

# **Feedback App**

**DIPLOMARBEIT**

verfasst im Rahmen der

**Reife- und Diplomprüfung**

an der

**Höheren Abteilung für Informatik**

Eingereicht von:

Mirzet Sankonjic  
Stefano Pyringer

Betreuer:

Josef Fürlinger

Leonding, September 2022

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, September 2022

M. Sankonjic & S. Pyringer

# Abstract



## Initial position

Feedback is a way for people and companies to improve themselves and their products. Especially in the age of the internet, evaluation is quick and easy. In education, it is rarely possible to give feedback, as teachers immediately move on to the next lesson. Our goal is to provide students and teachers with an easy way to do a quick assessment.

## The Application

The feedback app is based on .NET technologies. The data is stored on a server, which was developed with ASP.NET Core and enables external data access via an HTTP API. This makes an implementation of a Frontend very easy. This project focuses on a user interface for mobile operating systems Android and iOS, which was implemented with the help of Xamarin.

## Result

Our application allows teachers to give feedback to students on their teaching units in a simple and quick way.

# Zusammenfassung



## Ausgangslage

Feedback ist eine Möglichkeit für Personen und Unternehmen, sich und ihre Produkte und Dienstleistungen zu verbessern. Besonders im Zeitalter des Internets ist eine Bewertung einfach und schnell möglich. Im Bildungsbereich ist es selten möglich, ein sofortiges Feedback zu geben, da die Lehrenden sofort in die nächste Unterrichtseinheit wechseln. Unser Ziel ist es, den Schülern und Lehrenden eine einfache Möglichkeit für eine schnelle Bewertung zu bieten.

## Die Applikation

Die Feedback App basiert auf .NET Technologien. Die Daten werden auf einem Server gespeichert, der mit ASP.NET Core entwickelt wurde und mittels einem HTTP-API den Datenzugriff nach außen ermöglicht, welches die Umsetzung eines Frontend sehr flexibel macht. Dieses Projekt konzentriert sich auf eine Benutzeroberfläche für die mobilen Betriebssysteme Android und iOS, die mithilfe von Xamarin umgesetzt wurde.

## Ergebnis

Unsere Anwendung ermöglicht es Lehrenden den Schülern, auf eine einfache und schnelle Weise Feedback zu ihren Lehreinheiten zu geben.

## **Danksagung**

Wir möchten uns an dieser Stelle herzlich bei unserem Diplomarbeitsbetreuer Prof. Josef Fürlinger bedanken, dass er uns in allen Besprechungen stets mit seinem Fachwissen zur Seite gestanden ist und uns durch dieses Projekt geleitet hat. Des Weiteren möchten wir uns auch noch bei unseren Familien und Freunden bedanken welche uns, während diesem Projekt immer unterstützt haben.

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Ausgangslage . . . . .	1
1.2 Zielsetzung . . . . .	1
1.2.1 Geplantes Ergebnis . . . . .	2
1.3 Theoretischer Hintergrund . . . . .	2
1.3.1 Was ist Feedback? . . . . .	2
1.3.2 Johari-Fenster . . . . .	3
1.4 Geplantes Ergebnis . . . . .	5
1.5 Aufbau der Diplomarbeit . . . . .	5
<b>2 Systemarchitektur</b>	<b>6</b>
2.1 Übersicht . . . . .	6
2.2 Backend . . . . .	7
2.2.1 Datenstruktur . . . . .	7
2.2.2 Models . . . . .	10
2.2.3 API-Services . . . . .	10
2.3 Frontend . . . . .	11
2.3.1 Xamarin.Forms . . . . .	11
2.3.2 Anwendungsgrundlagen . . . . .	13
<b>3 Technologien</b>	<b>14</b>
3.1 .NET 6 . . . . .	14
3.1.1 Allgemeine Eigenschaften . . . . .	14
3.1.2 Sprachunabhängigkeit . . . . .	15
3.1.3 Architektur und Anwendungsbereiche . . . . .	15
3.2 ASP.NET Core . . . . .	17
3.2.1 Anwendungsbereiche . . . . .	18
3.2.2 Identity . . . . .	20

3.3	Entity Framework Core . . . . .	21
3.3.1	Überblick . . . . .	21
3.3.2	Überblick ORM . . . . .	21
3.3.3	Architektur . . . . .	22
3.3.4	Überblick wichtiger EF-Objekte . . . . .	22
3.3.5	EF-Modellieransätze . . . . .	23
3.3.6	Validierungsmöglichkeiten . . . . .	24
3.4	OpenAPI und Swagger . . . . .	26
3.4.1	Anwendungsbereiche . . . . .	27
3.4.2	Vorteile von OpenAPI . . . . .	28
3.5	JWT Token . . . . .	29
3.5.1	Aufbau . . . . .	29
3.5.2	Funktionsweise . . . . .	30
3.6	Xamarin . . . . .	33
<b>4</b>	<b>Entwicklungsumgebungen</b>	<b>35</b>
4.1	Visual Studio 2022 . . . . .	35
4.2	Visual Studio Code . . . . .	37
4.2.1	Unterschiede zu Visual Studio . . . . .	38
4.2.2	Verwendungsgrund . . . . .	38
4.2.3	Thunder Client Extension . . . . .	39
4.2.4	ERD Editor Extension . . . . .	39
4.3	Github . . . . .	40
4.4	Testen auf dem Endgerät . . . . .	42
4.4.1	Android Emulator . . . . .	42
4.4.2	iOS Emulator . . . . .	42
<b>5</b>	<b>Produktübersicht</b>	<b>44</b>
5.1	Login . . . . .	45
5.2	Startseite . . . . .	47
5.2.1	Schüler . . . . .	47
5.2.2	Lehrer . . . . .	50
5.3	Registrierung . . . . .	52
5.4	Benutzerkontoverwaltung . . . . .	54
5.5	Statistiken . . . . .	59
5.5.1	Schüler . . . . .	59

5.5.2	Lehrer . . . . .	60
<b>6</b>	<b>Umsetzung</b>	<b>63</b>
6.1	Backend . . . . .	63
6.1.1	Datenbank (Persistance) . . . . .	64
6.1.2	Web API . . . . .	66
6.2	Frontend . . . . .	70
6.2.1	Login . . . . .	70
6.2.2	Registrierung . . . . .	72
6.2.3	Benutzerkontoverwaltung . . . . .	73
6.2.4	Statistik . . . . .	75
<b>7</b>	<b>Nicht realisierte Funktionen</b>	<b>77</b>
7.1	Feedback Kategorien . . . . .	77
7.2	Export als PDF . . . . .	77
7.3	digitale Erfolge/Abzeichen . . . . .	77
7.4	Benachrichtigungen . . . . .	77
7.5	Two-Faktor Authentifizierung . . . . .	78
7.6	Benutzerkonto Profilfoto . . . . .	78
<b>8</b>	<b>Zusammenfassung</b>	<b>79</b>
<b>Literaturverzeichnis</b>		<b>VIII</b>
<b>Abbildungsverzeichnis</b>		<b>XI</b>
<b>Tabellenverzeichnis</b>		<b>XIII</b>
<b>Quellcodeverzeichnis</b>		<b>XIV</b>
<b>Anhang</b>		<b>XV</b>
		VI

# 1 Einleitung

## 1.1 Ausgangslage

Feedback ist eine Möglichkeit für Personen und Unternehmen, sich und ihre Produkte und Dienstleistungen zu verbessern. Besonders im Zeitalter des Internets ist eine Bewertung einfach und schnell möglich. Erfolgreiche Unternehmen verwenden Rezensionssysteme, um ihre Dienstleistungen nachhaltig zu verbessern.

Aufbauend auf eigene Erfahrungen gab vor diesem Projekt keine zufriedenstellende Möglichkeit, ein sofortiges ausführliches Feedback zu geben, da die Lehrenden oft sofort in die nächste Unterrichtseinheit wechseln. Zudem tauchten Probleme öfters erst später beim Lernen auf. Aktuelle Methoden sind für alle Beteiligten nicht benutzerfreundlich genug.

Eine E-Mail zu schreiben ist für die meisten Schüler zeitaufwendig und auch nicht notwendig, wenn es nur um eine Bewertung in einem 1-5 Notenschema handelt. Für den Lehrenden, der die E-Mail erhältet ist es auch schwierig zuzuordnen, zu welcher Unterrichtseinheit oder Thema die Rückmeldung sich bezieht. Zudem fehlt den Lehrenden die Möglichkeit, auf eine einfache Art und Weise, nachträglich um Feedback zu bitten.

## 1.2 Zielsetzung

Unser Ziel ist es, den Schülern und Lehrenden eine einfache Möglichkeit für eine schnelle einfache Bewertung zu bieten. Folgende Anforderungen sollte die Anwendung erfüllen:

*Schüler/Lehrerverwaltung*

- Anlegen und Bearbeitung der Benutzerkonten
- Übersicht über die Schüler

*Verwaltung der Feedback*

- Vorlesungs- und Lehrerbewertungssystem
- Schüler- und Klassenbewertungssystem
- Kommentarmöglichkeit
- Statistik in Form einer PDF teilen oder ausdrucken
- Login mit der Schul-E-Mail
- Überblick und Vergleich die Feedbacks

### 1.2.1 Geplantes Ergebnis

Unser Ziel ist es, dass mindestens 70% der Schüler der HTL-Leonding die Feedback-App benutzen werden. Es soll zu einer sichtbaren Verbesserung der Unterrichtsgestaltung kommen und auch damit zu weniger Missverständnissen und Konflikten zwischen Lehrer und Schülern kommen. Die Motivation der Schüler wird durch die verbesserten Lernbedingungen gesteigert und somit auch deren Lernerfolg. Diese Ziele definieren einen guten Erfolg dieses Projektes.

## 1.3 Theoretischer Hintergrund

[1] [2]

### 1.3.1 Was ist Feedback?

Ein Feedback ist eine offene Rückmeldung an eine Person oder Gruppe, wie ihr Verhalten von anderen wahrgenommen oder gedeutet wird. Diese Rückmeldung dient dazu, voneinander zu lernen. Denn Menschen lernen besonders gut, wenn sie die Ergebnisse ihrer Handlung sehen. Das Feedback macht diese Ergebnisse transparenter. Zudem hilft das Feedback, die Beziehungen zwischen den Personen zu klären, um die andere Person besser zu verstehen.

Rückmeldungen korrigieren die Verhaltensweisen, die der betroffenen Person und der Gruppe nicht weiterhelfen oder die der eigentlichen Intention nicht genügend angepasst oder konform sind. Erst nach Erhalt dieser Informationen ist der Feedback-Empfänger in der Lage, wie er sein Verhalten verändern könnte, damit das von ihm beabsichtigte

Ergebnis eintritt. Wichtig dabei ist, dass nur die betroffene Person über die Verhaltensänderung entscheidet und nicht der Feedbackgeber. Feedbacks werden auch dazu verwendet, positive Ergebnisse mitzuteilen, was dem Empfänger bestätigt, dass sein Handeln richtig ist. Im Falle eines Fehlverhaltens wird das Feedback-Gespräch als Kritikgespräch bezeichnet.

Beim Feedback geben sollte man sich an diese Empfehlungen halten, damit eine Rückmeldung wirksam und lehrreich ist:

- zuerst die positiven, dann die negativen Beobachtungen besprechen und mit einer positiven Rückmeldung enden (Sandwich-Feedback)
- Konkrete Situationen und Verhaltensweisen beschreiben
- Subjektive Satzformulierungen, z. B. "Ich hatte den Eindruck ..."'
- Eigene Interpretationen und Verallgemeinerungen vermeiden
- Nach einer negativen Rückmeldung konstruktive Verbesserungsvorschläge machen
- Das Feedback möglichst zeitnah geben

Der Feedback-Empfänger sollte beachten, dass:

- Feedback eine Chance zur individuellen Weiterentwicklung ist.
- den Gesprächspartner ausreden lässt.
- man zu seinen Schwächen stehen soll.
- Missverständnisse durch Nachfragen vermieden werden.
- man das Feedback auf sich wirken lassen sollte, bevor man sich dazu äußert oder Stellung nimmt.
- man nach dem Feedback darüber nachdenken sollte, welche Wünsche der anderen angenommen werden sollte und welche nicht.

### 1.3.2 Johari-Fenster

Die Persönlichkeitsmerkmale können unterschiedlich von Menschen oder von einem selbst wahrgenommen werden. Das Johari-Fenster stellt die Unterschiede zwischen Selbst- und Fremdwahrnehmung dar. Es wurde 1955 von den amerikanischen Sozialpsychologen Joseph Luft und Harry Ingham entwickelt. Mithilfe des Feedbacks soll der öffentliche Bereich der Person erweitert werden.

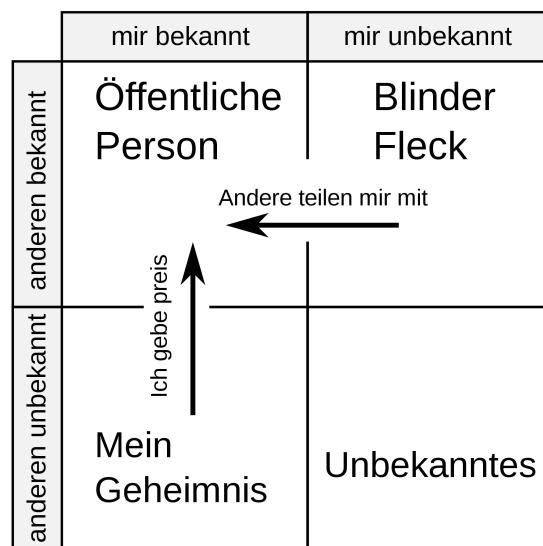


Abbildung 1: Johari-Fenster [3]

### Öffentliche Person

In diesem Feld versammeln sich Merkmale, die der Person selbst und anderen Personen bekannt sind. Das Handeln der Person ist frei und unbeeinträchtigt von Ängsten und Vorbehalten.

### Offenkundige Person - blinder Fleck

Hier finden sich Persönlichkeitsmerkmale, die der eigenen Person kaum bewusst ist, jedoch von anderen deutlich wahrgenommen werden können. Diese Merkmale sind sichtbar durch unbewusste Gewohnheiten und Verhaltensweisen, Vorurteile und persönliche Zu- und Abneigungen.

### Verbogene Person

Diese Persönlichkeitsmerkmale werden von der Person bewusst verborgen sind nur dieser bekannt. Dies können heimliche Wünsche oder wunde Punkte sein.

### Unbewusste Person

In diesem Feld sind die Merkmale der Person sich selbst und anderen Personen verborgen. Beispiel dafür sind verborgene Talente oder ungenützte Begabungen.

## 1.4 Geplantes Ergebnis

## 1.5 Aufbau der Diplomarbeit

### 2. Systemarchitektur

Als erstes wird das Datenmodel, Struktur und Kommunikation der Applikation erklärt.

### 3. Technologien

In diesem Kapitel werden die verwendeten Technologien erklärt.

### 4. Entwicklungsumgebungen

Hier werden die verwendeten Werkzeuge für die Entwicklung der Feedback App erklärt und wieso sie für dieses Projekt ausgewählt wurden.

### 5. Produktübersicht

In der Produktübersicht werden die einzelnen Views der Benutzeroberfläche der mobilen Anwendung präsentiert und ihre Funktionen erklärt.

### 6. Umsetzung

In diesem Kapitel wird die Umsetzung dargelegt, wie die Diplomarbeit realisiert wurde.

### 7. Nicht realisierte Funktionen

Zum Abschluss werden die nicht umgesetzten Funktionen der App kurz beschrieben.

# 2 Systemarchitektur

## 2.1 Übersicht

Diese Abbildung bietet einen groben Überblick über die Technologien, die für die Erstellung und Betrieb der Feedback App verwendet wurden. Für die Umsetzung der Applikation wurde die Programmiersprache C# verwendet. Die Technologien werden in Kapitel 3 ausführlich erklärt.

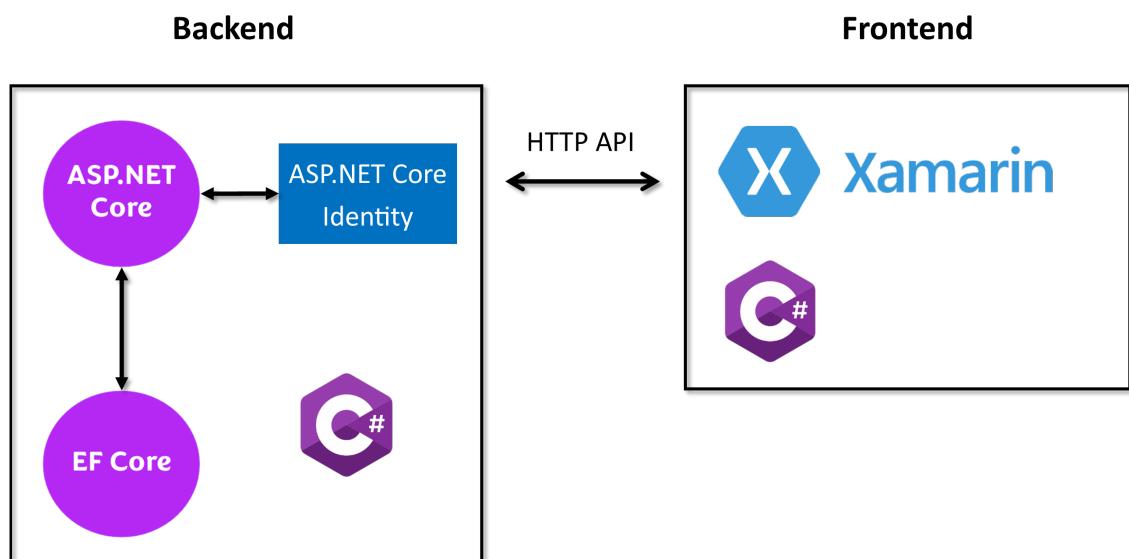


Abbildung 2: Systemarchitektur Überblick [4] [5]

### Backend

- .NET 6
  - – ASP.NET Core
    - Entity Framework Core
    - ASP.NET Core Identity

Für das Backend wurde die Software-Entwicklungsplattform .NET 6 verwendet, um Datenstruktur, Repositories und Authentifizierung zu realisieren. Es wird mittels HTTP-API mit dem Frontend kommuniziert.

## Frontend

Das Frontend handhabt die Interaktionen des Benutzers und stellt die von der API zur Verfügung gestellten Daten dar. Der Client ist auf mobilen Endgeräten mit dem Betriebssystem Android und iOS lauffähig. Das Frontend wurde mit Xamarin realisiert.

## 2.2 Backend

### 2.2.1 Datenstruktur

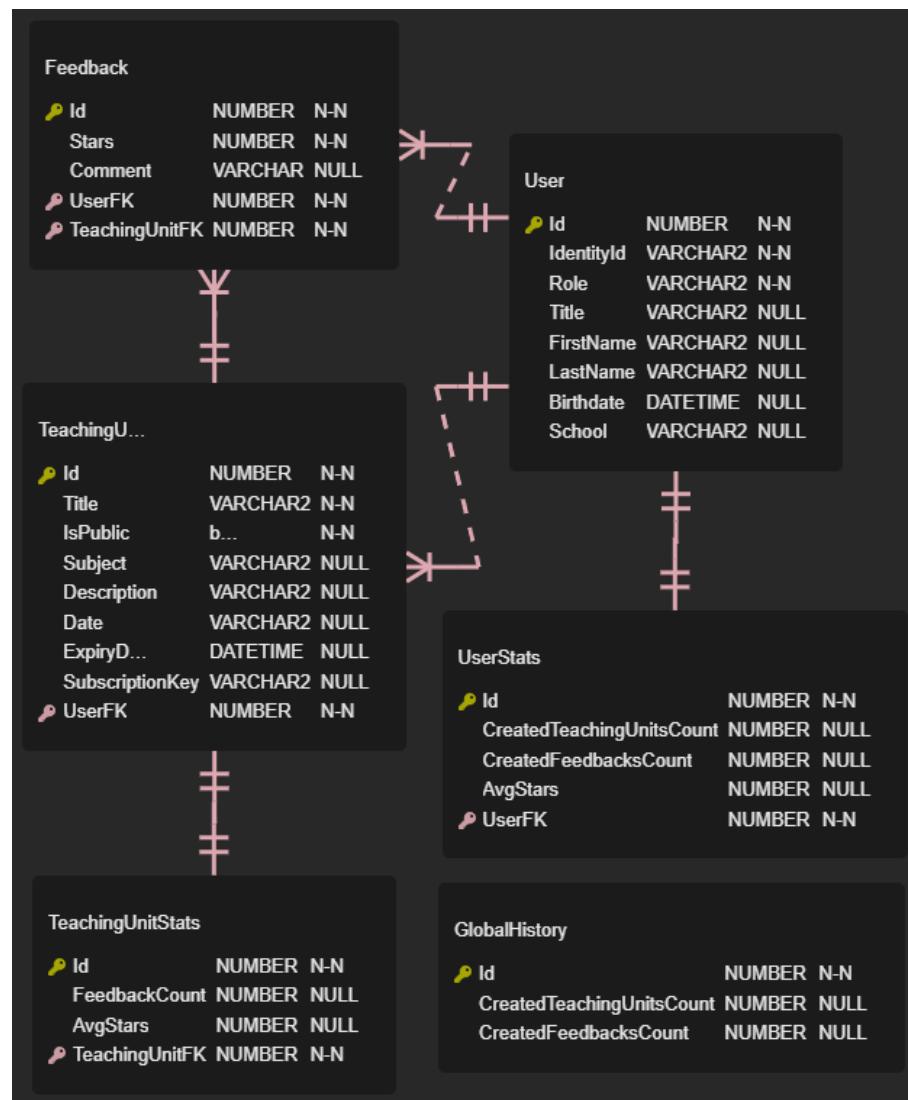


Abbildung 3: FeedbackDb Entity Relationship Diagramm

**Tabelle Feedback**

Diese Tabelle beschreibt das Feedback für eine Lehreinheit. Der User kann die Lehreinheit in Sternen (1-5) bewerten und einen Kommentar hinzufügen.

**Tabelle User**

Beinhaltet alle zusätzlichen personenbezogene Daten des Benutzers und die Rolle. Der Fremdschlüssel IdentityId verbindet die Tabelle 1:1 zu der User-Tabelle der Identity-Datenbank. Der User kann mehrere Feedbacks und Lehreinheiten erstellen. Je Benutzer ist eine Statistiktable verbunden.

**Tabelle TeachingUnit**

Es ist die Hauptkomponente der gesamten Datenbank. Hier werden alle notwendigen Daten gespeichert, die für das Erstellen einer Lehreinheit notwendig sind. Sie ist verbunden mit mehreren Feedbacks von anderen Usern und ist mit einer Tabelle für die Statistiken der Lehreinheit verbunden.

**Tabelle TeachingUnitStats**

Die Statistik für eine Lehreinheit speichert die Anzahl der gegebenen Feedbacks und die durchschnittliche Bewertungszahl einer Lehreinheit.

**Tabelle UserStats**

Diese Tabelle beinhaltet Statistiken für einen User. Es werden die Anzahl der erstellten Lehreinheiten und Bewertungen mit deren durchschnittlichen Bewertungszahl erfasst.

**Tabelle GlobalHistory**

Die globale Statistik Tabelle zählt alle jemals erstellten Lehreinheiten und Bewertungen des Servers an. Diese sind öffentlich zugänglich. D. h. es ist keine Autorisierung notwendig.

## ASP.NET CORE Identity

Die Daten für die ASP.NET Core Identity Authentifizierungsschnittstelle ist in der SarathalDb realisiert. Die Tabellen entsprechen dem vorgefertigten Schemas des NuGet-Package.

