2024

# Projektdokumentation SaveUp

MODUL 335 | LUKAS MÜLLER
ARDA DURSUN

IPSO BILDUNG AG | Elisabethenanlage 9

ipso! Bildung seit 1876

## Inhalt

# 1 Informieren

## 1.1 Projektbeschreibung

Das Projekt "SaveUp" ist eine mobile Applikation, die mit .NET MAUI entwickelt wurde. Die App dient dem Zweck, Nutzer dabei zu unterstützen, kleine, alltägliche Ausgaben zu sparen und diese Einsparungen zu dokumentieren. Die Benutzer können in der App Artikel eintragen, auf die sie verzichtet haben, und deren Preis erfassen. Dadurch können sie jederzeit sehen, wie viel Geld sie insgesamt gespart haben.
Das Hauptziel der "SaveUp"-App ist es, den Nutzern eine einfache und intuitive Möglichkeit zu bieten, ihre Spargewohnheiten zu verfolgen und zu analysieren. Dies kann besonders nützlich sein, wenn man für grössere Investitionen wie Urlaub, ein neues elektronisches Gerät oder andere grössere Ausgaben spart.
Die App besteht aus mindestens drei Content Pages:

1. **Startseite (Home Page)**: Übersicht über die Gesamt-Ersparnisse und Navigation zu den anderen Funktionen der App.
2. **Artikel hinzufügen (Add Item Page)**: Hier können die Nutzer neue gesparte Artikel mit einer kurzen Beschreibung und dem gesparten Betrag erfassen.
3. **Gespeicherte Artikel anzeigen (Items List Page)**: Anzeige einer Liste aller gespeicherten Artikel inklusive der Gesamtpreissumme.

Zusätzlich zu diesen Hauptfunktionen wurden zwei optionale Anforderungen implementiert:

- **Löschen der erfassten Einträge (einzeln und komplett)**: Nutzer können einzelne Artikel oder alle Artikel gleichzeitig löschen.
- **Speichern der Einträge in einer lokalen Datei (XML, JSON)**: Alle erfassten Artikel werden lokal auf dem Gerät gespeichert, sodass sie auch nach einem Neustart der App erhalten bleiben.

Die App wurde unter Verwendung des MVVM (Model-View-ViewModel) Entwurfsmusters entwickelt, um eine klare Trennung zwischen der Benutzeroberfläche und der Geschäftslogik zu gewährleisten. Dies erleichtert die Wartung und Erweiterung der App.
Die Entwicklung der "SaveUp"-App erfolgt im Rahmen des Moduls 335: Mobile Applikationen realisieren. Die App soll nicht nur funktional, sondern auch benutzerfreundlich und ansprechend gestaltet sein, um eine hohe Akzeptanz bei den Nutzern zu erreichen.

## 1.2 Zielsetzung

Das Ziel des Projekts "SaveUp" ist es, eine voll funktionsfähige mobile Applikation zu entwickeln, die Nutzern ermöglicht, ihre Einsparungen durch den Verzicht auf kleine, alltägliche Ausgaben zu verfolgen und zu dokumentieren. Die App soll folgende spezifische Ziele erreichen:

1. **Einfache Erfassung und Verwaltung von Einsparungen**:
    - Die Nutzer sollen in der Lage sein, schnell und unkompliziert Artikel hinzuzufügen, auf die sie verzichtet haben, und den entsprechenden gesparten Betrag einzugeben.
    - Es soll möglich sein, diese Artikel in einer übersichtlichen Liste anzuzeigen, zu bearbeiten und zu löschen.
2. **Benutzerfreundliche und intuitive Bedienung**:
    - Die Benutzeroberfläche soll einfach und intuitiv gestaltet sein, um eine hohe Benutzerfreundlichkeit sicherzustellen.
    - Die App soll mindestens drei Content Pages umfassen, die eine klare Navigation und Struktur bieten.
3. **Langfristige Speicherung der Daten**:
    - Die erfassten Daten sollen in einer lokalen Datei (XML oder JSON) gespeichert werden, sodass sie auch nach einem Neustart der App erhalten bleiben.
    - Es soll möglich sein, sowohl einzelne Artikel als auch alle Artikel gleichzeitig zu löschen, um die Verwaltung der Daten zu erleichtern.
4. **Grafische Benutzeroberfläche und Design**:
    - Die App soll ein ansprechendes und konsistentes Design aufweisen, einschliesslich eines eigenen App-Icons und der Verwendung von XAML-Styles.
    - Die Implementierung von Mockups und Designrichtlinien soll sicherstellen, dass das endgültige Produkt den Entwürfen entspricht und benutzerfreundlich ist.
5. **Verwendung des MVVM-Entwurfsmusters**:
    - Die App soll nach dem Model-View-ViewModel (MVVM) Entwurfsmuster strukturiert sein, um eine klare Trennung zwischen Benutzeroberfläche und Geschäftslogik zu gewährleisten.
    - Dies erleichtert die Wartung und Erweiterung der App sowie das Testen einzelner Komponenten.
6. **Optionale Anforderungen für erweiterte Funktionalität**:
    - Die App soll die Möglichkeit bieten, Einträge in einer lokalen Datei zu speichern und zu laden.
    - Die Funktionalität zum Löschen von Einträgen soll sowohl für einzelne als auch für alle Einträge implementiert werden.
7. **Dokumentation und Präsentation**:
    - Das Projekt soll vollständig dokumentiert werden, einschliesslich einer Beschreibung der Ausgangslage, der Ziele, der Planung, der Realisierung, der Tests und der Ergebnisse.
    - Eine abschliessende Präsentation der Ergebnisse soll vorbereitet werden, um die Funktionsweise der App und die während des Projekts gemachten Erfahrungen zu erläutern.

