ipso! Bildung seit 1876

SAVEUP

Verzicht für Reichtum

Fokko Vos

ICT 335 – Mobile Applikationen realisieren

ipso! Bildung seit 1876

Inhaltsverzeichnis

ipso! Bildung seit 1876

1 Versionierung

Version	Autor	Datum	Beschreibung
0.1	Fokko Vos	17.06.2024	Erstellung der Dokumentation
0.2	Fokko Vos	17.06.2024	Ausgangslage und Ziele Dokumentieren
0.3	Fokko Vos	17.06.2024	Informationen zu den Anforderungen Dokumentiert
0.4	Fokko Vos	17.06.2024	Mockups Dokumentiert
0.5	Fokko Vos	24.06.2024	Dokumentation der Realisierung
0.6	Fokko Vos	29.06.2024	Dokumentation der Unit-Tests
0.7	Fokko Vos	30.06.2024	Finalisierung der Dokumentation
1.0	Fokko Vos	30.06.2024	Aktualisieren aller Verweise

1 - Versionierung

2 Ausgangslage

Im Zuge der wachsenden Bedeutung von finanzieller Selbstverwaltung und Sparsamkeit hat sich die Notwendigkeit einer mobilen Applikation ergeben, die Nutzern hilft, ihre kleinen, alltäglichen Ausgaben zu verwalten und zu reduzieren. Viele Menschen neigen dazu, regelmässig Geld für Dinge wie Kaffee, Snacks oder andere kleine Annehmlichkeiten auszugeben, ohne die langfristigen Auswirkungen dieser Ausgaben auf ihre Sparziele zu berücksichtigen. Eine intuitive und einfach zu bedienende App könnte hierbei unterstützen, indem sie die Erfassung und Übersicht dieser Ausgaben erleichtert und so das Sparen für grössere Ziele wie Urlaub oder andere bedeutende Anschaffungen fördert.

3 Ziele

Das Hauptziel des Projekts ist die Entwicklung einer .NET MAUI App, genannt "SaveUp", welche die Nutzer darin unterstützt, Geld zu sparen, indem sie Verzicht auf kleinere, regelmässige Ausgaben üben. Die App soll mindestens drei Content Pages umfassen und es den Nutzern ermöglichen, sowohl eine Kurzbeschreibung als auch den Preis des verzichteten Artikels zu erfassen und zu speichern. Diese Daten sollen in einer übersichtlichen Liste, inklusive der gesamten Ersparnis, angezeigt werden. Weitere Ziele umfassen:

- **Benutzerfreundlichkeit:** Eine einfache und intuitive Bedienung der App, die durch ein effektives GUI-Design und eine geeignete Layoutgestaltung erreicht wird.
- **Effektive Datenverwaltung:** Implementierung von Funktionalitäten zum Speichern und Abrufen der Daten sowie optional das Löschen der Einträge und Persistenz der Daten entweder lokal oder über eine Backend-Datenbank.
- **Visuelle Aufbereitung:** Grafische Darstellung der eingesparten Beträge, um die Nutzer visuell über ihre Fortschritte zu informieren und zu motivieren.
- **Dokumentation und Testing:** Eine umfassende Dokumentation des Entwicklungsprozesses sowie des finalen Produkts und das Durchführen von Tests zur Sicherstellung der Funktionalität und Benutzerfreundlichkeit der App.

4 Informieren

Die Entwicklung der SaveUp-App basiert auf spezifischen Anforderungen, die im Rahmen der Projektinitiierung definiert wurden. Diese Anforderungen stellen die Basis für die Ausrichtung des gesamten Entwicklungsprozesses dar und sind in der folgenden Tabelle zusammengefasst:

Anforderungskategorie	**Details**
App-Struktur	Mindestens drei Content Pages
Benutzereingaben	Erfassung von Kurzbeschreibung und Preis
Datenanzeige	Anzeige der gesparten Produkte und Summen
Menüfunktionen	Speicherung und Aufruf der Listendarstellung
Benutzerfreundlichkeit	Einfache und intuitive Bedienung
Technologie	Umsetzung mit .NET MAUI / C#
Design	Anwendung von XAML-Styles
Zusätzliche Funktionen	Optionale Erweiterungen wie Datenpersistenz und grafische Darstellungen

