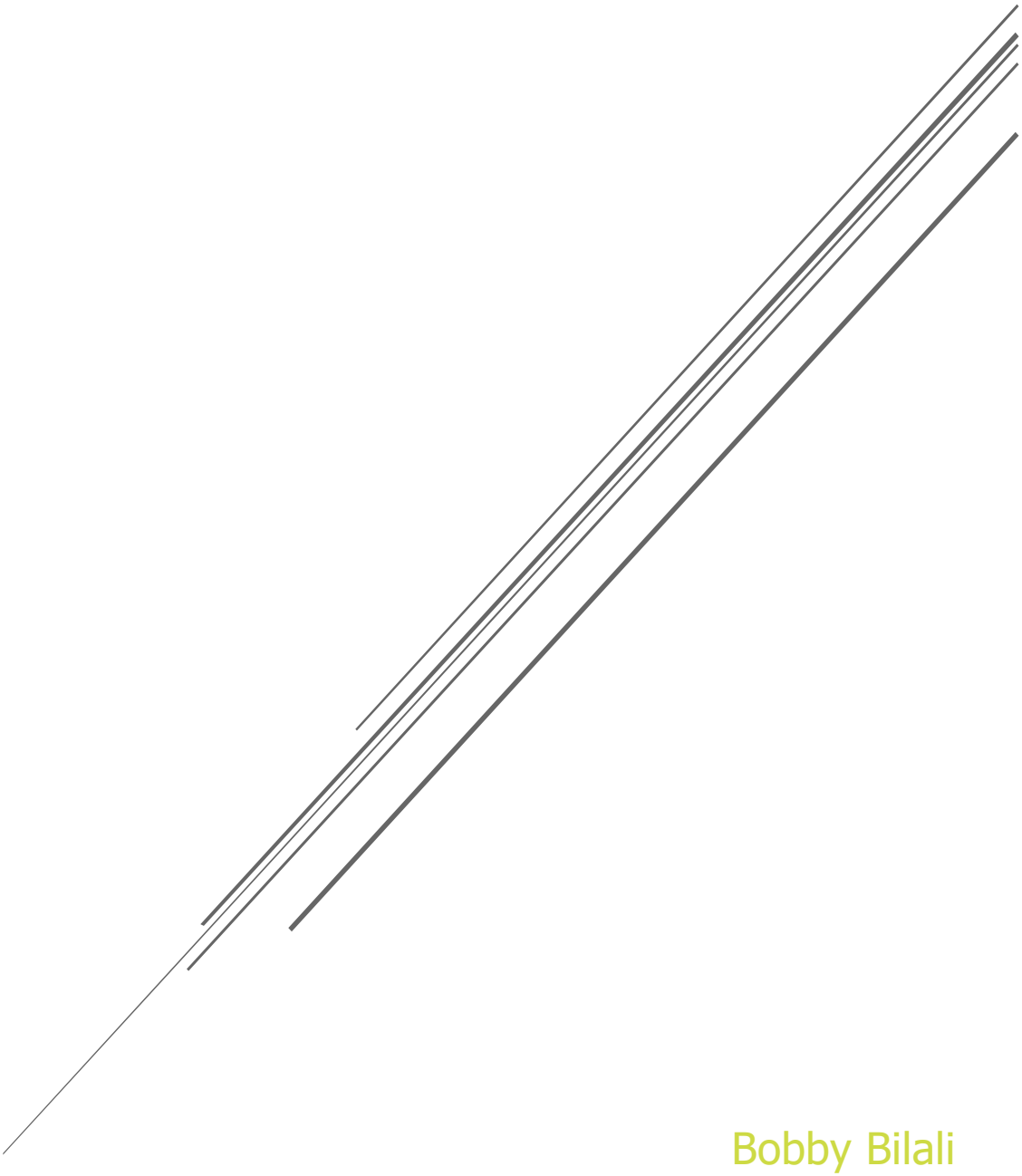
ipso! Bildung seit 1876

MODUL 335

Projektdokumentation



Bobby Bilali

.NET MAUI SAVEUP APP

ipso! Bildung seit 1876

Inhaltsverzeichnis

1 Vesrsionsverzeichnis

Version	Autor	Datum	Änderung
1.0	Bobby Bilali	16.06.2024	Erstellung des Dokuments
1.1	Bobby Bilali	17.06.2024	Erstellung der Planung
1.2	Bobby Bilali	18.06.2024	Grobe Fertigstellung des Dokuments
1.3	Bobby Bilali	19.06.2024	Rechtschreibprüfung und Textkorrekturen

Tabelle 1: Versionsverzeichnis

2 Executive Summary: "SaveUp-App"

Im Rahmen des Moduls 335 "Mobile-Applikation realisieren" wurde die SaveUp-App entwickelt, um Nutzern dabei zu helfen, Geld für grössere private Investitionen, wie z.B. Ferien, zu sparen. Die App ermöglicht es den Nutzern, auf kleine übliche Ausgaben zu verzichten und diese Ersparnisse in der App festzuhalten.

Das Projekt umfasst die Entwicklung einer .NET MAUI Anwendung mit mindestens drei Content Pages, die es dem Nutzer ermöglichen, eine Kurzbeschreibung und den Preis der gesparten Artikel zu erfassen und zu speichern. Die gespeicherten Artikel werden in einer Liste angezeigt, die auch die Gesamtsumme der Ersparnisse darstellt.

3 Einleitung

Dieses Dokument dient als grundlegende Projektdokumentation für die Entwicklung der SaveUp-App im Rahmen des Moduls 335 "Mobile-Applikation realisieren". Das Hauptziel dieses Projekts ist die Erstellung einer mobilen Anwendung, die Nutzern ermöglicht, durch den Verzicht auf kleine Ausgaben Geld zu sparen und diese Ersparnisse zu verfolgen. Ein wesentlicher Schwerpunkt liegt auf der Implementierung einer .NET MAUI App mit mindestens drei Content Pages, die eine intuitive und benutzerfreundliche Erfassung und Verwaltung der gesparten Beträge ermöglicht. Die App unterstützt die Benutzer dabei, ihre finanziellen Ziele zu erreichen, indem sie eine klare Übersicht über die gesparten Beträge bietet.

