ipso! Bildung seit 1876

SaveUp

By David R.

ipso! Bildung seit 1876

Inhaltsverzeichnis

1 Versionsverzeichnis

Version	Autor	Datum
1.0.0	David Reichert	30.06.2024
1.1.0	David Reichert	01.07.2024
2.0.0	David Reichert	05.07.2024

2 Einleitung

Das Projekt SaveUp wurde ins Leben gerufen, um eine effiziente und benutzerfreundliche mobile Anwendung zur Verwaltung von Ausgaben zu entwickeln. In der heutigen digitalen Ära ist das Management persönlicher Finanzen ein wesentlicher Bestandteil des täglichen Lebens. Viele Menschen haben jedoch Schwierigkeiten, ihre Ausgaben zu verfolgen und zu kontrollieren. Hier setzt SaveUp an.

SaveUp ist eine mobile App, die es Benutzern ermöglicht, ihre finanziellen Ausgaben auf einfache Weise zu erfassen, zu kategorisieren und zu verwalten. Die App bietet eine intuitive Benutzeroberfläche, die es ermöglicht, neue Ausgaben schnell und unkompliziert hinzuzufügen. Durch die Verwendung des .NET MAUI-Frameworks wird eine plattformübergreifende Kompatibilität gewährleistet, sodass die Anwendung sowohl auf Android- als auch auf iOS-Geräten genutzt werden kann.

Ein zentrales Element des Projekts ist die Implementierung des MVVM (Model-View-ViewModel) Designmusters. Dieses Muster stellt sicher, dass die Geschäftslogik und die Benutzeroberfläche klar voneinander getrennt sind, was die Wartbarkeit und Erweiterbarkeit der Anwendung erheblich verbessert. Darüber hinaus ermöglicht die MVVM-Architektur eine saubere Datenbindung, wodurch Änderungen an den Datenmodellen automatisch in der Benutzeroberfläche reflektiert werden.

Ein weiteres Schlüsselelement des Projekts ist die Speicherung der Daten in JSON-Dateien. Dies ermöglicht eine einfache und flexible Datenverwaltung, da die Ausgaben lokal auf dem Gerät gespeichert werden können. Beim Start der Anwendung werden die gespeicherten Daten geladen und angezeigt, sodass der Benutzer stets einen aktuellen Überblick über seine finanziellen Ausgaben hat.

Das Projekt SaveUp richtet sich an alle, die ihre Finanzen besser verwalten möchten, sei es, um Einsparungen zu erzielen, Budgetziele zu erreichen oder einfach nur einen besseren Überblick über ihre Ausgaben zu haben. Durch die Kombination aus modernem Design, benutzerfreundlicher Oberfläche und robuster Architektur bietet SaveUp eine leistungsstarke Lösung für das Finanzmanagement.

3 Anforderungen

3.1 Funktionale Anforderungen

- **Login**: Benutzer sollen sich mit Benutzernamen und Passwort anmelden können.
- **Ausgaben erfassen**: Benutzer sollen neue Ausgaben mit einer Beschreibung und einem Betrag hinzufügen können.
- **Ausgaben anzeigen**: Alle erfassten Ausgaben sollen in einer Liste angezeigt werden.
- **Ausgaben löschen**: Benutzer sollen Ausgaben löschen können.
- **Daten speichern**: Ausgaben sollen in einer JSON-Datei gespeichert und beim Start der App geladen werden.

3.2 Nicht-funktionale Anforderungen

- **Benutzerfreundlichkeit**: Die App soll eine intuitive und einfache Benutzeroberfläche bieten.
- **Performance**: Die App soll auch bei vielen gespeicherten Ausgaben performant bleiben.
- **Sicherheit**: Benutzerdaten sollen sicher gespeichert werden.

