

# M335

Teile die wichtigen Dinge  
im Leben. Memes!

## MemeChat — MAUI-App



Manuel Schumacher

pebe AG

IN20-24c

Michael Heizmann

07.10.2022

# INHALTSVERZEICHNIS

1	Einleitung .....	5
1.1	Zeitplan .....	5
2	Konzept .....	6
2.1	App - Möglichkeiten.....	6
2.1.1	Flutter .....	7
2.1.2	Cordova .....	8
2.1.3	Xamarin.....	8
2.1.4	MAUI .....	9
2.1.5	Verwendung .....	10
2.2	Sensoren.....	10
2.2.1	Verwendung .....	11
2.3	Datenbank.....	11
2.3.1	Relational .....	11
2.3.2	NoSQL .....	11
2.3.3	Graph.....	12
2.3.4	Verwendung .....	12
2.4	Persistenz.....	13
2.4.1	Datei Persistenz .....	13
2.4.2	Synchronisation.....	14
2.5	Entwicklungsumgebungen.....	14
2.5.1	Node .....	14
2.5.2	Gradle .....	14
2.5.3	Frameworks .....	14
2.6	IDEs .....	15
2.6.1	Android Studio .....	15
2.6.2	Visual Studio Code.....	16
2.6.3	Visual Studio .....	16
2.6.4	Verwendung .....	17
2.7	Usability.....	17

2.8	Testverfahren .....	18
2.8.1	Arten .....	19
2.8.2	Integration Testing .....	19
2.8.3	Load & Stress Testing .....	19
2.8.4	UI-Testing .....	20
2.8.5	Unit Test .....	20
2.8.6	Mocking .....	20
2.8.7	Blackbox .....	20
2.8.8	Whitebox .....	21
2.8.9	A / B .....	21
2.8.10	Testfälle .....	22
2.9	Veröffentlichen .....	22
2.9.1	Beachten .....	23
2.9.2	Release .....	24
2.10	Planung .....	24
2.10.1	Name .....	25
2.10.2	Beschreibung .....	25
2.10.3	Icons .....	25
2.10.4	Hintergrund .....	26
2.10.5	Mockups .....	26
2.10.6	Meine Wireframes .....	26
2.10.7	Anmeldeseite .....	28
2.10.8	Chats .....	29
2.10.9	Tablett .....	30
2.10.10	Konkurrenz .....	31
3	Systemdokumentation .....	32
3.1	Installation .....	32
3.1.1	Cordova .....	33
3.1.2	Visual Studio .....	34
3.1.3	Versionen .....	36

3.1.4	Umgebungsvariablen.....	37
3.2	Umsetzung Sensor .....	38
3.3	Datenbank.....	39
3.3.1	Installation .....	40
3.3.2	Unterstützte Datenbanken .....	41
3.4	Persistenz.....	41
3.5	Tests.....	41
3.6	Simulationsumgebungen .....	42
3.6.1	Unit Test.....	42
3.6.2	Emulator .....	42
3.7	Probleme.....	43
3.8	Bedienungsanleitung .....	44
4	Testdokumentation .....	46
5	Implementation.....	47
5.1	Sensor .....	47
5.1.1	Probleme.....	49
5.2	Datenbank.....	49
5.2.1	Abfragen.....	50
5.2.2	Speichern .....	50
5.2.3	Backup.....	50
5.2.4	Aufsetzen.....	51
5.2.5	Probleme.....	51
5.3	Projekt.....	52
5.3.1	Cordova Projekt.....	52
5.3.2	MAUI-Projekt .....	52
5.3.3	MVVM .....	54
5.3.4	.NET .....	55
5.3.5	MudBlazor.....	55
5.4	Ordnerstruktur .....	57
5.4.1	Komponenten .....	58

# MemeChat

5.4.2	Datenbank.....	58
5.4.3	Tests .....	58
5.5	Einstellungen .....	59
5.5.1	Informationen.....	59
5.5.2	Icons & Startlogo .....	60
5.5.3	Same-Origin-Policy.....	60
5.5.4	Sicherheit.....	61
5.6	Veröffentlichung.....	61
5.6.1	App Store.....	62
5.6.2	Signieren.....	65
6	Schlusswort .....	67

# 1 Einleitung

Jeder Messenger, welcher auf sich auf einem Mobilgerät vorinstalliert ist, kann Nachrichten versenden. Doch manchmal findet man in einer Konversation nicht die richtigen Worte. Genau dieses Problem sollte MemeChat beheben. Durch das Versenden von Memes wird die Unterhaltung auf die wichtigen Dinge im Leben gerichtet. Das Projekt<sup>1</sup> ist auf jeder Plattform (Android, iOS, macOS & Windows) ausführbar, da sie mit MAUI entwickelt wurde.

Heutzutage hat jede Person mindestens ein Mobiltelefon und ist ständig mit dem Internet verbunden. Jedes dieser Geräte bietet die Möglichkeit an Apps zu installieren, was ein Markt mit unheimlich viel Potenzial zur Verfügung stellt. Für jede vorstellbare Idee gibt es auch eine App, welche das Leben vereinfachen oder unterhalten kann. Allein auf dem Apple App Store<sup>2</sup> gibt es zurzeit über 2 Millionen unterschiedliche Apps. Der Google Play Store übertrifft diese Nummer nochmals um die Hälfte.

Jede Person mit einem Laptop und Internetverbindung kann eine App entwickeln. Wie dies gemacht wird und was es dabei zu beachten gibt wird in dieser Dokumentation beschrieben. Zudem enthält sie die Implementation eines eigenen Apps. Dieses Dokument wurde im ÜK 335 erstellt, in welchem das Ziel ist eine App zu realisieren. Das umfangreiche Projekt wurde innerhalb einer Woche während und nach den Schullektionen umgesetzt.

## 1.1 Zeitplan

Für diesen ÜK sind 5 Tage vorgesehen, in welchen das ganze Projekt fertiggestellt wird. Im ganzen wurde jedoch 23 Stunden mehr daran gearbeitet als eigentlich geplant.

TAG	AUFGABE
MONTAG	Grundlagen & Konzept
DIENSTAG	Design & Entwicklungsumgebung
MITTWOCH	Datenbanken & Sensoren
DONNERSTAG	Umsetzung & Verbesserung
FREITAG	Tests & Abgabe

---

<sup>1</sup> <https://github.com/21r8390/m335>

<sup>2</sup> <https://www.statista.com/statistics/779768/number-of-available-apps-in-the-apple-app-store-quarter/>

## 2 Konzept

Die Planung einer Applikation ist der wichtigste Schritt bei der Entwicklung. Nicht nur können unzählige Fehler, sondern auch Schwierigkeiten, welche ansonsten viel Zeitaufwand benötigen würden, behoben werden. Aus diesem Grund wurde dafür sehr viel Zeit aufgewendet. Als erstes wird auf die unterschiedlichen Arten von Apps eingegangen. Dies muss vor der Entwicklung bekannt sein, damit die Einschränkungen bekannt sind. Daraufhin können die Frameworks und IDEs ausgesucht werden, welche die meiste Arbeit der Entwicklung übernehmen sollten. Wenn alle technischen Anforderungen geklärt sind, dann kann es zum Design kommen. Dort eignet sich ein Wireframe oder Mockup, um das Navigationskonzept zu erstellen. Dabei muss auf Usability geachtet werden, damit sichergestellt wird, dass alle Personen die App verwenden können. Zum Schluss muss die Persistenz der Daten zum Beispiel über eine Datenbank gewährleistet werden, damit die Daten auch bei einem Neustart auffindbar sind. Wenn die Applikation fertiggestellt und getestet wurde, dann kann sie veröffentlicht werden.

### 2.1 App - Möglichkeiten

Es gibt drei verschiedene Arten, wie man ein App entwickeln kann. Je nach gewählter Möglichkeit<sup>3</sup> gibt es Vor- und Nachteile. Sie können Native, Hybrid oder Webbasieren erstellt werden. Native-Apps sind in einer spezifischen Programmiersprache für genau eine einzige Plattform entwickelt. Somit kann man entweder iOS oder Android als Zielplattform haben. Für iOS würde man die hauseigene Programmiersprache Swift<sup>4</sup> verwenden. Für die Entwicklung auf Android Plattformen eignet sich Java<sup>5</sup> oder Kotlin<sup>6</sup> ausgezeichnet. Der Vorteil von diesen Apps ist, dass sie sehr performant sind und alle Sensoren des Betriebssystems unterstützen. Der Nachteil ist jedoch, dass es für jede Plattform eine eigene Codebasis erstellt werden muss. Dies kann sehr schnell teuer werden, da beide Basen regelmässige Updates brauchen. Die Lösung dafür sind Hybrid-Apps, welche die Codebasis über mehrere Plattformen teilen. So kann nicht nur Geld, sondern auch Ummengen an Geld gespart werden. Für grosse Firmen ist diese Art besser geeignet, da es ohne Probleme auf grössere Benutzergruppen skaliert werden kann. Der Zugang zu Gerätefeatures besteht, ist jedoch nur eingeschränkt verfügbar. Der Nachteil ist, dass es auf jedem Gerät testen und sicherstellen muss, dass alle Funktionen funktionieren. Cordava<sup>7</sup> ist einer dieser Frameworks, welche das Entwickeln über mehrere Geräte gleichzeitig erlaubt. Alternativen dazu wären Flutter, Xamarin oder Ionic. Das letzte wäre die Entwicklung mithilfe von Webapplikationen. Früher war es so,

<sup>3</sup> <https://www.mobiloud.com/blog/native-web-or-hybrid-apps#5>

<sup>4</sup> <https://developer.apple.com/swift/>

<sup>5</sup> <https://www.java.com/>

<sup>6</sup> <https://kotlinlang.org/>

<sup>7</sup> <https://cordova.apache.org/>

dass Webseiten keine Mitteilungen erstellen könnten. Durch Progressive-Web-Apps (PWAs<sup>8</sup>) hat sich dies in den letzten Jahren geändert. Nun können mit HTML und JavaScript Webseiten erstellt werden, welche als Applikation auf dem System laufen. Dabei wird ein Browser im App selbst geöffnet, welcher die Webseite darstellt. Zugriff auf alle Sensoren gibt es immer noch nicht und nicht alles sieht auf jedem Betriebssystem gleich aus. Bei der Wahl der Entwicklungsart sollte man genau die Zielgruppe definieren und wissen was für Funktionen man braucht. Je nach Umfang muss eine andere Art verwendet werden. In der folgenden Liste werden die Vor- und Nachteile nochmals aufgelistet:

	WEB - APP	Hybrid - APP	Native - APP
Entwicklungskosten	klein	klein	hoch
Entwicklungszeit	klein	klein	lang
APP - Portabilität	gut	gut	schlecht
Geschwindigkeit	schnell	langsamer	sehr schnell
Funktionen	wenige	viele	alle
Native erstellbar	nein	ja	ja
Veröffentlichung	nein	ja	ja
Erweiterbar	nein	ja	ja

Abbildung 1 Tabelle Vor- & Nachteile

### 2.1.1 Flutter

Flutter<sup>9</sup> ist ein Open Source Framework von Google, welches die Entwicklung von Multiplattform Applikationen mit derselben Codebasis erlaubt. Als Zielsystem können Handys, Tablets, Desktops, Web und Embedded ausgewählt werden. Dafür wird die Programmiersprache Dart verwendet, welche vergleichbar mit der Syntax von Java und C# ist. Es ist noch aktiv in der Entwicklung, weswegen es öfters noch Funktionen gibt, welche nicht überall das machen, was sie sollten. Die Lernkurve ist zu Beginn sehr steil und mit vielen Hindernissen versehen. Sobald man sich mit der Materie auseinandergesetzt hat, wird der Umgang unglaublich einfach und es können wunderschöne UIs erstellt werden. Bei der Entwicklung ist Hot Reload behilflich, um schnell Anpassungen zu machen, ohne das App komplett neu zu starten.