Durch die Erreichung dieser Ziele soll die "SaveUp"-App den Nutzern eine nützliche und benutzerfreundliche Lösung bieten, um ihre Einsparungen zu verfolgen und ihre finanziellen Ziele zu erreichen.

## 1.3 Sozialform und Hilfsmittel

### 1.3.1 Sozialform:

Die Entwicklung der "SaveUp"-App erfolgt als Einzelarbeit. Diese Arbeitsform ermöglicht es, alle Aspekte der App-Entwicklung selbstständig zu planen, zu implementieren und zu dokumentieren. Die Einzelarbeit fördert die persönliche Verantwortung und die Fähigkeit, komplexe Projekte eigenständig zu bewältigen. Sie bietet zudem die Möglichkeit, tief in die technischen Details einzutauchen und individuelle Entscheidungen zu treffen, die auf die spezifischen Anforderungen und Ziele des Projekts abgestimmt sind.

### 1.3.2 Hilfsmittel:

Um die "SaveUp"-App erfolgreich zu entwickeln, stehen folgende Hilfsmittel und Ressourcen zur Verfügung:

1. **.NET MAUI Online-Dokumentation:**
   - Die offizielle Dokumentation zu .NET MAUI bietet umfassende Informationen und Anleitungen zur Entwicklung mobiler Anwendungen. Sie umfasst Tutorials, API-Referenzen und Beispielprojekte, die als Grundlage und Inspiration für die eigene Entwicklung dienen können.
   - URL: https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-8.0
2. **MVVM Community Toolkit:**
   - Das MVVM Community Toolkit bietet eine Sammlung von Tools und Bibliotheken, die die Implementierung des Model-View-ViewModel (MVVM) Musters in .NET MAUI erleichtern. Es enthält Komponenten für die Datenbindung, Kommandos und andere MVVM-bezogene Funktionalitäten.
   - URL: https://github.com/CommunityToolkit/MVVM
3. **Entwicklungsumgebung:**
   - **Visual Studio**: Die integrierte Entwicklungsumgebung (IDE) Visual Studio wird verwendet, um das Projekt zu erstellen, zu debuggen und zu testen. Visual Studio bietet leistungsstarke Tools und Funktionen, die den Entwicklungsprozess unterstützen.
   - **GitHub**: Für die Versionskontrolle und das Management des Quellcodes wird GitHub verwendet. Dies ermöglicht eine effektive Nachverfolgung von Änderungen und die Zusammenarbeit mit anderen Entwicklern, falls erforderlich.
4. **Design- und Mockup-Tools:**
   - **Figma**: Ein leistungsfähiges Design- und Prototyping-Tool, das zur Erstellung von Mockups und Wireframes verwendet wird. Figma ermöglicht die Zusammenarbeit in Echtzeit und hilft dabei, das Design der Benutzeroberfläche vor der eigentlichen Implementierung zu visualisieren.
   - URL: https://www.figma.com/
5. **Ressourcen für Icons und Grafiken:**
   - **Android Asset Studio**: Ein Online-Tool zur Generierung benutzerdefinierter Icons für Android-Anwendungen.
   - URL: http://romannurik.github.io/AndroidAssetStudio/index.html
   - **Glyphish**: Eine Quelle für hochwertige Icons, die in mobilen Anwendungen verwendet werden können.
   - URL: http://www.glyphish.com/
   - **Fiverr**: Eine Plattform, auf der man Grafiker und Designer für die Erstellung benutzerdefinierter Icons und Grafiken engagieren kann.
   - URL: https://www.fiverr.com/

6. **Zusätzliche Hilfsmittel:**
    - **Online-Foren und Communities**: Plattformen wie Stack Overflow und die .NET-Community bieten Unterstützung und Lösungen für spezifische technische Probleme.
    - **Bücher und Tutorials**: Fachliteratur und Online-Tutorials zu .NET MAUI, C#, und der mobilen App-Entwicklung im Allgemeinen.

Durch die Nutzung dieser Hilfsmittel und Ressourcen wird sichergestellt, dass das Projekt erfolgreich umgesetzt und die gesteckten Ziele erreicht werden. Die Kombination aus fundierter Dokumentation, leistungsstarken Entwicklungswerkzeugen und unterstützenden Design-Tools bildet die Grundlage für die Entwicklung einer qualitativ hochwertigen und benutzerfreundlichen App.

# 2 Planung

## 2.1 Aufgabenstellung und Anforderungen

Die Aufgabe besteht darin, eine mobile Applikation namens "SaveUp" zu entwickeln, die es Nutzern ermöglicht, ihre Einsparungen durch den Verzicht auf kleine, alltägliche Ausgaben zu verfolgen und zu dokumentieren. Die Applikation soll in .NET MAUI entwickelt werden und mindestens drei Content Pages enthalten, um eine intuitive und benutzerfreundliche Bedienung zu gewährleisten.