4 Informieren

4.1.1 Unterrichtssitzungen

Die Unterrichtseinheiten zur mobilen Entwicklung mit .NET MAUI und dem MVVM-Designmuster spielten eine zentrale Rolle bei der Umsetzung des Projekts SaveUp. Diese Sitzungen boten eine umfassende Einführung in die Konzepte und Best Practices, die für die Entwicklung moderner mobiler Anwendungen erforderlich sind.

Zu den behandelten Themen gehörten:

- **Einführung in .NET MAUI**: Überblick über das plattformübergreifende Framework, seine Vorteile und Anwendungsfälle.
- **MVVM-Architektur**: Detaillierte Erklärung des Model-View-ViewModel-Musters, einschließlich der Trennung von Logik und Benutzeroberfläche, Datenbindung und Kommandos.
- **Datenbindung**: Techniken zur Bindung von Datenmodellen an UI-Komponenten, um eine reaktive und dynamische Benutzeroberfläche zu schaffen.
- **State Management**: Verwaltung des Zustands in mobilen Anwendungen, um eine konsistente Benutzererfahrung zu gewährleisten.
- **Kommandos und Events**: Implementierung von Benutzeraktionen und deren Handhabung in der MVVM-Architektur.
- **Best Practices**: Empfehlungen für die Strukturierung des Codes, Optimierung der Performance und Sicherstellung der Wartbarkeit.

Diese Unterrichtseinheiten vermittelten nicht nur theoretisches Wissen, sondern boten auch praktische Übungen und Beispiele, die direkt auf das SaveUp-Projekt angewendet werden konnten. Dies ermöglichte eine fundierte Vorbereitung auf die tatsächliche Entwicklungsarbeit und stellte sicher, dass alle Teammitglieder ein gemeinsames Verständnis der verwendeten Technologien und Methoden hatten.

4.1.2 OneNote des Dozenten

Das OneNote des Dozenten diente als zentrales Repository für alle projektrelevanten Informationen, Materialien und Anleitungen. Es enthielt detaillierte Beschreibungen der verschiedenen Themenbereiche, die im Unterricht behandelt wurden, sowie zusätzliche Ressourcen und weiterführende Literatur.

Inhalte des OneNote:

- **Dokumentation und Tutorials**: Ausführliche Anleitungen zur Einrichtung der Entwicklungsumgebung, Nutzung von .NET MAUI und Implementierung des MVVM-Musters.

- **Beispielcode**: Beispielprojekte und Code-Snippets, die verschiedene Aspekte der mobilen Entwicklung und MVVM-Architektur illustrieren.
- **Problemlösungen**: Lösungen für häufig auftretende Probleme und Herausforderungen, auf die Entwickler stoßen könnten.
- **Best Practices und Empfehlungen**: Hinweise zur Optimierung der Entwicklungspraxis, Verbesserung der Codequalität und Sicherstellung der Performance.

Das OneNote war ein unverzichtbares Werkzeug, das als Referenz und Nachschlagewerk diente. Es ermöglichte es den Entwicklern, schnell auf relevante Informationen zuzugreifen und diese direkt in ihre Arbeit zu integrieren.

4.1.3 Projekt-Dokument

Das SaveUp-Projekt-Dokument lieferte eine umfassende Grundlage für die Planung, Entwicklung und Implementierung der Anwendung. Es definierte die technischen Anforderungen, die gewünschten Funktionalitäten und die Rahmenbedingungen für das Projekt.

Inhalte des Projekt-Dokuments:

- **Projektbeschreibung**: Detaillierte Beschreibung der Ziele und des Umfangs des Projekts.
- **Anforderungen**: Spezifische funktionale und nicht-funktionale Anforderungen, die die Anwendung erfüllen musste.
- **Architektur und Design**: Übersicht über die geplante Architektur der Anwendung, einschließlich des Einsatzes von .NET MAUI und MVVM.
- **Ressourcenplanung**: Zuweisung von Aufgaben und Verantwortlichkeiten innerhalb des Entwicklungsteams.
- **Meilensteine und Zeitplan**: Festlegung wichtiger Meilensteine und Deadlines für die verschiedenen Phasen des Projekts.
- **Qualitätssicherung**: Strategien und Methoden zur Sicherstellung der Qualität und Stabilität der Anwendung.