4 Ausgangslage

In der heutigen Zeit möchten viele Menschen ihre Ausgaben besser verwalten und für größere Investitionen sparen. Dazu gehört der Verzicht auf alltägliche kleine Ausgaben wie Kaffee oder Süßigkeiten, deren Ersparnisse über die Zeit einen erheblichen Betrag ausmachen können. Allerdings fehlt es oft an geeigneten Werkzeugen, um diese Ersparnisse effizient zu verfolgen und zu visualisieren.
Die SaveUp-App wurde entwickelt, um dieses Problem zu lösen. Sie ermöglicht es den Nutzern, auf einfache und intuitive Weise ihre gesparten Beträge zu erfassen und den Gesamtbetrag ihrer Ersparnisse im Blick zu behalten.
Das Projekt umfasst die folgenden Hauptaufgaben:

- **Design und Implementierung der App:** Entwicklung einer .NET MAUI Anwendung mit drei Content Pages zur Erfassung und Anzeige der gesparten Beträge.
- **Datenverwaltung:** Speichern der erfassten Daten lokal oder in einer Backend-Datenbank, um die Datenpersistenz zu gewährleisten.
- **Benutzerfreundlichkeit:** Gestaltung eines intuitiven und ansprechenden User Interfaces, das die einfache Bedienung der App sicherstellt.
- **Erweiterte Funktionen:** Optionales Hinzufügen von Funktionen wie dem Löschen von Einträgen und der grafischen Darstellung der Ersparnisse über verschiedene Zeiträume.
- **Dokumentation und Testing:** Ausführliche Dokumentation des Projekts sowie umfassende Tests zur Sicherstellung der Funktionalität und Benutzerfreundlichkeit.

Die erfolgreiche Umsetzung dieses Projekts wird den Nutzern helfen, ihre finanziellen Ziele zu erreichen, indem sie ihre Ersparnisse besser verwalten und visualisieren können. Dies trägt zur Förderung eines bewussteren Umgangs mit den täglichen Ausgaben bei und unterstützt die Nutzer dabei, größere Investitionen effektiver zu planen und zu realisieren.

5 Anforderungen

Für die SaveUp-App sind folgende Kernfunktionen erforderlich:

- **Artikel erfassen:** Eingabe einer Kurzbeschreibung und des Preises des gesparten Artikels.
- **Liste der gesparten Artikel:** Anzeigen der erfassten Artikel in einer Liste mit der Gesamtsumme der Ersparnisse.
- **Navigation:** Mindestens drei Content Pages zur Strukturierung der App.
- **Speicherung:** Möglichkeit zur Speicherung und zum Aufruf der gesparten Artikel.

Zusätzliche Anforderungen zur Erweiterung der Funktionalität:

- **Persistenz:** Speichern der Einträge in einer lokalen Datei (XML, JSON) oder einer Backend-Datenbank.
- **Löschfunktion:** Löschen einzelner oder aller Einträge.
- **Zeitstempel:** Erfassen von Datum und Uhrzeit des Verzichts.
- **Grafische Darstellung:** Visualisierung der Ersparnisse über verschiedene Zeiträume (z.B. pro Tag, Woche).

Diese Anforderungen bilden die Grundlage für die Entwicklung der SaveUp-App und stellen sicher, dass die Anwendung den Bedürfnissen der Nutzer gerecht wird, indem sie eine benutzerfreundliche und effiziente Verwaltung ihrer Ersparnisse ermöglicht.

5.1 Zusammenfassung der Anforderungen

Nr.	Beschreibung
1	Name und Titel der App: SaveUp
2	App besteht aus mindestens 3 Content Pages
3	GUI Design (Mock-ups)
4	Produkterfassung mit mindestens 2 Eingaben (Kurzbeschreibung und Preis)
5	Zwei Menüfunktionen zur Speicherung und zum Aufruf der Listendarstellung
6	Einfache und intuitive Bedienung, geeignetes Layout (Styles)
7	Codestrukturierung nach Model View ViewModel (MVVM) Entwurfsmuster
8	Verwendung von XAML-Styles für die Steuerelemente
9	Eigenes App-Icon

Tabelle 1: Zusammenfassung Anforderungen

5.2 Zusätzliche Anforderungen

Nr.	Beschreibung
1	Speichern der Einträge in einer lokalen Datei (XML, JSON)
2	Speichern der Einträge in einer Backend-Datenbank mit REST Implementierung
3	Löschen der erfassten Einträge (Clear), einzeln und komplett
4	Datum/Uhrzeit als zusätzliches Attribut, um den Zeitpunkt des Kaufverzichts zu erfassen
5	Grafische Darstellung der gesparten Verzichtsprodukte, z.B. pro Tag, Woche usw.

Tabelle 2: Zusätzliche Anforderungen

6 Informieren

In dieser Phase liegt der Fokus auf der umfassenden Informationsbeschaffung, die für die Entwicklung der SaveUp-App von entscheidender Bedeutung ist. Hierzu gehört die gründliche Analyse der technischen Anforderungen, die Erkundung der erforderlichen Systemarchitektur und die Bewertung von Benutzerfreundlichkeits- und Designaspekten.

Das Hauptziel besteht darin, ein tiefes Verständnis für die technischen Herausforderungen und die spezifischen Bedürfnisse der SaveUp-App zu entwickeln. Dies bildet die Grundlage für eine erfolgreiche Umsetzung der App und gewährleistet, dass die zukünftige SaveUp-Infrastruktur den operativen Anforderungen und der Benutzererfahrung gerecht wird.

6.1 Vorgabe/Anforderungen

Im Rahmen des SaveUp-App-Projekts ergeben sich folgende Kernelemente und Hauptziele:

- **Entwicklung einer stabilen und benutzerfreundlichen App:** Die App soll Nutzern ermöglichen, gesparte Beträge durch Verzicht auf kleine Ausgaben zu erfassen und anzuzeigen.
- **Implementierung einer effizienten Datenverwaltung:** Dies umfasst die Speicherung der Einträge entweder lokal (XML, JSON) oder in einer Backend-Datenbank.
- **Gestaltung einer intuitiven Benutzeroberfläche:** Die App muss ein ansprechendes und einfach zu bedienendes Design bieten, unterstützt durch Mock-ups und XAML-Styles.
- **Sicherstellung der Datenintegrität und Benutzerfreundlichkeit:** Die App soll eine zuverlässige Speicherung und Verwaltung der Daten gewährleisten, inklusive optionaler Funktionen wie dem Löschen von Einträgen und der grafischen Darstellung der Ersparnisse.
- **Einhaltung der strukturellen Vorgaben:** Die Anwendung soll nach dem Model View ViewModel (MVVM) Entwurfsmuster strukturiert und umgesetzt werden.