2 - Grundlegende Anforderungen der SaveUp-App

5 Planen

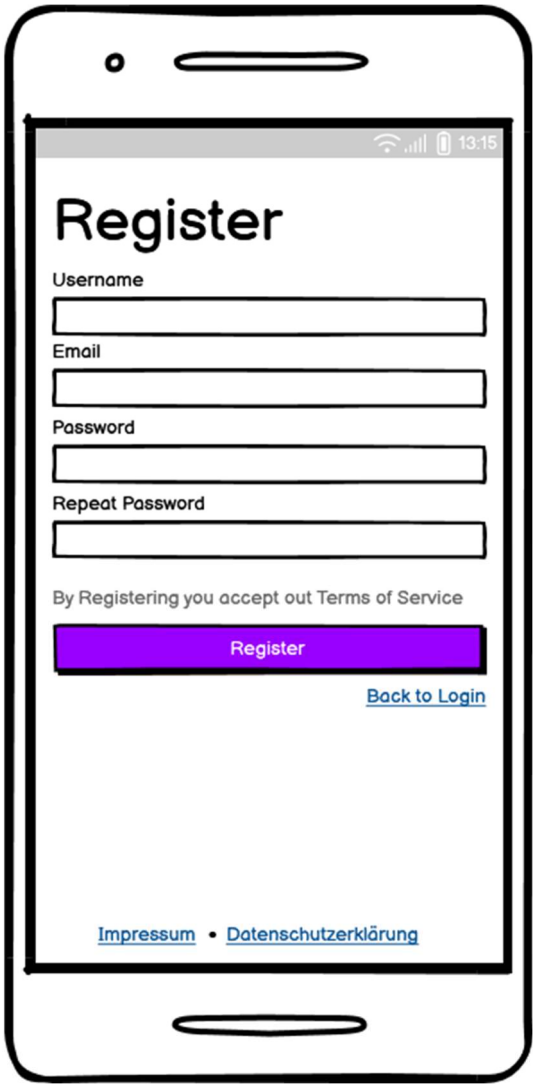
5.1 Gantt

SaveUp																			
Projektleiter: Fokko Vos		Projektstart: **17.06.2024**			KW 25								KW 26						
					Mo	Di	Mi	Do	Fr	Sa	So		Mo	Di	Mi	Do	Fr	Sa	So
Aufgabe	**SOLL**	**IST**	**Erledigt**		17	18	19	20	21	22	23		24	25	26	27	28	29	30
Informieren	**1**	**1**	**100%**																
Projektanforderungen überprüfen	1	1	100%																
Planen	**3.5**	**4.5**	**100%**																
Entwicklungsumgebung einrichten	0.5	1	100%																
Design und Mockups erstellen	2	3	100%																
Dokumentationsstruktur erstellen	1	0.5	100%																
Entscheiden	**1**	**1**	**100%**																
Grundstruktur und Technologien festlegen	1	1	100%																
Realisieren	**12**	**13**	**100%**																
Erstellung des Projektes	4	5	100%																
Entwicklung der Content Pages	2	4	100%																
Implementierung der Produkterfassung	3	1	100%																
Entwicklung der Menüfunktionen	1	2	100%																
Implementierung der Backend-Datenbankverbindung	2	1	100%																
Kontrollieren	**1**	**0.5**	**100%**																
Unit-Tests Durchführen	1	0.5	100%																
Auswerten	**4.5**	**3.5**	**100%**																
Fortlaufende Dokumentation vervollständigen	1	1	100%																
Präsentation vorbereiten	3	2	100%																
Abgabe	0.5	0.5	100%																
Gesamt	**23**	**23.5**	**100%**		17	18	19	20	21	22	23		24	25	26	27	28	29	30