---

<sup>8</sup> <https://web.dev/progressive-web-apps/>

<sup>9</sup> <https://flutter.dev/>

## 2.1.2 Cordova

Für die Entwicklung von Hybrid-Apps stehen verschiedenen Frameworks zur Verfügung. Eine der größten Entwicklungsplattformen ist Cordova der Apache Software Foundation und das darauf basierende PhoneGap von Adobe. Es bieten weitere Unternehmen wie Anscia Mobile mit dem Corona SDK und Drifty mit Ionic Lösungen zur Entwicklung von Hybrid-Apps an. Cordova wird in diesem ÜK verwendet, da es kostenlos und einfach zu erlernen ist. Mit einfachem Wissen über HTML, CSS und JavaScript kann man sehr schnell eine lauffähige Applikation erstellen. Es kann auf jedem Gerät ausgeführt werden, solange einen Browser vorhanden ist, da der Code zu 100% zwischen den Plattformen geteilt wird. Cordova ist eigentlich nur eine Render-Engine, welche das HTML UI darstellt. Durch diverse Plug-Ins können weitere APIs hinzugefügt werden, welche das Zugreifen auf Sensoren erlauben.

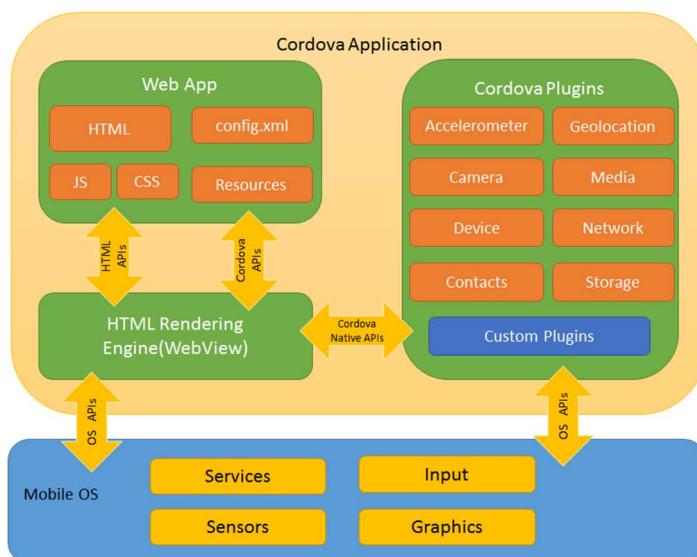


Abbildung 2 Cordova Aufbau<sup>10</sup>

Auf die Sensoren kann auch zugegriffen werden, wenn keine Internetverbindung vorhanden ist. Mit Leichtigkeit können Daten lokal auf dem Gerät gespeichert werden. Der Nachteil ist, dass es weniger performant als andere Entwicklungsumgebungen ist. Durch die Einfachheit und Flexibilität ist es nicht geeignet für größere Apps, da die Struktur schnell verläuft. Zusätzlich funktionieren die Interfaces nicht auf allen Versionen.

## 2.1.3 Xamarin

Xamarin ist für Entwickler gedacht, welche sich die Arbeit mit C# und .NET gewohnt sind. Für die Entwicklung ist ein Vorwissen von C# dringend notwendig, da auf den bestehenden Konzepten aufbaut. Es wird XAML geschrieben, welches dann für die verschiedenen Plattformen kompiliert

---

<sup>10</sup> <https://cordova.apache.org/docs/en/11.x/guide/overview/index.html>

wird, sodass es ein natives Erlebnis erreicht. Die Basis wird mit C# geschrieben und optimal nach dem Design Pattern Model View View-Model (MVVM<sup>11</sup>), damit das UI vom Source Code getrennt ist. Der Syntax von XAML ist gewöhnungsbedürftig und nicht für jedermann. Wenn man sich aber in die Projektstruktur und Funktionen eingearbeitet hat, dann kann man wunderschöne und performante Applikationen erstellen. Innerhalb der letzten 10 Jahren hat sich .NET und Xamarin unheimlich verändert.

#### 2.1.4 MAUI

Xamarin wurde 2022 von Multi-Plattform App UI (MAUI<sup>12</sup>) abgelöst. Das Konzept bleibt gleich, jedoch hat es im Hintergrund unzählige Änderungen gegeben. Unter anderem können nun Applikationen auch auf Desktops veröffentlicht werden. Das UI kann immer noch mit einem XML ähnlichen Syntax geschrieben werden. Es gibt jedoch auch die Möglichkeit mit Blazor<sup>13</sup>, was HTML als Darstellung verwendet, zu entwickeln. Es wird empfohlen, dass die Ordnerstruktur mit MVVM aufgebaut wird. MAUI kompiliert das App so, dass es sich wie nativ verhält, was eine bessere Benutzerführung erlaubt. Über .NET 6 wird der Code in ein Binärformat gebracht, welches dann über Common Language Runtime (CLR<sup>14</sup>) auf jeder Plattform ausgeführt werden kann. Im folgenden Bild wird dieses Verfahren aufgezeigt:

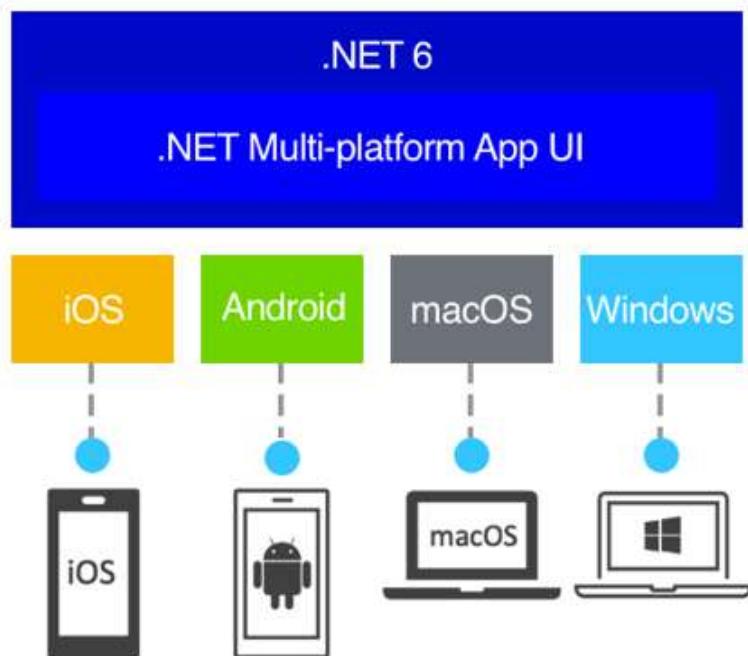


Abbildung 3 MAUI-Aufbau

<sup>11</sup> <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>

<sup>12</sup> <https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui>

<sup>13</sup> <https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor>

<sup>14</sup> <https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/>

### 2.1.5 Verwendung

Für mein Projekt verwende ich das neue Microsoft Framework MAUI, da ich im Betrieb bereits damit gearbeitet habe. Dort verwenden wir Blazor für das Server-Seitige-Rendering. Aus diesem Grund habe ich fast keine Einarbeitungszeit und minimaler Zeitaufwand. Durch die Verwendung von Blazor kann ich mein Wissen über HTML anwenden und muss den komplizierten XML-Syntax, welcher eigentlich verwendet wird, nicht benutzen. Ein weiterer Anlass für die Verwendung von MAUI ist, dass ich mein Wissen auffrischen und vertiefen möchte. Mir ist dabei klar, dass es noch ein neueres Produkt ist und Fehler existieren. Ich vertraue dem .NET-Team, dass sie diese Fehler bald beheben und Verbesserungen bieten.

## 2.2 Sensoren

Ein Smartphone hat unzählige Sensoren, welche Zugriffe auf interne Eigenschaften oder Services erlauben. Heutzutage können eigentlich alle Frameworks auf alle Sensoren zugreifen. Aus diesem Grund gibt es nur beim Zugreifen Unterschiede. In der folgenden Tabelle mögliche Sensoren mit ihren Eigenschaften aufgelistet. Selbstverständlich enthält diese Liste nur die wichtigsten Sensoren, welche in den meisten Fällen verwendet werden. Unter [diesem Link](#) gibt es eine komplette Liste.

SENSOR	BESCHREIBUNG
BROWSER	Über den Browser können Apps einen Link im Systembrowser oder in der App öffnen
LAUNCHER	Damit kann eine URI geöffnet werden, welche auf eine andere App zeigt
MAPS	Die Karte ermöglicht eine Navigation zu einem Standort oder das Herausfinden der aktuellen Position
KONTAKTE	Darüber können auf die auf dem Gerät gespeicherten Kontakte zugegriffen werden
NETZWERK	Die Internetverbindung hat unzählige Eigenschaften, auf welche zugegriffen werden kann (Geschwindigkeit, Verbunden, SSID, ...)
BATERIE	Es kann der aktuelle Akkustand abgefragt werden, um Batterieschonender zu arbeiten
SMS / E-MAIL	Selbstverständlich sind die Standardnachrichtenservices aufrufbar
VIBRATION	Die Vibration oder haptisches Feedback kann gesteuert werden
MEDIA PICKER	Auf die aktuellen Daten oder die Kamera kann auf unterschiedliche Weise zugegriffen werden

### 2.2.1 Verwendung

In dieser Applikation werden zwei unterschiedliche Sensoren verwendet. Da das App intensiv mit Bildern und Dateien arbeitet werden die zwei dafür zuständigen Sensoren verwendet. Zum einen wird der Media Picker, welcher für das Holen von auf dem Gerät gespeicherten Grafiken zuständig ist, welche dann versendet werden können. Zusätzlich kann auch über die Kamera ein Foto gemacht und verschickt werden. Diese zwei Sensoren brauchen besondere Berechtigungen, damit sie funktionieren, welche unbedingt vor dem Verwenden geprüft werden müssen.

Zusätzlich zu den Bildern brauche ich die Netzwerkschnittstelle, um zu wissen, ob Nachrichten versendet werden können. Wenn dies nicht der Fall ist, dann werden die Nachrichten lokal gespeichert und beim nächsten Mal publiziert.

## 2.3 Datenbank

Eine Datenbank ist eine strukturierte Sammlung von Informationen, welche digital gespeichert werden. Mithilfe von SQL können CRUD-Operationen betätigt werden. Das bedeutet, dass Daten hinzugefügt, modifiziert und gelöscht werden können. Es gibt drei verschiedene Arten von Datenbanken<sup>15</sup>. In den nächsten Kapiteln werden diese Arten von Datenbanken vorgestellt und erklärt.

### 2.3.1 Relational

Die uralte Lösung, Daten in einer abfragbaren Art und Weise zu persistieren, hat den Test der Zeit bestanden. Sie begann mit dem Web 1.0 und wird auch im Web 2.0 noch verwendet. SQL-Datenbanken speichern Daten, bei denen die Platzierung der Zeilen Vorrang vor den Spalten hat, d. h., es wird erwartet, dass einzelne Zeilen von größerer Bedeutung sind, um Fremdschlüssel-Beschränkungen über Tabellen hinweg sicherzustellen. Dies gewährleistet eine starke Konsistenz der Schlüssel innerhalb der Tabelle und auch über die Tabellen hinweg. Darüber hinaus bietet SQL eine Vielzahl präziser Filter für die Bearbeitung von Attributen innerhalb der Tabelle und komplexe Joins, die über eine Tabelle hinausgehen. Es bietet auch erweiterte Funktionen für den Inhalt Ihrer Tabelle, d. h. Sie können andere Einschränkungen als Ihre Schlüssel definieren.

### 2.3.2 NoSQL

NoSQL entstand mit der Entstehung des Web 2.0. Das Web 2.0 wurde schnell reich an Daten im Ruhezustand und bei der Übertragung. SQL hatte mit den katastrophalen Folgen der großen Datenmengen zu kämpfen, die ein- und ausgelagert wurden. Das waren die wenigen Herausforderungen, mit denen SQL-Datenbanken konfrontiert waren. NoSQL ging das Problem an, indem es eine horizontale Schichtung ermöglichte. Um jedoch ein horizontales Sharding zu ermöglichen, wurde

<sup>15</sup> <https://dev.to/akhan/graph-vs-sql-vs-nosql-part-1-2cap>

bei Shards das Konzept der Hash-Maps zur Verteilung der Tabellenschlüssel verwendet. Bei den Zugriffsmustern hatten Spalten anstelle von Zeilen den ausschließlichen Vorrang. Dies hatte einen Grund: Um Werte in Massen zu laden, musste das zeilenweise Zugriffsmuster aufgehoben werden. Damit wurden zwei Probleme gelöst, zum einen das Massenladen von Daten durch Abrufen der gesamten Spalten und zum anderen die Tatsache, dass Spaltenwerte eine größere Ähnlichkeit mit Zeilenwerten aufweisen, so dass bessere Komprimierungsraten erzielt werden konnten. Die Einbettung von Daten und die Herstellung von Beziehungen zwischen Tabellen wurde jedoch zu einem Problem, da es keinen Fremdschlüssel zur Erweiterung der Beziehungen gibt. Dieses Problem wurde teilweise durch die massive Senkung der Speicherpreise gelöst. NoSQL-SQL bot die Speicherung von Daten mit Beziehungen innerhalb derselben Tabelle an, auch wenn dies die Replikation einiger Attribute bedeutete, um die Geschwindigkeit zu erhöhen. NoSQL bietet keine Beschränkungen für Attribute. Das bedeutet eine höhere Geschwindigkeit bei geringerem Zugriff auf die Daten als bei SQL.

