26.3.2023

Simon Stauffer

Irasil & Co.

Dokumentation: SaveUp

Modul 335 – Mobile Applikation realisieren

Inhaltsverzeichnis

[2 Tabellenverzeichnis 2](#_Toc130658504)

[3 Abkürzungsverzeichnis (Chronologisch) 2](#_Toc130658505)

[4 Einleitung 3](#_Toc130658506)

[4.1 Verwendet Tools: 3](#_Toc130658507)

[4.2 Informieren 4](#_Toc130658508)

[4.3 Projektumfeld 4](#_Toc130658509)

[4.3.1 Projektorganisation 4](#_Toc130658510)

[4.3.2 Projektziele 4](#_Toc130658511)

[5 Planen 5](#_Toc130658512)

[5.1 Gantt 6](#_Toc130658513)

[5.1.1 Möglichkeiten der Umsetzung 7](#_Toc130658514)

[5.2 Use Cases 7](#_Toc130658515)

[5.2.1 Testprotokoll 7](#_Toc130658516)

[5.3 Wireframe erstellen 8](#_Toc130658517)

[5.3.1 Variante A 8](#_Toc130658518)

[5.3.2 Variante B 8](#_Toc130658519)

[6 Entscheiden 9](#_Toc130658520)

[6.1 Architektur der Applikation 9](#_Toc130658521)

[7 Realisieren 10](#_Toc130658522)

[7.1 API 10](#_Toc130658523)

[7.2 Datenbank 10](#_Toc130658524)

[7.2.1 Testdaten 10](#_Toc130658525)

[7.3 Maui App 11](#_Toc130658526)

[7.3.1 Modell & Views 11](#_Toc130658527)

[7.3.2 ViewModel 11](#_Toc130658528)

[7.4 Sequenzdiagramme 12](#_Toc130658529)

[7.4.1 Speichern eines Eintrages 12](#_Toc130658530)

[7.4.2 Abrufen der gespeicherten Einträge 13](#_Toc130658531)

[7.4.3 Eintrag löschen 14](#_Toc130658532)

[7.5 Designe & Icon 14](#_Toc130658533)

[8 Kontrollieren 15](#_Toc130658534)

[8.1 Manuelles Testen 15](#_Toc130658535)

[8.1.1 API 15](#_Toc130658536)

[8.1.2 Maui APP 15](#_Toc130658537)

[8.2 Anforderungen erfüllt 15](#_Toc130658538)

[9 Auswerten 15](#_Toc130658539)

[9.1 Zukunft der App 16](#_Toc130658540)

[10 Fazit 16](#_Toc130658541)

[11 Testprotokoll 17](#_Toc130658542)

[12 Abbildungsverzeichnis 18](#_Toc130658543)

[13 Quellen 18](#_Toc130658544)

# Tabellenverzeichnis

[Tabelle 1: Entscheid Maui vs. Xamarin 9](#_Toc130657439)

[Tabelle 2: Entscheid Speicherung der Daten 9](#_Toc130657440)

[Tabelle 3: Entscheid Designe 9](#_Toc130657441)

[Tabelle 4: Testdaten 10](#_Toc130657442)

[Tabelle 5: Testprotokoll 17](#_Toc130657443)

# Abkürzungsverzeichnis (Chronologisch)

GUI Graphical User Interface

MVVM Model View ViewModel

XAML Extensible Application Markup Language

MS SQL Microsoft SQL Server

API Application Programming Interface

# Einleitung

Das Projekt «SaveUp-App» zielte darauf ab, ein Mobile Applikation mit Maui zu erstellen. Dieses Projekt umfasst folgenden Punkte:

* App besteht min. aus 2 Content Pages
* GUI-Design
* Produkterfassung besteht aus min. 2 Eingaben (Kurzbeschreibung u. Preis)
* Bietet zwei Menüfunktionen (Action) zur Speicherung und Aufruf der Listendarstellung an.
* Einfache bzw. intuitive Bedienung, geeignetes Layout (Styles)
* Codestrukturierung nach Model View ViewModel (MVVM) Entwurfsmuster
* Verwendung von XAML-Styles der Steuerelemente
* Eigenes App-Icon
* Dokumentation u. Testing

Das Projekt wurde von Simon Stauffer realisiert und dokumentiert. Die Dokumentation, sowie die Umsetzung der SaveUp-App basieren auf dem IPERKA-Modell. Das Vorgehen wurde phasenweise dokumentiert.

## Verwendet Tools:

Als IDE wurde Visual Studio 2022 (VS) 17.5.2 genutzt. Zusätzlich wurden mehrere NuGets von VS verwendet:

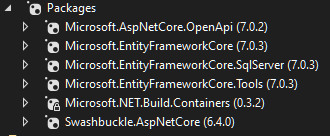
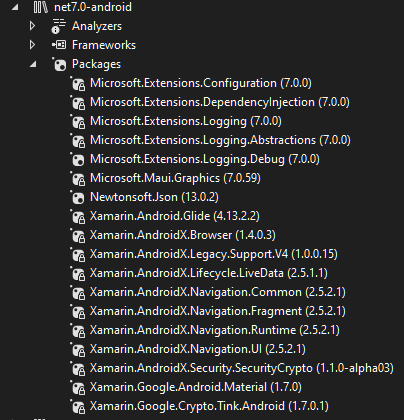


Abbildung : Nuget Liste API

Abbildung : Nuget Liste Maui App



Um die Daten speichern zu können habe ich MS SQL Version MS SQL 2022 Express verwendet und diese auf Somee.com gehostet. Für die API habe ich .net7.0 verwendet und diese auf Azure Free Plan gehostet. Für die Maui App habe ich auch .net7.0 verwendet und für die Verbindung zu der Datenbank habe ich SQL Server Management Studio (SSMS) Version 15.0.18424.0 benutzt.

