

Early Earthquake and Tsunami Warning Viewer

AQA A-Level Computing NEA Report

Yicheng Shao (Eason)

Candidate Number: 7616

21st March 2025

Abstract

Give a brief summary outline of your project.

©2024–2025 Yicheng Shao (Eason).

This work is licensed under a CC BY-NC-ND 4.0 licence.

This work is formatted using X_ELATEX.

The references are managed using JabRef [37] and compiled using BibLATEX.

The source code is available at [GitHub](#) EasonSYC/nea-report .

The relevant program source code is available at [GitHub](#) EasonSYC/EEETWV , licensed under the MIT Licence.

Contents

1 Analysis	15
1.1 Background Information	15
1.1.1 The Early Earthquake Warning System	15
1.1.2 Earthquake Terminology	15
1.2 Problem Area	16
1.3 Client and End User	16
1.4 Research Methodology	17
1.4.1 Client Interview	17
1.4.2 Existing Applications and Solutions	18
1.5 Features of proposed solution	23
1.6 Critical Path	25
1.6.1 Part 1. NIED and DM-D.S.S. API/WebSocket Connection	25
1.6.2 Part 2a. Real-time Monitoring Map	25
1.6.3 Part 2b. Past-earthquake Viewing	26
1.6.4 Part 3. Joint functionality	26
1.6.5 Part 4. Setting page for customisation	26
1.6.6 Part 5. Additional functionalities	26
1.7 Requirements Specification	27
1.8 Initial Modelling	27
1.8.1 Colour Scheme to Use	29
1.8.2 Details to Display	29
2 Design	32
2.1 Hierarchy Chart	32
2.2 External Data Sources	32
2.2.1 NIED Data Source	32
2.2.2 DM-D.S.S. Data Source	44
2.3 Algorithms	67
2.3.1 Wavefront Calculation	67
2.3.2 Shake Detection	70
2.4 User Interface	70
2.4.1 Map	72
2.4.2 Real-Time Monitoring Screen	72
2.4.3 Past Earthquakes	74
2.4.4 Settings Page	74
2.4.5 Joint Functionalities and Technicalities	75
2.5 OOP Model	76
2.5.1 Design Patterns	76
2.5.2 Abstractions (Interfaces), Options and Services (Classes)	78
2.5.3 Design of UI Classes with MVVM Pattern	88
2.5.4 Design of DTOs (Record Classes)	100
2.6 Hardware and Software Requirements	104
3 Technical Implementation	105
3.1 JMA Travel Time	105
3.2 Kyoshin Monitor	107
3.3 DM-D.S.S. Authentication	112

4 Testing	125
4.1 Test Strategy	125
4.2 Testing Video	125
4.3 System Tests (against original requirements' specification)	125
5 Evaluation	126
5.1 Requirements Specification Evaluation	126
5.2 Independent End-User Feedback	126
5.3 Improvements	126
A Code Listing	128
A.1 Polynomial Fit for Colour	128
A.2 JMA Time Table	134
A.3 Kyoshin Monitor (Kmoni)	139
A.4 DM-D.S.S. Common DTOs	153
A.5 DM-D.S.S. Authentication	170
A.6 DM-D.S.S. API	201
A.7 DM-D.S.S. Telegrams	229
A.8 DM-D.S.S. WebSocket	259
A.9 Kmoni Settings Helper	284
A.10 GUI	289
A.11 Use of AI	395
B Use of AI	397
C Bibliography	400

List of Tables

1.3.1	Comparison of target users and clients	16
1.4.1	Supported platforms of existing solutions	18
1.4.2	Feature comparison in monitoring of existing solutions	18
1.4.3	Feature comparison in configuration and customisability of existing solutions	20
1.7.1	Measurement methods	27
1.7.2	Requirements for Part 1	27
1.7.3	Requirements for Part 2(a)	28
1.7.4	Requirements for Part 2(b)	28
1.7.5	Requirements for Part 3	28
1.7.6	Requirements for Part 4 and 5	29
1.8.1	Colour Scheme for Application	30
2.2.1	Data available in 'Kyoshin' monitor	36
2.2.2	Sensors available in 'Kyoshin' monitor	36
2.2.3	Initial values for f_H	39
2.2.4	Initial values for f_S	40
2.2.5	Initial values for f_V	41
2.2.6	Error codes for authorisation code step in OAuth2	47
2.2.7	Error codes for access token step in OAuth2	48
2.2.8	Necessary access permissions for DM-D.S.S	50
2.2.9	Standard errors for API	51
2.2.10	Errors for <code>socket.start</code>	55
2.2.11	Errors for WebSocket connection	61
2.2.12	Telegrams used in the application	65
2.3.1	Distance intervals in JMA 2001	69
2.6.1	Hardware Specification for App	102
4.3.1	Table of Tests	125
5.1.1	Table of Evaluation	126
5.2.1	Table of Feedback	126

List of Figures

1.4.1	Feature introduction of JQuake	19
1.4.2	Feature introduction of Quarog	19
1.4.3	Customisable colour scheme of KEVI	20
1.4.4	Customisability of Quarog	20
1.4.5	Past earthquake pages for KEVI and SREV	21
1.4.6	Display of observation stations and regions of KEVI	21
1.4.7	Intensity tree on side panel of KEVI	22
1.4.8	Sidebar options of JQuake	22
1.4.9	Real-time and tsunami pages of KEVI	23
1.4.10	Real-time and tsunami pages of SREV	24
1.4.11	Main page for JQuake	24
1.8.1	JMA Colour Scheme for Measured Intensities	29
1.8.2	JMA Colour Scheme for Tsunamis	30
2.1.1	Hierarchy chart of the whole application	33
2.1.2	Hierarchy chart of the GUI Component	34
2.2.1	A comparison of the K-NET, F-net and Hi-net	35
2.2.2	Distribution of the sensors of different nets	35
2.2.3	Sample GIF image achieved from 'Kyoshin' Monitor	36
2.2.4	Scale colours of different measurements	37
2.2.5	The hue scale in HSL/HSV encoding	38
2.2.6	The values of (H, S, V) against pixel row r	38
2.2.7	The values of (H, S, V) against normalised height h	39
2.2.8	The fit result for $f_H : h \mapsto H$	40
2.2.9	The fit result for $f_S : h \mapsto S$	41
2.2.10	The fit result for $f_V : h \mapsto V$	42
2.2.11	Flow of data in NIED data sources	44
2.2.12	Relation between abstract variables	45
2.2.13	Colour generated using fitted functions	45
2.2.14	Control panel for API Keys	45
2.2.15	OAuth Procedure Outline	46
2.2.16	Flowchart for OAuth 2.0 Authorisation Code and New Refresh Token	49
2.2.17	Flowchart for OAuth 2.0 Acquire Access Token using Refresh Token	50
2.2.18	Flowchart of WebSocket Wrapper	63
2.2.19	Flowchart of dealing with WebSocket Response	64
2.3.1	Shake detection flowchart and diagram	71
2.4.1	Design of GUI for Real-Time Page	73
2.4.2	Design of GUI for Past Earthquakes Page	74
2.4.3	Design of GUI for Settings Page	75
2.5.1	Class Diagram for JMA Time Table	79
2.5.2	Class Diagram for Kmoni	80
2.5.3	Class Diagram for Extra Services of Kmoni	81
2.5.4	Class Diagram for DM-D.S.S. Authentication	83
2.5.5	Class Diagram for DM-D.S.S. API Calls	84
2.5.6	Class Diagram for DM-D.S.S Telegram Fetching and Parsing	85
2.5.7	Class Diagram for DM-D.S.S WebSocket	86
2.5.8	Class Diagram for DM-D.S.S WebSocket DTOs	87
2.5.9	Class Diagram for Logging Services	88

2.5.10	Class Diagram for Map Resources	89
2.5.11	Class Diagram for Time Abstraction	89
2.5.12	Class Diagram for Kmoni Settings Manager	90
2.5.13	Class Diagram for View Models	91
2.5.14	Class Diagram for Window	93
2.5.15	Class Diagram for Settings	94
2.5.16	Class Diagram for Past Earthquake	95
2.5.17	Class Diagram for Real-Time Page	98
2.5.18	Class Diagram for DTOs for API Calls	102
2.5.19	Class Diagram for DTOs for Telegrams	103
B.0.1	Prompt to ChatGPT in [59], Part 1	397
B.0.2	Prompt to ChatGPT in [59], Part 2	398
B.0.3	Prompt to ChatGPT in [59], Part 3	399

List of Listings

2.2.1	Response for OAuth Access Token Request	48
2.2.2	Error for OAuth Access Token Request	48
2.2.3	ok status for API	51
2.2.4	error status for API	51
2.2.5	Cursor token sample JSON	52
2.2.6	Contract list sample JSON	52
2.2.7	Socket start sample request JSON	53
2.2.8	Socket start sample response JSON	54
2.2.9	Socket list sample response JSON	56
2.2.10	Earthquake parameter sample response JSON	56
2.2.11	Past earthquake list sample response JSON	58
2.2.12	Past earthquake event sample response JSON	59
2.2.13	WebSocket Ping JSON	60
2.2.14	WebSocket Pong JSON	61
2.2.15	WebSocket Error JSON	61
2.2.16	WebSocket Data JSON	62
2.2.17	Head of JSON Schema	66
2.2.18	Comments in JSON Schema	67
2.3.1	JMA 2001 Wave Travel Tables	68
3.1.1	ITimeTable Interface	105
3.1.2	TimeTableEntry DTO	105
3.1.3	JmaTimeTableBuilder Factory Class	106
3.1.4	JmaTimeTable Service Class	107
3.1.5	JmaTimeTableOptions Options Record	107
3.1.6	JmaTimeTableServiceCollectionExtensions DI Injection Extension	107
3.2.1	IImageFetch Interface	108
3.2.2	IPointExtract Interface	108
3.2.3	ImageFetch Service Class	109
3.2.4	EnumToUriStringExtensions Extensions Class	109
3.2.5	PointExtract Service Class	110
3.2.6	KmoniServiceCollectionExtensions DI Injection Extension	110
3.2.7	ColourConversionExtensions Extensions Class	112
3.3.1	IAuthenticator Interface	112
3.3.2	NullAuthenticator Service Class	113
3.3.3	ApiKeyAuthenticator Service Class	113
3.3.4	OAuth2Authenticator Service Class	116
3.3.5	OAuth2Helper Service Class	118
3.3.6	Error, TokenRefresh, TokenRefresh DTO Records	119
3.3.7	OAuth2Provider Factory Methods	120
3.3.8	IAuthenticationHelper Interface	120
3.3.9	AuthenticationHelper Service Class	123
3.3.10	AuthenticationStatus Enumeration	123
3.3.11	AuthenticationStatusChangedEventArgs Event	123
3.3.12	OAuth2Options Options Class	124
3.3.13	AuthenticationHelperServiceCollectionExtensions DI Injection Extension	124
A.1.1	Code for Polynomial Fit of Colour	134

A.2.1	EasonEetwViewer.JmaTravelTime/Options/JmaTimeTableOptions.cs	134
A.2.2	EasonEetwViewer.JmaTravelTime/Extensions/JmaTimeTableServiceCollectionExtensions.cs	134
A.2.3	EasonEetwViewer.JmaTravelTime/Dtos/TimeTableEntry.cs	135
A.2.4	EasonEetwViewer.JmaTravelTime/Abstractions/ITimeTable.cs	135
A.2.5	EasonEetwViewer.JmaTravelTime/Logging/JmaTimeTableLogs.cs	137
A.2.6	EasonEetwViewer.JmaTravelTime/Services/JmaTimeTableBuilder.cs	137
A.2.7	EasonEetwViewer.JmaTravelTime/Services/JmaTimeTable.cs	139
A.3.1	EasonEetwViewer.KyoshinMonitor/Extensions/KmoniServiceCollectionExtensions.cs	140
A.3.2	EasonEetwViewer.KyoshinMonitor/Extensions/ColourConversionExtensions.cs	143
A.3.3	EasonEetwViewer.KyoshinMonitor/Extensions/ByteArrayToBitmapExtensions.cs	144
A.3.4	EasonEetwViewer.KyoshinMonitor/Extensions/EnumToStringExtensions.cs	145
A.3.5	EasonEetwViewer.KyoshinMonitor/Extensions/ObservationPointCollectionExtensions.cs	145
A.3.6	EasonEetwViewer.KyoshinMonitor/Dtos/PointType.cs	145
A.3.7	EasonEetwViewer.KyoshinMonitor/Dtos/KmoniHelperOptions.cs	146
A.3.8	EasonEetwViewer.KyoshinMonitor/Dtos/PixelCoordinate.cs	146
A.3.9	EasonEetwViewer.KyoshinMonitor/Dtos/ObservationPoint.cs	147
A.3.10	EasonEetwViewer.KyoshinMonitor/Dtos/GeographicCoordinate.cs	148
A.3.11	EasonEetwViewer.KyoshinMonitor/Abstractions/MeasurementType.cs	148
A.3.12	EasonEetwViewer.KyoshinMonitor/Abstractions/IImageFetch.cs	149
A.3.13	EasonEetwViewer.KyoshinMonitor/Abstractions/SensorType.cs	149
A.3.14	EasonEetwViewer.KyoshinMonitor/Abstractions/IPointExtract.cs	150
A.3.15	EasonEetwViewer.KyoshinMonitor/Logging/ImageFetchLogs.cs	151
A.3.16	EasonEetwViewer.KyoshinMonitor/Services/ImageFetch.cs	152
A.3.17	EasonEetwViewer.KyoshinMonitor/Services/PointExtract.cs	153
A.4.1	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Magnitude.cs	153
A.4.2	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Coordinate.cs	154
A.4.3	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Distance.cs	154
A.4.4	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/CoordinateComponent.cs	155
A.4.5	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Height.cs	156
A.4.6	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Hypocentre.cs	157
A.4.7	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Auxiliary.cs	157
A.4.8	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/HypocentreDetail.cs	158
A.4.9	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Depth.cs	158
A.4.10	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/EarthquakeComponent.cs	159
A.4.11	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/Geodetic.cs	159
A.4.12	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/Source.cs	160
A.4.13	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/DepthCondition.cs	161
A.4.14	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/MagnitudeUnit.cs	161
A.4.15	EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/MagnitudeCondition.cs	162
A.4.16	EasonEetwViewer.Dmdata.Dtos/Enum/Intensity.cs	163
A.4.17	EasonEetwViewer.Dmdata.Dtos/Enum/Classification.cs	164
A.4.18	EasonEetwViewer.Dmdata.Dtos/Enum/LgIntensity.cs	164
A.4.19	EasonEetwViewer.Dmdata.Dtos/Enum/LgCategory.cs	165
A.4.20	EasonEetwViewer.Dmdata.Dtos/Enum/WebSocket/TestStatus.cs	165
A.4.21	EasonEetwViewer.Dmdata.Dtos/Enum/WebSocket/FormatType.cs	166
A.4.22	EasonEetwViewer.Dmdata.Dtos/Telegram/XmlControl.cs	167
A.4.23	EasonEetwViewer.Dmdata.Dtos/Telegram/TelegramType.cs	167
A.4.24	EasonEetwViewer.Dmdata.Dtos/Telegram/XmlHead.cs	169
A.4.25	EasonEetwViewer.Dmdata.Dtos/Telegram/SchemaVersionInformation.cs	169
A.4.26	EasonEetwViewer.Dmdata.Dtos/Telegram/XmlReport.cs	170
A.4.27	EasonEetwViewer.Dmdata.Dtos/Telegram/TelegramStatus.cs	170
A.5.1	EasonEetwViewer.Dmdata.Authentication/Options/OAuth2Options.cs	171
A.5.2	EasonEetwViewer.Dmdata.Authentication/Exceptions/OAuthException.cs	171

A.5.3	EasonEetwViewer.Dmdata.Authentication/Exceptions/NullAuthenticatiorExc_	ption.cs	172
A.5.4	EasonEetwViewer.Dmdata.Authentication/Exceptions/OAuthJsonException.cs	172
A.5.5	EasonEetwViewer.Dmdata.Authentication/Exceptions/OAuthSecurityExceptio_	n.cs	173
A.5.6	EasonEetwViewer.Dmdata.Authentication/Exceptions/OAuthErrorException.c_	s	173
A.5.7	EasonEetwViewer.Dmdata.Authentication/Extensions/AuthenticationHelperS_	erviceCollectionExtensions.cs	174
A.5.8	EasonEetwViewer.Dmdata.Authentication/Dtos/Error.cs	175	
A.5.9	EasonEetwViewer.Dmdata.Authentication/Dtos/TokenRefresh.cs	175	
A.5.10	EasonEetwViewer.Dmdata.Authentication/Dtos/TokenRequest.cs	176	
A.5.11	EasonEetwViewer.Dmdata.Authentication/Abstractions/AuthenticationStatu_	s.cs	176
A.5.12	EasonEetwViewer.Dmdata.Authentication/Abstractions/IAuthenticator.cs . .	176	
A.5.13	EasonEetwViewer.Dmdata.Authentication/Abstractions/IAuthenticationHelp_	er.cs	177
A.5.14	EasonEetwViewer.Dmdata.Authentication/Events/AuthenticationStatusChang_	edEventArgs.cs	178
A.5.15	EasonEetwViewer.Dmdata.Authentication/Logging/OAuth2HelperLogs.cs	180	
A.5.16	EasonEetwViewer.Dmdata.Authentication/Logging/AuthenticationHelperLogs_	.cs	182
A.5.17	EasonEetwViewer.Dmdata.Authentication/Logging/OAuth2AuthenticatorLogs.	_cs	184
A.5.18	EasonEetwViewer.Dmdata.Authentication/Services/OAuth2Provider.cs	185	
A.5.19	EasonEetwViewer.Dmdata.Authentication/Services/AuthenticationHelper.cs	189
A.5.20	EasonEetwViewer.Dmdata.Authentication/Services/OAuth2SharedMethod.cs . .	190	
A.5.21	EasonEetwViewer.Dmdata.Authentication/Services/NullAuthenticator.cs . . .	190	
A.5.22	EasonEetwViewer.Dmdata.Authentication/Services/OAuth2Helper.cs	195	
A.5.23	EasonEetwViewer.Dmdata.Authentication/Services/OAuth2Authenticator.cs .	200	
A.5.24	EasonEetwViewer.Dmdata.Authentication/Services/ApiKeyAuthenticator.cs .	200	
A.6.1	EasonEetwViewer.Dmdata.Api/Extensions/ApiCallerServiceCollectionExtens_	ions.cs	201
A.6.2	EasonEetwViewer.Dmdata.Api/Extensions/NameValueCollectionExtensions.cs	202
A.6.3	EasonEetwViewer.Dmdata.Api/Extensions/EnumToStringExtensions.cs	203	
A.6.4	EasonEetwViewer.Dmdata.Api/Dtos/Record/ErrorDetails.cs	203	
A.6.5	EasonEetwViewer.Dmdata.Api/Dtos/Record/GdEarthquake/EarthquakeInfoWith_	Telegrams.cs	204
A.6.6	EasonEetwViewer.Dmdata.Api/Dtos/Record/GdEarthquake/EarthquakeInfo.cs	205
A.6.7	EasonEetwViewer.Dmdata.Api/Dtos/Record/GdEarthquake/TelegramHead.cs	206
A.6.8	EasonEetwViewer.Dmdata.Api/Dtos/Record/GdEarthquake/TelegramItem.cs	207
A.6.9	EasonEetwViewer.Dmdata.Api/Dtos/Record/WebSocket/WebSocketUrl.cs	208
A.6.10	EasonEetwViewer.Dmdata.Api/Dtos/Record/WebSocket/WebSocketDetails.cs	209
A.6.11	EasonEetwViewer.Dmdata.Api/Dtos/Record/Contract/UnitPrice.cs	210	
A.6.12	EasonEetwViewer.Dmdata.Api/Dtos/Record/Contract/Contract.cs	211	
A.6.13	EasonEetwViewer.Dmdata.Api/Dtos/Record/EarthquakeParameter/StationPosi_	tion.cs	211
A.6.14	EasonEetwViewer.Dmdata.Api/Dtos/Record/EarthquakeParameter/Station.cs	212
A.6.15	EasonEetwViewer.Dmdata.Api/Dtos/Response/WebSocketList.cs	213	
A.6.16	EasonEetwViewer.Dmdata.Api/Dtos/Response/GdEarthquakeList.cs	213	
A.6.17	EasonEetwViewer.Dmdata.Api/Dtos/Response/Error.cs	213	
A.6.18	EasonEetwViewer.Dmdata.Api/Dtos/Response/ContractList.cs	214	
A.6.19	EasonEetwViewer.Dmdata.Api/Dtos/Response/GdEarthquakeEvent.cs	214	
A.6.20	EasonEetwViewer.Dmdata.Api/Dtos/Response/WebSocketStart.cs	215	
A.6.21	EasonEetwViewer.Dmdata.Api/Dtos/Response/EarthquakeParameter.cs	215	
A.6.22	EasonEetwViewer.Dmdata.Api/Dtos/Enum/ResponseStatus.cs	216	
A.6.23	EasonEetwViewer.Dmdata.Api/Dtos/Enum/WebSocket/FormatMode.cs	216	
A.6.24	EasonEetwViewer.Dmdata.Api/Dtos/Enum/WebSocket/ConnectionStatus.cs	217

A.6.25	EasonEetwViewer.Dmdata.Api/Dtos/Enum/Earthquake/EarthquakeType.cs	217
A.6.26	EasonEetwViewer.Dmdata.Api/Dtos/Enum/Station/Status.cs	218
A.6.27	EasonEetwViewer.Dmdata.Api/Dtos/Request/WebSocketStartPost.cs	219
A.6.28	EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/ListBase.cs	219
A.6.29	EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/SuccessBase.cs	219
A.6.30	EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/ApiBase.cs	220
A.6.31	EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/PoolingBase.cs	221
A.6.32	EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase TokenNameBase.cs	221
A.6.33	EasonEetwViewer.Dmdata.Api/Abstractions/IApiCaller.cs	222
A.6.34	EasonEetwViewer.Dmdata.Api/Logging/ApiCallerLogs.cs	224
A.6.35	EasonEetwViewer.Dmdata.Api/Services/ApiCaller.cs	229
A.7.1	EasonEetwViewer.Dmdata.Telegram/Exceptions/TelegramParserFormatException.cs	230
A.7.2	EasonEetwViewer.Dmdata.Telegram/Exceptions/TelegramParserException.cs . .	230
A.7.3	EasonEetwViewer.Dmdata.Telegram/Exceptions/TelegramParserUnsupportedException.cs	231
A.7.4	EasonEetwViewer.Dmdata.Telegram/Extensions/TelegramRetrieverServiceCollectionExtensions.cs	231
A.7.5	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Tsunami.cs	232
A.7.6	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/FirstHeight.cs . .	232
A.7.7	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Kind.cs	232
A.7.8	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Forecast.cs . .	233
A.7.9	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Body.cs	233
A.7.10	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/SimpleKind.cs . .	233
A.7.11	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/MaxHeight.cs . .	234
A.7.12	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Comments.cs . .	234
A.7.13	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/TsunamiHeight.cs	234
A.7.14	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Enum/Revise.cs . .	234
A.7.15	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Enum/MaxHeightCondition.cs	235
A.7.16	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Enum/HeightCondition.cs	235
A.7.17	EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Enum/FirstHeightCondition.cs	235
A.7.18	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/RegionIntensity.cs	236
A.7.19	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Sva.cs	236
A.7.20	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Body.cs	236
A.7.21	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/PeriodicBanned.cs	236
A.7.22	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/PrePeriod.cs	237
A.7.23	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/IntensityDetails.cs	237
A.7.24	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Comments.cs . .	237
A.7.25	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/StationIntensity.cs	238
A.7.26	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/CityIntensity.cs	238
A.7.27	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Enum/ReviseStatus.cs	239
A.7.28	EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Enum/IntensityWithUnreceived.cs	240
A.7.29	EasonEetwViewer.Dmdata.Telegram/Dtos/Schema/TsunamiInformationSchema.cs	240
A.7.30	EasonEetwViewer.Dmdata.Telegram/Dtos/Schema/EewInformationSchema.cs	240

A.7.31	EasonEetwViewer.Dmdata.Telegram/Dtos/Schema/EarthquakeInformationSchema.cs	240
A.7.32	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/FromTo.cs	241
A.7.33	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Region.cs	241
A.7.34	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/ReducedName.cs	241
A.7.35	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Appendix.cs	242
A.7.36	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Earthquake.cs	242
A.7.37	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Body.cs	243
A.7.38	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Hypocentre.cs	243
A.7.39	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/SimpleKind.cs	243
A.7.40	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/AreaKind.cs	244
A.7.41	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Accuracy.cs	244
A.7.42	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Area.cs	244
A.7.43	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Comments.cs	244
A.7.44	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Station.cs	245
A.7.45	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/IntensityInfo.cs	245
A.7.46	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/HypocentrePosition.cs	245
A.7.47	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Change/Reason.cs	246
A.7.48	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Change/MaxInt.cs	246
A.7.49	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Accuracy/Magnitude.cs	246
A.7.50	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Accuracy/MagnitudePoint.cs	247
A.7.51	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Accuracy/EpicentreDepth.cs	247
A.7.52	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Range/IntensityUpper.cs	249
A.7.53	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Range/LgIntensityUpper.cs	249
A.7.54	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Range/IntensityLower.cs	250
A.7.55	EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Range/LgIntensityLower.cs	251
A.7.56	EasonEetwViewer.Dmdata.Telegram/Dtos/TelegramBase/AdditionalComment.cs	251
A.7.57	EasonEetwViewer.Dmdata.Telegram/Dtos/TelegramBase/Head.cs	252
A.7.58	EasonEetwViewer.Dmdata.Telegram/Abstractions/ITelegramRetriever.cs	252
A.7.59	EasonEetwViewer.Dmdata.Telegram/Abstractions/ITelegramParser.cs	253
A.7.60	EasonEetwViewer.Dmdata.Telegram/Logging/TelegramRetrieverLogs.cs	254
A.7.61	EasonEetwViewer.Dmdata.Telegram/Logging/TelegramParserLogs.cs	255
A.7.62	EasonEetwViewer.Dmdata.Telegram/Services/TelegramRetriever.cs	257
A.7.63	EasonEetwViewer.Dmdata.Telegram/Services/TelegramParser.cs	259
A.8.1	EasonEetwViewer.Dmdata.WebSocket/Exceptions/WebSocketClientException.cs	259
A.8.2	EasonEetwViewer.Dmdata.WebSocket/Exceptions/WebSocketClientUnsupportedException.cs	260
A.8.3	EasonEetwViewer.Dmdata.WebSocket/Exceptions/WebSocketClientFormatException.cs	261
A.8.4	EasonEetwViewer.Dmdata.WebSocket/Extensions/WebSocketServiceCollectionExtensions.cs	261
A.8.5	EasonEetwViewer.Dmdata.WebSocket/Dtos/HeadInfo.cs	262
A.8.6	EasonEetwViewer.Dmdata.WebSocket/Dtos/MessageType.cs	263
A.8.7	EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/DataResponse.cs	264
A.8.8	EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/ResponseBase.cs	265
A.8.9	EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/PingResponse.cs	265
A.8.10	EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/PongResponse.cs	266

A.8.11	EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/ErrorResponse.cs	266
A.8.12	EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/StartResponse.cs	267
A.8.13	EasonEetwViewer.Dmdata.WebSocket/Dtos/Request/PongRequest.cs	268
A.8.14	EasonEetwViewer.Dmdata.WebSocket/Dtos/Request/PingRequest.cs	268
A.8.15	EasonEetwViewer.Dmdata.WebSocket/Dtos/Data/CompressionType.cs	269
A.8.16	EasonEetwViewer.Dmdata.WebSocket/Dtos/Data/EncodingType.cs	269
A.8.17	EasonEetwViewer.Dmdata.WebSocket/Dtos/Data/PassingDetail.cs	269
A.8.18	EasonEetwViewer.Dmdata.WebSocket/Abstractions/IWebSocketClient.cs	270
A.8.19	EasonEetwViewer.Dmdata.WebSocket/Events/StatusChangedEventArgs.cs	270
A.8.20	EasonEetwViewer.Dmdata.WebSocket/Events/DataReceivedEventArgs.cs	271
A.8.21	EasonEetwViewer.Dmdata.WebSocket/Logging/WebSocketClientLogs.cs	275
A.8.22	EasonEetwViewer.Dmdata.WebSocket/Services/WebSocketClient.cs	284
A.9.1	EasonEetwViewer.Services.Kmoni/Options/KmoniSettingsHelperOptions.cs	284
A.9.2	EasonEetwViewer.Services.Kmoni/Extensions/KmoniSettingsHelperServiceCollectionExtensions.cs	285
A.9.3	EasonEetwViewer.Services.Kmoni/Dtos/KmoniSettings.cs	286
A.9.4	EasonEetwViewer.Services.Kmoni/Abstractions/IKmoniSettingsHelper.cs	286
A.9.5	EasonEetwViewer.Services.Kmoni/Events/KmoniSettingsChangedEventArgs.cs	287
A.9.6	EasonEetwViewer.Services.Kmoni/Logging/KmoniSettingsHelperLogs.cs	288
A.9.7	EasonEetwViewer.Services.Kmoni/Services/KmoniSettingsHelper.cs	289
A.10.1	EasonEetwViewer/appsettings.json	290
A.10.2	EasonEetwViewer/ViewLocator.cs	291
A.10.3	EasonEetwViewer/App.axaml	291
A.10.4	EasonEetwViewer/App.axaml.cs	294
A.10.5	EasonEetwViewer/Program.cs	295
A.10.6	EasonEetwViewer/ViewModels/MainWindowViewModel.cs	297
A.10.7	EasonEetwViewer/ViewModels/RealTimePageViewModel.cs	308
A.10.8	EasonEetwViewer/ViewModels/PastPageViewModel.cs	316
A.10.9	EasonEetwViewer/ViewModels/SettingPageViewModel.cs	323
A.10.10	EasonEetwViewer/ViewModels/ViewModelBases/ViewModelBase.cs	323
A.10.11	EasonEetwViewer/ViewModels/ViewModelBases/MapViewModelBase.cs	325
A.10.12	EasonEetwViewer/ViewModels/ViewModelBases/PageViewModelBase.cs	326
A.10.13	EasonEetwViewer/Converters/CultureInfoConverter.cs	326
A.10.14	EasonEetwViewer/Converters/UnitValueDisplayValueConverter.cs	327
A.10.15	EasonEetwViewer/Converters/EnumDisplayTextConverter.cs	327
A.10.16	EasonEetwViewer/Converters/UnitValueDisplayUnitConverter.cs	328
A.10.17	EasonEetwViewer/Converters/EnumDisplayColourConverter.cs	328
A.10.18	EasonEetwViewer/Models/MainWindow/SidebarItemTemplate.cs	329
A.10.19	EasonEetwViewer/Models/PastPage/DisplayMode.cs	329
A.10.20	EasonEetwViewer/Models/PastPage/EarthquakeDetailsTemplate.cs	330
A.10.21	EasonEetwViewer/Models/PastPage/DetailIntensityTemplate.cs	331
A.10.22	EasonEetwViewer/Models/PastPage/EarthquakeItemTemplate.cs	331
A.10.23	EasonEetwViewer/Models/SettingPage/EmptyWebSocketConnectionTemplate.cs	332
A.10.24	EasonEetwViewer/Models/SettingPage/IWebSocketConnectionTemplate.cs	333
A.10.25	EasonEetwViewer/Models/SettingPage/WebSocketConnectionTemplate.cs	333
A.10.26	EasonEetwViewer/Models/RealTimePage/TsunamiDetailsTemplate.cs	334
A.10.27	EasonEetwViewer/Models/RealTimePage/TsunamiWarningType.cs	334
A.10.28	EasonEetwViewer/Models/RealTimePage/EewDetailsTemplate.cs	336
A.10.29	EasonEetwViewer/Models/RealTimePage/EewWarningType.cs	337
A.10.30	EasonEetwViewer/Extensions/EnumDisplayColourExtensions.cs	340
A.10.31	EasonEetwViewer/Extensions/InformationalTextExtensions.cs	341
A.10.32	EasonEetwViewer/Extensions/MapStyleExtensions.cs	344
A.10.33	EasonEetwViewer/Extensions/EnumDisplayTextExtensions.cs	350
A.10.34	EasonEetwViewer/Extensions/StringColourExtensions.cs	351
A.10.35	EasonEetwViewer/Extensions/MapsuiLoggingExtensions.cs	351
A.10.36	EasonEetwViewer/Extensions/UnitValueExtensions.cs	352
A.10.37	EasonEetwViewer/Extensions/WarningTypeExtensions.cs	353
A.10.38	EasonEetwViewer/Extensions/EnumConversionExtensions.cs	355

A.10.39 EasonEetwViewer/Extensions/LoggingExtensions.cs	356
A.10.40 EasonEetwViewer/Extensions/LinqExtensions.cs	357
A.10.41 EasonEetwViewer/Views/mainwindow.axaml	358
A.10.42 EasonEetwViewer/Views/SettingPageView.axaml.cs	358
A.10.43 EasonEetwViewer/Views/PastPageView.axaml.cs	359
A.10.44 EasonEetwViewer/Views/SettingPageView.axaml	363
A.10.45 EasonEetwViewer/Views/RealTimePageView.axaml.cs	363
A.10.46 EasonEetwViewer/Views/PastPageView.axaml	370
A.10.47 EasonEetwViewer/Views/mainwindow.axaml.cs	370
A.10.48 EasonEetwViewer/Views/RealTimePageView.axaml	380
A.10.49 EasonEetwViewer/Assets/Icons.axaml	384
A.10.50 EasonEetwViewer/Logging/ViewModels/SettingPageViewModelLogs.cs	387
A.10.51 EasonEetwViewer/Logging/ViewModels/mainwindowViewModelLogs.cs	388
A.10.52 EasonEetwViewer/Logging/ViewModels/ViewModelBases/MapViewModelBaseLogs.cs	389
A.10.53 EasonEetwViewer/Logging/ViewModels/ViewModelBases/ViewModelBaseLogs.cs	389
A.10.54 EasonEetwViewer/Services/MapResourcesProvider.cs	391
A.10.55 EasonEetwViewer/Services/PrefectureData.cs	391
A.10.56 EasonEetwViewer/Services/TimeProvider/ITimeProvider.cs	392
A.10.57 EasonEetwViewer/Services/TimeProvider/DefaultTimeProvider.cs	392
A.10.58 EasonEetwViewer/Services/Logging/FileLoggerExtensions.cs	392
A.10.59 EasonEetwViewer/Services/Logging/AvaloniaToMicrosoftLoggingAdaptor.cs	393
A.10.60 EasonEetwViewer/Services/Logging/FileLoggerProvider.cs	394
A.10.61 EasonEetwViewer/Services/Logging/FileLogger.cs	395
A.11.1 Code generated by ChatGPT in [59]	396

List of Algorithms

2.2.1	Algorithm for f^{-1}	43
2.3.1	Algorithm for Finding Distance based on Time	69

Chapter 1

Analysis

Overview

This section introduces the background for the EEW system in place in Japan, consisting of EEW Forecasts and EEW Warnings, and together with the tsunami warnings. Concepts such as intensities, magnitude and epicentres are defined. Two key users and targets are identified, specifically passionate geologists and earthquake-sensitive industries, each having different requirements. A client in the former was interviewed, together with a thorough analysis and comparison of some existing solutions such as SREV, JQuake, KEVI and Quarog. A detailed requirement specification and the critical path is also outlined, consisting of DM-D.S.S. parsing, real-time monitoring, past-earthquake viewing and joint functionalities.

1.1 Background Information

1.1.1 The Early Earthquake Warning System

Earthquake is one of the most common natural disasters in the whole world, and direct consequences of earthquakes include tsunamis which could be catastrophic.

Japan, sitting on the intersection of the Eurasian, the Philippine and the North-American plates, is the countries with most earthquakes. Historically, the Great Kantō Earthquake (関東大震災) in 1923, the Great East Japan Earthquake (東日本大震災) in 2011 (a.k.a. the Tōhoku Earthquake) and the recent 2024 Noto Peninsula Earthquake (能登半島地震) all caused hundreds of deaths, both due to the result of the earthquake(s) and the resulting tsunami.

To provide protection to its residents, the Japan Meteorological Agency (JMA, 気象庁), together with the National Research Institute for Earth Science and Disaster Resilience (NIED, 防災科研) placed thousands of **earthquake sensors** across Japan (the Hi-net, the K-NET, the KiK-net and the F-NET), with several of them lying deep in the sea bed, measuring displacement, velocity and acceleration, which are connected to multiple servers, including two located in Ōsaka and Tōkyo.

Using data obtained from the sensors, computers do some complicated algorithms (mentioned below) to send out **early earthquake warnings (EEWs, 緊急地震速報)** automatically within milliseconds. There are two types of EEWs:

1. **EEW (Forecast, 予報).** Sent out to **highly-dependent industries** (e.g. rail industry, power plants) and **subscribed users**, when maximum intensity level of more than 3, or a magnitude of more than 3.5 is expected.
2. **EEW (Warning, 警報).** Sent out to **everyone** via TV, Radio, Mobile Phone, SMS, etc., when a maximum intensity level of more than 4 is expected.

After the earthquake, JMA staff will determine the location and severity of tsunami warnings to be issued, if necessary.

1.1.2 Earthquake Terminology

- **Intensity (震度).** The intensity describes the intensity vibration of a point due to an earthquake. It is not unique to an earthquake - **different places can have different intensities** due to the

distance to the epicentre, and intensity will also change over time. JMA measures intensity using **9 levels: 1, 2, 3, 4, 5–, 5+, 6–, 6+ and 7** in increasing order.

- **Magnitude/Scale.** The magnitude of an earthquake describes the energy released in the earthquake in a logarithmic scale. **It is unique to an earthquake.**
- **Epicentre/Hypocentre.** The epicentre is the surface point directly above the true centre of the earthquake.
- **Focal Depth.** The focal depth is the depth of the true centre of the earthquake.
- **P-Wave and S-Wave.** These are seismic waves, sourced from the true centre of the earthquake, travelling at different speeds, with Primary (P)-Wave travelling faster and Secondary (S)-Wave travelling slower.

1.2 Problem Area

The main goal of this application is to provide a visualisation of the earthquake/tsunami related data feed(s) provided by JMA's affiliated institution, Disaster Mitigation Data Send Service (DM-D.S.S), and some other data feeds as well. There are numerous apps providing a list of recent earthquakes, the real-time data measured by the sensors, and the real time earthquake warning displayed on a map, but rarely are there good apps that combine all those features together satisfactorily, with just the necessary features the author needs.

Some applications are no longer being updated due to change in the user's policy of the related data feed. Furthermore, most of the apps available are only in Japanese, not in English or my home language Chinese, which can create trouble for the author to understand.

1.3 Client and End User

The primary target of this application will be passionate geographers and geologists who are interested in the study of earthquake observations and predictions. The age group of this vary all the way from primary-school students to adults, including the author who has been amazed by the technology since the age of 12. They could take any employment, ranging from students to full-time jobs. Their proficiency usually varies, since there are people new to this field who probably does not have much knowledge, so the interface of the application should be relatively user-friendly and understandable, hiding unnecessary technical complexities.

Another target client could be industries which highly rely on earthquake predictions due to the risk imposed by earthquakes. High-speed railway and nuclear power plants are good examples of this. Therefore, the staff in charge monitoring will usually have higher proficiency and would like more detailed data of the earthquake. However, they will only need the necessary data from earthquakes happening close to them and only require intensity data of the point in interest (e.g. the power plant). To put this into context, an earthquake happening 1000 km away from them does not need to be fed into their system, while they would like to see the intensity of the shock and the arrival time of the seismic waves to decide the actions. In fact, the author really likes investigating on the rail industry, whose infrastructure could be greatly affected by earthquakes.

Table 1.3.1 compares relative features of these two target users/clients.

Feature	Primary Target	Secondary Target
Description	Passionate Geologists	Earthquake-Sensitive Industries
Age	Varies (Middle School – Adults)	Work Age
Reason	Monitor Live & Latest Earthquakes	Monitor Risks to Infrastructure
Proficiency	Varies (Beginner – Amateur)	Trained Professional
General Requirements	Monitor Overall Movement	Alert intensity and arrival time at certain location

Table 1.3.1: Comparison of target users and clients

1.4 Research Methodology

1.4.1 Client Interview

The author interviewed my friend Wesley Ma, who is a passionate geologist on earthquake studies and also monitor earthquakes regularly.

1. *Which earthquake monitoring apps do you use?*

Response: JQuake, SREV, Quarog, KEVI, Kyoshin-Monitor (Support discontinued), Kiwi Monitor (Support discontinued)

2. *Do you subscribe/pay to services such as DM-D.S.S. to use earthquake monitoring apps, and do you think it is worth the price?*

Response: Yes. However, DM-D.S.S. is a little expensive. However, the price becomes more affordable considering the information provided by the subscription.

3. *Do you watch YouTube livestreams on earthquake monitoring?*

Response: I do not usually watch the live streams, as almost no one who has already has a monitoring app will use the stream. They provide mostly the same information as the applications, just real-time streaming the windows.

4. *Why do you use earthquake monitoring apps?*

Response: To monitor the earthquake. This is derived from my interest in broadcasting culture in Japan. This eventually led me to be intrigued with the development of earthquake monitoring technologies and theories in Japan.

5. *How often do you use earthquake monitoring apps (e.g. all the time/after school/only after big earthquakes)?*

Response: After big earthquakes. But I usually open one or two apps for all-day monitoring to catch potential major (or medium) earthquakes.

6. *Describe the advantages and disadvantages of each of them, mentioning the specific features.*

Response:

- SREV is a good one, but it is only available in a browser with no app.
- Kyoshin-Monitor and Kiwi Monitor are relatively more stable, but the source is not the same as that from the former two apps, and its support is also discontinued.
- Quarog has weak response time and interacting interface.
- JQuake is the most developed app, which includes nearly all functions that can be thought of. However sometimes the connection of WebSocket is unstable.
- KEVI does not have the sound files configured by default. Some information are not displayed clearly enough.

7. *What features do you use the most/least?*

Response: Basic earthquake notifications, and should be including sufficient and prompt information.

8. *What features are redundant in the earthquake monitoring apps you use?*

Response: KEVI has a weather monitoring function, which could be redundant. But that could be caused by the different purposes of the app. Hence, no further comments. It is not a bad thing.

9. *What are the critical features of an earthquake monitoring app?*

Response: To provide accurate, prompt and detailed information according to the source released by the JMA, the information display interface should be easy to read and understand. This is not only helping the people who like to monitor, but more importantly, provides the easiest way to the people who really need to seek information to minimise the harm brought by earthquakes and successive disaster.

10. *What additional features would you like to have in those existing apps?*

Response: Summarise all the useful features from different apps into one app. Stable connection. Nothing else.

1.4.2 Existing Applications and Solutions

Based on the applications the author uses and the feedback from interviewee, there are the following commonly-used applications:

- JQuake
- Scratch Real-time Earthquake Viewer (SREV), available in compiled form at [kotoho7/scratch-realtime-earthquake-viewer-page](#)
- Kyoshin EEW Viewer for Ingen (KEVI), available at [ingen084/KyoshinEewViewerIngen](#)
- Quarog

1.4.2.1 Supported Platforms

Supported platforms of those apps are listed in Table 1.4.1. In particular, note that SREV is a web-based and GitHub Pages-hosted application therefore supporting all platforms. KEVI is written in .NET Framework and supports the second most platforms, with JQuake not supporting Linux and Quarog only supporting Windows.

Platform	JQuake	SREV	KEVI	Quarog
Windows	✓	✓	✓	✓
macOS	✓	✓	✓	
Linux	✓	✓	✓	
Android/iOS		✓		

Table 1.4.1: Supported platforms of existing solutions

1.4.2.2 Earthquake Monitoring

An overview feature table of monitoring is included in Table 1.4.2. In particular, note that due to the nature of SREV being Scratch-programmed and web-based, it reached a special agreement with DM-D.S.S. to use the API without the need of all users paying for this, since it is hard to integrate such function into a web application.

Quarog is a relatively new application and only supports basic functionalities of EEW Viewing and past earthquake listing, as shown in Figure 1.4.2. JQuake is solely dedicated to earthquake and tsunami monitoring as shown in Figure 1.4.1, while KEVI has features like rain clouds map and natural disaster warning which is beyond the scope of this analysis (which is a burden to users only requiring earthquake monitoring and a waste of storage space).

It is worth noting that for SREV, DM-D.S.S. is integrated with special permission with no login required. For JQuake, although it has tsunami warnings/special warnings, the tsunami forecast is not available, since they use a different data source.

Feature	JQuake	SREV	KEVI	Quarog
NIED Real-time Shake Support	✓	✓	✓	
DM-D.S.S. WebSocket Support	✓	✓	✓	✓
Real-time Sensor Data	✓	✓	✓	
Vibration Alert	✓	✓	✓	
Past Earthquake List	✓	✓	✓	✓
Past Earthquake Details		✓	✓	✓
Tsunami Warning	✓	✓	✓	
Real-time EEW	✓	✓	✓	✓
Calculated Seismic Wavefronts	✓	✓	✓	✓
User-Defined Key Monitor Point	✓		✓	
Sub-Map for the Okinawa Area	✓		✓	
Replay	✓		✓	

Table 1.4.2: Feature comparison in monitoring of existing solutions



Figure 1.4.1: Feature introduction of JQuake, screenshot from website



Figure 1.4.2: Feature introduction of Quarog, screenshot from website
Top-left: Past earthquake information; Top-right: Real-time EEW;
Bottom-Left: Past earthquake list; Bottom-Right: Details of EEW

1.4.2.3 Configuration Options

There are also a variety of configuration options available for all apps, as listed in Table 1.4.3. Both KEVI and Quarog supports the adjustment of the colour theme, and Quarog even supports changing the style of how blocks are displayed and coloured as shown in Figure 1.4.3 and Figure 1.4.4. Playing a sound on the speaker is also common among the apps to remind the user of earthquakes.

Feature	JQuake	SREV	KEVI	Quarog
DM-D.S.S. Login	✓		✓	✓
Sound Alert	✓	✓	✓	✓
System Notification			✓	
Colour Theme			✓	✓
Map Colouring Style		✓		✓

Table 1.4.3: Feature comparison in configuration and customisability of existing solutions

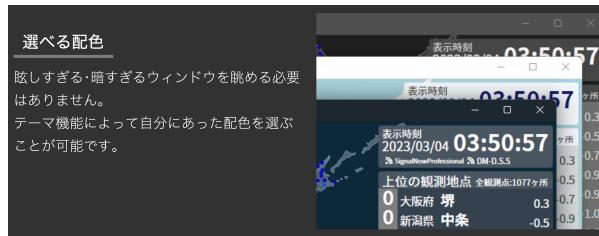


Figure 1.4.3: Customisable colour scheme of KEVI, screenshot from website



Figure 1.4.4: Customisability of Quarog, screenshot from website

1.4.2.4 GUI Analysis

All applications are similar in a way, that their past earthquakes list is displayed on a sidebar.

For Quarog and KEVI, an earthquake can be selected from the sidebar to view the details of, as in Figure 1.4.5. They will both colour on the map the maximum observed intensities by regions. While KEVI chooses to display the regions as colours and also provide an overlay of the observed intensities by observation stations (which will only appear after zoomed in enough, as in Figure 1.4.6), Quarog chooses to have a separate button to switch between display of regions, and the display of observation stations.

In particular for KEVI, there is also a tree structure on a panel displaying the details of the observed intensities by hierarchy structure, consisting of prefectures, skipping regions, then districts, then the exact observation stations, as shown in Figure 1.4.7.

As for SREV, the selected earthquake will be displayed on the map and coloured by region, but no information on observation stations will be provided. For JQuake, it is impossible to select a particular earthquake, but only possible to replay the past earthquake, and jump to an external weather service to view the details, as shown in Figure 1.4.8.

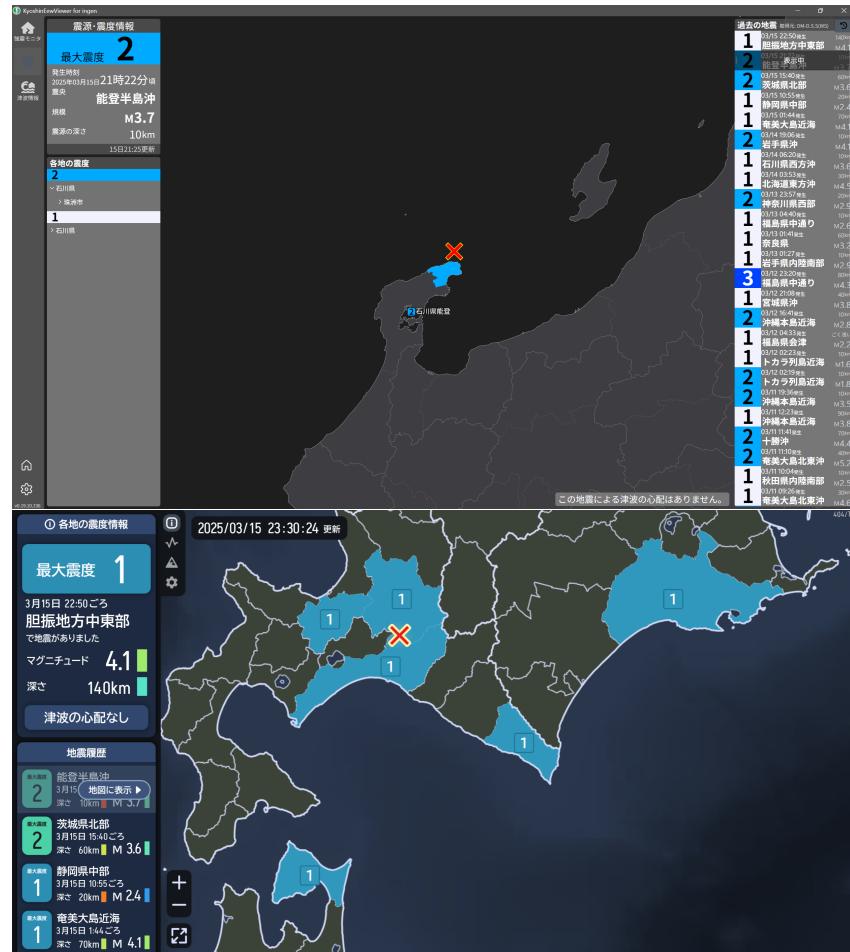


Figure 1.4.5: Past earthquake pages
Top: KEVI; Bottom: SREV

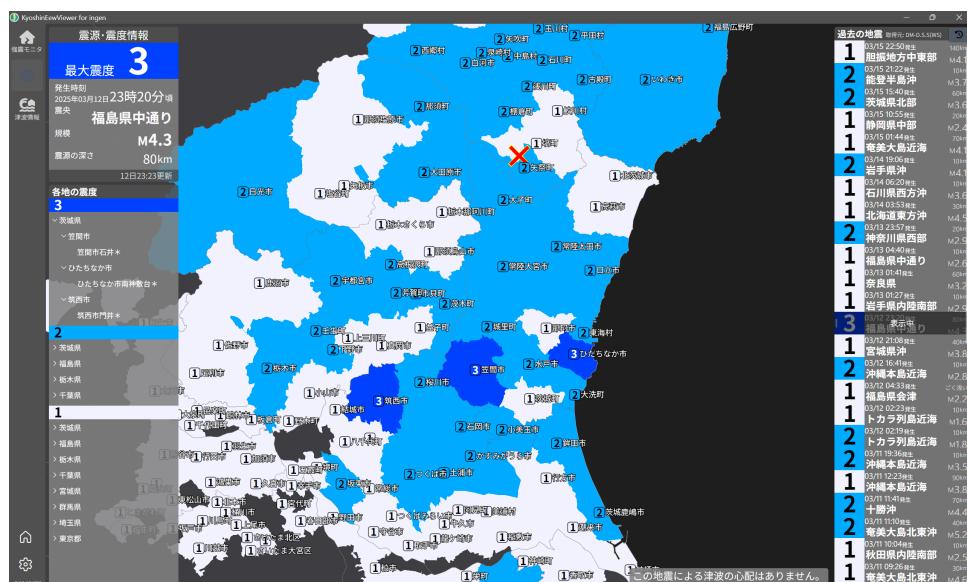


Figure 1.4.6: Display of observation stations and regions of KEVI



Figure 1.4.7: Intensity tree on side panel of KEVI



Figure 1.4.8: Sidebar options of JQuake
 Globe: Link to Yahoo! weather service page for earthquake;
 Time-lapse: Replay of earthquake

A significant difference between the applications is how they split the different functionalities. For KEVI and SREV, there are separate pages for real-time monitoring, past-earthquake viewing and tsunami warnings, as shown in Figure 1.4.9 for KEVI, and Figure 1.4.10 for SREV.

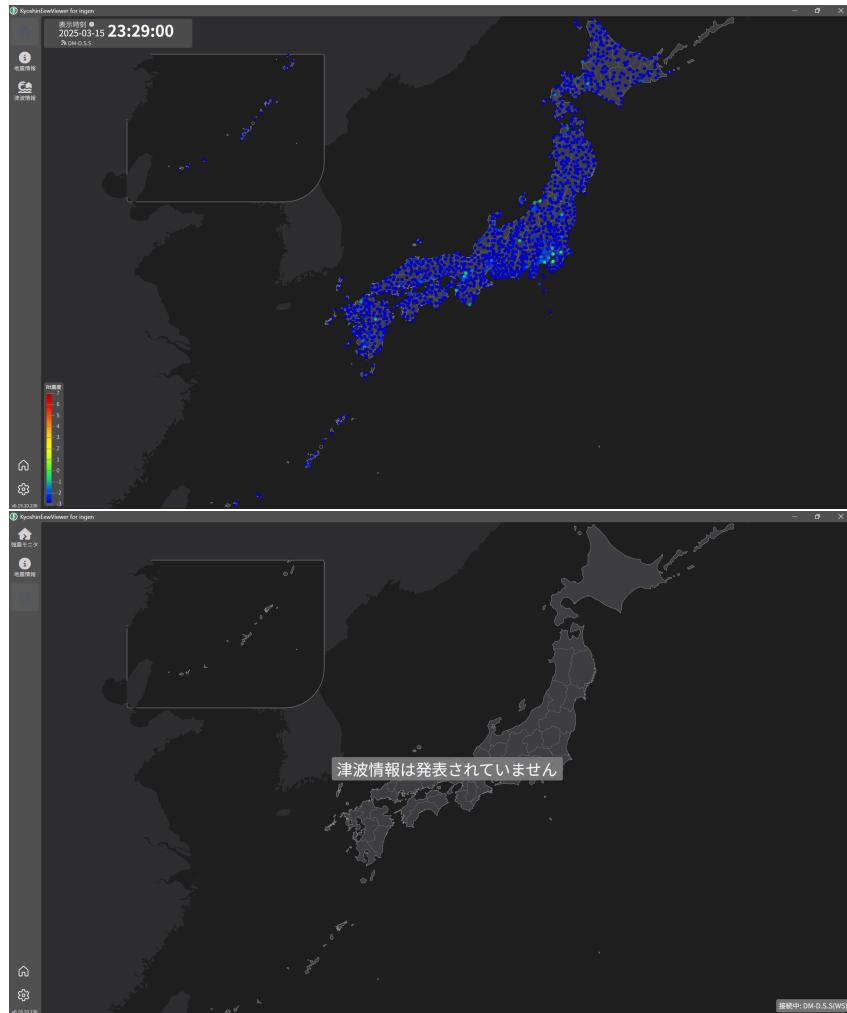


Figure 1.4.9: Real-time and tsunami pages of KEVI

On the contrary, since JQuake's map only supports replay and real-time functionality (EEW and tsunami), there is only one page, as shown in Figure 1.4.11. Quarog will automatically switch the map to display EEWs if EEWs are issued, and change the sidebar to display the predicted intensities of EEWs, as shown in Figure 1.4.2.

1.5 Features of proposed solution

Based on the potential user/client interview results, and the research into the existing solutions, the following key features should appear in the solution, since they are essential to earthquake monitoring applications:

1. DM-D.S.S. Login functionality (for the data source)
2. Real-Time Sensor Shake Intensity Data
3. EEW Visualisation (w/ Calculated Seismic Wavefronts)
4. Past Earthquake List (w/ Option to review details)
5. Tsunami Warning Visualisation

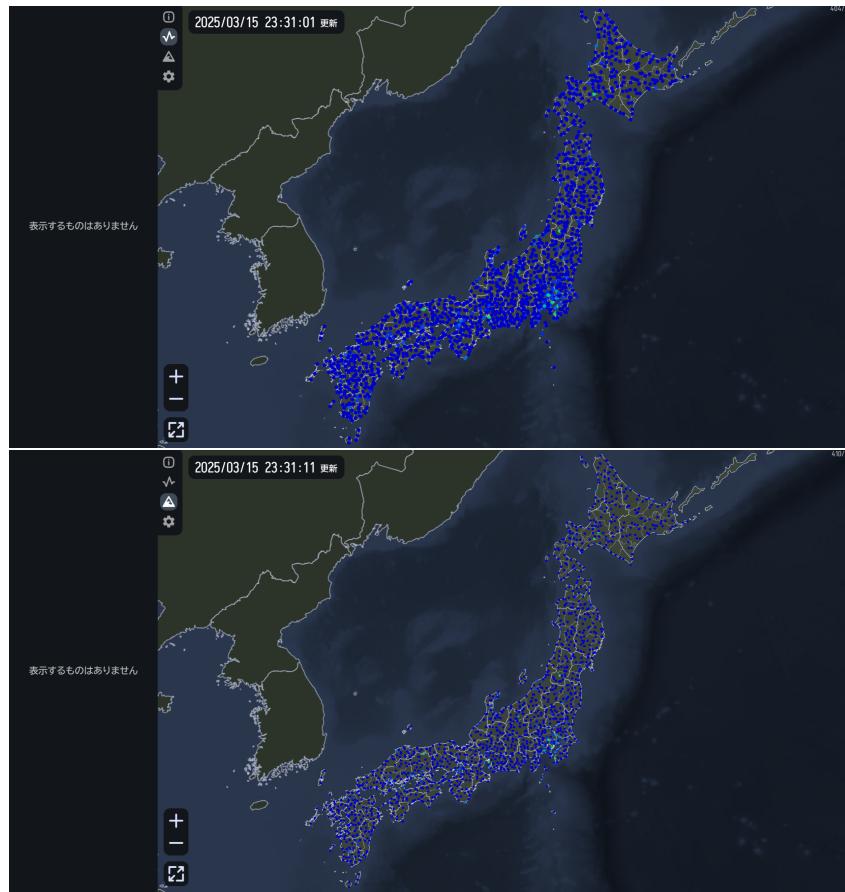


Figure 1.4.10: Real-time and tsunami pages of SREV

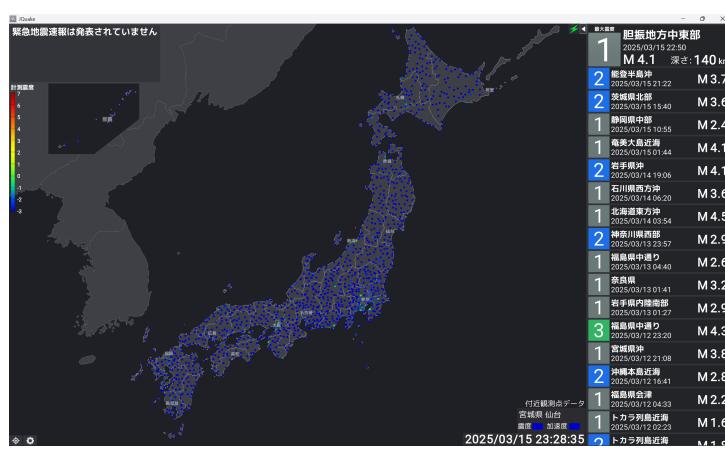


Figure 1.4.11: Main page for JQuake

However, due to the limitation of time, the following functionalities are excluded from the scope of this project, which include mostly the customisation parts, ranked in decreasing importance:

1. Shake detection algorithm (based on real-time sensor data)
2. User-Defined Key Monitor Point
3. (Customisable) Sound Alert
4. Customisable Colour Theme

The following features are considered too complicated for the scope for this project and the time constraints, hence not considered:

1. Replay (due to a server needing to store past data)
2. Sub-Map for the Okinawa Area (due to the difficulty in implementation)

With those key features implemented, the application should mirror all essential functionalities of an earthquake monitoring application, as compared to the four existing solutions above.

1.6 Critical Path

The functionality of this application can essentially be divided into $1 + 2 + 1 + a + b$ steps, where the 1 is to receive information from the API/WebSocket provided by DM-D.S.S. and NIED, and 2 is to, briefly saying:

1. Real-time Monitoring: Produce a real-time map to monitor real time vibration and plot real-time EEW/tsunami warnings
2. Past-earthquake Viewing: Produce a menu to select an earthquake to display information on the map.

The next 1 is to merge these two functions, specifically the functionality to switch back to the real-time monitoring option immediately when a new EEW is released (to make sure the user does not miss any information on real-time EEWs while looking at past earthquake information).

The a is to implement a setting page for customisation and some additional features, and the b is to implement additional functionalities that are excluded from this project, but could be completed if there is time.

1.6.1 Part 1. NIED and DM-D.S.S. API/WebSocket Connection

1. Investigate into the list of APIs/WebSockets that should be used by the application.
2. Investigate into patterns of the response data, and design the classes/DTOs using correct OOP patterns such as inheritance and composition.
3. Implement classes/DTOs to convert the information into C# objects.
4. Implement classes/DTOs to convert telegrams into C# objects.
5. Use API Key to test the functionality of this system.
6. Implement OAuth 2 login functionality to reduce complexity and increase security.

1.6.2 Part 2a. Real-time Monitoring Map

1. Display a map in the application.
2. Extract the data from the real-time measured intensities of observation stations from the data source.
3. Display the data on the map using a default colour scheme at the correct positions.
4. Colour the monitoring points using a colour scheme on the map.

5. Investigate into and implement an algorithm to calculate seismic wave fronts, using necessary data provided by the JMA.
6. Plot and colour real time EEWs, considering special cases such as:
 - Cancellation of EEW;
 - Upgrade from EEW Forecast to EEW Warning;
 - Multiple Earthquakes (hence displaying multiple epicentres, and switching between earthquakes in details display);
 - Assumed Hypocentre (hence using a different hypocentre symbol to display).
7. Colour areas on the map with the maximum expected intensity.
8. Plot on the map the hypocentre of the earthquake, using the correct symbol.
9. Plot and colour real time tsunami warnings.

1.6.3 Part 2b. Past-earthquake Viewing

1. Plot a sidebar of a list to display path earthquakes.
2. Provide a button to refresh the sidebar, and another one to load more earthquakes.
3. Provide a functionality to choose an earthquake from the sidebar to view the detailed information on the earthquake (e.g. magnitude, epicentre, time, additional message).
4. Plot the hypocentre of the earthquake.
5. Plot the observed maximum intensities on the map by regions.
6. Plot the intensities measured by each observation station on the map.
7. Provide external link to view earthquake in the weather service provider website.

1.6.4 Part 3. Joint functionality

1. Provide sidebar to switch between real-time monitoring and past-earthquake viewing, and settings options.
2. Implement automatic functionality to switch back to real-time earthquake when an EEW/Tsunami Warning is being published.

1.6.5 Part 4. Setting page for customisation

1. Provide a text box to input the API Key, and a button to confirm the API Key.
2. Provide a button to connect and authorise with OAuth 2.0.
3. Provide a button to connect to the WebSocket.
4. Display a list of current WebSocket connections, and provide a button to refresh such list.
5. Provide two dropdown menus, to choose the sensors to display on the real-time map (all sensors, or borehole only), and the data to display (real time intensity, PGA, PGV, PGD, etc.).

1.6.6 Part 5. Additional functionalities

1. Investigate into and implement an algorithm to detect shakes based on the observed data.
2. Provide functionality to define a specific point on the map, and display the measured intensity at that point.
3. Play sound alerts when EEW/Tsunami warnings are received.
4. Provide custom sound alert settings to specify the file to the sound alert.
5. Provide custom colour theme settings to specify the colour scheme of the colouring on the map.

1.7 Requirements Specification

A detailed specification is defined here that is to be aimed to be fulfilled at the end of the project, split into four sections:

1. **NIED and DM-D.S.S. Functionality** – Corresponding to **Part 1**, in Table 1.7.2
2. **Real-Time Earthquake Monitoring** – Corresponding to **Part 2a**, in Table 1.7.3
3. **Past-earthquake Viewing** – Corresponding to **Part 2b**, in Table 1.7.4
4. **Joint functionalities** – Corresponding to **Part 3**, in Table 1.7.5
5. **Settings Page** – Corresponding to **Part 4**, in Table 1.7.6
6. **Additional functionalities** – Corresponding to **Part 5**, also in Table 1.7.6, marked with a *

Those objectives marked with an * are optional depending on time, since they are not core functionalities for the project.

The objectives here are SMART, meaning they are specific, measurable, achievable, realistic and timely.

Table 1.7.1 gives abbreviations for the types of testing.

Test Method	Abbr.
Unit Testing	UT
Integrated Testing	IT

Table 1.7.1: Measurement methods

Req. №	Description	Success Criteria	Test
1(i)	Connect to DM-D.S.S. using an API Key	Included in header for requests	UT
1(ii)	Call HTTP-Based APIs	Calls to correct URLs with specified parameters where applicable	UT
1(iii)	Connect to WebSocket Data Feed	Successful connection with regular pings/pongs	IT
1(iv)	Obtain stable connection on WebSocket	Connected for 1 hour under reliable internet connection	IT
1(v)	Parse API calls JSON into C# objects	Information parsed with correct values	UT
1(vi)	Recognise schema for JSON Telegrams	Correct JSON schema recognised	UT
1(vii)	Parse JSON Telegrams into C# objects	Parsed into correct type with correct values	UT
1(viii)	Exception handling for incorrect JSON	Exceptions thrown	UT
1(ix)	Exception handling for failed connections	Exceptions thrown	UT
1(x)	Integrated DM-D.S.S. functionalities	Correct objects created	IT
1(xi)	Connect to DM-D.S.S. using OAuth2	Access token included in header for requests	IT
1(xii)	Refresh tokens when expired using OAuth2	New access token successfully obtained with the refresh token	IT

Table 1.7.2: Requirements for Part 1

1.8 Initial Modelling

Some initial ideas on the applications are considered in this section, but more detailed design will be reflected in the GUI design section in the next chapter.

Req. №	Description	Success Criteria	Test
2a(i)	Display real-time observed data on map	Coloured points at correct locations	IT
2a(ii)	Display current time at the corner	Time displayed and refresh every second	IT
2a(iii)	Display details of current EEW	Details displaying correctly	IT
2a(iv)	Plot epicentre of EEW on map	Epicentre at correct position with correct symbol	IT
2a(v)	Calculate seismic wavefronts position	Correct result from dataset	UT
2a(vi)	Plot seismic wavefronts	Correct wavefronts at correct positions	IT
2a(vii)	Colour expected max. intensity of regions	Correctly coloured	IT
2a(viii)	Update/Cancel EEW when appropriate	Update reflecting on both the details and the map	IT
2a(ix)	Handle multiple simultaneous EEWs	Switch between details of EEWs at regular intervals	IT
2a(x)	Display tsunami warnings when issued	Details displaying correctly	IT
2a(xi)	Colour shorelines from tsunami warnings	Coloured at correct location	IT

Table 1.7.3: Requirements for Part 2(a)

Req. №	Description	Success Criteria	Test
2b(i)	Display past-earthquake side list using data from DM-D.S.S.	Displayed and updated	IT
2b(ii)	Refresh button to refresh the latest earthquakes	Loaded and displayed	IT
2b(iii)	Load more button to load extra earthquakes	Loaded and displayed	IT
2b(iv)	Display the time, location, depth, intensity and magnitude of earthquake on the side list	Correctly formatted and displayed	IT
2b(v)	Select an earthquake item from list	Selection functioning	UT
2b(vi)	Display details of selected earthquake on sidebar	Correctly formatted and displayed	IT
2b(vii)	Colour map by the maximum intensity of regions of selected earthquake	Correctly coloured	IT
2b(viii)	Plot hypocentre of selected earthquake	Correctly plotted	IT
2b(ix)	Plot observed maximum intensities of observation stations of selected earthquake	Correctly plotted and coloured	IT
2b(x)	Provide link to external weather services for details	Linked to correct website and earthquake	IT

Table 1.7.4: Requirements for Part 2(b)

Req. №	Description	Success Criteria	Test
3(i)	Provide sidebar to switch between real-time, past earthquake and settings	Sidebar functioning and able to switch pages	IT
3(ii)	Sidebar has an expansion/hide button to show/hide names	Button working as described	IT
3(iii)	Switch to real-time monitoring map when EEWs and tsunami warnings are issued	Page switched	IT

Table 1.7.5: Requirements for Part 3

Req. №	Description	Success Criteria	Test
4(i)	Provide with functionality to set API Key or connect to OAuth2 to authenticate	Authentication functioning	IT
4(ii)	Provide button to connect and disconnect from WebSocket	Button functioning	IT
4(iii)	Provide button to view a list of WebSocket connections	List displayed	IT
4(iv)	Provide button in the list to disconnect a particular WebSocket	Button functioning to disconnect	IT
4(v)	Provide dropdown menus to select sensors and data to display on the map	Correct data displayed on the real-time map	IT
4(vi)	Provide an easy-to-use GUI	Potential user gives rating of more than 7 out of 10	UT
4(vii)*	Provide page to design colour scheme	Functional design page and applied to the whole UI	UT
4(viii)*	Play sound when events happen and provide option to customise sound files	Correct sound played when required	IT
4(ix)*	Provide option to define a point on the map to monitor	Functioning as specified in Part 2a	IT
4(x)*	Display real-time shake data at a user-defined point	Display the name of the point with acceleration and intensity	IT
4(xi)*	Provide sub-map to display Okinawa Area	Map functioning as specified in Part 2	IT

Table 1.7.6: Requirements for Part 4 and 5

1.8.1 Colour Scheme to Use

Although most built-in applications tend to introduce their own colour schemes, the author decided to follow the standard guidance provided by the JMA [73] and use the official colour scheme, to prevent any confusion and make the application generally accessible.

Figure 1.8.1 shows the JMA Colour Scheme for measured intensities, and Figure 1.8.2 shows the JMA Colour Scheme for different levels of tsunami warnings (the row for the tsunami is highlighted).

表 2-2 数値が高くなるほど危険度が増す情報の配色（震度）								
使用する基本配色 (RGB 値)	■ 180,0,104	■ 165,0,33	■ 255,40,0	■ 255,153,0	■ 255,230,0	■ 250,230,150	■ 0,65,255	■ 0,170,255
震度	震度7	震度6強	震度6弱	震度5強	震度5弱	震度4	震度3	震度2

Figure 1.8.1: JMA Colour Scheme for Measured Intensities

In addition to these colours, for tsunami informational advisory, the colour 80FFFF ■ is used, in accordance with the colour used for no release of river flooding warning, and for endings of tsunamis warnings, the colour F2F2FF ■ is used, in accordance with the minimal level (the extremely light grey) in Figure 1.8.2.

The colour F0F0F0 ■ is used for intensity zero (i.e. any intensity less than intensity 1), and C8C8CB ■ is used for unknown intensities.

A similar colour scheme is used for EEW warnings. Specifically, the same colour as tsunami warning is used for EEW warnings, and the same colour as endings of tsunami warnings is used for EEW cancellation. For EEW forecasts, the colour FFAA00 ■ (in accordance with the orange colour in Figure 1.8.2) is used, and for EEW final reports, the same colour is used as unknown intensities.

A summary of the colours used is in the table below.

1.8.2 Details to Display

Comparing the information provided by the different applications, the following information about an earthquake will be displayed in the sidebar, as consistent with most applications analysed:

- Maximum observed intensity of earthquake;

表 1-2 警報・注意報及びこれに類する情報の配色（台風・津波・噴火等の情報）

使用する基本配色 (RGB 値)	← 警戒度 大 →					発表なし 200,200,203
	200,0,255	255,40,0	255,170,0	250,245,0	242,242,255	
台風情報(経路図)		暴風域		強風域		
台風情報(暴風域に入る確率) 時間変化	70~100%	30~70%		5~30%		
台風情報(暴風域に入る確率) 分布表示				発表中		発表なし
海上警報	台風警報	暴風警報	強風警報	風警報		
大津波警報・津波警報・津波注意報	大津波警報(幅2倍)	津波警報		津波注意報		
噴火警報・予報 (噴火警戒レベル導入火山)	レベル 5 避難	レベル 4 避難準備	レベル 3 入山規制	レベル 2 火口周辺規制	レベル 1 活火山であることに留意	
噴火警報・予報 (噴火警戒レベル未導入火山)	居住地域 厳重警戒		入山危険	火口周辺危険	活火山であることに留意	
噴火警報・予報 (海底火山)			周辺海域警戒		活火山であることに留意	
レーダー・ナウキヤスト(雷)	活動度4	活動度3	活動度2	活動度1		
レーダー・ナウキヤスト(竜巻)		発生確度2		発生確度1		

Figure 1.8.2: JMA Colour Scheme for Tsunamis
Purple: Special Warning; Red: Tsunami Warning; Yellow: Tsunami Caution

Intensity	Colour	EEW Level	Colour	Tsunami Warning Level	Colour
?	C8C8CB				
0	F0F0F0				
1	F2F2FF				
2	00AAFF	Forecast	FFAA00	Information	80FFFF
3	0041FF	Warning	FF2800	Caution	FAF500
4	FAE696	Final	C8C8CB	Warning	FF2800
5-	FFE600	Cancelled	F2F2FF	Special Warning	C800FF
5+	FF9900			None	F2F2FF
6-	FF2800				
6+	A50021				
7	B40068				

Table 1.8.1: Colour Scheme for Application

- Time of the earthquake;
- Name of position of hypocentre;
- Depth of hypocentre;
- Magnitude of the earthquake.

In addition to these, on the side panel, the following information will be displayed in addition to the previous ones:

- Last updated time of information;
- The tree of intensities measured by observation stations, in hierarchy structure of:
 1. Intensity; then by
 2. Prefecture; then by
 3. Regions; then by
 4. Cities/Districts; finally
 5. Observation Stations.
- Any additional text provided by the JMA on the report, where applicable.

Note there is an extra level of regions in the hierarchy tree, compared to KEVI.

As for the EEW warning, there are slightly different information displayed, due to different information provided, and the nature of the EEW. The following information will be displayed:

- The level of the EEW warning;
- The serial number of the EEW warning,
- Maximum predicted intensity of earthquake;
- Origin time of the earthquake;
- Name of position of hypocentre;
- Warning text if the hypocentre is assumed (as a JMA requirement);
- Depth of hypocentre;
- Magnitude of the earthquake;
- Precision/Accuracy of information on hypocentre, depth and magnitude;
- Warning text if the earthquake has low accuracy (as a JMA requirement);
- Last updated time of the information,
- Any additional text provided by the JMA on the report, where applicable.

Note that (as a JMA requirement), if the hypocentre is assumed for algorithmic reasons, then the hypocentre details and the magnitude have no seismological meaning, and the information on origin time, the position of hypocentre, the depth of hypocentre and the magnitude of the earthquake will be hidden

The information displayed for a tsunami warning will be relatively simple:

- The level of the tsunami warning;
- Any additional text provided by the JMA on the report, where applicable.

Chapter 2

Design

Overview

This section includes a breakdown of the application into sections of data processing, GUI functionalities and joint functionalities. The use of external sources including DM-D.S.S. and NIED data sources is discussed, together with the relevant formats (JSON) and the objects related. UML class diagrams are included to discuss OOP relations of classes, records (record classes) and enums including the use of inheritance, composition, association and aggregation. They also implement different interfaces. Design patterns that should be included in the application are also introduced and outlined. An outline of the design of the user interface is included. The expected hardware requirements of the systems are also listed, but any laptop with an up-to-date operating system (running Windows or macOS) should be able to run the program.

2.1 Hierarchy Chart

As discussed in the analysis section, the program consists of three parts: data-parsing from external data sources, GUI functionalities and joint functionalities, where the GUI part will be divided into two parts focusing on real-time monitoring and past-earthquake information, respectively. Here, functionalities for each particular module is further split up. Figure 2.1.1 is a hierarchy diagram for the whole application. This shows how **decomposition** technique is applied to reduce a sophisticated problem into more attackable problems.

Since the GUI part is quite complicated, it is included in this separate diagram, in Figure 2.1.2.

The whole application (apart from the polynomial fitting part which was done in Python) will be written in C# using the .NET Core Version 9.0, since the .NET Core is designed oriented around OOP techniques, and some JSON properties (e.g. the `JsonStringEnumMemberName` to customise the parsing of enums) will require the use of .NET 9.

2.2 External Data Sources

There are two data sources this program will use: the NIED and the DM-D.S.S. Specifically, the former one is used to achieve the real-time shake data of the sensor points which were set up by the government (whose data is free to use), and the latter one is used to achieve past earthquake information and EEW information sent out by the JMA (which is pay-to-use). Note that DM-D.S.S. does also provide the real-time intensity data of the observation points, however it is pay-to-use only for companies and institutions on request. Therefore, it will not be feasible to use this data source in the program since one of the principle target users is people passionate in monitoring earthquakes.

2.2.1 NIED Data Source

As mentioned before, NIED has numerous 'earthquake observation nets' across Japan. Specifically, there is the K-NET and KiK-net [80], which is dedicated to the observation of strong seismic motion. The K-NET consists of approximately 1000 sensors located across Japan, while the KiK-net also includes some sensors which are located within the earth, which will often have different readings compared to those

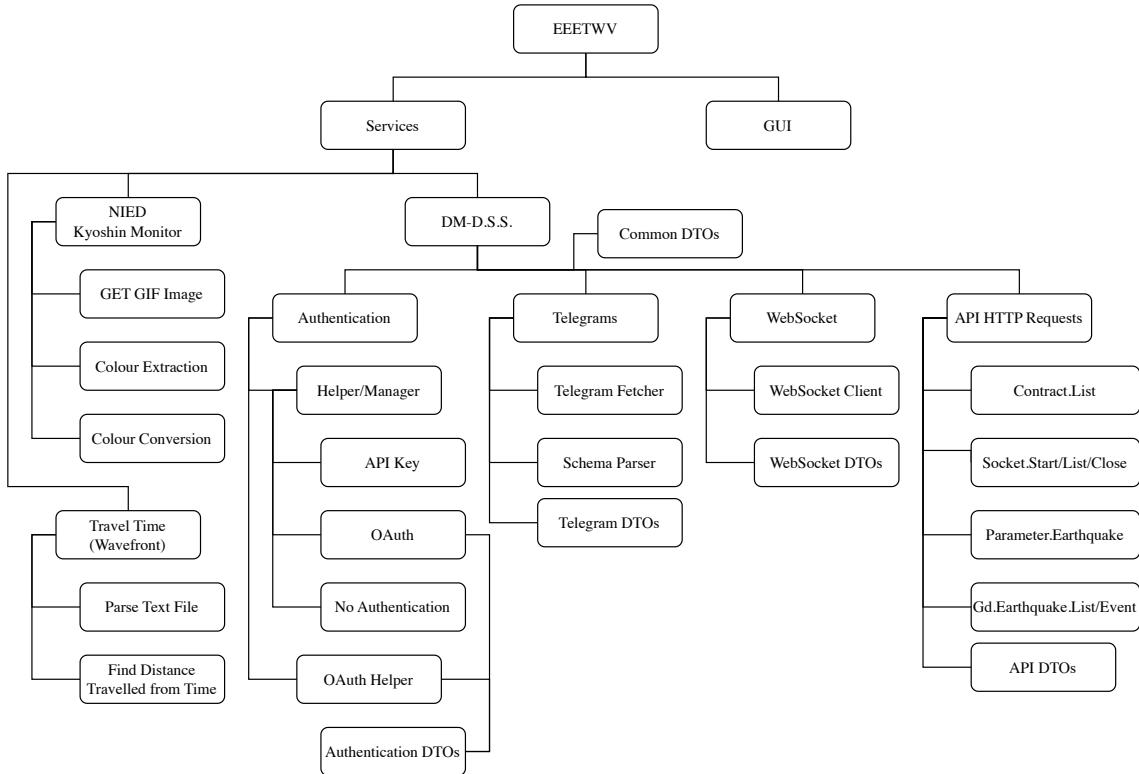


Figure 2.1.1: Hierarchy chart of the whole application

located on the surface. They are extremely capable of detecting strong motion of ground. Furthermore, the K-NET and the KiK-net provides real-time intensity data webpage of two types, the 'Kyoshin' (強震) monitor and the long-period ground motion (LPGM, 長周期地震動) monitor (not working at the time of investigation). The Hi-net [79] stands for high-sensitivity seismograph network, and it is dedicated for observation of minor motions of the ground. They release the waveforms to those who are researching seismic movements. As for the F-net [78] which stands for the Full Range Seismograph Network of Japan, which is used to analyse the mechanism of a certain earthquake by analysing movements. None of the three nets provide a real-time API data feed.

Having compared the functionalities described above of the K/KiK-net, Hi-net and F-net and how they feed the data sources, the most suitable data source to reflect real-time motion of ground movements will be the **K/KiK-net**'s data feed, since it detects strong ground movements and is available real-time for the purpose of the application. (This is also the data source that JQuake and KEVI use in fact.)

In fact, in addition to these three networks, there are also the S-net, the DONET and the N-net, which detects the ground seismic movements in the sea. These data were adapted by SREV, but this is beyond the scope of this NEA analysis.

A comparison from the official website of MOWLAS (Monitoring of Waves on Land and Seafloor) [81] of the three nets are included in Figure 2.2.1 and a map of the distribution of the sensors are included in Figure 2.2.2.

2.2.1.1 Achieving image format data source

The data source fed by the 'Kyoshin' Monitor is split into 8 types (detailed below in Table 2.2.1), and each type split into 2 types of data sources, surface sensors and borehole (earth) sensors, with codes in Table 2.2.2. The link to the GIF image is in the following format:

`http://www.kmoni.bosai.go.jp/data/map_img/RealTimeImg/[#1]_[#2]/[yyyyMMdd]/[yyyyMMdd][hhmmss].[#1]_[#2].gif`

In the link, the [yyyyMMdd] and the [hhmmss] part should be replaced with the date and time respectively (in JST, UTC+8), and the #1 replaced with the codes detailed below for the data types,

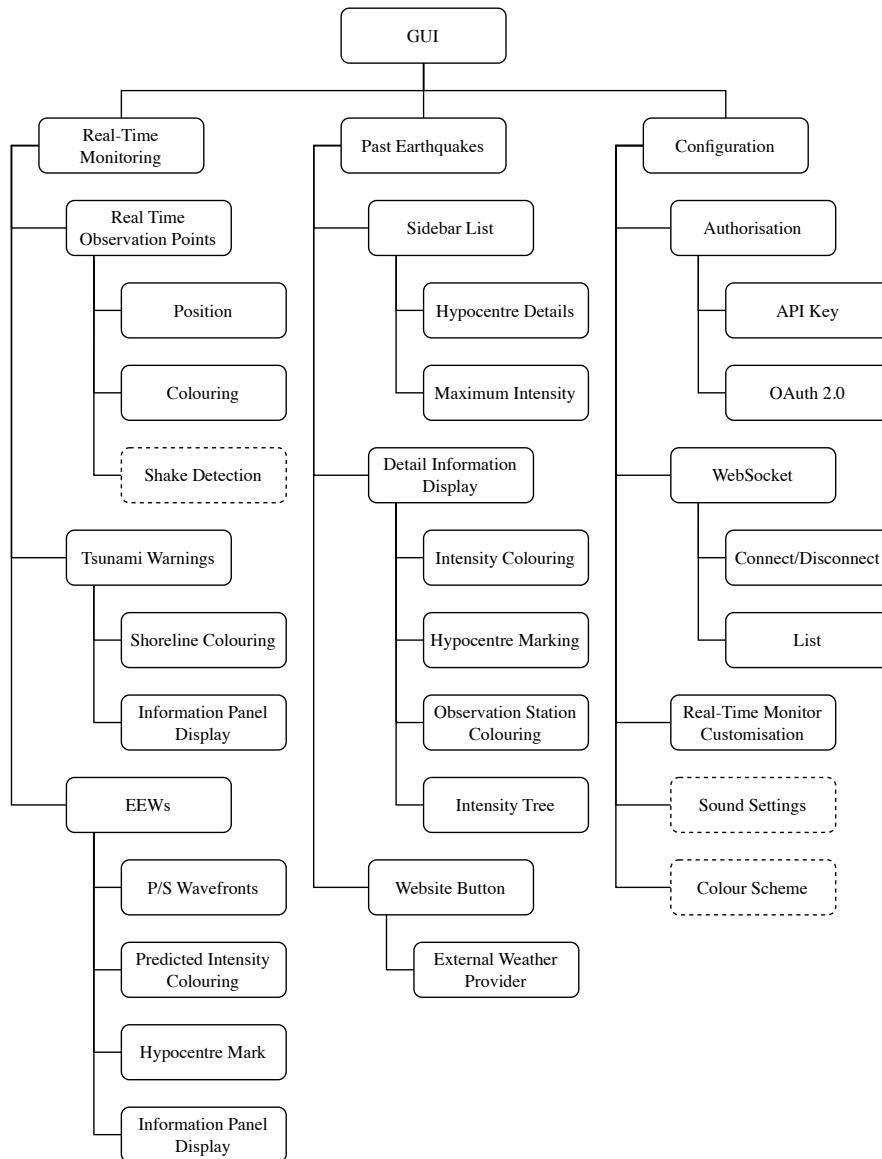


Figure 2.1.2: Hierarchy chart of the GUI Component

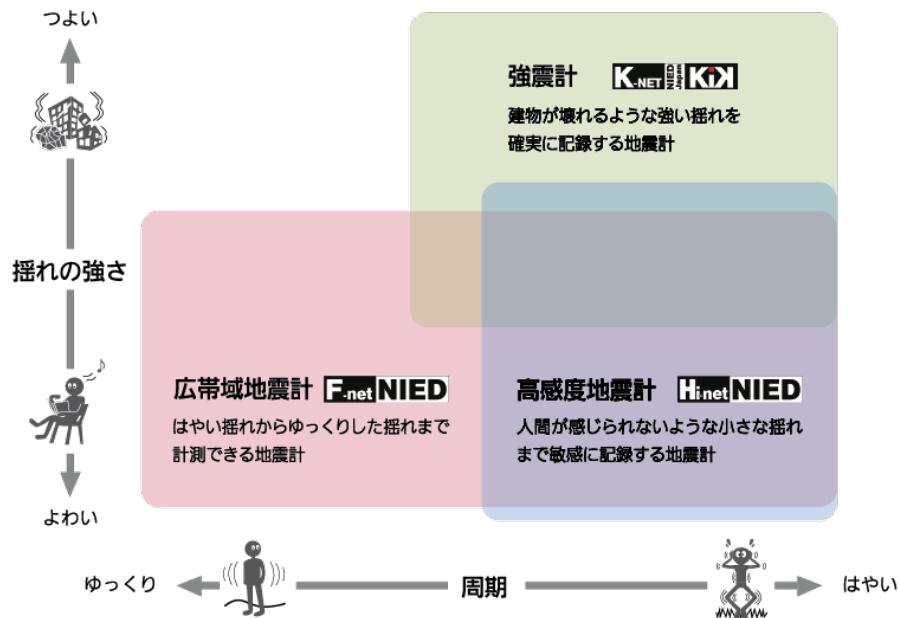


Figure 2.2.1: A comparison of the K-NET, F-net and Hi-net



Figure 2.2.2: Distribution of the sensors of different nets

and #2 replaced with the codes detailed below for data sources. An example of the imaged achieved is in Figure 2.2.3.

Data Type	Description/Meaning	Code in #1
Real-time Shindo	Real-time Measured Intensity	jma
PGA	Peak (Maximal) Ground Acceleration	acmap
PGV	Peak (Maximal) Ground Velocity	vcmap
PGD	Peak (Maximal) Ground Displacement	dcmap
Response 0.125Hz	Response spectrum for 0.125Hz PGV	rsp0125
Response 0.250Hz	Response spectrum for 0.250Hz PGV	rsp0250
Response 0.500Hz	Response spectrum for 0.500Hz PGV	rsp0500
Response 1.000Hz	Response spectrum for 1.000Hz PGV	rsp1000
Response 2.000Hz	Response spectrum for 2.000Hz PGV	rsp2000
Response 4.000Hz	Response spectrum for 4.000Hz PGV	rsp4000

Table 2.2.1: Data available in 'Kyoshin' monitor

Sensor Type	Description/Meaning	Code in #2
Surface	K-NET and KiK-net sensors	s
Borehole	KiK-net sensors within earth	b

Table 2.2.2: Sensors available in 'Kyoshin' monitor

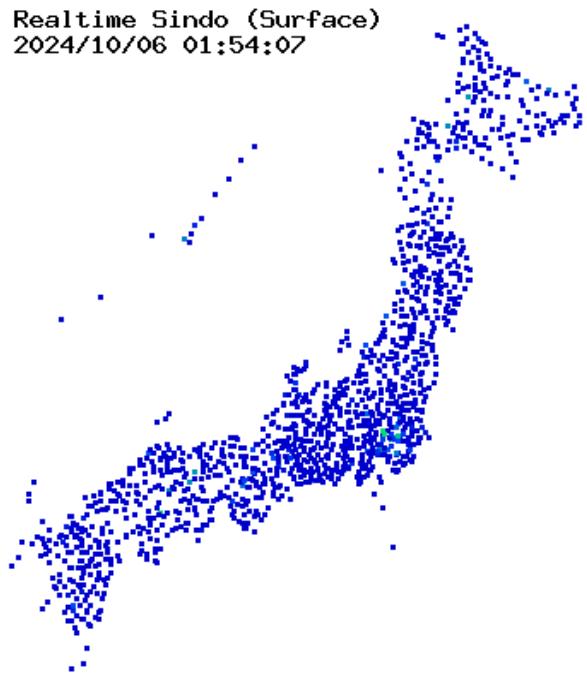


Figure 2.2.3: Sample GIF image achieved from 'Kyoshin' Monitor

2.2.1.2 Extracting colour for each observation point

Unfortunately, it seems to the author (and is widely accepted in the EEW monitoring app development society) that the position of the points (squares) on the image does not follow any significant pattern of position, i.e. there is no obvious conversion of coordinates to us from the official longitude/latitude locations to the positions on the image. Therefore, a manual conversion one-to-one mapping has to be developed.

NIED does have an official released list of observation points, which include their names and positions. This list has around 1700 of those observation points. However, in the actual image (like those in

Figure 2.2.3), there are only 1000 of those in use in real time, consistent with K-NET's official introduction, and the rest 700 of those are invalid observation points. Therefore, it will be worth removing them from the list of earthquake monitoring points, before attempting to make the dictionary.

Unfortunately, 1000 is still quite a lot for us to deal with. Luckily, Ingen who used a similar approach to develop the KEVI application has already made such a mapping inside his open-source application in the file ShindoObsPoints.mpk.lz4 within [Q ingen084/KyoshinEewViewerIngen](#), and even developed an editor for this at [Q ingen084/KyoshinShindoPlaceEditor](#).

Due to the limited time for this NEA, the author will primarily use the pre-determined observation points for the K-NET and the KiK-net by Ingen in JSON format.

This paragraph referred to [35].

2.2.1.3 Converting colour to number format for further processing

The true numerical data does not seem fully necessary at the first glance (since we might just as well just achieve the colour from the image and just plot them on the map, without the need to convert to a colour and back). However, for us to detect the shake in certain regions, it is necessary for us to achieve the numerical value to run the algorithm on it. Nevertheless, it is just good to have the number for us to have the numerical value for potential future developments.

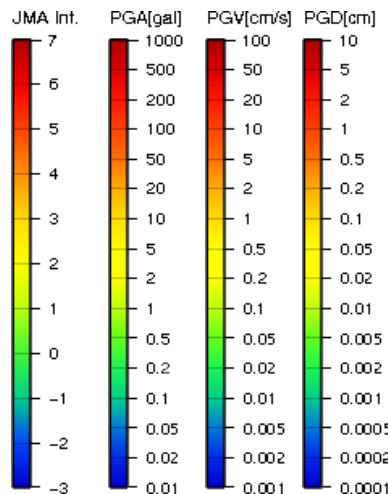


Figure 2.2.4: Scale colours of different measurements

As shown in Figure 2.2.4, the NIED 'Kyoshin' Monitor does indeed provide a scale of colours and reference to numerical values. However, there is a chance that a certain colour is not 'exactly' mapped on the scale, and further concerning that it is very slow and difficult to 'loop over' a colour legend, it is necessary to have an algorithmic-approach (numerical mapping-based approach) to map the colours in the colour space to numerical values (and back) is necessary.

Abstraction of colour scale Notice that the scale for PGA/PGV/PGD follow a logarithmic scale, while measured intensity follows a linear scale (though noting that the way intensity and magnitude is calculated is logarithmic as well). Therefore, if we normalise the vertical distance from the bottom of the axis h to $0 \leq h \leq 1$ (i.e. $h = 0$ at the bottom of the scale, $h = 1$ at the top of the scale), and if we denote intensity using I in JMA scale, PGA as a in gal, PGV as v in cm per second, and PGD as s in cm, from the scale, the following transforming formulae obviously hold:

$$\begin{aligned} I &= 10h - 3 \iff h = \frac{I + 3}{10}, \\ a &= 10^{5h-2} \iff h = \frac{\lg a + 2}{5}, \\ v &= 10^{5h-3} \iff h = \frac{\lg v + 3}{5}, \\ x &= 10^{5h-4} \iff h = \frac{\lg x + 4}{5}. \end{aligned}$$

However, it is worth noting that NIED did use 1, 2, 5, 10 on the logarithmic scale at equal intervals, so it is not a perfect logarithmic scale. The author is unsure why they designed the scale like this, nor if it's an intended approximation. Nevertheless, the logarithmic scale is a good enough approximation.

The next step is to develop a mapping from this colour space \mathcal{C} to h , which of course should be invertible. Denote this as $f : [0, 1] \rightarrow \mathcal{C}$.

Describing colour numerically We consider using a suitable base to decompose \mathcal{C} . The colour of the given scale is an immediate suggestion to use a base containing **hue**, which in fact is designed to describe how human perceive colour, and unlike RGB and CMYK which uses principle colours to describe colour. A hue scale is shown in Figure 2.2.5.

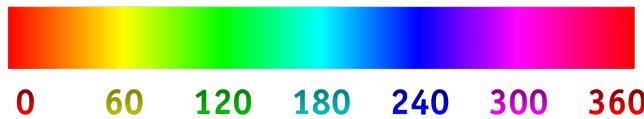


Figure 2.2.5: The hue scale in HSL/HSV encoding

Therefore, a colour in the colour space \mathcal{C} can be represented as a 3-D vector $\mathcal{C} \ni C = (H, S, V)$, where $H \in [0, 360]$ in degrees is the hue value, $S \in [0, 1]$ stands for the saturation, and $V \in [0, 1]$ stands for the value (a brightness). And hence we will be able to decompose f into three components $f = (f_H, f_S, f_V)$.

Figure 2.2.6 plots the values of H, S and V against h (this is the graph of f and its components) of discrete values of h , and depending on the result we will attempt some fit/regression to a suitable function.

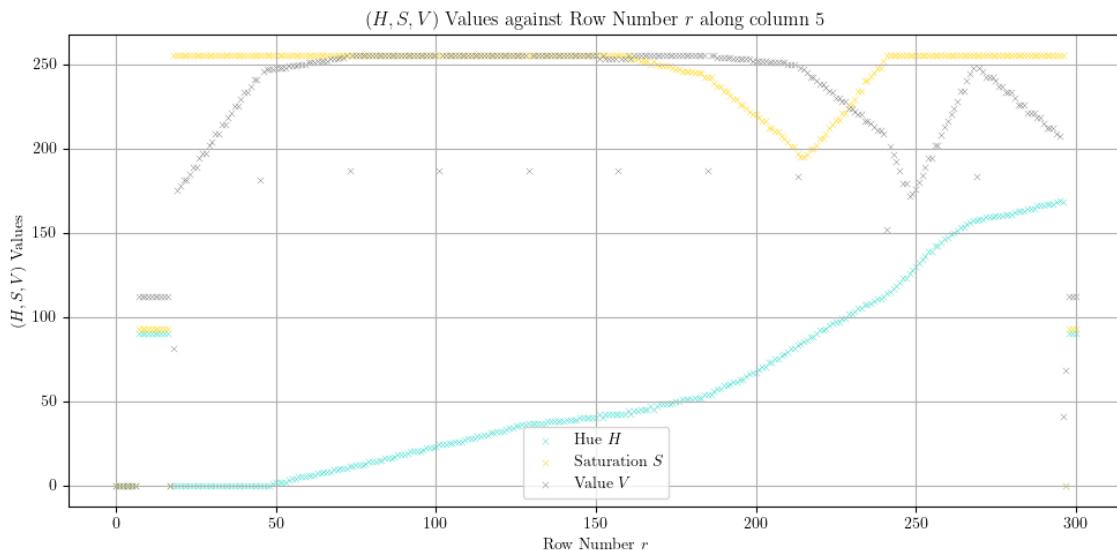


Figure 2.2.6: The values of (H, S, V) against pixel row r

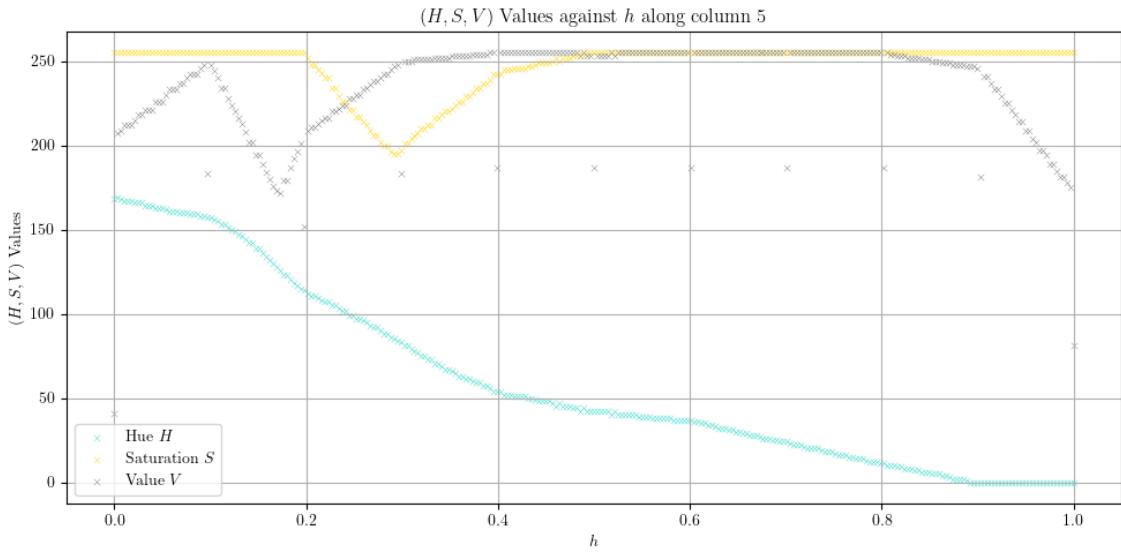
Notice that in this plot, all values of (H, S, V) in fact range from 0 to 255, as in an 8-bit binary.

It is worth noting that the scale has some space on the top (to show the type), and some space at the bottom. Notice that when the row $r = 17$ and $r = 297$ have values significantly different, so we extract the rows $r = 18$ and $r = 296$ to correspond (linearly) to $h = 1$ and $h = 0$, i.e.,

$$h = 1 - \frac{r - 18}{278}.$$

Figure 2.2.7 shows the result of this transformation being applied.

From here onwards, all values of (H, S, V) will be adjusted to be within the range which they should be in, i.e. $H \in [0, 360], S \in [0, 1], V \in [0, 1]$.

Figure 2.2.7: The values of (H, S, V) against normalised height h

Finding f_H in terms of h We consider finding f_H first, which is the cyan line. Notice that its trend can be split into 4 parts:

- $h \in [0, 0.1]$: linear;
- $h \in [0.1, 0.6]$: curving, ideally a cubic;
- $h \in [0.6, 0.9]$: linear;
- $h \in [0.9, 1]$: constant (0).

h	$H = f_H(h)$
0	237
0.1	222
0.6	51
0.9	0
1	0

Table 2.2.3: Initial values for f_H

Furthermore, boundary conditions in Table 2.2.3 are applied to ensure that the function is continuous and nicely-behaving while matching the existing data. We use the following function to apply the fit:

$$f_H(h) = \begin{cases} -150h + 237, & h \in [0, 0.1], \\ \odot, & h \in [0.1, 0.6], \\ -170h + 153, & h \in [0.6, 0.9], \\ 0, & h \in [0.9, 1]. \end{cases}$$

Here,

$$\begin{aligned} \odot &= \frac{222 \cdot (h - 0.3) \cdot (h - 0.4) \cdot (h - 0.6)}{(0.1 - 0.3) \cdot (0.1 - 0.4) \cdot (0.1 - 0.6)} \\ &+ \frac{y_1 \cdot (h - 0.1) \cdot (h - 0.4) \cdot (h - 0.6)}{(0.3 - 0.1) \cdot (0.3 - 0.4) \cdot (0.3 - 0.6)} \\ &+ \frac{y_2 \cdot (h - 0.1) \cdot (h - 0.3) \cdot (h - 0.6)}{(0.4 - 0.1) \cdot (0.4 - 0.3) \cdot (0.4 - 0.6)} \\ &+ \frac{51 \cdot (h - 0.1) \cdot (h - 0.3) \cdot (h - 0.4)}{(0.6 - 0.1) \cdot (0.6 - 0.3) \cdot (0.6 - 0.4)}. \end{aligned}$$

Here, m_1 is the gradient of the line for $h \in [0, 0.1]$, $y_1 = f_H(0.3), y_2 = f_H(0.4)$ for $h \in [0.1, 0.6]$ (using Lagrange Polynomial), and the equation between $h \in [0.6, 0.9]$ is in fact fixed due to the initial conditions.

By applying a curve fit to the original data, the following results are obtained:

$$(y_1, y_2) = (115, 79.5).$$

Plotting H and $f_H(h)$ against h gives us Figure 2.2.8, which is decent.

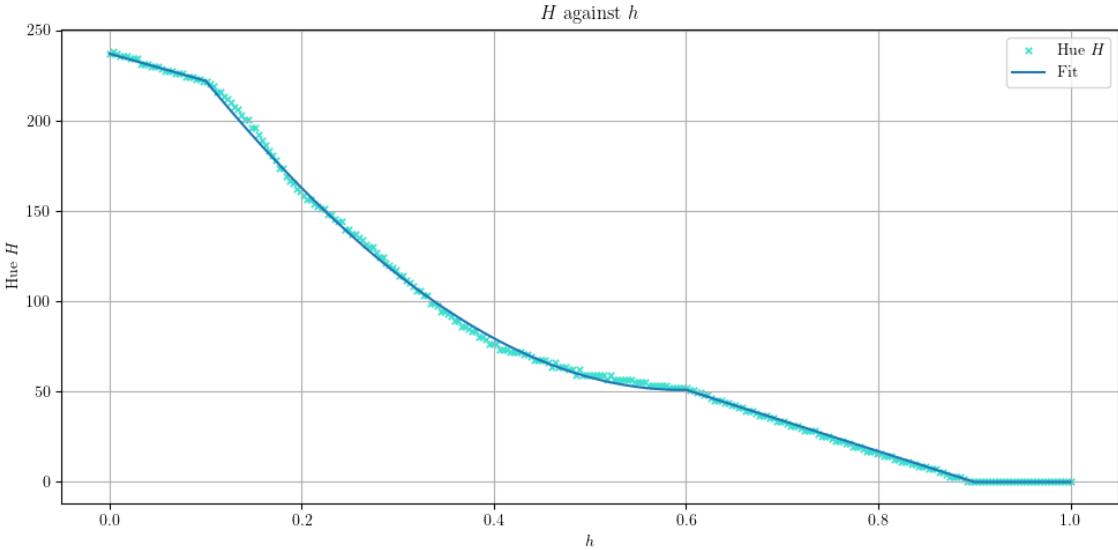


Figure 2.2.8: The fit result for $f_H : h \mapsto H$

Finding f_S in terms of h As for $f_S(h)$, the obvious thing to do is to split it into 5 (4) piecewise functions, specifically $f_S = 1$ for $h \in [0, 0.2] \cup [0.5, 1]$, and three linear functions for $h \in [0.2, 0.29], h \in [0.29, 0.4]$ and $h \in [0.4, 0.5]$. Initial values are included in Table 2.2.4.

h	$S = f_S(h)$
0	1
0.2	1
0.29	0.765
0.4	0.95
0.5	1
1	1

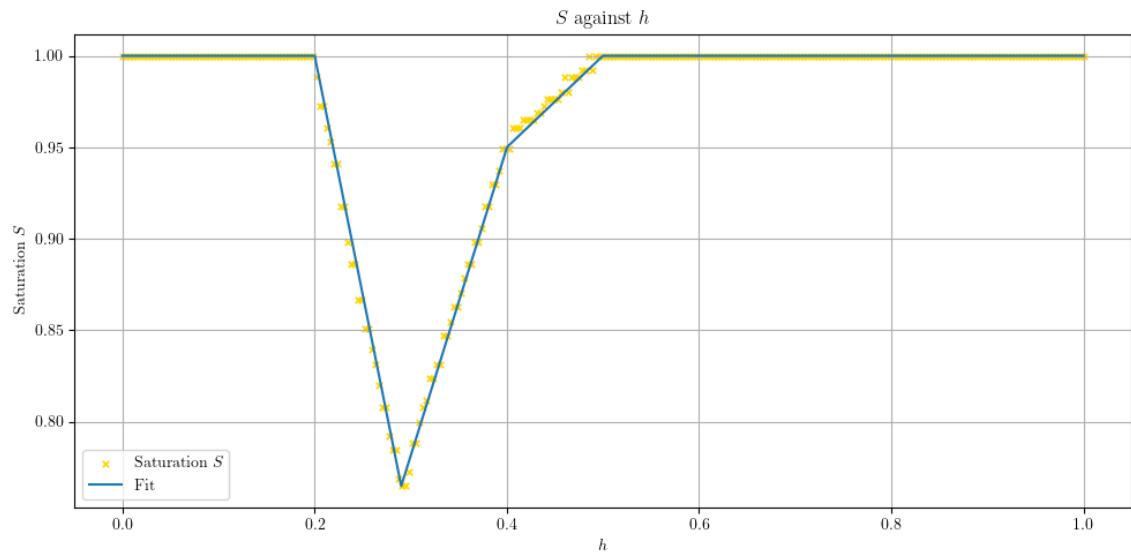
Table 2.2.4: Initial values for f_S

This gives us that

$$f_S(h) = \begin{cases} 1, & h \in [0, 0.2], \\ -2.611h + 1.522 & h \in [0.2, 0.29], \\ 1.682h + 0.277, & h \in [0.29, 0.4], \\ 0.5h + 0.75 & h \in [0.4, 0.5], \\ 1, & h \in [0.5, 1]. \end{cases}$$

Plotting this out gives Figure 2.2.9.

Finding f_V in terms of h As for $f_V(h)$, we shall divide it into even more piecewise linear functions. Specifically, I chose to divide the interval $[0, 1]$ at 0.1, 0.172, 0.2, 0.3, 0.4, 0.8 and 0.9. Initial values are included in Table 2.2.5.

Figure 2.2.9: The fit result for $f_S : h \mapsto S$

h	$V = f_V(h)$
0	0.8
0.1	0.98
0.172	0.66
0.2	0.82
0.3	0.98
0.4	1
0.8	1
0.9	0.97
1	0.68

Table 2.2.5: Initial values for f_V

This gives us the piecewise function

$$f_V(h) = \begin{cases} 1.8h + 0.8, & h \in [0, 0.1], \\ -4.444h + 1.424, & h \in [0.1, 0.172], \\ 5.714h - 0.323, & h \in [0.172, 0.2], \\ 1.6h + 0.5, & h \in [0.2, 0.3], \\ 0.2h + 0.92, & h \in [0.3, 0.4], \\ 1, & h \in [0.4, 0.8], \\ -0.3h + 1.24, & h \in [0.8, 0.9], \\ -2.9h + 3.58, & h \in [0.9, 1]. \end{cases}$$

Plotting this out gives us Figure 2.2.10.

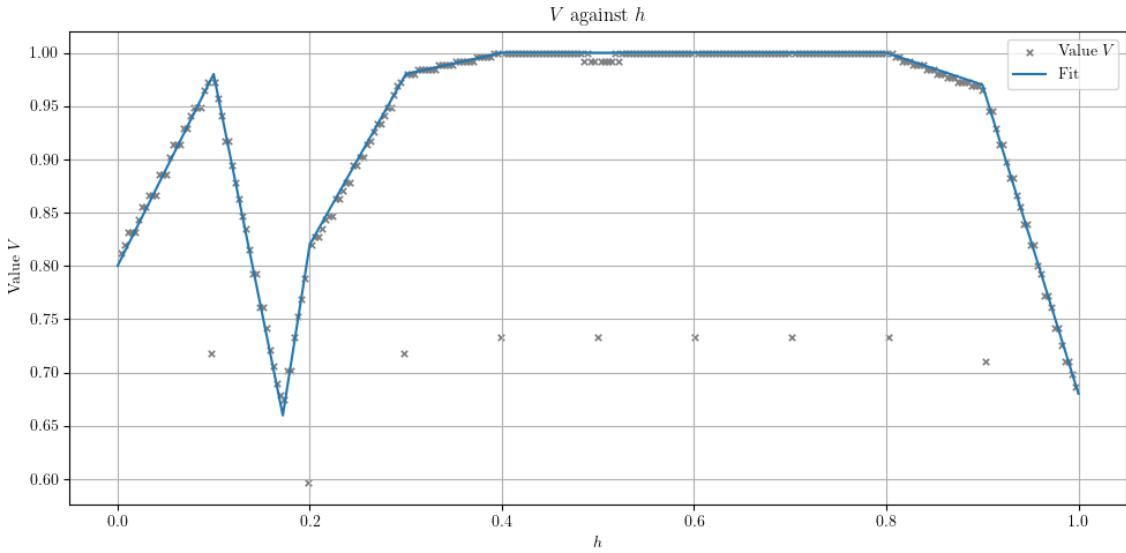


Figure 2.2.10: The fit result for $f_V : h \mapsto V$

Note that in this plot, V when $h = 0$ or $h = 1$ is excluded, since just like every $h = 0.1k$ for some $k \in \mathbb{N}$, they are anomalies created by the horizontal black line in the scale.

Finding f^{-1} To find $f^{-1} : \mathcal{C} \rightarrow [0, 1]$, we do not need necessarily to find an expression of h in terms of (H, V, S) . If we notice that f_H is one-to-one on $h \in [0, 0.9]$, and f_V is one-to-one on $h \in [0.9, 1]$, we can use f_H^{-1} to determine h from H only if H is non-zero, and use f_V^{-1} otherwise.

$$f^{-1}(H, S, V) = \begin{cases} f_H^{-1}(H), & H \neq 0, \\ f_V^{-1}(V), & H = 0. \end{cases}$$

Notice that for $h \in [0.1, 0.6]$, f_H is a cubic and is not easily invertible. However, it would be plausible to use a binary-search algorithm to find h based on H since it is monotonic, and it is within a reasonable amount of time, to relatively good precision. Otherwise, on the linear parts, it is fine to simply mathematically invert it.

Algorithm 2.2.1 describes the logic. Note that, this algorithm uses different V ranges from before (which is consistent with SkiaSharp library in C#), and the ranges is $V \in [0, 100]$. Notice that, $\epsilon(h)n$ and $\epsilon(H)$ are two constant values that determines the precision of such binary search, and they are chosen to be 0.01 and 0.5 in the implementation.

2.2.1.4 Flowchart of data and sidenotes

To summarise, we discussed the mapping from the colour space \mathcal{C} to the normalised height h , and back, and we also discussed how h is related with the measured intensity I , the PGA a , the PGV v , and the

Algorithm 2.2.1 Algorithm for f^{-1}

Require: $H \in [0, 360]$
Require: $S \in [0, 100]$
Require: $V \in [0, 100]$
Ensure: $h \in [0, 1]$

```

 $V \leftarrow V/100$   $\triangleright$  Normalise  $V$ 
if  $V$  is 0 then  $\triangleright$  Use  $V$ 
|   return  $(V - 3.50)/(-2.9)$ 
else  $\triangleright$  Use  $H$ 
|    $\triangleright$  Deal with out-of-range  $H$ s, and use inverse linear functions
|   if  $H \geq 222$  then
|   |   return  $(H - 237)/(-150)$ 
|   else if  $H \leq 51$  then
|   |   return  $(H - 153)/(-170)$ 
|   |    $\triangleright$  Inverse-Cubic by Binary Search, with set errors
|   else
|   |    $l \leftarrow 0.1$ 
|   |    $r \leftarrow 0.6$ 
|   |    $\triangleright$  Checks if the normalised height is already precise enough
|   |   while  $r - l \geq \epsilon(h)$  do
|   |   |    $m \leftarrow (l + r)/2$ 
|   |   |    $c \leftarrow \text{HEIGHT TO HUE}(m)$ 
|   |   |    $\triangleright$  Checks if the calculated hue is already precise enough
|   |   |   if  $|c - H| \leq \epsilon(H)$  then
|   |   |   |   return  $m$ 
|   |   |   else if  $c > H$  then
|   |   |   |    $l \leftarrow m$ 
|   |   |   else if  $c < H$  then
|   |   |   |    $r \leftarrow m$ 
|   |   |   end if
|   |   end while
|   |   return  $(l + r)/2$ 
|   end if
end if

```

PGD x . They can be transformed forwards and backwards using simple mathematical explicit relations, and specifically for $f_H^{-1}(H)$ will use a binary search algorithm.

Figure 2.2.11 shows the data flow, Figure 2.2.12 shows the relation between abstract variables, and Figure 2.2.13 shows the result of colour generated compared with the original.

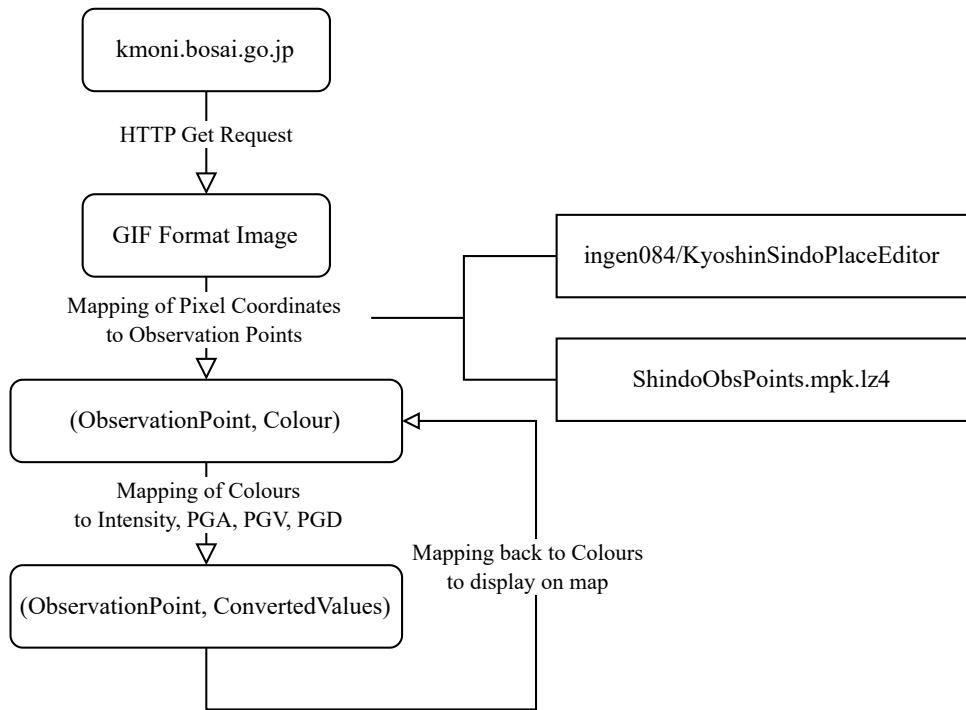


Figure 2.2.11: Flow of data in NIED data sources

It is worth noting that an existing NuGet Library, `ingen084/KyoshinMonitorLib` and introduced in [31] which is designated to manage intensities, as well as extracting intensities from the 'Kyoshin' monitor. This NEA did refer to this for some guidance but is not dependent on this library, and its necessary functionalities within the scope of this NEA is realised again using the author's own code. The developer of this library did also mention that it is quite purpose-built so might not be suitable for general use.

It is also worth noting that, technically, scraping the data from the 'Kyoshin' monitor page of NIED is not explicitly allowed, but not explicitly banned either. However, extracting and displaying numerical data in the application is strictly banned by the NIED, and therefore the numerical values will only serve as internal values of the application and will not be displayed to the users in any way.

This paragraph referred to [58]. The code used for this section is in Listing A.1.1.

2.2.2 DM-D.S.S. Data Source

DM-D.S.S. is a well-structured official data source with low latency and reliable information and services. This is going to be the primary data source for most part of the application.

Their APIs are split into two types: HTTP based requests and WebSocket based connections. HTTP based requests are typically for more static information, while WebSocket connections are for live time-essential data feeds, such as the EEW warnings, tsunami warnings and latest earthquake information.

2.2.2.1 Authorisation

There are two types of authorisation that DM-D.S.S. supports, API Keys and OAuth2 Access Tokens.

