SWP 2024/25 Crash & Burn

Leitfaden für Änderungen

2. Februar 2025

Im Entwurf der Anwendung wurde auf Modularität und Austauschbarkeit geachtet. Gewährleistet wird das durch die Aufteilung der Funktionalität auf Provider und die Einführung von Datentypen zur Strukturierung der Anwendungsdaten. So wird die Erweiterung und Änderung der Anwendung einheitlich und relativ einfach. Bei Anpassungen sollte darauf geachtet werden, dieses Modell beizubehalten. Die beste Anpassung ist die, die nicht erforderlich ist, darum wurde z.B. darauf geachtet, dass sich die Zahl der Sensorboards und PBRs (sowie die Zahl der von ihnen verbauten Sensoren) flexibel ändern kann ohne eine händische Anpassung zu benötigen.

Im Anhang befindet sich eine Übersicht über die implementierten Provider, die deren Zusammenwirken verdeutlicht.

Änderungen an der Anwendung sind hauptsächlich vorzunehmen, wenn Annahmen, die während der Entwicklung der Anwendung getroffen wurden, später auf geänderte Anforderungen treffen und dementsprechend angepasst werden müssen. Pro Quelltext-Einheit (Bibliothek) wird hier eine Liste von Annahmen mit Hinweisen zur Anpassung an geänderte Anforderungen angegeben.

1 Getroffene Annahmen

1.1 AQI-Provider

lib/application/providers/aqi_provider.dart

Berechnungsverfahren des AQI

Der AQI wird derzeit aus den Durchschnittswerten der Messgrößen O_2, CO_2, CO, Luftfeuchtigkeit und O_3 durch Anwendung einer stückweise linearen Funktion gebildet. Die folgenden möglichen Änderungen sind derzeit denkbar:

- Parameter der stückweise linearen Funktionen können angepasst werden
  - Parameter sind für jede Messgröße einzeln angegeben
- Andere Zuordnungen von einer Messgröße zum AQI
  - Jede Zuordnung wird als Funktion `int messgroesseAqi(double concentration)` angegeben
- Neue Messgrößen werden hinzugefügt
  - Es wird eine neue Funktion `int messgroesseAqi(double concentration)` benötigt
  - Die Funktion und ein aufruf auf `watch` des jeweiligen Providers muss in `aqi` und in `aqiCurrent` hinterlegt werden

Heuristik für Verlaufs-Relevanz

- `_addAqi` könnte um eine bessere Heuristik ergänzt werden, ob ein neu berechneter Wert relevant genug ist, um in den Verlauf aufgenommen zu werden

1.2 Chat-Provider

lib/application/providers/chat_provider.dart

Größe des Ringpuffers

Während der Entwicklung der Anwendung wurde davon ausgegangen, dass ein Verlauf über 1000 Nachrichten ausreichen sollte. Diese Größe kann hier beliebig angepasst werden.

Chat per MQTT

MQTT ist einer eher ungewöhnliche Entscheidung für ein Protokoll zur Übertragung von Textnachrichten und Binärdateien. Die Unterstützung für ein anderes Protokoll könnte hier ergänzt werden. Dabei würde absehbar auch eine Änderung von `lib/models/chat_message.dart` notwendig.

1.3 Influx Init Provider

lib/application/providers/influx_init_provider.dart

Anzahl der geladenen Datenpunkte

Derzeit werden für jede Zeitreihe aus der InfluxDB die letzten 1000 Messpunkte geladen. Dieser Wert lässt sich anpassen, indem die InfluxQL-Abfrage angepasst wird.

Verfahren zur Umbenennung von Topics

Topics werden derzeit umbenannt, wenn sie in die InfluxDB eingetragen werden, indem Schrägstriche (`/`) durch Unterstriche (`_`) ersetzt werden. Aus dem MQTT-Topic `board1/temp1_am` wird also die InfluxDB-Zeitreihe `board1_temp1_am`. Nun ist also nicht mehr ersichtlich, ob das ursprüngliche Topic `board1/temp1_am` oder `board1/temp1/am` war. Der Mechanismus in der Anwendung geht bisher nur vom ersten Fall aus, tauscht also nur den ersten Unterstrich zurück zum Schrägstrich. Wenn dieser Mechanismus sinnvollerweise auf Publisher-Seite und im Adapter von MQTT zur InfluxDB angepasst wird, muss die Rück-Umsetzung auch hier angepast werden.

Unterstützung für InfluxDB v2

Die Anwendung ist bisher nur auf Kompatibilität zur InfluxDB v1.7.4 ausgelegt und getestet. Unterstützung für InfluxDB v2 müsste hier ergänzt werden, sollte aber nur wenige Zeilen Code benötigen.