Die Umsetzung dieser Anforderungen zielt darauf ab, eine leistungsfähige und skalierbare Architektur zu schaffen, die den spezifischen Bedürfnissen der Nutzer gerecht wird. Darüber hinaus wird die App auf die nahtlose Integration zusätzlicher Funktionen und zukünftiger Erweiterungen vorbereitet.

6.2 Zusätzliche Anforderungen

Es müssen mindestens zwei zusätzliche Anforderungen umgesetzt werden:

- **Speichern der Einträge in Backend-Datenbank mit REST Implementierung:** Entwicklung einer Backend-Schnittstelle zur Speicherung der Daten in einer Datenbank, die über RESTful API angesprochen wird.
- **Datum/Uhrzeit als zusätzliches Attribut:** Erfassen und Speichern des Datums und der Uhrzeit, wann der Verzicht erfolgte, um eine detailliertere Nachverfolgung zu ermöglichen.

7 Planen

In der Planungsphase geht es darum, die in der Informationsphase gesammelten Daten und Erkenntnisse in einen strukturierten und umsetzbaren Plan zu überführen. Diese Phase ist entscheidend, um die Grundlage für die erfolgreiche Realisierung des Projekts zu legen.

7.1 Zeitplanung

Das Projekt findet innerhalb von 10 Tagen statt und wird in mehreren Phasen durchgeführt

Projektphase	Geplante Zeit
Informieren	3h
Planung, Entwurf, Entscheidung	6.50h
Realisierung	9h
Abnahme	5.50h
Dokumentation	8h
Gesamt	**32h**

Tabelle 2: Grobe Planung

7.2 Projektplanung

Ein Projektplan ist ein unverzichtbares Instrument im Projektmanagement, welches für die Definition und Visualisierung der Struktur und des Umfangs eines Projekts verwendet wird. Durch den Plan wird das gesamte Projekt in überschaubare Abschnitte oder Arbeitspakete unterteilt, was die Planung, Überwachung und Steuerung, vereinfacht.

Das Gantt ermöglicht es, den Projektumfang klar zu definieren und die benötigten Ressourcen für jede Phase festzulegen.

	Arbeitspaket	Verantwortlich	Aufwand (h) Sollzeit	Aufwand (h) Istzeit	Woche 1
					So 16.06.2024: 10 12 15 17 · Mo 17.06.2024: 10 12 15 17 · Di 18.06.2024: 10 12 15 17 · Mi 19.06.2024: 10 12 15 17
	Dokumentation	BB	**8.00**	8.00	
Informieren	Aufbau der Dokumentation	BB	**1.00**	0.50	
Informieren	Grobe Version des Zeitplanes erstellen	BB	**1.00**	0.50	
Informieren	Informationsbeschaffung	BB	**0.50**	0.50	
Informieren	Anforderungsanalyse	BB	**0.50**	0.50	
Planen	Zeitplan fertigstellen	BB	**1.00**	1.00	
Planen	Spezifikation Software (Frontend/Backend/Datenbank)	BB	**0.50**	0.50	
Planen	Spezifikation Software	BB	**0.50**	0.50	
Planen	Mockup erstellen	BB	**2.00**	2.00	
Planen	Test Konzept erstellen	BB	**1.00**	1.00	
Entscheiden	Entscheidung zur Softwarelösung	BB	**0.50**	0.50	
Entscheiden	Entscheidung Mockup	BB	**0.50**	0.50	
Entscheiden	Entscheidung über zusätzliche Features	BB	**0.50**	0.50	
Realisieren	Programmierung des Backends	BB	**2.00**	2.00	
Realisieren	Programmierung des Frontends	BB	**4.00**	5.00	
Realisieren	Datenbank erstellen	BB	**1.00**	0.00	
Realisieren	Backend mit Frontend verbinden	BB	**2.00**	4.00	
Kontrollieren	Testkonzept überprüfen ggf anpassen	BB	**1.00**	1.00	
Kontrollieren	Tests durchführen	RB	**1.00**	1.00	
Kontrollieren	SaveUp App Prototype	BB	**0.50**	0.50	
Auswerten	Überprüfung der Projektziele: Evaluierung des Projekterfolgs anhand der festgelegten Ziele.	BB	**1.00**	1.00	
Auswerten	Reflexion & Fazit	BB	**1.00**	1.00	
Auswerten	Projektabschluss	BB	**1.00**	1.00	
Reserve		BB	**4.00**		
	Total		**36.00**	33.00	

Tabelle 3: Gantt - Planung und Phasen

7.3 Zu verwendende Tools

Für die Entwicklung und Verwaltung des Projekts werden verschiedene Tools verwendet:

GitHub: Dient als zentrales Repository für den Quellcode, ermöglichte Versionskontrolle.
Visual Studio und Visual Studio Code: Diese integrierten Entwicklungsumgebungen (IDEs) werden für die Programmierung und das Debugging des Codes eingesetzt.
MongoDB: Wird verwendet, um die Datenbank zu verwalten, zu konfigurieren und zu testen.
SwaggerUI: Dient zur Dokumentation und zum Testen der RESTful APIs.

7.4 Systemarchitektur

Die Systemarchitektur der SaveUp-App-Projekts ist in drei Schichten aufgeteilt, welche die Kommunikation und den Datenfluss zwischen dem Client-Browser, dem Backend-Server und der Datenbank verdeutlichen.