API Keys access tokens are extremely easy to program, since it simply uses Basic Authorisation in the header, and uses the key as the username (without a password). It uses the Basic Base64 Encoding. However, this introduces an extra layer of complexity for the users, since they would have to go to the settings of the DM-D.S.S. webpage and achieve an API Key to paste into the application, as shown in Figure 2.2.14.

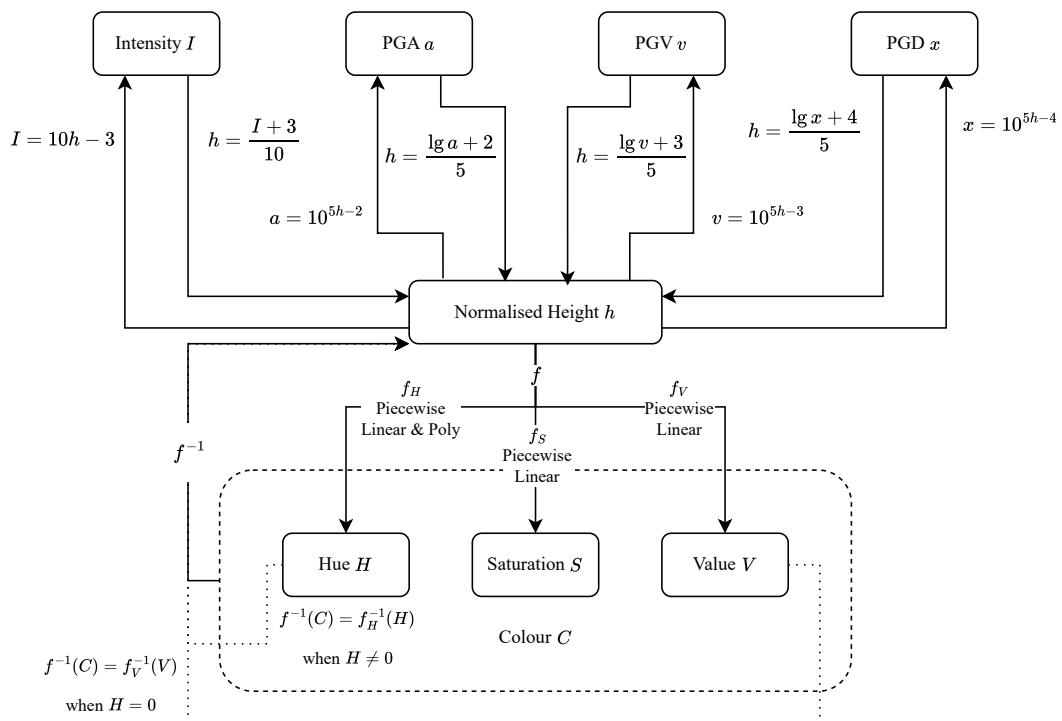


Figure 2.2.12: Relation between abstract variables

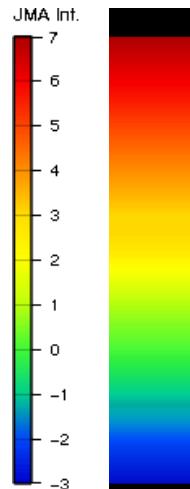


Figure 2.2.13: Colour generated using fitted functions



Figure 2.2.14: Control panel for API Keys

As for OAuth2, it will be much simpler for the users, since it will provide the user with a login interface on the website, and ask them to give the program certain permissions, which is just a few simple clicks. Rather than using the user as a bridge for sharing the credentials with the application, they are shared between the authorisation server (DM-D.S.S.) and the application directly, without the need for the user to deal with such human-unreadable codes.

OAuth 2.0 OAuth 2.0 is a standard protocol that allows a user to authorise an application without sharing any of their credentials. Figure 2.2.15 gives a brief outline of the procedure involved.

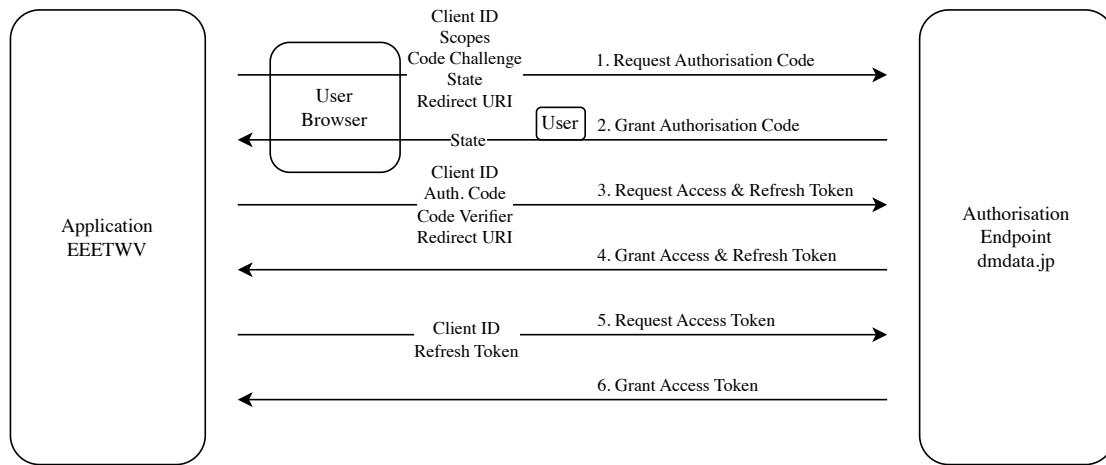


Figure 2.2.15: OAuth Procedure Outline

Authorisation Code The authorisation begins with the application requesting an **authorisation token** from the authorisation endpoint, via the user's browser. This is the only step where the user has to be involved, but all they have to do is log-in to their DM-D.S.S. account and click the 'grant' button on the page.

First, the user launches the browser to the authorisation endpoint (Step 1)

<https://manager.dmda.jp/account/oauth2/v1/auth>

with the following parameters:

- **client_id**: the client ID for the application to grant (which is public and safe to be seen by the user);
- **redirect_uri**: redirect URL for the endpoint to redirect to, to inform the application of the authorisation results;
- **response_type**: `code`
- **response_mode**: the way to response to the application (we will only use the response type `query`);
- **scope**: the string containing the permissions the application is requesting, separated by a space;
- **state**: the state which will be returned by the return query to prevent CSRF attacks;
- **code_challenge**: the code challenge for PKCE;
- **code_challenge_method**: the way to encode the code challenge (we will use `S256`).

At the same time as launching the browser, the program should start a listener on the local IP address and port specified in the redirect URL, and wait until it receives a request from the browser. In the response (Step 2), it will contain two parameters: `code` which contains the authorisation code (which is only valid for 10 minutes), and `state` which should be the same as the state sent in the request.

The reason why the `state` is essential is to prevent CSRF (Cross-Site Request Forgery) attacks, where the application can make sure the response is truly corresponding to the request the application

Error Code	Description
invalid_request	The parameters are invalid.
invalid_client	The Client ID is invalid.
invalid_redirect_url	The redirect URL is not supported by the Client ID.
invalid_scope	The scope string is invalid.
unauthorized_client	The client to authorise using this method is invalid.
access_denied	The access is denied.
recaptcha_verification_failed	The reCAPTCHA verification failed.
unsupported_response_type	The response type is not recognised.
unsupported_code_challenge_method	The code challenge method is not recognised.
no_signin	The user is not signed in (usually should not happen).
server_error	The internal server encountered an error.

Table 2.2.6: Error codes for authorisation code step in OAuth2

made (not by any other applications), and no one forged a response to the application (pretended to be the browser acting as authorisation endpoint). Verifying the state is essential for the security of the application.

If the request fails, the return query will return the `error`, and with the `state` as well. Table 2.2.6 includes a table of errors that could occur in the authorisation code request.

In the application, only the errors `access_denied`, `recaptcha_verification_failed`, `no_signin` and `server_error` should appear (and the final two shouldn't appear usually), since all other errors are due to issue with design of the application.

Refresh and Access Token After this step, the program will use the authorisation token to request an **access token** which comes with a **request token** (Step 3). The program will send an HTTP POST request to the token endpoint

<https://manager.dmdata.jp/account/oauth2/v1/token>

with the following query parameters, in URL Encoded form:

- `client_id`: the client ID for the application to grant (which is public and safe to be seen by the user);
- `grant_type`: `authorization_code`;
- `code`: the authorisation code received from the previous step;
- `redirect_uri`: the redirect URL used in the previous step;
- `code_verifier`: the original version of the code challenge used in the previous step.

Now the reason of the code challenge and code verifier is apparent (they are part of PKCE, Proof Key for Code Exchange): since the code challenge is a hashed version of the code verifier and is exchanged in the first step, the code verifier (the plain version) is used to prove that the application is the same as the one that requested the authorisation code. This means, if a man-in-the-middle acquired the authorisation code and the code challenge, they would not be able to use it, since it requires them to reverse hash the code challenge to get the code verifier, which is impossible in a reasonable amount of time. To ensure the security, the code verifier should be generated randomly and be very long, and disposed after each use.

The response (Step 4) will be JSON format, as shown in Listing 2.2.1.

The properties `token_type`, and `expires_in` are constants, the `scope` are the scopes which are authorised for, and the most important two are the `access_token` and `refresh_token`.

The refresh token is a long-living token, and it has lifetime 183 days in DM-D.S.S., resetting every time a new access token is acquired (see below). On the other hand, the access token is a short-living token, and it has lifetime 6 hours only. This does not mean that the access token should be disposed per-use – it should be reused within the application, and only when it expires, a new access token should be requested using the refresh token. However, when the application is closed, only the refresh token should be stored, and the access token revoked – next time the application launches, the access token should be acquired again.

```

1  {
2      "access_token": "ATn.TTTTTTTTTTTTTTTTTTTTTTTTT",
3      "token_type": "Bearer",
4      "expires_in": 21600,
5      "refresh_token": "ARh.RRRRRRRRRRRRRRRRRRRRRRRRRRR",
6      "scope": "telegram.list telegram.get.earthquake telegram.data"
7  }

```

Listing 2.2.1: Response for OAuth Access Token Request

```

1  {
2      "error": "invalid_request",
3      "error_description": "The client_id is missing."
4  }

```

Listing 2.2.2: Error for OAuth Access Token Request

If the request failed, then an error will be returned, also in JSON format, as shown in Listing 2.2.2. Table 2.2.7 includes a table of errors that could occur in the authorisation code request.

The only error that should occur here is `server_error`.

Refreshing Access Token Finally, Step 5 and 6 will be repeated multiple times in the application, to request a new access token using the provided refresh token. Similarly, the request will be a post request, but only requiring the `client_id`, the `grant_type` (as `refresh_token`) and `refresh_token` in the URL Encoded form, to the same endpoint.

The response JSON will be exactly the same, except the refresh token is not included again. It includes the newly acquired access token. In case of an error, all errors in Table 2.2.7 apart from `invalid_redirect_url` and `invalid_code_verifier` could occur. In this case however, there could be a chance of `invalid_grant`, if the application is closed for a very long period of time and the refresh token expired. In this case, the user should be asked to re-authorise the application.

Revoking Tokens Tokens should be revoked whenever they are no longer used. The revoke is done by sending a POST request to the revoke endpoint, which is

<https://manager.dmdata.jp/account/oauth2/v1/revoke>,

with the following parameters:

- `client_id`, the client ID;
- `token`, the token to be revoked.

The API will not return anything if successful, and if the token is already invalid, it will still be successful.

An error will be similar to that in Table 2.2.7, but only `invalid_request`, `invalid_client` and `server_error` could occur. Nonetheless, only `server_error` should occur in the application.

Error Code	Description
<code>invalid_request</code>	The parameters are invalid.
<code>invalid_client</code>	The Client ID is invalid.
<code>invalid_redirect_url</code>	The redirect URL is not supported by the Client ID.
<code>invalid_grant</code>	The authorisation code is invalid.
<code>invalid_code_verifier</code>	The PKCE verification failed.
<code>unauthorized_client</code>	The client to authorise using this method is invalid.
<code>unsupported_grant_type</code>	The grant type is not recognised.
<code>server_error</code>	The internal server encountered an error.

Table 2.2.7: Error codes for access token step in OAuth2

Flowchart In terms of the design of classes to support OAuth 2.0, there should be two separate classes responsible for this: one for acquiring the authorisation code and the refresh token, and the other for acquiring the access token using the refresh token.

Figure 2.2.16 shows the flowchart for the former, and Figure 2.2.17 shows the flowchart for the latter.

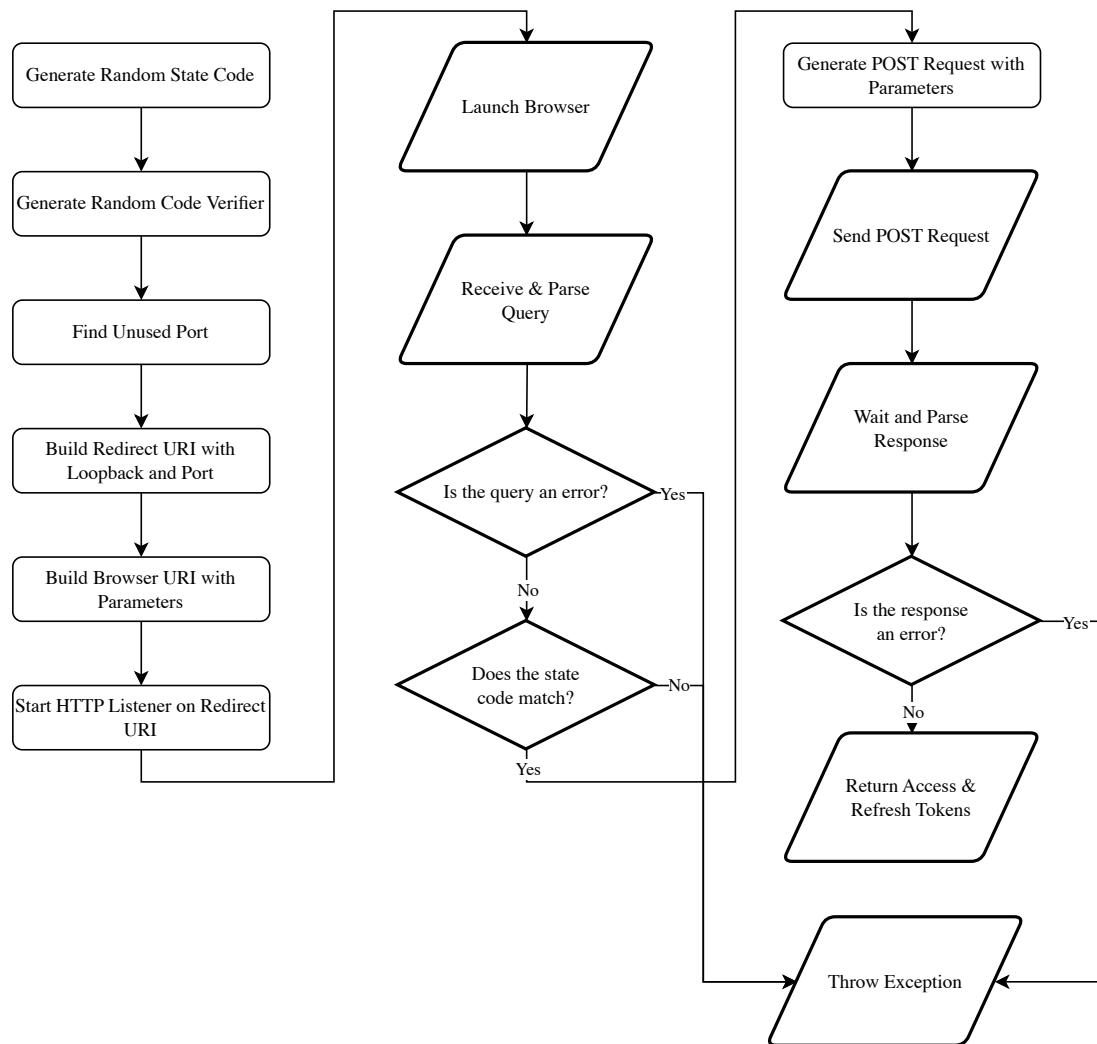


Figure 2.2.16: Flowchart for OAuth 2.0 Authorisation Code and New Refresh Token

Figure 2.2.17 represents the behaviour of the OAuth2 main class for authentication, while the previous figure Figure 2.2.16 represents the behaviour of the helper.

Notes For the purposes of this NEA, we will primarily use the API Key to test the application for accessing way of accessing DM-D.S.S. since it will be easier to code and debug, and OAuth2 will introduce quite a lot of complexity to the program. However, the program should be designed to be able to modify to OAuth2 authentication without much modification, and if time permits OAuth2 will be implemented in the application.

2.2.2.2 HTTP Based API Requests

The base URL of all requests is <https://api.dmdata.jp/v2/> which will be indicated as `base://` from now on.

Table 2.2.8 shows the necessary permissions would be necessary for the application to function. How each of them functions will be discussed below.

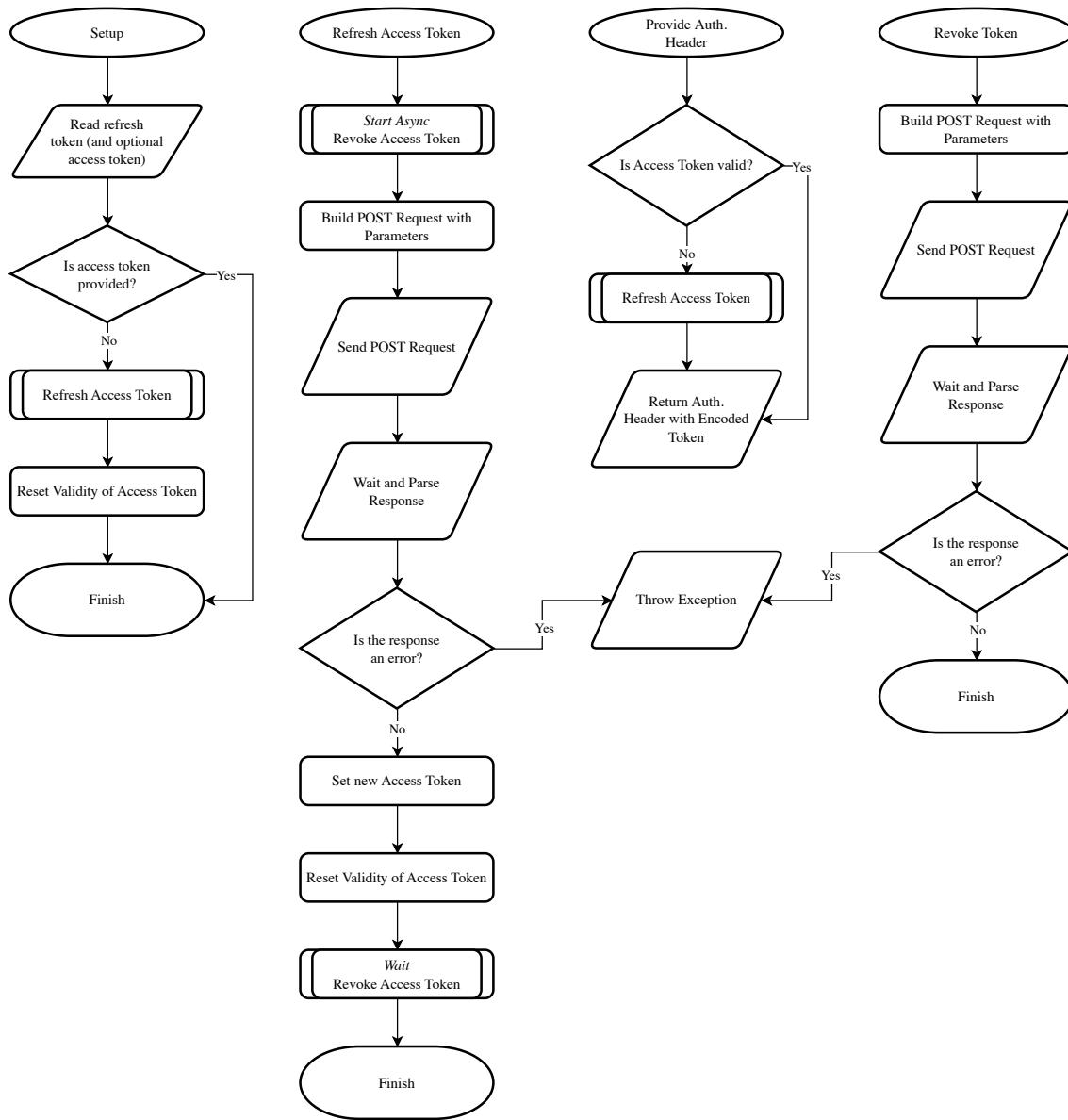


Figure 2.2.17: Flowchart for OAuth 2.0 Acquire Access Token using Refresh Token

Permission Code	Permission Details
<code>contract.list</code>	Get the list of subscriptions the user has.
<code>gd.earthquake</code>	Get list of past earthquakes.
<code>parameter.earthquake</code>	Get details of observation points for earthquake intensities.
<code>socket.start</code>	Start a new WebSocket connection.
<code>socket.list</code>	Get list of existing WebSocket connections.
<code>socket.close</code>	Close an existing WebSocket connection.
<code>telegram.data</code>	Get specific telegram released by the JMA.
<code>telegram.get.earthquake</code>	Allows program to access telegrams on earthquake information.
<code>eew.get.forecast</code>	Allows program to access telegrams on EEW forecasts (including warnings).

Table 2.2.8: Necessary access permissions for DM-D.S.S

Standard Return Information and Errors There are two status that an API call could return: a successful `ok` status, or an unsuccessful `error` status.

Listing 2.2.3 shows the JSON for a successful `ok` response, and Listing 2.2.4 shows the JSON for a `error` response.

```

1  {
2      "responseId": "66d23c0cede77d82",
3      "responseTime": "2021-04-01T00:00:00.000Z",
4      "status": "ok"
5 }
```

Listing 2.2.3: `ok` status for API

```

1  {
2      "responseId": "66d23c0cede77d82",
3      "responseTime": "2021-04-01T00:00:00.000Z",
4      "status": "error",
5      "error": {
6          "message": "Authentication required.",
7          "code": 401
8      }
9 }
```

Listing 2.2.4: `error` status for API

In both responses, there is an attribute `responseId` which gives the unique ID for each response as a `string`, and an attribute `responseTime` which gives the date and time of response in `ISO8601Time` format.

The `status` attribute in `string` indicates whether it is a successful response (in which case it will be "`ok`") and if it is an error response (in which case it will be "`error`" and gives the relevant HTTP error as well). The error will be indicated in the object `error` and will give the relevant message and code.

The list of standard errors together with their meanings is discussed in Table 2.2.9.

Error Code	Error Message
400	The query parameters are required.
400	The post parameters are required.
400	Unexpected data of search query <code>cursorToken</code> .
401	Authorisation required.
403	Insufficient scope for
403	Requests are not allowed.

Table 2.2.9: Standard errors for API

The only error that might occur in the application is 401 authorisation required, in the case when the OAuth 2 token expired, or the user inputted an incorrect API Key.

From here onwards, the `responseId`, `responseTime` and `status` will be removed from the sample response by default.

Cursor Token For some API data calls, and specifically for `telegram.list` that we are going to use, there is too much data to be returned in one API call. Therefore, in a response, there will be a specified attribute named `nextToken` indicated with type `string`, sometimes as well as `nextPooling` and `nextPoolingInterval`, as shown in Listing 2.2.5 as an example call of

```
base://telegram?type=VXSE53.
```

Specifically, the calls for `socket.list` and `gd.eew` will only return a `nextToken` for the next call, while for `telegram.list` and `gd.earthquake` it will return a `nextPooling` as well.

```

1  {
2      "responseId": "8359288359fc5bb9",
3      "responseTime": "2021-04-01T00:00:00.000Z",
4      "status": "ok",
5      "items": [
6          {},
7          {},
8          {}
9      ],
10     "nextToken": "bmV4dCAgICAgICAgNTc0MzI",
11     "nextPooling": "cG9sbGluZyAgICAgNzM0MzE",
12     "nextPoolingInterval": 1600
13 }

```

Listing 2.2.5: Cursor token sample JSON

If one would like to do the next call to find the next few items of the list, the `cursorToken` parameter should be specified as the same as the `nextToken` or `nextPooling`, with all the rest of the parameters identical to the previous request, i.e.,

```
base://telegram?type=VXSE53&cursorToken=bmV4dCAgICAgICAgNTc0MzI
```

if the `nextToken` is used, and

```
base://telegram?type=VXSE53&cursorToken=cG9sbGluZyAgICAgNzM0MzE
```

if `nextPooling` is used (which should be after `nextPoolingInterval` in milliseconds).

The document suggested that, due to performance issues, the `nextPooling` should be used wherever possible. Note that the `nextToken` and `nextPooling` will change for every single call of the API.

From here onwards, in the API sample results, `nextToken`, `nextPooling` and `nextPoolingInterval` will not be included.

Contract API There is only one contract-related API, which is `contract.list`. It is an HTTP GET request, on `base://contract` with no parameters.

It will return a list of contracts (subscriptions) of DM-D.S.S., whether or not the user has subscribed to it.

Listing 2.2.6 shows a sample return of this API call.

```

1  {
2      "items": [
3          {
4              "id": 92,
5              "planId": 1,
6              "planName": "地震・津波関連",
7              "classification": "telegram.earthquake",
8              "price": {
9                  "day": 15,
10                 "month": 350
11             },
12             "start": "2021-01-01T01:01:00.000Z",
13             "isValid": true,
14             "connectionCounts": 1
15         }
16     ]
17 }

```

Listing 2.2.6: Contract list sample JSON

Here, `items` is the list of contracts (subscription plans), with each of its element being an object representing a plan. For each of them, there are the properties:

- `id` (nullable) which stands for the subscription ID (could be `null` if the user is not subscribed);
- `planId` which stands for the subscription plan ID, which is unique for each plan;
- `planName` which stands for the name of the subscription plan;
- `classifications` which stands for the classification code of the subscription plan;
- `price.day` and `price.month` which stands for the price of the plan per day/per month;
- `start` (nullable) which stands for the start of the subscription;
- `isValid` which is a `boolean` to represent whether the plan is valid; and
- `connectionCounts` which stands for the number of extra WebSocket connections this subscription plan provides.

Note that how `id` and `start` can be `null` since it lists all the subscriptions, whether or not the user has been subscribed to it.

This API will be used to calculate the number of WebSocket connections, to display empty WebSocket slots in the listing WebSocket functionality.

WebSocket APIs There are three WebSocket-Related APIs, specifically, `socket.start`, to start a socket, `socket.list` to list all sockets, and `socket.close`, to close a socket. They are all called on `base://socket`.

Socket Start `socket.start` is a POST method which has a request body in JSON, as shown in Listing 2.2.7.

```

1  {
2      "classifications": [
3          "eew.forecast",
4          "telegram.earthquake"
5      ],
6      "types": [
7          "VXSE45",
8          "VXSE51",
9          "VXSE52",
10         "VXSE53"
11     ],
12     "test": "including",
13     "appName": "Application Test",
14     "formatMode": "json"
15 }
```

Listing 2.2.7: Socket start sample request JSON

The attributes are:

- `classifications`, which is a list of classifications of the subscription plans (which include all the types below);
- `types` (optional), which is a list of telegram wished to be transmitted through the WebSocket;
- `test` (optional, default to "`no`"), which indicates whether test telegrams should be received ("`including`") or not ("`no`");
- `appName` (optional), which is the name of the application to be recorded with the WebSocket; and

- **formatMode** (optional, default to "raw"), which is either "json" or "raw" depending on the desired return type.

The detailed classifications and types will be discussed in the next section on WebSocket connections. When the **types** is excluded, the WebSocket will return all the telegrams included in the classification. It will have a response as in Listing 2.2.8.

```

1  {
2      "ticket": "Tik....",
3      "websocket": {
4          "id": 0,
5          "url": "wss://ws003.api.dmdata.jp/v2/websocket?ticket=Tik....",
6          "protocol": [
7              "dmdata.v2"
8          ],
9          "expiration": 300
10     },
11     "classifications": [
12         "eew.forecast",
13         "telegram.earthquake"
14     ],
15     "test": "including",
16     "types": [
17         "VXSE45",
18         "VXSE51",
19         "VXSE52",
20         "VXSE53"
21     ],
22     "formats": [
23         "xml",
24         "a/n",
25         "binary"
26     ],
27     "appName": "Application Test"
28 }
```

Listing 2.2.8: Socket start sample response JSON

Some attributes are exactly the same as the JSON sent has header, **classifications**, **types** (nullable), **test** and **appName**. The additional attributes are:

- **ticket**, which is a code in **string** for the connection. This is also included in the **websocket** object.
- **websocket**, which is the object containing details for the connections. Its attribute **id** is unique for each connection and **url** is the WebSocket (**wss://**) URL to use to connect to. **protocol** being a constant array containing "**dmdata.v2**" indicating that this is the 2nd version of the protocol. **expiration** is in seconds the time the connection will expire if no transmission handshakes are made (see below), and is always 300 as constant.
- **formats**, which is a list of formats used to encode the data transmitted in the WebSocket. This is a constant list including "**xml**", "**a/n**" and "**binary**" if **formatMode** is set to "**raw**", and a list with only "**json**" if set to "**json**" in the request JSON.

Apart from the standard errors, it will also output errors shown in Table 2.2.10

The only error that should occur in the application is the final one, where the maximum number of simultaneous connections is exceeded. To allow the user to deal with this situation, a disconnect button for each connection in the current WebSocket list is provided.

Error Code	Error Message
400	The body of the request is not JSON.
400	At least one element of <code>classifications</code> is required.
400	The <code>types</code> is not a string or has more than 30 elements.
400	The <code>appName</code> is up to 24 bytes.
400	You have entered a string that is not defined in <code>formatMode</code> .
402	No contract.
409	The maximum number of simultaneous connections is exceeded.

Table 2.2.10: Errors for `socket.start`

Socket Close `socket.close` is a DELETE method, which specifies the ID of the WebSocket port to be closed in the URL parameter `:id`, e.g., `base://socket/30`. If the result is successful, the request will not return anything. If the status is error, it will return a standard error (detailed above), or a 404 error, which indicates that the specified WebSocket `id` is not found.

There is a chance of a 404 error for WebSocket ID not found, if the list has not been refreshed, and the user attempts to disconnect to the WebSocket which was already disconnected. Due to this, the refresh WebSocket list button is provided should such error occur.

Socket List `socket.list` is a GET method which supports `cursorToken` (optional) as discussed above. There are other parameters in the query:

- `id` (optional), which stands for the ID of the WebSocket connection that details is wished to be retrieved;
- `status` (optional), a `string` indicating the status of the WebSocket, including `open` which stands for operating live, `waiting` which means idle, and `closed` standing for closed connections;
- `limit` (optional, default 20), of type `integer` with a maximum of 100, indicating the number of WebSockets listed in a single response.

Listing 2.2.9 shows a response from the call. The `items` is a list of objects, each representing a WebSocket, with the following properties:

- `id`, `ticket`, `classifications`, `test`, `types`, `formats`, `appName`, identical to described above;
- `start`, a `ISO0601Time` representing the time of the start of the connection;
- `end`, a nullable `ISO0601Time` representing the time of the end of the connection;
- `ping`, a nullable `ISO0601Time` representing the previous time of ping-pong;
- `ipAddress`, a nullable `string` representing the IP address of the source of connection;
- `server`, a nullable `string` representing the connected WebSocket server; and
- `status`, a `string` representing the connection status.

However, it is worth noting that even though the document claims that `ticket` is not `null`, it will only be not null when the WebSocket is waiting connection, and null otherwise. Furthermore, `formats` never appears in the result, even though the document claims that it should always appear.

These API calls will be used to manage (open new, list existing and close) WebSockets in the application.

Parameter API The program would need to have a list of observation points of earthquakes to plot on the map the observed stations' intensity on the past-page map. Therefore, we would need to call `parameter.earthquake`, which is available at `base://parameter/earthquake/station`.

This is a GET request with no parameters to specify, and will return all stations in one go (which is time-costly). Listing 2.2.10 shows a sample response.

The `changeTime` and `version` of this list of parameters is often the same for calls since the list is rarely updated, and they give the version of the list and the last time of the update.

For each object in `items`, they represent an earthquake observation point. The attributes are:

```

1  {
2      "items": [
3          {
4              "id": 0,
5              "ticket": "Tik....",
6              "types": [
7                  "VXSE45",
8                  "VXSE51",
9                  "VXSE52",
10                 "VXSE53"
11             ],
12             "test": "including",
13             "classifications": [
14                 "eew.forecast",
15                 "telegram.earthquake"
16             ],
17             "ipAddress": "192.168.0.0",
18             "status": "open",
19             "server": "websocket-03",
20             "start": "2021-04-01T00:00:00.000Z",
21             "end": null,
22             "ping": "2021-04-01T00:00:00.000Z",
23             "appName": "ApplicationTest"
24         }
25     ]
26 }
```

Listing 2.2.9: Socket list sample response JSON

```

1  {
2      "changeTime": "2024-07-18T12:00:00+09:00",
3      "version": "20240718",
4      "items": [
5          {
6              "region": {
7                  "code": "100",
8                  "name": "石狩地方北部",
9                  "kana": "イシカリチホウホクブ"
10             },
11             "city": {
12                 "code": "0123500",
13                 "name": "石狩市",
14                 "kana": "イシカリシ"
15             },
16             "noCode": "1000000",
17             "code": "0123500",
18             "name": "石狩市花川",
19             "kana": "イシカリシハナカワ",
20             "status": "現",
21             "owner": "気象庁",
22             "latitude": "43.1714",
23             "longitude": "141.3156"
24         }
25     ]
26 }
```

Listing 2.2.10: Earthquake parameter sample response JSON

- `region` and `city`, both containing `code`, `name` and `kana` (Kana (カナ, 仮名) represents the pronunciation), which is unique for each region and city;
- `noCode`, the unique code, and `code`, the code used in XML (WebSocket data feeds);
- the `name` and `kana`, which are not necessarily unique for each observation point;
- `status` standing for whether the station is in use. "現" (now) means the data hasn't changed in the update, "変更" (change) means the data has been altered, "新規" (new) stands for a new observation point, and "廃止" (abolished) means the observation point has been discontinued;
- the `owner`, representing the owner of the observation point; and
- the `latitude` and `longitude`, which as the name suggests, represent the position of the observation point.

Due to the nature of JSON being very long and barely updated (an email will be sent to users of DM-D.S.S. every time its updated in fact), it will be obtained upon application launch/authentication by the user, and stored in the main memory for the lifetime of the application, rather than obtaining such every time when necessary.

Past Earthquake API The API `gd.earthquake` allows us to use a GET request to obtain a series of earthquake on `base://gd/earthquake`, with the option to obtain a list, or to obtain only a certain event.

Past Earthquake List The API call `gd.earthquake.list`, which supports `cursorToken` and `poolingToken` allows us to obtain a list of past earthquakes, with the ability to specify the following parameters:

- `hypocenter`, the 3-digit code to filter the hypocentre position;
- `maxInt`, the lower bound of the filter for the maximum measured intensity of the earthquake;
- `date`, the date when the earthquake is observed;
- `limit`, the number of earthquakes returned in one go, with a maximum of 100.

A sample response is shown in Listing 2.2.11.

These APIs will be used to display a list of past earthquakes. For each object representing an earthquake, this includes:

- Its unique `id`, `eventId`, `originTime` (if it is not a report on the intensity), and `arrivalTime`;
- its `type` (which is `normal` for earthquakes in Japan, and `distant` for earthquake reported by other countries);
- its `hypocenter`, including the `code`, `name`, the `coordinate` and the `depth`;
- its `magnitude` (an object representing units and values); and
- its `maxInt`, the maximum intensity observed.

It might also include LPGM information, in `maxLgint` and `lgCategory`, if such information is released, usually only for huge earthquakes.

The data used will be used to create the sidebar (list of past earthquakes) of the past page in the application, and will be called upon the click of refresh button/load more button.

```
1  {
2      "items": [
3          {
4              "id": 1584,
5              "type": "normal",
6              "eventId": "20210808085414",
7              "originTime": "2021-08-08T08:54:00+09:00",
8              "arrivalTime": "2021-08-08T08:54:00+09:00",
9              "hypocenter": {
10                  "code": "787",
11                  "name": "鹿児島湾",
12                  "coordinate": {
13                      "latitude": {
14                          "text": "31.3°N",
15                          "value": "31.3000"
16                      },
17                      "longitude": {
18                          "text": "130.6°E",
19                          "value": "130.6000"
20                      },
21                      "height": {
22                          "type": "高さ",
23                          "unit": "m",
24                          "value": "0"
25                      },
26                      "geodeticSystem": "日本測地系"
27                  },
28                  "depth": {
29                      "type": "深さ",
30                      "unit": "km",
31                      "value": "0",
32                      "condition": "ごく浅い"
33                  }
34              },
35              "magnitude": {
36                  "type": "マグニチュード",
37                  "unit": "Mj",
38                  "value": "2.6"
39              },
40              "maxInt": "2"
41          }
42      ]
43  }
```

Listing 2.2.11: Past earthquake list sample response JSON

```

1  {
2      "event": {
3          "telegrams": [
4              {
5                  "serial": 0,
6                  "id": "...",
7                  "originalId": "...",
8                  "classification": "telegram.earthquake",
9                  "head": {
10                      "type": "VXSE53",
11                      "author": "RJTD",
12                      "time": "2021-08-07T23:58:00.000Z",
13                      "designation": null,
14                      "test": false
15                  },
16                  "receivedTime": "2021-08-07T23:58:10.311Z",
17                  "xmlReport": {
18                      "head": {
19                          "title": "震源・震度情報",
20                          "serial": "1",
21                          "eventId": "20210808085414",
22                          "headline": "8日08時54分ころ、地震がありました。",
23                          "infoKind": "地震情報",
24                          "infoType": "発表",
25                          "reportDateTime": "2021-08-08T08:58:00+09:00",
26                          "targetDateTime": "2021-08-08T08:58:00+09:00",
27                          "infoKindVersion": "1.0_1"
28                      },
29                      "control": {
30                          "title": "震源・震度に関する情報",
31                          "status": "通常",
32                          "dateTime": "2021-08-07T23:58:08Z",
33                          "editorialOffice": "気象庁本庁",
34                          "publishingOffice": "気象庁"
35                      }
36                  },
37                  "schema": {
38                      "type": "earthquake-information",
39                      "version": "1.0.0"
40                  },
41                  "format": "json",
42                  "url": "https://data.api.dmdata.jp/v1/..."
43              }
44          ]
45      }
46  }

```

Listing 2.2.12: Past earthquake event sample response JSON

Past Earthquake Event If the call `base://gd/earthquake/:eventId` is used, it will return the object representing the earthquake in the `event` attribute (most attributes removed since identical to Listing 2.2.11), as well as a list of telegrams in the `telegrams` attribute, which was detailed above in the `telegram.list` section, as shown in Listing 2.2.12.

In the `telegrams` list, each object represents a telegram, with certain properties:

- `serial` stands for the serial ID of this transmission, while `id` is a unique ID for the JSON telegram (non-serial);
- `originalId` gives the original ID of the telegram in XML format;
- `classification` gives the classification of this telegram. For the purpose of this application, all classification will be of type `telegram.earthquake`;
- `head` gives the head information of this telegram, including the `type`, the `author`, the `time`, and whether it is `test`;
- `receivedTime` gives the time of this telegram being released,
- `xmlReport` details parts of information encoded in the XML;
- `schema` details the JSON schema used for the telegram, including the `type` and `version` of the schema used;
- `url` gives the URL for the detailed information of this telegram (that can be retrieved using another GET request), and `format` gives its format.

All telegrams provided in this list are in JSON format.

This API will be used to load the details for a certain earthquake to display on the sidebar panel.

2.2.2.3 WebSocket Connections

WebSocket connections are essential to real-time applications, since it provides a real-time feed from the server to the client on latest data, without the need of the client to query (which consumes a lot of resources). DM-D.S.S. provides WebSockets to initiate real-time connection to the server to achieve the latest telegrams and earthquake warnings.

The URL of the WebSocket will be `wss://[#1].api.dmdata.jp/v2/websocket`, where the #1 part could be `ws001` to `ws004`, but it is optional. If not included, the server will simply distribute the load.

WebSocket Responses There are four types of responses that a WebSocket could return, indicated in the `type` attribute:

- a `start`. This only happens when the WebSocket is started, and this includes the exact same information in the `socket.start` call in Listing 2.2.8, with an additional `time` to indicate the time of the response;
- a (server-side initiated) `ping`. This will happen regularly, and the server will send a JSON in the format in Listing 2.2.13.

```

1  {
2      "type": "ping",
3      "pingId": "012345"
4 }
```

Listing 2.2.13: WebSocket Ping JSON

Upon receiving, the client should return a `pong` JSON in the same format, with the same `pingId`, as in Listing 2.2.14.

This is in place to ensure that there are no 'dead' WebSocket connections, and the server will close the WebSocket connection after 120s of no response in such ping-pong calls, to not take up the limited number of WebSocket connection slots;

```

1  {
2      "type": "pong",
3      "pingId": "012345"
4 }
```

Listing 2.2.14: WebSocket Pong JSON

- a client-side initiated `ping` as in Listing 2.2.13, which the server will return a `pong` as in Listing 2.2.14. This works almost identically as before, apart from that such `pingId` is optional. This is in place to ensure that the application would be able to know if the connection is broken, for example due to unstable internet connections, and would be able to restart another WebSocket connection;
- a `error`, which will be the format in Listing 2.2.15. There will also be a boolean, `close`, to indicate whether the WebSocket connection is closed or not.

```

1  {
2      "type": "error",
3      "error": "Server error.",
4      "code": 4503,
5      "close": true
6 }
```

Listing 2.2.15: WebSocket Error JSON

A table of possible WebSocket errors is included in Table 2.2.11, with the relevant WebSocket error codes.

Error Code	Error Details
4400	Compulsory parameters are not specified when starting WebSocket.
4404	Ticket is not specified when starting WebSocket.
4409	Number of connection limit is reached.
4503	Internal server error.
4640	<code>pingId</code> does not match for a server-initiated <code>ping</code> .
4641	Incorrect data (non-JSON) received from client.
4808	Client requested closure of WebSocket.
4807	Contracts are discontinued in connection.

Table 2.2.11: Errors for WebSocket connection

In the application, only 4409 (when the connection number exceeded the maximum), 4503 (when the server has an internal error), 4808 (when the client requested closure of WebSocket), and 4807 (contract discontinued) should occur in the application.

The program should be designed to be able to handle these errors, and to close the connection if necessary;

- a `data`, an example shown in Listing 2.2.16.

This will include a `classification`, a version `version`, and a unique `id`. A list named `passing` gives the `name` and `time` of each passing location of this data piece inside the server.

Furthermore, the `head` and `xmlReport` gives details of the telegram received, similar to in Listing 2.2.12.

The `format`, `compression` (`gzip`, `zip` or `null`) and `encoding` (`base64` or `utf-8`) gives information on how the `body` should be processed and interpreted.

Note that the `body` should always be dealt with in the following sequence: decode; decompress; and deserialise.

This gives the key piece of information that will be passed between the data component of the software and the UI component of the software.

```
1  {
2      "type": "data",
3      "version": "2.0",
4      "classification": "telegram.weather",
5      "id": "123456789abcdef...",
6      "passing": [
7          {
8              "name": "websocket-01",
9              "time": "2020-01-01T00:00:00.120Z"
10         }
11     ],
12     "head": {
13         "type": "VPWW54",
14         "author": "JPTD",
15         "time": "2020-01-01T00:00:00.000Z",
16         "test": false,
17         "xml": true
18     },
19     "xmlReport": {
20         "control": {
21             "title": " 気象警報・注意報（H 2 7）",
22             "dateTime": "2020-02-27T00:00:00Z",
23             "status": " 通常",
24             "editorialOffice": " 気象庁本庁",
25             "publishingOffice": " 気象庁予報部"
26         },
27         "head": {
28             "title": " 東京都気象警報・注意報",
29             "reportDateTime": "2020-02-27T09:00:00+09:00",
30             "targetDateTime": "2020-02-27T09:00:00+09:00",
31             "eventId": null,
32             "serial": null,
33             "infoType": " 発表",
34             "infoKind": " 気象警報・注意報",
35             "infoKindVersion": "1.2_1",
36             "headline": " 注意報を解除します。"
37         }
38     },
39     "format": "xml",
40     "compression": "gzip",
41     "encoding": "base64",
42     "body": "H4sIAAAAAAAA... "
43 }
```

Listing 2.2.16: WebSocket Data JSON

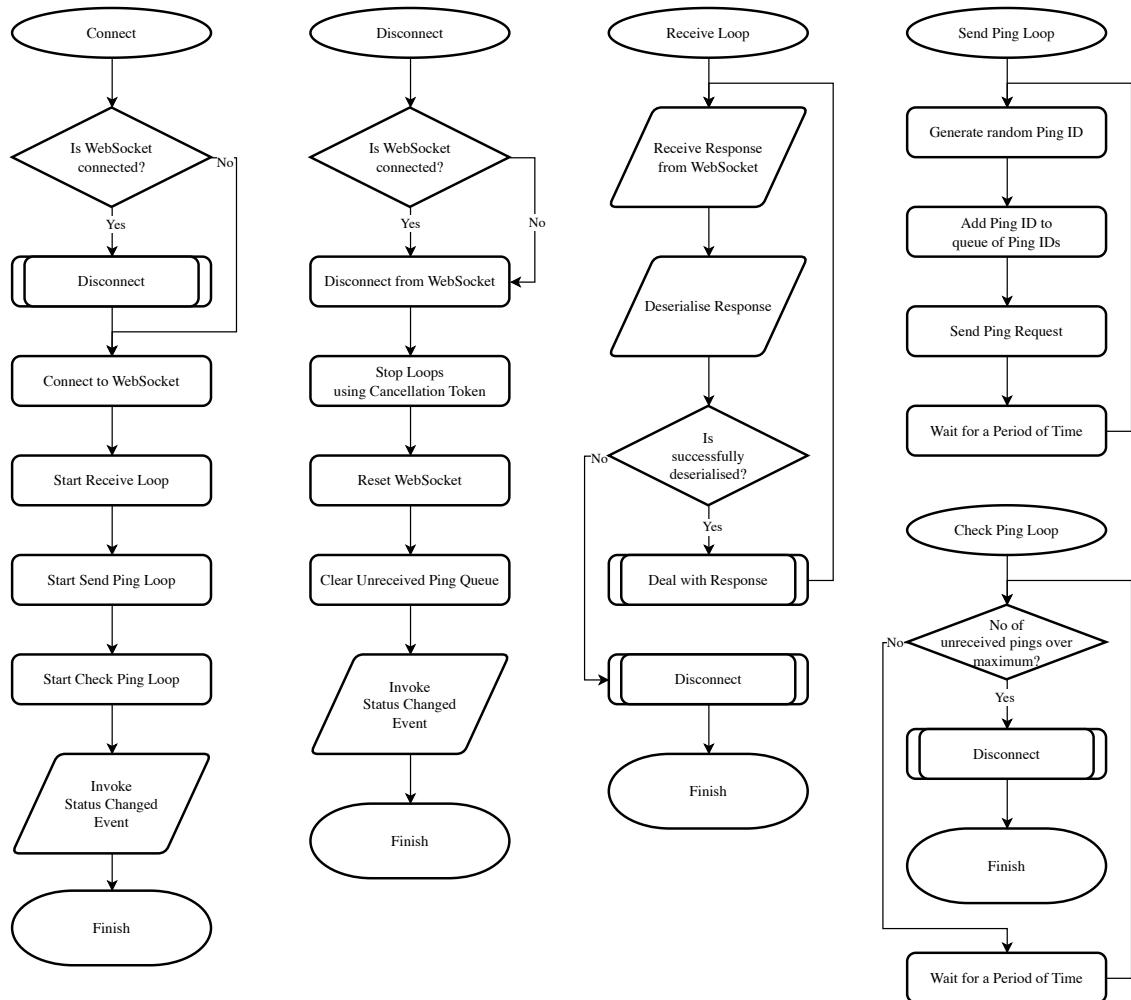


Figure 2.2.18: Flowchart of WebSocket Wrapper

JSON v.s. XML Most data nowadays on the internet are serialised in JSON, but the JMA provides its data in XML format. DM-D.S.S. does also provide JSON formatted schemas for most of the XML data files which are also processed real time. Even though there is a high chance of delay (up to 1 second) if the JSON format is used, not to say the potential of an error in processing which lead to undefined behaviour in the JSON format, the JSON format is much easier to parse and deserialise into C# DTO objects, and the schemas are very well-defined on the DM-D.S.S. references [16], in contrast to the JMA documentation [74] which is in Japanese (which the author is not an expert in). Therefore, JSON format telegrams and data will be used in the application.

Design of WebSocket Wrapper .NET has a built-in sealed `ClientWebSocket` class, and therefore to use it, we would have to wrap a class around it (i.e. a class that composites `ClientWebSocket` class), since it is responsible for the life-cycle of the WebSocket as well. The wrapper is re-used over multiple connections, while the backing `ClientWebSocket` class is disposed per-use.

Events are used here (Data Received and Status Changed) to use callbacks to send back the data to the necessary component, such as the View Model for the GUI (introduced later). Furthermore, the FIFO data structure queue is used to store pings sent by the client, since a later ping received would indicate that the connection is still alive, and the ping, together with all pings before that ping (and including that ping itself), could be removed from the queue.

The overall flow structure is shown in Figure 2.2.18 for the overall functionality of the WebSocket flowchart, and in Figure 2.2.19 for dealing with the received response.

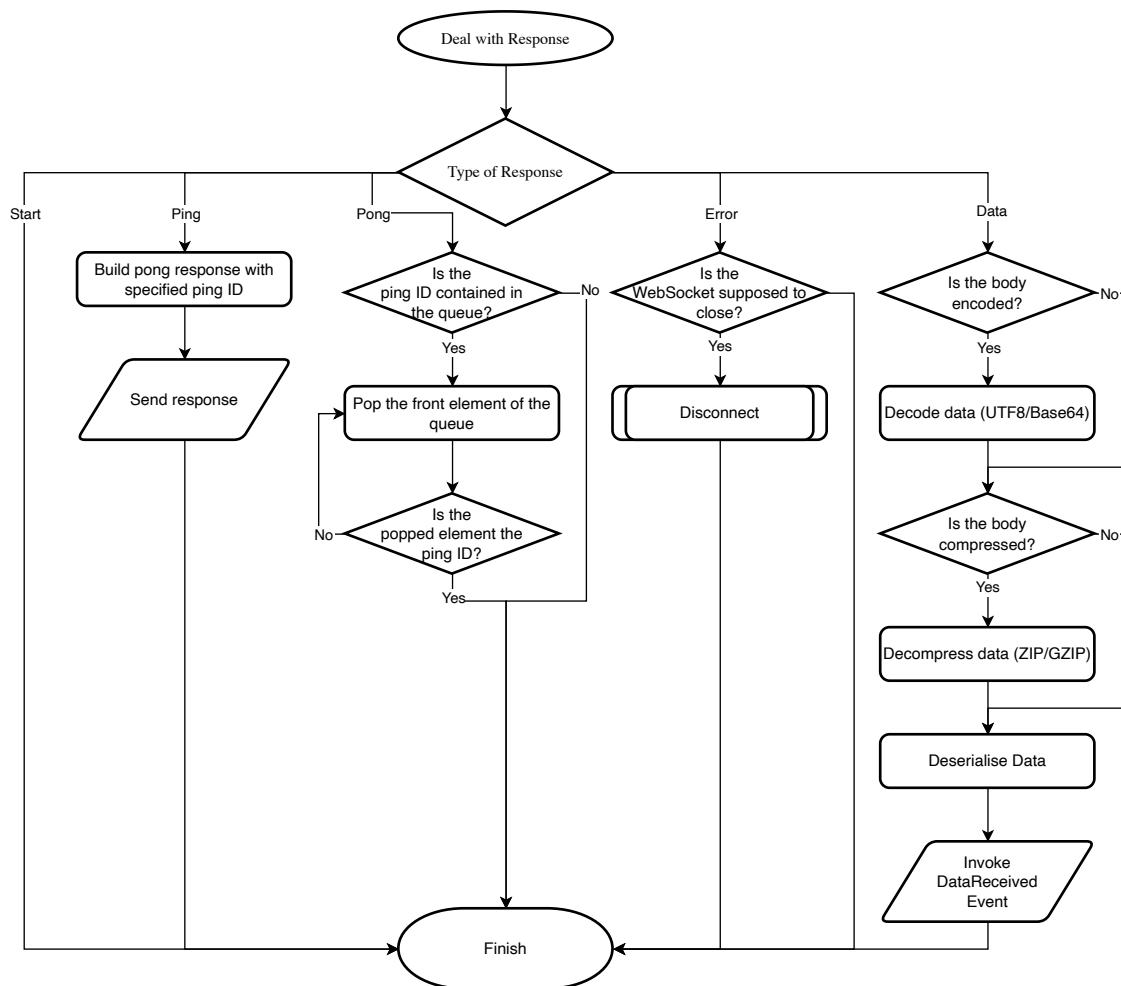


Figure 2.2.19: Flowchart of dealing with WebSocket Response

2.2.2.4 Telegrams

As discussed in the previous section, JSON telegrams is going to be used in the application, and there is no particular reason why telegrams in general (i.e. for the earthquake telegrams in `gd.earthquake.event` for the past-page) should be retrieved in XML format compared to JSON format. In this section we will discuss the telegrams used, and the structure of a telegram in DM-D.S.S.

Telegrams Used Considering the uses of the application, Table 2.2.12 gives an overview of the telegrams used in the application.

Code	Telegram Classification	JSON Schema	Description
VXSE45	<code>eew.forecast</code>	<code>eew-information</code>	All EEW telegrams including forecasts and warnings.
VXSE51	<code>telegram.earthquake</code>	<code>earthquake-information</code>	Preliminary report on the measured intensity of earthquake.
VXSE52	<code>telegram.earthquake</code>	<code>earthquake-information</code>	Preliminary report on the hypocentre of earthquake.
VXSE53	<code>telegram.earthquake</code>	<code>earthquake-information</code>	Full report on intensity and hypocentre of earthquake.
VTSE41	<code>telegram.earthquake</code>	<code>tsunami-information</code>	Tsunami warnings of all types.

Table 2.2.12: Telegrams used in the application

In the real-time page, VXSE45 and VTSE41 telegrams are fed via the WebSocket connection detailed above to be displayed. On the past page, VXSE51 53 telegrams are used to display the past earthquake information. In specific, the following logic will be used to determine which telegram to use: (in the following, $x = 1, 2, 3$)

- Filter the telegram to only include VXSE5 x ;
- Group telegrams by the telegram code;
- Find the latest telegram (by serial number) in each of the category;
- Use the telegram with the biggest x (i.e. VXSE53 > VXSE52 > VXSE51).

The particular choice for the final one is due to the fact that VXSE53 usually contains the most information, while VXSE51 and VXSE52 usually contains less information than VXSE53. (They are usually released earlier than VXSE53 though).

JSON Schemas All JSON Schemas include a part called the **schema head**, and a sample is shown in Listing 2.2.17.

There are the following properties:

- `_originalId`, the original ID of the telegram;
- `_schema`, the DM-D.S.S. JSON Schema used for this telegram, including the type and the version;
- `type`, the name of the telegram, and `title`, the title of the telegram;
- `status`, the status of the telegram, whether it is normal (通常), a training telegram (訓練), or a test telegram (試験). Note that telegrams should be filtered in the application to only display normal telegrams to the user;
- `infoType`, the type of information, whether it is a release (発表), a correction (訂正), a delay (遅延), or a cancellation (取消). Note that in EEW telegrams, there is also an indication in the body part whether it is a cancellation;
- `editorialOffice`, and a list of `publishingOffice`;
- `pressDateTime` when the telegram is made, `reportDateTime` when the telegram is released;
- `targetDateTime` which is nullable, describes the base time of the report, and `targetDatetimeDoubtful` giving the error;

```

1  {
2      "_originalId": "...",
3      "_schema": {
4          "type": "earthquake-information",
5          "version": "1.0.0"
6      },
7      "type": "震源・震度に関する情報",
8      "title": "震源・震度情報",
9      "status": "通常",
10     "infoType": "発表",
11     "editorialOffice": "気象庁本庁",
12     "publishingOffice": [
13         "気象庁"
14     ],
15     "pressDateTime": "2021-08-12T06:05:36Z",
16     "reportDateTime": "2021-08-12T15:05:00+09:00",
17     "targetDateTime": "2021-08-12T15:05:00+09:00",
18     "eventId": "20210812150301",
19     "serialNo": "1",
20     "infoKind": "地震情報",
21     "infoKindVersion": "1.0_1",
22     "headline": "12日15時02分ころ、地震がありました。",
23     "body": {}
24 }

```

Listing 2.2.17: Head of JSON Schema

- `validDateTime` for which the report is valid until (e.g. for tsunami information, there will not be cancellation reports, only a date time the information is valid until);
- `eventId`, the unique ID for the event that this describes. This should be used to identify earthquakes in the EEW;
- `serialNo`, the number that is released for the specific event. In EEW, if a telegram with lower serial is received after a telegram with a higher serial (e.g. due to delay reasons), then the telegram should not be used and should be disposed of;
- `infoKind` and `infoKindVersion`, the schema used in the XML report;
- `headline`, which includes the summary headline of the information;
- `body`, the body of all the information.

In particular, the information of `_schema`, `status`, `pressDateTime` (for displaying the last updated time of the information), `eventId`, `serialNo` and the `body` will be useful for the application.

The `body` part of the telegram will be omitted for brevity, since they are mostly just convoluted JSON objects. Anything interesting in particular will be mentioned later in the design for the DTOs.

Text and Comments component of the body For all schemas used in the application, in the `body` component, there will be properties `text` (of type string) and `comments` which includes text and comments on the telegram.

In the `comments` property, there could always be a `free` property where any string could be recorded, and some other properties (whose names may vary: `warning` for `tsunami-information` and `eew-inf` `ormation`, and `forecast` and `var` for `earthquake-information`), each containing a list of strings and codes which are paired, where the code is a 3-digit code representing the code of the comment.

An example of this is shown in Listing

These should be used, together with the `headline` property, to create the informational text to display with a telegram.

```

1  {
2      "body": {
3          "comments": {
4              "forecast": {
5                  "text": "津波警報等（大津波警報・津波警報あるいは津波注意報）を発表中です。",
6                  "→ \nこの地震について、緊急地震速報を発表しています。",
7                  "codes": [
8                      "0211",
9                      "0241"
10                 ]
11            },
12            "var": {
13                "text": " *印は気象庁以外の震度観測点についての情報です。",
14                "codes": [
15                    "0262"
16                ]
17            }
18        }
19    }

```

Listing 2.2.18: Comments in JSON Schema

2.2.2.5 Sidenotes

It is worth noting that an existing NuGet Library, [ingen084/DmdataSharp](#) supports dealing with the DM-D.S.S. data flow and converting them to C# objects (and exceptions). However, for the purposes of this NEA, we will implement our own way to interact with the APIs and the correlated C# DTOs. The developer of this library did also mention that it is quite purpose-built so might not be suitable for general use.

This section referred to [14, 16, 17, 19].

2.3 Algorithms

There are two key 'algorithms' that was designed for this application (in addition to the inverse-cubic for converting colours as described previously): the first one used to calculate the position of the P/S wavefronts, and the other one used to detect shake on the real-time monitoring screen.

2.3.1 Wavefront Calculation

Earthquake waves do not travel at uniform rate as they spread out due to the decrease in energy, and so there is no uniform velocity that the seismic waves travel in. We cannot do a simple equation like $s = vt$ to calculate the distance of the wavefront of a seismic wave from the epicentre.

However, JMA provides tables for the time that seismic waves take to travel a certain distance in [71], together with [75]. It will be possible for us to look up the table and plot the seismic waves at correct distance at a certain time.

Comparing the different wave travel tables that JMA provides, the most suitable one for this application will be the **JMA 2001**, since it is designed for general purpose uses for both land-based and ocean-based earthquakes, and does not take into account the effect of the altitude of the observation points. Listing 2.3.1 provides a preview of the first 10 lines of this file.

The table is a lookup table from horizontal distance to time in terms of the depth of the epicentre and the type of wave (P or S) to the time needed to travel such distance. Each row represents the P-Wave, time taken for the P-wave to travel (in seconds), S-wave, time taken for the S-wave to travel (in seconds), depth of epicentre (in kilometres), distance from epicentres (in kilometres).

This file will be read by the program upon launch every time and stored in the active memory when the program is running.

```

1 P    0.000 S    0.000 0      0
2 P    0.416 S    0.703 0      2
3 P    0.831 S    1.403 0      4
4 P    1.243 S    2.099 0      6
5 P    1.650 S    2.786 0      8
6 P    2.052 S    3.465 0     10
7 P    2.448 S    4.135 0     12
8 P    2.838 S    4.794 0     14
9 P    3.221 S    5.444 0     16
10 P   3.600 S    6.084 0     18

```

Listing 2.3.1: JMA 2001 Wave Travel Tables

2.3.1.1 Format of Rows

The format of the table is as follows:

- **Column 1.** The wave (P);
- **Column 2.** Whitespace;
- **Column 3-10.** The time in seconds, with exactly 4 characters (whitespace padding to the left) before the decimal point, and 3 after;
- **Column 11.** Whitespace;
- **Column 12.** The wave (S);
- **Column 13.** Whitespace;
- **Column 14-21.** The time in seconds, with exactly 4 characters (whitespace padding to the left) before the decimal point, and 3 after;
- **Column 22.** Whitespace;
- **Column 23-25.** The depth of the epicentre in kilometres, with spaces padding to the left;
- **Column 26-27.** Whitespace;
- **Column 28-32.** The distance from the epicentre in kilometres, with spaces padding to the left.

Therefore, the following regular expression is designed to match each row (with the `\A` anchor at the start, and `\$` anchor at the end):

```
P ((?=.{4}) *\d+)\.\d{3} S ((?=.{4}) *\d+)\.\d{3} ((?=.{3}) *\d+) ((?=.{5}) *\d+)
```

Most part of this is quite straightforward, but it is worth explaining what `((?=.{x}) *\d+)` means here. The `(?=.{x})` is a positive lookahead looking x digits afterwards, but it is not really part of the match. The pattern `*\d+` is the pattern that this lookahead should match to. Therefore, `((?=.{x}) *\d+)` matches to a number that has left padding to reach exactly x digits, which is exactly what is required for the numbers in the time, depth and distance column, providing the matching for the padding.

2.3.1.2 Format of Table

The depth and the distance from epicentre is provided in the intervals specified in Table 2.3.1. They are provided for distances up to 2000 kilometres and depths up to 700 kilometres.

The table is ordered in increasing order of depth, then by distance. This means that there are

$$\frac{50}{2} + \frac{200 - 50}{5} + \frac{700 - 200}{10} + \frac{2000 - 700}{10} + 1 = 25 + 30 + 50 + 130 + 1 = 236$$

rows corresponding to each depth.

Distance	Interval for Depth	Interval for distance
0 ~ 50	2	2
50 ~ 200	5	5
200 ~ 700	10	10
700 ~ 2000	—	10

Table 2.3.1: Distance intervals in JMA 2001

2.3.1.3 Algorithm of Finding Distance from Time and Depth

However, the table gives times based on distance and depth, but this does not provide us with the distance we would like, which is drawing a circle based on how long it has been since the earthquake has happened – the other way around.

Luckily, for a fixed depth, the list is sorted since the distance is always increasing, and so is the time. Therefore, after fixing the region of rows for the given depth, we could apply a linear search (a binary search would not make much of a difference here since there are only so few rows for each depth), and find the neighbouring two times such that the current time elapsed from the earthquake happening is between them. After doing this, a linear interpolation will be done on the neighbouring time-distance (t, d) pairs to find the accurate distance (radius) corresponding with the time and the depth.

The algorithm is outlined in Algorithm 2.3.1.

Algorithm 2.3.1 Algorithm for Finding Distance based on Time

```

Require:  $0 \leq d \leq 700, d \in \mathbb{Z}$   $\triangleright$  Depth of earthquake
Require:  $0 < t, t \in \mathbb{R}$   $\triangleright$  Time of travel
Require:  $i \in \{0, 1\}$   $\triangleright$  Index of time for the particular wave (0 for P, 1 for S)
 $r \leftarrow 236$   $\triangleright$  Number of rows per depth
     $\triangleright$  Find Starting Line
    if  $d \geq 0$  and  $d \leq 50$  then
         $s \leftarrow d/2 * r$ 
    else if  $d \geq 50$  and  $d \leq 200$  then
         $s \leftarrow 25 + (d - 5)/5 * r$ 
    else if  $d \geq 200$  and  $d \leq 700$  then
         $s \leftarrow 55 + (d - 200)/10 * r$ 
    else
        return
    end if
     $e \leftarrow s + r$   $\triangleright$  End Line is exclusive
     $l \leftarrow s$   $\triangleright$  Index of Left Point used for Linear Inter/Extrapolation
    if  $T[s][i] < t$  then
         $\triangleright$  Time is shorter than first time (i.e. seismic wave hasn't reached surface)
        return 0
    else if  $T[e - 1][i] > t$  then
         $\triangleright$  Time is longer than final time
         $l \leftarrow e - 2$ 
    else
         $\triangleright$  Find the time before the given time
        while  $l < e$  and  $T[l][i] \leq t$  do
            end while
             $l \leftarrow l - 1$ 
    end if
     $\triangleright$  Coordinates for Linear Interpolation
     $x_1 = T[l][i]$ 
     $y_1 = T[l].R$ 
     $x_2 = T[l + 1][i]$ 
     $y_2 = T[l + 1].R$ 
     $\triangleright$  Linear Interpolation
    return  $y_1 + (t - x_1) * (y_2 - y_1) / (x_2 - x_1)$ 

```

2.3.2 Shake Detection

As discussed in the analysis section, one of the common features of applications with real-time intensity display map is the shake detecting feature. Specifically, if the measured intensities within a certain region shows an increasing tendency, the application will display a box around that area to indicate there is a chance that the earthquake will happen.

However, there is a chance that certain observation points will malfunction and suddenly show a big increase in intensity, which does not indicate an earthquake, and we would like to prevent this from happening. Furthermore, we wish to be able to distinguish between a big earthquake and a small earthquake on the display when such shake event has occurred.

It is worth thinking how human would see an earthquake from the real-time intensity map. First, we would notice a certain observation point has an increase in the intensity for a certain observation point. Then, we will naturally look at the observation points near it, and see whether they also have an increase, or if that was just an anomaly. If the points nearby also has increasing intensity, then we would think it is probably an event, and we will look at the observation points near that, etc. If after a certain time period, that the intensity is no longer increasing, then we may say that the shake event is over.

The algorithm used in KEVI used a similar idea, and this is the algorithm that we are going to implement in our solution. Specifically, it will create a list of observation points that are near each of the observation points (we will refer to them as neighbouring points) upon launch of the program. When there is a significant increase of the measured value of an observation point, the program will look at the neighbouring points, and if they show an increase (not necessarily significant), this will create a new event including the observation point and the neighbouring points. If, further, that the neighbouring points also observe a significant increase in intensity, then they will be added to the event, and their neighbouring points will be added to the list of neighbouring points as well.

An event could be represented by three things: the time it has happened, the observation point in the event, and the neighbouring points of those points. The neighbouring points will become observation points themselves if they have significant increase in intensity, and then their neighbouring points will become some new neighbouring points. If no such increase has been found, then after a certain time period of waiting, this event would end, and it could be seen that the primary motion of the earthquake is over.

Classification of the significance of an event is straightforward – it can be determined by the maximum intensity observed within the event. A suitable scale would be split at intensity 1 (which is which human can feel a shake), and at intensity 3 (which is when earthquake might actually cause damage), and it has to have minimum measured intensity 0.0. Most real-time measurements have negative measured intensity, so this should not lead to detection of anomalies.

Figure 2.3.1 shows the concept of an observation point and neighbouring points together with a flowchart of how the algorithm operates. The time constants of 10 seconds and 30 seconds are chosen to reflect the behaviour of common applications including JQuake and KEVI, but should be left to be adjustable in the code to ensure the maintainability.

This section referred to [34].

2.4 User Interface

The GUI will be designed using Avalonia, which is designed to develop cross-platform applications in .NET.

The nature of this application decided that the UI will be a graphical interface. Comparing MAUI in .NET with Avalonia, due to the fact that MAUI is more suitable for a mobile touch-based platform (that also works on desktop), while Avalonia is designed to support the desktop paradigm, the latter will be used to implement the graphical interface.

As discussed in the previous sections, there are three main pages that the program will include: the page for real-time observations, the page for viewing past earthquakes, and the page for controls and settings.

A sidebar is used to switch between the three pages.

The mock-ups of GUI Pages are demonstrated using [11].

Note that the following mock-up images are for demonstration purposes only – it is not an actual earthquake. They only include functionalities identified as 'key' in the analysis section, and therefore additional functionalities like shake-detection boxes, as well as customisation features are not included.

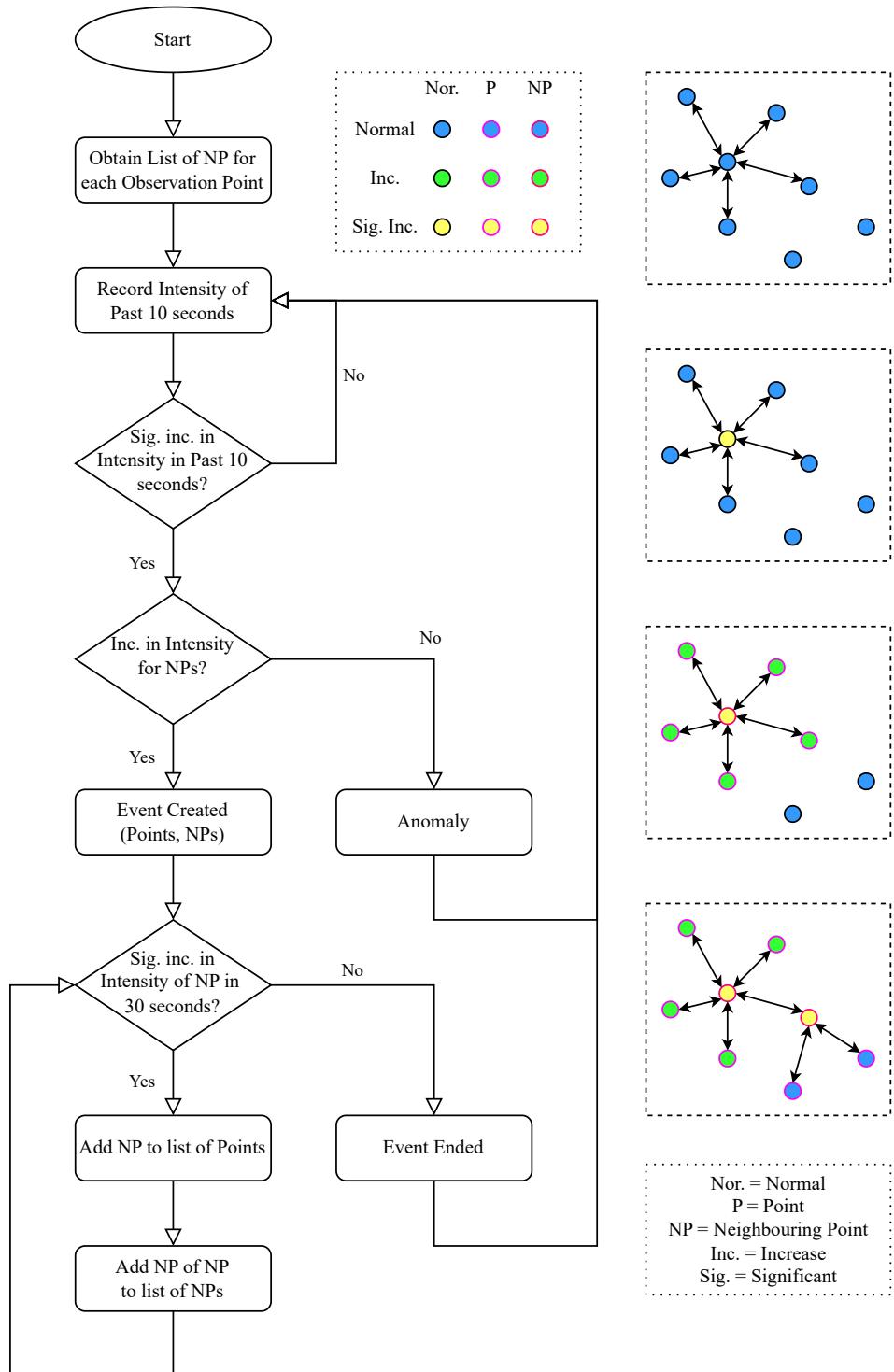


Figure 2.3.1: Shake detection flowchart and diagram

2.4.1 Map

The package Mapsui [43] will be used to draw the maps in the application, since it is one of the best-supported candidates in .NET.

The OpenStreetMap layer will be used as the background, first because it is free to use and supported very well, also does it allow users to see the actual map of the area of which the earthquake occurred.

To colour the regions and the shorelines, Shape Files (.shp) has to be used in the application. The Shape Files are sourced from the JMA webpage [72].

However, the shape files are extremely precise and accurate, tracing each and every coastline and border of the regions. However, this is not necessary for the purposes of the application (it does not make much difference zoomed out), and this will cause performance issues (loading the original shape files will cause a performance issue using more than 4 GB of RAM and will cause the application to halt).

Therefore, using the tool available on [13], the shape files are simplified using the Ramer–Douglas–Peucker algorithm, preserving 5% of the original details (which is still extremely accurate), and the line intersections are also repaired (which is a natural consequence of the algorithm). However, this algorithm is not a key part to this application, so the existing implementation on [13] is used.

2.4.2 Real-Time Monitoring Screen

The real-time monitoring screen holds the following three functionalities:

- real-time intensity display;
- tsunami warning display (including shoreline colouring); and
- EEW display (including hypocentre marking, predicted intensity colouring and real-time wavefront positions).

Figure 2.4.1 shows a design of this page that should be implemented.

The real-time intensity colour points, the current time and the colour scale will be displayed at all times. On the colour scale, there will be the name of the current data displayed, with the number to indicate the scale of the colour. The colour scale and time are situated in the bottom-left corner of the page.

If a EEW is received, there will be a side panel appearing in the top-left corner of the screen. It should display the level of the EEW, together with the serial of the EEW. The level of the EEW has priority follows: Cancellation > Final > Warning > Forecast. The EEW will take level the first in the list that it is in. The details of the EEW are displayed, together with the informational text in scrollable form.

Upon receiving or updating an EEW, the hypocentre of the earthquake will be marked on the map. If the hypocentre is a seismological hypocentre, a red cross will be used. However, if the hypocentre is assumed and does not seismological meaning, as discussed in the analysis section, a different symbol will be used. The author chose to use a dashed circle.

The predicted maximum intensity of regions will also be coloured, and the wavefronts as well, with the P-Wave being blue (since less destructive) and S-Wave being orange (since more destructive).

If multiple EEWs are in effect simultaneously, then the top-left panel will switch between the earthquakes at a fixed rate. However, the information on the map for both earthquakes will be displayed simultaneously.

The EEW shall be removed (and the layers on the map as well) after a certain amount of time.

The tsunami warning will be displayed only if it is released, or updated. The shorelines will be coloured in accordance of the predicted warning level of the tsunami on that particular shore line region. In the bottom-right corner, the level of the tsunami warning (which is the maximum of the tsunami warnings for each shoreline) will be displayed including its colour, and the last update time. There will be a scrollable bar for the informational text, since it is usually very long.

There are no buttons for the user to interact with. However, there will be scroll bars for users to view the informational text for the particular earthquake, and the user will be able to zoom/move the map (as supported by Mapsui).

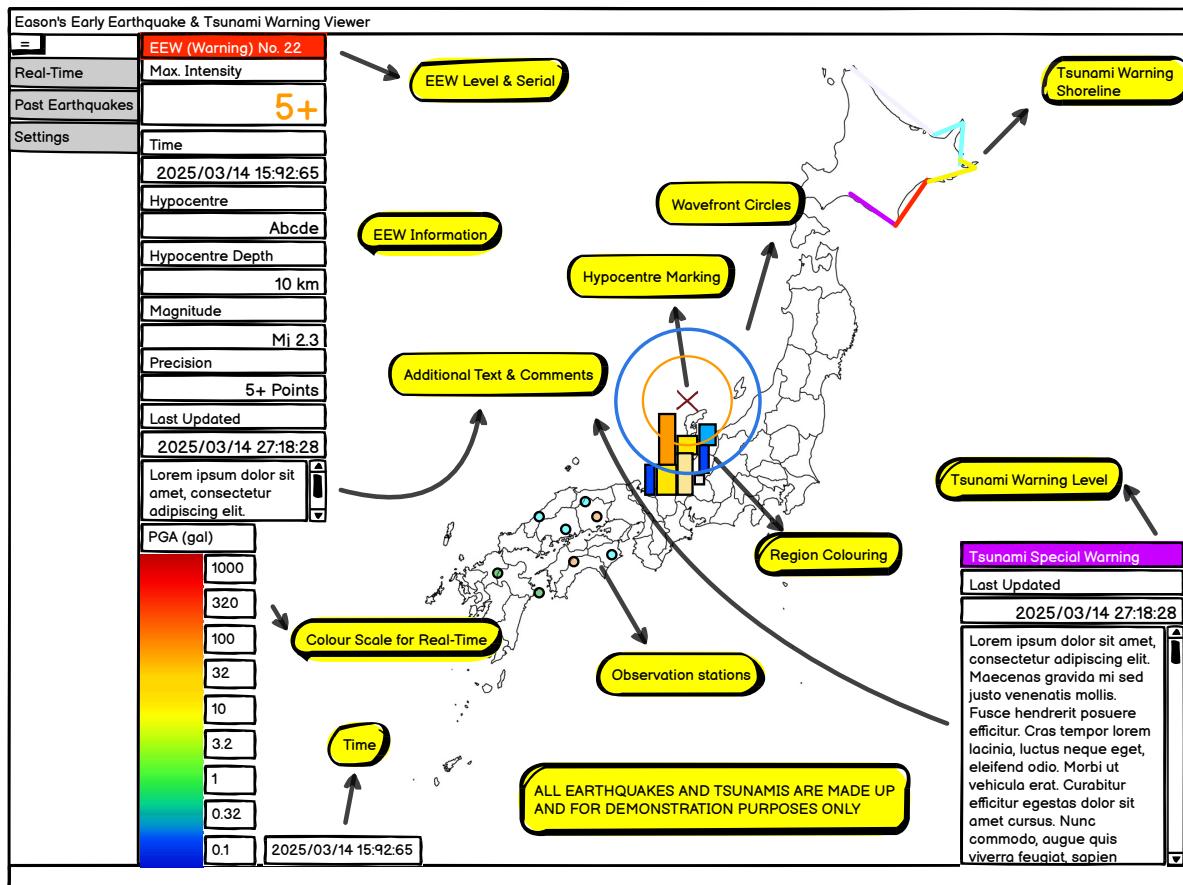


Figure 2.4.1: Design of GUI for Real-Time Page

2.4.3 Past Earthquakes

The past-earthquake screen regularly displays a sidebar to select the earthquake from, with a map in the middle, and on the left a panel displaying the detailed information of the selected earthquake. Figure 2.4.2 shows a design of this page.

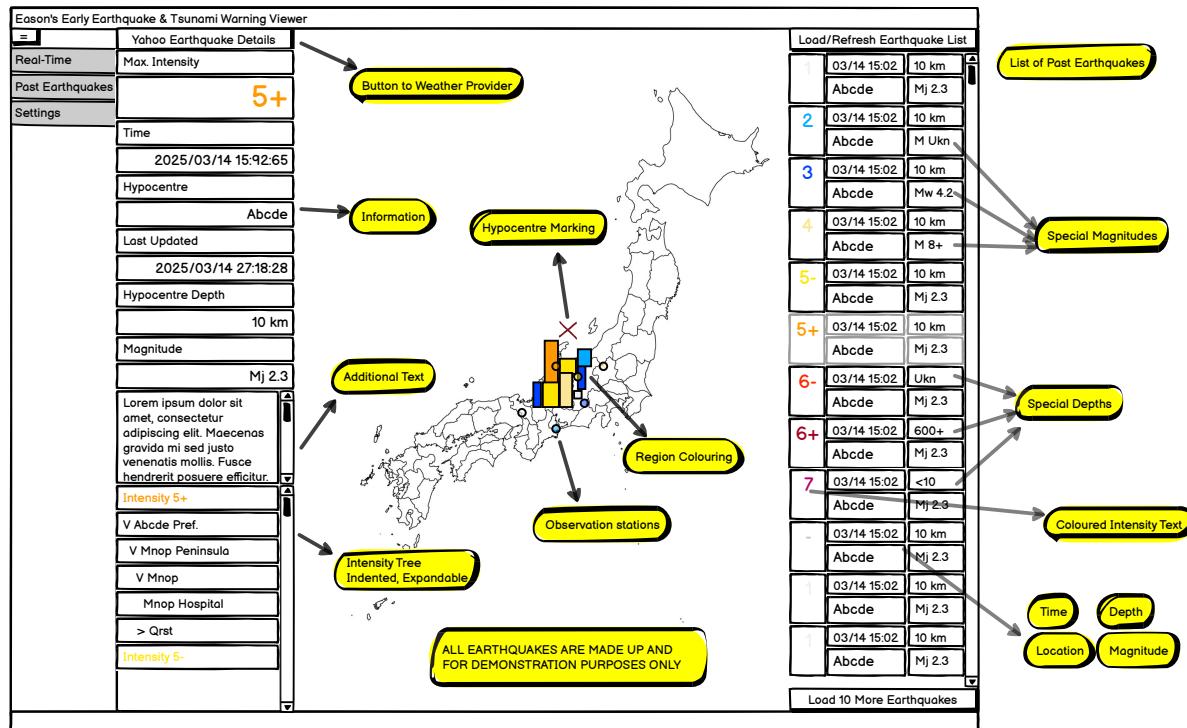


Figure 2.4.2: Design of GUI for Past Earthquakes Page

On the sidebar, for each earthquake, the intensity will be displayed, coloured as defined in the colour scheme. The name of the hypocentre's location, the depth, the magnitude, and the time of the earthquake is also included.

Notice that there are special cases of magnitudes, such as in different units, huge earthquakes or unknown, and special cases of depths as well, such as very deep, shallow or unknown.

Above the sidebar there is a load/refresh button, and below a button to load earthquakes. The sidebar is scrollable.

In the centre of the map, there is a red cross indicating the position of the hypocentre. Regions are coloured based on the maximum observed intensity in the region, and there are dots on the map indicating the observed intensity at that particular observation station.

On the left, the details of the selected earthquake will be displayed. On the top there is the button to link to Yahoo Earthquake Information. The information will be displayed more accurately here, together with the last updated time of the information. Then, there is the additional text and comments in scrollable form (since they could be quite long), and finally there is the intensity tree, first grouping by intensity, then by region, then by city, and finally the exact observation stations. The intensity level should be coloured in correspondence with the colour for that intensity.

The interactions the users is allowed to do on a page is to select an earthquake from the sidebar, and to click the load/refresh button to load the earthquakes. The user is also allowed to click on the Yahoo Earthquake Information button to view the information on the Yahoo Earthquake Information page. They are allowed to interact with the map (zoom/move), as well as using the scrollbars.

2.4.4 Settings Page

Figure 2.4.3 shows a mock-up of the design of the customisation page.

In the settings page, according to the JMA requirements, there has to be explanatory text explaining the limitations and possible errors of EEWs.

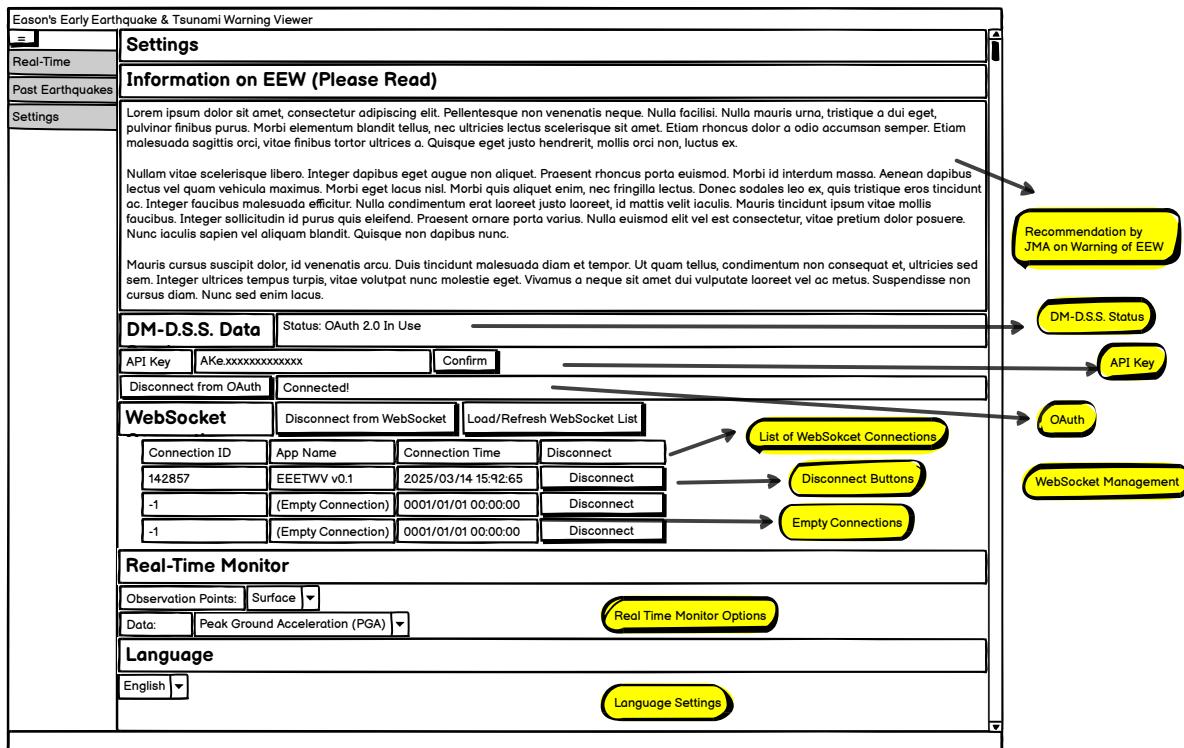


Figure 2.4.3: Design of GUI for Settings Page

There then should be a section to set up DM-D.S.S., where the user is allowed to enter an API Key and there should be a place to input an API Key from DM-D.S.S., or click a button to connect to OAuth 2.0.

Since OAuth 2.0 takes priority over API Key, the connect/disconnect button for OAuth is enabled all the time, allowing the user to authorise/revoke authorisation of the application.

The button to confirm the API Key should only be enabled if and only if all the following conditions are satisfied:

1. The API Key input box has changed;
2. The API Key has the correct format (begins with AKe .);
3. OAuth 2.0 is not in use.

Next up there is the section for the WebSocket, where a button is provided to connect to/disconnect from the WebSocket, and to load/refresh the list of WebSockets. In the list of WebSockets, the ID, App Name, and Connection Name of a connection is displayed, and a disconnect button is provided per line for each line, which will be disabled upon successful disconnection.

Finally, there are two dropdowns corresponding to the sensors and data of the real-time monitoring screen. The user should be able to select the sensors/data that they would like to view in the dropdown, and the data in the real-time page will reflect accordingly to the user's choice, including the text and the numbers on the scale.

At the end there is also a dropdown for the user to select the language of the application, which will change the language of the application, necessarily working upon restart.

2.4.5 Joint Functionalities and Technicalities

There is a joint functionality between the real-time page and the past earthquakes page, that is, when a new EEW/Tsunami Warning is released, the application should automatically jump to the real-time page to display the latest earthquake information to the user.

Since this a GUI-based application, there will be a significant use of asynchronous programming and events, which will be detailed later in the OOP section.

2.5 OOP Model

The program mainly uses the OOP (Object-Oriented Programming) paradigm, since it allows code object to be modelled like real-live objects. This is particularly useful in this application, since each service could be represented by an object (e.g. the API caller, the WebSocket connection), and each operation on it is modelled as a method.

This section will discuss some design patterns to use in the application, and will include class diagrams for modelling of services and DTOs.

2.5.1 Design Patterns

2.5.1.1 Dependency Injection and Singleton Pattern

Dependency injection is an extremely helpful tool dealing with complex classes with multiple dependencies to initialise. It would be a terrible idea for each class to instantiate the dependency themselves (e.g. the View Model for the Real-Time, Past, Setting Page instantiating their own services for API Calls, Authentications, Telegrams, etc., or the telegram fetcher and the WebSocket both each instantiating their own services to parse telegrams), since firstly it breaks the principle of single responsibility, and secondly it is performance-costly (the garbage collector has to collect garbage all the time, heap has to be assigned and de-assigned), and will lead to a lot of code duplication.

Dependency injection is achieved by sharing a **common container** of services, which .NET has a built-in **IServiceCollection** with concrete implementation **ServiceCollection**. The service container will deal with how to provide in the correct dependencies to the class that requires it. This avoids the need of complicated constructors in the code.

Singleton pattern is a pattern commonly used with dependency injection. Restricted to this application, **all services are registered as singletons**, meaning that there will only be **one** instance of each service in the application, and all services which depends on it, will **share this common instance**. Its lifetime will be managed by the container of the dependency injection (and we will not worry about it, it will get dealt with after the application stops).

The only exception to this rule is the loggers, since two loggers are involved, both implementing **ILogger**: the debug logger (by Microsoft) and my custom file logger, having different concrete implementations to the same base interface. This means they are to be registered as **enumerable** in the logger builder.

This section referred to [26], introducing the basics of dependency injection and how to use dependency injection with the Microsoft packages.

2.5.1.2 Logging

Logging is essential to the application, since it allows the developer to see what is happening, and allows the developer to deal with bugs that are reported by the users. It allows for tracking of what is happening with the application at all times.

Microsoft has a built-in **ILogger** and **ILoggerProvider** service. **ILogger** is the actual logger, and **ILoggerProvider** is a provider (factory) of the logger.

When the application is debugging, the built-in .NET Debug Logger will be used, and the logs will be logger to the output in debug. However, when the application is in production, the custom file logger will be used, and the logs will be logged to a file (a **StreamWriter** which is a file stream).

Logging is used in almost every service class of the application, unless the functionality is trivial, or any actions on the class is always visible.

This section referred to [29] introducing the Microsoft **ILogging** interface, and how to use it with dependency injection.

2.5.1.3 Adaptor Pattern

Adaptor pattern is about adapting one interface to another. Here, we will not detail in about the adaptor pattern involved between the results achieved from the API calls and the models that stores the data for the UI (this will be discussed later in the MVVM pattern).

The only two other adaptor patterns involved here, is the adaptor for logging for Avalonia and Mapsui.

Avalonia uses a very similar (but different) logging interface **ILogSink** to log, however we used the more modern **ILogSink** interface, and we will need to adapt the **ILogger** to **ILogSink**. This is achieved by an adaptor class **AvaloniaToMicrosoftLoggingAdaptor**, which will implement the **ILogSink** interface,

and aggregate a `ILogger`. The `Log` method will be implemented to call the `Log` method of the `ILogger`, and `IsEnabled` to call the `.IsEnabled` method of the `ILogger` as well. The `ILogger` provided to the adaptor class will be retrieved from the dependency injection container.

On the other hand, Mapsui uses a delegate (a callback function) to achieve logging. This is relatively simpler to achieve, since we could just attach to the `Logger.LogDelegate` method of Mapsui the log delegate we want, in our case the `ILogger.Log` method, where the `ILogger` is retrieved from the dependency injection container.

This section referred to [7, 42] on how logging is achieved in MapsUI and Avalonia.

2.5.1.4 Options Pattern

It is almost certainly a bad idea to hard-encode some configurations (especially strings) in different services, such as the location of the file for the JMA Time Table, the base URI of API Calls, and the location to store the authenticator. Therefore, a commonly-used approach in .NET is to use a file named `appsettings.json`. The bad thing about this approach, is that it is not typed, and it is stored like a dictionary with strings as keys. Luckily, Microsoft has an Options Binding pattern, which allows us to bind sections of the configuration JSON file, to an options class, which could then be injected into the dependency injection container, and used by the services.

The options pattern is used for the configuration for the JMA Time Table (for the location of the file), the configuration for the DM-D.S.S. (for the base URIs, the OAuth 2 setup options, and the start post data for WebSocket), and the configuration for the Kmoni components (for the base URI, relative URI, and the file path to which the observation points are stored). Other options will not be type bound, but retrieved in a similar method as the string key dictionary as described before.

A small caveat here is that the options bound using type `T` will be used as a `IOptions<T>` in the dependency injection container, and the options will be retrieved using the `Value` property of the `IOptions<T>`.

This section referred to [27] on best practices in binding options and how to do this with dependency injection.

2.5.1.5 Factory Pattern

First, there is the `FileLoggerProvider` (which is the provider for my custom file logger), which implements a `ILoggerProvider` to be used in the logger builder in dependency injection. Rather than providing a constructor for the user to directly call, there is a create method in the provider to create an instance, and that is essentially what factory pattern is about.

Some constructors (setting-up) of the services involved is relatively complicated (e.g. involved the base API which is a string, hence cannot be dependency injected), and others might require setting up two classes which is quite complicated (e.g. in OAuth2 services, if a new service is being set-up, then two classes are involved, one is the one getting the authorisation code and the other involves refreshing the token; while if reading refresh token from a file only the latter is needed). In these cases, factory classes or methods will hide this complexity to the users.

These will be introduced in detail later.

2.5.1.6 MVVM Pattern

The MVVM (Model-View-View Model) pattern is a commonly-used pattern in UI design. The main reason it is used is for a clear separation of concerns – the **view** describes the windows, and **binds** to certain properties in the **view model**, which is the model that the view binds to, and actual data is stored in the **models**.

As a brief description of the pattern, the following interfaces will be discussed. The detailed discussion of the view models and the models will be left to later.

INotifyPropertyChanged This interface is the most commonly-used interface in MVVM – it allows the object to notify other objects (that subscribes to it) whenever its properties has changed. It is especially useful in MVVM pattern, since it allows the model or the view model to notify the view model or the view that its properties to change, which allows the view model to deal with this accordingly, or let the GUI refresh the data of the view to update the window.

`INotifyPropertyChanged` only defines one particular thing in the interface – the `PropertyChanged` event, which is an event that is raised whenever a property has changed.

This section referred to [9].

ObservableObject, ObservableProperty and RelayCommand It would be painful to implement this interface for every single model/view model, since we would need to implement a backing (private) field, and a public property whose set property raises the `PropertyChanged` event.

`CommunityToolkit.Mvvm` addresses this problem by providing a base class (`ObservableObject`) and source generators `ObservableProperty` and `RelayCommand` to mark as attributes to use on the fields or on the methods, to generate the properties or the commands, respectively. It reduces our workload of implementing the same interface for each property repeatedly, and is very useful in the MVVM pattern. We will use this significantly to implement the view models in the MVVM pattern.

This section referred to [63].

2.5.1.7 Reflection

There are multiple enumerations that we would like to bind to in our view. Therefore, extension methods are provided to convert the enumerations to a readable string, or to a colour string, to bind to the view. However, the implementation of the Avalonia binding system decided that binding will not be possible with the extension methods.

Therefore, **converters** are used here. Specifically, the typed one-way converter `FuncValueConverter` is used. However, it would be ridiculous to write such a converter for each enumeration.

Therefore, the code is designed to put all extension methods for converting to a string in a single class, and all extension methods for converting to a colour in another class.

A converter is then written for the generic `Enum` class to convert to a string. Then, **reflection** will be used to find the list of all the methods provided by the particular extension classes, and filtering the method that takes the specific concrete type of the enumeration as the parameter, and invoke the method on it to convert to string. This is still type-safe, since the type check for parameters is used, but we have achieved 'type-safe dynamic typing' using reflection!

This section referred to [6, 48, 55].

2.5.1.8 Events

Events are raised when something of interest happened, and a **handler** is attached to the event to handle the event (i.e. executes some code). As an example, there are two events for WebSocket, the data received (which invokes the event handler in the real time view model), and status changed (which invokes the event handler in the setting view model).

The detailed use of events will be introduced in the separate components.

This section referred to [28], the .NET guidance on how to raise and handle events.

2.5.1.9 Exceptions

Exceptions are thrown when unexpected things (errors) happen in the application code (e.g. when an API call fails, when a WebSocket disconnects unexpectedly, when an attempt to write to a file fails). Exceptions are to be **handled** by the application (and should not cause the application to crash any way) and logged into the logs.

Custom exceptions are designed in the authentication, telegram, and WebSocket components in the application, and will be demonstrated later within the class diagrams.

This section referred to [22], the .NET guidance on how to create custom exceptions, and best practices on handling exceptions.

2.5.2 Abstractions (Interfaces), Options and Services (Classes)

Interfaces is an abstraction of a service, and is a contract between the service and the user on what the service will be able to do. It defines the functionality of the service, but does not provide the implementation. This is especially useful for mocking the service for unit testing purposes, and provide the possibility for polymorphism, which could be achieved by different implementation of the same service.

Options on the other hand gives the service the necessary setup data. In most cases, it makes the constructor more meaningful, and allows for the options pattern and binding to an external settings file (see later for a detailed description of the settings pattern).

Each section below is the description of one particular responsibility (service) that is necessary for the application, and in separate namespaces.

2.5.2.1 JMA Time Table

For the service of the JMA Time Table, **options pattern** is used to provide the location of the raw text file (the only setup required).

Abstraction is provided for the service that calculates the distance based on the time and depth, and this is the only responsibility of the interface, **ITimeTable**.

The services will be split into two parts: the actual service for the Time Table, **TimeTable**, and the builder of the Time Table Database, **TimeTableBuilder**, which is responsible for reading the file, verifying that it has the correct format via the Regular Expression, and parsing the raw text file into the DTO format to be transferred into the Time Table service provider. This is an example of the **factory pattern**, and it will return an object with the type of the interface, rather than the concrete class itself. This hides the detail implementation from the user, and provides the user with only an interface to interact with.

Logging will be provided for the service class **TimeTable**.

Extension methods on **IServiceCollection** is provided to **inject** a singleton of **ITimeTable** into the service collection, and the options pattern is used to provide the location of the file. This also hides the factory from the user as well.

This functionality does not have any dependencies on other functionalities.

Figure 2.5.1 is the class diagram for this functionality.

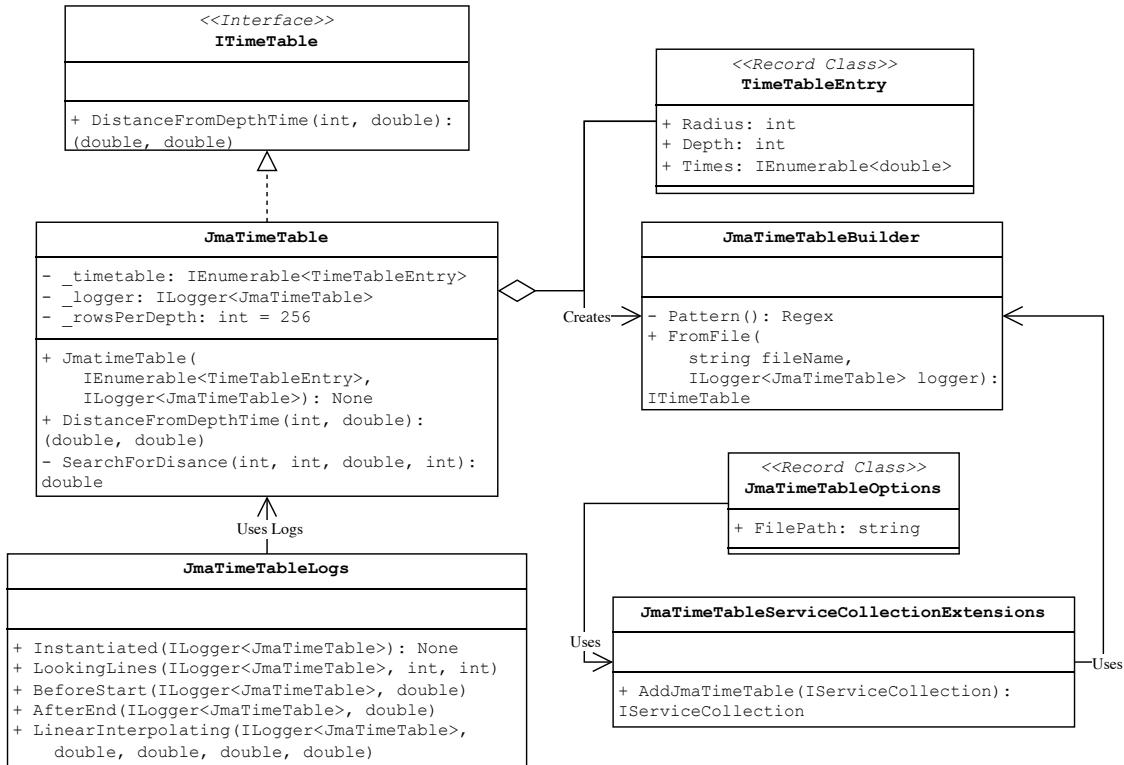


Figure 2.5.1: Class Diagram for JMA Time Table

2.5.2.2 Kyoshin Monitor

To manipulate with colours and images, the C# library SkiaSharp is used, specifically the classes **SKColor**, **SKBitmap** and **SKImage**.

The functionality is split into two parts, the fetching of the images from the webpage, and the extraction of the points using give data on the extraction of points in JSON format. As for the first

functionality, the **abstraction** used is the interface `IImageFetch`, and `ImageFetch` is the concrete implementation of it. They all depend on two enumerations, the `MeasurementType` and the `SensorType`, with an extension class providing conversion of the enumerations to a string in the URI. **Logging** is achieved by `ImageFetchLogs`.

The second functionality is the extraction of the points from the JSON data, and the **abstraction** used is the interface `IPointExtraction`, and `PointExtraction` is the concrete implementation of it.

For the DTOs to deserialise the JSON file which stores the mapping information of pixel position and geographical location, the DTO `ObservationPoint` is used, **composed** in the class `PointExtract` (i.e. they have the same life-cycle). `GeographicCoordinate`, `PixelCoordinate` and the enumeration `PointType` has composition relationship with the `ObservationPoint` class as well, since they are considered part of the observation point.

To achieve **dependency injection**, extension methods on `IServiceCollection` is provided to inject `IImageFetch` and `IPointExtract` into the service collection, using **options pattern** to set up the base URI, relative URI format string, and the file path to the JSON file storing the mapping of pixels to geographical locations.

This functionality does not have any dependencies on other functionalities.

Figure 2.5.2 is the class diagram for this functionality.

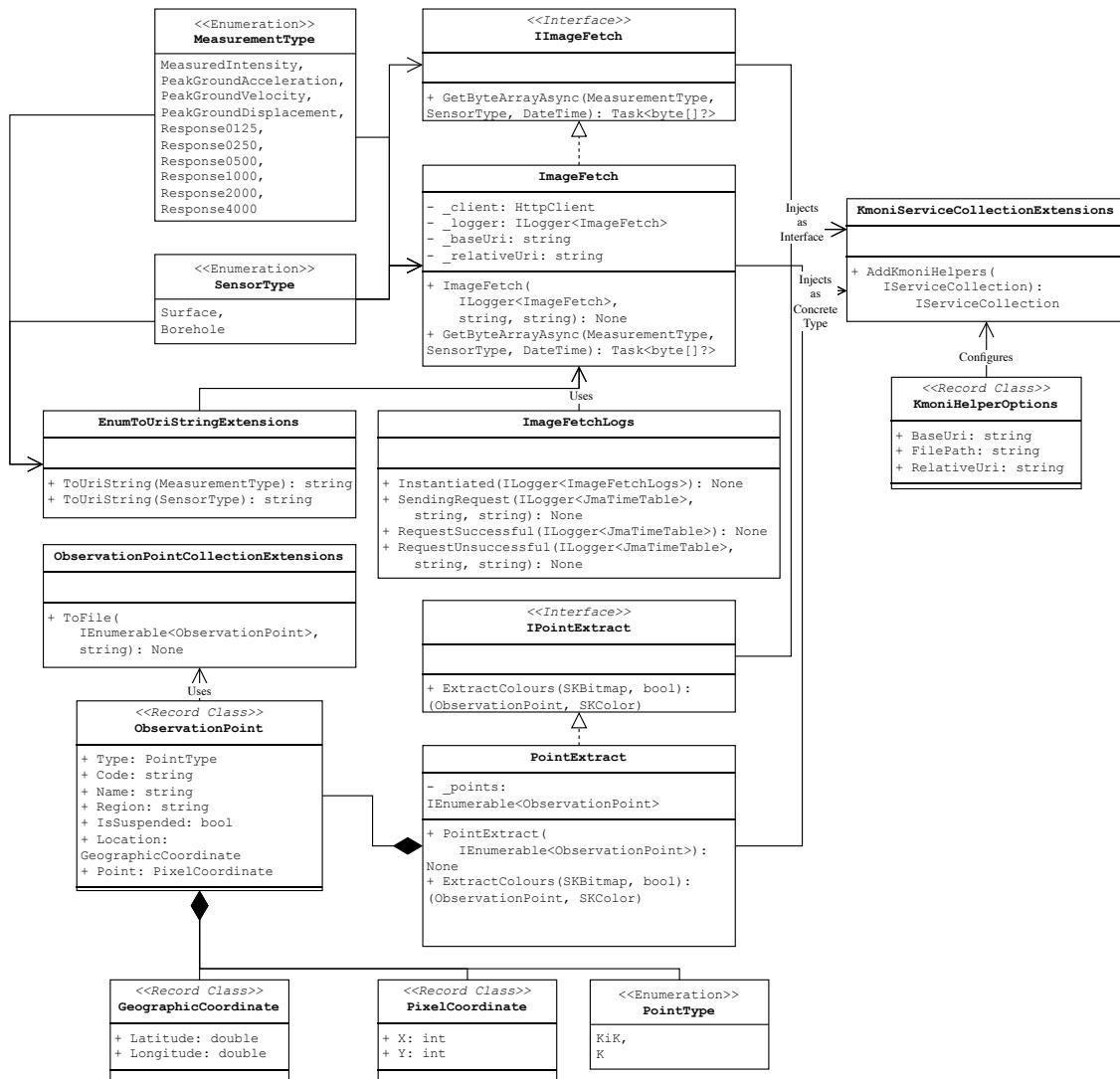


Figure 2.5.2: Class Diagram for Kmoni

Apart from those basic services, the colour conversion is included as extension methods provided in a static class. Also, an extension method is used in the conversion from the `byte[]` to `SKBitmap` to

connect between `IImageFetch` and `IPointExtraction`. Figure 2.5.3 gives the class diagram for these two Functionalities.

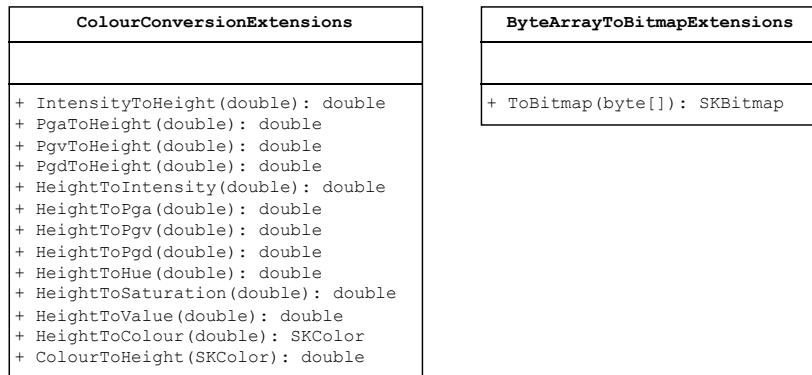


Figure 2.5.3: Class Diagram for Extra Services of Kmoni

2.5.2.3 DM-D.S.S. Authentication

As briefly discussed previously, there will be three types of authenticators involved here: not authenticated (null authenticator), API key authenticator, and OAuth 2 authenticator. To **abstract** their behaviour, the interface `IAuthenticator` is introduced. To manage the authenticators (to achieve a singleton pattern), the manager which acts as a **factory** is abstracted as `IAuthenticatorHelper`. The enumeration `AuthenticationStatus` is introduced to get the status of the current authenticator.

In terms of the concrete implementations of the interface, `NullAuthenticator` is the easiest. It will throw a custom exception `NullAuthenticatorException` when attempting to get a header from the class. However, since its behaviour is identical, it will be meaningless to create a new instance of it/dispose it when its necessary/unnecessary. Therefore, a common shared static instance is used, as in the `Instance` property of the class. The `ApiKeyAuthenticator` class also has a very simple implementation. It only has one thing, the API Key.

The OAuth 2 classes are slightly more convoluted. `OAuth2Authenticator` is the key part of the functionality, and it implements `IAuthenticator`. However, it is only its responsibility to acquire access token from refresh token, and to revoke tokens. The responsibilities of acquiring an authorisation code and the first refresh token is done by the `OAuth2Helper` class, which shows a clear separate of concern. Since there are multiple parameters necessary to set up OAuth2, the **options pattern** is used and `OAuth2Options` class provide the configuration. To hide this complexity, **factory methods** are also used, as in `OAuth2Provider`. (Shared static methods between OAuth 2 classes are in `OAuth2SharedMethod`).

Very simple DTO objects are used to store the JSON objects returned from the authentication endpoint. The `Error` DTO class represents an error response, `TokenRefresh` represents a token refresh response, and `TokenRequest` represents a token request response. However, since `TokenRequest` response is just basically `TokenRefresh` response plus the refresh token, **inheritance** is used here to reduce redundant code.

All concrete implementations of `IAuthenticator` must override the `ToString` method to provide means of saving the authenticator to a file.

The concrete implementation of `IAuthenticatorHelper` is `AuthennticatorHelper`. It is responsible for creating the authenticator based the new choice of the authenticator (null, API Key, OAuth), and a method is also provided to read and setup from the string representation of the current authenticator from a file.

Furthermore, there is an **event** in `IAuthenticatorHelper` which is `AuthennticatonStatusChanged`, which notifies other parts of the code that it has changed. The refresh of the components of GUI Setting Page is achieved by subscribing to this event (adding event handler), and the writing of the current authenticator used to a file is also achieved this way (rather than hard-coding into the code, since that would break single-responsibility).

Extension methods are provided in `AuthenticationHelperServiceCollectionExtension` to inject services into the dependency injection container. Logging is provided for `AuthenticationHelper`, `OAuth2Authenticator` and `OAuth2Helper`, since they have quite convoluted functionalities.

Figure 2.5.4 gives the class diagram for this functionality.

2.5.2.4 DM-D.S.S. API Calls

The service for API calls is relatively trivial, with `IApiCaller` representing the **abstraction**, `ApiCaller` being the **concrete implementation**, and `ApiCallerLogs` providing the **logs** for API calls. Extension methods for **dependency injection** are provided in `ApiCallerServiceCollectionExtensions`, and some other service extensions are provided.

To improve code reusability (and share error-handling code and behaviour), there are several private methods introduced in `ApiCaller` for this. For example, `CreateHttpRequest` is used to create HTTP Requests, `HandleError` is used to handle errors in HTTP Requests, and `SendRequestAsync` is used to send requests.

Notice that this class should not expose API errors to the external code that is using it – it will just return null if the API call fails.

The class diagram is in Figure 2.5.5. Notice that this does not include the DTOs, since they are not part of the service. (They will be discussed later.)

This service depends on the authentication service, and the common DTOs.

2.5.2.5 DM-D.S.S. Telegram Fetching and Parsing

The responsibilities of parsing the schema for the JSON telegram and parsing the telegram, is separate responsibilities from fetching the JSON telegram. Therefore, there are two **interfaces** involved here: `ITelegramParser` for parsing the JSON into the DTO of the corresponding schema, and determining the C# type of the object corresponding to a schema; and `ITelegramRetriever` for acquiring the telegram itself from the API.

Their concrete services are `TelegramParser` and `TelegramRetriever` respectively. However, notice that `TelegramRetriever` would aggregate an object of `TelegramParser`. This is an **aggregation** since their life cycle are not fully (though highly) correlated – the parser is provided in the constructor for the retriever, and could as well exist without the retriever.

Both concrete services have **logging**, and extension methods for **dependency injection** is also provided. Similar to previous classes, due to the existence of the dependency injection methods, the service classes are not exposed to external code, and the only way to create an instance of the service is through the dependency injection container.

The class diagram is in Figure 2.5.6.

This component has dependency on the authentication class, and the shared common DTOs.

2.5.2.6 DM-D.S.S. WebSocket

There is one **interface** for this, `IWebSocketClient`, and the concrete **service** is `WebSocketClient`.

There are two events: data received (with event arguments `DataReceivedEventArgs`, containing the telegram received), and status changed (with event arguments `StatusChangedEventArgs`, containing a boolean indicating if the new connection is connected). **Exceptions** are designed as well, but this is mainly for internal purposes, and they will not be exposed to the users of this (all handled internally).

The class diagram is in Figure 2.5.7. Note that DTOs are excluded from this.

The modelling of the DTOs for WebSocket is particularly interesting. They are all internal, since they are just designed for internal usage, and not for exposing to external code. First, a **base record class** `ResponseBase` is implemented, which contains the type of the response in an enumeration, `MessageType`. Contra-intuitively, it is **not virtual**, since we would like to first parse the data to this, determine the type, and then parse further to get the concrete type of the response depending on the type received. Different concrete types will have different `MessageType`, which is an example of **polymorphism**.

The five different types of responses, and including the two requests are all inherited from this. The reason is that they all share the same behaviour with the `MessageType`.

The `CompressionType` and `EncodingType` are described as enumerations rather than strings, since they can only take finitely many values, and this allows for easier comparison, and preserves type safety (unlike strings, which might lead to meaningless results).

Apart from this, other compositions of record classes to represents DTOs is just normal behaviour, same as what the API described. Figure 2.5.8 gives the class diagram for the DTOs involved. Notice how inheritance and polymorphism is involved.