## Informieren

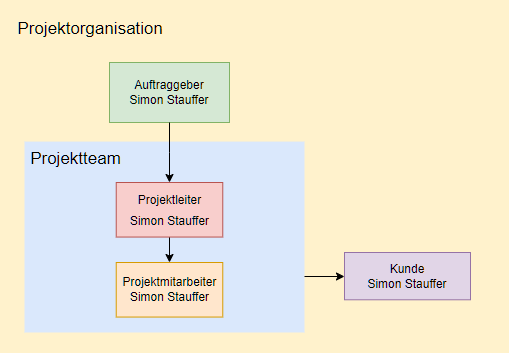
Als Erstes las ich den Arbeitsauftrag durch und besprachen bestehende Unklarheiten mit Herrn Müller. Nachdem die Unklarheiten geklärt waren, inspizierte ich die im Arbeitsauftrag beschriebenen Schritte zur Umsetzung des Projektes. Danach überlegte ich mir Möglichkeiten, diese umzusetzen. Weiter haben wir uns über verschiedene Tools informiert, die wir für die Umsetzung des Projekts benutzen werden. Zusätzlich setzten wir uns mit den Bewertungskriterien auseinander. Anschliessend habe ich im Internet nach Möglichkeiten gesucht, wie ich dieses Projekt umsetzen könnte.

## Projektumfeld

Ich möchte für eine grössere private Investition (z.B. Ferien etc.) Geld sparen und habe ich aus diesem Grund für den Verzicht von üblichen kleinen Ausgaben wie z.B. Kaffee, Süssigkeiten etc. entschieden. Alle gesparten Kleineinkäufe möchte ich in einer App festhalten, sodass ich mich laufend über den angesparten Geldbetrag informieren kann.

### Projektorganisation

Abbildung : Projektorganisation



### Projektziele

Das Projekt wird mit dem Ziel durchgeführt, dass ich meine Verzichte eintragen und speichern kann, damit ich eine Übersicht all meiner Verzichte und den dadurch gewonnenen Betrag bekomme. Ausserdem möchte ich mich mit dem Hosten von einer SQL-Datenbank und der API auseinandersetzen, um dies in Zukunft ohne Probleme machen zu können.

# Planen

Die Projektplanung werde ich nach dem IPERKA Modell machen. Der Grund dafür ist, dass ich schon viel mit diesem Modell gearbeitet habe und weil es für mich die beste Methode ist, um ein Projekt zu planen und dies zugleich eine Vorgabe ist.

**Ablauf- und Terminplanung**

1. Arbeitspakete erstellen

2. Reihenfolge der Arbeitspakete erstellen

3. Vorgangsdauer und Aufwand pro Arbeitspaket definieren.

4. Gesamtlänge und Gesamtaufwand des ganzen Projektes schätzen.

Nachdem ich diese Arbeitstakte erstellt, deren Reihenfolge und Aufwand geschätzt habe, entsteht folgende Planung:

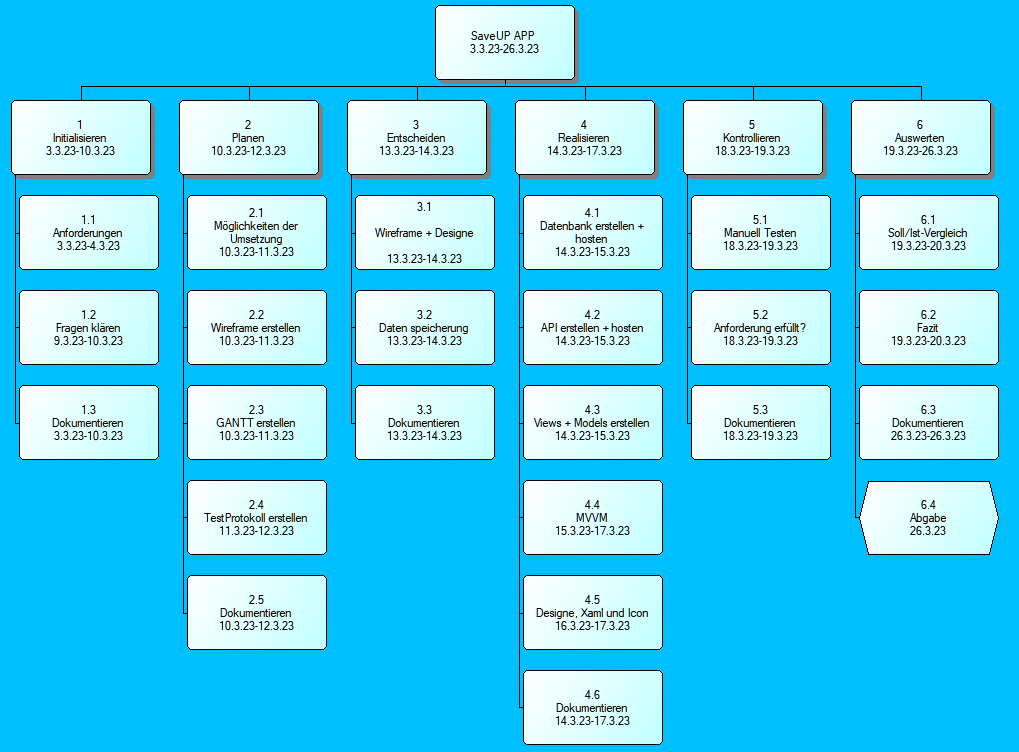


Abbildung : PSP SaveUp App

## Gantt

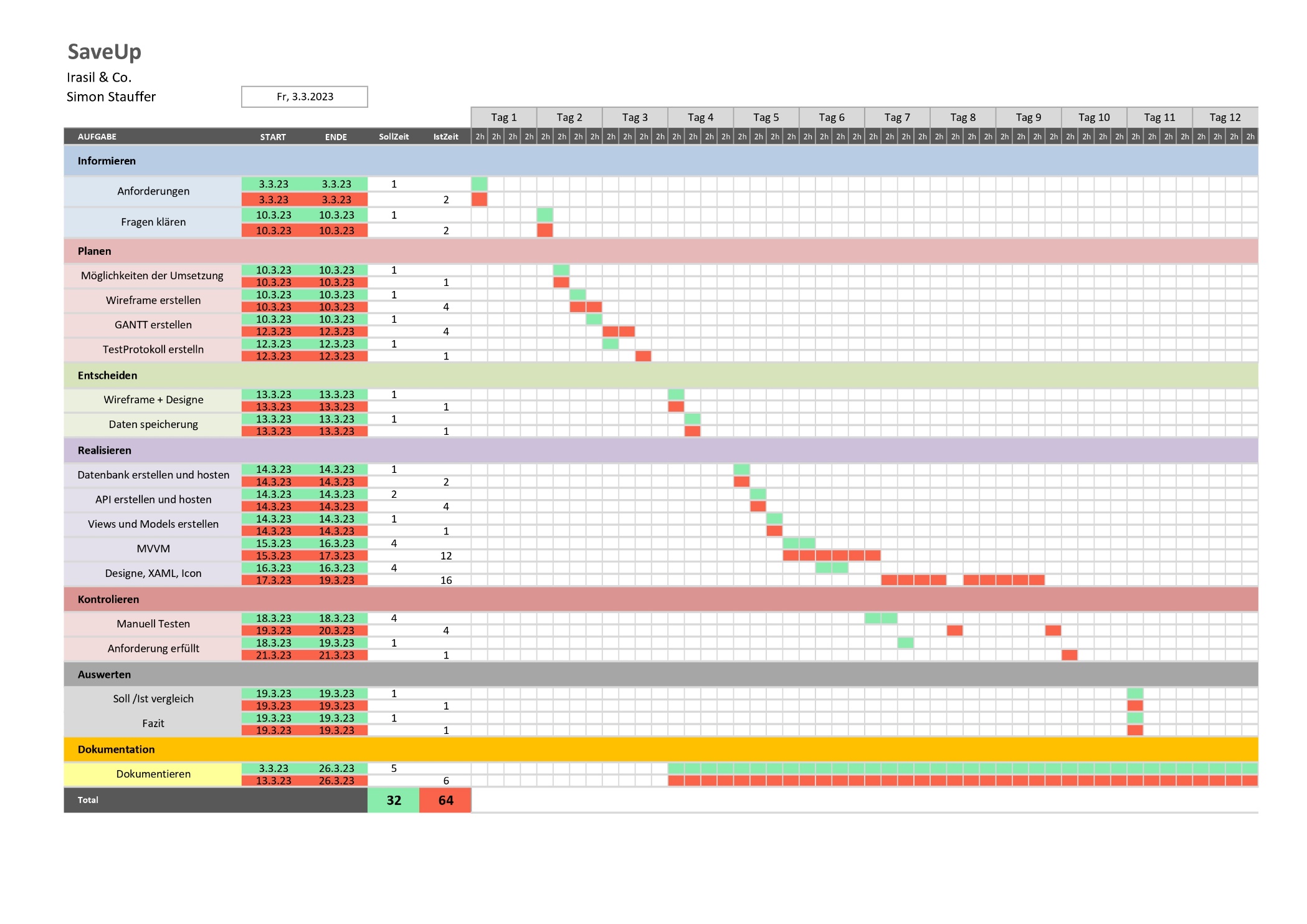


Abbildung : Gantt mit Soll und Ist Zeiten

### Möglichkeiten der Umsetzung

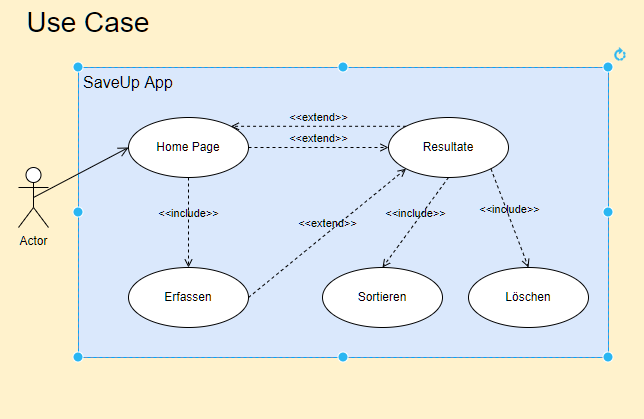
Um das Projekt umsetzen zu können, gibt es mehrere Möglichkeiten. Die Aufgabe liess offen, welches Framework ich verwende. Ich konnte es entweder mit Maui oder Xamarin machen.

Ausserdem war auch die Speicherung der Einträge auf zwei verschiedene Arten möglich. Die lokale Speicherung, also auf dem Endgerät selbst oder über eine API und einer Datenbank dahinter.

Auch das Design der App hatte keine Vorgaben, ausser dass es Benutzerfreundlich sein soll und es aus zwei Content-Pages bestehen muss. Dabei habe ich mir Gedanken über die Use Cases gemacht.

## Use Cases

Abbildung : Use Cases SaveUp



Bei den Use Cases bin ich zu folgendem Schluss gekommen. Als erstes soll der User auf die Hauptseite kommen, hier kann er einen neuen Eintrag erstellen oder über einen Button zu der Resultate-Seite gelangen.

Auf der Resultate-Seite kann der User die erfassten Einträge sortieren oder diese löschen. Ausserdem kann er wieder zurück auf die Homepage gehen, um einen neuen Eintrag zu erfassen.

### Testprotokoll

Auch das Testprotokoll habe ich in der Planungsphase erstellt, da mir die Use Cases bereits bekannt waren, konnte ich die Tests dementsprechend definieren. Dabei habe ich mich auf die Anforderungen an die Applikation konzentriert und deren möglichen Fehler bei der Eingabe oder Ausgabe der Einträge.

## Wireframe erstellen

Um ein Gespür für das UI zu bekommen, habe ich bereits in der Planungsphase mehrere Wireframes der Content-Pages erstellt. Diese konnte ich dann mit Benutzern teilen und so Feedback erhalten, welches mir die spätere Entscheidung erleichtern wird.

