

CO565
Web Services

CW1

21611431

16th May 2018

Computing & Web Development
Buckinghamshire New University

Summary

A Research	2
1 Technology - Wider Context	3
1.1 Considerations on the challenges when building distributed systems	3
1.2 Different Distributed Computing technologies	4
B Report	9
2 Project Proposal	10
2.1 Project Name	10
2.2 Overview of the Project	10
2.3 External Web Services	11
2.4 Data Exposed By The Service	11
2.5 Database Schema	11
2.6 User Interface Ideas	11
3 Engineering Approach	18
3.1 Requirements specification	18
3.2 Designing	19
3.3 Implementation	19
3.4 Testing	19
4 Project Management	20
5 Development	21
5.1 Requirements elicitation	21
5.2 Design	22
5.3 Implementation	22
5.4 Testing	40
References	41

Part A

Research

Section 1

Technology - Wider Context

1.1 Considerations on the challenges when building distributed systems

When creating distributing systems, different challenges had to be overcome by developers in other to develop such systems:

- Inherent complexities: Inter-node communication needs different mechanisms than intra-node communication, as well as policies and protocols. Synchronization and coordination is more complicated, and problems such as latency, jitter, transient failures and overload are also problems to consider.
- Accidental complexities: Those happen because of limitations with software tools and development techniques. Most of these happen because of choices made when the time arose to build the system.
- Inadequate methods and techniques: For a long time, software development studies and practices have focused on single-process and single-threaded applications, which has lead to a lack of expertise on software techniques distributed systems.
- Continuous re-invention and re-discovery of core concepts and techniques: For a long time, time has been used on solving the same problems over and over again, instead of trying to develop techniques that would facilitate the reuse of code across different platforms, which lead to less time being available to invest in innovation.

1.2 Different Distributed Computing technologies

1.2.1 Ad hoc Network Programming

Ad hoc network programming was one of the first solutions to allow the development of distributed computing systems. That technology used Interprocess Communication mechanisms. In IPC, there was a use of shared memory, pipes and sockets, which lead to a big coupling between the application code and the socket API, as well as paradigms mismatches, since local communication techniques use object-oriented techniques and remote communications use function-oriented techniques.

1.2.2 Structured communication

This technology was an improvement to ad hoc since it doesn't couple application code to low-level IPC mechanisms, offers higher-level communication to distributed systems, encapsulates machine-level details and embodies types and communication style closer to the application domain. One particular example of this technology are Remote Procedure Call platforms. They have the following characteristics:

- Allow distributed applications to cooperate with one another by:
 - Invoking functions on each other
 - Passing parameters along with each invocation
 - Receiving results from the function that was invoked

With this technique, however, components are still aware of their peers' remoteness, therefore it does not fulfil the following distributed systems requirements:

- Location-independence of components
- Flexible component deployment

1.2.3 Middleware

Middleware is used as a bridge between the remote system and the application, being a distribution infrastructure software. It resides between an application and the operating system, network or database. Different types of middleware have appeared over time, which are listed below.

1.2.3.1 Distributed Object Computing

Also known as DOC, it represents the confluence of two major information technologies: RPC-based distributed computing systems and object-oriented design and programming.

COBRA 2.x and Java RMI are examples of DOC. They focus on interfaces, which are contracts between clients and servers that define a location-independent means for clients to view and access object services provided by a server. Such technologies also define communication protocols and object information models to enable interoperability.

Their key limitations are:

- Lack of functional boundaries: This type of middleware treats all interfaces as client/server contracts and don't provide standard assembly mechanisms to decouple dependencies among collaborating object implementations. This leads to a requirement for the explicit discovery and connection to objects that are only dependencies for the actual object that the application is trying to reach.
- Lack of software deployment and configuration standards: Results in systems that are harder to maintain and software component implementations that are much harder to reuse.

1.2.3.2 Component middleware

This technology emerged to address the following limitations of DOC:

- Lack of functional boundaries: Allows a group of cohesive component objects to interact with each other through multiple provided and required interfaces and defines standard runtime mechanisms needed to execute these component objects in generic application servers.
- Lack of standard deployment and configuration mechanisms: Component middleware specifies the infrastructure to packages, which makes it easier to customize, assemble and disseminate components throughout a distributed system.

Examples of this technology are Enterprise JavaBeans and COBRA Component Model.

For this technology, a set of rules and definitions has been applied:

- Component is an implementation entity that exposes a set of named interfaces and connection points that components use to communicate with each other.
- Containers provide the server runtime environment for component implementations. Contains various pre-defined hooks and operations that give components access to strategies and services, such as persistence, event notification, transaction,

replication, load balancing and security. Each container is also responsible for initializing and providing runtime contexts for the managed components.

Component middleware also automates aspects of various stages in the application development lifecycle like component implementation, packaging, assembly and deployment. This approach enables developers to create applications more rapidly and robustly than DOC.

1.2.3.3 Publish/Subscribe and Message-Oriented Services

RPC, DOC and component middleware are all based on request/response communication. Aspects of this type of communication are:

- Synchronous communication: The client waits for a response and stops the rest of its process
- Designated communication: The client must know the identity of the server, which leads to tight coupling between the application code and the API
- Point-to-Point communication: Client talks to just one server at a time

In order to fix the problems that that type of communication created for some systems, the following alternatives were created:

- Message-Oriented Middleware: Applied in technologies such as IBM MQ Series, BEA MessageQ and TIBCO Rendezvous
- Publish/Subscribe middleware: Applied in technologies such as Java Messaging Service (JMS), Data Distribution Service (DDS) and WS-NOTIFICATION

Specifications of Message-Oriented:

- Support for asynchronous communication
- Transactional properties

On top of that, Publish/Subscribe also allows:

- Anonymous communication (loose coupling)
- Group communication

On the Publish/Subscribe paradigm:

- Publishers are the source of events. They may need to describe the type of event they generate sometimes
- Subscribers are the event sinks; they consume data on topics of interest to them. They may need to declare filtering information sometimes
- Event channels: Components in the system that propagate events. They can perform services such as filtering and routing, Quality of Service enforcement and fault management.

In these types of systems, events can be represented in many ways, and the interfaces can also be generic or specialized.

1.2.3.4 Service-Oriented Architectures and Web Services

Service-Oriented Architecture (SOA) is a style of organizing and utilizing distributed capabilities that may be controlled by different organizations and owners. It provides a uniform means to offer, discover, interact with and use capabilities of loosely coupled and interoperable software services to support the requirements of the business processes and application users.

From the emergence of this technology along with the increasing popularity of the World Wide Web, SOAP was created. It is a protocol to exchange XML-based messages over a network, normally using HTTP. SOAP spawned a popular new variant of SOA called Web Services. It allows developers to package application logic into services whose interfaces are described with the Web Service Description Language (WSDL). WSDL-based services are often accessed using higher-level Internet protocols. They can also be used to build an Enterprise Service Bus(ESB), which is a distributed computing architecture that simplifies interworking between disparate systems.

Web services have established themselves as the technology of choice for most enterprise business applications. They complement earlier successful middleware technologies and provide standard mechanisms for interoperability. Examples of Web Services technologies are:

- Microsoft Windows Component Foundation (WCF)
- Service Component Architecture (SCA)

Web Services combine aspects of component-based development and Web technologies. They also provide black-box functionality that can be described and reused without concern for how a service is implemented. They are not accessed using the object model-specific protocols, but rather using Web protocols and data formats.

Web service technologies focus on middleware integration & allow component reuse across an organization's entire application set, regardless of the technology implemented for those. In general, web services make it relatively easy to reuse and share common logic with such diverse clients as mobile, desktop and web applications. The broad reach of web services is possible because they rely on open standards that are ubiquitous, interoperable across different computing platforms and independent of the underlying execution technologies.