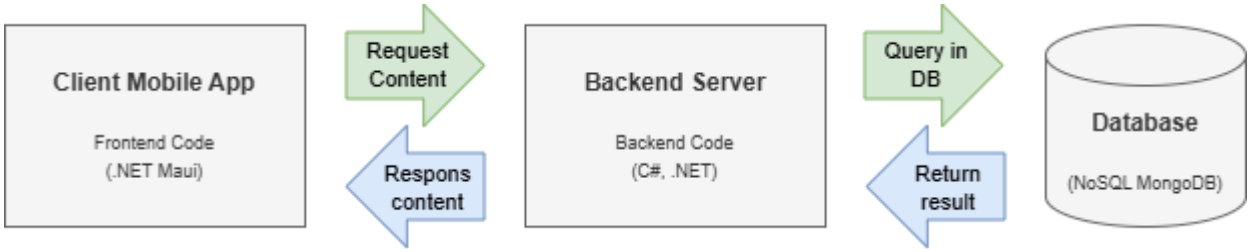


Abbildung 3: Systemarchitektur

- **Client Mobile App:** Die Benutzeroberfläche der SaveUp-App wird mithilfe von .NET MAUI erstellt. Die App läuft auf verschiedenen Plattformen (iOS, Android, Windows) und bietet eine konsistente Benutzererfahrung. Die Oberfläche wird durch XAML-Styles gestaltet, um ein ansprechendes und einheitliches Design sicherzustellen.
- **Backend:** Der Backend-Teil der Anwendung ist ebenfalls in .NET/C# implementiert. Er verarbeitet Benutzeranfragen, speichert und verwaltet die Daten. Die Architektur folgt dem Model-View-View-Model (MVVM) Muster, was eine klare Trennung von Logik und Darstellung ermöglicht.
- **Datenverwaltung:** Die App kann Daten lokal speichern, indem sie Dateien im XML- oder JSON-Format verwendet. Alternativ können Daten in einer Backend-Datenbank gespeichert werden, auf die über eine REST API zugegriffen wird. Dies stellt sicher, dass die Daten persistent und sicher gespeichert werden.

Diese strukturierte Architektur ermöglicht eine effiziente und robuste Verwaltung der Anwendungsdaten und unterstützt die Implementierung der SaveUp-App, entsprechend den festgelegten Anforderungen und Zielen.

7.4.1 Datenbankstruktur

Die Datenbank für die SaveUp-App ist so konzipiert, dass sie effizient die gesparten Beträge und zugehörigen Informationen verwalten kann. Sie verwendet ein einfaches, aber effektives Schema, das die Integrität und Sicherheit der Daten gewährleistet.

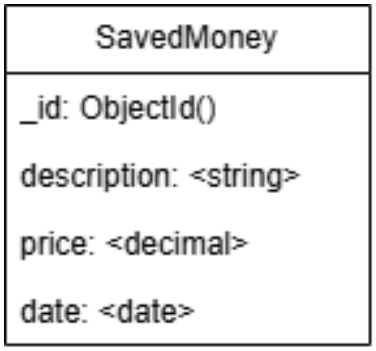


Abbildung 4: Datenbankdiagramm

7.4.2 Web-API

Die Web-API ist das Herzstück des Backends und ermöglicht eine nahtlose Kommunikation zwischen dem Frontend und der Datenbank.

7.5 Technologien

Die Auswahl der Technologien für die Entwicklung der SaveUp-App erfolgt mit dem Ziel, ein robustes und skalierbares System zu schaffen, das die spezifischen Anforderungen des Projekts erfüllt. Die gewählten Technologien umfassen:

- **Frontend-Entwicklung: .NET MAUI**
  - **Beschreibung:** .NET MAUI (Multi-platform App UI) wird verwendet, um eine plattformübergreifende Benutzeroberfläche zu erstellen. Es ermöglicht die Entwicklung einer einzigen Codebasis für Anwendungen, die auf Android, iOS und Windows laufen.
  - **Vorteile:** Konsistentes UI-Design, Wiederverwendbarkeit des Codes, Unterstützung für XAML-Styles zur Gestaltung der Benutzeroberfläche.
- **Backend-Entwicklung: .NET Core**
  - **Beschreibung:** .NET Core wird für die serverseitige Programmierung eingesetzt. Es bietet eine robuste und leistungsstarke Plattform für die Entwicklung von Backend-Services.
  - **Vorteile:** Hohe Performance, Sicherheit, Flexibilität bei der Integration verschiedener Dienste und eine breite Unterstützung für moderne Entwicklungspraktiken.
- **Datenbanktechnologie: NoSQL (MongoDB)**
  - **Beschreibung:** MongoDB wird als NoSQL-Datenbankmanagementsystem verwendet, um die gesammelten Daten effizient zu speichern und abzurufen. MongoDB bietet Flexibilität bei der Speicherung und Abfrage von Daten sowie eine einfache Integration in moderne Anwendungen.
  - **Vorteile:** Hohe Skalierbarkeit, flexible Datenmodelle, Unterstützung für umfangreiche Abfragen und Aggregationen, einfache Integration mit .NET Core.

8 Entscheiden

In der Entscheidungsphase der Entwicklung der SaveUp-App wurden strategische Entscheidungen getroffen, um die Ziele des Projekts zu erreichen und eine effiziente, sichere und benutzerfreundliche Anwendung zu gewährleisten. Diese Entscheidungen basieren auf den gesammelten Informationen und den definierten Anforderungen.

Technologie-Stack:

- **Frontend-Entwicklung:** .NET MAUI wurde aufgrund seiner Fähigkeit zur plattformübergreifenden Entwicklung und seiner Unterstützung für XAML-Styles ausgewählt.
- **Datenbanktechnologie:** MongoDB NoSQL wurde aufgrund seiner Flexibilität, Leistungsfähigkeit und Skalierbarkeit ausgewählt.

Architektur-Entscheidungen:

- **Modulare Architektur:** Zur Gewährleistung von Flexibilität und Skalierbarkeit wurde eine modulare Architektur gewählt, die die unabhängige Entwicklung und Wartung von Systemkomponenten ermöglicht.
- **API-Strategie:** Entwicklung von RESTful-APIs zur standardisierten und plattformunabhängigen Kommunikation zwischen Frontend und Backend.

Auswahl der Entwicklungswerkzeuge:

- **Entwicklungsumgebungen:** Visual Studio und Visual Studio Code wurden aufgrund ihrer umfangreichen Funktionen und Unterstützung für .NET MAUI als primäre Entwicklungswerkzeuge gewählt.
- **Versionskontrolle:** GitHub wird als zentrales Tool für Versionskontrolle verwendet.

Erstellung von Mockups (siehe Anhang):

- **Hauptseite:** Anzeige der Gesamtersparnisse und der heutigen Einsparungen, sowie Optionen zur Navigation und zum Hinzufügen von Artikeln.
- **Artikelliste:** Darstellung aller gespeicherten Artikel mit Datum und Preis.
- **Artikel hinzufügen:** Formular zur Eingabe von Beschreibung, Preis und Datum des gesparten Artikels.

Diese Entscheidungen bilden die Basis für die weitere Umsetzung und Entwicklung der SaveUp-App und sorgen dafür, dass die Anwendung den technischen und geschäftlichen Anforderungen gerecht wird.

