WETTERSTATION

Projekt des Wahlpflichtfaches 2022/23

Autor: Manuel Yates

INHALTSVERZEICHNIS

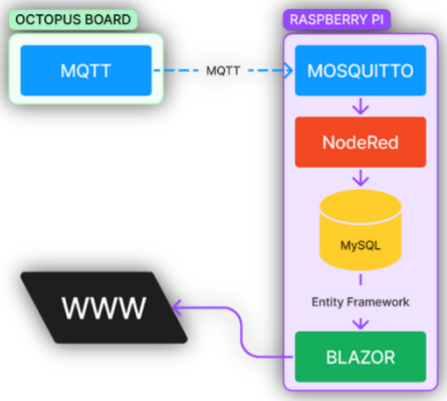
PROJEKTBESCHREIBUNG:

Im vergangenen Schuljahr erhielte die Gruppe des Wahlpflichtfaches im Rahmen des Unterrichts eine Wetterstation zu konzeptionieren und im besten Falle bereits mit der Entwicklung zu beginnen. Zum Ende des Schuljahres stand ein erster Prototyp.

In diesem Jahr sollte dieser Prototyp verbessert und um Funktionen erweitert werden. Das Team bestand zum Anfang des Schuljahres noch aus Peter Rietz und Manuel Yates. Ab Herbst 2022 hat Herr Rietz leider die Schule verlassen, womit Herr Yates das Projekt allein vollzog.

IST-ANALYSE:

Im Prototyp kam folgende Software-Architektur zustande:



Wie leicht zu erkennen ist, gibt es hier 4 Kernprogramme, auf welche gesetzt wird:

- **Mosquitto** als MQTT Broker
- **NodeRed** als Schnittstelle zwischen Broker und Datenbank
- **MySQL** als Datenbank
- Ein **Blazor** Webserver

Durch die Vielzahl an Diensten entstehen einige Probleme:

- Mehr Expertise der Entwickler benötigt:
  Die Entwickler müssen sich mit alles Tools auskennen, um diese vernünftig und sinnvoll weiterzuentwickeln oder zu verwenden.
- Ressourcen-„hungriger“:
  Durch die Installation von NodeRed wird im Hintergrund die Javascript Anwendung Node.js heruntergeladen und muss immer aktiv sein, damit NodeRed funktioniert.
  Um die MySQL-Datenbank besser zu verwalten, wurde die Anwendung phpMyAdmin verwendet, welche im Hintergrund einen Apache-PHP Server verwendet.
  Beide Anwendungen nutzen Webserver, um ihre Anwendung anzubieten, wobei diese keinem anderen Zweck dienen und dementsprechend nur unnötige Auslastung für das Gerät bedeuten.
- Abhängigkeiten:

Durch die Verwendung der Tools, sind die Entwickler abhängig von den Entwicklern und Herausgebern der Tools bzgl. Neuer Updates, Funktionen oder auch Fehlerbehebungen. Alle verwendeten Tools sind zwar OpenSource, jedoch teilweise mit etwas mangelhafter oder überwältigenden Menge an Dokumentationen und Einarbeitungsaufwand.

SOLL-KONZEPT:

Zu Beginn des Schuljahres, besprachen wir mit Herrn John die Projektanforderungen:

- Die Architektur sollte vereinfacht und Resourcenschonender gestaltet werden
- MQTT sollte weiterhin als Protokoll für die Kommunikation mit den Octopus-Boards verwendet werden.
- Eine Kamera-Anbindung mit einem möglichen Livefeed um den schuleigenen Bienenstock über das Internet anzuzeigen.
- Die Oberfläche des Webservers sollte um einige Analyse-Funktionen und Dashboards erweitert werden.