Das Projekt-Dokument war ein entscheidendes Instrument für die strukturierte und zielgerichtete Umsetzung des Projekts SaveUp. Es bot einen klaren Fahrplan und half dabei, die Erwartungen und Anforderungen aller Beteiligten zu steuern.

5 Plannung

Das Projekt SaveUp startete letzte Woche Dienstag und folgt einem strukturierten Plan, um sicherzustellen, dass alle Aufgaben effizient und rechtzeitig abgeschlossen werden. Hier sind die detaillierten Schritte und Zeitpläne für die verschiedenen Phasen des Projekts:

5.1 Meilensteine definieren

Die Festlegung von Meilensteinen ist entscheidend, um den Fortschritt des Projekts zu überwachen und sicherzustellen, dass alle Aufgaben rechtzeitig abgeschlossen werden. Für das SaveUp-Projekt wurden die folgenden Meilensteine definiert:

1. **Erstellung der Modellklassen**: Definition der Datenmodelle für Ausgaben.
    - **Ziel**: Erstellung der grundlegenden Datenstruktur, die zur Speicherung der Ausgaben verwendet wird.
    - **Zeitplan**: 25.06.2024 - 27.06.2024
    - **Soll**: Datenmodell für Ausgaben erstellt und getestet.
2. **UI-Design und Implementierung**: Entwurf und Implementierung der Benutzeroberfläche.
    - **Ziel**: Entwicklung einer benutzerfreundlichen und intuitiven Benutzeroberfläche.
    - **Zeitplan**: 28.06.2024 - 04.07.2024
    - **Soll**: Fertigstellung des UI-Designs und Implementierung der grundlegenden Benutzeroberfläche.
3. **Integration des MVVM-Musters**: Implementierung des MVVM-Musters zur Trennung von Logik und UI.
    - **Ziel**: Sicherstellung der Trennung von Logik und Benutzeroberfläche zur Verbesserung der Wartbarkeit und Erweiterbarkeit.
    - **Zeitplan**: 05.07.2024 - 09.07.2024
    - **Soll**: MVVM-Architektur implementiert und getestet.
4. **JSON-Speicherung**: Implementierung der Speicherung und des Ladens von Daten in JSON-Dateien.
    - **Ziel**: Speicherung der Ausgaben in JSON-Dateien und Laden der Daten beim Start der Anwendung.
    - **Zeitplan**: 10.07.2024 - 14.07.2024
    - **Soll**: JSON-Speicherung und -Laden implementiert und getestet.
5. **Testen und Debuggen**: Umfangreiche Tests zur Sicherstellung der Funktionalität und Performance.
    - **Ziel**: Durchführung von manuellen und automatisierten Tests, um die Stabilität und Funktionalität der Anwendung sicherzustellen.
    - **Zeitplan**: 15.07.2024 - 19.07.2024
    - **Soll**: Alle Funktionen getestet und Fehler behoben.

5.2 Ressourcenplanung

Die Ressourcenplanung umfasst die Zuweisung von Aufgaben basierend auf den Fachkenntnissen der Teammitglieder sowie die Bereitstellung der notwendigen Werkzeuge und Umgebungen für die Entwicklung und das Testen der Anwendung.

5.2.1 Entwickler

- **David Reichert**: Projektleiter und Hauptentwickler, verantwortlich für die Gesamtarchitektur und die Implementierung des MVVM-Musters.
- **Teammitglieder**: Zuweisung spezifischer Aufgaben wie UI-Design, Implementierung der Geschäftslogik, JSON-Speicherung und Testen.