All web services use HTTP and leverage data-interchange standards like XML and JSON, as well as common media types, but differ on the way they may use HTTP:

- As an application protocol to define standard service behaviours
- HTTP as a transport mechanism to convey data

In Web services, there are two prominent design models:

- SOAP
 - Language, platform and transport independent
 - Works well in distributed enterprise environments
 - Provides significant pre-build extensibility in the form of WS* standards
 - Built-in error handling
 - XML message-format only
 - Automation when used with certain language products
 - Uses WSDL as a description language
- REST
 - Uses easy to understand standards like swagger and OpenAPI Specification 3.0
 - Can use different types of message formatting like JSON
 - Fast (no extensive processing required)
 - Close to other Web technologies in design philosophy
 - Uses WADL as a description language

Part B

Report

Section 2

Project Proposal

2.1 Project Name

The name for the project is CryptoStats

2.2 Overview of the Project

The aim of the project is to provide users with relevant information regarding cryptocurrency exchanges. By analysing the data provided by the external API on the values of the cryptocurrencies, the service will provide the user with information about the patterns in the value of the cryptocurrency and data such as the average number of days that the currency's value declines and increases in one go, if there is any resemblance in the fluctuation between the currency and other major currencies.

The goal with the project is to be able to provide users with enough information to help them make informed decisions when doing an exchange for a particular coin, and also gather information about what makes cryptocurrencies fluctuate in value. Rather than just showing the value of the coin, the service aims to provide way more valuable information by analysing the data in the currency's history, which allows the user to possess very important and essential information when trading coins.

The information processed will be presented to the users in two different ways: The first in a web site that will display the data in a user-friendly manner, built using ASP.NET Core 2 for the back-end and Angular and Bootstrap for the front-end. The second is by presenting the processed information to the user through a RESTful API with a JSON response, in order to facilitate the integration of the information with other applications or services.

For the submission of the project, I only intend to provide information for major coins

such as Bitcoin, Ethereum, Litecoin and Bitcoin Cash, since they are the most known and there is many information and exchanges that trade these coins.

2.3 External Web Services

In order to obtain information about the coins, I will be using the [GDAX exchange API](#).

2.4 Data Exposed By The Service

The service aims to expose the following data:

- Average fluctuation for the last day, week, month, and year
- Average time when in growth
- Average time when in decline
- Difference between highest value and latest value

2.5 Database Schema

2.6 User Interface Ideas

The main concept for the User Interface of the web page is for it to be displayed on a single page, and using Angular components allow the interface to be responsive and show the right information that the user is looking for without having to load the page again or another page.

The first sketches for the interface are the following:

The Web application will have some additional pages explaining the project and how it came about, as well as a form allowing users to leave messages and an API page explaining how to work with the API.

The following images are wireframes that were produced based on the sketches:

Figure 2.1: Database Schema

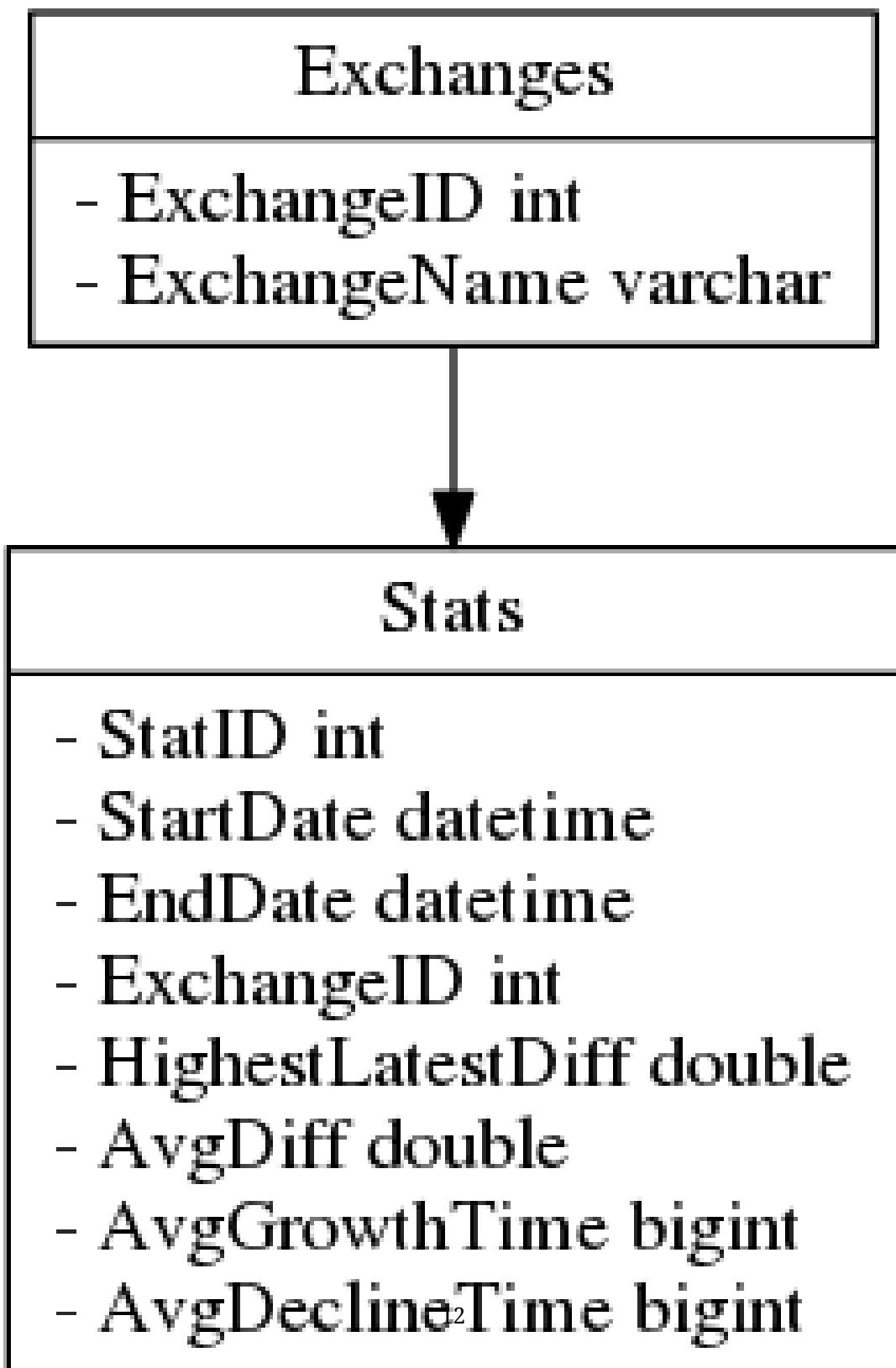


Figure 2.2: 1st Sketches

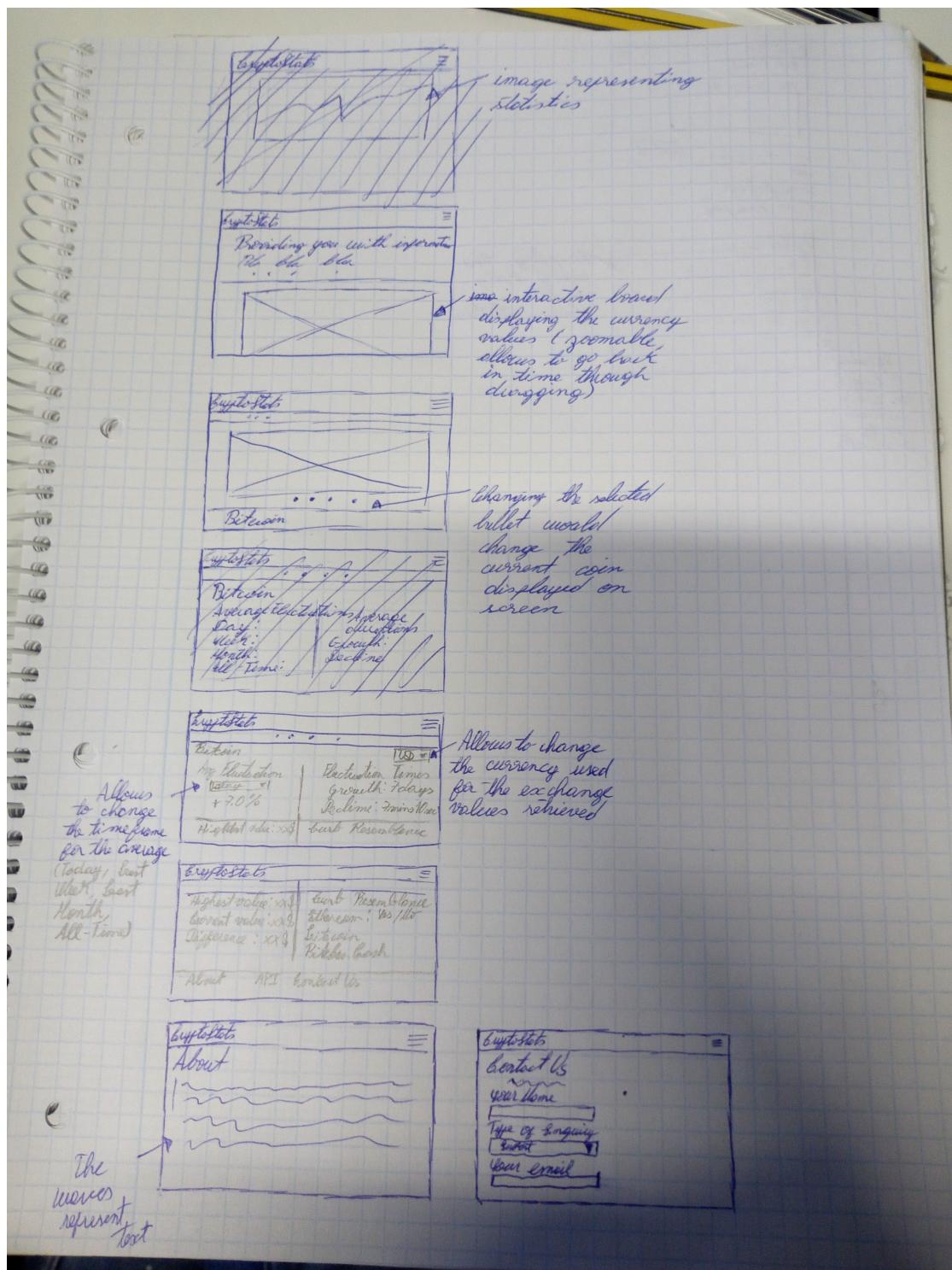


Figure 2.3: 2nd Sketches

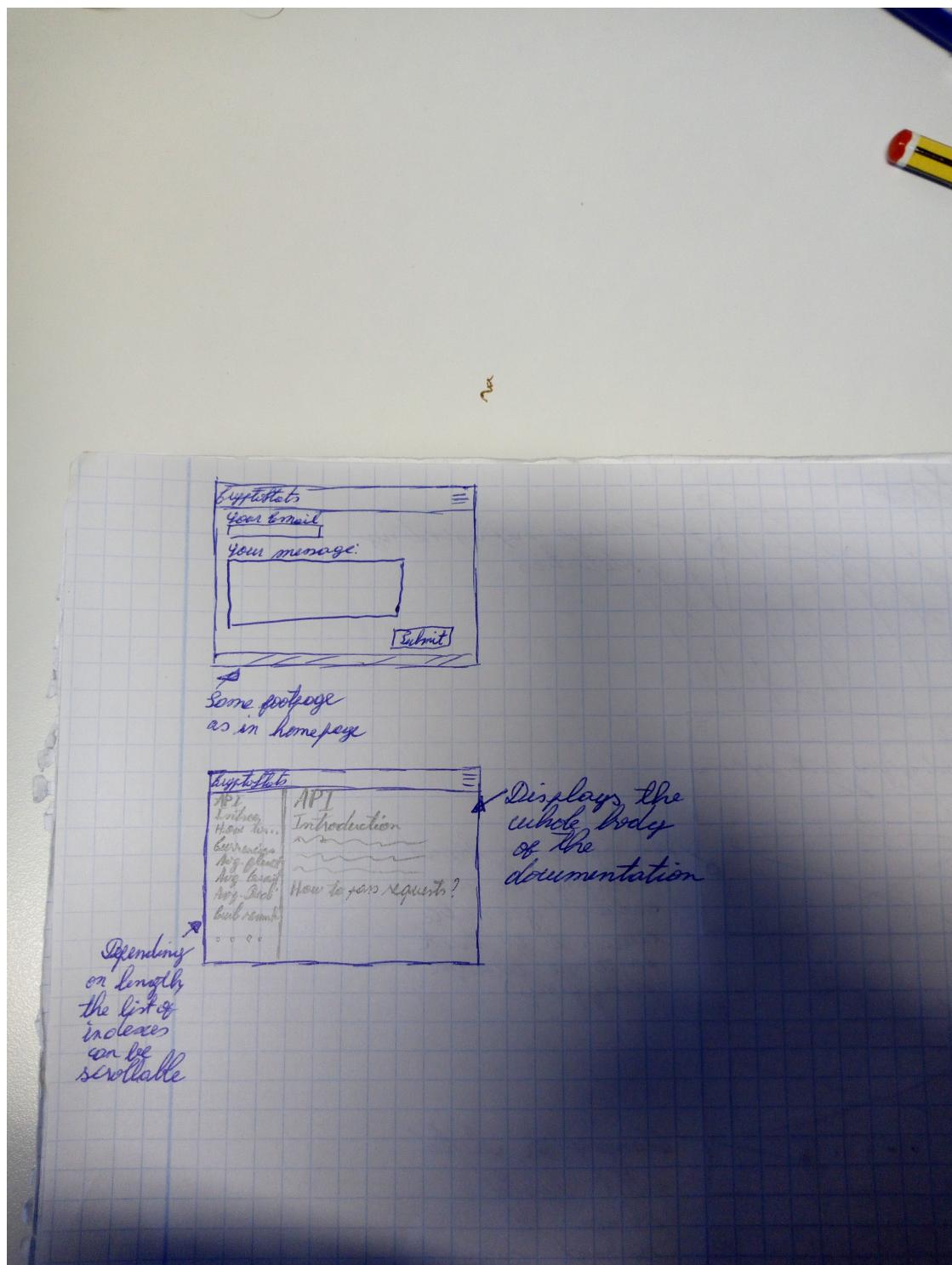


Figure 2.4: Homepage

The screenshot shows the top portion of the CryptoStats homepage. At the top left is the logo "CryptoStats". To the right is a blue navigation bar with three horizontal lines indicating a menu. Below the logo is a section titled "Providing you with information you need". Underneath this title is a paragraph of placeholder text (Lorem ipsum) followed by a large blue rectangular placeholder area labeled "Market values". At the bottom of this placeholder are four small circular icons.

Figure 2.5: Homepage Bottom

This screenshot shows the bottom section of the CryptoStats homepage for the "Bitcoin" category. It features a grid layout with several data points:

Average Fluctuation	Fluctuation Times
Today <input type="button" value="▼"/>	Growth: 7 days
+ 7%	Decline: 7 mins 10 secs
Value Differences	Curb Resemblance
Highest value: xx\$	Ethereum: Yes
Current value: xy\$	Litecoin: No
Difference: xz\$	Bitcoin Cash: Yes

At the bottom of the page, there is a blue footer bar with links for "About", "API", and "Contact Us".

Figure 2.6: About Page

The screenshot shows the 'About' section of the CryptoStats website. The page has a blue header bar with the 'CryptoStats' logo on the left and a three-line menu icon on the right. Below the header, the word 'About' is highlighted in bold. A large amount of placeholder text (Lorem ipsum) follows, divided into two paragraphs. At the bottom of the page is a blue footer bar with three white links: 'About', 'API', and 'Contact Us'.

Figure 2.7: Contact Us Page

The screenshot shows the 'Contact Us' section of the CryptoStats website. The page has a blue header bar with the 'CryptoStats' logo on the left and a three-line menu icon on the right. Below the header, the word 'Contact Us' is highlighted in bold. A message to the user is displayed: 'We would like you to be able to contact us if you have questions regarding the service we provide and to give us feedback. Please fill in the form below to leave us a message!'. The form consists of several input fields: 'Your Name:' with a text input field, 'Your Email:' with a text input field, 'Type of Enquiry:' with a dropdown menu set to 'Suggestion', and 'Your Message:' with a large text area. A blue 'Submit' button is located at the bottom right of the form. At the very bottom of the page is a blue footer bar with three white links: 'About', 'API', and 'Contact Us'.

Figure 2.8: API Page

CryptoStats

API

Introduction

Introduction

Currencies

Average Fluctuations

Average Growth

Average Decline

Curb Resemblance

Nunc ac ipsum rhoncus mauris viverra finibus euismod vel dolor. Curabitur quis purus condimentum, dapibus arcu sit amet, tempor metus. Vestibulum egestas, erat eget vehicula placerat, metus nisl aliquam ex, nec consequat ex arcu eu mi. Donec tellus libero, ultricies a sagittis eu, finibus quis nisl. Aenean vulputate, justo nec pellentesque laoreet, tortor dolor tristique tortor, nec iaculis turpis ipsum nec lacus. In vehicula magna id risus consectetur, ut pharetra lectus tincidunt. Quisque vestibulum in enim a blandit. Fusce et velit fermentum, aliquam tortor id, fringilla urna.