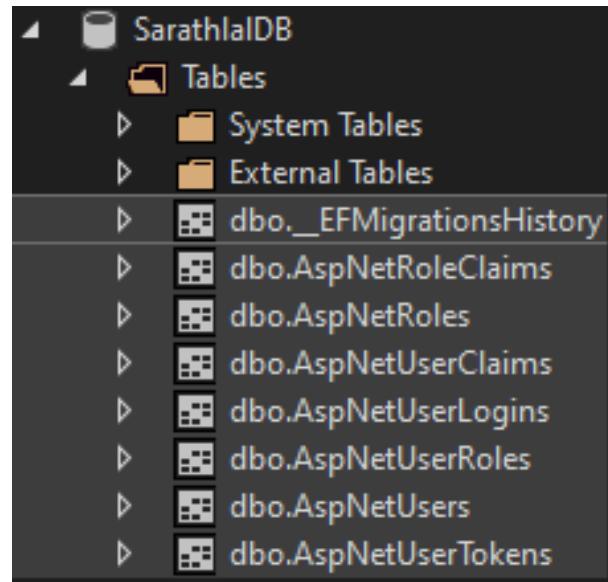


Abbildung 4: Tabellen der Idendity Datenbank

## 2.2.2 Models

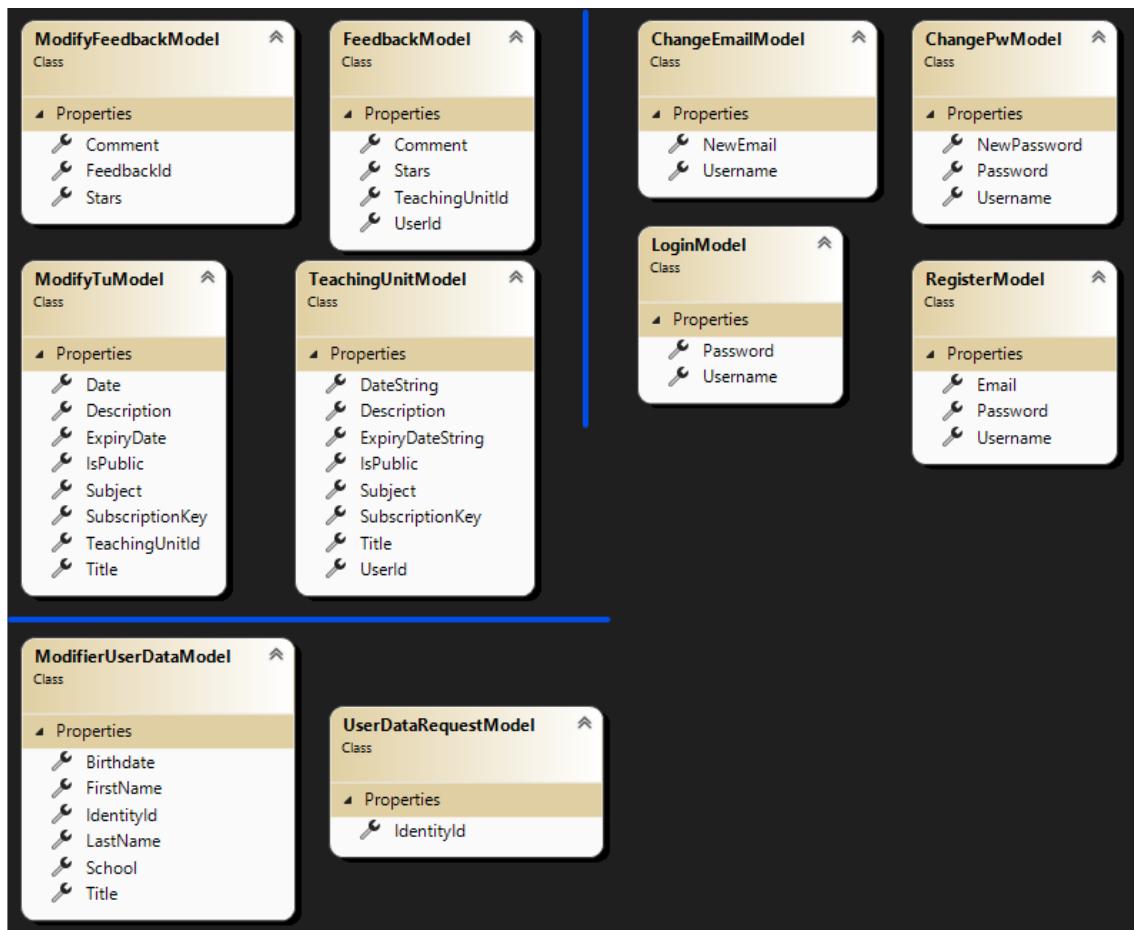


Abbildung 5: Model Feedback Web-API | links oben: Feedback, links unten: Account-Management, rechts: Feedback

Diese Models dienen als Schnittstelle für den Zugriff aus der HTTP-API zum Controller. Die Models definieren, welche Felder im HTTP-Body einer HTTP-Request haben soll, um von der Methode angenommen zu werden. Zudem wird eine gewisse Sicherheit gewährleistet, da nur die definierten Felder erstellt oder bearbeitet werden dürfen.

## 2.2.3 API-Services

Das Backend stellt einen API-Service für das Frontend zur Verfügung. Mittel HTTP-Requests können Daten empfangen und gesendet werden. Die Autorisierung wird mittels Anhang eines JWT-Tokens realisiert. Die API-Methoden werden mit OpenAPI Swagger beschrieben und vereinfachte so die Entwicklung der HTTP-Zugriffsmethoden für das Frontend. Eine genaue Beschreibung der Umsetzung ist in Kapitel 6.1 zu finden.

## 2.3 Frontend

### 2.3.1 Xamarin.Forms

In Visual Studio wird Code in Projektmappen und Projekten organisiert. Eine Projektmappe ist ein Container, der wenigstens ein Projekt beinhaltet. Ein Projekt kann z.B. eine Applikation, eine unterstützende Bibliothek oder eine Testanwendung sein. Die Notes-Anwendung besteht wie im nächsten Screenshot gezeigt aus einer Projektmappe mit drei Projekten:

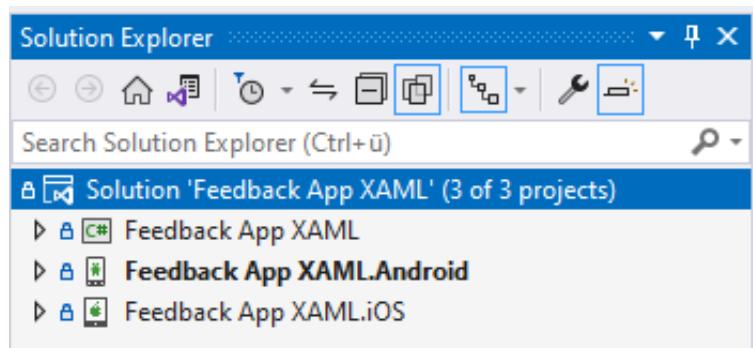


Abbildung 6: Xamarin.Forms Project Ansicht

- Feedback App XAML: Dieses Projekt ist das .NET Standard-Klassenbibliotheksprojekt, das den vollständigen freigegebenen Code und die komplette freigegebene Benutzeroberfläche beinhaltet.
- Feedback App XAML.Android: Dieses Projekt beinhaltet Android-spezifischen Code und ist der Einstiegspunkt für die Android-Anwendung.
- Feedback App XAML.iOS: Dieses Projekt beinhaltet iOS-spezifischen Code und ist der Einstiegspunkt für die iOS-Anwendung.

Diese Projekte sind folgende:

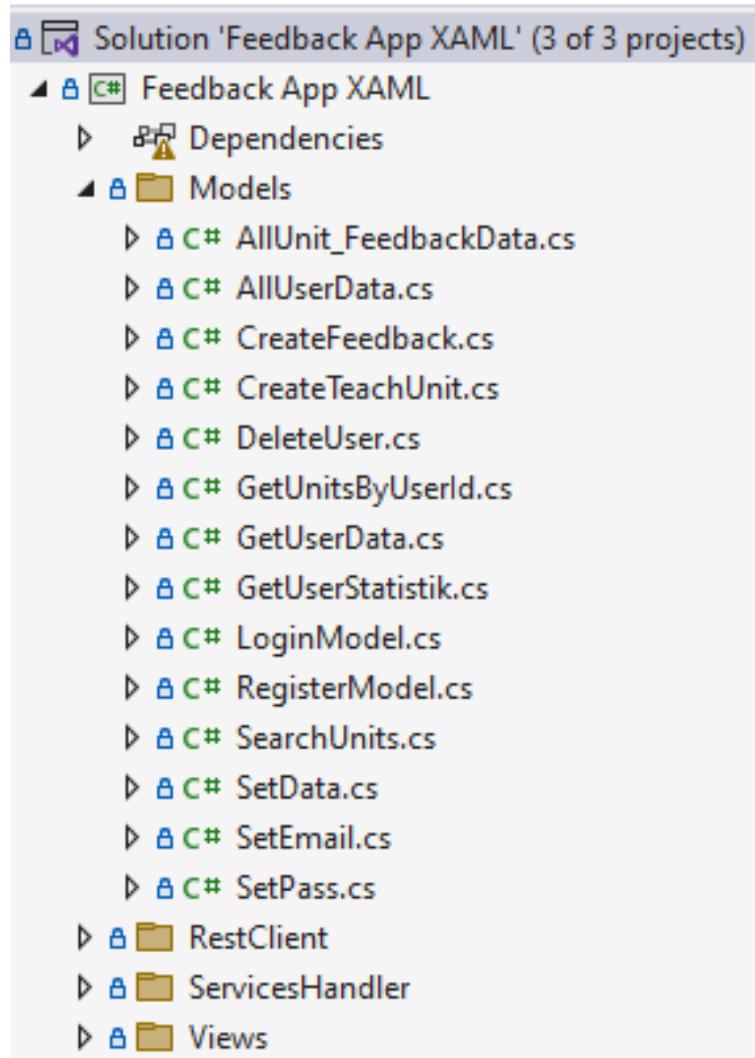


Abbildung 7: Project Ansicht

Das Projekt verfügt über den Knoten Abhängigkeiten, der die Knoten NuGet und SDK beinhaltet:

- NuGet: die dem Projekt hinzugefügten NuGet-Pakete für Xamarin.Forms, Xamarin.Essentials, Newtonsoft.Json und sqlite-net-pcl.
- SDK: Das NETStandard.Library-Metapaket, das alle NuGet-Pakete referenziert, die .NET Standard definieren.

### 2.3.2 Anwendungsgrundlagen

Eine Xamarin.Forms-Anwendung ist gleichwohl aufgebaut wie eine herkömmliche plattformübergreifende Applikation. Freigegebener Code wird meistens in einer .NET Standard-Bibliothek platziert und von plattformspezifischen Apps genutzt. Die sich anschließende Abbildung offeriert für die Notes-Anwendung einen Überblick über ebendiese Beziehung:

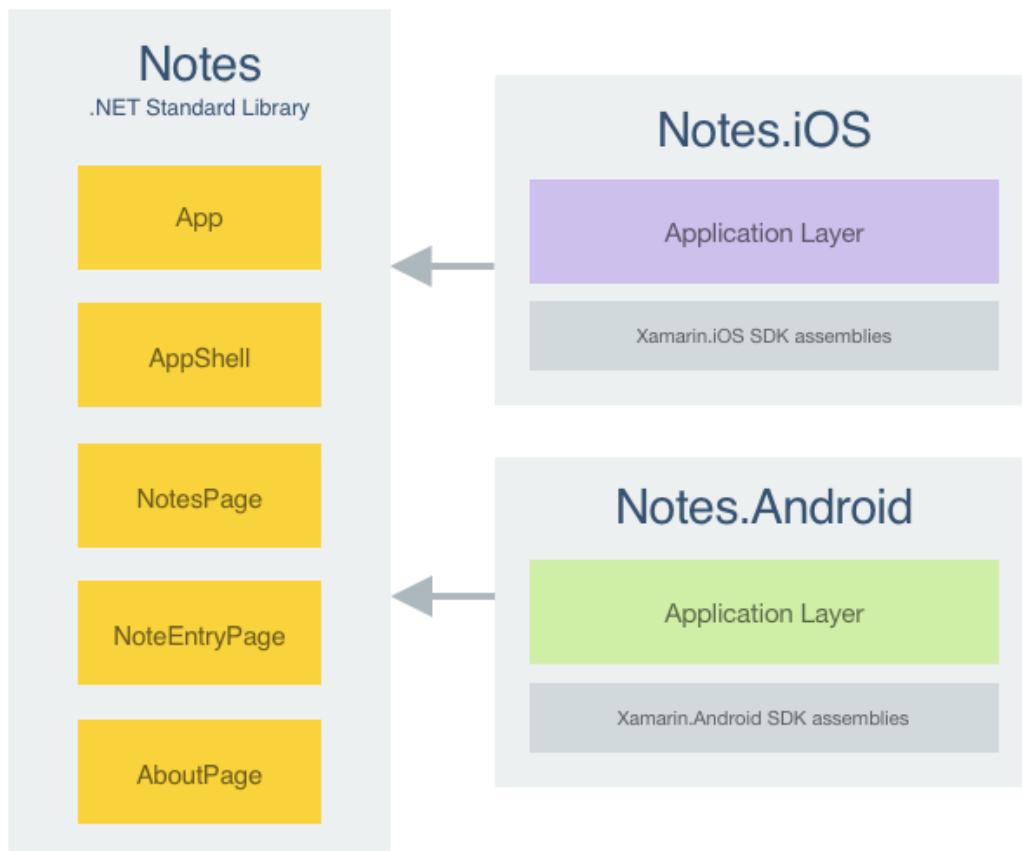


Abbildung 8: Anwendungsgrundlagen

# 3 Technologien

## 3.1 .NET 6

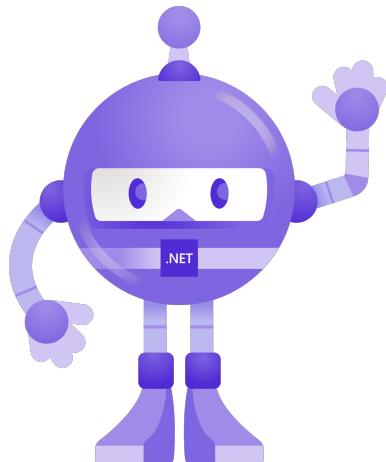


Abbildung 9: Community Maskottchen von .NET [6]

[7] [8] [9]

Das .NET wird für die Entwicklung und Ausführung von Anwendungsprogrammen verwendet und ist Teil der .NET-Plattform. Sie ist mit einer MIT-Lizenz lizenziert und ist somit eine freie open-source Software. Das .NET-Framework wurde von Microsoft 2016 unter den Namen .NET Core modernisiert. .NET ist der Nachfolger von .NET Core seit Dezember 2020 und ist in der Version 6.0.7 verfügbar (Stand Juli 2022).

### 3.1.1 Allgemeine Eigenschaften

Microsoft die Konzeptideen von Java, aufgrund des hohen Erfolgs der Sprache, übernommen und versucht, die bekannten Schwachstellen der Sprache auszumerzen und dadurch die Messlatte mit .NET spürbar höher gelegt.

Folgende Eigenschaften bringt .NET mit sich:

#### Objektorientierung

.NET ist komplett objektbasiert. Auch einfache Datentypen wie Integer werden als Objekte behandelt.

## WinAPI-32-Ersatz

Microsoft will langfristig die Win32-API durch Klassen des .NET-Frameworks ersetzen. Alle Sprachen greifen daher auf die gleiche Bibliothek zurück. Dadurch verwischen die charakteristischen Merkmale der verschiedenen Sprachen. Die Wahl einer Sprache im .NET ist nicht mehr mit den Entscheidungen gleichzusetzen, wie effizient eine Anwendung geschrieben werden kann oder was sie zu leisten imstande ist.

## Plattformunabhängigkeit

Auf .NET basierenden Anwendungen laufen in einer Umgebung, die mit der virtuellen Maschine von Java verglichen werden kann. Erst zur Laufzeit der Anwendung wird der Maschinencode generiert. Aufgrund der öffentlichen Dokumentation der Spezifikation von der Common Language Runtime (CLR) lässt sich sie sich auf Plattformen portieren wie Unix oder Linux. Quelloffene Beispiele sind Mono oder CoreCLR.

### 3.1.2 Sprachunabhängigkeit

Eine in C# geschriebene Klasse kann aus jeder anderen .NET-kompatiblen Sprache wie F# aufgerufen werden, ohne die Verwendung über eine spezifizierte Schnittstellenlentechnologie wie COM/COM+ gehen zu müssen. Beispielweise lässt sich eine C# implementierte Klasse aus einer VB.NET-Klasse ableiten und auch umgekehrt ist dies möglich.

### 3.1.3 Architektur und Anwendungsbereiche

Das Software-Development-Kit (SDK) unterstützt Windows (ab 7, 32/64 Bit und Arm), macOS (ab 10.12), Linux-Distributionen(64 Bit, Arm) und für Docker oder Snappy existieren offizielle Images. NET besteht aus 2 Hauptkomponenten CoreCLR und CoreFX. Sie sind vergleichbar mit der Common Language Runtime (CLR) und der Framework Class Library (FCL) von NET Framework's Common Language Infrastructure (CLI). NET unterstützt

Folgende Entwicklungsumgebungen werden von der SDK unterstützt:

- Visual Studio (ab 2022 auch für macOS)
- Visual Studio Code

- per Kommandozeile mit dem .NET SDK
- JetBrains Rider

Standardmäßig können für die Entwicklung von .NET Apps die Programmiersprachen C#, F# oder Visual Basic verwendet werden. .NET kann in der Funktionalität mithilfe von NuGet-Packages erweitert werden. Folgende Anwendungsgebiete unterstützt .NET 6:

- Web-Apps und Micro-Services (ASP.NET Core)
- Kommandozeilen Programme
- Klassenbibliotheken
- GUI-Applikationen für Windows (UWP, WPF) und Cross-Plattform-Apps (.NET MAUI, Xamarin)
- Machine Learning (ML.NET, Apache Spark for .NET)
- Game Development (Unity, Cryengine, MonoGame, etc.)
- Internet of Things

## 3.2 ASP.NET Core

[10] [11] [12]

ASP.NET Core ist ein modulares Open-Source Web-Framework für die Entwicklung von modernen Web-Anwendungen. Es wurde von Microsoft entwickelt und ist der Nachfolger von ASP.NET seit .NET 5. Das Web-Framework ist ein Bestandteil von .NET. ASP.NET Core wurde von Grund auf neu entwickelt und unterscheidet sich wesentlich von Vorgänger Versionen.

Die wichtigsten Vorteile zu ASP.NET sind:

- Entwicklung von Web-UI und Web-APIs in einer einheitlichen Umgebung
- Open-Source und Community freundlich
- Plattformunabhängigkeit (Windows, MacOS, Linux)
- Moderne Entwicklungstools
- Leichtgewichtige, modulare und leistungsstarke HTTP-Request Pipeline
- Umfassende Testmöglichkeiten
- Dependency Injection
- Razor Pages und Blazor ermöglichen dynamische Webseiten mit .NET Programmiersprachen
- Cloud kompatibel
- Mehrere Hosting Möglichkeiten:
  - Kestrel
  - IIS
  - HTTP.sys
  - Nginx
  - Apache
  - Docker

### 3.2.1 Anwendungsgebiete

#### Interaktive clientseitige Web-Apps mit Blazor/Razor View Engine

Mithilfe von Blazor ist das Erstellen von clientseitigen Webbenutzeroberflächen möglich. Zudem bietet die Razor View Engine die Möglichkeit, .NET-Sprachblöcke in HTML-Seiten einzubetten. Die UI wird anschließend als HTML und CSS gerendert inklusive umfassender Browserunterstützung (auch mobile Browser). Mit der Technik des Client-Side-Renderings werden die Skripte im Browser ausgeführt und verarbeitet.

#### Web-APIs

Mit ASP.NET Core können RESTful-basierte Webservices (HTTP-API) entwickelt werden. Die Services werden von Controllern gesteuert und diese unterstützen CRUD-Datenoperationen (Create, Read, Update, Delete). Die Schnittstellen können mit Swagger/Open-API dokumentiert werden.

#### Web-Apps und APIs mit MVC

ASP.NET Core MVC ist ein zusätzliches umfassendes Framework basierend auf dem Model-View-Controller-Entwurfsmusters. Damit werden die einzelnen Bereiche getrennt, was die Entwicklung, Organisation und Testbarkeit erleichtert. ASP.NET Core MVC wird parallel zu Razor Pages als Alternative weiterhin unterstützt.

#### Web-Apps und APIs mit Razor Pages

Razor Pages ist der Nachfolger von MVC in ASP.NET Core und ersetzt ASP.NET Web Forms. Sie basiert auf dem MVC-Framework, jedoch ist die Komplexität reduziert. Razor Pages verwendet das MVVM-Entwurfsmuster (Model-View-View-Model). Somit entfällt der Controller.

#### Echtzeit-Web-Apps mithilfe von SignalR

SignalR ist eine Open-Source-Bibliothek mithilfe die Hinzufügung von Echtzeitwebfunktionen zu Apps zu vereinfachen. Diese Funktionen ermöglichen serverseitigen Code, Inhalte sofort an die Clients zu senden. SignalR verarbeitet die Verbindungsverwaltung automatisch.

SignalR ist besonders für Szenarien geeignet, wo man Daten vom Server mit einer hohen Aktualisierungsfrequenz benötigt. Solche Kandidaten können sein:

- Dashboard oder Überwachungsdienste wie Sofortupdates wie Reisehinweise oder Börsenwerte.
- Soziale Netzwerke-Apps mit Chatroom-Funktion
- Applikationen, die wiederholt in kürzester Zeit Daten benötigen (z. B. Gaming, GPS, Voting, Auktionen, etc.)

## **gRPC-Dienste**

RPC steht für Remote Procedure Call und eignet sich dafür, Mikroservices miteinander kommunizieren zu lassen. Das dRPC-Framework ist sprachunabhängig und hochleistungsfähig.

Es eignet sich besonders für folgende Szenarien:

- Simple Mikroservices, wo Effizienz wichtig ist
- Mehrsprachige Systeme
- Point-to-Point-Dienste, die in Echtzeit Streaminganforderungen oder -antworten verarbeiten müssen.

## **Datengesteuerte Web-Apps**

Erstellung von Web-Applikationen, die mit Datenbankensysteme zusammenarbeiten. Daten können im Browser mittels CRUD-Operationen verändert werden, ohne auf eine spezielle Software, wie den Oracle SQL-Explorer, zuzugreifen.

## **Hybrid-App Development**

Verschiedene Technologien können miteinander kombiniert werden. So können alle vorherigen genannten Anwendungsmöglichkeiten mit ASP.NET Core in einer App verwendet werden. Es ist auch ein ist auch eine Verschmelzung von Benutzeroberflächen mit .NET MAUI, WPF und Windows Forms möglich. Diese laufen nicht im Browser, sondern in einer eigenen Anwendung dargestellt, ohne die z. B. existierende Oberfläche neu für die anzuwendende UI zu programmieren.

### 3.2.2 Identity

[13] [14]

ASP.NET Core Identity ist eine API, dass das Benutzermanagement (Login-Interface) ermöglicht. Die Benutzer können einen Account erstellen und mit diesen einloggen. Die API verwendet dafür eine SQL-Datenbank, die die Benutzernamen, Rollen, Passwörter, E-Mail Adressen und weitere benutzerbezogene Daten verwaltet. Als Alternative ist die Speicherung der Datenbank auf externen Datenbanksystemen wie Azure möglich.

Ein externer Login von Login-Providern wie Google, Microsoft, Twitter und Facebook wird von ASP.NET Core Identity ebenfalls unterstützt. Der Identity Source Code ist auf GitHub erhältlich.

Eine erweiterte Version von ASP.NET Core Identity ist IdentityServer4, welche ein OpenID Connect und OAuth 2.0 Framework bietet. Zudem sind folgende erweiterte Sicherheitsfeatures möglich:

- Authentication as a Service (AaaS)
- Single sign-on/off (SSO) Zugangsverfahren für Multi-Anwendungsszenarien
- Zugriffskontrolle für APIs
- Federation Gateway

## 3.3 Entity Framework Core

### 3.3.1 Überblick

[7] [15] [16]

Das Entity Framework (EF) ist ein moderner Objektdatenbank-Mapper für .NET-Objektstrukturen. Es wurde von Microsoft entwickelt und die erste Version erschien als Teil des .NET Framework 3.5 im Jahr 2008. Damals gehörte es noch zu ADO.NET, welches seit 2002 existiert. Es ist als Ergänzung zu verstehen, der die Differenzen zwischen der objektorientierten Programmierung und relationalen Datenbanken adressiert.