### Variante A



Abbildung : Resultate

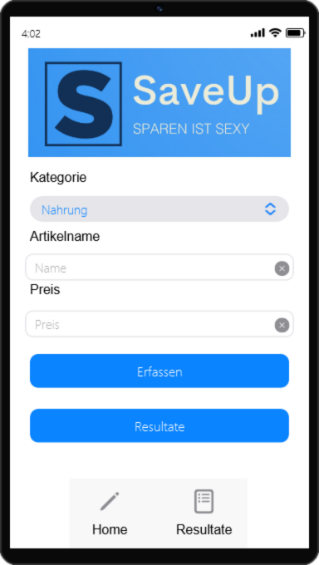


Abbildung : Homepage

Diese Variante besteht aus zwei Seiten, der Homepage, um Einträge zu erfassen und der Resultate-Seite, um die Liste aller Verzichte anzeigen zu lassen. Die Navigation erfolgt über ein Tabbarmenü.

### Variante B

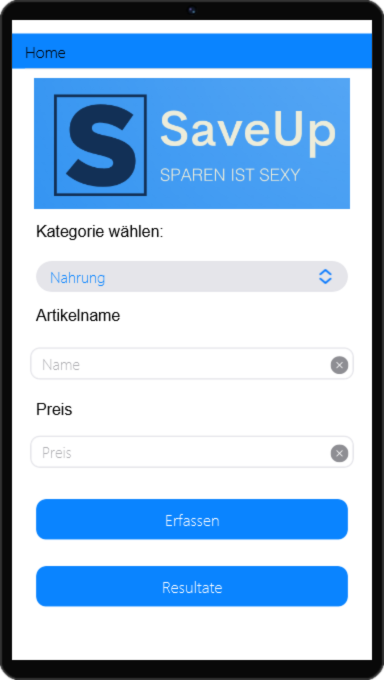


Abbildung : Homepage Version B

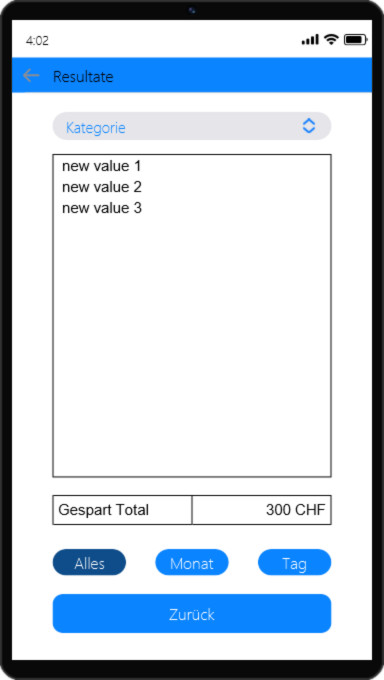


Abbildung :Resultate Version B

Die zweite Variante unterscheidet sich nur leicht von der ersten. Ich habe das Tabbarmenü für eine sauberere Darstellung entfernt und den aktiven Button dunkler eingefärbt als die anderen Buttons. Zusätzlich habe ich einen Seitentitel hinzugefügt, damit man weiss, wo man sich derzeit befindet.

# Entscheiden

Da ich mehrere Möglichkeiten hatte, um das Projekt umsetzen zu können, habe ich für jede Entscheidung eine Matrix mit gewissen Kriterien, die ich mit Punkten bewertet habe, erstellt.

|  |  |  |  |
| --- | --- | --- | --- |
| Entscheidung | .Net Maui | .Net Xamarin | Begründung |
| Kriterien / Gewichtung | Benutzerfreundlichkeit (8) Entwicklungszeit (9)  Verfügbares Wissen (9) | Benutzerfreundlichkeit (8)  Entwicklungszeit (5)  Verfügbares Wissen (2) | Maui kenne ich bereits und dadurch resultiert eine geringere Entwicklungszeit |
| Gesamtgewichtung | 27 | 15 |  |

Tabelle : Entscheid Maui vs. Xamarin

|  |  |  |  |
| --- | --- | --- | --- |
| Entscheidung | SQL-Datenbank /API | Lokale Speicherung | Begründung |
| Kriterien / Gewichtung | Skalierbarkeit (9)  Sicherheit (5)  Hosting (8) | Skalierbarkeit (5)  Sicherheit (6)  Hosting (0) | Eine Datenbank bietet bessere Skalierbarkeit. Ausserdem wollte ich das externe Hosten von SQL und API lernen |
| Gesamtgewichtung | 22 | 11 |  |

Tabelle : Entscheid Speicherung der Daten

|  |  |  |  |
| --- | --- | --- | --- |
| Entscheidung | Designe A | Designe B | Begründung |
| Kriterien / Gewichtung | Benutzerwertung (6) | Benutzerwertung (9) | Design B hat eine höhere Benutzerwertung als Design A |
| Gesamtgewichtung | 6 | 9 |  |

Tabelle : Entscheid Designe

Wie aus den Tabellen hervorgeht, haben Maui und die Datenbank die höhere Gesamtpunktzahl und sind daher die besten Optionen. Design B hat bei den befragten Benutzern besser abgeschnitten. Aufgrund dieser Matrix habe ich mich für Maui mit einer Datenbankanbindung über eine API mit dem Design B entschieden.

Bei dem Icon für die App habe ich mich für ein schlichtes, blaugehaltenes, grosses "S" entschieden. Bei den farblichen Hervorhebungen der App habe ich mich für ein helles Blau entschieden, wie ich es bereits in den Wireframes dargestellt hatte.