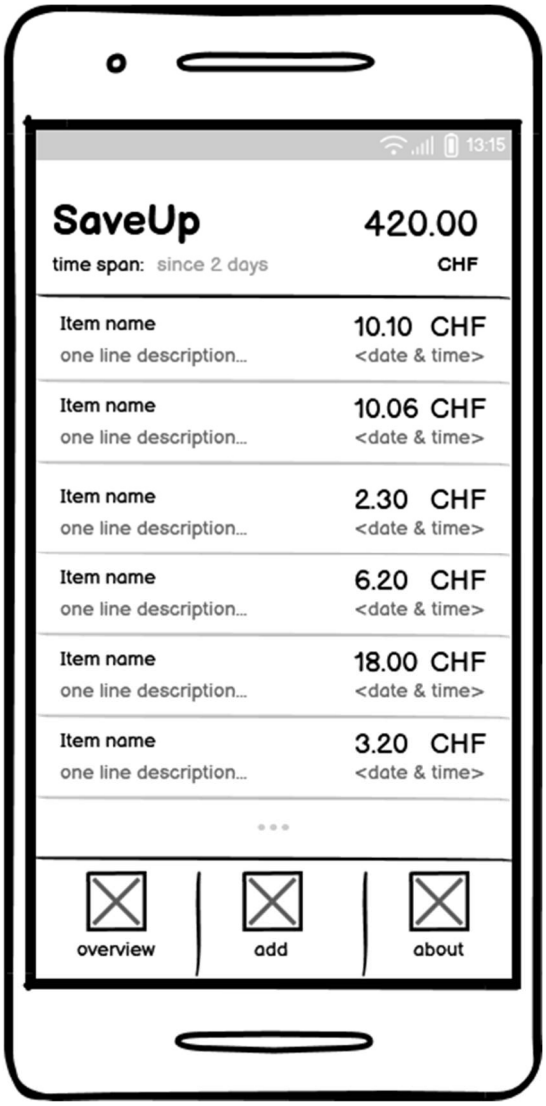
3 - Zeitplanung

5.2 Mockups

Die Anmeldeseite ermöglicht den Zugang zur App durch Eingabe von E-Mail und Passwort. Es gibt auch eine Option für Benutzer, die ihr Passwort vergessen haben oder ein neues Konto erstellen möchten. Die Passwort vergessen Funktion wird für die erste Version ausgegraut.



Die Registrierungsseite ermöglicht es Nutzern, ein neues Konto zu erstellen, indem sie Benutzername, E-Mail und Passwort eingeben. Durch Klicken auf "Register" akzeptieren Nutzer die Nutzungsbedingungen.



1 - Mockup: HomePage

2 - Mockup: HomePage Detailansicht

Die Startseite zeigt eine Zusammenfassung der gesparten Beträge an, mit einer Auflistung einzelner Einträge, die den Namen, Preis und eine kurze Beschreibung enthalten, sowie Datum und Uhrzeit der Einsparung.

In der Detailansicht eines Produkts werden der Preis und eine detaillierte Beschreibung des Produkts angezeigt, mit Optionen zum Entfernen des Eintrags aus der Liste.

3 - Mockup: AddPage

Diese Seite ermöglicht es Nutzern, Produkte hinzuzufügen, indem sie Details wie Namen, Preis und Beschreibung eingeben. Die gespeicherten Daten können dazu verwendet werden, die Einsparungen durch Verzicht zu überwachen.

4 - Mockup: AboutPage

Die Entwicklerseite enthält Informationen über die Entwickler der App, mit Links zu weiteren Einstellungen, einer Option zum Unterstützen des Entwicklers ("Buy me a coffee") und Zugang zu Impressum und Datenschutzerklärung.

5 - Mockup: SettingsPage

Die Einstellungsseite bietet Optionen zur Anpassung der App, einschliesslich der Einstellung der Währung (CHF), der Sprache (Deutsch) und des Designs. Es gibt auch eine Option zum Zurücksetzen des Zeitintervalls, das die Dauer der Einsparungen anzeigt. Sowie die Option sich auszuloggen.

6 Entscheiden

6.1 Platform und Technologie Stack

Die Entwicklung der SaveUp App erfolgt auf der .NET MAUI-Plattform, welche eine einheitliche Codebasis für iOS, Android und Windows ermöglicht. Die Verwendung von .NET MAUI unterstützt eine effiziente, plattformübergreifende Entwicklung, während C# als Programmiersprache für die Backend-Logik und XAML für das Frontend-Design genutzt wird.

6.2 Architekturmodell

Die App basiert auf dem Model-View-ViewModel (MVVM) Muster. Dieses Muster fördert eine klare Trennung zwischen der Präsentationslogik und der Geschäftslogik. Die Datenbindung zwischen den Views und den ViewModels erleichtert die Wartung und Erweiterung der App, indem sie eine lose Kopplung und eine verbesserte Testbarkeit ermöglicht.