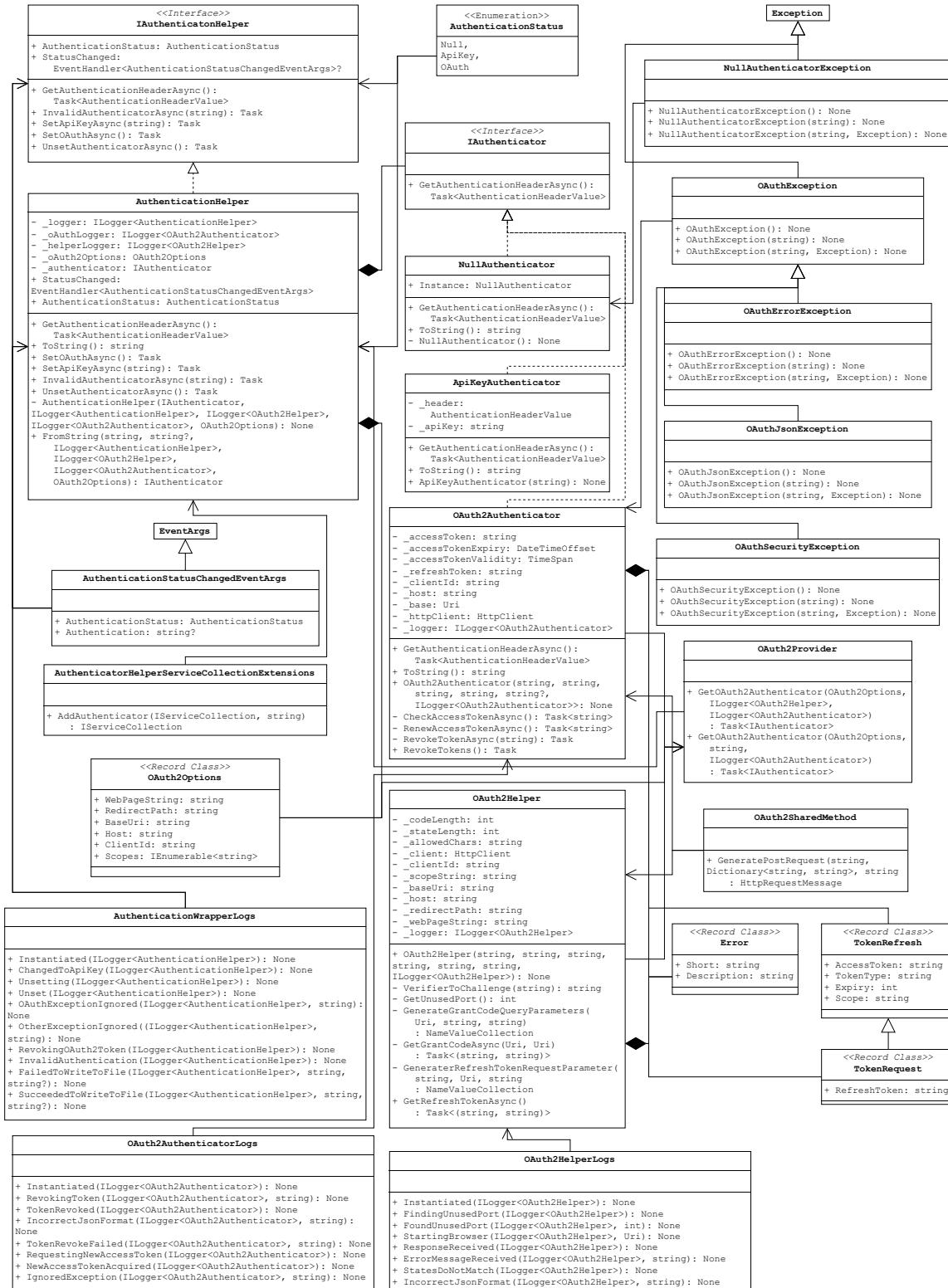


Figure 2.5.4: Class Diagram for DM-D.S.S. Authentication

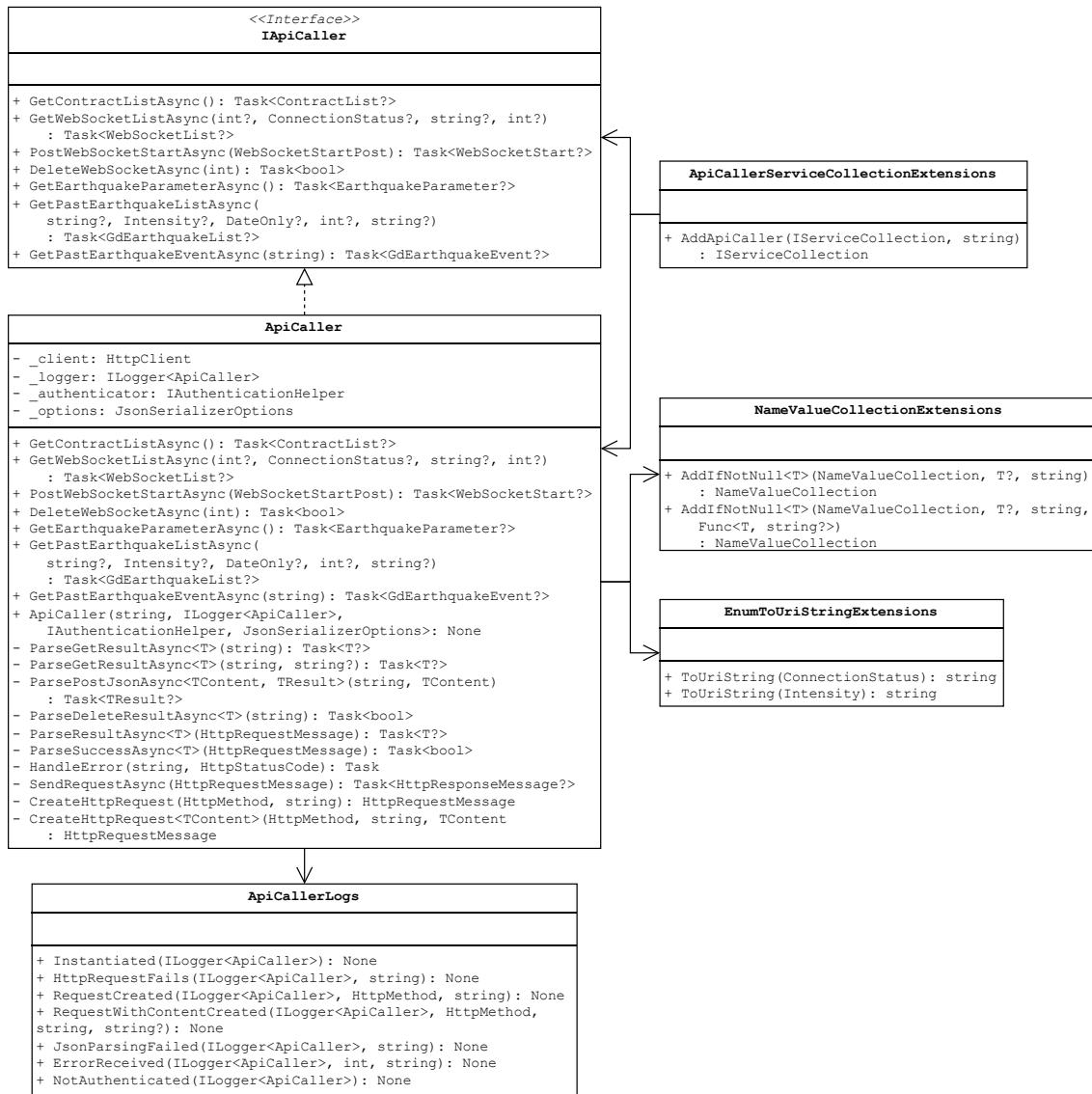


Figure 2.5.5: Class Diagram for DM-D.S.S. API Calls

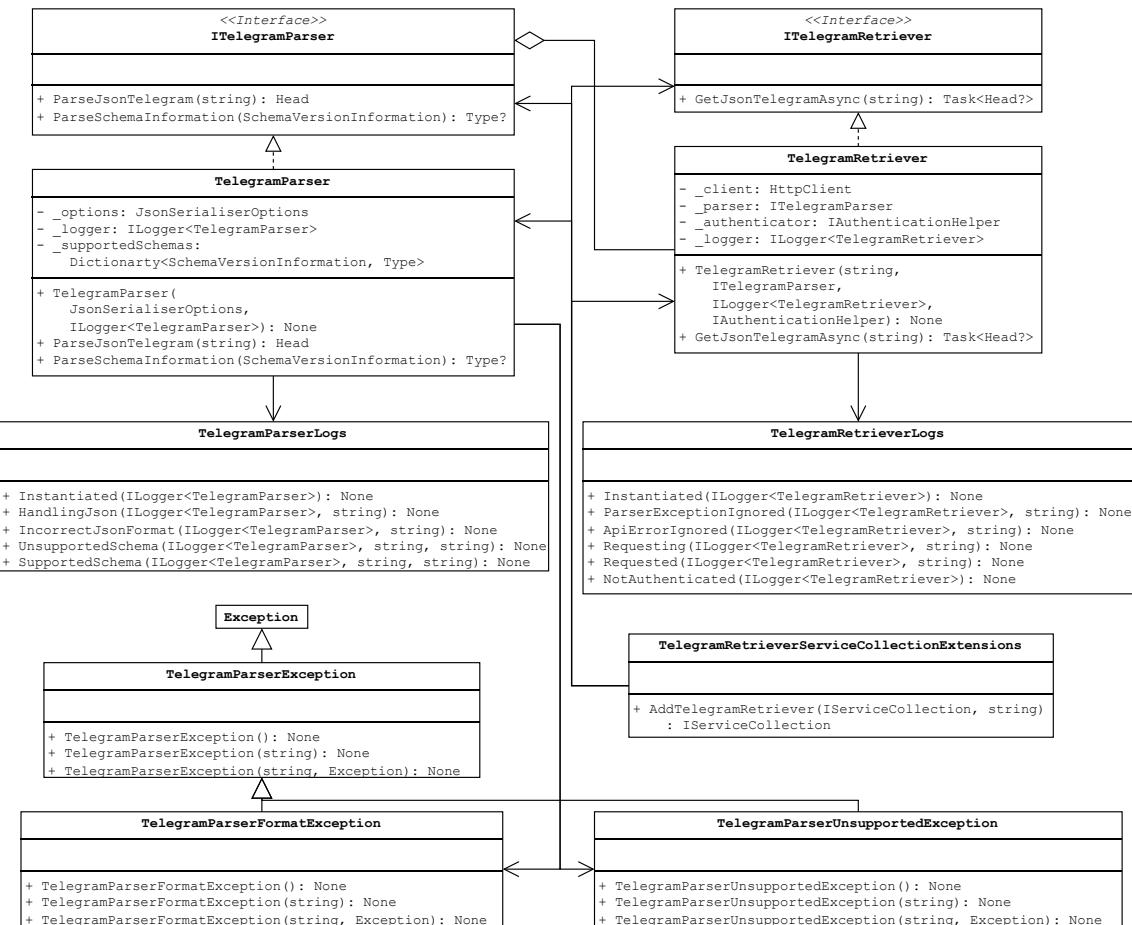


Figure 2.5.6: Class Diagram for DM-D.S.S Telegram Fetching and Parsing

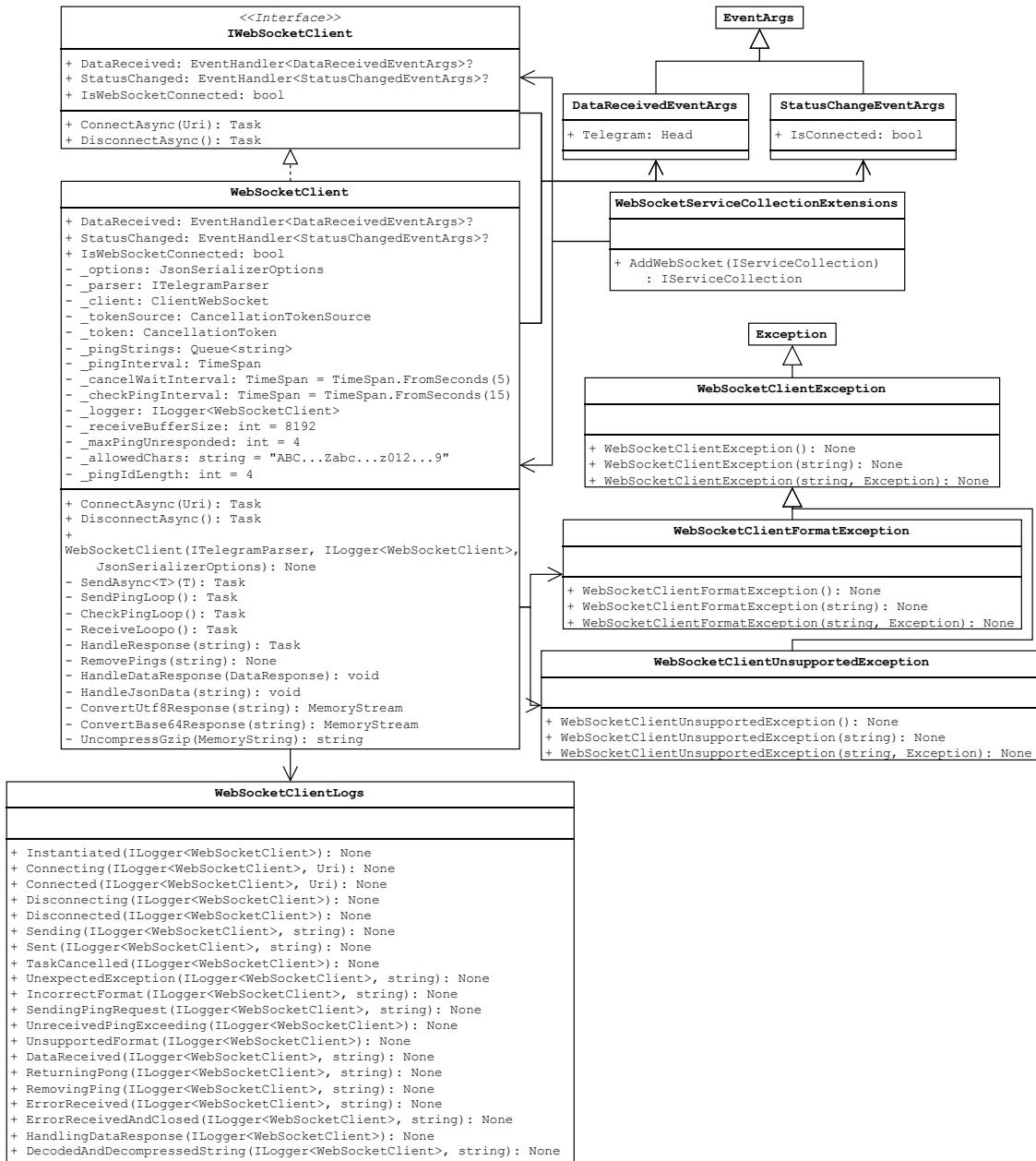


Figure 2.5.7: Class Diagram for DM-D.S.S WebSocket

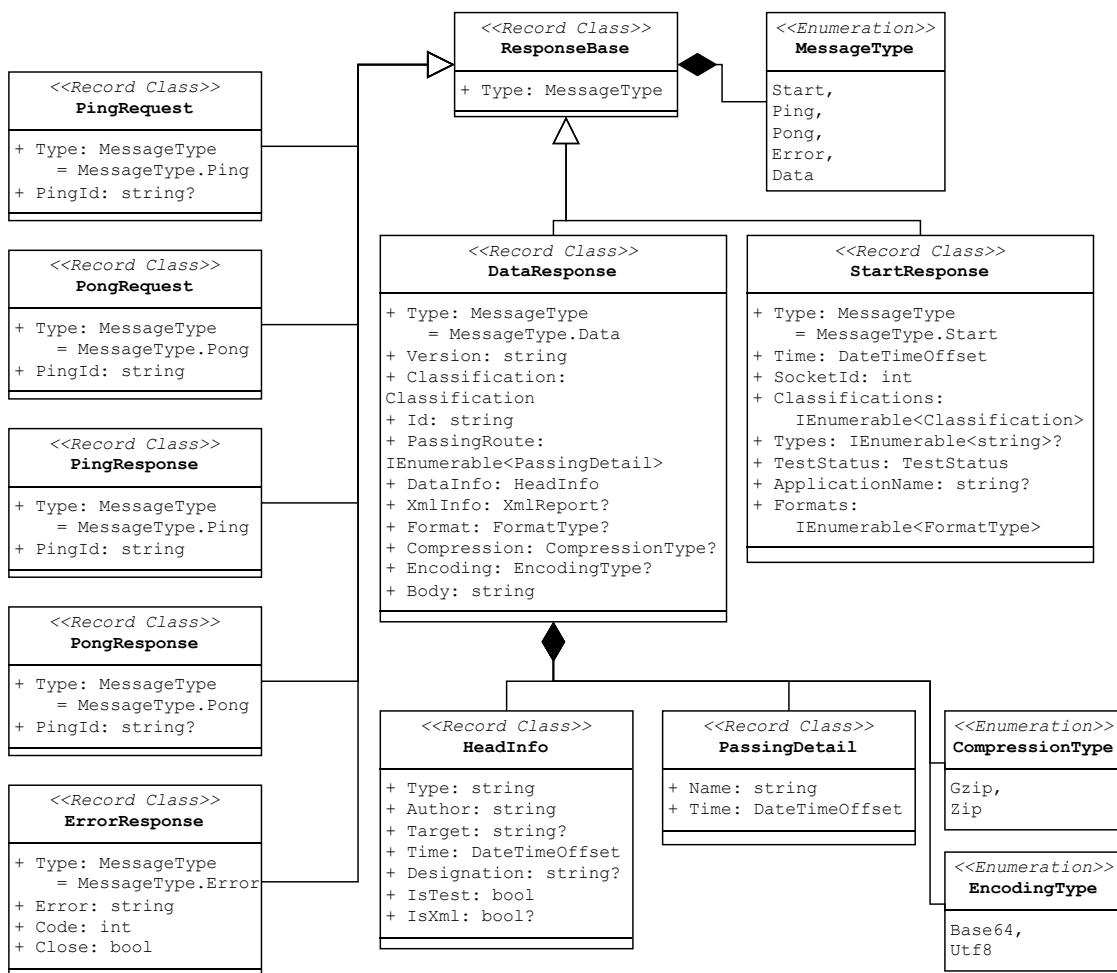


Figure 2.5.8: Class Diagram for DM-D.S.S WebSocket DTOs

This service depends on the common DTOs, and the telegram parser (but not the telegram fetcher, which releases its dependency on the authentication). This means that `ITelegramParser` is **aggregated** within `IWebSocketClient`, but not **composed**, just like with `ITelegramRetriever`. In the dependency injection, the `ITelegramParser` will be injected as well, but **singleton** pattern ensures there is only one of them in play, commonly shared.

2.5.3 Design of UI Classes with MVVM Pattern

2.5.3.1 Services

Logging Since the use of the **adaptor pattern** and **factory pattern** in logging is described before (for Avalonia and File Logger respectively), a class diagram is included here on how this is in Figure 2.5.9.

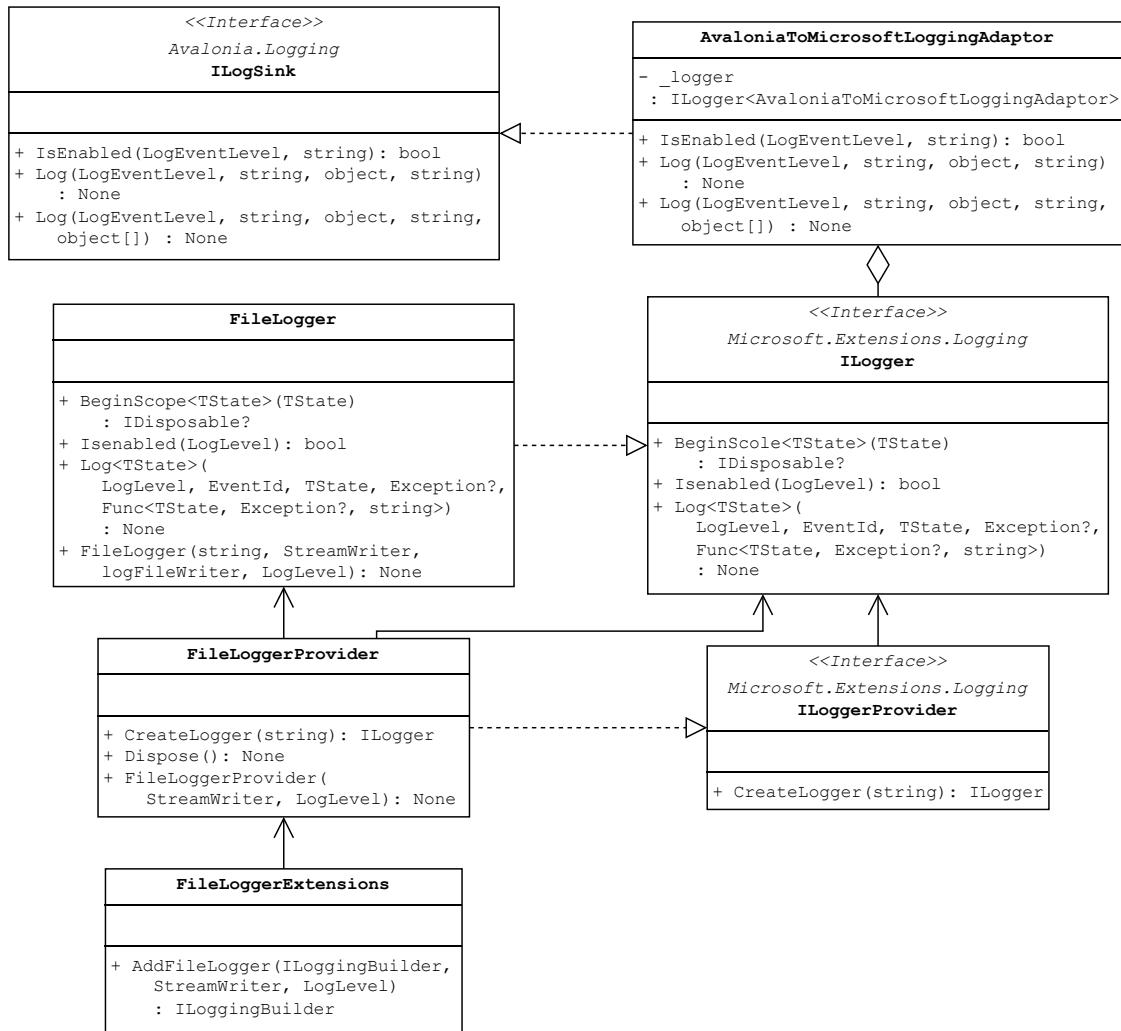


Figure 2.5.9: Class Diagram for Logging Services

Resources There are multiple map resources shared across different pages, and they are stored in the `MapResourcesProvider` class, which includes the shape files, the styles for the hypocentres, and the prefectures of Japan (which is further stored in the `PrefectureData` class). The class diagram for this is in Figure 2.5.10.

Time Abstraction Although there is an abstraction for a time provider provided by Microsoft as well, the author preferred to implement his own in the application, since the Microsoft one is too heavy

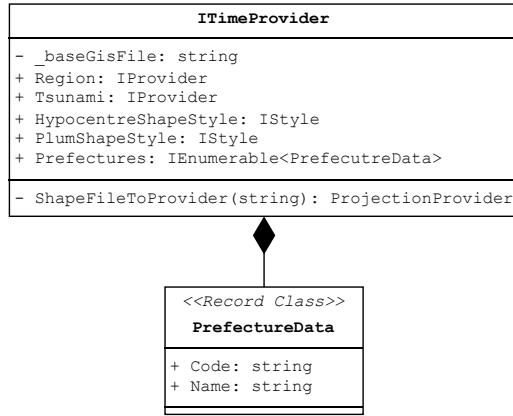


Figure 2.5.10: Class Diagram for Map Resources

but also lacks some functionality at the same time. It only achieves two things: getting the current time, and convert the time to JST, Japanese Standard Time, for communication with the Kmoni APIs. The class diagram for this is in Figure 2.5.11.

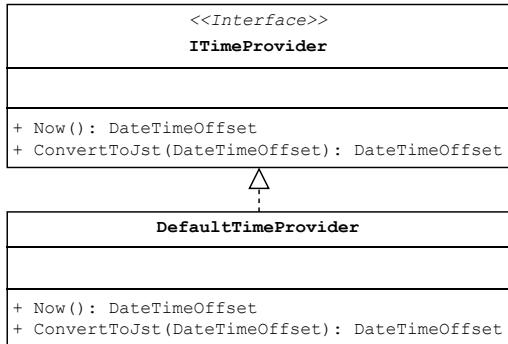


Figure 2.5.11: Class Diagram for Time Abstraction

This design might look trivial, but this first means that time could be mocked up for unit testing, and secondly past earthquakes are easier to be recreated, since a mock time provider could be provided to mock time in the past, and finally it improves code readability and code reusability, especially with the conversion of time to JST. There could also be a further functionality added, in the future where necessary, for the user to indicate whether they would like time to be displayed locally, or in JST, or in UTC.

Kmoni Settings Manager The reason why this (Kmoni) settings in particular is separated from other settings and have a full set of abstractions/options/DTO, is that it has its unique complexities: it is shared between the setting page and the real-time page, it involves the writing of the current option to a file, and it has default options if the file was not able to read/does not exist.

The **abstraction** involved here is the interface **IKmoniSettingsHelper**, which exposes the current measurement choice, and the current sensor choice. It also has an event to notify when the settings have changed (which is then used to write the new settings to the file, and in the real-time page to update the scale of the GUI to correspond to the new choice). Therefore, in the **event arguments**, it also includes a copy of the new settings.

There is a **DTO** called **KmoniSettings**, which stores the sensor type and measurement type, and is serialisable/deserialisable to JSON format, which then could be written to a file/read from a file. Furthermore, this class has **options** in class **KmoniSettingsHelperOptions**, which contains the file path to store the settings, and the default settings, both to be read from the **appsettings.json** file.

The concrete implementation is `KmoniSettingsHelper`, which is responsible for exposing the property as defined by the interface, while at the same time raising the event if the new sensor choice has changed. It also has logging involved.

Furthermore, there is a **factory method** provided as extension to the dependency injection container, which reads the current options from the files, and adds the write-to-file method to the event handler. It also does the error handling of the file read and write, and the JSON serialisation/deserialisation – this shows a clear **separate of concerns** in the design.

The class diagram for this is in Figure 2.5.12.

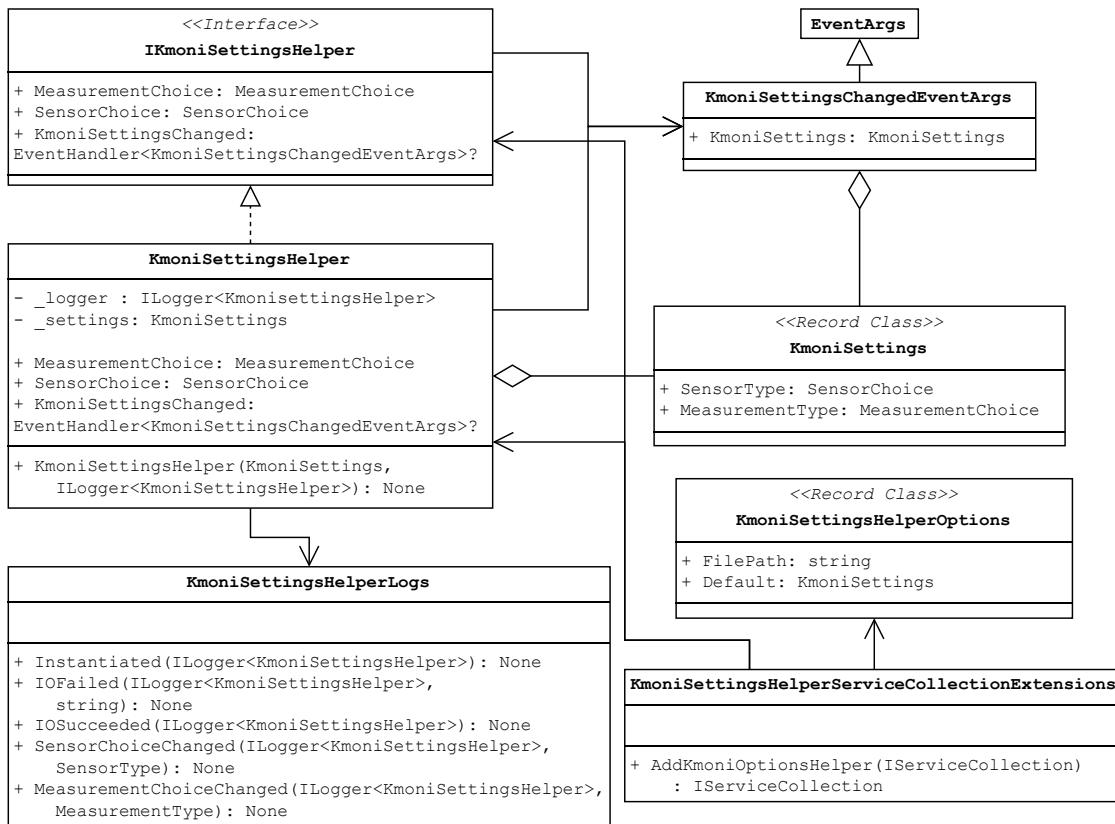


Figure 2.5.12: Class Diagram for Kmoni Settings Manager

2.5.3.2 Views, View Models, and Models

One-to-one correspondence between view and view model This principle is just as straightforward as the title suggests – there is one view model corresponding to each view, for the view to bind to.

There are four view models involved in the application: the main window (which contains functionality of the sidebar and switching the view-model), the setting page, the past earthquake page, and the real-time page. Figure 2.5.13 is a class diagram for the design of the view models.

The details of the view models of the pages will be detailed later.

Even though each view model might look bulky and heavy, the author believes this still does not break the principle of single responsibility – since each view model is only responsible for one page after all.

Abstract Base Classes **Abstract base classes (ABCs)** are used here which describes the shared parts for view models. All view models inherit from `ViewModelBase`. Then there is a `PageViewModelBase` which is the base view model for view models for a page (i.e. excluding the view model for the window itself). The `MapViewModelBase` inherits from that and provides additional functionalities for the map. **Protected fields and methods** are used to provide shared functionalities.

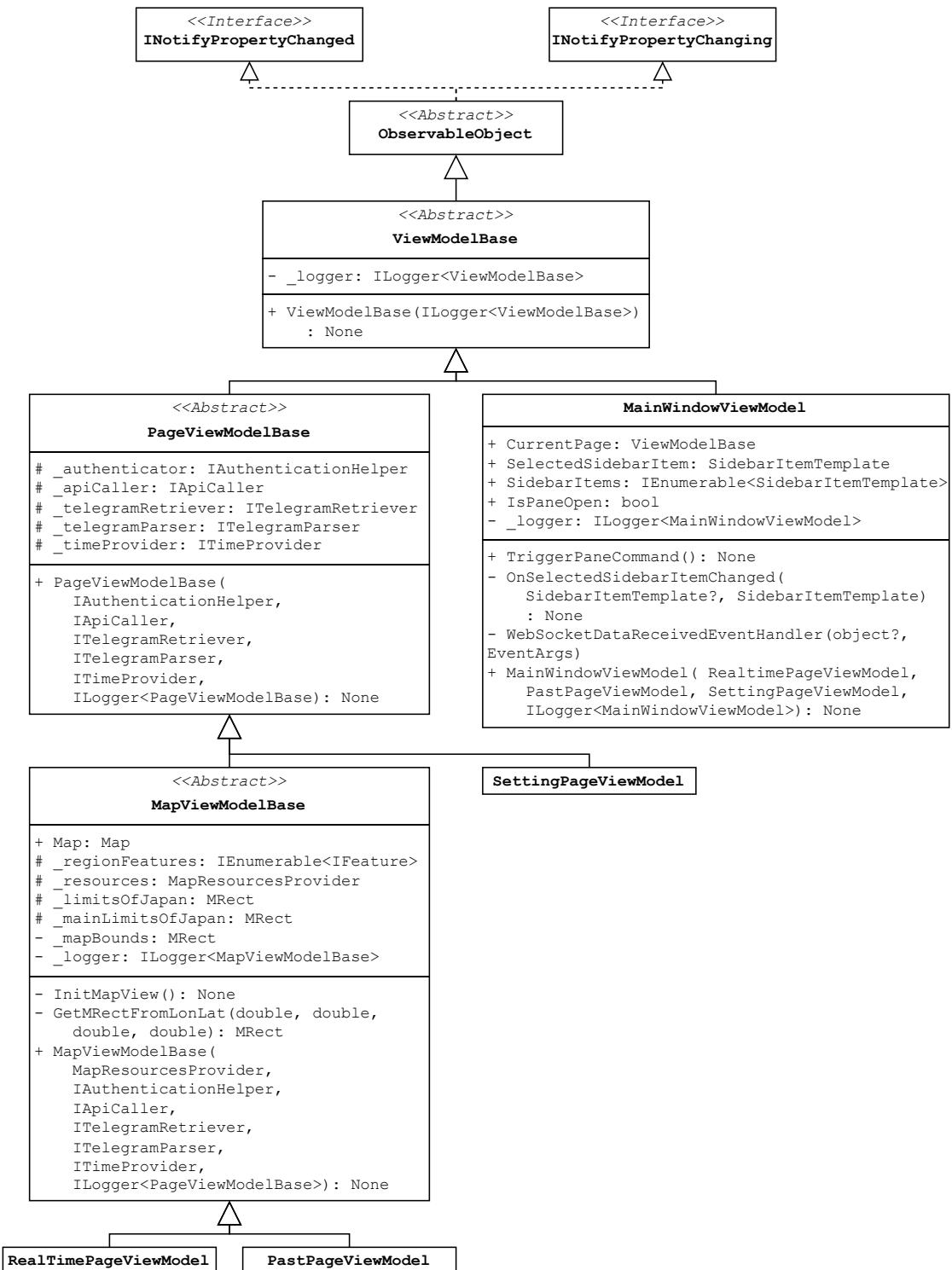


Figure 2.5.13: Class Diagram for View Models

ViewModelBase This is a class which inherits from the `ObservableObject` from the MVVM toolkit, and only involves the `ILogger` object. This might look unnecessary, but it would be extremely useful in the future, in case we would like to change the MVVM toolkit we use, or if we would like to add more shared functionalities.

PageViewModelBase This is a class which inherits from the `ViewModelBase`. It has five protected fields: the authentication helper, the API caller, the telegram retriever, the telegram parser, and the time provider for the application. They are all **aggregated** but not composed in the classes, since they are not part of the life cycle of the view model, and are passed in as constructors for the view model (as in dependency injection).

MapViewModelBase This is a class which inherits from the `PageViewModelBase`. It has a private field marked as an observable property (which makes it public anyway) for the map, and four protected fields for:

- the features of the regions read from the shape file;
- the static resources for the map, such as the shape files themselves, and the two styles for the hypocentre to be reused;
- the limits of Japan; and
- the main limits of Japan (containing the tiny islands as well).

The Window The view model for the main window has quite simple purposes: providing the sidebar to switch between views (view models), and provide a button to expand/retract the side pane as well.

The property `IsPaneOpen` is a boolean indicating whether the sidebar pane is open, and it is bound to the property of the pane. The button to expand/retract side pane is bound to `TriggerPaneCommand`, which changes the boolean value (by XOR with true).

The only model associated with this, is the item displayed in the sidebar, `SidebarItemTemplate`. It is initiated with the instance of the view model, the icon key to display on the sidebar, and the text to display on the sidebar. The view will be able to bind the related properties by accessing the list of sidebar items. Since this instance is created in the constructor of the view model, it is considered as **aggregation**, since their life cycle is strongly correlated.

Upon the chosen item changing in the sidebar, the `CurrentPage` will change as well, allowing the view of the window to change the binding.

It will also handle the event raised by the real time view model requesting to change the view model to the real time view model, and this will be done within the constructor.

Figure 2.5.14 is the class diagram for the window view model and the related components.

Models for Setting Page There are **four** primary functionalities of the settings page: changing the language setting, the Kmoni setting, the WebSocket settings, and the authentication settings.

Figure 2.5.15 is the class diagram for the settings view model and related components.

Language Settings There is an **event**, `LanguageChanged` to be invoked when the language is changed. This is not used in particular for external code – this is used as a means to clean up what is to be done when the language has changed. In the current design, only two tasks are to be performed: change the current language of the resources files, and to write it to the file which stores the language. The existence of this event allows the functionality to be extended, such as popping a window for the user to confirm, in the future.

The property `LanguageChoice` is exposed to the view model to bind to the current language choice, and `LanguageChoices` is the list of languages available. To be consistent with the resources file, the type `CultureInfo` is used to represent the language.

Kmoni Settings For the Kmoni settings, there is a `IKmoniSettingsHelper` **aggregated** within the class, and two lists which contain the choice of the sensor type and the measurement type. The properties in the settings helper are directly bound to the view model, and all properties changed are handled within the settings helper.

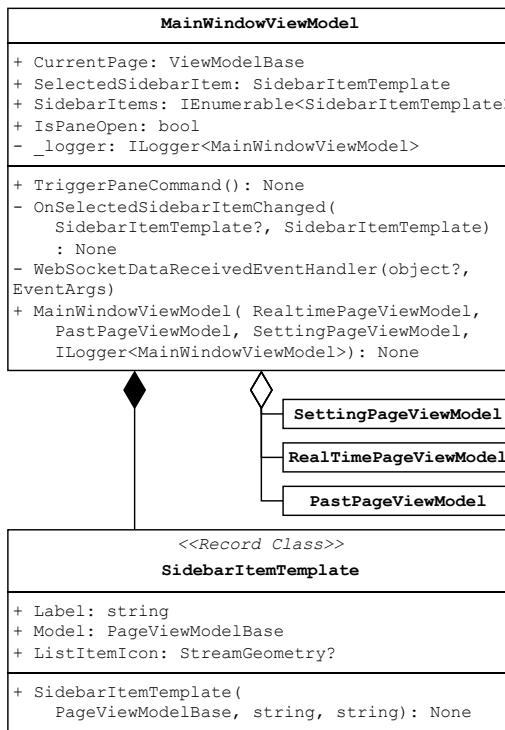


Figure 2.5.14: Class Diagram for Window

WebSocket Settings This part aggregates a **IWebSocketClient**, which is the actual WebSocket connection. There is an event handler for the WebSocket connection status changing, since the view model has to notify the view that certain properties have changed (the current WebSocket status). There is a text to bind to the WebSocket connect/disconnect button (which will change when the WebSocket status is changed), and there is a command bound to it as well to connect/disconnect.

In the table there is a list of current **WebSocketConnections**, which is an **ObservableCollection** of the interface **IWebSocketConnectionTemplate**. The use of the observable collection is since it will be able to notify the view whenever its members have changed. The interface of the WebSocket Connection declares four properties: the WebSocket ID, the application name, the starting time of the connection, and whether the disconnect button is enabled. A method for disconnecting is also provided. This is an example of the **adaptor** pattern, since it adapts the objects returned by an API call to the contract we have with the view for binding.

The reason why there is a use of the interface here, is because of **polymorphism**: there are two concrete implementations of this, the first an empty WebSocket connection, which just stays put (and shares a common instance), and the other an actual WebSocket connection, as will be shown in the class diagram below. They behave differently (especially their disconnection command). The actual WebSocket connection also inherits from Observable Object to allow the notification to the view that its properties have changed.

To implement the refreshing the list of current WebSocket connections, there is a separate method to get the number of total WebSockets available to the user (which looks through the user's contracts), and a separate method to list all currently active WebSockets. The observable collection then could be altered to reflect the latest result.

Authentication Settings For the setting for authentication, there is first property exposing the current **AuthenticationStatus** to the view via the authenticator, to display the current authenticator. There is also an event handler to handle the authentication status changed event, which notifies the properties that are related with the current authentication status.

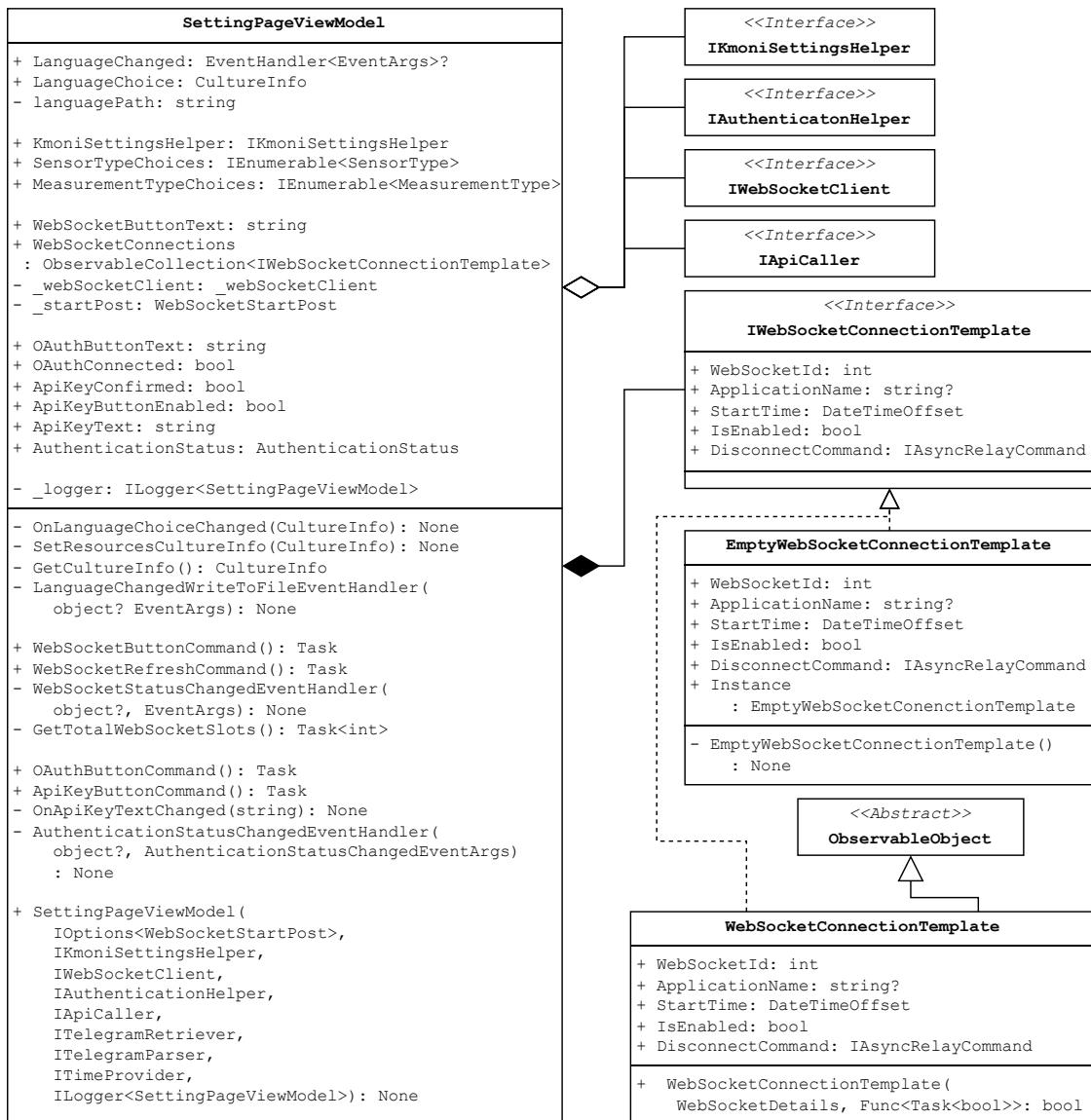


Figure 2.5.15: Class Diagram for Settings

There are two commands exposing to the view model, the button to set the current API Key, and the button to connect/disconnect from OAuth 2.0. There is also a string for the current API Key Text, a string for the text to display on the OAuth button, a boolean for whether OAuth connection is connected, a boolean for whether API Key is confirmed, and button for whether the API Key button is enabled (i.e. the API Key fits the format, which will be checked every time if the API Key is currently valid).

Past Earthquake Page Splitting up the functionality for the past earthquake page, there are two main functionalities: the past earthquake list, and the past earthquake details display. Hidden behind this is a functionality to retrieve and cache the list of earthquake observation stations (which we will regard as the third functionality).

Figure 2.5.16 is the class diagram for the past earthquake view model and related components.

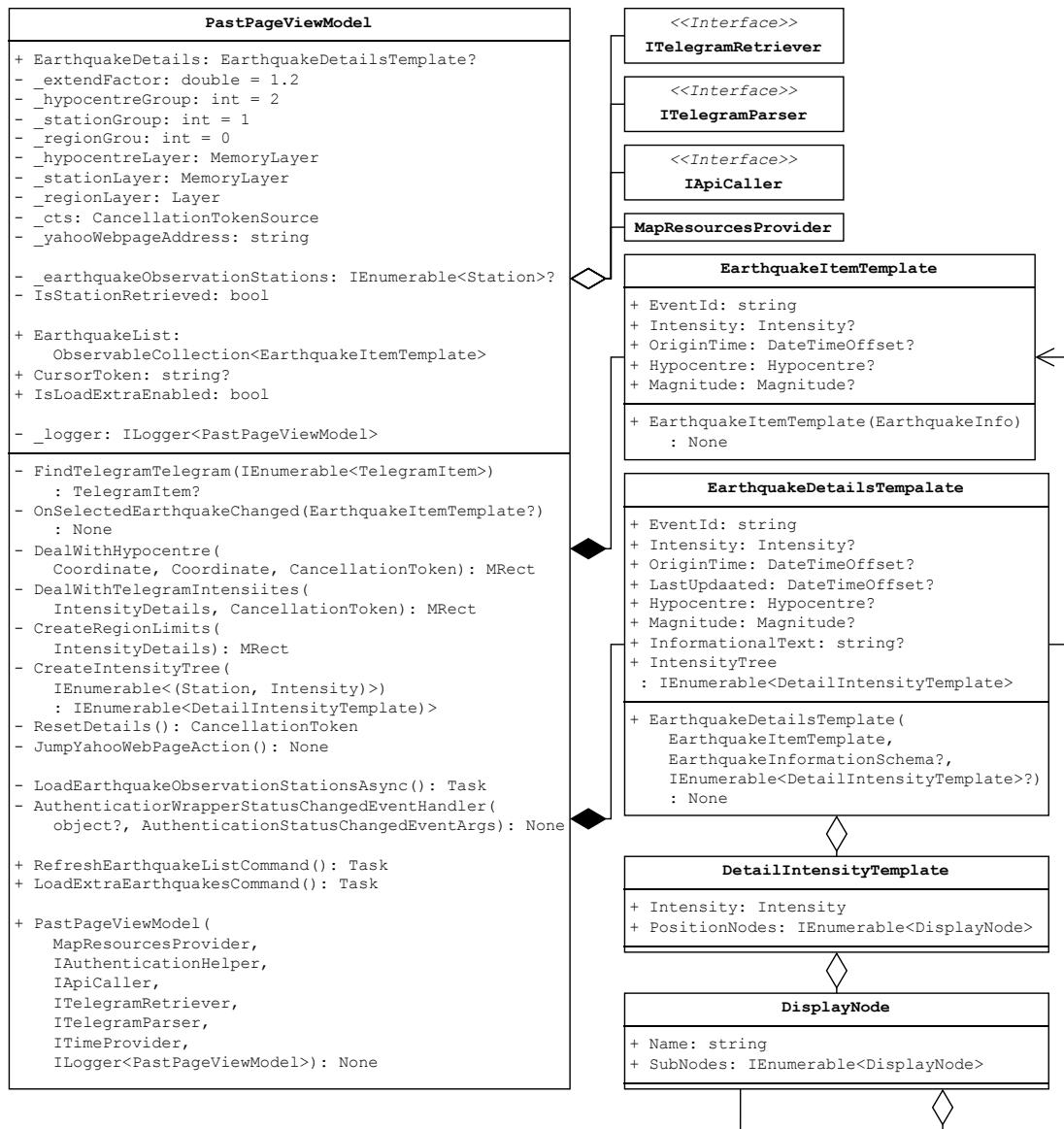


Figure 2.5.16: Class Diagram for Past Earthquake

Earthquake List To achieve the list of earthquakes, the user would have to press the refresh button, which calls the command associated with it, and attempts to fetch a list of earthquakes, stored

into an observable collection. **Note that the observable collection is annotated with observable property as well**, since we could just totally re-set the list to a new one, and the view will be able to reflect the change. In this method, the cursor token will also be stored to a private field, and used to load extra earthquakes. There is a boolean field reflecting whether the load extra button is enabled, by checking if the cursor token is null.

The model to store the earthquake list item takes one earthquake from the API call as in the constructor, and changing it to the contract to the view, which is another demonstration of the **adaptor pattern**. It only keeps the necessary properties for binding as well.

Observation Stations To load the list of earthquake observation stations, the user has to be authenticated first. If the constructor detects that the authenticator is in an authenticated stage, the constructor will attempt to fetch the station list at the start. When the authentication status is changed, there is also an event handler that attempts to fetch the list as well.

There is a method to load the stations manually as well, and allows for methods to check for if the list is initialised, before attempting to use it.

Earthquake Details This section has the most details and has the most complicated functionalities. First, three layers are repeatedly used over the life cycle of the application: the layer for the hypocentre, for the observation stations, and for the regional intensities. When a new earthquake is selected, it will not attempt to create a new layer – instead, it will change the data/style of a layer, and notify the layer that the data has changed.

Secondly there are constants defining the layer of the groups. It is designed such that the hypocentre always stays at the top of the map, stations the next, and the hypocentre in the bottom layer. There is also a **cancellation token source** – this is involved, since if the user has changed the mind of which earthquake they want to view the details of, before the details have been reflected on the map, we do not want it to be reflected any more – and we would like to cancel the asynchronous task. Therefore, in the rest details method, it will cancel and dispose of the current cancellation token source, and set up a new one for the next chosen earthquake.

There is a method to jump to the Yahoo earthquake page for the details of the earthquake as well, using a constant private field of string for the URL to be formatted.

Now comes the interesting part on how to deal with when a new earthquake item is selected.

- MVVM invokes the `OnSelectedEarthquakeChanged` method, which first resets the current earthquake details, and then attains a new cancellation token for the current earthquake.
- The method gets a list of telegrams for the current earthquake, and uses **filter, group by, select and order** methods as in `FindEarthquakeTelegram` to decide the most suitable telegram to get the information from, as described earlier in the design for the UI page.
- It creates the **intensity tree** and the **regional limits** (map bounds of regions which observed intensity) for the earthquake by calling the `DealWithTelegramIntensities` method.
- It marks the hypocentre on map and obtains limits for the hypocentre by `DealWithHypocentre` method.
- It zooms the map to the union of the regional limits and hypocentre, multiplied by a constant `_extendFactor`.
- It creates the details of the earthquake to display on the side panel, using the intensity tree created, the acquired telegram, and the earthquake list item.

In the `DealWithHypocentre` method, it will change the features of the hypocentre layer and notify the feature has changed, and return the limits for the hypocentre.

In the `DealWithTelegramIntensities` method, it will first **filter** where the intensity is not unknown, and **join** it with the list of observation stations.

Then, it will call method `CreateRegionLimits` which **aggregates** the region limits of each region involved using the union operation, and the `CreateIntensityTree` method which **joins** the station data further with a list of prefectures, and group by intensities, and group by prefecture, region, and finally by city, to create a **tree** structure.

There is a significant use of **list-manipulation methods** here, using existing **LINQ** methods including join, aggregate, select, where, group by, order by, and first.

As for the models used, `EarthquakeDetailsTemplate` is just `EarthquakeItemTemplate` with some more properties to display on the sidebar, and takes it and the telegram and the intensity tree within the constructor.

The particular interesting thing here is the **intensity tree**: it is a tree **with different data types** on different depths. Its root could be considered as the earthquake detail object, and on its first layer, the **intensity** is stored, but afterwards it stores a location, which could simply be it stores a **string**.

Therefore, there are two classes involved here: `DetailIntensityTemplate`, which is for the first layer, having an intensity and a list of `DisplayNode` (which is its children), which has a name, and a list of `DisplayNode` (which is its children). The whole tree is stored in a list of detail intensity template, which is a tree itself. Therefore, as shown in the class diagram, display node is aggregated within itself and detail intensity template, which is further aggregated into earthquake details template (it is not composition since it is provided at the constructor of earthquake details template).

Real-Time Page This is the view model with the most functionalities, but we can split it up into five parts: common parts of the view model, handling EEW, handling Tsunami, handling WebSocket, and Kmoni monitor functionality. Figure 2.5.17 is the overall class diagram for the real-time view model and related components.

Common Parts There are two **timers** shared by all the parts: a timer that invokes events every second (which includes refreshing the time display, removing expired EEW/Tsunami warnings, and refreshing the Kmoni layer), and a second one that invokes events every 2.5 seconds (which includes the switching of the EEW, if applicable). These event subscribers will be added to the timer within the constructor.

Furthermore, the management of group layers from top to bottom is as follows, the layer number defined as a constant integer field (for code reusability and readability):

- Hypocentre,
- Wavefront,
- Tsunami,
- Kmoni,
- Intensity Regions.

EEW The key part of the EEW is that there could be multiple EEW in effect at the same time. However, there is one thing that is unique for each EEW: their event ID. Therefore, a suitable data structure to store this collection of EEWs is a **dictionary**, with the event ID in **string** as the key, and the EEW in `EewDetailsTemplate` as the value.

This naturally leads to switching the current display EEW. This is controlled by `CurrentEewIndex`, a nullable integer for the current EEW index, and `CurrentDisplayEew`, a nullable `EewDetailsTemplate` which has a get method using the index to display the earthquake. When the `SwitchEew` event handler is invoked, it will do the follows:

- If the number of current EEWs is zero, set the index to null;
- Else, if the current index is null, set the index to zero;
- Else, increment the current index by one, and modulo the length of the number of current EEWs.

The design of the `EewDetailsTemplate` is as follows: apart from the standard information to display, there are two properties of type string, `Before` and `AfterIntensityText`, which gives the text to display before/after the intensity (e.g. if the intensity provided in the EEW is 'above 6+', then the before text will be 'above'. In Japanese the after text will be '以上', since the sequence in the language is different.) There is also an enumeration determining which enumeration to display (if it is (6-, 6+), then it will display 6+; but if it is above 6- i.e. (6-, above), then it will display 6-). Finally, notice there is a cancellation token source and cancellation token here – this indicates whether this EEW has already been removed from the collection of EEWs, to allow for asynchronous operation to halt.

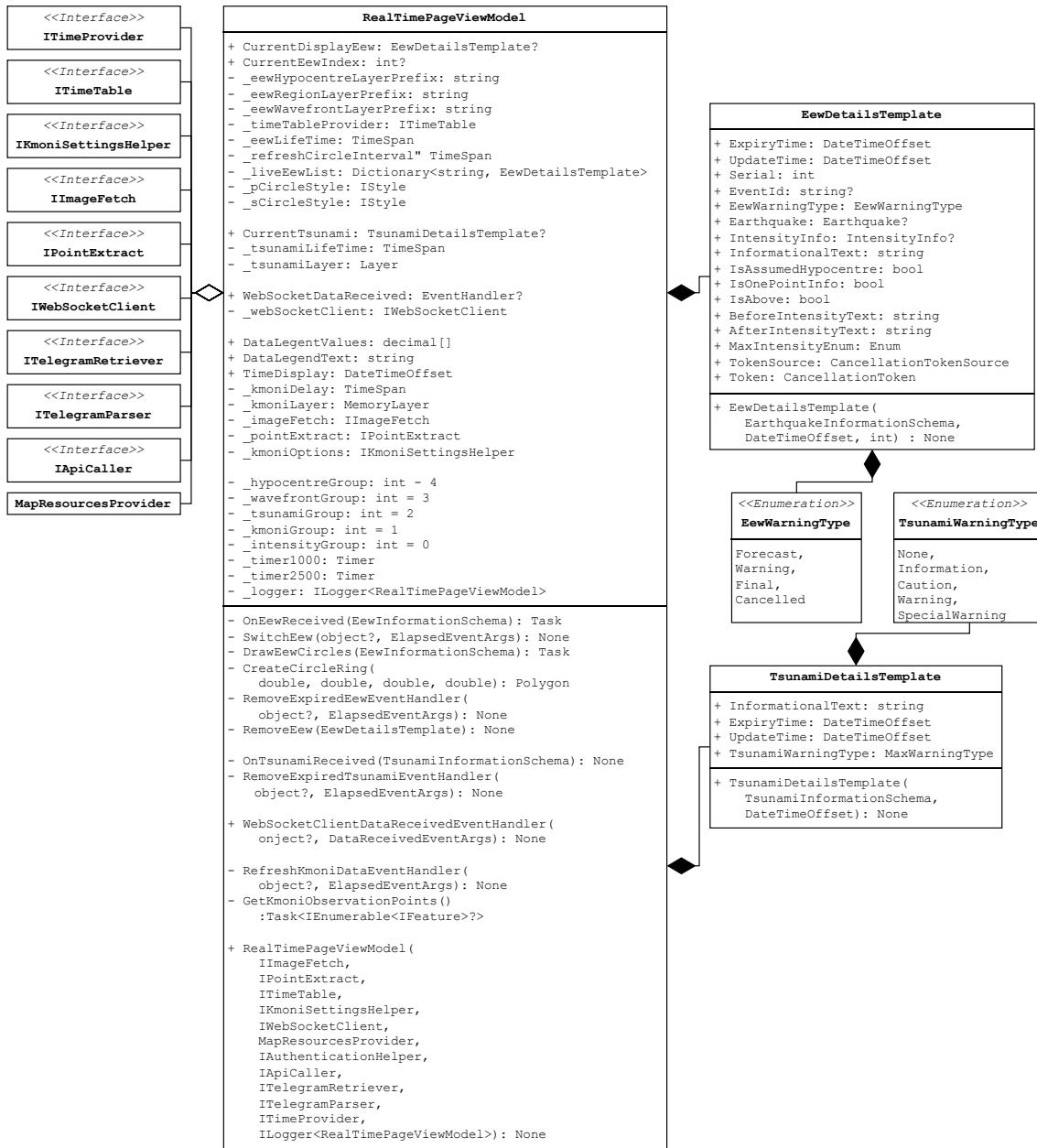


Figure 2.5.17: Class Diagram for Real-Time Page

The type of the EEW is described by the enumeration `EewWarningType`, and the logic of deciding the type of the EEW was discussed before in the design of the UI.

The handler for receiving an EEW is `OnEewReceived`. First, it determines if the EEW is newer than the current EEW in the collection, with the same event ID (making the decision based on the serial number). If not, then it will do nothing; if yes, then it will remove the EEW in display. Then, it will plot the coordinates of the hypocentre on the map, using the corresponding shape style depending on whether the hypocentre is assumed, and also display the regional intensities colouring on the maps. Finally, it will start a long-running background task on a separate thread to draw the wavefronts for the EEW.

This is the asynchronous `DrawEewCircles`. First it checks if the EEW fits the condition to draw the wavefronts (e.g. is the earthquake, hypocentre, origin null, or is it an assumed hypocentre). If not, it will then check if the depth is within the acceptable range. If all the checks passed, it will then insert two layers onto the map, one for the P-Wave and another for the S-Wave, which will be re-used throughout the duration of the EEW, updating only the features.

Then there is a while loop, which will continue until the cancellation is requested. During the loop, it will find the time that the earthquake has elapsed, and find the distance the seismic waves has travelled through by checking with `ITimeTable`, and adding the feature onto the layer, and notifying the layer that its data has changed. The loop will sleep for a certain amount of time, and then repeat the process. After quitting the loop, the layers will be removed from the map.

`CreateCircleRing` is a method designed to draw a circle in `Polygon` at the given position and radius, to a good quality. Basic geometry is used here, but since the projection of the map changes the length, a magic constant is introduced to the radius to correctly plot the circle with approximately the correct radius.

The method to remove an EEW is as follows (assuming the EEW is removed from the collection): the cancellation token source will be cancelled and disposed of, and then the maps layer of hypocentre and regions, with the correct event ID attached to, will be removed. This will also be regularly called by a method which is invoked by the timer to regularly iterate through the dictionary to see if there are any expired EEWs.

Tsunami Since it is almost certain that there is only one tsunami warning in effect at one time, the model `CurrentTsunami` is simply a single object of type `TsunamiDetailsTemplate?`. The tsunami layer is also initialised once and used throughout the lifetime of the application, since the data source (shape file) stays the same, and only the colouring of the shape file (the style) changes.

The first method involved here is `OnTsunamiReceived`, which handles the tsunami when it is received. It is in charge of finding the valid date time of the tsunami (which is as indicated on the telegram, or if not then `_tsunamiLifeTime` which is 2 days in the program), and setting the `CurrentTsunami` object and changing the style of the map.

The second method involved is `RemoveExpiredTsunamiEventHandler` which is subscribed to the one-second timer. It checks if the current tsunami has expiry time before the current time (as indicated by the time provider), and if so, set the current tsunami to null and set the layer style to null as well.

In terms of the models involved, `TsunamiDetailsTemplate` is a record class that holds the details of the tsunami, including its informational text, expiry time, update time, and maximum tsunami warning type of the tsunami warnings of shorelines issued (using the `max` method in `LINQ`, which is an `aggregate`). The tsunami warning type is an enumeration, containing the tsunami warning type level. JMA uses a code to describe the tsunami warning type, so a conversion method is provided as well.

WebSocket This part consists of two functionalities: handling the WebSocket data received event (passing it to the appropriate handler for EEW or for Tsunami), and a separate event indicating data received from WebSocket (requesting main window to switch view model to the real-time page) named `WebSocketDataReceived`. The former is achieved by subscribing to the event in the WebSocket client (which is done in the constructor), and the latter will be raised in the event handler. Notice that subscription to the event raised by this will be done in the window view model.

Kmoni Monitor The first functionality this part achieves is displaying the `legend` for the colours of the Kmoni layer. This consists of two parts: the text for the legend in string `DataLegendtext`, displaying the type and units of the data measured, and the values on the scales of the legend in a decimal array `DataLegendValues`. The latter involves `list manipulation` using `LINQ method select` to `map` normalised height to the scale of the data that is concerned.

The second functionality is to display the Kmoni data on the map. To improve performance, the same layer `_kmoniLayer` is used, and only the data within the layer will be updated to be reflected on the map. To prevent 404 requests, a time delay is in `_kmoniDelay` which is subtracted from the current time when request is sent to fetch the image and extract the points. This is done by `RefreshKmoniDataEventHandler`, which fetches the Kmoni data, and updates the layer if the fetch is successful. The `GetKmoniObservationPoints` is the method called which actually fetches the data and converts them to the feature to display on the map.

2.5.4 Design of DTOs (Record Classes)

2.5.4.1 Common (Reusable) DTOs

In principle, the DTOs should be contained in the namespace for the abstraction requiring the use of the DTO, such as the API calls, the telegrams, or the WebSockets, and exposed to the user where necessary. However, there are some DTOs to be reused, primarily because they are shared across the different parts of the DM-D.S.S. API, WebSocket and telegram services.

JSON Components DM-D.S.S. designed **JSON components** for the application [16], which improves the reusability of typed records over the application. These components include:

- The description of the earthquake; which further incorporates:
 - The hypocentre (including the depth and the coordinate);
 - The magnitude;
- The coordinate position of some property.

This makes the records highly reusable, and these will be reused for the API calls (since it is included in the earthquake list and event calls), and the telegrams as well.

Enumerations are used wherever possible. In this case, it means that strings which can only take finitely-many enumerable values will be modelled as an enumeration. In this component, it means that:

- The condition for the depth (very shallow 'ごく浅い', very deep '700km以上', unclear '不明') are modelled with `DepthCondition`;
- The condition for the magnitude (unknown 'M不明', very big 'M8を超える巨大地震') are modelled with `MagnitudeCondition`;
- The unit for the magnitude (JMA magnitude 'Mj', normal magnitude 'M') are modelled with `MagnitudeUnit`;
- The source for the information of the earthquake (the earthquake advisory centres available) are modelled with `Source`;
- The geodetic coordinate used (world geodetic coordinate '世界測地系', Japanese geodetic coordinate '日本測地系') are modelled with `Geodetic`.

Telegrams Several components of the telegram are shared between different parts of the DM-D.S.S. functionality. Those include:

- `XmlReport` which composes of `XmlHead`, `XmlControl`, and describes the key components of the XML telegram. This is used within the API call for the earthquake event, and the data response from the WebSocket.
- `SchemaVersionInformation`, which includes the type and the version for the JSON schema used. It is shared with the API component (because in each telegram retrieved for earthquake event the schema will be included), and for parsing the telegram.
- `TelegramStatus` and `TelegramType`, which describes the status and the type of the telegram. They are composed within `XmlHead`, and for `Head` in the telegram component as well (described later).

Note that `TelegramStatus` and `TelegramType` are modelled as enumerations, since they only take a finite number of values, although provided by string from the original API.

Enumerations There are also a couple of other shared enumerations within the DM-D.S.S. services. They are all modelled from strings which can only take finitely many values.

- **Classification**, which could be used to describe the classification for a contract that user subscribed to, or the classification of a telegram.
- **Intensity** for earthquake intensity, **LgCategory** for LPGM motion category, and **LgIntensity** for LPGM intensity;
- **FormatType** and **TestStatus** for describing the format of a particular telegram (JSON, XML, Alpha Numeric or Binary) and the test status (whether it is a test or not), both used for setting up the WebSocket and in the WebSocket service.

2.5.4.2 API Calls

Since all API Calls must share the response ID, response time, and response status properties, this could be modelled as an **abstract base class** (record class) for the API calls. This is the **ApiBase** class, on the top of the diagram, composing of an enumeration for the status of the class.

Inheriting from it are two types of responses: a **successful** response (which is abstract as well), and an **error** response (which is concrete). The error response composes the details of the error within as a DTO.

There are three responses further abstracted, each inheriting from the previous one:

- a **list** response, which contains a **ItemList** containing all the items in the **item** property in the serialised JSON;
- a **token** response, which contains a **NextToken** property for the next API call; and
- a **pooling** response, which contains the next pooling and next pooling interval properties for the next API call.

As described in the previous behaviour of the API responses:

- **WebSocketStart** and **GdEarthquakeEvent** response are success responses and compose of only the details of the response;
- **ContractList** and **EarthquakeParameter** contains a list of contracts/stations within the response. In the parameter response, in addition to this, it also contains the last changed time for the data, and the version information;
- **WebSocketList** contains a token but not a pooling token, and contains a list of WebSocket details; and finally
- **GdEarthquakeList** contains a list of past earthquake information, and is a pooling response.

This design is accurately reflected in Figure 2.5.18 which shows a clear use of **inheritance, composition and abstract classes**.

Most other parts of the DTOs for API Calls is just simply the use of composition as modelled by the DM-D.S.S. API documentation, and the use of enumerations to model 'finitely many values'.

2.5.4.3 Telegrams

Similar to the API calls, a base class **Head** which contains the head information of the telegram is modelled. And then, classes are inherited from this class to provide the body part of the telegram. The reason why **Head** is not abstract, is because telegrams would have to be deserialised to it first and find the JSON schema, and then re-deserialised to the specific type of the telegram.

This is reflected in Figure 2.5.19 which shows a clear use of **inheritance and composition**.

Otherwise, this is simply the use of composition as modelled by the DM-D.S.S. JSON Documentation [19].

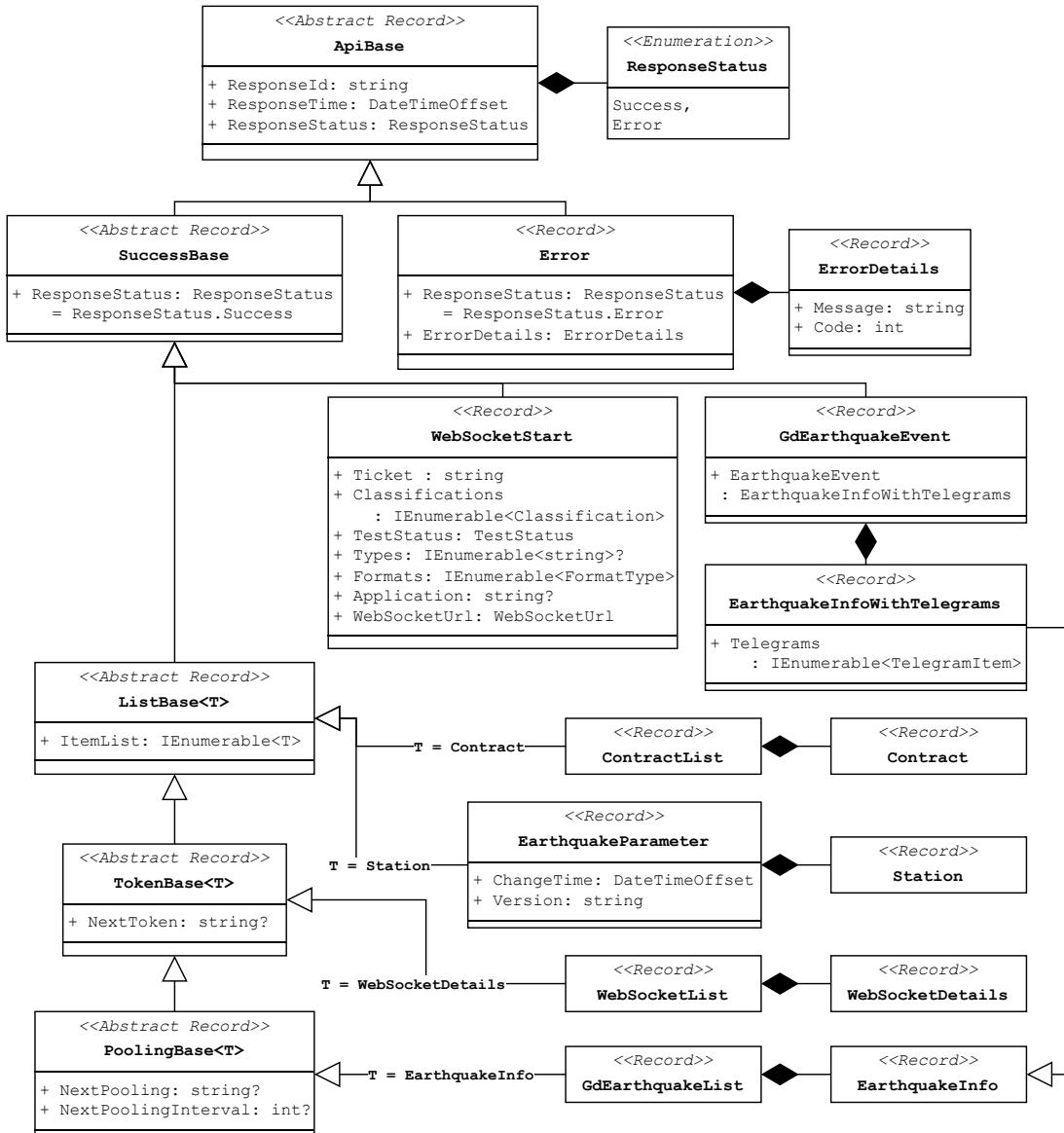


Figure 2.5.18: Class Diagram for DTOs for API Calls

Type	Minimum	Recommended
RAM	4 GB	8 GB
Storage Device	HDD	SSD
CPU	Intel Core 8th Gen (or equiv.)	Intel Core 10th Gen (or equiv.)
Display Resolution	720p	1080p
Display Size	9"	12"

Table 2.6.1: Hardware Specification for App

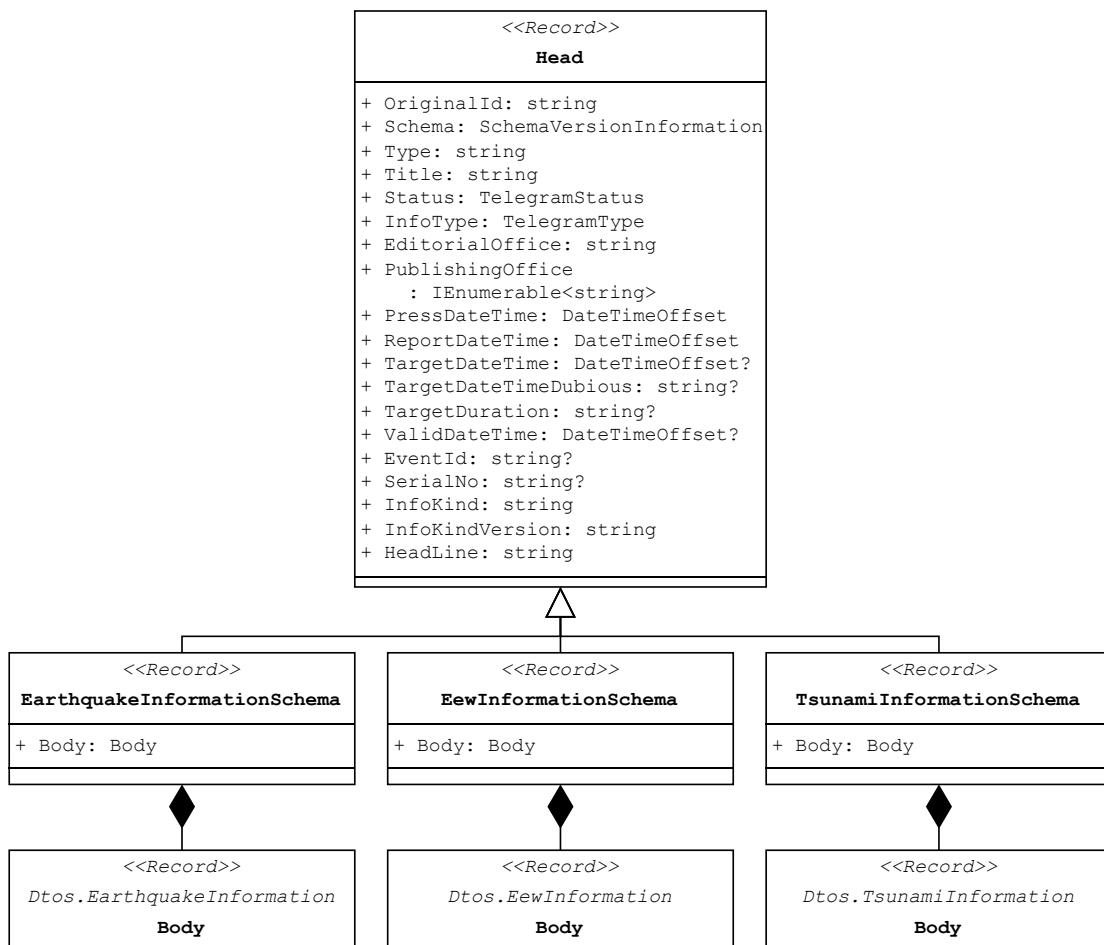


Figure 2.5.19: Class Diagram for DTOs for Telegrams

2.6 Hardware and Software Requirements

Table 2.6.1 outlines the expected basic hardware requirements for the application.

The amount of RAM is due to the amount of data that is processed (and considering other applications running as well). Storage space is not a substantial requirement of this application, while the CPU has to be of high standards to process all the data. To display the application properly (with appropriate size), a display of 1080p 12" is recommended.

It will be able to run on up-to-date Windows, macOS and Linux distributions (both x64 and ARM) due to the cross-platform nature of .NET, but out-of-date x86 platforms will not be explicitly supported.

The author uses a macOS 15 (beta) machine with 2.5K 13" display, Intel Core i5-1038NG7 (with Intel Iris Plus Graphics) and 16 GB of RAM (MacBook Pro 2020, 4 Thunderbolt Ports) and a Windows 11 (beta) machine with 2.5K 15" display, AMD Ryzen 7 5800H and 32 GB of RAM (with RTX 3070 for Laptop) (Legion R9000K 2021) to test the application, and with 2.5K 24" external display as well. An Ubuntu 22.04 virtual machine will be used to test the compatibility for Linux systems.

The application will be self-contained (i.e. comes with .NET runtime) to prevent the user from unnecessary technical complications.

The device needs to have stable connection to the internet using Wi-Fi/Ethernet/other means, to establish connection with the relevant APIs.

Chapter 3

Technical Implementation

Note that in this section, only key code segments are listed. Documentation comments, namespace declarations, using statements, and code that are not key functionalities are omitted for brevity (and this is why line numbers sometimes do skip). The full code is available in the appendix.

3.1 JMA Travel Time

```
5  public interface ITimeTable
6  {
7
13     (double pDistance, double sDistance) DistanceFromDepthTime(int depth, double
    ↵   timeSecond);
14 }
```

Listing 3.1.1: `ITimeTable` Interface
Use of Interface
Full Code at Listing A.2.4

```
6  internal record TimeTableEntry
7  {
8
11     public required int Radius { get; init; }
15     public required int Depth { get; init; }
19     public required IEnumerable<double> Times { get; init; }
20 }
```

Listing 3.1.2: `TimeTableEntry` DTO
User-defined DTO for Data
Full Code at Listing A.2.3

```
11 internal static partial class JmaTimeTableBuilder
12 {
13
22     public static ITimeTable FromFile(string fileName, ILogger<JmaTimeTable> logger)
23     {
24         string[] rows = File.ReadAllLines(fileName);
25
26         string? unmatchingRow = rows.FirstOrDefault(r => !Pattern().IsMatch(r));
27         return unmatchingRow is not null
28             ? throw new FormatException($"'{unmatchingRow}' does not fit format of time
    ↵   table")
29     }
```

```

29     : new JmaTimeTable(rows.Select(row =>
30         new TimeTableEntry()
31     {
32         Depth = int.Parse(row[22..25]),
33         Radius = int.Parse(row[27..32]),
34         Times = [double.Parse(row[2..10]),
35                   double.Parse(row[13..21])]
36     }), logger);
37 }
38 [GeneratedRegex(@"^P ((?=.{4}) *\d+)\.\d{3} S ((?=.{4}) *\d+)\.\d{3} ((?=.{3})
39   \w+\d+) ((?=.{5}) *\d+)$")]
40 private static partial Regex Pattern();
41 }
```

Listing 3.1.3: JmaTimeTableBuilder Factory Class
 Factory Pattern, Handling of External Data Source, Deserialisation
 Full Code at Listing A.2.6

```

9  internal sealed partial class JmaTimeTable : ITimeTable
10 {
11
12     private const int _rowsPerDepth = 236;
13
14     public (double pDistance, double sDistance) DistanceFromDepthTime(int depth,
15       double timeSecond)
16     {
17         ArgumentOutOfRangeException.ThrowIfNegative(timeSecond, nameof(timeSecond));
18         int startIndex = depth switch
19         {
20             >= 0 and <= 50 => depth / 2 * _rowsPerDepth,
21             >= 50 and <= 200 => (25 + ((depth - 50) / 5)) * _rowsPerDepth,
22             >= 200 and <= 700 => (55 + ((depth - 200) / 10)) * _rowsPerDepth,
23             _ => throw new ArgumentOutOfRangeException(nameof(depth)),
24         };
25         int endIndex = startIndex + _rowsPerDepth;
26         _logger.LookingLines(startIndex, endIndex);
27         return (SearchForDistance(startIndex, endIndex, timeSecond, 0),
28             SearchForDistance(startIndex, endIndex, timeSecond, 1));
29     }
30
31     private double SearchForDistance(int startIndex, int endIndex, double timeSecond,
32       int index)
33     {
34         int leftPoint;
35         if (timeSecond <= _timeTable.ElementAt(startIndex).Times.ElementAt(index))
36         {
37             _logger.BeforeStart(timeSecond);
38             return 0;
39         }
40         else if (timeSecond >= _timeTable.ElementAt(endIndex -
41             1).Times.ElementAt(index))
42         {
43             _logger.AfterEnd(timeSecond);
44             leftPoint = endIndex - 2;
45         }
46         else
47         {
48             for (leftPoint = startIndex; leftPoint < endIndex &&
49                 _timeTable.ElementAt(leftPoint).Times.ElementAt(index) <= timeSecond;
50                 ++leftPoint)
51             ;
52         }
53     }
54 }
```

```

75     {
76         ;
77     }
78
79     leftPoint -= 1;
80 }
81
82     double x1 = _timeTable.ElementAt(leftPoint).Times.ElementAt(index);
83     double y1 = _timeTable.ElementAt(leftPoint).Radius;
84     double x2 = _timeTable.ElementAt(leftPoint + 1).Times.ElementAt(index);
85     double y2 = _timeTable.ElementAt(leftPoint + 1).Radius;
86     _logger.LinearPolating(x1, y1, x2, y2);
87
88     return y1 + ((timeSecond - x1) * (y2 - y1) / (x2 - x1));
89 }
90 }
```

Listing 3.1.4: JmaTimeTable Service Class
 Implementation of Interface, Algorithm (Line 59 to 89), Logging
 Full Code at Listing A.2.7

```

7 public sealed record JmaTimeTableOptions
8 {
9
12     public required string FilePath { get; set; }
13 }
```

Listing 3.1.5: JmaTimeTableOptions Options Record
 Options Pattern
 Full Code at Listing A.2.1

```

12 public static class JmaTimeTableServiceCollectionExtensions
13 {
14
19     public static IServiceCollection AddJmaTimeTable(this IServiceCollection services)
20         => services.AddSingleton(sp
21             => JmaTimeTableBuilder.FromFile(
22                 sp.GetRequiredService<IOptions<JmaTimeTableOptions>>().Value.FilePath,
23                 sp.GetRequiredService<ILogger<JmaTimeTable>>()));
24 }
```

Listing 3.1.6: JmaTimeTableServiceCollectionExtensions DI Injection Extension
 Dependency Injection and Singleton Pattern
 Full Code at Listing A.2.2

3.2 Kyoshin Monitor

```

6 public interface IImageFetch
7 {
8
15     Task<byte[]?> GetByteArrayAsync(MeasurementType measurementType, SensorType
16         ↳ sensorType, DateTime dateTime);
16 }
```

Listing 3.2.1: IImageFetch Interface
Use of Interface
Full Code at Listing A.3.12

```

9  public interface IPointExtract
10 {
11
17      IEnumerable<(ObservationPoint point, SKColor colour)> ExtractColours(SKBitmap
18          ↪ bitmap, bool kikNetOnly = false);
19 }
```

Listing 3.2.2: IPointExtract Interface
Use of Interface
Full Code at Listing A.3.14

```

10 internal sealed class ImageFetch : IImageFetch
11 {
12
13     public ImageFetch	ILogger<ImageFetch> logger, string baseUri, string relativeUri)
14     {
15         _logger = logger;
16         _client = new()
17         {
18             BaseAddress = new(baseUri)
19         };
20         _baseUri = baseUri;
21         _relativeUri = relativeUri;
22         _logger.Instantiated();
23     }
24
25     public async Task<byte[]?> GetByteArrayAsync(MeasurementType measurementType,
26         ↪ SensorType sensorType, DateTime dateTime)
27     {
28         string relativeUri = string.Format(_relativeUri,
29             ↪ measurementType.ToUriString(), sensorType.ToUriString(), dateTime);
30
31         _logger.SendingRequest(_baseUri, relativeUri);
32         using HttpResponseMessage request = new( HttpMethod.Get, relativeUri );
33         try
34         {
35             using HttpResponseMessage response = await _client.SendAsync(request);
36
37             if (response.IsSuccessStatusCode)
38             {
39                 _logger.RequestSuccessful();
40                 byte[] imageBytes = await response.Content.ReadAsByteArrayAsync();
41                 return imageBytes;
42             }
43             else
44             {
45                 _logger.RequestUnsuccessful(_baseUri, relativeUri);
46                 return null;
47             }
48         }
49         catch (HttpRequestException)
50         {
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71 }
```

```

72     _logger.RequestUnsuccessful(_baseUri, relativeUri);
73     return null;
74   }
75 }
76 }
```

Listing 3.2.3: ImageFetch Service Class
 Implementation of Interface, Use of Parametrised Query, API Calls and Error Handling, Logging
 Full Code at Listing A.3.16

```

8  internal static class EnumToStringExtensions
9  {
10
11    public static string ToUriString(this MeasurementType kmoniDataType)
12      => kmoniDataType switch
13    {
14      MeasurementType.MeasuredIntensity
15        => "jma",
16      MeasurementType.PeakGroundAcceleration
17        => "acmap",
18      MeasurementType.PeakGroundVelocity
19        => "vcmap",
20      MeasurementType.PeakGroundDisplacement
21        => "dcmap",
22      MeasurementType.Response0125
23        => "rsp0125",
24      MeasurementType.Response0250
25        => "rsp0250",
26      MeasurementType.Response0500
27        => "rsp0500",
28      MeasurementType.Response1000
29        => "rsp1000",
30      MeasurementType.Response2000
31        => "rsp2000",
32      MeasurementType.Response4000
33        => "rsp4000",
34      -       => throw new UnreachableException(),
35    };
36
37    public static string ToUriString(this SensorType sensorType)
38      => sensorType switch
39    {
40      SensorType.Surface
41        => "s",
42      SensorType.Borehole
43        => "b",
44      -       => throw new UnreachableException(),
45    };
46 }
```

Listing 3.2.4: EnumToStringExtensions Extensions Class
 Helper for Parametrised Query
 Full Code at Listing A.3.4

```

9  internal sealed class PointExtract : IPointExtract
10 {
```

```

21     public IEnumerable<(ObservationPoint point, SKColor colour)>
22         ExtractColours(SKBitmap bitmap, bool kikNetOnly = false)
23         => _points
24             .Where(p => !p.IsSuspended && p.Point is not null)
25             .Where(p => !(kikNetOnly && p.Type != PointType.KiK))
26             .Select(p => (p, colour: bitmap.GetPixel(p.Point.X, p.Point.Y)))
27             .ToArray() // deferred execution throws System.ExecutionEngineException
28             .Where(pc => pc.colour.Alpha != 0);
}

```

Listing 3.2.5: PointExtract Service Class
Implementation of Interface, List Operations (LINQ)
Full Code at Listing A.3.17

```

13  public static class KmoniServiceCollectionExtensions
14  {
15
16      public static IServiceCollection AddKmoniHelpers(this IServiceCollection services)
17          => services
18              .AddSingleton<IImageFetch, ImageFetch>(static provider
19                  =>
20                  {
21                      KmoniHelperOptions options = provider.GetRequiredService<IOptions<
22                          <KmoniHelperOptions>>().Value;
23                      ILogger<ImageFetch> logger =
24                          provider.GetRequiredService<ILogger<ImageFetch>>();
25                      return new(logger, options.BaseUri, options.RelativeUri);
26                  })
27              .AddSingleton<IPointExtract, PointExtract>(static provider
28                  =>
29                  {
30                      KmoniHelperOptions options = provider.GetRequiredService<IOptions<
31                          <KmoniHelperOptions>>().Value;
32                      return new(JsonSerializer.Deserialize<IEnumerable<ObservationPoint>>(File.ReadAllText(options.FilePath)) ?? []);
33                  });
34  }
}

```

Listing 3.2.6: KmoniServiceCollectionExtensions DI Injection Extension
Dependency Injection and Singleton Pattern
Full Code at Listing A.3.1

```

8  public static class ColourConversionExtensions
9  {
10
11      public static double IntensityToHeight(this double intensity)
12          => (intensity + 3) / 10;
13
14      public static double PgaToHeight(this double pga)
15          => (Math.Log(pga, 10) + 2) / 5;
16
17      public static double PgvToHeight(this double pgv)
18          => (Math.Log(pgv, 10) + 3) / 5;
19
20      public static double PgdToHeight(this double pgd)
21          => (Math.Log(pgd, 10) + 4) / 5;
22
23      public static double HeightToIntensity(this double height)
24          => (10 * height) - 3;
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

```

```

50     public static double HeightToPga(this double height)
51         => Math.Pow(10, (5 * height) - 2);
52
53     public static double HeightToPgv(this double height)
54         => Math.Pow(10, (5 * height) - 3);
55
56     public static double HeightToPgd(this double height)
57         => Math.Pow(10, (5 * height) - 4);
58
59     public static double HeightToHue(this double height)
60         => height switch
61     {
62         <= 0.1 => (-150 * height) + 237,
63         >= 0.1 and <= 0.6 => (222 * (height - 0.3) * (height - 0.4) * (height -
64             → 0.6) / ((0.1 - 0.3) * (0.1 - 0.4) * (0.1 - 0.6)))
65             + (115 * (height - 0.1) * (height - 0.4) * (height - 0.6)
66                 → / ((0.3 - 0.1) * (0.3 - 0.4) * (0.3 - 0.6)))
67             + (79.5 * (height - 0.1) * (height - 0.3) * (height -
68                 → 0.6) / ((0.4 - 0.1) * (0.4 - 0.3) * (0.4 - 0.6)))
69             + (51 * (height - 0.1) * (height - 0.3) * (height - 0.4)
70                 → / ((0.6 - 0.1) * (0.6 - 0.3) * (0.6 - 0.4))),
71         >= 0.6 and <= 0.9 => (-170 * height) + 153,
72         _ => 0
73     };
74
75     public static double HeightToSaturation(this double height)
76         => height switch
77     {
78         <= 0.2 => 1,
79         >= 0.2 and <= 0.29 => (-2.611 * height) + 1.522,
80         >= 0.29 and <= 0.4 => (1.682 * height) + 0.277,
81         >= 0.4 and <= 0.5 => (0.5 * height) + 0.75,
82         _ => 1
83     };
84
85     public static double HeightToValue(this double height)
86         => height switch
87     {
88         <= 0.1 => (1.8 * height) + 0.8,
89         >= 0.1 and <= 0.172 => (-4.444 * height) + 1.424,
90         >= 0.172 and <= 0.2 => (5.714 * height) - 0.323,
91         >= 0.2 and <= 0.3 => (1.6 * height) + 0.5,
92         >= 0.3 and <= 0.4 => (0.2 * height) + 0.92,
93         >= 0.4 and <= 0.8 => 1,
94         >= 0.8 and <= 0.9 => (-0.3 * height) + 1.24,
95         _ => (-2.9 * height) + 3.58
96     };
97
98     public static SKColor HeightToColour(this double height)
99         => SKColor.FromHsv((float)height.HeightToHue(),
100            → (float)height.HeightToSaturation() * 100, (float)height.HeightToValue() *
101                → 100);
102
103     public static double ColourToHeight(this SKColor colour)
104     {
105         colour.ToHsv(out float h, out float _, out float v);
106
107         v /= 100; // Since SkiaSharp use percentage for v without the pecentage sign
108
109         if (h == 0) // Use v
110         {
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132

```

```

133         return (v - 3.58) / -2.9;
134     }
135     else // Use h
136     {
137         if (h >= 222)
138         {
139             return (h - 237) / -150;
140         }
141
142         if (h <= 51)
143         {
144             return (h - 153) / -170;
145         }
146
147         // Binary Search for Numerical Inverse Cubic
148
149         double left = 0.1;
150         double right = 0.6;
151         const double heightEpsilon = 0.01;
152         const double hueEpsilon = 0.5;
153         while (right - left >= heightEpsilon)
154         {
155             double mid = (right + left) / 2;
156             double calculatedHue = mid.HeightToHue();
157             if (Math.Abs(calculatedHue - h) <= hueEpsilon)
158             {
159                 return mid;
160             }
161
162             if (calculatedHue > h)
163             {
164                 left = mid;
165             }
166
167             if (calculatedHue < h)
168             {
169                 right = mid;
170             }
171         }
172
173         return (right + left) / 2;
174     }
175 }
176 }
```

Listing 3.2.7: ColourConversionExtensions Extensions Class
 Mathematical Algorithm, Binary Search (Line 147 to 173)
 Full Code at Listing A.3.2

3.3 DM-D.S.S. Authentication

```

8 internal interface IAuthenticator
9 {
10
11     Task<AuthenticationHeaderValue> GetAuthenticationHeaderAsync();
12 }
13 }
```

Listing 3.3.1: `IAuthenticator` Interface
 Use of Interface, Asynchronous Programming
 Full Code at Listing A.5.12

```

9  internal sealed class NullAuthenticator : IAuthenticator
10 {
11
13    public Task<AuthenticationHeaderValue> GetAuthenticationHeaderAsync()
14      => throw new NullAuthenticatiorException();
15
16    public override string? ToString()
17      => null;
18
19    private NullAuthenticator() { }
20
21    public static IAuthenticator Instance { get; } = new NullAuthenticator();
22
23 }
```

Listing 3.3.2: `NullAuthenticator` Service Class
 Implementation of Interface, Singleton Pattern, Custom Exceptions
 Full Code at Listing A.5.21

```

10 internal sealed class ApiKeyAuthenticator : IAuthenticator
11 {
12
13    public Task<AuthenticationHeaderValue> GetAuthenticationHeaderAsync()
14      => Task.FromResult(_header);
15
16    public override string ToString()
17      => _apiKey;
18
19    public ApiKeyAuthenticator(string apiKey)
20    {
21        if (!apiKey.StartsWith("AKe."))
22        {
23            throw new ArgumentException($"API Key does not have valid format:
24              ↪ {apiKey}", nameof(apiKey));
25        }
26
27        _apiKey = apiKey;
28        byte[] plainTextBytes = Encoding.UTF8.GetBytes($"{apiKey}:");
29        string val = Convert.ToBase64String(plainTextBytes);
30        _header = new("Basic", val);
31    }
32 }
```

Listing 3.3.3: `ApiKeyAuthenticator` Service Class
 Implementation of Interface, Singleton Pattern, Defensive Programming for Inputs
 Full Code at Listing A.5.24

```

13 internal sealed class OAuth2Authenticator : IAuthenticator
14 {
15
16    public OAuth2Authenticator(string clientId, string basePath, string host, string
17      ↪ refreshToken, string? accessToken, ILogger<OAuth2Authenticator> logger)
18    {
19        if (!refreshToken.StartsWith("ARh."))
20        {
21            throw new ArgumentException($"Refresh token must start with ARh.");
22        }
23
24        _clientId = clientId;
25        _basePath = basePath;
26        _host = host;
27        _refreshToken = refreshToken;
28        _accessToken = accessToken;
29        _logger = logger;
30    }
31 }
```

```
63     {
64         throw new ArgumentException($"{refreshToken} does not match format of
65             → refresh token.", nameof(refreshToken));
66     }
67
68     if (accessToken is not null && !accessToken.StartsWith("ATn."))
69     {
70         throw new ArgumentException($"{accessToken} does not match format of
71             → access token.", nameof(accessToken));
72     }
73
74     _clientId = clientId;
75     _host = host;
76     _base = new(basePath);
77     _httpClient = new()
78     {
79         BaseAddress = _base
80     };
81     _refreshToken = refreshToken;
82     _logger = logger;
83     _accessToken = accessToken ?? Task.Run(RenewAccessTokenAsync).Result;
84     _accessTokenExpiry = DateTimeOffset.Now + _accessTokenValidity;
85     _logger.Instantiated();
86 }
87
88 public async Task<AuthenticationHeaderValue> GetAuthenticationHeaderAsync()
89     => new("Bearer", await CheckAccessTokenAsync());
90
91 public override string ToString()
92     => _refreshToken;
93
94 private async Task<string> CheckAccessTokenAsync()
95     => _accessTokenExpiry > DateTimeOffset.Now
96         ? _accessToken
97         : await RenewAccessTokenAsync();
98
99 private async Task<string> RenewAccessTokenAsync()
100 {
101     if (_accessToken is not null)
102     {
103         try
104         {
105             await RevokeTokenAsync(_accessToken);
106         }
107         catch (Exception ex)
108         {
109             _logger.IgnoredException(ex.ToString());
110         }
111
112         _accessTokenExpiry = DateTimeOffset.MinValue;
113     }
114
115     _logger.RequestingNewAccessToken();
116     Dictionary<string, string> requestParams = new(){
117         { "client_id", _clientId },
118         { "grant_type", "refresh_token" },
119         { "refresh_token", _refreshToken }
120     };
121     HttpRequestMessage request = OAuth2SharedMethod.GeneratePostRequest("token",
122         → requestParams, _host);
```

```
132
133     HttpResponseMessage response = await _httpClient.SendAsync(request);
134
135     if (response.IsSuccessStatusCode)
136     {
137         string responseBody = await response.Content.ReadAsStringAsync();
138         try
139         {
140             TokenRefresh token =
141                 JsonSerializer.Deserialize<TokenRefresh>(responseBody)
142                 ?? throw new OAuthJsonException($"Cannot deserialise:
143                     {responseBody}");
144             _accessTokenExpiry = DateTimeOffset.Now + _accessTokenValidity;
145             _accessToken = token.AccessToken;
146             _logger.NewAccessTokenAcquired();
147             return _accessToken;
148         }
149         catch (OAuthJsonException)
150         {
151             _logger.IncorrectJsonFormat(responseBody);
152             throw;
153         }
154         catch (JsonException ex)
155         {
156             _logger.IncorrectJsonFormat(responseBody);
157             throw new OAuthJsonException($"Cannot deserialise: {responseBody}",
158                 ex);
159         }
160     }
161     else
162     {
163         string responseBody = await response.Content.ReadAsStringAsync();
164         try
165         {
166             Error error = JsonSerializer.Deserialize<Error>(responseBody)
167                 ?? throw new OAuthJsonException($"Cannot deserialise:
168                     {responseBody}");
169             _logger.TokenRevokeFailed(error.Description);
170             throw new OAuthErrorException($"'{error.Short} {error.Description}'");
171         }
172         catch (OAuthJsonException)
173         {
174             _logger.IncorrectJsonFormat(responseBody);
175             throw;
176         }
177         catch (JsonException ex)
178         {
179             _logger.IncorrectJsonFormat(responseBody);
180             throw new OAuthJsonException($"Cannot deserialise: {responseBody}",
181                 ex);
182         }
183     }
184
185     private async Task RevokeTokenAsync(string token)
186     {
187         Dictionary<string, string> requestParams = new(){
188             { "client_id", _clientId },
189             { "token", token }
190         };
191     }
192 }
```

```

194     _logger.RevokingToken(token);
195     HttpRequestMessage request = OAuth2SharedMethod.GeneratePostRequest("revoke",
196         → requestParams, _host);
197     HttpResponseMessage response = await _httpClient.SendAsync(request);
198     if (response.IsSuccessStatusCode)
199     {
200         _logger.TokenRevoked();
201     }
202     else
203     {
204         string responseBody = await response.Content.ReadAsStringAsync();
205         try
206         {
207             Error error = JsonSerializer.Deserialize<Error>(responseBody)
208                 ?? throw new OAuthJsonException($"Cannot deserialise:
209                     → {responseBody}");
210             _logger.TokenRevokeFailed(error.Description);
211             throw new OAuthErrorException($"'{error.Short} {error.Description}'");
212         }
213         catch (OAuthJsonException)
214         {
215             _logger.IncorrectJsonFormat(responseBody);
216             throw;
217         }
218         catch (JsonException ex)
219         {
220             _logger.IncorrectJsonFormat(responseBody);
221             throw new OAuthJsonException($"Cannot deserialise: {responseBody}",
222                 → ex);
223         }
224     }
225 }
226
227 public async Task RevokeTokens()
228 {
229     Task revokeAccessToken = RevokeTokenAsync(_accessToken);
230     Task revokeRefreshToken = RevokeTokenAsync(_refreshToken);
231     await Task.WhenAll(revokeAccessToken, revokeRefreshToken);
232 }
233
234 }
```

Listing 3.3.4: OAuth2Authenticator Service Class

Implementation of Interface, Defensive Programming in Constructor, Serialisation and Deserialisation, Parametrised API Calls, Exception Handling, Custom Exceptions
Full Code at Listing A.5.23

```

17 internal sealed class OAuth2Helper
18 {
19
105     // https://stackoverflow.com/a/150974/
110     private static int GetUnusedPort()
111     {
112         TcpListener listener = new(IPAddress.Loopback, 0);
113         listener.Start();
114         int port = ((IPEndPoint)listener.LocalEndpoint).Port;
115         listener.Stop();
116         return port;
117     }
118 }
```

```
147     private async Task<(<string> grantCode, <string> requestState)> GetGrantCodeAsync(Uri
148         ↪ redirectUri, Uri browserUri)
149     {
150         using HttpListener listener = new();
151         listener.Prefixes.Add(redirectUri.ToString());
152         listener.Start();
153
154         _logger.StartingBrowser(browserUri);
155         // https://stackoverflow.com/a/61035650/
156         _ = Process.Start(new ProcessStartInfo
157         {
158             FileName = browserUri.ToString(),
159             UseShellExecute = true
160         });
161
162         HttpListenerContext context = await listener.GetContextAsync();
163         Uri responseUri = context.Request.Url!;
164         NameValueCollection query = HttpUtility.ParseQueryString(responseUri.Query);
165         _logger.ResponseReceived(responseUri);
166
166         byte[] buffer = Encoding.UTF8.GetBytes(_webPageString);
167         context.Response.ContentLength64 = buffer.Length;
168         await context.Response.OutputStream.WriteAsync(buffer.AsMemory(0,
169             ↪ buffer.Length));
170         context.Response.OutputStream.Close();
171         listener.Stop();
172
172         if (query["error"] is not null)
173         {
174             _logger.ErrorMessageReceived(query["error"]!);
175             throw new OAuthErrorException(query["error"]!);
176         }
177         else
178         {
179             return (query["code"]!, query["state"]!);
180         }
181     }
182
204     public async Task<(<string> refreshToken, <string> accessToken)>
205         ↪ GetRefreshTokenAsync()
206     {
207         string state = RandomNumberGenerator.GetString(_allowedChars, _stateLength);
208         string codeVerifier = RandomNumberGenerator.GetString(_allowedChars,
209             ↪ _codeLength);
210
211         _logger.FindingUnusedPort();
212         int port = GetUnusedPort();
213         _logger.FoundUnusedPort(port);
214
215         Uri redirectUri = new UriBuilder(IPAddress.Loopback.ToString())
216         {
217             Port = port,
218             Path = _redirectPath
219         }.Uri;
220
221         Uri browserUri = new UriBuilder(new Uri(new Uri(_baseUri), "auth"))
222         {
222             Query = GenerateGrantCodeQueryParameters(redirectUri, state,
223                 ↪ codeVerifier).ToString()
224         }
225     }
226 }
```

```

223     .Uri;
224
225     (string grantCode, string requestState) = await
226     ↳ GetGrantCodeAsync(redirectUri, browserUri);
227
228     if (requestState != state)
229     {
230         _logger.StateDoesNotMatch();
231         throw new OAuthSecurityException("The states do not match.");
232     }
233
234     Dictionary<string, string> requestParams =
235     ↳ GenerateRefreshTokenRequestParameter(grantCode, redirectUri,
236     ↳ codeVerifier);
236     HttpRequestMessage request = OAuth2SharedMethod.GeneratePostRequest("token",
237     ↳ requestParams, _host);
237     HttpResponseMessage response = await _client.SendAsync(request);
238
239     string responseBody = await response.Content.ReadAsStringAsync();
240
241     try
242     {
243         if (response.IsSuccessStatusCode)
244         {
245             TokenRequest token =
246             ↳ JsonSerializer.Deserialize<TokenRequest>(responseBody)
247             ?? throw new OAuthJsonException($"Cannot deserialise:
248             ↳ {responseBody}.");
249
250             return (token.RefreshToken, token.AccessToken);
251         }
252         else
253         {
254             Error error = JsonSerializer.Deserialize<Error>(responseBody)
255             ?? throw new OAuthJsonException($"Cannot deserialise:
256             ↳ {responseBody}.");
257             _logger.ErrorMessageReceived(error.Description);
258             throw new OAuthErrorException($"'{error.Short} {error.Description}'");
259         }
260     }
261     catch (OAuthJsonException)
262     {
263         _logger.IncorrectJsonFormat(responseBody);
264         throw;
265     }
266     catch (JsonException ex)
267     {
268         _logger.IncorrectJsonFormat(responseBody);
269         throw new OAuthJsonException($"Cannot deserialise: {responseBody}", ex);
270     }
271 }

```

Listing 3.3.5: OAuth2Helper Service Class
 Parameterised API Calls, Deserialisation, Exception Handling, Custom Exceptions
 Full Code at Listing A.5.22

⁷ internal sealed record Error

```

8   {
12     [JsonPropertyName("error")]
13     public required string Short { get; init; }
17     [JsonPropertyName("error_description")]
18     public required string Description { get; init; }
19   }
8   internal record TokenRefresh
9   {
13     [JsonPropertyName("access_token")]
14     public required string AccessToken { get; init; }
18     [JsonPropertyName("token_type")]
19     public string TokenType { get; } = "Bearer";
23     [JsonPropertyName("expires_in")]
24     public int Expiry { get; } = 21600;
28     [JsonPropertyName("scope")]
29     public required string Scope { get; init; }
30 }
8   internal sealed record TokenRequest : TokenRefresh
9   {
13     [JsonPropertyName("refresh_token")]
14     public required string RefreshToken { get; init; }
15 }

```

Listing 3.3.6: Error, TokenRefresh, TokenRefresh DTO Records
 DTOs for API calls, Inheritance
 Full Code at ??

```

9   internal static class OAuth2Provider
10  {
18    public static async Task<IAuthenticator> GetOAuth2Authenticator(OAuth2Options
19      ↪ options, ILogger<OAuth2Helper> helperLogger, ILogger<OAuth2Authenticator>
20      ↪ logger)
21    {
22      (string refreshToken, string accessToken) = await new OAuth2Helper(
23        options.ClientId,
24        string.Join(' ', options.Scopes),
25        options.BaseUri,
26        options.Host,
27        options.RedirectPath,
28        options.WebPageString,
29        helperLogger)
30      .GetRefreshTokenAsync();
31      return new OAuth2Authenticator(
32        options.ClientId,
33        options.BaseUri,
34        options.Host,
35        refreshToken,
36        accessToken,
37        logger);
38    }

```

```

44     public static IAuthenticator GetOAuth2Authenticator(OAuth2Options options, string
45         ↪ refreshToken, ILogger<OAuth2Authenticator> logger)
46         => new OAuth2Authenticator(
47             options.ClientId,
48             options.BaseUri,
49             options.Host,
50             refreshToken,
51             null,
52             logger);

```

Listing 3.3.7: OAuth2Provider Factory Methods
 Factory Pattern (Methods), Options Pattern, OOP
 Full Code at Listing A.5.18

```

9  public interface IAuthenticationHelper
10 {
11
12     AuthenticationStatus AuthenticationStatus { get; }
13
14     event EventHandler<AuthenticationStatusChangedEventArgs>? StatusChanged;
15
16     Task<AuthenticationHeaderValue?> GetAuthenticationHeaderAsync();
17
18     Task InvalidAuthenticatorAsync(string message);
19
20     Task SetApiKeyAsync(string apiKey);
21
22     Task SetOAuthAsync();
23
24     Task UnsetAuthenticatorAsync();
25 }

```

Listing 3.3.8: IAuthenticationHelper Interface
 Use of Interface, Asynchronous Programming, Factory Class
 Full Code at Listing A.5.13

```

12 internal sealed class AuthenticationHelper : IAuthenticationHelper
13 {
14
15     public async Task<AuthenticationHeaderValue?> GetAuthenticationHeaderAsync()
16     {
17         if (AuthenticationStatus is AuthenticationStatus.Null)
18         {
19             return null;
20         }
21         else
22         {
23             try
24             {
25                 return await _authenticator.GetAuthenticationHeaderAsync();
26             }
27             catch (OAuthException ex)
28             {
29                 _logger.OAuthExceptionIgnored(ex.ToString());
30                 await UnsetAuthenticatorAsync();
31                 return null;
32             }
33         }
34     }

```

```
56     public override string? ToString()
57         => _authenticator.ToString();
58
59     internal static IAuthenticationHelper FromString(
60         string filePath,
61         string? str,
62         ILogger<AuthenticationHelper> logger,
63         ILogger<OAuth2Helper> helperLogger,
64         ILogger<OAuth2Authenticator> oAuthLogger,
65         OAuth2Options oAuth2Options)
66     {
67         IAuthenticator authenticator;
68         if (str is null)
69         {
70             authenticator = NullAuthenticator.Instance;
71         }
72         else if (str.StartsWith("AKe."))
73         {
74             authenticator = new ApiKeyAuthenticator(str);
75         }
76         else if (str.StartsWith("ARh."))
77         {
78             try
79             {
80                 authenticator = OAuth2Provider.GetOAuth2Authenticator(oAuth2Options,
81                 & str, oAuthLogger);
82             }
83             catch (OAuthException ex)
84             {
85                 logger OAuthExceptionIgnored(ex.ToString());
86                 authenticator = NullAuthenticator.Instance;
87             }
88             catch (Exception ex)
89             {
90                 logger OtherExceptionIgnored(ex.ToString());
91                 authenticator = NullAuthenticator.Instance;
92             }
93         }
94         else
95         {
96             authenticator = NullAuthenticator.Instance;
97         }
98
99         AuthenticationHelper authenticationHelper = new(
100             authenticator,
101             logger,
102             helperLogger,
103             oAuthLogger,
104             oAuth2Options);
105         authenticationHelper.StatusChanged += (sender, e) =>
106         {
107             string? content = e.Authentication;
108             try
109             {
110                 File.WriteAllText(filePath, content);
111                 logger.SucceededToWriteToFile(filePath, content);
112             }
113             catch (Exception)
114             {
115                 logger FailedToWriteToFile(filePath, content);
116             }
117         }
118     }
119 }
```

```
124     }
125     };
126     return authenticationHelper;
127 }

129     public AuthenticationStatus AuthenticationStatus
130         => _authenticator is NullAuthenticator
131             ? AuthenticationStatus.Null
132             : _authenticator is ApiKeyAuthenticator
133                 ? AuthenticationStatus.ApiKey
134                 : AuthenticationStatus.OAuth;

158     public async Task SetOAuthAsync()
159     {
160         await UnsetAuthenticatorAsync();
161         try
162         {
163             _authenticator = await
164                 OAuth2Provider.GetOAuth2Authenticator(_oAuth2Options, _helperLogger,
165                 _oAuthLogger);
166         }
167         catch (OAuthException ex)
168         {
169             _logger OAuthExceptionIgnored(ex.ToString());
170         }
171         catch (Exception ex)
172         {
173             _logger OtherExceptionIgnored(ex.ToString());
174         }
175         StatusChanged?.Invoke(this, new()
176         {
177             AuthenticationStatus = AuthenticationStatus,
178             Authentication = ToString()
179         });
180     }

181     public async Task SetApiKeyAsync(string apiKey)
182     {
183         await UnsetAuthenticatorAsync();
184         _authenticator = new ApiKeyAuthenticator(apiKey);
185         _logger.ChangedToApiKey();
186         StatusChanged?.Invoke(this, new()
187         {
188             AuthenticationStatus = AuthenticationStatus,
189             Authentication = ToString()
190         });
191     }

193     public async Task InvalidAuthenticatorAsync(string message)
194     {
195         _logger.InvalidAuthentication(message);
196         await UnsetAuthenticatorAsync();
197     }

199     public async Task UnsetAuthenticatorAsync()
200     {
201         _logger.Unsetting();
202         if (_authenticator is OAuth2Authenticator oAuth)
203         {
```

```

204     try
205     {
206         _logger.RevokingOAuth2Token();
207         await oAuth.RevokeTokens();
208     }
209     catch (OAuthException ex)
210     {
211         _logger OAuthExceptionIgnored(ex.ToString());
212     }
213     catch (Exception ex)
214     {
215         _logger OtherExceptionIgnored(ex.ToString());
216     }
217 }
218
219     _authenticator = NullAuthenticator.Instance;
220     _logger.Unset();
221     StatusChanged?.Invoke(this, new()
222     {
223         AuthenticationStatus = AuthenticationStatus,
224         Authentication = ToString()
225     });
226 }
227
228 public event EventHandler<AuthenticationChangedEventArgs>? StatusChanged;
229 }
```

Listing 3.3.9: AuthenticationHelper Service Class
 Implementation of Interface, Error Handling, Factory Method (Line 68 to 127), Events, Logging
 Full Code at Listing A.5.19

```

5  public enum AuthenticationStatus
6  {
7
8      Null = 0,
9
10     ApiKey = 1,
11
12     OAuth = 2
13 }
```

Listing 3.3.10: AuthenticationStatus Enumeration
 Abstraction
 Full Code at Listing A.5.11

```

8  public class AuthenticationChangedEventArgs : EventArgs
9  {
10
11     public required AuthenticationStatus AuthenticationStatus { get; init; }
12
13     public required string? Authentication { get; init; }
14 }
```

Listing 3.3.11: AuthenticationChangedEventArgs Event
 Custom Events
 Full Code at Listing A.5.14

```

5  public sealed record OAuth2Options
6  {
10     public required string WebPageString { get; init; }
14     public required string RedirectPath { get; init; }
18     public required string BaseUri { get; init; }
22     public required string Host { get; init; }
26     public required string ClientId { get; init; }
30     public required IEnumerable<string> Scopes { get; init; }
31 }

```

Listing 3.3.12: OAuth2Options Options Class
Options Pattern
Full Code at Listing A.5.1

```

11 public static class AuthenticationHelperServiceCollectionExtensions
12 {
19     public static IServiceCollection AddAuthenticator(this IServiceCollection
20             services, string filePath)
21             => services.AddSingleton(sp
22             =>
23             {
24                 string fileContent;
25                 try
26                 {
27                     fileContent = File.ReadAllText(filePath);
28                 }
29                 catch
30                 {
31                     fileContent = string.Empty;
32                 }
33
34                 return AuthenticationHelper.FromString(
35                     filePath,
36                     fileContent,
37                     sp.GetRequiredService<ILogger<AuthenticationHelper>>(),
38                     sp.GetRequiredService<ILogger<OAuth2Helper>>(),
39                     sp.GetRequiredService<ILogger<OAuth2Authenticator>>(),
40                     sp.GetRequiredService<IOptions<OAuth2Options>>().Value);
41             });
}

```

Listing 3.3.13: AuthenticationHelperServiceCollectionExtensions DI Injection Extension
Dependency Injection and Singleton Pattern, File IO, Error Handling
Full Code at Listing A.5.7

Chapter 4

Testing

Consider how you will test your project. You should devise a test strategy that encompasses a range of methods.

4.1 Test Strategy

- Unit testing (of individual functions)
- Integration testing (e.g. different modules/class files)
- Robustness (demonstrating defensive programming skills/exception handling)
- Requirements testing (against your initial requirements - a table with test number, description, test data, expected result, evidence (screenshot/video time link) would be suitable)
- Independent end user beta testing (this will assist with your evaluation)

4.2 Testing Video

- You can include a video to assist (but you will need to reference the time point at which relevant evidence appears)
- If you include a video you will need to have it publicly available.
- It is suggested that you include a QR code in your testing to give a link to the video (for the moderator) rather than just giving a long URL on its own.

4.3 System Tests (against original requirements' specification)

You need to give evidence in support of requirements that have been met e.g. reference to a relevant test/screenshot/relevant code.

Requirement №	Description	Success Criteria	Tests + Evidence

Table 4.3.1: Table of Tests

Chapter 5

Evaluation

5.1 Requirements Specification Evaluation

Personal evaluation

- Copy and paste your original requirements from your project analysis
- You need to review each requirement and comment objectively on whether it was *fully met/partially met/not met*.

Requirement №	Description	Success Criteria	Fully/Partial/Not met (Reflective Comment)

Table 5.1.1: Table of Evaluation

5.2 Independent End-User Feedback

End user/client evaluation

- there **must** be meaningful end user feedback
- You should hold a review meeting with your end user
- Write down any key feedback that they give you. E.g. Agreement that a particular requirement has been meet/comments as to aspects that they find suboptimal/comments as to additions they would like to see

Requirement №	Description	Acceptance Y/N	Additional Comments

Table 5.2.1: Table of Feedback

5.3 Improvements

You need to give consideration to a number of potential future improvements that could be made. They may arise from either your experience or from feedback given to you by your end user. Ideally at least one should be in response to end user feedback.

- Write a paragraph for each potential improvement/change
- The improvements/changes could result from additional functionality that has been identified as being beneficial or could be as a result of required efficiencies if some processes are clunky or require faster run-times
- You should then comment on how the proposed change could be implemented moving forward. i.e. what would need to be changed/developed and how? You are not expected to actually make any changes; just comment on the possibilities.