ARCHITEKTUR-VERÄNDERUNGEN

Wir beschlossen drastische Veränderungen an der Architektur vorzunehmen, um die bereits genannten Probleme zu attackieren:

- Es sollte ein eigener MQTT Broker auf Basis des MQTTnet Packets für C# entwickelt werden, welcher intern direkt über eine Datenbankanbindung verfügt.
- Für die Datenbank Anbindung sollte der „Object-Relational-Mapper“ Entity Framework verwendet werden.
- Als Datenbank sollte anstelle einer MySQL eine MSSQL Express genutzt werden.
- Da wir die Architektur von Anfang an möglichst zukunftsorientiert und erweiterbar halten wollten entschlossen wir uns, neben dem vorher bereits verwendeten Repository Pattern, auch das Unit-Of-Work Pattern zu verwenden um eine weitere Abstraktionsebene hinzuzufügen.
- Um einen zentralen Ort für die verwendeten Methoden und Frameworks zu bieten, sollte die eigene Klassenbibliothek aufgefrischt und ebenfalls umstrukturiert werden. Außerdem
- Zusätzlich wurde noch ein Daten-Generator angelegt, welcher uns lediglich mit Daten zum Testen bedienen sollte.

Mit diesem Lösungsansatz konnten wir direkt den Großteil der Probleme lösen:

- Mehr Expertise der Entwickler benötigt:
  Die Entwickler müssen nun lediglich in C# entwickeln und sich nur noch zu MQTTnet sowie dem Entity Framework belesen. Das Entity Framework verfügt über exzellente Dokumentationen von Microsoft und es gibt eine Menge kostenfreier Lernmaterialien. Das MQTTnet – Paket wird vom Autor selbst in GitHub gepflegt und dort sind die wichtigsten Grundfunktionen ausreichend erklärt. Der Einstiegsaufwand ist damit deutlich geringer.
- Ressourcen-„hungriger“:
  Da nun das Projekt aus lediglich zwei aktiven Anwendungen und insgesamt vier Komponenten insgesamt besteht, welche alle mit dem gleichen Framework arbeiten, sind keinerlei unnötige Hintergrunddienste mehr von Nöten.
- Abhängigkeiten:
  Mit den Änderungen sind wir grundlegend, lediglich von Microsoft abhängig, wobei diese für ihre Update-Geschwindigkeit und niedrige Fehlerquote sind. Da Kompatibilität eines der Grundsätze von Microsoft ist, müssen wir in der kommenden Zeit mit keinen Updates rechnen, welche den Betrieb der Anwendung stören sollten.

KAMERA-ANBINDUNG

Für die Kamera beschlossen wir vorerst eine einfache USB-Kamera zu verwenden und diese an einen Raspberry Pi anzuschließen. Um die Kamera softwareseitig einzubinden, wollten wir die OpenCV2 Bibliothek verwenden, da diese einfach und doch umfangreich zu verwenden ist.

Es sollte ein einfaches Python Skript erstellt werden, welches in Regelmäßigen Abständen Fotos macht, diese in ein Byte-Array konvertiert und anschließend mit einem POST an den Api-Controller des Webservers schickt.

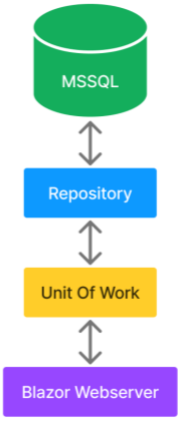
UMSETZUNGSPHASE:

ENTWICKLUNG DES MQTT BROKERS

Bei der Entwicklung des Broker setzten wir, wie vorhin beschrieben auf das Paket MQTTNet. Als Basis verwendeten wir eine Vorlage, die vom Herausgeber des Pakets gegeben wird. Diese passten wir so an, dass die gepublishten Nachrichten automatisch mit den jeweils korrekten Werten über das Entity Framework in die Datenbank geschrieben werden.