5.2.2 Testwerkzeuge

- **Manuelle Tests**: Durchführung von Usability-Tests und Überprüfung der Benutzeroberfläche durch das Entwicklungsteam.
- **Automatisierte Tests**: Implementierung von Unit-Tests zur Sicherstellung der Logik und Funktionalität der Anwendung.

5.2.3 Entwicklungsumgebung

- **Visual Studio**: Hauptentwicklungsumgebung zur Entwicklung und Debugging der Anwendung.
- **.NET MAUI**: Framework zur plattformübergreifenden Entwicklung von mobilen und Desktop-Anwendungen.
- **GitHub**: Plattform für Versionskontrolle und Zusammenarbeit im Team.

5.3 Zeitplan

Ein detaillierter Zeitplan stellt sicher, dass alle Aufgaben innerhalb des festgelegten Zeitrahmens abgeschlossen werden. Der Zeitplan für das SaveUp-Projekt ist wie folgt:

Meilenstein	Startdatum	Enddatum	Aufgaben
Erstellung der Modellklassen	25.06.2024	27.06.2024	Definition der Datenmodelle, Testen der Modelle
UI-Design und Implementierung	28.06.2024	04.07.2024	Entwurf des UI, Implementierung der Oberfläche
Integration des MVVM-Musters	05.07.2024	09.07.2024	Implementierung des MVVM, Datenbindung
JSON-Speicherung	10.07.2024	14.07.2024	Implementierung der Speicherung und des Ladens
Testen und Debuggen	15.07.2024	19.07.2024	Manuelle und automatisierte Tests, Fehlerbehebung

5.4 Soll-Ist-Vergleich

Um den Fortschritt des Projekts zu überwachen und sicherzustellen, dass alle Aufgaben rechtzeitig abgeschlossen werden, wird ein regelmäßiger Soll-Ist-Vergleich durchgeführt. Dieser Vergleich ermöglicht es, Abweichungen vom Plan frühzeitig zu erkennen und entsprechende Maßnahmen zu ergreifen.

5.4.1 Erfindung von Soll-Ist-Daten

In diesem Abschnitt werden fiktive Soll-Ist-Daten präsentiert, um den Fortschritt und die Herausforderungen des Projekts zu illustrieren.

Meilenstein	Soll-Datum	Ist-Datum	Abweichung	Maßnahmen
Erstellung der Modellklassen	27.06.2024	26.06.2024	+1 Tag	Keine Maßnahmen erforderlich
UI-Design und Implementierung	04.07.2024	05.07.2024	-1 Tag	Zusätzliche Ressourcen zugewiesen
Integration des MVVM-Musters	09.07.2024	09.07.2024	0 Tage	Planmäßige Fertigstellung
JSON-Speicherung	14.07.2024	13.07.2024	+1 Tag	Keine Maßnahmen erforderlich
Testen und Debuggen	19.07.2024	20.07.2024	-1 Tag	Zusätzliche Testtage eingeplant

5.4.2 Abweichungsanalyse

- **Erstellung der Modellklassen**: Dieser Meilenstein wurde einen Tag vor dem geplanten Datum abgeschlossen. Keine zusätzlichen Maßnahmen waren erforderlich.
- **UI-Design und Implementierung**: Dieser Meilenstein verzögerte sich um einen Tag aufgrund unerwarteter Designanpassungen. Zusätzliche Entwickler wurden zugewiesen, um die Verzögerung aufzuholen.
- **Integration des MVVM-Musters**: Dieser Meilenstein wurde planmäßig abgeschlossen. Keine Abweichungen traten auf.
- **JSON-Speicherung**: Dieser Meilenstein wurde einen Tag früher als geplant abgeschlossen. Keine zusätzlichen Maßnahmen waren erforderlich.
- **Testen und Debuggen**: Dieser Meilenstein verzögerte sich um einen Tag aufgrund zusätzlicher Fehlerbehebungen. Zusätzliche Testtage wurden eingeplant, um sicherzustellen, dass alle Funktionen ordnungsgemäß funktionieren.