Appendix A

Code Listing

A.1 Polynomial Fit for Colour

```
1 # %% [markdown]
2 # # Polynomial fitting for  $f : [0, 1] \rightarrow \mathcal{C}$ 
3
4 # %%
5 from PIL import Image, PyAccess # for image reading
6 import matplotlib.pyplot as plt # to plot graphs
7 from scipy import optimize # to do fitting
8
9 plt.rcParams['text.usetex'] = True # LaTeX in plots
10 plt.rcParams['mathtext.fontset'] = 'stix'
11 plt.rcParams['font.family'] = 'STIXGeneral'
12
13 # %% [markdown]
14 # ## Plot  $C = (H, S, V)$ 
15
16 # %%
17 orig_img_path: str = 'jma-scale.png'
18 orig_img: Image.Image = Image.open(orig_img_path) # read original image
19
20 orig_img_hsv: Image.Image = orig_img.convert('HSV') # convert to HSV colour format
21
22 img_width: int
23 img_height: int
24 img_width, img_height = orig_img.size # get size of image
25
26 img_width, img_height
27
28 # %%
29 # read the (H, S, V) in a fixed column
30
31 column: int = 5
32 hsv_list: list[tuple[int, int, int]] = [(orig_img_hsv.getpixel((column, y))) for y in
   range(img_height)] # type: ignore
33
34 hsv_list
35
36 # %% [markdown]
37 # ### Plot  $f_H, f_S, f_V$  against  $r$ 
38
39 # %%
40 r_list: list[int] = list(range(img_height))
41
```

```
42 # split up list into separate lists
43
44 hsv_h_list: list[int] = [c[0] for c in hsv_list]
45 hsv_s_list: list[int] = [c[1] for c in hsv_list]
46 hsv_v_list: list[int] = [c[2] for c in hsv_list]
47
48 # %%
49 # setup options for plotting graphs
50
51 marker_size: int = 15
52 marker_char: str = 'x'
53 line_width: float = 0.3
54
55 fig_size: tuple[int, int] = (10, 5)
56
57 h_colour: str = 'turquoise'
58 s_colour: str = 'gold'
59 v_colour: str = 'gray'
60
61 h_label: str = 'Hue $H$'
62 s_label: str = 'Saturation $S$'
63 v_label: str = 'Value $V$'
64
65 # %%
66 def init_plt(xlabel: str, ylabel: str, title: str) -> None:
67     plt.figure(figsize = fig_size)
68     plt.xlabel(xlabel)
69     plt.ylabel(ylabel)
70     plt.title(title)
71     plt.grid(True)
72     plt.tight_layout()
73
74 # %%
75 # plot graph of (H, S, V) against r
76
77 init_plt('Row Number $r$', '$(H, S, V)$ Values', f'$(H, S, V)$ Values against Row
    Number $r$ along column {column}')
78
79 plt.scatter(r_list, hsv_h_list, label=h_label, color=h_colour, s=marker_size,
    marker=marker_char, linewidths=line_width)
80 plt.scatter(r_list, hsv_s_list, label=s_label, color=s_colour, s=marker_size,
    marker=marker_char, linewidths=line_width)
81 plt.scatter(r_list, hsv_v_list, label=v_label, color=v_colour, s=marker_size,
    marker=marker_char, linewidths=line_width)
82
83 plt.legend()
84
85 plt.savefig('hsv-against-row.png')
86
87 # %%
88 plt.close()
89
90 # %% [markdown]
91 # ### Plot  $f_H, f_S, f_V$  against  $h$ 
92
93 # %%
94 # setup the initial values for  $h$  and  $r$ 
95
96 h_0 = 18
```

```
97 h_1 = 296 + 1
98 r_len = h_1 - h_0
99
100 def r_to_h(r: int) -> float:
101     return 1 - ((r - h_0) / (r_len - 1))
102
103 h_list: list[float] = [r_to_h(r) for r in r_list] # convert list of r to list of h
104
105 h_r_list: list[float] = h_list[h_0:h_1] # extract the values of h within range
106
107 hsv_h_r_list: list[int] = hsv_h_list[h_0:h_1] # extract the values of (H, S, V) within
108     ↪ range
109 hsv_s_r_list: list[int] = hsv_s_list[h_0:h_1]
110 hsv_v_r_list: list[int] = hsv_v_list[h_0:h_1]
111
112 h_r_list
113 # %%
114 # plot graph of (H, S, V) against h
115
116 init_plt('$h$', '$(H, S, V)$ Values', f'$(H, S, V)$ Values against $h$ along column
117     ↪ {column}')
118 plt.scatter(h_r_list, hsv_h_r_list, label=h_label, color=h_colour, s=marker_size,
119     ↪ marker=marker_char, linewidth=line_width)
120 plt.scatter(h_r_list, hsv_s_r_list, label=s_label, color=s_colour, s=marker_size,
121     ↪ marker=marker_char, linewidth=line_width)
122 plt.scatter(h_r_list, hsv_v_r_list, label=v_label, color=v_colour, s=marker_size,
123     ↪ marker=marker_char, linewidth=line_width)
124
125 plt.legend()
126
127 plt.savefig('hsv-against-h.png')
128
129 # %% [markdown]
130 # ## Regression
131
132 # %% [markdown]
133 # ### Prepare Lists to do Regression
134
135 # %%
136 # normalise values of h, s, v to our desired values
137
138 hsv_h_n_list: list[float] = [hsv_h_r_list[i] / 255 * 360 for i in range(279)]
139 hsv_s_n_list: list[float] = [hsv_s_r_list[i] / 255 * 1 for i in range(279)]
140 hsv_v_n_list: list[float] = [hsv_v_r_list[i] / 255 * 1 for i in range(279)]
141
142 # %% [markdown]
143 # ### Regression on f_H
144
145 # %%
146 # define f_H with unknown parameters y_1, y_2 which operates on a list
147
148 def f_h_params(xl: list[float], y_1: float, y_2: float) -> list[float]:
149     yl: list[float] = [0 for _ in xl]
150     for i, x in enumerate(xl):
```

```

151     if 0 <= x <= 0.1:
152         yl[i] = -150 * x + 237
153     elif 0.1 <= x <= 0.6:
154         yl[i] = (
155             ((222 * (x - 0.3) * (x - 0.4) * (x - 0.6)) / ((0.1 - 0.3) * (0.1 -
156             ↪ 0.4) * (0.1 - 0.6))) +
157             ((y_1 * (x - 0.1) * (x - 0.4) * (x - 0.6)) / ((0.3 - 0.1) * (0.3 -
158             ↪ 0.4) * (0.3 - 0.6))) +
159             ((y_2 * (x - 0.1) * (x - 0.3) * (x - 0.6)) / ((0.4 - 0.1) * (0.4 -
160             ↪ 0.3) * (0.4 - 0.6))) +
161             ((51 * (x - 0.1) * (x - 0.3) * (x - 0.4)) / ((0.6 - 0.1) * (0.6 -
162             ↪ 0.3) * (0.6 - 0.4)))
163         )
164     elif 0.6 <= x <= 0.9:
165         yl[i] = -170 * x + 153
166     elif 0.9 <= x <= 1:
167         yl[i] = 0
168     else:
169         yl[i] = 0
170
171     return yl
172
173 # %%
174 h_init: list[int] = [0, 0] # initial guess to indicate number of parameters
175
176 h_params: list[int]
177
178 h_params, _ = optimize.curve_fit(f_h_params, h_r_list, hsv_h_n_list, p0=h_init) #
179   ↪ optimised parameters
180
181 h_p_list: list[float] = [i / 1000 for i in range(0, 1001, 1)] # list to make sure the
182   ↪ plotted graph looks quite smooth
183 hsv_h_fitted_list: list[float] = f_h(h_p_list) # fitted line
184
185 # %%
186 line_width = 1.25 # make the markers look a bit thicker
187
188 # %%
189 # plot result of fit
190
191 init_plt('$h$', h_label, '$H$ against $h$')
192 plt.scatter(h_r_list, hsv_h_n_list, label=h_label, color=h_colour, s=marker_size,
193   ↪ marker=marker_char, linewidth=line_width)
194 plt.plot(h_p_list, hsv_h_fitted_list, label='Fit')
195
196 plt.legend()
197 plt.savefig('h-against-h.png')
198
199 # %%
200 plt.close()
201 # %% [markdown]
```

```

202 # ### Regression on  $f_S$ 
203
204 # %%
205 # define  $f_S$ 
206
207 def f_s(xl: list[float]) -> list[float]:
208     yl: list[float] = [0 for _ in xl]
209     for i, x in enumerate(xl):
210         if 0 <= x <= 0.2:
211             yl[i] = 1
212         elif 0.2 <= x <= 0.29:
213             yl[i] = -2.611 * x + 1.522
214         elif 0.29 <= x <= 0.4:
215             yl[i] = 1.682 * x + 0.277
216         elif 0.4 <= x <= 0.5:
217             yl[i] = 0.5 * x + 0.75
218         elif 0.5 <= x <= 1:
219             yl[i] = 1
220         else:
221             yl[i] = 0
222     return yl
223
224 hsv_s_fitted_list: list[float] = f_s(h_p_list) # fitted line
225
226 # %%
227 # plot result of fit
228
229 init_plt('$h$', s_label, '$S$ against $h$')
230
231 plt.scatter(h_r_list, hsv_s_n_list, label=s_label, color=s_colour, s=marker_size,
232             marker=marker_char, linewidth=line_width)
233 plt.plot(h_p_list, hsv_s_fitted_list, label='Fit')
234
235 plt.legend()
236 plt.savefig('s-against-h.png')
237
238 # %% [markdown]
239 # ### Regression on  $f_V$ 
240
241 # %%
242 # define  $f_V$ 
243
244 def f_v(xl: list[float]) -> list[float]:
245     yl: list[float] = [0 for _ in xl]
246     for i, x in enumerate(xl):
247         if 0 <= x <= 0.1:
248             yl[i] = 1.8 * x + 0.8
249         elif 0.1 <= x <= 0.172:
250             yl[i] = -4.444 * x + 1.424
251         elif 0.172 <= x <= 0.2:
252             yl[i] = 5.714 * x - 0.323
253         elif 0.2 <= x <= 0.3:
254             yl[i] = 1.6 * x + 0.5
255         elif 0.3 <= x <= 0.4:
256             yl[i] = 0.2 * x + 0.92
257         elif 0.4 <= x <= 0.8:
258             yl[i] = 1
259         elif 0.8 <= x <= 0.9:
260             yl[i] = -0.3 * x + 1.24

```

```

260     elif 0.9 <= x <= 1:
261         yl[i] = -2.9 * x + 3.58
262     else:
263         yl[i] = 0
264     return yl
265
266 hsv_v_fitted_list: list[float] = f_v(h_p_list) # fitted line
267
268 # %%
269 # plot result of fit
270
271 init_plt('$h$', v_label, '$V$ against $h$')
272
273 plt.scatter(h_r_list[1:r_len - 1], hsv_v_n_list[1:r_len - 1], label=v_label,
274             color=v_colour, s=marker_size, marker=marker_char, linewidth=line_width) # the
275             # first and last points are excluded from the plot
276 plt.plot(h_p_list, hsv_v_fitted_list, label='Fit')
277
278 plt.legend()
279 plt.savefig('v-against-h.png')
280
281 # %% [markdown]
282 # ## Back-Generate the image
283
284 # %%
285 # setup an HSV formatted image and get its pixels
286
287 image_generated: Image.Image = Image.new('HSV', (img_width, img_height))
288 generated_pixels: PyAccess.PyAccess = image_generated.load() # type: ignore
289
290 # returns a list of tuples of (H, S, V) for a column
291
292 def f(yl: list[int]) -> list[tuple[int, int, int]]:
293     hl: list[float] = [r_to_h(y) for y in yl]
294
295     hsv_h: list[float] = f_h(hl)
296     hsv_s: list[float] = f_s(hl)
297     hsv_v: list[float] = f_v(hl)
298
299     hsv_h_i: list[int] = [int(h / 360 * 255) for h in hsv_h]
300     hsv_s_i: list[int] = [int(s * 255) for s in hsv_s]
301     hsv_v_i: list[int] = [int(v * 255) for v in hsv_v]
302
303     return [(hsv_h_i[i], hsv_s_i[i], hsv_v_i[i]) for i in yl]
304
305 my_pixels: list[list[tuple[int, int, int]]] = [f(list(range(img_height))) for _ in
306     range(img_width)]
307
308 # copy this into the generated_pixels reference
309
310 for x in range(img_width):
311     for y in range(img_height):
312         generated_pixels[x, y] = my_pixels[x][y]
313
314 # convert to RGB and output
315
316 image_generated_rgb: Image.Image = image_generated.convert('RGB')
317 image_generated_rgb.save('generated-colour.png')
318

```

Listing A.1.1: Code for Polynomial Fit of Colour

A.2 JMA Time Table

```

1  using EasonEetwViewer.JmaTravelTime.Services;
2
3  namespace EasonEetwViewer.JmaTravelTime.Options;
4  ///<summary>
5  ///<summary>
6  ///</summary>
7  public sealed record JmaTimeTableOptions
8  {
9      ///<summary>
10     ///<summary>
11     ///</summary>
12     public required string FilePath { get; set; }
13 }
```

Listing A.2.1: EasonEetwViewer.JmaTravelTime/Options/JmaTimeTableOptions.cs

```

1  using EasonEetwViewer.JmaTravelTime.Abstractions;
2  using EasonEetwViewer.JmaTravelTime.Options;
3  using EasonEetwViewer.JmaTravelTime.Services;
4  using Microsoft.Extensions.DependencyInjection;
5  using Microsoft.Extensions.Logging;
6  using Microsoft.Extensions.Options;
7
8  namespace EasonEetwViewer.JmaTravelTime.Extensions;
9  ///<summary>
10 // Provides extensions to inject service of <see cref="ITimeTable"/> with JMA
11 // → implementation to the service collection.
12 ///</summary>
13 public static class JmaTimeTableServiceCollectionExtensions
14 {
15     ///<summary>
16     ///<summary>
17     ///<param name="services">The <see cref="IServiceCollection"/> where the services
18     ///<param name="services">The <see cref="IServiceCollection"/> where the services
19     public static IServiceCollection AddJmaTimeTable(this IServiceCollection services)
20         => services.AddSingleton(sp
21             => JmaTimeTableBuilder.FromFile(
22                 sp.GetRequiredService<IOptions<JmaTimeTableOptions>>().Value.FilePath,
23                 sp.GetRequiredService<ILogger<JmaTimeTable>>()));
24 }
```

Listing A.2.2:

EasonEetwViewer.JmaTravelTime/Extensions/JmaTimeTableServiceCollectionExtensions.cs

```

1  namespace EasonEetwViewer.JmaTravelTime.Dtos;
2
3  /// <summary>
4  /// Represents a row in the time table.
5  /// </summary>
6  internal record TimeTableEntry
7  {
8      /// <summary>
9      /// The distance that the waves travelled.
10     /// </summary>
11     public required int Radius { get; init; }
12     /// <summary>
13     /// The depth of the hypocentre.
14     /// </summary>
15     public required int Depth { get; init; }
16     /// <summary>
17     /// An array with lenght 2 of times, the time needed for P-Wave and S-Wave to
18     /// travel to such position.
19     /// </summary>
20     public required IEnumerable<double> Times { get; init; }
}

```

Listing A.2.3: EasonEetwViewer.JmaTravelTime/Dtos/TimeTableEntry.cs

```

1  namespace EasonEetwViewer.JmaTravelTime.Abstractions;
2  /// <summary>
3  /// Represents a time table for seismic waves travel time.
4  /// </summary>
5  public interface ITimeTable
{
    /// <summary>
    /// Gives the distance that seismic waves travel from the depth of the hypocentre
    /// and the time elapsed.
    /// </summary>
    /// <param name="depth">The depth of the hypocentre</param>
    /// <param name="timeSecond">The time elapsed.</param>
    /// <returns>A pair of <see cref="double"/>, representing the distance travelled
    /// by the P-wave and S-wave respectively.</returns>
    (double pDistance, double sDistance) DistanceFromDepthTime(int depth, double
    timeSecond);
}

```

Listing A.2.4: EasonEetwViewer.JmaTravelTime/Abstractions/ITimeTable.cs

```

1  using EasonEetwViewer.JmaTravelTime.Services;
2  using Microsoft.Extensions.Logging;
3
4  namespace EasonEetwViewer.JmaTravelTime;
5  /// <summary>
6  /// Represents logs used in <see cref="JmaTimeTable"/>.
7  /// </summary>
8  internal static partial class JmaTimeTableLogs
{
    /// <summary>
    /// Log when instantiated.
    /// </summary>

```

```
13     /// <param name="logger">The logger to be used.</param>
14     [LoggerMessage(
15         EventId = 0,
16         EventName = nameof(Instantiated),
17         Level = LogLevel.Information,
18         Message = "Instantiated.")]
19     public static partial void Instantiated(
20         this ILogger<JmaTimeTable> logger);
21     /// <summary>
22     /// Log when iterating between two indices.
23     /// </summary>
24     /// <param name="logger">The logger to be used.</param>
25     /// <param name="startIndex">The start index.</param>
26     /// <param name="endIndex">The end index.</param>
27     [LoggerMessage(
28         EventId = 1,
29         EventName = nameof(LookingLines),
30         Level = LogLevel.Trace,
31         Message = "Searching for time between `{{StartIndex}}` and `{{EndIndex}}`.")]
32     public static partial void LookingLines(
33         this ILogger<JmaTimeTable> logger, int startIndex, int endIndex);
34     /// <summary>
35     /// Log when the time was shorter than the data.
36     /// </summary>
37     /// <param name="logger">The logger to be used.</param>
38     /// <param name="time">The time elapsed.</param>
39     [LoggerMessage(
40         EventId = 2,
41         EventName = nameof(BeforeStart),
42         Level = LogLevel.Debug,
43         Message = "The time `{{Time}}` is before the data in first line.")]
44     public static partial void BeforeStart(
45         this ILogger<JmaTimeTable> logger, double time);
46     /// <summary>
47     /// Log when the time was longer than the data.
48     /// </summary>
49     /// <param name="logger">The logger to be used.</param>
50     /// <param name="time">The time elapsed.</param>
51     [LoggerMessage(
52         EventId = 3,
53         EventName = nameof(AfterEnd),
54         Level = LogLevel.Debug,
55         Message = "The time `{{Time}}` is after the data in final line.")]
56     public static partial void AfterEnd(
57         this ILogger<JmaTimeTable> logger, double time);
58     /// <summary>
59     /// Log the linear polation points.
60     /// </summary>
61     /// <param name="logger">The logger to be used.</param>
62     /// <param name="x1">The <c>x</c>-coordinate of the first point used.</param>
63     /// <param name="y1">The <c>y</c>-coordinate of the first point used.</param>
64     /// <param name="x2">The <c>x</c>-coordinate of the second point used.</param>
65     /// <param name="y2">The <c>y</c>-coordinate of the second point used.</param>
66     [LoggerMessage(
67         EventId = 4,
68         EventName = nameof(LinearPolating),
69         Level = LogLevel.Debug,
70         Message = "Linear polating using `{{X1}}, {{Y1}}` and `{{X2}}, {{Y2}}`.")]
71     public static partial void LinearPolating()
```

```

72     this ILogger<JmaTimeTable> logger, double x1, double y1, double x2, double
    ↵   y2);
73 }
74 }
```

Listing A.2.5: EasonEetwViewer.JmaTravelTime/Logging/JmaTimeTableLogs.cs

```

1  using System.Text.RegularExpressions;
2  using EasonEetwViewer.JmaTravelTime.Abstractions;
3  using EasonEetwViewer.JmaTravelTime.Dtos;
4  using EasonEetwViewer.JmaTravelTime.Services;
5  using Microsoft.Extensions.Logging;
6
7  namespace EasonEetwViewer.JmaTravelTime;
8  ///<summary>
9  /// Provides static methods to create a JMA TimeTable from.
10 ///</summary>
11 internal static partial class JmaTimeTableBuilder
12 {
13
14     ///<summary>
15     /// Create a new instance of <see cref="JmaTimeTable"/> from a file.
16     ///</summary>
17     ///<param name="fileName">The file that the time table is stored in.</param>
18     ///<param name="logger">The logger to be used.</param>
19     ///<returns>The instance of <see cref="JmaTimeTable"/> created.</returns>
20     ///<exception cref="FormatException">When a line does not have the correct
    ↵   format.</exception>
21     ///<remarks>See <see href="https://www.data.jma.go.jp/egev/data/bulletin/catalog_
    ↵   /appendix/trttime/tttfmt_j.html">JMA Webpage</see> for format
    ↵   definition.</remarks>
22     public static ITimeTable FromFile(string fileName, ILogger<JmaTimeTable> logger)
23     {
24         string[] rows = File.ReadAllLines(fileName);
25
26         string? unmatchingRow = rows.FirstOrDefault(r => !Pattern().IsMatch(r));
27         return unmatchingRow is not null
28             ? throw new FormatException($"'{unmatchingRow}' does not fit format of time
    ↵   table")
29             : new JmaTimeTable(rows.Select(row =>
30                 new TimeTableEntry()
31                 {
32                     Depth = int.Parse(row[22..25]),
33                     Radius = int.Parse(row[27..32]),
34                     Times = [double.Parse(row[2..10]),
    ↵           double.Parse(row[13..21])]
35                 }), logger);
36     }
37     [GeneratedRegex(@^P ((?=.{4}) *\d+)\.\d{3} S ((?=.{4}) *\d+)\.\d{3} ((?=.{3})
    ↵   *\d+) ((?=.{5}) *\d+$")]
38     private static partial Regex Pattern();
39 }
40 }
```

Listing A.2.6: EasonEetwViewer.JmaTravelTime/Services/JmaTimeTableBuilder.cs

```

1  using EasonEetwViewer.JmaTravelTime.Abstractions;
2  using EasonEetwViewer.JmaTravelTime.Dtos;
```

```

3  using Microsoft.Extensions.Logging;
4
5  namespace EasonEtwViewer.JmaTravelTime.Services;
6  ///<summary>
7  /// Default implementation for <see cref="ITimeTable"/> for JMA Time Tables. See <see
8  <see href="https://www.data.jma.go.jp/eqev/data/bulletin/catalog/appendix/trtime/trt_j_
9  .html">JMA Webpage Files</see>.
10 ///</summary>
11 internal sealed partial class JmaTimeTable : ITimeTable
12 {
13     ///<summary>
14     /// The collection of <see cref="TimeTableEntry"/> for the timetable.
15     ///</summary>
16     private readonly IEnumerable<TimeTableEntry> _timeTable;
17     ///<summary>
18     /// The logger to be used.
19     ///</summary>
20     private readonly ILogger<JmaTimeTable> _logger;
21     ///<summary>
22     /// Creates the instance of the class with the specified timetable.
23     ///</summary>
24     ///<param name="timeTable">The collection of <see cref="TimeTableEntry"/> for the
25     <param href="https://www.data.jma.go.jp/eqev/data/bulletin/catalog/appendix/trtime/trt_j_
26     .html">timetable.</param>
27     ///<param name="logger">The logger to be used.</param>
28     public JmaTimeTable(IEnumerable<TimeTableEntry> timeTable, ILogger<JmaTimeTable>
29     <param href="https://www.data.jma.go.jp/eqev/data/bulletin/catalog/appendix/trtime/trt_j_
30     .html">logger)
31     {
32         _timeTable = timeTable;
33         _logger = logger;
34         logger.Instantiated();
35     }
36     ///<summary>
37     /// The number of rows for each depth in the time table.
38     ///</summary>
39     private const int _rowsPerDepth = 236;
40     ///<inheritdoc/>
41     ///<exception cref="ArgumentOutOfRangeException">When the depth is not between
42     <exception href="https://www.data.jma.go.jp/eqev/data/bulletin/catalog/appendix/trtime/trt_j_
43     .html">the range of 0 to 700, or when the time second is negative.</exception>
44     public (double pDistance, double sDistance) DistanceFromDepthTime(int depth,
45     <param href="https://www.data.jma.go.jp/eqev/data/bulletin/catalog/appendix/trtime/trt_j_
46     .html">double timeSecond)
47     {
48         ArgumentOutOfRangeException.ThrowIfNegative(timeSecond, nameof(timeSecond));
49         int startIndex = depth switch
50         {
51             >= 0 and <= 50 => depth / 2 * _rowsPerDepth,
52             >= 50 and <= 200 => (25 + ((depth - 50) / 5)) * _rowsPerDepth,
53             >= 200 and <= 700 => (55 + ((depth - 200) / 10)) * _rowsPerDepth,
54             _ => throw new ArgumentOutOfRangeException(nameof(depth)),
55         };
56         int endIndex = startIndex + _rowsPerDepth;
57         _logger.LookingLines(startIndex, endIndex);
58         return (SearchForDistance(startIndex, endIndex, timeSecond, 0),
59             SearchForDistance(startIndex, endIndex, timeSecond, 1));
60     }
61     ///<summary>
62     /// Find the distance that is possible to travel within the given range, for the
63     <param href="https://www.data.jma.go.jp/eqev/data/bulletin/catalog/appendix/trtime/trt_j_
64     .html">given index of the distances array.
65     ///</summary>

```

```

54     /// <param name="startIndex">The start index to look at.</param>
55     /// <param name="endIndex">The end index to look at.</param>
56     /// <param name="timeSecond">The time elapsed.</param>
57     /// <param name="index">The element to use to refer to for the distance.</param>
58     /// <returns>The distance that the specified wave has travelled.</returns>
59     private double SearchForDistance(int startIndex, int endIndex, double timeSecond,
60                                     int index)
61     {
62         int leftPoint;
63         if (timeSecond <= _timeTable.ElementAt(startIndex).Times.ElementAt(index))
64         {
65             _logger.BeforeStart(timeSecond);
66             return 0;
67         }
68         else if (timeSecond >= _timeTable.ElementAt(endIndex -
69             1).Times.ElementAt(index))
70         {
71             _logger.AfterEnd(timeSecond);
72             leftPoint = endIndex - 2;
73         }
74         else
75         {
76             for (leftPoint = startIndex; leftPoint < endIndex &&
77                 _timeTable.ElementAt(leftPoint).Times.ElementAt(index) <= timeSecond;
78                 ++leftPoint)
79             {
80                 ;
81             }
82             leftPoint -= 1;
83         }
84
85         double x1 = _timeTable.ElementAt(leftPoint).Times.ElementAt(index);
86         double y1 = _timeTable.ElementAt(leftPoint).Radius;
87         double x2 = _timeTable.ElementAt(leftPoint + 1).Times.ElementAt(index);
88         double y2 = _timeTable.ElementAt(leftPoint + 1).Radius;
89         _logger.LinearPolating(x1, y1, x2, y2);
90
91     }
92 }
```

Listing A.2.7: EasonEetwViewer.JmaTravelTime/Services/JmaTimeTable.cs

A.3 Kyoshin Monitor (Kmoni)

```

1  using System.Text.Json;
2  using EasonEetwViewer.KyoshinMonitor.Abstractions;
3  using EasonEetwViewer.KyoshinMonitor.Dtos;
4  using EasonEetwViewer.KyoshinMonitor.Services;
5  using Microsoft.Extensions.DependencyInjection;
6  using Microsoft.Extensions.Logging;
7  using Microsoft.Extensions.Options;
8
9  namespace EasonEetwViewer.KyoshinMonitor.Extensions;
10 
```

```

11  /// Provides extensions to inject services of <see cref="IImageFetch"/> and <see
12  /// cref="IPointExtract"/> to the service collection.
13  /// </summary>
14  public static class KmoniServiceCollectionExtensions
15  {
16      /// <summary>
17      /// Add default implementations of <see cref="IImageFetch"/> and <see
18      /// cref="IPointExtract"/> to the service collection.
19      /// </summary>
20      /// <param name="services">The <see cref="IServiceCollection"/> where the services
21      /// is to be added.</param>
22      /// <returns>The instance of <see cref="IServiceCollection"/> to be
23      /// chained.</returns>
24      public static IServiceCollection AddKmoniHelpers(this IServiceCollection services)
25          => services
26              .AddSingleton<IImageFetch, ImageFetch>(static provider
27                  =>
28                  {
29                      KmoniHelperOptions options = provider.GetRequiredService<IOptions<
30                          <KmoniHelperOptions>>().Value;
31                      ILogger<ImageFetch> logger =
32                          provider.GetRequiredService<ILogger<ImageFetch>>();
33                      return new(logger, options.BaseUri, options.RelativeUri);
34                  })
35              .AddSingleton<IPointExtract, PointExtract>(static provider
36                  =>
37                  {
38                      KmoniHelperOptions options = provider.GetRequiredService<IOptions<
39                          <KmoniHelperOptions>>().Value;
40                      return new(JsonSerializer.Deserialize<IEnumerable<ObservationPoin
41                          t>>(File.ReadAllText(options.FilePath)) ?? []);
42                  });
43  }

```

Listing A.3.1:

EasonEetwViewer.KyoshinMonitor/Extensions/KmoniServiceCollectionExtensions.cs

```

1  using SkiaSharp;
2
3  namespace EasonEetwViewer.KyoshinMonitor.Extensions;
4
5  /// <summary>
6  /// Provides methods to convert between different values.
7  /// </summary>
8  public static class ColourConversionExtensions
9  {
10     /// <summary>
11     /// Converts intensity to normalised height.
12     /// </summary>
13     /// <param name="intensity">The intensity.</param>
14     /// <returns>The normalised height.</returns>
15     public static double IntensityToHeight(this double intensity)
16         => (intensity + 3) / 10;
17     /// <summary>
18     /// Converts PGA to normalised height.
19     /// </summary>
20     /// <param name="pga">The Peak Ground Acceleration.</param>
21     /// <returns>The normalised height.</returns>

```

```

22     public static double PgaToHeight(this double pga)
23         => (Math.Log(pga, 10) + 2) / 5;
24     /// <summary>
25     /// Converts PGV to normalised height.
26     /// </summary>
27     /// <param name="pgv">The Peak Ground Velocity.</param>
28     /// <returns>The normalised height.</returns>
29     public static double PgvToHeight(this double pgv)
30         => (Math.Log(pgv, 10) + 3) / 5;
31     /// <summary>
32     /// Converts PGD to normalised height.
33     /// </summary>
34     /// <param name="pgd">The Peak Ground Displacement.</param>
35     /// <returns>The normalised height.</returns>
36     public static double PgdToHeight(this double pgd)
37         => (Math.Log(pgd, 10) + 4) / 5;
38     /// <summary>
39     /// Converts normalised height to intensity.
40     /// </summary>
41     /// <param name="height">The normalised height.</param>
42     /// <returns>The intensity.</returns>
43     public static double HeightToIntensity(this double height)
44         => (10 * height) - 3;
45     /// <summary>
46     /// Converts normalised height to PGA.
47     /// </summary>
48     /// <param name="height">The normalised height.</param>
49     /// <returns>The Peak Ground Acceleration.</returns>
50     public static double HeightToPga(this double height)
51         => Math.Pow(10, (5 * height) - 2);
52     /// <summary>
53     /// Converts normalised height to PGV.
54     /// </summary>
55     /// <param name="height">The normalised height.</param>
56     /// <returns>The Peak Ground Velocity.</returns>
57     public static double HeightToPgv(this double height)
58         => Math.Pow(10, (5 * height) - 3);
59     /// <summary>
60     /// Converts normalised height to PGD.
61     /// </summary>
62     /// <param name="height">The normalised height.</param>
63     /// <returns>The Peak Ground Displacement.</returns>
64     public static double HeightToPgd(this double height)
65         => Math.Pow(10, (5 * height) - 4);
66     /// <summary>
67     /// Converts normalised height to hue.
68     /// </summary>
69     /// <param name="height">The normalised height.</param>
70     /// <returns>The hue component of the colour.</returns>
71     public static double HeightToHue(this double height)
72         => height switch
73         {
74             <= 0.1 => (-150 * height) + 237,
75             >= 0.1 and <= 0.6 => (222 * (height - 0.3) * (height - 0.4) * (height -
76                 <= 0.6) / ((0.1 - 0.3) * (0.1 - 0.4) * (0.1 - 0.6)))
77                 + (115 * (height - 0.1) * (height - 0.4) * (height - 0.6)
78                     <= / ((0.3 - 0.1) * (0.3 - 0.4) * (0.3 - 0.6)))
79                     + (79.5 * (height - 0.1) * (height - 0.3) * (height -
80                         <= 0.6) / ((0.4 - 0.1) * (0.4 - 0.3) * (0.4 - 0.6)))

```

```

78             + (51 * (height - 0.1) * (height - 0.3) * (height - 0.4))
79             ↪ / ((0.6 - 0.1) * (0.6 - 0.3) * (0.6 - 0.4))),
80         >= 0.6 and <= 0.9 => (-170 * height) + 153,
81         - => 0
82     };
83     /// <summary>
84     /// Converts normalised height to saturation.
85     /// </summary>
86     /// <param name="height">The normalised height.</param>
87     /// <returns>The saturation component of the colour.</returns>
88     public static double HeightToSaturation(this double height)
89     => height switch
90     {
91         <= 0.2 => 1,
92         >= 0.2 and <= 0.29 => (-2.611 * height) + 1.522,
93         >= 0.29 and <= 0.4 => (1.682 * height) + 0.277,
94         >= 0.4 and <= 0.5 => (0.5 * height) + 0.75,
95         - => 1
96     };
97     /// <summary>
98     /// Converts normalised height to value.
99     /// </summary>
100    /// <param name="height">The normalised height.</param>
101    /// <returns>The value component of the colour.</returns>
102    public static double HeightToValue(this double height)
103    => height switch
104    {
105        <= 0.1 => (1.8 * height) + 0.8,
106        >= 0.1 and <= 0.172 => (-4.444 * height) + 1.424,
107        >= 0.172 and <= 0.2 => (5.714 * height) - 0.323,
108        >= 0.2 and <= 0.3 => (1.6 * height) + 0.5,
109        >= 0.3 and <= 0.4 => (0.2 * height) + 0.92,
110        >= 0.4 and <= 0.8 => 1,
111        >= 0.8 and <= 0.9 => (-0.3 * height) + 1.24,
112        - => (-2.9 * height) + 3.58
113    };
114    /// <summary>
115    /// Converts normalised height to <see cref="SKColor"/>.
116    /// </summary>
117    /// <param name="height">The normalised height.</param>
118    /// <returns>The colour.</returns>
119    public static SKColor HeightToColour(this double height)
120    => SKColor.FromHsv((float)height.HeightToHue(),
121        ↪ (float)height.HeightToSaturation() * 100, (float)height.HeightToValue() *
122        ↪ 100);
123    /// <summary>
124    /// Converts the <see cref="SKColor"/> to the normalised height.
125    /// </summary>
126    /// <param name="colour">The colour.</param>
127    /// <returns>The normalised height.</returns>
128    public static double ColourToHeight(this SKColor colour)
129    {
130        colour.ToHsv(out float h, out float _, out float v);
131
132        v /= 100; // Since SkiaSharp use percentage for v without the percentage sign
133
134        if (h == 0) // Use v
135        {
136            return (v - 3.58) / -2.9;

```

```

134     }
135     else // Use h
136     {
137         if (h >= 222)
138         {
139             return (h - 237) / -150;
140         }
141
142         if (h <= 51)
143         {
144             return (h - 153) / -170;
145         }
146
147         // Binary Search for Numerical Inverse Cubic
148
149         double left = 0.1;
150         double right = 0.6;
151         const double heightEpsilon = 0.01;
152         const double hueEpsilon = 0.5;
153         while (right - left >= heightEpsilon)
154         {
155             double mid = (right + left) / 2;
156             double calculatedHue = mid.HeightToHue();
157             if (Math.Abs(calculatedHue - h) <= hueEpsilon)
158             {
159                 return mid;
160             }
161
162             if (calculatedHue > h)
163             {
164                 left = mid;
165             }
166
167             if (calculatedHue < h)
168             {
169                 right = mid;
170             }
171         }
172
173         return (right + left) / 2;
174     }
175 }
176 }
```

Listing A.3.2: EasonEetwViewer.KyoshinMonitor/Extensions/ColourConversionExtensions.cs

```

1  using EasonEetwViewer.KyoshinMonitor.Abstractions;
2  using SkiaSharp;
3
4  namespace EasonEetwViewer.KyoshinMonitor.Extensions;
5  ///<summary>
6  /// Provides extension method to bridge between <see cref="IImageFetch"/> and <see
7  /// cref="IPointExtract"/>.
8  ///</summary>
9  public static class ByteArrayToBitmapExtensions
{
10    ///<summary>
11    /// Converts a <see cref="byte"/> array to <see cref="SKBitmap"/>.
```

```

12     /// </summary>
13     /// <param name="bytes">The byte array to be converted.</param>
14     /// <returns>The converted bitmap.</returns>
15     public static SKBitmap ToBitmap(this byte[] bytes)
16         => SKBitmap.FromImage(SKImage.FromEncodedData(bytes));
17 }

```

Listing A.3.3: EasonEetwViewer.KyoshinMonitor/Extensions/ByteArrayToBitmapExtensions.cs

```

1  using System.Diagnostics;
2  using EasonEetwViewer.KyoshinMonitor.Abstractions;
3
4  namespace EasonEetwViewer.KyoshinMonitor.Extensions;
5  /// <summary>
6  /// Provides extension methods for converting enums to URI strings.
7  /// </summary>
8  internal static class EnumToUriStringExtensions
{
    /// <summary>
    /// Represents the enum in a string that is used in the URI of the kmoni.
    /// </summary>
    /// <param name="kmoniDataType">The current instance of
    /// <c>KmoniDataType</c></param>
    /// <returns>A string that is used in the URI of kmoni.</returns>
    public static string ToUriString(this MeasurementType kmoniDataType)
        => kmoniDataType switch
    {
        MeasurementType.MeasuredIntensity
            => "jma",
        MeasurementType.PeakGroundAcceleration
            => "acmap",
        MeasurementType.PeakGroundVelocity
            => "vcmap",
        MeasurementType.PeakGroundDisplacement
            => "dcmap",
        MeasurementType.Response0125
            => "rsp0125",
        MeasurementType.Response0250
            => "rsp0250",
        MeasurementType.Response0500
            => "rsp0500",
        MeasurementType.Response1000
            => "rsp1000",
        MeasurementType.Response2000
            => "rsp2000",
        MeasurementType.Response4000
            => "rsp4000",
        -
            => throw new UnreachableException(),
    };
    /// <summary>
    /// Represents the enum in a string that is used in the URI of the kmoni.
    /// </summary>
    /// <param name="sensorType">The current instance of <c>SensorType</c></param>
    /// <returns>A string that is used in the URI of kmoni.</returns>
    public static string ToUriString(this SensorType sensorType)
        => sensorType switch
    {

```

```

49     SensorType.Surface
50         => "s",
51     SensorType.Borehole
52         => "b",
53     -
54         => throw new UnreachableException(),
55     );
56 }

```

Listing A.3.4: EasonEetwViewer.KyoshinMonitor/Extensions/EnumToUriStringExtensions.cs

```

1  using System.Text.Json;
2  using EasonEetwViewer.KyoshinMonitor.Dtos;
3
4  namespace EasonEetwViewer.KyoshinMonitor.Extensions;
5  /// <summary>
6  /// Provides extension methods to write a collection of observation points to a
7  /// specified file path.
8  /// </summary>
9  public static class ObservationPointCollectionExtensions
{
10    /// <summary>
11    /// Writes the list of observation points to the specified file path.
12    /// </summary>
13    /// <param name="observationpoints">The collection of points.</param>
14    /// <param name="filePath">The path to write the file to.</param>
15    public static void ToFile(this IEnumerable<ObservationPoint> observationpoints,
16        string filePath)
17        => File.WriteAllText(filePath, JsonSerializer.Serialize(observationpoints));
}

```

Listing A.3.5:

EasonEetwViewer.KyoshinMonitor/Extensions/ObservationPointCollectionExtensions.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.KyoshinMonitor.Dtos;
4  /// <summary>
5  /// Represents the type of an observation point.
6  /// </summary>
7  [JsonConverter(typeof(JsonStringEnumConverter<PointType>))]
8  public enum PointType
{
9
10    /// <summary>
11    /// KiK-net. The value <c>KiK_net</c>.
12    /// </summary>
13    [JsonStringEnumMemberName("KiK_net")]
14    KiK,
15    /// <summary>
16    /// K-NET. The value <c>K_NET</c>.
17    /// </summary>
18    [JsonStringEnumMemberName("K_NET")]
19    K
}

```

Listing A.3.6: EasonEetwViewer.KyoshinMonitor/Dtos/PointType.cs

```

1  using EasonEetwViewer.KyoshinMonitor.Services;
2
3  namespace EasonEetwViewer.KyoshinMonitor.Dtos;
4  /// <summary>
5  /// The options to create an instance of <see cref="ImageFetch"/> and an instance of
6  /// <see cref="PointExtract"/>.
7  /// </summary>
8  public sealed record KmoniHelperOptions
9  {
10    /// <summary>
11    /// The base URI for the images.
12    /// </summary>
13    public required string BaseUri { get; set; }
14    /// <summary>
15    /// The relative URI for the images, with data yet to be filled in.
16    /// </summary>
17    public required string RelativeUri { get; set; }
18    /// <summary>
19    /// The path to the path of the JSON file which stores the observation points.
20    /// </summary>
21    public required string FilePath { get; set; }
}

```

Listing A.3.7: EasonEetwViewer.KyoshinMonitor/Dtos/KmoniHelperOptions.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.KyoshinMonitor.Dtos;
4  /// <summary>
5  /// Represents the position of a pixel.
6  /// </summary>
7  public record PixelCoordinate
8  {
9    /// <summary>
10   /// The property <c>x</c>, representing the position of the pixel on the width
11   /// axis.
12   /// </summary>
13   [JsonPropertyName("x")]
14   public required int X { get; init; }
15   /// <summary>
16   /// The property <c>y</c>, representing the position of the pixel on the height
17   /// axis.
18   /// </summary>
19   [JsonPropertyName("y")]
20   public required int Y { get; init; }
}

```

Listing A.3.8: EasonEetwViewer.KyoshinMonitor/Dtos/PixelCoordinate.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.KyoshinMonitor.Dtos;
4  /// <summary>
5  /// Represents an observation point.
6  /// </summary>
7  public record ObservationPoint
8  {

```

```

9   ///<summary>
10  /// The property <c>type</c>, whether it is in <c>K-NET</c> or <c>KiK-net</c>.
11  ///</summary>
12  [JsonPropertyName("type")]
13  public required PointType Type { get; init; }
14  ///<summary>
15  /// The property <c>code</c>, the unique code of the observation point.
16  ///</summary>
17  [JsonPropertyName("code")]
18  public required string Code { get; init; }
19  ///<summary>
20  /// The property <c>name</c>, the unique name of the observation point.
21  ///</summary>
22  [JsonPropertyName("name")]
23  public required string Name { get; init; }
24  ///<summary>
25  /// The property <c>region</c>, the region of the observation point.
26  ///</summary>
27  [JsonPropertyName("region")]
28  public required string Region { get; init; }
29  ///<summary>
30  /// The property <c>isSuspended</c>, whether the observation point is in use or is
31  → suspended.
32  ///</summary>
33  [JsonPropertyName("isSuspended")]
34  public required bool IsSuspended { get; init; }
35  ///<summary>
36  /// The property <c>location</c>, the geographic location of the observation
37  → point.
38  ///</summary>
39  [JsonPropertyName("location")]
40  public required GeographicCoordinate Location { get; init; }
41  ///<summary>
42  /// The property <c>point</c>, the pixel position of the observation point.
43  ///</summary>
44  [JsonInclude]
45  [JsonPropertyName("point")]
46  public required PixelCoordinate Point { get; init; }
47 }
```

Listing A.3.9: EasonEetwViewer.KyoshinMonitor.Dtos/ObservationPoint.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.KyoshinMonitor.Dtos;
4  ///<summary>
5  /// Represents a physical position.
6  ///</summary>
7  public record GeographicCoordinate
8  {
9    ///<summary>
10   /// The property <c>latitude</c>, the latitude of the position.
11   ///</summary>
12   [JsonPropertyName("latitude")]
13   public required double Latitude { get; init; }
14   ///<summary>
15   /// The property <c>longitude</c>, the longitude of the position.
16   ///</summary>
```

```

17     [JsonPropertyName("longitude")]
18     public required double Longitude { get; init; }
19 }
```

Listing A.3.10: EasonEetwViewer.KyoshinMonitor/Dtos/GeographicCoordinate.cs

```

1  namespace EasonEetwViewer.KyoshinMonitor.Abstractions;
2  /// <summary>
3  /// Represents the data type for the data map in kmoni.
4  /// </summary>
5  public enum MeasurementType
{
6      /// <summary>
7      /// Real-time measured intensity. The value <c>jma</c>.
8      /// </summary>
9      MeasuredIntensity,
10     /// <summary>
11     /// Peak (maximal) ground acceleration. The value <c>acmap</c>.
12     /// </summary>
13     PeakGroundAcceleration,
14     /// <summary>
15     /// Peak (maximal) ground velocity. The value <c>ucmap</c>.
16     /// </summary>
17     PeakGroundVelocity,
18     /// <summary>
19     /// Peak (maximal) ground displacement. The value <c>demap</c>.
20     /// </summary>
21     PeakGroundDisplacement,
22     /// <summary>
23     /// Response spectrum for 0.125Hz PGV. The value <c>rsp0125</c>.
24     /// </summary>
25     Response0125,
26     /// <summary>
27     /// Response spectrum for 0.250Hz PGV. The value <c>rsp0250</c>.
28     /// </summary>
29     Response0250,
30     /// <summary>
31     /// Response spectrum for 0.500Hz PGV. The value <c>rsp0500</c>.
32     /// </summary>
33     Response0500,
34     /// <summary>
35     /// Response spectrum for 1.000Hz PGV. The value <c>rsp1000</c>.
36     /// </summary>
37     Response1000,
38     /// <summary>
39     /// Response spectrum for 2.000Hz PGV. The value <c>rsp2000</c>.
40     /// </summary>
41     Response2000,
42     /// <summary>
43     /// Response spectrum for 4.000Hz PGV. The value <c>rsp4000</c>.
44     /// </summary>
45     Response4000
46 }
```

Listing A.3.11: EasonEetwViewer.KyoshinMonitor/Abstractions/MeasurementType.cs

```

1  namespace EasonEetwViewer.KyoshinMonitor.Abstractions;
```

```

2  ///<summary>
3  /// Represents the functionality to fetch a kmoni image.
4  ///</summary>
5  public interface IImageFetch
6  {
7      ///<summary>
8      /// Gets the GIF image as a byte array with the specified parameters.
9      ///</summary>
10     ///<param name="measurementType">The data type plotted on the image.</param>
11     ///<param name="sensorType">The sensor type included on the image.</param>
12     ///<param name="dateTime">The date and time to be fetched, in JST.</param>
13     ///<returns>The byte array obtained, <see langword="null"/> if attempt is
14     → unsuccessful.</returns>
15     Task<byte[]?> GetByteArrayAsync(MeasurementType measurementType, SensorType
16     → sensorType, DateTime dateTime);
16 }

```

Listing A.3.12: EasonEetwViewer.KyoshinMonitor/Abstractions/IImageFetch.cs

```

1  namespace EasonEetwViewer.KyoshinMonitor.Abstractions;
2  ///<summary>
3  /// Represents the sensor type for the data map in kmoni.
4  ///</summary>
5  public enum SensorType
6  {
7      ///<summary>
8      /// Surface sensors, both K-NET and KiK-net. The value <c>s</c>.
9      ///</summary>
10     Surface,
11     ///<summary>
12     /// Borehole sensors, only KiK-net sensors. The value <c>b</c>.
13     ///</summary>
14     Borehole
15 }

```

Listing A.3.13: EasonEetwViewer.KyoshinMonitor/Abstractions/SensorType.cs

```

1  using EasonEetwViewer.KyoshinMonitor.Dtos;
2  using SkiaSharp;
3
4  namespace EasonEetwViewer.KyoshinMonitor.Abstractions;
5
6  ///<summary>
7  /// Represents the functionality to extract the colour of points from a <see
8  → cref="SKBitmap"/>.
9  ///</summary>
10    public interface IPointExtract
11    {
12        ///<summary>
13        /// Extract the colours from the Bitmap.
14        ///</summary>
15        ///<param name="bitmap">The <c>SKBitmap</c> to have its pixels extracted from the
16        → <see cref="SKBitmap"/>.</param>
17        ///<param name="kikNetOnly">Whether <c>KiK-net</c> sensors only should be
18        → extracted.</param>

```

```

16     ///<returns>A collection of pairs of observation points and their colours in <see
17     <see cref="SKColor"/>.</returns>
18     IEnumerable<(ObservationPoint point, SKColor colour)> ExtractColours(SKBitmap
19     bitmap, bool kikNetOnly = false);
20 }
```

Listing A.3.14: EasonEetwViewer.KyoshinMonitor/Abstractions/IPointExtract.cs

```

1  using EasonEetwViewer.KyoshinMonitor.Services;
2  using Microsoft.Extensions.Logging;
3
4  namespace EasonEetwViewer.KyoshinMonitor;
5  ///<summary>
6  /// Represents the log messages used in <see cref="ImageFetch"/>.
7  ///</summary>
8  internal static partial class ImageFetchLogs
9  {
10     ///<summary>
11     /// Log when instantiated.
12     ///</summary>
13     ///<param name="logger">The logger to be used.</param>
14     [LoggerMessage(
15         EventId = 0,
16         EventName = nameof(Instantiated),
17         Level = LogLevel.Information,
18         Message = "Instantiated.")]
19     public static partial void Instantiated(
20         this ILogger<ImageFetch> logger);
21
22     ///<summary>
23     /// Log when sending a request.
24     ///</summary>
25     ///<param name="logger">The logger to be used.</param>
26     ///<param name="baseUri">The base URI of request.</param>
27     ///<param name="relativeUri">The relative URI of request.</param>
28     [LoggerMessage(
29         EventId = 1,
30         EventName = nameof(SendingRequest),
31         Level = LogLevel.Trace,
32         Message = "Sending HTTP Request to `{BaseUri}` and `{RelativeUri}`.")]
33     public static partial void SendingRequest(
34         this ILogger<ImageFetch> logger, string baseUri, string relativeUri);
35
36     ///<summary>
37     /// Log when a request was successful.
38     ///</summary>
39     ///<param name="logger">The logger to be used.</param>
40     [LoggerMessage(
41         EventId = 2,
42         EventName = nameof(RequestSuccessful),
43         Level = LogLevel.Debug,
44         Message = "HTTP Request was successful.")]
45     public static partial void RequestSuccessful(
46         this ILogger<ImageFetch> logger);
47
48     ///<summary>
49     /// Log when a request was unsuccessful.
50     ///</summary>
```

```

51     ///<param name="logger">The logger to be used.</param>
52     ///<param name="baseUri">The base URI of request.</param>
53     ///<param name="relativeUri">The relative URI of request.</param>
54     [LoggerMessage(
55         EventId = 3,
56         EventName = nameof(RequestUnsuccessful),
57         Level = LogLevel.Warning,
58         Message = "HTTP Request to `{BaseUri}` and `{RelativeUri}` was
59             → unsuccessful.")]
60     public static partial void RequestUnsuccessful(
61         this ILogger<ImageFetch> logger, string baseUri, string relativeUri);
62 }
```

Listing A.3.15: EasonEetwViewer.KyoshinMonitor/Logging/ImageFetchLogs.cs

```

1  using EasonEetwViewer.KyoshinMonitor.Abstractions;
2  using EasonEetwViewer.KyoshinMonitor.Extensions;
3  using Microsoft.Extensions.Logging;
4
5  namespace EasonEetwViewer.KyoshinMonitor.Services;
6
7  ///<summary>
8  /// Default implementation of <see cref="IImageFetch"/>.
9  ///</summary>
10 internal sealed class ImageFetch : IImageFetch
11 {
12     ///<summary>
13     /// The <see cref="HttpClient"/> used to make GET requests.
14     ///</summary>
15     private readonly HttpClient _client;
16     ///<summary>
17     /// The logger to be used for logging.
18     ///</summary>
19     private readonly ILogger<ImageFetch> _logger;
20     ///<summary>
21     /// The base URI for the images.
22     ///</summary>
23     private readonly string _baseUri;
24     ///<summary>
25     /// The relative URI for the images, with data yet to be filled in.
26     ///</summary>
27     private readonly string _relativeUri;
28
29     ///<summary>
30     /// Creates an instance of the class with the parameters specified.
31     ///</summary>
32     ///<param name="logger">The <see
33     → cref="ILogger{ImageFetch}">ILogger<ImageFetch>;</see> to be used for
34     → logging.</param>
35     ///<param name="baseUri">The base URI for the images.</param>
36     ///<param name="relativeUri">The relative URI for the images, with data yet to be
37     → filled in.</param>
38     public ImageFetch(ILogger<ImageFetch> logger, string baseUri, string relativeUri)
39     {
40         _logger = logger;
41         _client = new()
42         {
43             BaseAddress = new(baseUri)
44         }
45     }
46 }
```

```

41     };
42     _baseUri = baseUri;
43     _relativeUri = relativeUri;
44     _logger.Instantiated();
45 }
46
47 /// <inheritdoc/>
48 public async Task<byte[]> GetByteArrayAsync(MeasurementType measurementType,
49     → SensorType sensorType, DateTime dateTime)
50 {
51     string relativeUri = string.Format(_relativeUri,
52         → measurementType.ToString(), sensorType.ToString(), dateTime);
53
54     _logger.SendingRequest(_baseUri, relativeUri);
55     using HttpRequestMessage request = new(HttpMethod.Get, relativeUri);
56     try
57     {
58         using HttpResponseMessage response = await _client.SendAsync(request);
59
60         if (response.IsSuccessStatusCode)
61         {
62             _logger.RequestSuccessful();
63             byte[] imageBytes = await response.Content.ReadAsByteArrayAsync();
64             return imageBytes;
65         }
66         else
67         {
68             _logger.RequestUnsuccessful(_baseUri, relativeUri);
69             return null;
70         }
71     }
72     catch (HttpRequestException)
73     {
74         _logger.RequestUnsuccessful(_baseUri, relativeUri);
75         return null;
76     }
77 }
78 }
```

Listing A.3.16: EasonEetwViewer.KyoshinMonitor/Services/ImageFetch.cs

```

1  using EasonEetwViewer.KyoshinMonitor.Abstractions;
2  using EasonEetwViewer.KyoshinMonitor.Dtos;
3  using SkiaSharp;
4
5  namespace EasonEetwViewer.KyoshinMonitor.Services;
6  /// <summary>
7  /// Default implementation of <see cref="IPointExtract"/>
8  /// </summary>
9  internal sealed class PointExtract : IPointExtract
10 {
11     /// <summary>
12     /// Creates an instance of <see cref="PointExtract"/> by specifying the collection
13     → of observation points.
14     /// </summary>
15     /// <param name="points">The collection of observation points.</param>
16     internal PointExtract(IEnumerable<ObservationPoint> points) => _points = points;
17     /// <summary>
```

```

17  ///> The collection of observation points.
18  ///> </summary>
19  private readonly IEnumerable<ObservationPoint> _points;
20  ///> <inheritdoc/>
21  public IEnumerable<(ObservationPoint point, SKColor colour)>
22  ↳ ExtractColours(SKBitmap bitmap, bool kikNetOnly = false)
23  => _points
24  .Where(p => !p.IsSuspended && p.Point is not null)
25  .Where(p => !(kikNetOnly && p.Type != PointType.KiK))
26  .Select(p => (p, colour: bitmap.GetPixel(p.Point.X, p.Point.Y)))
27  .ToArray() // deferred execution throws System.ExecutionEngineException
28  .Where(pc => pc.colour.Alpha != 0);
}

```

Listing A.3.17: EasonEetwViewer.KyoshinMonitor/Services/PointExtract.cs

A.4 DM-D.S.S. Common DTOs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
5  ///> <summary>
6  ///> Describes the magnitude of an earthquake.
7  ///> </summary>
8  public record Magnitude
{
9
10  ///> <summary>
11  ///> The property <c>type</c>. A constant <c> マグニチュード </c> for magnitude.
12  ///> </summary>
13  [JsonPropertyName("type")]
14  public string Type { get; } = " マグニチュード";
15  ///> <summary>
16  ///> The property <c>unit</c>. The units of the magnitude.
17  ///> </summary>
18  [JsonPropertyName("unit")]
19  public required MagnitudeUnit Unit { get; init; }
20  ///> <summary>
21  ///> The property <c>value</c>. The magnitude of the earthquake.
22  ///> </summary>
23  ///> <remarks>
24  ///> <see langword="null"/> when unclear or greater than M8.
25  ///> </remarks>
26  [JsonPropertyName("value")]
27  public required float? Value { get; init; }
28  ///> <summary>
29  ///> The property <c>condition</c>. Abnormal behaviours of the magnitude.
30  ///> </summary>
31  ///> <remarks>
32  ///> <see langword="null"/> when <see cref="Value"/> is not <see langword="null"/>.
33  ///> </remarks>
34  [JsonPropertyName("condition")]
35  public MagnitudeCondition? Condition { get; init; }
36 }

```

Listing A.4.1: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Magnitude.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
4
5  /// <summary>
6  /// Represents a coordinate, either longitude or latitude.
7  /// </summary>
8  public record Coordinate
{
9
10     /// <summary>
11     /// The property <c>text</c>. The displayed text of the coordinate.
12     /// </summary>
13     [JsonPropertyName("text")]
14     public required string DisplayText { get; init; }
15
16     /// <summary>
17     /// The property <c>value</c>. The double value of the coordinate.
18     /// </summary>
19     [JsonPropertyName("value")]
20     public required double DoubleValue { get; init; }
}

```

Listing A.4.2: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Coordinate.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
4
5  /// <summary>
6  /// Represents a distance.
7  /// </summary>
8  public record Distance
{
9
10     /// <summary>
11     /// The property <c>unit</c>. A constant <c>km</c>.
12     /// </summary>
13     [JsonPropertyName("unit")]
14     public string Unit { get; } = "km";
15
16     /// <summary>
17     /// The property <c>value</c>. The value of the distance.
18     /// </summary>
19     [JsonPropertyName("value")]
20     public required int Value { get; set; }
}

```

Listing A.4.3: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Distance.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
5
6  /// <summary>
7  /// Represents a pair of coordinates describing a position.
8  /// </summary>
9  public record CoordinateComponent
{
11     /// <summary>

```

```

12     /// The property <c>latitude</c>. The latitude coordinate of the position.
13     /// </summary>
14     /// <remarks>
15     /// <see langword="null"/> when unclear.
16     /// </remarks>
17     [JsonPropertyName("latitude")]
18     public Coordinate? Latitude { get; init; }
19     /// <summary>
20     /// The property <c>longitude</c>. The longitude coordinate of the position.
21     /// </summary>
22     /// <remarks>
23     /// <see langword="null"/> when unclear.
24     /// </remarks>
25     [JsonPropertyName("longitude")]
26     public Coordinate? Longitude { get; init; }
27     /// <summary>
28     /// The property <c>height</c>. The height of the position.
29     /// </summary>
30     /// <remarks>
31     /// <see langword="null"/> when unclear or undefined.
32     /// </remarks>
33     [JsonPropertyName("height")]
34     public Height? Height { get; init; }
35     /// <summary>
36     /// The property <c>geodeticSystem</c>. The geodetic system of the coordinate.
37     /// </summary>
38     /// <remarks>
39     /// <see langword="null"/> depending on the situation.
40     /// </remarks>
41     [JsonPropertyName("geodeticSystem")]
42     public Geodetic? GeodeticSystem { get; init; }
43     /// <summary>
44     /// The property <c>condition</c>. Extra information provided for the coordinate.
45     /// </summary>
46     /// <remarks>
47     /// <see langword="null"/> when unnecessary.
48     /// <c>不明 </c> when coordinate is unclear.
49     /// </remarks>
50     [JsonPropertyName("condition")]
51     public string? Condition { get; init; }
52 }
```

Listing A.4.4: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/CoordinateComponent.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
4
5  /// <summary>
6  /// Represents the height of a position.
7  /// </summary>
8  public record Height
{
9
10    /// <summary>
11    /// The property <c>type</c>. A constant <c>高さ </c> for the height.
12    /// </summary>
13    [JsonPropertyName("type")]
14    public string Type { get; } = "高さ";
```

```

15     ///<summary>
16     /// The property <c>unit</c>. A constant <c>m</c> for the height.
17     ///</summary>
18     [JsonPropertyName("unit")]
19     public string Unit { get; } = "m";
20     ///<summary>
21     /// The property <c>value</c>. The value of the height.
22     ///</summary>
23     [JsonPropertyName("value")]
24     public required float Value { get; init; }
25 }
```

Listing A.4.5: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Height.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
5  ///<summary>
6  /// Represents the hypocentre of an earthquake.
7  ///</summary>
8  public record Hypocentre
{
    ///<summary>
    /// The property <c>name</c>. The name of the area of the hypocentre.
    ///</summary>
    [JsonPropertyName("name")]
    public required string Name { get; init; }

    ///<summary>
    /// The property <c>code</c>. The XML code of the area of the hypocentre.
    ///</summary>
    [JsonPropertyName("code")]
    public required int Code { get; init; }

    ///<summary>
    /// The property <c>coordinate</c>. The position of the hypocentre.
    ///</summary>
    [JsonPropertyName("coordinate")]
    public required CoordinateComponent Coordinates { get; init; }

    ///<summary>
    /// The property <c>depth</c>. The depth of the hypocentre.
    ///</summary>
    [JsonPropertyName("depth")]
    public required Depth Depth { get; init; }

    ///<summary>
    /// The property <c>detailed</c>. The extra details of the hypocentre.
    ///</summary>
    ///<remarks>
    ///<see langword="null"/> when the earthquake happens within Japan.
    ///</remarks>
    [JsonPropertyName("detailed")]
    public HypocentreDetail? Detail { get; init; }

    ///<summary>
    /// The property <c>auxiliary</c>. The auxiliary information of the hypocentre.
    ///</summary>
    ///<remarks>
    ///<see langword="null"/> depending on situation.
    ///</remarks>
    [JsonPropertyName("auxiliary")]
}
```

```

45     public Auxiliary? Auxiliary { get; init; }
46     /// <summary>
47     /// The property <c>source</c>. The source of the hypocentre information.
48     /// </summary>
49     /// <remarks>
50     /// <see langword="null"/> when the earthquake happens within Japan.
51     /// </remarks>
52     [JsonPropertyName("source")]
53     public Source? Source { get; init; }
54 }
```

Listing A.4.6: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Hypocentre.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
4
5  /// <summary>
6  /// Represents auxiliary information provided on the position of the hypocentre.
7  /// </summary>
8  public record Auxiliary
{
9
10    /// <summary>
11    /// The property <c>text</c>. The position where the hypocentre is captured.
12    /// </summary>
13    [JsonPropertyName("text")]
14    public required string Text { get; init; }
15    /// <summary>
16    /// The property <c>code</c>. The area code for which the hypocentre is captured.
17    /// </summary>
18    [JsonPropertyName("code")]
19    public required int Code { get; init; }
20    /// <summary>
21    /// The property <c>name</c>. The area name for which the hypocentre is captured.
22    /// </summary>
23    [JsonPropertyName("name")]
24    public required string Name { get; init; }
25    /// <summary>
26    /// The property <c>direction</c>. The direction in which the hypocentre is
27    → positioned relative to the area.
28    /// </summary>
29    [JsonPropertyName("direction")]
30    public required string Direction { get; init; }
31    /// <summary>
32    /// The property <c>distance</c>. The distance of the hypocentre to the area of
33    → capture.
34    /// </summary>
35    [JsonPropertyName("distance")]
36    public required Distance Distance { get; init; }
}
```

Listing A.4.7: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Auxiliary.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
4
```

```

5  ///<summary>
6  /// Provides details for the hypocentre area name.
7  ///</summary>
8  public record HypocentreDetail
9  {
10     ///<summary>
11     /// The property <c>code</c>. The XML code for the area.
12     ///</summary>
13     [JsonPropertyName("code")]
14     public required int Code { get; init; }
15     ///<summary>
16     /// The property <c>name</c>. The name for the area.
17     ///</summary>
18     [JsonPropertyName("name")]
19     public required string Name { get; init; }
20 }

```

Listing A.4.8: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/HypocentreDetail.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
5  ///<summary>
6  /// Describes the depth of an earthquake.
7  ///</summary>
8  public record Depth
9  {
10     ///<summary>
11     /// The property <c>type</c>. A constant <c>深さ</c> for depth.
12     ///</summary>
13     [JsonPropertyName("type")]
14     public string Type { get; } = "深さ";
15     ///<summary>
16     /// The property <c>unit</c>. A constant <c>km</c> for depth.
17     ///</summary>
18     [JsonPropertyName("unit")]
19     public string Unit { get; } = "km";
20     ///<summary>
21     /// The property <c>value</c>. The depth of the earthquake.
22     ///</summary>
23     ///<remarks>
24     ///<see langword="null"/> when unclear.
25     ///</remarks>
26     [JsonPropertyName("value")]
27     public required int? Value { get; init; }
28     ///<summary>
29     /// The property <c>condition</c>. Describes abnormal behaviours of the depth.
30     ///</summary>
31     ///<remarks>
32     ///<see langword="null"/> when <see cref="Value"/> is not <see langword="null"/>
33     ↵ and not 0 or 700.
34     ///</remarks>
35     [JsonPropertyName("condition")]
36     public DepthCondition? Condition { get; init; }
}

```

Listing A.4.9: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Depth.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
4  /// <summary>
5  /// Describes an earthquake.
6  /// </summary>
7  public record EarthquakeComponent
8  {
9      /// <summary>
10     /// The property <c>originTime</c>, the origin time of the earthquake.
11     /// </summary>
12     [JsonPropertyName("originTime")]
13     public required DateTimeOffset OriginTime { get; init; }
14     /// <summary>
15     /// The property <c>arrivalTime</c>, the arrival time of the earthquake.
16     /// </summary>
17     [JsonPropertyName("arrivalTime")]
18     public required DateTimeOffset ArrivalTime { get; init; }
19     /// <summary>
20     /// The property <c>hypocenter</c>, the hypocentre of the earthquake.
21     /// </summary>
22     [JsonPropertyName("hypocenter")]
23     public required Hypocentre Hypocentre { get; init; }
24     /// <summary>
25     /// The property <c>magnitude</c>, the magnitude of the earthquake.
26     /// </summary>
27     [JsonPropertyName("magnitude")]
28     public required Magnitude Magnitude { get; init; }
29 }

```

Listing A.4.10: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/EarthquakeComponent.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
4
5  /// <summary>
6  /// Represents the geodetic system used by a pair of coordinates.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<Geodetic>))]
9  public enum Geodetic
10 {
11     /// <summary>
12     /// The value <c>世界測地系 </c>, representing the world geodetic system.
13     /// </summary>
14     [JsonStringEnumMemberName(" 世界測地系")]
15     World,
16     /// <summary>
17     /// The value <c>日本測地系 </c>, representing the Japanese geodetic system.
18     /// </summary>
19     [JsonStringEnumMemberName(" 日本測地系")]
20     Japan
21 }

```

Listing A.4.11: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/Geodetic.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
4
5  /// <summary>
6  /// Represents the source of information of a distant earthquake
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<Source>))]
9  public enum Source
10 {
11     /// <summary>
12     /// The value <c> U S G S </c>, representing the United States Geological Survey.
13     /// </summary>
14     [JsonStringEnumMemberName(" U S G S ")]
15     USGS,
16     /// <summary>
17     /// The value <c> P T W C </c>, representing the Pacific Tsunami Warning Centre.
18     /// </summary>
19     [JsonStringEnumMemberName(" P T W C ")]
20     PTWC,
21     /// <summary>
22     /// The value <c> S C S T A C </c>, representing the South China Sea Tsunami
23     → Advisory Centre.
24     /// </summary>
25     [JsonStringEnumMemberName(" S C S T A C ")]
26     SCSTAC,
27     /// <summary>
28     /// The value <c> C A T A C </c>, representing the Central America Tsunami Advisory
29     → Centre.
30     /// </summary>
31     [JsonStringEnumMemberName(" C A T A C ")]
32     CATAc,
33     /// <summary>
34     /// The value <c> W C A T W C </c>, representing the West Coast/Alaska Tsunami
35     → Warning Centre.
36     /// </summary>
37     [JsonStringEnumMemberName(" W C A T W C ")]
38     WCATWC
39 }

```

Listing A.4.12: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/Source.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
4
5  /// <summary>
6  /// Describes the special situations of the depth value.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<DepthCondition>))]
9  public enum DepthCondition
10 {
11     /// <summary>
12     /// The value <c> ごく浅い </c>, representing too shallow.
13     /// </summary>
14     [JsonStringEnumMemberName(" ごく浅い")]
15     Shallow,
16     /// <summary>

```

```

17     /// The value <c> 700 km以上 </c>, representing too deep.
18     /// </summary>
19     [JsonStringEnumMemberName(" 700 km以上")]
20     Deep,
21     /// <summary>
22     /// The value <c> 不明 </c>, representing unclear.
23     /// </summary>
24     [JsonStringEnumMemberName(" 不明")]
25     Unclear
26 }

```

Listing A.4.13: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/DepthCondition.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
4
5  /// <summary>
6  /// Describes the units of the magnitude.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<MagnitudeUnit>))]
9  public enum MagnitudeUnit
{
10
11     /// <summary>
12     /// The value <c>Mj</c>, representing JMA magnitude.
13     /// </summary>
14     [JsonStringEnumMemberName("Mj")]
15     JmaMagnitude,
16     /// <summary>
17     /// The value <c>M</c>, representing normal magnitude.
18     /// </summary>
19     [JsonStringEnumMemberName("M")]
20     NormalMagnitude
21 }

```

Listing A.4.14: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/MagnitudeUnit.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
4
5  /// <summary>
6  /// Describes the special situations of the magnitude value.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<MagnitudeCondition>))]
9  public enum MagnitudeCondition
{
10
11     /// <summary>
12     /// The value <c> M不明 </c>, representing unclear.
13     /// </summary>
14     [JsonStringEnumMemberName(" M不明")]
15     Unclear,
16     /// <summary>
17     /// The value <c> M8 を超える巨大地震 </c>, representing too big an earthquake.
18     /// </summary>
19     [JsonStringEnumMemberName(" M8 を超える巨大地震")]
20     Huge
21 }

```

Listing A.4.15: EasonEetwViewer.Dmdata.Dtos/DmdataComponent/Enum/MagnitudeCondition.cs

```
1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Enum;
4
5  /// <summary>
6  /// Represents the JMA intensity.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<Intensity>))]
9  public enum Intensity
{
10
11    /// <summary>
12    /// The value <c>1</c>, representing intensity 1.
13    /// </summary>
14    [JsonStringEnumMemberName("1")]
15    One,
16
17    /// <summary>
18    /// The value <c>2</c>, representing intensity 2.
19    /// </summary>
20    [JsonStringEnumMemberName("2")]
21
22    Two,
23
24    /// <summary>
25    /// The value <c>3</c>, representing intensity 3.
26    /// </summary>
27    [JsonStringEnumMemberName("3")]
28
29    Three,
30
31    /// <summary>
32    /// The value <c>4</c>, representing intensity 4.
33    /// </summary>
34    [JsonStringEnumMemberName("4")]
35
36    Four,
37
38    /// <summary>
39    /// The value <c>5-</c>, representing intensity 5 weak.
40    /// </summary>
41    [JsonStringEnumMemberName("5-")]
42
43    FiveWeak,
44
45    /// <summary>
46    /// The value <c>5+</c>, representing intensity 5 strong.
47    /// </summary>
48    [JsonStringEnumMemberName("5+")]
49
50    FiveStrong,
51
52    /// <summary>
53    /// The value <c>6-</c>, representing intensity 6 weak.
54    /// </summary>
55    [JsonStringEnumMemberName("6-")]
56
57    SixWeak,
58
59    /// <summary>
60    /// The value <c>6+</c>, representing intensity 6 strong.
61    /// </summary>
62    [JsonStringEnumMemberName("6+")]
63
64    SixStrong,
65
66    /// <summary>
67    /// The value <c>7</c>, representing intensity 7.
68    /// </summary>
69    [JsonStringEnumMemberName("7")]
70
71    Seven
```

56 }

Listing A.4.16: EasonEetwViewer.Dmdata.Dtos/Enum/Intensity.cs

```
1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Enum;
4
5  /// <summary>
6  /// Represents classifications of telegrams and subscriptions.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<Classification>))]
9  public enum Classification
{
10
11    /// <summary>
12    /// The value <c>telegram.earthquake</c>. Earthquake telegrams.
13    /// </summary>
14    [JsonStringEnumMemberName("telegram.earthquake")]
15    TelegramEarthquake,
16
17    /// <summary>
18    /// The value <c>telegram.volcano</c>. Volcano telegrams.
19    /// </summary>
20    [JsonStringEnumMemberName("telegram.volcano")]
21    TelegramVolcano,
22
23    /// <summary>
24    /// The value <c>telegram.weather</c>. Weather telegrams.
25    /// </summary>
26    [JsonStringEnumMemberName("telegram.weather")]
27    TelegramWeather,
28
29    /// <summary>
30    /// The value <c>telegram.forecast</c>. Forecast telegrams.
31    /// </summary>
32    [JsonStringEnumMemberName("telegram.forecast")]
33    TelegramForecast,
34
35    /// <summary>
36    /// The value <c>telegram.scheduled</c>. Scheduled telegrams.
37    /// </summary>
38    [JsonStringEnumMemberName("telegram.scheduled")]
39    TelegramScheduled,
40
41    /// <summary>
42    /// The value <c>eew.warning</c>. EEW warnings.
43    /// </summary>
44    [JsonStringEnumMemberName("eew.warning")]
45    EewWarning,
46
47    /// <summary>
48    /// The value <c>eew.forecast</c>. EEW forecasts.
49    /// </summary>
50    [JsonStringEnumMemberName("eew.forecast")]
51    EewForecast,
52
53    /// <summary>
54    /// The value <c>websocket.plus5</c>. Add 5 WebSocket connections.
55    /// </summary>
56    [JsonStringEnumMemberName("websocket.plus5")]
57    WebSocketPlus5,
58
59    /// <summary>
60    /// The value <c>websocket.plus2</c>. Add 2 WebSocket connections.
61    /// </summary>
62    [JsonStringEnumMemberName("websocket.plus2")]
63}
```

```
55     WebSocketPlus2
56 }
```

Listing A.4.17: EasonEetwViewer.Dmdata.Dtos/Enum/Classification.cs

```
1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Enum;
4  /// <summary>
5  /// Describes the LPGM intensity.
6  /// </summary>
7  [JsonConverter(typeof(JsonStringEnumConverter<LgIntensity>))]
8  public enum LgIntensity
9  {
10     /// <summary>
11     /// The value <c>0</c>, representing LPGM intensity 0.
12     /// </summary>
13     [JsonStringEnumMemberName("0")]
14     Zero,
15     /// <summary>
16     /// The value <c>1</c>, representing LPGM intensity 1.
17     /// </summary>
18     [JsonStringEnumMemberName("1")]
19     One,
20     /// <summary>
21     /// The value <c>2</c>, representing LPGM intensity 2.
22     /// </summary>
23     [JsonStringEnumMemberName("2")]
24     Two,
25     /// <summary>
26     /// The value <c>3</c>, representing LPGM intensity 3.
27     /// </summary>
28     [JsonStringEnumMemberName("3")]
29     Three,
30     /// <summary>
31     /// The value <c>4</c>, representing LPGM intensity 4.
32     /// </summary>
33     [JsonStringEnumMemberName("4")]
34     Four,
35 }
```

Listing A.4.18: EasonEetwViewer.Dmdata.Dtos/Enum/LgIntensity.cs

```
1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Enum;
4
5  /// <summary>
6  /// Describes the category of the long period ground motion observed.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<LgCategory>))]
9  public enum LgCategory
10 {
11     /// <summary>
12     /// The value <c>1</c>, representing a maximum LPGM intensity 2 or less, and all
13     /// areas observing LPGM intensity 1 or above has maximum intensity 5 weak or
14     /// above.
```

```

13     /// </summary>
14     [JsonStringEnumMemberName("1")]
15     One,
16     /// <summary>
17     /// The value <c>2</c>, representing a maximum LPGM intensity 2 or less, but there
18     ↳ are areas which have maximum intensity 4 or less observing LPGM intensity 1 or
19     ↳ above.
20     /// </summary>
21     [JsonStringEnumMemberName("2")]
22     Two,
23     /// <summary>
24     /// The value <c>3</c>, representing a maximum LPGM intensity 3 or above, and all
25     ↳ areas observing LPGM intensity 3 or above has maximum intensity 5 weak or
26     ↳ above.
27     /// </summary>
28     [JsonStringEnumMemberName("3")]
29     Three,
30     /// <summary>
31     /// The value <c>4</c>, representing a maximum LPGM intensity 4 or above, but
32     ↳ there are areas which have maximum intensity 4 or less observing LPGM
33     ↳ intensity 3 or above.
34     /// </summary>
35     [JsonStringEnumMemberName("4")]
36     Four,
37 }

```

Listing A.4.19: EasonEetwViewer.Dmdata.Dtos/Enum/LgCategory.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Enum.WebSocket;
4
5  /// <summary>
6  /// Represents whether test telegrams are received by a WebSocket connection.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<TestStatus>))]
9  public enum TestStatus
10 {
11     /// <summary>
12     /// The value <c>including</c>, representing receiving test telegrams.
13     /// </summary>
14     [JsonStringEnumMemberName("including")]
15     Include,
16     /// <summary>
17     /// The value <c>no</c>, representing not receiving test telegrams.
18     /// </summary>
19     [JsonStringEnumMemberName("no")]
20     Exclude
21 }

```

Listing A.4.20: EasonEetwViewer.Dmdata.Dtos/Enum/WebSocket/TestStatus.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Enum.WebSocket;
4

```

```

5  ///<summary>
6  /// Represents the format of a telegram.
7  ///</summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<FormatType>))]
9  public enum FormatType
{
10
    ///<summary>
    /// The value <c>xml</c>, representing XML format.
11  ///</summary>
12  [JsonStringEnumMemberName("xml")]
13  Xml,
14
    ///<summary>
15  /// The value <c>json</c>, representing JSON format.
16  ///</summary>
17  [JsonStringEnumMemberName("json")]
18  Json,
19
    ///<summary>
20  /// The value <c>a/n</c>, representing alphanumeric format.
21  ///</summary>
22  [JsonStringEnumMemberName("a/n")]
23  AlphaNumeric,
24
    ///<summary>
25  /// The value <c>binary</c>, representing binary format.
26  ///</summary>
27  [JsonStringEnumMemberName("binary")]
28  Binary
29
30 }
31

```

Listing A.4.21: EasonEetwViewer.Dmdata.Dtos/Enum/WebSocket/FormatType.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Telemgram;
4  ///<summary>
5  /// Describes the control of a XML telegram.
6  ///</summary>
7  public record XmlControl
{
8
    ///<summary>
    /// The property <c>title</c>, the title of the telegram.
9    ///</summary>
10   [JsonPropertyName("title")]
11   public required string Title { get; init; }
12
    ///<summary>
13   /// The property <c>status</c>, the status of the telegram.
14   ///</summary>
15   [JsonPropertyName("status")]
16   public required TelegramStatus Status { get; init; }
17
    ///<summary>
18   /// The property <c>dateTime</c>, the date and time of the telegram.
19   ///</summary>
20   [JsonPropertyName("dateTime")]
21   public required DateTimeOffset Time { get; init; }
22
    ///<summary>
23   /// The property <c>editorialOffice</c>, the editorial office of the telegram.
24   ///</summary>
25   [JsonPropertyName("editorialOffice")]
26   public required string EditorialOffice { get; init; }
27
28 }

```

```

29     ///<summary>
30     ///<summary>The property <c>publishingOffice</c>, the publishing office of the telegram.
31     ///</summary>
32     [JsonPropertyName("publishingOffice")]
33     public required string PublishingOffice { get; init; }
34 }
```

Listing A.4.22: EasonEetwViewer.Dmdata.Dtos/Telegram/XmlControl.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Telegram;
4  ///<summary>
5  /// Describes the type of a telegram.
6  ///</summary>
7
8  [JsonConverter(typeof(JsonStringEnumConverter<TelegramType>))]
9  public enum TelegramType
{
10
    ///<summary>
    ///<summary>The value <c>発表</c>, representing a new release.
13    ///</summary>
14    [JsonStringEnumMemberName(" 発表")]
15    Release,
16    ///<summary>
17    ///<summary>The value <c>訂正</c>, representing a correction.
18    ///</summary>
19    [JsonStringEnumMemberName(" 訂正")]
20    Correction,
21    ///<summary>
22    ///<summary>The value <c>遅延</c>, representing a delay.
23    ///</summary>
24    [JsonStringEnumMemberName(" 遅延")]
25    Delay,
26    ///<summary>
27    ///<summary>The value <c>取消</c>, representing a cancellation.
28    ///</summary>
29    [JsonStringEnumMemberName(" 取消")]
30    Cancel
31 }
```

Listing A.4.23: EasonEetwViewer.Dmdata.Dtos/TelegramType.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Dtos.Telegram;
4  ///<summary>
5  /// Represents the head information of a XML telegram.
6  ///</summary>
7  public record XmlHead
{
8
    ///<summary>
    ///<summary>The property <c>title</c>, the title of the telegram.
11    ///</summary>
12    [JsonPropertyName("title")]
13    public required string Title { get; init; }
14    ///<summary>
```

```
15  /// The property <c>eventId</c>, the target event ID for the telegram.  
16  /// </summary>  
17  /// <remarks>  
18  /// <see langword="null"/> when it has no target event.  
19  /// </remarks>  
20  [JsonPropertyName("eventId")]  
21  public required string? EventId { get; init; }  
22  /// <summary>  
23  /// The property <c>serial</c>, the serial number of the telegram for the event.  
24  /// </summary>  
25  /// <remarks>  
26  /// <see langword="null"/> when it has no target event.  
27  /// </remarks>  
28  [JsonPropertyName("serial")]  
29  public required string? SerialNo { get; init; }  
30  /// <summary>  
31  /// The property <c>headline</c>, the headline of the telegram.  
32  /// </summary>  
33  /// <remarks>  
34  /// <see langword="null"/> when no headline is provided.  
35  /// </remarks>  
36  [JsonPropertyName("headline")]  
37  public required string? Headline { get; init; }  
38  /// <summary>  
39  /// The property <c>infoKind</c>, the schema name for the XML telegram.  
40  /// </summary>  
41  [JsonPropertyName("infoKind")]  
42  public required string InfoKind { get; init; }  
43  /// <summary>  
44  /// The property <c>infoKindVersion</c>, the schema version for the XML telegram.  
45  /// </summary>  
46  [JsonPropertyName("infoKindVersion")]  
47  public required string InfoKindVersion { get; init; }  
48  /// <summary>  
49  /// The property <c>infoType</c>, the type of the telegram.  
50  /// </summary>  
51  [JsonPropertyName("infoType")]  
52  public required TelegramType InfoType { get; init; }  
53  /// <summary>  
54  /// The property <c>reportDateTime</c>, the release date and time of the telegram.  
55  /// </summary>  
56  [JsonPropertyName("reportDateTime")]  
57  public required DateTimeOffset ReportDateTime { get; init; }  
58  /// <summary>  
59  /// The property <c>targetDateTime</c>, the target date and time for the telegram.  
60  /// </summary>  
61  /// <remarks>  
62  /// <see langword="null"/> when it has no target date or time.  
63  /// </remarks>  
64  [JsonPropertyName("targetDateTime")]  
65  public required DateTimeOffset? TargetDateTime { get; init; }  
66  /// <summary>  
67  /// The property <c>targetDateTimeDubious</c>, the target date and time for the  
68  → telegram with dubiousness.  
69  /// </summary>  
70  /// <remarks>  
71  /// <see langword="null"/> when it has no target date or time, or when it has no  
→ dubiousness.  
/// </remarks>
```

```

72 [JsonPropertyName("targetDateTimeDubious")]
73 public string? TargetDateTimeDubious { get; init; }
74 /// <summary>
75 /// The property <c>targetDuration</c>, the target duration for the telegram.
76 /// </summary>
77 /// <remarks>
78 /// <see langword="null"/> when it has no target duration.
79 /// </remarks>
80 [JsonPropertyName("targetDuration")]
81 public string? TargetDuration { get; init; }
82 /// <summary>
83 /// The property <c>validDateTime</c>, the date and time the telegram is valid
84 → until.
85 /// </summary>
86 /// <remarks>
87 /// <see langword="null"/> when it has no target duration.
88 /// </remarks>
89 [JsonPropertyName("validDateTime")]
90 public DateTimeOffset? ValidDateTime { get; init; }
91 }
```

Listing A.4.24: EasonEetwViewer.Dmdata.Dtos/Telegram/XmlHead.cs

```

1 using System.Text.Json.Serialization;
2
3 namespace EasonEetwViewer.Dmdata.Dtos.Telegram;
4 /// <summary>
5 /// Represents the version of a JSON Telegram Schema.
6 /// </summary>
7 public record SchemaVersionInformation
8 {
9     /// <summary>
10    /// The property <c>type</c>, the type of the schema.
11    /// </summary>
12    [JsonPropertyName("type")]
13    public required string Type { get; init; }
14    /// <summary>
15    /// The property <c>version</c>, the version of the schema.
16    /// </summary>
17    [JsonPropertyName("version")]
18    public required string Version { get; init; }
19 }
```

Listing A.4.25: EasonEetwViewer.Dmdata.Dtos/Telegram/SchemaVersionInformation.cs

```

1 using System.Text.Json.Serialization;
2
3 namespace EasonEetwViewer.Dmdata.Dtos.Telegram;
4 /// <summary>
5 /// Describes the head and control of a XML report.
6 /// </summary>
7 public class XmlReport
8 {
9     /// <summary>
10    /// The property <c>head</c>, the head component of the XML report.
11    /// </summary>
12    [JsonPropertyName("head")]
13 }
```

```

13     public required XmlHead Head { get; init; }
14     /// <summary>
15     /// The property <c>control</c>, the control component of the XML report.
16     /// </summary>
17     [JsonPropertyName("control")]
18     public required XmlControl Control { get; init; }
19 }
```

Listing A.4.26: EasonEetwViewer.Dmdata.Dtos/Telegram/XmlReport.cs

```

1  using System.Text.Json.Serialization;
2
3 namespace EasonEetwViewer.Dmdata.Dtos.Telegram;
4
5     /// <summary>
6     /// Describes the status of a telegram.
7     /// </summary>
8     [JsonConverter(typeof(JsonStringEnumConverter<TelegramStatus>))]
9     public enum TelegramStatus
{
10
11     /// <summary>
12     /// The value <c>通常</c>, a normal telegram.
13     /// </summary>
14     [JsonStringEnumMemberName(" 通常")]
15     Normal,
16     /// <summary>
17     /// The value <c>訓練</c>, a practise telegram.
18     /// </summary>
19     [JsonStringEnumMemberName(" 訓練")]
20     Practise,
21     /// <summary>
22     /// The value <c>試験</c>, a test telegram.
23     /// </summary>
24     [JsonStringEnumMemberName(" 試験")]
25     Test
26 }
```

Listing A.4.27: EasonEetwViewer.Dmdata.Dtos/Telegram/TelegramStatus.cs

A.5 DM-D.S.S. Authentication

```

1  namespace EasonEetwViewer.Dmdata.Authentication.Options;
2  /// <summary>
3  /// Represents options for OAuth2.
4  /// </summary>
5  public sealed record OAuth2Options
{
6
7      /// <summary>
8      /// The string to be displayed on the webpage after successful authentication.
9      /// </summary>
10     public required string WebPageString { get; init; }
11     /// <summary>
12     /// The path to redirect to after successful authentication.
13     /// </summary>
14     public required string RedirectPath { get; init; }
15     /// <summary>
```

```

16     /// The base URI of HTTP requests.
17     /// </summary>
18     public required string BaseUri { get; init; }
19     /// <summary>
20     /// The Host for the requests.
21     /// </summary>
22     public required string Host { get; init; }
23     /// <summary>
24     /// The Client ID to be used.
25     /// </summary>
26     public required string ClientId { get; init; }
27     /// <summary>
28     /// The scopes that are specified for the authentication.
29     /// </summary>
30     public required IEnumerable<string> Scopes { get; init; }
31 }
```

Listing A.5.1: EasonEetwViewer.Dmdata.Authentication/Options/OAuth2Options.cs

```

1 namespace EasonEetwViewer.Dmdata.Authentication.Exceptions;
2     /// <summary>
3     /// Represents errors that occurs during OAuth2.
4     /// </summary>
5     public class OAuthException : Exception
6     {
7         /// <summary>
8         /// Instantiates a new instance of the <see cref="OAuthException"/> class.
9         /// </summary>
10        public OAuthException() { }
11        /// <summary>
12        /// Instantiates a new instance of the <see cref="OAuthException"/> class with a
13        /// → specified error message.
14        /// </summary>
15        /// <param name="message">The error message.</param>
16        public OAuthException(string message)
17            : base(message) { }
18        /// <summary>
19        /// Instantiates a new instance of the <see cref="OAuthException"/> class with a
20        /// → specified error message and an inner exception.
21        /// </summary>
22        /// <param name="message">The error message.</param>
23        /// <param name="inner">The inner exception.</param>
24        public OAuthException(string message, Exception inner)
25            : base(message, inner) { }
26 }
```

Listing A.5.2: EasonEetwViewer.Dmdata.Authentication/Exceptions/OAuthException.cs

```

1 using EasonEetwViewer.Dmdata.Authentication.Services;
2
3 namespace EasonEetwViewer.Dmdata.Authentication.Exceptions;
4     /// <summary>
5     /// Represents error by attempting to get authentication from a <see
6     /// → cref="NullAuthenticator"/>.
7     /// </summary>
8     public sealed class NullAuthenticatiorException : Exception
```

```

8  {
9   /// <summary>
10  /// Instantiates a new instance of the <see cref="NullAuthenticatiiorException"/>
11  /// class.
12  /// </summary>
13  public NullAuthenticatiiorException() { }
14  /// <summary>
15  /// Instantiates a new instance of the <see cref="NullAuthenticatiiorException"/>
16  /// class with a specified error message.
17  /// </summary>
18  /// <param name="message">The error message.</param>
19  public NullAuthenticatiiorException(string message)
20  : base(message) { }
21  /// <summary>
22  /// Instantiates a new instance of the <see cref="NullAuthenticatiiorException"/>
23  /// class with a specified error message and an inner exception.
24  /// </summary>
25  /// <param name="message">The error message.</param>
26  /// <param name="inner">The inner exception.</param>
27  public NullAuthenticatiiorException(string message, Exception inner)
28  : base(message, inner) { }
29 }
```

Listing A.5.3:

EasonEetwViewer.Dmdata.Authentication/Exceptions/NullAuthenticatiiorException.cs

```

1  namespace EasonEetwViewer.Dmdata.Authentication.Exceptions;
2  /// <summary>
3  /// Represents errors that occurs during OAuth2 JSON Parsing.
4  /// </summary>
5  public sealed class OAuthJsonException : OAuthException
6  {
7   /// <summary>
8   /// Instantiates a new instance of the <see cref="OAuthJsonException"/> class.
9   /// </summary>
10  public OAuthJsonException() { }
11  /// <summary>
12  /// Instantiates a new instance of the <see cref="OAuthJsonException"/> class with
13  /// a specified error message.
14  /// </summary>
15  /// <param name="message">The error message.</param>
16  public OAuthJsonException(string message)
17  : base(message) { }
18  /// <summary>
19  /// Instantiates a new instance of the <see cref="OAuthJsonException"/> class with
20  /// a specified error message and an inner exception.
21  /// </summary>
22  /// <param name="message">The error message.</param>
23  /// <param name="inner">The inner exception.</param>
24  public OAuthJsonException(string message, Exception inner)
25  : base(message, inner) { }
26 }
```

Listing A.5.4: EasonEetwViewer.Dmdata.Authentication/Exceptions/OAuthJsonException.cs

```

1  namespace EasonEetwViewer.Dmdata.Authentication.Exceptions;
2  /// <summary>
```

```

3  /// Represents security errors that occurs during OAuth2 key change.
4  /// </summary>
5  public sealed class OAuthSecurityException : OAuthException
6  {
7      /// <summary>
8      /// Instantiates a new instance of the <see cref="OAuthSecurityException"/> class.
9      /// </summary>
10     public OAuthSecurityException() { }
11    /// <summary>
12    /// Instantiates a new instance of the <see cref="OAuthSecurityException"/> class
13    /// with a specified error message.
14    /// </summary>
15    /// <param name="message">The error message.</param>
16    public OAuthSecurityException(string message)
17        : base(message) { }
18    /// <summary>
19    /// Instantiates a new instance of the <see cref="OAuthSecurityException"/> class
20    /// with a specified error message and an inner exception.
21    /// </summary>
22    /// <param name="message">The error message.</param>
23    /// <param name="inner">The inner exception.</param>
24    public OAuthSecurityException(string message, Exception inner)
25        : base(message, inner) { }
26 }
```

Listing A.5.5: EasonEetwViewer.Dmdata.Authentication/Exceptions/OAuthSecurityException.cs

```

1  namespace EasonEetwViewer.Dmdata.Authentication.Exceptions;
2  /// <summary>
3  /// Represents errors thrown by online authenticators that occurs during OAuth2 key
4  /// change.
5  /// </summary>
6  public sealed class OAuthErrorException : OAuthException
7  {
8      /// <summary>
9      /// Instantiates a new instance of the <see cref="OAuthErrorException"/> class.
10     /// </summary>
11     public OAuthErrorException() { }
12     /// <summary>
13     /// Instantiates a new instance of the <see cref="OAuthErrorException"/> class
14     /// with a specified error message.
15     /// </summary>
16     /// <param name="message">The error message.</param>
17     public OAuthErrorException(string message)
18         : base(message) { }
19     /// <summary>
20     /// Instantiates a new instance of the <see cref="OAuthErrorException"/> class
21     /// with a specified error message and an inner exception.
22     /// </summary>
23     /// <param name="message">The error message.</param>
24     /// <param name="inner">The inner exception.</param>
25     public OAuthErrorException(string message, Exception inner)
26         : base(message, inner) { }
27 }
```

Listing A.5.6: EasonEetwViewer.Dmdata.Authentication/Exceptions/OAuthErrorException.cs

```

1  using EasonEetwViewer.Dmdata.Authentication.Options;
2  using EasonEetwViewer.Dmdata.Authentication.Services;
3  using Microsoft.Extensions.DependencyInjection;
4  using Microsoft.Extensions.Logging;
5  using Microsoft.Extensions.Options;
6
7  namespace EasonEetwViewer.Dmdata.Authentication.Extensions;
8  ///<summary>
9  /// Provides extension methods for <see cref="IServiceCollection"/> to add
10 /// authentication wrapper.
11 /// </summary>
12 public static class AuthenticationHelperServiceCollectionExtensions
13 {
14     ///<summary>
15     /// Injects a <see cref="AuthenticationHelper"/> with setup from provided
16     /// file path.
17     /// </summary>
18     ///<param name="services">The instance of <see cref="IServiceCollection"/> for
19     /// the service to be injected.</param>
20     ///<param name="filePath">The file path to where the string is stored.</param>
21     ///<returns>The <see cref="IServiceCollection"/> where the service is injected,
22     /// for chained calls.</returns>
23     public static IServiceCollection AddAuthenticator(this IServiceCollection
24         services, string filePath)
25         => services.AddSingleton(sp
26             =>
27             {
28                 string fileContent;
29                 try
30                 {
31                     fileContent = File.ReadAllText(filePath);
32                 }
33                 catch
34                 {
35                     fileContent = string.Empty;
36                 }
37
38                 return AuthenticationHelper.FromString(
39                     filePath,
40                     fileContent,
41                     sp.GetRequiredService<ILogger<AuthenticationHelper>>(),
42                     sp.GetRequiredService<ILogger<OAuth2Helper>>(),
43                     sp.GetRequiredService<ILogger<OAuth2Authenticator>>(),
44                     sp.GetRequiredService<IOptions<OAuth2Options>>().Value);
45             });
46 }

```

Listing A.5.7: EasonEetwViewer.Dmdata.Authentication/Extensions/AuthenticationHelperServiceCollectionExtensions.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Authentication.Dtos;
4  ///<summary>
5  /// Represents an error in the response from an OAuth HTTP Request.
6  /// </summary>
7  internal sealed record Error
8  {

```

```

9   /// <summary>
10  /// The property <c>error</c>, the error message.
11  /// </summary>
12  [JsonPropertyName("error")]
13  public required string Short { get; init; }
14  /// <summary>
15  /// The property <c>error_description</c>, the error description.
16  /// </summary>
17  [JsonPropertyName("error_description")]
18  public required string Description { get; init; }
19 }
```

Listing A.5.8: EasonEetwViewer.Dmdata.Authentication/Dtos/Error.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Authentication.Dtos;
4
5  /// <summary>
6  /// Represents a reponse from OAuth2 server to refresh a token.
7  /// </summary>
8  internal record TokenRefresh
{
9
10  /// <summary>
11  /// The property <c>access_token</c>, the new access token acquired.
12  /// </summary>
13  [JsonPropertyName("access_token")]
14  public required string AccessToken { get; init; }
15  /// <summary>
16  /// The property <c>token_type</c>, the type of token. A constant <c>Bearer</c>.
17  /// </summary>
18  [JsonPropertyName("token_type")]
19  public string TokenType { get; } = "Bearer";
20  /// <summary>
21  /// The property <c>expires_in</c>, the validity of the token. A constant
22  ↳ <c>21600</c>.
23  /// </summary>
24  [JsonPropertyName("expires_in")]
25  public int Expiry { get; } = 21600;
26  /// <summary>
27  /// The property <c>scope</c>, the scopes that the token is valid in.
28  /// </summary>
29  [JsonPropertyName("scope")]
30  public required string Scope { get; init; }
}
```

Listing A.5.9: EasonEetwViewer.Dmdata.Authentication/Dtos/TokenRefresh.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Authentication.Dtos;
4
5  /// <summary>
6  /// Represents a reponse from OAuth2 server to request a pair of token.
7  /// </summary>
8  internal sealed record TokenRequest : TokenRefresh
{

```

```

10     /// <summary>
11     /// The property <c>refresh_token</c>, the refresh token acquired.
12     /// </summary>
13     [JsonPropertyName("refresh_token")]
14     public required string RefreshToken { get; init; }
15 }
```

Listing A.5.10: EasonEetwViewer.Dmdata.Authentication/Dtos/TokenRequest.cs

```

1 namespace EasonEetwViewer.Dmdata.Authentication.Abstractions;
2     /// <summary>
3     /// Represents the current state of authentication.
4     /// </summary>
5     public enum AuthenticationStatus
6     {
7         /// <summary>
8         /// With no authentication.
9         /// </summary>
10        Null = 0,
11        /// <summary>
12        /// Using an API Key.
13        /// </summary>
14        ApiKey = 1,
15        /// <summary>
16        /// Using OAuth.
17        /// </summary>
18        OAuth = 2
19 }
```

Listing A.5.11:
EasonEetwViewer.Dmdata.Authentication/Abstractions/AuthenticationStatus.cs

```

1 using System.Net.Http.Headers;
2
3 namespace EasonEetwViewer.Dmdata.Authentication.Abstractions;
4
5     /// <summary>
6     /// Describes the interface of an authenticator for API calls.
7     /// </summary>
8     internal interface IAuthenticator
9     {
10         /// <summary>
11         /// Returns an <see cref="AuthenticationHeaderValue"/> to be used in a HTTP
12         /// request.
13         /// </summary>
14         /// <returns>An <see cref="AuthenticationHeaderValue"/>.</returns>
15         Task<AuthenticationHeaderValue> GetAuthenticationHeaderAsync();
16 }
```

Listing A.5.12: EasonEetwViewer.Dmdata.Authentication/Abstractions/IAuthenticator.cs

```

1 using System.Net.Http.Headers;
2 using EasonEetwViewer.Dmdata.Authentication.Events;
3 using EasonEetwViewer.Dmdata.Authentication.Services;
4
```

```

5  namespace EasonEetwViewer.Dmdata.Authentication.Abstractions;
6  ///<summary>
7  /// Wraps around an <see cref="IAuthenticator"/> to provide authentication
8  /// functionalities.
9  ///</summary>
10 public interface IAuthenticationHelper
11 {
12     ///<summary>
13     /// The current authentication status.
14     ///</summary>
15     AuthenticationStatus AuthenticationStatus { get; }
16     ///<summary>
17     /// When the status of the authentication has changed.
18     ///</summary>
19     event EventHandler<AuthenticationChangedEventArgs>? StatusChanged;
20     ///<summary>
21     /// Gets an authenticator header value to be used for authentication.
22     ///</summary>
23     ///<returns>The authenticator header value to be used, or <see langword="null"/>
24     /// if failed.</returns>
25     Task<AuthenticationHeaderValue?> GetAuthenticationHeaderValueAsync();
26     ///<summary>
27     /// Acts when the authenticator is invalid.
28     ///</summary>
29     ///<returns>A <see cref="Task"/> object that represents the asynchronous
30     /// operation.</returns>
31     Task InvalidAuthenticatorAsync(string message);
32     ///<summary>
33     /// Set the authenticator to <see cref="ApiKeyAuthenticator"/> with the specified
34     /// API Key.
35     ///</summary>
36     ///<param name="apiKey">The API Key.</param>
37     ///<returns>A <see cref="Task"/> object that represents the asynchronous
38     /// operation.</returns>
39     Task SetApiKeyAsync(string apiKey);
40     ///<summary>
41     /// Set the authenticator to <see cref="OAuth2Authenticator"/> which requires
42     /// input from the browser.
43     ///</summary>
44     ///<returns>A <see cref="Task"/> object that represents the asynchronous
45     /// operation.</returns>
46     Task SetOAuthAsync();
47     ///<summary>
48     /// Unset the authenticator to <see cref="NullAuthenticator"/>.
49     ///</summary>
50     ///<returns>A <see cref="Task"/> object that represents the asynchronous
51     /// operation.</returns>
52     Task UnsetAuthenticatorAsync();
53 }
```

Listing A.5.13:

EasonEetwViewer.Dmdata.Authentication/IAuthenticationHelper.cs

```

1  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
2
3  namespace EasonEetwViewer.Dmdata.Authentication.Events;
4
5  ///<summary>
```

```

6  /// The event arguments for when the authentication status changes.
7  /// </summary>
8  public class AuthenticationStatusChangedEventArgs : EventArgs
9  {
10     /// <summary>
11     /// The new authentication status.
12     /// </summary>
13     public required AuthenticationStatus AuthenticationStatus { get; init; }
14     /// <summary>
15     /// The new authentication.
16     /// </summary>
17     public required string? Authentication { get; init; }
18 }
```

Listing A.5.14:

EasonEetwViewer.Dmdata.Authentication/Events/AuthenticationStatusChangedEventArgs.cs

```

1  using Microsoft.Extensions.Logging;
2
3  namespace EasonEetwViewer.Dmdata.Authentication.Services;
4  /// <summary>
5  /// Represents the log messages used in <see cref="OAuth2Helper"/>.
6  /// </summary>
7  internal static partial class OAuth2HelperLogs
8  {
9      /// <summary>
10     /// Log when instantiated.
11     /// </summary>
12     /// <param name="logger">The logger to be used.</param>
13     [LoggerMessage(
14         EventId = 0,
15         EventName = nameof(Instantiated),
16         Level = LogLevel.Information,
17         Message = "Instantiated.")]
18     public static partial void Instantiated(
19         this ILogger<OAuth2Helper> logger);
20
21     /// <summary>
22     /// Log when finding unused port.
23     /// </summary>
24     /// <param name="logger">The logger to be used.</param>
25     [LoggerMessage(
26         EventId = 1,
27         EventName = nameof(FindingUnusedPort),
28         Level = LogLevel.Debug,
29         Message = "Finding unused port.")]
30     public static partial void FindingUnusedPort(
31         this ILogger<OAuth2Helper> logger);
32
33     /// <summary>
34     /// Log when found unused port.
35     /// </summary>
36     /// <param name="logger">The logger to be used.</param>
37     /// <param name="port">The port that was found.</param>
38     ///
39     [LoggerMessage(
40         EventId = 2,
41         EventName = nameof(FoundUnusedPort),
```

```
42     Level = LogLevel.Information,
43     Message = "Found unused port: `'{Port}`.")]
44 public static partial void FoundUnusedPort(
45     this ILogger<OAuth2Helper> logger, int port);
46
47     /// <summary>
48     /// Log when starting a webpage on the user's browser.
49     /// </summary>
50     /// <param name="logger">The logger to be used.</param>
51     /// <param name="uri">The URI of the page that was started.</param>
52 [LoggerMessage(
53     EventId = 3,
54     EventName = nameof(StartingBrowser),
55     Level = LogLevel.Information,
56     Message = "Starting browser page: `'{Uri}`.")]
57 public static partial void StartingBrowser(
58     this ILogger<OAuth2Helper> logger, Uri uri);
59
60     /// <summary>
61     /// Log when response is received from the user's browser.
62     /// </summary>
63     /// <param name="logger">The logger to be used.</param>
64     /// <param name="uri">The URI of the page where response was received.</param>
65 [LoggerMessage(
66     EventId = 4,
67     EventName = nameof(ResponseReceived),
68     Level = LogLevel.Debug,
69     Message = "Response received: `'{Uri}`.")]
70 public static partial void ResponseReceived(
71     this ILogger<OAuth2Helper> logger, Uri uri);
72
73     /// <summary>
74     /// Log when an error message was received.
75     /// </summary>
76     /// <param name="logger">The logger to be used.</param>
77     /// <param name="error">The error message received.</param>
78 [LoggerMessage(
79     EventId = 5,
80     EventName = nameof(ErrorMessageReceived),
81     Level = LogLevel.Error,
82     Message = "Error Message received: `'{Error}`.")]
83 public static partial void ErrorMessageReceived(
84     this ILogger<OAuth2Helper> logger, string error);
85
86     /// <summary>
87     /// Log when the states do not match.
88     /// </summary>
89     /// <param name="logger">The logger to be used.</param>
90 [LoggerMessage(
91     EventId = 6,
92     EventName = nameof(StateDoesNotMatch),
93     Level = LogLevel.Error,
94     Message = "State received from authenticator does not match.")]
95 public static partial void StateDoesNotMatch(
96     this ILogger<OAuth2Helper> logger);
97
98     /// <summary>
99     /// Log when unable to parse JSON.
100    /// </summary>
```

```

101    /// <param name="logger">The logger to be used.</param>
102    /// <param name="json">The JSON string that was unable to parse.</param>
103    [LoggerMessage(
104        EventId = 7,
105        EventName = nameof(IncorrectJsonFormat),
106        Level = LogLevel.Error,
107        Message = "Unable to parse JSON from server: `'{Json}`.")]
108    public static partial void IncorrectJsonFormat(
109        this ILogger<OAuth2Helper> logger, string json);
110 }

```

Listing A.5.15: EasonEetwViewer.Dmdata.Authentication/Logging/OAuth2HelperLogs.cs

```

1  using Microsoft.Extensions.Logging;
2
3  namespace EasonEetwViewer.Dmdata.Authentication.Services;
4  /// <summary>
5  /// Represents the log messages used in <see cref="AuthenticationHelper"/>.
6  /// </summary>
7  internal static partial class AuthenticationWrapperLogs
8  {
9      /// <summary>
10     /// Log when instantiated.
11     /// </summary>
12     /// <param name="logger">The logger to be used.</param>
13     [LoggerMessage(
14         EventId = 0,
15         EventName = nameof(Instantiated),
16         Level = LogLevel.Information,
17         Message = "Instantiated.")]
18     public static partial void Instantiated(
19         this ILogger<AuthenticationHelper> logger);
20
21     /// <summary>
22     /// Log when changed to API Key.
23     /// </summary>
24     /// <param name="logger">The logger to be used.</param>
25     [LoggerMessage(
26         EventId = 1,
27         EventName = nameof(ChangedToApiKey),
28         Level = LogLevel.Information,
29         Message = "Authenticator changed to API Key.")]
20
20
30     public static partial void ChangedToApiKey(
31         this ILogger<AuthenticationHelper> logger);
32
33     /// <summary>
34     /// Log when unsetting authenticator.
35     /// </summary>
36     /// <param name="logger">The logger to be used.</param>
37     [LoggerMessage(
38         EventId = 2,
39         EventName = nameof(Unsetting),
40         Level = LogLevel.Debug,
41         Message = "Unsetting authenticator.")]
42     public static partial void Unsetting(
43         this ILogger<AuthenticationHelper> logger);
44
45     /// <summary>

```

```
46     /// Log when authenticator unset.  
47     /// </summary>  
48     /// <param name="logger">The logger to be used.</param>  
49     [LoggerMessage(  
50         EventId = 3,  
51         EventName = nameof(Unset),  
52         Level = LogLevel.Information,  
53         Message = "Authenticator unset.")]  
54     public static partial void Unset(  
55         this ILogger<AuthenticationHelper> logger);  
56  
57     /// <summary>  
58     /// Log when OAuth exception ignored.  
59     /// </summary>  
60     /// <param name="logger">The logger to be used.</param>  
61     /// <param name="exception">The message of the exception.</param>  
62     [LoggerMessage(  
63         EventId = 4,  
64         EventName = nameof(OAuthExceptionIgnored),  
65         Level = LogLevel.Warning,  
66         Message = "OAuth Exception: `'{Exception}`.")]  
67     public static partial void OAuthExceptionIgnored(  
68         this ILogger<AuthenticationHelper> logger, string exception);  
69  
70     /// <summary>  
71     /// Log when other exceptions ignored.  
72     /// </summary>  
73     /// <param name="logger">The logger to be used.</param>  
74     /// <param name="exception">The message of the exception.</param>  
75     [LoggerMessage(  
76         EventId = 5,  
77         EventName = nameof(OtherExceptionIgnored),  
78         Level = LogLevel.Error,  
79         Message = "Exception: `'{Exception}`.")]  
80     public static partial void OtherExceptionIgnored(  
81         this ILogger<AuthenticationHelper> logger, string exception);  
82  
83     /// <summary>  
84     /// Log when revoking OAuth 2 tokens.  
85     /// </summary>  
86     /// <param name="logger">The logger to be used.</param>  
87     [LoggerMessage(  
88         EventId = 6,  
89         EventName = nameof(RevokingOAuth2Token),  
90         Level = LogLevel.Information,  
91         Message = "Revoking OAuth 2 tokens.")]  
92     public static partial void RevokingOAuth2Token(  
93         this ILogger<AuthenticationHelper> logger);  
94  
95     /// <summary>  
96     /// Log when invalid authentication message.  
97     /// </summary>  
98     /// <param name="logger">The logger to be used.</param>  
99     /// <param name="message">The message of the error.</param>  
100    [LoggerMessage(  
101        EventId = 7,  
102        EventName = nameof(InvalidAuthentication),  
103        Level = LogLevel.Error,  
104        Message = "Invalid authentication message: `'{Message}`.")]
```

```

105     public static partial void InvalidAuthentication(
106         this ILogger<AuthenticationHelper> logger, string message);
107
108     /// <summary>
109     /// Log when failed to write to file.
110     /// </summary>
111     /// <param name="logger">The logger to be used.</param>
112     /// <param name="filePath">The file path to be written to.</param>
113     /// <param name="content">The content that was written.</param>
114     [LoggerMessage(
115         EventId = 8,
116         EventName = nameof(FailedToWriteToFile),
117         Level = LogLevel.Error,
118         Message = "Failed to write to file: `'{FilePath}` `{Content}`.")]
119     public static partial void FailedToWriteToFile(
120         this ILogger<AuthenticationHelper> logger, string filePath, string? content);
121
122     /// <summary>
123     /// Log when successfullt to write to file.
124     /// </summary>
125     /// <param name="logger">The logger to be used.</param>
126     /// <param name="filePath">The file path to be written to.</param>
127     /// <param name="content">The content that was written.</param>
128     [LoggerMessage(
129         EventId = 9,
130         EventName = nameof(SucceededToWriteToFile),
131         Level = LogLevel.Information,
132         Message = "Succeeded to write to file: `'{FilePath}` `{Content}`.")]
133     public static partial void SucceededToWriteToFile(
134         this ILogger<AuthenticationHelper> logger, string filePath, string? content);
135 }

```

Listing A.5.16: EasonEtwViewer.Dmdata.Authentication/Logging/AuthenticationHelperLogs.cs

```

1  using Microsoft.Extensions.Logging;
2
3  namespace EasonEtwViewer.Dmdata.Authentication.Services;
4  /// <summary>
5  /// Represents the log messages used in <see cref="OAuth2Authenticator"/>.
6  /// </summary>
7  internal static partial class OAuth2AuthenticatorLogs
8  {
9      /// <summary>
10     /// Log when instantiated.
11     /// </summary>
12     /// <param name="logger">The logger to be used.</param>
13     [LoggerMessage(
14         EventId = 0,
15         EventName = nameof(Instantiated),
16         Level = LogLevel.Information,
17         Message = "Instantiated.")]
18     public static partial void Instantiated(
19         this ILogger<OAuth2Authenticator> logger);
20
21     /// <summary>
22     /// Log when revoking token.
23     /// </summary>
24     /// <param name="logger">The logger to be used.</param>

```

```
25     /// <param name="token">The token to be revoked.</param>
26     [LoggerMessage(
27         EventId = 1,
28         EventName = nameof(RevokingToken),
29         Level = LogLevel.Debug,
30         Message = "Revoking token: `{{Token}}`")]
31     public static partial void RevokingToken(
32         this ILogger<OAuth2Authenticator> logger, string token);
33
34     /// <summary>
35     /// Log when token revoked.
36     /// </summary>
37     /// <param name="logger">The logger to be used.</param>
38     [LoggerMessage(
39         EventId = 2,
40         EventName = nameof(TokenRevoked),
41         Level = LogLevel.Information,
42         Message = "Token revoked.")]
43     public static partial void TokenRevoked(
44         this ILogger<OAuth2Authenticator> logger);
45
46     /// <summary>
47     /// Log when unable to parse JSON.
48     /// </summary>
49     /// <param name="logger">The logger to be used.</param>
50     /// <param name="json">The JSON string that was unable to parse.</param>
51     [LoggerMessage(
52         EventId = 3,
53         EventName = nameof(IncorrectJsonFormat),
54         Level = LogLevel.Error,
55         Message = "Unable to parse JSON from server: `{{Json}}`")]
56     public static partial void IncorrectJsonFormat(
57         this ILogger<OAuth2Authenticator> logger, string json);
58
59     /// <summary>
60     /// Log when failed to revoke token.
61     /// </summary>
62     /// <param name="logger">The logger to be used.</param>
63     /// <param name="errorMessage">The exception caught.</param>
64     [LoggerMessage(
65         EventId = 4,
66         EventName = nameof(TokenRevokeFailed),
67         Level = LogLevel.Error,
68         Message = "Unable to revoke token: `{{ErrorMessage}}`")]
69     public static partial void TokenRevokeFailed(
70         this ILogger<OAuth2Authenticator> logger, string errorMessage);
71
72     /// <summary>
73     /// Log when requesting new access token.
74     /// </summary>
75     /// <param name="logger">The logger to be used.</param>
76     [LoggerMessage(
77         EventId = 5,
78         EventName = nameof(RequestingNewAccessToken),
79         Level = LogLevel.Debug,
80         Message = "Requesting new access token.")]
81     public static partial void RequestingNewAccessToken(
82         this ILogger<OAuth2Authenticator> logger);
83
```

```

84     ///<summary>
85     /// Log when new access token acquired.
86     ///</summary>
87     ///<param name="logger">The logger to be used.</param>
88     [LoggerMessage(
89         EventId = 6,
90         EventName = nameof(NewAccessTokenAcquired),
91         Level = LogLevel.Information,
92         Message = "New access token acquired.")]
93     public static partial void NewAccessTokenAcquired(
94         this ILogger<OAuth2Authenticator> logger);
95
96     ///<summary>
97     /// Log when an exception was thrown and handled, and the program continues
98     /// execution.
99     ///</summary>
100    ///<param name="logger">The logger to be used.</param>
101    ///<param name="exception">The exception caught.</param>
102    [LoggerMessage(
103        EventId = 7,
104        EventName = nameof(IgnoredException),
105        Level = LogLevel.Warning,
106        Message = "Exception: `'{Exception}``.")]
107    public static partial void IgnoredException(
108        this ILogger<OAuth2Authenticator> logger, string exception);
109 }
```

Listing A.5.17: EasonEetwViewer.Dmdata.Authentication/Logging/OAuth2AuthenticatorLogs.cs

```

1  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
2  using EasonEetwViewer.Dmdata.Authentication.Options;
3  using Microsoft.Extensions.Logging;
4
5  namespace EasonEetwViewer.Dmdata.Authentication.Services;
6  ///<summary>
7  /// Provides <see cref="OAuth2Authenticator"/> as <see cref="IAuthenticator"/>.
8  ///</summary>
9  internal static class OAuth2Provider
10 {
11     ///<summary>
12     /// Creates a new instance of <see cref="OAuth2Authenticator"/> by requesting a
13     /// new refresh token and a new access token.
14     ///</summary>
15     ///<param name="options">The options to be used for OAuth.</param>
16     ///<param name="helperLogger">The logger for the OAuth helper.</param>
17     ///<param name="logger">The logger for the OAuth authenticator.</param>
18     ///<returns>An asynchronous task whose result is the authenticator
19     /// desired.</returns>
20     public static async Task<IAuthenticator> GetOAuth2Authenticator(OAuth2Options
21         options, ILogger<OAuth2Helper> helperLogger, ILogger<OAuth2Authenticator>
22         logger)
23     {
24         (string refreshToken, string accessToken) = await new OAuth2Helper(
25             options.ClientId,
26             string.Join(' ', options.Scopes),
27             options.BaseUri,
28             options.Host,
29             options.RedirectPath,
```

```

26     options.WebPageString,
27     helperLogger)
28     .GetRefreshTokenAsync();
29     return new OAuth2Authenticator(
30         options.ClientId,
31         options.BaseUri,
32         options.Host,
33         refreshToken,
34         accessToken,
35         logger);
36 }
37 /// <summary>
38 /// Creates a new instance of <see cref="OAuth2Authenticator"/> using the existing
39 /// → refresh token.
40 /// </summary>
41 /// <param name="options">The options to be used for OAuth.</param>
42 /// <param name="refreshToken">The refresh token to be used.</param>
43 /// <param name="logger">The logger for the OAuth authenticator.</param>
44 /// <returns>The authenticator desired.</returns>
45 public static IAuthenticator GetOAuth2Authenticator(OAuth2Options options, string
46     → refreshToken, ILogger<OAuth2Authenticator> logger)
47     => new OAuth2Authenticator(
48         options.ClientId,
49         options.BaseUri,
50         options.Host,
51         refreshToken,
52         null,
53         logger);
54 }
```

Listing A.5.18: EasonEetwViewer.Dmdata.Authentication/Services/OAuth2Provider.cs

```

1  using System.Net.Http.Headers;
2  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
3  using EasonEetwViewer.Dmdata.Authentication.Events;
4  using EasonEetwViewer.Dmdata.Authentication.Exceptions;
5  using EasonEetwViewer.Dmdata.Authentication.Options;
6  using Microsoft.Extensions.Logging;
7
8  namespace EasonEetwViewer.Dmdata.Authentication.Services;
9  /// <summary>
10 /// Default implementation of <see cref="IAuthenticationHelper"/>.
11 /// </summary>
12 internal sealed class AuthenticationHelper : IAuthenticationHelper
13 {
14     /// <summary>
15     /// The logger used for the current instance.
16     /// </summary>
17     private readonly ILogger<AuthenticationHelper> _logger;
18     /// <summary>
19     /// The logger used for the <see cref="OAuth2Authenticator"/>.
20     /// </summary>
21     private readonly ILogger<OAuth2Authenticator> _oAuthLogger;
22     /// <summary>
23     /// The logger used for the <see cref="OAuth2Helper"/>.
24     /// </summary>
25     private readonly ILogger<OAuth2Helper> _helperLogger;
26     /// <summary>
```

```
27     /// The options for OAuth 2.
28     /// </summary>
29     private readonly OAuth2Options _oAuth2Options;
30     /// <summary>
31     /// The <see cref="IAuthenticator"/> to be used for the authentication.
32     /// </summary>
33     private IAAuthenticator _authenticator;
34     /// <inheritdoc/>
35     public async Task<AuthenticationHeaderValue?> GetAuthenticationHeaderAsync()
36     {
37         if (AuthenticationStatus is AuthenticationStatus.Null)
38         {
39             return null;
40         }
41         else
42         {
43             try
44             {
45                 return await _authenticator.GetAuthenticationHeaderAsync();
46             }
47             catch (OAuthException ex)
48             {
49                 _logger.OAuthExceptionIgnored(ex.ToString());
50                 await UnsetAuthenticatorAsync();
51                 return null;
52             }
53         }
54     }
55     /// <inheritdoc/>
56     public override string? ToString()
57     => _authenticator.ToString();
58     /// <summary>
59     /// Creates a new instance of <see cref="AuthenticationHelper"/> from a string.
60     /// </summary>
61     /// <param name="filePath">The file path to write to when the authentication
62     → status is updated.</param>
63     /// <param name="str">The string to be used to create the instance from.</param>
64     /// <param name="logger">The logger to be used for the instance</param>
65     /// <param name="helperLogger">The logger to be used for the <see
66     → cref="OAuth2Helper"/>.</param>
67     /// <param name="oAuthLogger">The logger to be used for the <see
68     → cref="OAuth2Authenticator"/>.</param>
69     /// <param name="oAuth2Options">The options for OAuth 2.</param>
70     /// <returns>The new instance of <see cref="AuthenticationHelper"/>.</returns>
71     internal static IAuthenticationHelper FromString(
72         string filePath,
73         string? str,
74         ILogger<AuthenticationHelper> logger,
75         ILogger<OAuth2Helper> helperLogger,
76         ILogger<OAuth2Authenticator> oAuthLogger,
77         OAuth2Options oAuth2Options)
78     {
79         IAAuthenticator authenticator;
80         if (str is null)
81         {
82             authenticator = NullAuthenticator.Instance;
83         }
84         else if (str.StartsWith("AKe."))
85         {
86             authenticator = AKeAuthenticator.Instance;
87         }
88         else
89         {
90             authenticator = OAuth2Authenticator.Create(str);
91         }
92         return new AuthenticationHelper(authenticator, logger, helperLogger, oAuthLogger, oAuth2Options);
93     }
```

