12.3.2023

Projektarbeit M335 SaveUp App

SaveUp App

Raphael Hug

Inhalt

1. Informieren

1.1. Einleitung

Ich möchtest Geld sparen für eine grössere private Investition, wie zum Beispiel für die Ferien. Aus diesem Grund hast du dich entschieden, auf übliche kleine Ausgaben wie Kaffee, Süssigkeiten usw. zu verzichten. Alle gesparten Kleineinkäufe möchtest du in einer App festhalten, damit du dich laufend über den angesparten Geldbetrag informieren kannst.

Durch den Verzicht auf kleine Ausgaben kannst du schnell und einfach Geld sparen, dass du dann für grössere und wichtigere Investitionen nutzen kannst. Das Festhalten aller gesparten Beträge in einer App ermöglicht es dir, jederzeit und überall auf den aktuellen Stand deines Sparziels zuzugreifen und deine Fortschritte zu verfolgen.

Durch das Verwenden einer App, um deine Ersparnisse zu verfolgen, kannst du ausserdem deine Ausgabenmuster analysieren und feststellen, wo du noch weitere Einsparungen vornehmen kannst. Dies hilft dir, deine Ziele effektiver zu erreichen und deine finanziellen Ziele zu erreichen.

Insgesamt denke ich, dass die Verwendung einer Spar-App eine effektive Methode ist, um gezieltes Sparen zu fördern und den Überblick über deine Finanzen zu behalten. Du bist zuversichtlich, dass du durch den Verzicht auf kleine Ausgaben und das Festhalten deiner Ersparnisse in der App dein Sparziel erreichen und eine finanzielle Stabilität erreichen wirst.

1.2. Vorgaben

Das Projekt befasst sich mit der Entwicklung einer innovativen Xamarin.Forms oder Net Maui Anwendung, die es dem Benutzer ermöglicht, seine gesparten Verzichtprodukte effizient und übersichtlich zu verwalten. Durch die Verwendung von zwei Content Pages kann der Anwender eine Kurzbeschreibung und den Preis jedes einzelnen Artikels erfassen und speichern.

Die App ist darauf ausgelegt, eine intuitive und benutzerfreundliche Umgebung zur Verfügung zu stellen, in der der Benutzer seine gesparten Verzichtprodukte einfach und effektiv verwalten kann. Die gespeicherten Daten werden in einer übersichtlichen Liste angezeigt, die neben der Beschreibung und dem Preis auch die Gesamtsumme aller gesparten Artikel enthält.

Dieses Projekt bietet eine großartige Möglichkeit, die Verwaltung von Verzichtprodukten zu vereinfachen und zu optimieren. Durch die Verwendung moderner Technologien und einer benutzerfreundlichen Oberfläche wird eine Lösung bereitgestellt, die es dem Benutzer ermöglicht, seine Sparziele zu erreichen und seine Ausgaben im Blick zu behalten.

1.3. Anforderungen

- Die App muss SaveUp heißen und ein eigenes App-Icon haben.
- Die App muss mindestens aus 2 Content Pages bestehen.
- Das GUI-Design der App, einschließlich der Mock-ups, muss umgesetzt werden.
- Der Benutzer muss die Möglichkeit haben, eine Kurzbeschreibung und den Preis des gesparten Artikels zu erfassen (mindestens 2 Eingaben).
- Die App muss zwei Menüfunktionen (Action) zur Speicherung und zum Aufruf der Listendarstellung bieten.
- Die App muss eine einfache bzw. intuitive Bedienung und ein geeignetes Layout (Styles) haben.
- Die Codestrukturierung der App muss dem MVVM-Entwurfsmuster entsprechen.
- Die App muss XAML-Styles der Steuerelemente verwenden.
- Datum/Uhrzeit des Kaufverzichts müssen als zusätzliches Attribut erfasst werden.
- Die App muss umfassend dokumentiert werden.
- Die App muss gründlich getestet werden, um eine fehlerfreie Funktionsweise zu gewährleisten.

1.4. Zusätzliche Anforderungen

- Zur Erreichung der maximalen Punktzahl müssen zwei optionale Anforderungen umgesetzt werden.
- Die erfassten Daten müssen in einer lokalen Datei (XML, JSON) gespeichert werden.
- Die erfassten Daten müssen in einer Backend-Datenbank mit REST-Implementierung gespeichert werden.
- Die App muss die Möglichkeit bieten, erfasste Einträge (Clear) einzeln oder komplett zu löschen.

1.5. Was muss sonst noch wissen?

Ich bin gerade dabei, ein Projekt mit Net Maui zu realisieren und die Anforderungen sind ähnlich wie in meinem aktuellen Projekt. Daher brauche ich keine weiteren spezifischen Informationen.

2. Planen

2.1. Zeitmanagement

Ich habe frühzeitig einen Termin gesetzt, um an meinem eigenen Projekt arbeiten zu können und sicherzustellen, dass ich genügend Zeit habe, um alles abzuschließen. In der Gantt-Darstellung ist deutlich zu erkennen, dass ich mir das Ziel gesetzt habe, das Projekt innerhalb von drei Tagen zu beenden. Ich habe das Gantt-Diagramm als Teil meiner Abgabe hinzugefügt, um den Leser mit einer übersichtlichen Darstellung meiner Fortschritte und des Zeitplans zu unterstützen.

Durch das Festlegen eines klaren Zeitrahmens und das Verwenden einer Gantt-Diagramm-Visualisierung habe ich meine Arbeit besser strukturieren und priorisieren können. Das Diagramm zeigt alle Aufgaben und Meilensteine des Projekts sowie die Abhängigkeiten zwischen ihnen auf, was es mir ermöglicht hat, den Fortschritt des Projekts in Echtzeit zu überwachen und bei Bedarf Anpassungen vorzunehmen.

Insgesamt denke ich, dass die Verwendung einer Gantt-Diagramm-Visualisierung eine effektive Methode ist, um Projekte zu planen und zu organisieren. Es gibt mir eine klare Vorstellung davon, was getan werden muss und wann es getan werden muss, und es erleichtert die Kommunikation mit anderen Beteiligten, indem es einen klaren Überblick über den Fortschritt und den Zeitplan des Projekts bietet. Schätzungsweise 17 Stunden werde ich an der Arbeit haben.

2.2. Vorgangsweise

In Bezug auf meine Vorgehensweise bei der Entwicklung der Software habe ich beschlossen, zuerst alle Views zu erstellen, um eine klare Vorstellung davon zu haben, wie die Benutzeroberfläche aussehen wird. Anschließend plane ich, alle Models zu implementieren, um sicherzustellen, dass das Backend reibungslos funktioniert und alle Daten korrekt verarbeitet werden.

