SAVEUP

Simon Müller

IBZ AARAU 28.02.2023

Versionsverzeichnis

Version	Datum	Änderung
1	01.03.2023	Erstellen des Dokuments
1.1	03.03.2023	Erweiterung Realisierung, Kontrollierung, Auswertung
1.2	04.03.2023	.NET MAUI-Applikation
1.3	21.03.2023	Fertigstellung Dokumentation

1 Versionsverzeichnis

Inhaltsverzeichnis

1 Einleitung

Im Rahmen des Moduls 335, wird das Projekt SaveUp umgesetzt.

Auftraggeber: Lukas Müller Dozent IBZ / IPSO

Das Projekt wird von Simon Müller umgesetzt.

1 Aufgabenstellung

Es handelt sich um eine **.NET MAUI-Applikation**, welche dazu dient, gesparte nicht Käufe einzutragen. In dieser lassen sich Einträge erstellen und abrufen die den Benutzer über den aktuellen Stand seines gesparten Gelds zu Informieren. Diese sollte primär auf dem Smartphone (iOS) funktionieren und sekundär auf einem Windows PC.

Die Aufgabestellung findet sich im Detail unter dem Ordner **Aufgabenstellung**.

In der Aufgabenstellung wird als Rahmenbedingung ausdrücklich eine Xamarin Forms App festgelegt. Dies wurde aber mit Lukas Müller auf Absprache auf eine .NET MAUI-App geändert.

2 Planung

Die Projektplanung und Umsetzung, ist nach IPERKA.

3 Technologien und Tools

Folgende Tools und Versionen werden verwendet:

- Microsoft SQL Server Azure SQL Edge (64-bit): 15.0.2000.1574
- Raspberry Pi 4:
  - OS: Ubuntu 20.04
- Dotnet SDK: 7.0.201
- Dotnet Runtimes:
  - Microsoft.AspNetCore.App: 7.0.3
  - Microsoft.NETCore.App: 7.0.3
- Visual Studio 2022 Professional: 17.5.0
  - .NET 7
- Docker: 23.0.1
- Microsoft Server Management Studio: 15.0.18424.0 (Für Überprüfung)
- Postman: 10.11.1 (Für Überprüfung)
- EF Core Power Tools: 2.5.1206
- Nginx: 1.17.0

Für die Code Struktur Überprüfung wird die databinding.editorconfig Version 2023.02.07.8 verwendet.

Azure DevOps Repository wird für die Source Kontrolle verwendent.

GitHub für das Veröffentlichen des Codes.

2 Informieren

1.1 .NET MAUI

Am erstem Modul 335 Tag, wurde der Auftrag erteilt. Die Informationen über die .NET MAUI-Applikation wurden aus dem erstem Modul Tag entnommen. Darunter auch Microsoft Webseiten.

Die .NET MAUI-Applikation kann nur mit einer Apple Developer Licence auf ein iOS Gerät hochgeladen werden. Diese wird von der databinding GmbH bereitgestellt.

1.2 ASP.NET Core Web API

Informationen über das Veröffentlichen einer ASP.NET Core Web API, wurden aus bestehendem wissen und diversen Internetseiten zusammengesetzt.

Siehe Quellenverzeichnis am Ende dieses Dokuments.

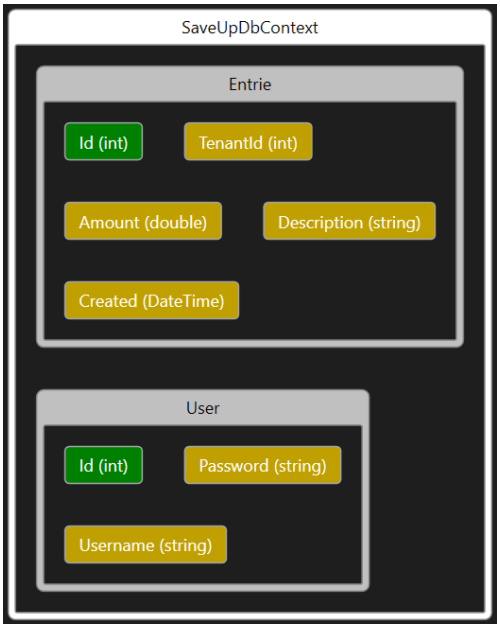
3 Planung

3.1 Zeitplan

Der Zeitplan findet sich unter dem Verzeichnis: **docs/Planung/Gantt-Zeitplan.xlsx**.