9 Realisieren

Die Realisierungsphase setzt die geplanten und entschiedenen Architektur- und Anforderungsdetails in eine funktionierende Lösung um. Während dieser Phase wird das Frontend-System entwickelt und die zugehörigen Tests erstellt.

9.1 Frontend-Entwicklung

Die Frontend-Entwicklung ist eine zentrale Säule des Gesamtsystems. Ziel ist es, eine benutzerfreundliche und ansprechende Benutzeroberfläche zu schaffen, die eine einfache Erfassung und Verwaltung der gesparten Beträge ermöglicht.

Aufbau der Frontend-Infrastruktur:

- **Einrichtung und Konfiguration:** Einrichtung der .NET MAUI Umgebung zur plattformübergreifenden App-Entwicklung.
- **GUI-Design:** Implementierung der Benutzeroberfläche anhand der erstellten Mockups und XAML-Styles (siehe Anhang).

Projektstruktur:

- **Pages:** Implementierung der Hauptseite, Artikelliste und Seite zum Hinzufügen von Artikeln.
- **ViewModels:** Anwendung des Model View ViewModel (MVVM) Musters zur Trennung von Logik und Darstellung.
- **Models:** Definition der Datenmodelle für die gesparten Artikel (Beschreibung, Preis, Datum).

Mockups:

- **Hauptseite:** Anzeige der Gesamtersparnisse und der heutigen Einsparungen, sowie Optionen zur Navigation und zum Hinzufügen von Artikeln.
- **Artikelliste:** Darstellung aller gespeicherten Artikel mit Datum und Preis.
- **Artikel hinzufügen:** Formular zur Eingabe von Beschreibung, Preis und Datum des gesparten Artikels.

API-Integration:

- **Implementierung von API-Calls:** Anbindung an das Backend zur Speicherung und Abruf der gesparten Artikel.

9.2 Sicherheitsimplementierung

Ein umfassendes Sicherheitskonzept wird implementiert, um die Integrität und den Schutz der Daten zu gewährleisten:

- **Einsatz von HTTPS:** Sicherstellung der sicheren Kommunikation.

9.3 Testverfahren und Testplan

Eine Teststrategie wird etabliert, um die Zuverlässigkeit des Frontends zu gewährleisten:

- **Benutzertests:** Sicherstellung der Benutzerfreundlichkeit und Funktionalität durch Endnutzer-Tests.

Testfall Nr.	Beschreibung des Tests	Erwartetes Ergebnis	Ergebnis
1	Starten der App	App startet ohne Fehler und zeigt die Hauptseite an	
2	Hinzufügen eines Artikels	Artikel wird korrekt mit Beschreibung, Preis und Datum gespeichert	
3	Anzeige der Artikelliste	Gespeicherte Artikel werden in der Liste korrekt angezeigt	

4	Navigation zwischen Seiten	Navigation funktioniert reibungslos zwischen Hauptseite, Artikelliste und Artikel hinzufügen Seite	
5	Validierung der Eingabedaten (Beschreibung)	Fehler wird angezeigt, wenn die Beschreibung leer ist	
6	Validierung der Eingabedaten (Preis)	Fehler wird angezeigt, wenn der Preis kein gültiges Dezimalformat hat	
7	Validierung der Eingabedaten (Datum)	Fehler wird angezeigt, wenn kein Datum ausgewählt wurde	
8	Anzeige der Gesamtersparnisse	Gesamtersparnisse werden korrekt auf der Listseite angezeigt	
9	Benutzerfreundlichkeit	Benutzer kann alle Funktionen intuitiv nutzen	

9.4 Dokumentation und Wartbarkeit

Die Dokumentation und Wartung des Systems werden mit folgenden Werkzeugen unterstützt:

- **SwaggerUI:** Detaillierte Dokumentation der API-Endpunkte.
- **GitHub:** Versionskontrolle und Kollaboration.

Die Realisierung des Frontend-Systems wird iterativ durchgeführt, wobei kontinuierliches Feedback aus dem Testing und von den Stakeholdern zur stetigen Verbesserung und Anpassung des Systems beigetragen hat.

10 Kontrollieren

In dieser Phase werden umfangreiche Tests durchgeführt, um die Funktionalität, Sicherheit und Stabilität der entwickelten SaveUp-App zu gewährleisten. Verschiedene Validierungstests werden etabliert, um sicherzustellen, dass die Anforderungen erfüllt werden und die App entsprechend den Spezifikationen funktioniert:

11 10.1 Usability Tests durchführen

Die Usability-Tests zielen darauf ab, die Benutzerfreundlichkeit und Funktionalität der SaveUp-App zu bewerten. Die folgenden Testfälle wurden definiert, um verschiedene Aspekte der App zu überprüfen:

Testfall Nr.	Beschreibung des Tests	Erwartetes Ergebnis	Ergebnis
1	Starten der App	App startet ohne Fehler und zeigt die Hauptseite an	Ok
2	Hinzufügen eines Artikels	Artikel wird korrekt mit Beschreibung, Preis und Datum gespeichert	Ok
3	Anzeige der Artikelliste	Gespeicherte Artikel werden in der Liste korrekt angezeigt	Ok
4	Navigation zwischen Seiten	Navigation funktioniert reibungslos zwischen Hauptseite, Artikelliste und Artikel hinzufügen Seite	Ok
5	Validierung der Eingabedaten (Beschreibung)	Fehler wird angezeigt, wenn die Beschreibung leer ist	Ok
6	Validierung der Eingabedaten (Preis)	Fehler wird angezeigt, wenn der Preis kein gültiges Dezimalformat hat	Ok
7	Validierung der Eingabedaten (Datum)	Fehler wird angezeigt, wenn kein Datum ausgewählt wurde	Ok
8	Anzeige der Gesamtersparnisse	Gesamtersparnisse werden korrekt auf der Listseite angezeigt	Ok
9	Benutzerfreundlichkeit	Benutzer kann alle Funktionen intuitiv nutzen	Ok

12 Auswerten

Die Auswertungsphase dient dazu, nach Fertigstellung des Projekts eine umfassende Bewertung der verschiedenen Aspekte der SaveUp-App vorzunehmen. Die Analyse befasst sich mit der Funktionalität, der technischen Umsetzung, der Effizienz der Entwicklung sowie den erreichten Geschäftszielen.