Nachdem alle Views und Models fertiggestellt sind, beabsichtige ich, die entsprechenden Viewmodels zu erstellen, um die Verbindung zwischen den Views und Models zu gewährleisten und die Geschäftslogik der Anwendung zu implementieren. Indem ich diese Vorgehensweise befolge, denke ich, dass ich in der Lage sein werde, das Projekt schrittweise voranzutreiben und eine gut strukturierte und effiziente Software zu entwickeln, die den Anforderungen entspricht.

Auf der Nächsten Seite sieht man mein UML-Plan:

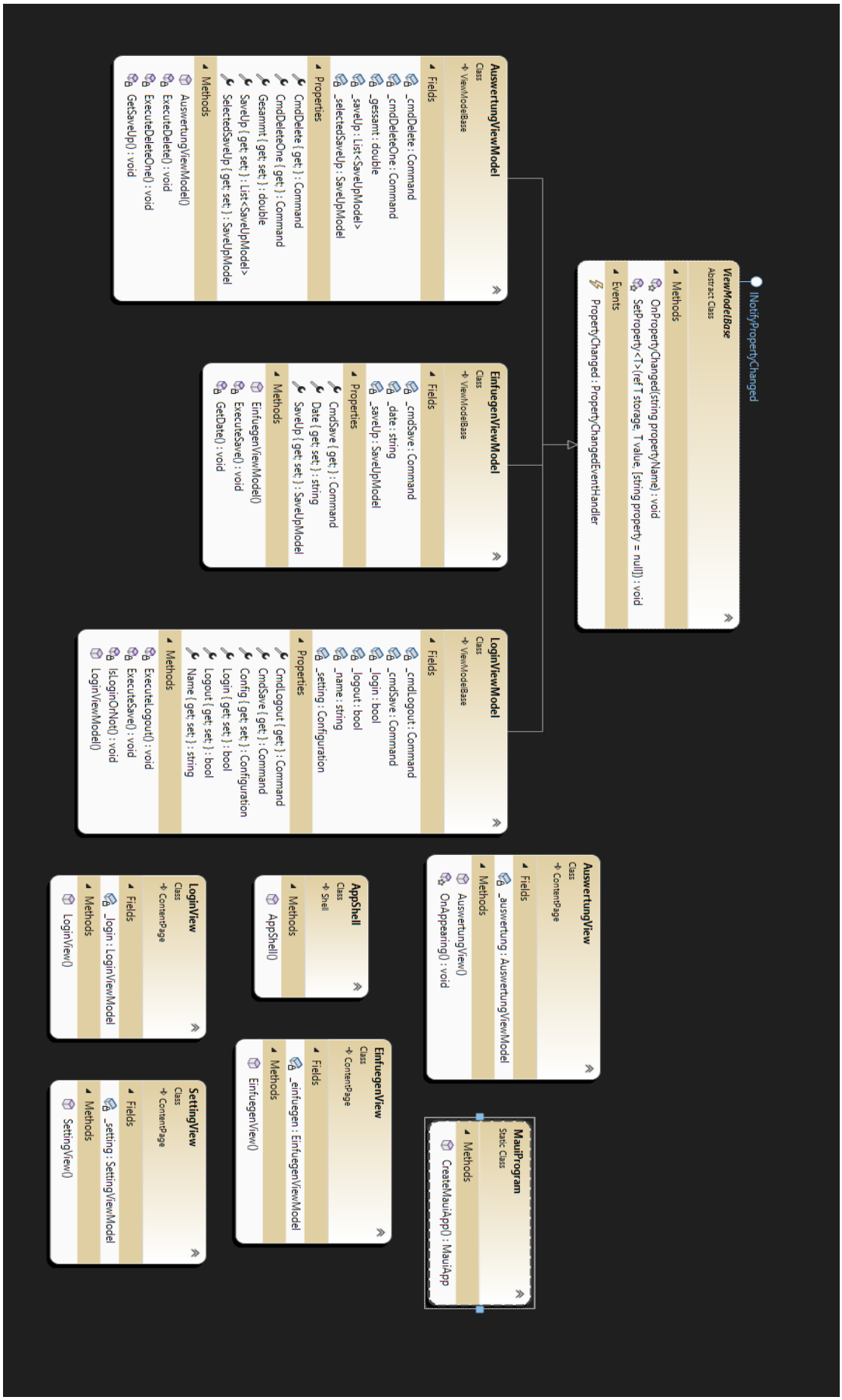

Abbildung 1 UML

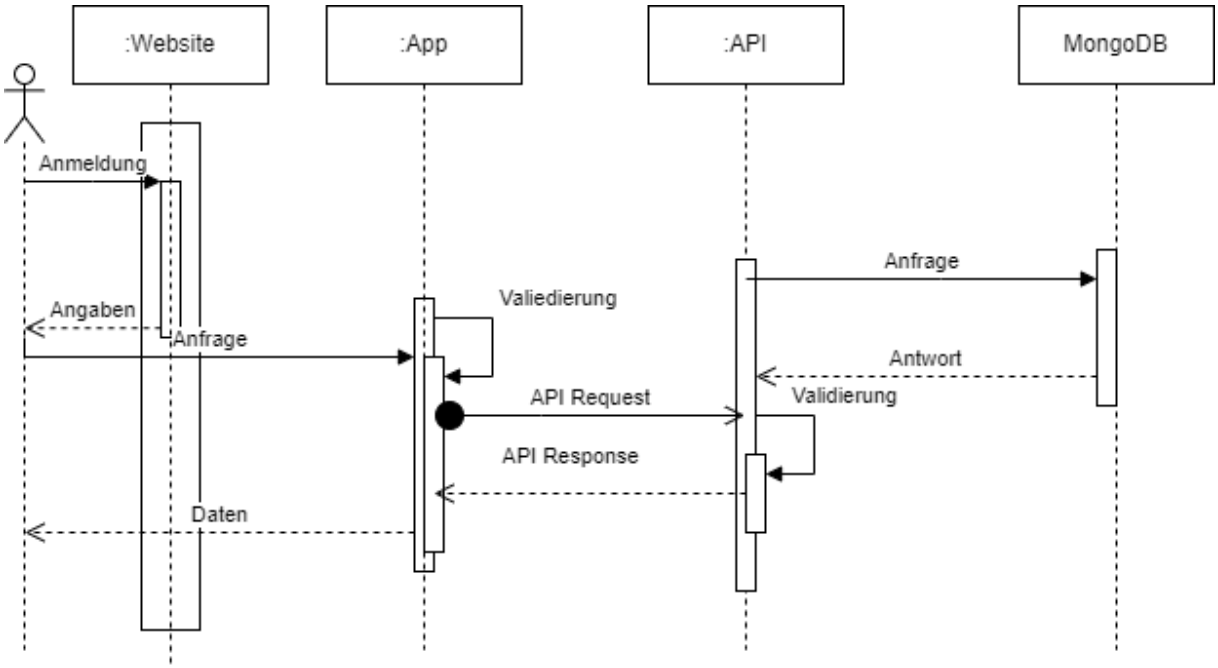

Abbildung 2 Sezuenz

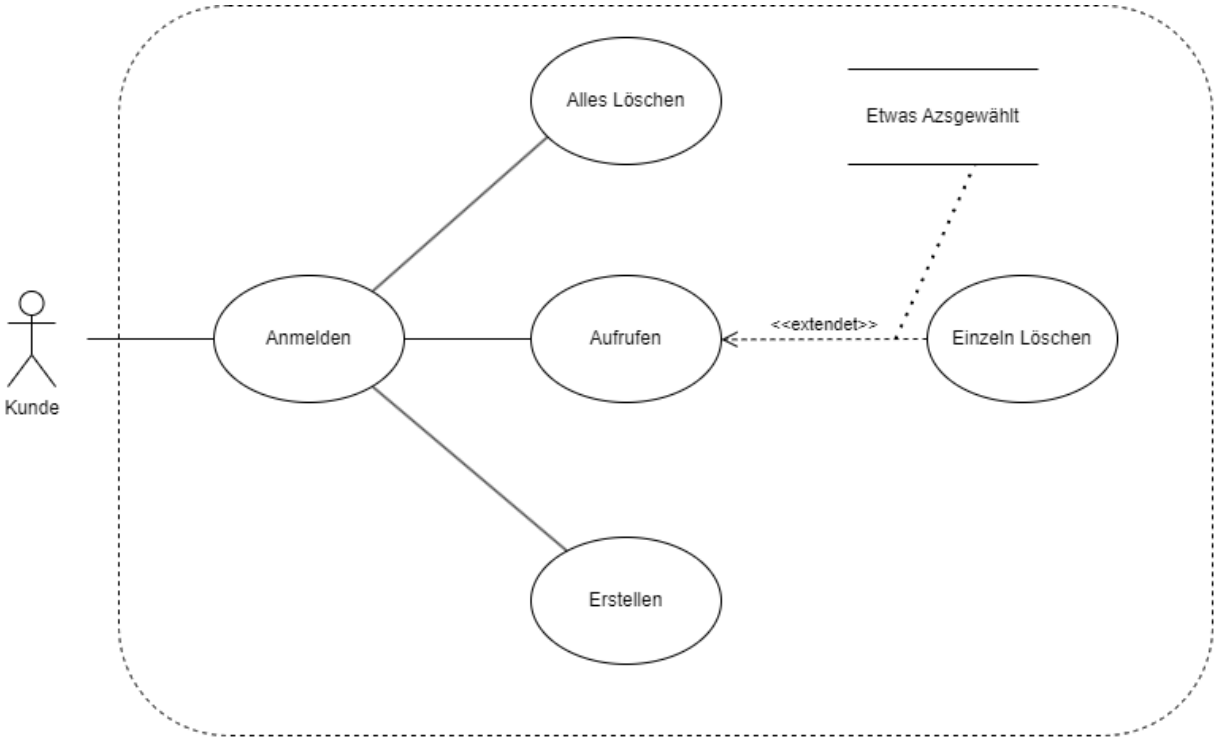

Abbildung 3 UsecAse

PSP

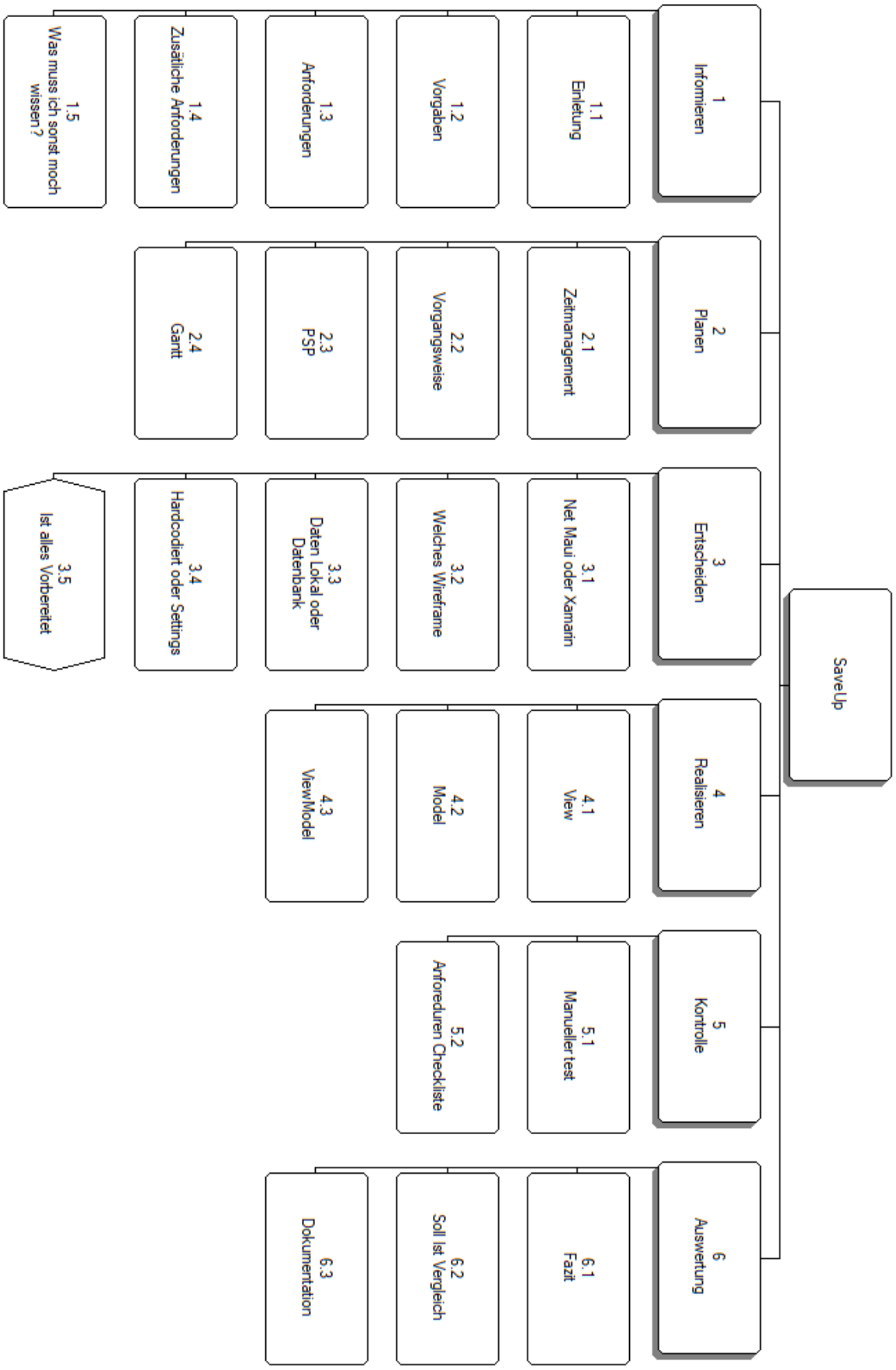

Abbildung 4 PSP

2.3. Gantt

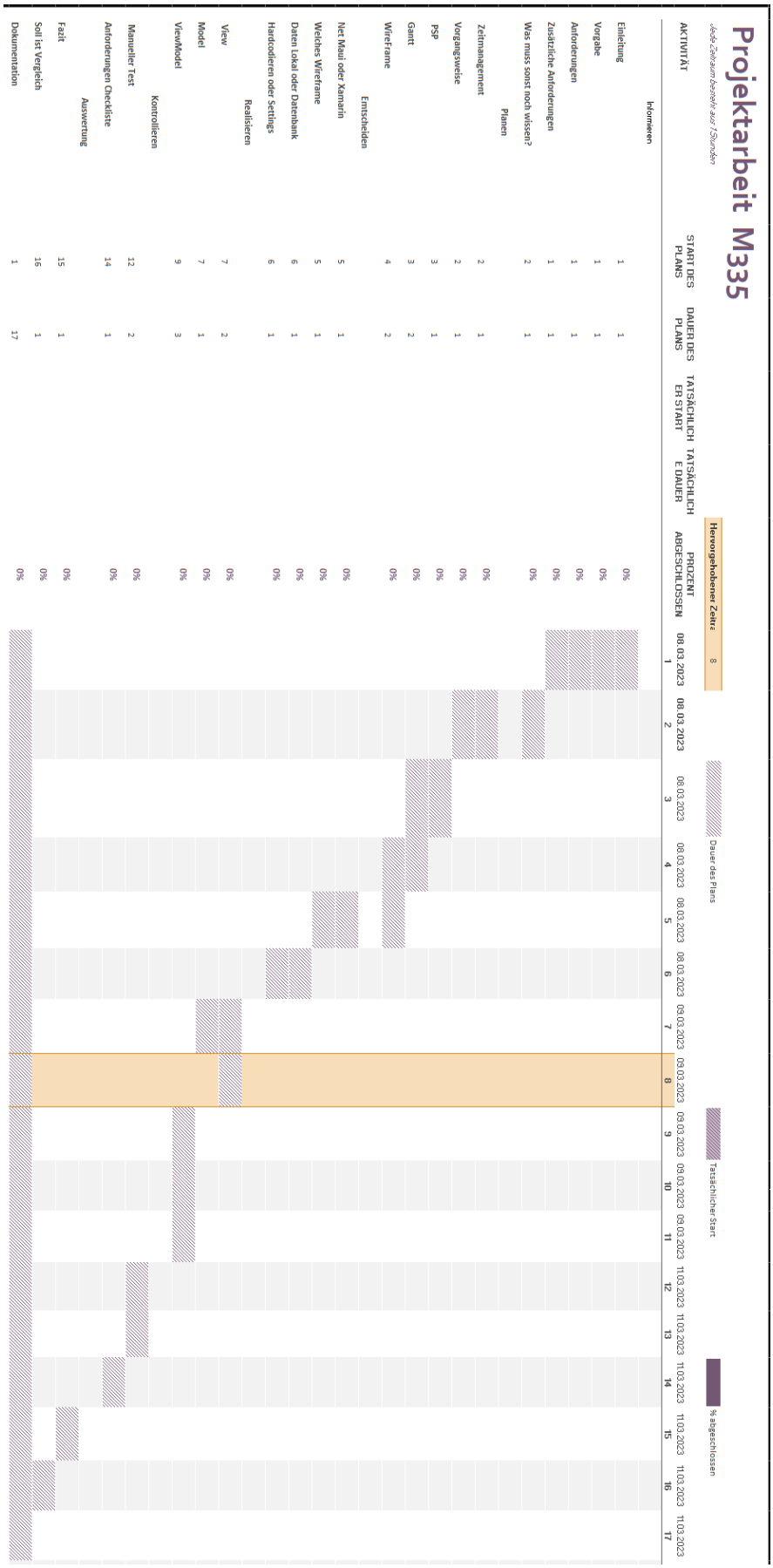

Abbildung 5 Gantt Vorlage

3. Entscheiden

3.1. Net Maui oder Xamarin

Tabelle 1 Net Maui oder Xamarin

Beschreibung:	Faktor	Net Maui	Xamarin
Was kann ich schon kann ich schon?	1	2	0
Ist es Neuwertig?	2	2	0
Native-UI-Unterstützung	3	3	1
Auswertung		15	3

Diese Beschreibung ist kurz, weil Net Maui eine Weiterentwicklung ist von Xamarin und Net Maui hat 15 Punkte dagegen hat Xamarin nur 3.

3.2. Welches Wireframe

Tabelle 2 Wireframe

Beschreibung:	Faktor	Vorlage 1	Vorlage 2
Klicks	2	3	0
Umfrage	3	2	1
Benutzerfreundlichkeit	2	3	1
Auswertung		18	5

Da die meisten Personen die Vorlage 1 gewählt hat und es weniger Klicks und Benutzerfreundlicher war habe ich Vorlage 1 genommen.