Phasellus accumsan lacinia aliquam. Praesent dapibus aliquam diam, at varius ante tristique ac. Ut aliquet sodales fermentum. Fusce blandit odio non lacus accumsan venenatis. Aliquam eu dui at leo rutrum elementum. Ut egestas eu nibh ac egestas. Curabitur sit amet luctus velit. Aliquam diam tortor, vehicula eget nulla at, pretium sagittis libero. Nulla ullamcorper convallis metus, in euismod metus dapibus ut. Morbi vel ligula vitae diam vehicula posuere ut id leo. Integer metus dolor, pellentesque quis justo non, venenatis tincidunt sapien. Suspendisse potenti.

About API Contact Us

Section 3

Engineering Approach

For this project, I worked in an agile and iterative manner, based on the SCRUM methodology.

3.1 Requirements specification

When specifying the requirements for the project, I started by choosing what I wanted the project to offer to the users initially. I started by thinking of the core functionality that I wanted for the project and what data I wanted my API to expose to the user. After deciding on the data that would be exposed, I started specifying the requirements that would lead to the exposure of the data, by doing a list of the core functionality that the API would have:

- Display all the exchanges and statistics
- Display exchanges and statistics by id
- Display statistics by start date
- Display statistics by end date
- Display statistics by start and end date
- Display statistics related to a single exchange
- Display statistics related to a single exchange by id
- Display statistics related to a single exchange by start date
- Display statistics related to a single exchange by end date
- Display statistics related to a single exchange by start and end date

Most of this functionality would allow the user to retrieve different series of data by the criteria that they choose (id and or range of dates) and the type of data that they are looking for (exchange or statistics).

3.2 Designing

I started designing the implementation of the features on paper, by writing down the parts of the algorithms that would implement the features that were the most difficult to implement.

In order to implement the features, I chose to go with a Model-Controller approach, where the data would be initially stored and retrieved through the models and the business logic and presentation of the data would occur in the controller. The business logic that would allow the API to seed the database and add data to it would also work in the models.

3.3 Implementation

For the implementation, I started by implementing the models and the database context that would allow the connection to the database, storage and retrieval of the data.

After that, I implemented the controllers and added the functionality that would display the JSON results to the user.

After all of that was done, I moved on to implement the features that would retrieve data from the input API and process it. For that, I studied and modified a library client that was available for the API I was using, so that only the functions that I needed for my project would be available, and created new namespaces for those functionalities.

After implementing the seeding and storage of new data functinoalities, I started implementing the front-end of my project. Based on my previous UI design and wireframes, I used the planned technologies to build a front-end that would resemble the original design as much as possible.

3.4 Testing

This project was throughly tested as it was being built. As soon as a new feature would be implemented, the project would be tested to make sure that everything was working as expected and that everything was also behaving and looking in an expected way.

Section 4

Project Management

In order to manage the project, the code as well as the project tasks were stored in a GitHub repository.

That choice was made because I already had previous experience storing my code and repositories in GitHub but also because the way GitHub does project management was very suitable for the approach that I was using to develop the project.

Using SCRUM terms, GitHub facilitates the planning of sprints and tasks by its use of issues and milestones, but also because it makes it very easy to keep track of how the project is evolving through the use of Kanban boards. It is also possible to assign the different tasks to different sprints and to users, so that everyone involved in the project knows where the project is and what they need to do.

In this particular case, I developed the whole project myself, but I still found it very useful to use the features provided by GitHub, especially because it allows me to comment on the tasks as I am getting them done, allowing me to have a log of the progress that I have made over time, as well as register the difficulties that I had to face and how I overcame them.

The project code and tasks can be found [here](#).

Section 5

Development

5.1 Requirements elicitation

5.1.1 API

- Display all stored exchanges and statistics
- Display exchanges with a specific id
- Display statistics with a specific id
- Display the statistics starting at a specific date
- Display the statistics ending at a specific date
- Display the statistics with a specific start and end dates
- Fetch all the statistics related to a specific exchange
- Fetch the statistic related to a particular exchange with a specific id
- Fetch all the statistics related to a specific exchange with a specific start date
- Fetch all the statistics related to a specific exchange with a particular end date
- Fetch a statistic related to a particular exchange with specific start and end dates

5.1.2 Front-End

- Display for a particular exchange:
 - Average fluctuation
 - Average time in growth

- Average time in decline
- Difference between highest and latest values
- Allow display of information for a period of the last day, week, month and year.

5.2 Design

5.3 Implementation

Here is the code for the API:

5.3.1 Main Program

Figure 5.1: Program.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using Microsoft.AspNetCore;
7  using Microsoft.AspNetCore.Hosting;
8  using Microsoft.Extensions.DependencyInjection;
9  using Microsoft.Extensions.Configuration;
10 using Microsoft.Extensions.Logging;
11 using CryptoStats.Models;

12 namespace CryptoStats
13 {
14     public class Program
15     {
16         public static void Main(string[] args)
17         {
18             var host = BuildWebHost(args);

19             using (var scope = host.Services.CreateScope())
20             {
21                 var services = scope.ServiceProvider;

22                 try
23                 {
24                     SeedData.Initialize(services);

```

```

25     }
26     catch (Exception ex)
27     {
28         var logger = services.GetRequiredService<ILogger<Program>>();
29         logger.LogError(ex, "An error occurred seeding the DB.");
30         Console.WriteLine(ex.Message);
31     }
32 }

33     host.Run();
34 }

35     public static IWebHost BuildWebHost(string[] args) =>
36         WebHost.CreateDefaultBuilder(args)
37             .UseStartup<Startup>()
38             .Build();
39     }
40 }
```

Figure 5.2: Startup.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Builder;
6  using Microsoft.AspNetCore.Hosting;
7  using Microsoft.AspNetCore.HttpsPolicy;
8  using Microsoft.AspNetCore.Mvc;
9  using Microsoft.Extensions.Configuration;
10 using Microsoft.Extensions.DependencyInjection;
11 using Microsoft.Extensions.Logging;
12 using Microsoft.Extensions.Options;
13 using Microsoft.EntityFrameworkCore;
14 using Microsoft.EntityFrameworkCore.Sqlite;
15 using CryptoStats.Models;

16 namespace CryptoStats
17 {
18     public class Startup
19     {
20         public Startup(IConfiguration configuration)
21         {
22             Configuration = configuration;
23         }
```

```

24     public IConfiguration Configuration { get; }
25
26     // This method gets called by the runtime. Use this method to add services to the container.
27     public void ConfigureServices(IServiceCollection services)
28     {
29         services.AddDbContext<CryptoContext>(opt => opt.UseSqlite("Data Source=app.db"));
30         services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
31
32     // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
33     public void Configure(IApplicationBuilder app, IHostingEnvironment env)
34     {
35         if (env.IsDevelopment())
36         {
37             app.UseDeveloperExceptionPage();
38         }
39         else
40         {
41             app.UseHsts();
42         }
43
44         app.UseHttpsRedirection();
45         app.UseMvc();
46     }
47 }

```

Figure 5.3: appsettings.json

```

1  {
2      "ConnectionStrings": {
3          "DefaultConnection": "DataSource=app.db"
4      },
5      "Logging": {
6          "LogLevel": {
7              "Default": "Warning"
8          }
9      },
10     "AllowedHosts": "*"
11 }

```

5.3.2 Models

Figure 5.4: Exchange.cs

```

1  using System.Collections.Generic;
2
3  namespace CryptoStats.Models
4  {
5      public class Exchange
6      {
7          public int ExchangeId{get;set;}
8          public string Name{get;set;}
9
10         //Setting list for foreign key
11         public List<Stat> Stats{get;set;}
12     }
13 }
```