6.3 App-Struktur und Navigation

Die App besteht aus drei Hauptseiten: der HomePage zur Anzeige gesparter Einträge, der AddPage für das Hinzufügen neuer Sparziele, und der AboutPage, die Informationen über die App und den Entwickler bietet. Die Navigation zwischen diesen Seiten ist durch eine konstante untere Menüleiste gewährleistet, die eine schnelle und einfache Benutzerführung ermöglicht.

6.4 Responsive Design und Usability

Um eine optimale Benutzererfahrung auf verschiedenen Geräten sicherzustellen, wird responsives Design verwendet. XAML-Styles sorgen für ein konsistentes und ansprechendes Erscheinungsbild, während intuitive Interaktionsmuster wie Touch-Eingaben und gestische Steuerungen integriert werden.

6.5 Datenmanagement und Persistenz

Für die SaveUp App wird ein zentrales Backend verwendet, das über RESTful APIs kommuniziert. Dies ermöglicht es der App, Daten effizient und sicher in einer zentralen Datenbank zu speichern und abzurufen. Die Verwendung von REST APIs unterstützt die Plattformunabhängigkeit und Skalierbarkeit der Anwendung und erleichtert die Wartung sowie das Update der Datenverarbeitungslogik auf dem Server, ohne Client-Updates erzwingen zu müssen. Dieser Ansatz garantiert eine robuste Datenkonsistenz und erleichtert die Implementierung weiterer Funktionen wie Multi-Device-Synchronisation und Echtzeit-Datenaktualisierungen.

6.6 Sicherheit und Datenschutz

Moderne Sicherheitsprotokolle und Authentifizierungsmechanismen, wie OAuth, schützen die Kommunikation zwischen dem Client und dem Server. Sensible Daten werden verschlüsselt übertragen und gespeichert, um Datenschutz und Compliance mit aktuellen Sicherheitsstandards zu gewährleisten. Die App implementiert ausserdem Strategien zur Fehlerbehandlung und Validierung der Datenintegrität, um die Sicherheit und Zuverlässigkeit der Nutzerdaten sicherzustellen.

7 Realisieren

7.1 Erstellung des Projektes

Das Repository wurde auf GitHub eingerichtet, um die Zusammenarbeit und Versionskontrolle zu erleichtern. Die Struktur umfasst Verzeichnisse für die App, Tests, Backend und Datenmodelle.

SaveUp
Hauptprojektverzeichnis, enthält die Quellcodedateien der Applikation.
SaveUp.Tests
Enthält die Unit-Testdateien.
SaveUpBackend
Backend für die Datenverarbeitung.
SaveUpModels
Enthält die Datenmodelle.

7.2 Entwicklung der Content Pages

Die Applikation umfasst mehrere Content Pages, die den Kern der Benutzeroberfläche bilden:

HomePage
Zeigt die Liste der gesparten Verzichtsprodukte an.
AddPage
Ermöglicht das Hinzufügen neuer Verzichtsprodukte.
AboutPage
Zeigt Informationen über die App und die Entwickler an.

Die Content Pages wurden unter Verwendung von XAML für das Design und C# für die Logik implementiert. Hierbei wurde das MVVM-Muster befolgt, um eine klare Trennung zwischen der Darstellung und der Geschäftslogik zu gewährleisten.

Benutzerzentriertes Design
Das Design der Benutzeroberfläche wurde mit einem Fokus auf Benutzerfreundlichkeit und Effizienz entwickelt. Besonderes Augenmerk wurde auf die Minimierung der erforderlichen Interaktionen (Tipp-Eingaben und Mausklicks) gelegt, um die Bedienung auch mit Handschuhen praktikabel zu machen.

MVVM für Pages
Die MVVM-Architektur wurde speziell für die Seiten (Pages) der App umgesetzt. Jede Seite hat ein zugehöriges ViewModel, das die Datenlogik und Interaktionen handhabt. Die Views binden sich an diese ViewModels, wodurch eine klare Trennung der Benutzeroberflächenlogik und der Geschäftslogik erreicht wird. Dies erleichtert das Testen und die Wiederverwendung von Code und ermöglicht ein reaktives Design, bei dem sich die Benutzeroberfläche dynamisch an die zugrunde liegenden Daten anpasst.