UMSTRUKTURIERUNG DER ARCHITEKTUR

Bei der Integration des Unit-Of-Work Patterns gab es einige Dinge zu beachten, damit eine reibungslose und korrekte Verwendung ermöglicht wird. Die Unit-Of-Work Ebene übernimmt die gesamte Datenbeschaffung aus Sicht des Webservers und die Repositorien bieten nur die Methoden für den eigentlichen Zugriff auf die Datenbank. So ist die die Unit-Of-Work auch um viele Schnittstellen, wie zum Beispiel einen Dateizugriff erweiterbar ohne das andere Bestandteile verändert werden müssen. Gleiches gilt auch für die Repositorien. Sollte also demnach die Entscheidung getroffen werden den Datenzugriff künftig anders zu lösen so ist dies kein Problem. Auch ein Wechsel der Datenbank ist durch das Entity Framework als ORM mit nur wenigen Handgriffen und Anpassungen möglich. So ist die Anwendung lose gekoppelt und sehr zukunftsicher.



TICKETS

Wir entschieden uns aus Interessensgründen vorab mit der Ticket-Plattform Jira auseinanderzusetzen, um dort unsere Aufgaben zentral zu verwalten und zu planen.

Jira ist für kleinere Teams und Organisationen vollständig kostenlos und verfügt neben einer Scrum und Kanban Oberfläche zur agilen Arbeit auch über viele hilfreiche Tools und Schnittstellen zu anderen Diensten. Da wir bereits bei den Prototypen auf GitHub als Versionsverwaltung gesetzt hatten, bot sich die Jira Integration perfekt dafür an.

In dieser kann leicht einen Automatismus gesteuert werden, welcher zum Beispiel einen Branch mit dem Ticketnamen erstellt, sobald dieses auf „In Arbeit“ gestellt wird. Anschließend kann man direkt im Jira die Veränderungen am Quellcode sehen und diese auch per Knopfdruck in den Haupt Branch mergen.

Da zu Beginn Herr Rietz noch am Projekt mitarbeitete, teilten wir die Aufgaben wie folgt auf:

- Manuel Yates:
  - Architekturumstellung
  - Entwicklung des MQTT-Brokers
  - Umstrukturierung der Bibliothek
- Peter Rietz:
  - Kamera-Anbindung (bspw. An den Raspberry Pi)
  - Vorbereiten der Schnittstellen / APIs
  - Erstellen grafischer Mockups für die Oberflächen

DEPLOYMENT / PRODUKTIONSUMGEBUNG:

Während der Entwicklungsphase beschäftigten wir uns ebenfalls um den Vorgang des sog. „Deployments“ also die Bereitstellung der Anwendung für die Kunden.

Um Kosten einzusparen beschlossen wir eine einfach Kombination aus einer Fritzbox und dem DynDNS Anbieter zu verwenden.

FUNKTIONSWEISE:

Ein normaler Internetanschluss für den Privatgebraucht besitzt keine statische IP-Adresse. Die IP-Adressen des Routers werden nach einem gewissen Zeitraum aktualisiert.

Damit entsteht das Problem, dass wenn man bspw. Eine Website über einen Heimcomputer hosten möchte, nach dem Abgelaufenen Zeitraum die IP-Adresse in den Websiteeinstellungen für die Weiterleitung händisch aktualisieren müsste.

Mit einem DynDNS Anbieter kann dies automatisch geschehen. Zu Beginn sucht man sich einen Anbieter aus. Einige bieten die Dienste kostenlos an, sind aber meist an Bedingungen geknüpft (z.B.: Verfall nach 30 Tagen ohne Login), andere erheben einen meist sehr niedrigen Preis und bieten dafür alle Freiheiten. Wir beschlossen zum Anbieter zu gehen, da dieser sowohl eine kostenlose als auch bezahlbare Dienstleistung anbietet. Wir beschlossen in unserem Fall die Bezahlversion zu nehmen. Nachdem man eine DynDNS-Domain angelegt und die Zugangsdaten im Portal des Anbieters gefunden hat, kann man diese in den Einstellungen der Fritz Box hinterlegen. Nun wird immer, wenn sich die IP-Adresse des Private Haushaltes verändert eine Mitteilung an den DynDNS geschickt und dieser verändert die Weiterleitung. So können auch privat-gehostete Webseiten permanent verfügbar gemacht werden.