Der Soll-Ist-Vergleich wird wöchentlich durchgeführt, und bei Abweichungen werden die Ursachen analysiert und Maßnahmen zur Behebung ergriffen. Dies kann die Umverteilung von Ressourcen, Anpassung des Zeitplans oder zusätzliche Unterstützung durch das Team umfassen.

5.5 Risikomanagement

Ein wesentlicher Bestandteil der Projektplanung ist das Risikomanagement. Mögliche Risiken werden identifiziert, bewertet und entsprechende Maßnahmen zur Risikominderung festgelegt.

Risiko	Wahrscheinlichkeit	Auswirkung	Maßnahmen
Verzögerungen im Zeitplan	Mittel	Hoch	Regelmäßige Fortschrittsüberwachung
Technische Probleme	Hoch	Mittel	Einplanung von Pufferzeiten
Mangel an Fachkenntnissen	Niedrig	Mittel	Schulungen und Workshops für das Team
Datenverlust	Niedrig	Hoch	Regelmäßige Backups und Versionierung
Performanceprobleme	Mittel	Mittel	Optimierung der Codebasis, Performance-Tests

6 Entscheidungen Treffen

6.1.1 Technologieauswahl

Die Auswahl der richtigen Technologien ist entscheidend für den Erfolg des Projekts. Im Fall von SaveUp wurden folgende Entscheidungen getroffen:

- **Framework**: .NET MAUI wurde aufgrund seiner Fähigkeit zur plattformübergreifenden Entwicklung ausgewählt. Dies ermöglicht die Entwicklung einer einzigen Codebasis für Android und iOS, was die Entwicklungszeit verkürzt und die Wartung erleichtert.
- **Datenformat**: JSON wurde als Datenformat für die Speicherung der Ausgaben gewählt. JSON ist leichtgewichtig, einfach zu verwenden und gut unterstützt, was die Speicherung und den Austausch von Daten vereinfacht.
- **Designmuster**: Das MVVM (Model-View-ViewModel) Muster wurde implementiert, um eine klare Trennung von Logik und Benutzeroberfläche zu gewährleisten. Dies verbessert die Wartbarkeit und Erweiterbarkeit der Anwendung.

6.1.2 Entwicklungsumgebung

Visual Studio wurde als Entwicklungsumgebung gewählt, um die Entwicklung und das Debugging der Anwendung zu erleichtern. Visual Studio bietet umfassende Unterstützung für .NET MAUI, hervorragende Debugging-Tools und eine Vielzahl von Erweiterungen, die die Entwicklung effizienter gestalten.

6.2 7. Realisierung des Projekts "SaveUp"

6.2.1 Initialisierung und Projektstruktur

Das Projekt wurde mit dem Ziel initiiert, eine benutzerfreundliche und performante mobile Anwendung zur Verwaltung von Ausgaben zu entwickeln. Die Projektstruktur umfasst Verzeichnisse für Modelle, Ansichten und ViewModels, um eine klare Trennung der Verantwortlichkeiten zu gewährleisten.

- **Modelle**: Enthält die Datenklassen, die die Ausgaben und andere Geschäftslogik repräsentieren.
- **Ansichten (Views)**: Enthält die XAML-Dateien, die die Benutzeroberfläche definieren.
- **ViewModels**: Enthält die Logik, die zwischen den Modellen und den Ansichten vermittelt, einschließlich Datenbindung und Kommandos.

6.2.2 Datenmodellierung und Integration

Die Datenmodellierung erfolgte mit Fokus auf die Anforderungen der Benutzer. Die Modelle für Ausgaben wurden entwickelt, um alle notwendigen Informationen effizient zu speichern und zu verwalten.

- **ExpenseItem**: Diese Klasse repräsentiert eine einzelne Ausgabe und enthält Eigenschaften wie Beschreibung, Preis und Datum.