Figure 5.5: Stat.cs

```

1  using System;
2
3  namespace CryptoStats.Models
4  {
5      public class Stat
6      {
7          public int StatId{get;set;}
8          public DateTime startDate {get;set;}
9          public DateTime endDate{get;set;}
10         public decimal HighestLatestDiff{get;set;}
11         public decimal avgDiff {get;set;}
12         public TimeSpan avgGrowthTime {get;set;}
13         public TimeSpan avgDeclineTime {get;set;}
14
15         //Setting variables for foreign key
16         public int ExchangeId{get;set;}
17         public Exchange Exchange {get;set;}
18     }
19 }
```

Figure 5.6: SeedData.cs

```

1  using Microsoft.EntityFrameworkCore;
2  using Microsoft.Extensions.DependencyInjection;
3  using System;
4  using System.Linq;
5  using System.Collections.Generic;
6  using CryptoStats.GDAX;
7  using CryptoStats.GDAX.Services.Products.Models;
```

```

8  using CryptoStats.GDAX.Services.Products.Types;
9  using CryptoStats.GDAX.Shared.Types;

10 namespace CryptoStats.Models
11 {
12     public static class SeedData
13     {
14         private static GDAXClient client = new GDAXClient();
15         public static void Initialize(IServiceProvider serviceProvider)
16         {
17             using(var context = new CryptoContext(serviceProvider.GetRequiredService<ICryptoContext>()))
18             {
19                 if(context.Exchanges.Any() && context.Stats.Any())
20                 {
21                     return;
22                 }
23
24                 if(context.Exchanges.Any() == false)
25                 {
26                     var products = client.ProductsService.GetAllProductsAsync().Result;
27                     foreach(Product prod in products)
28                     {
29                         Console.WriteLine(prod.Id.ToString());
30                         context.Exchanges.Add(new Exchange(){Name = prod.Id.ToString()});
31                     }
32                     context.SaveChanges();
33                 }
34
35                 if(!context.Stats.Any())
36                 {
37                     foreach(Exchange e in context.Exchanges)
38                     {
39                         seedStat(e.ExchangeId, DateTime.Now.AddDays(-7), DateTime.Now);
40                     }
41                 }
42
43             }
44
45             public static void seedStat(int exchangeID, DateTime startDate, DateTime endDate)
46             {
47                 using(context)
48                 {
49                     foreach(Product p in client.ProductsService.GetAllProductsAsync().Result)
50                     {
51                         if(p.Id == exchangeID)
52                         {
53                             var stats = context.Stats.Where(s => s.ExchangeId == p.Id).ToList();
54                             if(stats.Count == 0)
55                             {
56                                 var stat = new Stat{ExchangeId = p.Id, Date = startDate, Value = 0};
57                                 context.Stats.Add(stat);
58                             }
59                             else
60                             {
61                                 var stat = stats[0];
62                                 stat.Date = endDate;
63                                 stat.Value = 1;
64                             }
65                         }
66                     }
67                 }
68             }
69
70             public static void Main()
71             {
72                 Initialize();
73             }
74
75             static void Main(string[] args)
76             {
77                 Main();
78             }
79
80             static void Main()
81             {
82                 Main();
83             }
84
85             static void Main()
86             {
87                 Main();
88             }
89
90             static void Main()
91             {
92                 Main();
93             }
94
95             static void Main()
96             {
97                 Main();
98             }
99
100            static void Main()
101            {
102                Main();
103            }
104
105            static void Main()
106            {
107                Main();
108            }
109
110            static void Main()
111            {
112                Main();
113            }
114
115            static void Main()
116            {
117                Main();
118            }
119
120            static void Main()
121            {
122                Main();
123            }
124
125            static void Main()
126            {
127                Main();
128            }
129
130            static void Main()
131            {
132                Main();
133            }
134
135            static void Main()
136            {
137                Main();
138            }
139
140            static void Main()
141            {
142                Main();
143            }
144
145            static void Main()
146            {
147                Main();
148            }
149
150            static void Main()
151            {
152                Main();
153            }
154
155            static void Main()
156            {
157                Main();
158            }
159
160            static void Main()
161            {
162                Main();
163            }
164
165            static void Main()
166            {
167                Main();
168            }
169
170            static void Main()
171            {
172                Main();
173            }
174
175            static void Main()
176            {
177                Main();
178            }
179
180            static void Main()
181            {
182                Main();
183            }
184
185            static void Main()
186            {
187                Main();
188            }
189
190            static void Main()
191            {
192                Main();
193            }
194
195            static void Main()
196            {
197                Main();
198            }
199
200            static void Main()
201            {
202                Main();
203            }
204
205            static void Main()
206            {
207                Main();
208            }
209
210            static void Main()
211            {
212                Main();
213            }
214
215            static void Main()
216            {
217                Main();
218            }
219
220            static void Main()
221            {
222                Main();
223            }
224
225            static void Main()
226            {
227                Main();
228            }
229
230            static void Main()
231            {
232                Main();
233            }
234
235            static void Main()
236            {
237                Main();
238            }
239
240            static void Main()
241            {
242                Main();
243            }
244
245            static void Main()
246            {
247                Main();
248            }
249
250            static void Main()
251            {
252                Main();
253            }
254
255            static void Main()
256            {
257                Main();
258            }
259
260            static void Main()
261            {
262                Main();
263            }
264
265            static void Main()
266            {
267                Main();
268            }
269
270            static void Main()
271            {
272                Main();
273            }
274
275            static void Main()
276            {
277                Main();
278            }
279
280            static void Main()
281            {
282                Main();
283            }
284
285            static void Main()
286            {
287                Main();
288            }
289
290            static void Main()
291            {
292                Main();
293            }
294
295            static void Main()
296            {
297                Main();
298            }
299
300            static void Main()
301            {
302                Main();
303            }
304
305            static void Main()
306            {
307                Main();
308            }
309
310            static void Main()
311            {
312                Main();
313            }
314
315            static void Main()
316            {
317                Main();
318            }
319
320            static void Main()
321            {
322                Main();
323            }
324
325            static void Main()
326            {
327                Main();
328            }
329
330            static void Main()
331            {
332                Main();
333            }
334
335            static void Main()
336            {
337                Main();
338            }
339
340            static void Main()
341            {
342                Main();
343            }
344
345            static void Main()
346            {
347                Main();
348            }
349
350            static void Main()
351            {
352                Main();
353            }
354
355            static void Main()
356            {
357                Main();
358            }
359
360            static void Main()
361            {
362                Main();
363            }
364
365            static void Main()
366            {
367                Main();
368            }
369
370            static void Main()
371            {
372                Main();
373            }
374
375            static void Main()
376            {
377                Main();
378            }
379
380            static void Main()
381            {
382                Main();
383            }
384
385            static void Main()
386            {
387                Main();
388            }
389
390            static void Main()
391            {
392                Main();
393            }
394
395            static void Main()
396            {
397                Main();
398            }
399
400            static void Main()
401            {
402                Main();
403            }
404
405            static void Main()
406            {
407                Main();
408            }
409
410            static void Main()
411            {
412                Main();
413            }
414
415            static void Main()
416            {
417                Main();
418            }
419
420            static void Main()
421            {
422                Main();
423            }
424
425            static void Main()
426            {
427                Main();
428            }
429
430            static void Main()
431            {
432                Main();
433            }
434
435            static void Main()
436            {
437                Main();
438            }
439
440            static void Main()
441            {
442                Main();
443            }
444
445            static void Main()
446            {
447                Main();
448            }
449
450            static void Main()
451            {
452                Main();
453            }
454
455            static void Main()
456            {
457                Main();
458            }
459
460            static void Main()
461            {
462                Main();
463            }
464
465            static void Main()
466            {
467                Main();
468            }
469
470            static void Main()
471            {
472                Main();
473            }
474
475            static void Main()
476            {
477                Main();
478            }
479
480            static void Main()
481            {
482                Main();
483            }
484
485            static void Main()
486            {
487                Main();
488            }
489
490            static void Main()
491            {
492                Main();
493            }
494
495            static void Main()
496            {
497                Main();
498            }
499
500            static void Main()
501            {
502                Main();
503            }
504
505            static void Main()
506            {
507                Main();
508            }
509
510            static void Main()
511            {
512                Main();
513            }
514
515            static void Main()
516            {
517                Main();
518            }
519
520            static void Main()
521            {
522                Main();
523            }
524
525            static void Main()
526            {
527                Main();
528            }
529
530            static void Main()
531            {
532                Main();
533            }
534
535            static void Main()
536            {
537                Main();
538            }