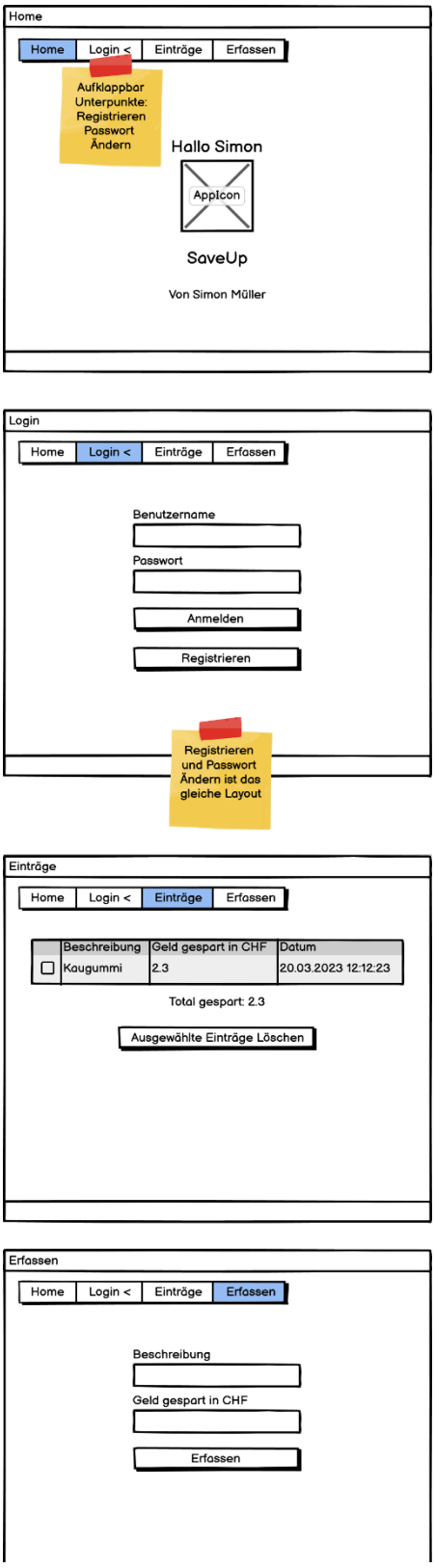
3.2 Datenbank Modell

Das Datenbank Modell wurde mit EF Core erzeugt. Mit EF Power Tool wurde ein Diagramm erstellt:



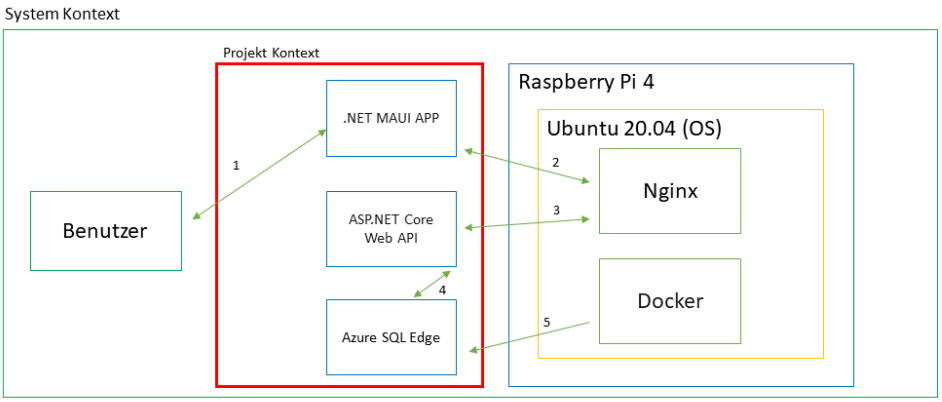
1 Datenbank Diagramm

3.3 Designentwurf



2 Designentwurf

3.4 Applikationsarchitektur



Nummer	Verbindung
1	Manuelle Eingabe auf Smartphone oder PC
2	HTTP Request/Response. Austausch von Daten.
3	Prozessweiterleitung. Leitet den HTTP Port zur API weiter.
4	Database Connection (über Connection String)
5	Isolierung der Datenbank mit Hilfe von Containervirtualisierung

3 Applikationsarchitektur

3.5 App-Icon

Das Icon für die Applikation wird von der Internetseite flaticon ausgesucht.

Dies weil ein Abonnement von databinding GmbH vorhanden ist.

3.6 Mandanten Basierte API

Für die Web API wurde eine Multi-Tenant Applikation geplant.

Dies wird mit Entity Framework Core und JWT Token Claims umgesetzt.

Somit kann die API von beliebig vielen Benutzer (soweit die Leistung des Raspberry Pi mag) nutzen. Dabei kann ein Benutzer nur die von ihm selbst erfassten Einträge Abfragen und nur für sich selbst Einträge erfassen.

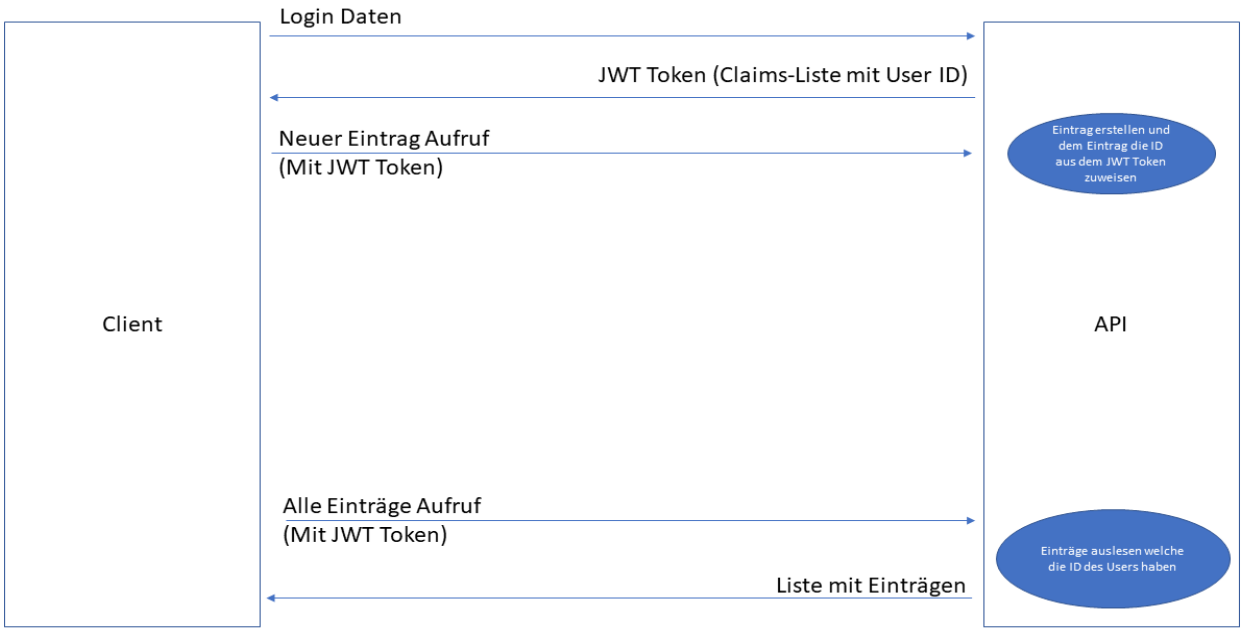
3.6.1 Registrierung

Bei der Registrierung wird eine ID für den Benutzer angelegt.