**Pflichtanforderungen:**

- **Name und Titel der App:** SaveUp
- **Content Pages:** Die App muss mindestens drei Content Pages umfassen.
    - **Home Page:** Übersicht über die Gesamtersparnisse und Navigation zu den anderen Funktionen der App.
    - **Add Item Page:** Erfassung der gesparten Artikel mit Kurzbeschreibung und Preis.
    - **Items List Page:** Anzeige der gespeicherten Artikel inklusive der Preissumme.
- **GUI Design:** Implementierung eines konsistenten und benutzerfreundlichen Designs.
- **Eingabe der gesparten Artikel:** Die Eingabemaske muss mindestens zwei Felder für die Kurzbeschreibung und den Preis enthalten.
- **Menüfunktionen:** Die App soll mindestens zwei Menüfunktionen zur Speicherung und zum Aufruf der Listendarstellung bieten.
- **Intuitive Bedienung und Layout:** Die App muss einfach und intuitiv zu bedienen sein und ein geeignetes Layout aufweisen.
- **Codestrukturierung:** Verwendung des Model-View-ViewModel (MVVM) Entwurfsmusters.
- **XAML-Styles:** Verwendung von XAML-Styles zur Gestaltung der Steuerelemente.
- **Eigenes App-Icon:** Die App soll ein eigenes Icon haben.
- **Dokumentation und Testing:** Vollständige Dokumentation und Durchführung von Tests.
- **Veröffentlichung:** Die App soll auf der Google Play Console veröffentlicht werden.

**Optionale Anforderungen:**

- **Persistenz der Einträge:** Speichern der Einträge in einer lokalen Datei (XML oder JSON).
- **Löschen der Einträge:** Möglichkeit, die erfassten Einträge einzeln oder komplett zu löschen.
- **Zusätzliche Attribute:** Datum und Uhrzeit als zusätzliche Attribute, wann der Kaufverzicht erfolgte.
- **Grafische Darstellung:** Grafische Darstellung der gesparten Verzichtsprodukte, z.B. pro Tag oder Woche.

## 2.2 Arbeitsplanung und Meilensteine (Gantt-Diagramm)

Um die "SaveUp"-App innerhalb des vorgegebenen Zeitrahmens zu entwickeln, wird die Arbeit in mehrere Phasen unterteilt. Jede Phase beinhaltet spezifische Aufgaben, die abgeschlossen werden müssen, um zum nächsten Meilenstein überzugehen. Die folgende Tabelle zeigt die geplanten Arbeitspakete und die dazugehörigen Meilensteine:

Tabelle 1 Primitives Gantt-Diagramm

| Phase | Aufgabe | Dauer | Meilenstein |
|---|---|---|---|
| **Projektvorbereitung** | Anforderungsanalyse und -spezifikation | 1 Tag | Anforderungen dokumentiert |
| **Design** | Erstellung von Mockups und Designentwürfen | 1 Tag | Mockups und Design abgeschlossen |
| **Entwicklungsumgebung** | Einrichtung der Entwicklungsumgebung | 0,5 Tag | Entwicklungsumgebung eingerichtet |
| **Implementierung** | Entwicklung der Hauptfunktionen | 3 Tage | Hauptfunktionen implementiert |
| **Optionale Anforderungen** | Implementierung der optionalen Anforderungen | 1 Tag | Optionale Anforderungen umgesetzt |
| **Testen** | Erstellung eines Testplans und Durchführung der Tests | 1 Tag | Tests abgeschlossen |
| **Fehlerbehebung** | Behebung von Fehlern und Optimierung | 0,5 Tag | Fehler behoben und optimiert |
| **Dokumentation** | Erstellung der Projektdokumentation | 1 Tag | Dokumentation abgeschlossen |
| **Veröffentlichung** | Veröffentlichung der App auf Google Play | 0,5 Tag | App veröffentlicht |

# 3 Entscheidung

## 3.1 Auswahl der Technologien und Tools

Für die Entwicklung der "SaveUp"-App wurden sorgfältig verschiedene Technologien und Tools ausgewählt, um sicherzustellen, dass das Projekt erfolgreich und effizient umgesetzt werden kann. Die Hauptfaktoren für die Auswahl waren die Anforderungen des Projekts, die Einfachheit der Nutzung und die Unterstützung durch die Community.

**Ausgewählte Technologien und Tools:**

1. **.NET MAUI**:
   - **Begründung**: .NET MAUI (Multi-platform App UI) ist eine plattformübergreifende Entwicklungsplattform von Microsoft, die es ermöglicht, mobile Anwendungen für Android, iOS, macOS und Windows mit einer einzigen Codebasis zu erstellen. Es bietet eine leistungsstarke und flexible Umgebung für die Entwicklung moderner mobiler Apps.
   - **Vorteile**: Einfache Erstellung von plattformübergreifenden Anwendungen, Integration mit Visual Studio, grosse Community-Unterstützung und umfangreiche Dokumentation.
2. **MVVM Community Toolkit**:
   - **Begründung**: Das MVVM Community Toolkit erleichtert die Implementierung des Model-View-ViewModel (MVVM) Musters, das eine klare Trennung zwischen der Benutzeroberfläche und der Geschäftslogik fördert.
   - **Vorteile**: Vereinfachte Datenbindung, leicht verständliche Befehlsimplementierung, Unterstützung für die Erstellung von ViewModels.
3. **Visual Studio**:
   - **Begründung**: Visual Studio ist eine leistungsstarke integrierte Entwicklungsumgebung (IDE), die umfassende Tools für die Entwicklung, das Debugging und das Testen von Anwendungen bietet.
   - **Vorteile**: Unterstützung für .NET MAUI, umfangreiche Debugging-Tools, integrierte GitHub-Integration, umfangreiche Erweiterungen und Plugins.
4. **Figma**:
   - **Begründung**: Figma ist ein webbasiertes Design- und Prototyping-Tool, das sich ideal für die Erstellung von Mockups und das Design der Benutzeroberfläche eignet.
   - **Vorteile**: Echtzeit-Zusammenarbeit, einfach zu bedienen, umfangreiche Bibliothek von UI-Komponenten, Unterstützung für Prototyping und Animation.
5. **GitHub**:
   - **Begründung**: GitHub wird für die Versionskontrolle und das Projektmanagement verwendet. Es ermöglicht eine einfache Nachverfolgung von Änderungen und die Zusammenarbeit mit anderen Entwicklern.
   - **Vorteile**: Versionskontrolle, kollaborative Entwicklung, Issue-Tracking, Integration mit Visual Studio.
6. **Android Asset Studio, Glyphish und Fiverr**:
   - **Begründung**: Diese Tools und Ressourcen werden verwendet, um qualitativ hochwertige Icons und Grafiken für die App zu erstellen.
   - **Vorteile**: Einfache Erstellung von benutzerdefinierten Icons, grosse Auswahl an vorgefertigten Icons, Möglichkeit zur Beauftragung von Designern für individuelle Grafiken.