539
540            static void Main()
541            {
542                Main();
543            }
544
545            static void Main()
546            {
547                Main();
548            }
549
550            static void Main()
551            {
552                Main();
553            }
554
555            static void Main()
556            {
557                Main();
558            }
559
560            static void Main()
561            {
562                Main();
563            }
564
565            static void Main()
566            {
567                Main();
568            }
569
570            static void Main()
571            {
572                Main();
573            }
574
575            static void Main()
576            {
577                Main();
578            }
579
580            static void Main()
581            {
582                Main();
583            }
584
585            static void Main()
586            {
587                Main();
588            }
589
590            static void Main()
591            {
592                Main();
593            }
594
595            static void Main()
596            {
597                Main();
598            }
599
600            static void Main()
601            {
602                Main();
603            }
604
605            static void Main()
606            {
607                Main();
608            }
609
610            static void Main()
611            {
612                Main();
613            }
614
615            static void Main()
616            {
617                Main();
618            }
619
620            static void Main()
621            {
622                Main();
623            }
624
625            static void Main()
626            {
627                Main();
628            }
629
630            static void Main()
631            {
632                Main();
633            }
634
635            static void Main()
636            {
637                Main();
638            }
639
640            static void Main()
641            {
642                Main();
643            }
644
645            static void Main()
646            {
647                Main();
648            }
649
650            static void Main()
651            {
652                Main();
653            }
654
655            static void Main()
656            {
657                Main();
658            }
659
660            static void Main()
661            {
662                Main();
663            }
664
665            static void Main()
666            {
667                Main();
668            }
669
670            static void Main()
671            {
672                Main();
673            }
674
675            static void Main()
676            {
677                Main();
678            }
679
680            static void Main()
681            {
682                Main();
683            }
684
685            static void Main()
686            {
687                Main();
688            }
689
690            static void Main()
691            {
692                Main();
693            }
694
695            static void Main()
696            {
697                Main();
698            }
699
700            static void Main()
701            {
702                Main();
703            }
704
705            static void Main()
706            {
707                Main();
708            }
709
710            static void Main()
711            {
712                Main();
713            }
714
715            static void Main()
716            {
717                Main();
718            }
719
720            static void Main()
721            {
722                Main();
723            }
724
725            static void Main()
726            {
727                Main();
728            }
729
730            static void Main()
731            {
732                Main();
733            }
734
735            static void Main()
736            {
737                Main();
738            }
739
740            static void Main()
741            {
742                Main();
743            }
744
745            static void Main()
746            {
747                Main();
748            }
749
750            static void Main()
751            {
752                Main();
753            }
754
755            static void Main()
756            {
757                Main();
758            }
759
760            static void Main()
761            {
762                Main();
763            }
764
765            static void Main()
766            {
767                Main();
768            }
769
770            static void Main()
771            {
772                Main();
773            }
774
775            static void Main()
776            {
777                Main();
778            }
779
780            static void Main()
781            {
782                Main();
783            }
784
785            static void Main()
786            {
787                Main();
788            }
789
790            static void Main()
791            {
792                Main();
793            }
794
795            static void Main()
796            {
797                Main();
798            }
799
800            static void Main()
801            {
802                Main();
803            }
804
805            static void Main()
806            {
807                Main();
808            }
809
810            static void Main()
811            {
812                Main();
813            }
814
815            static void Main()
816            {
817                Main();
818            }
819
820            static void Main()
821            {
822                Main();
823            }
824
825            static void Main()
826            {
827                Main();
828            }
829
830            static void Main()
831            {
832                Main();
833            }
834
835            static void Main()
836            {
837                Main();
838            }
839
840            static void Main()
841            {
842                Main();
843            }
844
845            static void Main()
846            {
847                Main();
848            }
849
850            static void Main()
851            {
852                Main();
853            }
854
855            static void Main()
856            {
857                Main();
858            }
859
860            static void Main()
861            {
862                Main();
863            }
864
865            static void Main()
866            {
867                Main();
868            }
869
870            static void Main()
871            {
872                Main();
873            }
874
875            static void Main()
876            {
877                Main();
878            }
879
880            static void Main()
881            {
882                Main();
883            }
884
885            static void Main()
886            {
887                Main();
888            }
889
890            static void Main()
891            {
892                Main();
893            }
894
895            static void Main()
896            {
897                Main();
898            }
899
900            static void Main()
901            {
902                Main();
903            }
904
905            static void Main()
906            {
907                Main();
908            }
909
910            static void Main()
911            {
912                Main();
913            }
914
915            static void Main()
916            {
917                Main();
918            }
919
920            static void Main()
921            {
922                Main();
923            }
924
925            static void Main()
926            {
927                Main();
928            }
929
930            static void Main()
931            {
932                Main();
933            }
934
935            static void Main()
936            {
937                Main();
938            }
939
940            static void Main()
941            {
942                Main();
943            }
944
945            static void Main()
946            {
947                Main();
948            }
949
950            static void Main()
951            {
952                Main();
953            }
954
955            static void Main()
956            {
957                Main();
958            }
959
960            static void Main()
961            {
962                Main();
963            }
964
965            static void Main()
966            {
967                Main();
968            }
969
970            static void Main()
971            {
972                Main();
973            }
974
975            static void Main()
976            {
977                Main();
978            }
979
980            static void Main()
981            {
982                Main();
983            }
984
985            static void Main()
986            {
987                Main();
988            }
989
990            static void Main()
991            {
992                Main();
993            }
994
995            static void Main()
996            {
997                Main();
998            }
999
1000 static void Main()
1001 {
1002    Main();
1003 }
```

```

48     System.Threading.Thread.Sleep(3000);
49     if(p.Id.ToString().Equals(context.Exchanges.Find(exchangeID).Name)
50     {
51         TimeSpan t = endDate.Subtract(startDate);
52         int days = (int)t.TotalDays;
53         for(int i = 0; i < days; i++)
54         {
55             System.Threading.Thread.Sleep(3000);
56             var candles = client.ProductsService.GetHistoricR
57             System.Threading.Thread.Sleep(1000);
58             Stat adder = new Stat(){
59                 startDate = candles.First().Time,
60                 endDate = candles.Last().Time
61             };
62
63             //calculate highestLatestDiff
64             decimal high = 0.00M;
65             foreach(Candle c in candles)
66             {
67                 if(c.High > high)
68                 {
69                     high = (decimal)c.High;
70                 }
71             }
72             adder.HighestLatestDiff = (decimal)candles.Last()
73
74             //calculate avgDiff
75             //Use difference between open and close of candle
76             decimal candlesDiff = 0.00M;
77             foreach(Candle c in candles)
78             {
79                 candlesDiff += Convert.ToDecimal(c.Close - c.
80             }
81             adder.avgDiff = candlesDiff / candles.Count();
82
83             //calculate average growth and decline times
84             //store candles in a temporary list to retrieve t
85             //growth and decline times are stored in a list o
86             //average is calculated afterwards
87             List<TimeSpan> growthTimes = new List<TimeSpan>()
88             List<TimeSpan> declineTimes = new List<TimeSpan>()
89             bool? inGrowth = null;
90             List<Candle> growthCandles = new List<Candle>();
91
92             foreach(Candle c in candles)
93             {
94                 if(c.High > high)
95                 {
96                     high = (decimal)c.High;
97                     growthCandles.Add(c);
98                 }
99                 else
100                 {
101                     declineTimes.Add(endDate - c.Time);
102                     declineCandles.Add(c);
103                 }
104             }
105             adder.GrowthAvg = growthTimes.Average();
106             adder.DeclineAvg = declineTimes.Average();
107             adder.GrowthCandles = growthCandles;
108             adder.DeclineCandles = declineCandles;
109         }
110     }
111 }
```

```

89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
{
    if(c.Close > c.Open && inGrowth == false)
    {
        inGrowth = true;
        declineTimes.Add(growthCandles.Last().Time);
        growthCandles.Clear();
        growthCandles.Add(c);
    }
    else if((c.Close > c.Open && inGrowth == true))
    {
        growthCandles.Add(c);
        inGrowth = true;
    }
    else if(c.Close < c.Open && inGrowth == true)
    {
        inGrowth = false;
        growthTimes.Add(growthCandles.Last().Time);
        growthCandles.Clear();
        growthCandles.Add(c);
    }
    else if((c.Close < c.Open && inGrowth == false))
    {
        growthCandles.Add(c);
        inGrowth = false;
    }
}
if(inGrowth == true)
{
    growthTimes.Add(growthCandles.Last().Time.Subtract(
}
else if (inGrowth == false)
{
    declineTimes.Add(growthCandles.Last().Time.Subtract(
}

double totalTimes = 0;
foreach(TimeSpan time in growthTimes)
{
    totalTimes += t.TotalMilliseconds;
}
adder.avgGrowthTime = TimeSpan.FromMilliseconds(totalTimes);

totalTimes = 0;
foreach(TimeSpan time in declineTimes)
{