```
83         authenticator = new ApiKeyAuthenticator(str);
84     }
85     else if (str.StartsWith("ARh."))
86     {
87         try
88     {
89         authenticator = OAuth2Provider.GetOAuth2Authenticator(oAuth2Options,
90             → str, oAuthLogger);
91     }
92     catch (OAuthException ex)
93     {
94         logger.OAuthExceptionIgnored(ex.ToString());
95         authenticator = NullAuthenticator.Instance;
96     }
97     catch (Exception ex)
98     {
99         logger.OtherExceptionIgnored(ex.ToString());
100        authenticator = NullAuthenticator.Instance;
101    }
102 }
103 {
104     authenticator = NullAuthenticator.Instance;
105 }
106
107 AuthenticationHelper authenticationHelper = new(
108     authenticator,
109     logger,
110     helperLogger,
111     oAuthLogger,
112     oAuth2Options);
113 authenticationHelper.StatusChanged += (sender, e) =>
114 {
115     string? content = e.Authentication;
116     try
117     {
118         File.WriteAllText(filePath, content);
119         logger.SucceededToWriteToFile(filePath, content);
120     }
121     catch (Exception)
122     {
123         logger FailedToWriteToFile(filePath, content);
124     }
125 };
126     return authenticationHelper;
127 }
128 /// <inheritdoc/>
129 public AuthenticationStatus AuthenticationStatus
130     => _authenticator is NullAuthenticator
131         ? AuthenticationStatus.Null
132         : _authenticator is ApiKeyAuthenticator
133             ? AuthenticationStatus.ApiKey
134             : AuthenticationStatus.OAuth;
135
136 /// <summary>
137 /// Creates a new instance of <see cref="AuthenticationHelper"/> with the options
138     → provided..
139 /// </summary>
140 /// <param name="currentAuthenticator">The current <see cref="IAuthenticator"/> to
141     → be used.</param>
```

```
139     /// <param name="logger">The logger to be used for the instance</param>
140     /// <param name="helperLogger">The logger to be used for the <see
141     →   cref="OAuth2Helper"/>.</param>
142     /// <param name="oAuthLogger">The logger to be used for the <see
143     →   cref="OAuth2Authenticator"/>.</param>
144     /// <param name="oAuth2Options">The options for OAuth 2.</param>
145     private AuthenticationHelper(
146         IAuthenticator currentAuthenticator,
147         ILogger<AuthenticationHelper> logger,
148         ILogger<OAuth2Helper> helperLogger,
149         ILogger<OAuth2Authenticator> oAuthLogger,
150         OAuth2Options oAuth2Options)
151     {
152         _authenticator = currentAuthenticator;
153         _logger = logger;
154         _helperLogger = helperLogger;
155         _oAuthLogger = oAuthLogger;
156         _oAuth2Options = oAuth2Options;
157         _logger.Instantiated();
158     }
159     /// <inheritdoc/>
160     public async Task SetOAuthAsync()
161     {
162         await UnsetAuthenticatorAsync();
163         try
164         {
165             _authenticator = await
166                 OAuth2Provider.GetOAuth2Authenticator(_oAuth2Options, _helperLogger,
167                 _oAuthLogger);
168         }
169         catch (OAuthException ex)
170         {
171             _logger.OAuthExceptionIgnored(ex.ToString());
172         }
173         catch (Exception ex)
174         {
175             _logger.OtherExceptionIgnored(ex.ToString());
176         }
177         StatusChanged?.Invoke(this, new()
178         {
179             AuthenticationStatus = AuthenticationStatus,
180             Authentication = ToString()
181         });
182     }
183     /// <inheritdoc/>
184     public async Task SetApiKeyAsync(string apiKey)
185     {
186         await UnsetAuthenticatorAsync();
187         _authenticator = new ApiKeyAuthenticator(apiKey);
188         _logger.ChangedToApiKey();
189         StatusChanged?.Invoke(this, new()
190         {
191             AuthenticationStatus = AuthenticationStatus,
192             Authentication = ToString()
193         });
194     }
195     /// <inheritdoc/>
196     public async Task InvalidAuthenticatorAsync(string message)
```

```

194     {
195         _logger.InvalidAuthentication(message);
196         await UnsetAuthenticatorAsync();
197     }
198 /// <inheritdoc/>
199 public async Task UnsetAuthenticatorAsync()
200 {
201     _logger.Unsetting();
202     if (_authenticator is OAuth2Authenticator oAuth)
203     {
204         try
205         {
206             _logger.RevokingOAuth2Token();
207             await oAuth.RevokeTokens();
208         }
209         catch (OAuthException ex)
210         {
211             _logger.OAuthExceptionIgnored(ex.ToString());
212         }
213         catch (Exception ex)
214         {
215             _logger.OtherExceptionIgnored(ex.ToString());
216         }
217     }
218
219     _authenticator = NullAuthenticator.Instance;
220     _logger.Unset();
221     StatusChanged?.Invoke(this, new()
222     {
223         AuthenticationStatus = AuthenticationStatus,
224         Authentication = ToString()
225     });
226 }
227 /// <inheritdoc/>
228 public event EventHandler<AuthenticationStatusChangedEventArgs>? StatusChanged;
229 }
```

Listing A.5.19: EasonEetwViewer.Dmdata.Authentication/Services/AuthenticationHelper.cs

```

1  using System.Net.Http.Headers;
2
3  namespace EasonEetwViewer.Dmdata.Authentication.Services;
4  /// <summary>
5  /// Provides shared methods where OAuth 2.0 requires.
6  /// </summary>
7  internal static class OAuth2SharedMethod
8  {
9      /// <summary>
10     /// Generates an HTTP POST Request using the specified URI and parameters, and
11     /// with media type <c>application/x-www-form-urlencoded</c>.
12     /// </summary>
13     /// <param name="requestUri">The URI of the POST Request.</param>
14     /// <param name="requestParams">The Parameters of the POST Request.-1</param>
15     /// <param name="host">The host for the request.</param>
16     /// <returns>The generated <see cref="HttpRequestMessage"/>.</returns>
17     public static HttpRequestMessage GeneratePostRequest(string requestUri,
18         Dictionary<string, string> requestParams, string host)
19     {
```

```

18     FormUrlEncodedContent content = new(requestParams);
19     HttpRequestMessage request = new(HttpMethod.Post, requestUri)
20     {
21         Content = content
22     };
23     request.Headers.Host = host;
24     request.Content.Headers.ContentType = new
25         MediaTypeHeaderValue("application/x-www-form-urlencoded");
26     return request;
27 }

```

Listing A.5.20: EasonEetwViewer.Dmdata.Authentication/Services/OAuth2SharedMethod.cs

```

1  using System.Net.Http.Headers;
2  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
3  using EasonEetwViewer.Dmdata.Authentication.Exceptions;
4
5  namespace EasonEetwViewer.Dmdata.Authentication.Services;
6  ///<summary>
7  /// Represents an authenticator without authentication.
8  ///</summary>
9  internal sealed class NullAuthenticator : IAuthenticator
10 {
11     ///<inheritdoc/>
12     ///<exception cref="NullAuthenticatiorException">This class does not support this
13     => throw new NullAuthenticatiorException();
14     ///<inheritdoc/>
15     public override string? ToString()
16     => null;
17     ///<summary>
18     /// Creates a new instance of <see cref="NullAuthenticator"/>.
19     ///</summary>
20     private NullAuthenticator() { }
21     ///<summary>
22     /// The shared instance of <see cref="NullAuthenticator"/>.
23     ///</summary>
24     public static IAuthenticator Instance { get; } = new NullAuthenticator();
25 }

```

Listing A.5.21: EasonEetwViewer.Dmdata.Authentication/Services/NullAuthenticator.cs

```

1  using System.Collections.Specialized;
2  using System.Diagnostics;
3  using System.Net;
4  using System.Net.Sockets;
5  using System.Security.Cryptography;
6  using System.Text;
7  using System.Text.Json;
8  using System.Web;
9  using EasonEetwViewer.Dmdata.Authentication.Dtos;
10 using EasonEetwViewer.Dmdata.Authentication.Exceptions;
11 using Microsoft.Extensions.Logging;
12
13 namespace EasonEetwViewer.Dmdata.Authentication.Services;

```

```
14  ///<summary>
15  /// A helper which gets the refresh token and access token for OAuth 2.0.
16  ///</summary>
17  internal sealed class OAuth2Helper
18  {
19      ///<summary>
20      /// Create a new instance of the class <see cref="OAuth2Helper"/> using the given
21      /// parameters.
22      ///</summary>
23      ///<param name="clientId">The client ID for OAuth.</param>
24      ///<param name="scopeString">The string containing the scopes to be
25      /// requested.</param>
26      ///<param name="baseUri">The base URI of requests.</param>
27      ///<param name="host">The host specified in the requests.</param>
28      ///<param name="redirectPath">The relative path used to communicate grant
29      /// codes.</param>
30      ///<param name="webPageString">The webpage contents to display after a successful
31      /// authentication.</param>
32      ///<param name="logger">The logger to be used for logging.</param>
33  public OAuth2Helper(
34      string clientId,
35      string scopeString,
36      string baseUri,
37      string host,
38      string redirectPath,
39      string webPageString,
40      ILogger<OAuth2Helper> logger)
41  {
42      _client = new() { BaseAddress = new(baseUri) };
43      _clientId = clientId;
44      _scopeString = scopeString;
45      _baseUri = baseUri;
46      _host = host;
47      _redirectPath = redirectPath;
48      _webPageString = webPageString;
49      _logger = logger;
50      _logger.Instantiated();
51  }
52  ///<summary>
53  /// The length of a code verifier.
54  ///</summary>
55  private const int _codeLength = 64;
56  ///<summary>
57  /// The length of a state.
58  ///</summary>
59  private const int _stateLength = 32;
60  ///<summary>
61  /// The allowed characters when generating a State or a Code Verifier.
62  ///</summary>
63  private const string _allowedChars =
64      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
65  ///<summary>
66  /// The HTTP Client to be used.
67  ///</summary>
68  private readonly HttpClient _client;
69  ///<summary>
70  /// The client ID for OAuth.
71  ///</summary>
72  private readonly string _clientId;
```

```
68     /// <summary>
69     /// The string containing the scopes to be requested.
70     /// </summary>
71     private readonly string _scopeString;
72     /// <summary>
73     /// The base URI of requests.
74     /// </summary>
75     private readonly string _baseUri;
76     /// <summary>
77     /// The host specified in the requests.
78     /// </summary>
79     private readonly string _host;
80     /// <summary>
81     /// The relative path used to communicate grant codes.
82     /// </summary>
83     private readonly string _redirectPath;
84     /// <summary>
85     /// The webpage contents to display after a successful authentication.
86     /// </summary>
87     private readonly string _webPageString;
88     /// <summary>
89     /// The logger to be used for logging.
90     /// </summary>
91     private readonly ILogger<OAuth2Helper> _logger;
92
93     // Modified from
94     //→ https://nicolaiarocci.com/how-to-implement-pkce-code-challenge-in-csharp/
95     /// <summary>
96     /// Encodes a code verifier to a code challenge use SHA256 and Base 64 encoding.
97     /// </summary>
98     /// <param name="verifier">The code verifier.</param>
99     /// <returns>The code challenge corresponding to the code verifier.</returns>
100    private static string VerifierToChallenge(string verifier)
101        => Convert.ToBase64String(SHA256.HashData(Encoding.UTF8.GetBytes(verifier)))
102            .Replace("+", "-")
103            .Replace("/", "_")
104            .TrimEnd('=');
105
106    // https://stackoverflow.com/a/150974/
107    /// <summary>
108    /// Finds an unused port on the local IP address <c>localhost</c>.
109    /// </summary>
110    /// <returns>The port that is found to be unused.</returns>
111    private static int GetUnusedPort()
112    {
113        TcpListener listener = new(IPAddress.Loopback, 0);
114        listener.Start();
115        int port = ((IPEndPoint)listener.LocalEndpoint).Port;
116        listener.Stop();
117        return port;
118    }
119
120    /// <summary>
121    /// Generates a grant code query parameter with the given parameters.
122    /// </summary>
123    /// <param name="redirectUri">The redirect URI.</param>
124    /// <param name="state">The state code.</param>
125    /// <param name="codeVerifier">The code verifier.</param>
126    /// <returns></returns>
```

```

126     private NameValueCollection GenerateGrantCodeQueryParameters(Uri redirectUri,
127         ↪ string state, string codeVerifier)
128     {
129         NameValueCollection queryParams = HttpUtility.ParseQueryString(string.Empty);
130         queryParams["client_id"] = _clientId;
131         queryParams["response_type"] = "code";
132         queryParams["redirect_uri"] = redirectUri.ToString();
133         queryParams["response_mode"] = "query";
134         queryParams["scope"] = _scopeString;
135         queryParams["state"] = state;
136         queryParams["code_challenge"] = VerifierToChallenge(codeVerifier);
137         queryParams["code_challenge_method"] = "S256";
138         return queryParams;
139     }
140
141     /// <summary>
142     /// Gets the grant code by executing on the relevant <see cref="Uri"/>s.
143     /// </summary>
144     /// <param name="redirectUri">The <see cref="Uri"/> to listen from.</param>
145     /// <param name="browserUri">The <see cref="Uri"/> to redirect the user
146     ↪ to.</param>
147     /// <returns>A pair of grant code and the request state obtained.</returns>
148     /// <exception cref="OAuthErrorException">When there is an error in the
149     ↪ response.</exception>
150     private async Task<(string grantCode, string requestState)> GetGrantCodeAsync(Uri
151         ↪ redirectUri, Uri browserUri)
152     {
153         using HttpListener listener = new();
154         listener.Prefixes.Add(redirectUri.ToString());
155         listener.Start();
156
157         _logger.StartingBrowser(browserUri);
158         // https://stackoverflow.com/a/61035650/
159         _ = Process.Start(new ProcessStartInfo
160         {
161             FileName = browserUri.ToString(),
162             UseShellExecute = true
163         });
164
165         HttpListenerContext context = await listener.GetContextAsync();
166         Uri responseUri = context.Request.Url!;
167         NameValueCollection query = HttpUtility.ParseQueryString(responseUri.Query);
168         _logger.ResponseReceived(responseUri);
169
170         byte[] buffer = Encoding.UTF8.GetBytes(_webPageString);
171         context.Response.ContentLength64 = buffer.Length;
172         await context.Response.OutputStream.WriteAsync(buffer.AsMemory(0,
173             ↪ buffer.Length));
174         context.Response.OutputStream.Close();
175         listener.Stop();
176
177         if (query["error"] is not null)
178         {
179             _logger.ErrorMessageReceived(query["error"]!);
180             throw new OAuthErrorException(query["error"]!);
181         }
182         else
183         {
184             return (query["code"]!, query["state"]!);
185         }
186     }

```

```
180     }
181 }
182 /// <summary>
183 /// Generates a refresh token request parameter with the given parameters.
184 /// </summary>
185 /// <param name="grantCode">The grant code.</param>
186 /// <param name="redirectUri">The redirect URI.</param>
187 /// <param name="codeVerifier">The code verifier.</param>
188 /// <returns>A dictionary which represents the parameters to be used.</returns>
189 private Dictionary<string, string> GenerateRefreshTokenRequestParameter(string
190     → grantCode, Uri redirectUri, string codeVerifier)
191     => new(){
192         { "client_id", _clientId },
193         { "grant_type", "authorization_code" },
194         { "code", grantCode },
195         { "redirect_uri", redirectUri.ToString() },
196         { "code_verifier", codeVerifier}
197     };
198 /// <summary>
199 /// Gets a new refresh token by prompting the user to login on a browser.
200 /// </summary>
201 /// <returns>A pair of tokens, the refresh token and the access token.</returns>
202 /// <exception cref="OAuthSecurityException">When the state returned by the state
203     → is not in accordance with the one sent.</exception>
204 /// <exception cref="OAuthJsonException">When the returned data cannot be parsed
205     → to JSON successfully.</exception>
206 /// <exception cref="OAuthErrorException">When there is an error in the
207     → response.</exception>
208 public async Task<(string refreshToken, string accessToken)>
209     → GetRefreshTokenAsync()
210 {
211     string state = RandomNumberGenerator.GetString(_allowedChars, _stateLength);
212     string codeVerifier = RandomNumberGenerator.GetString(_allowedChars,
213     → _codeLength);
214
215     _logger.FindingUnusedPort();
216     int port = GetUnusedPort();
217     _logger.FoundUnusedPort(port);
218
219     Uri redirectUri = new UriBuilder(IPAddress.Loopback.ToString())
220     {
221         Port = port,
222         Path = _redirectPath
223     }.Uri;
224
225     Uri browserUri = new UriBuilder(new Uri(new Uri(_baseUri), "auth"))
226     {
227         Query = GenerateGrantCodeQueryParameters(redirectUri, state,
228             → codeVerifier).ToString()
229     }
230     .Uri;
231
232     (string grantCode, string requestState) = await
233         → GetGrantCodeAsync(redirectUri, browserUri);
234
235     if (requestState != state)
236     {
237         _logger.StateDoesNotMatch();
238         throw new OAuthSecurityException("The states do not match.");
239 }
```

```

231     }
232
233     Dictionary<string, string> requestParams =
234         → GenerateRefreshTokenRequestParameter(grantCode, redirectUri,
235         → codeVerifier);
236     HttpRequestMessage request = OAuth2SharedMethod.GeneratePostRequest("token",
237         → requestParams, _host);
238     HttpResponseMessage response = await _client.SendAsync(request);
239
240     string responseBody = await response.Content.ReadAsStringAsync();
241
242     try
243     {
244         if (response.IsSuccessStatusCode)
245         {
246             TokenRequest token =
247                 → JsonSerializer.Deserialize<TokenRequest>(responseBody)
248                 ?? throw new OAuthJsonException($"Cannot deserialise:
249                 → {responseBody}.");
250
251             return (token.RefreshToken, token.AccessToken);
252         }
253         else
254         {
255             Error error = JsonSerializer.Deserialize<Error>(responseBody)
256                 ?? throw new OAuthJsonException($"Cannot deserialise:
257                 → {responseBody}.");
258             _logger.ErrorMessageReceived(error.Description);
259             throw new OAuthErrorException($"'{error.Short}' {error.Description}");
260         }
261     }
262     catch (OAuthJsonException)
263     {
264         _logger.IncorrectJsonFormat(responseBody);
265         throw;
266     }
267     catch (JsonException ex)
268     {
269         _logger.IncorrectJsonFormat(responseBody);
270         throw new OAuthJsonException($"Cannot deserialise: {responseBody}", ex);
271     }
272 }

```

Listing A.5.22: EasonEetwViewer.Dmdata.Authentication/Services/OAuth2Helper.cs

```

1  using System.Net.Http.Headers;
2  using System.Text.Json;
3  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
4  using EasonEetwViewer.Dmdata.Authentication.Dtos;
5  using EasonEetwViewer.Dmdata.Authentication.Exceptions;
6  using Microsoft.Extensions.Logging;
7
8  namespace EasonEetwViewer.Dmdata.Authentication.Services;
9
10 /// <summary>
11 /// An implementation of <see cref="IAuthenticator"/>, using OAuth 2.0 to
12     → authenticate.

```

```
12  /// </summary>
13  internal sealed class OAuth2Authenticator : IAuthenticator
14  {
15      /// <summary>
16      /// The access token.
17      /// </summary>
18      private string _accessToken;
19      /// <summary>
20      /// The time of expiry for the access token.
21      /// </summary>
22      private DateTimeOffset _accessTokenExpiry;
23      /// <summary>
24      /// The time span for the validity of an access token.
25      /// </summary>
26      private readonly TimeSpan _accessTokenValidity = TimeSpan.FromHours(6);
27      /// <summary>
28      /// The refresh token to be used.
29      /// </summary>
30      private readonly string _refreshToken;
31      /// <summary>
32      /// The Client ID for OAuth 2.0
33      /// </summary>
34      private readonly string _clientId;
35      /// <summary>
36      /// The Host indicated in POST requests.
37      /// </summary>
38      private readonly string _host;
39      /// <summary>
40      /// The Base URI for requests.
41      /// </summary>
42      private readonly Uri _base;
43      /// <summary>
44      /// The HttpClient used for requests.
45      /// </summary>
46      private readonly HttpClient _httpClient;
47      /// <summary>
48      /// The logger to be used for logging.
49      /// </summary>
50      private readonly ILogger<OAuth2Authenticator> _logger;
51      /// <summary>
52      /// Creates a new instance of the class with a set of given parameters.
53      /// </summary>
54      /// <param name="clientId">The Client ID for OAuth 2.0.</param>
55      /// <param name="basePath">The Base Path for HTTP requests.</param>
56      /// <param name="host">The Host indicated in POST requests.</param>
57      /// <param name="refreshToken">The refresh token to be used.</param>
58      /// <param name="accessToken">The access token to be used, or <see
59      → langword="null"/> if unprovided.</param>
60      /// <param name="logger">The logger to be used.</param>
61      public OAuth2Authenticator(string clientId, string basePath, string host, string
62      → refreshToken, string? accessToken, ILogger<OAuth2Authenticator> logger)
63      {
64          if (!refreshToken.StartsWith("ARh."))
65          {
66              throw new ArgumentException($"'{refreshToken}' does not match format of
67              → refresh token.", nameof(refreshToken));
68          }
69
70          if (accessToken is not null && !accessToken.StartsWith("ATn."))
71      }
```

```
68
69     {
70         throw new ArgumentException($"'{accessToken}' does not match format of
71             ← access token.", nameof(accessToken));
72     }
73
74     _clientId = clientId;
75     _host = host;
76     _base = new(basePath);
77     _httpClient = new()
78     {
79         BaseAddress = _base
80     };
81     _refreshToken = refreshToken;
82     _logger = logger;
83     _accessToken = accessToken ?? Task.Run(RenewAccessTokenAsync).Result;
84     _accessTokenExpiry = DateTimeOffset.Now + _accessTokenValidity;
85     _logger.Instantiated();
86 }
87
88 /// <inheritdoc/>
89 public async Task<AuthenticationHeaderValue> GetAuthenticationHeaderAsync()
90     => new("Bearer", await CheckAccessTokenAsync());
91
92 /// <inheritdoc/>
93 public override string ToString()
94     => _refreshToken;
95
96 /// <summary>
97 /// Checks if the access token is valid, and refresh its validity where necessary.
98 /// </summary>
99 /// <returns>A <see cref="Task"/> object that represents the asynchronous
100    ← operation.</returns>
101 private async Task<string> CheckAccessTokenAsync()
102     => _accessTokenExpiry > DateTimeOffset.Now
103         ? _accessToken
104         : await RenewAccessTokenAsync();
105
106 /// <summary>
107 /// Gets a new access token.
108 /// </summary>
109 /// <returns>A <see cref="Task"/> object that represents the asynchronous
110    ← operation.</returns>
111 /// <exception cref="OAuthJsonException">When the returned data cannot be parsed
112    ← to JSON successfully.</exception>
113 /// <exception cref="OAuthErrorException">When there is an error in the
114    ← response.</exception>
115 private async Task<string> RenewAccessTokenAsync()
116 {
117     if (_accessToken is not null)
118     {
119         try
120         {
121             await RevokeTokenAsync(_accessToken);
122         }
123         catch (Exception ex)
124         {
125             _logger.IgnoredException(ex.ToString());
126         }
127     }
128 }
```

```
122     _accessTokenExpiry = DateTimeOffset.MinValue;
123 }
124
125     _logger.RequestingNewAccessToken();
126     Dictionary<string, string> requestParams = new(){
127         { "client_id", _clientId },
128         { "grant_type", "refresh_token" },
129         { "refresh_token", _refreshToken }
130     };
131     HttpRequestMessage request = OAuth2SharedMethod.GeneratePostRequest("token",
132         → requestParams, _host);
133
134     HttpResponseMessage response = await _httpClient.SendAsync(request);
135
136     if (response.IsSuccessStatusCode)
137     {
138         string responseBody = await response.Content.ReadAsStringAsync();
139         try
140         {
141             TokenRefresh token =
142                 → JsonSerializer.Deserialize<TokenRefresh>(responseBody)
143                 ?? throw new OAuthJsonException($"Cannot deserialise:
144                     → {responseBody}");
145             _accessTokenExpiry = DateTimeOffset.Now + _accessTokenValidity;
146             _accessToken = token.AccessToken;
147             _logger.NewAccessTokenAcquired();
148             return _accessToken;
149         }
150         catch (OAuthJsonException)
151         {
152             _logger.IncorrectJsonFormat(responseBody);
153             throw;
154         }
155         catch (JsonException ex)
156         {
157             _logger.IncorrectJsonFormat(responseBody);
158             throw new OAuthJsonException($"Cannot deserialise: {responseBody}",
159                 → ex);
160         }
161     }
162     else
163     {
164         string responseBody = await response.Content.ReadAsStringAsync();
165         try
166         {
167             Error error = JsonSerializer.Deserialize<Error>(responseBody)
168                 ?? throw new OAuthJsonException($"Cannot deserialise:
169                     → {responseBody}");
170             _logger.TokenRevokeFailed(error.Description);
171             throw new OAuthErrorException($"{{error.Short}} {{error.Description}}");
172         }
173         catch (OAuthJsonException)
174         {
175             _logger.IncorrectJsonFormat(responseBody);
176             throw;
177         }
178         catch (JsonException ex)
179         {
180             _logger.IncorrectJsonFormat(responseBody);
```

```
176             throw new OAuthJsonException($"Cannot deserialise: {responseBody}",
177                                         ← ex);
178         }
179     }
180
181     /// <summary>
182     /// Revokes the specified token.
183     /// </summary>
184     /// <param name="token">The token to be revoked.</param>
185     /// <returns>A <see cref="Task"/> object that represents the asynchronous
186     → operation.</returns>
187     /// <exception cref="OAuthJsonException">When the returned data cannot be parsed
188     → to JSON successfully.</exception>
189     /// <exception cref="OAuthErrorException">When there is an error in the
190     → response.</exception>
191     private async Task RevokeTokenAsync(string token)
192     {
193         Dictionary<string, string> requestParams = new(){
194             { "client_id", _clientId },
195             { "token", token }
196         };
197         _logger.RevokingToken(token);
198         HttpRequestMessage request = OAuth2SharedMethod.GeneratePostRequest("revoke",
199                                     ← requestParams, _host);
200         HttpResponseMessage response = await _httpClient.SendAsync(request);
201         if (response.IsSuccessStatusCode)
202         {
203             _logger.TokenRevoked();
204         }
205         else
206         {
207             string responseBody = await response.Content.ReadAsStringAsync();
208             try
209             {
210                 Error error = JsonSerializer.Deserialize<Error>(responseBody)
211                 ?? throw new OAuthJsonException($"Cannot deserialise:
212                                         ← {responseBody}");
213                 _logger.TokenRevokeFailed(error.Description);
214                 throw new OAuthErrorException($"'{error.Short} {error.Description}'");
215             }
216             catch (OAuthJsonException)
217             {
218                 _logger.IncorrectJsonFormat(responseBody);
219                 throw;
220             }
221             catch (JsonException ex)
222             {
223                 _logger.IncorrectJsonFormat(responseBody);
224                 throw new OAuthJsonException($"Cannot deserialise: {responseBody}",
225                                         ← ex);
226             }
227         }
228     }
229
230     /// <summary>
231     /// Revokes the access token and the refresh token.
232     /// </summary>
```

```

228     /// <returns>A <see cref="Task"/> object that represents the asynchronous
229     // operation.</returns>
230     public async Task RevokeTokens()
231     {
232         Task revokeAccessToken = RevokeTokenAsync(_accessToken);
233         Task revokeRefreshToken = RevokeTokenAsync(_refreshToken);
234         await Task.WhenAll(revokeAccessToken, revokeRefreshToken);
235     }

```

Listing A.5.23: EasonEetwViewer.Dmdata.Authentication/Services/OAuth2Authenticator.cs

```

1  using System.Net.Http.Headers;
2  using System.Text;
3  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
4
5  namespace EasonEetwViewer.Dmdata.Authentication.Services;
6
7  /// <summary>
8  /// Implementation of <see cref="IAuthenticator"/>, using API Keys to authenticate.
9  /// </summary>
10 internal sealed class ApiKeyAuthenticator : IAAuthenticator
11 {
12     /// <summary>
13     /// The header for HTTP request.
14     /// </summary>
15     private readonly AuthenticationHeaderValue _header;
16     /// <summary>
17     /// The API Key.
18     /// </summary>
19     private readonly string _apiKey;
20     /// <inheritdoc/>
21     public Task<AuthenticationHeaderValue> GetAuthenticationHeaderAsync()
22         => Task.FromResult(_header);
23     /// <inheritdoc/>
24     public override string ToString()
25         => _apiKey;
26     /// <summary>
27     /// Creates a new instance of the class with a given API Key.
28     /// </summary>
29     /// <param name="apiKey">The API Key to be used.</param>
30     public ApiKeyAuthenticator(string apiKey)
31     {
32         if (!apiKey.StartsWith("AKe."))
33         {
34             throw new ArgumentException($"API Key does not have valid format:
35             → {apiKey}", nameof(apiKey));
36         }
37
38         _apiKey = apiKey;
39         byte[] plainTextBytes = Encoding.UTF8.GetBytes($"{apiKey}:");
40         string val = Convert.ToBase64String(plainTextBytes);
41         _header = new("Basic", val);
42     }

```

Listing A.5.24: EasonEetwViewer.Dmdata.Authentication/Services/ApiKeyAuthenticator.cs

A.6 DM-D.S.S. API

```

1  using System.Text.Json;
2  using EasonEetwViewer.Dmdata.Api.Abstractions;
3  using EasonEetwViewer.Dmdata.Api.Services;
4  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
5  using Microsoft.Extensions.DependencyInjection;
6  using Microsoft.Extensions.Logging;
7
8  namespace EasonEetwViewer.Dmdata.Api.Extensions;
9  ///<summary>
10 /// Provides extension methods for <see cref="IServiceCollection"/> to add API Caller.
11 ///</summary>
12 public static class ApiCallerServiceCollectionExtensions
13 {
14     ///<summary>
15     /// Injects a <see cref="IApiCaller"/> with the given base URI.
16     ///</summary>
17     ///<param name="services">The instance of <see cref="IServiceCollection"/> for
18     /// the service to be injected.</param>
19     ///<param name="baseUri">The base URI for API Calls.</param>
20     ///<returns>The <see cref="IServiceCollection"/> where the service is injected,
21     /// for chained calls.</returns>
22     public static IServiceCollection AddApiCaller(this IServiceCollection services,
23         string baseUri)
24         => services.AddSingleton<IApiCaller>(sp
25             => new ApiCaller(
26                 baseUri,
27                 sp.GetRequiredService<ILogger<ApiCaller>>(),
28                 sp.GetRequiredService<IAuthenticationHelper>(),
29                 sp.GetRequiredService<JsonSerializerOptions>()));
30 }
```

Listing A.6.1:

EasonEetwViewer.Dmdata.Api/Extensions/ApiCallerServiceCollectionExtensions.cs

```

1  using System.Collections.Specialized;
2
3  namespace EasonEetwViewer.Dmdata.Api.Extensions;
4  ///<summary>
5  /// Provides extension methods for <see cref="NameValueCollection"/>.
6  ///</summary>
7  internal static class NameValueCollectionExtensions
8  {
8     ///<summary>
9     /// Adds a parameter to the query if the value is not null.
10    ///</summary>
11    ///<typeparam name="T">The type of the value.</typeparam>
12    ///<param name="query">The query for it to be added to.</param>
13    ///<param name="value">The value to be added.</param>
14    ///<param name="key">The key for the value.</param>
15    ///<returns>The query where it was added to, to chain the calls.</returns>
16    public static NameValueCollection AddIfNotNull<T>(
17        this NameValueCollection query,
18        T? value,
19        string key)
20        => AddIfNotNull(query, value, key, x => x?.ToString());
21    ///<summary>
```

```

23     /// Adds a parameter to the query if the value is not null, using the given
24     /// selector.
25     /// </summary>
26     /// <typeparam name="T">The type of the value.</typeparam>
27     /// <param name="query">The query for it to be added to.</param>
28     /// <param name="value">The value to be added.</param>
29     /// <param name="key">The key for the value.</param>
30     /// <param name="parameterSelector">The function used to convert the value to the
31     /// parameter in string format.</param>
32     /// <returns>The query where it was added to, to chain the calls.</returns>
33     public static NameValueCollection AddIfNotNull<T>(
34         this NameValueCollection query,
35         T? value,
36         string key,
37         Func<T, string?> parameterSelector)
38     {
39         if (value is not null)
40         {
41             query.Add(key, parameterSelector(value));
42         }
43     }
44 }
```

Listing A.6.2: EasonEetwViewer.Dmdata.Api/Extensions/NameValueCollectionExtensions.cs

```

1  using System.Diagnostics;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Enum.WebSocket;
3  using EasonEetwViewer.Dmdata.Dtos.Enum;
4
5  namespace EasonEetwViewer.Dmdata.Api.Extensions;
6  /// <summary>
7  /// Provides extensions for enums to convert to URI string to be used in API calls.
8  /// </summary>
9  internal static class EnumToUriStringExtensions
10 {
11     /// <summary>
12     /// Converts the <see cref="ConnectionStatus"/> to a URI string.
13     /// </summary>
14     /// <param name="connectionStatus">The connection status to be converted.</param>
15     /// <returns>The URI string for the connection status</returns>
16     /// <exception cref="UnreachableException">When the code reaches an unreachable
17     /// state.</exception>
18     public static string ToUriString(this ConnectionStatus connectionStatus)
19         => connectionStatus switch
20         {
21             ConnectionStatus.Waiting
22                 => "waiting",
23             ConnectionStatus.Open
24                 => "open",
25             ConnectionStatus.Closed
26                 => "closed",
27                 -                => throw new UnreachableException()
28         };
29     /// <summary>
30     /// Converts the <see cref="Intensity"/> to a URI string.

```

```

31     ///</summary>
32     ///<param name="intensity">The intensity to be converted.</param>
33     ///<returns>The URI string for the intensity</returns>
34     ///<exception cref="UnreachableException">When the code reaches an unreachable
35     //→ state.</exception>
36     public static string ToUriString(this Intensity intensity)
37         => intensity switch
38         {
39             Intensity.One
40                 => "1",
41             Intensity.Two
42                 => "2",
43             Intensity.Three
44                 => "3",
45             Intensity.Four
46                 => "4",
47             Intensity.FiveWeak
48                 => "5-",
49             Intensity.FiveStrong
50                 => "5+",
51             Intensity.SixWeak
52                 => "6-",
53             Intensity.SixStrong
54                 => "6+",
55             Intensity.Seven
56                 => "7",
57             -                => throw new UnreachableException()
58         };
59     }

```

Listing A.6.3: EasonEetwViewer.Dmdata.Api/Extensions/EnumToUriStringExtensions.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Record;
4
5  ///<summary>
6  ///<summary>Represents an error thrown by the API call.
7  ///</summary>
8  public record ErrorDetails
9  {
10     ///<summary>
11     ///<summary>The property <c>message</c>. The message of the error.
12     ///</summary>
13     [JsonPropertyName("message")]
14     public required string Message { get; init; }
15     ///<summary>
16     ///<summary>The property <c>code</c>. The HTTP error status code.
17     ///</summary>
18     [JsonPropertyName("code")]
19     public required int Code { get; init; }
20 }

```

Listing A.6.4: EasonEetwViewer.Dmdata.Api/Dtos/Record/ErrorDetails.cs

```
1  using System.Text.Json.Serialization;
```

```

2
3 namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.GdEarthquake;
4 /// <summary>
5 /// Represents an earthquake in the API call <c>gd.earthquake.event</c> which includes
6   → the telegrams.
7 /// </summary>
8 public record EarthquakeInfoWithTelegrams : EarthquakeInfo
9 {
10   /// <summary>
11   /// The property <c>telegrams</c>. The telegrams related to the earthquake.
12   /// </summary>
13   [JsonPropertyName("telegrams")]
14   public required IEnumerable<TelegramItem> Telegrams { get; init; }
15 }

```

Listing A.6.5:

EasonEetwViewer.Dmdata.Api/Dtos/Record/GdEarthquake/EarthquakeInfoWithTelegrams.cs

```

1 using System.Text.Json.Serialization;
2 using EasonEetwViewer.Dmdata.Api.Dtos.Enum.Earthquake;
3 using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
4 using EasonEetwViewer.Dmdata.Dtos.Enum;
5
6 namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.GdEarthquake;
7
8 /// <summary>
9 /// Represents an earthquake in the API call <c>gd.earthquake.list</c>.
10 /// </summary>
11 public record EarthquakeInfo
12 {
13   /// <summary>
14   /// The property <c>id</c>. The ID of the earthquake.
15   /// </summary>
16   [JsonPropertyName("id")]
17   public required int EarthquakeId { get; init; }
18   /// <summary>
19   /// The property <c>type</c>. The type of the earthquake.
20   /// </summary>
21   [JsonPropertyName("type")]
22   public required EarthquakeType Type { get; init; }
23   /// <summary>
24   /// The property <c>eventId</c>. The Event ID of the earthquake.
25   /// </summary>
26   [JsonPropertyName("eventId")]
27   public required string EventId { get; init; }
28   /// <summary>
29   /// The property <c>originTime</c>. The time at which the earthquake happened.
30   /// </summary>
31   /// <remarks>
32   /// <see langword="null"/> when there is only observation report of the
33     → earthquake.
34   /// </remarks>
35   [JsonPropertyName("originTime")]
36   public DateTimeOffset? OriginTime { get; init; }
37   /// <summary>
38   /// The property <c>arrivalTime</c>. The time at which the earthquake is detected.
39   /// </summary>
40   [JsonPropertyName("arrivalTime")]

```

```

40     public required DateTimeOffset ArrivalTime { get; init; }
41     /// <summary>
42     /// The property <c>hypocenter</c>. The hypocentre of the earthquake.
43     /// </summary>
44     /// <remarks>
45     /// <see langword="null"/> when there is only observation report of the
46     → earthquake.
47     /// </remarks>
48     [JsonPropertyName("hypocenter")]
49     public Hypocentre? Hypocentre { get; init; }
50     /// <summary>
51     /// The property <c>magnitude</c>. The magnitude of the earthquake.
52     /// </summary>
53     /// <remarks>
54     /// <see langword="null"/> when there is only observation report of the
55     → earthquake.
56     /// </remarks>
57     [JsonPropertyName("magnitude")]
58     public Magnitude? Magnitude { get; init; }
59     /// <summary>
60     /// The property <c>maxInt</c>. The maximum intensity observed for the earthquake.
61     /// </summary>
62     /// <remarks>
63     /// <see langword="null"/> when this is a distant earthquake.
64     /// </remarks>
65     [JsonPropertyName("maxInt")]
66     public Intensity? MaxIntensity { get; init; }
67     /// <summary>
68     /// The property <c>maxLgInt</c>. The maximum LPGM intensity observed for the
69     → earthquake.
70     /// </summary>
71     /// <remarks>
72     /// <see langword="null"/> when no LPGM observation report is issued.
73     /// </remarks>
74     [JsonPropertyName("maxLgInt")]
75     public LgIntensity? MaxLgIntensity { get; init; }
76     /// <summary>
77     /// The property <c>lgCategory</c>. The LPGM intensity category of the earthquake.
78     /// </summary>
79     /// <remarks>
80     /// <see langword="null"/> when no LPGM observation report is issued.
81     /// </remarks>
82     [JsonPropertyName("lgCategory")]
83     public LgCategory? LgCategory { get; init; }
84 }
```

Listing A.6.6: EasonEetwViewer.Dmdata.Api/Dtos/Record/GdEarthquake/EarthquakeInfo.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.GdEarthquake;
4  /// <summary>
5  /// Represents the head data of a telegram, in <c>gd.earthquake.event</c>.
6  /// </summary>
7  public record TelegramHead
8  {
9      /// <summary>
10     /// The property <c>type</c>. The type of the telegram.
```

```

11     /// </summary>
12     [JsonPropertyName("type")]
13     public required string Type { get; init; }
14     /// <summary>
15     /// The property <c>author</c>. The author of the telegram.
16     /// </summary>
17     [JsonPropertyName("author")]
18     public required string Author { get; init; }
19     /// <summary>
20     /// The property <c>time</c>. The base time of the telegram.
21     /// </summary>
22     [JsonPropertyName("time")]
23     public required DateTimeOffset Time { get; init; }
24     /// <summary>
25     /// The property <c>designation</c>. The designation code of the telegram.
26     /// </summary>
27     /// <remarks>
28     /// <see langword="null"/> when the telegram does not have a designation.
29     /// </remarks>
30     [JsonPropertyName("designation")]
31     public required string? Designation { get; init; }
32     /// <summary>
33     /// The property <c>test</c>. Whether the telegram is a test telegram.
34     /// </summary>
35     /// <remarks>
36     /// A constant <see langword="false"/> for this API call.
37     /// </remarks>
38     [JsonPropertyName("test")]
39     public bool Test { get; } = false;
40 }

```

Listing A.6.7: EasonEetwViewer.Dmdata.Api/Dtos/Record/GdEarthquake/TelegramHead.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Dtos.Enum.WebSocket;
4  using EasonEetwViewer.Dmdata.Dtos.Telegram;
5
6  namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.GdEarthquake;
7  /// <summary>
8  /// Represents an item in the list of telegrams for <c>gd.earthquake.event</c>.
9  /// </summary>
10 public record TelegramItem
11 {
12     /// <summary>
13     /// The property <c>serial</c>. The serial number of the telegram.
14     /// </summary>
15     [JsonPropertyName("serial")]
16     public required int Serial { get; init; }
17     /// <summary>
18     /// The property <c>id</c>. The ID of the JSON version of the telegram.
19     /// </summary>
20     [JsonPropertyName("id")]
21     public required string Id { get; init; }
22     /// <summary>
23     /// The property <c>originalId</c>. The original ID of the XML version of the
24     /// → telegram.
25     /// </summary>

```

```

25 [JsonPropertyName("originalId")]
26 public required string OriginalId { get; init; }
27 /// <summary>
28 /// The property <c>classification</c>. The classification of the telegram.
29 /// </summary>
30 [JsonPropertyName("classification")]
31 public required Classification Classification { get; init; }
32 /// <summary>
33 /// The property <c>head</c>. The head data of the telegram.
34 /// </summary>
35 [JsonPropertyName("head")]
36 public required TelegramHead TelegramHead { get; init; }
37 /// <summary>
38 /// The property <c>receivedTime</c>. The time when the telegram was received.
39 /// </summary>
40 [JsonPropertyName("receivedTime")]
41 public required DateTimeOffset ReceivedTime { get; init; }
42 /// <summary>
43 /// The property <c>xmlReport</c>. The XML head and control of the original XML
44 → telegram.
45 /// </summary>
46 [JsonPropertyName("xmlReport")]
47 public required XmlReport XmlReport { get; init; }
48 /// <summary>
49 /// The property <c>schema</c>. The schema version of the JSON telegram.
50 /// </summary>
51 [JsonPropertyName("schema")]
52 public required SchemaVersionInformation SchemaVersion { get; init; }
53 /// <summary>
54 /// The property <c>format</c>. The format of the telegram.
55 /// </summary>
56 [JsonPropertyName("format")]
57 public required FormatType Format { get; init; }
58 /// <summary>
59 /// The property <c>url</c>. The URL of the JSON telegram.
60 /// </summary>
61 [JsonPropertyName("url")]
62 public required Uri Url { get; init; }
63 }

```

Listing A.6.8: EasonEetwViewer.Dmdata.Api/Dtos/Record/GdEarthquake/TelegramItem.cs

```

1 using System.Text.Json.Serialization;
2
3 namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.WebSocket;
4
5 /// <summary>
6 /// Represents the URL of a WebSocket.
7 /// </summary>
8 public record WebSocketUrl
9 {
10     /// <summary>
11     /// The property <c>id</c>. The ID of the WebSocket.
12     /// </summary>
13     [JsonPropertyName("id")]
14     public required int WebSocketId { get; init; }
15     /// <summary>
16     /// The property <c>url</c>. The url to the WebSocket connection, with the ticket.

```

```

17     /// </summary>
18     [JsonPropertyName("url")]
19     public required Uri Url { get; init; }
20     /// <summary>
21     /// The property <c>protocol</c>. The protocol used by the WebSocket, a constant
22     → with value <c>dmdata.v2</c>.
23     /// </summary>
24     [JsonPropertyName("protocol")]
25     public string Protocol { get; } = "dmdata.v2";
26     /// <summary>
27     /// The property <c>expiration</c>. The expiration of the key in seconds, a
28     → constant <c>300</c>.
29     /// </summary>
30     [JsonPropertyName("expiration")]
31     public int Expiration { get; } = 300;
32 }
```

Listing A.6.9: EasonEetwViewer.Dmdata.Api/Dtos/Record/WebSocketUrl.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Enum.WebSocket;
3  using EasonEetwViewer.Dmdata.Dtos.Enum;
4  using EasonEetwViewer.Dmdata.Dtos.Enum.WebSocket;
5
6  namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.WebSocket;
7
8  /// <summary>
9  /// Details of a WebSocket connection.
10 /// </summary>
11 public record WebSocketDetails
12 {
13     /// <summary>
14     /// The property <c>ticket</c>. The ticket for the WebSocket connection.
15     /// </summary>
16     /// <remarks>
17     /// <see langword="null"/> when status is not waiting.
18     /// </remarks>
19     [JsonPropertyName("ticket")]
20     public required string? Ticket { get; init; }
21     /// <summary>
22     /// The property <c>classifications</c>. The classifications of telegrams that the
23     → WebSocket receives.
24     /// </summary>
25     [JsonPropertyName("classifications")]
26     public required IEnumerable<Classification> Classifications { get; init; }
27     /// <summary>
28     /// The property <c>test</c>. Whether the WebSocket receives test telegrams.
29     /// </summary>
30     [JsonPropertyName("test")]
31     public required TestStatus Test { get; init; }
32     /// <summary>
33     /// The property <c>types</c>. The types of telegrams the WebSocket receives.
34     /// </summary>
35     /// <remarks>
36     /// <see langword="null"/> when receiving all types from the classifications.
37     /// </remarks>
38     [JsonPropertyName("types")]
39     public required IEnumerable<string>? Types { get; init; }

```

```

39     ///<summary>
40     /// The property <c>appName</c>. The application name of the WebSocket connection.
41     ///</summary>
42     ///<remarks>
43     ///<see langword="null"/> when not indicated.
44     ///</remarks>
45     [JsonPropertyName("appName")]
46     public required string? ApplicationName { get; init; }
47     ///<summary>
48     /// The property <c>id</c>. The ID of the connection.
49     ///</summary>
50     [JsonPropertyName("id")]
51     public required int WebSocketId { get; init; }
52     ///<summary>
53     /// The property <c>start</c>. The start time of the connection.
54     ///</summary>
55     [JsonPropertyName("start")]
56     public required DateTimeOffset StartTime { get; init; }
57     ///<summary>
58     /// The property <c>end</c>. The end time of the connection.
59     ///</summary>
60     ///<remarks>
61     ///<see langword="null"/> when the connection is still open.
62     ///</remarks>
63     [JsonPropertyName("end")]
64     public required DateTimeOffset? EndTime { get; init; }
65     ///<summary>
66     /// The property <c>ping</c>. The previous ping-pong time of the connection.
67     ///</summary>
68     ///<remarks>
69     ///<see langword="null"/> when no ping-pong has been initiated.
70     ///</remarks>
71     [JsonPropertyName("ping")]
72     public required DateTimeOffset? PingTime { get; init; }
73     ///<summary>
74     /// The property <c>ipAddress</c>. The IP Address of the connection.
75     ///</summary>
76     ///<remarks>
77     ///<see langword="null"/> when the connection has never started.
78     ///</remarks>
79     [JsonPropertyName("ipAddress")]
80     public required string? IpAddress { get; init; }
81     ///<summary>
82     /// The property <c>server</c>. The server that the connection connected to.
83     ///</summary>
84     [JsonPropertyName("server")]
85     public required string ServerLocation { get; init; }
86     ///<summary>
87     /// The property <c>status</c>. The status of the WebSocket connection.
88     ///</summary>
89     [JsonPropertyName("status")]
90     public required ConnectionStatus WebSocketStatus { get; init; }
91 }

```

Listing A.6.10: EasonEetwViewer.Dmdata.Api/Dtos/Record/WebSocket/WebSocketDetails.cs

```

1  using System.Text.Json.Serialization;
2

```

```

3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.Contract;
4
5  /// <summary>
6  /// Represents the price of a contract.
7  /// </summary>
8  public record UnitPrice
{
9
10    /// <summary>
11    /// The property <c>day</c>. The daily price of the contract.
12    /// </summary>
13    [JsonPropertyName("day")]
14    public required int Day { get; init; }
15
16    /// <summary>
17    /// The property <c>month</c>. The monthly price of the contract.
18    /// </summary>
19    [JsonPropertyName("month")]
20    public required int Month { get; init; }
}

```

Listing A.6.11: EasonEetwViewer.Dmdata.Api/Dtos/Record/Contract/UnitPrice.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.Contract;
5
6  /// <summary>
7  /// Represents a contract of subscription to <see href="dmdatas.jp"/>.
8  /// </summary>
9  public record Contract
{
10
11    /// <summary>
12    /// The property <c>id</c>. The subscription ID.
13    /// </summary>
14    /// <remarks>
15    /// <see langword="null"/> if not subscribed to the contract.
16    /// </remarks>
17    [JsonPropertyName("id")]
18    public required int? SubscriptionId { get; init; }
19
20    /// <summary>
21    /// The property <c>planId</c>. The contract plan ID.
22    /// </summary>
23    [JsonPropertyName("planId")]
24    public required int PlanId { get; init; }
25
26    /// <summary>
27    /// The property <c>planName</c>. The name of the contract.
28    /// </summary>
29    [JsonPropertyName("planName")]
30    public required string PlanName { get; init; }
31
32    /// <summary>
33    /// The property <c>classifications</c>. The classification of the contract.
34    /// </summary>
35    [JsonPropertyName("classification")]
36    public required Classification Classification { get; init; }
37    /// <summary>
38    /// The property <c>price</c>. The price of the contract.
39    /// </summary>
40    [JsonPropertyName("price")]

```

```

38     public required UnitPrice Price { get; init; }
39     /// <summary>
40     /// The property <c>start</c>. The start time of the subscription.
41     /// </summary>
42     /// <remarks>
43     /// <see langword="null"/> if not subscribed to the contract.
44     /// </remarks>
45     [JsonPropertyName("start")]
46     public required DateTimeOffset? StartTime { get; init; }
47     /// <summary>
48     /// The property <c>isValid</c>. Whether the user is subscribed to the contract.
49     /// </summary>
50     [JsonPropertyName("isValid")]
51     public required bool IsValid { get; init; }
52     /// <summary>
53     /// The property <c>connectionCounts</c>. The number of extra WebSocket connection
54     → this subscription gives.
55     /// </summary>
56     [JsonPropertyName("connectionCounts")]
57     public required int ConnectionCounts { get; init; }
}

```

Listing A.6.12: EasonEetwViewer.Dmdata.Api/Dtos/Record/Contract/Contract.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.EarthquakeParameter;
4
5  /// <summary>
6  /// Represents a position (city or region).
7  /// </summary>
8  public record StationPosition
{
    /// <summary>
    /// The <c>code</c> property. The unique code of the position.
    /// </summary>
    [JsonPropertyName("code")]
    public required string Code { get; init; }
    /// <summary>
    /// The <c>name</c> property. The Kanji name of the position.
    /// </summary>
    [JsonPropertyName("name")]
    public required string KanjiName { get; init; }
    /// <summary>
    /// The <c>kana</c> property. The Kana name of the position.
    /// </summary>
    [JsonPropertyName("kana")]
    public required string KanaName { get; init; }
}

```

Listing A.6.13:
EasonEetwViewer.Dmdata.Api/Dtos/Record/EarthquakeParameter/StationPosition.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Enum.Station;
3
4  namespace EasonEetwViewer.Dmdata.Api.Dtos.Record.EarthquakeParameter;

```

```

5  ///<summary>
6  /// Represents an earthquake observation station.
7  ///</summary>
8  public record Station
9  {
10     ///<summary>
11     /// The property <c>region</c>, the region of the city.
12     ///</summary>
13     [JsonPropertyName("region")]
14     public required StationPosition Region { get; init; }
15     ///<summary>
16     /// The property <c>city</c>, the city of the observation point.
17     ///</summary>
18     [JsonPropertyName("city")]
19     public required StationPosition City { get; init; }
20     ///<summary>
21     /// The property <c>noCode</c>, the unique code assigned to the observation point.
22     ///</summary>
23     [JsonPropertyName("noCode")]
24     public required string NoCode { get; init; }
25     ///<summary>
26     /// The property <c>code</c>, the code for the observation point in the XML.
27     ///</summary>
28     [JsonPropertyName("code")]
29     public required string XmlCode { get; init; }
30     ///<summary>
31     /// The property <c>name</c>, the Kanji Name of the observation point.
32     ///</summary>
33     [JsonPropertyName("name")]
34     public required string KanjiName { get; init; }
35     ///<summary>
36     /// The property <c>kana</c>, the Kana Name of the observation point.
37     ///</summary>
38     [JsonPropertyName("kana")]
39     public required string KanaName { get; init; }
40     ///<summary>
41     /// The property <c>status</c>, the status of the observation point.
42     ///</summary>
43     [JsonPropertyName("status")]
44     public required Status StationStatus { get; init; }
45     ///<summary>
46     /// The property <c>owner</c>, the owner of the observation point.
47     ///</summary>
48     [JsonPropertyName("owner")]
49     public required string Owner { get; init; }
50     ///<summary>
51     /// The property <c>latitude</c>, the latitude of the observation point.
52     ///</summary>
53     [JsonPropertyName("latitude")]
54     public required double Latitude { get; init; }
55     ///<summary>
56     /// The property <c>longitude</c>, the longitude of the observation point.
57     ///</summary>
58     [JsonPropertyName("longitude")]
59     public required double Longitude { get; init; }
60 }

```

Listing A.6.14: EasonEetwViewer.Dmdata.Api/Dtos/Record/EarthquakeParameter/Station.cs

```

1  using EasonEetwViewer.Dmdata.Api.Dtos.Record.WebSocket;
2  using EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
3
4  namespace EasonEetwViewer.Dmdata.Api.Dtos.Response;
5  /// <summary>
6  /// Represents the result of an API call on <c>socket.list</c> API.
7  /// </summary>
8  public record WebSocketList : TokenBase<WebSocketDetails>;

```

Listing A.6.15: EasonEetwViewer.Dmdata.Api/Dtos/Response/WebSocketList.cs

```

1  using EasonEetwViewer.Dmdata.Api.Dtos.Record.GdEarthquake;
2  using EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
3
4  namespace EasonEetwViewer.Dmdata.Api.Dtos.Response;
5  /// <summary>
6  /// Represents the result of an API call on <c>gd.earthquake.list</c> API.
7  /// </summary>
8  public record GdEarthquakeList : PoolingBase<EarthquakeInfo>;

```

Listing A.6.16: EasonEetwViewer.Dmdata.Api/Dtos/Response/GdEarthquakeList.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Api.Dtos.Record;
4  using EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
5
6  namespace EasonEetwViewer.Dmdata.Api.Dtos.Response;
7
8  /// <summary>
9  /// Represents an Error HTTP response.
10 /// </summary>
11 public record Error : ApiBase
{
12
13    /// <summary>
14    /// The <c>status</c> property. Always set to <see cref="ResponseStatus.Error"/>.
15    /// </summary>
16    [JsonPropertyName("status")]
17    public override ResponseStatus ResponseStatus { get; } = ResponseStatus.Error;
18
19    /// <summary>
20    /// The <c>error</c> property. An object representing the error returned by the
21    /// API Call.
22    /// </summary>
23    [JsonPropertyName("error")]
24    public required ErrorDetails ErrorDetails { get; init; }
}

```

Listing A.6.17: EasonEetwViewer.Dmdata.Api/Dtos/Response/Error.cs

```

1  using EasonEetwViewer.Dmdata.Api.Dtos.Record.Contract;
2  using EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
3
4  namespace EasonEetwViewer.Dmdata.Api.Dtos.Response;
5  /// <summary>

```

```

6  /// Represents the result of an API call on <c>contract.list</c> API.
7  /// </summary>
8  public record ContractList : ListBase<Contract>;

```

Listing A.6.18: EasonEetwViewer.Dmdata.Api/Dtos/Response/ContractList.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Record.GdEarthquake;
3  using EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
4
5  namespace EasonEetwViewer.Dmdata.Api.Dtos.Response;
6  /// <summary>
7  /// Represents the result of an API call on <c>gd.earthquake.event</c> API.
8  /// </summary>
9  public record GdEarthquakeEvent : SuccessBase
10 {
11     /// <summary>
12     /// The property <c>event</c>, the earthquake event.
13     /// </summary>
14     [JsonPropertyName("event")]
15     public required EarthquakeInfoWithTelegrams EarthquakeEvent { get; init; }
16 }

```

Listing A.6.19: EasonEetwViewer.Dmdata.Api/Dtos/Response/GdEarthquakeEvent.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Record.WebSocket;
3  using EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
4  using EasonEetwViewer.Dmdata.Dtos.Enum;
5  using EasonEetwViewer.Dmdata.Enum.WebSocket;
6
7  namespace EasonEetwViewer.Dmdata.Api.Dtos.Response;
8
9  /// <summary>
10 /// Represents the result of a POST request to start WebSocket.
11 /// </summary>
12 public record WebSocketStart : SuccessBase
13 {
14     /// <summary>
15     /// The property <c>ticket</c>. The ticket for the WebSocket connection.
16     /// </summary>
17     [JsonPropertyName("ticket")]
18     public required string Ticket { get; init; }
19     /// <summary>
20     /// The property <c>classifications</c>. The classifications of telegrams that the
21     /// → WebSocket receives.
22     /// </summary>
23     [JsonPropertyName("classifications")]
24     public required IEnumerable<Classification> Classifications { get; init; }
25     /// <summary>
26     /// The property <c>test</c>. Whether the WebSocket receives test telegrams.
27     /// </summary>
28     [JsonPropertyName("test")]
29     public required TestStatus TestStatus { get; init; }
30     /// <summary>
31     /// The property <c>types</c>. The types of telegrams the program receives.
32     /// </summary>

```

```

32     ///<remarks>
33     ///<see langword="null"/> when receiving all types from the classifications.
34     ///</remarks>
35     [JsonPropertyName("types")]
36     public required IEnumerable<string?> Types { get; init; }
37     ///<summary>
38     ///<c>formats</c>. A list of formats of telegrams the WebSocket
39     → receives.
40     ///</summary>
41     [JsonPropertyName("formats")]
42     public required IEnumerable<FormatType> Formats { get; init; }
43     ///<summary>
44     ///<c>appName</c>. The application name of the WebSocket connection.
45     ///</summary>
46     ///<remarks>
47     ///<see langword="null"/> when not indicated.
48     ///</remarks>
49     [JsonPropertyName("appName")]
50     public required string? ApplicationName { get; init; }
51     ///<summary>
52     ///<c>websocket</c>. Represents the connection details to the
53     → WebSocket.
54     ///</summary>
55     [JsonPropertyName("websocket")]
56     public required WebSocketUrl WebSockerUrl { get; init; }
57 }
```

Listing A.6.20: EasonEetwViewer.Dmdata.Api/Dtos/Response/WebSocketStart.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Record.EarthquakeParameter;
3  using EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
4
5  namespace EasonEetwViewer.Dmdata.Api.Dtos.Response;
6
7  ///<summary>
8  ///<c>parameter.earthquake</c>.
9  ///</summary>
10 public record EarthquakeParameter : ListBase<Station>
11 {
12     ///<summary>
13     ///<c>changeTime</c>, representing the last time the list was
14     → changed.
15     ///</summary>
16     [JsonPropertyName("changeTime")]
17     public required DateTimeOffset ChangeTime { get; init; }
18     ///<summary>
19     ///<c>version</c>, a string indicating the version of the list.
20     ///</summary>
21     [JsonPropertyName("version")]
22     public required string Version { get; init; }
23 }
```

Listing A.6.21: EasonEetwViewer.Dmdata.Api/Dtos/Response/EarthquakeParameter.cs

```

1  using System.Text.Json.Serialization;
2
```

```

3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Enum;
4
5  /// <summary>
6  /// Represents the status of the API response.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<ResponseStatus>))]
9  public enum ResponseStatus
{
10
11    /// <summary>
12    /// The value <c>ok</c>, representing a success.
13    /// </summary>
14    [JsonStringEnumMemberName("ok")]
15    Success,
16
17    /// <summary>
18    /// The value <c>error</c>, representing an error.
19    /// </summary>
20    [JsonStringEnumMemberName("error")]
21    Error
}

```

Listing A.6.22: EasonEetwViewer.Dmdata.Api/Dtos/Enum/ResponseStatus.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Enum.WebSocket;
4
5  /// <summary>
6  /// Represents the specified format mode of a WebSocket connection.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<FormatMode>))]
9  public enum FormatMode
{
10
11    /// <summary>
12    /// The value <c>raw</c>, representing raw formatting.
13    /// </summary>
14    [JsonStringEnumMemberName("raw")]
15    Raw,
16
17    /// <summary>
18    /// The value <c>json</c>, representing formatting in JSON.
19    /// </summary>
20    [JsonStringEnumMemberName("json")]
21    Json
}

```

Listing A.6.23: EasonEetwViewer.Dmdata.Api/Dtos/Enum/WebSocket/FormatMode.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Enum.WebSocket;
4
5  /// <summary>
6  /// Represents the status of the WebSocket connection.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<ConnectionStatus>))]
9  public enum ConnectionStatus
{
10
11    /// <summary>

```

```

12     /// The value <c>waiting</c>, representing awaiting connection.
13     /// </summary>
14     [JsonStringEnumMemberName("waiting")]
15     Waiting,
16     /// <summary>
17     /// The value <c>open</c>, representing an active connection.
18     /// </summary>
19     [JsonStringEnumMemberName("open")]
20     Open,
21     /// <summary>
22     /// The value <c>closed</c>, representing a closed connection.
23     /// </summary>
24     [JsonStringEnumMemberName("closed")]
25     Closed
26 }

```

Listing A.6.24: EasonEetwViewer.Dmdata.Api/Dtos/Enum/WebSocket/ConnectionStatus.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Enum.Earthquake;
4
5  /// <summary>
6  /// Describes the type of earthquake.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<EarthquakeType>))]
9  public enum EarthquakeType
{
10
11    /// <summary>
12    /// The value <c>normal</c>, representing an earthquake within close vicinity of
13    /// Japan.
14    /// </summary>
15    [JsonStringEnumMemberName("normal")]
16    Japan,
17    /// <summary>
18    /// The value <c>distant</c>, representing a distant earthquake.
19    /// </summary>
20    [JsonStringEnumMemberName("distant")]
21    Distant
}

```

Listing A.6.25: EasonEetwViewer.Dmdata.Api/Dtos/Enum/Earthquake/EarthquakeType.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.Enum.Station;
4
5  /// <summary>
6  /// The status of a station.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<Status>))]
9  public enum Status
{
10
11    /// <summary>
12    /// The value <c>新規</c>, representing a new station.
13    /// </summary>
14    [JsonStringEnumMemberName("新規")]
}

```

```

15     New,
16     /// <summary>
17     /// The value <c>変更</c>, representing a change in detail of the station.
18     /// </summary>
19     [JsonStringEnumMemberName("変更")]
20     Changed,
21     /// <summary>
22     /// The value <c>現</c>, representing the same station details.
23     /// </summary>
24     [JsonStringEnumMemberName("現")]
25     InUse,
26     /// <summary>
27     /// The value <c>廃止</c>, representing a discontinued station.
28     /// </summary>
29     [JsonStringEnumMemberName("廃止")]
30     Discontinued
31 }

```

Listing A.6.26: EasonEetwViewer.Dmdata.Api/Dtos/Enum/Station/Status.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Enum.WebSocket;
3  using EasonEetwViewer.Dmdata.Dtos.Enum;
4  using EasonEetwViewer.Dmdata.Dtos.Enum.WebSocket;
5
6  namespace EasonEetwViewer.Dmdata.Api.Dtos.Request;
7
8  /// <summary>
9  /// Outlines the necessary information provided in the JSON in the header in
10  /// <c>socket.start</c> API call.
11  /// </summary>
12  public record WebSocketStartPost
13  {
14      /// <summary>
15      /// The property <c>classifications</c>. The classifications of telegrams that the
16      /// <c>WebSocket receives.
17      /// </summary>
18      [JsonPropertyName("classifications")]
19      public required IEnumerable<Classification> Classifications { get; init; }
20      /// <summary>
21      /// The property <c>types</c>. The types of telegrams the program receives.
22      /// </summary>
23      /// <remarks>
24      /// <see langword="null"/> when receiving all types from the classifications.
25      /// </remarks>
26      [JsonPropertyName("types")]
27      public IEnumerable<string>? Types { get; init; }
28      /// <summary>
29      /// The property <c>test</c>. Whether the WebSocket receives test telegrams.
30      /// </summary>
31      [JsonPropertyName("test")]
32      public TestStatus? TestStatus { get; init; }
33      /// <summary>
34      /// The property <c>appName</c>. The application name of the WebSocket connection.
35      /// </summary>
36      /// <remarks>
37      /// <see langword="null"/> when not indicated.
38      /// </remarks>

```

```

37     [JsonPropertyName("appName")]
38     public string? AppName { get; init; }
39     /// <summary>
40     /// The property <c>formatMode</c>. Whether the WebSocket receives JSON serialised
41     /// → data or original data.
42     /// </summary>
43     [JsonPropertyName("formatMode")]
44     public FormatMode? FormatMode { get; init; }
45 }
```

Listing A.6.27: EasonEetwViewer.Dmdata.Api/Dtos/Request/WebSocketStartPost.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
4
5  /// <summary>
6  /// Outlines the model of a list response from an API call.
7  /// Abstract and cannot be instantiated.
8  /// </summary>
9  /// <typeparam name="T">The type of item in the list.</typeparam>
10 public abstract record ListBase<T> : SuccessBase
11 {
12     /// <summary>
13     /// The <c>items</c> property. The list of items returned by the API call.
14     /// </summary>
15     [JsonPropertyName("items")]
16     public required IEnumerable<T> ItemList { get; init; }
17 }
```

Listing A.6.28: EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/ListBase.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
5
6  /// <summary>
7  /// Outlines the model of an Error HTTP response.
8  /// Abstract and cannot be instantiated.
9  /// </summary>
10 public abstract record SuccessBase : ApiBase
11 {
12     /// <summary>
13     /// The <c>status</c> property. Always set to <see
14     /// → cref="ResponseStatus.Success"/>.
15     /// </summary>
16     [JsonPropertyName("status")]
17     public override ResponseStatus ResponseStatus { get; } = ResponseStatus.Success;
18 }
```

Listing A.6.29: EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/SuccessBase.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Enum;
```

```

3
4 namespace EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
5
6 /// <summary>
7 /// Outlines the model of a HTTP response in JSON format from an API Call.
8 /// Abstract and cannot be instantiated.
9 /// </summary>
10 public abstract record ApiBase
11 {
12     /// <summary>
13     /// The <c>responseId</c> property. An unique ID for the API Call.
14     /// </summary>
15     [JsonPropertyName("responseId")]
16     public required string ResponseId { get; init; }
17     /// <summary>
18     /// The <c>responseTime</c> property. The time when the call was received.
19     /// </summary>
20     [JsonPropertyName("responseTime")]
21     public required DateTimeOffset ResponseTime { get; init; }
22     /// <summary>
23     /// The <c>status</c> property. An enum representing the status of the call.
24     /// Abstract and has to be overridden.
25     /// </summary>
26     [JsonPropertyName("status")]
27     public abstract ResponseStatus ResponseStatus { get; }
28 }
```

Listing A.6.30: EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/ApiBase.cs

```

1 using System.Text.Json.Serialization;
2
3 namespace EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
4
5 /// <summary>
6 /// Outlines the model of a list response with next pooling options.
7 /// Abstract and cannot be instantiated.
8 /// </summary>
9 /// <typeparam name="T">The type of item in the list.</typeparam>
10 public abstract record PoolingBase<T> : TokenBase<T>
11 {
12     /// <summary>
13     /// The <c>nextPooling</c> property. The pooling token that should be specified in
14     /// the next API call.
15     /// </summary>
16     /// <remarks>
17     /// <see langword="null"/> when the current call is the final call.
18     /// </remarks>
19     [JsonPropertyName("nextPooling")]
20     public string? NextPooling { get; init; }
21     /// <summary>
22     /// The <c>nextPoolingInterval</c> property. The time in milliseconds that the
23     /// program should wait until the next API call.
24     /// </summary>
25     /// <remarks>
26     /// <see langword="null"/> when the current call is the final call.
27     /// </remarks>
28     [JsonPropertyName("nextPoolingInterval")]
29     public int? NextPoolingInterval { get; init; }
```

28 }

Listing A.6.31: EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/PoolingBase.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Api.Dtos.ResponseBase;
4
5  /// <summary>
6  /// Outlines the model of a list response with next token options.
7  /// Abstract and cannot be instantiated.
8  /// </summary>
9  /// <typeparam name="T">The type of item in the list.</typeparam>
10 public abstract record TokenBase<T> : ListBase<T>
11 {
12     /// <summary>
13     /// The <c>nextToken</c> property. The token that should be specified in the next
14     /// API call.
15     /// </summary>
16     /// <remarks>
17     /// <see langword="null"/> when the current call is the final call.
18     /// </remarks>
19     [JsonPropertyName("nextToken")]
20     public string? NextToken { get; init; }
}

```

Listing A.6.32: EasonEetwViewer.Dmdata.Api/Dtos/ResponseBase/TokenBase.cs

```

1  using EasonEetwViewer.Dmdata.Api.Dtos.Enum.WebSocket;
2  using EasonEetwViewer.Dmdata.Api.Dtos.Request;
3  using EasonEetwViewer.Dmdata.Api.Dtos.Response;
4  using EasonEetwViewer.Dmdata.Dtos.Enum;
5
6  namespace EasonEetwViewer.Dmdata.Api.Abstractions;
7  /// <summary>
8  /// Represents the interface for calling the <see href="dodata.jp"/> API.
9  /// </summary>
10 public interface IApiCaller
11 {
12     /// <summary>
13     /// Gets the list of contracts. <c>contract.list</c> API.
14     /// </summary>
15     /// <returns>The contract list. <see langword="null"/> if unsuccessful.</returns>
16     Task<ContractList?> GetContractListAsync();
17     /// <summary>
18     /// Gets the list of WebSockets. <c>socket.list</c> API.
19     /// </summary>
20     /// <param name="connectionStatus">The connection status of the WebSocket.</param>
21     /// <param name="cursorToken">The cursor token for calling a list.</param>
22     /// <param name="id">The ID of the WebSocket.</param>
23     /// <param name="limit">The length limit of the list.</param>
24     /// <returns>The WebSocket list. <see langword="null"/> if unsuccessful.</returns>
25     Task<WebSocketList?> GetWebSocketListAsync(
26         int? id = null,
27         ConnectionStatus? connectionStatus = null,
28         string? cursorToken = null,
29         int? limit = null);

```

```

30     /// <summary>
31     /// Start a new WebSocket connection. <c>socket.start</c> API.
32     /// </summary>
33     /// <param name="postData">The data to be included in the POST request.</param>
34     /// <returns>The details of the WebSocket. <see langword="null"/> if
35     /// → unsuccessful.</returns>
36     Task<WebSocketStart?> PostWebSocketStartAsync(WebSocketStartPost postData);
37     /// <summary>
38     /// Deletes a WebSocket connection. <c>socket.close</c> API.
39     /// </summary>
40     /// <param name="id">The ID of the WebSocket.</param>
41     /// <returns><see langword="true"/> if it was successful, <see langword="false"/>
42     /// → otherwise.</returns>
43     Task<bool> DeleteWebSocketAsync(int id);
44     /// <summary>
45     /// Gets the list of earthquake observation points. <c>parameter.earthquake</c>
46     /// API.
47     /// </summary>
48     /// <returns>The list of earthquake observation points. <see langword="null"/> if
49     /// → unsuccessful.</returns>
50     Task<EarthquakeParameter?> GetEarthquakeParameterAsync();
51     /// <summary>
52     /// Get the list of path earthquakes. <c>gd.earthquake.list</c> API.
53     /// </summary>
54     /// <param name="hypocentreCode">The code for the hypocentre.</param>
55     /// <param name="maxInt">The maximum intensity.</param>
56     /// <param name="date">The date that the earthquake happened.</param>
57     /// <param name="limit">The length limit of the list.</param>
58     /// <param name="cursorToken">The cursor token for calling a list.</param>
59     /// <returns>A list of past earthquakes. <see langword="null"/> if
60     /// → unsuccessful.</returns>
61     Task<GdEarthquakeList?> GetPastEarthquakeListAsync(
62         string? hypocentreCode = null,
63         Intensity? maxInt = null,
64         DateOnly? date = null,
65         int? limit = null,
66         string? cursorToken = null);
67     /// <summary>
68     /// Gets the details of a past earthquake event. <c>gd.earthquake.event</c> API.
69     /// </summary>
70     /// <param name="eventId">The event ID of the earthquake.</param>
71     /// <returns>The details of the past earthquake event. <see langword="null"/> if
72     /// → unsuccessful.</returns>
73     Task<GdEarthquakeEvent?> GetPathEarthquakeEventAsync(string eventId);
74 }
```

Listing A.6.33: EasonEetwViewer.Dmdata.Api/Abstractions/IApiCaller.cs

```

1  using EasonEetwViewer.Dmdata.Api.Services;
2  using Microsoft.Extensions.Logging;
3
4  namespace EasonEetwViewer.Dmdata.Api.Logging;
5
6  /// <summary>
7  /// Represents the log messages used in <see cref="ApiCaller"/>.
8  /// </summary>
9  internal static partial class ApiCallerLogs
10 {
```

```
11     /// <summary>
12     /// Log when instantiated.
13     /// </summary>
14     /// <param name="logger">The logger to be used.</param>
15     [LoggerMessage(
16         EventId = 0,
17         EventName = nameof(Instantiated),
18         Level = LogLevel.Information,
19         Message = "Instantiated.")]
20     public static partial void Instantiated(
21         this ILogger<ApiCaller> logger);
22     /// <summary>
23     /// Log when an HTTP Request failed.
24     /// </summary>
25     /// <param name="logger">The logger to be used.</param>
26     /// <param name="exception">The exception message.</param>
27     [LoggerMessage(
28         EventId = 1,
29         EventName = nameof(HttpRequestFails),
30         Level = LogLevel.Error,
31         Message = "Http Request Failed: `{{Exception}}`.")]
32     public static partial void HttpRequestFails(
33         this ILogger<ApiCaller> logger, string exception);
34     /// <summary>
35     /// Log when an HTTP Request was created.
36     /// </summary>
37     /// <param name="logger">The logger to be used.</param>
38     /// <param name="method">The HTTP Method.</param>
39     /// <param name="relativePath">The relative path.</param>
40     [LoggerMessage(
41         EventId = 2,
42         EventName = nameof(RequestCreated),
43         Level = LogLevel.Information,
44         Message = "Http Request Created: `{{Method}}` `{{RelativePath}}`.")]
45     public static partial void RequestCreated(
46         this ILogger<ApiCaller> logger, HttpMethod method, string relativePath);
47     /// <summary>
48     /// Log when an HTTP Request with content was created.
49     /// </summary>
50     /// <param name="logger">The logger to be used.</param>
51     /// <param name="method">The HTTP Method.</param>
52     /// <param name="relativePath">The relative path.</param>
53     /// <param name="content">The content in the request.</param>
54     [LoggerMessage(
55         EventId = 3,
56         EventName = nameof(RequestWithContentCreated),
57         Level = LogLevel.Information,
58         Message = "Http Request Created: `{{Method}}` `{{RelativePath}}` `{{Content}}`.")]
59     public static partial void RequestWithContentCreated(
60         this ILogger<ApiCaller> logger, HttpMethod method, string relativePath,
61         → string? content);
62     /// <summary>
63     /// Log when failed to parse a result string.
64     /// </summary>
65     /// <param name="logger">The logger to be used.</param>
66     /// <param name="json">The JSON string that is parsed.</param>
67     [LoggerMessage(
68         EventId = 4,
69         EventName = nameof(JsonParsingFailed),
```

```

69     Level = LogLevel.Information,
70     Message = "JSON Parsing Failed: `'{Json}`.")]
71     public static partial void JsonParsingFailed(
72         this ILogger<ApiCaller> logger, string json);
73     /// <summary>
74     /// Log when an error is received from the API call.
75     /// </summary>
76     /// <param name="logger">The logger to be used.</param>
77     /// <param name="errorCode">The error code received.</param>
78     /// <param name="errorMessage">The error message received.</param>
79     [LoggerMessage(
80         EventId = 5,
81         EventName = nameof(ErrorReceived),
82         Level = LogLevel.Error,
83         Message = "Error received from API call: `'{ErrorCode}` `'{ErrorMessage}`")]
84     public static partial void ErrorReceived(
85         this ILogger<ApiCaller> logger, int errorCode, string errorMessage);
86     /// <summary>
87     /// Log when not authenticated.
88     /// </summary>
89     /// <param name="logger">The logger to be used.</param>
90     [LoggerMessage(
91         EventId = 6,
92         EventName = nameof(NotAuthenticated),
93         Level = LogLevel.Warning,
94         Message = "Not authenticated.")]
95     public static partial void NotAuthenticated(
96         this ILogger<ApiCaller> logger);
97 }
```

Listing A.6.34: EasonEetwViewer.Dmdata.Api/Logging/ApiCallerLogs.cs

```

1  using System.Collections.Specialized;
2  using System.Net;
3  using System.Net.Http.Headers;
4  using System.Text;
5  using System.Text.Json;
6  using System.Web;
7  using EasonEetwViewer.Dmdata.Api.Abstractions;
8  using EasonEetwViewer.Dmdata.Api.Dtos.Enum.WebSocket;
9  using EasonEetwViewer.Dmdata.Api.Dtos.Request;
10 using EasonEetwViewer.Dmdata.Api.Dtos.Response;
11 using EasonEetwViewer.Dmdata.Api.Extensions;
12 using EasonEetwViewer.Dmdata.Api.Logging;
13 using EasonEetwViewer.Dmdata.Authentication.Abstractions;
14 using EasonEetwViewer.Dmdata.Dtos.Enum;
15 using Microsoft.Extensions.Logging;

16
17 namespace EasonEetwViewer.Dmdata.Api.Services;
18 /// <summary>
19 /// The default implementation for <see cref="IApiCaller"/>.
20 /// </summary>
21 internal sealed class ApiCaller : IApiCaller
22 {
23     private readonly HttpClient _client;
24     private readonly ILogger<ApiCaller> _logger;
25     private readonly IAuthenticationHelper _authenticator;
26     private readonly JsonSerializerOptions _options;
```

```
27     private static NameValueCollection EmptyQuery
28         => HttpUtility.ParseQueryString(string.Empty);
29     public ApiCaller(string baseApi, ILogger<ApiCaller> logger, IAuthenticationHelper
30         authenticator, JsonSerializerOptions jsonSerializerOptions)
31     {
32         _client = new()
33         {
34             BaseAddress = new(baseApi)
35         };
36         _authenticator = authenticator;
37         _options = jsonSerializerOptions;
38         _logger = logger;
39         _logger.Instantiated();
40     }
41     /// <inheritdoc/>
42     public async Task<ContractList?> GetContractListAsync()
43         => await ParseGetResultAsync<ContractList>("contract");
44     /// <inheritdoc/>
45     public async Task<WebSocketList?> GetWebSocketListAsync(
46         int? id = null,
47         ConnectionStatus? connectionStatus = null,
48         string? cursorToken = null,
49         int? limit = null)
50         => await ParseGetResultAsync<WebSocketList>("socket",
51             EmptyQuery
52             .AddIfNotNull(id, "id")
53             .AddIfNotNull(connectionStatus, "status", x => x?.ToString())
54             .AddIfNotNull(cursorToken, "cursorToken")
55             .AddIfNotNull(limit, "limit")
56             .ToString());
57     /// <inheritdoc/>
58     public async Task<WebSocketStart?> PostWebSocketStartAsync(WebSocketStartPost
59         postData)
60         => await ParsePostJsonAsync<WebSocketStart, WebSocketStartPost>(
61             "socket",
62             postData);
63     /// <inheritdoc/>
64     public async Task<bool> DeleteWebSocketAsync(int id)
65         => await ParseDeleteResultAsync($"socket/{id}");
66     /// <inheritdoc/>
67     public async Task<EarthquakeParameter?> GetEarthquakeParameterAsync()
68         => await
69             ParseGetResultAsync<EarthquakeParameter>("parameter/earthquake/station");
70     /// <inheritdoc/>
71     public async Task<GdEarthquakeList?> GetPastEarthquakeListAsync(
72         string? hypocentreCode = null,
73         Intensity? maxInt = null,
74         DateOnly? date = null,
75         int? limit = null,
76         string? cursorToken = null)
77         => await ParseGetResultAsync<GdEarthquakeList>("gd/earthquake",
78             EmptyQuery
79             .AddIfNotNull(hypocentreCode, "hypocenter")
80             .AddIfNotNull(maxInt, "maxInt", x => x?.ToString())
81             .AddIfNotNull(date, "date", x => x?.ToString("yyyy-MM-dd"))
82             .AddIfNotNull(limit, "limit")
83             .AddIfNotNull(cursorToken, "cursorToken")
84             .ToString());
85     /// <inheritdoc/>
```

```
83     public async Task<GdEarthquakeEvent?> GetPathEarthquakeEventAsync(string eventId)
84         => await ParseGetResultAsync<GdEarthquakeEvent>($"gd/earthquake/{eventId}");
85     private async Task<T?> ParseGetResultAsync<T>(string relativePath, string?
86         → queryString) where T : class
87         => await ParseGetResultAsync<T>(
88             queryString.IsNullOrEmpty(queryString)
89             ? relativePath
90             : $"{relativePath}?{queryString}");
91     /// <summary>
92     /// Parses the result of a GET request.
93     /// </summary>
94     /// <param name="relativePath">The relative path to the request.</param>
95     /// <returns>The result if successful, <see langword="null"/> otherwise.</returns>
96     private async Task<T?> ParseGetResultAsync<T>(string relativePath) where T : class
97     {
98         using HttpResponseMessage request = CreateHttpRequest(HttpMethod.Get,
99             → relativePath);
100        return await ParseResultAsync<T>(request);
101    }
102    /// <summary>
103    /// Parses the result of a POST request.
104    /// </summary>
105    /// <param name="relativePath">The relative path to the request.</param>
106    /// <param name="content">The content to be included in the request.</param>
107    /// <returns>The result if successful, <see langword="null"/> otherwise.</returns>
108    private async Task<TResult?> ParsePostJsonAsync<TResult, TContent>(string
109        → relativePath, TContent content) where TResult : class
110    {
111        using HttpResponseMessage request = CreateHttpRequest(
112            HttpMethod.Post,
113            relativePath,
114            content);
115        return await ParseResultAsync<TResult>(request);
116    }
117    /// <summary>
118    /// Parses the result of a DELETE request.
119    /// </summary>
120    /// <param name="relativePath">The relative path to the request.</param>
121    /// <returns><see langword="true"/> if successful, <see langword="false"/>
122    /// → otherwise.</returns>
123    private async Task<bool> ParseDeleteResultAsync(string relativePath)
124    {
125        using HttpResponseMessage request = CreateHttpRequest(HttpMethod.Delete,
126            → relativePath);
127        return await ParseSuccessAsync(request);
128    }
129    /// <summary>
130    /// Parses the result of the request.
131    /// </summary>
132    /// <typeparam name="T">The type to parse the result into</typeparam>
133    /// <param name="request">The <see cref="HttpRequestMessage"/> to be sent.</param>
134    /// <returns>The result if successful, <see langword="null"/> otherwise.</returns>
135    private async Task<T?> ParseResultAsync<T>(HttpRequestMessage request) where T :

```

```
136     }
137
138     string responseBody = await response.Content.ReadAsStringAsync();
139     if (response.IsSuccessStatusCode)
140     {
141         try
142         {
143             T? result = JsonSerializer.Deserialize<T>(responseBody, _options);
144
145             if (result is null)
146             {
147                 _logger.JsonParsingFailed(responseBody);
148                 return null;
149             }
150
151             return result;
152         }
153         catch (JsonException)
154         {
155             _logger.JsonParsingFailed(responseBody);
156             return null;
157         }
158     }
159     else
160     {
161         await HandleError(responseBody, response.StatusCode);
162         return null;
163     }
164 }
165 /// <summary>
166 /// Parses the result of the request.
167 /// </summary>
168 /// <param name="request">The <see cref="HttpRequestMessage"/> to be sent.</param>
169 /// <returns><see langword="true"/> if successful, <see langword="false"/>
170 //> otherwise.</returns>
171 private async Task<bool> ParseSuccessAsync(HttpRequestMessage request)
172 {
173     using HttpResponseMessage? response = await SendRequestAsync(request);
174     if (response is null)
175     {
176         return false;
177     }
178     else if (response.IsSuccessStatusCode)
179     {
180         return true;
181     }
182     else
183     {
184         string responseBody = await response.Content.ReadAsStringAsync();
185         await HandleError(responseBody, response.StatusCode);
186         return false;
187     }
188 }
189 /// <summary>
190 /// Handles the error received from the API and logs as appropriate.
191 /// </summary>
192 /// <param name="responseBody"></param>
193 /// <param name="statusCode"></param>
```

```
193     /// <returns>A <see cref="Task"/> object that represents the asynchronous
194     /// operation.</returns>
195     private async Task HandleError(string responseBody, HttpStatusCode statusCode)
196     {
197         try
198         {
199             Error? error = JsonSerializer.Deserialize<Error>(responseBody, _options);
200             if (error is null)
201             {
202                 _logger.JsonParsingFailed(responseBody);
203                 return;
204             }
205
206             _logger.ErrorReceived(error.ErrorDetails.Code,
207             → error.ErrorDetails.Message);
208
209             if (statusCode is HttpStatusCode.Unauthorized or HttpStatusCode.Forbidden)
210             {
211                 await _authenticator.InvalidAuthenticatorAsync(error.ErrorDetails.Message);
212                 → sage);
213             }
214         }
215     }
216
217     /// <summary>
218     /// Adds the authentication header to the request and sends it.
219     /// </summary>
220     /// <param name="request">The <see cref="HttpRequestMessage"/> to be sent.</param>
221     /// <returns>The <see cref="HttpResponseMessage"/> obtained. <see
222     /// langword="null"/> when unsuccessful.</returns>
223     private async Task<HttpResponseMessage?> SendRequestAsync(HttpRequestMessage
224     → request)
225     {
226         AuthenticationHeaderValue? header = await
227             → _authenticator.GetAuthenticationHeaderAsync();
228         if (header is null)
229         {
230             _logger.NotAuthenticated();
231             return null;
232         }
233
234         request.Headers.Authorization = header;
235         try
236         {
237             return await _client.SendAsync(request);
238         }
239         catch (HttpRequestException ex)
240         {
241             _logger.HttpRequestFails(ex.ToString());
242             return null;
243         }
244     }
245
246     /// <summary>
247     /// Creates a new <see cref="HttpRequestMessage"/> with the specified HTTP Method
248     /// and relative path.
249     /// </summary>
```

```

245     /// <param name="method">The <see cref="HttpMethod"/> specified.</param>
246     /// <param name="relativePath">The relative path.</param>
247     /// <returns>The <see cref="HttpRequestMessage"/> created.</returns>
248     private HttpRequestMessage CreateHttpRequest(HttpMethod method, string
249         ↪ relativePath)
250     {
251         _logger.RequestCreated(method, relativePath);
252         return new(method, relativePath);
253     }
254     /// <summary>
255     /// Creates a new <see cref="HttpRequestMessage"/> with the specified HTTP Method,
256     ↪ relative path and content.
257     /// </summary>
258     /// <typeparam name="TContent">The type of the content to be included.</typeparam>
259     /// <param name="method">The <see cref="HttpMethod"/> specified.</param>
260     /// <param name="relativePath">The relative path.</param>
261     /// <param name="content">The content to be included.</param>
262     /// <returns>The <see cref="HttpRequestMessage"/> created.</returns>
263     private HttpRequestMessage CreateHttpRequest<TContent>(HttpMethod method, string
264         ↪ relativePath, TContent content)
265     {
266         string serialisedContent = JsonSerializer.Serialize(content);
267         _logger.RequestWithContentCreated(method, relativePath, serialisedContent);
268         return new(method, relativePath)
269         {
270             Content = new StringContent(
271                 serialisedContent,
272                 Encoding.UTF8,
273                 "application/json")
274         };
275     }
276 }
```

Listing A.6.35: EasonEetwViewer.Dmdata.Api/Services/ApiCaller.cs

A.7 DM-D.S.S. Telegrams

```

1  using EasonEetwViewer.Dmdata.Telegram.Services;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Exceptions;
4  /// <summary>
5  /// Represents format errors that occurs in <see cref="TelegramParser"/>.
6  /// </summary>
7  public class TelegramParserFormatException : TelegramParserException
8  {
9      /// <summary>
10     /// Instantiates a new instance of the <see cref="TelegramParserFormatException"/>
11     ↪ class.
12     /// </summary>
13     public TelegramParserFormatException() { }
14     /// <summary>
15     /// Instantiates a new instance of the <see cref="TelegramParserFormatException"/>
16     ↪ class with a specified error message.
17     /// </summary>
18     /// <param name="message">The error message.</param>
19     public TelegramParserFormatException(string message)
20         : base(message) { }
```

```

19     ///<summary>
20     /// Instantiates a new instance of the <see cref="TelegramParserFormatException"/>
21     /// class with a specified error message and an inner exception.
22     ///</summary>
23     ///<param name="message">The error message.</param>
24     ///<param name="inner">The inner exception.</param>
25     public TelegramParserFormatException(string message, Exception inner)
26     : base(message, inner) { }

```

Listing A.7.1:
EasonEetwViewer.Dmdata.Telgram/Exceptions/TelgramParserFormatException.cs

```

1  using EasonEetwViewer.Dmdata.Telgram.Services;
2
3  namespace EasonEetwViewer.Dmdata.Telgram.Exceptions;
4  ///<summary>
5  /// Represents errors that occurs in <see cref="TelgramParser"/>.
6  ///</summary>
7  public class TelgramParserException : Exception
8  {
9      ///<summary>
10     /// Instantiates a new instance of the <see cref="TelgramParserException"/>
11     /// class.
12     ///</summary>
13     public TelgramParserException() { }
14     ///<summary>
15     /// Instantiates a new instance of the <see cref="TelgramParserException"/> class
16     /// with a specified error message.
17     ///</summary>
18     ///<param name="message">The error message.</param>
19     public TelgramParserException(string message)
20     : base(message) { }
21     ///<summary>
22     /// Instantiates a new instance of the <see cref="TelgramParserException"/> class
23     /// with a specified error message and an inner exception.
24     ///</summary>
25     ///<param name="message">The error message.</param>
26     ///<param name="inner">The inner exception.</param>
27     public TelgramParserException(string message, Exception inner)
28     : base(message, inner) { }
29 }

```

Listing A.7.2: EasonEetwViewer.Dmdata.Telgram/Exceptions/TelgramParserException.cs

```

1  using EasonEetwViewer.Dmdata.Telgram.Services;
2
3  namespace EasonEetwViewer.Dmdata.Telgram.Exceptions;
4  ///<summary>
5  /// Represents unsupported errors that occurs in <see cref="TelgramParser"/>.
6  ///</summary>
7  public class TelgramParserUnsupportedException : TelgramParserException
8  {
9      ///<summary>
10     /// Instantiates a new instance of the <see
11     /// cref="TelgramParserUnsupportedException"/> class.
12     ///</summary>
13 }

```

```

12     public TelegramParserUnsupportedException() { }
13     /// <summary>
14     /// Instantiates a new instance of the <see
15     →   cref="TelegramParserUnsupportedException"/> class with a specified error
16     →   message.
17     /// </summary>
18     /// <param name="message">The error message.</param>
19     public TelegramParserUnsupportedException(string message)
20       : base(message) { }
21     /// <summary>
22     /// Instantiates a new instance of the <see
23     →   cref="TelegramParserUnsupportedException"/> class with a specified error
24     →   message and an inner exception.
25     /// </summary>
26     /// <param name="message">The error message.</param>
27     /// <param name="inner">The inner exception.</param>
28     public TelegramParserUnsupportedException(string message, Exception inner)
29       : base(message, inner) { }
30   }

```

Listing A.7.3:

EasonEetwViewer.Dmdata.Telegram/Exceptions/TelegramParserUnsupportedException.cs

```

1  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
2  using EasonEetwViewer.Dmdata.Telegram.Abstractions;
3  using EasonEetwViewer.Dmdata.Telegram.Services;
4  using Microsoft.Extensions.DependencyInjection;
5  using Microsoft.Extensions.Logging;
6
7  namespace EasonEetwViewer.Dmdata.Telegram.Extensions;
8  /// <summary>
9  /// Provides extension methods for <see cref="IServiceCollection"/> to add telegram
10 → retriever.
11  /// </summary>
12  public static class TelegramRetrieverServiceCollectionExtensions
13  {
14      /// <summary>
15      /// Injects a <see cref="ITelegramRetriever"/> with the given base URI.
16      /// </summary>
17      /// <param name="services">The instance of <see cref="IServiceCollection"/> for
18      →   the service to be injected.</param>
19      /// <param name="baseUri">The base URI for telegrams.</param>
20      /// <returns>The <see cref="IServiceCollection"/> where the service is injected,
21      →   for chained calls.</returns>
22      public static IServiceCollection AddTelegramRetriever(this IServiceCollection
23      →   services, string baseUri)
24      => services
25          .AddSingleton<ITelegramParser, TelegramParser>()
26          .AddSingleton<ITelegramRetriever>(sp
27          => new TelegramRetriever(
28              baseUri,
              sp.GetRequiredService<ITelegramParser>(),
              sp.GetRequiredService<ILogger<TelegramRetriever>>(),
              sp.GetRequiredService<IAuthenticationHelper>())));

```

Listing A.7.4: EasonEetwViewer.Dmdata.Telegram/Extensions/TelegramRetrieverServiceCollectionExtensions.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
4  public record Tsunami
5  {
6      [JsonPropertyName("forecasts")]
7      public IEnumerable<Forecast>? Forecasts { get; init; }
8      //#[JsonPropertyName("observations")]
9      //public I Enumerable<Observation>? Observations { get; init; }
10     //#[JsonPropertyName("estimations")]
11     //public I Enumerable<Estimation>? Estimations { get; init; }
12 }

```

Listing A.7.5: EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Tsunami.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
5  public record FirstHeight
6  {
7      [JsonPropertyName("arrivalTime")]
8      public DateTimeOffset? ArrivalTime { get; init; }
9      [JsonPropertyName("condition")]
10     public FirstHeightCondition? Condition { get; init; }
11     [JsonPropertyName("revise")]
12     public Revise? Revise { get; init; }
13 }

```

Listing A.7.6: EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/FirstHeight.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
4  public record Kind
5  {
6      [JsonPropertyName("code")]
7      public required string Code { get; init; }
8      [JsonPropertyName("name")]
9      public required string Name { get; init; }
10     [JsonPropertyName("lastKind")]
11     public required SimpleKind LastKind { get; init; }
12 }

```

Listing A.7.7: EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Kind.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
4  public record Forecast
5  {
6      [JsonPropertyName("code")]
7      public required string Code { get; init; }
8      [JsonPropertyName("name")]

```

```

9     public required string Name { get; init; }
10    [JsonPropertyName("kind")]
11    public required Kind Kind { get; init; }
12    [JsonPropertyName("firstHeight")]
13    public FirstHeight? FirstHeight { get; init; }
14    [JsonPropertyName("maxHeight")]
15    public MaxHeight? MaxHeight { get; init; }
16    //#[JsonPropertyName("stations")]
17    //#[public IEnumerable<Station>? Stations { get; init; }]
18 }

```

Listing A.7.8: EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Forecast.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
5  public record Body
{
6
7      [JsonPropertyName("tsunami")]
8      public Tsunami? Tsunami { get; init; }
9      [JsonPropertyName("earthquakes")]
10     public IEnumerable<EarthquakeComponent>? Earthquakes { get; init; }
11     [JsonPropertyName("text")]
12     public string? Text { get; init; }
13     [JsonPropertyName("comments")]
14     public Comments? Comments { get; init; }
15 }

```

Listing A.7.9: EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/Body.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
4  public record SimpleKind
{
5
6      [JsonPropertyName("code")]
7      public required string Code { get; init; }
8      [JsonPropertyName("name")]
9      public required string Name { get; init; }
10 }

```

Listing A.7.10: EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/SimpleKind.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
5  public record MaxHeight
{
6
7      [JsonPropertyName("height")]
8      public required TsunamiHeight Height { get; init; }
9      [JsonPropertyName("condition")]
10     public MaxHeightCondition? Condition { get; init; }
11     [JsonPropertyName("revise")]
12     public Revise? Revise { get; init; }
13 }

```

Listing A.7.11: EasonEetwViewer.Dmdata.Telegram/Dtos/TsunamiInformation/MaxHeight.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telemgram.Dtos.TelemgramBase;
3
4  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation;
5  public record Comments
6  {
7      [JsonPropertyName("free")]
8      public string? FreeText { get; init; }
9      [JsonPropertyName("warning")]
10     public AdditionalComment? Warning { get; init; }
11 }

```

Listing A.7.12: EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation/Comments.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation;
5  public record TsunamiHeight
6  {
7      [JsonPropertyName("type")]
8      public string Type { get; } = "津波の高さ";
9      [JsonPropertyName("unit")]
10     public string Unit { get; } = "m";
11     [JsonPropertyName("value")]
12     public required double? Value { get; init; }
13     [JsonPropertyName("over")]
14     public bool? Over { get; init; }
15     [JsonPropertyName("condition")]
16     public HeightCondition? Condition { get; init; }
17 }

```

Listing A.7.13: EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation/TsunamiHeight.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation.Enum;
4  [JsonConverter(typeof(JsonStringEnumConverter<Revise>))]
5  public enum Revise
6  {
7      [JsonStringEnumMemberName("追加")]
8      Add,
9      [JsonStringEnumMemberName("更新")]
10     Renew
11 }

```

Listing A.7.14: EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation/Enum/Revise.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation.Enum;

```

```

4 [JsonConverter(typeof(JsonStringEnumConverter<MaxHeightCondition>))]
5 public enum MaxHeightCondition
6 {
7     [JsonStringEnumMemberName(" 重要")]
8     Important
9 }
10 }
```

Listing A.7.15:

EasonEetwViewer.Dmdata.Telemgram/Dtos/TsunamiInformation/Enum/MaxHeightCondition.cs

```

1 using System.Text.Json.Serialization;
2
3 namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation.Enum;
4 [JsonConverter(typeof(JsonStringEnumConverter<HeightCondition>))]
5 public enum HeightCondition
6 {
7     [JsonStringEnumMemberName(" 高い")]
8     High,
9     [JsonStringEnumMemberName(" 巨大")]
10    Huge
11 }
```

Listing A.7.16:

EasonEetwViewer.Dmdata.Telemgram/Dtos/TsunamiInformation/Enum/HeightCondition.cs

```

1 using System.Text.Json.Serialization;
2
3 namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.TsunamiInformation.Enum;
4
5 [JsonConverter(typeof(JsonStringEnumConverter<FirstHeightCondition>))]
6 public enum FirstHeightCondition
7 {
8     [JsonStringEnumMemberName(" 津波到達中と推測")]
9     Approaching,
10    [JsonStringEnumMemberName(" ただちに津波来襲と予測")]
11    Striking,
12    [JsonStringEnumMemberName(" 第1波の到達を確認")]
13    FirstWaveConfirmed
14 }
```

Listing A.7.17:

EasonEetwViewer.Dmdata.Telemgram/Dtos/TsunamiInformation/Enum/FirstHeightCondition.cs

```

1 using System.Text.Json.Serialization;
2 using EasonEetwViewer.Dmdata.Dtos.Enum;
3 using EasonEetwViewer.Dmdata.Telemgram.Dtos.EarthquakeInformation.Enum;
4
5 namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.EarthquakeInformation;
6
7 public record RegionIntensity
8 {
9     [JsonPropertyName("code")]
10    public required string Code { get; init; }
11    [JsonPropertyName("name")]
12 }
```

```

12     public required string Name { get; init; }
13     [JsonPropertyName("maxInt")]
14     public Intensity? MaxInt { get; init; }
15     [JsonPropertyName("maxLgInt")]
16     public LgIntensity? MaxLgInt { get; init; }
17     [JsonPropertyName("revise")]
18     public ReviseStatus? Revise { get; init; }
19 }
```

Listing A.7.18:

EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/RegionIntensity.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
4  public record Sva
{
5
6    [JsonPropertyName("unit")]
7    public string Unit { get; } = "cm/s";
8    [JsonPropertyName("value")]
9    public required float Value { get; init; }
10 }
```

Listing A.7.19: EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Sva.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
5  public record Body
{
6
7    [JsonPropertyName("earthquake")]
8    public EarthquakeComponent? Earthquake { get; init; }
9    [JsonPropertyName("intensity")]
10   public IntensityDetails? Intensity { get; init; }
11   [JsonPropertyName("text")]
12   public string? Text { get; init; }
13   [JsonPropertyName("comments")]
14   public Comments? Comments { get; init; }
15 }
```

Listing A.7.20: EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Body.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
4  public record PeriodicBand
{
5
6    [JsonPropertyName("unit")]
7    public string Unit { get; } = "秒台";
8    [JsonPropertyName("value")]
9    public required int Value { get; init; }
10 }
```

Listing A.7.21:

EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/PeriodicBand.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
5  public record PrePeriod
{
6      [JsonPropertyName("periodicBand")]
7      public required PeriodicBand Band { get; init; }
8      [JsonPropertyName("lgInt")]
9      public required LgIntensity LgInt { get; init; }
10     [JsonPropertyName("sva")]
11     public required Sva Sva { get; init; }
12 }
13 }
```

Listing A.7.22: EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/PrePeriod.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
5  public record IntensityDetails
{
6      [JsonPropertyName("maxInt")]
7      public required Intensity MaxInt { get; init; }
8      [JsonPropertyName("maxLgInt")]
9      public LgIntensity? MaxLgInt { get; init; }
10     [JsonPropertyName("lgCategory")]
11     public LgCategory? LgCategory { get; init; }
12     [JsonPropertyName("prefectures")]
13     public required IEnumerable<RegionIntensity> Prefectures { get; init; }
14     [JsonPropertyName("regions")]
15     public required IEnumerable<RegionIntensity> Regions { get; init; }
16     [JsonPropertyName("cities")]
17     public required IEnumerable<CityIntensity> Cities { get; init; }
18     [JsonPropertyName("stations")]
19     public required IEnumerable<StationIntensity> Stations { get; init; }
20 }
21 }
```

Listing A.7.23:
EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/IntensityDetails.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.TelegramBase;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
5  public record Comments
{
6      [JsonPropertyName("free")]
7      public string? FreeText { get; init; }
8      [JsonPropertyName("forecast")]
9      public AdditionalComment? Forecast { get; init; }
10     [JsonPropertyName("var")]
11     public AdditionalComment? Var { get; init; }
12 }
13 }
```

Listing A.7.24: EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Comments.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation.Enum;
4
5  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
6  public record StationIntensity
7  {
8      [JsonPropertyName("code")]
9      public required string Code { get; init; }
10     [JsonPropertyName("name")]
11     public required string Name { get; init; }
12     [JsonPropertyName("int")]
13     public IntensityWithUnreceived? MaxInt { get; init; }
14     [JsonPropertyName("lgInt")]
15     public LgIntensity? LgInt { get; init; }
16     [JsonPropertyName("sva")]
17     public Sva? Sva { get; init; }
18     [JsonPropertyName("prePeriods")]
19     public IEnumerable<PrePeriod>? PrePeriods { get; init; }
20     [JsonPropertyName("revise")]
21     public ReviseStatus? Revise { get; init; }
22     [JsonPropertyName("condition")]
23     public string? Condition { get; init; }
24 }

```

Listing A.7.25:

EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/StationIntensity.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation.Enum;
4
5  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
6  public record CityIntensity
7  {
8      [JsonPropertyName("code")]
9      public required string Code { get; init; }
10     [JsonPropertyName("name")]
11     public required string Name { get; init; }
12     [JsonPropertyName("maxInt")]
13     public Intensity? MaxInt { get; init; }
14     [JsonPropertyName("revise")]
15     public ReviseStatus? Revise { get; init; }
16     [JsonPropertyName("condition")]
17     public string? Condition { get; init; }
18 }

```

Listing A.7.26:

EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/CityIntensity.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation.Enum;
4
5  [JsonConverter(typeof(JsonStringEnumConverter<ReviseStatus>))]
6  public enum ReviseStatus
7  {

```

```
8     [JsonStringEnumMemberName("上方修正")]
9     Increase,
10    [JsonStringEnumMemberName("追加")]
11    Additional
12 }
```

Listing A.7.27:

EasonEetwViewer.Dmdata.Telemgram.Dtos.EarthquakeInformation.Enum/ReviseStatus.cs

```
1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.EarthquakeInformation.Enum;
4
5  [JsonConverter(typeof(JsonStringEnumConverter<IntensityWithUnreceived>))]
6  public enum IntensityWithUnreceived
7  {
8      /// <summary>
9      /// The value <c>1</c>, representing intensity 1.
10     /// </summary>
11     [JsonStringEnumMemberName("1")]
12     One,
13     /// <summary>
14     /// The value <c>2</c>, representing intensity 2.
15     /// </summary>
16     [JsonStringEnumMemberName("2")]
17     Two,
18     /// <summary>
19     /// The value <c>3</c>, representing intensity 3.
20     /// </summary>
21     [JsonStringEnumMemberName("3")]
22     Three,
23     /// <summary>
24     /// The value <c>4</c>, representing intensity 4.
25     /// </summary>
26     [JsonStringEnumMemberName("4")]
27     Four,
28     /// <summary>
29     /// The value <c>5-</c>, representing intensity 5 weak.
30     /// </summary>
31     [JsonStringEnumMemberName("5-")]
32     FiveWeak,
33     /// <summary>
34     /// The value <c>5+</c>, representing intensity 5 strong.
35     /// </summary>
36     [JsonStringEnumMemberName("5+")]
37     FiveStrong,
38     /// <summary>
39     /// The value <c>6-</c>, representing intensity 6 weak.
40     /// </summary>
41     [JsonStringEnumMemberName("6-")]
42     SixWeak,
43     /// <summary>
44     /// The value <c>6+</c>, representing intensity 6 strong.
45     /// </summary>
46     [JsonStringEnumMemberName("6+")]
47     SixStrong,
48     /// <summary>
49     /// The value <c>7</c>, representing intensity 7.
```

```

50     /// </summary>
51     [JsonStringEnumMemberName("7")]
52     Seven,
53     [JsonStringEnumMemberName("!5-")]
54     Unreceived,
55
56 }

```

Listing A.7.28: EasonEetwViewer.Dmdata.Telegram/Dtos/EarthquakeInformation/Enum/Intensit
yWithUnreceived.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.TelogramBase;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
4
5  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
6  public record TsunamiInformationSchema : Head
7  {
8      [JsonPropertyName("body")]
9      public required Body Body { get; init; }
10 }

```

Listing A.7.29: EasonEetwViewer.Dmdata.Telegram/Dtos/Schema/TsunamiInformationSchema.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.TelogramBase;
4
5  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
6  public record EewInformationSchema : Head
7  {
8      [JsonPropertyName("body")]
9      public required Body Body { get; init; }
10 }

```

Listing A.7.30: EasonEetwViewer.Dmdata.Telegram/Dtos/Schema/EewInformationSchema.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.TelogramBase;
4
5  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
6  public record EarthquakeInformationSchema : Head
7  {
8      [JsonPropertyName("body")]
9      public required Body Body { get; init; }
10 }

```

Listing A.7.31:
EasonEetwViewer.Dmdata.Telegram/Dtos/Schema/EarthquakeInformationSchema.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;

```

```

4  public record FromTo<TFrom, TTo>
5      where TFrom : struct, System.Enum
6      where TTo : struct, System.Enum
7  {
8      [JsonPropertyName("from")]
9      public required TFrom From { get; init; }
10     [JsonPropertyName(name: "to")]
11     public required TTo To { get; init; }
12 }

```

Listing A.7.32: EasonEetwViewer.Dmdata.Telemgram/Dtos/EewInformation/FromTo.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telemgram.Dtos.EewInformation.Enum.Range;
3
4  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.EewInformation;
5  public record Region
6  {
7      [JsonPropertyName("code")]
8      public required string Code { get; init; }
9      [JsonPropertyName("name")]
10     public required string Name { get; init; }
11     [JsonPropertyName("isPlum")]
12     public required bool IsPlum { get; init; }
13     [JsonPropertyName("isWarning")]
14     public required bool IsWarning { get; init; }
15
16     [JsonPropertyName("forecastMaxInt")]
17     public required FromTo<IntensityLower, IntensityUpper> ForecastMaxInt { get;
18         ← init; }
19     [JsonPropertyName("forecastMaxLgInt")]
20     public FromTo<LgIntensityLower, LgIntensityUpper>? ForecastMaxLgInt { get; init; }
21     [JsonPropertyName("kind")]
22     public required SimpleKind Kind { get; init; }
23     [JsonPropertyName("condition")]
24     public string? Condition { get; init; }
25     [JsonPropertyName("arrivalTime")]
26     public DateTimeOffset? ArrivalTime { get; init; }
}

```

Listing A.7.33: EasonEetwViewer.Dmdata.Telemgram/Dtos/EewInformation/Region.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.EewInformation;
4  public record ReducedName
5  {
6      [JsonPropertyName("code")]
7      public required string Code { get; init; }
8      [JsonPropertyName("name")]
9      public required string Name { get; init; }
10 }

```

Listing A.7.34: EasonEetwViewer.Dmdata.Telemgram/Dtos/EewInformation/ReducedName.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Change;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
5  public record Appendix
{
6      [JsonPropertyName("maxIntChange")]
7      public required MaxInt MaxIntensityChange { get; init; }
8      [JsonPropertyName("maxLgIntChange")]
9      public MaxInt? MaxLgIntensityChange { get; init; }
10     [JsonPropertyName("maxIntChangeReason")]
11     public required Reason MaxIntensityChangeReason { get; init; }
12 }
13 }
```

Listing A.7.35: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Appendix.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
5  public record Earthquake
{
6      [JsonPropertyName("originTime")]
7      public DateTimeOffset? OriginTime { get; init; }
8      [JsonPropertyName("arrivalTime")]
9      public required DateTimeOffset ArrivalTime { get; init; }
10     [JsonPropertyName("condition")]
11     public string? Condition { get; init; }
12     [JsonPropertyName("hypocenter")]
13     public required Hypocentre Hypocentre { get; init; }
14     [JsonPropertyName("magnitude")]
15     public required Magnitude Magnitude { get; init; }
16 }
17 }
```

Listing A.7.36: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Earthquake.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
4  public record Body
{
5      [JsonPropertyName("isLastInfo")]
6      public required bool IsLastInfo { get; init; }
7      [JsonPropertyName("isCancelled")]
8      public required bool IsCancelled { get; init; }
9      [JsonPropertyName("isWarning")]
10     public bool? IsWarning { get; init; }
11     [JsonPropertyName("zones")]
12     public IEnumerable<Area>? Zones { get; init; }
13     [JsonPropertyName("prefectures")]
14     public IEnumerable<Area>? Prefectures { get; init; }
15     [JsonPropertyName("regions")]
16     public IEnumerable<Area>? Regions { get; init; }
17     [JsonPropertyName("earthquake")]
18     public Earthquake? Earthquake { get; init; }
19     [JsonPropertyName("intensity")]
20     public IntensityInfo? Intensity { get; init; }
21 }
```

```

22     [JsonPropertyName("text")]
23     public string? Text { get; init; }
24     [JsonPropertyName("comments")]
25     public Comments? Comments { get; init; }
26 }
```

Listing A.7.37: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Body.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
5  public record Hypocentre
{
6
7      [JsonPropertyName("code")]
8      public required string Code { get; init; }
9      [JsonPropertyName("name")]
10     public required string Name { get; init; }
11     [JsonPropertyName("coordinate")]
12     public required CoordinateComponent Coordinate { get; init; }
13     [JsonPropertyName("depth")]
14     public required Depth Depth { get; init; }
15     [JsonPropertyName("reduce")]
16     public required ReducedName ShortName { get; init; }
17     [JsonPropertyName("landOrSea")]
18     public Enum.HypocentrePosition? Position { get; init; }
19     [JsonPropertyName("accuracy")]
20     public required Accuracy Accuracy { get; init; }
21 }
```

Listing A.7.38: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Hypocentre.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
4  public record SimpleKind
{
5
6      [JsonPropertyName("code")]
7      public required string Code { get; init; }
8      [JsonPropertyName("name")]
9      public required string Name { get; init; }
10 }
```

Listing A.7.39: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/SimpleKind.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
4  public record AreaKind
{
5
6      [JsonPropertyName("code")]
7      public string Code { get; } = "31";
8      [JsonPropertyName("name")]
9      public string Name { get; } = "緊急地震速報（警報）";
10     [JsonPropertyName("lastKind")]
11     public required SimpleKind LastKind { get; init; }
12 }
```

Listing A.7.40: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/AreaKind.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Accuracy;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
5  public record Accuracy
6  {
7      [JsonPropertyName("epicenters")]
8      public required EpicentreDepth[] Epicentres { get; init; }
9      [JsonPropertyName("depth")]
10     public required EpicentreDepth Depth { get; init; }
11     [JsonPropertyName("magnitudeCalculation")]
12     public required Magnitude Magnitude { get; init; }
13     [JsonPropertyName("numberOfMagnitudeCalculation")]
14     public required MagnitudePoint MagnitudePoint { get; init; }
15 }

```

Listing A.7.41: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Accuracy.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
4  public record Area
5  {
6      [JsonPropertyName("code")]
7      public required string Code { get; init; }
8      [JsonPropertyName("name")]
9      public required string Name { get; init; }
10     [JsonPropertyName("kind")]
11     public required AreaKind Kind { get; init; }
12 }

```

Listing A.7.42: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Area.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.TelegramBase;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
5  public record Comments
6  {
7      [JsonPropertyName("free")]
8      public string? FreeText { get; init; }
9      [JsonPropertyName("warning")]
10     public AdditionalComment? Warning { get; init; }
11 }

```

Listing A.7.43: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Comments.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
4  public record Station

```

```

5  {
6      [JsonPropertyName("code")]
7      public required string Code { get; init; }
8      [JsonPropertyName("name")]
9      public required string Name { get; init; }
10     [JsonPropertyName("int")]
11     public required string Int { get; init; }
12     [JsonPropertyName("k")]
13     public required float K { get; init; }
14 }
```

Listing A.7.44: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Station.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Range;
3
4  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
5  public record IntensityInfo
{
6      [JsonPropertyName("forecastMaxInt")]
7      public required FromTo<IntensityLower, IntensityUpper> ForecastMaxInt { get;
8          ← init; }
9      [JsonPropertyName("forecastMaxLgInt")]
10     public FromTo<LgIntensityLower, LgIntensityUpper>? ForecastMaxLgInt { get; init; }
11     [JsonPropertyName("appendix")]
12     public Appendix? Appendix { get; init; }
13     [JsonPropertyName("regions")]
14     public required IEnumerable<Region> Regions { get; init; }
15     [JsonPropertyName("stations")]
16     public IEnumerable<Station>? Stations { get; init; }
17 }
```

Listing A.7.45: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/IntensityInfo.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum;
4
5  [JsonConverter(typeof(JsonStringEnumConverter<HypocentrePosition>))]
6  public enum HypocentrePosition
{
7      [JsonStringEnumMemberName(" 内陸")]
8      Land,
9      [JsonStringEnumMemberName(" 海域")]
10     Sea
11 }
12 }
```

Listing A.7.46:

EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/HypocentrePosition.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Change;
4  [JsonConverter(typeof(JsonStringEnumConverter<Reason>))]
5  public enum Reason
```

```

6  {
7      [JsonStringEnumMemberName("0")]
8      None = 0,
9      [JsonStringEnumMemberName("1")]
10     Magnitude = 1,
11     [JsonStringEnumMemberName("2")]
12     Position = 2,
13     [JsonStringEnumMemberName("3")]
14     MagnitudeAndPosition = 3,
15     [JsonStringEnumMemberName("4")]
16     Depth = 4,
17     [JsonStringEnumMemberName("9")]
18     Plum = 9
19 }

```

Listing A.7.47: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Change/Reason.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Change;
4  [JsonConverter(typeof(JsonStringEnumConverter<MaxInt>))]
5  public enum MaxInt
6  {
7      [JsonStringEnumMemberName("0")]
8      NoChange = 0,
9      [JsonStringEnumMemberName("1")]
10     Increase = 1,
11     [JsonStringEnumMemberName("2")]
12     Decrease = 2
13 }

```

Listing A.7.48: EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Change/MaxInt.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Accuracy;
4  [JsonConverter(typeof(JsonStringEnumConverter<Magnitude>))]
5  public enum Magnitude
6  {
7      [JsonStringEnumMemberName("0")]
8      Unknown = 0,
9      [JsonStringEnumMemberName("2")]
10     SpeedMagnitude = 2,
11     [JsonStringEnumMemberName("3")]
12     FullPPhase = 3,
13     [JsonStringEnumMemberName("4")]
14     FullPPhaseMixed = 4,
15     [JsonStringEnumMemberName("5")]
16     FullPointPhase = 5,
17     [JsonStringEnumMemberName("6")]
18     Epos = 6,
19     [JsonStringEnumMemberName("8")]
20     LevelOrPlum = 8
21 }

```

Listing A.7.49:

EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Accuracy/Magnitude.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Accuracy;
4  [JsonConverter(typeof(JsonStringEnumConverter<MagnitudePoint>))]
5  public enum MagnitudePoint
{
6      [JsonStringEnumMemberName("0")]
7      Unknown = 0,
8      [JsonStringEnumMemberName("1")]
9      OneOrLevelOrPlum = 1,
10     [JsonStringEnumMemberName("2")]
11     Two = 2,
12     [JsonStringEnumMemberName("3")]
13     Three = 3,
14     [JsonStringEnumMemberName("4")]
15     Four = 4,
16     [JsonStringEnumMemberName("5")]
17     FiveOrAbove = 5
18 }
19 }
```

Listing A.7.50:

EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Accuracy/MagnitudePoint.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Accuracy;
4  [JsonConverter(typeof(JsonStringEnumConverter<EpicentreDepth>))]
5  public enum EpicentreDepth
{
6      [JsonStringEnumMemberName("0")]
7      Unknown = 0,
8      [JsonStringEnumMemberName("1")]
9      LevelIpf1Plum = 1,
10     [JsonStringEnumMemberName("2")]
11     Ipf2 = 2,
12     [JsonStringEnumMemberName("3")]
13     Ipf30Or4 = 3,
14     [JsonStringEnumMemberName("4")]
15     Ipf50OrMore = 4,
16     [JsonStringEnumMemberName("5")]
17     [Obsolete("No longer used after 2023/09/26 14:00 JST.", true)]
18     Bosai40OrLess = 5,
19     [JsonStringEnumMemberName("6")]
20     [Obsolete("No longer used after 2023/09/26 14:00 JST.", true)]
21     Bosai50OrMoreHinet = 6,
22     [JsonStringEnumMemberName("7")]
23     [Obsolete("No longer used after 2023/09/26 14:00 JST.", true)]
24     EposSea = 7,
25     [JsonStringEnumMemberName("8")]
26     [Obsolete("No longer used after 2023/09/26 14:00 JST.", true)]
27     EposLand = 8,
28     [JsonStringEnumMemberName("9")]
29     Final = 9
30 }
31 }
```

Listing A.7.51:

EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Accuracy/EpicentreDepth.cs

```
1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.EewInformation.Enum.Range;
4  [JsonConverter(typeof(JsonStringEnumConverter<IntensityUpper>))]
5  public enum IntensityUpper
6  {
7      [JsonPropertyEnumMemberName("0")]
8      Zero,
9      /// <summary>
10     /// The value <c>1</c>, representing intensity 1.
11     /// </summary>
12     [JsonPropertyEnumMemberName("1")]
13     One,
14     /// <summary>
15     /// The value <c>2</c>, representing intensity 2.
16     /// </summary>
17     [JsonPropertyEnumMemberName("2")]
18     Two,
19     /// <summary>
20     /// The value <c>3</c>, representing intensity 3.
21     /// </summary>
22     [JsonPropertyEnumMemberName("3")]
23     Three,
24     /// <summary>
25     /// The value <c>4</c>, representing intensity 4.
26     /// </summary>
27     [JsonPropertyEnumMemberName("4")]
28     Four,
29     /// <summary>
30     /// The value <c>5-</c>, representing intensity 5 weak.
31     /// </summary>
32     [JsonPropertyEnumMemberName("5-")]
33     FiveWeak,
34     /// <summary>
35     /// The value <c>5+</c>, representing intensity 5 strong.
36     /// </summary>
37     [JsonPropertyEnumMemberName("5+")]
38     FiveStrong,
39     /// <summary>
40     /// The value <c>6-</c>, representing intensity 6 weak.
41     /// </summary>
42     [JsonPropertyEnumMemberName("6-")]
43     SixWeak,
44     /// <summary>
45     /// The value <c>6+</c>, representing intensity 6 strong.
46     /// </summary>
47     [JsonPropertyEnumMemberName("6+")]
48     SixStrong,
49     /// <summary>
50     /// The value <c>7</c>, representing intensity 7.
51     /// </summary>
52     [JsonPropertyEnumMemberName("7")]
53     Seven,
54     [JsonPropertyEnumMemberName(" 不明")]
55     Unclear,
56     [JsonPropertyEnumMemberName("over")]
57     Above
58 }
```

Listing A.7.52:

EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Range/IntensityUpper.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Range;
4  [JsonConverter(typeof(JsonStringEnumConverter<LgIntensityUpper>))]
5  public enum LgIntensityUpper
{
6
7      /// <summary>
8      /// The value <c>0</c>, representing LPGM intensity 0.
9      /// </summary>
10     [JsonStringEnumMemberName("0")]
11     Zero,
12     /// <summary>
13     /// The value <c>1</c>, representing LPGM intensity 1.
14     /// </summary>
15     [JsonStringEnumMemberName("1")]
16     One,
17     /// <summary>
18     /// The value <c>2</c>, representing LPGM intensity 2.
19     /// </summary>
20     [JsonStringEnumMemberName("2")]
21     Two,
22     /// <summary>
23     /// The value <c>3</c>, representing LPGM intensity 3.
24     /// </summary>
25     [JsonStringEnumMemberName("3")]
26     Three,
27     /// <summary>
28     /// The value <c>4</c>, representing LPGM intensity 4.
29     /// </summary>
30     [JsonStringEnumMemberName("4")]
31     Four,
32     [JsonStringEnumMemberName(" 不明")]
33     Unclear,
34     [JsonStringEnumMemberName("over")]
35     Above
36 }

```

Listing A.7.53:

EasonEetwViewer.Dmdata.Telegram/Dtos/EewInformation/Enum/Range/LgIntensityUpper.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Range;
4  [JsonConverter(typeof(JsonStringEnumConverter<IntensityLower>))]
5  public enum IntensityLower
{
6
7      [JsonStringEnumMemberName("0")]
8      Zero,
8      /// <summary>
9      /// The value <c>1</c>, representing intensity 1.
10     /// </summary>
11     [JsonStringEnumMemberName("1")]
12     One,

```

```

14     ///<summary>
15     /// The value <c>2</c>, representing intensity 2.
16     ///</summary>
17     [JsonStringEnumMemberName("2")]
18     Two,
19     ///<summary>
20     /// The value <c>3</c>, representing intensity 3.
21     ///</summary>
22     [JsonStringEnumMemberName("3")]
23     Three,
24     ///<summary>
25     /// The value <c>4</c>, representing intensity 4.
26     ///</summary>
27     [JsonStringEnumMemberName("4")]
28     Four,
29     ///<summary>
30     /// The value <c>5-</c>, representing intensity 5 weak.
31     ///</summary>
32     [JsonStringEnumMemberName("5-")]
33     FiveWeak,
34     ///<summary>
35     /// The value <c>5+</c>, representing intensity 5 strong.
36     ///</summary>
37     [JsonStringEnumMemberName("5+")]
38     FiveStrong,
39     ///<summary>
40     /// The value <c>6-</c>, representing intensity 6 weak.
41     ///</summary>
42     [JsonStringEnumMemberName("6-")]
43     SixWeak,
44     ///<summary>
45     /// The value <c>6+</c>, representing intensity 6 strong.
46     ///</summary>
47     [JsonStringEnumMemberName("6+")]
48     SixStrong,
49     ///<summary>
50     /// The value <c>7</c>, representing intensity 7.
51     ///</summary>
52     [JsonStringEnumMemberName("7")]
53     Seven,
54     [JsonStringEnumMemberName(" 不明")]
55     Unclear
56 }

```

Listing A.7.54:

EasonEetwViewer.Dmdata.Telogram.Dtos.EewInformation.Enum.Range/IntensityLower.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telogram.Dtos.EewInformation.Enum.Range;
4  [JsonConverter(typeof(JsonStringEnumConverter<LgIntensityLower>))]
5  public enum LgIntensityLower
{
6
7     ///<summary>
8     /// The value <c>0</c>, representing LPGM intensity 0.
9     ///</summary>
10    [JsonStringEnumMemberName("0")]
11    Zero,

```

```

12     /// <summary>
13     /// The value <c>1</c>, representing LPGM intensity 1.
14     /// </summary>
15     [JsonStringEnumMemberName("1")]
16     One,
17     /// <summary>
18     /// The value <c>2</c>, representing LPGM intensity 2.
19     /// </summary>
20     [JsonStringEnumMemberName("2")]
21     Two,
22     /// <summary>
23     /// The value <c>3</c>, representing LPGM intensity 3.
24     /// </summary>
25     [JsonStringEnumMemberName("3")]
26     Three,
27     /// <summary>
28     /// The value <c>4</c>, representing LPGM intensity 4.
29     /// </summary>
30     [JsonStringEnumMemberName("4")]
31     Four,
32     [JsonStringEnumMemberName(" 不明")]
33     Unclear
34 }

```

Listing A.7.55:

EasonEetwViewer.Dmdata.Telemgram/Dtos/EewInformation/Enum/Range/LgIntensityLower.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.TelemgramBase;
4  public record AdditionalComment
5  {
6      [JsonPropertyName("text")]
7      public required string Text { get; init; }
8      [JsonPropertyName("codes")]
9      public required IEnumerable<string> Codes { get; init; }
10 }

```

Listing A.7.56: EasonEetwViewer.Dmdata.Telemgram/Dtos/TelemgramBase/AdditionalComment.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Telemgram;
3
4  namespace EasonEetwViewer.Dmdata.Telemgram.Dtos.TelemgramBase;
5
6  public record Head
7  {
8      [JsonPropertyName("_originalId")]
9      public required string OriginalId { get; init; }
10
11     [JsonPropertyName("_schema")]
12     public required SchemaVersionInformation Schema { get; init; }
13     [JsonPropertyName("type")]
14     public required string Type { get; init; }
15     [JsonPropertyName("title")]
16     public required string Title { get; init; }
17     [JsonPropertyName("status")]

```

```

18     public required TelegramStatus Status { get; init; }
19     [JsonPropertyName("infoType")]
20     public required TelegramType InfoType { get; init; }
21     [JsonPropertyName("editorialOffice")]
22     public required string EditorialOffice { get; init; }
23     [JsonPropertyName("publishingOffice")]
24     public required IEnumerable<string> PublishingOffice { get; init; }
25     [JsonPropertyName("pressDateTime")]
26     public required DateTimeOffset PressDateTime { get; init; }
27     [JsonPropertyName("reportDateTime")]
28     public required DateTimeOffset ReportDateTime { get; init; }
29     [JsonPropertyName("targetDateTime")]
30     public required DateTimeOffset? TargetDateTime { get; init; }
31     [JsonPropertyName("targetDateTimeDubious")]
32     public string? TargetDateTimeDubious { get; init; }
33     [JsonPropertyName("targetDuration")]
34     public string? TargetDuration { get; init; }
35     [JsonPropertyName("validDateTime")]
36     public DateTimeOffset? ValidDateTime { get; init; }
37     [JsonPropertyName("eventId")]
38     public required string? EventId { get; init; }
39     [JsonPropertyName("serialNo")]
40     public required string? SerialNo { get; init; }
41     [JsonPropertyName("infoKind")]
42     public required string InfoKind { get; init; }
43     [JsonPropertyName("infoKindVersion")]
44     public required string InfoKindVersion { get; init; }
45     [JsonPropertyName("headline")]
46     public required string? Headline { get; init; }
47 }

```

Listing A.7.57: EasonEetwViewer.Dmdata.Telemgram/Dtos/TelegramBase/Head.cs

```

1  using EasonEetwViewer.Dmdata.Telemgram.Dtos.TelemgramBase;
2
3  namespace EasonEetwViewer.Dmdata.Telemgram.Abstractions;
4  /// <summary>
5  /// Represents a retriever for JSON Telegrams.
6  /// </summary>
7  public interface ITelegramRetriever
8  {
9      /// <summary>
10     /// Retrieves a JSON Telegram with the specified ID.
11     /// </summary>
12     /// <param name="id">The ID of the telegram.</param>
13     /// <returns>The telegram that was parsed.</returns>
14     Task<Head?> GetJsonTelegramAsync(string id);
15 }

```

Listing A.7.58: EasonEetwViewer.Dmdata.Telemgram/Abstractions/ITelegramRetriever.cs

```

1  using EasonEetwViewer.Dmdata.Dtos.Telemgram;
2  using EasonEetwViewer.Dmdata.Telemgram.Dtos.TelemgramBase;
3
4  namespace EasonEetwViewer.Dmdata.Telemgram.Abstractions;
5  /// <summary>
6  /// Represents a parser for JSON Telegrams.

```

```

7  /// </summary>
8  public interface ITelegramParser
9  {
10     /// <summary>
11     /// Parses a JSON Telegram.
12     /// </summary>
13     /// <param name="json">The JSON string to be parsed.</param>
14     /// <returns>The telegram that was parsed.</returns>
15     Head ParseJsonTelegram(string json);
16     /// <summary>
17     /// Gives the corresponding to the <see cref="SchemaVersionInformation"/>.
18     /// </summary>
19     /// <param name="schema">The JSON schema version information.</param>
20     /// <returns>The type of the JSON schema, <see langword="null"/> if
21     → unknown.</returns>
22     Type? ParseSchemaInformation(SchemaVersionInformation schema);
}

```

Listing A.7.59: EasonEetwViewer.Dmdata.Telemgram/Abstractions/ITelegramParser.cs

```

1  using Microsoft.Extensions.Logging;
2
3  namespace EasonEetwViewer.Dmdata.Telemgram.Services;
4
5  /// <summary>
6  /// Represents the log messages used in <see cref="TelegramRetriever"/>.
7  /// </summary>
8  internal static partial class TelegramRetrieverLogs
{
9
10    /// <summary>
11    /// Log when instantiated.
12    /// </summary>
13    /// <param name="logger">The logger to be used.</param>
14    [LoggerMessage(
15      EventId = 0,
16      EventName = nameof(Instantiated),
17      Level = LogLevel.Information,
18      Message = "Instantiated.")]
19    public static partial void Instantiated(
20      this ILogger<TelegramRetriever> logger);
21    /// <summary>
22    /// Log when parser experiences exception.
23    /// </summary>
24    /// <param name="logger">The logger to be used.</param>
25    /// <param name="exception">The exception message thrown by the parser.</param>
26    [LoggerMessage(
27      EventId = 1,
28      EventName = nameof(ParserExceptionIgnored),
29      Level = LogLevel.Warning,
30      Message = "Parser Exception Ignored: `'{Exception}`.")]
31    public static partial void ParserExceptionIgnored(
32      this ILogger<TelegramRetriever> logger, string exception);
33    /// <summary>
34    /// Log when retriever experiences exception.
35    /// </summary>
36    /// <param name="logger">The logger to be used.</param>
37    /// <param name="error">The error message returned by the API Call.</param>
38    [LoggerMessage(

```

```

39     EventId = 2,
40     EventName = nameof(ApiErrorIgnored),
41     Level = LogLevel.Warning,
42     Message = "Error Message from API Ignored: `'{Error}``."]
43 public static partial void ApiErrorIgnored(
44     this ILogger<TelegramRetriever> logger, string error);
45 /// <summary>
46 /// Log when sending request.
47 /// </summary>
48 /// <param name="logger">The logger to be used.</param>
49 /// <param name="id">The telegram ID.</param>
50 [LoggerMessage(
51     EventId = 3,
52     EventName = nameof(Requesting),
53     Level = LogLevel.Debug,
54     Message = "Requesting telegram `'{Id}``."]
55 public static partial void Requesting(
56     this ILogger<TelegramRetriever> logger, string id);
57 /// <summary>
58 /// Log when request successfully sent.
59 /// </summary>
60 /// <param name="logger">The logger to be used.</param>
61 /// <param name="id">The telegram ID.</param>
62 [LoggerMessage(
63     EventId = 4,
64     EventName = nameof(Requested),
65     Level = LogLevel.Information,
66     Message = "Requested telegram `'{Id}``."]
67 public static partial void Requested(
68     this ILogger<TelegramRetriever> logger, string id);
69 /// <summary>
70 /// Log when not authenticated.
71 /// </summary>
72 /// <param name="logger">The logger to be used.</param>
73 [LoggerMessage(
74     EventId = 5,
75     EventName = nameof(NotAuthenticated),
76     Level = LogLevel.Warning,
77     Message = "Not authenticated.`")
78 public static partial void NotAuthenticated(
79     this ILogger<TelegramRetriever> logger);
80 }

```

Listing A.7.60: EasonEtwViewer.Dmdata.Telemgram/Logging/TelegramRetrieverLogs.cs

```

1 using Microsoft.Extensions.Logging;
2
3 namespace EasonEtwViewer.Dmdata.Telemgram.Services;
4
5 /// <summary>
6 /// Represents the log messages used in <see cref="TelegramParser"/>.
7 /// </summary>
8 internal static partial class TelegramParserLogs
{
9
10    /// <summary>
11    /// Log when instantiated.
12    /// </summary>
13    /// <param name="logger">The logger to be used.</param>

```

```
14 [LoggerMessage(
15     EventId = 0,
16     EventName = nameof(Instantiated),
17     Level = LogLevel.Information,
18     Message = "Instantiated.")]
19 public static partial void Instantiated(
20     this ILogger<TelegramParser> logger);
21 /// <summary>
22 /// Log when handling JSON.
23 /// </summary>
24 /// <param name="logger">The logger to be used.</param>
25 /// <param name="json">The JSON data that is being parsed.</param>
26 [LoggerMessage(
27     EventId = 1,
28     EventName = nameof(HandlingJson),
29     Level = LogLevel.Trace,
30     Message = "Handling JSON: `{{Json}}`.")]
31 public static partial void HandlingJson(
32     this ILogger<TelegramParser> logger, string json);
33 /// <summary>
34 /// Log when JSON has incorrect format.
35 /// </summary>
36 /// <param name="logger">The logger to be used.</param>
37 /// <param name="json">The JSON data that is being parsed.</param>
38 [LoggerMessage(
39     EventId = 2,
40     EventName = nameof(IncorrectJsonFormat),
41     Level = LogLevel.Error,
42     Message = "Incorrect Format JSON: `{{Json}}`.")]
43 public static partial void IncorrectJsonFormat(
44     this ILogger<TelegramParser> logger, string json);
45 /// <summary>
46 /// Log when JSON has unsupported schema.
47 /// </summary>
48 /// <param name="logger">The logger to be used.</param>
49 /// <param name="schemaName">The name of the schema.</param>
50 /// <param name="schemaVersion">The version of the schema.</param>
51 [LoggerMessage(
52     EventId = 3,
53     EventName = nameof(UnsupportedSchema),
54     Level = LogLevel.Error,
55     Message = "Unsupported schema: `{{SchemaName}}` `{{SchemaVersion}}`.")]
56 public static partial void UnsupportedSchema(
57     this ILogger<TelegramParser> logger, string schemaName, string schemaVersion);
58 /// <summary>
59 /// Log when JSON has supported schema.
60 /// </summary>
61 /// <param name="logger">The logger to be used.</param>
62 /// <param name="schemaName">The name of the schema.</param>
63 /// <param name="schemaVersion">The version of the schema.</param>
64 [LoggerMessage(
65     EventId = 4,
66     EventName = nameof(SupportedSchema),
67     Level = LogLevel.Information,
68     Message = "Supported schema: `{{SchemaName}}` `{{SchemaVersion}}`.")]
69 public static partial void SupportedSchema(
70     this ILogger<TelegramParser> logger, string schemaName, string schemaVersion);
71 }
```

Listing A.7.61: EasonEetwViewer.Dmdata.Telegram/Logging/TelegramParserLogs.cs

```

1  using System.Net;
2  using System.Net.Http.Headers;
3  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
4  using EasonEetwViewer.Dmdata.Telegram.Abstractions;
5  using EasonEetwViewer.Dmdata.Telegram.Dtos.TelegramBase;
6  using EasonEetwViewer.Dmdata.Telegram.Exceptions;
7  using Microsoft.Extensions.Logging;
8
9  namespace EasonEetwViewer.Dmdata.Telegram.Services;
10 /// <summary>
11 /// The default implementation of <see cref="ITelegramRetriever"/>.
12 /// </summary>
13 internal sealed class TelegramRetriever : ITelegramRetriever
14 {
15     /// <summary>
16     /// The <see cref="HttpClient"/> to be used.
17     /// </summary>
18     private readonly HttpClient _client;
19     /// <summary>
20     /// The parser for JSON to be used.
21     /// </summary>
22     private readonly ITelegramParser _parser;
23     /// <summary>
24     /// The authenticator to be used.
25     /// </summary>
26     private readonly IAuthenticationHelper _authenticator;
27     /// <summary>
28     /// The logger to be used.
29     /// </summary>
30     private readonly ILogger<TelegramRetriever> _logger;
31     /// <summary>
32     /// Initializes a new instance of the <see cref="TelegramRetriever"/> class.
33     /// </summary>
34     /// <param name="baseApi">The base API of the telegram to be retrieved.</param>
35     /// <param name="parser">The parser to be used to parse JSON telegrams.</param>
36     /// <param name="logger">The logger to be used.</param>
37     /// <param name="authenticator">The authenticator to be used.</param>
38     public TelegramRetriever(string baseApi, ITelegramParser parser,
39         → ILogger<TelegramRetriever> logger, IAuthenticationHelper authenticator)
40     {
41         _client = new()
42         {
43             BaseAddress = new(baseApi)
44         };
45         _authenticator = authenticator;
46         _parser = parser;
47         _logger = logger;
48     }
49     /// <inheritDoc/>
50     public async Task<Head?> GetJsonTelegramAsync(string id)
51     {
52         using HttpRequestMessage request = new(HttpMethod.Get, $"{id}");
53         AuthenticationHeaderValue? authenticationHeaderValue = await
54             → _authenticator.GetAuthenticationHeaderAsync();
55         if (authenticationHeaderValue is null)
56         {
57             _logger.NotAuthenticated();

```

```

56     return null;
57 }
58
59     request.Headers.Authorization = authenticationHeaderValue;
60     try
61     {
62         _logger.Requesting(id);
63         using HttpResponseMessage response = await _client.SendAsync(request);
64         _logger.Requested(id);
65         string responseBody = await response.Content.ReadAsStringAsync();
66         if (response.IsSuccessStatusCode)
67         {
68             return _parser.ParseJsonTelegram(responseBody);
69         }
70         else
71         {
72             _logger.ApiErrorIgnored(responseBody);
73
74             if (response.StatusCode is HttpStatusCode.Unauthorized or
75                 HttpStatusCode.Forbidden)
76             {
77                 await _authenticator.InvalidAuthenticatorAsync(responseBody);
78             }
79
80             return null;
81         }
82     catch (TelegramParserException ex)
83     {
84         _logger.ParserExceptionIgnored(ex.ToString());
85         return null;
86     }
87 }
88 }
```

Listing A.7.62: EasonEetwViewer.Dmdata.Telemgram/Services/TelogramRetriever.cs

```

1  using System.Text.Json;
2  using EasonEetwViewer.Dmdata.Dtos.Telemgram;
3  using EasonEetwViewer.Dmdata.Telemgram.Abstractions;
4  using EasonEetwViewer.Dmdata.Telemgram.Dtos.Schema;
5  using EasonEetwViewer.Dmdata.Telemgram.Dtos.TelemgramBase;
6  using EasonEetwViewer.Dmdata.Telemgram.Exceptions;
7  using Microsoft.Extensions.Logging;
8
9  namespace EasonEetwViewer.Dmdata.Telemgram.Services;
10 /// <summary>
11 /// The default implementation of the <see cref="ITelegramParser"/> interface.
12 /// </summary>
13 public sealed class TelegramParser : ITelegramParser
14 {
15     /// <summary>
16     /// The JSON serialisation options to be used.
17     /// </summary>
18     private readonly JsonSerializerOptions _options;
19     /// <summary>
20     /// The logger to be used for logging.
21     /// </summary>
```

```
22     private readonly ILogger<TelegramParser> _logger;
23     /// <summary>
24     /// Creates a new instance of the class <see cref="TelegramParser"/>.
25     /// </summary>
26     /// <param name="options">The serialisation options to be used for JSON.</param>
27     /// <param name="logger">The logger to be used.</param>
28     public TelegramParser(JsonSerializerOptions options, ILogger<TelegramParser>
29         logger)
30     {
31         _options = options;
32         _logger = logger;
33         _logger.Instantiated();
34     }
35     /// <summary>
36     /// A list of supported schemas.
37     /// </summary>
38     private readonly Dictionary<SchemaVersionInformation, Type> _supportedSchemas =
39         new()
40     {
41         new SchemaVersionInformation {
42             Type = "eew-information",
43             Version = "1.0.0" },
44             typeof(EewInformationSchema)
45         },
46         {
47             new SchemaVersionInformation {
48                 Type = "earthquake-information",
49                 Version = "1.1.0" },
50                 typeof(EarthquakeInformationSchema)
51         },
52         {
53             new SchemaVersionInformation {
54                 Type = "tsunami-information",
55                 Version = "1.0.0" },
56                 typeof(TsunamiInformationSchema)
57         },
58     };
59     /// <inheritdoc/>
60     /// <exception cref="TelegramParserFormatException">When there was an error in
61     /// parsing the telegram.</exception>
62     /// <exception cref="TelegramParserUnsupportedException">When the telegram has an
63     /// unsupported schema.</exception>
64     public Head ParseJsonTelegram(string json)
65     {
66         try
67         {
68             _logger.HandlingJson(json);
69             Head headData = JsonSerializer.Deserialize<Head>(json, _options)
70             ?? throw new TelegramParserFormatException($"Cannot deserialise:
71             {json}");
72
73             if (_supportedSchemas.TryGetValue(headData.Schema, out Type? type))
74             {
75                 _logger.SupportedSchema(headData.Schema.Type,
76                     headData.Schema.Version);
77                 Head? data = JsonSerializer.Deserialize(json, type, _options) as Head;
78                 return data ?? throw new TelegramParserFormatException($"Cannot
79                 deserialise: {json}");
80             }
81         }
82     }
```

```

74     }
75
76     _logger.UnsupportedSchema(headData.Schema.Type, headData.Schema.Version);
77     throw new TelegramParserUnsupportedException($"Schema is not supported:
78         → Type {headData.Schema.Type}, Version {headData.Schema.Version}");
79   }
80   catch (TelegramParserFormatException)
81   {
82     _logger.IncorrectJsonFormat(json);
83     throw;
84   }
85   catch (JsonException ex)
86   {
87     _logger.IncorrectJsonFormat(json);
88     throw new TelegramParserFormatException($"Cannot deserialised: {json}",
89         → ex);
90   }
91   /// <inheritdoc>
92   public Type? ParseSchemaInformation(SchemaVersionInformation schema)
93   => _supportedSchemas.TryGetValue(schema, out Type? type) ? type : null;
94 }
```

Listing A.7.63: EasonEetwViewer.Dmdata.Telemgram/Services/TelegramParser.cs

A.8 DM-D.S.S. WebSocket

```

1  using EasonEetwViewer.Dmdata.WebSocket.Services;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Exceptions;
4  /// <summary>
5  /// Represents errors that occurs in <see cref="WebSocketClient"/>.
6  /// </summary>
7  public class WebSocketClientException : Exception
8  {
9    /// <summary>
10   /// Instantiates a new instance of the <see cref="WebSocketClientException"/>
11   → class.
12   /// </summary>
13   public WebSocketClientException() { }
14   /// <summary>
15   /// Instantiates a new instance of the <see cref="WebSocketClientException"/>
16   → class with a specified error message.
17   /// </summary>
18   /// <param name="message">The error message.</param>
19   public WebSocketClientException(string message)
20     : base(message) { }
21   /// <summary>
22   /// Instantiates a new instance of the <see cref="WebSocketClientException"/>
23   → class with a specified error message and an inner exception.
24   /// </summary>
25   /// <param name="message">The error message.</param>
26   /// <param name="inner">The inner exception.</param>
27   public WebSocketClientException(string message, Exception inner)
28     : base(message, inner) { }
29 }
```

Listing A.8.1: EasonEetwViewer.Dmdata.WebSocket/Exceptions/WebSocketClientException.cs

```

1  using EasonEetwViewer.Dmdata.WebSocket.Services;
2  namespace EasonEetwViewer.Dmdata.WebSocket.Exceptions;
3  ///<summary>
4  /// Represents unsupported operations that occurs in <see cref="WebSocketClient"/>.
5  /// </summary>
6  public class WebSocketClientUnsupportedException : WebSocketClientException
7  {
8      ///<summary>
9      /// Instantiates a new instance of the <see
10     <see cref="WebSocketClientUnsupportedException"/> class.
11  }
12  ///<summary>
13  /// Instantiates a new instance of the <see
14  <see cref="WebSocketClientUnsupportedException"/> class with a specified error
15  <param name="message">The error message.</param>
16  public WebSocketClientUnsupportedException(string message)
17  : base(message) {}
18  ///<summary>
19  /// Instantiates a new instance of the <see
20  <see cref="WebSocketClientUnsupportedException"/> class with a specified error
21  <param name="message">The error message.</param>
22  <param name="inner">The inner exception.</param>
23  public WebSocketClientUnsupportedException(string message, Exception inner)
24  : base(message, inner) {}
25 }
```

Listing A.8.2:

EasonEetwViewer.Dmdata.WebSocket/Exceptions/WebSocketClientFormatException.cs

```

1  using EasonEetwViewer.Dmdata.WebSocket.Services;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Exceptions;
4  ///<summary>
5  /// Represents incorrected formatted data that occurs in <see
6  <see cref="WebSocketClient"/>.
7  /// </summary>
8  public class WebSocketClientFormatException : WebSocketClientException
{
9      ///<summary>
10     /// Instantiates a new instance of the <see
11     <see cref="WebSocketClientFormatException"/> class.
12  }
13  ///<summary>
14  /// Instantiates a new instance of the <see
15  <see cref="WebSocketClientFormatException"/> class with a specified error message.
16  /// </summary>
17  ///<param name="message">The error message.</param>
18  public WebSocketClientFormatException(string message)
19  : base(message) {}
```

```

19     ///<summary>
20     /// Instantiates a new instance of the <see
21     /// cref="WebSocketClientFormatException"/> class with a specified error message
22     /// and an inner exception.
23     ///</summary>
24     ///<param name="message">The error message.</param>
25     ///<param name="inner">The inner exception.</param>
26     public WebSocketClientFormatException(string message, Exception inner)
27         : base(message, inner) { }
28 }
```

Listing A.8.3:

EasonEetwViewer.Dmdata.WebSocket/Exceptions/WebSocketClientFormatException.cs

```

1  using EasonEetwViewer.Dmdata.Telegram.Abstractions;
2  using EasonEetwViewer.Dmdata.Telegram.Services;
3  using EasonEetwViewer.Dmdata.WebSocket.Abstractions;
4  using EasonEetwViewer.Dmdata.WebSocket.Services;
5  using Microsoft.Extensions.DependencyInjection;
6
7  namespace EasonEetwViewer.Dmdata.WebSocket.Extensions;
8  ///<summary>
9  /// Provides extension methods for <see cref="IServiceCollection"/> to add WebSocket.
10 ///</summary>
11 public static class WebSocketServiceCollectionExtensions
12 {
13     ///<summary>
14     /// Injects a <see cref="IWebSocketClient"/> with the given base URI.
15     ///</summary>
16     ///<param name="services">The instance of <see cref="IServiceCollection"/> for
17     /// the service to be injected.</param>
18     ///<returns>The <see cref="IServiceCollection"/> where the service is injected,
19     /// for chained calls.</returns>
20     public static IServiceCollection AddWebSocket(this IServiceCollection services)
21         => services
22             .AddSingleton<ITelegramParser, TelegramParser>()
23             .AddSingleton<IWebSocketClient, WebSocketClient>();
24 }
```

Listing A.8.4:

EasonEetwViewer.Dmdata.WebSocket/Extensions/WebSocketServiceCollectionExtensions.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos;
4  ///<summary>
5  /// The head information of the telegram.
6  ///</summary>
7  internal record HeadInfo
8  {
8     ///<summary>
9     /// The property <c>type</c>, the type code of the telegram.
10    ///</summary>
11    [JsonPropertyName("type")]
12    public required string Type { get; init; }
13    ///<summary>
14    /// The property <c>author</c>, the author of the telegram.
15 }
```

```

16     /// </summary>
17     [JsonPropertyName("author")]
18     public required string Author { get; init; }
19     /// <summary>
20     /// The property <c>target</c>, the target code of the telegram.
21     /// </summary>
22     /// <remarks>
23     /// <see langword="null"/> when no such target.
24     /// </remarks>
25     [JsonPropertyName("target")]
26     public string? Target { get; init; }
27     /// <summary>
28     /// The property <c>time</c>, the time base of the telegram.
29     /// </summary>
30     [JsonPropertyName("time")]
31     public required DateTimeOffset Time { get; init; }
32     /// <summary>
33     /// The property <c>designation</c>, the designation of the telegram.
34     /// </summary>
35     /// <remarks>
36     /// <see langword="null"/> when normal.
37     /// </remarks>
38     [JsonPropertyName("designation")]
39     public required string? Designation { get; init; }
40     /// <summary>
41     /// The property <c>test</c>, whether the telegram is a test telegram.
42     /// </summary>
43     [JsonPropertyName("test")]
44     public required bool IsTest { get; init; }
45     /// <summary>
46     /// The property <c>xml</c>, whether the telegram is an XML telegram.
47     /// </summary>
48     /// <remarks>
49     /// <see langword="null"/> when unknown.
50     /// </remarks>
51     [JsonPropertyName("xml")]
52     public bool? IsXml { get; init; }
53 }

```

Listing A.8.5: EasonEetwViewer.Dmdata.WebSocket/Dtos/HeadInfo.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos;
4
5  /// <summary>
6  /// Represents the type of response received by the WebSocket.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<MessageType>))]
9  internal enum MessageType
10 {
11     /// <summary>
12     /// The value <c>start</c>. Response after starting a WebSocket.
13     /// </summary>
14     [JsonStringEnumMemberName("start")]
15     Start,
16     /// <summary>
17     /// The value <c>ping</c>. A server-side initiated ping.

```

```

18     /// </summary>
19     [JsonStringEnumMemberName("ping")]
20     Ping,
21     /// <summary>
22     /// The value <c>pong</c>. Response of a client-side initiated ping.
23     /// </summary>
24     [JsonStringEnumMemberName("pong")]
25     Pong,
26     /// <summary>
27     /// The value <c>error</c>. An error response from the WebSocket.
28     /// </summary>
29     [JsonStringEnumMemberName("error")]
30     Error,
31     /// <summary>
32     /// The value <c>data</c>. A data returned by the WebSocket.
33     /// </summary>
34     [JsonStringEnumMemberName("data")]
35     Data
36 }

```

Listing A.8.6: EasonEetwViewer.Dmdata.WebSocket/Dtos/MessageType.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Dtos.Enum.WebSocket;
4  using EasonEetwViewer.Dmdata.Dtos.Telegram;
5  using EasonEetwViewer.Dmdata.WebSocket.Dtos.Data;
6
7  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
8  /// <summary>
9  /// Represents a data response.
10 /// </summary>
11 internal record DataResponse : ResponseBase
12 {
13     /// <summary>
14     /// The property <c>type</c>, a constant <see cref="MessageType.Data"/>.
15     /// </summary>
16     [JsonPropertyName("type")]
17     public override MessageType Type { get; init; } = MessageType.Data;
18     /// <summary>
19     /// The property <c>version</c>, representing the version of the data.
20     /// </summary>
21     [JsonPropertyName("version")]
22     public required string Version { get; init; }
23     /// <summary>
24     /// The property <c>classification</c>, representing the classification of the
25     /// data.
26     /// </summary>
27     [JsonPropertyName("classification")]
28     public required Classification Classification { get; init; }
29     /// <summary>
30     /// The property <c>id</c>, representing the ID of the data.
31     /// </summary>
32     [JsonPropertyName("id")]
33     public required string Id { get; init; }
34     /// <summary>
35     /// The property <c>passing</c>, representing the passing route of the data.
36     /// </summary>

```

```

36     [JsonPropertyName("passing")]
37     public required IEnumerable<PassingDetail> PassingRoute { get; init; }
38     /// <summary>
39     /// The property <c>head</c>, representing the head information of the telegram.
40     /// </summary>
41     [JsonPropertyName("head")]
42     public required HeadInfo DataInfo { get; init; }
43     /// <summary>
44     /// The property <c>xmlReport</c>, representing the head and control information
45     /// of the data.
46     /// </summary>
47     /// <remarks>
48     /// <see langword="null"/> when not an XML report or a JSON converted format.
49     /// </remarks>
50     [JsonPropertyName("xmlReport")]
51     public XmlReport? XmlInfo { get; init; }
52     /// <summary>
53     /// The property <c>format</c>, representing the format of the data.
54     /// </summary>
55     /// <remarks>
56     /// <see langword="null"/> when unknown.
57     /// </remarks>
58     [JsonPropertyName("format")]
59     public required FormatType? Format { get; init; }
60     /// <summary>
61     /// The property <c>compression</c>, representing the compression format of the
62     /// data.
63     /// </summary>
64     /// <remarks>
65     /// <see langword="null"/> when data is uncompressed.
66     /// </remarks>
67     [JsonPropertyName("compression")]
68     public required CompressionType? Compression { get; init; }
69     /// <summary>
70     /// The property <c>encoding</c>, representing the encoding format of the data.
71     /// </summary>
72     /// <remarks>
73     /// <see langword="null"/> when not encoded.
74     /// </remarks>
75     [JsonPropertyName("encoding")]
76     public required EncodingType? Encoding { get; init; }
77     /// <summary>
78     /// The property <c>body</c>, representing the body of the data.
79     /// </summary>
80     [JsonPropertyName("body")]
81     public required string Body { get; init; }
82 }
```

Listing A.8.7: EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/DataResponse.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
4
5  /// <summary>
6  /// Represents a response received from the WebSocket.
7  /// </summary>
8  internal record ResponseBase
```

```

9  {
10   /// <summary>
11   /// The property <c>type</c>, representing the type of the response the WebSocket
12   /// receives.
13   /// </summary>
14   [JsonPropertyName("type")]
15   public virtual MessageType Type { get; init; }
}

```

Listing A.8.8: EasonEetwViewer.Dmdata.WebSocket/Dtos/ResponseBase.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
4
5  /// <summary>
6  /// Represents a ping response from the WebSocket.
7  /// </summary>
8  internal record PingResponse : ResponseBase
{
9
10   /// <summary>
11   /// The property <c>type</c>, a constant <see cref="MessageType.Ping"/>.
12   /// </summary>
13   [JsonPropertyName("type")]
14   public override MessageType Type { get; init; } = MessageType.Ping;
15
16   /// <summary>
17   /// The property <c>pingId</c>, the Ping ID that should be included when returning
18   /// a corresponding Pong.
19   /// </summary>
20   [JsonPropertyName("pingId")]
21   public required string PingId { get; init; }
}

```

Listing A.8.9: EasonEetwViewer.Dmdata.WebSocket/Dtos/PingResponse.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
4
5  /// <summary>
6  /// Represents a pong response from the WebSocket.
7  /// </summary>
8  internal record PongResponse : ResponseBase
{
9
10   /// <summary>
11   /// The property <c>type</c>, a constant <see cref="MessageType.Pong"/>.
12   /// </summary>
13   [JsonPropertyName("type")]
14   public override MessageType Type { get; init; } = MessageType.Pong;
15
16   /// <summary>
17   /// The property <c>pingerId</c>, the Ping ID that was included when the user
18   /// initiated the Pong.
19   /// </summary>
20   /// <remarks>
21   /// <see langword="null"/> when not specified by the user-initiated ping.
22   /// </remarks>
}

```

```

21     [JsonPropertyName("pingId")]
22     public string? PingId { get; init; }
23 }
```

Listing A.8.10: EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/PongResponse.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
4
5  /// <summary>
6  /// Represents an error response from the WebSocket.
7  /// </summary>
8  internal record ErrorResponse : ResponseBase
{
    /// <summary>
    /// The property <c>type</c>, a constant <see cref="MessageType.Error"/>.
    /// </summary>
    [JsonPropertyName("type")]
    public override MessageType Type { get; init; } = MessageType.Error;
    /// <summary>
    /// The property <c>error</c>, the error message.
    /// </summary>
    [JsonPropertyName("error")]
    public required string Error { get; init; }
    /// <summary>
    /// The property <c>code</c>, the error code.
    /// </summary>
    [JsonPropertyName("code")]
    public required int Code { get; init; }
    /// <summary>
    /// The property <c>close</c>, whether the error was fatal and the WebSocket
    /// connection is closed.
    /// </summary>
    [JsonPropertyName("close")]
    public required bool Close { get; init; }
}
```

Listing A.8.11: EasonEetwViewer.Dmdata.WebSocket/Dtos/Response/ErrorResponse.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Dtos.Enum.WebSocket;
4
5  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
6  /// <summary>
7  /// Represents a start response.
8  /// </summary>
9  internal record StartResponse : ResponseBase
{
    /// <summary>
    /// The property <c>type</c>, a constant <see cref="MessageType.Start"/>.
    /// </summary>
    [JsonPropertyName("type")]
    public override MessageType Type { get; init; } = MessageType.Start;
    /// <summary>
    /// The property <c>time</c>, representing the time the WebSocket was started.
    /// </summary>
```

```

18     /// </summary>
19     [JsonPropertyName("time")]
20     public required DateTimeOffset Time { get; init; }
21     /// <summary>
22     /// The property <c>socketId</c>, representing the ID of the socket.
23     /// </summary>
24     [JsonPropertyName("socketId")]
25     public required int SocketId { get; init; }
26     /// <summary>
27     /// The property <c>classifications</c>. The classifications of telegrams that the
28     → WebSocket receives.
29     /// </summary>
30     [JsonPropertyName("classifications")]
31     public required IEnumerable<Classification> Classifications { get; init; }
32     /// <summary>
33     /// The property <c>types</c>. The types of telegrams the program receives.
34     /// </summary>
35     /// <remarks>
36     /// <see langword="null"/> when receiving all types from the classifications.
37     /// </remarks>
38     [JsonPropertyName("types")]
39     public required IEnumerable<string?> Types { get; init; }
40     /// <summary>
41     /// The property <c>test</c>. Whether the WebSocket receives test telegrams.
42     /// </summary>
43     [JsonPropertyName("test")]
44     public required TestStatus TestStatus { get; init; }
45     /// <summary>
46     /// The property <c>appName</c>. The application name of the WebSocket connection.
47     /// </summary>
48     /// <remarks>
49     /// <see langword="null"/> when not indicated.
50     /// </remarks>
51     [JsonPropertyName("appName")]
52     public required string? ApplicationName { get; init; }
53     /// <summary>
54     /// The property <c>formats</c>. A list of formats of telegrams the WebSocket
55     → receives.
56     /// </summary>
57     [JsonPropertyName("formats")]
58     public required IEnumerable<FormatType> Formats { get; init; }
59 }
```

Listing A.8.12: EasonEetwViewer.Dmdata.WebSocket/Dtos/StartResponse.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
3
4  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Request;
5
6  /// <summary>
7  /// Represents a pong request from the client.
8  /// </summary>
9  internal record PongRequest : ResponseBase
10 {
11     /// <summary>
12     /// The property <c>type</c>, a constant <see cref="MessageType.Pong"/>.
13     /// </summary>
```

```

14     [JsonPropertyName("type")]
15     public override MessageType Type { get; init; } = MessageType.Pong;
16     /// <summary>
17     /// The property <c>pingerId</c>, the Ping ID that was indicated by the server to be
18     → included.
19     /// </summary>
20     [JsonPropertyName("pingId")]
21     public required string PingId { get; init; }
22 }
```

Listing A.8.13: EasonEetwViewer.Dmdata.WebSocket.Dtos.Request/PongRequest.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
3
4  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Request;
5
6  /// <summary>
7  /// Represents a ping request from the client.
8  /// </summary>
9  internal record PingRequest : ResponseBase
{
10
11    /// <summary>
12    /// The property <c>type</c>, a constant <see cref="MessageType.Ping"/>.
13    /// </summary>
14    [JsonPropertyName("type")]
15    public override MessageType Type { get; init; } = MessageType.Ping;
16    /// <summary>
17    /// The property <c>pingerId</c>, the Ping ID that is to be included when the
18    → WebSocket returns the Pong.
19    /// </summary>
20    /// <remarks>
21    /// <see langword="null"/> when not specified by the user-initiated ping.
22    /// </remarks>
23    [JsonPropertyName("pingId")]
24    public string? PingId { get; init; }
}
```

Listing A.8.14: EasonEetwViewer.Dmdata.WebSocket.Dtos.Request/PingRequest.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Data;
4
5  /// <summary>
6  /// Represents the type of compression used in the WebSocket.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<CompressionType>))]
9  internal enum CompressionType
{
10
11    /// <summary>
12    /// The value <c>gzip</c>, representing Gzip compression.
13    /// </summary>
14    [JsonStringEnumMemberName("gzip")]
15    Gzip,
16    /// <summary>
17    /// The value <c>zip</c>, representing Zip compression.
18 }
```

```

18     /// </summary>
19     [JsonStringEnumMemberName("zip")]
20     Zip
21 }

```

Listing A.8.15: EasonEetwViewer.Dmdata.WebSocket.Dtos/Data/CompressionType.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Data;
4
5  /// <summary>
6  /// Represents the type of encoding used in the response.
7  /// </summary>
8  [JsonConverter(typeof(JsonStringEnumConverter<EncodingType>))]
9  internal enum EncodingType
10 {
11     /// <summary>
12     /// The value <c>base64</c>, representing Base64 encoding.
13     /// </summary>
14     [JsonStringEnumMemberName("base64")]
15     Base64,
16     /// <summary>
17     /// The value <c>utf8</c>, representing UTF-8 encoding.
18     /// </summary>
19     [JsonStringEnumMemberName("utf8")]
20     Utf8
21 }

```

Listing A.8.16: EasonEetwViewer.Dmdata.WebSocket.Dtos/Data/EncodingType.cs

```

1  using System.Text.Json.Serialization;
2
3  namespace EasonEetwViewer.Dmdata.WebSocket.Dtos.Data;
4  /// <summary>
5  /// Represents a passing detail of the specified data response.
6  /// </summary>
7  internal record PassingDetail
8  {
9      /// <summary>
10     /// The property <c>name</c>, the name of the passing.
11     /// </summary>
12     [JsonPropertyName("name")]
13     public required string Name { get; init; }
14     /// <summary>
15     /// The property <c>time</c>, the time of the passing.
16     /// </summary>
17     [JsonPropertyName("time")]
18     public required DateTimeOffset Time { get; init; }
19 }

```

Listing A.8.17: EasonEetwViewer.Dmdata.WebSocket.Dtos/Data/PassingDetail.cs

```

1  using EasonEetwViewer.Dmdata.WebSocket.Events;
2

```

```

3  namespace EasonEetwViewer.Dmdata.WebSocket.Abstractions;
4
5  /// <summary>
6  /// Represents a WebSocket connection to <see href="dmdata.jp"/>.
7  /// </summary>
8  public interface IWebSocketClient
{
9
10    /// <summary>
11    /// When data is received by the WebSocket connection.
12    /// </summary>
13    event EventHandler<DataReceivedEventArgs>? DataReceived;
14
15    /// <summary>
16    /// When the status of the WebSocket connection changes.
17    /// </summary>
18    event EventHandler<StatusChangedEventArgs>? StatusChanged;
19
20    /// <summary>
21    /// Indicates if the WebSocket is connected.
22    /// </summary>
23    bool IsWebSocketConnected { get; }
24
25    /// <summary>
26    /// Connect to the specified URL for the WebSocket
27    /// </summary>
28    /// <param name="webSocketUrl">The URL of the WebSocket to connect to.</param>
29    /// <returns>A <see cref="Task"/> object that represents the asynchronous
30    /// operation.</returns>
31    Task ConnectAsync(Uri webSocketUrl);
32
33    /// <summary>
34    /// Disconnect from the current WebSocket connection.
35    /// </summary>
36    /// <returns>A <see cref="Task"/> object that represents the asynchronous
37    /// operation.</returns>
38    Task DisconnectAsync();
}

```

Listing A.8.18: EasonEetwViewer.Dmdata.WebSocket/Abstractions/IWebSocketClient.cs

```

1  namespace EasonEetwViewer.Dmdata.WebSocket.Events;
2
3  /// <summary>
4  /// The event arguments for the status changed event.
5  /// </summary>
6  public class StatusChangedEventArgs : EventArgs
{
7
8    /// <summary>
9    /// Whether the WebSocket is connected or not.
10   /// </summary>
11   public required bool IsConnected { get; init; }
12 }

```

Listing A.8.19: EasonEetwViewer.Dmdata.WebSocket/Events/StatusChangedEventArgs.cs

```

1  using EasonEetwViewer.Dmdata.Telegram.Dtos.TelegramBase;
2  namespace EasonEetwViewer.Dmdata.WebSocket.Events;
3  /// <summary>
4  /// Represents the event arguments for a data being received.
5  /// </summary>
6  public sealed class DataReceivedEventArgs : EventArgs

```

```

7  {
8      /// <summary>
9      /// The telegram that is received.
10     /// </summary>
11     public required Head Telegram { get; init; }
12 }

```

Listing A.8.20: EasonEetwViewer.Dmdata.WebSocket/Events/DataReceivedEventArgs.cs

```

1  using EasonEetwViewer.Dmdata.WebSocket.Services;
2  using Microsoft.Extensions.Logging;
3
4  namespace EasonEetwViewer.WebSocket.Services;
5  /// <summary>
6  /// Represents the log messages used in <see cref="WebSocketClient"/>.
7  /// </summary>
8  internal static partial class WebSocketClientLogs
{
9
10     /// <summary>
11     /// Log when instantiated.
12     /// </summary>
13     /// <param name="logger">The logger to be used.</param>
14     [LoggerMessage(
15         EventId = 0,
16         EventName = nameof(Instantiated),
17         Level = LogLevel.Information,
18         Message = "Instantiated.")]
19     public static partial void Instantiated(
20         this ILogger<WebSocketClient> logger);
21
22     /// <summary>
23     /// Log when connecting to WebSocket.
24     /// </summary>
25     /// <param name="logger">The logger to be used.</param>
26     /// <param name="url">The URI of the WebSocket.</param>
27     [LoggerMessage(
28         EventId = 1,
29         EventName = nameof(Connecting),
30         Level = LogLevel.Debug,
31         Message = "Connecting to WebSocket: `'{Url}`.")]
32     public static partial void Connecting(
33         this ILogger<WebSocketClient> logger, Uri url);
34
35     /// <summary>
36     /// Log when connected to WebSocket.
37     /// </summary>
38     /// <param name="logger">The logger to be used.</param>
39     /// <param name="url">The URI of the WebSocket.</param>
40     [LoggerMessage(
41         EventId = 2,
42         EventName = nameof(Connected),
43         Level = LogLevel.Information,
44         Message = "Connected to WebSocket: `'{Url}`.")]
45     public static partial void Connected(
46         this ILogger<WebSocketClient> logger, Uri url);
47
48     /// <summary>
49     /// Log when disconnecting from WebSocket.

```

```
50     /// </summary>
51     /// <param name="logger">The logger to be used.</param>
52     [LoggerMessage(
53         EventId = 3,
54         EventName = nameof(Disconnecting),
55         Level = LogLevel.Debug,
56         Message = "Disconnecting from WebSocket.")]
57     public static partial void Disconnecting(
58         this ILogger<WebSocketClient> logger);
59
60     /// <summary>
61     /// Log when disconnected from WebSocket.
62     /// </summary>
63     /// <param name="logger">The logger to be used.</param>
64     [LoggerMessage(
65         EventId = 4,
66         EventName = nameof(Disconnected),
67         Level = LogLevel.Information,
68         Message = "Disconnected from WebSocket.")]
69     public static partial void Disconnected(
70         this ILogger<WebSocketClient> logger);
71
72     /// <summary>
73     /// Log when sending data to WebSocket.
74     /// </summary>
75     /// <param name="logger">The logger to be used.</param>
76     /// <param name="data">The data to be sent.</param>
77     [LoggerMessage(
78         EventId = 5,
79         EventName = nameof(Sending),
80         Level = LogLevel.Debug,
81         Message = "Sending data to WebSocket: `'{Data}`.")]
82     public static partial void Sending(
83         this ILogger<WebSocketClient> logger, string data);
84
85     /// <summary>
86     /// Log when data is sent to WebSocket.
87     /// </summary>
88     /// <param name="logger">The logger to be used.</param>
89     /// <param name="data">The data to be sent.</param>
90     [LoggerMessage(
91         EventId = 6,
92         EventName = nameof(Sent),
93         Level = LogLevel.Information,
94         Message = "Data sent to WebSocket: `'{Data}`.")]
95     public static partial void Sent(
96         this ILogger<WebSocketClient> logger, string data);
97
98     /// <summary>
99     /// Log when task cancelled.
100    /// </summary>
101    /// <param name="logger">The logger to be used.</param>
102    [LoggerMessage(
103        EventId = 7,
104        EventName = nameof(TaskCancelled),
105        Level = LogLevel.Debug,
106        Message = "Task is cancelled.")]
107    public static partial void TaskCancelled(
108        this ILogger<WebSocketClient> logger);
```

```
109
110     /// <summary>
111     /// Log when unexpected exception caught.
112     /// </summary>
113     /// <param name="logger">The logger to be used.</param>
114     /// <param name="exception">The description of the exception.</param>
115     [LoggerMessage(
116         EventId = 8,
117         EventName = nameof(UnexpectedException),
118         Level = LogLevel.Error,
119         Message = "Unexpected exception: `'{Exception}`.")]
120     public static partial void UnexpectedException(
121         this ILogger<WebSocketClient> logger, string exception);
122
123     /// <summary>
124     /// Log when data with incorrect format received.
125     /// </summary>
126     /// <param name="logger">The logger to be used.</param>
127     /// <param name="data">The incorrect data received.</param>
128     [LoggerMessage(
129         EventId = 10,
130         EventName = nameof(IncorrectFormat),
131         Level = LogLevel.Error,
132         Message = "Incorected format: `'{Data}`.")]
133     public static partial void IncorrectFormat(
134         this ILogger<WebSocketClient> logger, string data);
135
136     /// <summary>
137     /// Log when sending a ping request.
138     /// </summary>
139     /// <param name="logger">The logger to be used.</param>
140     /// <param name="pingId">The ID of the ping request.</param>
141     [LoggerMessage(
142         EventId = 11,
143         EventName = nameof(SendingPingRequest),
144         Level = LogLevel.Trace,
145         Message = "Sending ping request with ID: `'{PingId}`.")]
146     public static partial void SendingPingRequest(
147         this ILogger<WebSocketClient> logger, string pingId);
148
149     /// <summary>
150     /// Log when number of unreceived ping request exceeded a maximum.
151     /// </summary>
152     /// <param name="logger">The logger to be used.</param>
153     [LoggerMessage(
154         EventId = 13,
155         EventName = nameof(UnreceivedPingExceeding),
156         Level = LogLevel.Warning,
157         Message = "Number of unreceived ping requests exceeded maximum.")]
158     public static partial void UnreceivedPingExceeding(
159         this ILogger<WebSocketClient> logger);
160
161     /// <summary>
162     /// Log when the data received is in a format that is unsupported.
163     /// </summary>
164     /// <param name="logger">The logger to be used.</param>
165     [LoggerMessage(
166         EventId = 14,
167         EventName = nameof(UnsupportedFormat),
```

```
168     Level = LogLevel.Warning,
169     Message = "The format received is not supported."]
170 public static partial void UnsupportedFormat(
171     this ILogger<WebSocketClient> logger);
172
173     /// <summary>
174     /// Log when data is received.
175     /// </summary>
176     /// <param name="logger">The logger to be used.</param>
177     /// <param name="data">The data received.</param>
178     [LoggerMessage(
179         EventId = 15,
180         EventName = nameof(DataReceived),
181         Level = LogLevel.Information,
182         Message = "Data received: `{{Data}}`.")]
183     public static partial void DataReceived(
184         this ILogger<WebSocketClient> logger, string data);
185
186     /// <summary>
187     /// Log when responding to a ping request.
188     /// </summary>
189     /// <param name="logger">The logger to be used.</param>
190     /// <param name="pongId">The pong ID specified.</param>
191     [LoggerMessage(
192         EventId = 16,
193         EventName = nameof(ReturningPong),
194         Level = LogLevel.Trace,
195         Message = "Pong response: `{{PongId}}`.")]
196     public static partial void ReturningPong(
197         this ILogger<WebSocketClient> logger, string pongId);
198
199     /// <summary>
200     /// Log when a ping request is received.
201     /// </summary>
202     /// <param name="logger">The logger to be used.</param>
203     /// <param name="pingId">The ping ID received.</param>
204     [LoggerMessage(
205         EventId = 17,
206         EventName = nameof(RemovingPing),
207         Level = LogLevel.Trace,
208         Message = "Ping response: `{{PingId}}`.")]
209     public static partial void RemovingPing(
210         this ILogger<WebSocketClient> logger, string pingId);
211
212     /// <summary>
213     /// Log when an error is received
214     /// </summary>
215     /// <param name="logger">The logger to be used.</param>
216     /// <param name="error">The error message.</param>
217     [LoggerMessage(
218         EventId = 18,
219         EventName = nameof(ErrorReceived),
220         Level = LogLevel.Warning,
221         Message = "Error received: `{{Error}}`.")]
222     public static partial void ErrorReceived(
223         this ILogger<WebSocketClient> logger, string error);
224
225     /// <summary>
226     /// Log when an error is received and the connection is closed.
```

```

227     /// </summary>
228     /// <param name="logger">The logger to be used.</param>
229     /// <param name="error">The error message.</param>
230     [LoggerMessage(
231         EventId = 19,
232         EventName = nameof(ErrorReceivedAndClosed),
233         Level = LogLevel.Error,
234         Message = "Error received and disconnected: `'{Error}`.")]
235     public static partial void ErrorReceivedAndClosed(
236         this ILogger<WebSocketClient> logger, string error);
237
238     /// <summary>
239     /// Log when handling data response.
240     /// </summary>
241     /// <param name="logger">The logger to be used.</param>
242     [LoggerMessage(
243         EventId = 20,
244         EventName = nameof(HandlingDataResponse),
245         Level = LogLevel.Information,
246         Message = "Handling data response.")]
247     public static partial void HandlingDataResponse(
248         this ILogger<WebSocketClient> logger);
249
250     /// <summary>
251     /// Log when string is decoded and decompressed.
252     /// </summary>
253     /// <param name="logger">The logger to be used.</param>
254     /// <param name="decodedString">The string decoded.</param>
255     [LoggerMessage(
256         EventId = 21,
257         EventName = nameof(DecodedAndDecompressedString),
258         Level = LogLevel.Information,
259         Message = "Decoded string: `'{DecodedString}`.")]
260     public static partial void DecodedAndDecompressedString(
261         this ILogger<WebSocketClient> logger, string decodedString);
262 }

```

Listing A.8.21: EasonEetwViewer.Dmdata.WebSocket/Logging/WebSocketClientLogs.cs

```

1  using System.Diagnostics;
2  using System.IO.Compression;
3  using System.Net.WebSockets;
4  using System.Security.Cryptography;
5  using System.Text;
6  using System.Text.Json;
7  using EasonEetwViewer.Dmdata.Dtos.Enum.WebSocket;
8  using EasonEetwViewer.Dmdata.Telemgram.Abstractions;
9  using EasonEetwViewer.Dmdata.Telemgram.Dtos.TelemgramBase;
10 using EasonEetwViewer.Dmdata.Telemgram.Exceptions;
11 using EasonEetwViewer.Dmdata.WebSocket.Abstractions;
12 using EasonEetwViewer.Dmdata.WebSocket.Dtos;
13 using EasonEetwViewer.Dmdata.WebSocket.Dtos.Data;
14 using EasonEetwViewer.Dmdata.WebSocket.Dtos.Request;
15 using EasonEetwViewer.Dmdata.WebSocket.Dtos.Response;
16 using EasonEetwViewer.Dmdata.WebSocket.Events;
17 using EasonEetwViewer.Dmdata.WebSocket.Exceptions;
18 using EasonEetwViewer.WebSocket.Services;
19 using Microsoft.Extensions.Logging;

```

```
20
21 namespace EasonEetwViewer.Dmdata.WebSocket.Services;
22
23 /// <summary>
24 /// The default implementation of <see cref="IWebSocketClient"/>.
25 /// </summary>
26 internal sealed class WebSocketClient : IWebSocketClient
27 {
28     /// <summary>
29     /// The JSON Serializer Options to be used.
30     /// </summary>
31     private readonly JsonSerializerOptions _options;
32
33     /// <summary>
34     /// The parser used to parse telegrams.
35     /// </summary>
36     private readonly ITelegramParser _parser;
37
38     /// <summary>
39     /// The WebSocket client used for the connections.
40     /// </summary>
41     private ClientWebSocket _client;
42
43     /// <summary>
44     /// The cancellation token source.
45     /// </summary>
46     private CancellationTokenSource _tokenSource;
47
48     /// <summary>
49     /// The queue of ping codes that have not yet received a response.
50     /// </summary>
51     private readonly Queue<string> _pingStrings;
52
53     /// <summary>
54     /// The time span between two user-initiated ping requests.
55     /// </summary>
56     private readonly TimeSpan _pingInterval = TimeSpan.FromSeconds(30);
57
58     /// <summary>
59     /// The time span between the disposal and the cancellation of the cancellation
60     /// → token source.
61     /// </summary>
62     private readonly TimeSpan _cancelWaitInterval = TimeSpan.FromSeconds(5);
63
64     /// <summary>
65     /// The time span between two checks of the number of unresponded ping requests.
66     /// </summary>
67     private readonly TimeSpan _checkPingInterval = TimeSpan.FromSeconds(15);
68
69     /// <summary>
70     /// The logger to be used.
71     /// </summary>
72     private readonly ILogger<WebSocketClient> _logger;
73
74     /// <summary>
75     /// The receive buffer size.
76     /// </summary>
```

```
77     /// The allowed characters when generating a Ping Id.
78     /// </summary>
79     private const string _allowedChars =
80         "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
81     /// <summary>
82     /// The length of a Ping Id.
83     /// </summary>
84     private const int _pingIdLength = 4;
85
86     /// <inheritdoc/>
87     public event EventHandler<Events.DataReceivedEventArgs>? DataReceived;
88     /// <inheritdoc/>
89     public event EventHandler<StatusChangedEventArgs>? StatusChanged;
90     /// <inheritdoc/>
91     public bool IsWebSocketConnected
92         => _client.State is WebSocketState.Open;
93
94     /// <summary>
95     /// A task that sends ping requests to the WebSocket server with interval <see
96     /// cref="_pingInterval"/>, until the token has been cancelled.
97     /// </summary>
98     /// <returns>A <see cref="Task"/> that represents the asynchronous
99     /// operation.</returns>
100    private async Task SendPingLoop()
101    {
102        try
103        {
104            while (!_token.IsCancellationRequested)
105            {
106                PingRequest pingRequest = new()
107                {
108                    PingId = RandomNumberGenerator.GetString(_allowedChars,
109                        _pingIdLength)
110                };
111                _pingStrings.Enqueue(pingRequest.PingId);
112                _logger.SendingPingRequest(pingRequest.PingId);
113                await SendAsync(pingRequest);
114                await Task.Delay(_pingInterval);
115            }
116        }
117        catch (Exception ex)
118        {
119            _logger.UnexpectedException(ex.ToString());
120            await DisconnectAsync();
121        }
122    }
123
124    /// <summary>
125    /// A task that checks number of unresponded ping requests with interval <see
126    /// cref="_checkPingInterval"/>, until the token has been cancelled.
127    /// </summary>
128    /// <returns>A <see cref="Task"/> that represents the asynchronous
129    /// operation.</returns>
130    private async Task CheckPingLoop()
131    {
132        try
133        {
134            while (!_token.IsCancellationRequested)
135            {
```

```
130             if (_pingStrings.Count > _maxPingUnresponded)
131             {
132                 _logger.UnreceivedPingExceeding();
133                 await DisconnectAsync();
134             }
135
136             await Task.Delay(_checkPingInterval);
137         }
138     }
139     catch (Exception ex)
140     {
141         _logger.UnexpectedException(ex.ToString());
142         await DisconnectAsync();
143     }
144 }
145
146 /// <summary>
147 /// A task that receives message from the WebSocket connection and deals with them
148 /// as appropriate.
149 /// </summary>
150 /// <returns>A <see cref="Task"/> that represents the asynchronous
151 /// operation.</returns>
152 private async Task ReceiveLoop()
153 {
154     try
155     {
156         byte[] receiveBuffer = new byte[_receiveBufferSize];
157
158         // Modified from https://stackoverflow.com/a/65761228,
159         // https://stackoverflow.com/a/63574016
160         while (_client.State is WebSocketState.Open &&
161             !_token.IsCancellationRequested)
162         {
163             WebSocketReceiveResult result = await _client.ReceiveAsync(
164                 new ArraySegment<byte>(receiveBuffer),
165                 _token);
166
167             if (!result.EndOfMessage)
168             {
169                 _logger.UnsupportedFormat();
170             }
171             else
172             {
173                 string responseBody = Encoding.UTF8.GetString(receiveBuffer, 0,
174                     result.Count);
175                 _logger.DataReceived(responseBody);
176                 await HandleResponse(responseBody);
177             }
178         }
179     }
180 }
181
182 /// <summary>
183 /// Handles the response received from the WebSocket.
```

```
184     /// </summary>
185     /// <param name="responseBody">The response body received from the
186     /// <returns>A <see cref="Task"/> that represents the asynchronous
187     /// operation.</returns>
188     /// <exception cref="UnreachableException">When reaching a code that is
189     /// unreachable.</exception>
190     private async Task HandleResponse(string responseBody)
191     {
192         try
193         {
194             ResponseBase webSocketResponse =
195                 JsonSerializer.Deserialize<ResponseBase>(responseBody, _options)
196                 ?? throw new WebSocketClientFormatException();
197             switch (webSocketResponse.Type)
198             {
199                 case MessageType.Start:
200                     StartResponse startResponse =
201                         JsonSerializer.Deserialize<StartResponse>(responseBody,
202                             _options)
203                             ?? throw new WebSocketClientFormatException();
204                     break;
205
206                 case MessageType.Ping:
207                     PingResponse pingResponse =
208                         JsonSerializer.Deserialize<PingResponse>(responseBody,
209                             _options)
210                             ?? throw new WebSocketClientFormatException();
211
212                     PongRequest pongRequest = new()
213                     {
214                         PingId = pingResponse.PingId
215                     };
216                     _logger.ReturningPong(pingResponse.PingId);
217                     await SendAsync(pongRequest);
218
219                     break;
220
221                 case MessageType.Pong:
222                     PongResponse pongResponse =
223                         JsonSerializer.Deserialize<PongResponse>(responseBody,
224                             _options)
225                             ?? throw new WebSocketClientFormatException();
226
227                     if (pongResponse.PingId is not null)
228                     {
229                         _logger.RemovingPing(pongResponse.PingId);
230                         RemovePings(pongResponse.PingId);
231                     }
232
233                     break;
234
235                 case MessageType.Error:
236                     ErrorResponse errorResponse =
237                         JsonSerializer.Deserialize<ErrorResponse>(responseBody,
238                             _options)
239                             ?? throw new WebSocketClientFormatException();
240
241                     if (errorResponse.Close)
```

```
231         {
232             _logger.ErrorReceivedAndClosed(errorResponse.Error);
233             await DisconnectAsync();
234         }
235     else
236     {
237         _logger.ErrorReceived(errorResponse.Error);
238     }
239
240     break;
241
242     case MessageType.Data:
243         DataResponse dataResponse =
244             → JsonSerializer.Deserialize<DataResponse>(responseBody,
245             → _options)
246             ?? throw new WebSocketClientFormatException();
247
248         _logger.HandlingDataResponse();
249         HandleDataResponse(dataResponse);
250
251         break;
252
253     default:
254         throw new UnreachableException();
255     }
256 }
257 catch (JsonException)
258 {
259     _logger.IncorrectFormat(responseBody);
260 }
261 catch (WebSocketClientFormatException)
262 {
263     _logger.IncorrectFormat(responseBody);
264 }
265 catch (WebSocketClientUnsupportedException)
266 {
267     _logger.UnsupportedFormat();
268 }
269
270 /// <summary>
271 /// Removes the ping IDs before the specified ping ID.
272 /// </summary>
273 /// <param name="pingId">The ping ID to be removed until.</param>
274 private void RemovePings(string pingId)
275 {
276     if (_pingStrings.Contains(pingId))
277     {
278         string lastDequeue;
279         do
280         {
281             lastDequeue = _pingStrings.Dequeue();
282         } while (lastDequeue != pingId);
283     }
284
285     /// <summary>
286     /// Handles the data response received from the WebSocket.
287     /// </summary>
```

```
288     /// <param name="dataResponse">The data response to be handled.</param>
289     /// <exception cref="UnreachableException">When reaching a code that is
290     → unreachable.</exception>
291     /// <exception cref="WebSocketClientUnsupportedException">When the received data
292     → type is not supported.</exception>
293     private void HandleDataResponse(DataResponse dataResponse)
294     {
295         string bodyString = dataResponse.Body;
296         using MemoryStream decodedStringStream = dataResponse.Encoding switch
297         {
298             EncodingType.Base64
299                 => ConvertBase64Response(bodyString),
300             EncodingType.Utf8 or null
301                 => ConvertUtf8Response(bodyString),
302             _              => throw new UnreachableException(),
303         };
304
305         string finalString;
306         switch (dataResponse.Compression)
307         {
308             case CompressionType.Gzip:
309                 finalString = UncompressGzip(decodedStringStream);
310                 break;
311             case CompressionType.Zip:
312                 _logger.UnsupportedFormat();
313                 throw new WebSocketClientUnsupportedException();
314             case null:
315                 finalString = bodyString;
316                 break;
317             default:
318                 throw new UnreachableException();
319         }
320
321         _logger.DecodedAndDecompressedString(finalString);
322
323         switch (dataResponse.Format)
324         {
325             case FormatType.Json:
326                 HandleJsonData(finalString);
327                 break;
328             case FormatType.Xml:
329             case FormatType.AlphaNumeric:
330             case FormatType.Binary:
331             case null:
332                 _logger.UnsupportedFormat();
333                 throw new WebSocketClientUnsupportedException();
334             default:
335                 throw new UnreachableException();
336         }
337     }
338
339     /// <summary>
340     /// Handles the JSON data received from the WebSocket.
341     /// </summary>
342     /// <param name="finalString">The string of the JSON to be handled.</param>
343     private void HandleJsonData(string finalString)
344     {
345         try
```

```
345     {
346         Head data = _parser.ParseJsonTelegram(finalString);
347         DataReceived?.Invoke(this, new Events.DataReceivedEventArgs() { Telegram
348             => data });
349     }
350     catch (TelegramParserFormatException)
351     {
352         _logger.IncorrectFormat(finalString);
353     }
354     catch (TelegramParserUnsupportedException)
355     {
356         _logger.UnsupportedFormat();
357     }
358 }
359 /// <summary>
360 /// Converts the given string to a <see cref="MemoryStream"/> using UTF8.
361 /// </summary>
362 /// <param name="bodyString">The string to be converted.</param>
363 /// <returns>The converted <see cref="MemoryStream"/>.</returns>
364 private static MemoryStream ConvertUtf8Response(string bodyString)
365     => new(Encoding.UTF8.GetBytes(bodyString));
366
367 /// <summary>
368 /// Converts the given string to a <see cref="MemoryStream"/> using Base64.
369 /// </summary>
370 /// <param name="bodyString">The string to be converted.</param>
371 /// <returns>The converted <see cref="MemoryStream"/>.</returns>
372 private static MemoryStream ConvertBase64Response(string bodyString)
373     => new(Convert.FromBase64String(bodyString));
374
375 /// <summary>
376 /// Decompresses the given <see cref="MemoryStream"/> using GZip.
377 /// </summary>
378 /// <param name="stream">The given <see cref="MemoryStream"/>.</param>
379 /// <returns>A string representing the decoded data.</returns>
380 private static string UncompressGzip(MemoryStream stream)
381 {
382     using GZipStream gZipStream = new(stream, CompressionMode.Decompress);
383     using MemoryStream outputStream = new();
384     gZipStream.CopyTo(outputStream);
385     return Encoding.UTF8.GetString(outputStream.ToArray());
386 }
387
388 /// <summary>
389 /// Creates a new instance of the class with the given parameters.
390 /// </summary>
391 /// <param name="parser">The parser to be used to parse JSON telegrams.</param>>
392 /// <param name="logger">The logger to be used.</param>
393 /// <param name="jsonSerializerOptions">The JSON Serialisation options to be
394     => used.</param>
395 public WebSocketClient(ITelegramParser parser, ILogger<WebSocketClient> logger,
396     => JsonSerializerOptions jsonSerializerOptions)
397 {
398     _parser = parser;
399     _client = new();
400     _pingStrings = [];
401     _tokenSource = new();
402     _token = _tokenSource.Token;
```

```
401     _options = jsonSerializerOptions;
402     _logger = logger;
403     _logger.Instantiated();
404 }
405 /// <inheritdoc/>
406 public async Task ConnectAsync(Uri webSocketUrl)
407 {
408     if (_client.State == WebSocketState.Open)
409     {
410         await DisconnectAsync();
411     }
412
413     _logger.Connecting(webSocketUrl);
414     await _client.ConnectAsync(webSocketUrl, _token);
415     _ = await Task.Factory.StartNew(ReceiveLoop, _token,
416         TaskCreationOptions.LongRunning, TaskScheduler.Default);
417     _ = await Task.Factory.StartNew(SendPingLoop, _token,
418         TaskCreationOptions.LongRunning, TaskScheduler.Default);
419     _ = await Task.Factory.StartNew(CheckPingLoop, _token,
420         TaskCreationOptions.LongRunning, TaskScheduler.Default);
421     StatusChanged?.Invoke(this, new() { IsConnected = true });
422     _logger.Connected(webSocketUrl);
423 }
424 /// <inheritdoc/>
425 public async Task DisconnectAsync()
426 {
427     if (_client.State is WebSocketState.Closed)
428     {
429         return;
430     }
431
432     _logger.Disconnecting();
433     _tokenSource.CancelAfter(_cancelWaitInterval);
434     await _client.CloseAsync(
435         WebSocketCloseStatus.NormalClosure,
436         "WebSocket is closing.",
437         _token);
438
439     _tokenSource = new();
440     _token = _tokenSource.Token;
441     _pingStrings.Clear();
442     _client.Dispose();
443     _client = new ClientWebSocket();
444     StatusChanged?.Invoke(this, new() { IsConnected = false });
445     _logger.Disconnected();
446 }
447 /// <summary>
448 /// Sends the specified message to the client WebSocket.
449 /// 
450 /// <param name="T">The type of the message to be sent.</param>
451 /// <param name="message">The message to be sent.</param>
452 /// <returns>A <see cref="Task"/> object that represents the asynchronous
453 /// operation.</returns>
454 private async Task SendAsync<T>(T message)
455 {
456     string dataJson = JsonSerializer.Serialize(message);
457     byte[] messageBuffer = Encoding.UTF8.GetBytes(dataJson);
458
459     _logger.Sending(dataJson);
```

```

456     await _client.SendAsync(
457         new ArraySegment<byte>(messageBuffer),
458         WebSocketMessageType.Text,
459         true,
460         _token);
461
462     _logger.Sent(dataJson);
463 }
464 }
```

Listing A.8.22: EasonEetwViewer.Dmdata.WebSocket/Services/WebSocketClient.cs

A.9 Kmoni Settings Helper

```

1  using EasonEetwViewer.Services.Kmoni.Dtos;
2  using EasonEetwViewer.Services.Kmoni.Services;
3
4  namespace EasonEetwViewer.Services.Kmoni.Options;
5  ///<summary>
6  /// Represents the options for the <see cref="KmoniSettingsHelper"/>.
7  ///</summary>
8  public sealed record KmoniSettingsHelperOptions
9  {
10     ///<summary>
11     /// The filepath to which the setting is stored.
12     ///</summary>
13     public required string FilePath { get; init; }
14     ///<summary>
15     /// The default settings for <see cref="KmoniSettings"/>.
16     ///</summary>
17     public required KmoniSettings Default { get; init; }
18 }
```

Listing A.9.1: EasonEetwViewer.Services.Kmoni/Options/KmoniSettingsHelperOptions.cs

```

1  using System.Text.Json;
2  using EasonEetwViewer.KyoshinMonitor;
3  using EasonEetwViewer.Services.Kmoni.Abstractions;
4  using EasonEetwViewer.Services.Kmoni.Dtos;
5  using EasonEetwViewer.Services.Kmoni.Options;
6  using EasonEetwViewer.Services.Kmoni.Services;
7  using Microsoft.Extensions.DependencyInjection;
8  using Microsoft.Extensions.Logging;
9  using Microsoft.Extensions.Options;
10
11 namespace EasonEetwViewer.Services.Kmoni.Extensions;
12
13 ///<summary>
14 /// Provides extension methods for <see cref="IServiceCollection"/> to add <see
15     cref="KmoniSettingsHelper"/>.
16 ///</summary>
17 public static class KmoniSettingsHelperServiceCollectionExtensions
18 {
19     ///<summary>
20     /// Adds a <see cref="KmoniSettingsHelper"/> to the service collection.
21     ///</summary>
```

```

21     /// <param name="services">The services collection for which the service is added
22     /// to.</param>
23     /// <returns>The <see cref="IServiceCollection"/> so that calls can be
24     /// chained.</returns>
25     public static IServiceCollection AddKmoniOptionsHelper(this IServiceCollection
26         services)
27         => services.AddSingleton(sp
28             =>
29             {
30                 JsonSerializerOptions serializerOptions =
31                     sp.GetRequiredService<JsonSerializerOptions>();
32                 KmoniSettingsHelperOptions options = sp.GetRequiredService<IOptions<K_
33                     moniSettingsHelperOptions>>().Value;
34                 string filePath = options.FilePath;
35                 ILogger<KmoniSettingsHelper> logger =
36                     sp.GetRequiredService<ILogger<KmoniSettingsHelper>>();
37                 KmoniSettings? settings;
38                 try
39                 {
40                     settings = JsonSerializer.Deserialize<KmoniSettings>(File.ReadAllText(
41                         filePath), serializerOptions);
42                     logger.IOsucceeded();
43                 }
44                 catch
45                 {
46                     settings = options.Default;
47                     logger.IOFailed(filePath);
48                 }
49
50                 settings ??= options.Default;
51                 IKmoniSettingsHelper helper = new KmoniSettingsHelper(settings,
52                     logger);
53                 helper.KmoniSettingsChanged += (sender, e)
54                     =>
55                     {
56                         try
57                         {
58                             File.WriteAllText(
59                                 filePath,
60                                 JsonSerializer.Serialize(
61                                     e.KmoniSettings,
62                                     serializerOptions));
63                         }
64                         catch
65                         {
66                             logger.IOFailed(filePath);
67                         }
68                     };
69                 return helper;
70             });
71         }
72     }

```

Listing A.9.2: EasonEetwViewer.Services.Kmoni/Extensions/KmoniSettingsHelperServiceCollectionExtensions.cs

```

1  using System.Text.Json.Serialization;
2  using EasonEetwViewer.KyoshinMonitor.Abstractions;
3

```

```

4  namespace EasonEetwViewer.Services.Kmoni.Dtos;
5  ///<summary>
6  /// Represents the settings for the kmoni.
7  ///</summary>
8  public record KmoniSettings
{
9
10    ///<summary>
11    /// The sensor choice.
12    ///</summary>
13    [JsonPropertyName("SensorChoice")]
14    public required SensorType SensorChoice { get; set; }
15    ///<summary>
16    /// The measurement choice.
17    ///</summary>
18    [JsonPropertyName("MeasurementChoice")]
19    public required MeasurementType MeasurementChoice { get; set; }
20}

```

Listing A.9.3: EasonEetwViewer.Services.Kmoni/Dtos/KmoniSettings.cs

```

1  using EasonEetwViewer.KyoshinMonitor.Abstractions;
2  using EasonEetwViewer.Services.Kmoni.Events;
3
4  namespace EasonEetwViewer.Services.Kmoni.Abstractions;
5  ///<summary>
6  /// Represents a helper for managing kmoni settings.
7  ///</summary>
8  public interface IKmoniSettingsHelper
{
9
10    ///<summary>
11    /// The choice for the measurement.
12    ///</summary>
13    MeasurementType MeasurementChoice { get; set; }
14    ///<summary>
15    /// The choice for the sensor.
16    ///</summary>
17    SensorType SensorChoice { get; set; }
18    ///<summary>
19    /// The event raised when the settings changed.
20    ///</summary>
21    event EventHandler<KmoniSettingsChangedEventArgs>? KmoniSettingsChanged;
22}

```

Listing A.9.4: EasonEetwViewer.Services.Kmoni/Abstractions/IKmoniSettingsHelper.cs

```

1  using EasonEetwViewer.Services.Kmoni.Dtos;
2
3  namespace EasonEetwViewer.Services.Kmoni.Events;
4  ///<summary>
5  /// The event arguments for when kmoni settings are changed.
6  ///</summary>
7  public class KmoniSettingsChangedEventArgs : EventArgs
{
8
9    ///<summary>
10   /// The new settings that are changed to.
11   ///</summary>
12   public required KmoniSettings KmoniSettings { get; init; }
13}

```

Listing A.9.5: EasonEtwViewer.Services.Kmoni/Events/KmoniSettingsChangedEventArgs.cs

```
1  using EasonEtwViewer.KyoshinMonitor.Abstractions;
2  using EasonEtwViewer.Services.Kmoni.Services;
3  using Microsoft.Extensions.Logging;
4
5  namespace EasonEtwViewer.KyoshinMonitor;
6  /// <summary>
7  /// Represents the log messages used in <see cref="KmoniSettingsHelper"/>.
8  /// </summary>
9  internal static partial class KmoniSettingsHelperLogs
10 {
11     /// <summary>
12     /// Log when instantiated.
13     /// </summary>
14     /// <param name="logger">The logger to be used.</param>
15     [LoggerMessage(
16         EventId = 0,
17         EventName = nameof(Instantiated),
18         Level = LogLevel.Information,
19         Message = "Instantiated.")]
20     public static partial void Instantiated(
21         this ILogger<KmoniSettingsHelper> logger);
22
23     /// <summary>
24     /// Log when an I/O operation failed.
25     /// </summary>
26     /// <param name="logger">The logger to be used.</param>
27     /// <param name="filePath">The path to the file.</param>
28     [LoggerMessage(
29         EventId = 1,
30         EventName = nameof(IOFailed),
31         Level = LogLevel.Error,
32         Message = "File Read/Write Failed: '{FilePath}'")]
33     public static partial void IOFailed(
34         this ILogger<KmoniSettingsHelper> logger, string filePath);
35
36     /// <summary>
37     /// Log when an I/O operation succeeded.
38     /// </summary>
39     /// <param name="logger">The logger to be used.</param>
40     [LoggerMessage(
41         EventId = 1,
42         EventName = nameof(IOSucceeded),
43         Level = LogLevel.Debug,
44         Message = "File Read/Write Succeeded.")]
45     public static partial void IOSucceeded(
46         this ILogger<KmoniSettingsHelper> logger);
47
48     /// <summary>
49     /// Log when sensor choice changed.
50     /// </summary>
51     /// <param name="logger">The logger to be used.</param>
52     /// <param name="sensorChoice">The new sensor choice.</param>
53     [LoggerMessage(
54         EventId = 3,
55         EventName = nameof(SensorChoiceChanged),
```

```

56     Level = LogLevel.Information,
57     Message = "Sensor choice changed to: `'{SensorChoice}``."]
58     public static partial void SensorChoiceChanged(
59         this ILogger<KmoniSettingsHelper> logger, SensorType sensorChoice);
60
61     /// <summary>
62     /// Log when measurement choice changed.
63     /// </summary>
64     /// <param name="logger">The logger to be used.</param>
65     /// <param name="measurementChoice">The new measurement choice.</param>
66     [LoggerMessage(
67         EventId = 4,
68         EventName = nameof(MeasurementChoiceChanged),
69         Level = LogLevel.Information,
70         Message = "Measurement choice changed to: `'{MeasurementChoice}``."]
71     public static partial void MeasurementChoiceChanged(
72         this ILogger<KmoniSettingsHelper> logger, MeasurementType measurementChoice);
73 }
```

Listing A.9.6: EasonEetwViewer.Services.Kmoni/Logging/KmoniSettingsHelperLogs.cs

```

1  using EasonEetwViewer.KyoshinMonitor;
2  using EasonEetwViewer.KyoshinMonitor.Abstractions;
3  using EasonEetwViewer.Services.Kmoni.Abstractions;
4  using EasonEetwViewer.Services.Kmoni.Dtos;
5  using EasonEetwViewer.Services.Kmoni.Events;
6  using Microsoft.Extensions.Logging;
7
8  namespace EasonEetwViewer.Services.Kmoni.Services;
9
10    /// <summary>
11    /// The default implementation of <see cref="IKmoniSettingsHelper"/>.
12    /// </summary>
13    internal sealed class KmoniSettingsHelper : IKmoniSettingsHelper
14    {
15        /// <summary>
16        /// The logger to be used.
17        /// </summary>
18        private readonly ILogger<KmoniSettingsHelper> _logger;
19
20        /// <summary>
21        /// The settings for kmoni.
22        /// </summary>
23        private readonly KmoniSettings _settings;
24
25        /// <inheritdoc/>
26        public event EventHandler<KmoniSettingsChangedEventArgs>? KmoniSettingsChanged;
27
28        /// <inheritdoc/>
29        public SensorType SensorChoice
30        {
31            get
32            => _settings.SensorChoice;
33            set
34            {
35                if (_settings.SensorChoice != value)
36                {
37                    _settings.SensorChoice = value;
38                    KmoniSettingsChanged?.Invoke(this, new() { KmoniSettings = _settings
39                        => });
40                }
41            }
42        }
43    }
44 }
```

```

38             _logger.SensorChoiceChanged(value);
39         }
40     }
41 }
42
43 /// <inheritdoc/>
44 public MeasurementType MeasurementChoice
45 {
46     get
47         => _settings.MeasurementChoice;
48     set
49     {
50         if (_settings.MeasurementChoice != value)
51         {
52             _settings.MeasurementChoice = value;
53             KmoniSettingsChanged?.Invoke(this, new() { KmoniSettings = _settings
54             => });
55             _logger.MeasurementChoiceChanged(value);
56         }
57     }
58 }
59
60 /// <summary>
61 /// Initializes a new instance of the <see cref="KmoniSettingsHelper"/>.
62 /// </summary>
63 /// <param name="settings">The initial settings.</param>
64 /// <param name="logger">The logger to be used.</param>
65 public KmoniSettingsHelper(KmoniSettings settings, ILogger<KmoniSettingsHelper>
66     => logger)
67 {
68     _settings = settings;
69     _logger = logger;
70     _logger.Instantiated();
71 }
72 }
```

Listing A.9.7: EasonEetwViewer.Services.Kmoni/Services/KmoniSettingsHelper.cs

A.10 GUI

```

1  {
2      "KmoniHelperOptions": {
3          "BaseUri": "http://www.kmoni.bosai.go.jp/data/map_img/RealTimeImg/",
4          "RelativeUri": "{0}_{1}/{2:yyyyMMdd}/{2:yyyyMMddHHmmss}.{0}_{1}.gif",
5          "FilePath": "ObservationPoints.json"
6      },
7      "TimeTableOptions": {
8          "FilePath": "tjma2001.txt"
9      },
10     "OAuth2Options": {
11         "WebPageString": "<html><body><h1>You may close this
12             => window.</h1></body></html>",
13         "RedirectPath": "eeetwv-code-auth/",
14         "BaseUri": "https://manager.dmddata.jp/account/oauth2/v1/",
15         "Host": "manager.dmddata.jp",
16         "ClientId": "CId.RRV95iuUV9FrYzeIN_BYM9Z35MJwwQen5DIwJ8JQXaTm",
17         "Scopes": [
18             "read"
19         ]
20     }
21 }
```

```

17     "contract.list",
18     "eew.get.forecast",
19     "gd.earthquake",
20     "parameter.earthquake",
21     "socket.close",
22     "socket.list",
23     "socket.start",
24     "telegram.data",
25     "telegram.get.earthquake",
26     "telegram.list"
27   ]
28 },
29 "BaseUrls": {
30   "Api": "https://api.dmdata.jp/v2/",
31   "Telegram": "https://data.api.dmdata.jp/v1/"
32 },
33 "KmoniSettingsHelperOptions": {
34   "FilePath": "KmoniOptions.json",
35   "Default": {
36     "SensorChoice": 0,
37     "MeasurementChoice": 1
38   }
39 },
40 "ConfigPaths": {
41   "Authenticator": "Authenticator.txt",
42   "Language": "Lang.txt"
43 },
44 "WebSocketParams": {
45   "AppName": "EEETWV Test",
46   "Classifications": [
47     "EewForecast",
48     "TelegramEarthquake"
49   ],
50   "Types": [
51     "VXSE45",
52     "VTSE41"
53   ],
54   "TestStatus": "Exclude",
55   "FormatMode": "Json"
56 }
57 }

```

Listing A.10.1: EasonEetwViewer/appsettings.json

```

1 using Avalonia.Controls;
2 using Avalonia.Controls.Templates;
3 using EasonEetwViewer.ViewModels.ViewModelBases;
4
5 namespace EasonEetwViewer;
6 /// <summary>
7 /// The view locator for the application.
8 /// </summary>
9 internal class ViewLocator : IDataTemplate
{
10   /// <inheritdoc/>
11   public Control? Build(object? param)
12   {
13     if (param is null)

```

```

15     {
16         return null;
17     }
18
19     string name = param.GetType().FullName!.Replace("ViewModel", "View",
20         StringComparison.OrdinalIgnoreCase);
21     Type? type = Type.GetType(name);
22
23     return type is not null
24         ? (Control)Activator.CreateInstance(type)!
25         : new TextBlock { Text = "Not Found: " + name };
26 }
27 /// <inheritdoc/>
28 public bool Match(object? data)
29     => data is ViewModelBase;
}

```

Listing A.10.2: EasonEetwViewer/ViewLocator.cs

```

1 <Application xmlns="https://github.com/avaloniaui"
2     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
3     x:Class="EasonEetwViewer.App"
4     xmlns:local="using:EasonEetwViewer"
5     RequestedThemeVariant="Dark">
6     <!-- "Default" ThemeVariant follows system theme variant. "Dark" or
7         "Light" are other available options. -->
8
9     <Application.DataTemplates>
10    <local:ViewLocator/>
11  </Application.DataTemplates>
12
13  <Application.Styles>
14    <FluentTheme />
15    <StyleInclude
16        Source="avares://Avalonia.Controls.DataGrid/Themes/Fluent.xaml"/>
17    <StyleInclude Source="avares://EasonEetwViewer/Assets/Icons.axaml" />
18  </Application.Styles>
19 </Application>

```

Listing A.10.3: EasonEetwViewer/App.axaml

```

1 using System.Text.Json;
2 using System.Text.Json.Serialization;
3 using Avalonia;
4 using Avalonia.Controls.ApplicationLifetimes;
5 using Avalonia.Data.Core.Plugins;
6 using Avalonia.Logging;
7 using Avalonia.Markup.Xaml;
8 using EasonEetwViewer.Dmdata.Api.Dtos.Request;
9 using EasonEetwViewer.Dmdata.Api.Extensions;
10 using EasonEetwViewer.Dmdata.Authentication.Extensions;
11 using EasonEetwViewer.Dmdata.Authentication.Options;
12 using EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
13 using EasonEetwViewer.Dmdata.Telegram.Extensions;
14 using EasonEetwViewer.Dmdata.WebSocket.Extensions;
15 using EasonEetwViewer.Extensions;

```

```
16  using EasonEetwViewer.JmaTravelTime.Extensions;
17  using EasonEetwViewer.JmaTravelTime.Options;
18  using EasonEetwViewer.KyoshinMonitor.Dtos;
19  using EasonEetwViewer.KyoshinMonitor.Extensions;
20  using EasonEetwViewer.Services;
21  using EasonEetwViewer.Services.Kmoni.Extensions;
22  using EasonEetwViewer.Services.Kmoni.Options;
23  using EasonEetwViewer.Services.Logging;
24  using EasonEetwViewer.Services.TimeProvider;
25  using EasonEetwViewer.ViewModels;
26  using EasonEetwViewer.Views;
27  using Microsoft.Extensions.Configuration;
28  using Microsoft.Extensions.DependencyInjection;
29  using Microsoft.Extensions.Logging;
30
31  namespace EasonEetwViewer;
32  ///<inheritdoc/>
33  internal partial class App : Application
34  {
35      ///<inheritdoc/>
36      public override void Initialize()
37          => AvaloniaXamlLoader.Load(this);
38      ///<summary>
39      /// The service provider for the application.
40      ///</summary>
41      public static IServiceProvider? Service { get; private set; }
42
43      ///<inheritdoc/>
44      public override async void OnFrameworkInitializationCompleted()
45      {
46          IConfigurationRoot config = new
47              ConfigurationBuilder().AddJsonFile("appsettings.json").Build();
48
49          IConfigurationSection baseUrls = config.GetSection("BaseUrls");
50          string baseApi = baseUrls["Api"]!;
51          string baseTelegram = baseUrls["Telegram"]!;
52
53          IConfigurationSection configPaths = config.GetSection("ConfigPaths");
54          string authenticatorPath = configPaths["Authenticator"]!;
55          string languagePath = configPaths["Language"]!;
56
57          IServiceCollection collection = new ServiceCollection()
58              .AddLogging(loggingBuilder
59                  => loggingBuilder
60                      .AddFileLogger(
61                          new StreamWriter($"{DateTime.UtcNow:yyyyMMddHHmmss}.log"),
62                          LogLevel.Warning)
63 #if DEBUG
64                 .AddDebug()
65 #endif
66                 .SetMinimumLevel(LogLevel.Information))
67
67 .AddSingleton(sp
68     => new JsonSerializerOptions()
69     {
70         NumberHandling = JsonNumberHandling.AllowReadingFromString,
71         RespectNullableAnnotations = true,
72         RespectRequiredConstructorParameters = true
73     })
74
```

```
74
75     .AddApiCaller(baseApi)
76     .AddTelegramRetriever(baseTelegram)
77     .AddWebSocket()
78
79     .AddSingleton<ITimeProvider, DefaultTimeProvider>()
80
81     .AddKmoniOptionsHelper()
82     .AddOptions<KmoniSettingsHelperOptions>()
83         .Bind(config.GetSection("KmoniSettingsHelperOptions"))
84         .Services
85
86     .AddAuthenticator(authenticatorPath)
87     .AddOptions<OAuth2Options>()
88         .Bind(config.GetSection("OAuth2Options"))
89         .Services
90
91     .AddKmoniHelpers()
92     .AddOptions<KmoniHelperOptions>()
93         .Bind(config.GetSection("KmoniHelperOptions"))
94         .Services
95
96     .AddJmaTimeTable()
97     .AddOptions<JmaTimeTableOptions>()
98         .Bind(config.GetSection("TimeTableOptions"))
99         .Services
100
101    .AddSingleton<MapResourcesProvider>()
102
103    .AddSingleton<ILogSink, AvaloniaToMicrosoftLoggingAdaptor>()
104
105    .AddSingleton(languagePath)
106    .AddSingleton<MainWindowViewModel>()
107    .AddSingleton<RealtimePageViewModel>()
108    .AddSingleton<PastPageViewModel>()
109    .AddSingleton<SettingPageViewModel>()
110    .AddOptions<WebSocketStartPost>()
111        .Bind(config.GetSection("WebSocketParams"))
112        .Services;
113
114    Service = collection
115        .BuildServiceProvider()
116        .AttachMapsuiLogging();
117
118    Logger.Sink = Service.GetRequiredService<ILogSink>();
119
120    if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime desktop)
121    {
122        DisableAvaloniaDataAnnotationValidation();
123        desktop.MainWindow = new MainWindow
124        {
125            DataContext = Service.GetRequiredService<MainWindowViewModel>(),
126        };
127    }
128
129    base.OnFrameworkInitializationCompleted();
130
131 #if DEBUG
132     await TestTelegram();
```

```

133  #endif
134  }
135
136  private async Task TestTelegram()
137  {
138      await Task.Delay(5000);
139      EewInformationSchema eewData =
140          → JsonSerializer.Deserialize<EewInformationSchema>(
141              File.ReadAllText("TestEew.json"),
142              Service!.GetRequiredService<JsonSerializerOptions>()!;
143      Service!.GetRequiredService<RealtimePageViewModel>().WebSocketClientDataRecei
144          → vedEventHandler(this, new() { Telegram = eewData });
145      EewInformationSchema eewData2 =
146          → JsonSerializer.Deserialize<EewInformationSchema>(
147              File.ReadAllText("TestEew2.json"),
148              Service!.GetRequiredService<JsonSerializerOptions>()!;
149      Service!.GetRequiredService<RealtimePageViewModel>().WebSocketClientDataRecei
150          → vedEventHandler(this, new() { Telegram = eewData2 });
151
152
153
154
155     /// <summary>
156     /// Avoid duplicate validations from both Avalonia and the CommunityToolkit.
157     /// </summary>
158     /// <remarks>
159     /// <see href="https://docs.avaloniaui.net/docs/guides/development-guides/data-va
160         → lidation#manage-validationplugins">More Info</see>
161     /// </remarks>
162     private static void DisableAvaloniaDataAnnotationValidation()
163     {
164         // Get an array of plugins to remove
165         DataAnnotationsValidationPlugin[] dataValidationPluginsToRemove =
166             [.. BindingPlugins.DataValidators.OfType<DataAnnotationsValidationPlugin>
167                 ()];
168
169         // remove each entry found
170         foreach (DataAnnotationsValidationPlugin plugin in
171             → dataValidationPluginsToRemove)
172         {
173             _ = BindingPlugins.DataValidators.Remove(plugin);
174         }
175     }
176 }

```

Listing A.10.4: EasonEetwViewer/App.axaml.cs

```

1  using Avalonia;
2
3  namespace EasonEetwViewer;
4  /// <summary>
5  /// The entry point of the application.

```

```

6  /// </summary>
7  internal sealed class Program
8  {
9      // Initialization code. Don't use any Avalonia, third-party APIs or any
10     // SynchronizationContext-reliant code before AppMain is called: things aren't
11     // initialized
12     // yet and stuff might break.
13     /// <summary>
14     /// The entry point of the application.
15     /// </summary>
16     /// <param name="args">The arguments.</param>
17     /// <returns>The exit code.</returns>
18     [STAThread]
19     public static int Main(string[] args)
20         => BuildAvaloniaApp()
21             .StartWithClassicDesktopLifetime(args);
22
23     // Avalonia configuration, don't remove; also used by visual designer.
24     /// <summary>
25     /// The Avalonia configuration.
26     /// </summary>
27     /// <returns>The <see cref="AppBuilder"/>.</returns>
28     public static AppBuilder BuildAvaloniaApp()
29         => AppBuilder.Configure<App>()
30             .UsePlatformDetect()
31             .WithInterFont();
32 }
```

Listing A.10.5: EasonEetwViewer/Program.cs

```

1  using CommunityToolkit.Mvvm.ComponentModel;
2  using CommunityToolkit.Mvvm.Input;
3  using EasonEetwViewer.Lang;
4  using EasonEetwViewer.Models.MainWindow;
5  using EasonEetwViewer.ViewModels.ViewModelBases;
6  using Microsoft.Extensions.Logging;
7
8  namespace EasonEetwViewer.ViewModels;
9
10    // Adapted from https://github.com/MammaMiaDev/avaloniaui-the-series Episode 3 with
11    // edits
12    /// <summary>
13    /// The view model for the main window.
14    /// </summary>
15    internal sealed partial class MainWindowViewModel : ViewModelBase
16    {
17        /// <summary>
18        /// The currently selected view model.
19        /// </summary>
20        public ViewModelBase CurrentPage
21            => SelectedSidebarItem.Model;
22
23        /// <summary>
24        /// The currently selected sidebar item.
25        /// </summary>
26        [ObservableProperty]
27        [NotifyPropertyChangedFor(nameof(CurrentPage))]
28        private SidebarItemTemplate _selectedSidebarItem;
29        /// <summary>
```

```
28     /// When the selected sidebar item has changed.
29     /// </summary>
30     /// <param name="oldValue">The previously selected item.</param>
31     /// <param name="newValue">The new selected item.</param>
32     partial void OnSelectedSidebarItemChanged(SidebarItemTemplate? oldValue,
33         SidebarItemTemplate newValue)
34     {
35         _logger.ViewModelSwitched(newValue.Model.GetType());
36     }
37     /// <summary>
38     /// Event handler for when data is received on WebSocket.
39     /// </summary>
40     /// <param name="sender">The sender of the event.</param>
41     /// <param name="e">The event arguments.</param>
42     private void WebSocketDataReceivedEventHandler(object? sender, EventArgs e)
43     {
44         SelectedSidebarItem = SidebarItems.ElementAt(0);
45         _logger.SwitchingToRealTime();
46     }
47     /// <summary>
48     /// The logger to be used.
49     /// </summary>
50     private readonly ILogger<MainWindowViewModel> _logger;
51
52     /// <summary>
53     /// Creates a new instance of the <see cref="MainWindowViewModel"/> class.
54     /// </summary>
55     /// <param name="realtimePageViewModel">The realtime page view model
56     /// <param name="pastPageViewModel">The past page view model instance.</param>
57     /// <param name="settingPageViewModel">The setting page view model
58     /// <param name="logger">The logger to be used.</param>
59     public MainWindowViewModel(
60         RealtimePageViewModel realtimePageViewModel,
61         PastPageViewModel pastPageViewModel,
62         SettingPageViewModel settingPageViewModel,
63         ILogger<MainWindowViewModel> logger)
64         : base(logger)
65     {
66         _logger = logger;
67         SidebarItems =
68             [
69                 new SidebarItemTemplate(
70                     realtimePageViewModel,
71                     "LiveRegular",
72                     MainWindowResources.PageNameRealtime),
73                 new SidebarItemTemplate(
74                     pastPageViewModel,
75                     "HistoryRegular",
76                     MainWindowResources.PageNamePast),
77                 new SidebarItemTemplate(
78                     settingPageViewModel,
79                     "SettingsRegular",
80                     MainWindowResources.PageNameSettings),
81             ];
82         SelectedSidebarItem = SidebarItems.ElementAt(0);
83         realtimePageViewModel.WebSocketDataReceived +=
84             WebSocketDataReceivedEventHandler;
```

```

83     }
84
85     /// <summary>
86     /// The sidebar items.
87     /// </summary>
88     public IEnumerable<SidebarItemTemplate> SidebarItems { get; init; }
89     /// <summary>
90     /// Whether the sidebar pane is open.
91     /// </summary>
92     [ObservableProperty]
93     private bool _isPaneOpen = true;
94     /// <summary>
95     /// To open or close the sidebar pane.
96     /// </summary>
97     [RelayCommand]
98     private void TriggerPane()
99     {
100         IsPaneOpen ^= true;
101         _logger.PaneTriggered(IsPaneOpen);
102     }
103 }
```

Listing A.10.6: EasonEetwViewer/ViewModels/MainWindowViewModel.cs

```

1  using System.Diagnostics;
2  using System.Timers;
3  using CommunityToolkit.Mvvm.ComponentModel;
4  using EasonEetwViewer.Dmdata.Api.Abstractions;
5  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
6  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
7  using EasonEetwViewer.Dmdata.Telegram.Abstractions;
8  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
9  using EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
10 using EasonEetwViewer.Dmdata.Telegram.Dtos.TelegramBase;
11 using EasonEetwViewer.Dmdata.WebSocket.Abstractions;
12 using EasonEetwViewer.Extensions;
13 using EasonEetwViewer.JmaTravelTime.Abstractions;
14 using EasonEetwViewer.KyoshinMonitor.Abstractions;
15 using EasonEetwViewer.KyoshinMonitor.Extensions;
16 using EasonEetwViewer.Lang;
17 using EasonEetwViewer.Models.RealTimePage;
18 using EasonEetwViewer.Services;
19 using EasonEetwViewer.Services.Kmoni.Abstractions;
20 using EasonEetwViewer.Services.TimeProvider;
21 using EasonEetwViewer.ViewModels.ViewModelBases;
22 using Mapsui;
23 using Mapsui.Layers;
24 using Mapsui.Nts.Extensions;
25 using Mapsui.Projections;
26 using Mapsui.Styles;
27 using Microsoft.Extensions.Logging;
28 using NetTopologySuite.Geometries;
29 using Coordinate = NetTopologySuite.Geometries.Coordinate;
30 using DataReceivedEventArgs =
31     → EasonEetwViewer.Dmdata.WebSocket.Events.DataReceivedEventArgs;
31 using IFeature = Mapsui.IFeature;
32 using Polygon = NetTopologySuite.Geometries.Polygon;
33 using Timer = System.Timers.Timer;
```

```
34
35 namespace EasonEetwViewer.ViewModels;
36
37 /// <summary>
38 /// The view model for the real-time page.
39 /// </summary>
40 internal partial class RealtimePageViewModel : MapViewModelBase
{
    /// <summary>
    /// Creates a new instance of the <see cref="RealtimePageViewModel"/> class.
    /// </summary>
    /// <param name="imageFetch">The image fetcher to be used.</param>
    /// <param name="pointExtract">The point extractor to be used.</param>
    /// <param name="timeTable">The time table to be used.</param>
    /// <param name="kmoniOptions">The kmoni options to be used.</param>
    /// <param name="webSocketClient">The WebSocket client to be used.</param>
    /// <param name="resources">The map resources to be used.</param>
    /// <param name="authenticatorWrapper">The authenticator to be used.</param>
    /// <param name="apiCaller">The API caller to be used.</param>
    /// <param name="telegramRetriever">The telegram retriever to be used.</param>
    /// <param name="telegramParser">The telegram parser to be used.</param>
    /// <param name="timeProvider">The time provider to be used.</param>
    /// <param name="logger">The logger to be used.</param>
    public RealtimePageViewModel(
        IImageFetch imageFetch,
        IPPointExtract pointExtract,
        ITimeTable timeTable,
        IKmoniSettingsHelper kmoniOptions,
        IWebSocketClient webSocketClient,
        MapResourcesProvider resources,
        IAuthenticationHelper authenticatorWrapper,
        IApiCaller apiCaller,
        ITTelegramRetriever telegramRetriever,
        ITTelegramParser telegramParser,
        ITimeProvider timeProvider,
        ILogger<RealtimePageViewModel> logger)
        : base(resources,
            authenticatorWrapper,
            apiCaller,
            telegramRetriever,
            telegramParser,
            timeProvider,
            logger)
    {
        _logger = logger;

        _kmoniOptions = kmoniOptions;
        _kmoniOptions.KmoniSettingsChanged += (o, e)
            => OnPropertyChanged(nameof(DataLegendValues));
        _kmoniOptions.KmoniSettingsChanged += (o, e)
            => OnPropertyChanged(nameof(DataLegendText));

        _imageFetch = imageFetch;
        _pointExtract = pointExtract;

        _timer1000 = new Timer(1000)
        {
            AutoReset = true,
            Enabled = true
        }
    }
}
```

```
93     };
94     _timer1000.Elapsed += RefreshKmoniDataEventHandler;
95     _timer1000.Elapsed += RemoveExpiredTsunamiEventHandler;
96     _timer1000.Elapsed += RemoveExpiredEewEventHandler;
97     _timer1000.Elapsed += (o, e)
98         => OnPropertyChanged(nameof(TimeDisplay));
99
100    _timer2500 = new Timer(2500)
101    {
102        AutoReset = true,
103        Enabled = true
104    };
105    _timer2500.Elapsed += SwitchEew;
106
107    _webSocketClient = webSocketClient;
108    _webSocketClient.DataReceived += WebSocketClientDataReceivedEventHandler;
109
110    _timeTableProvider = timeTable;
111
112    _tsunamiLayer = new()
113    {
114        DataSource = _resources.Tsunami,
115        Style = null
116    };
117
118    Map.Layers.Add(_kmoniLayer, _kmoniGroup);
119    Map.Layers.Add(_tsunamiLayer, _tsunamiGroup);
120}
121/// <summary>
122/// The logger to be used.
123/// </summary>
124private readonly ILogger<RealtimePageViewModel> _logger;
125/// <summary>
126/// The timer to trigger events every second.
127/// </summary>
128/// <remarks>
129/// Event triggered includes: Refreshing Time Display, Remove Expired Warnings,
130/// → Refresh kmoni Layer.
131/// </remarks>
132private readonly Timer _timer1000;
133/// <summary>
134/// The timer to trigger events every 2.5 seconds.
135/// </summary>
136/// <remarks>
137/// Event triggered includes: Switching EEW.
138/// </remarks>
139private readonly Timer _timer2500;
140
141/// <summary>
142/// The group for the hypocentre layer, on the top.
143/// </summary>
144private const int _hypocentreGroup = 4;
145/// <summary>
146/// The group for the wavefront layer, below the hypocentre layer.
147/// </summary>
148private const int _wavefrontGroup = 3;
149/// <summary>
150/// The group for the tsunami warning layer, below the wavefront layer.
151/// </summary>
```

```
151     private const int _tsunamiGroup = 2;
152     /// <summary>
153     /// The group for the kmoni layer, below the tsunami warning layer.
154     /// </summary>
155     private const int _kmoniGroup = 1;
156     /// <summary>
157     /// The group for the intensity layer, at the bottom.
158     /// </summary>
159     private const int _intensityGroup = 0;
160
161     #region eew
162     private const string _eewHypocentreLayerPrefix = "Hypocentre";
163     private const string _eewRegionLayerPrefix = "Regions";
164
165     private readonly ITimeTable _timeTableProvider;
166     private readonly TimeSpan _eewLifeTime = TimeSpan.FromMinutes(3);
167
168     private readonly Dictionary<string, EewDetailsTemplate> _liveEewList = [];
169     public EewDetailsTemplate? CurrentDisplayEew
170         => CurrentEewIndex is null || CurrentEewIndex >= _liveEewList.Count
171             ? null
172             : _liveEewList.ToList()[(int)CurrentEewIndex].Value;
173
174     [ObservableProperty]
175     [NotifyPropertyChangedFor(nameof(CurrentDisplayEew))]
176     private int? _currentEewIndex = null;
177
178     private async Task OnEewReceived(EewInformationSchema eew)
179     {
180         if (!int.TryParse(eew.SerialNo, out int serial) || eew.EventId is null)
181         {
182             return;
183         }
184
185         if (_liveEewList.TryGetValue(eew.EventId, out EewDetailsTemplate?
186             => existingEew))
187         {
188             if (existingEew.Serial > serial)
189             {
190                 return;
191             }
192
193             RemoveEew(existingEew);
194         }
195
196         DateTimeOffset lifeTime = eew.PressDateTime + _eewLifeTime;
197         if (lifeTime < _timeProvider.Now())
198         {
199             return;
200         }
201
202         EewDetailsTemplate eewTemplate = new(eew, lifeTime, serial);
203         _liveEewList[eew.EventId] = eewTemplate;
204
205         if (eew.Body.IsCancelled)
206         {
207             return;
208         }
209     }
210
211 }
```

```
209     CoordinateComponent? coordinates = eew.Body.Earthquake?.Hypocentre.Coordinate;
210     if (coordinates?.Longitude is not null && coordinates?.Latitude is not null)
211     {
212         MPoint point = (
213             coordinates.Longitude.DoubleValue,
214             coordinates.Latitude.DoubleValue)
215             .LonLatToMPoint();
216
217         MemoryLayer layer = new()
218         {
219             Name = _eewHypocentreLayerPrefix + eew.EventId,
220             Features = [new PointFeature(point)],
221             Style = eewTemplate.IsAssumedHypocentre
222                 ? _resources.PlumShapeStyle
223                 : _resources.HypocentreShapeStyle
224         };
225
226         if (!eewTemplate.Token.IsCancellationRequested)
227         {
228             Map.Layers.Add(layer, _hypocentreGroup);
229         }
230     }
231
232     if (eew.Body.Intensity is not null)
233     {
234         IEnumerable<Region> regions = eew.Body.Intensity.Regions;
235
236         ILayer layer = new Layer()
237         {
238             Name = _eewRegionLayerPrefix + eew.EventId,
239             DataSource = _resources.Region,
240             Style = regions.ToRegionStyle()
241         };
242
243         if (!eewTemplate.Token.IsCancellationRequested)
244         {
245             Map.Layers.Add(new RasterizingLayer(layer), _intensityGroup);
246         }
247     }
248
249     _ = await Task.Factory.StartNew(() => DrawEewCircles(eewTemplate),
250         eewTemplate.Token, TaskCreationOptions.LongRunning,
251         TaskScheduler.Default);
252 }
253 private void SwitchEew(object? sender, ElapsedEventArgs e)
254     => CurrentEewIndex =
255         _liveEewList.Count == 0
256             ? null
257             : CurrentEewIndex is null
258                 ? 0
259                 : (CurrentEewIndex + 1) % _liveEewList.Count;
260
261     private readonly IStyle _pCircleStyle
262         = new VectorStyle
263         {
264             Outline = new Pen
265             {
266                 Color = Color.Blue,
267                 Width = 2
```

```
266     },
267     Fill = null
268 };
269 private readonly IStyle _sCircleStyle
270 = new VectorStyle
271 {
272     Outline = new Pen
273     {
274         Color = Color.Orange,
275         Width = 2
276     },
277     Fill = null
278 };
279 private const string _eewWavefrontLayerPrefix = "Wavefront";
280 private readonly TimeSpan _refreshCircleInterval = TimeSpan.FromSeconds(1d / 60d);
281 private async Task DrawEewCircles(EewDetailsTemplate eew)
282 {
283     if (eew.Earthquake is null ||
284         eew.Earthquake.Hypocentre.Depth.Value is null ||
285         eew.Earthquake.Hypocentre.Coordinate.Latitude is null ||
286         eew.Earthquake.Hypocentre.Coordinate.Longitude is null ||
287         eew.Earthquake.OriginTime is null ||
288         eew.Earthquake.Condition is "仮定震源要素")
289     {
290         return;
291     }
292
293     int depth = (int)eew.Earthquake.Hypocentre.Depth.Value;
294     if (depth is < 0 or > 700)
295     {
296         return;
297     }
298
299     string pLayerName = _eewWavefrontLayerPrefix + "P" + eew.EventId;
300     string sLayerName = _eewWavefrontLayerPrefix + "S" + eew.EventId;
301
302     double latitude = eew.Earthquake.Hypocentre.Coordinate.Latitude.DoubleValue;
303     double longitude = eew.Earthquake.Hypocentre.Coordinate.Longitude.DoubleValue;
304
305     MemoryLayer pLayer = new()
306     {
307         Name = pLayerName,
308         Features = [],
309         Style = _pCircleStyle
310     };
311     MemoryLayer sLayer = new()
312     {
313         Name = sLayerName,
314         Features = [],
315         Style = _sCircleStyle
316     };
317
318     Map.Layers.Add(pLayer, _wavefrontGroup);
319     Map.Layers.Add(sLayer, _wavefrontGroup);
320
321     while (!eew.Token.IsCancellationRequested)
322     {
323         double time = ((TimeSpan)(_timeProvider.Now() -
324             eew.Earthquake.OriginTime)).TotalSeconds;
```

```
324     if (time < 0)
325     {
326         continue;
327     }
328
329     (double pDistance, double sDistance) =
330     → _timeTableProvider.DistanceFromDepthTime(depth, time);
331
332     if (pDistance != 0)
333     {
334         Polygon pCirclePolygon = CreateCircleRing(latitude, longitude,
335             → pDistance);
336         pLayer.Features = [pCirclePolygon.ToFeature()];
337         pLayer.DataHasChanged();
338     }
339
340     if (sDistance != 0)
341     {
342         Polygon sCirclePolygon = CreateCircleRing(latitude, longitude,
343             → sDistance);
344         sLayer.Features = [sCirclePolygon.ToFeature()];
345         sLayer.DataHasChanged();
346     }
347
348     await Task.Delay(_refreshCircleInterval);
349
350     _ = Map.Layers.Remove(pLayer);
351     _ = Map.Layers.Remove(sLayer);
352 }
353 private static Polygon CreateCircleRing(
354     double latitude,
355     double longitude,
356     double radius,
357     double quality = 360)
358 {
359     (double x, double y) = SphericalMercator.FromLonLat(longitude, latitude);
360     radius *= 1000;
361     radius *= 1.23; // a magic constant found by experimenting
362
363     ICollection<Coordinate> outerRing = [];
364     double increment = 360d / (quality < 3 ? 3 : (quality > 360 ? 360 : quality));
365     for (double angle = 0; angle < 360; angle += increment)
366     {
367         double angleInRadians = angle / 180 * Math.PI;
368         outerRing.Add(new Coordinate(x + (Math.Sin(angleInRadians) * radius), y +
369             → (Math.Cos(angleInRadians) * radius)));
370     }
371
372     outerRing.Add(outerRing.ElementAt(0));
373
374     return new Polygon(new LinearRing([.. outerRing]));
375 }
376 /// <summary>
377 /// Removes the expired EEWs.
378 /// </summary>
379 /// <param name="sender">The sender of the event.</param>
380 /// <param name="e">The event arguments.</param>
```