### 2.3.3 Graph

Graph-Datenbanken enthalten eine Mischung aus beiden. Sie basieren auf der Graphentheorie und verwenden eine Datenstruktur mit verknüpften Listen, die die Eigenschaften von SQL- und NoSQL-Datenbanken vereint. Graph-Datenbanken bieten Vorrang vor den Zeilenoperationen, verteilen die Entitäten aber nach Hash-Knoten-IDs. Sie bieten eine NoSQL-ähnliche Spaltenverteilung, haben aber Relationen, die als Fremdschlüssel dienen, um Beziehungen zwischen verschiedenen Entitäten herzustellen, ohne replizierte Daten zu erzeugen. Komplexe Join-Operationen, die Fremdschlüssel erfordern, werden durch einen bereits etablierten Pfad zwischen den Knoten, den sogenannten Relationen, ersetzt. Labels sind lediglich eine spaltenbezogene Gruppierung von Knoten mit einer logischen Kategorie, die spaltenbezogene Abfragen beschleunigt. Es bietet auch Beschränkungen für Attribute und ist immer noch schema-los wie NoSQL, bietet aber den Zugriff auf Daten wie SQL.

### 2.3.4 Verwendung

In diesem Projekt wird für die Speicherung, von den Nachrichten auf dem Server, MariaDB<sup>16</sup> verwendet, was eine SQL-Datenbank ist. Es ist ein Fork von MySQL, um weiterhin Open-Source zu sein und schneller Funktionen zu implementieren. Es ist eine der beliebtesten relationalen Datenbanken und wird von diversen grossen Unternehmen verwendet. SQLite verwende ich lokal auf dem Gerät, damit die Daten auch offline sichtbar sind. Der Grund, wieso ich mich für MariaDB entschied, ist, da ich über Hostinger<sup>17</sup> eine Datenbank zur Verfügung habe, welche von überall

---

<sup>16</sup> <https://mariadb.org/>

<sup>17</sup> <https://www.hostinger.com/>

zugreifbar ist. Für SQLite entschied ich mich, da meine Daten strukturiert sind und ich darüber einfach speichern kann. Das Schema wurde mit PlantUML<sup>18</sup> modelliert und sieht wie folgt aus:

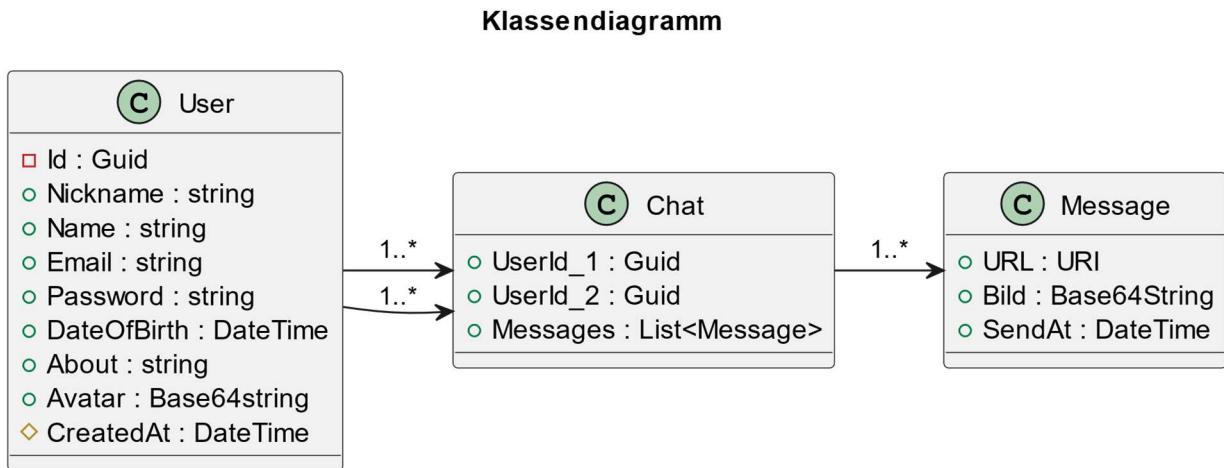


Abbildung 4 Datenbankschema

## 2.4 Persistenz

Als Persistenz<sup>19</sup> bezeichnet man die Speicherung der vorherigen Version. Wenn ein Benutzer eine Eingabe macht und danach das App schliesst, dann möchte er beim nächsten Mal die vorherige Eingabe wieder finden. Es ist visuell nicht sichtbar, dass die Daten im Hintergrund gespeichert werden. Eine Persistenz von Eingaben kann durch das ständige Speichern der Zeichenfolgen gemacht werden. Während dem Schreiben wird nicht nur das Feld, sondern auch eine Datei beschrieben. Das stellt sicher, dass die Persistenz mit Sicherheit vorhanden ist. Der Nachteil davon ist, dass das Speichern sich bei der Performance bemerkbar macht. Bei der Persistenz spielt die Internetverbindung eine wichtige Rolle.

### 2.4.1 Datei Persistenz

Wenn die Internetverbindung unterbrochen wird, dann können die Daten automatisch auf dem Gerät offline gespeichert werden. Sobald die Verbindung wieder besteht, werden dann die Daten an den Server gesendet. Wenn das aktiviert ist, dann sind die Daten auch bei einem Neustart vorhanden. Callbacks werden weiterhin auf lokale Updates angewendet. In einer Warteschlange können alle Schreibvorgänge gespeichert werden, welche auf die Datenbank sollten.

---

<sup>18</sup> <https://plantuml.com/class-diagram>

<sup>19</sup> <https://firebase.google.com/docs/database/android/offline-capabilities>

### 2.4.2 Synchronisation

Die Daten vom Server können automatisch lokal gespeichert werden, damit auch ohne Internet darauf zugegriffen werden kann. Abfragen werden dann ohne Internet auf der lokalen Datenbank gemacht, so dass der Benutzer möglichst keinen Unterschied bemerkt. Um zu grossen Datenmengen zu vermeiden wird ein Limit verwendet, welches zwingt, dass nur die neusten und relevanten Daten synchronisiert werden.

## 2.5 Entwicklungsumgebungen

Eine Entwicklungsumgebung ist ein Programm, das es einem Entwickler ermöglicht, Software zu entwickeln. Die Entwicklungsumgebung stellt dem Entwickler Werkzeuge zur Verfügung, die ihm die Arbeit erleichtern. Unter anderem erlaubt es das Kompilieren von Quellcode, das Debuggen von Programmen und das Verwalten von Versionskontrollsysteinen. Die Entwicklungsumgebung ist ein zentraler Bestandteil der Softwareentwicklung. Häufig wird sie auch als IDE (Integrated Development Environment) bezeichnet.

### 2.5.1 Node

Node.js ist eine serverseitige Plattform in der Softwareentwicklung zum Betrieb von Netzwerkanwendungen. Insbesondere lassen sich Webserver damit realisieren. Node.js wird in der JavaScript-Laufzeitumgebung „V8“ ausgeführt, die ursprünglich für Google Chrome entwickelt wurde, und bietet eine ressourcensparende Architektur, die eine besonders große Anzahl gleichzeitig bestehender Netzwerkverbindungen ermöglicht.

### 2.5.2 Gradle

Gradle ist ein auf Java basierendes Build-Management-Automatisierungs-Tool, vergleichbar mit Apache Ant und Apache Maven. Gradle nutzt eine auf Groovy basierende domänenpezifische Sprache (DSL) zur Beschreibung der zu bauenden Projekte. Im Gegensatz zu Maven-Projektdefinitionen (pom.xml) sind Gradle-Skripte direkt ausführbarer Code.

### 2.5.3 Frameworks

Ein Framework ist eine Struktur, auf der Sie Software aufbauen können. Es dient als Grundlage, damit Sie nicht ganz bei null anfangen müssen. Frameworks sind in der Regel mit einer bestimmten Programmiersprache verknüpft und eignen sich für verschiedene Arten von Aufgaben.<sup>20</sup> Ein Framework verhindert häufige Fehler und sollte Zeit bei der Entwicklung sparen. Wiederholender

---

<sup>20</sup> <https://www.codecademy.com/resources/blog/what-is-a-framework/>

Code sollte damit verhindert und vereinfacht werden. Je nachdem was für ein Framework verwendet wird steigt die Komplexität und Abstraktion. Aus diesem Grund lohnt es sich zu Beginn ein kleineres zu verwenden, welches nicht zu viel von allein macht.

	<b>AngularJS</b>	<b>Angular</b>	<b>Ember.js</b>	<b>Vue.js</b>	<b>Meteor</b>
<b>Erscheinungsjahr</b>	2009	2016	2011	2014	2012
<b>Maintainer</b>	Google	Google	Ember Core Team	Evan You	Meteor Development Group
<b>Lizenz</b>	MIT	MIT	MIT	MIT	MIT
<b>GitHub-Contributors</b>	ca. 1.600	ca. 570	ca. 700	ca. 700	ca. 370
<b>Architektur</b>	MVVM/MVVW	MVC	MVVM	MVVM	MVVM
<b>Besonderheiten</b>	Auch für Mobile- und Desktop-Apps einsetzbar	Auch für Mobile- und Desktop-Apps einsetzbar	Auch für Desktopanwendungen einsetzbar	Einfache Einarbeitung	Kombiniert Backend und Frontend

Abbildung 5 JavaScript-Frameworks<sup>21</sup>

## 2.6 IDEs

Mithilfe von IDEs können Entwickler schneller mit der Programmierung neuer Anwendungen beginnen, weil nicht mehrere Dienstprogramme als Teil des Setup-Prozesses manuell konfiguriert und integriert werden müssen. Die Entwickler müssen auch nicht mehr Stunden damit verbringen, die Bedienung unterschiedlicher Tools zu erlernen, da jedes dieser Dienstprogramme in derselben Workbench enthalten ist. Dies kann sich speziell beim Onboarding neuer Entwickler als nützlich erwiesen, da sich diese mithilfe der IDE bei den Standard-Tools und -Workloads eines Teams in kürzester Zeit auf den neuesten Stand bringen können. Tatsächlich ist es so, dass die meisten Features einer IDE konzipiert wurden, um Zeit zu sparen, sodass z. B. intelligente Code-Vervollständigung und automatische Code-Generierung dafür sorgen, dass komplette Zeichensequenzen nicht mehr komplett ausgeschrieben werden müssen.<sup>22</sup>

### 2.6.1 Android Studio

Android Studio ist eine freie Integrierte Entwicklungsumgebung (IDE) von Google und offizielle Entwicklungsumgebung für Android. Mit der Preview-Version 1.3 vom 28. Mai 2015, wurde der SDK-Manager vollständig in Android Studio integriert, des Weiteren ist nun die Unterstützung für das Android-NDK (Native Development Kit) vorhanden.

---

<sup>21</sup> <https://ict-berufsbildung.info/mod/h5pactivity/view.php?id=21634>

<sup>22</sup> <https://www.redhat.com/de/topics/middleware/what-is-ide>

## 2.6.2 Visual Studio Code

Visual Studio Code, kurz auch VS-Code, ist ein Quelltext-Editor von Microsoft. Er dient vorrangig der Entwicklung von Webanwendungen und ermöglicht die Programmierung mit Programmier- und Auszeichnungssprachen wie Batch, C#, C++, Clojure, CoffeeScript, CSS, Dart, Dockerfile, F#, Go, Groovy, Handlebars.js, HTML, Ini, Jade, Java, JavaScript, JSON, Less, Lua, Makefile, Markdown, Objective-C, Perl, PowerShell, PHP, Python, R, Razor, Ruby, Rust, Sass, SQL, Swift, TypeScript, Visual Basic und XML.