Diese dient dann als Query Filter für alle Einträge.

3.6.2 Login / Daten Erfassen und Lesen

Folgender Ablauf beschreibt das Login und das Erfassen und Lesen von Daten:



2 Mandantenbasiertes Login und Erstellen

Mit Entity Framework Core lassen sich die Einträge mit einem Globalen Filter Auslesen. Das heisst, es muss nicht mehr bei jeder Abfrage auf die Tenant-ID gesucht werden.

3.7 JWT-Token

Für die Authentifizierung, werden JWT Token gebraucht.

Die JWT-Tokens können sogenannte Claims beinhalten. Was es ermöglicht, die Benutzer ID in dem Token zu verbauen und auszulesen.

4 Entscheidung

4.1 MSSQL / Azure Edge SQL

Microsoft SQL Server 2019 sollte auf dem Ubuntu 20.04 installiert werden. Da es sich aber herausgestellt hat, dass dies nicht von Linux unterstützt wird, wurde sich für Azure SQL Edge entschieden.

Ausfolgenden Gründen:

- Die Datenbank ist optimiert für IoT Geräte wie Raspberry Pi.
- Um auf diese Datenbank zu wechseln, braucht es keine Änderungen am Entity Framework, da es die gleichen Funktionen bietet.
- Die Installation und Aufsetzung sind leicht gemacht mit Docker.

4.2 Optionale Anforderungen

Es wurde sich für Folgende Optionale Anforderungen entschieden:

- Speichern der Einträge in Backend Datenbank REST
  - Weil das Fachwissen bereits vorhanden ist, und es die App Client Unabhängig macht.
- Löschen der erfassten Einträge
  - Dabei können Einträge einzeln oder mehrere gelöscht werden.
- Datum/Uhrzeit als zusätzliches Attribut, wann der Kaufverzicht erfolgte.

4.3 Login Sperre

Damit das Login eines Benutzers nicht unendlich ausprobiert werden kann, wurde sich dafür entschieden, das Login nach drei Falschen Versuchen zu sperren.

In dem Fall kann das Login nur von einen System Admin wieder freigegeben werden.

5 Realisierung

5.1 ASP.NET Core Web API

Die API konnte schnell umgesetzt werden. Da die Fach- und Methodenkompetenzen bereits mehrheitlich vorhanden waren.

5.2 Host Umgebung

Damit ist das Raspberry Pi Einrichten, Aufsetzen, Docker und Nginx Installieren gemeint.

Diese war nicht Teil des Projekts und wird hier nicht weiter beschrieben.

Trotzdem wurden die Quellen der Informationen im Quellenverzeichnis zur Referenz aufgelistet.

5.3 Azure SQL Edge

Das Datenmodel wurde mit Entity Framework Core erstellt. Dies kann anhand C# Klassen ein Datenbank Modell ableiten.

Die daraus entstehenden Migrationen werden dann auf die Datenbank angewandt. Vorteil davon ist, dass die Datenbank einfach ausgetauscht werden kann. Zum Beispiel durch eine MariaDB.

Für das Datenbank Update wurde ein Projekt erstellt, welches die Datenbank mit einem Datenbank Kontext und einem Connection String Aktualisieren kann. Das Projekt kann Lokal gestartet werden. In dem Fall wird die Lokale Datenbank aktualisiert. Das Projekt wird auch in der Azure Pipeline beim Release gestartet. Dabei wird der Connection String (in Azure gespeichert) als Parameter beim Start mitgegeben, damit keine Sensiblen Daten im Git Repository sichtbar sind. Dabei wird die Datenbank auf dem Raspberry Pi aktualisiert.