Im Jahr 2021 mit der Version 5 machte Microsoft das Framework quelloffen verfügbar und hieß Entity Framework. Mit .NET Core gibt es seit 2016 das Framework als separates Zusatzpaket Entity Framework Core (EF Core). Die aktuellste Version ist EF Core 6.0 (Stand Juli 2022) und erhält Langzeit-Support (LTS) bis 8. November 2024.

Es unterstützt LINQ-Abfragen, Migrationen und Änderungsnachverfolgung (Tracking). EF-Core unterstützt viele Datenbanken wie SQL-Datenbanken (lokal und Azure), SQLite, MySQL, PostgreSQL und Azure Cosmos DB.

### 3.3.2 Überblick ORM

Objektrelationale Abbildung (engl. Object-relational mapping) kann mit einer objektorientierten Programmiersprache Objekte (Klassen) in einer relationalen Datenbank abgelegt werden. ORM ist somit eine Lösung für die Kommunikation zwischen Objektcode und relationaler Datenbank.

Zu den Vorteilen von ORM zählen neben der Datenzugriffstechnik auch:

- Vereinfachte Entwicklung, da die Konvertierung von Objekt und Tabelle und umgekehrt automatisiert wird, was zu geringeren Entwicklungs- und Wartungskosten führt.
- Weniger Code im Vergleich zu Embedded SQL und handgeschriebenen gespeicherten Prozeduren
- Transparente Objektzwischenspeicherung in der Anwendungsebene zur Verbesserung der Systemleistung

- Einfache und schnelle Wartung der Anwendung

Diese Technik bringt auch Nachteile mit sich:

- Langsamere Performance im Gegensatz zu gespeicherten Prozeduren
- ORM-Abhängigkeit kann unter bestimmten Umständen zu schlecht gestalteten Datenbanken führen

### 3.3.3 Architektur

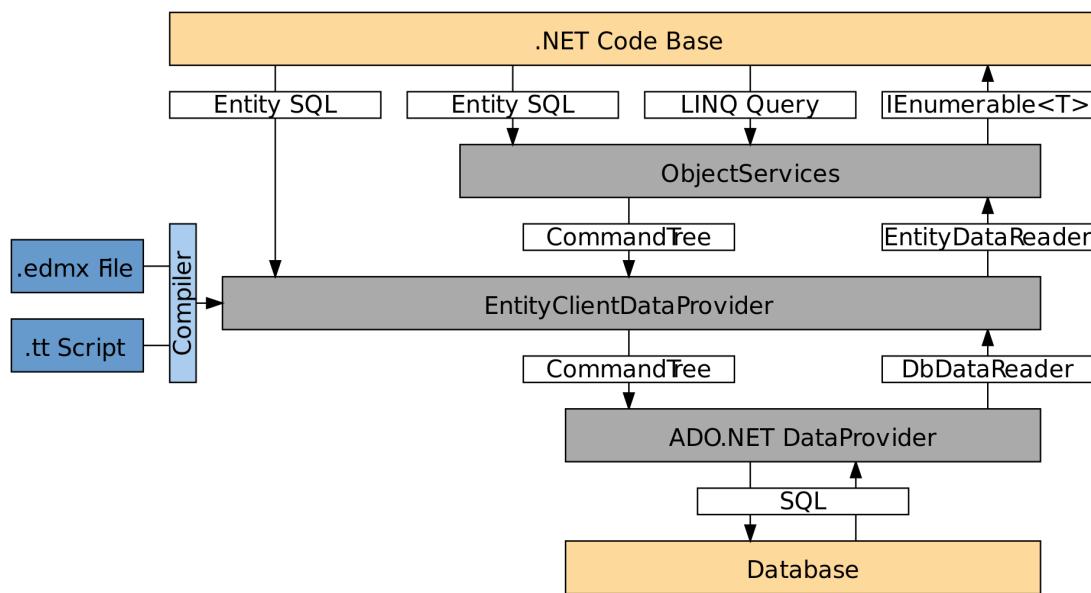


Abbildung 10: Prinzipielle Funktionsweise des ADO.NET Entity Framework [17]

### 3.3.4 Überblick wichtiger EF-Objekte

#### DbContext

Stellt eine Verbindung mit der Datenbank dar. Stellt Methoden für Abfragen (Query), Änderungsverfolgung (Tracking) und Speichern (Save) bereit.

#### DbQuery

Stellt Methoden für das Hinzufügen (Add), Anhängen (Attach) und Entfernen (Remove) von Entitäten bereit.

## DbSet

Erbt von DbQuery und stellt die entsprechenden Methoden für Entity-Typen bereit.

## Change Tracker API

Bietet Methoden an, um Änderungen verfolgen zu können.

## Validation API

Führt eine automatische Validierung der Daten im DataLayer durch.

## Code First Model Building

Erstellt eine Datenbank mithilfe von Code-basierten Klassen

### 3.3.5 EF-Modellieransätze

Für das Entity Framework gibt es 2 Modellieransätze, nämlich Code First und Model First. Ersteres setzt auf die Erzeugung einer Datenbankstruktur mit C# Code. Model First verwendet im Gegensatz Designer und Assistenten für die Erstellung der Datenbankstruktur.

Daraus ergeben sich folgende Szenarien:

#### Code First - keine Datenbank vorhanden

Bestehende Klassen (MVC-Model) werden mit Annotationen ausgezeichnet, welche die Abbildung auf eine Datenbank steuern. Darauf abbauend werden vom DbContext die Datenbank und die Datenbank-Tabellen modelliert und beim Aufruf der SaveChanges-Methode erstellt.

#### Model First - keine Datenbank vorhanden

Die Entity-Klassen werden mit einem grafischen Designer modelliert. Dieses Modell wird mit Hilfe des Text Template Transformation Toolkit (T4) und der dazugehörigen Skriptsprache in Entity-Klassen umgewandelt. Der Designer erstellt zusätzlich ein SQL-Skript, mit dem die Datenbank erstellt wird.

### **Code First - Verwendung einer bestehenden Datenbank**

Die Entity-Klassen können entsprechend der vorgegebenen Datenbank manuell erstellt werden. Dies ist jedoch sehr arbeitsintensiv.

### **Model First - Verwendung einer bestehenden Datenbank**

Mit Hilfe eines Assistenten wird die Datenbank abgefragt und entsprechend der Datenbankstruktur ein passendes Modell erstellt. Dieses wird mit einem T4-Script in die entsprechenden Klassen umgewandelt.

## **3.3.6 Validierungsmöglichkeiten**

Das Entity Framework bietet eine Vielzahl von Validierungsfeatures. Dieses Konzept gilt für mehrere Ebenen:

- Entity direkt → Domainspezifisch
- Persistenzschicht → Datenbankabhängig
- Web-Ui clientseitig → Validierung im Browser
- WPF → Validierung am Desktop

Die Validierungen können unterschiedlich implementiert werden:

### **Feld-Ebene**

Es werden Data-Annotationen in den Model-Felder verwendet. Diese sind in EF Core schon vorgefertigt. Es können aber auch eigene Annotationen(-klassen) erstellt werden (Custom Validation).

### **Objekt-Ebene**

Sie wird empfohlen, wenn die Validierung nicht genau einem Feld zugeordnet werden kann. Es wird die Methode Validate benutzt, wenn Entity IValidatableObject implementiert wird. Es können mehrere Validierungsfehler definiert werden.

### **Context-Ebene**

Sie ist objektübergreifend (mehrere Objekte) und benutzt die Validation unter Verwendung der Daten aus der Datenbank. Es wird verwendet, wenn eine Validation unter Berücksichtigung mehrerer oder aller Objekte notwendig ist (z. B. Unique-Constraint).

## 3.4 OpenAPI und Swagger

[18] [19] [20]

Die OpenAPI Spezifikation, auch als Swagger-Spezifikation bekannt, ist ein Standard für die Beschreibung von REST-Schnittstellen. Es war ursprünglich ein Bestandteil des Open-Source-Frameworks Swagger, welches HTTP-Webservices bereitstellt. Seit 2016 ist es ein eigenständiges Projekt, das von der OpenAPI Initiative verwaltet und gefördert wird. Mitglieder dieser Initiative sind große Unternehmen wie z. B. Microsoft, Google, IBM, Paypal und SAP sowie weitere Initiative wie Smartbear und die Linux Foundation. Ziel ist es, ein offenes und herstellerneutrales Beschreibungsformat für API-Dienste bereitzustellen. Bei den API-Spezifikationen kommen meistens die 2 Sprachen YAML oder JSON zum Einsatz. Die aktuellste der OpenApi-Spezifikation ist 3.1 (Stand Juli 2022).

Der OpenApi-Standard definiert eine Reihe von Eigenschaften, die auch als sogenannte Objekte zusammengefasst werden können. Folgende Objekte liegen in einer Dokumentation vor:

- Info Objekt: Version und Name der API
- Contact Info: Kontaktinformationen des API-Anbieters
- License Object: Unter welcher Lizenz die API lizenziert ist
- Server Object: Hostnamen, URL-Struktur und Ports der Server
- Components Object: gekapselte Objekte, die sich in einer API-Definition mehrfach verwenden lassen
- Paths Objects: relative Pfade zu Endpunkten der API, die mit dem Server Object gemeinsam genutzt werden
- Path Item Object: für einen spezifischen Pfad wie GET, PUT, POST, DELETE, etc.
- Operation Object: legt unter anderem die Parameter und die zu erwartenden Server-Responses für eine Operation fest

Da es um eine nur um eine technische Spezifikation handelt, ist es nicht an eine spezielle technische Implementierung gebunden. Dafür gibt es Tools wie Swagger. Swagger ist ein Sammlung von Open-Source-Werkzeugen, die helfen, die API-Dokumentation zu entwickeln.

- Swagger Editor: ein Browser-basierter Editor, wo die OpenAPI-Dokumentation geschrieben werden kann
- Swagger UI: erzeugt die OpenAPI-Dokumentation
- Swagger Codegen: generiert die Server Stubs und Client SDKs

Es existieren auch kostenpflichtige Tools wie SwaggerHub Enterprise, Swagger Inspector und API-Tree. Des Weiteren gibt es verschiedene Entwicklungsumgebungen und Erweiterungen zur Unterstützung der OpenAPI-Dokumentation. Nicht jede beliebige API lässt sich mittels OpenAPI abbilden. REST-APIs werden vollständig unterstützt.

### 3.4.1 Anwendungsgebiete

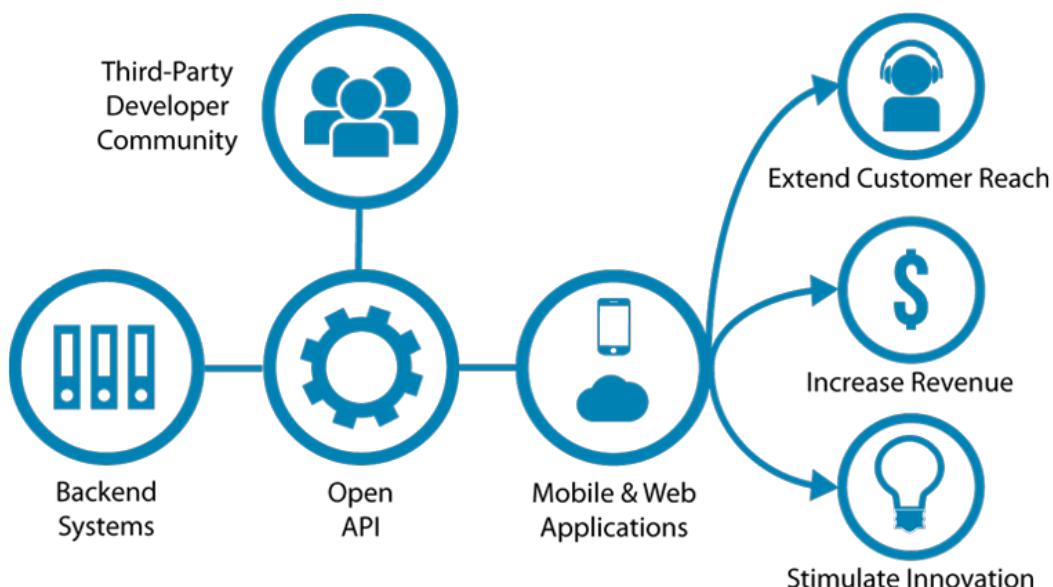


Abbildung 11: OpenAPI Business Diagramm [21]

Generell wird die OpenAPI eingesetzt, um REST-APIs auf eine einheitliche Weise zu beschreiben. Da diese Definition in einem maschinenlesbaren Format vorliegen, können automatisiert folgende Möglichkeiten erzeugt werden.

### Erzeugung der API-Dokumentation

Es kann aus den Code der API-Definition eine HTML-basierte Dokumentation erzeugt werden. Diese dient als Nachschlagewerk für jene Entwickler, die auf die APIs zugreifen wollen. Ändert sich die API-Definition, so wird die Dokumentation automatisch neu erzeugt.

## Erzeugung von Anbindungen in verschiedenen Programmiersprachen

Aus API-Definition können mit den passenden Tools eine Client-seitige Softwarebibliothek in einer unterstützten Programmiersprache erzeugt werden. Diese werden ganz normal eingebunden und die Zugriffe auf die API-Diensten erfolgen z. B. über Funktionsaufrufe innerhalb der gewohnten Entwicklungsumgebung. Damit werden Entwickler mit unterschiedlichen Programmiersprachen der Zugriff auf die API ermöglicht.

## Erzeugung von Testfällen

Aus den API-Defintion hat man die Möglichkeit, Testfälle zu definieren. Diese Test werden bei jeder Änderung der API automatisch durchgeführt, um sicherzustellen, dass jede Komponente der Software ordnungsgemäß funktioniert.

### 3.4.2 Vorteile von OpenAPI

Die Entwicklung und Pflege einer API wird in Einklang gehalten. Sie erlaubt es, dass die Koordination der API-Entwicklung zwischen Backend und Frontend verbessert wird. Beide Seiten können aus der API-Definition Code-Komponenten erzeugen lassen, so dass entwickelt und getestet werden kann, ohne auf die jeweils andere Seite warten zu müssen. Zudem ist es eine standardisierte Basis für die Entwicklung und Dokumentation von APIs.

Außerdem ergeben sich aus der Nutzung von OpenAPI weitere Vorteile:

- HTTP-APIs werden unabhängig von einer spezifischen Programmiersprache definiert
- Server-Code für eine OpenAPI definierte API generieren
- Client-Bibliotheken für eine OpenAPI-konforme API in mehr als 40 Programmiersprachen generieren
- OpenAPI-Defintionen mit geeigneten Tools verarbeiten
- Eine interaktive API-Dokumentation erstellen
- Auf API-Services mit minimalen Implementationsaufwand zugreifen
- Services verstehen, ohne dafür Einsicht in den Quelltext oder weiteren zusätzlichen Dokumentation nehmen zu müssen

## 3.5 JWT Token

[22] [23]

Der JSON Web Token (JWT) ist ein offener Standard (RFC 7519), mit dem JSON-Objekte zwischen den Kommunikationspartner mit dem JSON-Access Token Verfahren sicher und kompakt ausgetauscht werden können. JWTs sind digital signiert und können 2 verschiedene kryptographische Verfahren benutzen:

- Symmetrisches Verfahren mit HMAC
- Asymmetrische Verfahren mit RSA oder ECDSA

Der Token überträgt Claims, die die notwendigen Informationen für die Authentifizierung einer Entität enthalten. Dieser Token kann z. B. im HTTP-Header oder in der URL übertragen werden. JSON Web Tokens eignen sich für Authentifizierungsvorgänge zustandsloser Sessions und kommen im Webumfeld beispielsweise für Single Sign-on Anwendungen (SSO) zum Einsatz. Sie sind eine Alternative zu Session Cookies.

### 3.5.1 Aufbau

JWTs sind Zeichenstrings, die aus 3 Teilen bestehen, die mit einem Punkt (.) getrennt werden:

- Header
- Payload
- Signature

Der Token könnte in folgender Form dargestellt werden: *Header.Payload.Signature*

#### Header

Der Header beschreibt, welcher Typ der Token hat und welcher Signierungsalgorithmus verwendet wurde. Bei JSON Web Tokens ist immer der Typ als *JWT* angegeben.

#### Payload

Die Payload beinhaltet die Claims. Claims beinhalten Informationen über die Entität (den User) und zusätzliche Daten. Claim Bezeichnungen werden stets nur mit 3 Buch-

staben lang, da JWT ein kompakter Standard ist. Es existieren 3 verschiedene Arten von Claims.

**Registered Claims** sind ein Set von vordefinierten Claims, welche empfohlen werden für ein funktionelle verwendbare Claims. Diese können beispielweise *iss* (issuer), *exp* (expiration time), *sub* (subject) oder *aud* (audience) sein. Die vorgeschlagenen Claims müssen nicht verwendet werden.

**Public Claims** können von den Benutzern den Belieben nach vereinbart werden. Sie sollten in der IANA JSON Web Token Registry definiert sein um Kollisionen zu vermeiden. Als Alternative können sie auch als URI definiert sein, welche eine Kollisionsvermeidung für Namespaces hat.

**Private Claims** sind selbsterstellte Claims, die Informationen beinhalten, auf die sich die benutzenden Parteien geeinigt haben und weder den *registered* oder den *public* Claims zuzuordnen sind.

### **Signature**

Die Signatur wird dazu benutzt, um festzustellen, dass die Nachricht nicht verändert oder gefälscht wurde. Zudem kann ermittelt werden, wer der Absender ist. Die Signatur ist die Verschlüsselung vom Header, Payload, Passwort und im Header angegebene Verschlüsselungsmethode.

### **Erstellter Token**

Das Ergebnis sind 3 Base64-URL Strings die mit einem Punkt getrennt werden. Diese Strings sind platzsparend in HTML und HTTP Umgebungen im Gegensatz zu XML basierten Standards wie SAML. Der jwt.io Debugger kann JSON Web Tokens decodieren, was nützlich ist zum Debuggen. Da der JWT Token durch den Debugger alle Informationen einsehbar sind, sollten keine sensiblen Daten im Token angefügt werden.

### **3.5.2 Funktionsweise**

Wenn auf geschützte Ressourcen zugegriffen wird, so muss der Client (User) ein JSON Web Token senden, welcher im Idealfall im Authorizations-Header ist, welches das

The screenshot shows the jwt.io Debugger interface. At the top, there's a navigation bar with links for Debugger, Libraries, Introduction, Ask, and a Crafted by auth0 logo. Below the navigation, there's a dropdown menu for Algorithm set to HS256.

**Encoded:** PASTE A TOKEN HERE  
The input field contains a long base64-encoded JWT token:  
`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiJmNWUwZGI2ZC0xYWExLTQzY2UtOGNi0S1kYmN10TQ2YmJmNzYiLCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWR1bnRpdHkvY2xhaW1zL3JvbGUI0iJzdHVkZW50IiwiZXhwIjoxNjYwODE2OTY0LCJpc3Mi0iJodHRw0i8vbG9jYWxob3N00jYxOTU1IiwiYXVKIjoiaHR0cDovL2xvY2FsaG9zdDo0MjAwIn0.S7KyWDsnKkdzEILAiBXAezRaftw9DCQIKNifs79c1Ss`

**Decoded:** EDIT THE PAYLOAD AND SECRET  
The decoded token is shown in JSON format:

```
{
  "alg": "HS256",
  "typ": "JWT"
}

{
  "jti": "f5e0db6d-1aa1-43ce-8cb9-dbce946bbf76",
  "http://schemas.microsoft.com/ws/2008/06/identity/claim
s/role": "student",
  "exp": 1660816964,
  "iss": "http://localhost:61955",
  "aud": "http://localhost:4200"
}
```

**VERIFY SIGNATURE**  
The verification code is displayed:

```
 HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) □ secret base64 encoded
```

Abbildung 12: Beispiel JWT im jwt.io Debugger

Bearer-Schema verwendet. Der Token beinhaltet grob die Daten, auf welche Bereiche der API ein Benutzer zugreifen darf.

JWT Token, die im HTTP-Header gesendet werden sollen, sollten nicht zu groß sein, da einige Server Dateigrößen von mehr als 8 KB im Header nicht akzeptieren. Sollte dennoch mehr Informationen notwendig sein, so sollte eine Alternative wie *Auth0 Fine-Grained-Authorization* verwendet werden.

Folgendes Diagramm zeigt den Ablauf für eine Anforderung einer geschützten Ressource:

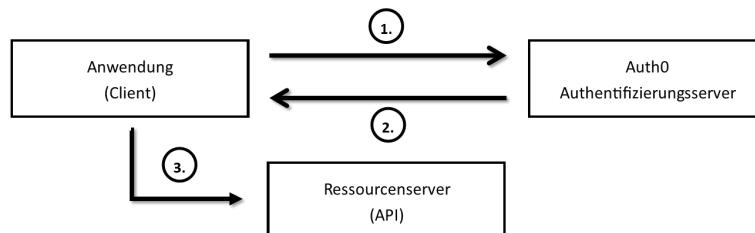


Abbildung 13: Diagramm JWT Authentifizierungsablauf

1. Die Anwendung sendet einen Authentifizierungs-Request an den Authentifizierungsserver. Diese führt einen bestimmten *authorization-code-flow* durch (z. B. OpenID)

2. Bei einer erfolgreichen Authentifizierung sendet der Authentisierungsserver ein Zugriffstoken zurück an den Client.
3. Die Anwendung kann mit dem erhaltenen Zugriffstoken auf die geschützte Resourcen zugreifen (z. B. API).

## 3.6 Xamarin

[24]



Abbildung 14: Xamarin Logo [25]

### Erklärung

Im Jahr 2011 starteten die ehemaligen Mono-Entwickler das Unternehmen namens Xamarin, um auf dem aufkommenden Markt der mobilen Betriebssysteme eine plattformübergreifende Entwicklungsumgebung zu schaffen. Mono ist eine weitere, quelloffene Implementierung von Microsofts .NET Framework. Sie erlaubt die Änderung von plattformunabhängiger Software auf den Direktiven der Common Language Infrastructure und der Programmiersprache C#. Das gleichnamige, ein Jahr später vorgestellte Produkt ermöglichte den Einsatz der Programmiersprache C-Sharp zur Entwicklung von plattformübergreifenden Anwendungen für Windows, aber auch für MacOS. Ab der Version 2.0, die 2013 vorgestellt wurde, kamen als Zielplattformen die mobilen Betriebssysteme iOS und Android dazu, so dass nun aus Visual Studio heraus Programme für MacOS, iOS und Android entwickelt werden konnten. Xamarin ist eine Open-Source-Plattform zum Aufbau moderner und leistungsfähiger Applikationen für das Betriebssystem Apples (iOs), Android und Windows mit .NET. Xamarin ist eine Abstraktionsschicht, die die Verständigung von freigegebenem Code mit dem zugrunde liegenden Plattformcode verwaltet. Xamarin wird in einer verwalteten Umgebung ausgeführt, die Annehmlichkeiten wie Speicherzuweisung und Garbage Collection offeriert.

### Funktionalität

Xamarin realisiert es Entwicklern, durchschnittlich 90 Prozent ihrer App plattformübergreifend miteinander zu nutzen. Dieses Muster erlaubt es Entwicklern, ihre restlose Geschäftslogik in einer einzigen Sprache zu schreiben (oder vorhandenen Anwendungscode wiederzuverwenden), allerdings auf jedweder Plattform native Leistung, Look und Verhalten zu erzielen.

## Begründung und Verwendung

Xamarin ist für Entwickler mit den folgenden Zielen:

1. Code, Test und Geschäftslogik plattformübergreifend teilen
2. Plattformübergreifende Anwendungen in C# mit Visual Studio schreiben

Es wurde für unsere Arbeit ausgewählt, weil es den zusätzlichen Tools der Visual Studio-Software durchaus nahe kommt und zudem für uns kostenlos nutzbar ist. Die Enterprise-Lizenz wird von der HTL bereitgestellt und darf nur für schulische Zwecke verwendet werden.