## Architektur der Applikation

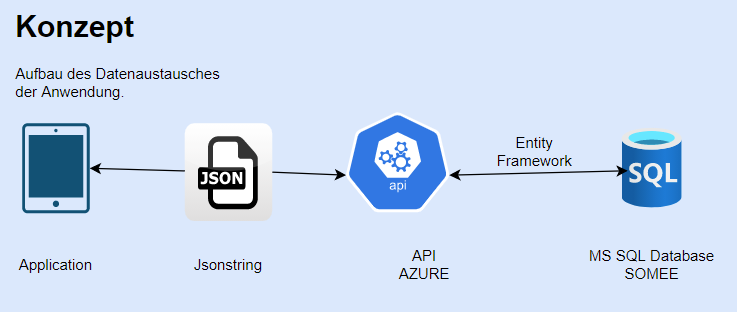


Abbildung : Konzept der Applikation

# Realisieren

## API

In dieser Phase begann ich mit dem eigentlichen Programmieren. Ich habe ein neues Web API Projekt (SaveUpAPI) erstellt und dieses sogleich mit GitHub synchronisiert. Dabei habe ich mich des Entity Frameworks bedient, um sogleich die Tabellen aus dem Modell erstellen zu können, sobald ich eine Datenbank habe. Die API habe ich folgendermassen aufgebaut:

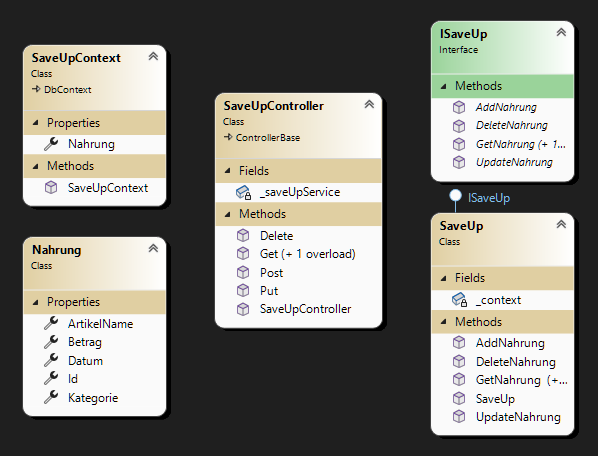


Abbildung : Klassen Diagramm für die API

## Datenbank

Für das Hosten der Datenbank habe ich einen Anbieter gesucht, der einen MS SQL Server gratis zur Verfügung stellt. Dabei bin ich auf somee.com gestossen. Für das Erstellen der Datenbank habe ich mich an die Anleitung von somee.com gehalten. Nach der Erstellung erhielt ich den Login Namen, das Passwort und den Connection-String. Diese konnte ich in den Settings der API eintragen und die Tabelle erstellen. Anschliessend habe ich folgende Einträge als Testdaten in der Datenbank gespeichert:

### Testdaten

|  |  |  |  |
| --- | --- | --- | --- |
| ArtikelName | Betrag | Kategorie | Datum |
| Milch | 10.50 | Nahrung | 2023-03-18 |
| Brot | 20.00 | Nahrung | 2023-03-19 |
| Eier | 15.75 | Nahrung | 2023-02-20 |
| Käse | 5.95 | Nahrung | 2023-02-21 |
| Kino | 50.00 | Ausgang | 2023-03-18 |
| Restaurant | 30.00 | Ausgang | 2023-03-19 |
| Bar | 25.00 | Ausgang | 2023-02-20 |
| Konzert | 15.00 | Ausgang | 2023-02-21 |
| Smartphone | 100.00 | Elektronik | 2023-03-18 |
| Laptop | 200.00 | Elektronik | 2023-03-19 |
| Fernseher | 150.00 | Elektronik | 2023-02-20 |
| Kopfhörer | 75.00 | Elektronik | 2023-02-21 |

Tabelle : Testdaten

Sobald ich die Daten gespeichert hatte, habe ich die API lokal und mit Swagger getestet. Als dies funktionierte, habe ich die API auf Azure gehostet. Dies war sehr einfach über Visual Studio möglich. Deshalb und weil es auch kein Teil der Aufgabe war, werde ich ihr nicht weiter darauf eingehen.

## Maui App

Für die App habe ich mir Gedanken über den Aufbau der Klassen gemacht und bin zu dem folgenden Schluss gekommen:

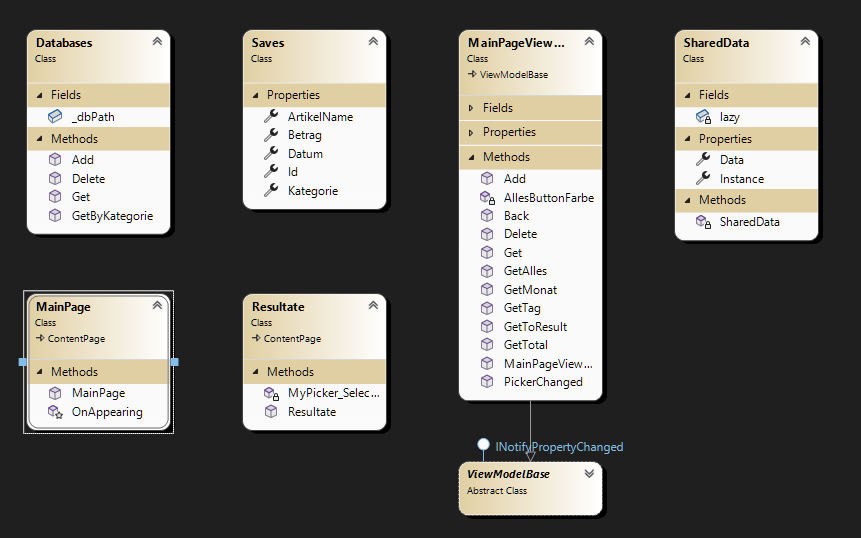


Abbildung : Klassen Diagramm SaveUp

### Modell & Views

Als Erstes habe ich eine Model-Klasse namens Saves erstellt, diese beinhaltet die Properties der zu speichernden Einträge.

Um die Einträge speichern und anzeigen zu können, habe ich eine Hauptseite und eine Resultate-Seite anhand des Wireframes erstellt. Ausserdem habe ich für die Verbindung zu der API eine Klasse Databases erstellt.

### ViewModel

Damit ich Code Behind bestmöglich vermeiden konnte, habe ich eine ViewModel Klasse erstellt. Um das ViewModel an beide UI-Seiten binden zu können, habe ich eine Klasse SharedData erstellt, da der Konstruktor dieser Klasse privat ist, wird/kann dieser nur einmal aufgerufen und gleichzeitig ein neues Objekt von ViewModel erstellt. Anhand dieses einmalig erstellten Objektes konnte ich das Databinding beider UI-Seiten an dasselbe Binden. Ansonsten wäre jeweils ein neues Objekt von ViewModel erstellt worden und ich hätte keinen Zugriff auf die Daten von der jeweils anderen UI-Seite erstellen Daten. Um Änderungen an Properties festzustellen, habe ich das Interface ViewModelBase benutzt.