3.3. Daten Lokal oder Datenbank

Tabelle 3 Lokal oder Datenbank

Beschreibung:	Faktor	Lokal	Datenbank
Daten immer vorhanden?	3	3	2
Daten auf andere Geräte verfügbar?	2	0	3
Weniger Aufwand	3	3	2
Auswertung		15	128

Ich nehme die Datenbank da ich schon eine API gemacht habe wie lauft ist der Aufwand nicht sehr gross und da wir heute überall Internet haben Mobile daten oder W-Lan spielt das keine grosse Rolle die Daten auf der Datenbank zu haben.

3.4. Hartcodieren oder Settings

Tabelle 4 Setting

Beschreibung:	Faktor	Hart	Setting 2
Wiederverwendbarkeit	3	0	3
Schneller	1	2	0
Auswertung		4	9

Es ist schneller Hartcodieren, aber so sollte es nicht machen und es ist die App so nicht wiederverwendbar.

3.5. Wireframe Vorlage 1

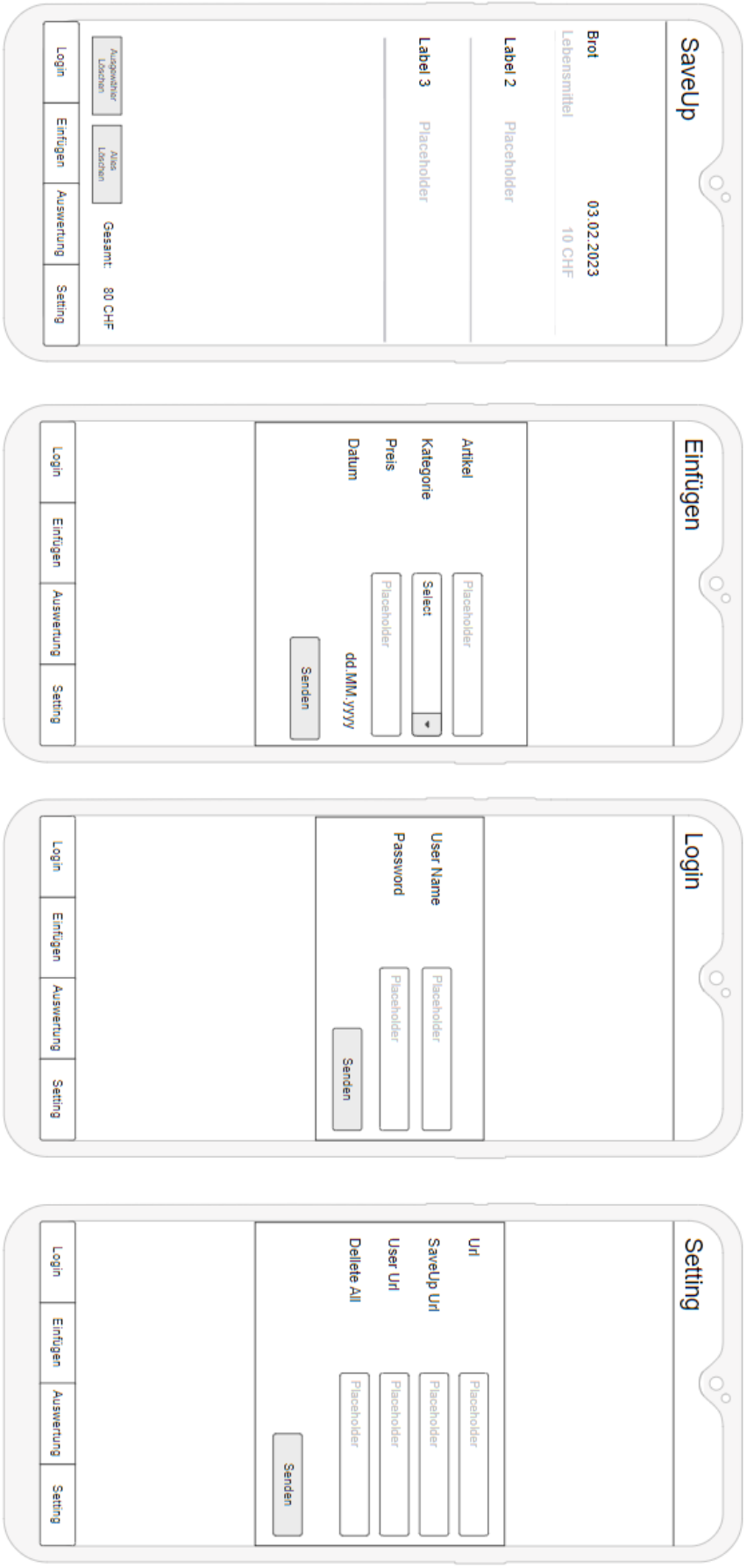

Abbildung 6 Wireframe Vorlage1

3.6. Wireframe Vorlage 2

Abbildung 7 Wireframe Vorlage2

4. Realisiere

4.1. Voraussetzung

Betriebssystem: Der Computer läuft unter Windows 11 Version 22H2.
Net 7: Die Anwendung erfordert Net 7, da es nicht auf Zertifikate prüft, wenn eine HTTPS-Verbindung hergestellt wird.
Entwicklungsumgebung: Die Version 17.5.1 von Visual Studio wird verwendet.
Android: Die App wird für die Version 13 des Android-Betriebssystems mit der API-Version 33 entwickelt.
Abhängigkeiten: Die Anwendung verwendet Newtonsoft.Json in Version 13.0.3.

Leider scheint es so, dass das Zertifikatsproblem aufgrund meines Raspberry Pi auftritt. Ich habe jedoch einen Intel NUC Mini-Computer gekauft, der noch nicht angekommen ist. Sobald ich ihn habe, werde ich einen virtuellen Rechner einrichten, auf dem ich Windows installieren kann. Ich vermute, dass das Zertifikatsproblem dann nicht mehr auftreten wird.

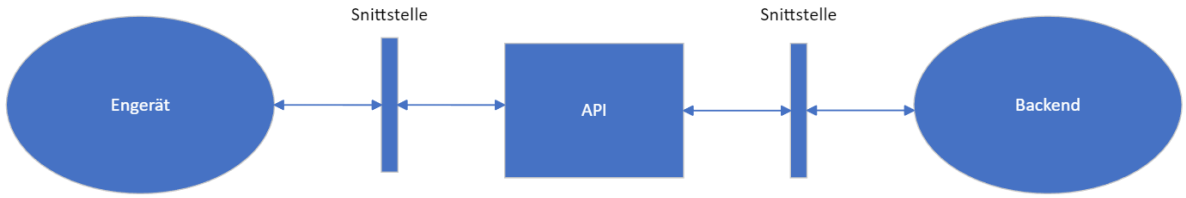

Abbildung 8 Schnittstellen

4.2. View

Zu Beginn meines Entwicklungsprojekts habe ich mich auf die Erstellung der Views konzentriert. Als erstes habe ich das Login-View entworfen und dafür gesorgt, dass Benutzer sich für meine API anmelden können, indem sie ihre Login-Daten eingeben und diese mit einem "Senden"-Button übermitteln können.