# 4 Entwicklungsumgebungen

## 4.1 Visual Studio 2022

[26]

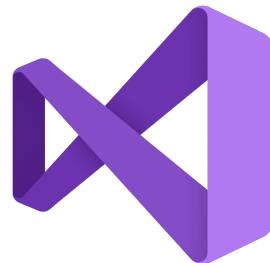


Abbildung 15: Visual Studio 2022 Logo [27]

### Erklärung

Visual Studio Enterprise 2022 ist eine integrierte Entwicklungsumgebung für verschiedene Hochsprachen von "Microsoft" welche im Jahr 2022 (Juni) veröffentlicht wurde. Visual Studio ermöglicht es Programmierern, sowohl Win32/Win64-Programme als auch Anwendungen für das .NET Framework zu entwickeln. Darüber hinaus lassen sich mit Visual Studio Windows-Apps, dynamische Webseiten bzw. Webservices für das Internet/Intranet oder Azure-Services entwickeln.

### Funktionalität

Visual Studio ist eine sehr umfangreiche und komfortable Entwicklungsumgebung. Sie lässt sich gezielt auf die Anforderungen von Projekten anpassen. Mit dem VS-Installer können zusätzliche Hochsprachen installiert oder deinstalliert werden. Neben der Erweiterbarkeit stellt Visual Studio einen integrierten Debugger zur Verfügung. Dieser enthält die Funktion „Bearbeiten und Fortfahren“ und erlaubt das nachträgliche Anhängen an bereits laufende Prozesse, sowohl am lokalen Rechner als auch über das Netzwerk. Neben dem Debugger wird der Softwareentwickler durch eine gute IntelliSense unterstützt.

## Begründung und Verwendung

Visual Studio ist die etablierteste Entwicklungsumgebung auf dem Markt, um .NET zu programmieren, da es sich durch seinen Umfang und die gute Bedienbarkeit auszeichnet. Aufgrund der vielen Funktion, der Marktposition und der Tatsache, dass sich Visual Studio bereits in vergangenen Projekten bewährt hat, wurde es für unsere Arbeit gewählt. Die Enterprise Lizenz wurde von der HTL zur Verfügung gestellt und darf ausschließlich für schulische Zwecke eingesetzt werden.

## 4.2 Visual Studio Code

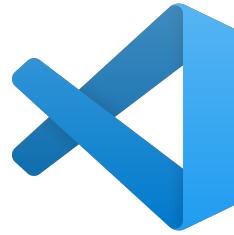


Abbildung 16: Visual Studio Code Logo [28]

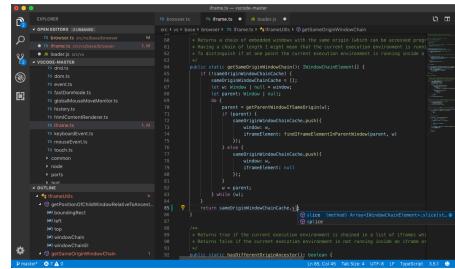


Abbildung 17: Visual Studio Code Screenshot [29]

[30] [31] [32]

Visual Studio Code, wird auch als VS Code bezeichnet, ist ein Code-Editor entwickelt von Microsoft. Seit November 2015 ist der Source-Code des Code-Editors auf GitHub verfügbar. Im April 2016 wurde die erste finale Version von Visual Studio Code veröffentlicht. Der Source-Code des Code-Editors ist mit einer MIT-Lizenz lizenziert und ist somit Open-Source. Das installierbare Programm hingegen enthält Microsoft-Binaries und ist deshalb keine Open-Source. Das Programm ist kostenlos verfügbar.

Der Texteditor basiert auf dem App-Framework Electron und läuft daher in der Chromium-Engine. Standardmäßig werden ohne zusätzliche Installationen die gängigen Programmier- und Scriptsprachen wie HTML, JavaScript, SQL, JSON, C, etc. unterstützt. Folgende Basisfunktionen besitzt das Entwickler-Tool:

- Syntaxhervorhebung
- Auto vervollständigung
- Code-Folding
- Debugging
- Versionsverwaltung
- Konfigurierbare Code-Snippets
- IntelliSense für TypeScript, JSON, CSS und HTML

- Extension-Support

### 4.2.1 Unterschiede zu Visual Studio

Entwicklungsumgebungen (IDEs) wie Visual Studio sind meistens für ein Betriebssystem entwickelt worden und funktionieren gar nicht oder nur eingeschränkt auf anderen Betriebssystemen. Die Auswahl von Programmiersprachen und Frameworks ist beschränkt. IDEs haben üblicherweise einen hohen Speicher- und Ressourcenbedarf.

Visual Studio Code hingegen ist ein Texteditor, der in der Chromium-Engine läuft und ist für alle 3 gängigen Betriebssysteme Windows, MacOs und Linux verfügbar. Aufgrund des niedrigen Ressourcenbedarfs ist es auch auf schwächeren oder älteren Computern und Laptops benutzbar. Ein großer Unterschied zu Visual Studio ist die Dateiveraltung. In Entwicklungsumgebungen wird ein Projekt in Projektdateien zusammengefasst, währenddessen der Code-Editor mit Workspaces arbeitet. Diese speichern den Bearbeitungszustand, Reihenfolge und Zeilenposition der geöffneten Dateien. In Visual Studio Code wird ein Workspace geöffnet, indem ein Ordner ausgewählt wird.

Ein Texteditor kann den Funktionsumfang und Komfort einer Entwicklungsumgebung nicht mithalten. Visual Studio Code kann dank Plug-ins, werden auch als Extensions bezeichnet, in der Funktionalität stark erweitert werden. Ein großer Vorteil zu Visual Studio ist der Extension-Marketplace, die von einer großen Gemeinschaft von Entwicklern gepflegt wird. Dank einer Vielzahl von Plug-ins ist fast jede beliebige Sprache mit umfangreichen Hilfsfunktionen wie IntelliSense oder Snippets verfügbar.

### 4.2.2 Verwendungsgrund

Visual Studio Code wurde für dieses Projekt für die Testung der HTML-Schnittstellen (APIs) des Programms und für die Planung der Datenbankobjekte verwendet. Es wurden im Unterricht bereits gute Erfahrungen mit dem Code-Editor gemacht. Ein großer Vorteil ist der sparsame Verbrauch von Ressourcen, da das Projekt großteils mit Laptops entwickelt wurde und der große Funktionsumfang von bekannten Entwicklerwerkzeugen nicht benötigt wurde.

Es wurden folgende Plug-ins verwendet:

- Thunder Client
- ERD Editor

### 4.2.3 Thunder Client Extension

[33]

Die Thunder Client Extension ist eine leichtgewichtete REST-API, die sich durch eine einfach zu bedienende GUI auszeichnet. Sie unterstützt API Collections, Environment Variablen und kommt auch mit großen Anfragen klar. Des Weiteren können die API Anfragen ohne Script mithilfe der GUI getestet werden. Sie ist eine schnelle und einfache Alternative zu Postman.

### 4.2.4 ERD Editor Extension

[34]

Der ERD Editor ist eine Extension für Visual Studio Code, mit der die Planung von Datenbankobjekten und deren Beziehungen ermöglicht wird. Mithilfe dieses Plug-ins sind verschiedene grafische Darstellungen des Entity Relation Diagramms möglich. Ein Code-Generator kann die geplanten Datenbankobjekte in SQL-DDL-Code umwandeln. Die Extension ist eine Alternative zu Planungswerkzeugen, die z. B. im Oracle SQL Developer enthalten sind.

## 4.3 Github

[35]



Abbildung 18: GitHub Logo [36]

### Erklärung

GitHub ist die primäre Plattform für Entwickler, um ihre Software zu hosten und zu verwalten. wurde 2008 gegründet und 2018 von Microsoft gekauft. Da GitHub dennoch zu einer wichtigen Anlaufstelle geworden ist, kann es in dieser Weise lange Zeit als soziales Netzwerk für Entwickler verstanden werden. Jeder hat dort sein eigenes Profil, einige arbeiten nur an Open-Source-Software und werden von Sendeanstalten finanziert. Andere führen ihre privaten Projekte nach der Arbeit durch oder unterstützen größere Projekte nur zum Spaß.

### Funktionalität

Um ein Programm, eine Website oder ähnliches bearbeiten oder durchsuchen zu können, muss das Repository oder Verzeichnis öffentlich sein oder Sie müssen dazu eingeladen werden. Und wenn dies nicht der Fall ist, kann nur der Ersteller des Repositorys daran arbeiten. An einem Verzeichnis können beliebig viele Entwickler arbeiten. Um einen Beitrag leisten zu können, müssen Sie das Repository forken, was bedeutet, dass Sie eine Kopie auf Ihrem eigenen Konto mit den aktuellen Daten erstellen. Wenn der Ersteller des ursprünglichen Repositorys dies später zulässt, können die beiden Repositorys wieder zusammengeführt werden. Dies wird als Zusammenführen bezeichnet. Sie können auch einfach dem Repository oder Entwickler folgen. GitHub wird von Microsoft betreut.

## Begründung und Verwendung

GitHub ist die seriöseste und bekannteste Plattform auf dem Markt für Teamprojekte und -arbeiten, da sie sich durch Geschwindigkeit und Benutzerfreundlichkeit auszeichnet. Es wurde aufgrund seiner zahlreichen Funktionen und der Kompatibilität mit Visual Studio für unsere Arbeit ausgewählt. Es gibt keine Lizenz und es kann kostenlos für jeden Zweck verwendet werden.

## 4.4 Testen auf dem Endgerät

### 4.4.1 Android Emulator

[37] In Visual Studio 2022 wird der Android x86-Emulator verwendet, der die Anwendung in einer simulierten Android Umgebung ausführen und debuggen kann. Dank Hyper-V Unterstützung wird die virtuelle Maschine beschleunigt. Der Emulator kann verschiedenste Sensoren wie GPS, Bewegungssensoren, Kamera, etc. simulieren.

Im Visual Studio Emulator können verschiedenste Geräteprofile ausgewählt werden, um die Kompatibilität der zu entwickelnden App zu gewährleisten. Diese Profile beinhalten verschiedene Hersteller und Android Versionen.

Der Emulator lässt sich auch mit ADB (Android Debug Bridge) verbinden, sodass andere beliebte Android Entwicklungstools wie Eclipse und Android Studio problemlos auf den Emulator zugreifen können.

#### Verwendungsgrund

Mit dem Visual Studio Android Emulator kann die Anwendung unabhängig eines echten Android-Smartphones getestet werden. Da der Emulator bereits in Visual Studio 2022 mit dem Instalationspaket Xamarin enthalten ist, ist die Instalation und Inbetriebnahme des Emulators einfacher als eine separate Installation. Zudem ist das Debugging bereits in Visual Studio integriert.

### 4.4.2 iOS Emulator

[38] [39]

#### Erklärung

Der Emulator kann in andersartigen Konfigurationen ausgeführt werden, um andersartige Apparate zu schauspielern. Jede Justierung wird als virtuelles Endgerät bezeichnet. Wenn Sie Ihre Application auf einem Emulator bereitstellen und prüfen, wählen Sie ein vorkonfiguriertes oder benutzerdefiniertes virtuelles Device aus, das ein physisches iOS-Gerät wie ein iPhone-Telefon simuliert.

## Funktionalität

Mit dem Remote-iOS-Simulator für Windows können Sie Ihre Applikationen auf einem iOS-Simulator prüfen, der unter Windows mit Visual Studio 2022 angezeigt wird. Der Remote-iOS-Simulator für Windows wird von selbst als Teil der Arbeitsauslastung der plattformübergreifenden .NET-Anwendungs-UI-Entwicklung in Visual Studio 2022 eingerichtet. Die Einrichtung besteht aus einigen Schritten. Mit Xamarin.Mac können Sie ohne Ausnahme native Mac-Anwendungen in C-Sharp und .NET erzeugen. Da Xamarin.Mac schnell in Xcode eingebettet ist, können Entwickler den Interface Builder von Xcode verwenden, um Anwendungsbenuzeroberflächen zu anlegen (optional kann dies genauso einfach in C-Sharp-Code erfolgen). Auf jene Weise war es ausgeprägt schneller, problemloser und effektiver, die iOS-Anwendung durch das Apple-Betriebssystems zu prüfen.

## Begründung und Verwendung

Xamarin iOS Emulator ist für das Testen mobiler Applikationen unverzichtbar, da es längst als Teil der Xamarin-Plattform eingerichtet ist. Das Testen geschieht gleichzeitig, was es außergewöhnlich nützlich macht. Wir haben uns für den iOs-Emulator entschieden, weil es wahrscheinlich ist, auf zwei differenzierten Plattformen (Windows und iOs) zu begutachten. Es existieren keine Lizenz, trotz alledem es ist Teil der Xamarin-Plattform.

## 5 Produktübersicht

Die Produktübersicht zeigt die einzelnen Ansichten der Website und erläutert deren Funktionen. Die mobile Version wird für jedes Display (Android oder iOS) verglichen, um zu zeigen, dass die Feedback-Anwendung auf allen derzeit gängigen Endgeräten funktionsfähig und nutzbar ist. Für die mobile Version wurde das iPhone 12 Pro ausgewählt, da es sich um ein Smartphone mit sehr großem Marktanteil handelt.

## 5.1 Login

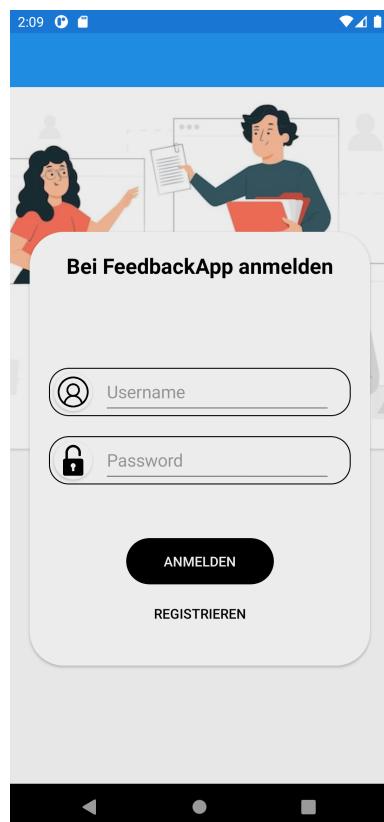


Abbildung 19: Login Ansicht

Durch den Aufruf der Anwendung gelangt der Nutzer auf die Anmeldeseite von Feedback. Die Anmeldeseite ist schwarz-weiß, schlicht und übersichtlich gestaltet. Im Login Menü E-Mail-Adresse und Passwort eingegeben, welche in der Datenbank hinterlegt sind. Button "Anmelden" überprüft unsere eingegebenen Werte, und wenn sie korrekt sind, gelangen wir auf die Startseite. Falls wir noch keine Daten eingegeben und ein Konto zum Anmelden erstellt haben, finden Sie unten die Schaltfläche zum Registrieren neuer Benutzer.

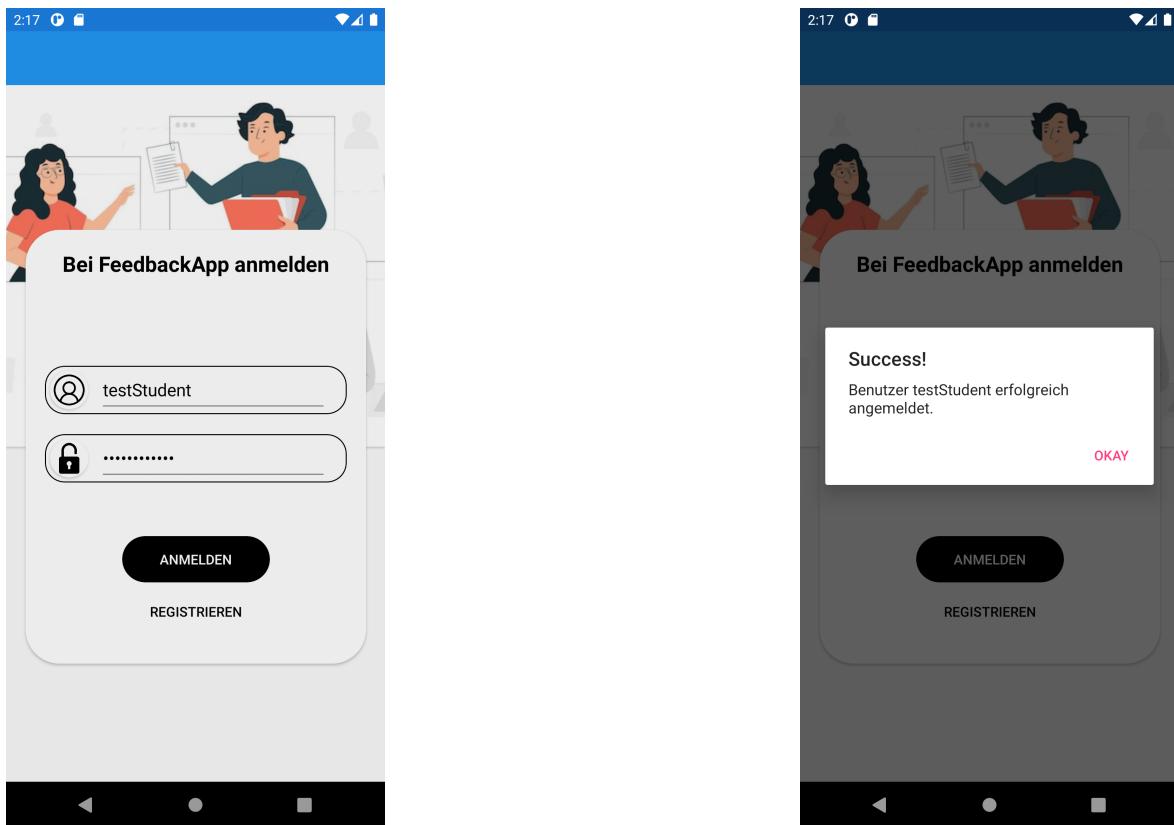


Abbildung 20: Login Daten einfügen

Durch die Eingabe von Benutzername und Passwort prüfen wir, ob bereits ein Account eingerichtet ist. Wenn das Konto bereits existiert, erhalten wir eine positive Antwort, andernfalls erhalten wir einen Fehler ("error"). Die Anmeldung ist für Schüler und Lehrer gleich, daher gibt es nur eine Anmeldeschaltfläche für alle Benutzer.

## 5.2 Startseite

### 5.2.1 Schüler

Die Startseite für Studierende ist einfach gehalten mit der Hauptfunktion der Fächersuche (Feedbackname). Ganz unten befindet sich ein Button "Feedback geben", der nur aktiviert wird, wenn "Einheit" gefunden wird. In der oberen rechten Ecke befindet sich eine Schaltfläche, die zu den persönlichen Daten des Benutzers führt, wo die Daten angepasst, geändert oder gelöscht werden können. In der Mitte der Seite befindet sich eine Suchmaschine für alle Fächer (Einheit), und der Name muss genau eingegeben werden, um die richtigen Ergebnisse zurückzugeben.

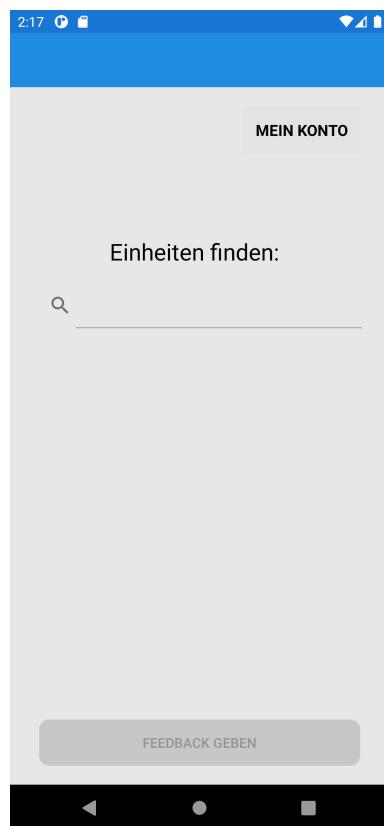


Abbildung 21: HomePage Student Ansicht

Zunächst muss der richtige Name des Themas oder zumindest der richtige Anfangsteil eingegeben werden, und nach der Suchmaschine erhalten wir korrekte Ergebnisse.

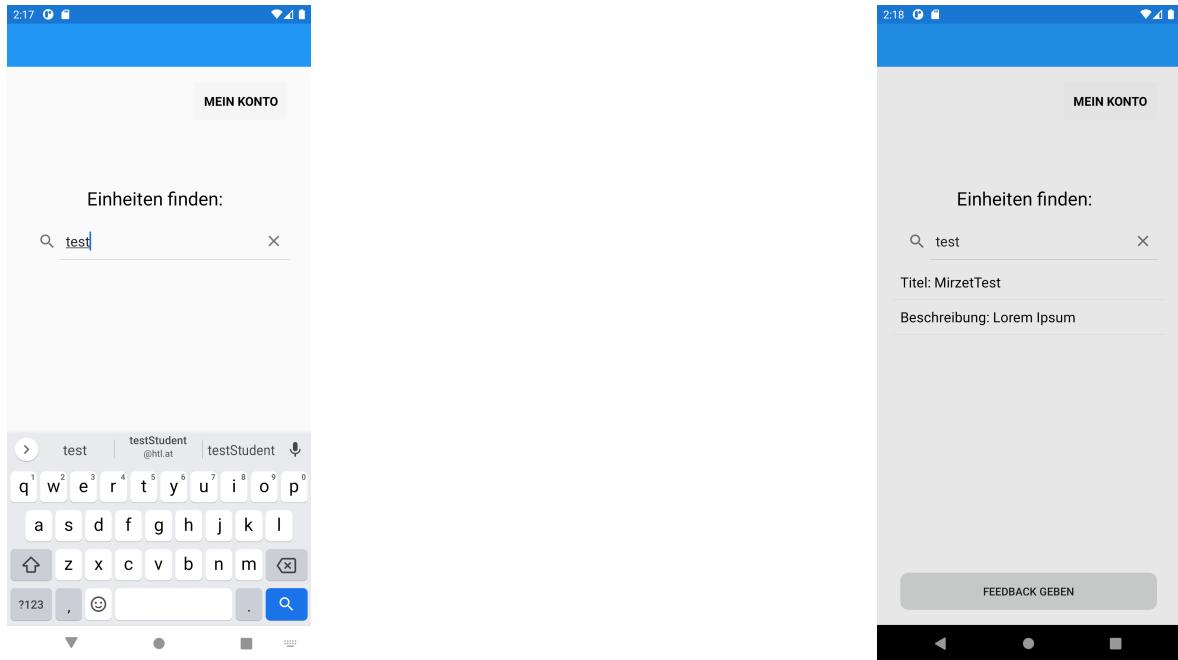


Abbildung 22: Einheiten Ansicht

Nachdem der Artikel gefunden wurde, wird die Schaltfläche unten auf der Seite sichtbar und aktiv, und wir können darauf klicken, wenn wir Feedback oder einen Kommentar hinterlassen möchten.