Die Datenmodelle wurden so gestaltet, dass sie leicht erweiterbar sind, um zukünftige Anforderungen zu berücksichtigen.

6.2.3 Implementierung des MVVM-Musters

Das MVVM-Muster wurde implementiert, um eine klare Trennung zwischen Logik und Benutzeroberfläche zu gewährleisten.

- **Model**: Repräsentiert die Daten und Geschäftslogik.
- **View**: Repräsentiert die Benutzeroberfläche.
- **ViewModel**: Vermittelt zwischen Model und View, stellt Daten bereit und verarbeitet Benutzeraktionen.

Dies ermöglichte eine einfache Wartung und Erweiterbarkeit der Anwendung. Die ViewModels enthalten die Logik für die Datenbindung und die Verarbeitung von Benutzeraktionen, während die Views für die Darstellung der Benutzeroberfläche verantwortlich sind.

6.2.4 Sicherheitsmaßnahmen und Datenzugriff

Sicherheitsmaßnahmen wurden implementiert, um die Benutzerdaten zu schützen. Der Zugriff auf die JSON-Daten wurde optimiert, um eine hohe Performance sicherzustellen.

- **Datenverschlüsselung**: Die gespeicherten Daten in JSON-Dateien wurden verschlüsselt, um unbefugten Zugriff zu verhindern.
- **Sichere Speicherung**: Daten wurden in einem sicheren Bereich des Dateisystems gespeichert, der nur von der Anwendung zugänglich ist.
- **Benutzer-Authentifizierung**: Ein Login-System wurde implementiert, um sicherzustellen, dass nur autorisierte Benutzer Zugriff auf ihre Daten haben.

6.3 8. Kontrolle und Qualitätssicherung

6.3.1 Verwendung von Testwerkzeugen

Um die Qualität und Funktionalität der Anwendung sicherzustellen, wurden verschiedene Teststrategien implementiert:

- **Manuelle Tests**: Umfassende Tests der Benutzeroberfläche und der Funktionalitäten wurden durchgeführt, um sicherzustellen, dass die Anwendung benutzerfreundlich und intuitiv ist.
- **Automatisierte Tests**: Unit-Tests wurden implementiert, um die Logik der Anwendung zu testen. Diese Tests wurden in die Continuous Integration (CI) Pipeline integriert, um sicherzustellen, dass jede Codeänderung automatisch getestet wird.
- **Integrationstests**: Tests, die die Interaktion zwischen verschiedenen Modulen und Komponenten der Anwendung sicherstellen.

Regelmäßige Code-Reviews und Peer-Reviews wurden durchgeführt, um die Codequalität zu gewährleisten und sicherzustellen, dass alle Teammitglieder die gleichen Standards einhalten.

6.3.2 Teststrategien

- **Smoke Testing**: Schnelltests, um sicherzustellen, dass die grundlegenden Funktionen der Anwendung nach einer neuen Codeänderung funktionieren.
- **Regression Testing**: Tests, um sicherzustellen, dass neue Änderungen keine bestehenden Funktionen beeinträchtigen.
- **Performance Testing**: Tests zur Überprüfung der Leistung und Skalierbarkeit der Anwendung, insbesondere bei großen Datenmengen.

7 Auswertung und Fazit des Projekts "SaveUp"

7.1 Persönliche Reflexion

Die Arbeit am Projekt "SaveUp" ermöglichte es, wertvolle Erfahrungen in der mobilen Entwicklung mit .NET MAUI und dem MVVM-Muster zu sammeln. Die Herausforderungen und Erfolge dieses Projekts trugen zu einem umfassenden Lernprozess bei. Die Entwicklung einer plattformübergreifenden Anwendung bot Einblicke in die Optimierung und Anpassung von Anwendungen für verschiedene Geräte und Betriebssysteme.