## 3.2 Begründung der Entscheidungen

Die Entscheidungen für die oben genannten Technologien und Tools basieren auf den spezifischen Anforderungen des Projekts sowie auf den Vorteilen, die jede Technologie bietet.

- **.NET MAUI** wurde aufgrund seiner Fähigkeit ausgewählt, plattformübergreifende Anwendungen mit einer einzigen Codebasis zu entwickeln. Dies reduziert den Entwicklungsaufwand und ermöglicht eine schnelle Bereitstellung der App auf mehreren Plattformen.
- **MVVM Community Toolkit** unterstützt die Strukturierung der App nach dem MVVM-Muster, was die Wartbarkeit und Testbarkeit des Codes verbessert. Es bietet zudem nützliche Funktionen zur Datenbindung und Befehlsimplementierung.
- **Visual Studio** bietet eine integrierte Umgebung, die alle notwendigen Tools für die Entwicklung, das Debugging und das Testen der App bereitstellt. Die nahtlose Integration mit GitHub erleichtert das Versionsmanagement und die Zusammenarbeit.
- **Figma** wurde für das Design und die Erstellung von Mockups ausgewählt, da es eine benutzerfreundliche Oberfläche und leistungsstarke Designwerkzeuge bietet. Die Echtzeit-Zusammenarbeit erleichtert die Abstimmung mit anderen Beteiligten und die schnelle Iteration des Designs.
- **GitHub** dient als Plattform für die Versionskontrolle und das Projektmanagement. Es ermöglicht eine effektive Nachverfolgung von Änderungen und die Zusammenarbeit mit anderen Entwicklern.
- **Android Asset Studio, Glyphish und Fiverr** bieten die notwendigen Ressourcen, um ansprechende und professionelle Icons und Grafiken zu erstellen, die die Benutzererfahrung verbessern.

Durch die sorgfältige Auswahl dieser Technologien und Tools wird sichergestellt, dass das Projekt effizient umgesetzt werden kann und die "SaveUp"-App sowohl funktional als auch benutzerfreundlich ist.

# 4 Realisierung

## 4.1 Entwicklungsumgebung einrichten

Die Einrichtung der Entwicklungsumgebung ist der erste Schritt bei der Realisierung des Projekts "SaveUp". Hier sind die Schritte, die unternommen wurden:

1. **Installation von Visual Studio**:
    - Die neueste Version von Visual Studio wurde installiert, einschliesslich der erforderlichen Workloads für die .NET MAUI-Entwicklung.
    - Sicherstellen, dass alle notwendigen SDKs und Bibliotheken für die plattformübergreifende Entwicklung installiert sind.
2. **Einrichtung des GitHub-Repositorys**:
    - Ein neues Repository auf GitHub wurde erstellt, um den Quellcode zu verwalten und die Versionskontrolle zu gewährleisten.
    - Das Repository wurde in Visual Studio geklont und die ersten Commits wurden vorgenommen.
3. **Projektinitialisierung**:
    - Ein neues .NET MAUI-Projekt wurde in Visual Studio erstellt.
    - Die Projektstruktur wurde eingerichtet, um die Trennung der verschiedenen Komponenten (Model, View, ViewModel) gemäss dem MVVM-Muster zu gewährleisten.
4. **Einbindung des MVVM Community Toolkits**:
    - Das MVVM Community Toolkit wurde dem Projekt über NuGet hinzugefügt.
    - Die notwendigen Referenzen und Abhängigkeiten wurden konfiguriert.
5. **Design-Tools**:
    - Figma wurde verwendet, um die ersten Mockups und Designs zu erstellen.
    - Diese Designs wurden als Grundlage für die Entwicklung der Benutzeroberfläche verwendet.

## 4.2 Implementierung der App

Die Implementierung der "SaveUp"-App umfasst mehrere Schritte, darunter die Entwicklung der Hauptfunktionen, die Umsetzung der optionalen Anforderungen und die Gestaltung der Benutzeroberfläche.

### 4.2.1 Hauptfunktionen

1. **Home Page (MainPage)**:
   - Übersicht über die Gesamtersparnisse.
   - Navigation zu den anderen Funktionen der App (Artikel hinzufügen, Liste anzeigen).