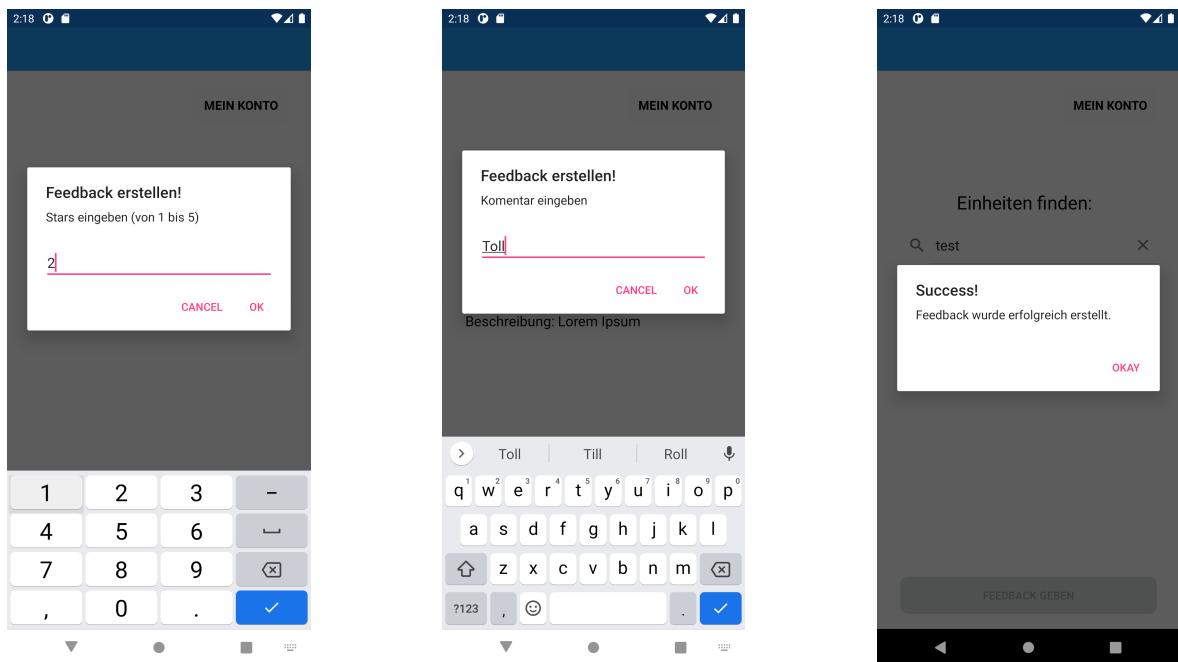


Abbildung 23: Feedback geben Ansicht

Wenn wir eine Bewertung über 5 oder unter 1 eingeben, erhalten wir eine Fehlermeldung und das Feedback wird nicht gespeichert. Nach erfolgreicher Abgabe von Feedback ist der Button „Feedback geben“ wieder inaktiv und wir können es nicht noch einmal geben.

### 5.2.2 Lehrer

Die Startseite für Lehrer ist die gleiche wie für Schüler, nur haben wir zusätzlich einen Button zum Anlegen von Fächern. Darüber hinaus gibt es auch eine Suchmaschine, um zu prüfen, ob der Artikel, den wir eingeben möchten, bereits existiert. Natürlich gibt es auch einen Button für das Benutzerkonto.

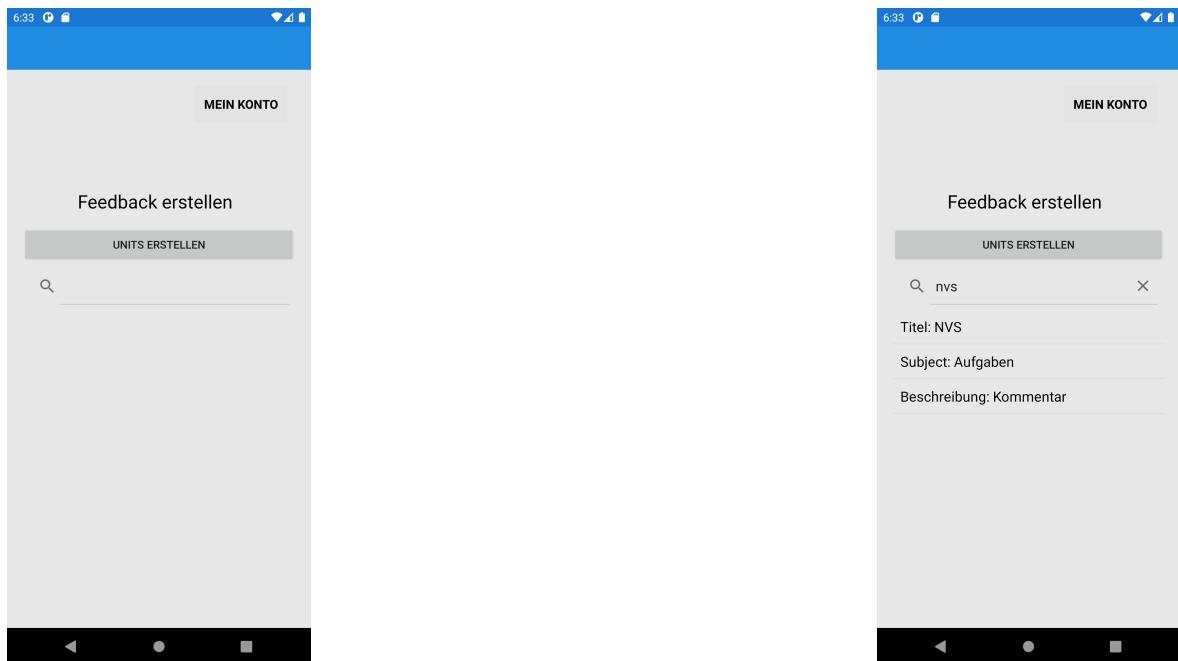


Abbildung 24: HomePage Lehrer Ansicht

Um einen neuen Betreff einzugeben, ist die Eingabe von Titel, Betreff und Beschreibung des Betreffs obligatorisch. Das Formular wird in 3 Schritten ausgefüllt, und wenn alles korrekt eingetragen ist, erhalten wir eine positive Antwort.

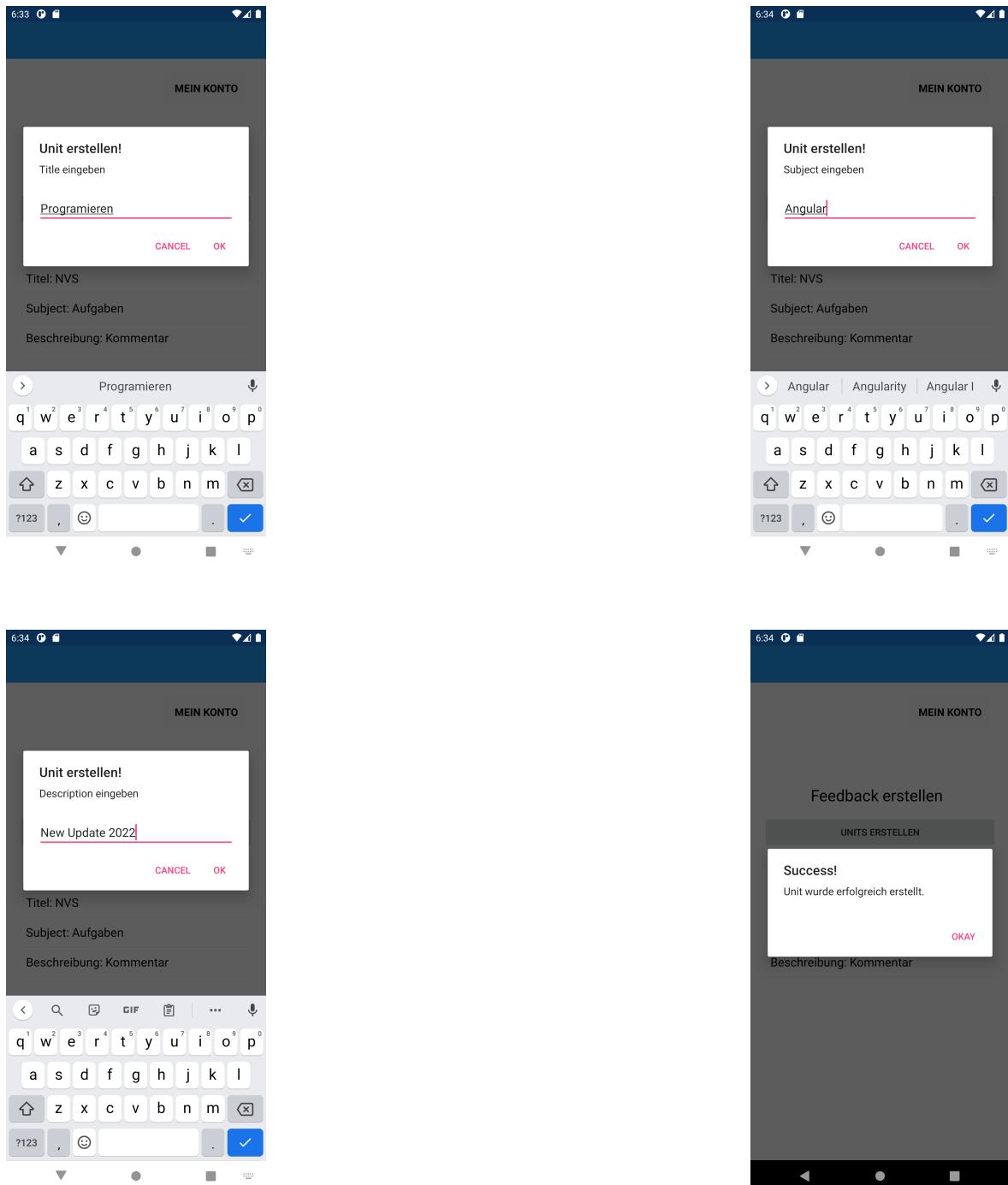


Abbildung 25: Unit erstellen

## 5.3 Registrierung

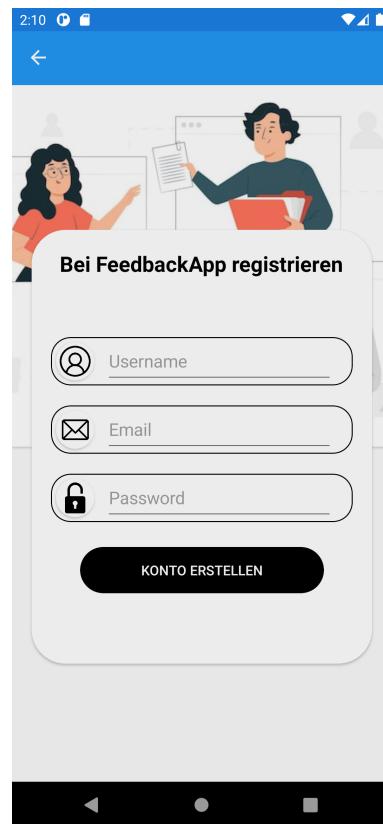


Abbildung 26: Registrierung Page Ansicht

Die Registrierungsseite ist einfach und übersichtlich gestaltet. Um einen Benutzer zu registrieren, müssen Sie einen Benutzernamen, eine E-Mail-Adresse und ein Passwort eingeben. Nur Schüler können sich mit dieser Methode registrieren, während die Lehrerregistrierung so geschützt ist, dass Schüler sich nicht als Lehrer registrieren können und Lehrer ihr Profil vom Admin-Team erhalten.

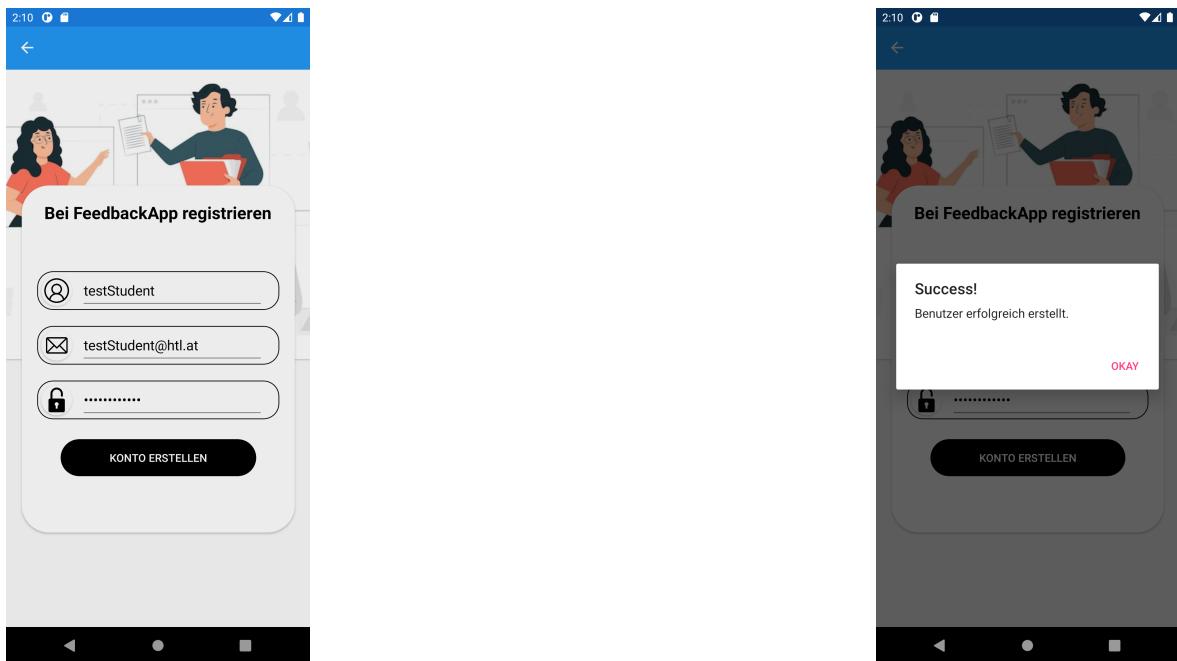


Abbildung 27: Benutzerkonto erstellen

Falls die Daten falsch sind, d.h. die nicht den Regeln des Codes unterliegen, erhalten wir per E-Mail eine Warnung und einen Fehler, um die Registrierungsaktion zu wiederholen.

## 5.4 Benutzerkontoverwaltung

Im Profil kann der Benutzer seine gespeicherten Daten wie Passwort, Vorname oder Schule aktualisieren. Die Seite ist einfach und geräumig gestaltet, um die Daten, die der Benutzer benötigt, klar zu sehen.

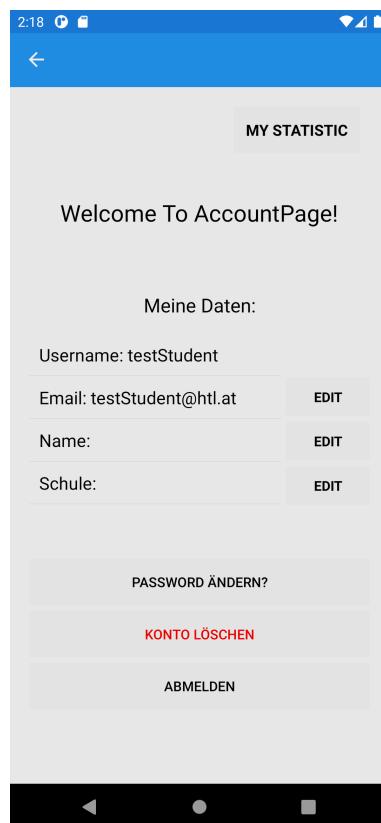


Abbildung 28: Benutzerkontoverwaltung Student Ansicht

Sobald wir die Benutzerkontoseite betreten, sehen wir, dass der Name, der Nachname und der Name der Schule, die wir besuchen, fehlt. Mit den Buttons auf der rechten Seite können wir jeden einzeln eingeben, ändern oder löschen. Auch die E-Mail ist bereits gespeichert, die wir bei der Registrierung eingegeben haben, die aber später geändert werden kann.

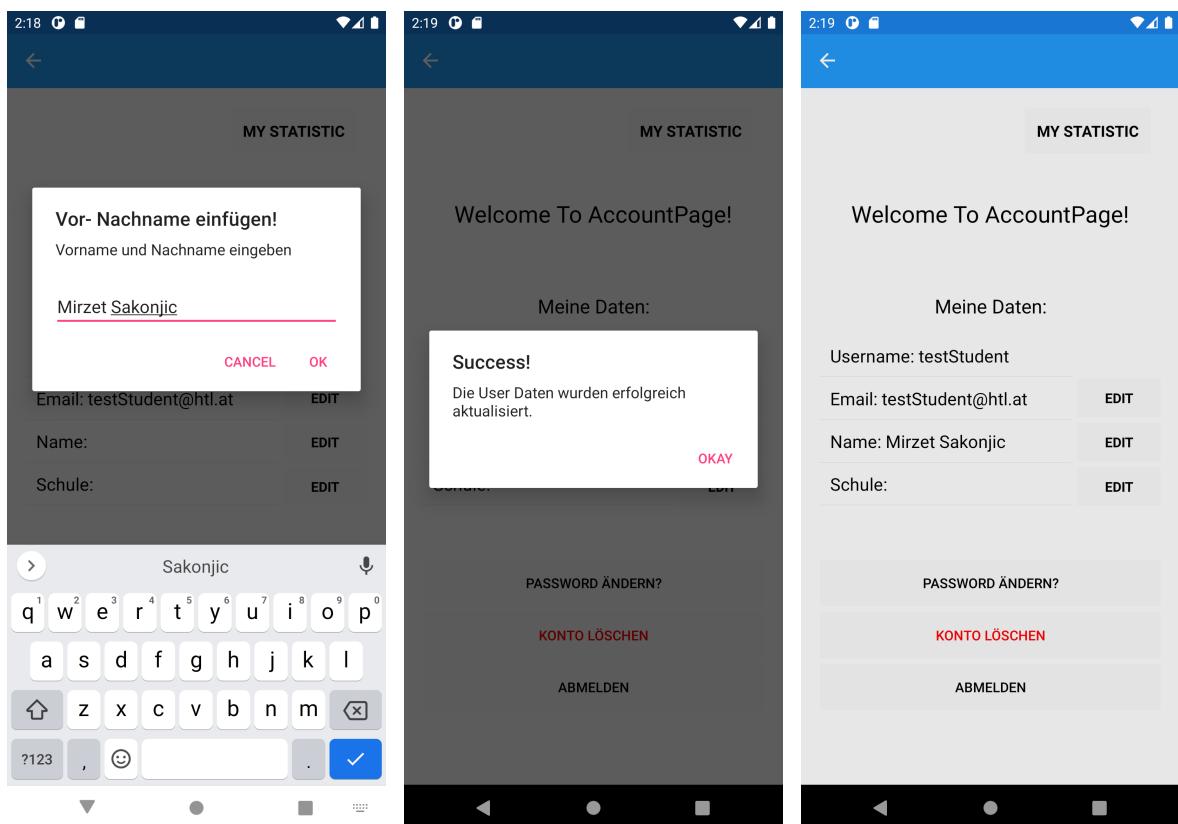


Abbildung 29: Namensänderung

Als nächstes geben Sie den Namen der Schule ein, die wir besuchen, oder des Hauptfachs, das wir studieren.

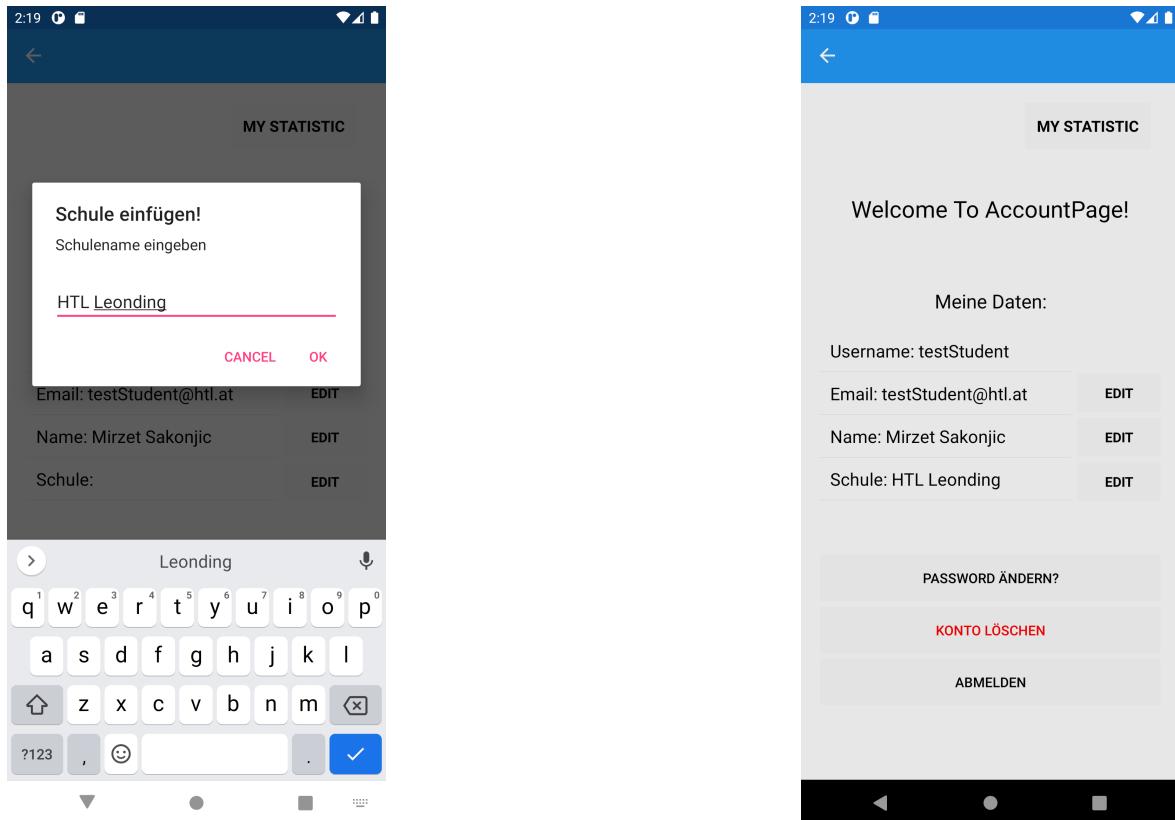


Abbildung 30: Umbenennung der Schule

Falls wir die E-Mail-Adresse ändern und eine neue eingeben möchten, ist dies ebenfalls möglich. Falls die neue E-Mail-Adresse falsch ist oder die Kriterien für die E-Mail-Adresse nicht erfüllt, erhalten wir eine Fehlermeldung.

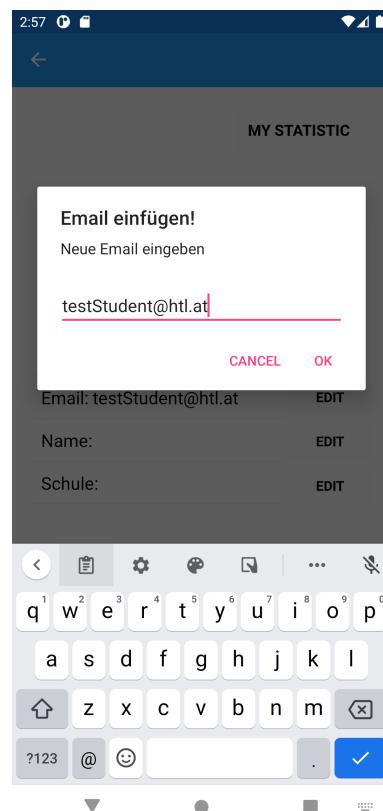
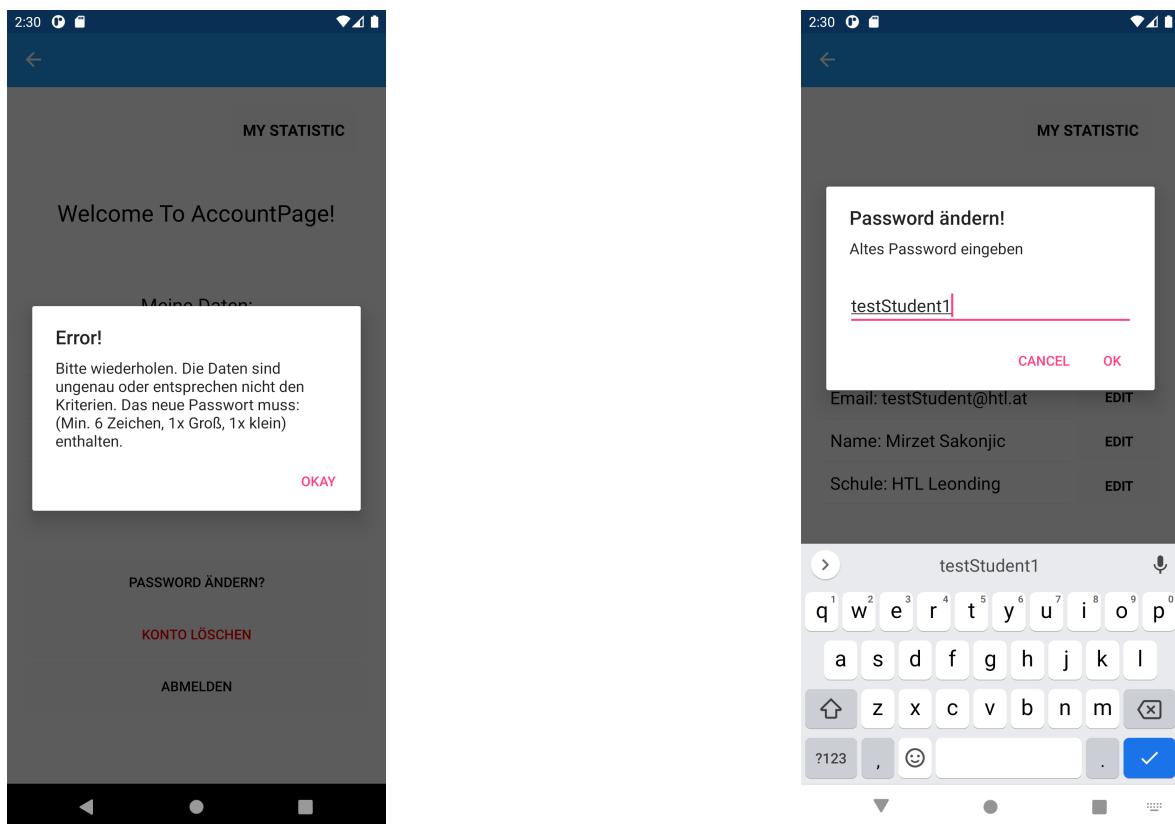


Abbildung 31: Änderung der E-Mail-Adresse

Wir werden das Passwort ändern, indem wir zuerst das alte Passwort und dann das neue eingeben. Wenn das alte Passwort korrekt ist und somit das neue Passwort den Passwortkriterien unterliegt, wird das neue Passwort gespeichert.