HARDWARE-RESSOURCEN:

Als Hardware kamen zum Einsatz:

- Die schuleigenen OctoBoards als Sensoren
- Einen Intel NUC Computer als Entwicklungsserver
- Einen Heim-PC als Produktionsserver mit Portfreigabe fürs Internet

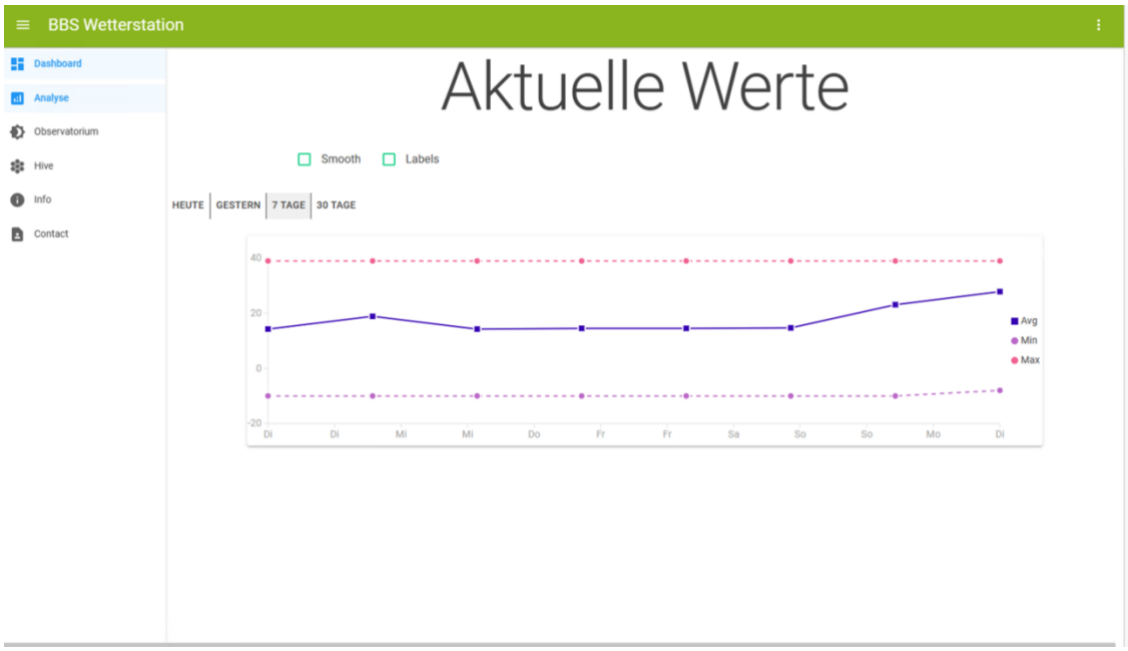
FAZIT:

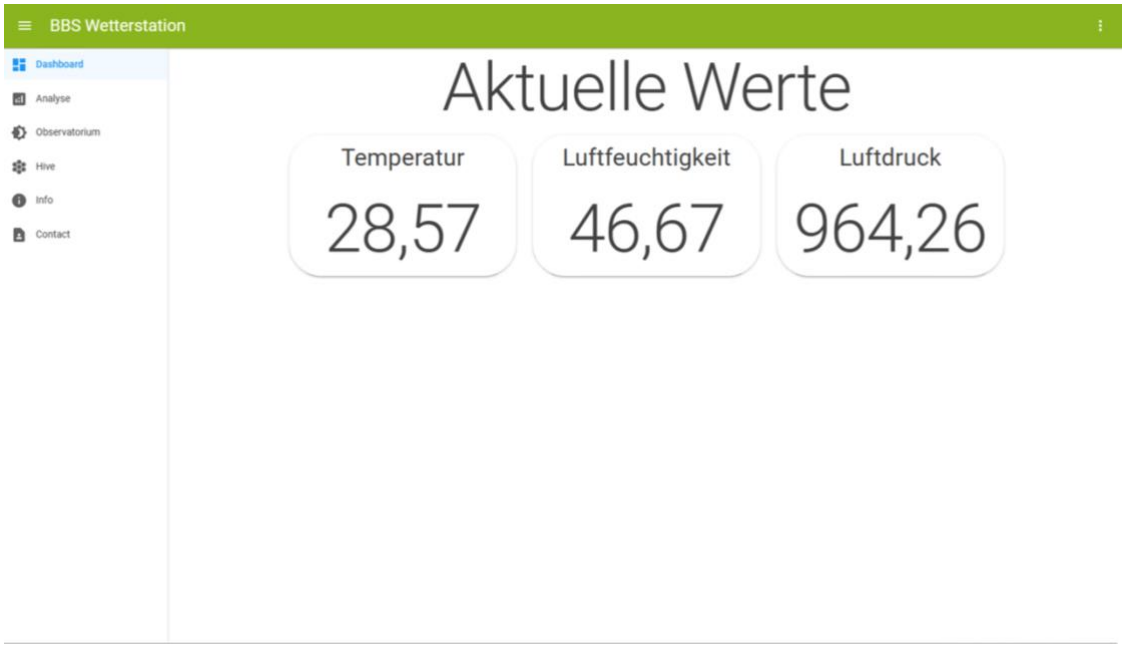
Durch das Wegfallen von Herrn Rietz bei der Entwicklung, blieb mir leider nicht mehr genügend Zeit, um auch die Kamera-Anbindung umzusetzen. Erschwerend kam in diesem Schuljahr hinzu, dass wir einige Stunden für Vorbereitungen für den Teil 1 der Abschlussprüfung im März 2023 verwendeten und allgemein weniger Zeit für das WPF hatten.

Nichtsdestotrotz konnte ich durch das Projekt mein Wissen mit dem MQTTnet Framework erweitern und viele neue Umsetzungsmöglichkeiten ausprobieren und verbessern.