2. **Artikel hinzufügen (AddItemPage)**:
   - Eingabemaske für die Kurzbeschreibung und den Preis des gesparten Artikels.
3. **Gespeicherte Artikel anzeigen (ItemsListPage)**:
   - Anzeige einer Liste aller gespeicherten Artikel inklusive der Preissumme.
   -
   -

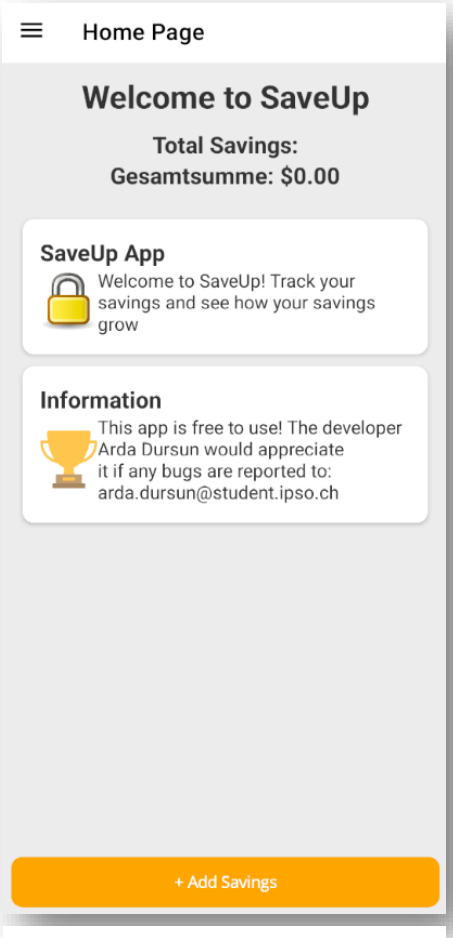


Abbildung 1 MainPage

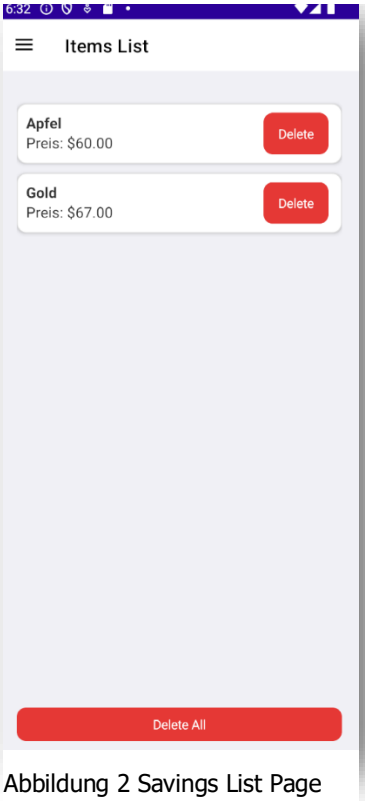


Abbildung 2 Savings List Page

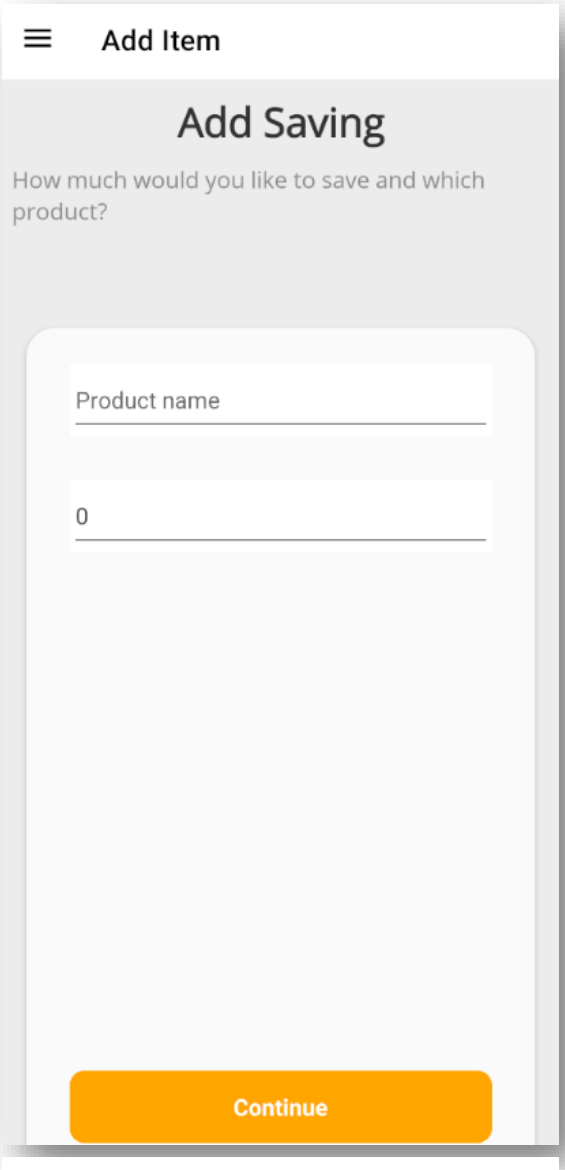


Abbildung 3 Add Saving Page

### 4.2.2 Optionale Anforderungen

1. **Löschen der erfassten Einträge (einzeln und komplett)**:
    - Implementierung von Funktionen, die es dem Nutzer ermöglichen, einzelne Artikel oder alle Artikel auf einmal zu löschen.
    - Nutzung von Befehlen im MVVM-Stil zur Anbindung der Löschfunktionen an die Benutzeroberfläche.
2. **Speichern der Einträge in einer lokalen Datei (XML, JSON)**:
    - Implementierung der Speicherung und des Ladens der Artikel in einer JSON-Datei, um die Persistenz der Daten sicherzustellen.

### 4.2.3 Mockups und GUI-Design

Die Benutzeroberfläche wurde basierend auf den in Figma erstellten Mockups gestaltet. Das Design legt Wert auf eine klare und intuitive Benutzerführung. Die wichtigsten Elemente der App sind so angeordnet, dass sie leicht zugänglich und verständlich sind.