Das Löschen des Kontos ist durch Eingabe eines Passworts möglich, und alle Kontodaten werden entfernt und gelöscht. Um sich abzumelden, müssen wir den Abmelden-Button drücken, und wir werden direkt auf die Login-Startseite zurückgeleitet. Der einzige Unterschied auf der Benutzerkontoseite zwischen Schülern und Lehrern ist die Statistik in Button. So zeigt es für jeden Benutzer unterschiedliche Statistiken, zB wie viele Fächer der Lehrer eingegeben hat oder wie viel Feedback der Schüler gegeben hat.

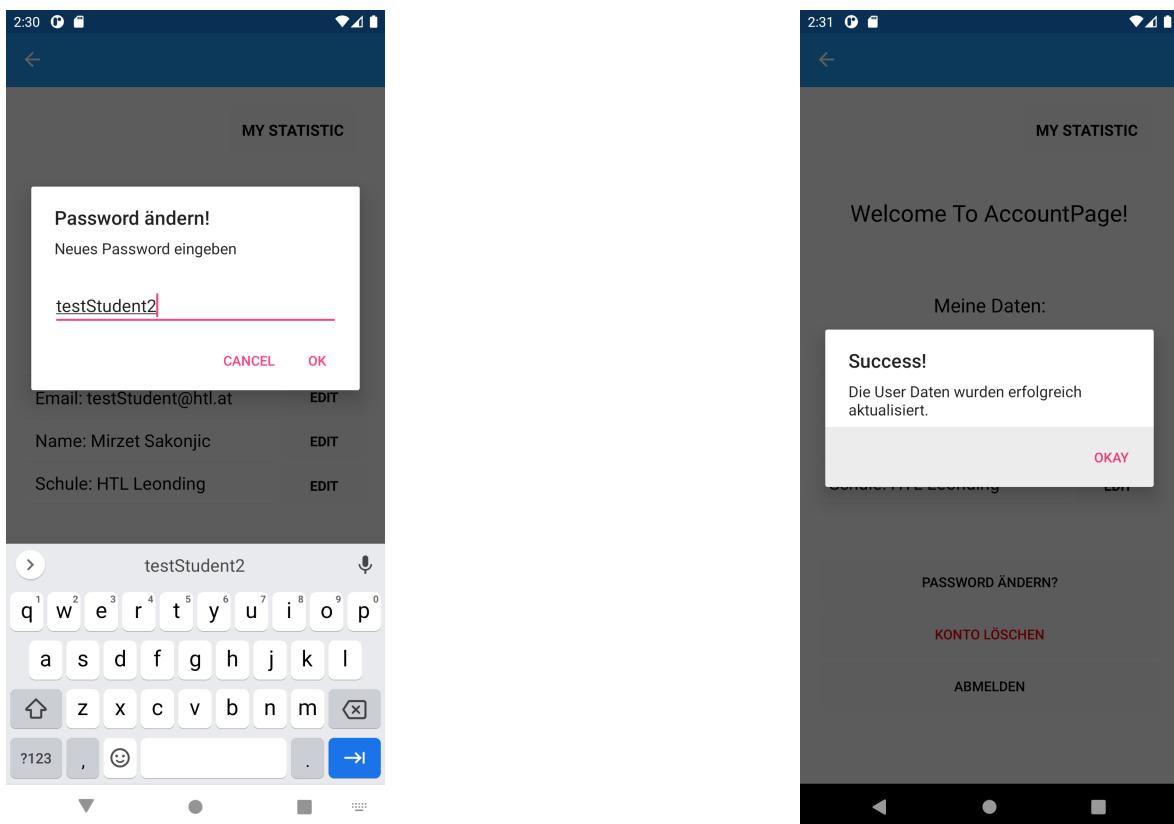
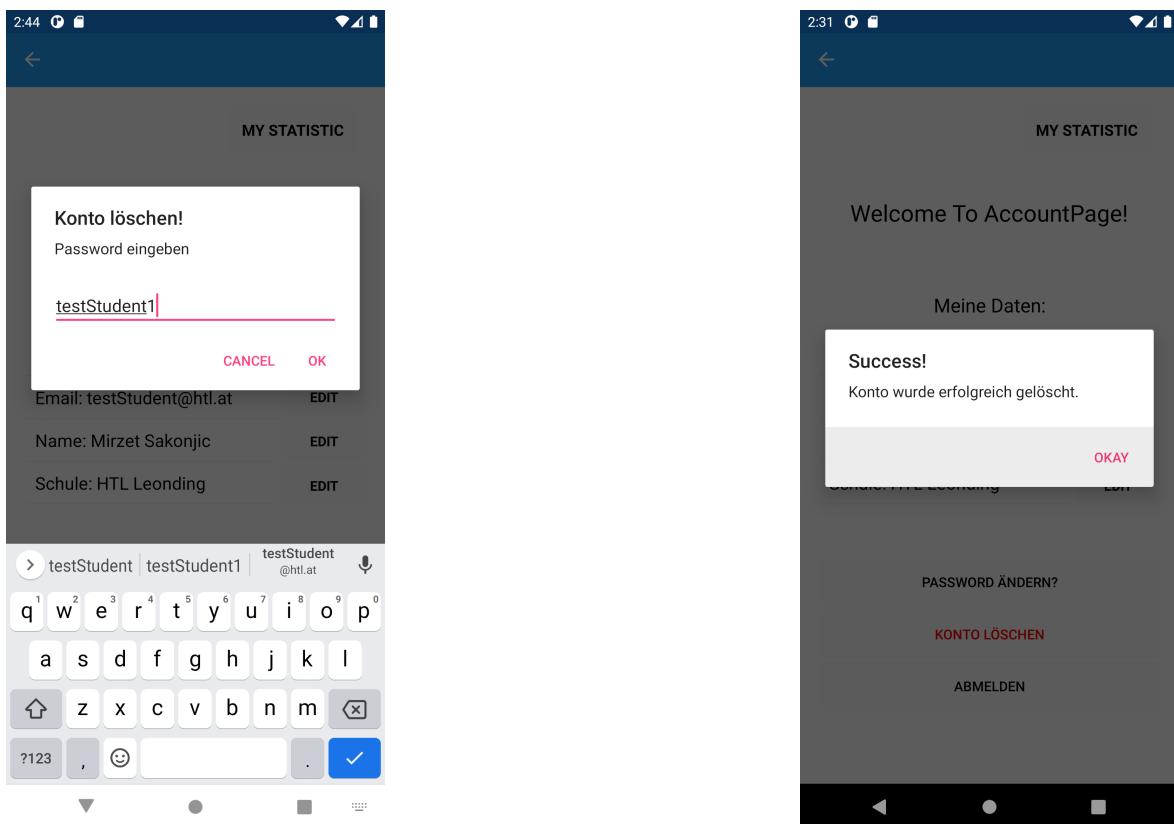


Abbildung 32: Passwort ändern

## 5.5 Statistiken

### 5.5.1 Schüler

Die Studentenstatistik wird in die Menge und Anzahl der abgegebenen Feedbacks eingerechnet. Das Messgerät zählt also alle Rückmeldungen und wirft eine Zahl als Summe aus.



## 5.5.2 Lehrer

Die Lehrerstatistik wird in der Menge und Anzahl der abgegebenen Feedbacks gezählt. Es zeigt zusätzlich die durchschnittliche Bewertung aller Feedbacks (AVG).

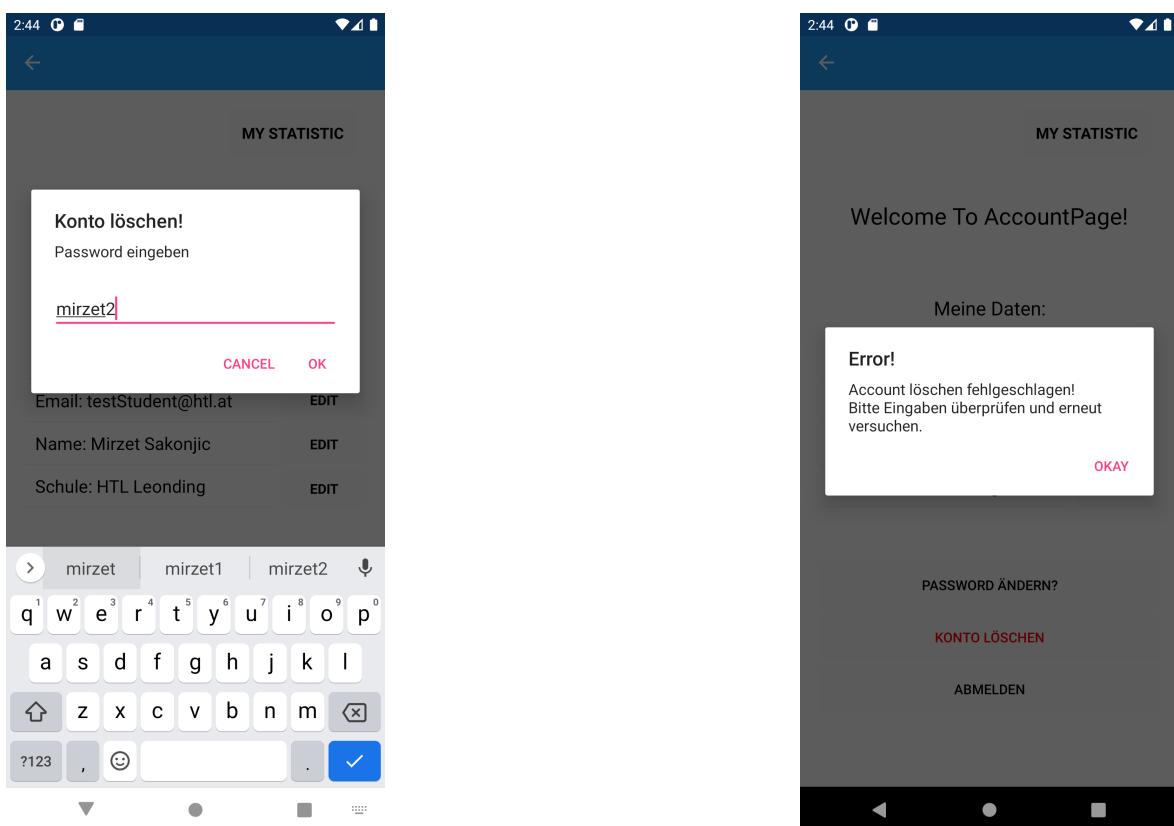


Abbildung 33: Konto löschen

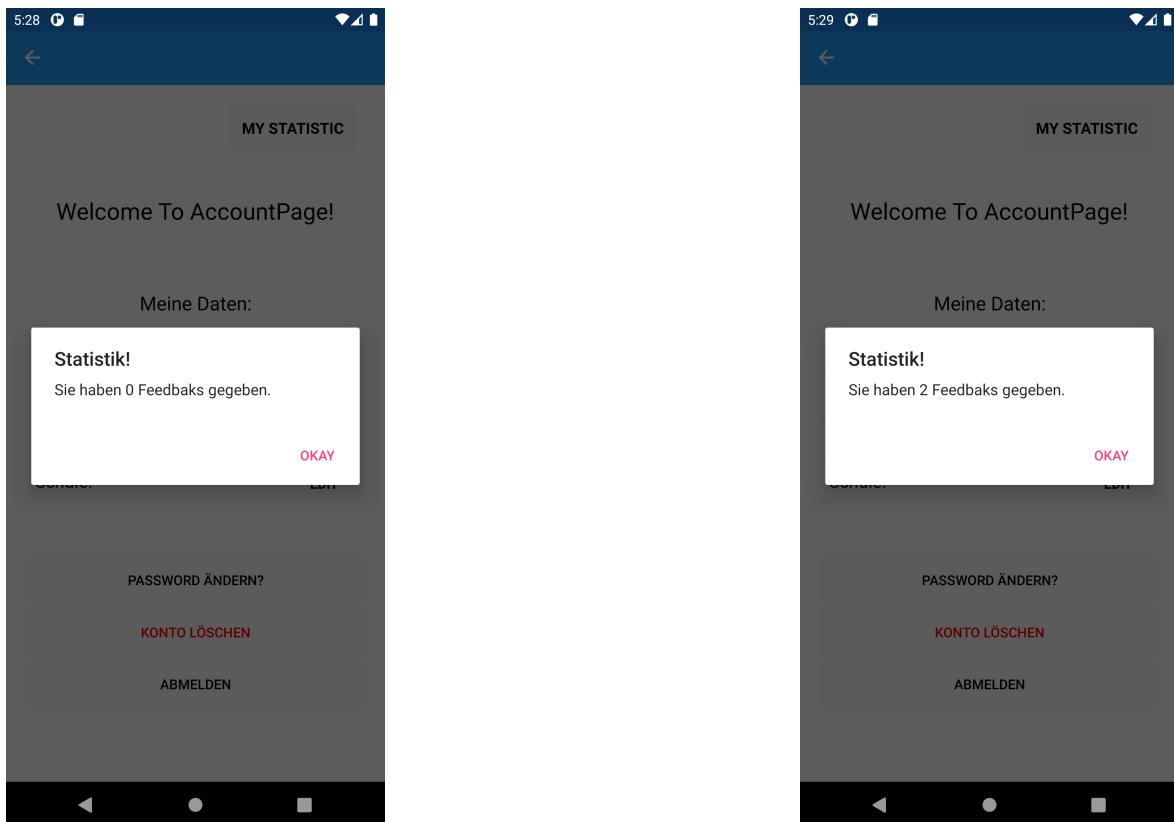


Abbildung 34: Statistiken Schüler

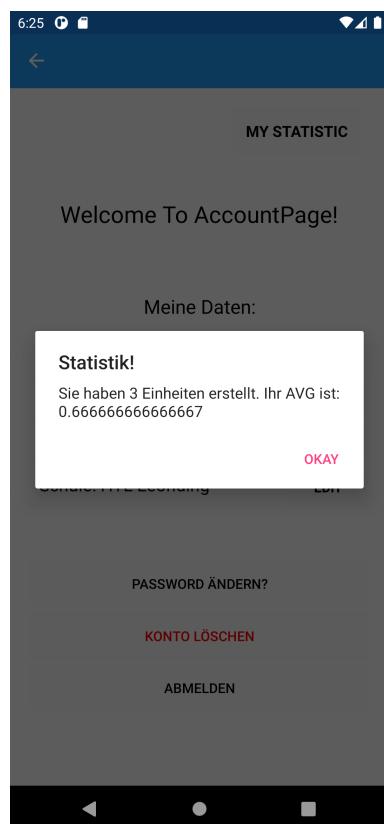


Abbildung 35: Statistiken Lehrer

# 6 Umsetzung

## 6.1 Backend

### Erstellung der Datenbank

Für die Erstellung der Datenbank wurde der Modellierungsansatz Code-First angewendet. Das Mapping der EF Core Datenbank ist in der Klasse FeedbackDbContext.cs definiert. Die Tabellen entsprechen der in Kapitel 2 Abbildung 3 abgebildeten ERD.

```
public class FeedbackDbContext : DbContext
{
    1 reference
    public FeedbackDbContext()
    {
    }

    12 references
    public FeedbackDbContext(DbContextOptions<FeedbackDbContext> options) : base(options)
    {
    }

    23 references
    public DbSet<User> Users => Set<User>();
    11 references
    public DbSet<TeachingUnit> TeachingUnits => Set<TeachingUnit>();
    10 references
    public DbSet<Feedback> Feedbacks => Set<Feedback>();
    6 references
    public DbSet<GlobalHistory> GlobalHistories => Set<GlobalHistory>();
    6 references
    public DbSet<UserStatistic> UserStatistics => Set<UserStatistic>();
    5 references
    public DbSet<TeachingUnitStatistic> TeachingUnitStatistics => Set<TeachingUnitStatistic>();

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
    }
}
```

Abbildung 36: FeedbackDbContext.cs Mapping

Die Tabellen wurde mit Klassen erstellt, die mit Annotationen ausgezeichnet wurden. Sie sind im Core Bereich der Solution unter Models zu finden. Eine Besonderheit ist die Model-Klasse TeachingUnit in diesem Projekt, da sie die Möglichkeit mithilfe eines bool-Atributes bietet, ob eine Lehreinheit öffentlich sichtbar ist.

Beide Datenbanken für Feedback und Identity an sich wird lokal in Microsoft SQL Server erstellt und initialisiert. Die Connection Strings sind in appsettings.json definiert.

```

public class TeachingUnit
{
    [Key]
    7 references
    public int Id { get; set; }

    [Required]
    5 references
    public bool IsPublic { get; set; }

    [Required]
    5 references
    public string Title { get; set; } = string.Empty;
    3 references
    public string? Subject { get; set; } = string.Empty;
    3 references
    public string? Description { get; set; } = string.Empty;
    3 references
    public DateTime? Date { get; set; }
    4 references
    public DateTime? ExpiryDate { get; set; }
    3 references
    public string? SubscriptionKey { get; set; } = string.Empty;

    5 references
    public int UserId { get; set; }
    1 reference
    public User User { get; set; } = new User();

}

```

Abbildung 37: Model-Klasse TeachingUnit.cs

### 6.1.1 Datenbank (Persistance)

Die Controller der Web API greifen auf die Daten im Persistance-Teil der Anwendung zu. Es existieren 3 Repositories die den Datenzugriff der Feedback API im Hintergrund ermöglichen. Alle Methoden der Repositories sind asynchron und ermöglichen somit das Multitasking von Requests. Für die Datenzugriffe in den Methoden wurde LINQ verwendet.

#### User-Repository

Dieses Repository stellt Datenzugriffsmethoden für die Benutzerkontoverwaltung bereit. Mit diesen Methoden werden User abgefragt, erstellt, geändert und gelöscht. Zusätzlich stellt es Zählmethoden für Statistiken bereit.

#### Statistic-Repository

Diese Methoden stellen Methoden für die Statistik der Anwendung bereit. Wenn ein Benutzer erstellt oder gelöscht wird, so muss auch die verknüpfte User-Statistik-

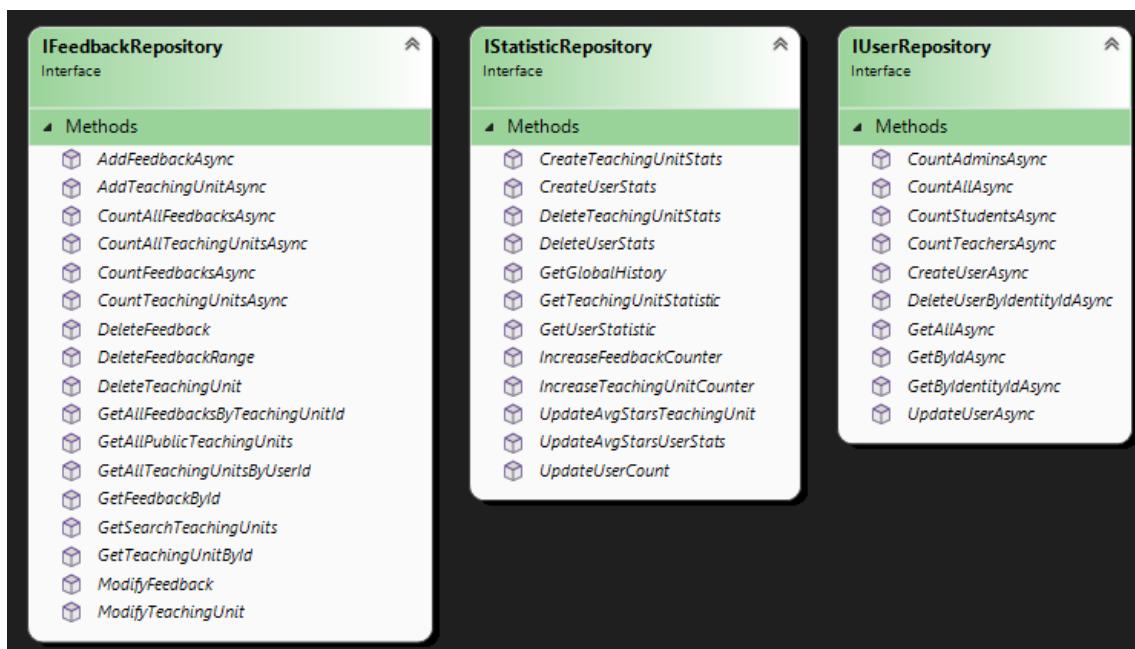


Abbildung 38: Überblick der Methoden der Repositories

Tabelle erstellt oder gelöscht werden. Die Umsetzung ist in der folgende Abbildung sichtbar.

## Feedback-Repository

Sie ist das Herzstück der Persistance-Schicht des Projektes. In diesem Repository werden die Lehreinheiten und Feedbacks verwaltet. Wie beim Statistik Repository müssen auch im Falle einer Löschung einer TeachingUnit alle dazugehörigen Feedbacks gelöscht werden.

```

3 references
public async Task CreateUserStats(User user)
{
    UserStatistic userStatistic = new() { UserId = user.Id, User = user };

    await _dbContext.UserStatistics.AddAsync(userStatistic);
}

2 references
public async Task DeleteTeachingUnitStats(int teachingUnitId)
{
    var teachingUnitStats = await GetTeachingUnitStatistic(teachingUnitId);

    _dbContext.TeachingUnitStatistics.Remove(teachingUnitStats);
}

2 references
public async Task DeleteUserStats(int userid)
{
    var userStats = await GetUserStatistic(userid);

    _dbContext.UserStatistics.Remove(userStats);
}

```

Abbildung 39: StatistikRepository Methoden Create/Delete

```

public async Task DeleteTeachingUnit(int teachingUnitId)
{
    await DeleteFeedbackRange(teachingUnitId);
    var teachingUnit = await _dbContext.TeachingUnits.FindAsync(teachingUnitId);
    _dbContext.TeachingUnits.Remove(teachingUnit);
}

```

Abbildung 40: Methode für Löschung einer TeachingUnit

## 6.1.2 Web API

### Benutzerverwaltung

Für das Usermanagement wird das NuGet Package ASP.NET Identity verwendet. Der Authenticate-Controller ermöglicht die Registrierung, den Login und die Löschung des Benutzerkontos. Zusätzlich kann das Passwort und die E-Mail Adresse nachträglich geändert werden. Schüler und Lehrende werden durch Rollen getrennt. Bei erfolgreicher Authentifizierung wird ein gültiger JSON Web Token zurückgesendet.