Code-Behind für Components
Für kleinere Komponenten und Dialoge haben wir uns entschieden, die traditionelle Code-Behind-Technik zu verwenden. Dieser Ansatz wurde gewählt, um die Komplexität zu reduzieren und die Entwicklungsgeschwindigkeit für kleinere, weniger komplexe Teile der Benutzeroberfläche zu erhöhen. Obwohl diese Komponenten nicht die MVVM-Architektur nutzen, sind sie dennoch so gestaltet, dass sie gut mit den MVVM-Teilen der App interagieren und eine konsistente Benutzererfahrung bieten.

Multi-Lingual Support
Die Benutzeroberfläche der SaveUp-App wurde mit umfangreichem Multi-Lingual Support entwickelt, um eine breite, weltweite Nutzerbasis anzusprechen. Durch die Integration von Sprachressourcen in .resx-Dateien bietet die App lokalisierte Texte für UI-Elemente wie Menüs, Dialoge und Anweisungen in Englisch und Deutsch. Dies verbessert die Zugänglichkeit und Benutzerfreundlichkeit erheblich, indem Benutzer die App in ihrer bevorzugten Sprache nutzen können.

7.2.1 Schlüsselkomponenten

AuthManager
Der AuthManager ist für die Handhabung der Authentifizierung zuständig. Er ermöglicht das Einloggen und Ausloggen von Benutzern, verwaltet den Zugriff auf Benutzerdaten wie Tokens und Benutzer-IDs und signalisiert Änderungen im Anmeldestatus durch Events. Dies gewährleistet eine sichere und reibungslose Benutzererfahrung, indem er sicherstellt, dass nur autorisierte Benutzer Zugriff auf bestimmte Funktionen und Daten haben. Der AuthManager ist somit zentral für die Sicherheit und Benutzerfreundlichkeit der App.

SettingsManager
Der SettingsManager verwaltet die Benutzereinstellungen. Er ermöglicht es Benutzern, ihre Präferenzen wie Sprache und Thema festzulegen und sorgt dafür, dass diese Einstellungen konsistent angewendet werden. Durch die Speicherung und das Laden von Benutzerpräferenzen trägt die Klasse wesentlich zur Personalisierung und Benutzerfreundlichkeit der App bei.

Localization
Die Lokalisierung ist ein wesentlicher Aspekt der Benutzeroberfläche, um Multi-Lingual Support zu bieten. Die Localization-Klasse verwaltet die Sprachressourcen und ermöglicht eine dynamische Sprachumschaltung in der App. Sie bindet UI-Elemente an lokalisierte Texte, die in .resx-Dateien definiert sind, und aktualisiert die Anzeige sofort bei einem Sprachwechsel.

7.3 Implementierung der Produkterfassung

Die Produkterfassung ermöglicht es Nutzern, Details zu den gesparten Verzichtsprodukten einzugeben und zu speichern

- Model: ItemResponse in SaveUpModels/DTOs/Responses/ItemResponse.cs
- Service: ItemAPIService in SaveUp/Services/API/ItemAPIService.cs
- ViewModel: AddViewModel in SaveUp/ViewModels/AddViewModel.cs
- View: AddPage.xaml in SaveUp/Views/AddPage.xaml

Die Eingaben der Benutzer (Name und Preis, evtl. Kurzbeschreibung) werden im ViewModel verarbeitet und mithilfe des ItemAPIService gespeichert. Die gespeicherten Produkte werden dann auf der HomePage angezeigt.

7.4 Entwicklung der Menüfunktionen

Die AppShell bindet die Hauptseiten der App (HomePage, AddPage, AboutPage) als Tabs ein. Dies ermöglicht eine schnelle und intuitive Navigation zwischen den verschiedenen Funktionen der App.

- HomePage: SaveUp/Views/HomePage.xaml
- AddPage: SaveUp/Views/AddPage.xaml
- AboutPage: SaveUp/Views/AboutPage.xaml

Die Einstellungen können über die AboutPage aufgerufen werden, um die Benutzeroberfläche nicht zu überladen und die Benutzerfreundlichkeit zu erhöhen.