Es handelt sich um einen kostenlosen und quelloffenen Texteditor, der plattformübergreifend für die Betriebssysteme Windows, macOS und Linux verfügbar ist. Bis auf den Namen, das Logo und einigen Funktionen wie IntelliSense hat VS-Code nichts mit Visual Studio gemeinsam. VS-Code basiert auf dem Framework Electron und ermöglicht auch Debugging, IntelliSense und Versionsverwaltung. Im Unterschied zu Visual Studio arbeitet VS-Code nicht mit Projektdateien, sondern auf Basis von Codedateien und Ordnern.<sup>23</sup>

## 2.6.3 Visual Studio

Visual Studio ist eine professionelle IDE<sup>24</sup>, welche alle vorstellbaren Anforderungen erfüllt. Es ist hauptsächlich für grössere Unternehmen, welche mit .NET entwickeln gedacht. Es enthält ein ausführliches Diagnosetool, welche automatisch den Code analysiert und visualisiert. Über einen eingebauten Git-Client können einzelne Zeilen und Merge Konflikte ohne Probleme behoben werden. Mit eingebautem IntelliSense wird das Refactoring ein Kinderspiel. Für die Zusammenarbeit kann der Code geteilt werden, sodass Änderungen gleich sichtbar sind. Eines der grössten Features sind die Live-Unit-Tests, welche im Hintergrund ausgeführt werden. Beim Ändern am Code werden die abhängigen Tests ausgeführt, um zu testen, ob die Änderung mit bestehenden Funktionen kompatibel ist.



Abbildung 6 Live-Unit-Test

<sup>23</sup> <https://ict-berufsbildung.info/mod/assign/view.php?id=19323>

<sup>24</sup> <https://visualstudio.microsoft.com/vs/>

Der Nachteil dieser umfangreichen IDE ist, dass sie kostenpflichtig ist und nur auf Windows und MacOS läuft. Wer also Linux verwendet muss auf Visual Studio Code zurückgreifen. Durch die ganzen Tools und Automationen muss ein moderner Computer mit mindestens 16 GB Arbeitsspeicher verwendet werden, damit es flüssig läuft.

#### 2.6.4 Verwendung

Meine Applikation werde ich mit C# und MAUI machen, was stark für die Verwendung von Visual Studio spricht. Ein weiterer Grund, wieso ich es benutze, ist, dass ich in meinem Lehrbetrieb C# mit Blazor verwende und somit die Lizenzen dafür bereits besitze. Dies ist ein riesiger Vorteil, da ich dadurch bei der Entwicklungsumgebung keine Einschränkungen habe. Ich persönlich mag die Umgebung sehr und habe die Lernkurve bereits gemeistert. Die benötigten Einstellungen und Tools habe ich deshalb bereits installiert und konfiguriert.

### 2.7 Usability

Bei der Usability kommt es für mich vor allem darauf an, dass der Benutzer Tätigkeiten nicht doppelt machen muss. Dies sollte ihn davon abhalten, dass er genervt oder ungeduldig wird. Zudem finde ich es nötig, dass die Eingaben, während dem Tippen bereits validiert werden und bei Fehlern frühzeitig reklamieren. So besteht ein ständiger Austausch von Feedbacks zwischen der Applikation und dem Benutzer, was Sicherheit und Selbstvertrauen bietet. Ein weiterer wichtiger Punkt ist, dass das Layout und die Farbgebung konsistent sind. Die Seiten sollten sich vom Aufbau nicht ändern, damit bekannte Einstellungen und Funktionen sich am gleichen Ort befinden. Dabei sollte das Rad nicht neu erfunden werden. Es gibt bereits unzählige Applikationen, welche ein Layout und Navigationskonzept implementierten. Von diesem abzuschauen hilft auch dem Endnutzer, da dieser sich an diese Konzepte gewöhnt hat. Tätigkeiten sollten ohne Verzögerung und in einer kurzen Zeit gelöst werden. Ständige Ladebalken oder unzählige Bestätigungen stoppen den Flow, was zu Unzufriedenheit führt. Bei diesem Projekt wurde auf die oben genannten Grundsätze zur Usability geachtet.

Benutzerfreundlichkeit ist ein einfaches Konzept, das jedoch so mächtig ist, dass es selbst die erstaunlichsten Produktkonzepte und elegantesten Designs zunichtemachen kann, wenn es nicht gut umgesetzt wird. Vereinfacht ausgedrückt, ist Benutzerfreundlichkeit die Leichtigkeit, mit der eine Person eine bestimmte Aufgabe mit Ihrem Produkt erledigen kann. Die Web Content Accessibility Guidelines (WCAG) erklären, wie Webinhalte zugänglich gemacht werden für Menschen mit Einschränkungen. Es gibt dafür drei verschiedene Stufen, welche die Niveaus (niedrig, mittel & hoch) beschreiben.

RICHTLINIE	ERKLÄRUNGEN
WAHRNEHMBARKEIT	Inhalte und Bedienelemente (Schaltflächen, Eingabefelder usw.) müssen den Nutzern derart präsentiert werden, dass diese auch mit Einschränkungen wahrgenommen werden können.
BEDIENBARKEIT	Die Bestandteile des User-Interfaces und der Navigationselemente müssen in jedem Falle bedienbar sein.
VERSTÄNDLICHKEIT	Informationen und die Nutzung der Bedienelemente müssen verständlich sein.
ROBUSTHEIT	Die Inhalte einer WCAG-konformen Webseite müssen robust genug sein, sodass sie von Endgeräten und Software aller Art (Browser, Lesegeräte usw.) samt assistierenden Techniken dargestellt und interpretiert werden können.

## 2.8 Testverfahren

Eines der wichtigsten Aufgaben und Abgabekriterien einer Software sind die Tests. Um sicherzustellen, dass eine Methode oder sogar eine ganze Applikation korrekt funktioniert, muss sie getestet werden. Da Applikationen immer grösser und komplexer werden, wird das manuelle Testen zeitaufwendiger. Um dieser Entwicklung entgegenzuwirken, werden automatische Tests geschrieben, welche das Testen übernehmen. Das Ziel von Testen ist es, die Funktionalität einer Applikation sicherzustellen. Wenn eine Änderung vorgenommen wird, dann sollte automatisch validiert werden, ob immer noch alles so funktioniert wie es sollte. Die Tests sollten kreativ und intellektuell herausfordernde Aufgaben sein. Die folgenden Abschnitte wurden aus meiner eigenen Dokumentation genommen.<sup>25</sup> Folgende Testverfahren gibt es:

Test	Beispiel
Unit-Testing	Methode liefert korrekte Rückgabewerte
Integration-Testing	Produkt kann auf einer Webseite gekauft werden
System-Integration-Testing	Installation & Inbetriebnahme erfolgreich
Load / Stress-Testing	Berechnung X kann Y Mal pro Sekunde ausgeführt werden
Abnahme / Nutzungs-Testing	Zusammenarbeit mit anderen Softwares / Microservice

*Abbildung 7 Testarten*

<sup>25</sup> [https://bztfinformatik.github.io/lbl\\_doku-21r8390/Erkl%C3%A4rungen/Testing/#integration-testing](https://bztfinformatik.github.io/lbl_doku-21r8390/Erkl%C3%A4rungen/Testing/#integration-testing)

### 2.8.1 Arten

In der Theorie kann jede Interaktion, die mit der Applikation gemacht wird, getestet werden. Je nach Umfang werden unterschiedlich viele Komponenten automatisiert getestet, da die Instandhaltung von den Tests nicht einfach ist. Heutzutage ist es Standard, dass einfache Methoden automatisiert getestet werden. Eine Übersicht, wie ein Entwicklungsprozess mit Tests ablaufen könnte, ist im folgenden Bild zu erkennen:

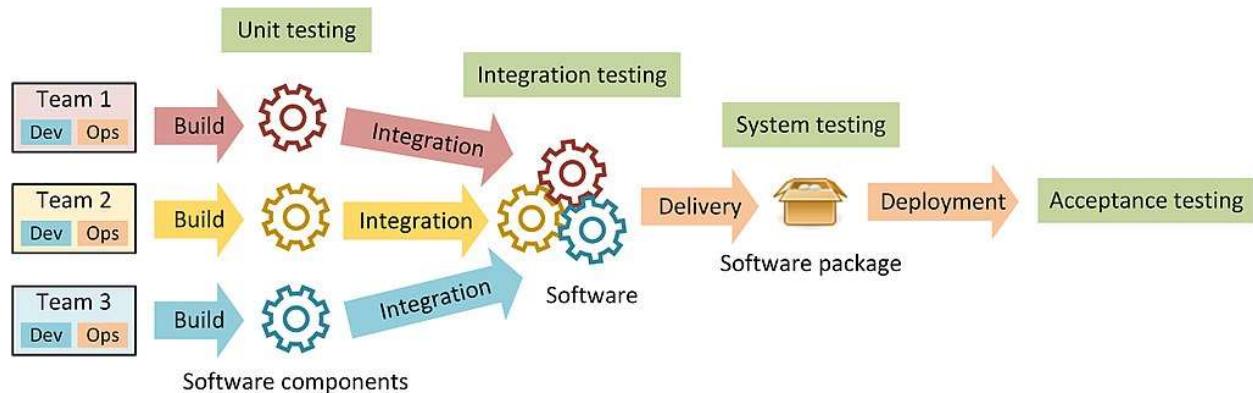


Abbildung 8 Wie wird getestet?

### 2.8.2 Integration Testing

Integrationstests werten eine Applikation in einem grösseren Bereich als Unit Tests aus. Das bedeutet, es werden mehrere Funktionalitäten gleichzeitig getestet, um sicherzustellen, dass die Zusammenarbeit davon reibungslos verläuft. Es wird zum Beispiel getestet, ob eine Datenbank oder ein Dateisystem richtig eingebunden wurde. Diese Tests haben meistens eine längere Ausführungszeit, da sie einen grossen Teil des Codes testen.

### 2.8.3 Load & Stress Testing

Load und Stress Testen sind wichtig, um sicherzustellen, dass eine Applikation performant und skalierbar ist. Das Ziel von beiden Tests ist unterschiedlich, auch wenn sie ähnliche Tests beinhalten. Load testet, wie stabil eine Applikation ist, wenn sie auf einem hohen Anwendungsvolumen läuft. So sollte sichergestellt werden, dass auch ganz viele Benutzer gleichzeitig auf der Applikation arbeiten können oder grosse Datenmengen unterstützt werden. Stress testet wie stabil eine Applikation auf extremen Anwendungsvolumen läuft, oft für eine lange Zeitperiode. Die Tests belasten die Anwendung mit einer Belastungsspitze, einer allmählichen ansteigender Last oder mit begrenzten Rechenressourcen. Es sollte sichergestellt werden, dass sich von Fehlern erholt und zur normalen Funktion zurückgekehrt werden kann.

#### 2.8.4 UI-Testing

Um sicherzustellen, dass die Benutzeroberfläche einer Applikation korrekt funktioniert, werden die Benutzeroberflächen getestet. Dies kann bei einer HTML-Seite zum Beispiel mit Selenium erfolgen. Es wird dabei ein Ablauf oder eine Tätigkeit schrittweise durchlaufen. Nach jeder Tätigkeit wird geprüft, ob die richtigen Werte gesetzt wurden und ob das Ziel erreichbar ist. So kann zum Beispiel sichergestellt werden, dass jeder Knopf immer eine Funktionalität hat.

Das UI ist jedoch etwas, was sehr oft manuell getestet wird, da das Schreiben und Instandhalten dieser Abläufe aufwendig ist. Zudem kann nicht erkannt werden, ob die Elemente korrekt aussehen und ob die Benutzerführung angenehm ist.

#### 2.8.5 Unit Test

Unit Tests sind kleine Tests, die einzelne Methoden testen. Es werden möglichst kleine Aufgaben getestet und deren Ergebnisse validiert. Das Ziel davon ist es, dass die Applikation ständig automatisiert getestet wird. Eine Methode wird dabei mit verschiedenen Grenzwerten ausgeführt. Die Rückgabe wird dann validiert. Es sollte ungefähr 80% aller Methoden mithilfe von Unit Tests getestet werden. So sollten Probleme frühzeitig erkannt werden. Diese Tests gehen nur wenige Millisekunden, weswegen sie im Hintergrund ausgeführt werden können. Eine verständliche Beschreibung zu guten Tests kann auf MS Docs gefunden werden.