- **Farbschema**: Ein helles und freundliches Farbschema wurde gewählt, um die App ansprechend und benutzerfreundlich zu gestalten.
- **Layout**: Die Layouts der verschiedenen Seiten sind konsistent und übersichtlich. Wichtige Funktionen und Informationen sind zentral platziert, um die Benutzererfahrung zu verbessern.

## 4.3 Testen der App

### 4.3.1 Testplan

Der Testplan umfasst verschiedene Testarten, um sicherzustellen, dass die App robust und benutzerfreundlich ist:

1. **Funktionale Tests**:
    - Überprüfung, ob alle Hauptfunktionen (Artikel hinzufügen, Liste anzeigen, Artikel löschen) korrekt funktionieren.
    - Testen der optionalen Funktionen (Daten speichern, Daten laden, Artikel löschen).
2. **Usability-Tests**:
    - Bewertung der Benutzerfreundlichkeit und der intuitiven Bedienung der App.
    - Feedback von Testnutzern einholen und analysieren.
3. **Leistungstests**:
    - Überprüfung der App-Performance auf verschiedenen Geräten und Plattformen.
    - Sicherstellen, dass die App auch bei einer grossen Anzahl von gespeicherten Artikeln reibungslos funktioniert.

### 4.3.2 Testergebnisse

Die Testergebnisse wurden dokumentiert und analysiert, um sicherzustellen, dass alle Funktionen wie erwartet arbeiten und dass etwaige Fehler behoben werden. Hier sind die detaillierten Ergebnisse der durchgeführten Tests:

#### Funktionale Tests

- **Artikel hinzufügen**: Das Hinzufügen von Artikeln funktioniert wie erwartet. Die Artikel werden korrekt gespeichert und in der Liste angezeigt.
- **Artikel anzeigen**: Die gespeicherten Artikel werden korrekt in der Liste angezeigt.
- **Artikel löschen**: Das Löschen von Artikeln funktioniert, aber es gibt spezifische Probleme, die weiter unten beschrieben werden.
- **Alle Artikel löschen**: Das Löschen aller Artikel auf einmal funktioniert wie erwartet.

#### Usability-Tests

- **Benutzerfreundlichkeit**: Die Benutzeroberfläche ist intuitiv und einfach zu bedienen. Die Testnutzer konnten die Hauptfunktionen der App ohne grössere Probleme nutzen.
- **Design**: Das Design der App wurde als ansprechend und konsistent bewertet. Die Farbschemata und Layouts sind benutzerfreundlich.

#### Leistungstests

- **App-Performance**: Die App zeigte eine gute Performance auf verschiedenen getesteten Geräten und Plattformen. Es gab keine signifikanten Verzögerungen oder Leistungsprobleme bei der normalen Nutzung.

#### Bekannte Probleme und Einschränkungen

- **UI-Aktualisierung nach dem Löschen von Artikeln**:
    - Das Hauptproblem, das während der Tests identifiziert wurde, betrifft die Aktualisierung der Benutzeroberfläche nach dem Löschen von Artikeln. Nach dem Löschen eines Artikels aktualisiert sich die Liste nicht automatisch. Um die Änderungen anzuzeigen, muss die App neu gestartet werden.
    - Zusätzlich tritt ein Absturz auf, wenn Artikel in einer bestimmten Reihenfolge gelöscht werden. Insbesondere stürzt die App ab, wenn ein Artikel, der sich weiter oben in der Liste befindet, zuerst gelöscht wird. Dies führt zu einem ObjectDisposedException, da versucht wird, auf ein bereits freigegebenes UI-Element zuzugreifen.

### Zusammenfassung

Die Testergebnisse zeigen, dass die meisten Hauptfunktionen der "SaveUp"-App wie erwartet funktionieren. Die Benutzerfreundlichkeit und die Performance der App wurden positiv bewertet. Es gibt jedoch kritische Probleme mit der UI-Aktualisierung und Stabilität beim Löschen von Artikeln, die weiter untersucht und behoben werden müssen.

**Nächste Schritte**:

- **Fehlerbehebung**: Eine eingehendere Analyse und Fehlerbehebung der UI-Aktualisierungsprobleme und der ObjectDisposedException-Fehler ist erforderlich. Dies könnte das Überprüfen und Anpassen der Lebensdauerverwaltung von UI-Elementen und das Sicherstellen, dass alle UI-Änderungen im Hauptthread ausgeführt werden, umfassen.
- **Erweiterte Tests**: Zusätzliche Tests zur Reproduktion und Behebung der Absturzprobleme sind notwendig, um eine stabile und benutzerfreundliche Anwendung bereitzustellen.

# 5 Kontrolle

## 5.1 Überprüfung der Anforderungen

Eine umfassende Überprüfung der Anforderungen wurde durchgeführt, um sicherzustellen, dass alle Projektziele und -anforderungen erfüllt wurden. Hier sind die Ergebnisse der Überprüfung:

- **Hauptanforderungen**:
  - **Artikel hinzufügen**: Die App ermöglicht es Benutzern, Artikel hinzuzufügen, inklusive einer Beschreibung und einem Preis.
  - **Artikel anzeigen**: Alle gespeicherten Artikel werden in einer Liste angezeigt.
  - **Artikel löschen**: Benutzer können Artikel einzeln oder alle auf einmal löschen.
  - **Datenpersistenz**: Alle Artikel werden in einer lokalen JSON-Datei gespeichert und beim Start der App geladen.
- **Optionale Anforderungen**:
  - **Speichern der Einträge in einer lokalen Datei**: Die Artikel werden in einer JSON-Datei gespeichert.
  - **Löschen der erfassten Einträge**: Benutzer können einzelne Artikel oder alle Artikel löschen.