1. **Funktionalität und Technische Umsetzung:**
    - Die in der App implementierten Funktionen und Benutzeroberflächen wurden eingehend getestet, insbesondere hinsichtlich ihrer Interaktion mit dem Backend.
    - Die Datenverarbeitung und -speicherung, insbesondere die Handhabung der gesparten Artikel, wurden entsprechend den Anforderungen verifiziert.
    - Die GUI wurde basierend auf den Mockups umgesetzt und auf Benutzerfreundlichkeit getestet.
2. **Systemleistung:**
    - Die App wurde auf ihre Leistungsfähigkeit hin ausgewertet, mit einem besonderen Fokus auf die Reaktionszeiten und die Zuverlässigkeit unter realen Bedingungen.
3. **Effizienz der Entwicklung:**
    - Die Entscheidungen für die verwendeten Technologien und Frameworks, wie .NET MAUI und MongoDB, sowie die Projektstruktur, wurden hinsichtlich ihrer Auswirkungen auf die Entwicklungsgeschwindigkeit und -qualität bewertet.
    - Die Verwendung von MVVM zur Trennung von Logik und Darstellung trug zur Effizienz der Entwicklung bei.
4. **Erreichte Geschäftsziele:**
    - Die App unterstützt nun effektiv das Verfolgen und Verwalten der gesparten Beträge, was zu einer verbesserten finanziellen Übersicht für die Nutzer führt.
    - Die Datenerfassung in der App ermöglicht es, zukünftige Ausgaben besser zu planen und zielgerichtet zu sparen.
5. **Kunden- und Nutzerfeedback:**
    - Rückmeldungen von Endnutzern und Stakeholdern wurden gesammelt und analysiert, um die Benutzerfreundlichkeit und Zufriedenheit mit den App-Funktionen zu bewerten.
    - Dieses Feedback wurde als Grundlage für zukünftige Verbesserungen und Funktionsupdates herangezogen.

Durch diese umfassende Auswertung konnten wichtige Erkenntnisse gewonnen werden, die zur kontinuierlichen Verbesserung der SaveUp-App beitragen und sicherstellen, dass die Anwendung den Bedürfnissen der Nutzer gerecht wird.

12.1 Fazit

Die Entwicklung der SaveUp-App war für mich ein bedeutender Schritt in der Umsetzung einer nützlichen und benutzerfreundlichen Anwendung zur Verwaltung von Ersparnissen durch den Verzicht auf kleine Ausgaben. Ich habe moderne Technologien und bewährte Methoden angewendet, um den spezifischen Anforderungen der Nutzer gerecht zu werden.

Die zentrale Herausforderung bestand darin, eine plattformübergreifende Anwendung mit .NET MAUI zu erstellen, die sowohl auf Android- als auch auf iOS-Geräten ein konsistentes Benutzererlebnis bietet. Durch die Anwendung des MVVM-Musters konnte ich eine klare Trennung von Geschäftslogik und Benutzeroberfläche erreichen, was die Wartbarkeit und Erweiterbarkeit der App erheblich erleichtert. Die Anbindung des Backends bereitete Schwierigkeiten, da das Verständnis fehlte und wenig Vorkenntnisse vorhanden waren.

Ein wesentlicher Erfolgsfaktor war die Integration von MongoDB als NoSQL-Datenbank. Diese Wahl bot die notwendige Flexibilität und Skalierbarkeit, um die erfassten Daten effizient zu verwalten und Abfragen schnell zu verarbeiten. Dies war entscheidend, um den Nutzern jederzeit eine aktuelle und zuverlässige Übersicht über ihre Ersparnisse zu bieten.

Die erfolgreiche Umsetzung der App zeigte mir die Bedeutung einer sorgfältigen Planung und der richtigen Technologieauswahl. Die SaveUp-App demonstriert technische Kompetenz und ein tiefes Verständnis für die Anwendung moderner Entwicklungswerkzeuge und -methoden. Sie bietet eine solide Grundlage für zukünftige Erweiterungen und Anpassungen, um den sich wandelnden Bedürfnissen der Nutzer gerecht zu werden.

13 Anhänge

13.1 Voraussetzungen für die Funktionalität des Backends und Frontends (siehe auch readme)

Um sicherzustellen, dass die SaveUp-App und das Backend ordnungsgemäß funktionieren, müssen die folgenden Voraussetzungen erfüllt sein:

1. **Emulator-Setup:**
   - Die App läuft im Emulator, daher muss der Emulator korrekt installiert und konfiguriert sein.
2. **Backend-Integration:**
   - Das Backend ist im selben Projekt wie die App integriert. Stellen Sie sicher, dass das Backend-Projekt erfolgreich kompiliert und gestartet werden kann.
3. **MongoDB Installation:**
   - MongoDB muss auf dem System installiert sein, auf dem die App ausgeführt wird.
   - Der Verbindungsstring (Connection String) zu MongoDB muss in der appsettings.json der App angepasst werden, um eine erfolgreiche Verbindung zur Datenbank herzustellen.
4. **Syncfusion Lizenz:**
   - Für den DatePicker in der App wird ein Lizenzschlüssel von Syncfusion benötigt.
   - Der Lizenzschlüssel muss in einer Datei namens syncfusion_license.txt im Resources-Ordner des Projekts gespeichert werden.

Durch die Erfüllung dieser Voraussetzungen wird sichergestellt, dass die SaveUp-App und das Backend ordnungsgemäß funktionieren und eine nahtlose Benutzererfahrung bieten.