7.5 Implementierung der Backend-Datenbankverbindung

Das Backend ist in SaveUpBackend implementiert und umfasst Controller für die API-Endpunkte, die die Datenverwaltung handhaben. Die App kommuniziert mit dem Backend über RESTful APIs. Diese APIs ermöglichen es, Daten effizient zu speichern, abzurufen und zu löschen. Die Datenmodelle und Endpunkte sind klar definiert, um eine nahtlose Integration zu gewährleisten.

Moderne Sicherheitsprotokolle und Authentifizierungsmechanismen (z.B. OAuth) sichern die Datenkommunikation. Sensible Daten werden verschlüsselt übertragen und gespeichert, um den Datenschutz zu gewährleisten.

API-Services Implementierung

BaseAPIService
Dies ist die Grundlage für alle API-Services. Es handhabt die grundlegenden CRUD-Operationen und stellt eine generische Implementierung zur Verfügung, die von spezifischen Services erweitert wird.
Spezifische API-Services
Für jede Art von Daten oder Aktion gibt es einen spezifischen Service, wie ItemAPIService und UserAPIService. Diese Services erben von BaseAPIService und implementieren teilweise zusätzliche spezifische Funktionen.

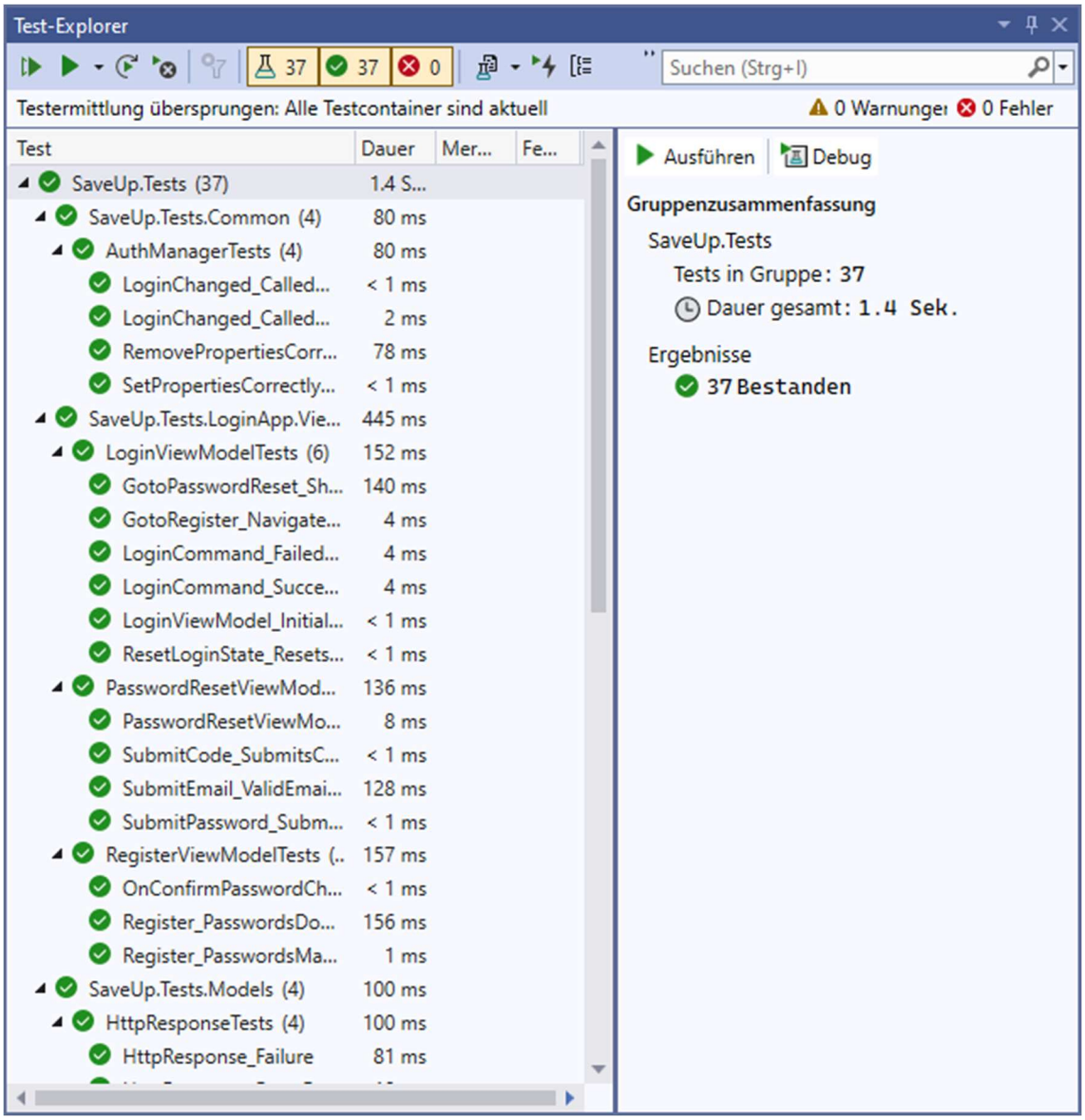
7.6 Kontrollieren

In dieser Phase des Projekts lag der Fokus auf der Durchführung und Dokumentation von Tests, um die Funktionalität und Zuverlässigkeit der entwickelten App sicherzustellen. Die Teststrategie umfasste Unit Test.

7.6.1 Unit-Tests

Die Unit Tests zielten darauf ab, die Funktionalität der einzelnen Komponenten isoliert zu überprüfen. Dies beinhaltete die Validierung der Kernfunktionalitäten, das korrekte Handling von Eingabeparametern und die angemessene Reaktion auf verschiedene Instanziierung-/Aufruf-möglichkeiten.
Die Ergebnisse der Unit Tests wurden in Visual Studio dokumentiert und sind im beigefügten Testbericht detailliert aufgeführt. Sie zeigen eine hohe Abdeckung und erfolgreiche Validierung der-Komponenten.

Verantwortlich: Fokko Vos