Nachdem ich das Login-View erstellt hatte, habe ich mich anschließend an die Erstellung der Setting-View gemacht. Diese View ermöglicht es Benutzern, weitere Einstellungen einzugeben, wie zum Beispiel die URL, bevor sie diese mit einem "Speichern"-Button bestätigen können.

Anschließend habe ich mich an die Erstellung des Einfügen-Views gewagt, wo Benutzer Artikel mit Informationen wie Kategorie, Preis und Datum eingeben können. Dabei habe ich auch einen praktischen Picker eingebaut, der es den Benutzern erleichtert, die Kategorie auszuwählen.

Abschließend habe ich die Auswertungs-View entwickelt, die Benutzern eine Übersicht über alle gespeicherten Artikel bietet. Hier können Benutzer jedes ausgewählte Element löschen oder alle Elemente auf einmal löschen. Zudem wird auch der Gesamtpreis aller gespeicherten Artikel angezeigt, um Benutzern eine schnelle und einfache Übersicht zu geben.

4.3. Model

Die Klasse "SaveUpModel" wird verwendet, um Daten von Artikeln in einer .NET MAUI-Anwendung zu speichern und zu verwalten. Die Klasse erbt von der Basisklasse "ObservableObject", um die Aktualisierung von Eigenschaften innerhalb der Anwendung zu erleichtern.
Die Klasse verfügt über Eigenschaften, um Informationen wie die eindeutige ID des Artikels, den Produktnamen, die Kategorie, den Preis, das Datum und den Namen des Benutzers zu speichern. Diese Eigenschaften sind mit Attributen wie "JsonPropertyName" versehen, um die Serialisierung und Deserialisierung zu erleichtern.
Die Eigenschaften "Produkt" und "Kategorie" sind Zeichenfolgen, während die Eigenschaft "Wert" eine Gleitkommazahl ist. Die Eigenschaft "Datum" ist ein DateTime-Objekt, das das Datum des Artikels speichert.
Die Eigenschaft "DayTime" ist eine private Zeichenfolge, die nicht serialisiert wird. Sie wird verwendet, um die Zeitangabe des Datums im Format "tt:mm" anzuzeigen.
Die Eigenschaft "Name" ist eine Zeichenfolge, die den Namen des Benutzers speichert, der den Artikel hinzugefügt hat.
Jede Eigenschaft, die eine Änderung an den Daten des Artikels darstellt, wird durch die Methode "SetProperty" aufgerufen, die das Standardverhalten zum Aktualisieren von Eigenschaften in .NET MAUI-Anwendungen implementiert.
Insgesamt bietet die Klasse "SaveUpModel" eine einfache Möglichkeit, Daten von Artikeln in einer .NET MAUI-Anwendung zu speichern und zu verwalten.

Die Klasse "Configuration" enthält Eigenschaften wie den API-Schlüssel, URLs und Benutzerdaten. Jede dieser Eigenschaften wird überwacht, um sicherzustellen, dass Aktualisierungen vorgenommen werden, wenn sich die Daten ändern.

Die Klasse "ConfigManager" bietet Methoden zum Laden und Speichern von Konfigurationsdaten in einer JSON-Datei im Dateisystem. Der Pfad zur JSON-Datei wird in der statischen Variable "configPath" definiert.

Insgesamt bietet dieser Code eine einfache Möglichkeit, Konfigurationsdaten in einer .NET MAUI-Anwendung zu speichern und zu laden. Die Verwendung von JSON als Speicherformat ermöglicht eine einfache Serialisierung und Deserialisierung der Daten. Die Klasse "Configuration" bietet einen einfachen Zugriff auf die verschiedenen Konfigurationsoptionen, während die Klasse "ConfigManager" die Funktionalität zum Laden und Speichern der Daten bereitstellt.

4.4. ViewModel

Dieses Code-Segment stellt eine ViewModel-Klasse für die Auswertungsseite in einer .NET MAUI-Anwendung bereit. Die Klasse enthält eine Liste von SaveUp-Model-Objekten, eine ausgewählte SaveUp-Model-Instanz, Methoden zum Löschen eines einzelnen SaveUp-Model-Objekts sowie aller SaveUp-Model-Objekte für einen bestimmten Benutzer und eine Eigenschaft für die Gesamtsumme aller Werte in der Liste.

Die abstrakte Klasse "ViewModelBase" dient als Basisklasse für alle ViewModel-Klassen in einer .NET MAUI-Anwendung. Sie implementiert das INotifyPropertyChanged-Interface, um PropertyChanged-Events auszulösen, wenn Eigenschaften in abgeleiteten Klassen geändert werden. Die Methode "SetProperty" wird verwendet, um eine Eigenschaft auf einen neuen Wert zu setzen und das PropertyChanged-Event auszulösen, wenn sich der Wert ändert. Die Methode vergleicht den neuen Wert mit dem aktuellen Wert und löst das Event nur aus, wenn sich die Werte unterscheiden. Die Methode "OnPropertyChanged" löst das PropertyChanged-Event explizit aus und gibt dabei den Namen der geänderten Eigenschaft an. Durch Ableiten von "ViewModelBase" und Verwendung der "SetProperty"-Methode in den Eigenschaften-Klassen können Änderungen an den Eigenschaften von Views und UI-Elementen in der Anwendung erkannt und automatisch aktualisiert werden.

Das LoginViewModel-ViewModel implementiert eine einfache Anmeldung mit Benutzername und Passwort über eine HTTP-POST-Anforderung. Es prüft, ob ein API-Schlüssel in der Konfiguration vorhanden ist und ändert den Anzeigezustand des Benutzerinterfaces entsprechend. Das Setting-ViewModel-ViewModel stellt eine Ansicht zur Verfügung, in der der Benutzer die Anwendungskonfiguration ändern kann. Das Einfuegen-ViewModel-ViewModel ermöglicht das Hinzufügen von Daten zu einer Datensammlung durch Senden eines POST-Requests an einen RESTful-Webdienst mit dem API-Schlüssel aus der Konfiguration. Es führt grundlegende Überprüfungen durch, um sicherzustellen, dass die erforderlichen Daten vorhanden sind, bevor die Anforderung gesendet wird.