#### 2.8.6 Mocking

Im vorherigen Beispiel wurde nur eine einfache Methode, welche keine Abhängigkeit hat, getestet. Dies ist nicht immer der Fall. Meistens ist noch eine Datenbank oder ein API-Abruf eingebunden. Die Rückgabewerte können sich ändern, was das erwartete Ergebnis verhindert. Aus diesem Grund sollten Tests so gut wie möglich isoliert sein. Dies wird durch Mocking erreicht. So kann sichergestellt werden, dass da Problem nur in der aktuellen Methode liegt. Dabei die Abhängigkeit überschrieben, sodass statische Werte zurückgegeben werden. Wenn Mocking verwendet wird, dann ist meistens die Frage, ob die Tests nicht bereits Integration Tests sind.

#### 2.8.7 Blackbox

In einem Blackbox-Test ist die interne Struktur des Programms nicht bekannt. Dabei wird das Ein- und Ausgabeverhalten getestet. Ein vollständiges Austesten des Codes ist nicht möglich, da die Limiten nicht bekannt sind. Es können nur so viel Fehler gefunden werden, wie visuell sichtbar sind. Es wird nicht überprüft, ob das Speichern in die Datenbank oder die Verbindungen richtig

aufgebaut sind. Blackbox-Tests sollten von manuellen Testern zur Validierung der Usability ausgeführt werden.<sup>26</sup>

### 2.8.8 Whitebox

Das Whitebox-Testing ist das Gegenteil des Blackbox-Tests. Der Unterschied ist, dass der Aufbau und die Limiten des Apps bekannt sind. Es wird dabei der vollständige Code mit der Datenüberdeckung geprüft. Ein Programm kann auch wegen fehlendem Code als fehlerhaft angesehen werden. Die Codeüberdeckung sagt nichts über datensible Fehler aus, welche beim Programmieren entstehen. Es wird getestet wieviel Zeit das Ausführen einer Methode benötigt oder wie viele Instanzen eines Objektes angelegt werden. Diese Tests werden am besten automatisiert ausgeführt, da ein Mensch schlecht ständig den Arbeitsspeicher überprüfen kann.

### 2.8.9 A / B

Das Ziel von A / B Tests ist, dass herausgefunden wird auf was die Benutzer am besten reagieren. Egal wie ausgeklügelt eine Marketingstrategie ist, die Zielgruppe könnte immer eine andere Meinung besitzen. Mithilfe von zwei Versionen wird verglichen was mehr Kunden zur gewünschten Aktion anregt. So vergrößert man die Chance, dass der Kunde mit zufrieden ist. Es ist ein sehr weit verbreitetes Verfahren, was heutzutage alle Webseitenentwickler verwenden.<sup>27</sup>

---

<sup>26</sup> [https://ict-berufsbildung.info/pluginfile.php/32613/mod\\_resource/content/0/SoftwareTesten.pdf](https://ict-berufsbildung.info/pluginfile.php/32613/mod_resource/content/0/SoftwareTesten.pdf)

<sup>27</sup> <https://mailchimp.com/marketing-glossary/ab-tests/>

## 2.8.10 Testfälle

Ein Testfall sollte eine Beschreibung haben, welche aussagt was und wie getestet wird. Der Testfall muss sehr genau definiert sein, damit es zu keinen Missverständnissen oder Doppeldeutigkeiten kommt. Tests müssen manchmal in unterschiedlichen Reihenfolgen ausgeführt werden. Fall dies der Fall ist muss dies auch vermerkt sein. Das erwartete Ergebnis ist mit Abstand das Wichtigste. Es definiert, ob ein Testfall erfolgreich oder fehlgeschlagen ist. Das Resultat wird üblicherweise dokumentiert. Je nachdem, was das Ergebnis ist können andere Aktionen folgen. Was dann unternommen wird muss dokumentiert werden.

### Testfälle erstellen und protokollieren

In der Dokumentation sollen Ihre Testszenarien und Ergebnisse festgehalten werden:

- Was wird getestet? (Navigation – Menü usw.)
- Wie wird getestet? (im Browser Device simuliert usw.)
- Wann wird getestet? (bei Projektende)
- Was wird erwartet? (auf Handy, Tablet und PC ist das Menü optimal)
- Wie ist das Resultat des Tests? (Menü wie erwartet, mit Screenshot dokumentiert)
- Welche Folgen hat das? (falls der Test fehlschlägt)
- Was wird dann unternommen? (wo wird der Fehler vermutet, was könnte unternommen werden)

*Abbildung 9 Testfälle definieren<sup>28</sup>*

## 2.9 Veröffentlichen

Sobald eine App fertiggestellt und getestet wurde kann sie veröffentlicht werden. Dazu gibt es zwei grosse Plattformen, welche die Veröffentlichung erlauben. Für Android wäre dies der Google Play Store, welcher von Beginn an auf den Geräten installiert ist. Dafür wird ein Google Konto und eine hinterlegte Kreditkarte benötigt. Android Apps werden als APK oder Bundle hochgeladen. Unter IOS gibt es mehrere Hindernisse, die das Publizieren verzögern können. Zunächst wird ein Mac Gerät benötigt, um eine ausführbare Datei zu erstellen. Diese kann dann über ein Tool in den Store hochgeladen werden. Dort wird ein verifiziertes Konto benötigt. Nach dem Hochladen dauert es einen bis drei Tage, bis ein Apple Mitarbeiter das App testet und sicherstellt, dass alle Funktionalitäten erfüllt sind. Wenn irgendetwas im Prozess fehlläuft, wird die Datei als ungültig vermerkt und kann nicht mehr benutzt werden.

---

<sup>28</sup> <https://ict-berufsbildung.info/mod/hvp/view.php?id=19464>

Wenn das App alle Anforderungen erfüllt, kann es dann veröffentlicht werden. Die Anforderungen sind, 5 Bilder je im Hoch und Querformat, welche den Inhalt und die Funktionalitäten aufzeigen. Dies sollte dem Benutzer helfen zu erkennen, ob es qualitativ und vertrauenswürdig ist. Zusätzlich müssen die benötigten Berechtigungen und Versionen angegeben werden. Bei Android sollte man deshalb auf mehreren Plattformen testen. Das Icon und der Name dürfen natürlich auch nicht fehlen. Eines der wichtigsten Schritte ist jedoch, dass es signiert ist, damit die Herkunft sichergestellt werden kann. Fall eine App nicht für Kinder gedacht ist muss dies dringend angegeben werden. Die Kosten für die Veröffentlichung sind abhängig von der Plattform und gehen von einmaligen 25 bis zu jährlichen 99 Fr.

### 2.9.1 Beachten

Beim Veröffentlichen müssen folgende Punkte beachtet werden:

- Name
- Zielgruppe
- Beschreibung
- Bewertung
- Wartbarkeit
- Werbung/Vermarktung
- Marktanalyse
- Kosten
- Kompetenzfähigkeit der App
- Richtlinien im Store
- Unterstützte Plattform

## 2.9.2 Release

Wenn eine Applikation als Release erstellt wird, dann wird der Code optimiert und ist später nicht mehr nachfolgbar. Debugg Informationen und ausführliche Fehlermeldungen gibt dann leider auch nicht mehr. MAUI bietet eine sehr einfache Veröffentlichung, welche auch über eine CI/CD-Server gemacht werden könnte. Die dazugehörige [Anleitung](#) ist auf der Microsoftseite zu finden. Kurzgesagt kann es über `dotnet publish` gemacht werden. Um eine Cordova Apps zu veröffentlichen, muss zuerst die Konfiguration richtig eingestellt werden. Danach kann die Applikation kompiliert und signiert werden. In den zwei Bildern wird dieses Verfahren gezeigt:



```
{
  "android": {
    "apk": {
      "keystore": "memechat.keystore",
      "storePassword": "memechat123!",
      "alias": "memechat",
      "password": "memechat123!",
      "keystoreType": "",
      "packageType": "apk"
    },
    "release": {
      "keystore": "memechat.keystore",
      "storePassword": "memechat123!",
      "alias": "memechat",
      "password": "memechat123!",
      "keystoreType": "jks",
      "packageType": "bundle"
    }
  }
}
```



```
Windows PowerShell
C:\> C:\Program Files\Java\jdk1.8.0_221\bin\keytool.exe -genkey -v -keystore memechat.keystore -alias memechar -keyalg RSA -keysize 2048 -validity 10000
0ms
```

Abbildung 10 Cordova Release

## 2.10 Planung

Bevor ich mir eine Idee suchte, las ich zuerst die Anforderungen an die App durch. Dadurch wusste ich, dass ich einen Sensor verwenden und Daten speichern muss. So konnte ich meine Einstellungen eingrenzen und fokussierte mich auf die möglichen Projekte. Durch Brainstorming bekam ich dann die Idee, dass ich eine Nachrichtenapp programmiere. Da es bereits mehrere von diesen gibt brauchte ich aber etwas Einzigartiges. Sobald ich eine Idee hatte, schaute ich mich im Internet nach ähnlichen Projekten um, damit ich den Umfang abschätzen kann. Zudem musste ich mir sicher sein, dass auch die nötigen Tools zur Verfügung stehen. Als nächstes erstellte ich eine kurze Beschreibung meiner Umsetzung.

### 2.10.1 Name

Was die Benutzer schlussendlich am meisten an der App erinnert ist der Name. Aus diesem Grund sollte dieser sorgfältig und mit Bedacht ausgewählt werden. Zusätzlich sollte der Name eine Verbindung mit dem App herstellen. Auf den Namen «MemeChat» bin ich gekommen, da ich nur Memes versendet werden können. So habe ich die zwei wesentlichen Punkte der App in ein Wort zusammengefasst. Wichtig war mir, dass der Name nicht länger als 8 Zeichen ist, damit er auf allen Geräten vollständig angezeigt wird. Im Internet gibt es Tools<sup>29</sup>, welche bei der Namensfindung behilflich sein könnten.

### 2.10.2 Beschreibung

MemeChat ist eine App, welche es ermöglicht, mit anderen Usern über ein Chatfenster zu kommunizieren. Ganz im Gegenteil zu allen anderen Nachrichtenapps kann in dieser nur über Bilder und Memes kommuniziert werden. Dies sollten die Konversationen unterhaltsamer und lustiger gestalten. Die Memes werden teilweise von der App bereitgestellt, können aber auch von den Usern selbst erstellt werden. Die App ist für alle Altersgruppen geeignet. Für maximale Kompatibilität kann die App unter Android und iOS genutzt werden.

### 2.10.3 Icons

Das Icon ist essenziell, um den Nutzer im Store auf die Applikation aufmerksam zu machen. Es sollte ein grosses und schlichtes Logo sein, welches einen Wiedererkennungswert besitzt. Am besten wird das Icon über [folgende Webseite](#) in die unterschiedlichen Formate gebracht. Dort werden automatisch die plattformspezifischen Icons erstellt, was unheimlich viel Zeit erspart. Das Icon, welche ich verwenden werde, ist schlicht und sie hat wie folgt aus:



Abbildung 11 App Icon<sup>30</sup>

---

<sup>29</sup> <https://namify.tech/app-name-generator>

<sup>30</sup> <https://looka.com/>

## 2.10.4 Hintergrund

Mobileapplikationen können geschlossen oder pausiert werden, wenn sie nicht mehr gebraucht werden. Je nachdem was für eine Art von App sollte dieser Status unterschiedlich behandelt werden. So können Push Mitteilungen angezeigt werden, um den Benutzer auf eine Änderung aufmerksam zu machen.

Wenn die Applikation geschlossen oder in den Hintergrund geschoben wird, dann werden zur Linderung der Komplexität keine Nachrichten empfangen. Dies hat den Nachteil, dass die App geöffnet werden muss, um zu sehen, ob eine ungelesene Nachricht vorhanden ist. Die Umsetzung dieser Funktionalität würde mit etwas mehr Zeit keine Probleme bereiten. Beim öffnen der App werden dann alle Daten synchronisiert.

## 2.10.5 Mockups

Ein Wireframe ist ein schematischer Entwurf einer Benutzeroberfläche. Darin wird definiert welche Elemente verwendet und positioniert werden. Zudem wie die Navigation zwischen den Seiten funktionieren soll. Es dient der Entwicklung frühzeitig Fehler und Hindernisse zu erkennen, damit diese nicht beim Implementieren passieren. Unschönheiten oder Denkfehler werden beim Erstellen von Wireframes erkannt und behoben.