```

```

132                         totalTimes += t.TotalMilliseconds;
133                     }
134                 adder.avgDeclineTime = TimeSpan.FromMilliseconds(
135                     context.Stats.Add(add);
136                 startDate.AddDays(1);
137             }

138             startDate.AddDays(-days);
139             for(int i = 0; i < days; i++)
140             {
141                 System.Threading.Thread.Sleep(1000);
142                 var candles = client.ProductsService.GetHistoricR
143                 System.Threading.Thread.Sleep(1000);
144                 Stat adder = new Stat(){
145                     startDate = candles.First().Time,
146                     endDate = candles.Last().Time
147                 };

148             //calculate highestLatestDiff
149             decimal high = 0.00M;
150             foreach(Candle c in candles)
151             {
152                 if(c.High > high)
153                 {
154                     high = (decimal)c.High;
155                 }
156             }
157             adder.HighestLatestDiff = (decimal)candles.Last();

158             //calculate avgDiff
159             //Use difference between open and close of candle
160             decimal candlesDiff = 0.00M;
161             foreach(Candle c in candles)
162             {
163                 candlesDiff += (decimal)(c.Close - c.Open);
164             }
165             adder.avgDiff = candlesDiff / candles.Count();

166             //calculate average growth and decline times
167             //store candles in a temporary list to retrieve t
168             //growth and decline times are stored in a list o
169             //average is calculated afterwards
170             List<TimeSpan> growthTimes = new List<TimeSpan>()
171             List<TimeSpan> declineTimes = new List<TimeSpan>()
172             bool inGrowth = false;

```

```

173     List<Candle> growthCandles = new List<Candle>();
174
175     foreach(Candle c in candles)
176     {
177         if(c.Close > c.Open && inGrowth == false)
178         {
179             inGrowth = true;
180             declineTimes.Add(growthCandles.Last().Time);
181             growthCandles.Clear();
182             growthCandles.Add(c);
183         }
184         else if(c.Close > c.Open && inGrowth == true)
185         {
186             growthCandles.Add(c);
187         }
188         else if(c.Close < c.Open && inGrowth == true)
189         {
190             inGrowth = false;
191             growthTimes.Add(growthCandles.Last().Time);
192             growthCandles.Clear();
193             growthCandles.Add(c);
194         }
195         else if(c.Close < c.Open && inGrowth == false)
196         {
197             growthCandles.Add(c);
198         }
199         if(inGrowth)
200         {
201             growthTimes.Add(growthCandles.Last().Time.Subtract(
202             growthCandles.First().Time));
203         }
204         else
205         {
206             declineTimes.Add(growthCandles.Last().Time.Subtract(
207             growthCandles.First().Time));
208         }
209
210         double totalTimes = 0;
211         foreach(TimeSpan time in growthTimes)
212         {
213             totalTimes += t.TotalMilliseconds;
214         }
215         adder.avgGrowthTime = TimeSpan.FromMilliseconds(totalTimes / growthTimes.Count);
216
217         totalTimes = 0;
218         foreach(TimeSpan time in declineTimes)
219         {
220             totalTimes += t.TotalMilliseconds;
221         }
222         adder.avgDeclineTime = TimeSpan.FromMilliseconds(totalTimes / declineTimes.Count);
223
224     }
225
226     return adder;
227 }

```

```

215             {
216                 totalTimes += t.TotalMilliseconds;
217             }
218             adder.avgDeclineTime = TimeSpan.FromMilliseconds(
219                 context.Stats.Add(add);
220                 endDate.AddDays(-1);
221             }
222             context.SaveChanges();
223         }
224     }
225 }
226 }
227 }
228 }
```

Figure 5.7: DbContext.cs

```

1  using Microsoft.EntityFrameworkCore;
2
3  namespace CryptoStats.Models
4  {
5      public class CryptoContext: DbContext
6      {
7          public CryptoContext(DbContextOptions<CryptoContext> options) : base(options)
8          {
9
10             public DbSet<Exchange> Exchanges{get;set;}
11             public DbSet<Stat> Stats{get;set;}
12         }
13     }
```

5.3.3 Controllers

Figure 5.8: ExchangesController.cs

```

1  using Microsoft.AspNetCore.Mvc;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System;
5  using CryptoStats.Models;
6
7  namespace CryptoStats.Controllers
8  {
9      [Route("api/[controller]")]
10 }
```

```

9     [ApiController]
10    public class ExchangesController : ControllerBase
11    {
12        private readonly CryptoContext _context;
13
14        public ExchangesController(CryptoContext context)
15        {
16            _context = context;
17        }
18
19        [HttpGet]
20        public ActionResult<List<Exchange>> GetAll()
21        {
22            return _context.Exchanges.ToList();
23        }
24
25        [HttpGet("{id}", Name="GetExchange")]
26        public ActionResult<Exchange> GetById(int id)
27        {
28            var item = _context.Exchanges.Find(id);
29            if(item == null)
30            {
31                return NotFound();
32            }
33            return item;
34        }
35    }

```

Figure 5.9: ExchangeStatsController.cs

```

1  using Microsoft.AspNetCore.Mvc;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System;
5  using CryptoStats.Models;
5
6  namespace CryptoStats.Controllers
7  {
8      [Route("api/")]
9      [ApiController]
10     public class ExchangeStatsController : ControllerBase
11     {
12         private readonly CryptoContext _context;

```

```

13     public ExchangeStatsController(CryptoContext context)
14     {
15         _context = context;
16     }
17
18     [HttpGet("exchanges/{id}/stats")]
19     public ActionResult<List<Stat>> GetAll(int id)
20     {
21         var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
22         if(exchange == null)
23         {
24             return NotFound();
25         }
26         else
27         {
28             return exchange.Stats;
29         }
30     }
31
32     [HttpGet("exchanges/{id}/stats/id/{statId}")]
33     public ActionResult<Stat> GetByID(int id, int statId)
34     {
35         var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
36         if(exchange == null)
37         {
38             return NotFound();
39         }
40         else
41         {
42             return exchange.Stats.Where(s => s.StatId == statId).First();
43         }
44     }
45
46     [HttpGet("exchanges/{id}/stats/startDate/{startDate}")]
47     public ActionResult<List<Stat>> GetByStartDate(int id, DateTime startDate)
48     {
49         var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
50         if(exchange == null)
51         {
52             return NotFound();
53         }
54         else
55         {
56             return exchange.Stats.Where(s => s.startDate.Equals(startDate)).ToList();
57         }
58     }

```

```

55     }
56
57     [HttpGet("exchanges/{id}/stats/endDate/{endDate}")]
58     public ActionResult<List<Stat>> GetByEndDate(int id, DateTime endDate)
59     {
60         var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
61         if(exchange == null)
62         {
63             return NotFound();
64         }
65         else
66         {
67             return exchange.Stats.Where(s => s.endDate.Equals(endDate)).ToList();
68         }
69
70     [HttpGet("exchanges/{id}/stats/dates/{startDate}/{endDate}")]
71     public ActionResult<Stat> GetByDates(int id, DateTime startDate, DateTime endDate)
72     {
73         var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
74         if(exchange == null)
75         {
76             return NotFound();
77         }
78         else
79         {
80             return exchange.Stats.Where(s => s.startDate.Equals(startDate) && s.endDate.Equals(endDate)).FirstOrDefault();
81         }
82     }
83 }

```

Figure 5.10: StatsController.cs

```

1  using Microsoft.AspNetCore.Mvc;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System;
5  using CryptoStats.Models;

6  namespace CryptoStats.Controllers
7  {
8      [Route("api/[controller]")]
9      [ApiController]
10     public class StatsController : ControllerBase