Ich werde hier nicht weiter auf die Programmierung eingehen, da diese aus den folgenden Sequenzdiagrammen ersichtlich wird.

## Sequenzdiagramme

Ich werde hier nicht alle Sequenzdiagramme einfügen. Ich habe mich für die wichtigsten und auch kürzesten entschieden. Alle anderen finden Sie unter Bilder/Sequenzdiagramme.

### Speichern eines Eintrages

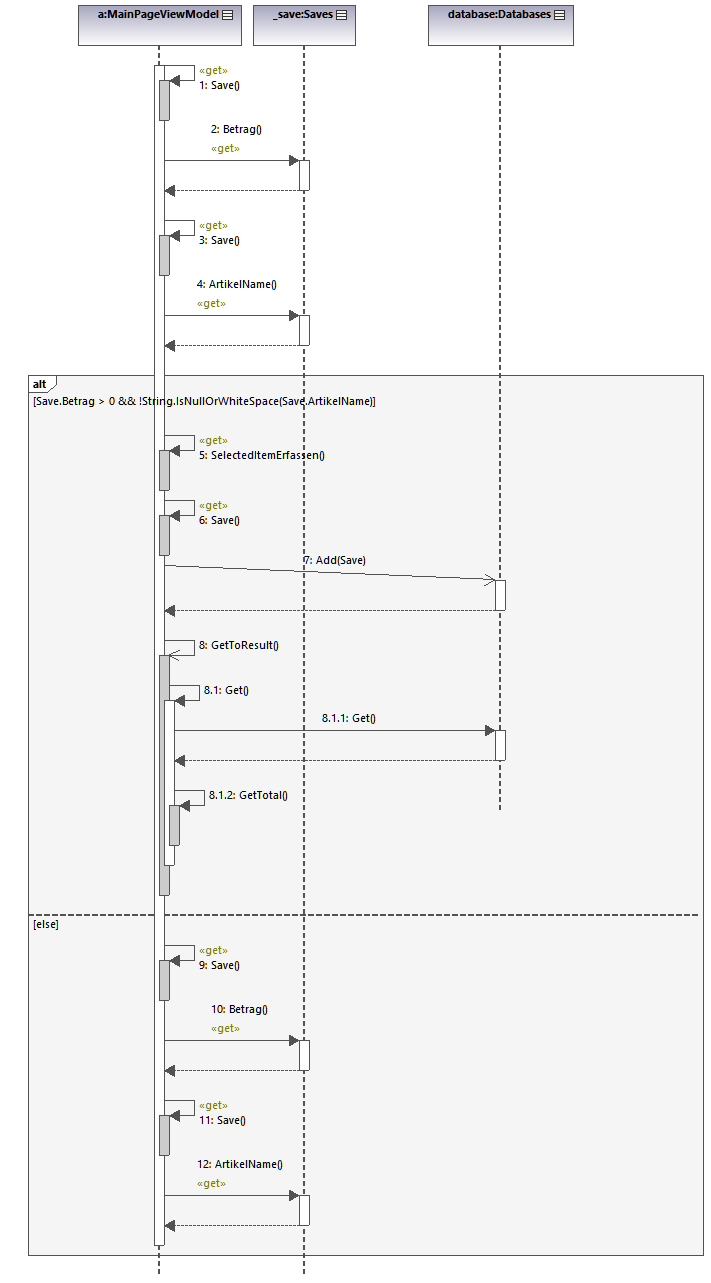


Abbildung : Sequenzdiagramm hinzufügen

### Abrufen der gespeicherten Einträge

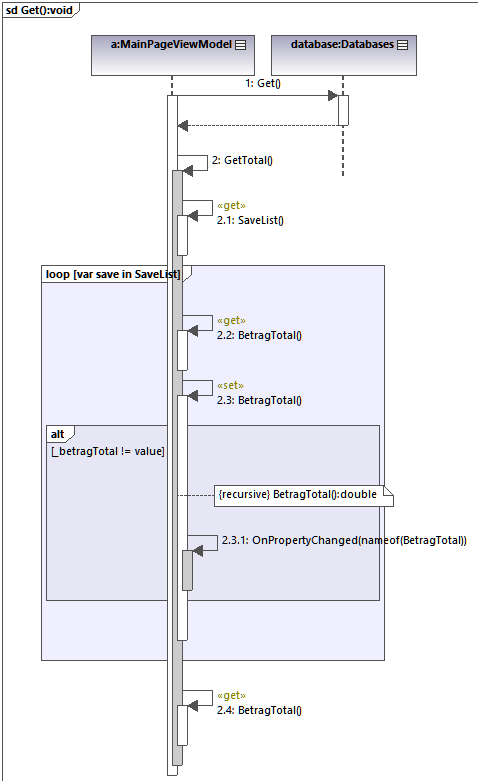


Abbildung : Sequenzdiagramm Abrufen

### Eintrag löschen

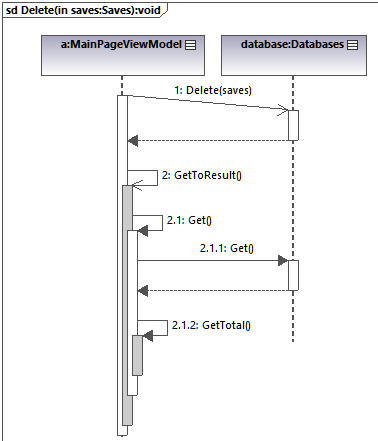


Abbildung : Sequenzdiagramm löschen

## Designe & Icon

Bei dem Design habe ich mich auch vollständig an die Vorlage des Wireframes gehalten. Ein helles Blau und ein dunkleres bei dem aktiven Button. Dies konnte ich in den Syles definieren. Das Finale Logo und Icon:



Abbildung : Logo



Abbildung : Icon

# Kontrollieren

## Manuelles Testen

### API

Nachdem ich eine neue Methode implementiert habe, habe ich diese sogleich mit Swagger getestet. Als alle CRUD-Methoden liefen, habe ich wie beschrieben die API auf Azure veröffentlicht und sogleich über Postman getestet. Da ich schon eine gewisse Erfahrung mit dem Erstellen einer API habe, lief dies ohne Probleme und ich musste nichts weiter daran verbessern.