```

379     private void RemoveExpiredEewEventHandler(object? sender, ElapsedEventArgs e)
380     {
381         foreach (string eventId in _liveEewList.Keys.ToList())
382         {
383             if (!_liveEewList.TryGetValue(eventId, out EewDetailsTemplate? eew))
384             {
385                 continue;
386             }
387
388             if (eew.ExpiryTime < _timeProvider.Now())
389             {
390                 RemoveEew(eew);
391                 _ = _liveEewList.Remove(eventId);
392             }
393         }
394     }
395     /// <summary>
396     /// Removes the EEW from the map.
397     /// </summary>
398     /// <param name="eew">The EEW that is to be removed.</param>
399     private void RemoveEew(EewDetailsTemplate eew)
400     {
401         eew.TokenSource.Cancel();
402         eew.TokenSource.Dispose();
403         _ = Map.Layers.Remove(x => x.Name == (_eewHypocentreLayerPrefix +
404             eew.EventId));
405         _ = Map.Layers.Remove(x => x.Name == (_eewRegionLayerPrefix + eew.EventId));
406     }
407     #endregion
408
409     #region tsunami
410     /// <summary>
411     /// The current tsunami.
412     /// </summary>
413     [ObservableProperty]
414     private TsunamiDetailsTemplate? _currentTsunami = null;
415     /// <summary>
416     /// The default lifetime for a tsunami.
417     /// </summary>
418     private readonly TimeSpan _tsunamiLifeTime = TimeSpan.FromDays(2);
419     /// <summary>
420     /// The map layer for tsunami warnings.
421     /// </summary>
422     private readonly Layer _tsunamiLayer;
423     /// <summary>
424     /// Handles when a new tsunami telegram is received.
425     /// </summary>
426     /// <param name="tsunami">The tsunami telegram.</param>
427     private void OnTsunamiReceived(TsunamiInformationSchema tsunami)
428     {
429         DateTimeOffset validDateTime = tsunami.ValidDateTime ?? tsunami.PressDateTime
430             + _tsunamiLifeTime;
431         if (validDateTime < _timeProvider.Now())
432         {
433             return;
434         }
435
436         if (tsunami.Body.Tsunami?.Forecasts is not null)
437         {

```

```
436         _tsunamiLayer.Style = tsunami.Body.Tsunami.Forecasts.ToRegionStyle();
437         _tsunamiLayer.DataHasChanged();
438     }
439
440     CurrentTsunami = new(tsunami, validDateTime);
441 }
442 /// <summary>
443 /// Clears the expired tsunami warnings.
444 /// </summary>
445 /// <param name="source">The source of the event.</param>
446 /// <param name="e">The event arguments.</param>
447 private void RemoveExpiredTsunamiEventHandler(object? source, ElapsedEventArgs e)
448 {
449     if (CurrentTsunami is not null && CurrentTsunami.ExpiryTime <
450         → _timeProvider.Now())
451     {
452         CurrentTsunami = null;
453         _tsunamiLayer.Style = null;
454         _tsunamiLayer.DataHasChanged();
455     }
456 }
457 #endregion
458 #region websocket
459 /// <summary>
460 /// The WebSocket client to be used.
461 /// </summary>
462 private readonly IWebSocketClient _webSocketClient;
463 /// <summary>
464 /// Raises when there is data received over the WebSocket connection.
465 /// </summary>
466 public event EventHandler? WebSocketDataReceived;
467 /// <summary>
468 /// Handles when data is received by the WebSocket connection.
469 /// </summary>
470 /// <param name="sender">The sender of hte event.</param>
471 /// <param name="e">The event arguments.</param>
472 internal async void WebSocketClientDataReceivedEventHandler(object? sender,
473     → DataReceivedEventArgs e)
474 {
475     WebSocketDataReceived?.Invoke(this, new());
476
477     Head telegram = e.Telegram;
478     if (telegram is EewInformationSchema eew)
479     {
480         await OnEewReceived(eew);
481     }
482     else if (telegram is TsunamiInformationSchema tsunami)
483     {
484         OnTsunamiReceived(tsunami);
485     }
486 }
487 #endregion
488 #region kmoni
489 /// <summary>
490 /// The values to display for the current kmoni data choice.
491 /// </summary>
492 public decimal[] DataLegendValues
```

```
493     => [.. Enumerable.Range(0, 11)
494         .Select(x => (double)x / 10)
495         .Select(x => _kmoniOptions.MeasurementChoice switch
496         {
497             MeasurementType.MeasuredIntensity
498                 => x.HeightToIntensity(),
499             MeasurementType.PeakGroundAcceleration
500                 => x.HeightToPga(),
501             MeasurementType.PeakGroundVelocity or
502             MeasurementType.Response0125 or
503             MeasurementType.Response0250 or
504             MeasurementType.Response0500 or
505             MeasurementType.Response1000 or
506             MeasurementType.Response2000 or
507             MeasurementType.Response4000
508                 => x.HeightToPgv(),
509             MeasurementType.PeakGroundDisplacement
510                 => x.HeightToPgd(),
511             -           => throw new UnreachableException(),
512         })
513         .Select(x => x.ToSignificantFigures(2))];
515
516     /// <summary>
517     /// The data string to display for the current kmoni data choice.
518     /// </summary>
519     public string DataLegendText
520         => _kmoniOptions.MeasurementChoice switch
521         {
522             MeasurementType.MeasuredIntensity
523                 => RealtimePageResources.KmoniLegendTextIntensity,
524             MeasurementType.PeakGroundAcceleration
525                 => RealtimePageResources.KmoniLegendTextPga,
526             MeasurementType.PeakGroundVelocity or
527             MeasurementType.Response0125 or
528             MeasurementType.Response0250 or
529             MeasurementType.Response0500 or
530             MeasurementType.Response1000 or
531             MeasurementType.Response2000 or
532             MeasurementType.Response4000
533                 => RealtimePageResources.KmoniLegendTextPgv,
534             MeasurementType.PeakGroundDisplacement
535                 => RealtimePageResources.KmoniLegendTextPgd,
536             -           => throw new UnreachableException(),
537         };
539
540     /// <summary>
541     /// The options for the kmoni layer.
542     /// </summary>
543     private readonly IKmoniSettingsHelper _kmoniOptions;
544     /// <summary>
545     /// The time display.
546     /// </summary>
547     public DateTimeOffset TimeDisplay
548         => _timeProvider.Now();
549     /// <summary>
550     /// The image fetcher to be used.
551     /// </summary>
```

```
552     private readonly IImageFetch _imageFetch;
553     /// <summary>
554     /// The point extractor to be used.
555     /// </summary>
556     private readonly IPointExtract _pointExtract;
557     /// <summary>
558     /// The delay for the kmoni layer when fetching data.
559     /// </summary>
560     private readonly TimeSpan _kmoniDelay = new(0, 0, 1);
561     /// <summary>
562     /// The kmoni layer.
563     /// </summary>
564     private readonly MemoryLayer _kmoniLayer = new() { Style = null };
565
566     // Adapted from https://mapsui.com/samples/- Info - Custom Callout
567     /// <summary>
568     /// Handles the timed event, by changing the displaying time, and refreshing the
569     /// → Kmoni layer.
570     /// </summary>
571     /// <param name="source">The source of the event.</param>
572     /// <param name="e">The event arguments.</param>
573     private async void RefreshKmoniDataEventHandler(object? source, ElapsedEventArgs
574     → e)
575     {
576         IEnumerable<IFeature>? kmoniObservationPoints = await
577         → GetKmoniObservationPoints();
578         if (kmoniObservationPoints is null)
579         {
580             return;
581         }
582
583         _kmoniLayer.Features = kmoniObservationPoints;
584         _kmoniLayer.DataHasChanged();
585     }
586
587     /// <summary>
588     /// Gets the observation points from Kmoni, and convert them to features.
589     /// </summary>
590     /// <returns>The list of features, or <see langword="null"/> if it was
591     /// → unsuccessful.</returns>
592     private async Task<IEnumerable<IFeature>?> GetKmoniObservationPoints()
593     {
594         try
595         {
596             byte[]? imageBytes = await _imageFetch
597             .GetByteArrayAsync(
598                 _kmoniOptions.MeasurementChoice,
599                 _kmoniOptions.SensorChoice,
600                 _timeProvider
601                     .ConvertToJst(_timeProvider.Now())
602                     .Subtract(_kmoniDelay)
603                     .DateTime);
604
605             return imageBytes is null
606                 ? null
607                 : _pointExtract
608                     .ExtractColours(imageBytes.ToBitmap(), _kmoniOptions.SensorChoice
609                         → is SensorType.Borehole)
610                         .Select(pc => pc.ToStationFeature());
611         }
612     }
```

```

606     }
607     catch (ArgumentException) { }
608
609     return null;
610 }
611 #endregion
612 }
```

Listing A.10.7: EasonEetwViewer/ViewModels/RealTimePageViewModel.cs

```

1  using System.Collections.ObjectModel;
2  using System.Diagnostics;
3  using CommunityToolkit.Mvvm.ComponentModel;
4  using CommunityToolkit.Mvvm.Input;
5  using DynamicData;
6  using EasonEetwViewer.Dmdata.Api.Abstractions;
7  using EasonEetwViewer.Dmdata.Api.Dtos.Record.EarthquakeParameter;
8  using EasonEetwViewer.Dmdata.Api.Dtos.Record.GdEarthquake;
9  using EasonEetwViewer.Dmdata.Api.Dtos.Response;
10 using EasonEetwViewer.Dmdata.Authentication.Abstractions;
11 using EasonEetwViewer.Dmdata.Authentication.Events;
12 using EasonEetwViewer.Dmdata.DmdataComponent;
13 using EasonEetwViewer.Dmdata.Dtos.Enum;
14 using EasonEetwViewer.Dmdata.Telegram.Abstractions;
15 using EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
16 using EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
17 using EasonEetwViewer.Extensions;
18 using EasonEetwViewer.Models.PastPage;
19 using EasonEetwViewer.Services;
20 using EasonEetwViewer.Services.TimeProvider;
21 using EasonEetwViewer.ViewModels.ViewModelBases;
22 using Mapsui;
23 using Mapsui.Extensions;
24 using Mapsui.Layers;
25 using Mapsui.Projections;
26 using Microsoft.Extensions.Logging;
27
28 namespace EasonEetwViewer.ViewModels;
29 /// <summary>
30 /// The view model for the past page.
31 /// </summary>
32 internal partial class PastPageViewModel : MapViewModelBase
33 {
34     /// <summary>
35     /// Initializes a new instance of the <see cref="PastPageViewModel"/> class.
36     /// </summary>
37     /// <param name="resources">The map resources to be used.</param>
38     /// <param name="authenticatorWrapper">The authenticator to be used.</param>
39     /// <param name="apiCaller">The API caller to be used.</param>
40     /// <param name="telegramRetriever">The telegram retriever to be used.</param>
41     /// <param name="telegramParser">The telegram parser to be used.</param>
42     /// <param name="timeProvider">The time provider to be used.</param>
43     /// <param name="logger">The logger to be used.</param>
44     public PastPageViewModel(
45         MapResourcesProvider resources,
46         IAuthenticationHelper authenticatorWrapper,
47         IApiCaller apiCaller,
48         ITelegramRetriever telegramRetriever,
```

```
49     ITelegramParser telegramParser,
50     ITimeProvider timeProvider,
51     ILogger<PastPageViewModel> logger) : base(
52         resources,
53         authenticatorWrapper,
54         apiCaller,
55         telegramRetriever,
56         telegramParser,
57         timeProvider,
58         logger)
59     {
60         _logger = logger;
61
62         if (authenticatorWrapper.AuthenticationStatus is not
63             AuthenticationStatus.Null)
64         {
65             Task.Run(LoadEarthquakeObservationStationsAsync).Wait();
66         }
67
68         authenticatorWrapper.StatusChanged +=
69             AuthenticatorWrapperStatusChangedEventHandler;
70
71         _hypocentreLayer = new()
72         {
73             Style = _resources.HypocentreShapeStyle
74         };
75         _regionLayer = new()
76         {
77             DataSource = _resources.Region,
78             Style = null
79         };
80
81         Map.Layers.Add(_hypocentreLayer, _hypocentreGroup);
82         Map.Layers.Add(_stationLayer, _stationGroup);
83         Map.Layers.Add(_regionLayer, _regionGroup);
84     }
85
86     /// <summary>
87     /// The logger to be used.
88     /// </summary>
89     private readonly ILogger<PastPageViewModel> _logger;
90
91     #region earthquakeDetails
92     /// <summary>
93     /// The earthquake details for the currently selected earthquake, displayed on the
94     /// left.
95     /// </summary>
96     [ObservableProperty]
97     private EarthquakeDetailsTemplate? _earthquakeDetails;
98
99     /// <summary>
100    /// The factor to extend the view by when zooming to the earthquake.
101    /// </summary>
102    private const double _extendFactor = 1.2;
103
104    /// <summary>
105    /// The layer group for the hypocentre, on the top.
106    /// </summary>
107    private const int _hypocentreGroup = 2;
```

```

105     ///<summary>
106     /// The layer group for the observation stations, in the middle.
107     ///</summary>
108     private const int _stationGroup = 1;
109     ///<summary>
110     /// The layer group for the regions, on the bottom.
111     ///</summary>
112     private const int _regionGroup = 0;
113
114     ///<summary>
115     /// The layer for the hypocentre.
116     ///</summary>
117     private readonly MemoryLayer _hypocentreLayer;
118     ///<summary>
119     /// The layer for the observation station.
120     ///</summary>
121     private readonly MemoryLayer _stationLayer = new()
122     {
123         Style = null,
124     };
125     ///<summary>
126     /// The layer for the regions.
127     ///</summary>
128     private readonly Layer _regionLayer;
129
130     ///<summary>
131     /// The cancellation token source for setting an earthquake detail.
132     ///</summary>
133     private CancellationTokenSource _cts = new();
134
135     ///<summary>
136     /// Selects the best telegram to display the earthquake information.
137     ///</summary>
138     ///<param name="telegrams">The list of earthquake telegrams available.</param>
139     ///<returns>The selected eearthquake telegram.</returns>
140     private TelegramItem? FindEarthquakeTelegram(IEnumerable<TelegramItem> telegrams)
141     => telegrams
142         .Where(x
143             => _telegramParser.ParseSchemaInformation(x.SchemaVersion) ==
144                 typeof(EarthquakeInformationSchema))
145         .Where(x
146             => x.TelegramHead.Type is "VXSE51" or "VXSE52" or "VXSE53")
147         .GroupBy(x
148             => x.TelegramHead.Type)
149         .Select(x
150             => x.OrderByDescending(y
151                 => y.Serial)
152                 .First())
153             .OrderByDescending(x
154                 => x.TelegramHead.Type)
155                 .FirstOrDefault();
156
157     ///<summary>
158     /// Handles the selected earthquake changed.
159     ///</summary>
160     ///<param name="value">The new selected earthquake.</param>
161     async partial void OnSelectedEarthquakeChanged(EarthquakeItemTemplate? value)
162     {
163         CancellationToken token = ResetDetails();

```

```

163
164     if (value is null)
165     {
166         return;
167     }
168
169     GdEarthquakeEvent? rsp = await
170     → _apiCaller.GetPathEarthquakeEventAsync(value.EventId);
171     IEnumerable<TelegramItem> telegrams = rsp?.EarthquakeEvent.Telegrams ?? [];
172     TelegramItem? telegram = FindEarthquakeTelegram(telegrams);
173     EarthquakeInformationSchema? telegramInfo = telegram is not null
174     ? await _telegramRetriever.GetJsonTelegramAsync(telegram.Id) as
175     → EarthquakeInformationSchema
176     : null;
177
178     IEnumerable<DetailIntensityTemplate>? tree = null;
179     MRect? regionLimits = null;
180     if (telegramInfo?.Body.Intensity is not null)
181     {
182         await LoadEarthquakeObservationStationsAsync();
183         (regionLimits, tree) =
184         → DealWithTelegramIntensities(telegramInfo.Body.Intensity, token);
185     }
186
187     // Mark Hypocentre
188     MRect? hypocentreLimits = null;
189     CoordinateComponent? coordinate =
190     → telegramInfo?.Body.Earthquake?.Hypocentre.Coordinates ??
191     → value.Hypocentre?.Coordinates;
192     if (coordinate?.Longitude is not null && coordinate?.Latitude is not null)
193     {
194         hypocentreLimits = DealWithHypocentre(coordinate.Longitude,
195         → coordinate.Latitude, token);
196     }
197
198     // Zoom to Box
199     MRect limits = regionLimits ?? _mainLimitsOfJapan;
200     limits = limits.Join(hypocentreLimits);
201     if (regionLimits is not null || hypocentreLimits is not null)
202     {
203         limits = limits.Multiply(_extendFactor);
204     }
205
206     if (!token.IsCancellationRequested)
207     {
208         EarthquakeDetails = new(value, telegramInfo, tree);
209         Map.Navigator.ZoomToBox(limits);
210     }
211
212     /// <summary>
213     /// Deals with the hypocentre of the earthquake.
214     /// </summary>
215     /// <param name="longitude">The longitude of the hypocentre.</param>
216     /// <param name="latitude">The latitude of the hypocentre.</param>
217     /// <param name="token">The token with whether to cancel the operation.</param>
218     /// <returns>The limits of the earthquakes in <see cref="MRect"/>.</returns>
219     private MRect DealWithHypocentre(
220         Coordinate longitude,

```

```

216     Coordinate latitude,
217     CancellationToken token)
218     {
219         MPoint convertedCoordinates = SphericalMercator
220             .FromLonLat(
221                 longitude.DoubleValue,
222                 latitude.DoubleValue)
223             .ToMPoint();
224         MRect limits = convertedCoordinates.MRect;
225
226         if (!token.IsCancellationRequested)
227         {
228             _hypocentreLayer.Features = [new PointFeature(convertedCoordinates)];
229             _hypocentreLayer.DataHasChanged();
230         }
231
232         return limits;
233     }
234
235     /// <summary>
236     /// Deals with the intensities of the earthquake in the telegram.
237     /// </summary>
238     /// <param name="intensityDetails">The details of the intensities in the
239     → telegram.</param>
240     /// <param name="token">The token with whether to cancel the operation.</param>
241     /// <returns>A tuple of: the limits of the regions, and the intensity
242     → tree.</returns>
243     private (
244         MRect regionLimits,
245         IEnumerable<DetailIntensityTemplate> intensityTree)
246             → DealWithTelegramIntensities(
247                 IntensityDetails intensityDetails,
248                 CancellationToken token)
249             {
250                 // Join the station data with the stations retrieved from the API
251                 IEnumerable<(Station station, Intensity intensity)> stationData =
252                     → intensityDetails
253                         .Stations
254                         .Where(x
255                             => x.MaxInt?.ToIntensity() is Intensity)
256                         .Join(
257                             _earthquakeObservationStations!,
258                             si => si.Code,
259                             s => s.XmlCode,
260                             (si, s)
261                             => (s, (Intensity)si.MaxInt?.ToIntensity()!));
262
263         if (!token.IsCancellationRequested)
264         {
265             _regionLayer.Style = intensityDetails.Regions.ToRegionStyle();
266             _regionLayer.DataHasChanged();
267             _stationLayer.Features = stationData.Select(x => x.ToStationFeature());
268             _stationLayer.DataHasChanged();
269         }
270
271         return (
272             CreateRegionLimits(intensityDetails),
273             CreateIntensityTree(stationData));
274     }

```

```

271     ///<summary>
272     ///Creates the limits of the regions from the intensity details.
273     ///</summary>
274     ///<param name="intensityDetails">The intensity details from a tekegram.</param>
275     ///<returns>The intensity details.</returns>
276     private MRect CreateRegionLimits(IntensityDetails intensityDetails)
277         => _regionFeatures
278             .Join(intensityDetails.Regions,
279                 f => (string)f["code"]!,
280                 r => r.Code,
281                 (f, r) => f.Extent)
282             .NotNull()
283             .Aggregate((region1, region2) => region1.Join(region2));
284
285     ///<summary>
286     ///Creates a tree of intensities from the station data.
287     ///</summary>
288     ///<param name="stationData">The data of the stations.</param>
289     ///<returns>The list of detail intensity template, which represents the tree, in
290     ///descending intensity.</returns>
291     private IEnumerable<DetailIntensityTemplate>
292         => CreateIntensityTree(IEnumerable<(Station station, Intensity intensity)>
293             => stationData
294             => stationData
295                 .Join(_resources.Prefectures,
296                     x
297                         => x.station.City.Code[0..2],
298                         prefecture
299                             => prefecture.Code,
300                             (si, prefecture)
301                             => (prefecture, si.station, si.intensity))
302             .GroupBy(x => x.intensity)
303             .Select(x => new DetailIntensityTemplate()
304             {
305                 Intensity = x.Key,
306                 PositionNodes = x
307                     .GroupBy(y => y.prefecture)
308                     .Select(y => new DisplayNode()
309                     {
310                         Name = y.Key.Name,
311                         SubNodes = y
312                             .GroupBy(z => z.station.Region)
313                             .Select(z => new DisplayNode()
314                             {
315                                 Name = z.Key.KanjiName,
316                                 SubNodes = z
317                                     .GroupBy(a => a.station.City)
318                                     .Select(a => new DisplayNode()
319                                     {
320                                         Name = a.Key.KanjiName,
321                                         SubNodes = a
322                                             .Select(a => new DisplayNode()
323                                             {
324                                                 Name = a.station.KanjiName
325                                             })
326                                         })
327                                     })
328                                 })
329                             })
330                         })
331                     })
332                 })
333             })
334         )
335     }

```

```
327     .OrderByDescending(x => x.Intensity);
328     /// <summary>
329     /// Resets the details of the earthquake.
330     /// </summary>
331     /// <returns>A cancellation token for setting the next earthquake.</returns>
332     private CancellationToken ResetDetails()
333     {
334         _cts.Cancel();
335         _cts.Dispose();
336
337         _hypocentreLayer.Features = [];
338         _hypocentreLayer.DataHasChanged();
339         _stationLayer.Features = [];
340         _stationLayer.DataHasChanged();
341         _regionLayer.Style = null;
342         _regionLayer.DataHasChanged();
343
344         Map.Navigator.ZoomToBox(_mainLimitsOfJapan);
345         EarthquakeDetails = null;
346
347         _cts = new CancellationTokenSource();
348         CancellationToken token = _cts.Token;
349         return token;
350     }
351
352     /// <summary>
353     /// The address for the Yahoo webpage for earthquakes.
354     /// </summary>
355     private const string _yahooWebpageAddress =
356         "https://typhoon.yahoo.co.jp/weather/jp/earthquake/{0}.html";
357
358     /// <summary>
359     /// Jumps to the Yahoo webpage for the currently selected earthquake.
360     /// </summary>
361     [RelayCommand]
362     private void JumpYahooWebpage()
363     {
364         if (EarthquakeDetails is null)
365         {
366             return;
367         }
368
369         _ = Process.Start(new ProcessStartInfo //
370             "https://stackoverflow.com/a/61035650/"
371             {
372                 FileName = string.Format(
373                     _yahooWebpageAddress,
374                     EarthquakeDetails.EventId),
375                 UseShellExecute = true
376             });
377     }
378     #endregion
379
380     #region earthquakeObservationStations
381     /// <summary>
382     /// The list of earthquake observation stations.
383     /// </summary>
384     private IEnumerable<Station>? _earthquakeObservationStations = null;
385     /// <summary>
```

```
384     /// Whether the earthquake observation stations have been retrieved.  
385     /// </summary>  
386     private bool IsStationsRetrieved  
387         => _earthquakeObservationStations is not null;  
388     /// <summary>  
389     /// Loads the earthquake observation stations.  
390     /// </summary>  
391     /// <returns>A <see cref="Task"/> object representing the asynchronous  
392     /// operation.</returns>  
393     private async Task LoadEarthquakeObservationStationsAsync()  
394     {  
395         if (IsStationsRetrieved)  
396         {  
397             return;  
398         }  
399         EarthquakeParameter? rsp = await _apiCaller.GetEarthquakeParameterAsync();  
400         _earthquakeObservationStations = rsp?.ItemList;  
401     }  
402     /// <summary>  
403     /// Event handler for the authenticator status changed event.  
404     /// </summary>  
405     /// <param name="sender">The sender of the event.</param>  
406     /// <param name="e">The event arguments.</param>  
407     private async void AuthenticatorWrapperStatusChangedEventHandler(object? sender,  
408         => AuthenticationStatusChangedEventArgs e)  
409         => await LoadEarthquakeObservationStationsAsync();  
410     #endregion  
411     #region loadEarthquakeAction  
412     /// <summary>  
413     /// The list of earthquakes on the sidebar.  
414     /// </summary>  
415     [ObservableProperty]  
416     private ObservableCollection<EarthquakeItemTemplate> _earthquakeList = [];  
417     /// <summary>  
418     /// The current selected earthquake.  
419     /// </summary>  
420     [ObservableProperty]  
421     private EarthquakeItemTemplate? _selectedEarthquake;  
422     /// <summary>  
423     /// The cursor token for the earthquake list.  
424     /// </summary>  
425     [ObservableProperty]  
426     [NotifyPropertyChangedFor(nameof(IsLoadExtraEnabled))]  
427     private string? _cursorToken = null;  
428  
429     /// <summary>  
430     /// Whether the load extra button is enabled.  
431     /// </summary>  
432     public bool IsLoadExtraEnabled  
433         => CursorToken is not null;  
434  
435     /// <summary>  
436     /// Refreshes the earthquake list, and clears existing earthquakes and the  
437     /// selected earthquake.  
438     /// </summary>  
439     /// <returns>A <see cref="Task"/> object representing the asynchronous  
440     /// operation.</returns>
```

```

439     [RelayCommand]
440     private async Task RefreshEarthquakeList()
441     {
442         GdEarthquakeList? rsp = await _apiCaller.GetPastEarthquakeListAsync(limit:
443             ↪ 50);
444         IEnumerable<EarthquakeInfo> eqList = rsp?.ItemList ?? [];
445
446         CursorToken = rsp?.NextToken ?? null;
447         SelectedEarthquake = null;
448         EarthquakeList.Clear();
449         EarthquakeList.AddRange(eqList.Select(x
450             => new EarthquakeItemTemplate(x)));
451     }
452
453     /// <summary>
454     /// Loads 10 extra earthquakes to the list using the cursor token.
455     /// </summary>
456     /// <returns>A <see cref="Task"/> object representing the asynchronous
457     ↪ operation.</returns>
458     [RelayCommand]
459     private async Task LoadExtraEarthquakes()
460     {
461         if (!IsLoadExtraEnabled)
462         {
463             return;
464         }
465
466         GdEarthquakeList? rsp = await _apiCaller.GetPastEarthquakeListAsync(limit:
467             ↪ 10, cursorToken: CursorToken);
468         IEnumerable<EarthquakeInfo> eqList = rsp?.ItemList ?? [];
469         CursorToken = rsp?.NextToken;
470         EarthquakeList.AddRange(eqList.Select(x
471             => new EarthquakeItemTemplate(x)));
472     }
473     #endregion
474 }
```

Listing A.10.8: EasonEetwViewer/ViewModels/PastPageViewModel.cs

```

1  using System.Collections.ObjectModel;
2  using System.Globalization;
3  using CommunityToolkit.Mvvm.ComponentModel;
4  using CommunityToolkit.Mvvm.Input;
5  using DynamicData;
6  using EasonEetwViewer.Dmdata.Api.Abstractions;
7  using EasonEetwViewer.Dmdata.Api.Dtos.Enum.WebSocket;
8  using EasonEetwViewer.Dmdata.Api.Dtos.Record.Contract;
9  using EasonEetwViewer.Dmdata.Api.Dtos.Record.WebSocket;
10 using EasonEetwViewer.Dmdata.Api.Dtos.Request;
11 using EasonEetwViewer.Dmdata.Api.Dtos.Response;
12 using EasonEetwViewer.Dmdata.Authentication.Abstractions;
13 using EasonEetwViewer.Dmdata.Authentication.Events;
14 using EasonEetwViewer.Dmdata.Telegram.Abstractions;
15 using EasonEetwViewer.Dmdata.WebSocket.Abstractions;
16 using EasonEetwViewer.KyoshinMonitor.Abstractions;
17 using EasonEetwViewer.Lang;
18 using EasonEetwViewer.Models.SettingPage;
19 using EasonEetwViewer.Services.Kmoni.Abstractions;
```

```
20  using EasonEetwViewer.Services.TimeProvider;
21  using EasonEetwViewer.ViewModels.ViewModelBases;
22  using Microsoft.Extensions.Logging;
23  using Microsoft.Extensions.Options;
24
25  namespace EasonEetwViewer.ViewModels;
26
27  /// <summary>
28  /// The view model for the settings page.
29  /// </summary>
30  internal sealed partial class SettingPageViewModel : PageViewModelBase
31  {
32      /// <summary>
33      /// Instantiates a new <see cref="SettingPageViewModel"/>.
34      /// </summary>
35      /// <param name="startPost">The post data when the WebSocket starts.</param>
36      /// <param name="kmoniOptions">The options helper for kmoni.</param>
37      /// <param name="webSocketClient">The WebSocket client to be used.</param>
38      /// <param name="logger">The logger to be used.</param>
39      /// <param name="authenticator">The authenticator to be used.</param>
40      /// <param name="apiCaller">The API caller to be used.</param>
41      /// <param name="telegramRetriever">The telegram retriever to be used.</param>
42      /// <param name="telegramParser">The telegram parser to be used.</param>
43      /// <param name="timeProvider">The time provider to be used.</param>
44      /// <param name="languagePath">The path to the language file.</param>
45  public SettingPageViewModel(
46      IOptions<WebSocketStartPost> startPost,
47      IKmoniSettingsHelper kmoniOptions,
48      IWebSocketClient webSocketClient,
49      IAuthenticationHelper authenticator,
50      IApiCaller apiCaller,
51      ITelegramRetriever telegramRetriever,
52      ITelegramParser telegramParser,
53      ITimeProvider timeProvider,
54      ILogger<SettingPageViewModel> logger,
55      string languagePath)
56      : base(
57          authenticator,
58          apiCaller,
59          telegramRetriever,
60          telegramParser,
61          timeProvider,
62          logger)
63  {
64      _webSocketClient = webSocketClient;
65      _startPost = startPost.Value;
66      _logger = logger;
67      KmoniSettingsHelper = kmoniOptions;
68      _authenticator.StatusChanged += AuthenticationStatusChangedEventHandler;
69      _webSocketClient.StatusChanged += WebSocketStatusChangedEventHandler;
70
71      _languageFilePath = languagePath;
72      LanguageChanged += (o, e) => SetResourcesCultureInfo(LanguageChoice);
73      LanguageChanged += LanguageChangedWriteToFileEventHandler;
74      LanguageChoice = GetCultureInfo();
75  }
76  /// <summary>
77  /// The logger to be used.
78 }
```

```
79     /// </summary>
80     private readonly ILogger<SettingPageViewModel> _logger;
81     #region languageSettings
82     /// <summary>
83     /// Invoked when the language choice is changed.
84     /// </summary>
85     public event EventHandler<EventArgs>? LanguageChanged;
86     /// <summary>
87     /// The current language.
88     /// </summary>
89     [ObservableProperty]
90     private CultureInfo _languageChoice = MainWindowResources.Culture;
91     /// <summary>
92     /// Executes when the language choice is changed.
93     /// </summary>
94     /// <param name="value">The new language.</param>
95     partial void OnLanguageChoiceChanged(CultureInfo value)
96         => LanguageChanged?.Invoke(this, new());
97     /// <summary>
98     /// Sets the resources culture info to the language specified.
99     /// </summary>
100    /// <param name="cultureInfo">The new language.</param>
101    private static void SetResourcesCultureInfo(CultureInfo cultureInfo)
102    {
103        MainWindowResources.Culture = cultureInfo;
104        EarthquakeResources.Culture = cultureInfo;
105        PastPageResources.Culture = cultureInfo;
106        SettingPageResources.Culture = cultureInfo;
107        RealtimePageResources.Culture = cultureInfo;
108    }
109    /// <summary>
110    /// Gets the language from the file.
111    /// </summary>
112    /// <returns>The language obtained, or invariant culture if
113    /// → unsuccessful.</returns>
114    private CultureInfo GetCultureInfo()
115    {
116        CultureInfo culture;
117        try
118        {
119            culture = new CultureInfo(File.ReadAllText(_languageFilePath));
120            _logger.ReadLanguage(_languageFilePath, culture);
121        }
122        catch
123        {
124            culture = CultureInfo.InvariantCulture;
125            _loggerFailedToReadLanguage(_languageFilePath);
126        }
127        return culture;
128    }
129    /// <summary>
130    /// Event handler for language changed by writing the language to the file.
131    /// </summary>
132    /// <param name="sender">The sender of the event.</param>
133    /// <param name="e">The event arguments.</param>
134    private void LanguageChangedWriteToFileEventHandler(object? sender, EventArgs e)
135    {
136        try
```

```
137     {
138         File.WriteAllText(_languageFilePath, LanguageChoice.Name);
139         _logger.WriteLine(_languageFilePath, LanguageChoice);
140     }
141     catch
142     {
143         _loggerFailedToWriteLanguage(_languageFilePath);
144     }
145 }
146 /// <summary>
147 /// The path to the file where the language is stored.
148 /// </summary>
149 private readonly string _languageFilePath;
150 /// <summary>
151 /// The list of languages available.
152 /// </summary>
153 public IEnumerable<CultureInfo> LanguageChoices { get; init; } = [
154     CultureInfo.InvariantCulture,
155     new CultureInfo("ja-JP"),
156     new CultureInfo("zh-CN")];
157 #endregion
158
159 #region kmoniSettings
160 /// <summary>
161 /// The helper for kmoni options.
162 /// </summary>
163 public IKmoniSettingsHelper KmoniSettingsHelper { get; init; }
164 /// <summary>
165 /// The choices for the sensor type.
166 /// </summary>
167 public IEnumerable<SensorType> SensorTypeChoices { get; init; }
168     = Enum.GetValues<SensorType>();
169 /// <summary>
170 /// The choices for the measurement type.
171 /// </summary>
172 public IEnumerable<MeasurementType> MeasurementTypeChoices { get; init; }
173     = Enum.GetValues<MeasurementType>();
174 #endregion
175
176 #region webSocketSettings
177 /// <summary>
178 /// The WebSocket client to be used.
179 /// </summary>
180 private readonly IWebSocketClient _webSocketClient;
181 /// <summary>
182 /// The data to include in the POST request when starting the WebSocket.
183 /// </summary>
184 private readonly WebSocketStartPost _startPost;
185 /// <summary>
186 /// The text on the WebSocket connect/disconnect button.
187 /// </summary>
188 public string WebSocketButtonText
189     => _webSocketClient.IsWebSocketConnected
190         ? SettingPageResources.WebSocketButtonTextConnected
191         : SettingPageResources.WebSocketButtonTextDisconnected;
192 /// <summary>
193 /// Handles the WebSocket status changed event by notifying the relevant
194     → properties.
195 /// </summary>
```

```
195     /// <param name="sender">The sender of the event.</param>
196     /// <param name="e">The arguments of the event.</param>
197     private void WebSocketStatusChangedEventHandler(object? sender, EventArgs e)
198         => OnPropertyChanged(nameof(WebSocketButtonText));
199     /// <summary>
200     /// The command to execute when the connect/disconnect button for WebSocket is
201     /// pressed.
202     /// </summary>
203     /// <returns>A <see cref="Task"/> object representing the asynchronous
204     /// operation.</returns>
205     [RelayCommand]
206     private async Task WebSocketButton()
207     {
208         if (!_webSocketClient.IsWebSocketConnected)
209         {
210             _logger.ConnectingWebSocket();
211             WebSocketStart? webSocket = await
212                 _apiCaller.PostWebSocketStartAsync(_startPost);
213             if (webSocket is not null)
214             {
215                 await _webSocketClient.ConnectAsync(webSocket.WebSockerUrl.Url);
216                 _logger.ConnectedWebSocket();
217             }
218         }
219         else
220         {
221             _logger.DisconnectingWebSocket();
222             await _webSocketClient.DisconnectAsync();
223         }
224     }
225     /// <summary>
226     /// The list of WebSocket connections.
227     /// </summary>
228     public ObservableCollection<IWebSocketConnectionTemplate> WebSocketConnections {
229         get; init; }
230         = [];
231     /// <summary>
232     /// Gets the number of available websocket connections.
233     /// </summary>
234     /// <returns>The number of connections available.</returns>
235     private async Task<int> GetTotalWebSocketSlots()
236     {
237         _logger.RequestingAvailableConnections();
238         ContractList? contractList = await _apiCaller.GetContractListAsync();
239         IEnumerable<Contract> contracts = contractList?.ItemList ?? [];
240         int slotNumber = contracts.Sum(contract
241             => contract.IsValid
242                 ? contract.ConnectionCounts
243                     : 0);
244         _logger.RequestedAvailableConnections(slotNumber);
245         return slotNumber;
246     }
247     /// <summary>
248     /// The action to be executed when refreshing the WebSocket list.
249     /// </summary>
250     /// <returns>A <see cref="Task"/> object representing the asynchronous
251     /// operation.</returns>
252     [RelayCommand]
253     private async Task WebSocketRefresh()
```

```
249     {
250         _logger.RequestingAvailableConnections();
251         IList<WebSocketDetails> wsList = [];
252         string? currentCursorToken = null;
253         do
254         {
255             WebSocketList? webSocketList = await _apiCaller.GetWebSocketListAsync(
256                 limit: 100,
257                 connectionStatus: ConnectionStatus.Open,
258                 cursorToken: currentCursorToken);
259             wsList.AddRange(webSocketList?.ItemList ?? []);
260
261             currentCursorToken = webSocketList?.NextToken;
262         } while (currentCursorToken is not null);
263
264         _logger.DisplayingActiveConnections();
265         WebSocketConnections.Clear();
266         WebSocketConnections.AddRange(wsList.Select(x
267             => new WebSocketConnectionTemplate(x, async ()
268                 => await _apiCaller.DeleteWebSocketAsync(x.WebSocketId))));
```

269

```
270         int availableConnection = await GetTotalWebSocketSlots();
271         _logger.DisplayingAvailableConnections();
272         WebSocketConnections.AddRange(Enumerable.Repeat(
273             EmptyWebSocketConnectionTemplate.Instance,
274             availableConnection - WebSocketConnections.Count));
275     }
276 #endregion
```

277

```
278 #region authSettings
279 /// <summary>
280 /// The OAuth Button text.
281 /// </summary>
282 public string OAuthButtonText
283     => OAuthConnected
284         ? SettingPageResources.OAuthButtonTextConnected
285         : SettingPageResources.OAuthButtonTextDisconnected;
286 /// <summary>
287 /// Whether OAuth is connected.
288 /// </summary>
289 public bool OAuthConnected
290     => AuthenticationStatus is AuthenticationStatus.OAuth;
291 /// <summary>
292 /// Whether the API key is confirmed.
293 /// </summary>
294 public bool ApiKeyConfirmed
295     => AuthenticationStatus is AuthenticationStatus.ApiKey;
296 /// <summary>
297 /// Whether the API key button is enabled.
298 /// </summary>
299 public bool ApiKeyButtonEnabled
300     => AuthenticationStatus is AuthenticationStatus.Null &&
301         → ApiKeyText.StartsWith("AKe.");
302 /// <summary>
303 /// The API Key Text input.
304 /// </summary>
305 [ObservableProperty]
306 [NotifyPropertyChangedFor(nameof(ApiKeyButtonEnabled))]
307 private string _apiKeyText = string.Empty;
```

```
307     ///<summary>
308     ///<summary>The OAuth Button click option.
309     ///</summary>
310     ///<returns>A <see cref="Task"/> object that represents the asynchronous
311     ///<summary>A <see cref="Task"/> object that represents the asynchronous
312     ///<summary>A <see cref="Task"/> object that represents the asynchronous
313     [RelayCommand]
314     private async Task OAuthButton()
315     {
316         if (AuthenticationStatus is AuthenticationStatus.OAuth)
317         {
318             _logger.UnsettingOAuth();
319             await _authenticator.UnsetAuthenticatorAsync();
320         }
321         else
322         {
323             _logger.SettingOAuth();
324             await _authenticator.SetOAuthAsync();
325             _logger.OAuthSet();
326         }
327     }
328     ///<summary>
329     ///<summary>Handles the authentication status changed event by notifying the relevant
330     ///<summary>Handles the authentication status changed event by notifying the relevant
331     ///<summary>Handles the authentication status changed event by notifying the relevant
332     ///<summary>Handles the authentication status changed event by notifying the relevant
333     ///<summary>The sender of the event.</param>
334     ///<summary>The arguments of the event.</param>
335     private void AuthenticationStatusChangedEventHandler(object? sender,
336     ///<summary>The sender of the event.</param>
337     ///<summary>The arguments of the event.</param>
338     {
339         OnPropertyChanged(nameof(AuthenticationStatus));
340         OnPropertyChanged(nameof(OAuthConnected));
341         OnPropertyChanged(nameof(OAuthButtonText));
342         OnPropertyChanged(nameof(ApiKeyConfirmed));
343         OnPropertyChanged(nameof(ApiKeyButtonEnabled));
344     }
345     ///<summary>
346     ///<summary>The API Key Button click option.
347     ///</summary>
348     ///<summary>A <see cref="Task"/> object that represents the asynchronous
349     ///<summary>A <see cref="Task"/> object that represents the asynchronous
350     ///<summary>A <see cref="Task"/> object that represents the asynchronous
351     [RelayCommand]
352     private async Task ApiKeyButton()
353     {
354         _logger.SettingApiKey();
355         await _authenticator.SetApiKeyAsync(ApiKeyText);
356     }
357     ///<summary>
358     ///<summary>Executes when the API Key text is changed, to unset the authenticator if it
359     ///<summary>Executes when the API Key text is changed, to unset the authenticator if it
360     ///<summary>Executes when the API Key text is changed, to unset the authenticator if it
```

```

361     }
362     /// <summary>
363     /// The current authentication status.
364     /// </summary>
365     public AuthenticationStatus AuthenticationStatus
366         => _authenticator.AuthenticationStatus;
367     #endregion
368 }
```

Listing A.10.9: EasonEetwViewer/ViewModels/SettingPageViewModel.cs

```

1  using CommunityToolkit.Mvvm.ComponentModel;
2  using Microsoft.Extensions.Logging;
3
4  namespace EasonEetwViewer.ViewModels.ViewModelBases;
5  /// <summary>
6  /// The base class for all view models.
7  /// </summary>
8  internal abstract class ViewModelBase : ObservableObject
{
9
10    /// <summary>
11    /// The logger to be used.
12    /// </summary>
13    private readonly ILogger<ViewModelBase> _logger;
14    /// <summary>
15    /// Creates a new instance of the <see cref="ViewModelBase"/> class.
16    /// </summary>
17    /// <param name="logger">The logger to be used.</param>
18    public ViewModelBase(ILogger<ViewModelBase> logger)
19    {
20        _logger = logger;
21        _logger.Instantiated();
22    }
23}
```

Listing A.10.10: EasonEetwViewer/ViewModels/ViewModelBases/ViewModelBase.cs

```

1  using CommunityToolkit.Mvvm.ComponentModel;
2  using EasonEetwViewer.Dmdata.Api.Abstractions;
3  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
4  using EasonEetwViewer.Dmdata.Telgram.Abstractions;
5  using EasonEetwViewer.Services;
6  using EasonEetwViewer.Services.TimeProvider;
7  using Mapsui;
8  using Mapsui.Limiting;
9  using Mapsui.Projections;
10 using Microsoft.Extensions.Logging;
11
12 namespace EasonEetwViewer.ViewModels.ViewModelBases;
13 /// <summary>
14 /// The base view model for all view models that use a map.
15 /// </summary>
16 internal abstract partial class MapViewModelBase : PageViewModelBase
{
17
18    /// <summary>
19    /// The map to be used in the view.
20    /// </summary>
```

```
21 [ObservableProperty]
22 private Map _map = new();
23
24 // Adapted from https://mapsui.com/samples/ - Navigation - Keep within Extent
25 /// <summary>
26 /// Creates a new instance of the <see cref="MapViewModelBase"/> class.
27 /// </summary>
28 /// <param name="resources">The map resources to be used.</param>
29 /// <param name="authenticatorWrapper">The authenticator to be used.</param>
30 /// <param name="apiCaller">The API caller to be used.</param>
31 /// <param name="telegramRetriever">The telegram retriever to be used.</param>
32 /// <param name="telegramParser">The telegram parser to be used.</param>
33 /// <param name="timeProvider">The time provider to be used.</param>
34 /// <param name="logger">The logger to be used.</param>
35 public MapViewModelBase(
36     MapResourcesProvider resources,
37     IAuthenticationHelper authenticatorWrapper,
38     IApiCaller apiCaller,
39     ITelegramRetriever telegramRetriever,
40     ITelegramParser telegramParser,
41     ITimeProvider timeProvider,
42     ILogger<MapViewModelBase> logger)
43 : base(
44     authenticatorWrapper,
45     apiCaller,
46     telegramRetriever,
47     telegramParser,
48     timeProvider,
49     logger)
50 {
51     _resources = resources;
52     _logger = logger;
53     _regionFeatures = _resources.Region.GetFeaturesAsync(new(new
54         → MSection(_limitsOfJapan, 1))).Result;
55     InitMapView();
56 }
57 /// <summary>
58 /// The features within the regions.
59 /// </summary>
60 protected readonly IEnumerable<IFeature> _regionFeatures;
61 /// <summary>
62 /// The logger to be used;
63 /// </summary>
64 private readonly ILogger<MapViewModelBase> _logger;
65 /// <summary>
66 /// The map resources to be used.
67 /// </summary>
68 protected readonly MapResourcesProvider _resources;
69
70 /// <summary>
71 /// Initialises the view of the map.
72 /// </summary>
73 private void InitMapView()
74 {
75     Map.Layers.Add(Mapsui.Tiling.OpenStreetMap.CreateTileLayer());
76     Map.Navigator.RotationLock = true;
77     Map.Navigator.Limiter = new ViewportLimiterKeepWithinExtent();
78     Map.Navigator.OverridePanBounds = _mapBounds;
    Map.Navigator.ZoomToBox(_mainLimitsOfJapan);
```

```

79         _logger.MapInitialised();
80     }
81
82     /// <summary>
83     /// Gives a <see cref="MRect"/> of the limits of Japan.
84     /// </summary>
85     protected readonly MRect _limitsOfJapan
86         = GetMRectFromLonLat(122, 20, 154, 46);
87     /// <summary>
88     /// Gives a <see cref="MRect"/> of the main limits of Japan.
89     /// </summary>
90     protected readonly MRect _mainLimitsOfJapan
91         = GetMRectFromLonLat(122, 27, 154, 46);
92     /// <summary>
93     /// Gives a <see cref="MRect"/> of the map bounds.
94     /// </summary>
95     private readonly MRect _mapBounds
96         = GetMRectFromLonLat(-180, -85, 180, 85);
97     /// <summary>
98     /// Creates a <see cref="MRect"/> object from pair of longitude and latitude.
99     /// </summary>
100    /// <param name="minLon">The minimal longitude.</param>
101    /// <param name="minLat">The minimal latitude.</param>
102    /// <param name="maxLon">The maximal longitude.</param>
103    /// <param name="maxLat">The maximal latitude.</param>
104    /// <returns>The <see cref="MRect"/> object created with the specific
105    /// → parameters.</returns>
106    private static MRect GetMRectFromLonLat(double minLon, double minLat, double
107    → maxLon, double maxLat)
108    {
109        (double minX, double minY) = SphericalMercator.FromLonLat(minLon, minLat);
110        (double maxX, double maxY) = SphericalMercator.FromLonLat(maxLon, maxLat);
111        return new MRect(minX, minY, maxX, maxY);
112    }
113 }
```

Listing A.10.11: EasonEetwViewer/ViewModels/ViewModelBases/MapViewModelBase.cs

```

1  using EasonEetwViewer.Dmdata.Api.Abstractions;
2  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
3  using EasonEetwViewer.Dmdata.Telemgram.Abstractions;
4  using EasonEetwViewer.Services.TimeProvider;
5  using Microsoft.Extensions.Logging;
6
7  namespace EasonEetwViewer.ViewModels.ViewModelBases;
8  /// <summary>
9  /// The base view model for all pages.
10  /// </summary>
11  /// <param name="authenticator">The authenticator to be used.</param>
12  /// <param name="apiCaller">The API caller to be used.</param>
13  /// <param name="telegramRetriever">The telegram retriever to be used.</param>
14  /// <param name="telegramParser">The telegram parser to be used.</param>
15  /// <param name="timeProvider">The time provider to be used.</param>
16  /// <param name="logger">The logger to be used.</param>
17  internal abstract partial class PageViewModelBase(
18      IAuthenticationHelper authenticator,
19      IApiCaller apiCaller,
20      ITelegramRetriever telegramRetriever,
```

```

21     ITelegramParser telegramParser,
22     ITimeProvider timeProvider,
23     ILogger<PageViewModelBase> logger) : ViewModelBase(logger)
24 {
25     /// <summary>
26     /// The authenticator to be used.
27     /// </summary>
28     protected readonly IAuthenticationHelper _authenticator = authenticator;
29     /// <summary>
30     /// The API caller to be used.
31     /// </summary>
32     protected readonly IApiCaller _apiCaller = apiCaller;
33     /// <summary>
34     /// The telegram retriever to be used.
35     /// </summary>
36     protected readonly ITelegramRetriever _telegramRetriever = telegramRetriever;
37     /// <summary>
38     /// The telegram parser to be used.
39     /// </summary>
40     protected readonly ITelegramParser _telegramParser = telegramParser;
41     /// <summary>
42     /// The time provider to be used.
43     /// </summary>
44     protected readonly ITimeProvider _timeProvider = timeProvider;
45 }

```

Listing A.10.12: EasonEetwViewer/ViewModels/ViewModelBases/PageViewModelBase.cs

```

1  using System.Globalization;
2  using Avalonia.Data.Converters;
3
4  namespace EasonEetwViewer.Converters;
5  internal class CultureInfoConverter : FuncValueConverter<CultureInfo, string>
6  {
7      public CultureInfoConverter() : base(value
8          => value is CultureInfo c
9              ? c.Equals(CultureInfo.InvariantCulture)
10                 ? "English"
11                 : c.Equals(new CultureInfo("ja-JP"))
12                     ? "日本語"
13                     : c.Equals(new CultureInfo("zh-CN"))
14                         ? "中文"
15                         : string.Empty
16                     : string.Empty)
17      { }
18  }

```

Listing A.10.13: EasonEetwViewer/Converters/CultureInfoConverter.cs

```

1  using System.Reflection;
2  using Avalonia.Data.Converters;
3  using EasonEetwViewer.Extensions;
4
5  namespace EasonEetwViewer.Converters;
6
7  internal class UnitValueDisplayValueConverter : FuncValueConverter<object, string>
8  {

```

```

9   public UnitValueDisplayValueConverter() : base(value
10    => (string)typeof(UnitValueExtensions)
11     .GetMethods(BindingFlags.Static | BindingFlags.Public)
12     .Where(method
13       => method.Name is nameof(UnitValueExtensions.ToValueString))
14     .Where(method
15       => method.GetParameters().Length is 1)
16     .SingleOrDefault(method
17       => method.GetParameters()[0].ParameterType == value?.GetType())?
18     .Invoke(null, [value])!)
19   { }
20 }
```

Listing A.10.14: EasonEetwViewer/Converters/UnitValueDisplayValueConverter.cs

```

1  using System.Reflection;
2  using Avalonia.Data.Converters;
3  using EasonEetwViewer.Extensions;
4
5  namespace EasonEetwViewer.Converters;
6
7  internal class EnumDisplayTextConverter : FuncValueConverter<Enum?, string>
8  {
9    public EnumDisplayTextConverter() : base(value
10     => (string)typeof(EnumDisplayTextExtensions)
11       .GetMethods(BindingFlags.Static | BindingFlags.Public)
12       .Where(method
13         => method.Name is nameof(EnumDisplayTextExtensions.ToString))
14       .Where(method
15         => method.GetParameters().Length is 1)
16       .SingleOrDefault(method
17         => method.GetParameters()[0].ParameterType == value?.GetType())?
18         .Invoke(null, [value])!)
19   { }
20 }
```

Listing A.10.15: EasonEetwViewer/Converters/EnumDisplayTextConverter.cs

```

1  using System.Reflection;
2  using Avalonia.Data.Converters;
3  using EasonEetwViewer.Extensions;
4
5  namespace EasonEetwViewer.Converters;
6
7  internal class UnitValueDisplayUnitConverter : FuncValueConverter<object, string>
8  {
9    public UnitValueDisplayUnitConverter() : base(value
10     => (string)typeof(UnitValueExtensions)
11       .GetMethods(BindingFlags.Static | BindingFlags.Public)
12       .Where(method
13         => method.Name is nameof(UnitValueExtensions.ToString))
14       .Where(method
15         => method.GetParameters().Length is 1)
16       .SingleOrDefault(method
17         => method.GetParameters()[0].ParameterType == value?.GetType())?
18         .Invoke(null, [value])!)
19   { }
20 }
```

Listing A.10.16: EasonEetwViewer/Converters/UnitValueDisplayUnitConverter.cs

```

1  using System.Reflection;
2  using Avalonia.Data.Converters;
3  using EasonEetwViewer.Extensions;
4
5  namespace EasonEetwViewer.Converters;
6
7  internal class EnumDisplayColourConverter : FuncValueConverter<Enum?, string>
8  {
9      public EnumDisplayColourConverter() : base(value
10         => (string)typeof(EnumDisplayColourExtensions)
11             .GetMethodsWith(BindingFlags.Static | BindingFlags.Public)
12             .Where(method
13                 => method.Name == nameof(EnumDisplayColourExtensions.ToColourString))
14             .Where(method
15                 => method.GetParameters().Length == 1)
16             .SingleOrDefault(method
17                 => method.GetParameters()[0].ParameterType == value?.GetType())?
18                 .Invoke(null, [value])!)
19     { }
20 }

```

Listing A.10.17: EasonEetwViewer/Converters/EnumDisplayColourConverter.cs

```

1  using Avalonia;
2  using Avalonia.Controls;
3  using Avalonia.Media;
4  using EasonEetwViewer.ViewModels.ViewModelBases;
5
6  namespace EasonEetwViewer.Models.MainWindow;
7  /// <summary>
8  /// Represents a sidebar item.
9  /// </summary>
10 internal record SidebarItemTemplate
11 {
12     /// <summary>
13     /// Creates a new instance of the <see cref="SidebarItemTemplate"/> record.
14     /// </summary>
15     /// <param name="instance">The instance for the view model of the page.</param>
16     /// <param name="iconKey">The key to the icon.</param>
17     /// <param name="displayLabel">The label to display the text for the
18     // sidebar.</param>
19     public SidebarItemTemplate(
20         PageViewModelBase instance,
21         string iconKey,
22         string displayLabel)
23     {
24         Label = displayLabel;
25         Model = instance;
26         ListItemIcon =
27             Application.Current!.TryFindResource(iconKey, out object? res)
28             ? res as StreamGeometry
29             : null;
30     }
31     /// <summary>

```

```

31     /// The label to display the text for the sidebar.
32     /// </summary>
33     public string Label { get; init; }
34     /// <summary>
35     /// The instance for the view model of the page.
36     /// </summary>
37     public PageViewModelBase Model { get; init; }
38     /// <summary>
39     /// The icon to display on the sidebar.
40     /// </summary>
41     public StreamGeometry? ListItemIcon { get; init; }
42 }
```

Listing A.10.18: EasonEetwViewer/Models/MainWindow/SidebarItemTemplate.cs

```

1  namespace EasonEetwViewer.Models.PastPage;
2  /// <summary>
3  /// Represents a node in the tree of positions.
4  /// </summary>
5  internal record DisplayNode
{
6
7      /// <summary>
8      /// The name of the node.
9      /// </summary>
10     public required string Name { get; init; }
11     /// <summary>
12     /// The subnodes of the node.
13     /// </summary>
14     public IEnumerable<DisplayNode> SubNodes { get; init; } = [];
15 }
```

Listing A.10.19: EasonEetwViewer/Models/PastPage/DisplayNode.cs

```

1  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
4  using EasonEetwViewer.Extensions;
5
6  namespace EasonEetwViewer.Models.PastPage;
7  /// <summary>
8  /// Describes the details of an earthquake.
9  /// </summary>
10    internal record EarthquakeDetailsTemplate
{
11
12    /// <summary>
13    /// Initializes a new instance of the <see cref="EarthquakeDetailsTemplate"/>
14    /// class.
15    /// </summary>
16    /// <param name="earthquakeItem">The <see cref="EarthquakeItemTemplate"/>, which
17    /// contains information for a fallback.</param>
18    /// <param name="telegram">The <see cref="EarthquakeInformationSchema"/>, which
19    /// contains most of the information.</param>
20    /// <param name="tree">A tree of intensities.</param>
21    public EarthquakeDetailsTemplate(
        EarthquakeItemTemplate earthquakeItem,
        EarthquakeInformationSchema? telegram,
```

```

21     IEnumerable<DetailIntensityTemplate>? tree)
22 {
23     EventId = telegram?.EventId ?? earthquakeItem.EventId;
24     Intensity = telegram?.Body.Intensity?.MaxInt ?? earthquakeItem.Intensity;
25     OriginTime = telegram?.Body.Earthquake?.OriginTime ??
26         → earthquakeItem.OriginTime;
27     Hypocentre = telegram?.Body.Earthquake?.Hypocentre ??
28         → earthquakeItem.Hypocentre;
29     Magnitude = telegram?.Body.Earthquake?.Magnitude ?? earthquakeItem.Magnitude;
30     LastUpdated = telegram?.ReportDateTime;
31     InformationalText = telegram?.ToInformationalString();
32     IntensityTree = tree ?? [];
33 }
34 /// <summary>
35 /// The Event ID of the earthquake.
36 /// </summary>
37 public string EventId { get; private init; }
38 /// <summary>
39 /// The maximum intensity observed for the earthquake.
40 /// </summary>
41 public Intensity? Intensity { get; private init; }
42 /// <summary>
43 /// The time at which the earthquake originated.
44 /// </summary>
45 public DateTimeOffset? OriginTime { get; private init; }
46 /// <summary>
47 /// The time at which the information is last updated.
48 /// </summary>
49 public DateTimeOffset? LastUpdated { get; private init; }
50 /// <summary>
51 /// The hypocentre of the earthquake.
52 /// </summary>
53 public Hypocentre? Hypocentre { get; private init; }
54 /// <summary>
55 /// The magnitude of the earthquake.
56 /// </summary>
57 public Magnitude? Magnitude { get; private init; }
58 /// <summary>
59 /// The informational text from the telegram.
60 /// </summary>
61 public string? InformationalText { get; private init; }
62 /// <summary>
63 /// The detailed intensity tree.
64 /// </summary>
65 public IEnumerable<DetailIntensityTemplate> IntensityTree { get; private init; }
66 }
```

Listing A.10.20: EasonEetwViewer/Models/PastPage/EarthquakeDetailsTemplate.cs

```

1  using EasonEetwViewer.Dmdata.Dto.Enum;
2
3  namespace EasonEetwViewer.Models.PastPage;
4  /// <summary>
5  /// Represents the node for an intensity in the intensity display tree.
6  /// </summary>
7  internal record DetailIntensityTemplate
8  {
9      /// <summary>
```

```

10     /// The intensity of the node.
11     /// </summary>
12     public required Intensity Intensity { get; init; }
13     /// <summary>
14     /// The nodes that has this intensity.
15     /// </summary>
16     public required IEnumerable<DisplayNode> PositionNodes { get; init; }
17 }
```

Listing A.10.21: EasonEetwViewer/Models/PastPage/DetailIntensityTemplate.cs

```

1  using EasonEetwViewer.Dmdata.Api.Dtos.Record.GdEarthquake;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
3  using EasonEetwViewer.Dmdata.Dtos.Enum;
4
5  namespace EasonEetwViewer.Models.PastPage;
6  /// <summary>
7  /// Describes an earthquake item in the sidebar earthquake list.
8  /// </summary>
9  internal record EarthquakeItemTemplate
{
10
11     /// <summary>
12     /// Initializes a new instance of the <see cref="EarthquakeItemTemplate"/> class
13     /// from a <see cref="EarthquakeInfo"/>.
14     /// </summary>
15     /// <param name="earthquakeInfo">The <see cref="EarthquakeInfo"/> used as the data
16     /// source.</param>
17     public EarthquakeItemTemplate(EarthquakeInfo earthquakeInfo)
18     {
19         EventId = earthquakeInfo.EventId;
20         Intensity = earthquakeInfo.MaxIntensity;
21         OriginTime = earthquakeInfo.OriginTime;
22         Hypocentre = earthquakeInfo.Hypocentre;
23         Magnitude = earthquakeInfo.Magnitude;
24     }
25     /// <summary>
26     /// The Event ID of the earthquake.
27     /// </summary>
28     public string EventId { get; private init; }
29     /// <summary>
30     /// The maximum intensity observed for the earthquake.
31     /// </summary>
32     public Intensity? Intensity { get; private init; }
33     /// <summary>
34     /// The time at which the earthquake originated.
35     /// </summary>
36     public DateTimeOffset? OriginTime { get; private init; }
37     /// <summary>
38     /// The hypocentre of the earthquake.
39     /// </summary>
40     public Hypocentre? Hypocentre { get; private init; }
41     /// <summary>
42     /// The magnitude of the earthquake.
43     /// </summary>
44     public Magnitude? Magnitude { get; private init; }
45 }
```

Listing A.10.22: EasonEetwViewer/Models/PastPage/EarthquakeItemTemplate.cs

```

1  using CommunityToolkit.Mvvm.Input;
2  using EasonEetwViewer.Lang;
3
4  namespace EasonEetwViewer.Models.SettingPage;
5  /// <summary>
6  /// Describes an empty WebSocket connection.
7  /// </summary>
8  internal partial class EmptyWebSocketConnectionTemplate : IWebSocketConnectionTemplate
9  {
10     /// <inheritdoc/>
11     public int WebSocketId
12         => -1;
13     /// <inheritdoc/>
14     public string ApplicationName
15         => SettingPageResources.WebSocketListTextEmptyConnection;
16     /// <inheritdoc/>
17     public DateTimeOffset StartTime
18         => new();
19     /// <inheritdoc/>
20     public bool IsEnabled
21         => false;
22     /// <inheritdoc cref="IWebSocketConnectionTemplate.DisconnectCommand"/>
23     /// <returns>A <see cref="Task"/> object that represents the asynchronous
24     /// operation.</returns>
25     [RelayCommand]
26     private static Task Disconnect()
27         => Task.CompletedTask;
28     /// <summary>
29     /// The instance of <see cref="EmptyWebSocketConnectionTemplate"/>.
30     /// </summary>
31     public static EmptyWebSocketConnectionTemplate Instance
32         => new();
33     /// <summary>
34     /// Initializes a new instance of the <see
35     /// cref="EmptyWebSocketConnectionTemplate"/> class.
36     /// </summary>
37     private EmptyWebSocketConnectionTemplate() { }
38 }
```

Listing A.10.23: EasonEetwViewer/Models/SettingPage/EmptyWebSocketConnectionTemplate.cs

```

1  using CommunityToolkit.Mvvm.Input;
2
3  namespace EasonEetwViewer.Models.SettingPage;
4  /// <summary>
5  /// Describes a template for a WebSocket connection.
6  /// </summary>
7  internal interface IWebSocketConnectionTemplate
8  {
9      /// <summary>
10     /// The WebSocket connection ID.
11     /// </summary>
12     int WebSocketId { get; }
13     /// <summary>
14     /// The name of the application.
15     /// </summary>
16     string? ApplicationName { get; }
```

```

17     /// <summary>
18     /// The start time of the connection.
19     /// </summary>
20     DateTimeOffset StartTime { get; }
21     /// <summary>
22     /// Whether the disconnect button is enabled.
23     /// </summary>
24     bool IsEnabled { get; }
25     /// <summary>
26     /// The async relay command to disconnect the WebSocket connection.
27     /// </summary>
28     IAsyncRelayCommand DisconnectCommand { get; }
29 }
```

Listing A.10.24: EasonEetwViewer/Models/SettingPage/IWebSocketConnectionTemplate.cs

```

1  using CommunityToolkit.Mvvm.ComponentModel;
2  using CommunityToolkit.Mvvm.Input;
3  using EasonEetwViewer.Dmdata.Api.Dtos.Record.WebSocket;
4
5  namespace EasonEetwViewer.Models.SettingPage;
6  /// <summary>
7  /// Describes a normal WebSocket connection.
8  /// </summary>
9  /// <param name="disconnectTask">The action to execute when disconnecting the
10    → WebSocket.</param>
11  /// <param name="webSocket">The WebSocket connection details.</param>
12  internal partial class WebSocketConnectionTemplate(WebSocketDetails webSocket,
13    → Func<Task<bool>> disconnectTask) : ObservableObject, IWebSocketConnectionTemplate
14  {
15    /// <inheritdoc/>
16    public int WebSocketId { get; private init; } = webSocket.WebSocketId;
17    /// <inheritdoc/>
18    public string? ApplicationName { get; private init; } = webSocket.ApplicationName;
19    /// <inheritdoc/>
20    public DateTimeOffset StartTime { get; private init; } = webSocket.StartTime;
21    /// <inheritdoc cref="IWebSocketConnectionTemplate.IsEnabled"/>
22    [ObservableProperty]
23    private bool _isEnabled = true;
24    /// <summary>
25    /// The action to execute when disconnecting the WebSocket.
26    /// </summary>
27    public Func<Task<bool>> DisconnectTask { get; private init; } = disconnectTask;
28    /// <inheritdoc cref="IWebSocketConnectionTemplate.DisconnectCommand"/>
29    /// <returns>A <see cref="Task"/> object that represents the asynchronous
30      → operation.</returns>
31    [RelayCommand]
32    private async Task Disconnect()
33    {
34      bool isSuccessful = await DisconnectTask();
35      if (isSuccessful)
36      {
37        IsEnabled = false;
38      }
39    }
40 }
```

Listing A.10.25: EasonEetwViewer/Models/SettingPage/WebSocketConnectionTemplate.cs

```

1  using EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
2  using EasonEetwViewer.Extensions;
3
4  namespace EasonEetwViewer.Models.RealTimePage;
5  /// <summary>
6  /// Represents a template for the current tsunami details.
7  /// </summary>
8  internal record TsunamiDetailsTemplate
{
9
10    /// <summary>
11    /// The informational text of the tsunami.
12    /// </summary>
13    public string InformationalText { get; private init; }
14    /// <summary>
15    /// The expiry time of the information.
16    /// </summary>
17    public DateTimeOffset ExpiryTime { get; private init; }
18    /// <summary>
19    /// The update time of the information.
20    /// </summary>
21    public DateTimeOffset UpdateTime { get; private init; }
22    /// <summary>
23    /// The maximum warning type of the tsunami.
24    /// </summary>
25    public TsunamiWarningType MaxWarningType { get; private init; }
26    /// <summary>
27    /// Initializes a new instance of the <see cref="TsunamiDetailsTemplate"/> class.
28    /// </summary>
29    /// <param name="tsunami">The tsunami information telegram.</param>
30    /// <param name="validDateTime">The time the information is valid until.</param>
31    public TsunamiDetailsTemplate(TsunamiInformationSchema tsunami, DateTimeOffset
32        ↪ validDateTime)
33    {
34        InformationalText = tsunami.ToInformationString();
35        ExpiryTime = validDateTime;
36        UpdateTime = tsunami.PressDateTime;
37        MaxWarningType = tsunami.Body.Tsunami?.Forecasts?
38            .Max(forecast => forecast.Kind.Code.ToTsunamiWarningType())
39            ?? TsunamiWarningType.None;
40    }
}

```

Listing A.10.26: EasonEetwViewer/Models/RealTimePage/TsunamiDetailsTemplate.cs

```

1  namespace EasonEetwViewer.Models.RealTimePage;
2  internal enum TsunamiWarningType
3  {
4      None,
5      Information,
6      Caution,
7      Warning,
8      SpecialWarning
9 }

```

Listing A.10.27: EasonEetwViewer/Models/RealTimePage/TsunamiWarningType.cs

```
1  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Range;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
4  using EasonEetwViewer.Extensions;
5  using EasonEetwViewer.Lang;
6
7  namespace EasonEetwViewer.Models.RealTimePage;
8  ///<summary>
9  /// Represents the template for the details of an EEW.
10 ///</summary>
11 internal record EewDetailsTemplate
12 {
13     ///<summary>
14     /// The expiry time of the EEW.
15     ///</summary>
16     public DateTimeOffset ExpiryTime { get; private init; }
17     ///<summary>
18     /// The last updated time of the EEW Information.
19     ///</summary>
20     public DateTimeOffset UpdateTime { get; private init; }
21     ///<summary>
22     /// The serial number of the EEW.
23     ///</summary>
24     public int Serial { get; private init; }
25     ///<summary>
26     /// The event ID of the EEW.
27     ///</summary>
28     public string? EventId { get; private init; }
29     ///<summary>
30     /// The warning type of the EEW.
31     ///</summary>
32     public EewWarningType EewWarningType { get; private init; }
33     ///<summary>
34     /// The earthquake reported by the EEW.
35     ///</summary>
36     public Earthquake? Earthquake { get; private init; }
37     ///<summary>
38     /// The intensity reported by the EEW.
39     ///</summary>
40     public IntensityInfo? IntensityInfo { get; private init; }
41     ///<summary>
42     /// The informational text of the EEW.
43     ///</summary>
44     public string InformationalText { get; private init; }
45     ///<summary>
46     /// Whether the hypocentre is assumed.
47     ///</summary>
48     public bool IsAssumedHypocentre { get; private init; }
49     ///<summary>
50     /// Whether the information is based on only one observation point.
51     ///</summary>
52     public bool IsOnePointInfo { get; private init; }
53     ///<summary>
54     /// Whether the maximum intensity is exact, or is a lower bound.
55     ///</summary>
56     public bool IsAbove { get; private init; }
57     ///<summary>
58     /// The text before the intensity text.
59     ///</summary>
```

```

60     public string BeforeIntensityText
61         => IsAbove
62             ? RealtimePageResources.IntensityOverTextBefore
63             : string.Empty;
64     /// <summary>
65     /// The text after the intensity text.
66     /// </summary>
67     public string AfterIntensityText
68         => IsAbove
69             ? RealtimePageResources.IntensityOverTextAfter
70             : string.Empty;
71     /// <summary>
72     /// The maximum intensity of the EEW, to be displayed.
73     /// </summary>
74     public Enum? MaxIntensityEnum
75         => IsAbove
76             ? IntensityInfo?.ForecastMaxInt.From
77             : IntensityInfo?.ForecastMaxInt.To;
78     /// <summary>
79     /// The cancellation token source related with this EEW issue.
80     /// </summary>
81     public CancellationTokenSource TokenSource { get; private init; }
82     /// <summary>
83     /// The cancellation token related with this EEW issue.
84     /// </summary>
85     public CancellationToken Token { get; private init; }
86     /// <summary>
87     /// Initializes a new instance of the <see cref="EewDetailsTemplate"/> class.
88     /// </summary>
89     /// <param name="eew">The EEW Telegram.</param>
90     /// <param name="expiryTime">The expiry time of this information.</param>
91     /// <param name="serial">The serial of this information.</param>
92     public EewDetailsTemplate(EewInformationSchema eew, DateTimeOffset expiryTime,
93         → int serial)
94     {
95         ExpiryTime = expiryTime;
96         UpdateTime = eew.PressDateTime;
97         Serial = serial;
98         EventId = eew.EventId;
99         EewWarningType = eew.ToWarningType();
100        Earthquake = eew.Body.Earthquake;
101        IntensityInfo = eew.Body.Intensity;
102        InformationalText = eew.ToInformationString();
103        IsAssumedHypocentre =
104            eew.Body.Earthquake is not null &&
105            eew.Body.Earthquake.IsAssumedHypocentre();
106        IsOnePointInfo =
107            eew.Body.Earthquake is not null &&
108            eew.Body.Earthquake.Hypocentre.Accuracy.IsOnePointInfo();
109        IsAbove =
110            eew.Body.Intensity is not null &&
111            eew.Body.Intensity.ForecastMaxInt.To is IntensityUpper.Above;
112        TokenSource = new();
113        Token = TokenSource.Token;
114    }

```

Listing A.10.28: EasonEetwViewer/Models/RealTimePage/EewDetailsTemplate.cs

```

1  namespace EasonEetwViewer.Models.RealTimePage;
2
3  /// <summary>
4  /// Represents the EEW type that is to be displayed.
5  /// </summary>
6  internal enum EewWarningType
7  {
8      /// <summary>
9      /// An EEW Forecast (that is not cancelled or the final info).
10     /// </summary>
11     Forecast,
12     /// <summary>
13     /// An EEW Warning (that is not cancelled or the final info).
14     /// </summary>
15     Warning,
16     /// <summary>
17     /// An EEW that is the final information (that is not cancelled).
18     /// </summary>
19     Final,
20     /// <summary>
21     /// An EEW that has been cancelled.
22     /// </summary>
23     Cancelled
24 }

```

Listing A.10.29: EasonEetwViewer/Models/RealTimePage/EewWarningType.cs

```

1  using System.Diagnostics;
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Range;
4  using EasonEetwViewer.Lang;
5  using EasonEetwViewer.Models.RealTimePage;
6
7  namespace EasonEetwViewer.Extensions;
8
9  /// <summary>
10    /// Provides extension methods for enums to have <c>ToColourString</c> method.
11    /// </summary>
12  internal static class EnumDisplayColourExtensions
13  {
14      /// <summary>
15      /// Converts <see cref="TsunamiWarningType"/> to colour string.
16      /// </summary>
17      /// <param name="tsunamiWarningType">The enum to be converted.</param>
18      /// <returns>The string representing the colour.</returns>
19      /// <exception cref="UnreachableException">When the program reaches an unreachable
20      → state.</exception>
21      public static string ToColourString(this TsunamiWarningType tsunamiWarningType)
22          => tsunamiWarningType switch
23          {
24              TsunamiWarningType.Information
25                  => RealtimePageResources.TsunamiColourInformation,
26              TsunamiWarningType.Caution
27                  => RealtimePageResources.TsunamiColourCaution,
28              TsunamiWarningType.Warning
29                  => RealtimePageResources.TsunamiColourWarning,
30              TsunamiWarningType.SpecialWarning
31                  => RealtimePageResources.TsunamiColourSpecialWarning,

```

```
31     TsunamiWarningType.None
32         => RealtimePageResources.TsunamiColourNone,
33     -
34         => throw new UnreachableException()
35     };
36
37     /// <summary>
38     /// Converts <see cref="Intensity"/> to colour string.
39     /// </summary>
40     /// <param name="intensity">The enum to be converted.</param>
41     /// <returns>The string representing the colour.</returns>
42     /// <exception cref="UnreachableException">When the program reaches an unreachable
43     → state.</exception>
43     public static string ToColourString(this Intensity intensity)
44         => intensity switch
45     {
46         Intensity.One
47             => EarthquakeResources.IntensityColourOne,
48         Intensity.Two
49             => EarthquakeResources.IntensityColourTwo,
50         Intensity.Three
51             => EarthquakeResources.IntensityColourThree,
52         Intensity.Four
53             => EarthquakeResources.IntensityColourFour,
54         Intensity.FiveWeak
55             => EarthquakeResources.IntensityColourFiveWeak,
56         Intensity.FiveStrong
57             => EarthquakeResources.IntensityColourFiveStrong,
58         Intensity.SixWeak
59             => EarthquakeResources.IntensityColourSixWeak,
60         Intensity.SixStrong
61             => EarthquakeResources.IntensityColourSixStrong,
62         Intensity.Seven
63             => EarthquakeResources.IntensityColourSeven,
64     -
65         => throw new UnreachableException()
66     };
67
68     /// <summary>
69     /// Converts <see cref="EewWarningType"/> to colour string.
70     /// </summary>
71     /// <param name="eew">The enum to be converted.</param>
72     /// <returns>The string representing the colour.</returns>
73     /// <exception cref="UnreachableException">When the program reaches an unreachable
74     → state.</exception>
74     public static string ToColourString(this EewWarningType eew)
75         => eew switch
76     {
77         EewWarningType.Cancelled
78             => RealtimePageResources.EewColourCancelled,
79         EewWarningType.Final
80             => RealtimePageResources.EewColourFinal,
81         EewWarningType.Warning
82             => RealtimePageResources.EewColourWarning,
83         EewWarningType.Forecast
84             => RealtimePageResources.EewColourForecast,
85     -
86         => throw new UnreachableException()
87     };

```

```
88
89     /// <summary>
90     /// Converts <see cref="IntensityLower"/> to colour string.
91     /// </summary>
92     /// <param name="intensity">The enum to be converted.</param>
93     /// <returns>The string representing the colour.</returns>
94     /// <exception cref="UnreachableException">When the program reaches an unreachable
95     /// state.</exception>
96     public static string ToColourString(this IntensityLower intensity)
97         => intensity switch
98         {
99             IntensityLower.Zero
100                => EarthquakeResources.IntensityColourZero,
101                IntensityLower.One
102                    => EarthquakeResources.IntensityColourOne,
103                    IntensityLower.Two
104                        => EarthquakeResources.IntensityColourTwo,
105                        IntensityLower.Three
106                            => EarthquakeResources.IntensityColourThree,
107                            IntensityLower.Four
108                                => EarthquakeResources.IntensityColourFour,
109                                IntensityLower.FiveWeak
110                                    => EarthquakeResources.IntensityColourFiveWeak,
111                                    IntensityLower.FiveStrong
112                                        => EarthquakeResources.IntensityColourFiveStrong,
113                                        IntensityLower.SixWeak
114                                            => EarthquakeResources.IntensityColourSixWeak,
115                                            IntensityLower.SixStrong
116                                                => EarthquakeResources.IntensityColourSixStrong,
117                                                IntensityLower.Seven
118                                                    => EarthquakeResources.IntensityColourSeven,
119                                                    IntensityLower.Unclear
120                                                        => EarthquakeResources.IntensityColourUnknown,
121                                                        -=> throw new UnreachableException()
122         };
123
124     /// <summary>
125     /// Converts <see cref="IntensityUpper"/> to colour string.
126     /// </summary>
127     /// <param name="intensity">The enum to be converted.</param>
128     /// <returns>The string representing the colour.</returns>
129     /// <exception cref="UnreachableException">When the program reaches an unreachable
130     /// state.</exception>
131     public static string ToColourString(this IntensityUpper intensity)
132         => intensity switch
133         {
134             IntensityUpper.Zero
135                 => EarthquakeResources.IntensityColourZero,
136                 IntensityUpper.One
137                     => EarthquakeResources.IntensityColourOne,
138                     IntensityUpper.Two
139                         => EarthquakeResources.IntensityColourTwo,
140                         IntensityUpper.Three
141                             => EarthquakeResources.IntensityColourThree,
142                             IntensityUpper.Four
143                                 => EarthquakeResources.IntensityColourFour,
144                                 IntensityUpper.FiveWeak
145                                     => EarthquakeResources.IntensityColourFiveWeak,
```

```

145     IntensityUpper.FiveStrong
146         => EarthquakeResources.IntensityColourFiveStrong,
147     IntensityUpper.SixWeak
148         => EarthquakeResources.IntensityColourSixWeak,
149     IntensityUpper.SixStrong
150         => EarthquakeResources.IntensityColourSixStrong,
151     IntensityUpper.Seven
152         => EarthquakeResources.IntensityColourSeven,
153     IntensityUpper.Unclear or IntensityUpper.Above
154         => EarthquakeResources.IntensityColourUnknown,
155     -
156         => throw new UnreachableException()
157     };
158 }

```

Listing A.10.30: EasonEetwViewer/Extensions/EnumDisplayColourExtensions.cs

```

1  using System.Text;
2  using EasonEetwViewer.Dmdata.Telemgram.Dtos.Schema;
3
4  namespace EasonEetwViewer.Extensions;
5
6  /// <summary>
7  /// Provides extension methods for converting telegram to a information string.
8  /// </summary>
9  internal static class InformationalTextExtensions
10 {
11     /// <summary>
12     /// Appends a line to the string builder if the value is not null or whitespace.
13     /// </summary>
14     /// <param name="stringBuilder">The instance of the string builder.</param>
15     /// <param name="value">The value to be appended.</param>
16     /// <returns>The instance of the string builder, for calls to be
17     → chained.</returns>
18     private static StringBuilder AppendLineIfVisible(this StringBuilder
19         → stringBuilder, string? value)
20         => !string.IsNullOrEmptyWhiteSpace(value)
21         ? stringBuilder.AppendLine(value)
22         : stringBuilder;
23     /// <summary>
24     /// Converts the earthquake information schema telegram to a information string.
25     /// </summary>
26     /// <param name="earthquake">The <see cref="EarthquakeInformationSchema"/> to be
27     → converted.</param>
28     /// <returns>The converted information string.</returns>
29     public static string ToInformationalString(this EarthquakeInformationSchema
30         → earthquake)
31         => new StringBuilder()
32             .AppendLineIfVisible(earthquake.Headline)
33             .AppendLineIfVisible(earthquake.Body.Text)
34             .AppendLineIfVisible(earthquake.Body.Comments?.FreeText)
35             .AppendLineIfVisible(earthquake.Body.Comments?.Forecast?.Text)
36             .AppendLineIfVisible(earthquake.Body.Comments?.Var?.Text)
37             .ToString();
38     /// <summary>
39     /// Converts the tsunami information schema telegram to a information string.
40     /// </summary>
41     /// <param name="tsunami">The <see cref="TsunamiInformationSchema"/> to be
42     → converted.</param>

```

```

38     /// <returns>The converted information string.</returns>
39     public static string ToInformationString(this TsunamiInformationSchema tsunami)
40         => new StringBuilder()
41             .AppendLineIfVisible(tsunami.Headline)
42             .AppendLineIfVisible(tsunami.Body.Text)
43             .AppendLineIfVisible(tsunami.Body.Comments?.FreeText)
44             .AppendLineIfVisible(tsunami.Body.Comments?.Warning?.Text)
45             .ToString();
46     /// <summary>
47     /// Converts the eew information schema telegram to a information string.
48     /// </summary>
49     /// <param name="eew">The <see cref="EewInformationSchema"/> to be
50     /// converted.</param>
51     /// <returns>The converted information string.</returns>
52     public static string ToInformationString(this EewInformationSchema eew)
53         => new StringBuilder()
54             .AppendLineIfVisible(eew.Headline)
55             .AppendLineIfVisible(eew.Body.Text)
56             .AppendLineIfVisible(eew.Body.Comments?.FreeText)
57             .AppendLineIfVisible(eew.Body.Comments?.Warning?.Text)
58             .ToString();
}

```

Listing A.10.31: EasonEetwViewer/Extensions/InformationalTextExtensions.cs

```

1  using EasonEetwViewer.Dmdata.Dtos.Enum;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
4  using EasonEetwViewer.Dmdata.Telegram.Dtos.TsunamiInformation;
5  using EasonEetwViewer.KyoshinMonitor.Dtos;
6  using EasonEetwViewer.KyoshinMonitor.Extensions;
7  using Mapsui;
8  using Mapsui.Extensions;
9  using Mapsui.Layers;
10 using Mapsui.Projections;
11 using Mapsui.Rendering.Skia.Extensions;
12 using Mapsui.Styles;
13 using Mapsui.Styles.Thematics;
14 using SkiaSharp;
15 using Station = EasonEetwViewer.Dmdata.Api.Dtos.Record.EarthquakeParameter.Station;
16
17 namespace EasonEetwViewer.Extensions;
18
19 // Adapted from https://mapsui.com/v5/samples/ - Styles - ThemeStyle on ShapeFile
20 /// <summary>
21 /// Provides extensions to deal with <see cref="Mapsui"/>.
22 /// </summary>
23 internal static class MapStyleExtensions
{
24
25     /// <summary>
26     /// Creates a theme style for the regions.
27     /// </summary>
28     /// <param name="regions">The regions to be converted.</param>
29     /// <returns>The converted style.</returns>
30     public static ThemeStyle ToRegionStyle(this IEnumerable<RegionIntensity> regions)
31         => new(feature =>
32         {
33             RegionIntensity? region = regions

```

```
34         .SingleOrDefault(region => region.Code == (string)feature["code"]!);
35
36     return region?.MaxInt is Intensity intensity
37         ? new VectorStyle()
38         {
39             Fill = new Brush(intensity.ToString().ToColour(0.60f))
40         }
41         : null;
42     );
43 /// <summary>
44 /// Creates a theme style for the regions.
45 /// </summary>
46 /// <param name="regions">The regions to be converted.</param>
47 /// <returns>The converted style.</returns>
48 public static ThemeStyle ToRegionStyle(this IEnumerable<Region> regions)
49     => new(feature =>
50     {
51         Region? region = regions
52             .SingleOrDefault(region => region.Code == (string)feature["code"]!);
53         return region?.ForecastMaxInt.From.ToIntensity() is Intensity intensity
54             ? new VectorStyle()
55             {
56                 Fill = new Brush(intensity.ToString().ToColour(0.60f))
57             }
58             : null;
59     );
59 /// <summary>
60 /// Creates a theme style for the regions.
61 /// </summary>
62 /// <param name="forecasts">The regions to be converted.</param>
63 /// <returns>The converted style.</returns>
64 public static ThemeStyle ToRegionStyle(this IEnumerable<Forecast> forecasts)
65     => new(feature =>
66     {
67         Forecast? forecast = forecasts
68             .SingleOrDefault(forecast => forecast.Code ==
69                 (string)feature["code"]!);
70         return forecast is null
71             ? null
72             : new VectorStyle()
73             {
74                 Line = new Pen(forecast.Kind.Code.TsunamiWarningType().ToColour_
75                     .ToString().ToColour(0.80f), 2.5)
76             };
77     );
78 /// <summary>
79 /// Creates a style for the intensity.
80 /// </summary>
81 /// <param name="intensity">The intensity for which the station has.</param>
82 /// <returns>The converted style.</returns>
83 public static SymbolStyle ToStationStyle(this Intensity intensity)
84     => new()
85     {
86         SymbolScale = 0.25,
87         Fill = new Brush(intensity.ToString().ToColour()),
88         Outline = new Pen { Color = Color.Black }
89     };
90 }
```

```
91     /// <summary>
92     /// Creates a style for the <see cref="SKColor"/> of the station.
93     /// </summary>
94     /// <param name="colour">The colour extracted from the station.</param>
95     /// <returns>The style converted.</returns>
96     public static SymbolStyle ToStationStyle(this SKColor colour)
97         => new()
98     {
99         SymbolScale = 0.1,
100        Fill = new Brush(colour.ColourToHeight().HeightToColour().ToMapsui())
101    };
102
103
104    /// <summary>
105    /// Creates a feature for the station.
106    /// </summary>
107    /// <param name="pointColour">The tuple containing the station and the
108    → colour.</param>
109    /// <returns>The converted feature.</returns>
110    public static PointFeature ToStationFeature(this (ObservationPoint point, SKColor
111        → colour) pointColour)
112        => new((pointColour.point.Location.Longitude,
113            → pointColour.point.Location.Latitude).LonLatToMPoint())
114    {
115        Styles = [pointColour.colour.ToStationStyle()]
116    };
117
118
119    /// <summary>
120    /// Creates a feature for the station.
121    /// </summary>
122    /// <param name="stationIntensity">The tuple containing the station and the
123    → intensity.</param>
124    /// <returns>The converted feature.</returns>
125    public static PointFeature ToStationFeature(this (Station station, Intensity
126        → intensity) stationIntensity)
127        => new((stationIntensity.station.Longitude,
128            → stationIntensity.station.Latitude).LonLatToMPoint())
129    {
130        Styles = [stationIntensity.intensity.ToStationStyle()]
131    };
132
133
134    /// <summary>
135    /// Converts a pair of longitude and latitude to a <see cref="MPoint"/>.
136    /// </summary>
137    /// <param name="coordinate">The pair of coordinates to convert.</param>
138    /// <returns>The converted <see cref="MPoint"/>.</returns>
139    public static MPoint LonLatToMPoint(this (double longitude, double latitude)
140        → coordinate)
141        => SphericalMercator
142            .FromLonLat(
143                coordinate.longitude,
144                coordinate.latitude)
145            .ToMPoint();
146
147
148    /// <summary>
149    /// Round a double to the certain number of significant figures, in decimal.
150    /// </summary>
151    /// <param name="value">The value to be rounded.</param>
152    /// <param name="figures">The number of significant figures desired.</param>
```

```

143     ///<returns>The decimal with the desired number of significant figures.</returns>
144     public static decimal ToSignificantFigures(this double value, int figures)
145     {
146         if (value == 0)
147         {
148             return 0;
149         }
150
151         decimal d = (decimal)value;
152         int digits = (int)Math.Floor(Math.Log10(Math.Abs(value)) + 1);
153
154         decimal scaleFactor = (decimal)Math.Pow(10, digits);
155         return scaleFactor * Math.Round(d / scaleFactor, figures);
156     }
157 }
```

Listing A.10.32: EasonEetwViewer/Extensions/MapStyleExtensions.cs

```

1  using System.Diagnostics;
2  using EasonEetwViewer.Dmdata.Authentication.Abstractions;
3  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
4  using EasonEetwViewer.Dmdata.Dtos.Enum;
5  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Accuracy;
6  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Range;
7  using EasonEetwViewer.KyoshinMonitor.Abstractions;
8  using EasonEetwViewer.Lang;
9  using EasonEetwViewer.Models.RealTimePage;
10
11 namespace EasonEetwViewer.Extensions;
12
13 ///<summary>
14 /// Provides extension methods for enums to have <c>ToDisplayString</c> method.
15 ///</summary>
16 internal static class EnumDisplayTextExtensions
17 {
18     ///<summary>
19     /// Converts <see cref="MeasurementType"/> to display string.
20     ///</summary>
21     ///<param name="measurementType">The enum to be converted.</param>
22     ///<returns>The converted display string.</returns>
23     ///<exception cref="UnreachableException">When the program reaches an unreachable
24     → state.</exception>
25     public static string ToDisplayString(this MeasurementType measurementType)
26         => measurementType switch
27         {
28             MeasurementType.MeasuredIntensity
29                 => SettingPageResources.KmoniMeasurementTextMeasuredIntensity,
30             MeasurementType.PeakGroundAcceleration
31                 => SettingPageResources.KmoniMeasurementTextPeakGroundAcceleration,
32             MeasurementType.PeakGroundVelocity
33                 => SettingPageResources.KmoniMeasurementTextPeakGroundVelocity,
34             MeasurementType.PeakGroundDisplacement
35                 => SettingPageResources.KmoniMeasurementTextPeakGroundDisplacement,
36             MeasurementType.Response0125
37                 => SettingPageResources.KmoniMeasurementTextResponse0125,
38             MeasurementType.Response0250
39                 => SettingPageResources.KmoniMeasurementTextResponse0250,
                    MeasurementType.Response0500
}
```

```
40             => SettingPageResources.KmoniMeasurementTextResponse0500,
41             MeasurementType.Response1000
42                 => SettingPageResources.KmoniMeasurementTextResponse1000,
43             MeasurementType.Response2000
44                 => SettingPageResources.KmoniMeasurementTextResponse2000,
45             MeasurementType.Response4000
46                 => SettingPageResources.KmoniMeasurementTextResponse4000,
47
48             -           => throw new UnreachableException(),
49         };
50     /// <summary>
51     /// Converts <see cref="SensorType"/> to display string.
52     /// </summary>
53     /// <param name="sensorType">The enum to be converted.</param>
54     /// <returns>The converted display string.</returns>
55     /// <exception cref="UnreachableException">When the program reaches an unreachable
56     → state.</exception>
56     public static string ToDisplayString(this SensorType sensorType)
57         => sensorType switch
58     {
59         SensorType.Surface
60             => SettingPageResources.KmoniSensorTextSurface,
61         SensorType.Borehole
62             => SettingPageResources.KmoniSensorTextBorehole,
63
64             -           => throw new UnreachableException(),
65     };
66     /// <summary>
67     /// Converts <see cref="AuthenticationStatus"/> to display string.
68     /// </summary>
69     /// <param name="authenticationStatus">The enum to be converted.</param>
70     /// <returns>The converted display string.</returns>
71     /// <exception cref="UnreachableException">When the program reaches an unreachable
72     → state.</exception>
72     public static string ToDisplayString(this AuthenticationStatus
73         → authenticationStatus)
74         => authenticationStatus switch
75     {
76         AuthenticationStatus.ApiKey
77             => SettingPageResources.AuthenticationStatusTextApiKey,
78         AuthenticationStatus.OAuth
79             => SettingPageResources.AuthenticationStatusTextOAuth,
80         AuthenticationStatus.Null
81             => SettingPageResources.AuthenticationStatusTextNull,
82
83             -           => throw new UnreachableException(),
84     };
85     /// <summary>
86     /// Converts <see cref="TsunamiWarningType"/> to display string.
87     /// </summary>
88     /// <param name="tsunamiWarningType">The enum to be converted.</param>
89     /// <returns>The converted display string.</returns>
90     /// <exception cref="UnreachableException">When the program reaches an unreachable
91     → state.</exception>
90     public static string ToDisplayString(this TsunamiWarningType tsunamiWarningType)
91         => tsunamiWarningType switch
92     {
93         TsunamiWarningType.None
94             => RealtimePageResources.TsunamiTextNone,
```

```
95     TsunamiWarningType.Information  
96         => RealtimePageResources.TsunamiTextInformation,  
97     TsunamiWarningType.Caution  
98         => RealtimePageResources.TsunamiTextCaution,  
99     TsunamiWarningType.Warning  
100        => RealtimePageResources.TsunamiTextWarning,  
101     TsunamiWarningType.SpecialWarning  
102        => RealtimePageResources.TsunamiTextSpecialWarning,  
103     -  
104        => throw new UnreachableException()  
105    };  
106 /// <summary>  
107 /// Converts <see cref="Intensity"/> to display string.  
108 /// </summary>  
109 /// <param name="intensity">The enum to be converted.</param>  
110 /// <returns>The converted display string.</returns>  
111 /// <exception cref="UnreachableException">When the program reaches an unreachable  
112    ↳ state.</exception>  
113 public static string ToDisplayString(this Intensity intensity)  
114     => intensity switch  
115    {  
116        Intensity.One  
117            => EarthquakeResources.IntensityTextOne,  
118        Intensity.Two  
119            => EarthquakeResources.IntensityTextTwo,  
120        Intensity.Three  
121            => EarthquakeResources.IntensityTextThree,  
122        Intensity.Four  
123            => EarthquakeResources.IntensityTextFour,  
124        Intensity.FiveWeak  
125            => EarthquakeResources.IntensityTextFiveWeak,  
126        Intensity.FiveStrong  
127            => EarthquakeResources.IntensityTextFiveStrong,  
128        Intensity.SixWeak  
129            => EarthquakeResources.IntensityTextSixWeak,  
130        Intensity.SixStrong  
131            => EarthquakeResources.IntensityTextSixStrong,  
132        Intensity.Seven  
133            => EarthquakeResources.IntensityTextSeven,  
134     -  
135        => throw new UnreachableException()  
136    };  
137 /// <summary>  
138 /// Converts <see cref="EpicentreDepth"/> to display string.  
139 /// </summary>  
140 /// <param name="epicentreDepth">The enum to be converted.</param>  
141 /// <returns>The converted display string.</returns>  
142 /// <exception cref="UnreachableException">When the program reaches an unreachable  
143    ↳ state.</exception>  
144 public static string ToDisplayString(this EpicentreDepth epicentreDepth)  
145 #pragma warning disable IDE0072 // Add missing cases  
146     => epicentreDepth switch  
147    {  
148        EpicentreDepth.LevelIpf1Plum  
149            => RealtimePageResources.EpicentreDepthAccuracy1,  
150        EpicentreDepth.Ipf2  
151            => RealtimePageResources.EpicentreDepthAccuracy2,  
151        EpicentreDepth.Ipf30r4  
151            => RealtimePageResources.EpicentreDepthAccuracy3,
```

```
152     EpicentreDepth.Ipf50rMore
153         => RealtimePageResources.EpicentreDepthAccuracy4,
154     EpicentreDepth.Final
155         => RealtimePageResources.EpicentreDepthAccuracy9,
156     EpicentreDepth.Unknown
157         => EarthquakeResources.UnknownText,
158     -
159         => throw new UnreachableException(),
160     };
161 #pragma warning restore IDE0072 // Add missing cases
162     /// <summary>
163     /// Converts <see cref="Magnitude"/> to display string.
164     /// </summary>
165     /// <param name="magnitude">The enum to be converted.</param>
166     /// <returns>The converted display string.</returns>
167     /// <exception cref="UnreachableException">When the program reaches an unreachable
168     → state.</exception>
169     public static string ToDisplayString(this Magnitude magnitude)
170         => magnitude switch
171     {
172         Magnitude.SpeedMagnitude
173             => RealtimePageResources.MagnitudeAccuracy2,
174         Magnitude.FullPPPhase
175             => RealtimePageResources.MagnitudeAccuracy3,
176         Magnitude.FullPPPhaseMixed
177             => RealtimePageResources.MagnitudeAccuracy4,
178         Magnitude.FullPointPhase
179             => RealtimePageResources.MagnitudeAccuracy5,
180         Magnitude.Epos
181             => RealtimePageResources.MagnitudeAccuracy6,
182         Magnitude.Level0rPlum
183             => RealtimePageResources.MagnitudeAccuracy8,
184         Magnitude.Unknown
185             => EarthquakeResources.UnknownText,
186         -
187             => throw new UnreachableException(),
188     };
189     /// <summary>
190     /// Converts <see cref="MagnitudePoint"/> to display string.
191     /// </summary>
192     /// <param name="magnitudePoint">The enum to be converted.</param>
193     /// <returns>The converted display string.</returns>
194     /// <exception cref="UnreachableException">When the program reaches an unreachable
195     → state.</exception>
196     public static string ToDisplayString(this MagnitudePoint magnitudePoint)
197         => magnitudePoint switch
198     {
199         MagnitudePoint.OneOrLevel0rPlum
200             => RealtimePageResources.Point1,
201         MagnitudePoint.Two
202             => RealtimePageResources.Point2,
203         MagnitudePoint.Three
204             => RealtimePageResources.Point3,
205         MagnitudePoint.Four
206             => RealtimePageResources.Point4,
207         MagnitudePoint.FiveOrAbove
208             => RealtimePageResources.Point5,
209         MagnitudePoint.Unknown
210             => EarthquakeResources.UnknownText,
```

```
209         - => throw new UnreachableException(),
210     };
211     /// <summary>
212     /// Converts <see cref="DepthCondition"/> to display string.
213     /// </summary>
214     /// <param name="depthCondition">The enum to be converted.</param>
215     /// <returns>The converted display string.</returns>
216     /// <exception cref="UnreachableException">When the program reaches an unreachable
217     → state.</exception>
218     public static string ToDisplayString(this DepthCondition depthCondition)
219         => depthCondition switch
220     {
221         DepthCondition.Deep
222             => EarthquakeResources.DepthDeep,
223         DepthCondition.Shallow
224             => EarthquakeResources.DepthShallow,
225         DepthCondition.Unclear
226             => EarthquakeResources.UnknownText,
227
228         - => throw new UnreachableException()
229     };
230     /// <summary>
231     /// Converts <see cref="MagnitudeCondition"/> to display string.
232     /// </summary>
233     /// <param name="magnitudeCondition">The enum to be converted.</param>
234     /// <returns>The converted display string.</returns>
235     /// <exception cref="UnreachableException">When the program reaches an unreachable
236     → state.</exception>
237     public static string ToDisplayString(this MagnitudeCondition magnitudeCondition)
238         => magnitudeCondition switch
239     {
240         MagnitudeCondition.Huge
241             => EarthquakeResources.MagnitudeHuge,
242         MagnitudeCondition.Unclear
243             => EarthquakeResources.UnknownText,
244
245         - => throw new UnreachableException()
246     };
247     /// <summary>
248     /// Converts <see cref="MagnitudeUnit"/> to display string.
249     /// </summary>
250     /// <param name="magnitudeUnit">The enum to be converted.</param>
251     /// <returns>The converted display string.</returns>
252     /// <exception cref="UnreachableException">When the program reaches an unreachable
253     → state.</exception>
254     public static string ToDisplayString(this MagnitudeUnit magnitudeUnit)
255         => magnitudeUnit switch
256     {
257         MagnitudeUnit.JmaMagnitude
258             => EarthquakeResources.MagnitudeUnitJma,
259         MagnitudeUnit.NormalMagnitude
260             => EarthquakeResources.MagnitudeUnitMoment,
261
262         - => throw new UnreachableException()
263     };
264     /// <summary>
265     /// Converts <see cref="EewWarningType"/> to display string.
```

```
265     /// </summary>
266     /// <param name="eew">The enum to be converted.</param>
267     /// <returns>The converted display string.</returns>
268     /// <exception cref="UnreachableException">When the program reaches an unreachable
269     //→ state.</exception>
270     public static string ToDisplayString(this EewWarningType eew)
271         => eew switch
272         {
273             EewWarningType.Cancelled
274                 => RealtimePageResources.EewTextCancelled,
275             EewWarningType.Final
276                 => RealtimePageResources.EewTextFinal,
277             EewWarningType.Warning
278                 => RealtimePageResources.EewTextWarning,
279             EewWarningType.Forecast
280                 => RealtimePageResources.EewTextForecast,
281             -           => throw new UnreachableException()
282         };
283
284     /// <summary>
285     /// Converts <see cref="IntensityLower"/> to display string.
286     /// </summary>
287     /// <param name="intensity">The enum to be converted.</param>
288     /// <returns>The converted display string.</returns>
289     /// <exception cref="UnreachableException">When the program reaches an unreachable
290     //→ state.</exception>
291     public static string ToDisplayString(this IntensityLower intensity)
292         => intensity switch
293         {
294             IntensityLower.Zero
295                 => EarthquakeResources.IntensityTextZero,
296             IntensityLower.One
297                 => EarthquakeResources.IntensityTextOne,
298             IntensityLower.Two
299                 => EarthquakeResources.IntensityTextTwo,
300             IntensityLower.Three
301                 => EarthquakeResources.IntensityTextThree,
302             IntensityLower.Four
303                 => EarthquakeResources.IntensityTextFour,
304             IntensityLower.FiveWeak
305                 => EarthquakeResources.IntensityTextFiveWeak,
306             IntensityLower.FiveStrong
307                 => EarthquakeResources.IntensityTextFiveStrong,
308             IntensityLower.SixWeak
309                 => EarthquakeResources.IntensityTextSixWeak,
310             IntensityLower.SixStrong
311                 => EarthquakeResources.IntensityTextSixStrong,
312             IntensityLower.Seven
313                 => EarthquakeResources.IntensityTextSeven,
314             IntensityLower.Unclear
315                 => EarthquakeResources.IntensityTextUnknown,
316             -           => throw new UnreachableException()
317         };
318
319     /// <summary>
320     /// Converts <see cref="IntensityUpper"/> to display string.
321     /// </summary>
```

```

322     /// <param name="intensity">The enum to be converted.</param>
323     /// <returns>The converted display string.</returns>
324     /// <exception cref="UnreachableException">When the program reaches an unreachable
325     //→ state.</exception>
326     public static string ToDisplayString(this IntensityUpper intensity)
327         => intensity switch
328         {
329             IntensityUpper.Zero
330                 => EarthquakeResources.IntensityTextZero,
331             IntensityUpper.One
332                 => EarthquakeResources.IntensityTextOne,
333             IntensityUpper.Two
334                 => EarthquakeResources.IntensityTextTwo,
335             IntensityUpper.Three
336                 => EarthquakeResources.IntensityTextThree,
337             IntensityUpper.Four
338                 => EarthquakeResources.IntensityTextFour,
339             IntensityUpper.FiveWeak
340                 => EarthquakeResources.IntensityTextFiveWeak,
341             IntensityUpper.FiveStrong
342                 => EarthquakeResources.IntensityTextFiveStrong,
343             IntensityUpper.SixWeak
344                 => EarthquakeResources.IntensityTextSixWeak,
345             IntensityUpper.SixStrong
346                 => EarthquakeResources.IntensityTextSixStrong,
347             IntensityUpper.Seven
348                 => EarthquakeResources.IntensityTextSeven,
349             IntensityUpper.Unclear or IntensityUpper.Above
350                 => EarthquakeResources.IntensityTextUnknown,
351             -           => throw new UnreachableException()
352         };
353     }

```

Listing A.10.33: EasonEetwViewer/Extensions/EnumDisplayTextExtensions.cs

```

1  using Mapsui.Styles;
2
3  namespace EasonEetwViewer.Extensions;
4
5  /// <summary>
6  /// Provides extension methods to convert a string to a colour.
7  /// </summary>
8  internal static class StringColourExtensions
{
9
10    /// <summary>
11    /// Converts a string to a colour.
12    /// </summary>
13    /// <param name="colourString">The string of the color to be converted.</param>
14    /// <returns>The converted colour.</returns>
15    public static Color ToColour(this string colourString)
16        => Color.FromString(colourString);
17
18    /// <summary>
19    /// Converts a string to a colour with the specified opacity.
20    /// </summary>
21    /// <param name="colourString">The string of the color to be converted.</param>
22    /// <param name="opacity">The specified opacity.</param>

```

```

23     /// <returns>The converted colour.</returns>
24     public static Color ToColour(this string colourString, float opacity)
25         => Color.Opacity(colourString.ToColour(), opacity);
26 }
```

Listing A.10.34: EasonEetwViewer/Extensions/StringColourExtensions.cs

```

1  using System.Diagnostics;
2  using Mapsui;
3  using Mapsui.Logging;
4  using Microsoft.Extensions.DependencyInjection;
5  using Microsoft.Extensions.Logging;
6
7  namespace EasonEetwViewer.Extensions;
8  /// <summary>
9  /// Provides extensions to attach MapsUI logging to the <see cref="ILogger"/>.
10 /// </summary>
11 internal static class MapsuiLoggingExtensions
12 {
13     // https://mapsui.com/documentation/logging.html
14     /// <summary>
15     /// Attaches MapsUI logging to the <see cref="ILogger"/>.
16     /// </summary>
17     /// <param name="serviceProvider">The <see cref="ServiceProvider"/> of the
18     /// application.</param>
19     /// <returns>The <see cref="ServiceProvider"/> so that calls can be
20     /// chained.</returns>
21     /// <exception cref="UnreachableException">When the application reaches an
22     /// unreachable state.</exception>
23     public static IServiceProvider AttachMapsuiLogging(this IServiceProvider
24         serviceProvider)
25     {
26         ILogger<Map> logger = serviceProvider.GetRequiredService<ILogger<Map>>();
27
28         Logger.Settings.LogMapEvents = true;
29         Logger.Settings.LogWidgetEvents = true;
30         Logger.LogDelegate += (level, message, ex)
31             => logger.Log(level.ToMicrosoftLogLevel(), ex, "{Message}", message);
32
33         return serviceProvider;
34     }
35 }
```

Listing A.10.35: EasonEetwViewer/Extensions/MapsuiLoggingExtensions.cs

```

1  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent;
2  using EasonEetwViewer.Dmdata.Dtos.DmdataComponent.Enum;
3  using EasonEetwViewer.Lang;
4
5  namespace EasonEetwViewer.Extensions;
6  /// <summary>
7  /// Provides extension methods for enums to have <c>ToUnitString</c> and
7  /// <c>ToValueString</c> method.
8  /// </summary>
9  internal static class UnitValueExtensions
10 {
```

```

11     /// <summary>
12     /// Converts a <see cref="Depth"/> to a unit string.
13     /// </summary>
14     /// <param name="depth">The value to be converted.</param>
15     /// <returns>The converted unit.</returns>
16     public static string ToUnitString(this Depth depth)
17         => depth.Condition is DepthCondition
18             ? string.Empty
19             : depth.Unit;
20     /// <summary>
21     /// Converts a <see cref="Depth"/> to a value string.
22     /// </summary>
23     /// <param name="depth">The value to be converted.</param>
24     /// <returns>The converted value.</returns>
25     public static string ToValueString(this Depth depth)
26         => depth.Condition is DepthCondition condition
27             ? condition.ToString()
28             : depth.Value?.ToString()
29             ?? EarthquakeResources.UnknownText;
30     /// <summary>
31     /// Converts a <see cref="Magnitude"/> to a unit string.
32     /// </summary>
33     /// <param name="magnitude">The value to be converted.</param>
34     /// <returns>The converted unit.</returns>
35     public static string ToUnitString(this Magnitude magnitude)
36         => magnitude.Condition is MagnitudeCondition
37             ? EarthquakeResources.MagnitudeUnitDefault
38             : magnitude.Unit.ToString();
39     /// <summary>
40     /// Converts a <see cref="Magnitude"/> to a value string.
41     /// </summary>
42     /// <param name="magnitude">The value to be converted.</param>
43     /// <returns>The converted value.</returns>
44     public static string ToValueString(this Magnitude magnitude)
45         => magnitude.Condition is MagnitudeCondition condition
46             ? condition.ToString()
47             : magnitude.Value.ToString()
48             ?? EarthquakeResources.UnknownText;
49
50 }

```

Listing A.10.36: EasonEetwViewer/Extensions/UnitValueExtensions.cs

```

1  using System.Diagnostics;
2  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Accuracy;
4  using EasonEetwViewer.Dmdata.Telegram.Dtos.Schema;
5  using EasonEetwViewer.Models.RealTimePage;
6
7  namespace EasonEetwViewer.Extensions;
8  /// <summary>
9  /// Provides extension methods for warning type conversion.
10 /// </summary>
11 internal static class WarningTypeExtensions
12 {
13     /// <summary>
14     /// Converts the code string to a tsunami warning type.
15     /// </summary>

```

```

16     /// <param name="code">The code to be converted.</param>
17     /// <returns>The converted tsunami warning type.</returns>
18     /// <exception cref="UnreachableException">When the program reaches an unreachable
19     → state.</exception>
20     public static TsunamiWarningType ToTsunamiWarningType(this string code)
21         => code switch
22         {
23             "00" or "50" or "60"
24                 => TsunamiWarningType.None,
25             "71" or "72" or "73"
26                 => TsunamiWarningType.Information,
27             "62"
28                 => TsunamiWarningType.Caution,
29             "51"
30                 => TsunamiWarningType.Warning,
31             "52" or "53"
32                 => TsunamiWarningType.SpecialWarning,
33             - => throw new UnreachableException()
34         };
35
36     /// <summary>
37     /// Converts the EEW Information Schema to an EEW Warning Type.
38     /// </summary>
39     /// <param name="eew">The EEW to be converted.</param>
40     /// <returns>The converted EEW Warning Type.</returns>
41     public static EewWarningType ToWarningType(this EewInformationSchema eew)
42         => eew.Body.IsCancelled
43             ? EewWarningType.Cancelled
44             : eew.Body.IsLastInfo
45                 ? EewWarningType.Final
46                 : eew.Body.IsWarning ?? false
47                     ? EewWarningType.Warning
48                     : EewWarningType.Forecast;
49
50     /// <summary>
51     /// Determines if the accuracy is a one-point information.
52     /// </summary>
53     /// <param name="accuracy">The accuracy to be determined.</param>
54     /// <returns>Whether it is a one-point information.</returns>
55     public static bool IsOnePointInfo(this Accuracy accuracy)
56         => accuracy.Epicentres[0] == EpicentreDepth.LevelIpf1Plum
57             && accuracy.Epicentres[1] == EpicentreDepth.LevelIpf1Plum
58             && accuracy.Depth == EpicentreDepth.LevelIpf1Plum
59             && accuracy.Magnitude == Magnitude.LevelOrPlum
60             && accuracy.MagnitudePoint == MagnitudePoint.OneOrLevelOrPlum;
61
62     /// <summary>
63     /// Determines if the earthquake has an assumed hypocentre.
64     /// </summary>
65     /// <param name="earthquake">The earthquake to be determined.</param>
66     /// <returns>Whether it has an assumed hypocentre.</returns>
67     public static bool IsAssumedHypocentre(this Earthquake earthquake)
68         => earthquake.Condition is "仮定震源要素";
69 }

```

Listing A.10.37: EasonEetwViewer/Extensions/WarningTypeExtensions.cs

```
1  using System.Diagnostics;
```

```
2  using EasonEetwViewer.Dmdata.Dtos.Enum;
3  using EasonEetwViewer.Dmdata.Telegram.Dtos.EarthquakeInformation.Enum;
4  using EasonEetwViewer.Dmdata.Telegram.Dtos.EewInformation.Enum.Range;
5
6  namespace EasonEetwViewer.Extensions;
7
8  internal static class EnumConversionExtensions
9  {
10     public static Intensity? ToIntensity(this IntensityWithUnreceived intensity)
11         => intensity switch
12         {
13             IntensityWithUnreceived.One
14                 => Intensity.One,
15             IntensityWithUnreceived.Two
16                 => Intensity.Two,
17             IntensityWithUnreceived.Three
18                 => Intensity.Three,
19             IntensityWithUnreceived.Four
20                 => Intensity.Four,
21             IntensityWithUnreceived.FiveWeak
22                 => Intensity.FiveWeak,
23             IntensityWithUnreceived.FiveStrong
24                 => Intensity.FiveStrong,
25             IntensityWithUnreceived.SixWeak
26                 => Intensity.SixWeak,
27             IntensityWithUnreceived.SixStrong
28                 => Intensity.SixStrong,
29             IntensityWithUnreceived.Seven
30                 => Intensity.Seven,
31             IntensityWithUnreceived.Unreceived
32                 => null,
33
34             -> throw new UnreachableException()
35         };
36
37     public static Intensity? ToIntensity(this IntensityLower intensity)
38         => intensity switch
39         {
40             IntensityLower.One
41                 => Intensity.One,
42             IntensityLower.Two
43                 => Intensity.Two,
44             IntensityLower.Three
45                 => Intensity.Three,
46             IntensityLower.Four
47                 => Intensity.Four,
48             IntensityLower.FiveWeak
49                 => Intensity.FiveWeak,
50             IntensityLower.FiveStrong
51                 => Intensity.FiveStrong,
52             IntensityLower.SixWeak
53                 => Intensity.SixWeak,
54             IntensityLower.SixStrong
55                 => Intensity.SixStrong,
56             IntensityLower.Seven
57                 => Intensity.Seven,
58             IntensityLower.Zero or IntensityLower.Unclear
59                 => null,
60
61             ->
```

```

61             => throw new UnreachableException()
62     };
63
64     public static Intensity? ToIntensity(this IntensityUpper intensity)
65         => intensity switch
66     {
67         IntensityUpper.One
68             => Intensity.One,
69         IntensityUpper.Two
70             => Intensity.Two,
71         IntensityUpper.Three
72             => Intensity.Three,
73         IntensityUpper.Four
74             => Intensity.Four,
75         IntensityUpper.FiveWeak
76             => Intensity.FiveWeak,
77         IntensityUpper.FiveStrong
78             => Intensity.FiveStrong,
79         IntensityUpper.SixWeak
80             => Intensity.SixWeak,
81         IntensityUpper.SixStrong
82             => Intensity.SixStrong,
83         IntensityUpper.Seven
84             => Intensity.Seven,
85         IntensityUpper.Zero or IntensityUpper.Unclear or IntensityUpper.Above
86             => null,
87         -             => throw new UnreachableException()
88     };
89 }

```

Listing A.10.38: EasonEetwViewer/Extensions/EnumConversionExtensions.cs

```

1  using System.Diagnostics;
2  using Avalonia.Logging;
3  using Microsoft.Extensions.Logging;
4
5  namespace EasonEetwViewer.Extensions;
6
7  /// <summary>
8  /// Provides extensions for logging services.
9  /// </summary>
10 internal static class LoggingExtensions
11 {
12     /// <summary>
13     /// Converts an Avalonia <see cref="LogEventLevel"/> to a Microsoft <see
14     /// cref="LogLevel"/>.
15     /// </summary>
16     /// <param name="level">The log event level to be converted.</param>
17     /// <returns>The converted log level.</returns>
18     /// <exception cref="UnreachableException">When the program reaches a state that
19     /// was thought to be unreachable.</exception>
20     public static LogLevel ToMicrosoftLogLevel(this LogEventLevel level)
21         => level switch
22     {
23         LogEventLevel.Verbose
24             => LogLevel.Trace,
25         LogEventLevel.Debug

```

```

24             => LogLevel.Debug,
25             LogEventLevel.Information
26                 => LogLevel.Information,
27             LogEventLevel.Warning
28                 => LogLevel.Warning,
29             LogEventLevel.Error
30                 => LogLevel.Error,
31             LogEventLevel.Fatal
32                 => LogLevel.Critical,
33
34             -           => throw new UnreachableException()
35         };
36     /// <summary>
37     /// Converts an MapsUI <see cref="Mapsui.Logging.LogLevel"/> to a Microsoft <see
38     ///     cref="LogLevel"/>.
39     /// </summary>
40     /// <param name="level">The log event level to be converted.</param>
41     /// <returns>The converted log level.</returns>
42     /// <exception cref="UnreachableException">When the program reaches a state that
43     ///     was thought to be unreachable.</exception>
44     public static LogLevel ToMicrosoftLogLevel(this Mapsui.Logging.LogLevel level)
45         => level switch
46         {
47             Mapsui.Logging.LogLevel.Trace
48                 => LogLevel.Trace,
49             Mapsui.Logging.LogLevel.Debug
50                 => LogLevel.Debug,
51             Mapsui.Logging.LogLevel.Information
52                 => LogLevel.Information,
53             Mapsui.Logging.LogLevel.Warning
54                 => LogLevel.Warning,
55             Mapsui.Logging.LogLevel.Error
56                 => LogLevel.Error,
57
58             -           => throw new UnreachableException()
59         };
60     }

```

Listing A.10.39: EasonEetwViewer/Extensions/LoggingExtensions.cs

```

1  namespace EasonEetwViewer.Extensions;
2
3  /// <summary>
4  /// Provides extension methods for LINQ.
5  /// </summary>
6  internal static class LinqExtensions
7  {
8      /// <summary>
9      /// Filters a sequence of values for values that are not <see langword="null"/>.
10     /// </summary>
11     /// <typeparam name="T">The type in the enumerable to be filtered.</typeparam>
12     /// <param name="enumerable">The enumerable that is to be filtered.</param>
13     /// <returns>An enumerable that contains the elements that are not null.</returns>
14     public static IEnumerable<T> NotNull<T>(this IEnumerable<T?> enumerable) where T
15         : class
16         => enumerable
17             .Where(e => e is not null)
18             .Select(e => e!);

```

18 }

Listing A.10.40: EasonEetwViewer/Extensions/LinqExtensions.cs