1.4 MQTT Updates Provider

lib/application/providers/mqtt_updates_provider.dart

WebSocket-Unterstützung

Aktuell wird nur eine MQTT-Verbindung per WebSocket unterstützt. Die Unterstützung für die Verbindung über „reines“ TCP/TLS könnte hier ergänzt werden.

1.5 Setpoint Rules Provider

lib/application/providers/setpoint_rules_provider.dart

Standardwerte wie in Aufgabenstellung

Sollten sich die Standardwerte für die Grenzwerte als untauglich herausstellen, können wie hier angepasst werden. Eine Änderung zur Laufzeit der Anwendung ist natürlich immer möglich. Neue Regeln können hier ebenfalls ergänzt werden, indem ein weiterer Eintrag zur Liste hinzugefügt wird. Dies wird insbesondere dann relevant, wenn neue Topics dazukommen, die nicht einfach nur einen Zähler im Topic ändern, sondern z.B. eine neue Messgröße darstellen.

1.6 Settings-Provider

lib/application/providers/settings_provider.dart

Klartext-Speicherung der Passwörter

Bisher werden alle Einstellungen im Klartext gespeichert, weil die Integration mit der verschlüsselten Speicherung nicht auf allen Plattformen zuverlässig funktioniert. Sobald diese Probleme behoben sind, könnte eine verschlüsselte Speicherung ergänzt werden

String und Integer als einzige Datentypen

Alle bisher benötigten Einstellungen können bereits als String abgebildet werden. Möglicherweise bieten sich noch bool und double als Erweiterungskandidaten an, sollten Einstellungen diese benötigen. Weiterhin müssen dann `lib/models/setting.dart` und `lib/presentation/pages/settings_general_page.dart` angepasst werden.

Standardwerte für die Einstellungen

Die Standardwerte der bisher vorhandenen Einstellungen können geändert werden, indem hier der jeweilge Eintrag in der Liste geändert wird. Neue Einstellungen können hinzugefügt werden, indem die Liste um Einträge ergänzt wird.

1.7 TimeSeriesNotifierProvider

lib/application/providers/timeseries_notifier_provider.dart

Stetig wachsender Datenspeicher

Bisher werden stets nur neue Daten hinzugefügt, keine Bestandsdaten gelöscht. Dank einiger Maßnahmen zur Optimierung der Berechnungen über die Daten funktioniert das auch mit weit über 1000 Messpunkten pro Zeitreihe noch gut. Wenn die Anwendung für eine unbeschränkte Zeit laufen soll, sollte hier ein Aufräummechanismus eingeführt werden.

1.8 VisualPrimaryStatusProvider

lib/application/providers/visual_primary_status_provider.dart

Statusanzeigen aus Aufgabenstellung

Falls die Statuslampen aus der Aufgabenstellung ergänzt werden sollen (z.B., weil eine neue Gerätekategorie eingeführt wurde) ist hier der richtige Ort dafür.

1.9 Marszeit-Hilfsmittel

lib/application/util/current_time.dart

Erdzeit ist unzureichend

Die aus Dart bekannten Funktionen zur Repräsentation und Verarbeitung von Zeitdaten sind für die Behandlung von Problemen mit Zeit auf dem Mars (andere Länge von Tagen und Jahren, Zeitdilatation) vollkommen unzureichend. Um nur an einer Stelle im Code die notwendigen Änderungen zu machen, wurde ein neuer Datentyp für Marszeitstempel eingeführt (MartianTimestamp). Alle Funktionen, die zwischen Zeitdarstellungen konvertieren sind hier abstrahiert und können den Anforderungen entsprechend angepasst werden.

1.10 Zeitreihenprovider-Schnittstelle

lib/application/time_series_provider_selector.dart

Drei Arten von Zeitreihen-Providern

Operationen auf Zeitreihen (z.B. den Durchschnitt zu bilden oder den AQI zu berechnen) können neue Zeitreihen ergeben. Um eine einheitliche Schnittstelle zu den entsprechenden Providern zu haben existiert diese Klasse. Wenn eine neue „Art“ von Zeitreihen-Providern eingeführt wird, müssen Anpassungen an den folgenden Stellen vorgenommen werden:

- Diese Klasse
- `lib/presentation/widgets/gauge_widget.dart`
- `lib/presentation/widgets/graph_widget.dart`

1.11 Datenmodell: Chatnachricht

lib/models/chat_message.dart

Untypisierte Übertragung beliebiger Daten

Bisher ist vorgesehen, dass alle Daten (also Textnachrichten, aber auch Binärdateien) ohne weitere Änderungen verschickt werden. Alternativ könnte hier ein Format zur Beschreibung der übertragenen Daten gewählt werden. Eine Möglichkeit könnte z.B. MIME (insbesondere multipart/form-data) sein.

Einfache Erkennungsheuristik

Derzeit werden die verschickten Daten anhand einiger Magic Bytes oder alternativ anhand der verwendeten Werte identifiziert. Falls nötig, könnte dieser Erkennungsmechanismus aufgebohrt werden.