5.4 HttpClient

Der HttpClient wird mit einem Interceptor versehen. Dieser fängt die nicht erfolgreichen http Request ab. Bei einem 401 Nicht authentifiziert, wird die Login Seite aufgerufen. Bei anderen Fehlern kommt eine Toast Meldung.

5.5 .NET MAUI-Applikation

Bei der Realisierung der .NET MAUI-Applikation gab es viele Fehler. Dabei sind einige davon:

- Das Debuggen auf Windows, Mac, iOS und Android Einrichten.
- Die Listdarstellung.
- Das Layout Responsive machen.

6 Kontrollieren

6.1 NUnit Tests

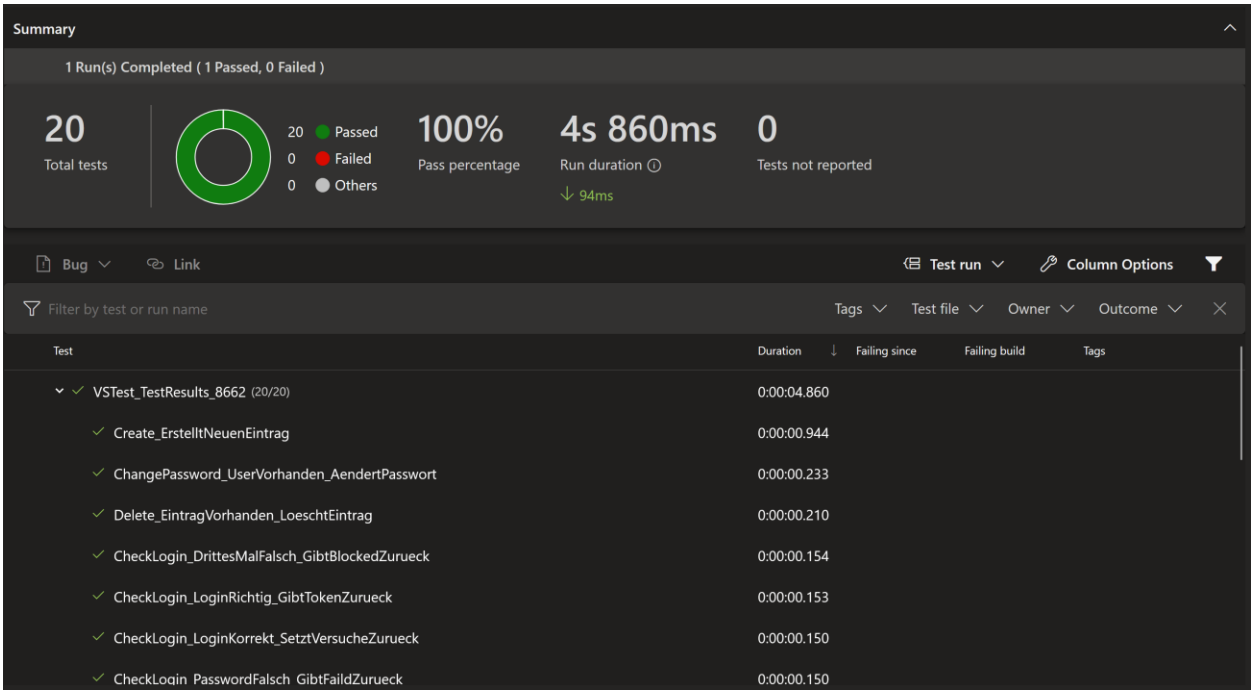
Die Logik der Web API findet in Services statt. Diese werden über NUnit Tests auf ihre Funktionalität getestet.

Somit wird sichergestellt, dass die Services bei Änderungen im Code nicht kaputt gehen.

Die Tests werden in der Azure Pipeline bei jedem neues Release ausgeführt. Falls auch nur ein Test fehlschlägt, wird der Release abgebrochen.