Sie dienen auch dazu, dass man dem Kunden einen ersten Entwurf zeigen kann, an welchem noch Änderungen gemacht werden können. So können auch Missverständnisse behoben oder bessere Entscheidungen getroffen werden.

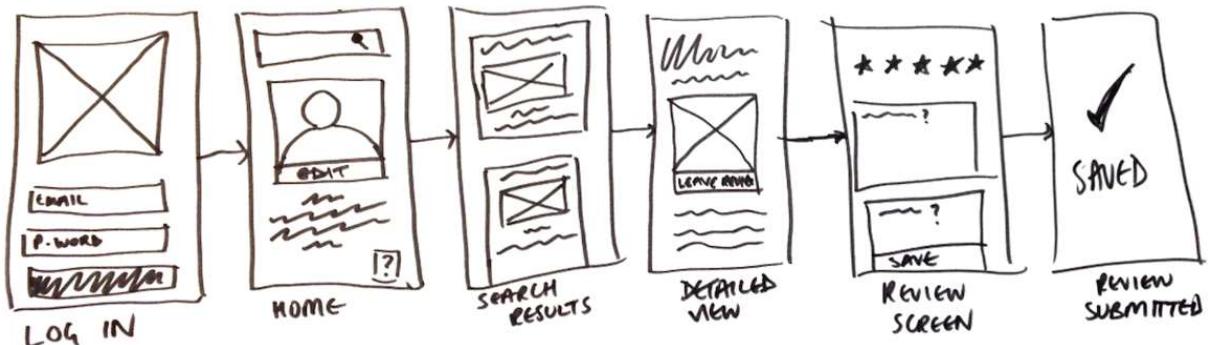


Abbildung 12 Wireframes<sup>31</sup>

## 2.10.6 Meine Wireframes

Die Wireframes habe ich mithilfe von Figma<sup>32</sup> erstellt, was ein Designtool ist. Es kann mit Leichtigkeit ein Entwurf entwickelt werden. Der Vorteil davon ist, dass es sehr einfach zu erlernen und

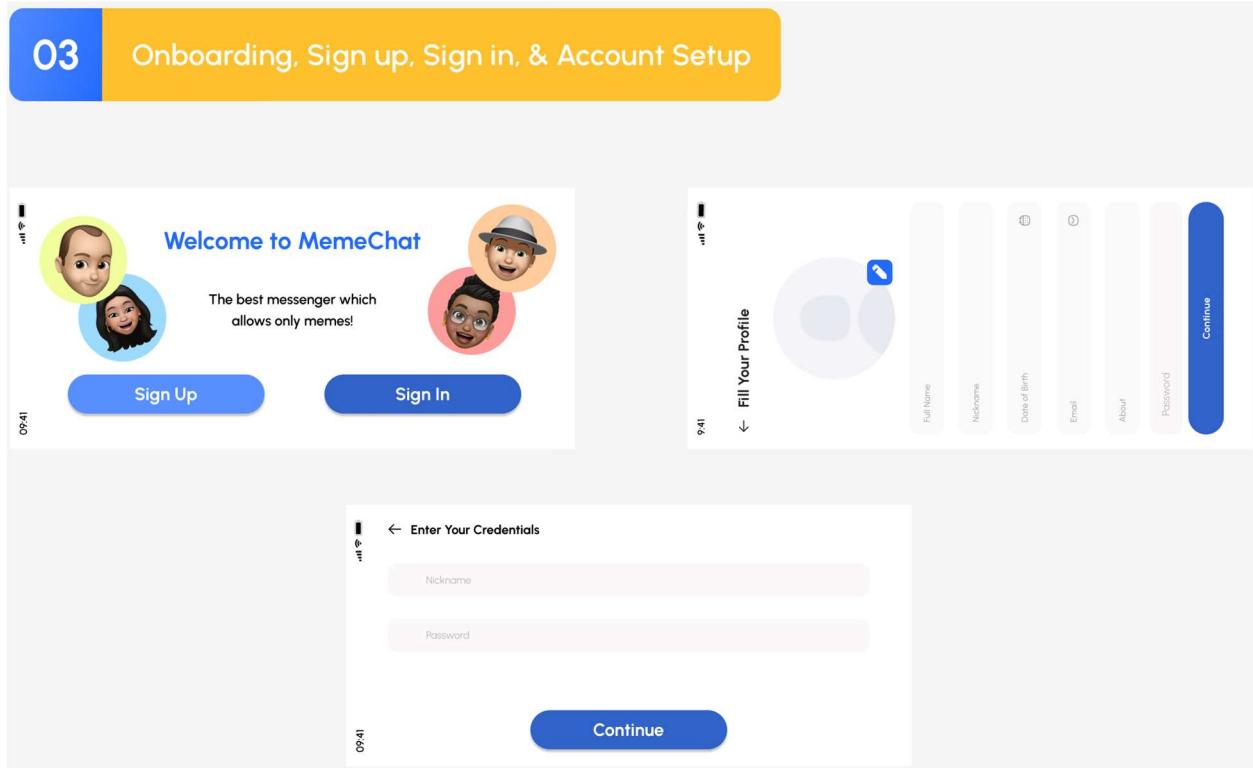
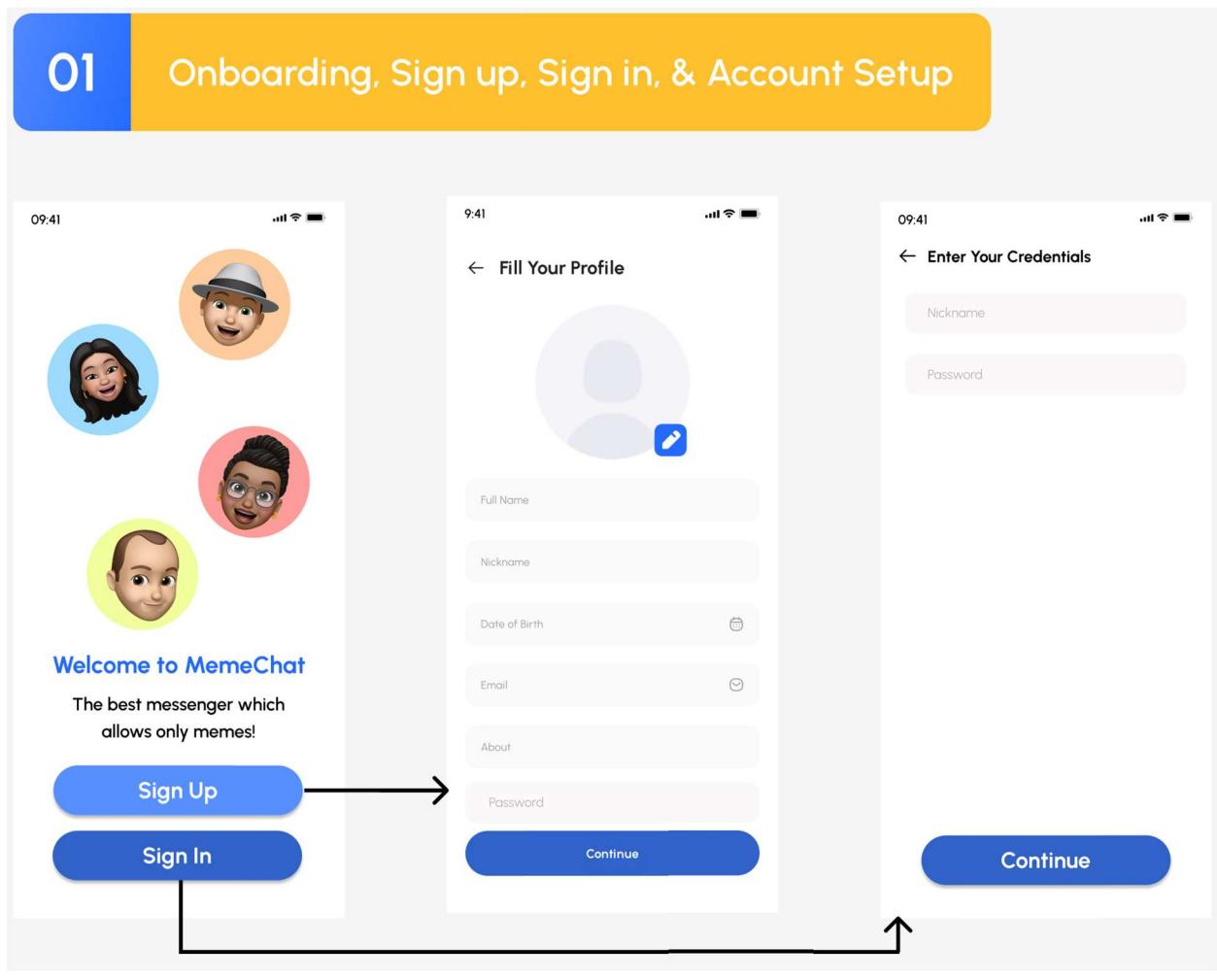
<sup>31</sup> <https://careerfoundry.com/en/blog/ux-design/what-is-a-wireframe-guide/>

<sup>32</sup> <https://www.figma.com/>

kostenlos verwendbar ist. Unter [diesem Link](#) können sie aufgerufen und genauer betrachtet werden. Alle Seiten bis auf die Anmeldung sind jeweils im Hoch- und Querformat verfügbar. Bei der Anmeldung macht es mit den vielen Feldern keinen Sinn das Querformat zu unterstützen. Die Tablett Ansicht verhält und sieht exakt gleich wie die Mobileapplikation. Der einzige Unterschied ist, dass es grösser skaliert ist. Hier im Dokument sind die Ausschnitte der einzelnen Seiten.