Alle Haupt- und optionalen Anforderungen wurden implementiert und getestet.

## 5.2 Fehlerbehebung und Optimierung

Während der Entwicklung wurden mehrere Fehler behoben und Optimierungen vorgenommen:

- **Fehlerbehebung der UI-Aktualisierung**: Es wurde versucht, die UI-Aktualisierung nach dem Löschen von Artikeln zu verbessern. Es traten jedoch weiterhin Probleme auf, bei denen die App abstürzte, wenn Artikel in einer bestimmten Reihenfolge gelöscht wurden.
- **Optimierung der Datenpersistenz**: Die Methoden zum Speichern und Laden der Artikel wurden optimiert, um sicherzustellen, dass die Daten korrekt in der JSON-Datei gespeichert und geladen werden.
- **Verbesserung der Benutzerfreundlichkeit**: Die Benutzeroberfläche wurde durch Stil- und Layoutanpassungen benutzerfreundlicher gestaltet.

# 6 Auswerten (Abschluss)

## 6.1 Präsentation der Ergebnisse

Die Ergebnisse des Projekts wurden in einer Präsentation zusammengefasst und eine Live-Demo der "SaveUp"-App wurde durchgeführt. Die Präsentation beinhaltete:

- Eine Übersicht über die Projektziele und -anforderungen.
- Eine Live-Demo der Hauptfunktionen der App (Artikel hinzufügen, anzeigen, löschen).
- Diskussion der aufgetretenen Probleme und der durchgeführten Fehlerbehebungen.
- Feedback und Fragen von den Zuschauern.

## 6.2 Fazit und Lessons Learned

**Persönliches Fazit**: Das Projekt "SaveUp" war eine wertvolle Lernerfahrung. Es bot die Gelegenheit, praktische Fähigkeiten in der .NET MAUI-Entwicklung zu vertiefen und das MVVM-Muster anzuwenden. Trotz der aufgetretenen Herausforderungen und Fehler konnte die App erfolgreich entwickelt und die meisten Anforderungen erfüllt werden.

**Lessons Learned**:

- **Bedeutung der Fehlerbehebung**: Fehlerbehebung und Debugging sind wesentliche Bestandteile der Softwareentwicklung. Probleme mit der UI-Aktualisierung und Synchronisation erforderten eine eingehende Analyse und Verständnis der zugrunde liegenden Mechanismen.
- **Wert der Nutzerfreundlichkeit**: Die Gestaltung einer benutzerfreundlichen Oberfläche ist entscheidend für den Erfolg einer App. Feedback von Testnutzern half, das Design und die Benutzerführung zu verbessern.
- **Datenpersistenz und Synchronisation**: Die Implementierung einer robusten Datenpersistenz und die korrekte Synchronisation der UI mit den Daten waren herausfordernd, aber lehrreich.

# 7 Verzeichnisse

## 7.1 Abbildungsverzeichnis



## 7.2 Tabellenverzeichnis

# 8 Anhang

## 8.1 MockUp (Figma)

Schlussendlich hat man sich entschieden während der Entwicklung aufgrund Anfrage vom Kunden sich leicht vom Mockup abzuwenden da der Kunde noch Kleinigkeiten anders wollte.