### Maui APP

Das Kontrollieren der App habe ich hauptsächlich im Emulator gemacht. Dabei habe ich mich an das Testprotokoll gehalten. Jedoch habe ich es auch auf einen lokalen Device im Debugg- und Release-Modus getestet. Dabei viel mir auf, dass sich der Release-Modus nicht immer, wie der Debugg-Modus verhält.

Zum Schluss habe ich bemerkt, dass ich bei jeder Änderung der anzuzeigenden Einträge in der App einen neuen Request an die API sende und dies Performanz technisch sehr schlecht ist. Dies konnte ich lösen, indem ich eine dritte Liste erstellt habe, die jeweils als Referenzlist bestehen bleibt. Somit wird bei jeder Anzeigeänderung die anzuzeigende Liste angepasst und nur bei Löschen oder Einfügen von Einträgen wird eine Verbindung zu der API aufgebaut.

## Anforderungen erfüllt

Bei diesem Punkt habe ich kontrolliert, ob ich die grundlegenden und zusätzlichen Anforderungen des Auftrages erfüllt habe.

# Auswerten

Nach der Fertigstellung des Projektes habe ich mir Gedanken zu den positiven und negativen Momenten bei der Realisierung gemacht und die Soll- und Ist-Zeit verglichen.

Hierbei ist mir wieder einmal aufgefallen, dass meine Planung immer noch hinterherhinkt. Ich habe doppelt so viel Zeit gebraucht als geplant. Dies liegt sicher auch daran, dass ich viel Zeit in Kleinigkeiten und Dinge, die nicht machbar sind, gesteckt habe.

Dabei ist die Tabbar ein gutes Beispiel, für diese habe ich enorm viel Zeit investiert, um am Ende zum Schluss zu kommen, dass diese in Kombination mit Button-Navigation nicht funktioniert. Auch habe ich sehr viel Zeit in die Hintergrundfarbe des selektierten Elements in der ListView gesteckt. Diese habe ich bis heute nicht ändern können. Ich habe es mit einem sehr unkonventionellen/nicht absichtlichen Workaround geschafft, dass man kein Element mehr selektieren kann. Dabei habe ich es auch belassen, da so diese orange Default-Farbe von Android nicht gezeigt wird.

Auch für die Implementierung des Icons habe ich sehr viel Zeit investieren müssen, da die Grösse nie stimmte, egal wie gross das originale SVG war.

Zum Schluss habe ich auch viel Zeit für das Veröffentlichen auf dem Google Play Store verwendet, da ich nicht gewartet habe, bis wir es im Unterricht machten.

Hier der Link zum Play Store https://play.google.com/store/apps/details?id=com.Irasil.SaveUp&hl=de oder Sie suchen im Play Store nach SaveUp, dann können Sie Teil des offenen Tests werden. Da es sich um einen Free Plan von Azure handelt, dauert die Abfrage dementsprechend lange.

Mir ist bewusst, dass diese App alles andere als sicher ist, da der Connection String zu der Datenbank im API Projekt Hard-Codiert ist, jeder auf die gleiche Datenbank zugreift und somit die Daten von den Anderen User auch sieht. Somit ist diese App nicht nutzbar, aber da es sich um eine Übung im Zusammenhang mit Maui handelt, habe ich diese Aspekte ignoriert.

## Zukunft der App

In Zukunft werde ich der App noch eine "Eigene Ziele" Seite hinzufügen, bei der man Ziele eingeben kann und durch Erreichen dieser, gewisse Medaillen erhält. Diese kann man dann wiederum auf den gängigsten Sozialen Medien Plattformen teilen.

Zusätzlich denke ich, dass man sich mit Freunden vergleichen können muss, um einen weiteren Ansporn zum Sparen zu erhalten. Diese zwei Seiten stelle ich mir etwa so vor:

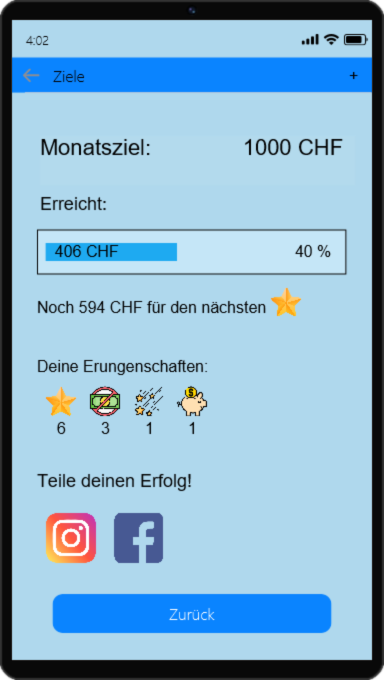


Abbildung : Wireframe Eigene Ziele definieren

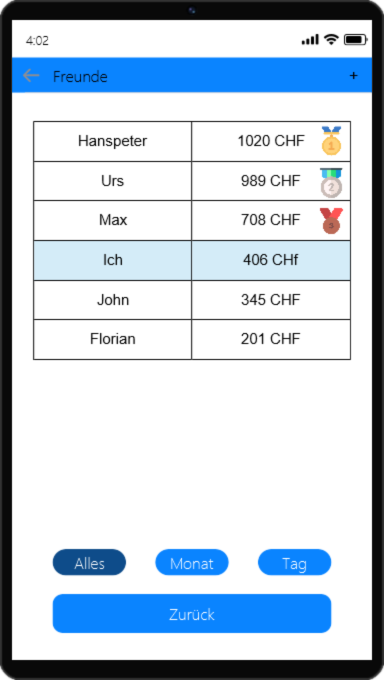


Abbildung : Wireframe Vergleichen mit Freunden

# Fazit

Dieses Projekt war mit sehr viel Frust verbunden, da Maui noch sehr neu ist und somit viele Bugs enthält. Wie zum Beispiel, das Einstellen des Default-Pickers, der maxLength bei Entrys, eine einfache Handhabung der Icons und nicht zu vergessen die Dokumentation von Microsoft, welche teilweise fehlerhaft ist.