1.12 Seite: Chat

lib/presentation/pages/chat_page.dart

Nur Bilder und PDF können angezeigt werden

Falls sich zukünftig die Notwendigkeit ergeben sollte, neben Textnachrichten, Bildern und PDF-Dokumenten weitere Medientypen anzuzeigen, könnte die Unterstützung dafür hier ergänzt werden. Es könnte außerdem sinnvoll sein, für die Betrachtung unbekannter Datentypen einen Hexadezimal-Betrachter einzubauen.

1.13 Widget: Graph

lib/presentation/widgets/graph_widget.dart

Zoom-Auflösung begrenzt auf 5 Minuten

Im Moment ist die Auflösung beim Hereinzoomen in Graphen so begrenzt, dass mindestens ein Verlauf von 5 Minuten angezeigt werden muss. Falls höherfrequente Daten dargestellt werden sollen, könnte dieser Wert an der entsprechenden Stelle verringert werden.

1.14 Widget: Habitat

lib/presentation/widgets/habitat.dart

3D-Modell aus „Supplemental Resources“

Wenn bauliche Veränderungen am Habitat vorgenommen werden, sollte auch das 3D-Modell mit der Viertelschnittansicht diese Änderungen reflektieren. Dafür kann das neue Modell zu den Assets hinzugefügt und hier ausgetauscht werden. Zu beachten ist dabei, dass die Anzahl der verwendeten Polygone nicht zu hoch sein darf, um den verwendeten Renderer nicht zu überfordern.

1.15 Widget: Hw3dDisplay

lib/presentation/widgets/hw_3d_display.dart

1.15.1 3D-Modelle aus „Supplemental Resources“

Die Modelle für den PBR und die Sensorboards können hier trivial durch andere Widgets ersetzt werden. Wenn neue HwCategory-Einträge dazukommen, sollten hier ebenfalls Ergänzungen vorgenommen werden.

1.16 Widget: Menü / Navigation

lib/presentation/widgets/main_menu.dart

5 Seiten innerhalb der Anwendung

Wenn neue Seiten zur Anwendung hinzugefügt werden sollen, muss hier und im `AppRouter` ein neuer Eintrag hinzugefügt werden. Außerdem können hier die verwendeten Icons geändert werden, sollte das nötig werden.

1.17 Widget: Mehrere Zeitreihen mit gemeinsamer Zeitachse

lib/presentation/widgets/multiple_time_series_display.dart

Festgelegte Höhe der Einträge in der Zeitreihenübersicht

Die genannte Höhe kann hier händisch angepasst werden, wenn sich die Anforderungen bzgl. der verwendeten Geräte ändern (z.B. für Hochkant-Bildschirme oder XR-Brillen).

1.18 main.dart

lib/main.dart

Initialisierung der Provider InfluxInitProvider, ChatProvider, MqttMessagesAdapterProvider

Weitere benötigte Initialisierungslogik kann hier platziert werden.

2 Komplexere Änderungen

Weitere Änderungen, die weitreichendere Einflüsse haben, werden im folgenden beschrieben.

2.1 Topic-Format ändern:

Die gesamte Anwendung ist darauf ausgelegt, Zeitreihendaten mit einem Topic zu markieren und sie dem Topic entsprechend zu behandeln. Wenn sich nun das Format der verwendeten Topics ändert, sind weitreichende Änderungen erforderlich. Insbesondere die folgenden Stellen müssen bedacht werden:

- InfluxInitProvider
  - Hier muss der Mechanismus angepasst werden, der InfluxDB-Zeitreihentitel in die Topics rücküberführt
- MqttUpdatesProvider
  - Im Moment werden alle Topics abonniert (`#`), um Sensoren nicht erst händisch in der Anwendung konfigurieren zu müssen. Ein naheliegender Weg zur Erweiterung könnte darin bestehen, zunächst die topics wie im folgenden Beispiel umzubenennen: `board1/temp1_am` → `board/1/temp_am/1`. Für jede `HwCategory` könnte dann eine Wildcard wie z.B. `board/#` abonniert werden.
- MqttTimeSeriesAdapterProvider

  - **lib/application/providers/mqtt_timeseries_adapter_provider.dart**
  - Topics, die tatsächlich Zeitreihen darstellen, werden hier anhand eines regulären Ausdrucks erkannt. Dieser Ausdruck muss dem neuen Format entsprechend angepasst werden.
- TimeSeriesNotifierProvider
  - Das Format der Topics ist hier dadurch festgeschrieben, dass die `HwInstance` ebenfalls durch einen regulären Ausdruck aus dem Topic bestimmt wird. Dieser muss ebenfalls angepasst werden.