13.2 Glossar

Begriff	Definition
.NET MAUI	Eine plattformübergreifende Entwicklungsplattform für mobile und Desktop-Anwendungen.
API	Application Programming Interface, eine Schnittstelle zur Kommunikation zwischen verschiedenen Softwarekomponenten.
Backend	Der serverseitige Teil einer Anwendung, der die Datenverarbeitung und -speicherung übernimmt.
Content Pages	Seiten innerhalb einer Anwendung, die Inhalte und Funktionen präsentieren.
Datenpersistenz	Die dauerhafte Speicherung von Daten, sodass diese auch nach einem Neustart der Anwendung verfügbar bleiben.
Frontend	Der clientseitige Teil einer Anwendung, der die Benutzeroberfläche darstellt und mit dem Benutzer interagiert.
GitHub	Eine Plattform für Versionskontrolle und kollaborative Softwareentwicklung.
GUI	Graphical User Interface, die grafische Benutzeroberfläche einer Anwendung.
JSON	JavaScript Object Notation, ein leichtgewichtiges Datenformat zur Datenübertragung.
MongoDB	Ein NoSQL-Datenbankmanagementsystem, das flexible, dokumentenbasierte Datenbanken ermöglicht.
MVVM	Model View ViewModel, ein Entwurfsmuster zur Trennung von Logik und Benutzeroberfläche in Softwareanwendungen.
REST API	Representational State Transfer, ein Architekturstil für Netzwerkkommunikation, der auf HTTP-Protokollen basiert.
SwaggerUI	Ein Tool zur Dokumentation und Visualisierung von RESTful APIs.
Syncfusion	Ein Anbieter von Softwarekomponenten für die Entwicklung von Benutzeroberflächen, darunter auch Lizenzierte DatePicker.
Versionskontrolle	Ein System zur Verwaltung von Änderungen an Dokumenten, Programmen und anderen Informationsquellen.
Visual Studio	Eine integrierte Entwicklungsumgebung (IDE) von Microsoft für die Entwicklung von Software.
Visual Studio Code	Ein kostenloser Quelltext-Editor von Microsoft mit Unterstützung für verschiedene Programmiersprachen.
XAML	Extensible Application Markup Language, eine Markup-Sprache zur Gestaltung von Benutzeroberflächen.
XML	Extensible Markup Language, ein Markup-Datenformat zur Strukturierung und Speicherung von Daten.

Tabelle 4: Glossar

13.3 Mock – ups

Das dritte bzw. immer das Mockup ganz rechts ist das, welches umgesetzt wurde.