```

```

11     {
12         private readonly CryptoContext _context;
13
14         public StatsController(CryptoContext context)
15         {
16             _context = context;
17         }
18
19         [HttpGet]
20         public ActionResult<List<Stat>> GetAll()
21         {
22             return _context.Stats.ToList();
23         }
24
25         [HttpGet("/id/{id}")]
26         public ActionResult<Stat> GetById(int id)
27         {
28             var item = _context.Stats.Find(id);
29             if(item == null)
30             {
31                 return NotFound();
32             }
33             return item;
34         }
35
36         [HttpGet("/startDate/{startDate}")]
37         public ActionResult<List<Stat>> GetByStartDate(DateTime startDate)
38         {
39             var items = _context.Stats.Where(s => s.startDate.Equals(startDate)).ToList();
40             if(items == null)
41             {
42                 return NotFound();
43             }
44             return items;
45         }
46
47         [HttpGet("/endDate/{endDate}")]
48         public ActionResult<List<Stat>> GetByEndDate(DateTime endDate)
49         {
50             var items = _context.Stats.Where(s => s.endDate.Equals(endDate)).ToList();
51             if(items == null)
52             {
53                 return NotFound();
54             }
55             return items;

```

```

51     }

52     [HttpGet("/dates/{startDate}/{endDate}")]
53     public ActionResult<Stat> GetByDates(DateTime startDate, DateTime endDate)
54     {
55         var item = _context.Stats.Where(s => s.startDate.Equals(startDate) && s.e
56         if(item == null)
57         {
58             return NotFound();
59         }
60         return item;
61     }
62 }
63 }
```

5.3.4 GDAX Client

Figure 5.11: GDAXClient.cs

```

1  using System;
2  using CryptoStats.GDAX.Network.HttpClient;
3  using CryptoStats.GDAX.Network.HttpRequest;
4  using CryptoStats.GDAX.Services.Products;
5  using CryptoStats.GDAX.Shared.Utilities.Clock;
6  using CryptoStats.GDAX.Shared.Utilities.Queries;

7  namespace CryptoStats.GDAX
8  {
9      public class GDAXClient
10     {
11         public GDAXClient(bool sandBox = false) : this( new HttpClient(), sandBox){}

12         public GDAXClient(IHttpClient httpClient, bool sandBox = false)
13         {
14             var clock = new Clock();
15             httpRequestMessageService = new HttpRequestMessageService(clock, sand
16             var queryBuilder = new QueryBuilder();

17             ProductsService = new ProductsService(httpClient, httpRequestMessageService
18         }

19         public ProductsService ProductsService{get;}
20     }
21 }
```

Figure 5.12: ProductsService.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Globalization;
4  using System.Linq;
5  using System.Net.Http;
6  using System.Threading.Tasks;
7  using CryptoStats.GDAX.Services;
8  using CryptoStats.GDAX.Network.HttpClient;
9  using CryptoStats.GDAX.Network.HttpRequest;
10 using CryptoStats.GDAX.Services.Products.Models;
11 using CryptoStats.GDAX.Services.Products.Models.Responses;
12 using CryptoStats.GDAX.Services.Products.Types;
13 using CryptoStats.GDAX.Shared.Types;
14 using CryptoStats.GDAX.Shared.Utilities.Extensions;
15 using CryptoStats.GDAX.Shared.Utilities.Queries;

16 namespace CryptoStats.GDAX.Services.Products
17 {
18     public class ProductsService: AbstractService
19     {
20         private readonly IQueryBuilder queryBuilder;

21         public ProductsService(IHttpClient httpClient, IHttpRequestMethodService http
22         {
23             this.queryBuilder = queryBuilder;
24         }

25         public async Task<List<Product>> GetAllProductsAsync()
26         {
27             return await SendServiceCall<List<Product>>(HttpMethod.Get, "/products");
28         }

29         public async Task<ProductsOrderBookResponse> GetProductOrderBookAsync(
30             ProductType productId,
31             ProductLevel productLevel = ProductLevel.One)
32         {
33             var productsOrderBookJsonResponse = await SendServiceCall<ProductsOrderBo
34             var productOrderBookResponse = ConvertProductOrderBookResponse(productsOrde

35             return productOrderBookResponse;
36         }

37         public async Task<ProductTicker> GetProductTickerAsync(ProductType productId)
```

```
38
39     {
40         return await SendServiceCall<ProductTicker>(HttpMethod.Get, $"products/{productType}/ticker");
41     }
42
43     public async Task<ProductStats> GetProductStatsAsync(ProductType productId)
44     {
45         return await SendServiceCall<ProductStats>(HttpMethod.Get, $"products/{productId}/stats");
46     }
47
48     public async Task<IList<IList<ProductTrade>>> GetTradesAsync(
49         ProductType productId,
50         int limit = 100,
51         int numberofPages = 0)
52     {
53         var httpResponseMessage = await SendHttpRequestMessagePagedAsync<ProductTrade>(productId, limit, numberofPages);
54
55         return httpResponseMessage;
56     }
57
58     public async Task<IList<Candle>> GetHistoricRatesAsync(
59         ProductType productPair,
60         DateTime start,
61         DateTime end,
62         CandleGranularity granularity)
63     {
64         const int maxPeriods = 300;
65
66         var candleList = new List<Candle>();
67
68         DateTime? batchEnd = end;
69         DateTime batchStart;
70
71         var maxBatchPeriod = (int)granularity * maxPeriods;
72
73         do
74         {
75             if (batchEnd == null) {
76                 break;
77             }
78
79             batchStart = batchEnd.Value.AddSeconds(-maxBatchPeriod);
80             if (batchStart < start) batchStart = start;
81
82             candleList.AddRange(await GetHistoricRatesAsync(productPair, batchStart, batchEnd));
83         }
84
85         return candleList;
86     }
87 }
```

```

72         batchEnd = candleList.LastOrDefault()?.Time;
73
74     } while (batchStart > start);
75
76     return candleList;
77 }
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109

    private async Task<IList<Candle>> GetHistoricRatesAsync(
        ProductType productId,
        DateTime start,
        DateTime end,
        int granularity)
    {
        var isoStart = start.ToString("s");
        var isoEnd = end.ToString("s");

        var queryString = queryBuilder.BuildQuery(
            new KeyValuePair<string, string>("start", isoStart),
            new KeyValuePair<string, string>("end", isoEnd),
            new KeyValuePair<string, string>("granularity", granularity.ToString()));

        return await SendServiceCall<IList<Candle>>(HttpMethod.Get, $""/products/{productId}/candles?{queryString}");
    }

    private ProductsOrderBookResponse ConvertProductOrderBookResponse(
        ProductsOrderBookJsonResponse productsOrderBookJsonResponse,
        ProductLevel productLevel)
    {
        var askList = productsOrderBookJsonResponse.Asks.Select(product => product
        {
            OrderId = productLevel == ProductLevel.Three
                ? new Guid(askArray[2])
                : (Guid?)null,
            NumberOfOrders = productLevel == ProductLevel.Three
                ? (decimal?)null
                : Convert.ToDecimal(askArray[2], CultureInfo.InvariantCulture)
        }).ToArray();

        var bidList = productsOrderBookJsonResponse.Bids.Select(product => product
        {
            OrderId = productLevel == ProductLevel.Three
                ? new Guid(bidArray[2])
                : (Guid?)null,
            NumberOfOrders = productLevel == ProductLevel.Three
                ? (decimal?)null

```

```

110             : Convert.ToDecimal(bidArray[2], CultureInfo.InvariantCulture)
111         });
112 
113         var productOrderBookResponse = new ProductsOrderBookResponse(productsOrderBook);
114         return productOrderBookResponse;
115     }
116 }

```

For the GDAX Client, an instance of HttpClient, and JSON deserializers have been used to send HTTP requests to the input API and deserialize the data sent back from the request. The output API was built using REST technology, and sends JSON responses that the system automatically serializes from the ActionResults that the controllers return.

5.4 Testing

Testing as been done alongside the build of the project, as features were being implemented to make sure that everything was being working as it was supposed to.

Testing as been done on different browsers (Google Chrome, Chromium, Firefox) and also using Sqliteman to make sure that the database was being populated properly.

For debugging, when the API wouldn't throw an exception to the browser, but the application wouldn't behave as expected, then I would use the debugger from Visual Studio Code for C# and put breakpoints in the most critical and error-prone parts of the code until I stumbled upon the exception or behaviour that would be making the application behave in an unexcepted manner, to then later on fix it.

References