Der Benutzername muss folgende Kriterien erfüllen:

- mindestens 6 Zeichen
- maximal 26 Zeichen
- keine Leerzeichen

**Authenticate** Main User Account Management

- POST** `/api/login` get a login token
- POST** `/api/register` register a student
- POST** `/api/register-teacher` register a teacher
- POST** `/api/deleteAccount` deletes a student or teacher account (require token)
- POST** `/api/changePw` change the password of a student or teacher
- POST** `/api/changeEmail` change the e-mail address of a student or teacher (require token)

**UserAccount** Additional User Account Management

- POST** `/api/user/getData` get the additional user data (require token)
- POST** `/api/user/modifierData` modify the additional user data (require token)

Abbildung 41: Screenshot Swagger Web API Dokumentation Benutzerverwaltung

- keine Sonderzeichen (Umlaute sind erlaubt)

Das Passwort hat folgende Mindestanforderungen:

- mindestens 6 Zeichen
- keine Leerzeichen
- mindestens einen Großbuchstaben, einen Kleinbuchstaben und eine Zahl

Diese Anforderungen sind in der Start-Up Konfiguration der ASP.NET Core Web API festgelegt und werden zusätzlich von der Klasse AuthenticateValidations.cs kontrolliert. Im Fehlerfall wird ein entsprechender HTTP-Fehlercode zurückgegeben.

Der UserAccount-Controller ermöglicht das Hinzufügen folgender Daten:

- Titel
- Vorname
- Nachname

- Geburtsdatum
- Lehranstalt

```
// For Identity
services.AddIdentity<ApplicationUser, IdentityRole>(options =>
{
    // Username settings
    var allowed = options.User.AllowedUserNameCharacters + "äöÄÜÖß";
    options.User.AllowedUserNameCharacters = allowed;
    options.User.RequireUniqueEmail = true;
    // Username Length = 6 --> AuthenticateValidations.cs

    // Password settings
    options.Password.RequireDigit = true;
    options.Password.RequireLowercase = true;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequireUppercase = true;
    options.Password.RequiredLength = 6;
    options.Password.RequiredUniqueChars = 0;
})
    .AddEntityFrameworkStores<ApplicationContext>()
    .AddDefaultTokenProviders();
```

Abbildung 42: Screenshot Startup.cs Benutzername und Passwort Anforderungen

## Authentifizierung

Die Login-Methode sendet ein HTTP OK Response mit dem gültigen JWT Token als Objekt zurück. Der Token ist 15 Minuten lang gültig und beinhaltet die Id des Benutzerkontos und deren Rolle. Dieser erlaubt es, je nach Rolle, Zugriff auf die geschützten Bereiche der Web API.

```
var authSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JWT:Secret"]));  
  
var token = new JwtSecurityToken  
(  
    issuer: _configuration["JWT:ValidIssuer"],  
    audience: _configuration["JWT:ValidAudience"],  
    expires: DateTime.Now.AddMinutes(15),  
    claims: authClaims,  
    signingCredentials: new SigningCredentials(authSigningKey, SecurityAlgorithms.HmacSha256)  
>;  
  
//var UserInDb = await _unitOfWork.UserRepository.GetByIdentityIdAsync(user.Id);  
  
return Ok(new  
{  
    token = new JwtSecurityTokenHandler().WriteToken(token),  
    expiration = token.ValidTo,  
  
    //Klarertext für Entwicklungszwecke Xamarin!!!  
    //userId = UserInDb.Id,  
    //identityId = user.Id,  
    //role = userRoles.FirstOrDefault(),  
    //username = user.UserName,  
    //email = user.Email  
});
```

Abbildung 43: Screenshot JWT Token Erstellung detailliert

## 6.2 Frontend

### 6.2.1 Login

Die Anwendungsseite wird ausschließlich in der Oberfläche realisiert, indem das Bild in einem separaten Ressourcenordner gespeichert und mit <img> angezeigt wird. Das Bild auf der Startseite ist urheberrechtlich geschützt, da sich dieses eine Bild nur auf der Anmeldeseite befindet. Beim Einloggen gibt der Nutzer seine E-Mail-Adresse und sein Passwort ein, die bereits ausgelesen und im Hintergrund in der Datenbank gespeichert wurden. Bei Bedarf kann ein neuer Benutzer angelegt werden. Die Validierung überprüft, ob die E-Mail-Adresse und das Passwort eingegeben wurden. Es überprüft auch Passwortregeln wie Länge, Großbuchstaben oder zusätzliche Zeichen.

```
0 Verweise
private async void ButtonAnmelden_Clicked(object sender, EventArgs e)
{
    LoginService serviceData = new LoginService();
    var getData = await serviceData.GetData(EntryUsername.Text, EntryPassword.Text);
```

Abbildung 44: Button Login

Wenn der User auf den Button „Login“ drückt, wird ein LoginService-Request mit der eingegebenen E-Mail und Passwort ans Backend gesendet. Wenn die Zugangsdaten stimmen und der User vorhanden ist, werden die Logindaten inklusive Username zurückgesendet und in der globalen Komponente abgespeichert, damit im gesamten Projekt darauf zugegriffen werden kann.

```
30 Verweise
public class LoginService
{
    RestClient<LoginModel> _restClient = new RestClient<LoginModel>();

    1 Verweis
    public async Task<bool> CheckLoginIfExists(string userName, string password)
    {
        var check = await _restClient.checkLogin(userName, password);
        return check;
    }
}
```

Abbildung 45: RestClient

Mit Hilfe von RestClient erstellen wir ein LoginModel und fügen es in eine bool-Variable ein, um zu antworten, ob die Anmeldung erfolgreich war oder nicht, d.h. ob das Profil existiert oder nicht.

```
1Verweis
public class RestClient<T>
{
    private const string MainWebServiceUrl = "https://10.0.2.2:5081"; // FOR ANDROID EMULATOR
    private const string LoginWebServiceUrl = MainWebServiceUrl + "/api/Login";

    1Verweis
    public async Task<bool> checkLogin(string userName, string password)
    {
        LoginModel model = new LoginModel() { Username = userName, Password = password };

        HttpClientHandler httpClientHandler;
        #if (DEBUG)
            httpClientHandler = new HttpClientHandler { ServerCertificateCustomValidationCallback = (o, cert, chain, errors) => true };
        #else
            httpClientHandler = new HttpClientHandler(); /*endif using (var client = new HttpClient(httpClientHandler));
        #endif

        var client = new HttpClient(httpClientHandler);
        var json = JsonConvert.SerializeObject(model);
        var content = new StringContent(json, Encoding.UTF8, "application/json");
        string debugContentJson = await content.ReadAsStringAsync();
        var result = await client.PostAsync(LoginWebServiceUrl, content).ConfigureAwait(false);
        return result.IsSuccessStatusCode;
    }
}
```

Abbildung 46: HttpClient

Sollten die Zugangsdaten nicht stimmen, wird eine Fehlermeldung angezeigt. Nach dem erfolgreichen Laden aller Daten wird der User zu der Startseite weitergeleitet.

```
public async void GetUserData()
{
    LoginService serviceData = new LoginService();
    var getData = await serviceData.GetData(EntryUsername.Text, EntryPassword.Text);

    AllUserData jsonToken = JsonConvert.DeserializeObject<AllUserData>(getData);
    string token = jsonToken.Token;
    AllUserData jsonExpiration = JsonConvert.DeserializeObject<AllUserData>(getData);
    string expiration = jsonExpiration.Expiration;
    AllUserData jsonUserId = JsonConvert.DeserializeObject<AllUserData>(getData);
    int userId = jsonUserId.UserId;
    AllUserData jsonIdentityId = JsonConvert.DeserializeObject<AllUserData>(getData);
    string identityId = jsonIdentityId.IdentityId;
    AllUserData jsonRole = JsonConvert.DeserializeObject<AllUserData>(getData);
    string role = jsonRole.Role;
    AllUserData jsonUsername = JsonConvert.DeserializeObject<AllUserData>(getData);
    string username = jsonUsername.Username;
    AllUserData jsonEmail = JsonConvert.DeserializeObject<AllUserData>(getData);
    string email = jsonEmail.Email;
}
```

Abbildung 47: UserData laden

Nachdem die Logindaten erfolgreich geladen wurden, werden die Profildaten wie Vorna-  
me, Nachname oder Schule mithilfe der userId geladen.

```
1Verweis
public async Task<string> getData(string userName, string password)
{
    LoginModel model = new LoginModel() { Username = userName, Password = password };

    HttpClientHandler httpClientHandler;
    #if (DEBUG)
        httpClientHandler = new HttpClientHandler { ServerCertificateCustomValidationCallback = (o, cert, chain, errors) => true };
    #else
        httpClientHandler = new HttpClientHandler(); /*endif using (var client = new HttpClient(httpClientHandler));
    #endif

    var client = new HttpClient(httpClientHandler);
    var json = JsonConvert.SerializeObject(model);
    var content = new StringContent(json, Encoding.UTF8, "application/json");
    string debugContentJson = await content.ReadAsStringAsync();
    var result = await client.PostAsync(LoginWebServiceUrl, content).ConfigureAwait(false);

    string responseString = await result.Content.ReadAsStringAsync();
    var text = JsonConvert.DeserializeObject(responseString);
    return responseString;
}
```

Abbildung 48: UserData laden

## 6.2.2 Registrierung

Bei der Registrierung werden die eingegebenen Werte zunächst daraufhin überprüft, ob sie den Standards entsprechen. Danach wird registerService erstellt und Daten werden über restClient an RegisterModel übergeben. Feedback kann eine Bestätigung oder ein Fehler sein.



```

2 Verweise
private async void SchulerReg_Clicked(object sender, EventArgs e)
{
    if ((string.IsNullOrEmpty(EntryUsername.Text)) ||
        (string.IsNullOrEmpty(EntryEmail.Text)) ||
        (string.IsNullOrEmpty(EntryPassword.Text)))
    {
        await DisplayAlert("Error!", "Benutzer erstellen fehlgeschlagen! Bitte Eingaben überprüfen und erneut versuchen.", "Okay");
    }
    else
    {
        RegisterService service = new RegisterService();
        var response = await service.CheckRegisterIfExists(EntryUsername.Text, EntryEmail.Text, EntryPassword.Text);

        if(response)
        {
            await DisplayAlert("Success!", "Benutzer erfolgreich erstellt.", "Okay");
            await Navigation.PushAsync(new LoginPage());
        }
        else
        {
            await DisplayAlert("Error!", "Benutzer erstellen fehlgeschlagen! Bitte Eingaben überprüfen und erneut versuchen oder Benutzer existiert bereits!", "Okay");
        }
    }
}

```

Abbildung 49: Konto erstellen

Mit Hilfe von HttpClient werden die Daten im Backend geprüft und wir erhalten das Ergebnis.



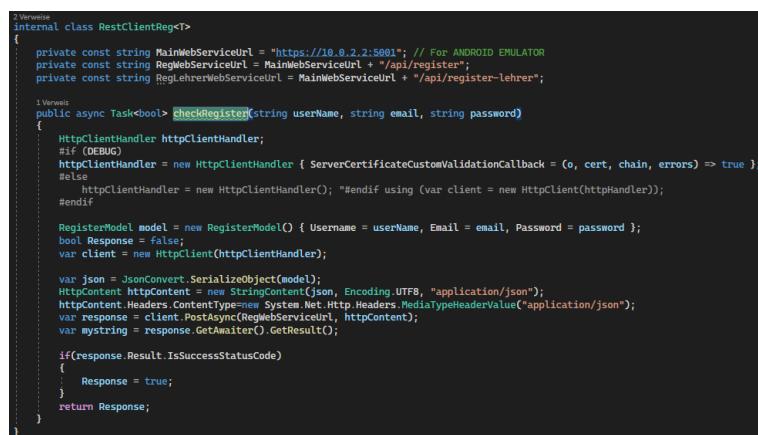
```

2 Verweise
public class RegisterService
{
    RestClientReg<RegisterModel> _restClient = new RestClientReg<RegisterModel>();

    1 Verweis
    public async Task<bool> CheckRegisterIfExists(string userName, string email, string password)
    {
        var check = await _restClient.checkRegister(userName, email, password);
        return check;
    }
}

```

Abbildung 50: RestClient



```

2 Verweise
internal class RestClientReg<T>
{
    private const string MainWebServiceUrl = "https://19.0.2.2:5001"; // For ANDROID EMULATOR
    private const string ReglehrServiceUrl = MainWebServiceUrl + "/api/register";
    private const string RegLehrerWebServiceUrl = MainWebServiceUrl + "/api/register-lehrer";

    1 Verweis
    public async Task<bool> checkRegister(string userName, string email, string password)
    {
        HttpClientHandler httpClientHandler;
        #if DEBUG
        httpClientHandler = new HttpClientHandler { ServerCertificateCustomValidationCallback = (o, cert, chain, errors) => true };
        #else
        httpClientHandler = new HttpClientHandler(); "#endif using (var client = new HttpClient(httpClientHandler));
        #endif

        RegisterModel model = new RegisterModel() { Username = userName, Email = email, Password = password };
        bool Response = false;
        var client = new HttpClient(httpClientHandler);

        var json = JsonConvert.SerializeObject(model);
        HttpContent httpContent = new StringContent(json, Encoding.UTF8, "application/json");
        httpContent.Headers.ContentType = System.Net.Http.Headers.MediaTypeHeaderValue("application/json");
        var response = client.PostAsync(RegWebServiceUrl, httpContent);
        var myString = response.GetAwaiter().GetResult();

        if(response.Result.IsSuccessStatusCode)
        {
            Response = true;
        }
        return Response;
    }
}

```

Abbildung 51: HttpClient

### 6.2.3 Benutzerkontoverwaltung

Bei der Anmeldung des Benutzers ruft die Methode sofort die Daten des Benutzers ab und sie werden auf der Kontoseite des Benutzers per Ansichtsliste angezeigt.

```

2 Verweise
public MyAccountPage()
{
    InitializeComponent();
    ListViewUserData();
}

5 Verweise
public void ListViewUserData()
{
    var username = Application.Current.Properties["username"];
    var email = Application.Current.Properties["email"];
    var firstName = Application.Current.Properties["firstName"];
    var lastName = Application.Current.Properties["lastName"];
    var school = Application.Current.Properties["school"];

    var userData = new List<string>();
    userData.Add("Username: " + username);
    userData.Add("Email: " + email);
    userData.Add("Name: " + firstName + " " + lastName);
    userData.Add("Schule: " + school);

    MainListView.ItemsSource = userData;
}

```

Abbildung 52: Benutzerkontoverwaltung Daten auslesen

Wenn wir Daten wie E-Mail, Vorname, Nachname oder Schulname ändern möchten, tun wir dies über Tasks.

```

// RestClient<LoginModel> _restClient = new RestClient<LoginModel>();
// Verweis
public async Task<bool> CheckLoginInfoExists(string userName, string password)
{
    var check = await _restClient.CheckLogin(userName, password);
    return check;
}

// Verweis
public async Task<string> GetData(string userName, string password)
{
    var check = await _restClient.GetData(userName, password);
    return check;
}

// Verweis
public async Task<string> GetUserData(string token, string identityId)
{
    var check = await _restClient.GetUserData(token, identityId);
    return check;
}

// Verweis
public async Task<bool> SetData(string firstName, string lastName, string token, string identityId, string school)
{
    var check = await _restClient.SetData(firstName, lastName, token, identityId, school);
    return check;
}

// Verweis
public async Task<bool> SetPass(string username, string password, string newPassword)
{
    var check = await _restClient.SetPass(username, password, newPassword);
    return check;
}

// Verweis
public async Task<bool> SetEmail(string username, string newEmail, string token)
{
    var check = await _restClient.SetEmail(username, newEmail, token);
    return check;
}

```

```

public async Task<bool> DeleteUserAcc(string username, string password, string token)
{
    var check = await _restClient.DeleteUserAcc(username, password, token);
    return check;
}

public async Task<string> SearchData(string searchKeyValue)
{
    var check = await _restClient.SearchData(searchKeyValue);
    return check;
}

public async Task<bool> CreateFeedback(int teachingInstId, int userId, int star, string comment, string token)
{
    var check = await _restClient.CreateFeedback(teachingInstId, userId, star, comment, token);
    return check;
}

public async Task<bool> CreateUnit(int userId, string title, string subject, string description, string subscriptionKey, string token)
{
    var check = await _restClient.CreateUnit(userId, title, subject, description, subscriptionKey, token);
    return check;
}

public async Task<string> GetUnitByUserId(string token, int userId)
{
    var check = await _restClient.GetUnitByUserId(token, userId);
    return check;
}

public async Task<string> GetUserStatistics(string token, int userId)
{
    var check = await _restClient.GetUserStatistics(token, userId);
    return check;
}

```

Abbildung 53: Tasks

Durch Drücken des Buttons Abmelden gelangen wir auf die Login-Homepage.

```

0 Verweise
private async void btnAbmelden_Clicked(object sender, EventArgs e)
{
    await Navigation.PushAsync(new LoginPage());
}

```

Abbildung 54: Abmelden Button

Die Änderung der persönlichen Daten des Benutzers erfolgt auf die gleiche Weise. Die eingegebenen Daten werden an den LoginService gesendet und dann über den RestClient mit bool Tasks überprüft. Wenn die Antwort positiv ist, wurden die Daten geändert und gespeichert, andernfalls wird ein Fehler ausgegeben.

Nachfolgend finden Sie Beispiele für die Änderung von Daten wie E-Mail, Vorname, Nachname, E-Mail und Passwort.

```

private async void btnUpdateEmail_Clicked(object sender, EventArgs e)
{
    var token = Application.Current.Properties["token"] as string;
    var newEmail = await DisplayPromptAsync("Email", "Neue Email eingeben", keyboard: Keyboard.Email, initialValue:email);
    if (newEmail is null)
    {
        ListviewUserData();
        return;
    }
    if (newEmail.Length == 0)
    {
        await DisplayAlert("Error", "Bitte wiederholen.", "Okay");
    }
    else
    {
        var token = Application.Current.Properties["token"] as string;
        var username = Application.Current.Properties["username"] as string;
        var password = Application.Current.Properties["password"] as string;
        var tokenString = Application.Current.Properties["tokenString"] as string;
        var loginService = new LoginService();
        var serviceSetEmail = new ServiceSetEmail();
        serviceSetEmail.Username = username;
        serviceSetEmail.Password = password;
        serviceSetEmail.Tokens = token;
        serviceSetEmail.TokensString = tokenString;
        if (loginService != null)
        {
            if (loginService.Login("Success", "Ihr User Daten wurden erfolgreich aktualisiert.", "Okay"))
            {
                Application.Current.Properties["email"] = newEmail;
            }
        }
        ListviewUserData();
    }
}

```

```

public async Task<bool> SetEmail(string username, string newEmail, string token)
{
    SetEmailModel model = new SetEmailModel() { Username = username, NewEmail = newEmail, Token = token };
    HttpClientHandler httpclientHandler = new HttpClientHandler();
    httpclientHandler.ServerCertificateCustomValidationCallback = (s, cert, chain, errors) => true;
    using (var client = new HttpClient(httpclientHandler))
    {
        client.DefaultRequestHeaders.Add("Accept", "application/json");
        client.DefaultRequestHeaders.Add("Content-Type", "application/json");
        client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);
        var request = new HttpRequestMessage();
        request.Method = HttpMethod.Put;
        request.RequestUri = new Uri("https://thunderclient.com/api/changeEmail");
        request.Headers.Add("Content-Type", "application/json");
        var bodyString = "{ \"username\": " + username + ", \"newEmail\": " + newEmail + "}";
        var content = new StringContent(bodyString, Encoding.UTF8, "application/json");
        request.Content = content;
        var response = await client.SendAsync(request);
        var result = await response.Content.ReadAsStringAsync();
        return response.IsSuccessStatusCode;
    }
}

```

Abbildung 55: Email ändern

```

private async void btnUpdateName_Clicked(object sender, EventArgs e)
{
    if (Application.Current.Properties["token"] == null)
    {
        var token = Application.Current.Properties["token"] as string;
        var firstName = Application.Current.Properties["firstName"] as string;
        var lastName = Application.Current.Properties["lastName"] as string;
        var schoolId = Application.Current.Properties["schoolId"] as string;
        var inputSchool = Application.Current.Properties["inputSchool"] as string;
        var identityId = Application.Current.Properties["identityId"] as string;
        var tokenString = Application.Current.Properties["tokenString"] as string;
        var loginService = new LoginService();
        var serviceSetName = new ServiceSetName();
        serviceSetName.Username = username;
        serviceSetName.FirstName = firstName;
        serviceSetName.LastName = lastName;
        serviceSetName.SchoolId = schoolId;
        serviceSetName.IdentityId = identityId;
        serviceSetName.Token = token;
        serviceSetName.TokenString = tokenString;
        if (loginService != null)
        {
            if (loginService.Login("Success", "Ihr Name wurde erfolgreich aktualisiert.", "Okay"))
            {
                Application.Current.Properties["firstName"] = firstName;
                Application.Current.Properties["lastName"] = lastName;
                Application.Current.Properties["schoolId"] = schoolId;
                Application.Current.Properties["identityId"] = identityId;
            }
        }
        ListviewUserData();
    }
}

```

```

public async Task<bool> SetName(string username, string firstName, string lastName)
{
    SetNameModel model = new SetNameModel() { Username = username, FirstName = firstName, LastName = lastName };
    HttpClientHandler httpclientHandler = new HttpClientHandler();
    httpclientHandler.ServerCertificateCustomValidationCallback = (s, cert, chain, errors) => true;
    using (var client = new HttpClient(httpclientHandler))
    {
        client.DefaultRequestHeaders.Add("Accept", "application/json");
        client.DefaultRequestHeaders.Add("Content-Type", "application/json");
        client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);
        var request = new HttpRequestMessage();
        request.Method = HttpMethod.Put;
        request.RequestUri = new Uri("https://thunderclient.com/api/changeName");
        request.Headers.Add("Content-Type", "application/json");
        var bodyString = "{ \"username\": " + username + ", \"firstName\": " + firstName + ", \"lastName\": " + lastName + "}";
        var content = new StringContent(bodyString, Encoding.UTF8, "application/json");
        request.Content = content;
        var response = await client.SendAsync(request);
        var result = await response.Content.ReadAsStringAsync();
        return response.IsSuccessStatusCode;
    }
}

```

Abbildung 56: Vorname und Nachname ändern

```

private async void btnUpdateSchool_Clicked(object sender, EventArgs e)
{
    var token = Application.Current.Properties["token"] as string;
    var identityId = Application.Current.Properties["identityId"] as string;
    if (Application.Current.Properties["firstName"] is null)
    {
        Application.Current.Properties["firstName"] = String.Empty;
    }
    if (Application.Current.Properties["lastName"] is null)
    {
        Application.Current.Properties["lastName"] = String.Empty;
    }
    var firstName = Application.Current.Properties["firstName"] as string;
    var lastName = Application.Current.Properties["lastName"] as string;
    if (Application.Current.Properties["schoolId"] is null)
    {
        Application.Current.Properties["schoolId"] = String.Empty;
    }
    var school = Application.Current.Properties["school"] as string;
    var inputSchool = Application.Current.Properties["inputSchool"] as string;
    var loginService = new LoginService();
    var serviceSetSchool = new ServiceSetSchool();
    serviceSetSchool.FirstName = firstName;
    serviceSetSchool.LastName = lastName;
    serviceSetSchool.SchoolId = school;
    serviceSetSchool.InputSchool = inputSchool;
    if (loginService != null)
    {
        if (serviceSetSchool != null)
        {
            if (serviceSetSchool.SetData(firstName, lastName, token, identityId, inputSchool))
            {
                if (serviceSetSchool.IsSuccessful)
                {
                    await DisplayAlert("Success", "Die User Daten wurden erfolgreich aktualisiert.", "Okay");
                    Application.Current.Properties["school"] = inputSchool;
                }
                else
                {
                    await DisplayAlert("Error", "Bitte wiederholen.", "Okay");
                }
            }
        }
        ListviewUserData();
    }
}