Durchführung: 29.06.2024
Ort: files/Berichte/test_results.trx



6 - Unit Tests

8 Auswerten

8.1 Fazit

Im Verlauf meines Projekts, der Entwicklung der Ausgaben-Tracking-Applikation SaveUp, stand ich vor verschiedenen Herausforderungen und konnte zahlreiche Chancen nutzen, die mir die Arbeit mit .NET MAUI bot. Trotz einer anfänglichen Einarbeitungszeit und der relativen Neuheit von .NET MAUI, die einige Aspekte erschwerte, erwies sich die Wahl dieses Frameworks als bereichernd. Dank meiner gestiegenen Erfahrung und der flexiblen Programmierung aus früheren Projekten konnte ich die meisten Probleme effizient lösen.

Ein positiver Aspekt war die schnelle und reibungslose Integration des Backends in die neue .NET MAUI-Anwendung. Trotz einiger verbliebener Herausforderungen, wie leichten Lags in der UI und der Notwendigkeit verbesserter Fehlerbehandlung, hat mir das Projekt gezeigt, wie flexibel und kompatibel .NET MAUI mit bestehenden Technologien ist.

Obwohl das Projekt zum aktuellen Zeitpunkt noch nicht produktionsbereit ist, bin ich stolz auf meine Arbeit und fühle mich durch die überwundenen Herausforderungen bereichert. Die Kombination aus technischem Fortschritt, kreativer Problemlösung und dem erfolgreichen Abschluss komplexer Aufgaben hinterlässt ein tiefes Gefühl der Zufriedenheit.

Für zukünftige Projekte würde ich .NET MAUI aufgrund seiner Vorteile, des Spassfaktors und der fortschrittlichen Technologie erneut wählen, trotz gelegentlicher, unerwarteter Fehler, die typisch für Microsoft-spezifische Produkte sein können. Mein Projekt hat nicht nur meine technischen Fähigkeiten verbessert, sondern auch mein Vertrauen in die Wahl moderner Entwicklungswerkzeuge gestärkt. Ich betrachte mein Projekt als einen Erfolg im Rahmen eines Schulprojekts und freue mich auf zukünftige Entwicklungen mit .NET MAUI. Am liebsten arbeite ich dennoch in Rust.

9 Verweise

9.1 Tabellenverzeichnis

9.2 Abbildungsverzeichnis