Ein Testprotokoll wird automatisch von Azure DevOps erstellt. Dies kann vom Entwicklerteam jederzeit eingesehen werden.

Hier ist das Testprotokoll vom 20.03.2023:



4 Testprotokoll

6.2 Swagger

Während dem Entwickeln der Web API, wurde lokal Swagger UI verwendet, um die Endpunkte zu testen.

Dabei können die API-Endpunkte mit OpenAPI Visuell dargestellt werden.

6.3 Postman

Um die Endpunkte zu testen, nachdem diese auf dem Ubuntu Server sind, wurde Postman verwendet.

Mit diesem Programm können HTTP (Internetanfragen) gespeichert, gruppiert und getestet werden.

Die Postman Datei findet sich im Ordner **/docs/Postman/SaveUp_API_Collection.json**

7 Auswerten

7.1 Soll / Ist Vergleich

Siehe Gantt-Zeitplan.

7.2 Fazit

Ich konnte viel lernen. Sowohl über .NET MAUI als auch über Hosting.

Ich hatte die Möglichkeit eine eigene API zu programmieren und diese auf meinem Raspberry lauf zu lassen. Zudem wurde die mit Azure DevOps Pipelines automatisiert.

Ich habe Docker und Nginx auf dem Ubuntu Installieren können und dabei einen SQL-Server ansprechen können.

Die API und Datenbank sind auf jeden Fall gelungen. Mandantenbassiert und selbst gehostet.

Bestehendes Wissen gut angewandt

Ich habe die Zeit überzogen, aber es lohnte sich, um sich damit besser beschäftigen zu können.

Ich habe .NET MAUI-Erfragungen gesammelt. Dabei ist mir Aufgefallen, dass .NET MAUI noch nicht ausgereift ist. Es gibt viele Fehler und Unklarheiten. Auch im Internet findet sich nicht immer eine gute Antwort da es auch Neu ist.

8 Abbildungsverzeichnis

9 Tabellenverzeichnis

10 Quellenverzeichnis

Kontext	Link
dotnet runtime installieren auf Ubuntu	linux - Cannot install .NET on Ubuntu 18.04 - Stack Overflow
dotnet Versionen	Download ASP.NET Core 7.0 Runtime (v7.0.3) - Linux Arm64 Binaries (microsoft.com)
Autostart von Ubuntu Services	Howto disable or enable apache2 autostart on Ubuntu - ubuntu|dog (ubuntudog.com)
Azure SQL Edge mit Docker auf Raspberry Pi	https://hevodata.com/learn/sql-server-on-raspberry-pi/
Dotnet SDK, Runtime bei reboot von Raspberry Pi verfügbar machen	.net - Dotnet command not recognized after reboot Raspbian - Stack Overflow
Hosten einer ASP.NET Core App in Linux	Hosten von ASP.NET Core unter Linux mit Nginx | Microsoft Learn
Installation dotnet ef tools unter Ubuntu	.net - How to properly install dotnet ef on Ubuntu? - Stack Overflow
Entfernung von Docker Container	IBM Documentation
ASP.NET Core bei start von Raspberry Pi ausführen	Raspberry Pi: Run ASP.NET Core on Startup - Code it Yourself... (mendible.com)
Azure DevOps Pipeline mit .NET MAUI App	https://www.andreasnesheim.no/setting-up-ci-for-your-net-maui-android-app-in-azure-devops/
Tabs in .NET MAUI	.NET MAUI Shell tabs - .NET MAUI | Microsoft Learn
.NET MAUI DI	Dependency Injection | Microsoft Learn
.NET MAUI DataGrid	themes/material/components/DataGrid | EnisnProjects (enisn-projects.io)
.NET MAUI http Interceptor	HttpInterceptor implementation in C# and .NET - Stack Overflow

3 Quellenverzeichnis