```
1 <Window xmlns="https://github.com/avaloniaui"
2   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
3   xmlns:vm="using:EasonEetwViewer.ViewModels"
4   xmlns:m="using:EasonEetwViewer.Models.MainWindow"
5   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
7   mc:Ignorable="d"
8   d:DesignWidth="800"
9   d:DesignHeight="450"
10  x:Class="EasonEetwViewer.Views.MainWindow"
11  x:DataType="vm:MainWindowViewModel"
12  Icon="/Assets/avalonia-logo.ico"
13  Title="Eason's Early Earthquake and Tsunami Warning Viewer">
14
15  <Design.DataContext>
16    <vm:MainWindowViewModel/>
17  </Design.DataContext>
18
19  <SplitView IsPaneOpen="{Binding IsPaneOpen}"
20    CompactPaneLength="48"
21    DisplayMode="CompactInline"
22    OpenPaneLength="200"
23    Background="#1e1e1e"
24    PaneBackground="#1e1e1e">
25
26    <!--Sidebar-->
27
28    <SplitView.Pane>
29      <StackPanel Orientation="Vertical"
30        Spacing="5"
31        HorizontalAlignment="Stretch"
32        VerticalAlignment="Top">
33
34      <!--Sidebar Button-->
35
36      <Button HorizontalAlignment="Left"
37        Width="32"
38        Height="32"
39        Margin="8 10"
40        Command="{Binding TriggerPaneCommand}">
41        <PathIcon Height="11"
42          Data="{StaticResource SidebarRegular}" />
43      </Button>
44
45      <!--Sidebar Options-->
46
47      <ListBox Margin="5 0"
48        Padding="0"
49        ItemsSource="{Binding SidebarItems}"
50        SelectedItem="{Binding SelectedSidebarItem}">
51        <ListBox.Styles>
52          <Style Selector="ListBoxItem">
53            <Setter Property="Padding"
54              Value="12 8"/></Setter>
```

```

55             </Style>
56         </ListBox.Styles>
57         <ListBox.ItemTemplate>
58             <DataTemplate DataType="{x:Type m:SidebarItemTemplate}">
59                 <StackPanel Spacing="15"
60                     Orientation="Horizontal">
61                     <PathIcon Data="{Binding ListItemIcon}"
62                         Width="14" />
63                     <TextBlock Text="{Binding Label}" />
64                 </StackPanel>
65             </DataTemplate>
66         </ListBox.ItemTemplate>
67     </ListBox>
68 </StackPanel>
69 </SplitView.Pane>
70
71 <SplitView.Content>
72     <Border CornerRadius="12 0 0 0"
73         Background="#2d2d2d">
74         <TransitioningContentControl Content="{Binding CurrentPage}" />
75     </Border>
76 </SplitView.Content>
77 </SplitView>
78 </Window>

```

Listing A.10.41: EasonEetwViewer/Views/MainWindow.axaml

```

1  using Avalonia.Controls;
2
3  namespace EasonEetwViewer.Views;
4  /// <summary>
5  /// Code-behind for SettingPageView.axaml
6  /// </summary>
7  public partial class SettingPageView : UserControl
8  {
9      /// <summary>
10     /// Initializes a new instance of the <see cref="SettingPageView"/> class.
11     /// </summary>
12     public SettingPageView()
13         => InitializeComponent();
14 }

```

Listing A.10.42: EasonEetwViewer/Views/SettingPageView.axaml.cs

```

1  using Avalonia.Controls;
2  using EasonEetwViewer.ViewModels;
3  using Microsoft.Extensions.DependencyInjection;
4
5  namespace EasonEetwViewer.Views;
6
7  // Adapted from https://github.com/dev-elian/DemoMapsui
8  /// <summary>
9  /// Code-behind for PastPageView.axaml
10 /// </summary>
11 public partial class PastPageView : UserControl
12 {
13     /// <summary>

```

```

14     /// Initializes a new instance of the <see cref="PastPageView"/> class.
15     /// </summary>
16     public PastPageView()
17     {
18         InitializeComponent();
19         MapControl.Map = App.Service!.GetRequiredService<PastPageViewModel>().Map;
20     }
21 }
```

Listing A.10.43: EasonEetwViewer/Views/PastPageView.axaml.cs

```

1  <UserControl xmlns="https://github.com/avaloniaui"
2      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
3      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
4      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5      xmlns:vm="using:EasonEetwViewer.ViewModels"
6      xmlns:m="using:EasonEetwViewer.Models.SettingPage"
7      xmlns:conv="using:EasonEetwViewer.Converters"
8      xmlns:lang="using:EasonEetwViewer.Lang"
9      mc:Ignorable="d"
10     d:DesignWidth="800"
11     d:DesignHeight="450"
12     x:Class="EasonEetwViewer.Views.SettingPageView"
13     x:DataType="vm:SettingPageViewModel">
14
15     <Design.DataContext>
16         <vm:SettingPageViewModel/>
17     </Design.DataContext>
18
19     <UserControl.Resources>
20         <conv:EnumDisplayTextConverter x:Key="EnumDisplayTextConverter"/>
21         <conv:CultureInfoConverter x:Key="LanguageChoiceConverter"/>
22     </UserControl.Resources>
23
24     <ScrollViewer>
25         <StackPanel Margin="20">
26
27             <!--Header-->
28
29             <TextBlock Margin="0 20"
30                     FontSize="20"
31                     FontWeight="Bold"
32                     Text="{x:Static lang:SettingPageResources.HeaderText}"/>
33
34             <!--General Text-->
35
36             <TextBlock Margin="0 20"
37                     FontSize="18"
38                     FontWeight="Bold"
39                     Text="{x:Static lang:SettingPageResources.GeneralTextHeader}"/>
40             <TextBlock Margin="10 0"
41                     FontSize="15"
42                     TextWrapping="Wrap"
43                     LineHeight="25">
44                 <Run Text="{x:Static lang:SettingPageResources.GeneralTextBullet}"/>
45                 <Run Text="{x:Static lang:SettingPageResources.GeneralText1}"/>
46                 <LineBreak/>
47                 <Run Text="{x:Static lang:SettingPageResources.GeneralTextBullet}"/>
```

```
48     <Run Text="{x:Static lang:SettingPageResources.GeneralText2}" />
49     <LineBreak/>
50     <Run Text="{x:Static lang:SettingPageResources.GeneralTextBullet}" />
51     <Run Text="{x:Static lang:SettingPageResources.GeneralText3}" />
52     <LineBreak/>
53     <Run Text="{x:Static lang:SettingPageResources.GeneralTextBullet}" />
54     <Run Text="{x:Static lang:SettingPageResources.GeneralText4}" />
55     <LineBreak/>
56     <Run Text="{x:Static lang:SettingPageResources.GeneralTextBullet}" />
57     <Run Text="{x:Static lang:SettingPageResources.GeneralText5}" />
58     <LineBreak/>
59     <Run Text="{x:Static lang:SettingPageResources.GeneralTextBullet}" />
60     <Run Text="{x:Static lang:SettingPageResources.GeneralText6}" />
61 </TextBlock>
62
63 <!--DM-D.S.S. Authorisation-->
64
65 <StackPanel Margin="0 0"
66             Orientation="Horizontal">
67     <TextBlock Margin="0 20"
68                 FontSize="18"
69                 FontWeight="Bold"
70                 Text="{x:Static lang:SettingPageResources.DmdssText}" />
71     <TextBlock Margin="10 0"
72                 FontSize="15"
73                 VerticalAlignment="Center">
74         <Run Text="{x:Static
75             → lang:SettingPageResources.AuthenticationStatusText}" />
76         <Run Text="{Binding AuthenticationStatus,
77                         Converter={StaticResource EnumDisplayTextConverter}}"/>
78     </TextBlock>
79 </StackPanel>
80
81 <!--Api Key-->
82
83 <StackPanel Margin="10 0"
84             Orientation="Horizontal">
85     <TextBlock Margin="10 0"
86                 FontSize="15"
87                 VerticalAlignment="Center"
88                 Text="{x:Static lang:SettingPageResources.ApiText}" />
89     <TextBox Margin="10 0"
90                 FontSize="15"
91                 Width="410"
92                 PasswordChar="*"
93                 Watermark="AKexxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
94                 Text="{Binding ApiKeyText}" />
95     <Button IsEnabled="{Binding ApiKeyButtonEnabled}"
96                 Margin="10 0"
97                 FontSize="15"
98                 Command="{Binding ApiKeyButtonCommand}"
99                 Content="{x:Static
100             → lang:SettingPageResources.ApiTextConfirm}" />
101    <TextBlock Margin="10 0"
102                 FontSize="15"
103                 VerticalAlignment="Center"
104                 Text="{x:Static
105             → lang:SettingPageResources.ApiTextConfirmed}"
106                 IsVisible="{Binding ApiKeyConfirmed}" />
```

```
104 </StackPanel>
105
106 <!--OAuth-->
107
108 <StackPanel Margin="10"
109     Orientation="Horizontal">
110     <Button Margin="0 0"
111         FontSize="15"
112         Content="{Binding OAuthButtonText}"
113         Command="{Binding OAuthButtonCommand}"/>
114     <TextBlock Margin="10 0"
115         FontSize="15"
116         VerticalAlignment="Center"
117         Text="{x:Static
118             → lang:SettingPageResources.OAuthConnectedText}"
119             IsVisible="{Binding OAuthConnected}"/>
120 </StackPanel>
121
122 <!--WebSocket Settings-->
123
124 <StackPanel Margin="0"
125     Orientation="Horizontal">
126     <TextBlock Margin="0 20"
127         FontSize="18"
128         FontWeight="Bold"
129         Text="{x:Static lang:SettingPageResources.WebSocketText}"/>
130     <Button Margin="10 0"
131         FontSize="15"
132         Content="{Binding WebSocketButtonText}"
133         Command="{Binding WebSocketButtonCommand}"/>
134     <Button Margin="10 0"
135         FontSize="15"
136         Content="{x:Static
137             → lang:SettingPageResources.WebSocketTextRefreshList}"
138             Command="{Binding WebSocketRefreshCommand}"/>
139 </StackPanel>
140
141 <!--WebSocket Connections-->
142
143 <DataGrid Margin="20"
144     GridLinesVisibility="All"
145     CanUserSortColumns="False"
146     BorderThickness="1"
147     BorderBrush="Gray"
148     ItemsSource="{Binding WebSocketConnections}">
149     <DataGrid.Columns>
150         <DataGridTextColumn Header="{x:Static
151             → lang:SettingPageResources.WebSocketListTextWebSocketId}"
152                 Binding="{Binding WebSocketId}"/>
153         <DataGridTextColumn Header="{x:Static
154             → lang:SettingPageResources.WebSocketListTextApplicationName}"
155                 Binding="{Binding ApplicationName,
156                     FallbackValue={x:Static
157                         → lang:SettingPageResources.WebSocketListTe
158                         → xtApplicationNameUnknown},
159                     TargetNullValue={x:Static
160                         → lang:SettingPageResources.WebSocketListTe
161                         → xtApplicationNameUnknown}}"/>
```

```
154 <DataGridTextColumn Header="{x:Static  
155     lang:SettingPageResources.WebSocketListTextConnectionTime}"  
156         Binding="{Binding StartTime,  
157             StringFormat='yyyy/MM/dd HH:mm:ss'}"/>  
158 <DataGridTemplateColumn Header="{x:Static  
159     lang:SettingPageResources.WebSocketListTextDisconnect}">  
160     <DataTemplate DataType="{x:Type  
161         m:IWebSocketConnectionTemplate}">  
162         <Button HorizontalAlignment="Center"  
163             IsEnabled="{Binding IsEnabled}"  
164             Content="{x:Static lang:SettingPageResources.WebS  
165                 ocketListTextDisconnect}"  
166             Command="{Binding DisconnectCommand}"/>  
167     </DataTemplate>  
168 </DataGridTemplateColumn>  
169 </DataGrid.Columns>  
170 </DataGrid>  
171  
172 <!--Kmoni Settings-->  
173  
174 <TextBlock Margin="0 20"  
175     FontSize="18"  
176     FontWeight="Bold"  
177     Text="{x:Static lang:SettingPageResources.KmoniText}"/>  
178  
179 <StackPanel Margin="10 0"  
180     Orientation="Horizontal">  
181     <TextBlock Margin="10 0"  
182         FontSize="15"  
183         VerticalAlignment="Center"  
184         Text="{x:Static  
185             lang:SettingPageResources.KmoniSensorText}"/>  
186     <ComboBox ItemsSource="{Binding SensorTypeChoices}"  
187         SelectedItem="{Binding KmoniSettingsHelper.SensorChoice}">  
188         <ComboBox.ItemTemplate>  
189             <DataTemplate>  
190                 <TextBlock Text="{Binding Converter={StaticResource  
191                     EnumDisplayTextConverter}}"/>  
192             </DataTemplate>  
193         </ComboBox.ItemTemplate>  
194     </ComboBox>  
195 </StackPanel>  
196  
197 <StackPanel Margin="10"  
198     Orientation="Horizontal">  
199     <TextBlock Margin="10 0"  
200         FontSize="15"  
201         VerticalAlignment="Center"  
202         Text="{x:Static  
203             lang:SettingPageResources.KmoniMeasurementText}"/>  
204     <ComboBox ItemsSource="{Binding MeasurementTypeChoices}"  
205         SelectedItem="{Binding  
206             KmoniSettingsHelper.MeasurementChoice}">  
207         <ComboBox.ItemTemplate>  
208             <DataTemplate>  
209                 <TextBlock Text="{Binding Converter={StaticResource  
210                     EnumDisplayTextConverter}}"/>  
211             </DataTemplate>  
212         </ComboBox.ItemTemplate>
```

```

204             </ComboBox>
205         </StackPanel>
206
207         <!--Language Settings-->
208
209         <TextBlock Margin="0 20"
210             FontSize="18"
211             FontWeight="Bold"
212             Text="{x:Static lang:SettingPageResources.LangText}"/>
213
214         <ComboBox ItemsSource="{Binding LanguageChoices}"
215             SelectedItem="{Binding LanguageChoice}">
216             <ComboBox.ItemTemplate>
217                 <DataTemplate>
218                     <TextBlock Text="{Binding Converter={StaticResource
219                         LanguageChoiceConverter}}"/>
220                 </DataTemplate>
221             </ComboBox.ItemTemplate>
222         </ComboBox>
223     </StackPanel>
224 </ScrollViewer>
225 </UserControl>

```

Listing A.10.44: EasonEetwViewer/Views/SettingPageView.xaml

```

1  using Avalonia.Controls;
2  using EasonEetwViewer.ViewModels;
3  using Microsoft.Extensions.DependencyInjection;
4
5  namespace EasonEetwViewer.Views;
6
7  // Adapted from https://github.com/dev-elian/DemoMapsui
8  /// <summary>
9  /// Code-behind for RealtimePageView.xaml
10 /// </summary>
11 public partial class RealtimePageView : UserControl
12 {
13     /// <summary>
14     /// Initializes a new instance of the <see cref="RealtimePageView"/> class.
15     /// </summary>
16     public RealtimePageView()
17     {
18         InitializeComponent();
19         MapControl.Map = App.Service!.GetRequiredService<RealtimePageViewModel>().Map;
20     }
21 }

```

Listing A.10.45: EasonEetwViewer/Views/RealTimePageView.xaml.cs

```

1  <UserControl xmlns="https://github.com/avaloniaui"
2      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
3      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
4      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5      xmlns:vm="using:EasonEetwViewer.ViewModels"
6      xmlns:m="using:EasonEetwViewer.Models.PastPage"
7      xmlns:conv="using:EasonEetwViewer.Converters"
8      xmlns:lang="using:EasonEetwViewer.Lang"

```

```
9      xmlns:mapsui="using:Mapsui.UI.Avalonia"
10     mc:Ignorable="d"
11     d:DesignWidth="800"
12     d:DesignHeight="450"
13     x:Class="EasonEetwViewer.Views.PastPageView"
14     x:DataType="vm:PastPageViewModel">
15
16     <Design.DataContext>
17         <vm:PastPageViewModel/>
18     </Design.DataContext>
19
20     <UserControl.Resources>
21         <conv:EnumDisplayColourConverter x:Key="EnumDisplayColourConverter"/>
22         <conv:EnumDisplayTextConverter x:Key="EnumDisplayTextConverter"/>
23         <conv:UnitValueDisplayUnitConverter x:Key="UnitValueDisplayUnitConverter"/>
24         <conv:UnitValueDisplayValueConverter x:Key="UnitValueDisplayValueConverter"/>
25     </UserControl.Resources>
26
27     <Grid>
28         <Grid.ColumnDefinitions>
29             <ColumnDefinition Width="300"/>
30             <ColumnDefinition Width="*"/>
31             <ColumnDefinition Width="300"/>
32         </Grid.ColumnDefinitions>
33
34         <!--Earthquake Info Display-->
35
36         <Grid Grid.Column="0"
37             Grid.Row="0"
38             IsVisible="{Binding EarthquakeDetails,
39             Converter={x:Static ObjectConverters.IsNotNull}}">
40             <Grid.RowDefinitions>
41                 <RowDefinition Height="Auto"/>
42                 <RowDefinition Height="*"/>
43             </Grid.RowDefinitions>
44
45         <!--Earthquake Info Display-->
46
47         <StackPanel Grid.Column="0"
48             Grid.Row="0">
49
50             <!--Jump to Yahoo Weather Button-->
51
52             <Button HorizontalAlignment="Center"
53                 FontSize="16"
54                 Content="{x:Static lang:PastPageResources.YahooText}"
55                 Command="{Binding JumpYahooWebpageCommand}"
56                 Margin="5"/>
57
58             <!--Intensity Display-->
59
60             <TextBlock HorizontalAlignment="Left"
61                 FontSize="16"
62                 FontWeight="SemiBold"
63                 Text="{x:Static
64                     lang:EarthquakeResources.DetailsMaxIntensityText}"
65                 Margin="5 5"/>
66             <TextBlock HorizontalAlignment="Right"
67                 FontSize="52"
```

```
67     FontWeight="ExtraBold"
68     Text="{Binding EarthquakeDetails.Intensity,
69     Converter={StaticResource EnumDisplayTextConverter},
70     FallbackValue={x:Static
71       ↳ lang:EarthquakeResources.IntensityTextUnknown},
72     TargetNullValue={x:Static
73       ↳ lang:EarthquakeResources.IntensityTextUnknown}}"
74     Foreground="{Binding EarthquakeDetails.Intensity,
75     Converter={StaticResource EnumDisplayColourConverter},
76     FallbackValue={x:Static
77       ↳ lang:EarthquakeResources.IntensityColourUnknown},
78     TargetNullValue={x:Static
79       ↳ lang:EarthquakeResources.IntensityColourUnknown}}"
80     Margin="5 0"/>
81
82     <!--Origin Time Display-->
83
84     <TextBlock HorizontalAlignment="Left"
85       FontSize="14"
86       FontWeight="SemiBold"
87       Text="{x:Static lang:EarthquakeResources.DetailsTimeText}"
88       Margin="5 0"/>
89     <TextBlock HorizontalAlignment="Right"
90       FontSize="24"
91       FontWeight="SemiBold"
92       Text="{Binding EarthquakeDetails.OriginTime,
93         StringFormat='yyyy/MM/dd HH:mm',
94         FallbackValue={x:Static
95           ↳ lang:EarthquakeResources.UnknownText},
96         TargetNullValue={x:Static
97           ↳ lang:EarthquakeResources.UnknownText}}"
98       Margin="5 0"/>
99
100    <!--Hypocentre Position Display-->
101
102    <TextBlock HorizontalAlignment="Left"
103      FontSize="14"
104      FontWeight="SemiBold"
105      Text="{x:Static
106        ↳ lang:EarthquakeResources.DetailsHypocentreText}"
107      Margin="5 0"/>
108    <TextBlock HorizontalAlignment="Right"
109      FontSize="24"
110      FontWeight="SemiBold"
111      Text="{Binding EarthquakeDetails.Hypocentre.Name,
112        FallbackValue={x:Static
113          ↳ lang:EarthquakeResources.UnknownText},
114        TargetNullValue={x:Static
115          ↳ lang:EarthquakeResources.UnknownText}}"
116      Margin="5 0"/>
117
118    <!--Update Time Display-->
119
120    <TextBlock HorizontalAlignment="Left"
121      FontSize="14"
122      FontWeight="SemiBold"
123      Text="{x:Static
124        ↳ lang:EarthquakeResources.DetailsLastUpdatedText}"
125      Margin="5 0"/>
```

```
116     <TextBlock HorizontalAlignment="Right"
117         FontSize="18"
118         FontWeight="SemiBold"
119         Text="{Binding EarthquakeDetails.LastUpdated,
120             StringFormat='yyyy/MM/dd HH:mm',
121             FallbackValue={x:Static
122                 → lang:EarthquakeResources.UnknownText},
123                 TargetNullValue={x:Static
124                     → lang:EarthquakeResources.UnknownText}}"
125         Margin="5 0"/>
126
126     <!--Depth Display-->
127
127     <TextBlock HorizontalAlignment="Left"
128         FontSize="14"
129         FontWeight="SemiBold"
130         Text="{x:Static lang:EarthquakeResources.DetailsDepthText}"
131         Margin="5 0"/>
132     <TextBlock HorizontalAlignment="Right"
133         FontWeight="SemiBold"
134         Margin="5 0">
135         <Run FontSize="24"
136             Text="{Binding EarthquakeDetails.Hypocentre.Depth,
137                 Converter={StaticResource UnitValueDisplayValueConverter},
138                 FallbackValue={x:Static
139                     → lang:EarthquakeResources.UnknownText},
140                 TargetNullValue={x:Static
141                     → lang:EarthquakeResources.UnknownText}}"/>
140         <Run FontSize="18"
141             Text="{Binding EarthquakeDetails.Hypocentre.Depth,
142                 Converter={StaticResource UnitValueDisplayUnitConverter},
143                 FallbackValue={},
144                 TargetNullValue={}"/>
145     </TextBlock>
146
146     <!--Magnitude Display-->
147
147     <TextBlock HorizontalAlignment="Left"
148         FontSize="14"
149         FontWeight="SemiBold"
150         Text="{x:Static
151             → lang:EarthquakeResources.DetailsMagnitudeText}"
152         Margin="5 0"/>
153     <TextBlock HorizontalAlignment="Right"
154         FontWeight="SemiBold"
155         Margin="5 0">
156         <Run FontSize="18"
157             Text="{Binding EarthquakeDetails.Magnitude,
158                 Converter={StaticResource UnitValueDisplayUnitConverter},
159                 FallbackValue={x:Static
160                     → lang:EarthquakeResources.MagnitudeUnitDefault},
161                 TargetNullValue={x:Static
162                     → lang:EarthquakeResources.MagnitudeUnitDefault}}"/>
162         <Run FontSize="24"
163             Text="{Binding EarthquakeDetails.Magnitude,
164                 Converter={StaticResource UnitValueDisplayValueConverter},
165                 FallbackValue={x:Static
165                     → lang:EarthquakeResources.UnknownText}},
```

```
166             TargetNullValue={x:Static
167                 ↳ lang:EarthquakeResources.UnknownText}}"/>
168         </TextBlock>
169
170         <TextBlock HorizontalAlignment="Left"
171             FontSize="14"
172             Text="{Binding EarthquakeDetails.InformationalText}"
173             TextWrapping="Wrap"
174             Margin="8"/>
175     </StackPanel>
176
177     <!--Intensity Detail Tree-->
178
179     <ScrollViewer Grid.Column="0"
180                 Grid.Row="1"
181                 Margin="0 5">
182         <ItemsControl ItemsSource="{Binding EarthquakeDetails.IntensityTree}">
183             <ItemsControl.ItemTemplate>
184                 <DataTemplate>
185                     <StackPanel>
186
187                         <!--Intensity Text-->
188
189                         <TextBlock HorizontalAlignment="Stretch"
190                             TextAlignment="Left"
191                             FontSize="14"
192                             FontWeight="SemiBold"
193                             Foreground="{Binding Intensity,
194                             Converter={StaticResource
195                                 ↳ EnumDisplayColourConverter}}"
196                             Margin="5">
197                             <Run Text="{x:Static, lang:EarthquakeResources.De
198                                 ↳ tailsIntensityText}"/>
199                             <Run Text="{Binding Intensity,
200                             Converter={StaticResource
201                                 ↳ EnumDisplayTextConverter}}"/>
202                         </TextBlock>
203
204                         <!--Observation Point Tree-->
205
206                         <TreeView ItemsSource="{Binding PositionNodes}">
207                             <TreeView.ItemTemplate>
208                                 <TreeDataTemplate ItemsSource="{Binding
209                                     ↳ SubNodes}">
210                                     <TextBlock Text="{Binding Name}"/>
211                                 </TreeDataTemplate>
212                             </TreeView.ItemTemplate>
213                         </TreeView>
214                     </StackPanel>
215                 </DataTemplate>
216             </ItemsControl.ItemTemplate>
217         </ItemsControl>
218     </ScrollViewer>
219 </Grid>
220
221 <Grid Grid.Column="1"
222       Grid.Row="0">
223     <DockPanel>
224         <mapsui:MapControl x:Name="MapControl"/>
```

```
220     </DockPanel>
221     <Canvas>
222     </Canvas>
223   </Grid>
224
225   <!--Past Earthquake List-->
226
227   <Grid Grid.Column="2"
228         Grid.Row="0">
229     <Grid.RowDefinitions>
230       <RowDefinition Height="Auto"/>
231       <RowDefinition Height="*"/>
232       <RowDefinition Height="Auto"/>
233     </Grid.RowDefinitions>
234
235   <!--Refresh List Button-->
236
237   <Button Grid.Column="0"
238         Grid.Row="0"
239         Content="{x:Static lang:PastPageResources.RefreshListText}"
240         Command="{Binding RefreshEarthquakeListCommand}"
241         HorizontalAlignment="Center"
242         Margin="10"/>
243
244   <!--Past Earthquake List-->
245
246   <ListBox Grid.Column="0"
247             Grid.Row="1"
248             Padding="0"
249             ItemsSource="{Binding EarthquakeList}"
250             SelectedItem="{Binding SelectedEarthquake}">
251     <ListBox.ItemTemplate>
252       <DataTemplate DataType="{x:Type m:EarthquakeItemTemplate}">
253         <Grid>
254           <Grid.ColumnDefinitions>
255             <ColumnDefinition Width="40"/>
256             <ColumnDefinition Width="*"/>
257             <ColumnDefinition Width="65"/>
258           </Grid.ColumnDefinitions>
259           <Grid.RowDefinitions>
260             <RowDefinition Height="2*"/>
261             <RowDefinition Height="3*"/>
262           </Grid.RowDefinitions>
263
264         <TextBlock Text="{Binding Intensity,
265                         Converter={StaticResource
266                           → EnumDisplayTextConverter},
267                         FallbackValue={x:Static lang:EarthquakeResourc
268                           → es.IntensityTextUnknown},
269                         TargetNullValue={x:Static lang:EarthquakeResou
270                           → rces.IntensityTextUnknown}}"
271                         VerticalAlignment="Center"
272                         HorizontalAlignment="Center"
273                         FontSize="28"
274                         FontWeight="ExtraBold"
275                         Foreground="{Binding Intensity,
276                             Converter={StaticResource
277                               → EnumDisplayColourConverter},
```

```
274     FallbackValue={x:Static lang:EarthquakeResourc_  
275         ↵ es.IntensityColourUnknown},  
276     TargetNullValue={x:Static lang:EarthquakeResou_  
277         ↵ rces.IntensityColourUnknown}}}"  
278     Grid.Column="0"  
279     Grid.Row="0"  
280     Grid.RowSpan="2"/>  
281  
282     <TextBlock Text="{Binding OriginTime,  
283         StringFormat='MM/dd HH:mm',  
284         FallbackValue={x:Static  
285             ↵ lang:EarthquakeResources.UnknownText},  
286         TargetNullValue={x:Static  
287             ↵ lang:EarthquakeResources.UnknownText}}"  
288         HorizontalAlignment="Left"  
289         VerticalAlignment="Bottom"  
290         Grid.Column="1"  
291         Grid.Row="0"/>  
292  
293     <TextBlock Text="{Binding Hypocentre.Name,  
294         FallbackValue={x:Static  
295             ↵ lang:EarthquakeResources.UnknownText},  
296         TargetNullValue={x:Static  
297             ↵ lang:EarthquakeResources.UnknownText}}"  
298         HorizontalAlignment="Left"  
299         VerticalAlignment="Bottom"  
300         FontSize="18"  
301         FontWeight="SemiBold"  
302         Grid.Column="1"  
303         Grid.Row="1"/>  
304  
305     <TextBlock HorizontalAlignment="Right"  
306         VerticalAlignment="Bottom"  
307         Grid.Column="2"  
308         Grid.Row="0">  
309         <Run Text="{Binding Hypocentre.Depth,  
310             Converter={StaticResource  
311                 ↵ UnitValueDisplayValueConverter},  
312             FallbackValue={x:Static  
313                 ↵ lang:EarthquakeResources.UnknownText},  
314             TargetNullValue={x:Static  
315                 ↵ lang:EarthquakeResources.UnknownText}}"  
316             FontWeight="SemiBold"/>  
317         <Run Text="{Binding Hypocentre.Depth,  
318             Converter={StaticResource  
319                 ↵ UnitValueDisplayUnitConverter},  
320             FallbackValue={},  
321             TargetNullValue={}"}"  
322             FontWeight="SemiLight"/>  
323     </TextBlock>  
324  
325     <TextBlock HorizontalAlignment="Right"  
326         VerticalAlignment="Bottom"  
327         Grid.Column="2"  
328         Grid.Row="1">  
329         <Run Text="{Binding Magnitude,  
330             Converter={StaticResource  
331                 ↵ UnitValueDisplayUnitConverter},  
332             FallbackValue=0.0, TargetNullValue=0.0}"/>
```

```
321             FallbackValue={x:Static lang:EarthquakeResources.MagnitudeUnitDefault},
322             TargetNullValue={x:Static lang:EarthquakeResources.UnknownText},
323             FontWeight="SemiLight"/>
324         <Run Text="{Binding Magnitude,
325             Converter={StaticResource UnitValueDisplayValueConverter},
326             FallbackValue={x:Static lang:EarthquakeResources.UnknownText},
327             TargetNullValue={x:Static lang:EarthquakeResources.UnknownText}}"
328             FontSize="18"
329             FontWeight="SemiBold"/>
330     </TextBlock>
331   </Grid>
332 </DataTemplate>
333 </ListBox.ItemTemplate>
334 </ListBox>
335
336
337 <!--Load 10 More Earthquakes-->
338
339 <Button Grid.Column="0"
340         Grid.Row="2"
341         Content="{x:Static lang:PastPageResources.LoadMoreText}"
342         Command="{Binding LoadExtraEarthquakesCommand}"
343         IsEnabled="{Binding IsLoadExtraEnabled}"
344         HorizontalAlignment="Center"
345         Margin="10"/>
346   </Grid>
347 </Grid>
348 </UserControl>
```

Listing A.10.46: EasonEetwViewer/Views/PastPageView.axaml

```
1  using Avalonia.Controls;
2
3  namespace EasonEetwViewer.Views;
4  ///<summary>
5  ///<summary>Code-behind for MainWindow.axaml</summary>
6  ///</summary>
7  public partial class MainWindow : Window
8  {
9      ///<summary>
10     ///<summary>Initializes a new instance of the <see cref="MainWindow"/> class.</summary>
11     ///</summary>
12     public MainWindow()
13         => InitializeComponent();
14 }
```

Listing A.10.47: EasonEetwViewer/Views/mainwindow.axaml.cs

```
1 <UserControl xmlns="https://github.com/avaloniaui"  
2     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
3     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
4      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5      xmlns:vm="using:EasonEetwViewer.ViewModels"
6      xmlns:m="using:EasonEetwViewer.Models"
7      xmlns:conv="using:EasonEetwViewer.Converters"
8      xmlns:lang="using:EasonEetwViewer.Lang"
9      xmlns:mapsui="using:Mapsui.UI.Avalonia"
10     mc:Ignorable="d"
11     d:DesignWidth="800"
12     d:DesignHeight="450"
13     x:Class="EasonEetwViewer.Views.RealtimePageView"
14     x:DataType="vm:RealtimePageViewModel">
15
16     <Design.DataContext>
17         <vm:RealtimePageViewModel/>
18     </Design.DataContext>
19
20     <UserControl.Resources>
21         <conv:EnumDisplayColourConverter x:Key="EnumDisplayColourConverter"/>
22         <conv:EnumDisplayTextConverter x:Key="EnumDisplayTextConverter"/>
23         <conv:UnitValueDisplayUnitConverter x:Key="UnitValueDisplayUnitConverter"/>
24         <conv:UnitValueDisplayValueConverter x:Key="UnitValueDisplayValueConverter"/>
25     </UserControl.Resources>
26
27     <Grid>
28         <DockPanel>
29             <mapsui:MapControl x:Name="MapControl"/>
30         </DockPanel>
31         <Canvas>
32             <!--Scale Display-->
33             <Border Canvas.Bottom="0"
34                 Canvas.Left="0"
35                 CornerRadius="0 5 0 0"
36                 Background="{StaticResource SystemChromeBlackMediumColor}">
37                 <Canvas Margin="10"
38                     Height="320"
39                     Width="75">
40                     <Image Canvas.Bottom="0"
41                         Canvas.Left="0"
42                         Source="avares://EasonEetwViewer/Assets/colourScale.png"/>
43                     <TextBlock Canvas.Bottom="0"
44                         Canvas.Left="40"
45                         FontFamily="Bold"
46                         Text="{Binding DataLegendValues[0]}"
47                         FontSize="12"/>
48                     <TextBlock Canvas.Bottom="28"
49                         Canvas.Left="40"
50                         FontFamily="Bold"
51                         Text="{Binding DataLegendValues[1]}"
52                         FontSize="12"/>
53                     <TextBlock Canvas.Bottom="56"
54                         Canvas.Left="40"
55                         FontFamily="Bold"
56                         Text="{Binding DataLegendValues[2]}"
57                         FontSize="12"/>
58                     <TextBlock Canvas.Bottom="84"
59                         Canvas.Left="40"
60                         FontFamily="Bold"
61                         Text="{Binding DataLegendValues[3]}"
62                         FontSize="12"/>
```

```
63 <TextBlock Canvas.Bottom="112"  
64     Canvas.Left="40"  
65     FontFamily="Bold"  
66     Text="{Binding DataLegendValues[4]}"  
67     FontSize="12"/>  
68 <TextBlock Canvas.Bottom="140"  
69     Canvas.Left="40"  
70     FontFamily="Bold"  
71     Text="{Binding DataLegendValues[5]}"  
72     FontSize="12"/>  
73 <TextBlock Canvas.Bottom="168"  
74     Canvas.Left="40"  
75     FontFamily="Bold"  
76     Text="{Binding DataLegendValues[6]}"  
77     FontSize="12"/>  
78 <TextBlock Canvas.Bottom="196"  
79     Canvas.Left="40"  
80     FontFamily="Bold"  
81     Text="{Binding DataLegendValues[7]}"  
82     FontSize="12"/>  
83 <TextBlock Canvas.Bottom="224"  
84     Canvas.Left="40"  
85     FontFamily="Bold"  
86     Text="{Binding DataLegendValues[8]}"  
87     FontSize="12"/>  
88 <TextBlock Canvas.Bottom="252"  
89     Canvas.Left="40"  
90     FontFamily="Bold"  
91     Text="{Binding DataLegendValues[9]}"  
92     FontSize="12"/>  
93 <TextBlock Canvas.Bottom="280"  
94     Canvas.Left="40"  
95     FontFamily="Bold"  
96     Text="{Binding DataLegendValues[10]}"  
97     FontSize="12"/>  
98 <TextBlock Canvas.Bottom="300"  
99     Canvas.Left="0"  
100    FontFamily="Bold"  
101    Text="{Binding DataLegendText}"/>  
102   </Canvas>  
103 </Border>  
104  
105 <!--Time Display-->  
106 <Border Canvas.Bottom="0"  
107     Canvas.Left="95"  
108     Background="{StaticResource SystemChromeBlackMediumColor}"  
109     CornerRadius="0 5 0 0">  
110     <TextBlock FontSize="20"  
111         FontWeight="Bold"  
112         Text="{Binding TimeDisplay, StringFormat='yyyy/MM/dd  
113             HH:mm:ss'}"/>  
114 </Border>  
115  
116 <!--Eew Info Display-->  
117 <Border Canvas.Top="0"  
118     Canvas.Left="0"  
119     IsVisible="{Binding CurrentDisplayEew,  
120     Converter={x:Static ObjectConverters.IsNotNull}}"
```

```
121     Background="{StaticResource SystemChromeBlackMediumColor}"
122     Width="400"
123     CornerRadius="0 0 10 0">
124     <StackPanel>
125
126         <!--Eew Colour Display-->
127
128         <Border Background="{Binding CurrentDisplayEew.EewWarningType,
129             Converter={StaticResource EnumDisplayColourConverter},
130             TargetNullValue={x:Static
131                 ↪ lang:RealtimePageResources.EewColourCancelled},
132             FallbackValue={x:Static
133                 ↪ lang:RealtimePageResources.EewColourCancelled}}"
134             Height="10">
135             </Border>
136
137         <Border Background="{Binding CurrentDisplayEew.EewWarningType,
138             Converter={StaticResource EnumDisplayColourConverter},
139             TargetNullValue={x:Static
140                 ↪ lang:RealtimePageResources.EewColourCancelled},
141             FallbackValue={x:Static
142                 ↪ lang:RealtimePageResources.EewColourCancelled}}"
143             Margin="0 5"
144             Height="5">
145             </Border>
146
147             <!--Eew Serial Display-->
148
149             <TextBlock FontWeight="SemiBold"
150                 Margin="5 5">
151                 <Run FontSize="24"
152                     Text="{x:Static lang:RealtimePageResources.EewText}"/>
153                 <Run FontSize="24"
154                     Text="{x:Static
155                         ↪ lang:RealtimePageResources.RoundBracketsStart}"/>
156                 <Run FontSize="24"
157                     Text="{Binding CurrentDisplayEew.EewWarningType,
158                         Converter={StaticResource EnumDisplayTextConverter},
159                         TargetNullValue={x:Static
160                             ↪ lang:RealtimePageResources.EewTextCancelled},
161                         FallbackValue={x:Static
162                             ↪ lang:RealtimePageResources.EewTextCancelled}}"/>
163                 <Run FontSize="24"
164                     Text="{x:Static
165                         ↪ lang:RealtimePageResources.RoundBracketsEnd}"/>
166                 <Run FontSize="18"
167                     Text="{x:Static
168                         ↪ lang:RealtimePageResources.EewNumberTextBefore}"/>
169                 <Run FontSize="18"
170                     Text="{Binding CurrentDisplayEew.Serial}"/>
171                 <Run FontSize="18"
172                     Text="{x:Static
173                         ↪ lang:RealtimePageResources.EewNumberTextAfter}"/>
174             </TextBlock>
175
176             <!--Intensity Display-->
177
178             <TextBlock HorizontalAlignment="Left"
179                         FontSize="16"
```

```
170     FontWeight="SemiBold"
171     Text="{x:Static
172         → lang:EarthquakeResources.DetailsMaxIntensityText}"
173     Margin="5 5"/>
174
175     <TextBlock HorizontalAlignment="Right"
176         Margin="5 0"
177         FontWeight="ExtraBold">
178         <Run FontSize="20"
179             Text="{Binding CurrentDisplayEew.BeforeIntensityText}"/>
180         <Run FontSize="52"
181             Text="{Binding CurrentDisplayEew.MaxIntensityEnum,
182             Converter={StaticResource EnumDisplayTextConverter},
183             TargetNullValue={x:Static
184                 → lang:EarthquakeResources.IntensityTextUnknown},
185             FallbackValue={x:Static
186                 → lang:EarthquakeResources.IntensityTextUnknown}}"
187             Foreground="{Binding CurrentDisplayEew.MaxIntensityEnum,
188             Converter={StaticResource EnumDisplayColourConverter},
189             TargetNullValue={x:Static
190                 → lang:EarthquakeResources.IntensityColourUnknown},
191             FallbackValue={x:Static
192                 → lang:EarthquakeResources.IntensityColourUnknown}}"/>
193         <Run FontSize="20"
194             Text="{Binding CurrentDisplayEew.AfterIntensityText}"/>
195     </TextBlock>
196
197     <!--Origin time Display-->
198
199     <TextBlock HorizontalAlignment="Left"
200         FontSize="14"
201         FontWeight="SemiBold"
202         Text="{x:Static
203             → lang:EarthquakeResources.DetailsTimeText}"
204         Margin="5 0"/>
205     <TextBlock HorizontalAlignment="Right"
206         FontSize="24"
207         FontWeight="SemiBold"
208         Text="{Binding CurrentDisplayEew.Earthquake.OriginTime,
209             StringFormat='yyyy/MM/dd HH:mm:ss',
210             FallbackValue={x:Static
211                 → lang:EarthquakeResources.UnknownText},
212             TargetNullValue={x:Static
213                 → lang:EarthquakeResources.UnknownText}}"
214         Margin="5 0"/>
215
216     <!--Hypocentre Position Display-->
217
218     <TextBlock HorizontalAlignment="Left"
219         FontSize="14"
220         FontWeight="SemiBold"
221         Text="{x:Static
222             → lang:EarthquakeResources.DetailsHypocentreText}"
223         Margin="5 0"/>
224     <TextBlock HorizontalAlignment="Right"
225         FontSize="24"
226         FontWeight="SemiBold"
227         Text="{Binding
228             → CurrentDisplayEew.Earthquake.Hypocentre.Name,
```

```
219             FallbackValue={x:Static
220                 ↳ lang:EarthquakeResources.UnknownText},
221             TargetNullValue={x:Static
222                 ↳ lang:EarthquakeResources.UnknownText}}"
223             Margin="5 0"/>
224
225             <!--Assumed Hypocentre Text-->
226
227             <TextBlock HorizontalAlignment="Left"
228                 FontSize="18"
229                 FontWeight="SemiBold"
230                 Text="{x:Static
231                     ↳ lang:RealtimePageResources.EewAssumedSpecificText}"
232                 TextWrapping="Wrap"
233                 IsVisible="{Binding
234                     ↳ CurrentDisplayEew.IsAssumedHypocentre}"
235                 Margin="5 5"/>
236
237             <!--Depth Display-->
238
239             <TextBlock HorizontalAlignment="Left"
240                 FontSize="14"
241                 FontWeight="SemiBold"
242                 Text="{x:Static
243                     ↳ lang:EarthquakeResources.DetailsDepthText}"
244                 IsVisible="{Binding
245                     ↳ !CurrentDisplayEew.IsAssumedHypocentre}"
246                 Margin="5 0"/>
247             <TextBlock HorizontalAlignment="Right"
248                 FontWeight="SemiBold"
249                 Margin="5 0"
250                 IsVisible="{Binding
251                     ↳ !CurrentDisplayEew.IsAssumedHypocentre}">
252                 <Run FontSize="24"
253                     Text="{Binding
254                         ↳ CurrentDisplayEew.Earthquake.Hypocentre.Depth,
255                         Converter={StaticResource
256                             ↳ UnitValueDisplayValueConverter},
257                         FallbackValue={x:Static
258                             ↳ lang:EarthquakeResources.UnknownText},
259                         TargetNullValue={x:Static
260                             ↳ lang:EarthquakeResources.UnknownText}}"/>
261                 <Run FontSize="18"
262                     Text="{Binding
263                         ↳ CurrentDisplayEew.Earthquake.Hypocentre.Depth,
264                         Converter={StaticResource UnitValueDisplayUnitConverter},
265                         FallbackValue={},
266                         TargetNullValue={}"/>
267             </TextBlock>
268
269             <!--Magnitude Display-->
270
271             <TextBlock HorizontalAlignment="Left"
272                 FontSize="14"
273                 FontWeight="SemiBold"
274                 Text="{x:Static
275                     ↳ lang:EarthquakeResources.DetailsMagnitudeText}"
276                 IsVisible="{Binding
277                     ↳ !CurrentDisplayEew.IsAssumedHypocentre}"
```

```
264             Margin="5 0"/>
265         <TextBlock HorizontalAlignment="Right"
266             FontWeight="SemiBold"
267             Margin="5 0"
268             IsVisible="{Binding
269                 ↳ !CurrentDisplayEew.IsAssumedHypocentre}">
270             <Run FontSize="18"
271                 Text="{Binding CurrentDisplayEew.Earthquake.Magnitude,
272                 Converter={StaticResource UnitValueDisplayUnitConverter},
273                 FallbackValue={x:Static
274                     ↳ lang:EarthquakeResources.MagnitudeUnitDefault},
275                 TargetNullValue={x:Static
276                     ↳ lang:EarthquakeResources.MagnitudeUnitDefault}}"/>
277             <Run FontSize="24"
278                 Text="{Binding CurrentDisplayEew.Earthquake.Magnitude,
279                 Converter={StaticResource
280                     ↳ UnitValueDisplayValueConverter},
281                 FallbackValue={x:Static
282                     ↳ lang:EarthquakeResources.UnknownText},
283                 TargetNullValue={x:Static
284                     ↳ lang:EarthquakeResources.UnknownText}}"/>
285         </TextBlock>
286
287         <!--Precision Display-->
288
289         <TextBlock HorizontalAlignment="Left"
290             FontSize="14"
291             FontWeight="SemiBold"
292             Text="{x:Static
293                 ↳ lang:RealtimePageResources.EewPrecisionText}"
294             Margin="5 0"/>
295         <Grid Margin="5.0">
296             <Grid.RowDefinitions>
297                 <RowDefinition Height="Auto"/>
298                 <RowDefinition Height="Auto"/>
299                 <RowDefinition Height="Auto"/>
300                 <RowDefinition Height="Auto"/>
301                 <RowDefinition Height="Auto"/>
302             </Grid.RowDefinitions>
303             <Grid.ColumnDefinitions>
304                 <ColumnDefinition Width="Auto"/>
305                 <ColumnDefinition Width="*"/>
306             </Grid.ColumnDefinitions>
307             <TextBlock Grid.Row="0"
308                 Grid.Column="0"
309                 HorizontalAlignment="Left"
310                 FontWeight="Normal"
311                 FontSize="14"
312                 Text="{x:Static lang:RealtimePageResources.Accurac
313                     ↳ yHypocentrePosition}"/>
314             <TextBlock Grid.Row="1"
315                 Grid.Column="0"
316                 HorizontalAlignment="Left"
317                 FontWeight="Normal"
318                 FontSize="14"
319                 Text="{x:Static lang:RealtimePageResources.Accurac
320                     ↳ yHypocentreCoordinate}"/>
321             <TextBlock Grid.Row="2"
322                 Grid.Column="0"
```

```
314     HorizontalAlignment="Left"
315     FontWeight="Normal"
316     FontSize="14"
317     Text="{x:Static
318         <-- lang:RealtimePageResources.AccuracyDepth}"/>
319     <TextBlock Grid.Row="3"
320         Grid.Column="0"
321         HorizontalAlignment="Left"
322         FontWeight="Normal"
323         FontSize="14"
324         Text="{x:Static lang:RealtimePageResources.Accurac
325             <-- yMagnitude}"/>
326     <TextBlock Grid.Row="4"
327         Grid.Column="0"
328         HorizontalAlignment="Left"
329         FontWeight="Normal"
330         FontSize="14"
331         Text="{x:Static lang:RealtimePageResources.Accurac
332             <-- yMagnitudePoints}"/>
333     <TextBlock Grid.Row="0"
334         Grid.Column="1"
335         HorizontalAlignment="Right"
336         FontWeight="SemiBold"
337         FontSize="16"
338         Text="{Binding CurrentDisplayEew.Earthquake.Hypoce
339             <-- ntre.Accuracy.Epicentres[0],
340             Converter={StaticResource
341                 <-- EnumDisplayTextConverter},
342             TargetNullValue={x:Static
343                 <-- lang:EarthquakeResources.UnknownText},
344             FallbackValue={x:Static
345                 <-- lang:EarthquakeResources.UnknownText}}"/>
346     <TextBlock Grid.Row="1"
347         Grid.Column="1"
348         HorizontalAlignment="Right"
349         FontWeight="SemiBold"
350         FontSize="16"
351         Text="{Binding CurrentDisplayEew.Earthquake.Hypoce
352             <-- ntre.Accuracy.Epicentres[1],
353             Converter={StaticResource
354                 <-- EnumDisplayTextConverter},
355             TargetNullValue={x:Static
356                 <-- lang:EarthquakeResources.UnknownText},
357             FallbackValue={x:Static
358                 <-- lang:EarthquakeResources.UnknownText}}"/>
359     <TextBlock Grid.Row="2"
360         Grid.Column="1"
361         HorizontalAlignment="Right"
362         FontWeight="SemiBold"
363         FontSize="16"
364         Text="{Binding CurrentDisplayEew.Earthquake.Hypoce
365             <-- ntre.Accuracy.Depth,
366             Converter={StaticResource
367                 <-- EnumDisplayTextConverter},
368             TargetNullValue={x:Static
369                 <-- lang:EarthquakeResources.UnknownText},
370             FallbackValue={x:Static
371                 <-- lang:EarthquakeResources.UnknownText}}"/>
372     <TextBlock Grid.Row="3"
```

```
358             Grid.Column="1"
359             HorizontalAlignment="Right"
360             FontWeight="SemiBold"
361             FontSize="16"
362             Text="{Binding CurrentDisplayEew.Earthquake.Hypoce_
363             ntre.Accuracy.Magnitude,
364             Converter={StaticResource
365             → EnumDisplayTextConverter},
366             TargetNullValue={x:Static
367             → lang:EarthquakeResources.UnknownText},
368             FallbackValue={x:Static
369             → lang:EarthquakeResources.UnknownText}}"/>
370             <TextBlock Grid.Row="4"
371             Grid.Column="1"
372             HorizontalAlignment="Right"
373             FontWeight="SemiBold"
374             FontSize="16"
375             Text="{Binding CurrentDisplayEew.Earthquake.Hypoce_
376             ntre.Accuracy.MagnitudePoint,
377             Converter={StaticResource
378             → EnumDisplayTextConverter},
379             TargetNullValue={x:Static
380             → lang:EarthquakeResources.UnknownText},
381             FallbackValue={x:Static
382             → lang:EarthquakeResources.UnknownText}}"/>
383             </Grid>
384
385             <!--One Point Warning Text-->
386
387             <TextBlock HorizontalAlignment="Left"
388             FontSize="18"
389             FontWeight="SemiBold"
390             Text="{x:Static
391             → lang:RealtimePageResources.EewOnePointWarningText}"
392             TextWrapping="Wrap"
393             IsVisible="{Binding CurrentDisplayEew.IsOnePointInfo}"
394             Margin="5 5"/>
395
396             <!--Update Time Display-->
397
398             <TextBlock HorizontalAlignment="Left"
399             FontSize="14"
400             FontWeight="SemiBold"
401             Text="{x:Static
402             → lang:EarthquakeResources.DetailsLastUpdatedText}"
403             Margin="5 0"/>
404             <TextBlock HorizontalAlignment="Right"
405             FontSize="18"
406             FontWeight="SemiBold"
407             Text="{Binding CurrentDisplayEew.UpdateTime,
408             StringFormat='yyyy/MM/dd HH:mm:ss',
409             TargetNullValue={x:Static
410             → lang:EarthquakeResources.UnknownText},
411             FallbackValue={x:Static
412             → lang:EarthquakeResources.UnknownText}}"
413             Margin="5 0"/>
414
415             <!--Informational Text-->
```

```
405 <TextBlock HorizontalAlignment="Left"
406     FontSize="14"
407     Text="{Binding CurrentDisplayEew.InformationalText}"
408     TextWrapping="Wrap"
409     Margin="8"/>
410 </StackPanel>
411 </Border>
412
413
414
415 <!--Tsunami Info Display-->
416
417 <Border Canvas.Bottom="0"
418     Canvas.Right="0"
419     IsVisible="{Binding CurrentTsunami,
420     Converter={x:Static ObjectConverters.IsNotNull}}"
421     Background="{StaticResource SystemChromeBlackMediumColor}"
422     Width="350"
423     CornerRadius="10 0 0 0">
424     <StackPanel>
425
426         <!--Tsunami Colour Display-->
427
428         <Border Background="{Binding CurrentTsunami.MaxWarningType,
429             Converter={StaticResource EnumDisplayColourConverter},
430             TargetNullValue={x:Static
431                 ↳ lang:RealtimePageResources.TsunamiColourNone},
432             FallbackValue={x:Static
433                 ↳ lang:RealtimePageResources.TsunamiColourNone}}"
434             Height="10"
435             CornerRadius="10 0 0 0">
436             </Border>
437
438             <Border Background="{Binding CurrentTsunami.MaxWarningType,
439                 Converter={StaticResource EnumDisplayColourConverter},
440                 TargetNullValue={x:Static
441                     ↳ lang:RealtimePageResources.TsunamiColourNone},
442                     FallbackValue={x:Static
443                         ↳ lang:RealtimePageResources.TsunamiColourNone}}"
444                 Margin="0 5"
445                 Height="5">
446             </Border>
447
448         <!--Tsunami Text Display-->
449
450         <TextBlock FontWeight="SemiBold"
451             Margin="5 5"
452             FontSize="24"
453             Text="{Binding CurrentTsunami.MaxWarningType,
454                 Converter={StaticResource EnumDisplayTextConverter},
455                 TargetNullValue={x:Static
456                     ↳ lang:RealtimePageResources.TsunamiTextNone},
457                     FallbackValue={x:Static
458                         ↳ lang:RealtimePageResources.TsunamiTextNone}}"/>
459
460         <!--Update Time Display-->
461
462         <TextBlock HorizontalAlignment="Left"
463             FontSize="14"
```

```
458     FontWeight="SemiBold"
459     Text="{x:Static
460         ↳ lang:EarthquakeResources.DetailsLastUpdatedText}"
461     Margin="5 0"/>
462     <TextBlock HorizontalAlignment="Right"
463         FontSize="18"
464         FontWeight="SemiBold"
465         Text="{Binding CurrentTsunami.UpdateTime,
466             StringFormat='yyyy/MM/dd HH:mm:ss',
467             TargetNullValue={x:Static
468                 ↳ lang:EarthquakeResources.UnknownText},
469             FallbackValue={x:Static
470                 ↳ lang:EarthquakeResources.UnknownText}}"
471             Margin="5 0"/>
472
473     <!--Informational Text-->
474
475     <ScrollViewer Margin="8"
476         Height="300">
477         <TextBlock HorizontalAlignment="Left"
478             FontSize="14"
479             Text="{Binding CurrentTsunami.InformationalText}"
480             TextWrapping="Wrap"/>
481     </ScrollViewer>
482     </StackPanel>
483 </Border>
484 </Canvas>
485 </Grid>
486 </UserControl>
```

Listing A.10.48: EasonEtwViewer/Views/RealTimePageView.axaml

```
1  <Styles xmlns="https://github.com/avaloniaui"
2      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
3      <Design.PreviewWith>
4          <Border Padding="20">
5              <!-- Add Controls for Previewer Here -->
6          </Border>
7      </Design.PreviewWith>
8
9      <Style>
10         <Style.Resources>
```

```
11 <StreamGeometry x:Key="CellularData1Regular">M3.75,17 C4.12969577,17  
→ 4.44349096,17.2821539 4.49315338,17.6482294 L4.5,17.75 L4.5,19.25  
→ C4.5,19.6642136 4.16421356,20 3.75,20 C3.37030423,20  
→ 3.05650904,19.7178461 3.00684662,19.3517706 L3,19.25 L3,17.75  
→ C3,17.3357864 3.33578644,17 3.75,17 Z M11.75,11 C12.1296958,11  
→ 12.443491,11.2821539 12.4931534,11.6482294 L12.5,11.75 L12.5,19.25  
→ C12.5,19.6642136 12.1642136,20 11.75,20 C11.3703042,20  
→ 11.056509,19.7178461 11.0068466,19.3517706 L11,19.25 L11,11.75  
→ C11,11.3357864 11.3357864,11 11.75,11 Z M15.75,8 C16.1296958,8  
→ 16.443491,8.28215388 16.4931534,8.64822944 L16.5,8.75 L16.5,19.25  
→ C16.5,19.6642136 16.1642136,20 15.75,20 C15.3703042,20  
→ 15.056509,19.7178461 15.0068466,19.3517706 L15,19.25 L15,8.75  
→ C15,8.33578644 15.3357864,8 15.75,8 Z M19.742726,5  
→ C20.1224039,4.99635605 20.4389211,5.27522768 20.492132,5.64050924  
→ L20.4999655,5.74212651 L20.4999655,19.2431626 C20.5039832,19.6570224  
→ 20.1714696,19.9957764 19.7572755,19.999897 C19.3775976,20.0034768  
→ 19.0610804,19.7245997 19.0078695,19.3593177 L19.000036,19.2577004  
→ L19.000036,5.75666426 C18.9960183,5.3428045 19.3285319,5.00405049  
→ 19.742726,5 Z M7.75,14 C8.12969577,14 8.44349096,14.2821539  
→ 8.49315338,14.6482294 L8.5,14.75 L8.5,19.2487201 C8.5,19.6629337  
→ 8.16421356,19.9987201 7.75,19.9987201 C7.37030423,19.9987201  
→ 7.05650904,19.7165662 7.00684662,19.3504907 L7,19.2487201 L7,14.75  
→ C7,14.3357864 7.33578644,14 7.75,14 Z</StreamGeometry>  
12 <StreamGeometry x:Key="LinkSquareRegular">M6.25 3C4.45507 3 3 4.45507 3  
→ 6.25V12.25C3 14.0449 4.45507 15.5 6.25 15.5H7.5V14H6.25C5.2835 14 4.5  
→ 13.2165 4.5 12.25V6.25C4.5 5.2835 5.2835 4.5 6.25 4.5H12.25C13.2165  
→ 4.5 14 5.2835 14 6.25V12.25C14 13.2165 13.2165 14 12.25  
→ 14H11V15.5H12.25C14.0449 15.5 15.5 14.0449 15.5 12.25V6.25C15.5  
→ 4.45507 14.0449 3 12.25 3H6.25Z M10 11.75C10 10.7835 10.7835 10 11.75  
→ 10H12.9982V8.5H11.75C9.95507 8.5 8.5 9.95507 8.5 11.75V17.75C8.5  
→ 19.5449 9.95507 21 11.75 21H17.75C19.5449 21 21 19.5449 21  
→ 17.75V11.75C21 9.95507 19.5449 8.5 17.75 8.5H16.5V10H17.75C18.7165 10  
→ 19.5 10.7835 19.5 11.75V17.75C19.5 18.7165 18.7165 19.5 17.75  
→ 19.5H11.75C10.7835 19.5 10 18.7165 10 17.75V11.75Z</StreamGeometry>  
13 <StreamGeometry x:Key="SidebarRegular">M2 4.5C2 4.22386 2.22386 4 2.5  
→ 4H17.5C17.7761 4 18 4.22386 18 4.5C18 4.77614 17.7761 5 17.5  
→ 5H2.5C2.22386 5 2 4.77614 2 4.5Z M2 9.5C2 9.22386 2.22386 9 2.5  
→ 9H17.5C17.7761 9 18 9.22386 18 9.5C18 9.77614 17.7761 10 17.5  
→ 10H2.5C2.22386 10 2 9.77614 2 9.5Z M2.5 14C2.22386 14 2 14.2239 2  
→ 14.5C2 14.7761 2.22386 15 2.5 15H17.5C17.7761 15 18 14.7761 18  
→ 14.5C18 14.2239 17.7761 14 17.5 14H2.5Z</StreamGeometry>
```

14 <StreamGeometry x:Key="SettingsRegular">M14 9.50006C11.5147 9.50006 9.5
→ 11.5148 9.5 14.0001C9.5 16.4853 11.5147 18.5001 14 18.5001C15.3488
→ 18.5001 16.559 17.9066 17.3838 16.9666C18.0787 16.1746 18.5 15.1365
→ 18.5 14.0001C18.5 13.5401 18.431 13.0963 18.3028 12.6784C17.7382
→ 10.8381 16.0253 9.50006 14 9.50006ZM11 14.0001C11 12.3432 12.3431
→ 11.0001 14 11.0001C15.6569 11.0001 17 12.3432 17 14.0001C17 15.6569
→ 15.6569 17.0001 14 17.0001C12.3431 17.0001 11 15.6569 11 14.0001Z
→ M21.7093 22.3948L19.9818 21.6364C19.4876 21.4197 18.9071 21.4515
→ 18.44 21.7219C17.9729 21.9924 17.675 22.4693 17.6157 23.0066L17.408
→ 24.8855C17.3651 25.273 17.084 25.5917 16.7055 25.682C14.9263 26.1061
→ 13.0725 26.1061 11.2933 25.682C10.9148 25.5917 10.6336 25.273 10.5908
→ 24.8855L10.3834 23.0093C10.3225 22.4731 10.0112 21.9976 9.54452
→ 21.7281C9.07783 21.4586 8.51117 21.4269 8.01859 21.6424L6.29071
→ 22.4009C5.93281 22.558 5.51493 22.4718 5.24806 22.1859C4.00474
→ 20.8536 3.07924 19.2561 2.54122 17.5137C2.42533 17.1384 2.55922
→ 16.7307 2.8749 16.4977L4.40219 15.3703C4.83721 15.0501 5.09414
→ 14.5415 5.09414 14.0007C5.09414 13.4598 4.83721 12.9512 4.40162
→ 12.6306L2.87529 11.5051C2.55914 11.272 2.42513 10.8638 2.54142
→ 10.4882C3.08038 8.74734 4.00637 7.15163 5.24971 5.82114C5.51684
→ 5.53528 5.93492 5.44941 6.29276 5.60691L8.01296 6.36404C8.50793
→ 6.58168 9.07696 6.54881 9.54617 6.27415C10.0133 6.00264 10.3244
→ 5.52527 10.3844 4.98794L10.5933 3.11017C10.637 2.71803 10.9245
→ 2.39704 11.3089 2.31138C12.19 2.11504 13.0891 2.01071 14.0131
→ 2.00006C14.9147 2.01047 15.8128 2.11485 16.6928 2.31149C17.077
→ 2.39734 17.3643 2.71823 17.4079 3.11017L17.617 4.98937C17.7116
→ 5.85221 18.4387 6.50572 19.3055 6.50663C19.5385 6.507 19.769 6.45838
→ 19.9843 6.36294L21.7048 5.60568C22.0626 5.44818 22.4807 5.53405
→ 22.7478 5.81991C23.9912 7.1504 24.9172 8.74611 25.4561 10.487C25.5723
→ 10.8623 25.4386 11.2703 25.1228 11.5035L23.5978 12.6297C23.1628 12.95
→ 22.9 13.4586 22.9 13.9994C22.9 14.5403 23.1628 15.0489 23.5988
→ 15.3698L25.1251 16.4965C25.441 16.7296 25.5748 17.1376 25.4586
→ 17.5131C24.9198 19.2536 23.9944 20.8492 22.7517 22.1799C22.4849
→ 22.4657 22.0671 22.5518 21.7093 22.3948ZM16.263 22.1966C16.4982
→ 21.4685 16.9889 20.8288 17.6884 20.4238C18.5702 19.9132 19.6536
→ 19.8547 20.5841 20.2627L21.9281 20.8526C22.791 19.8538 23.4593
→ 18.7013 23.8981 17.4552L22.7095 16.5778L22.7086 16.5771C21.898 15.98
→ 21.4 15.0277 21.4 13.9994C21.4 12.9719 21.8974 12.0195 22.7073
→ 11.4227L22.7085 11.4218L23.8957 10.545C23.4567 9.2988 22.7881 8.14636
→ 21.9248 7.1477L20.5922 7.73425L20.5899 7.73527C20.1844 7.91463
→ 19.7472 8.00722 19.3039 8.00663C17.6715 8.00453 16.3046 6.77431
→ 16.1261 5.15465L16.1259 5.15291L15.9635 3.69304C15.3202 3.57328
→ 14.6677 3.50872 14.013 3.50017C13.3389 3.50891 12.6821 3.57367
→ 12.0377 3.69328L11.8751 5.15452C11.7625 6.16272 11.1793 7.05909
→ 10.3019 7.56986C9.41937 8.0856 8.34453 8.14844 7.40869
→ 7.73694L6.07273 7.14893C5.20949 8.14751 4.54092 9.29983 4.10196
→ 10.5459L5.29181 11.4233C6.11115 12.0269 6.59414 12.9837 6.59414
→ 14.0007C6.59414 15.0173 6.11142 15.9742 5.29237 16.5776L4.10161
→ 17.4566C4.54002 18.7044 5.2085 19.8585 6.07205 20.8587L7.41742
→ 20.2682C8.34745 19.8613 9.41573 19.9215 10.2947 20.4292C11.174 20.937
→ 11.7593 21.832 11.8738 22.84L11.8744 22.8445L12.0362 24.3088C13.3326
→ 24.5638 14.6662 24.5638 15.9626 24.3088L16.1247 22.8418C16.1491
→ 22.6217 16.1955 22.4055 16.263 22.1966Z</StreamGeometry>

```

15 <StreamGeometry x:Key="LiveRegular">M5.98959236,4.92893219
  ↵ C6.28248558,5.22182541 6.28248558,5.69669914 5.98959236,5.98959236
  ↵ C2.67013588,9.30904884 2.67013588,14.6909512 5.98959236,18.0104076
  ↵ C6.28248558,18.3033009 6.28248558,18.7781746 5.98959236,19.0710678
  ↵ C5.69669914,19.363961 5.22182541,19.363961 4.92893219,19.0710678
  ↵ C1.02368927,15.1658249 1.02368927,8.83417511 4.92893219,4.92893219
  ↵ C5.22182541,4.63603897 5.69669914,4.63603897 5.98959236,4.92893219 Z
  ↵ M19.0710678,4.92893219 C22.9763107,8.83417511 22.9763107,15.1658249
  ↵ 19.0710678,19.0710678 C18.7781746,19.363961 18.3033009,19.363961
  ↵ 18.0104076,19.0710678 C17.7175144,18.7781746 17.7175144,18.3033009
  ↵ 18.0104076,18.0104076 C21.3298641,14.6909512 21.3298641,9.30904884
  ↵ 18.0104076,5.98959236 C17.7175144,5.69669914 17.7175144,5.22182541
  ↵ 18.0104076,4.92893219 C18.3033009,4.63603897 18.7781746,4.63603897
  ↵ 19.0710678,4.92893219 Z M8.81801948,7.75735931 C9.1109127,8.05025253
  ↵ 9.1109127,8.52512627 8.81801948,8.81801948 C7.06066017,10.5753788
  ↵ 7.06066017,13.4246212 8.81801948,15.1819805 C9.1109127,15.4748737
  ↵ 9.1109127,15.9497475 8.81801948,16.2426407 C8.52512627,16.5355339
  ↵ 8.05025253,16.5355339 7.75735931,16.2426407 C5.41421356,13.8994949
  ↵ 5.41421356,10.1005051 7.75735931,7.75735931 C8.05025253,7.46446609
  ↵ 8.52512627,7.46446609 8.81801948,7.75735931 Z M16.2426407,7.75735931
  ↵ C18.5857864,10.1005051 18.5857864,13.8994949 16.2426407,16.2426407
  ↵ C15.9497475,16.5355339 15.4748737,16.5355339 15.1819805,16.2426407
  ↵ C14.8890873,15.9497475 14.8890873,15.4748737 15.1819805,15.1819805
  ↵ C16.9393398,13.4246212 16.9393398,10.5753788 15.1819805,8.81801948
  ↵ C14.8890873,8.52512627 14.8890873,8.05025253 15.1819805,7.75735931
  ↵ C15.4748737,7.46446609 15.9497475,7.46446609 16.2426407,7.75735931 Z
  ↵ M12,10.5 C12.8284271,10.5 13.5,11.1715729 13.5,12 C13.5,12.8284271
  ↵ 12.8284271,13.5 12,13.5 C11.1715729,13.5 10.5,12.8284271 10.5,12
  ↵ C10.5,11.1715729 11.1715729,10.5 12,10.5 Z</StreamGeometry>
16 <StreamGeometry x:Key="HistoryRegular">M12,3 C16.9705627,3 21,7.02943725
  ↵ 21,12 C21,16.9705627 16.9705627,21 12,21 C10.2903875,21
  ↵ 8.64944804,20.5217936 7.23080614,19.6337823 C6.65294072,19.2720624
  ↵ 6.11769706,18.8456754 5.63566512,18.3635872 C5.15289898,17.8807646
  ↵ 4.72601217,17.3445768 4.36400875,16.7656632 C3.47740079,15.3478054
  ↵ 3,13.7081751 3,12 C3,11.725846 3.01227986,11.4530822
  ↵ 3.03669655,11.1822711 C3.07389172,10.769731 3.43847425,10.4654537
  ↵ 3.85101443,10.5026489 C4.2635546,10.5398441 4.56783184,10.9044266
  ↵ 4.53063667,11.3169668 C4.51025648,11.5430081 4.5,11.7708281 4.5,12
  ↵ C4.5,13.4248663 4.89726709,14.7892778 5.63582603,15.9703766
  ↵ C5.93763522,16.4530279 6.2936779,16.9002326 6.69638725,17.3029889
  ↵ C7.09848441,17.7051331 7.54490219,18.0607594 8.02668093,18.362333
  ↵ C9.20843557,19.1020627 10.5739361,19.5 12,19.5 C16.1421356,19.5
  ↵ 19.5,16.1421356 19.5,12 C19.5,7.85786438 16.1421356,4.5 12,4.5
  ↵ C9.60270786,4.5 7.39952516,5.63281093 5.99774512,7.50196302
  ↵ L8.75418677,7.50209339 C9.16840034,7.50209339 9.50418677,7.83787982
  ↵ 9.50418677,8.25209339 C9.50418677,8.63178915 9.22203289,8.94558435
  ↵ 8.85595733,8.99524677 L8.75418677,9.00209339 L4.25,9.00209339
  ↵ C3.87030423,9.00209339 3.55650904,8.71993951 3.50684662,8.35386394
  ↵ L3.5,8.25209339 L3.5,3.75209339 C3.5,3.33787982 3.83578644,3.00209339
  ↵ 4.25,3.00209339 C4.62969577,3.00209339 4.94349096,3.28424727
  ↵ 4.99315338,3.65032283 L5,3.75209339 L4.99900166,6.34348083
  ↵ C6.68586937,4.25603382 9.23627771,3 12,3 Z M11.25,7 C11.6295,7
  ↵ 11.9434583,7.28233333 11.9931493,7.64827431 L12,7.75 L12,12 L14.25,12
  ↵ C14.664,12 15,12.336 15,12.75 C15,13.1295 14.7176667,13.4434583
  ↵ 14.3517257,13.4931493 L14.25,13.5 L11.25,13.5 C10.8705,13.5
  ↵ 10.5565417,13.2176667 10.5068507,12.8517257 L10.5,12.75 L10.5,7.75
  ↵ C10.5,7.336 10.836,7 11.25,7 Z</StreamGeometry>
17 </Style.Resources>
```

```
18  </Style>
19  </Styles>
```

Listing A.10.49: EasonEetwViewer/Assets/Icons.axaml

```
1  using System.Globalization;
2  using Microsoft.Extensions.Logging;
3
4  namespace EasonEetwViewer.ViewModels.ViewModelBases;
5  ///<summary>
6  /// Logs for <see cref="SettingPageViewModel"/>.
7  ///</summary>
8  internal static partial class SettingPageViewModelLogs
9  {
10     ///<summary>
11     /// Log when text in API Key input box is changed.
12     ///</summary>
13     ///<param name="logger">The logger to be used.</param>
14     [LoggerMessage(
15         EventId = 0,
16         EventName = nameof(TextChanged),
17         Level = LogLevel.Information,
18         Message = "The text in API Key box is changed.")]
19     public static partial void TextChanged(
20         this ILogger<SettingPageViewModel> logger);
21     ///<summary>
22     /// Log when editing API Key.
23     ///</summary>
24     ///<param name="logger">The logger to be used.</param>
25     [LoggerMessage(
26         EventId = 1,
27         EventName = nameof(UnsettingApiKey),
28         Level = LogLevel.Debug,
29         Message = "Unsetting API Key since the text in API Key box is changed.")]
30     public static partial void UnsettingApiKey(
31         this ILogger<SettingPageViewModel> logger);
32     ///<summary>
33     /// Log when setting API Key.
34     ///</summary>
35     ///<param name="logger">The logger to be used.</param>
36     [LoggerMessage(
37         EventId = 2,
38         EventName = nameof(SettingApiKey),
39         Level = LogLevel.Information,
40         Message = "Setting API Key.")]
41     public static partial void SettingApiKey(
42         this ILogger<SettingPageViewModel> logger);
43     ///<summary>
44     /// Log when unsetting OAuth.
45     ///</summary>
46     ///<param name="logger">The logger to be used.</param>
47     [LoggerMessage(
48         EventId = 3,
49         EventName = nameof(UnsettingOAuth),
50         Level = LogLevel.Information,
51         Message = "Unsetting OAuth.")]
52     public static partial void UnsettingOAuth(
53         this ILogger<SettingPageViewModel> logger);
```

```
54     /// <summary>
55     /// Log when setting OAuth.
56     /// </summary>
57     /// <param name="logger">The logger to be used.</param>
58     [LoggerMessage(
59         EventId = 4,
60         EventName = nameof(SettingOAuth),
61         Level = LogLevel.Information,
62         Message = "Setting OAuth.")]
63     public static partial void SettingOAuth(
64         this ILogger<SettingPageViewModel> logger);
65     /// <summary>
66     /// Log when setting OAuth set finished.
67     /// </summary>
68     /// <param name="logger">The logger to be used.</param>
69     [LoggerMessage(
70         EventId = 5,
71         EventName = nameof(OAuthSet),
72         Level = LogLevel.Information,
73         Message = "OAuth set finished.")]
74     public static partial void OAuthSet(
75         this ILogger<SettingPageViewModel> logger);
76     /// <summary>
77     /// Log when connecting to WebSocket.
78     /// </summary>
79     /// <param name="logger">The logger to be used.</param>
80     [LoggerMessage(
81         EventId = 6,
82         EventName = nameof(ConnectingWebSocket),
83         Level = LogLevel.Information,
84         Message = "Connecting to WebSocket.")]
85     public static partial void ConnectingWebSocket(
86         this ILogger<SettingPageViewModel> logger);
87     /// <summary>
88     /// Log when connected to WebSocket.
89     /// </summary>
90     /// <param name="logger">The logger to be used.</param>
91     [LoggerMessage(
92         EventId = 7,
93         EventName = nameof(ConnectedWebSocket),
94         Level = LogLevel.Information,
95         Message = "Connected to WebSocket.")]
96     public static partial void ConnectedWebSocket(
97         this ILogger<SettingPageViewModel> logger);
98     /// <summary>
99     /// Log when connected to WebSocket.
100    /// </summary>
101   /// <param name="logger">The logger to be used.</param>
102   [LoggerMessage(
103       EventId = 8,
104       EventName = nameof(DisconnectingWebSocket),
105       Level = LogLevel.Information,
106       Message = "Disconnecting from WebSocket.")]
107   public static partial void DisconnectingWebSocket(
108       this ILogger<SettingPageViewModel> logger);
109   /// <summary>
110   /// Log when getting number of available WebSocket connections.
111   /// </summary>
112   /// <param name="logger">The logger to be used.</param>
```

```
113 [LoggerMessage(
114     EventId = 9,
115     EventName = nameof(RequestingAvailableConnections),
116     Level = LogLevel.Information,
117     Message = "Requesting available connections.")]
118 public static partial void RequestingAvailableConnections(
119     this ILogger<SettingPageViewModel> logger);
120 /// <summary>
121 /// Log when successfully get the number of available WebSocket connections.
122 /// </summary>
123 /// <param name="logger">The logger to be used.</param>
124 /// <param name="number">The number of available connections</param>
125 [LoggerMessage(
126     EventId = 10,
127     EventName = nameof(RequestedAvailableConnections),
128     Level = LogLevel.Information,
129     Message = "Available connections: `'{Number}`.")]
130 public static partial void RequestedAvailableConnections(
131     this ILogger<SettingPageViewModel> logger, int number);
132 /// <summary>
133 /// Log when requesting current WebSocket connections.
134 /// </summary>
135 /// <param name="logger">The logger to be used.</param>
136 [LoggerMessage(
137     EventId = 11,
138     EventName = nameof(RequestingCurrentConnections),
139     Level = LogLevel.Information,
140     Message = "Requesting current WebSocket connections.")]
141 public static partial void RequestingCurrentConnections(
142     this ILogger<SettingPageViewModel> logger);
143 /// <summary>
144 /// Log when adding active WebSocket connections to current display.
145 /// </summary>
146 /// <param name="logger">The logger to be used.</param>
147 [LoggerMessage(
148     EventId = 12,
149     EventName = nameof(DisplayingActiveConnections),
150     Level = LogLevel.Trace,
151     Message = "Displaying active connections.")]
152 public static partial void DisplayingActiveConnections(
153     this ILogger<SettingPageViewModel> logger);
154 /// <summary>
155 /// Log when adding empty WebSocket connections to current display.
156 /// </summary>
157 /// <param name="logger">The logger to be used.</param>
158 [LoggerMessage(
159     EventId = 13,
160     EventName = nameof(DisplayingAvailableConnections),
161     Level = LogLevel.Trace,
162     Message = "Displaying available connections.")]
163 public static partial void DisplayingAvailableConnections(
164     this ILogger<SettingPageViewModel> logger);
165 /// <summary>
166 /// Log when failed to read language from file.
167 /// </summary>
168 /// <param name="logger">The logger to be used.</param>
169 /// <param name="filePath">The file path of the language file.</param>
170 [LoggerMessage(
171     EventId = 14,
```

```

172     EventName = nameof(FailedToReadLanguage),
173     Level = LogLevel.Warning,
174     Message = "Failed to read language from file: `'{FilePath}``.");
175     public static partial void FailedToReadLanguage(
176         this ILogger<SettingPageViewModel> logger, string filePath);
177     /// <summary>
178     /// Log when succeeded in reading language from file.
179     /// </summary>
180     /// <param name="logger">The logger to be used.</param>
181     /// <param name="filePath">The file path of the language file.</param>
182     /// <param name="language">The language read.</param>
183     [LoggerMessage(
184         EventId = 15,
185         EventName = nameof(ReadLanguage),
186         Level = LogLevel.Information,
187         Message = "Read language `{Language}` from file: `'{FilePath}``.");
188     public static partial void ReadLanguage(
189         this ILogger<SettingPageViewModel> logger, string filePath, CultureInfo
190             ↳ language);
191     /// <summary>
192     /// Log when failed to write language to file.
193     /// </summary>
194     /// <param name="logger">The logger to be used.</param>
195     /// <param name="filePath">The file path of the language file.</param>
196     [LoggerMessage(
197         EventId = 16,
198         EventName = nameof(FailedToWriteLanguage),
199         Level = LogLevel.Warning,
200         Message = "Failed to write language to file: `'{FilePath}``.");
201     public static partial void FailedToWriteLanguage(
202         this ILogger<SettingPageViewModel> logger, string filePath);
203     /// <summary>
204     /// Log when succeeded in writing language to file.
205     /// </summary>
206     /// <param name="logger">The logger to be used.</param>
207     /// <param name="filePath">The file path of the language file.</param>
208     /// <param name="language">The language read.</param>
209     [LoggerMessage(
210         EventId = 17,
211         EventName = nameof(WriteLanguage),
212         Level = LogLevel.Information,
213         Message = "Wrote language `{Language}` to file: `'{FilePath}``.");
214     public static partial void WriteLanguage(
215         this ILogger<SettingPageViewModel> logger, string filePath, CultureInfo
216             ↳ language);
215 }

```

Listing A.10.50: EasonEetwViewer/Logging/ViewModels/SettingPageViewModelLogs.cs

```

1  using Microsoft.Extensions.Logging;
2
3  namespace EasonEetwViewer.ViewModels.ViewModelBases;
4  /// <summary>
5  /// Logs for <see cref="MainWindowViewModel"/>.
6  /// </summary>
7  internal static partial class MainWindowViewModelLogs
8  {
9      /// <summary>

```

```

10     /// Log when pane is triggered.
11     /// </summary>
12     /// <param name="logger">The logger to be used.</param>
13     /// <param name="isPaneOpen">The new state of the pane.</param>
14     [LoggerMessage(
15         EventId = 0,
16         EventName = nameof(PaneTriggered),
17         Level = LogLevel.Information,
18         Message = "Pane triggered to state `'{IsPaneOpen}`.")]
19     public static partial void PaneTriggered(
20         this ILogger<MainWindowViewModel> logger, bool isPaneOpen);
21     /// <summary>
22     /// Log when view model is switched.
23     /// </summary>
24     /// <param name="logger">The logger to be used.</param>
25     /// <param name="newViewModel">The new view model switched to.</param>
26     [LoggerMessage(
27         EventId = 1,
28         EventName = nameof(ViewModelSwitched),
29         Level = LogLevel.Debug,
30         Message = "View model is switched to: `'{NewViewModel}`.")]
31     public static partial void ViewModelSwitched(
32         this ILogger<MainWindowViewModel> logger, Type newViewModel);
33     /// <summary>
34     /// Log when switching to real time view model due to event received.
35     /// </summary>
36     /// <param name="logger">The logger to be used.</param>
37     [LoggerMessage(
38         EventId = 2,
39         EventName = nameof(SwitchingToRealTime),
40         Level = LogLevel.Debug,
41         Message = "View model is switched to Realtime due to event received.")]
42     public static partial void SwitchingToRealTime(
43         this ILogger<MainWindowViewModel> logger);
44 }

```

Listing A.10.51: EasonEetwViewer/Logging/ViewModels/MainWindowViewModelLogs.cs

```

1  using Microsoft.Extensions.Logging;
2
3  namespace EasonEetwViewer.ViewModels.ViewModelBases;
4  /// <summary>
5  /// Logs for <see cref="MapViewModelBase"/>.
6  /// </summary>
7  internal static partial class MapViewModelBaseLogs
8  {
9      /// <summary>
10     /// Log when map initialised.
11     /// </summary>
12     /// <param name="logger">The logger to be used.</param>
13     [LoggerMessage(
14         EventId = 0,
15         EventName = nameof(MapInitialised),
16         Level = LogLevel.Information,
17         Message = "Map initialised.")]
18     public static partial void MapInitialised(
19         this ILogger<PageViewModelBase> logger);
20 }

```

Listing A.10.52:
EasonEetwViewer/Logging/ViewModels/ViewModelBases/MapViewModelBaseLogs.cs

```

1  using Microsoft.Extensions.Logging;
2
3  namespace EasonEetwViewer.ViewModels.ViewModelBases;
4  ///<summary>
5  /// Logs for <see cref="ViewModelBase"/>.
6  ///</summary>
7  internal static partial class MainWindowViewModelLogs
8  {
9      ///<summary>
10     /// Log when instantiated.
11     ///</summary>
12     ///<param name="logger">The logger to be used.</param>
13     [LoggerMessage(
14         EventId = 0,
15         EventName = nameof(Instantiated),
16         Level = LogLevel.Information,
17         Message = "Instantiated.")]
18     public static partial void Instantiated(
19         this ILogger<ViewModelBase> logger);
20 }

```

Listing A.10.53: EasonEetwViewer/Logging/ViewModels/ViewModelBases/ViewModelBaseLogs.cs

```

1  using Mapsui.Nts.Providers.Shapefile;
2  using Mapsui.Providers;
3  using Mapsui.Styles;
4
5  namespace EasonEetwViewer.Services;
6
7  ///<summary>
8  /// Provides resources for the application.
9  ///</summary>
10 internal sealed class MapResourcesProvider
11 {
12     ///<summary>
13     /// The base path to the shape files.
14     ///</summary>
15     private const string _baseGisFile = "Content/GisFiles/";
16     ///<summary>
17     /// The shape files for earthquake forecast regions.
18     ///</summary>
19     public IProvider Region { get; private init; }
20     = ShapeFileToProvider("Simp_20240520_AreaForecastLocaleE_GIS/Regions.shp");
21     ///<summary>
22     /// The shape files for tsunami forecast regions.
23     ///</summary>
24     public IProvider Tsunami { get; private init; }
25     = ShapeFileToProvider("Simp_20240520_AreaTsunami_GIS/Tsunamis.shp");
26
27     // Adapted from https://mapsui.com/samples/ - Projection - Shapefile with
28     // → Projection
29     ///<summary>
30     /// Reads the shape file from the path and converts to a provider.

```

```
30     /// </summary>
31     /// <param name="shapeFilePath">The path to the shapefile.</param>
32     /// <returns>The <see cref="ProjectingProvider"/> that contains the <see
33     ↪ cref="ShapeFile"/> read.</returns>
34     private static ProjectingProvider ShapeFileToProvider(string shapeFilePath)
35         => new(
36             new ShapeFile(
37                 _baseGisFile + shapeFilePath,
38                 true,
39                 true)
40                 { CRS = "EPSG:4326" })
41                 { CRS = "EPSG:3857" };
42
43     /// <summary>
44     /// The style for the epicenter for normal hypocentres.
45     /// </summary>
46     public IStyle HypocentreShapeStyle { get; private init; }
47         = new ImageStyle
48         {
49             Image = "embedded://EasonEetwViewer.Resources.cross.svg"
50         };
51
52     /// <summary>
53     /// The style for the epicenter for assumed hypocentre methods.
54     /// </summary>
55     public IStyle PlumShapeStyle { get; private init; }
56         = new ImageStyle
57         {
58             Image = "embedded://EasonEetwViewer.Resources.circle.svg"
59         };
60
61     /// <summary>
62     /// The regions of Japan.
63     /// </summary>
64     public IEnumerable<PrefectureData> Prefectures { get; private init; } = [
65         new PrefectureData() { Code = "01", Name = " 北海道" },
66         new PrefectureData() { Code = "02", Name = " 青森県" },
67         new PrefectureData() { Code = "03", Name = " 岩手県" },
68         new PrefectureData() { Code = "04", Name = " 宮城県" },
69         new PrefectureData() { Code = "05", Name = " 秋田県" },
70         new PrefectureData() { Code = "06", Name = " 山形県" },
71         new PrefectureData() { Code = "07", Name = " 福島県" },
72         new PrefectureData() { Code = "08", Name = " 茨城県" },
73         new PrefectureData() { Code = "09", Name = " 栃木県" },
74         new PrefectureData() { Code = "10", Name = " 群馬県" },
75         new PrefectureData() { Code = "11", Name = " 埼玉県" },
76         new PrefectureData() { Code = "12", Name = " 千葉県" },
77         new PrefectureData() { Code = "13", Name = " 東京都" },
78         new PrefectureData() { Code = "14", Name = " 神奈川県" },
79         new PrefectureData() { Code = "15", Name = " 新潟県" },
80         new PrefectureData() { Code = "16", Name = " 富山県" },
81         new PrefectureData() { Code = "17", Name = " 石川県" },
82         new PrefectureData() { Code = "18", Name = " 福井県" },
83         new PrefectureData() { Code = "19", Name = " 山梨県" },
84         new PrefectureData() { Code = "20", Name = " 長野県" },
85         new PrefectureData() { Code = "21", Name = " 岐阜県" },
86         new PrefectureData() { Code = "22", Name = " 静岡県" },
87         new PrefectureData() { Code = "23", Name = " 愛知県" },
88         new PrefectureData() { Code = "24", Name = " 三重県" },
89         new PrefectureData() { Code = "25", Name = " 滋賀県" },
90         new PrefectureData() { Code = "26", Name = " 京都府" },
```

```

88     new PrefectureData() { Code = "27", Name = " 大阪府" },
89     new PrefectureData() { Code = "28", Name = " 兵庫県" },
90     new PrefectureData() { Code = "29", Name = " 奈良県" },
91     new PrefectureData() { Code = "30", Name = " 和歌山県" },
92     new PrefectureData() { Code = "31", Name = " 鳥取県" },
93     new PrefectureData() { Code = "32", Name = " 島根県" },
94     new PrefectureData() { Code = "33", Name = " 岡山县" },
95     new PrefectureData() { Code = "34", Name = " 広島県" },
96     new PrefectureData() { Code = "35", Name = " 山口県" },
97     new PrefectureData() { Code = "36", Name = " 德島県" },
98     new PrefectureData() { Code = "37", Name = " 香川県" },
99     new PrefectureData() { Code = "38", Name = " 愛媛県" },
100    new PrefectureData() { Code = "39", Name = " 高知県" },
101    new PrefectureData() { Code = "40", Name = " 福岡県" },
102    new PrefectureData() { Code = "41", Name = " 佐賀県" },
103    new PrefectureData() { Code = "42", Name = " 長崎県" },
104    new PrefectureData() { Code = "43", Name = " 熊本県" },
105    new PrefectureData() { Code = "44", Name = " 大分県" },
106    new PrefectureData() { Code = "45", Name = " 宫崎県" },
107    new PrefectureData() { Code = "46", Name = " 鹿児島県" },
108    new PrefectureData() { Code = "47", Name = " 沖縄県" }
109  ];
110 }

```

Listing A.10.54: EasonEetwViewer/Services/MapResourcesProvider.cs

```

1  namespace EasonEetwViewer.Services;
2  internal record PrefectureData
3  {
4      public required string Code { get; init; }
5      public required string Name { get; init; }
6  }

```

Listing A.10.55: EasonEetwViewer/Services/PrefectureData.cs

```

1  namespace EasonEetwViewer.Services.TimeProvider;
2  ///<summary>
3  /// Provides time-related information.
4  ///</summary>
5  internal interface ITimeProvider
6  {
7      ///<summary>
8      /// Returns the current time.
9      ///</summary>
10     ///<returns>The current time in <see cref="DateTimeOffset"/>. </returns>
11     ///<remarks>
12     /// It is NOT guaranteed that the offset will be in the current timezone.
13     ///</remarks>
14     DateTimeOffset Now();
15     ///<summary>
16     /// Converts the given <see cref="DateTimeOffset"/> to JST.
17     ///</summary>
18     ///<param name="dateTimeOffset">The <see cref="DateTimeOffset"/> to be
19     /// converted. </param>
20     ///<returns>The new <see cref="DateTimeOffset"/> whose time offset is in
21     /// JST. </returns>
22     DateTimeOffset ConvertToJst(DateTimeOffset dateTimeOffset);
23 }

```

Listing A.10.56: EasonEetwViewer/Services/TimeProvider/ITimeProvider.cs

```

1  namespace EasonEetwViewer.Services.TimeProvider;
2  ///<summary>
3  ///<summary>The default implementation of <see cref="ITimeProvider"/>.</summary>
4  ///</summary>
5  internal class DefaultTimeProvider : ITimeProvider
6  {
7      ///<inheritdoc/>
8      public DateTimeOffset Now()
9          => DateTimeOffset.Now;
10     ///<inheritdoc/>
11     public DateTimeOffset ConvertToJst(DateTimeOffset dateTimeOffset)
12         => dateTimeOffset.ToOffset(new TimeSpan(9, 0, 0));
13 }

```

Listing A.10.57: EasonEetwViewer/Services/TimeProvider/DefaultTimeProvider.cs

```

1  using Microsoft.Extensions.DependencyInjection;
2  using Microsoft.Extensions.DependencyInjection.Extensions;
3  using Microsoft.Extensions.Logging;
4
5  namespace EasonEetwViewer.Services.Logging;
6  ///<summary>
7  ///<summary>Provides extension methods to add <see cref="FileLogger"/> on a <see
8  → cref="ILoggingBuilder"/>.</summary>
9  ///</summary>
10 internal static class FileLoggerExtensions
11 {
12     ///<summary>
13     ///<summary>Adds a file logger to the <see cref="ILoggingBuilder"/> that writes logs to a
14     → <see cref="StreamWriter"/>.</summary>
15     ///</summary>
16     ///<param name="builder">The <see cref="ILoggingBuilder"/> where the logger is
17     → added to.</param>
18     ///<param name="logFileWriter">The <see cref="StreamWriter"/> to write logs
19     → to.</param>
20     ///<param name="minimumLogLevel">The minimum log level for the logger.</param>
21     ///<returns>The <see cref="ILoggingBuilder"/> so that calls can be
22     → chained.</returns>
23     public static ILoggingBuilder AddFileLogger(
24         this ILoggingBuilder builder,
25         StreamWriter logFileWriter,
26         LogLevel minimumLogLevel)
27     {
28         builder.Services.TryAddEnumerable(
29             ServiceDescriptor.Singleton<ILoggerProvider, FileLoggerProvider>(sp
30                 => new(logFileWriter, minimumLogLevel)));
31         return builder;
32     }
33 }

```

Listing A.10.58: EasonEetwViewer/Services/Logging/FileLoggerExtensions.cs

```

1  using Avalonia.Logging;
2  using EasonEetwViewer.Extensions;
3  using Microsoft.Extensions.Logging;
4
5  namespace EasonEetwViewer.Services.Logging;
6
7  /// <summary>
8  /// Adapts Avalonia logging <see cref="ILogSink"/> to Microsoft logging <see
9  /// cref="ILogger"/>.
10 /// </summary>
11 /// <param name="logger">The Microsoft <see cref="ILogger"/> to be used.</param>
12 internal class
13     AvaloniaToMicrosoftLoggingAdaptor(ILogger<AvaloniaToMicrosoftLoggingAdaptor>
14     logger) : ILogSink
15 {
16     /// <summary>
17     /// The logger to be used.
18     /// </summary>
19     private readonly ILogger<AvaloniaToMicrosoftLoggingAdaptor> _logger = logger;
20
21     /// <inheritdoc/>
22     public bool IsEnabled(LogEventLevel level, string area)
23         => _logger.IsEnabled(level.ToMicrosoftLogLevel());
24
25     /// <inheritdoc/>
26     public void Log(LogEventLevel level, string area, object? source, string
27         messageTemplate)
28     {
29         if (IsEnabled(level, area))
30         {
31             _logger.Log(
32                 level.ToMicrosoftLogLevel(),
33                 "[{Area}] {Source} : {MessageTemplate}",
34                 area,
35                 source,
36                 messageTemplate);
37         }
38     }
39
40     /// <inheritdoc/>
41     public void Log(LogEventLevel level, string area, object? source, string
42         messageTemplate, params object?[] propertyValues)
43     {
44         if (IsEnabled(level, area))
45         {
46             _logger.Log(
47                 level.ToMicrosoftLogLevel(),
48                 "[{Area}] {Source} : {MessageTemplate} {PropertyValues}",
49                 area,
50                 source,
51                 messageTemplate,
52                 propertyValues);
53         }
54     }
55 }

```

Listing A.10.59: EasonEetwViewer/Services/Logging/AvaloniaToMicrosoftLoggingAdaptor.cs

```
1  using Microsoft.Extensions.Logging;
```

```

2
3 namespace EasonEetwViewer.Services.Logging;
4 /// <summary>
5 /// A logger provider that creates <see cref="FileLogger"/> instances.
6 /// </summary>
7 /// <param name="logFileWriter">The <see cref="StreamWriter"/> to write logs
8 //→ to.</param>
9 /// <param name="minimumLogLevel">The minimum log level for the logger.</param>
10 internal class FileLoggerProvider(StreamWriter logFileWriter, LogLevel
11 //→ minimumLogLevel) : ILoggerProvider
12 {
13     /// <inheritdoc/>
14     public ILogger CreateLogger(string categoryName)
15         => new FileLogger(categoryName, logFileWriter, minimumLogLevel);
16     /// <inheritdoc/>
17     public void Dispose()
18         => logFileWriter.Dispose();
19 }

```

Listing A.10.60: EasonEetwViewer/Services/Logging/FileLoggerProvider.cs

```

1 using Microsoft.Extensions.Logging;
2
3 namespace EasonEetwViewer.Services.Logging;
4 /// <summary>
5 /// A logger that writes log messages to a stream, usually a file.
6 /// </summary>
7 /// <param name="name">The name of the logger.</param>
8 /// <param name="logFileWriter">The <see cref="StreamWriter"/> to write logs
9 //→ to.</param>
10 internal class FileLogger
11     (string name,
12     StreamWriter
13     logFileWriter,
14     LogLevel minimumLogLevel) : ILogger
15 {
16     /// <inheritdoc/>
17     public IDisposable? BeginScope<TState>(TState state) where TState : notnull
18         => null;
19     /// <inheritdoc/>
20     public bool IsEnabled(LogLevel logLevel)
21         => logLevel >= minimumLogLevel;
22     /// <inheritdoc/>
23     public void Log<TState>(
24         LogLevel logLevel,
25         EventId eventId,
26         TState state,
27         Exception? exception,
28         Func<TState, Exception?, string> formatter)
29     {
30         if (IsEnabled(logLevel))
31         {
32             logFileWriter.WriteLine($"[{DateTime.UtcNow:yyyy-MM-dd HH:mm:ss}] [{name}]
33 //→ [{logLevel}] : {formatter(state, exception)}");
34             logFileWriter.Flush();
35         }
36     }

```

36 }

Listing A.10.61: EasonEetwViewer/Services/Logging/FileLogger.cs

A.11 Use of AI

```

1  using System;
2  using System.Net.WebSockets;
3  using System.Text;
4  using System.Threading;
5  using System.Threading.Tasks;
6
7  class Program
8  {
9      static async Task Main(string[] args)
10     {
11         using (ClientWebSocket webSocket = new ClientWebSocket())
12         {
13             Uri serverUri = new Uri("wss://example.com/socket"); // Replace with your
14             ↳ WebSocket server URI
15
16             try
17             {
18                 Console.WriteLine("Connecting to WebSocket...");
19                 await webSocket.ConnectAsync(serverUri, CancellationToken.None);
20                 Console.WriteLine("Connected!");
21
22                 // Sending a message
23                 string messageToSend = "Hello, WebSocket!";
24                 byte[] messageBytes = Encoding.UTF8.GetBytes(messageToSend);
25                 await webSocket.SendAsync(new ArraySegment<byte>(messageBytes),
26                 ↳ WebSocketMessageType.Text, true, CancellationToken.None);
27                 Console.WriteLine($"Sent: {messageToSend}");
28
29                 // Receiving a message
30                 byte[] buffer = new byte[1024];
31                 WebSocketReceiveResult result = await webSocket.ReceiveAsync(new
32                 ↳ ArraySegment<byte>(buffer), CancellationToken.None);
33                 string messageReceived = Encoding.UTF8.GetString(buffer, 0,
34                 ↳ result.Count);
35                 Console.WriteLine($"Received: {messageReceived}");
36
37                 // Closing the WebSocket
38                 Console.WriteLine("Closing WebSocket...");
39                 await webSocket.CloseAsync(WebSocketCloseStatus.NormalClosure,
40                 ↳ "Goodbye", CancellationToken.None);
41                 Console.WriteLine("WebSocket closed.");
42             }
43             catch (WebSocketException ex)
44             {
45                 Console.WriteLine($"WebSocket error: {ex.Message}");
46             }
47             catch (Exception ex)
48             {
49                 Console.WriteLine($"Unexpected error: {ex.Message}");
50             }
51         }
52     }
53 }
```

```
47     }  
48 }
```

Listing A.11.1: Code generated by ChatGPT in [59]

Appendix B

Use of AI

The detailed prompt of the use of ChatGPT 4o [59] is as in Figure B.0.1, Figure B.0.2 and Figure B.0.3. For the ease of reference, the code is also included in Listing A.11.1.

Connecting to a WebSocket in C#

WebSocket connections in C# are typically handled using the `System.Net.WebSockets` namespace. Here's an example of how you can create a WebSocket connection in C#, send and receive messages.

Step-by-Step Guide

1. Add the Necessary Namespace

Ensure your project has access to the `System.Net.WebSockets` namespace. You might need to install the `System.Net.WebSockets.Client` package if you're targeting .NET Framework.

2. Create a WebSocket Connection

You use the `ClientWebSocket` class to create a client connection.

Figure B.0.1: Prompt to ChatGPT in [59], Part 1

This is used to outline a working example of `System.Net.WebSocket` class and the author made numerous edits and additions to eventually create commit `8f8f6e9` on the same day, and other references such as StackOverflow [20, 23] and the official Microsoft Documentation [49] has also been used to create the relevant code in WebSocket connections.

This code has eventually been refactored and removed in `60b660a`, still referencing the sources [20, 23].

Example Code <pre>csharp using System; using System.Net.WebSockets; using System.Text; using System.Threading; using System.Threading.Tasks; class Program { static async Task Main(string[] args) { using (ClientWebSocket webSocket = new ClientWebSocket()) { Uri serverUri = new Uri("wss://your-websocket-endpoint"); CancellationTokenSource cts = new CancellationTokenSource(); try { // Connect to the WebSocket server Console.WriteLine("Connecting to WebSocket server..."); await webSocket.ConnectAsync(serverUri, cts.Token); Console.WriteLine("Connected!"); // Send a message string message = "Hello, WebSocket!"; byte[] bytesToSend = Encoding.UTF8.GetBytes(message); await webSocket.SendAsync(new ArraySegment<byte>(bytesToSend), WebSocketMessageType.Text, true, cts.Token); Console.WriteLine(\$"Sent: {message}"); // Receive a message byte[] buffer = new byte[1024]; WebSocketReceiveResult result = await webSocket.ReceiveAsync(new ArraySegment<byte>(buffer), cts.Token); string receivedMessage = Encoding.UTF8.GetString(buffer, 0, result.Count); Console.WriteLine(\$"Received: {receivedMessage}"); // Close the connection await webSocket.CloseAsync(WebSocketCloseStatus.NormalClosure, "Connection closed."); Console.WriteLine("Connection closed."); } catch (WebSocketException e) { Console.WriteLine(\$"WebSocket error: {e.Message}"); } catch (Exception ex) { Console.WriteLine(\$"Error: {ex.Message}"); } } } }</pre>	Example Code <pre>csharp //Websocket() //socket-endpoint"; //ationTokenSource(); //socket server..."; //i, cts.Token); //getBytes(message); //gment<byte>(bytesToSend), WebSocketMessageType.Text, true, cts.Token); //; //: webSocket.ReceiveAsync(new ArraySegment<byte>(buffer), cts.Token); //8.GetString(buffer, 0, result.Count); //redMessage"); //closeStatus.NormalClosure, "Closing", cts.Token); //"); //{e.Message}); //{e.Message}); </pre>
---	--

Figure B.0.2: Prompt to ChatGPT in [59], Part 2

Explanation of Key Parts

1. `ClientWebSocket` : This is the main class for handling WebSocket connections.
2. `ConnectAsync` : Establishes the WebSocket connection to the specified URI.
3. `SendAsync` : Sends data to the WebSocket server.
4. `ReceiveAsync` : Receives data from the WebSocket server.
5. `CloseAsync` : Closes the WebSocket connection gracefully.
6. **Buffer Management:** The buffer is a byte array used to store incoming data temporarily.

Notes

- Replace `"wss://your-websocket-endpoint"` with your actual WebSocket server URI.
- Ensure you handle exceptions like `WebSocketException` for scenarios where the connection might fail.
- Use proper cancellation tokens to handle task cancellation in production code.

This code provides a basic framework for working with WebSocket connections in C#. You can expand it based on your application's needs.



Figure B.0.3: Prompt to ChatGPT in [59], Part 3

Appendix C

Bibliography

- [1] E. Aparicio. *DemoMapsui*. 2024. URL: <https://github.com/dev-elian/DemoMapsui>.
- [2] E. Aparicio. *MVVM with Maosui / Maps with .NET*. 3rd July 2024. URL: https://www.youtube.com/watch?v=5zcDIxUsV_Y (visited on 10/03/2025).
- [3] Avalonia, ed. *Avalonia Docs*. URL: <https://docs.avaloniaui.net/docs/welcome> (visited on 10/03/2025).
- [4] Avalonia, ed. *Built-in Controls*. URL: <https://docs.avaloniaui.net/docs/reference/controls/> (visited on 10/03/2025).
- [5] Avalonia. *Fluent Icons for Avalonia*. URL: <https://avaloniaui.github.io/icons.html> (visited on 10/03/2025).
- [6] Avalonia, ed. *How to Create a Custom Data Binding Converter*. URL: <https://docs.avaloniaui.net/docs/guides/data-binding/how-to-create-a-custom-data-binding-converter> (visited on 10/03/2025).
- [7] Avalonia, ed. *How To Log Errors and Warnings*. URL: <https://docs.avaloniaui.net/docs/guides/implementation-guides/logging-errors-and-warnings> (visited on 10/03/2025).
- [8] Avalonia, ed. *How To Use Design-time Data*. URL: <https://docs.avaloniaui.net/docs/guides/implementation-guides/how-to-use-design-time-data> (visited on 10/03/2025).
- [9] Avalonia, ed. *How to use INotifyPropertyChanged*. URL: <https://docs.avaloniaui.net/docs/guides/data-binding/inotifypropertychanged> (visited on 10/03/2025).
- [10] Avalonia, ed. *Localizing using ResX*. URL: <https://docs.avaloniaui.net/docs/guides/implementation-guides/localizing> (visited on 10/03/2025).
- [11] *Balsamiq*. Version 4.8.0. 2024. URL: <https://balsamiq.com>.
- [12] BillWagner et al., eds. *Documentation comments*. 15th June 2023. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/xmldoc/> (visited on 10/12/2024).
- [13] M. Bloch. *mapshaper*. 2025. URL: <https://mapshaper.org/>.
- [14] dmdata.jp, ed. *API v2*. 1st Dec. 2024. URL: <https://dmdata.jp/docs/reference/api/v2/> (visited on 07/12/2024).
- [15] dmdata.jp, ed. *EEW*について. 7th Feb. 2025. URL: <https://dmdata.jp/docs/eew/> (visited on 10/03/2025).
- [16] dmdata.jp, ed. *JSON*化データについて. 7th Feb. 2025. URL: <https://dmdata.jp/docs/reference/conversion/json/> (visited on 10/03/2025).
- [17] dmdata.jp, ed. *OAuth 2 v1*. 1st Dec. 2024. URL: <https://dmdata.jp/docs/reference/oauth2/v1> (visited on 10/12/2024).
- [18] dmdata.jp, ed. サンプル *JavaScript*. 1st Dec. 2024. URL: <https://dmdata.jp/docs/sample/javascript> (visited on 10/12/2024).
- [19] dmdata.jp, ed. 電文データ配信データの区分. 17th Feb. 2025. URL: <https://dmdata.jp/docs/telegrams/> (visited on 10/03/2025).
- [20] FloriUni. *Create a client websocket based console app*. 17th Jan. 2021. URL: <https://stackoverflow.com/a/65761228> (visited on 22/11/2024).

- [21] gewarren and BillWagner, eds. *JSON serialization and deserialization (marshalling and unmarshalling) in .NET*. 25th Oct. 2023. URL: <https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/overview> (visited on 01/12/2024).
- [22] gewarren et al., eds. *Handling and throwing exceptions in .NET*. 15th Sept. 2021. URL: <https://learn.microsoft.com/en-us/dotnet/standard/exceptions/> (visited on 10/03/2025).
- [23] Harry. *Connecting to websocket using C# (I can connect using JavaScript, but C# gives Status code 200 error)*. Ed. by kevinarpe. 25th Aug. 2020. URL: <https://stackoverflow.com/a/63574016> (visited on 22/11/2024).
- [24] J. Hunter et al., eds. *Matplotlib 3.9.3 documentation*. 2002. URL: <https://matplotlib.org/stable/users/index.html> (visited on 07/10/2024).
- [25] N. Iarocci. *How to implement a PKCE code challenge in C#*. 17th Jan. 2024. URL: <https://nicolaiarocci.com/how-to-implement-pkce-code-challenge-in-csharp/> (visited on 01/01/2024).
- [26] IEvangelist et al., eds. *.NET dependency injection*. 18th July 2024. URL: <https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection> (visited on 10/03/2025).
- [27] IEvangelist et al., eds. *Configuration in .NET*. 9th Oct. 2024. URL: <https://learn.microsoft.com/en-us/dotnet/core/extensions/configuration> (visited on 09/03/2025).
- [28] IEvangelist et al., eds. *Handle and raise events*. 4th Oct. 2022. URL: <https://learn.microsoft.com/en-us/dotnet/standard/events/> (visited on 10/03/2024).
- [29] IEvangelist et al., eds. *Logging in C# and .NET*. 17th July 2024. URL: <https://learn.microsoft.com/en-us/dotnet/core/extensions/logging> (visited on 09/03/2025).
- [30] IEvangelist et al., eds. *Make HTTP requests with the HttpClient class*. 8th Mar. 2025. URL: <https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient> (visited on 10/03/2025).
- [31] ingen084. 【初心者向け】*KyoshinMonitorLib* チュートリアル. 9th Dec. 2020. URL: <https://qiita.com/ingen084/items/b9961576d1af1b140f10>.
- [32] ingen084. *KyoshinShindoPlaceEditor*. Version 0.0.6.0. 2018. URL: <https://github.com/ingen084/KyoshinShindoPlaceEditor/>.
- [33] ingen084. *ShindoObsPoints.mpk.lz4*. 2024. URL: <https://github.com/ingen084/KyoshinEewViewerIngen/blob/develop/src/KyoshinEewViewer/Assets/ShindoObsPoints.mpk.lz4> (visited on 08/03/2025).
- [34] ingen084. 強震モニタの画像から揺れていることを検知する. 14th Dec. 2022. URL: <https://qiita.com/ingen084/items/82985e8d3227c97c608d> (visited on 13/10/2024).
- [35] ingen084. 強震モニタの画像から震度と地点を特定するまで. 2nd Dec. 2020. URL: <https://qiita.com/ingen084/items/7e91f8da2996972ac586#fnref2> (visited on 13/10/2024).
- [36] JGraph. *diagrams.net, draw.io*. Version 15.5.2. 2021. URL: <https://www.diagrams.net/>.
- [37] O. Kopp, C. C. Snethlage and C. Schwentker. ‘JabRef: BibTeX-based literature management software’. In: *TUGboat* 44.3 (138 2023), pp. 441–447. ISSN: 0896-3207. DOI: 10.47397/tb/44-3/tb138kopp-jabref.
- [38] F. Lundh, J. A. Clark and et al., eds. *Pillow (PIL Fork) 11.0.0 documentation*. 1995. URL: <https://pillow.readthedocs.io/en/stable/> (visited on 07/10/2024).
- [39] Mamma Mia Dev. *Avalonia UI - 03 - Sidebar Menu*. 3rd July 2024. URL: <https://www.youtube.com/watch?v=UDbKVheMBY8> (visited on 10/03/2025).
- [40] MammaMiaDev. *avalonia-the-series*. 2024. URL: <https://github.com/MammaMiaDev/avaloniaui-the-series>.
- [41] Mapsui, ed. *API*. URL: <https://mapsui.com/v5/api/Mapsui.html> (visited on 10/03/2025).
- [42] Mapsui, ed. *Logging*. URL: <https://mapsui.com/documentation/logging.html> (visited on 10/03/2025).
- [43] Mapsui. *Mapsui*. Version 5.0.0 Beta 9. 10th Mar. 2025. URL: <https://github.com/Mapsui/Mapsui>.
- [44] Mapsui, ed. *Mapsui Samples*. URL: <https://mapsui.com/samples/> (visited on 10/03/2025).
- [45] Mapsui, ed. *Mapsui Samples*. URL: <https://mapsui.com/v5/samples/> (visited on 10/03/2025).

- [46] Mapsui, ed. *Projections*. URL: <https://mapsui.com/documentation/projections.html> (visited on 10/03/2025).
- [47] Matthias247. *Choosing a buffer size for a WebSocket response*. Ed. by Johan. 29th Jan. 2017. URL: <https://stackoverflow.com/a/41926694> (visited on 22/11/2024).
- [48] Microsoft, ed. *BindingFlags Enum*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.reflection.bindingflags?view=net-9.0> (visited on 10/03/2025).
- [49] Microsoft, ed. *ClientWebSocket Class*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.net.websockets.clientwebsocket?view=net-9.0> (visited on 22/11/2024).
- [50] Microsoft, ed. *HttpClient Class*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=net-9.0> (visited on 01/12/2024).
- [51] Microsoft, ed. *HttpListener Class*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.net.httplistener?view=net-9.0> (visited on 01/12/2024).
- [52] Microsoft, ed. *HttpRequest Class*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.net.http.httprequestmessage?view=net-9.0> (visited on 01/12/2024).
- [53] Microsoft, ed. *HttpRequestMessage Class*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.net.http.httprequestmessage?view=net-9.0> (visited on 01/12/2024).
- [54] Microsoft, ed. *MethodInfo Class*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.reflection.methodinfo?view=net-9.0> (visited on 10/03/2025).
- [55] Microsoft, ed. *Type.GetMethods Method*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.type.getmethods?view=net-9.0> (visited on 10/03/2025).
- [56] Microsoft, ed. *Uri Class*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.uri?view=net-9.0> (visited on 01/12/2024).
- [57] Microsoft, ed. *UriBuilder Class*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.uribuilder?view=net-9.0> (visited on 01/12/2024).
- [58] NoneType1 (フランソワ). 多項式補間を使用して強震モニタ画像から数値データを決定する. 22nd Dec. 2020. URL: <https://qiita.com/NoneType1/items/a4d2cf932e20b56ca444> (visited on 07/10/2024).
- [59] OpenAI ChatGPT-4o. *Response on Connecting to a WebSocket in C#*. 22nd Nov. 2024. URL: <https://openai.com/chatgpt> (visited on 22/11/2024).
- [60] Overleaf. *Bibliography management with biblatex*. URL: https://www.overleaf.com/learn/latex/Bibliography_management_with_biblatex (visited on 10/12/2024).
- [61] Overleaf. *Code Highlighting with minted*. URL: https://www.overleaf.com/learn/latex/Code_Highlighting_with_minted (visited on 14/10/2024).
- [62] SciPy community, ed. *SciPy documentation*. 2008. URL: <https://docs.scipy.org/doc/scipy/> (visited on 14/10/2024).
- [63] Sergio0694, niels9001 and michael-hawker, eds. *Introduction to the MVVM Toolkit*. 7th Nov. 2024. URL: <https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/> (visited on 10/03/2025).
- [64] TheSeeker. *Find the next TCP port in .NET*. 29th Sept. 2008. URL: <https://stackoverflow.com/a/150974/> (visited on 03/12/2024).
- [65] YoryelNathan. *Process.Start(url) fails*. 4th Apr. 2020. URL: <https://stackoverflow.com/a/61035650/> (visited on 01/12/2024).
- [66] こんぽ. 地震情報アプリ界隈 Advent Calendar 2021. 2021. URL: <https://adventar.org/calendars/6729> (visited on 15/10/2024).
- [67] こんぽ. 地震界隈 Advent Calendar 2020. 2020. URL: <https://adventar.org/calendars/5903> (visited on 15/10/2024).
- [68] こんぽ. 防災アプリ Advent Calendar 2022. 2022. URL: <https://adventar.org/calendars/7488> (visited on 15/10/2024).
- [69] こんぽ. 防災アプリ開発 Advent Calendar 2023. 2023. URL: <https://adventar.org/calendars/9301> (visited on 15/10/2024).

- [70] こんぽ. 防災アプリ開発 *Advent Calendar 2024*. 2024. URL: <https://adventar.org/calendars/9939> (visited on 10/12/2024).
- [71] 気象庁 / Japan Meterological Agency. *JMA2001 走時表*. 2024. URL: https://www.data.jma.go.jp/svd/eqev/data/bulletin/catalog/appendix/trttime/trt_j.html (visited on 17/10/2024).
- [72] 気象庁 / Japan Meterological Agency. 予報区等 *GIS* データ. 2001. URL: <https://www.data.jma.go.jp/developer/gis.html> (visited on 14/03/2025).
- [73] 気象庁 / Japan Meterological Agency, ed. 気象庁ホームページにおける気象情報の配色に関する設定指針. July 2020. URL: https://www.jma.go.jp/jma/kishou/info/colorguide/HPColordGuide_202007.pdf (visited on 10/03/2025).
- [74] 気象庁 / Japan Meterological Agency, ed. 気象庁防災情報 XML フォーマット 技術資料. 9th Dec. 2022. URL: https://xml.kishou.go.jp/tec_material.html (visited on 10/12/2024).
- [75] 気象庁 / Japan Meterological Agency, ed. 走時表フォーマット / *Time Table Format*. URL: https://www.data.jma.go.jp/svd/eqev/data/bulletin/catalog/appendix/trttime/tttfmt_j.html (visited on 17/10/2024).
- [76] 気象庁 / Japan Meterological Agency and dmdata.jp. *Sample Data*. URL: <https://sample.dmdata.jp/> (visited on 10/03/2025).
- [77] 防災科研 / National Research Institute for Earth Science and Disaster Resilience. *K-NET & KiK-net: 観測点一覧*. URL: <https://www.kyoshin.bosai.go.jp/kyoshin/db/index.html> (visited on 10/03/2025).
- [78] 防災科研 / National Research Institute for Earth Science and Disaster Resilience. *NIED F-net*. 29th Mar. 2019. DOI: <https://doi.org/10.17598/NIED.0005>. URL: <http://www.fnet.bosai.go.jp/auth/dataset/>.
- [79] 防災科研 / National Research Institute for Earth Science and Disaster Resilience. *NIED Hi-net*. 27th Feb. 2019. DOI: <https://doi.org/10.17598/NIED.0003>. URL: <https://hinetwww11.bosai.go.jp/auth/download/cont/>.
- [80] 防災科研 / National Research Institute for Earth Science and Disaster Resilience. *NIED K-NET, KiK-net*. 28th Mar. 2019. DOI: <https://doi.org/10.17598/nied.0004>. URL: <http://www.kyoshin.bosai.go.jp/kyoshin/>.
- [81] 防災科研 / National Research Institute for Earth Science and Disaster Resilience. *NIED MOWLAS (Monitoring of Waves on Land and Seafloor)*. 29th Mar. 2019. DOI: <https://doi.org/10.17598/NIED.0009>. URL: <http://www.mowlas.bosai.go.jp/mowlas/>.