7.2 Neue Kenntnisse und Fähigkeiten

Durch die intensive Auseinandersetzung mit .NET MAUI und MVVM konnten die Kenntnisse in der mobilen Entwicklung vertieft und erweitert werden. Die Implementierung

der JSON-Speicherung bot praktische Erfahrungen im Umgang mit Dateisystemen in mobilen Anwendungen. Wichtige Fähigkeiten, die während des Projekts entwickelt wurden, umfassen:

- **Datenbindung und Kommandos**: Effiziente Implementierung von Datenbindung und Kommandos im MVVM-Muster.
- **Plattformübergreifende Entwicklung**: Entwicklung und Optimierung einer einzigen Codebasis für mehrere Plattformen.
- **Benutzerfreundlichkeit**: Gestaltung einer intuitiven und ansprechenden Benutzeroberfläche.
- **Leistungsoptimierung**: Verbesserung der Performance der Anwendung, insbesondere bei der Handhabung großer Datenmengen.

7.3 Spaß und Motivation

Die erfolgreiche Realisierung der Projektziele führte zu großer persönlicher Zufriedenheit und verstärkte das Interesse an der weiteren Erforschung moderner Entwicklungsmethoden und Technologien. Die Zusammenarbeit im Team und die kontinuierlichen Verbesserungen der Anwendung waren besonders motivierend. Jeder Meilenstein, der erreicht wurde, bot ein Gefühl der Erfüllung und ermutigte dazu, weiterzumachen und das Beste aus den verfügbaren Technologien herauszuholen.

8 Anhänge

8.1 NuGet-Pakete

- **CommunityToolkit.Mvvm**:
- **Microsoft.Maui.Controls**:
- **Microsoft.Maui.Controls.Compatibility**:
- **Microsoft.Extensions.Logging.Debug**:
- **Newtonsoft.Json**:

8.2 Glossar

Begriff	Definition
MVVM	Model-View-ViewModel, ein Designmuster zur Trennung von Logik und Benutzeroberfläche. Es ermöglicht die Trennung der Zuständigkeiten und erleichtert die Wartung.
JSON	JavaScript Object Notation, ein leichtgewichtiges Datenformat zur Speicherung und Übertragung von Daten. Es ist menschenlesbar und wird häufig in Webanwendungen verwendet.
.NET MAUI	.NET Multi-platform App UI, ein Framework zur plattformübergreifenden Entwicklung von mobilen und Desktop-Anwendungen. Es ermöglicht die Erstellung einer einzigen Codebasis für mehrere Plattformen.
Data Binding	Mechanismus zur Synchronisation von Daten zwischen UI-Komponenten und Datenquellen. In MVVM wird es verwendet, um die Ansicht (View) an das Datenmodell (ViewModel) zu binden.
Command	Ein MVVM-Mechanismus zur Bindung von Benutzeraktionen an Logik in ViewModels. Es handelt sich um eine Implementierung des Kommando-Musters, das Aktionen kapselt und wiederverwendbar macht.
CommunityToolkit.Mvvm	Ein NuGet-Paket, das MVVM-Unterstützung durch einfache und intuitive APIs bietet. Es erleichtert die Implementierung des MVVM-Musters in .NET-Anwendungen.
Microsoft.Maui.Controls	Die Basisbibliothek für die Entwicklung plattformübergreifender Benutzeroberflächen in .NET MAUI. Sie enthält UI-Komponenten und Layouts.
Microsoft.Maui.Controls.Compatibility	Ein NuGet-Paket, das Abwärtskompatibilität für frühere Xamarin.Forms-Komponenten bietet, um die Migration auf .NET MAUI zu erleichtern.
Microsoft.Extensions.Logging.Debug	Ein Logging-Framework für .NET, das Debug-Logging während der Entwicklung unterstützt. Es hilft Entwicklern, die Anwendung zu überwachen und Fehler zu beheben.