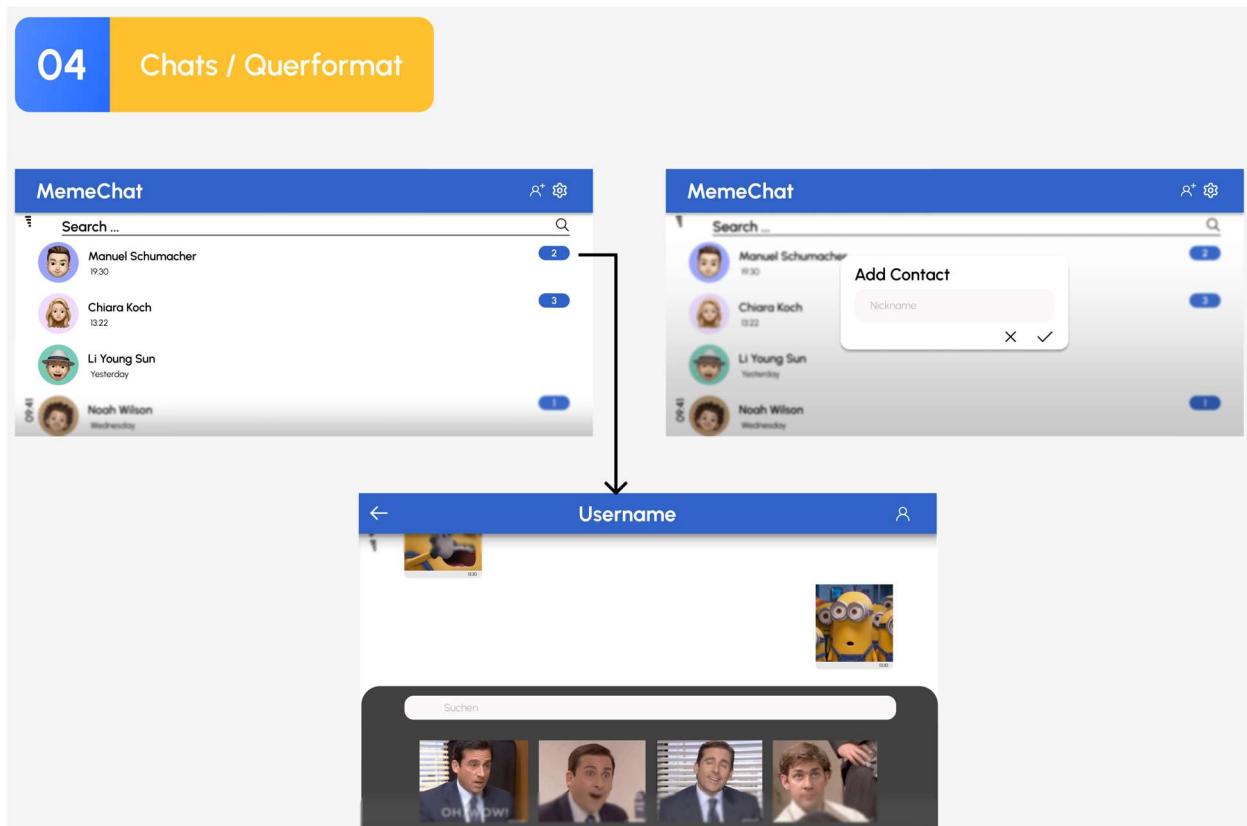
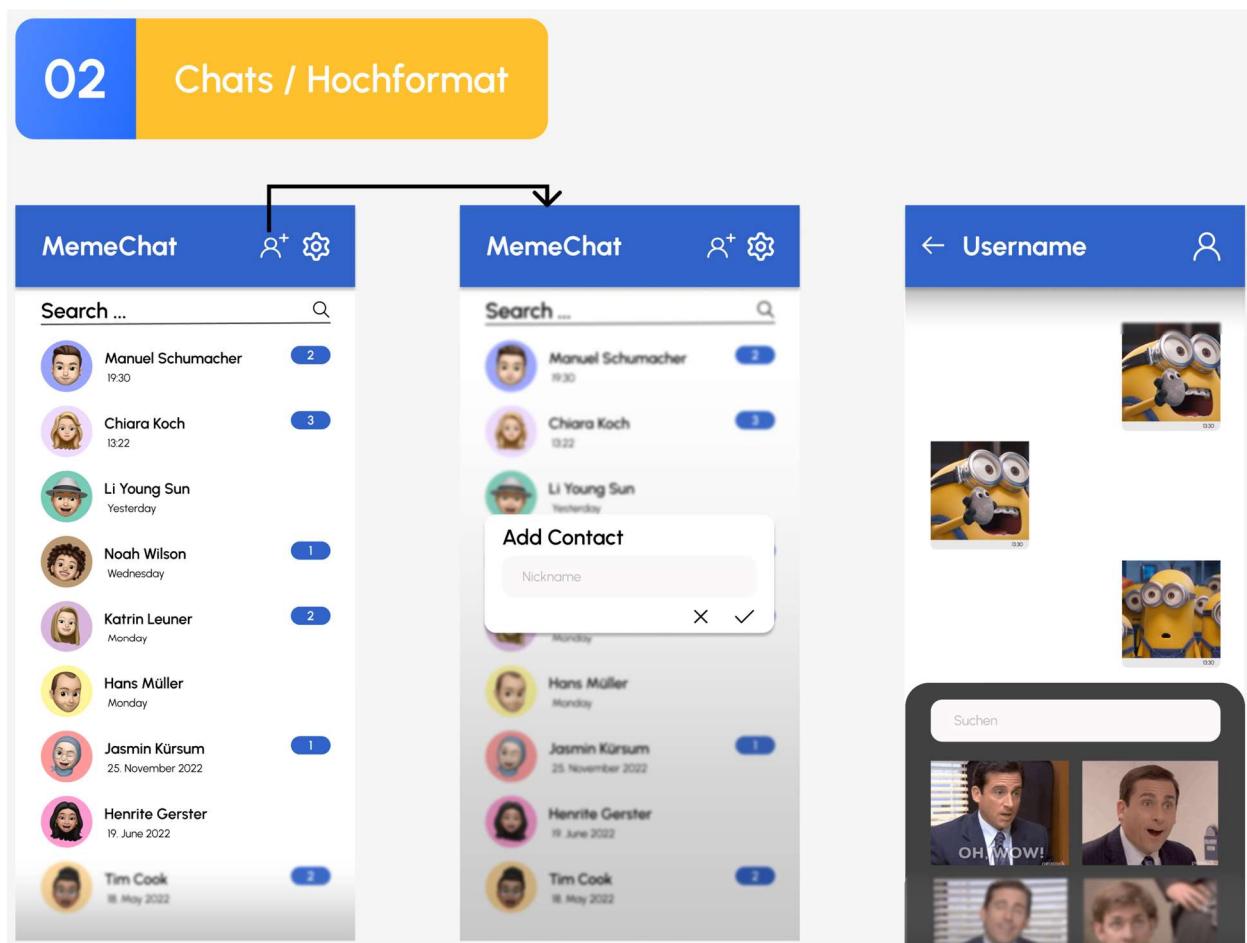
# MemeChat

## 2.10.7 Anmeldeseite



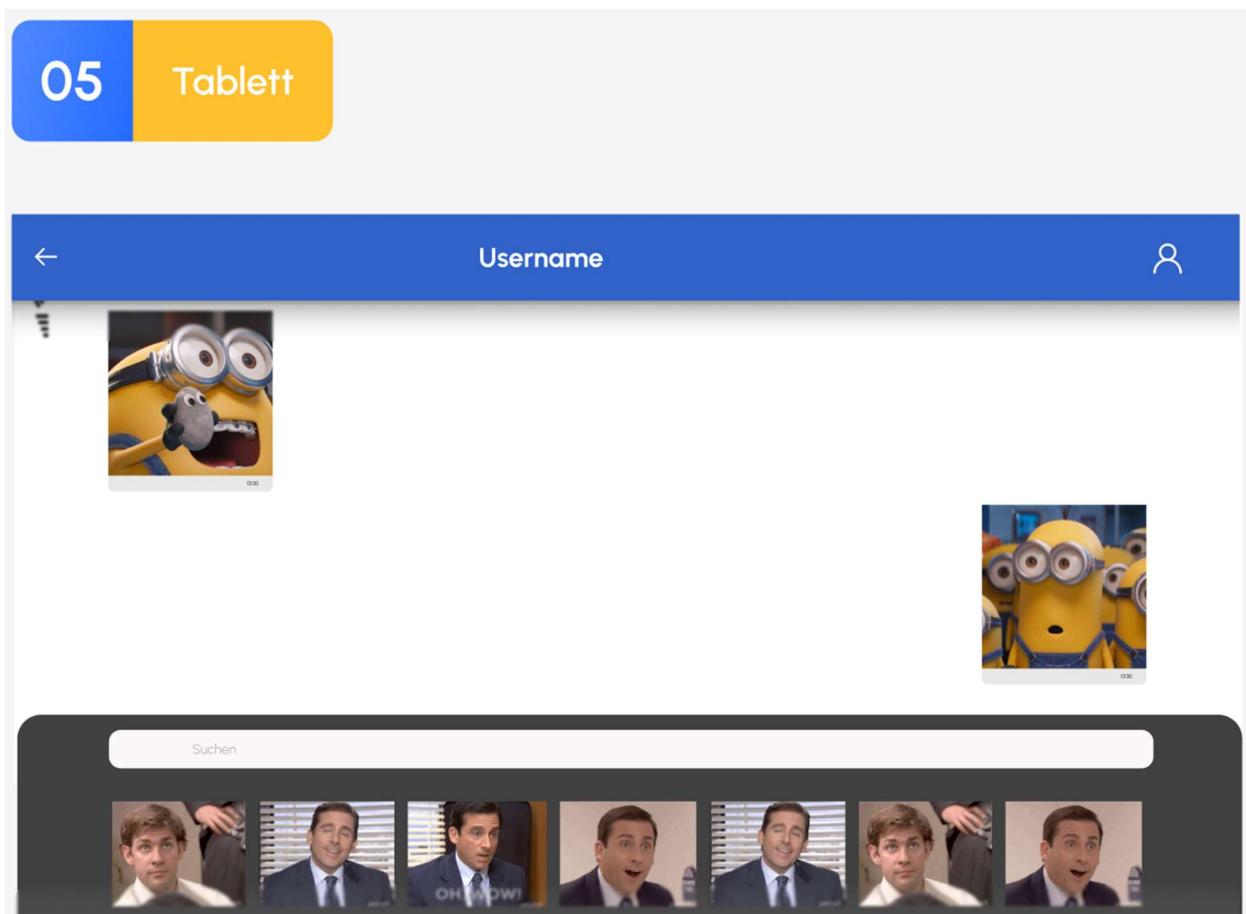
## MemeChat

### 2.10.8 Chats



## 2.10.9 Tablett

Die Tablett Ansicht ist dieselbe wie die Quer, mit dem Unterschied das der Chat grösser und skaliert ist.



## 2.10.10 Konkurrenz

Ein Messenger ist keine neue Idee und wurde bereits vermehrt umgesetzt. Was diese App einzigartig macht, ist, dass alle anderen das Schreiben mit einer Tastatur erlauben und diese nicht. So können die Konversationen auf die wichtigen Dinge im Leben fokussiert werden, Memes. Die Zielgruppen sind Jugendliche im Alter von 13 – 30 Jahren, welche nur kurz Gefühle und Interessen austauschen wollten. Durch die Einzigartigkeit und Spezifizierung der Kunden könnte diese App mit mehr Arbeit konkurrenzfähig werden.

Im Vergleich zu anderen Apps wie Signal<sup>33</sup> ist MemeChat natürlich ohne Widerspruch unterlegen. Nicht nur wird End-To-End-Verschlüsselung nicht erlaubt. Hier wird eine Liste der genauen Vergleiche angezeigt:

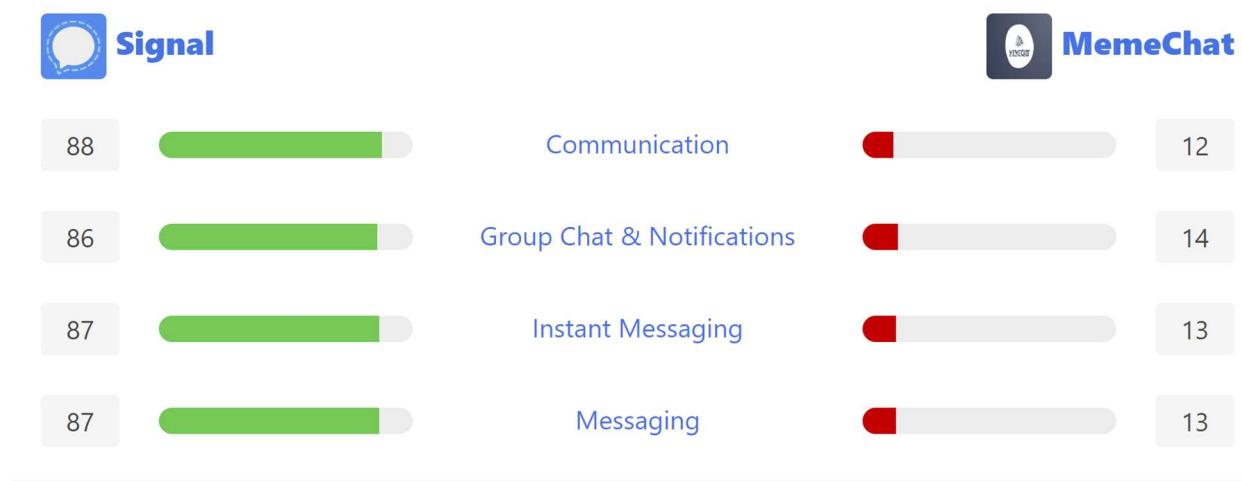


Abbildung 13 Vergleich Signal vs MemeChat

<sup>33</sup> <https://signal.org/en/>

### 3 Systemdokumentation

Ein gut dokumentiertes System nimmt viel Zeit in Anspruch. Wenn kontinuierlich daran gearbeitet wird, dann ist sie auf lange Sicht für die Wartung und Einarbeitung von neuen Mitarbeitern eine hilfreiche Referenz. In dieser Systemdokumentation werden Informationen über die Lösung zu Problemen und Verwendung von Komponenten aufgezeigt. Wenn etwas unklar ist, dann sollte hier eine Beschreibung oder ein Hinweis enthalten sein. In der Bedienungsanleitung wird festgehalten wie die App verwendet werden sollte und was dabei zu beachten ist.

#### 3.1 Installation

In diesem ÜK wird Cordova empfohlen, weswegen hier die Installation beschrieben wird. Zusätzlich befindet sich am Schluss dieses Unterkapitels eine kurze Anleitung für die Installation der Entwicklungsumgebung von MAUI.