Die Get SaveUp ()-Methode ruft die Liste der SaveUp-Model-Objekte für den aktuellen Benutzer aus der Datenbank ab und ordnet jedes Objekt einem Datum zu. Es wird auch die Gesamtsumme der Werte berechnet. Die Execute Delete One ()-Methode löscht ein einzelnes SaveUp-Model-Objekt aus der Datenbank, basierend auf der ausgewählten Instanz. Die Execute Delete ()-Methode löscht alle SaveUp-Model-Objekte für den aktuellen Benutzer aus der Datenbank.

Die Klasse verwendet die ViewModelBase-Klasse als Basisklasse und implementiert die INotifyPropertyChanged-Schnittstelle, um die Bindung der Benutzeroberfläche an die Eigenschaften des ViewModel zu unterstützen. Die SaveUp-Model-Klasse ist ein separates Model-Objekt, das die Eigenschaften des einzufügenden Datensatzes definiert.

Zusammenfassend kann dieses Beispiel als Grundlage für die Implementierung eines einfachen RESTful-Clients mit .NET Maui dienen. Es zeigt, wie ViewModel-Klassen verwendet werden können, um die Insgesamt zeigt dieses Beispiel die Verwendung von ViewModel-Klassen in einer .NET MAUI-Anwendung, um Daten von einem RESTful-Webdienst abzurufen, zu bearbeiten und zu löschen. Es wird eine Konfigurationsdatei verwendet, um den API-Schlüssel und andere Anwendungsparameter zu speichern.

Die Login-View-Model-Klasse implementiert die Authentifizierung von Benutzern über eine API, während das Setting-View-Model die Einstellungen der Anwendung verwaltet. Das Einfügen View-Model-ViewModel ermöglicht das Hinzufügen von Daten zu einer Datensammlung über einen RESTful-Webdienst.

Die Auswertung-View-Model-Klasse enthält die Logik für die Auswertung von Daten, die von einem RESTful-Webdienst abgerufen wurden, einschließlich der Anzeige einer Liste von SaveUp-Model-Objekten und der Berechnung der Gesamtsumme der Werte in dieser Liste. Es enthält auch Methoden zum Löschen einzelner Datensätze oder aller Datensätze für einen bestimmten Benutzer aus der Datenbank.

Durch die Verwendung von ViewModel-Klassen in einer .NET MAUI-Anwendung können Entwickler komplexe Funktionalitäten implementieren und die Anwendungskomplexität reduzieren, indem sie die Datenverarbeitung von der Benutzeroberfläche trennen. Dies ermöglicht es Entwicklern, die Codequalität zu verbessern und die Wartbarkeit zu erhöhen.

4.5. API-Dokumentation

Übersicht

Dieses Projekt beinhaltet drei Teile: Models, Controllers und Service. Es verwendet eine bereits vorhandene API, die für die Verwendung von MongoDB als NoSQL-Datenbank optimiert ist.
Ich musste nur das Model kopieren, das Controller anpassen und den Service ebenfalls kurz kopieren und umbenennen. Die Zeit, die ich dafür benötigt habe ich, betrug nur 10 Minuten. Und ich habe das nicht im PSP erwähnt weil das Ganze auf einem anderem Projekt lief.

Models

Dieser Teil definiert das Modell für die SaveUP-Daten, die gespeichert werden sollen. Es enthält Felder wie ID, Produkt, Kategorie, Wert, Datum und Name. Diese Felder werden mithilfe von MongoDB.Bson und System.Text.Json serialisiert.

Controllers

Dieser Teil definiert das Verhalten der API, indem es HTTP-Anforderungen wie GET, POST und DELETE verarbeitet. Es nutzt den SaveUpController und verwendet eine Instanz des ISaveUp-Services, um die Datenbankoperationen durchzuführen.

Service

Dieser Teil definiert die Schnittstelle ISaveUp für die Datenbankoperationen. Es wird von einer konkreten Klasse namens SaveUp_MongoDB implementiert, die auf MongoDB basiert und die Methoden zum Hinzufügen, Löschen und Abrufen von Daten bereitstellt.

Zusammenfassung

Dieses Projekt verwendet eine bereits vorhandene API, die für die Verwendung von MongoDB als NoSQL-Datenbank optimiert ist. Ich musste nur das Modell kopieren, das Controller anpassen und den Service ebenfalls kurz kopieren und umbenennen. Die Zeit, die du dafür benötigt hast, betrug nur 10 Minuten. Die Daten werden mithilfe von MongoDB.Bson und System.Text.Json serialisiert und das Projekt ist so aufgebaut, dass es einfach zu warten und zu aktualisieren ist.

5. Kontrollieren

5.1. Manueller Test (White Box)

Tabelle 5 White Box

Beschreibung	Check	Anmerkung
Alle Buttons geprüft	x	
Request geprüft	x	Zertifikat Probleme
Setting speichern geprüft	x	
Falsche daten geprüft	x	Gibt Meldung
Gesamtrechnung geprüft	x	

5.2. Anforderungen Checkliste

Tabelle 6 Checkliste

Beschreibung	Check	Anmerkung
Die App muss SaveUp heißen und ein eigenes App-Icon haben.	x	
Die App muss mindestens aus 2 Content Pages bestehen.	x	
Das GUI-Design der App, einschließlich der Mock-ups, muss umgesetzt werden.	x	
Der Benutzer muss die Möglichkeit haben, eine Kurzbeschreibung und den Preis des gesparten Artikels zu erfassen (mindestens 2 Eingaben).	x	
Die App muss zwei Menüfunktionen (Action) zur Speicherung und zum Aufruf der Listendarstellung bieten	x	
Die App muss eine einfache bzw. intuitive Bedienung und ein geeignetes Layout (Styles) haben.	x	
Die Codestrukturierung der App muss dem MVVM-Entwurfsmuster entsprechen.	x	
Die App muss XAML-Styles der Steuerelemente verwenden.	x	
Die App muss die Möglichkeit bieten, erfasste Einträge (Clear) einzeln oder komplett zu löschen.	x	
Datum/Uhrzeit des Kaufverzichts müssen als zusätzliches Attribut erfasst werden.	x	
Die App muss umfassend dokumentiert werden.	x	
Die App muss gründlich getestet werden, um eine fehlerfreie Funktionsweise zu gewährleisten.	x	
Die erfassten Daten müssen in einer Backend-Datenbank mit REST-Implementierung gespeichert werden.	x	

6. Auswertung

6.1. Fazit

Es war eine wahre Freude, das Projekt umzusetzen. Dank meiner Erfahrungen in meinem Smart Home Projekt war es mir möglich, schnell Fortschritte zu machen und die Anforderungen umzusetzen. Lediglich die Schwierigkeiten mit dem Zertifikat und der Umstellung des Projekts von .NET 6 auf .NET 7 haben mich etwas Zeit gekostet.