2.2 Neue Geräteklassen / Sensortypen hinzufügen

Es wird angenommen, dass die topics weiterhin dem Format
<gerät><instanz>/<sensorbeschreibung> entsprechen werden.
Ansonsten müssen die Änderungen aus „Topic-Format ändern“ umgesetzt werden.

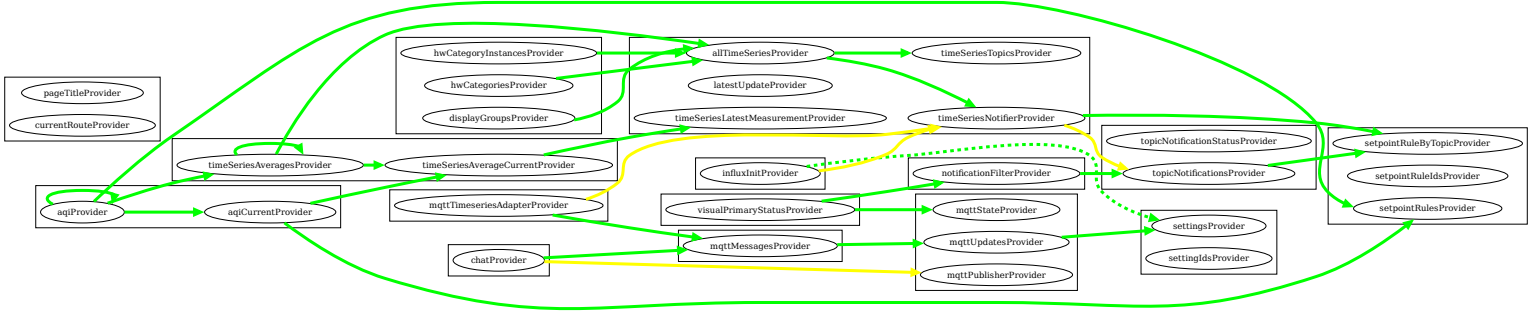
- MqttTimeSeriesAdapterProvider
  - Der reguläre Ausdruck, der hier feststellt, ob es sich bei einem Topic um eine für die Anwendung relevante Zeitreihe (und nicht um beliebige andere Daten) handelt muss um <gerät> ergänzt werden.
- HwCategory
  - **lib/models/display_group.dart**
  - Der Enum muss ergänzt werden um die neue Geräteklasse
  - Optional kann außerdem Hw3dDisplay um ein 3D-Modell für die Geräteklasse ergänzt werden
- DisplayGroup
  - **lib/models/display_group.dart**
  - Der Enum muss ggf. ergänzt werden um die Darstellungsgruppe des neuen Sensors, dabei wird die vorher definierte `HwCategory` angegeben
- SetpointRulesProvider
  - Eine Regel muss für jeden neuen Sensortypen, der angezeigt werden soll, definiert werden. Dabei muss ein RegExp zur Identifikation, ein Titel, optional eine Einheit, Standardgrenzwerte zur Alarmierung und die eben erstellte `DisplayGroup` angegeben werden.

3 Anhang

3.1 Provider-Übersicht

Die folgende Übersicht kann bei Änderungen an der Anwendung ggf. hilfreich sein, um zunächst einen Überblick über das Backend zu erhalten. Die Kästchen stellen je eine Bibliothek dar, die runden Blasen je einen Provider bzw. eine Provider-Familie. Grüne Pfeile bedeuten hier, dass ein Provider auf Statusänderungen eines anderen Providers reagiert (watch), grün gestrichelte Pfeile, dass der Wert des Providers einmalig gelesen wird (read) und gelbe Pfeile bedeuten, dass der notifier eines anderen Providers gelesen wird und mindestens eine Funktion darüber aufgerufen wird.

Die UI darf nun prinzipiell an jeder Stelle auf jeden Provider zugreifen, wobei die Provider jeweils für sich sicherstellen, dass kein invalider Zustand entstehen kann. Um eine relative Übersichtlichkeit zu gewährleisten, kann das hier allerdings nicht dargestellt werden.


pageTitleProvider
currentRouteProvider
hwCategoryInstancesProvider
hwCategoriesProvider
displayGroupsProvider
allTimeSeriesProvider
timeSeriesTopicsProvider
latestUpdateProvider
timeSeriesLatestMeasurementProvider
timeSeriesNotifierProvider
topicNotificationStatusProvider
topicNotificationsProvider
setpointRuleByTopicProvider
setpointRuleIdsProvider
setpointRulesProvider
timeSeriesAveragesProvider
timeSeriesAverageCurrentProvider
aqiProvider
aqiCurrentProvider
mqttTimeseriesAdapterProvider
influxInitProvider
notificationFilterProvider
visualPrimaryStatusProvider
mqttStateProvider
mqttUpdatesProvider
mqttMessagesProvider
mqttPublisherProvider
chatProvider
settingsProvider
settingIdsProvider