```

```

public async Task<bool> SetSchool(string firstName, string lastName, string school, string inputSchool, string identityId, string token)
{
    ServiceSetSchool model = new ServiceSetSchool() { FirstName = firstName, LastName = lastName, SchoolId = school, InputSchool = inputSchool, IdentityId = identityId, Token = token };
    HttpClientHandler httpclientHandler = new HttpClientHandler();
    httpclientHandler.ServerCertificateCustomValidationCallback = (s, cert, chain, errors) => true;
    using (var client = new HttpClient(httpclientHandler))
    {
        client.DefaultRequestHeaders.Add("Accept", "application/json");
        client.DefaultRequestHeaders.Add("Content-Type", "application/json");
        client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);
        var request = new HttpRequestMessage();
        request.Method = HttpMethod.Put;
        request.RequestUri = new Uri("https://thunderclient.com/api/changeSchool");
        request.Headers.Add("Content-Type", "application/json");
        var bodyString = "{ \"firstName\": " + firstName + ", \"lastName\": " + lastName + ", \"school\": " + school + ", \"inputSchool\": " + inputSchool + ", \"identityId\": " + identityId + ", \"token\": " + token + "}";
        var content = new StringContent(bodyString, Encoding.UTF8, "application/json");
        request.Content = content;
        var response = await client.SendAsync(request);
        var result = await response.Content.ReadAsStringAsync();
        return response.IsSuccessStatusCode;
    }
}

```

Abbildung 57: Schule ändern

Um ein Profil zu löschen, müssen Sie eine gültige E-Mail-Adresse eingeben. Dann werden die Daten an den LoginService gesendet und über RestClient prüfen wir, ob das Passwort korrekt ist. HttpClient sendet uns eine Antwort, und wenn dies zutrifft, werden die Daten und das Konto aus der Datenbank und dem Server gelöscht.

```

private void btnPasswortAendern_Clicked(object sender, EventArgs e)
{
    var oldPass = await DisplayPromptAsync("Passwort ändern", "Alt Passsword eingeben");
    if (oldPass == null)
    {
        return;
    }
    else
    {
        var newPass = await DisplayPromptAsync("Passwort ändern", "Neues Passsword eingeben");
        if (newPass == null)
        {
            return;
        }
        else
        {
            var username = Application.Current.Properties["username"].ToString();
            LoginService serviceLogin = new LoginService();
            var client = new HttpClient();
            var token = Application.Current.Properties["token"];
            var password = Application.Current.Properties["password"];
            var newPassword = Application.Current.Properties["newpassword"];
            var confirmPassword = Application.Current.Properties["confirmnewpass"];
            var model = new SetPassModel();
            model.oldPass = oldPass;
            model.newPass = newPassword;
            model.confirmNewPass = confirmPassword;
            var response = await serviceLogin.setPass(model);
            if (response != null)
            {
                await DisplayAlert("Success", "Ihre Daten wurden erfolgreich aktualisiert.", "Okay");
            }
            else
            {
                await DisplayAlert("Error", "Bitte schließen. Die Daten sind ungültig oder entsprechen nicht den Kriterien. Das neue Passwort muss: (Mindestens 8 Zeichen, mindestens eine Ziffer, mindestens ein Sonderzeichen und darf kein Passwort aus der Vergangenheit sein)", "Okay");
            }
        }
    }
}

```

```

public static Task<bool> setPass(string username, string password, string newPassword)
{
    SetPass model = new SetPass();
    model.Username = username;
    model.Password = password;
    model.NewPass = newPassword;
    model.ConfirmNewPass = confirmPassword;

    HttpClient client = new HttpClient();
    client.DefaultRequestHeaders.Add("Content-Type", "application/json");
    client.DefaultRequestHeaders.Add("User-Agent", "Thunder Client (https://www.thunderclient.com)");
    var request = new HttpRequestMessage();
    request.Method = HttpMethod.Post;
    request.Headers.Add("Accept", "*/*");
    request.Headers.Add("User-Agent", "Thunder Client (https://www.thunderclient.com)");
    request.Headers.Add("Content-Type", "application/json");
    request.Content = new StringContent(model.ToString(), Encoding.UTF8, "application/json");
    var context = new StringContent("body=try{JSON.parse(response)}catch(e){e.message}", Encoding.UTF8, "application/json");
    request.Content = context;
    var result = await client.PostAsync(request);
    var response = await result.Content.ReadAsStringAsync();
    var resultString = JsonConvert.DeserializeObject<SetPassResponse>(resultString);
    return response.IsSuccessStatusCode;
}

```

Abbildung 58: Password ändern

```

private async void btnDeleteKonto_Clicked(object sender, EventArgs e)
{
    var password = await DisplayPromptAsync("Konto löschen", "Passwort eingeben");
    if (password == null)
    {
        return;
    }
    else
    {
        if (password.Length == 0)
        {
            await DisplayAlert("Error", "Bitte wiederholen.", "Okay");
        }
        else
        {
            var token = Application.Current.Properties["token"].ToString();
            var username = Application.Current.Properties["username"].ToString();

            LoginService serviceDeleteUser = new LoginService();
            var delete = await serviceDeleteUser.deleteUser(username, password, token);
            if (delete == true)
            {
                await DisplayAlert("Success", "Ihre Konto wurde erfolgreich gelöscht.", "Okay");
            }
            else
            {
                await DisplayAlert("Error", "Account wurde fehlgeschlagen. Bitte Loggen überprüfen und erneut versuchen.", "Okay");
            }
        }
    }
}

```

```

public async Task<bool> deleteUser(string username, string password, string token)
{
    DeleteUser model = new DeleteUser();
    model.Username = username;
    model.Password = password;
    model.Token = token;

    HttpClient client = new HttpClient();
    client.DefaultRequestHeaders.Add("Content-Type", "application/json");
    client.DefaultRequestHeaders.Add("User-Agent", "Thunder Client (https://www.thunderclient.com)");
    var request = new HttpRequestMessage();
    request.Method = HttpMethod.Delete;
    request.Headers.Add("Accept", "*/*");
    request.Headers.Add("User-Agent", "Thunder Client (https://www.thunderclient.com)");
    request.Headers.Add("Content-Type", "application/json");
    var context = new StringContent("body=try{JSON.parse(response)}catch(e){e.message}", Encoding.UTF8, "application/json");
    request.Content = context;
    var response = await client.SendAsync(request);
    var resultString = JsonConvert.DeserializeObject<DeleteUserResponse>(resultString);
    return response.IsSuccessStatusCode;
}

```

Abbildung 59: Konto löschen

## 6.2.4 Statistik

Beim Aufrufen der Statistikseite ziehen wir die Daten vom Server aus dem Backend. Aus den gespeicherten Daten prüfen wir, ob es sich um einen Schüler oder einen Lehrer handelt, und rufen die korrekten Daten über LoginService und Statistics Task ab.

```

private void buttonMyStat_Clicked(object sender, EventArgs e)
{
    var token = Application.Current.Properties["token"].ToString();
    var userId = (int)Application.Current.Properties["userId"];
    LoginService serGetUserStat = new LoginService();
    var getStat = await serGetUserStat.GetUserStatistic(token, userId);

    AllUnit_FeedbackData jsonCreatedTeachingUnitsCount = JsonConvert.DeserializeObject<AllUnit_FeedbackData>(getStat);
    int createdTeachingUnitsCount = jsonCreatedTeachingUnitsCount.CreatedTeachingUnitsCount;

    AllUnit_FeedbackData jsonCreatedFeedbacksCount = JsonConvert.DeserializeObject<AllUnit_FeedbackData>(getStat);
    int createdFeedbacksCount = jsonCreatedFeedbacksCount.CreatedFeedbacksCount;

    AllUnit_FeedbackData jsonAvgStars = JsonConvert.DeserializeObject<AllUnit_FeedbackData>(getStat);
    double avgStars = jsonAvgStars.AvgStars;

    Application.Current.Properties["createdTeachingUnitsCount"] = createdTeachingUnitsCount;
    Application.Current.Properties["createdFeedbacksCount"] = createdFeedbacksCount;
    Application.Current.Properties["avgStars"] = avgStars;

    var role = Application.Current.Properties["role"].ToString();

    if(role == "student")
    {
        await DisplayAlert("Statistik!", "Sie haben " + createdFeedbacksCount + " Feedbacks gegeben.", "Okay");
    }
    else
    {
        await DisplayAlert("Statistik!", "Sie haben " + createdTeachingUnitsCount + " Einheiten erstellt. Ihr AVG ist: " + avgStars, "Okay");
    }
}

```

Abbildung 60: Statistik Button

Wir holen uns die Daten per httpClient vom Server und übertragen die Daten in den Response-String, damit wir ihn lesen können.

```
1 Verweis
public async Task<String> GetUserStatistik(string token, int userId)
{
    GetUserStatistik model = new GetUserStatistik() {Token=token, UserId=userId};

    HttpClientHandler httpClientHandler;
    #if (DEBUG)
    httpClientHandler = new HttpClientHandler { ServerCertificateCustomValidationCallback = (o, cert, chain, errors) => true };
    #else
    httpClientHandler = new HttpClientHandler(); /*endif using (var client = new HttpClient(httpClientHandler));
    #endif

    var client = new HttpClient(httpClientHandler);
    var request = new HttpRequestMessage();
    request.RequestUri = new Uri("https://10.0.2.2:5001/api/statistic/userStats?userId=" + userId);
    request.Method = HttpMethod.Get;

    request.Headers.Add("Accept", "*/*");
    request.Headers.Add("User-Agent", "Thunder Client (https://www.thunderclient.com)");
    request.Headers.Add("Authorization", "Bearer " + token);

    var response = await client.SendAsync(request);
    var result = await response.Content.ReadAsStringAsync();
    return result;
}
```

Abbildung 61: HttpClient

# **7 Nicht realisierte Funktionen**

Aufgrund von diversen Schwierigkeiten, die wir während der Diplomarbeit konfrontiert waren, konnten einige Funktionen der App nicht rechtzeitig implementiert werden.

## **7.1 Feedback Kategorien**

Der Lehrende hätte bei der Erstellung bei der Lehreinheit, eigene Kategorien mit Sternen oder Kommentar anlegen können, um ein präziseres Feedback von den Schülern erhalten zu können.

## **7.2 Export als PDF**

Diese Funktion sollte den Ausdruck und externe Speicherung von Statistiken, Lehreinheiten und deren Bewertungen vereinfachen.

## **7.3 digitale Erfolge/Abzeichen**

Bei dieser Idee wurde die PC-Spiele Plattform Steam als Vorbild genommen, die gewisse Erfolge mittels Abzeichen die im Benutzerprofil öffentlich sichtbar gemacht werden können, um die Motivation für gewisse nicht verpflichtende Tätigkeiten im Spiel zu steigern. Dies sollte in der Feedback App ebenfalls möglich gemacht werden. Zudem kann der Lehrende auch sehen, ob der Bewerter aufgrund bestimmter Erfolgsabzeichen eventuell mehr Erfahrung mit dem Feedback geben hat und somit die Bewertung ehrlicher und eine mehr konstruierte Rückmeldung gegenüber anderen hat.

## **7.4 Benachrichtigungen**

Auf den mobilen Endgerät sollte der Benutzer mittels benachrichtigt werden, wenn er Feedback von anderen Benutzern erhalten hat oder seine Lehreinheit das Ablaufdatum erreicht hat. Zudem war eine Benachrichtigung mittel E-Mail geplant.

## 7.5 Two-Faktor Authentifizierung

Der Benutzer hätte entscheiden können, ob er die sichere 2-Faktor Anmeldemethode verwendet. Dabei handelt es sich um eine Eingabe eines zweiten Kennworts, dass automatisch von einem externen Dienst wie Google Authenticator erstellt wurde. Dieser Pin ist zeitlich begrenzt und erhöht damit erheblich die Sicherheit, um den Missbrauch eines Accounts zu verhindern.

## 7.6 Benutzerkonto Profilfoto

Die Hinzufügung eines Profilbilds hätte der User sein Benutzerkonto individueller gestalten können. Diese Bild wäre in einem platzsparenden Format (JPEG) mit Höhen- und Längenbegrenzung auf dem Server mit der Feedback Datenbank in der Tabelle User hochgeladen worden.

# **8 Zusammenfassung**

Im Laufe der Arbeiten traten Verdrießlichkeiten auf, die zwar nicht unlösbar waren, trotz alledem indes langfristige Ergebnisse erforderten. Einige Fakten waren schwierig zusammenzustellen. Es hat einige Zeit gedauert, sämtliche Einfälle zusammenzubringen und Leistungsmerkmale zu erhalten, die miteinander wirken. Das erstmalige Problem war die plötzliche Kündigung von Teammitgliedern, sodass wir ausschließlich noch 2 von 4 Teammitgliedern hatten. Ein weiteres Problem sind fehlende Kenntnisse und Erfahrungen mit Xamarin, wie auch Tutorials (Anleitungen) aus dem World Wide Web, die mit unserer Ausführung lediglich schwer oder keineswegs zu inkludieren sind. Da der Xamarin-Emulator ausgeprägt Zeit braucht, um das virtuelle Endgerät vorzubereiten und die App auszuführen (dies tut er innerhalb jedem Start), erfordert er enorm Mühe und Geduld beim Testen, was für den Abschluss der Arbeit am essentiellsten ist.



# Literaturverzeichnis

- [1] M. Kronawetter, G. Andre *et al.*, *Betriebswirtschaft und Management IV/V Auflage 2018*. MANZ Verlag, 2018.
- [2] D. J. Fleig, „Lernen durch Feedback,” Business-Wissen, November 2020. Online verfügbar: [www.business-wissen.de/hb/feedback-geben-warum-feedback-wichtig-ist/#:~:text=Feedback%20sorgt%20fÃijr%20Transparenz&text=Das%20Feedback%20geben%20dient%20dazu,undere%20Person%20besser%20zu%20verstehen](http://www.business-wissen.de/hb/feedback-geben-warum-feedback-wichtig-ist/#:~:text=Feedback%20sorgt%20fÃijr%20Transparenz&text=Das%20Feedback%20geben%20dient%20dazu,undere%20Person%20besser%20zu%20verstehen).
- [3] S. emu, „Johari Fenster,” via Wikimedia Commons - CC BY-SA 4.0, Juli 2018. Online verfügbar: <https://commons.wikimedia.org/wiki/File:Johari2.svg>
- [4] Andres15alvarez, „logo de C#,“ via Wikimedia Commons - CC BY-SA 4.0, Februar 2018. Online verfügbar: [https://commons.wikimedia.org/wiki/File:Csharp\\_Logo.png](https://commons.wikimedia.org/wiki/File:Csharp_Logo.png)
- [5] Microsoft, „Logo of Xamarin,” via Wikimedia Commons - Public Domain. Online verfügbar: <https://commons.wikimedia.org/wiki/File:Xamarin-logo.svg?uselang=de>
- [6] K. Wilkinson, „DotNet-Bot,” via Wikimedia Commons - CC0 - .NET Foundation, Mai 2020. Online verfügbar: <https://commons.wikimedia.org/wiki/File:Dotnet-bot.svg>
- [7] A. Kühnel, *C# 8 mit Visual Studio 2019, Das umfassende Handbuch, 8. Auflage*, A. Scheibe, Hrsg. Rheinwerk Verlag, 2019.
- [8] „.NET documentation,” Microsoft .NET Documentation. Online verfügbar: <https://docs.microsoft.com/en-us/dotnet/fundamentals/>
- [9] „.NET,” Wikipedia. Online verfügbar: <https://en.wikipedia.org/wiki/.NET>
- [10] „ASP.NET documentation,” Microsoft ASP.NET documentation. Online verfügbar: <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0>
- [11] D. H. Schwichtenberg, „Was ist ASP.NET Core Razor Pages?” IT Visions de. Online verfügbar: <https://www.dotnetframework.de/%7B8B3FCCE5-01A5-42A6-9781-3032AB9CAA6F%7D.aspx>
- [12] „ASP.NET Core,” Wikipedia. Online verfügbar: [https://en.wikipedia.org/wiki/ASP.NET\\_Core](https://en.wikipedia.org/wiki/ASP.NET_Core)
- [13] „Introduction to Identity on ASP.NET Core,” Microsoft ASP.NET Documentation. Online verfügbar: <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio>
- [14] „Introduction to Identity,” Microsoft Documentation, 2016. Online verfügbar: <https://jakeydocs.readthedocs.io/en/latest/security/authentication/identity.html>

- [15] „Entity Framework Core,” Microsoft EF Documentation. Online verfügbar: <https://docs.microsoft.com/en-us/ef/core/>
- [16] „Entity Framework,” Wikipedia. Online verfügbar: [https://en.wikipedia.org/wiki/Entity\\_Framework](https://en.wikipedia.org/wiki/Entity_Framework)
- [17] MovGP0, „ADO.NET Entity Framework,” via Wikimedia Commons - CC BY-SA 3.0, November 2012. Online verfügbar: [https://commons.wikimedia.org/wiki/File:ADO.NET\\_EF.svg](https://commons.wikimedia.org/wiki/File:ADO.NET_EF.svg)
- [18] „What Is OpenAPI?” Official OpenAPI Swagger Documentation. Online verfügbar: <https://swagger.io/docs/specification/about/>
- [19] „OpenAPI,” Wikipedia. Online verfügbar: <https://de.wikipedia.org/wiki/OpenAPI>
- [20] „Was ist OpenAPI?” Digital Guide IONOS Artikel, Juni 2022. Online verfügbar: <https://www.ionos.at/digitalguide/websites/web-entwicklung/was-ist-openapi/>
- [21] Camwilliams96, „Open API Chart,” via Wikimedia Commons - CC BY-SA 4.0, November 2015. Online verfügbar: <https://commons.wikimedia.org/wiki/File:Open-APIs-v5.png>
- [22] „Introduction to JSON Web Tokens,” JSON Web Token Official Website. Online verfügbar: <https://jwt.io/introduction>
- [23] D.-I. F. S. Luber, „Was ist JSON Web Token (JWT)?” Security Insider Artikel, Jänner 2022. Online verfügbar: <https://www.security-insider.de/was-ist-json-web-token-jwt-a-1094265/>
- [24] „Xamarin.” Online verfügbar: <https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin>
- [25] „Xamarin Logo.” Online verfügbar: [https://de.wikipedia.org/wiki/Datei:Xamarin\\_logo\\_and\\_wordmark.png#/media/Datei:Xamarin\\_logo\\_and\\_wordmark.png](https://de.wikipedia.org/wiki/Datei:Xamarin_logo_and_wordmark.png#/media/Datei:Xamarin_logo_and_wordmark.png)
- [26] „Visual Studio.” Online verfügbar: [https://de.wikipedia.org/wiki/Visual\\_Studio#Version\\_2022](https://de.wikipedia.org/wiki/Visual_Studio#Version_2022)
- [27] „Visual Studio Logo.” Online verfügbar: [https://logos.fandom.com/wiki/Microsoft\\_Visual\\_Studio](https://logos.fandom.com/wiki/Microsoft_Visual_Studio)
- [28] Microsoft, „VS Code Logo,” via Wikimedia Commons - Public domain, Juni 2021. Online verfügbar: [https://commons.wikimedia.org/wiki/File:Visual\\_Studio\\_Code\\_1.35\\_icon.svg](https://commons.wikimedia.org/wiki/File:Visual_Studio_Code_1.35_icon.svg)
- [29] Serogen, „VS Code 1.36.0-insider,” via Wikimedia Commons - MIT, Juni 2019. Online verfügbar: [https://commons.wikimedia.org/wiki/File:VS\\_Code\\_1.36.0-insider.png](https://commons.wikimedia.org/wiki/File:VS_Code_1.36.0-insider.png)
- [30] „Why did we build Visual Studio Code?” VS Code Official Documentation. Online verfügbar: <https://code.visualstudio.com/docs/editor/whyvscode>
- [31] „Visual Studio Code,” Wikipedia. Online verfügbar: [https://en.wikipedia.org/wiki/Visual\\_Studio\\_Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)
- [32] Lars, „Microsoft Visual Studio Code: Wir erklären die geniale Entwicklerumgebung,” Windows United Article, Dezember 2018. Online verfügbar: <https://windowsunited.de/microsoft-visual-studio-code-wir-erklaeren-die-geniale-entwicklerumgebung/>

- [33] R. Vadhineni, „Thunder Client — lightweight alternative to Postman,” Medium Blog, März 2021.
- [34] S. Lee, „ERD Editor Extension,” Github. Online verfügbar: <https://github.com/vuerd/vuerd#readme>
- [35] „GitHub.” Online verfügbar: <https://de.wikipedia.org/wiki/GitHub>
- [36] „GitHub Logo.” Online verfügbar: <https://github.com/logos>
- [37] Microsoft, „Visual Studio Emulator für Android,” MS Android Emulator Homepage. Online verfügbar: <https://visualstudio.microsoft.com/de/vs/msft-android-emulator/>
- [38] „iOs Emulator on Windows.” Online verfügbar: <https://docs.microsoft.com/de-de/visualstudio/mac/xamarin?view=vsmac-2022>
- [39] „iOs Emulator on Windows.” Online verfügbar: <https://docs.microsoft.com/en-us/xamarin/tools/ios-simulator/>

# Abbildungsverzeichnis

1	Johari-Fenster . . . . .	4
2	overview architecture . . . . .	6
3	ERD FeedbackDb . . . . .	7
4	Identity Tables . . . . .	9
5	models API . . . . .	10
6	Xamarin.Forms Project Ansicht . . . . .	11
7	Project Ansicht . . . . .	12
8	Anwendungsgrundlagen Ansicht . . . . .	13
9	Community Maskottchen von .NET . . . . .	14
10	Prinzipielle Funktionsweise des ADO.NET Entity Framework . . . . .	22
11	OpenAPI Business Diagramm . . . . .	27
12	Bsp JWT in Debugger . . . . .	31
13	Ablauf JWT Authentifizierung . . . . .	31
14	Xamarin Logo . . . . .	33
15	VS 2022 Logo . . . . .	35
16	Visual Studio Code Logo [28] . . . . .	37
17	Visual Studio Code Screenshot [29] . . . . .	37
18	GitHub Logo . . . . .	40
19	LoginPage Ansicht . . . . .	45
20	LoginPage Ansicht . . . . .	46
21	HomePage Student Ansicht . . . . .	47
22	HomePage Einheiten Ansicht . . . . .	48
23	HomePage Feedback geben . . . . .	49
24	HomePage Lehrer Ansicht . . . . .	50
25	HomePage Unit Ansicht . . . . .	51
26	Registrierung Ansicht . . . . .	52
27	Registrierung Ansicht . . . . .	53
28	MyAccount Ansicht . . . . .	54
29	MyAccount Namensänderung . . . . .	55
30	MyAccount Umbenennung der Schule . . . . .	56
31	MyAccount Änderung der E-Mail-Adresse . . . . .	57
32	MyAccount Passwort . . . . .	59
33	MyAccount Konto löschen . . . . .	61
34	Statistiken Schüler Ansicht . . . . .	61
35	Statistiken Lehrer Ansicht . . . . .	62
36	FeedbackDbContext . . . . .	63
37	TeachingUnitModel . . . . .	64
38	overview Repos . . . . .	65
39	Userstats . . . . .	66
40	DelTeachingUnit . . . . .	66
41	Swagger Benutzerverwaltung . . . . .	67
42	PW User Requirements Startup . . . . .	68

43	JWT create . . . . .	69
44	Login . . . . .	70
45	Login . . . . .	70
46	Login . . . . .	71
47	Login . . . . .	71
48	Login . . . . .	71
49	Registrierung . . . . .	72
50	Registrierung . . . . .	72
51	Registrierung . . . . .	72
52	MyAccount . . . . .	73
53	MyAccount . . . . .	73
54	MyAccount . . . . .	73
55	MyAccount . . . . .	74
56	MyAccount . . . . .	74
57	MyAccount . . . . .	74
58	MyAccount . . . . .	75
59	MyAccount . . . . .	75
60	Statistik . . . . .	75
61	Statistik . . . . .	76

# **Tabellenverzeichnis**

# **Quellcodeverzeichnis**

# **Anhang**