Abbildung 5: Mock-up - MainPage

Abbildung 6: Mock-up - ListPage

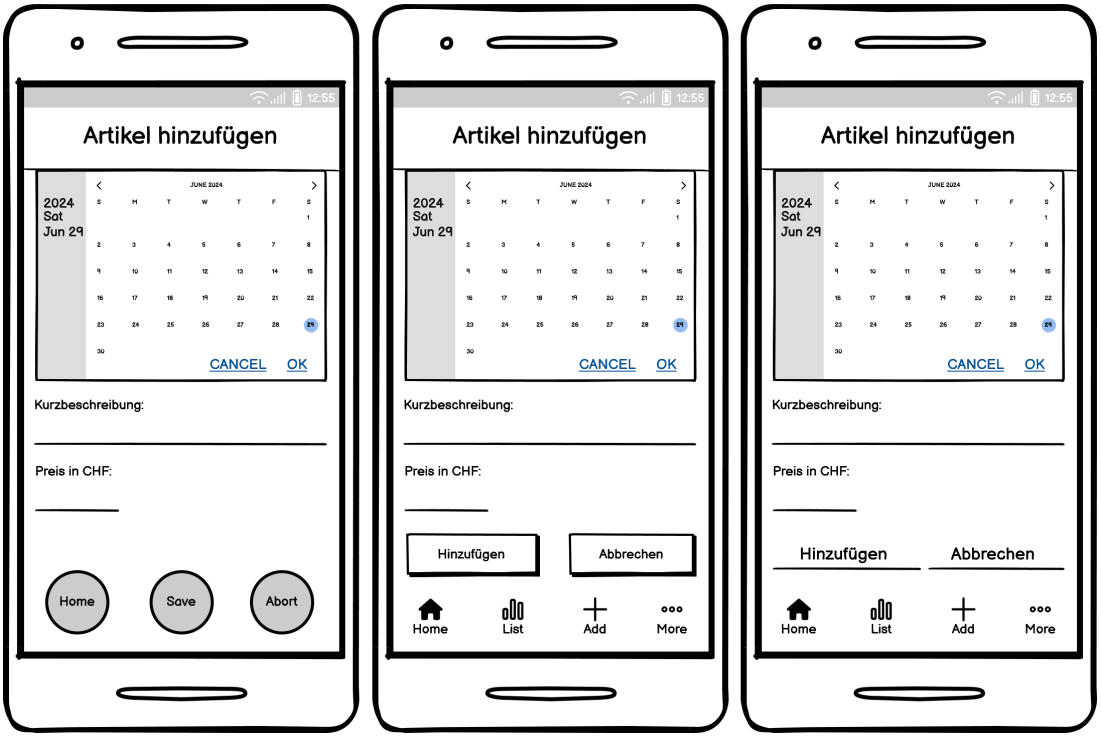


Abbildung 7: Mock-up - AddPage

13.4 App Icon und Splash screen

Abbildung 9: App icon

Abbildung 8: Splashscreen

13.5 Umgesetzte App - Pages

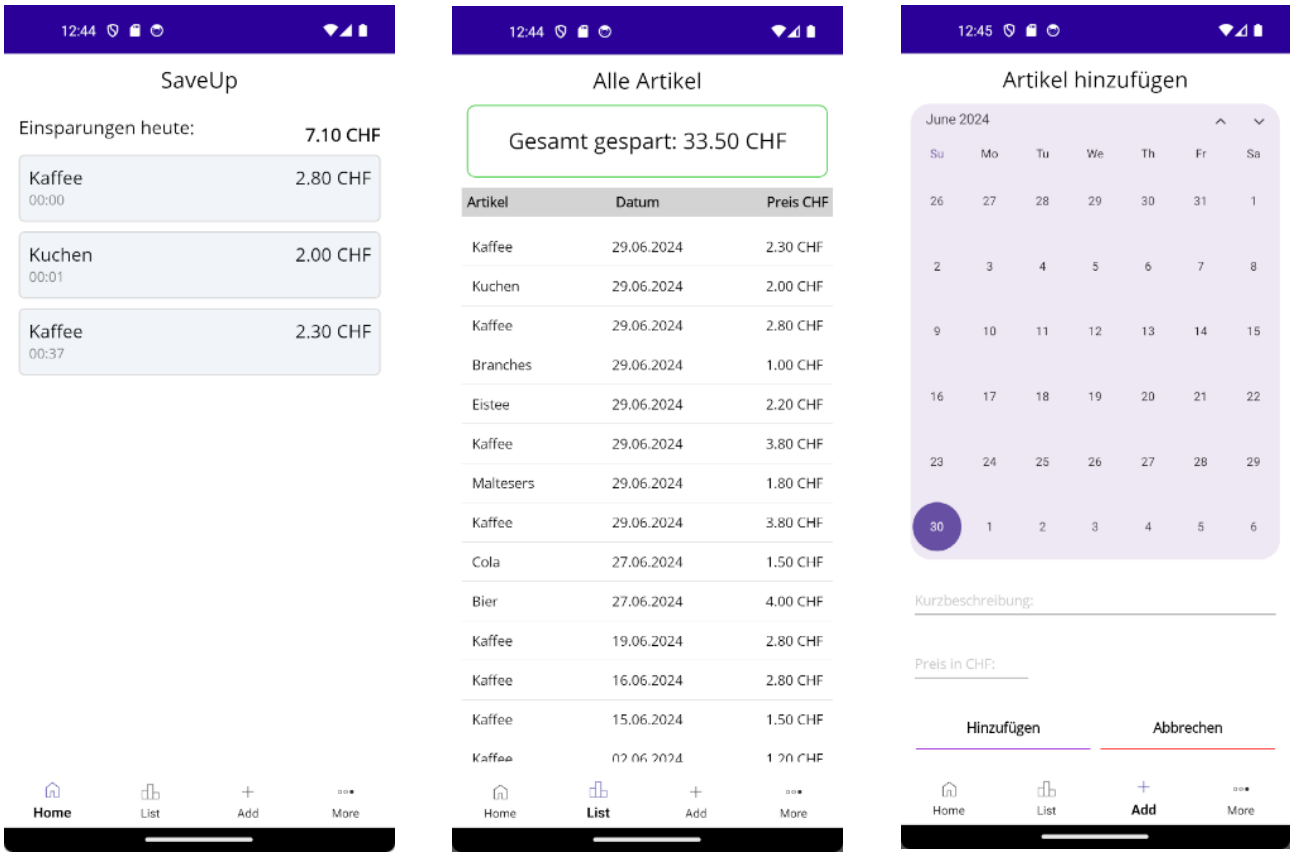


Abbildung 10: Umgesetzte App in .NET Maui

13.6 NuGet Packages Screenshot

```
<Project Sdk="Microsoft.NET.Sdk">

    <PropertyGroup>
        <EmbedAssembliesIntoApk>true</EmbedAssembliesIntoApk>
        <TargetFrameworks>net8.0-android;net8.0-ios;net8.0-maccatalyst</TargetFrameworks>
        <TargetFrameworks Condition="$([MSBuild]::IsOSPlatform('windows'))">$(TargetFrameworks);net8.0-windows10.0.19041.0</TargetFrameworks>
        <!-- Uncomment to also build the tizen app. You will need to install tizen by following this: https://github.com/Samsung/Tizen.NET -->
        <!-- <TargetFrameworks>$(TargetFrameworks);net8.0-tizen</TargetFrameworks> -->

  <ItemGroup>
      <PackageReference Include="Microsoft.Maui.Controls" Version="$(MauiVersion)" />
      <PackageReference Include="Microsoft.Maui.Controls.Compatibility" Version="$(MauiVersion)" />
      <PackageReference Include="Microsoft.Extensions.Logging.Debug" Version="8.0.0" />
      <PackageReference Include="Newtonsoft.Json" Version="13.0.3" />
      <PackageReference Include="SkiaSharp.Svg" Version="1.60.0" />
      <PackageReference Include="Svg" Version="3.4.7" />
      <PackageReference Include="Syncfusion.Maui.Calendar" Version="26.1.35" />
  </ItemGroup>
```

Abbildung 11: NuGet Packages Screenshot

13.7 Swagger Screenshots

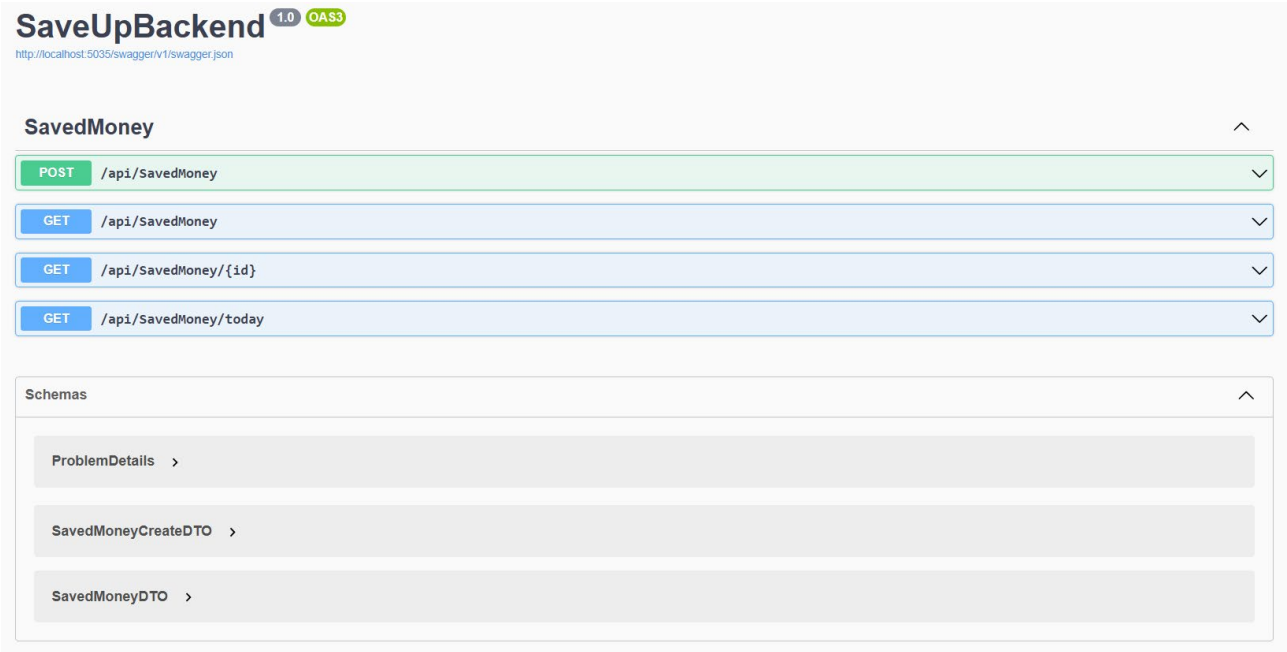


Abbildung 12: SwaggerUI - API Endpoints und Schemas

13.8 Verweise

13.8.1 Quellenverzeichnis

Microsoft. (19. Juni 2024). *.NET MAUI documentation*. Von Microsoft Learn: https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-8.0 abgerufen

13.8.2 Tabellenverzeichnis

13.8.3 Abbildungsverzeichnis