Nichtsdestotrotz konnte ich bei diesem Projekt viel lernen. Die Zusatzaufgaben wie das externe Hosten von API und Datenbank haben mir Spass gemacht und das Wissen werde ich in Zukunft sicher wieder gebrauchen können. Auch die Erstellung einer mobilen App war sehr interessant, da ich dies zum ersten Mal gemacht habe. Ich bin gespannt, was Microsoft mit Maui noch alles vorhat, und schaue zuversichtlich in die Zukunft, in der ich Maui noch häufiger verwenden werde.

# Testprotokoll

Tabelle : Testprotokoll

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Was wir Testen | Wie wir testen | Erwartetes Ergebnis | Tatsächliches Ergebnis | Testergebnis | Wurde es gelöst | Wie wurde es gelöst |
| 1 | Home-Seite wird angezeigt | Anzeige | Home Seite wird angezeigt | Home Seite wird angezeigt | Fehlerfrei | - |  |
| 2 | Resultat Button bring uns auf die Resultate Seite | Resultate-Button klicken | Anzeigen der Resultate-Seite | Anzeige der Resultate-Seite | Fehlerfrei | - |  |
| 3 | Erfassen mit lehren Entrys nicht möglich | Erfassen-Button klicken | Alert, dass alle Felder ausgefüllt sein müssen. | Alert | Fehlerfrei | - |  |
| 4 | Bei Betrag dürfen nur positive Zahlen eingegeben werden | Buchstaben und negative Zahlen eingeben. | Alert. | Negative Zahlen gingen. | Fehlerhaft | Ja | Ein If-Statement  -> Entry = >0 |
| 5 | Maximale Länge der Artikelnamen =25 | Mehr als 25 Zeichen eingeben | Nicht möglich | Möglich | Fehlerhaft | Nein | Bug in Maui, kein Workaround gemacht, da keine Anforderung |
| 6 | Einträge erfassen | Erfassen von Eintrag | Eintrag wird über API in DB gespeichert | Eintrag wurde gespeichert | Fehlerfrei | - |  |
| 7 | Einträge löschen | Einträge löschen | Eintrag wird aus DB gelöscht. | Eintrag wurde aus DB gelöscht. | Fehlerfrei | - |  |
| 8 | Zurück Button | Button klicken | Zurück auf Home-Seite | Home-Seite wurde angezeigt | Fehlerfrei | - |  |
| 9 | Auswahl der Kategorie | Kategorie auswählen | Kategorie wird ausgewählt | Kategorie wurde ausgewählt | Fehlerfrei | - |  |
| 10 | Einträge anzeigen | Resultate Button klicken | Einträge werden angezeigt | Einträge wurden angezeigt | Fehlerfrei | - |  |
| 11 | Total anzeigen | Einträge anzeigen | Total aller Einträge wird angezeigt | Total wurde nach löschen eines Eintrages nicht geändert. | Fehlerhaft | Ja | Das gelöschte Element nicht nur aus DB löschen, sondern auch aus der Liste die Angezeigt wird |
| 12 | Auswahl der Zeitspanne die angezeigt wird | Alles, Monat oder Tag klicken | Nur Einträge des gewählten Zeitraumes werden angezeigt | Nur Einträge des gewählten Zeitraumes wurden angezeigt | Fehlerfrei | - |  |
| 13 | Icon wird richtig Angezeigt | Installieren auf Lokalem Gerät | Icon wird richtig angezeigt | Icon war zu gross Skaliert | Fehlerhaft | Ja | Workaround indem ich das Icon kleiner als das gesamte Bild gemacht habe |
| 14 | Runden aller Beträge auf 0.05 oder 0.00 | Einträge anzeigen | Alle Beträge sind auf 0.05 oder 0.00 gerundet | Beträge waren richtig gerundet | Fehlerfrei | - |  |
| 15 | Richtige Darstellung im Dark-Mode | Emulator auf Dark-Mode umgestellt | UI ist leserlich und gut erkennbar | Beschriftung der Buttons war nicht mehr zu lesen | Fehlerhaft | - | Anpassen in den Styles |

# Abbildungsverzeichnis

[Abbildung 1: Nuget Liste API 3](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657470)

[Abbildung 2: Nuget Liste Maui App 3](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657471)

[Abbildung 3: Projektorganisation 4](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657472)

[Abbildung 4: PSP SaveUp App 5](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657473)

[Abbildung 5: Gantt mit Soll und Ist Zeiten 6](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657474)

[Abbildung 6: Use Cases SaveUp 7](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657475)

[Abbildung 7: Resultate 8](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657476)

[Abbildung 8: Homepage 8](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657477)

[Abbildung 9: Homepage Version B 8](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657478)

[Abbildung 10:Resultate Version B 8](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657479)

[Abbildung 11: Konzept der Applikation 9](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657480)

[Abbildung 12: Klassen Diagramm für die API 10](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657481)

[Abbildung 13: Klassen Diagramm SaveUp 11](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657482)

[Abbildung 14: Sequenzdiagramm hinzufügen 12](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657483)

[Abbildung 15: Sequenzdiagramm Abrufen 13](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657484)

[Abbildung 16: Sequenzdiagramm löschen 14](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657485)

[Abbildung 17: Logo 14](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657486)

[Abbildung 18: Icon 14](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657487)

[Abbildung 20: Wireframe Eigene Ziele definieren 16](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657488)

[Abbildung 19: Wireframe Vergleichen mit Freunden 16](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20335/Projektarbeit/Dokumentation/SaveUpDokumentation.docx#_Toc130657489)

# Quellen

https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0

https://learn.microsoft.com/en-us/dotnet/maui/android/deployment/overview?view=net-maui-7.0

https://www.youtube.com/watch?v=jfSVb\_RR7X0