Ich bin sehr beeindruckt von der Leistungsfähigkeit von .NET MAUI und sehe darin die Zukunft der Entwicklung plattformübergreifender Anwendungen. Obwohl das Framework noch einige Probleme hat, ist es verständlich, da es noch neu ist. Ich werde mich auf jeden Fall weiterhin mit .NET MAUI beschäftigen, da ich der Meinung bin, dass es eine vielversprechende Technologie ist. Es ist großartig, dass .NET MAUI auf allen Betriebssystemen lauffähig ist und somit eine breite Zielgruppe ansprechen kann.

6.2. Soll ist Vergleich

Es ist wichtig, bei der Planung von Projekten immer genügend Zeit für unvorhergesehene Probleme einzuplanen. In diesem Fall waren es insbesondere Probleme mit der API, die zu Verzögerungen geführt haben. Es ist gut, dass du diese Herausforderungen im Vorfeld erwähnt hast, um sicherzustellen, dass alle Beteiligten darauf vorbereitet sind.

Es ist auch sinnvoll, den Gantt-Chart regelmässig zu überprüfen und gegebenenfalls anzupassen, um sicherzustellen, dass das Projekt auf Kurs bleibt. Es ist gut, dass ich den Gantt-Chart aktualisiert hast, um die tatsächliche Projektdauer widerzuspiegeln. Das wird dazu beitragen, zukünftige Projekte besser zu planen und realistischere Zeitrahmen zu setzen.

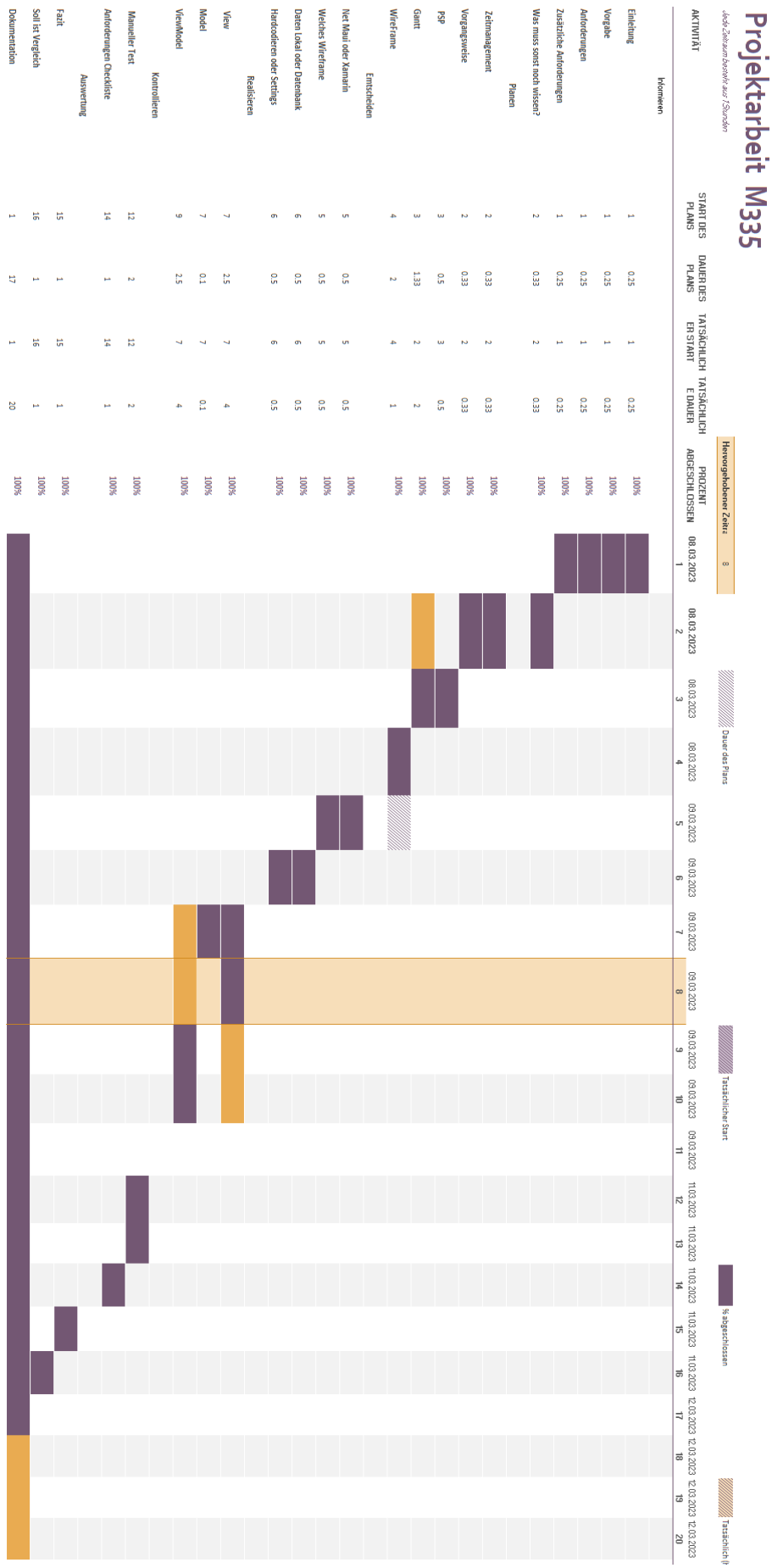

Abbildung 9 Gantt Finish

7. Anhänge

7.1. Tabellenverzeichnis

7.2. Bilderverzeichnis

7.3. Links

Alle Bilder, wo ich in der Projekt benutzt habe sind von einer Website die gratis Icon herunterladen kann Link: https://icons8.de/icons/set/free-icons
Mein Net Maui Projekt: https://github.com/Raphi93/SmartHome_Frontend
Mein API: https://github.com/Raphi93/SmartHome_Backend

7.4. API URL

Da ich dieses Projekt auf GitHub Öffentlich ist werde ich meine Daten per Teams als ein Text File zur Verfügung stellen.