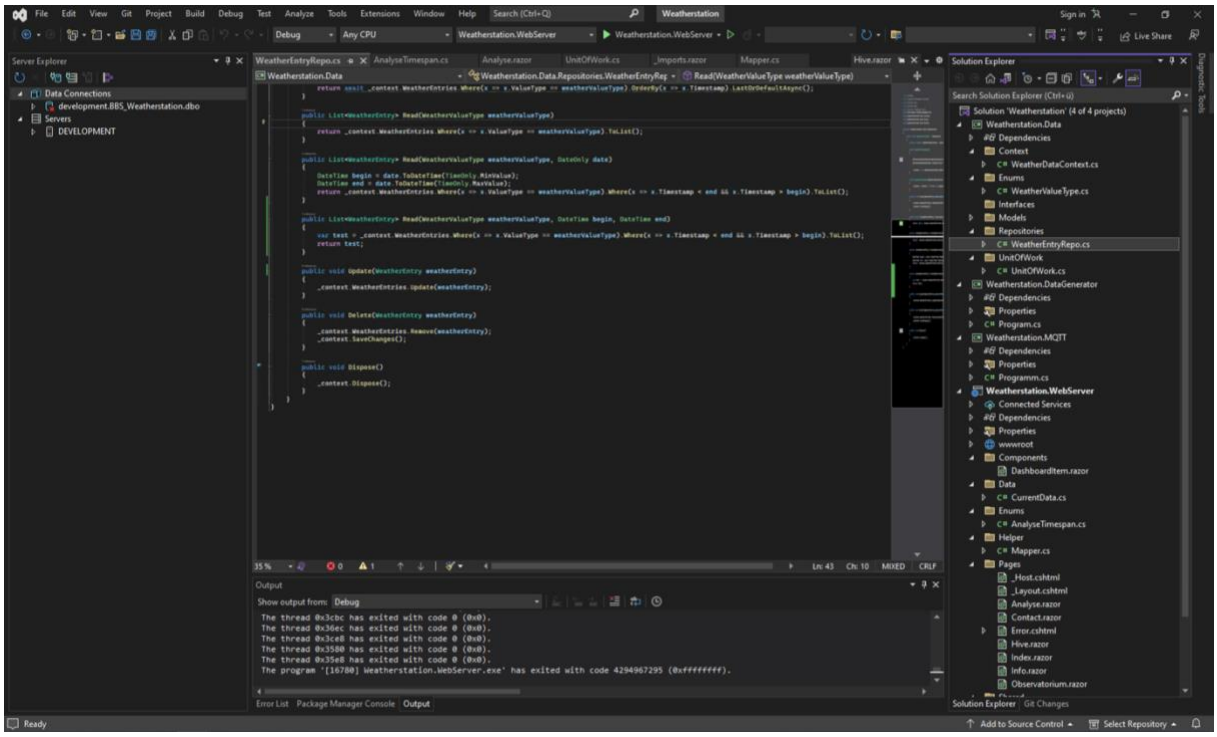
ANHÄNGE

AUFNAHMEN DER OBERFLÄCHEN

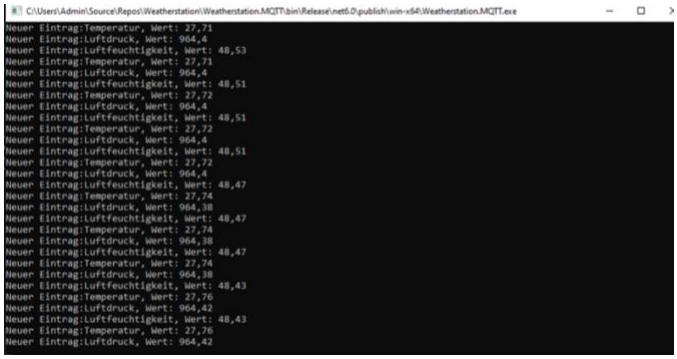




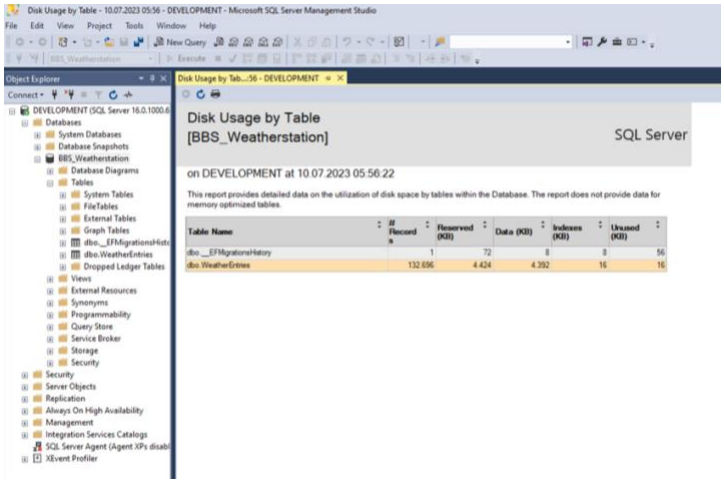
AUFNAHMEN DER ENTWICKLUNGSUMGEBUNG

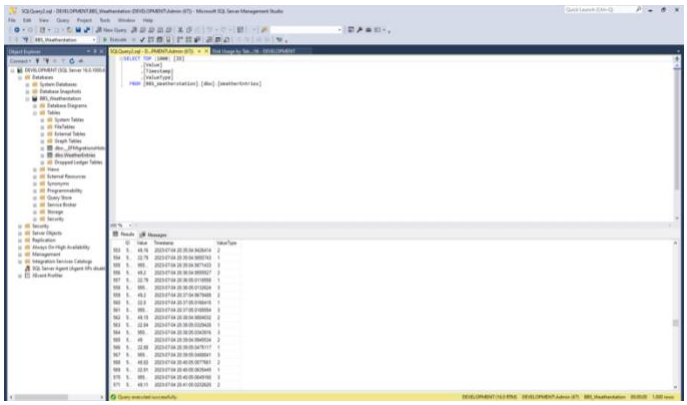


DER MQTT BROKER



DATENBANK





LINKS

Link zum Repository: https://github.com/porschemanu/Weatherstation

Link zu MQTTnet: https://github.com/dotnet/MQTTnet