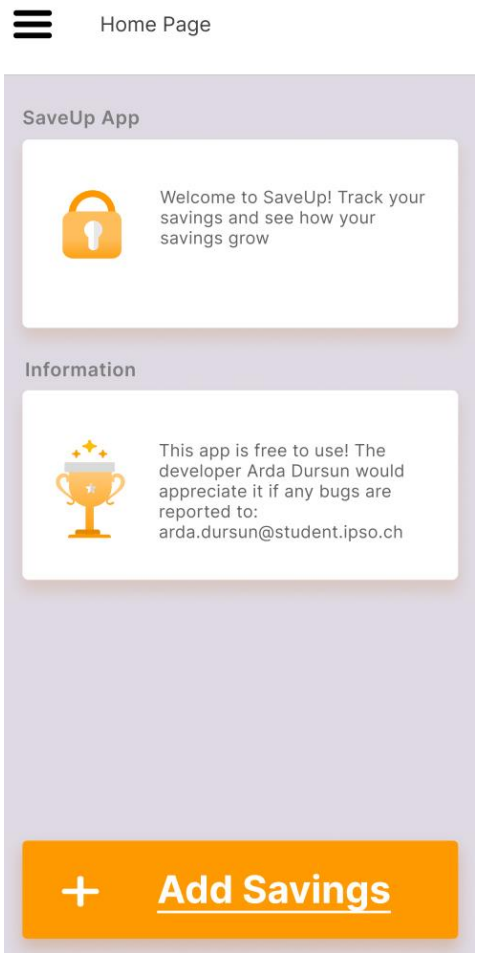


Abbildung 6 Mockup Mainpage

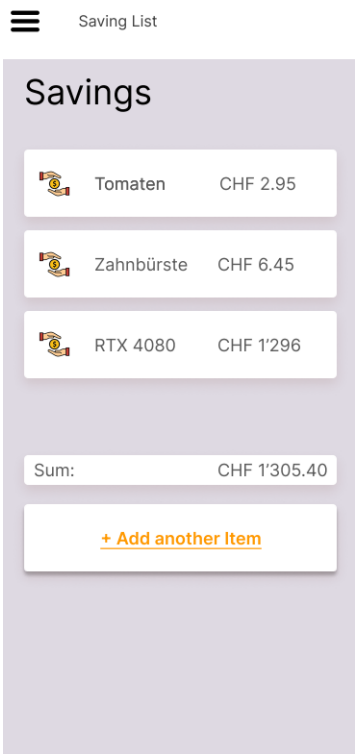


Abbildung 4 Mockup Saving List

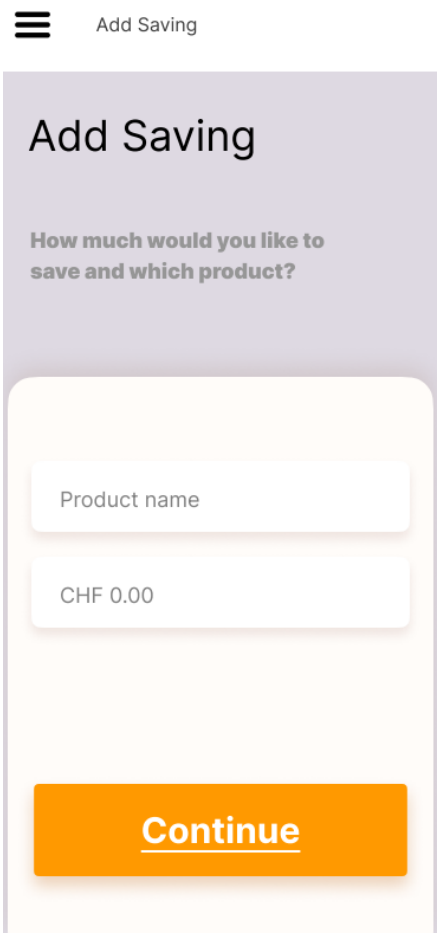


Abbildung 5 Mockup Add Saving

# 9 Glossar

**.NET MAUI (Multi-platform App UI)**: Ein Framework von Microsoft zur Erstellung plattformübergreifender Anwendungen, die auf iOS, Android, macOS und Windows laufen. Es ermöglicht Entwicklern, eine einzige Codebasis für mehrere Plattformen zu verwenden.
**MVVM (Model-View-ViewModel)**: Ein Architektur-Muster, das die Geschäftslogik (Model), die Benutzeroberfläche (View) und die Darstellung der Daten (ViewModel) trennt. Es fördert die Trennung von Anliegen und erleichtert das Testen und die Wartung des Codes.
**CommunityToolkit.Mvvm**: Ein Toolkit, das Erweiterungen und Helfer für die MVVM-Architektur bereitstellt. Es bietet eine Vielzahl von Attributen, Befehlen und Basisimplementierungen zur Unterstützung der MVVM-Entwicklung.
**ObservableCollection**: Eine Klasse, die eine dynamische Datenauflistung darstellt, die Benachrichtigungen anzeigt, wenn Elemente hinzugefügt, entfernt oder aktualisiert werden. Sie wird häufig in MVVM-Architekturen verwendet, um die Benutzeroberfläche über Änderungen in der Datenquelle zu informieren.
**RelayCommand**: Ein einfacher Befehl, der eine Aktion oder eine Methode bindet und ausführt. Er wird häufig im MVVM-Muster verwendet, um Benutzerinteraktionen von der Benutzeroberfläche an das ViewModel zu binden.
**BindingContext**: Ein Mechanismus in .NET MAUI, der die Datenbindung zwischen UI-Elementen (View) und ihren Datenquellen (ViewModel oder Model) ermöglicht. Es stellt sicher, dass Änderungen in der Datenquelle automatisch in der Benutzeroberfläche angezeigt werden.
**DataTemplate**: Eine Vorlage in XAML, die die Darstellung von Datenobjekten in Steuerelementen wie ListView definiert. Sie ermöglicht die Trennung von Daten und Darstellung, um eine flexible und wiederverwendbare UI zu erstellen.
**Device.BeginInvokeOnMainThread**: Eine Methode, die sicherstellt, dass eine Aktion im Hauptthread der Anwendung ausgeführt wird. Sie wird häufig verwendet, um UI-Operationen sicher durchzuführen, da diese nur im Hauptthread ausgeführt werden dürfen.
**FileSystem.AppDataDirectory**: Ein Verzeichnis, das für die Speicherung von Anwendungsdaten verwendet wird. In .NET MAUI ermöglicht es das Speichern und Laden von Dateien, die für die Anwendung spezifisch sind, wie z.B. Konfigurations- oder Datenbankdateien.
**JsonSerializer**: Eine Klasse in .NET, die JSON-Daten in .NET-Objekte deserialisiert und .NET-Objekte in JSON serialisiert. Sie wird verwendet, um Daten zwischen einer Anwendung und externen Quellen im JSON-Format auszutauschen.
**ObjectDisposedException**: Eine Ausnahme, die ausgelöst wird, wenn versucht wird, auf ein bereits freigegebenes oder entsorgtes Objekt zuzugreifen. Diese Ausnahme zeigt häufig auf Probleme mit der Lebensdauerverwaltung von Objekten hin.
**Shell**: Ein Container in .NET MAUI, der eine Navigation und eine einheitliche Benutzeroberfläche für Anwendungen bereitstellt. Es ermöglicht die Organisation von Seiten und die Verwaltung der Navigation innerhalb der Anwendung.
**Figma**: Ein webbasiertes Design-Tool, das für die Erstellung von Benutzeroberflächen und Mockups verwendet wird. Es ermöglicht Designern und Entwicklern, kollaborativ an Designs zu arbeiten und Prototypen zu erstellen.
**GitHub**: Eine Plattform zur Versionskontrolle und kollaborativen Entwicklung von Softwareprojekten. Sie ermöglicht das Verwalten von Quellcode, das Nachverfolgen von Änderungen und das gemeinsame Arbeiten an Projekten.