Als Umgebung wird Visual Studio Code<sup>34</sup> verwendet, was sehr einfach zu installieren ist. Wie im vorherigen Kapitel erwähnt, ist VS-Code an sich nur ein Texteditor. Mithilfe folgender Extension kann es jedoch zu einer kompletten und umfangreichen IDE werden:

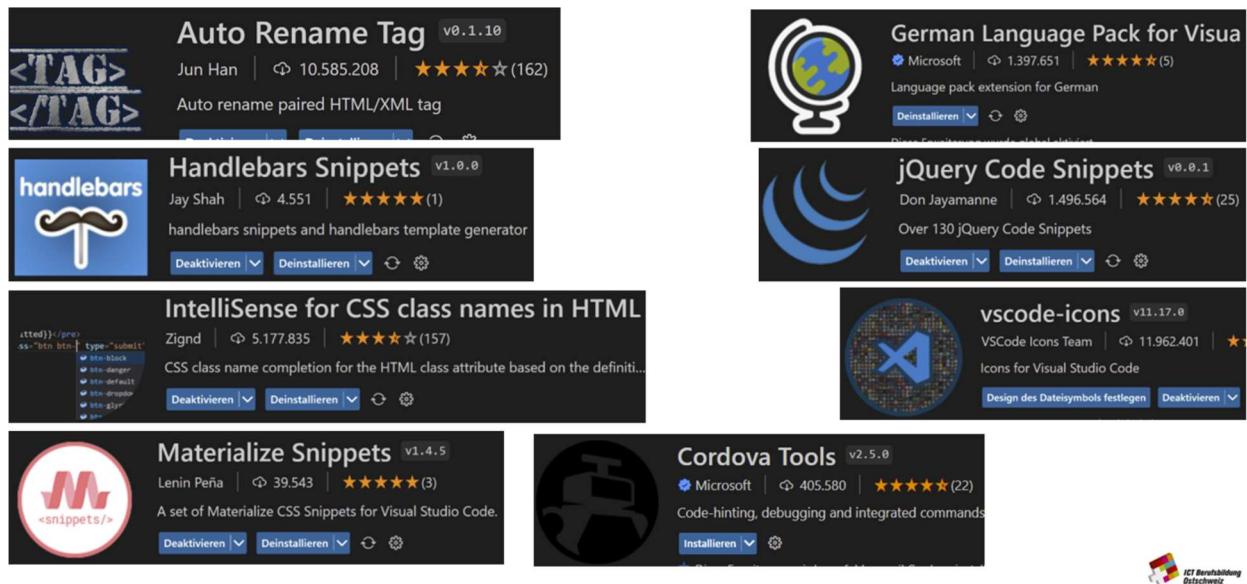


Abbildung 14 VS-Code Extension

Unter dem Standardlaufwerk sollte ein Ordner erstellt werden, welcher keine Leerzeichen oder Sonderzeichen enthält. Zusätzlich muss das Terminal auf Command Prompt eingestellt werden. Wie dies gemacht werden kann ist im folgenden Bild beschrieben. Wenn diese Vorgaben nicht eingehalten werden, hat Cordova beim Kompilieren Probleme.

<sup>34</sup> <https://code.visualstudio.com/>

## MemeChat

Damit ein APK erstellt werden kann und der Android Emulator existiert muss Android Studio<sup>35</sup> installiert werden. Dies dauert sehr lange und könnte bei nicht ausreichenden Berechtigungen zu Problemen führen. Im SDK-Manager sollte das SDK 32 und Android 12 installiert werden:

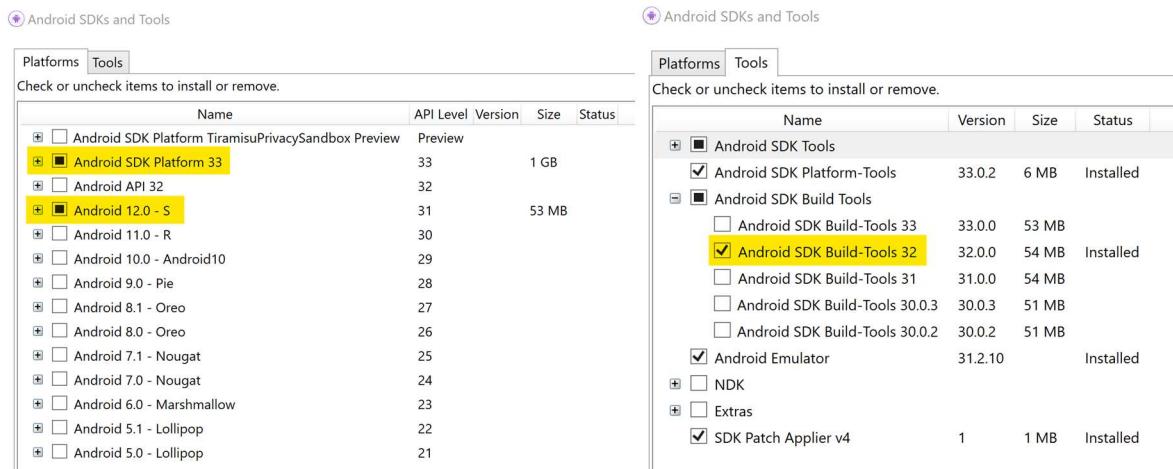


Abbildung 15 SDK-Manager

Damit Lokal ohne ein Gerät programmiert werden kann wird ein Emulator benötigt. Dieser kann im Device Manager erstellt werden. Er verhält sich exakt gleich wie ein richtiges Gerät mit der Ausnahme, dass die Sensoren gesteuert werden können. Der Emulator kann wie folgt erstellt werden:

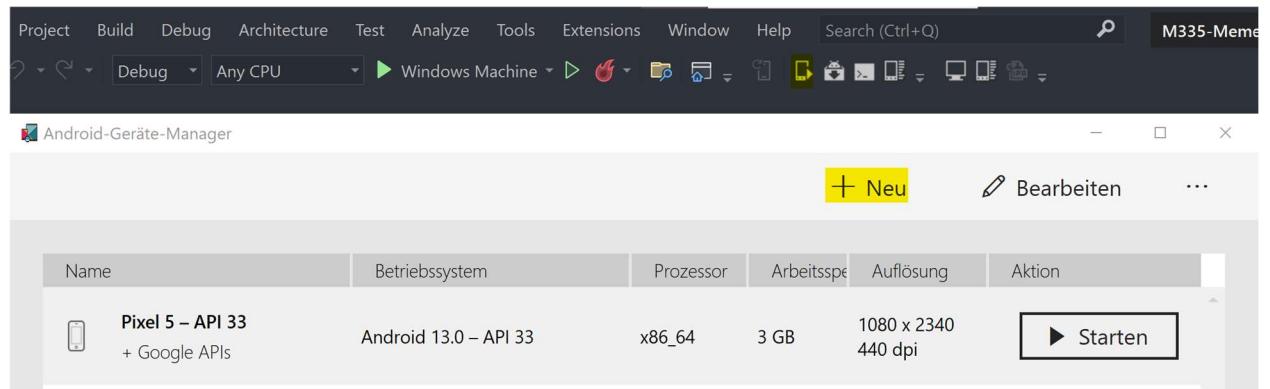
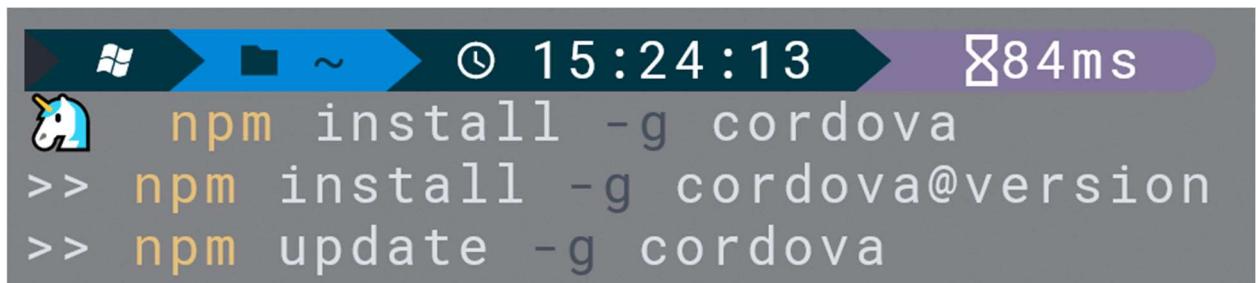


Abbildung 16 Android Emulator hinzufügen

### 3.1.1 Cordova

Cordova kann über den Node-Package-Manager, welcher automatisch mit Node.js mitgeliefert wird, installiert werden. Mithilfe des Flags «-g» wird Cordova global installiert. Wenn eine spezifische Version benötigt wird, dann kann dies über das @ und dann die Versionsnummer gemacht werden. Über Update wird die neuste Version auf die Geräte geladen:

<sup>35</sup> <https://developer.android.com/studio>



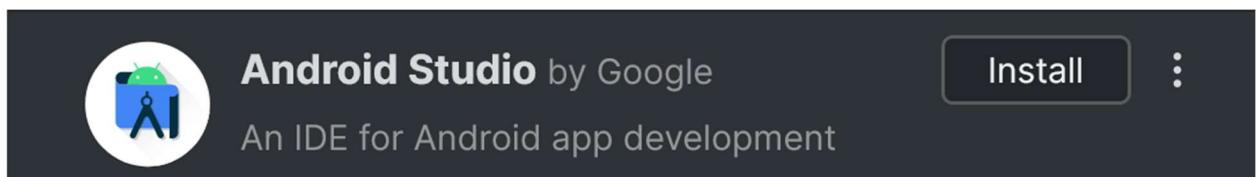
```

Windows [ ~ ] 15:24:13 84ms
npm install -g cordova
>> npm install -g cordova@version
>> npm update -g cordova

```

*Abbildung 17 Cordova über npm installieren*

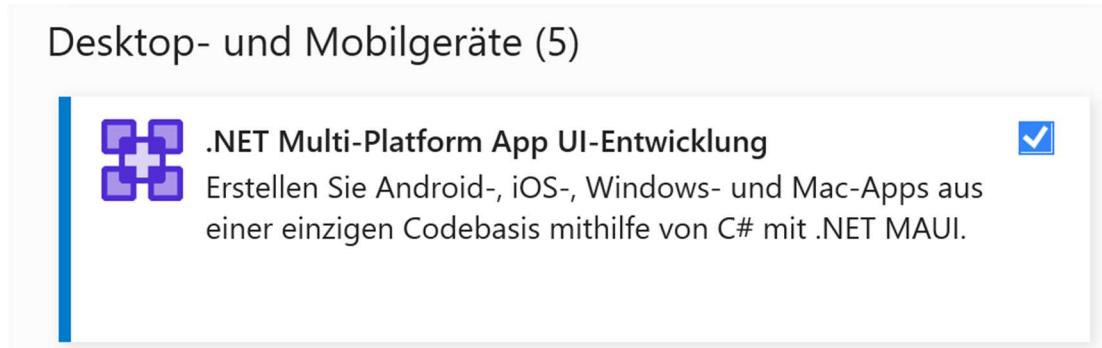
Damit eine APK erstellt werden kann wird zusätzlich zu Cordova eine Installation von Android Studio und Gradle benötigt. Die Gradle Installation wird am besten auf das C Laufwerk gemacht und über die Umgebungsvariablen darauf verwiesen. Android Studio wird am besten über die Jet Brains Toolbox installiert:



*Abbildung 18 Android Studio installieren*

### 3.1.2 Visual Studio

Die Installation<sup>36</sup> von Visual Studio ist sehr einfach und eigentlich mit keinen Problemen verbunden, da die ganze Entwicklungsumgebung mit allen benötigten Daten eingerichtet werden. Es wird empfohlen, einen Rechner mit mindestens 16 GB Arbeitsspeicher und einer modernen CPU zu verwenden, damit alles flüssig läuft. Der Installer kann auf der Microsoft Webseite<sup>37</sup> heruntergeladen werden. Während der Installation müssen den AGBs zugestimmt werden. Im Installer kann dann die Version und die benötigten Komponenten ausgewählt werden. Hauptsächlich muss das Packet «.NET MAUI» installiert werden, damit ein App-Projekt erstellt werden kann:



*Abbildung 19 MAUI installieren*

<sup>36</sup> <https://learn.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2022>

<sup>37</sup> <https://visualstudio.microsoft.com/vs/>

Falls noch weitere Komponenten wie unterschiedliche Versionen oder Git benötigt werden, können diese unter den «individuellen Komponenten» ausgewählt werden. Die Sprache könnte unter diesen Tabs auch geändert werden:

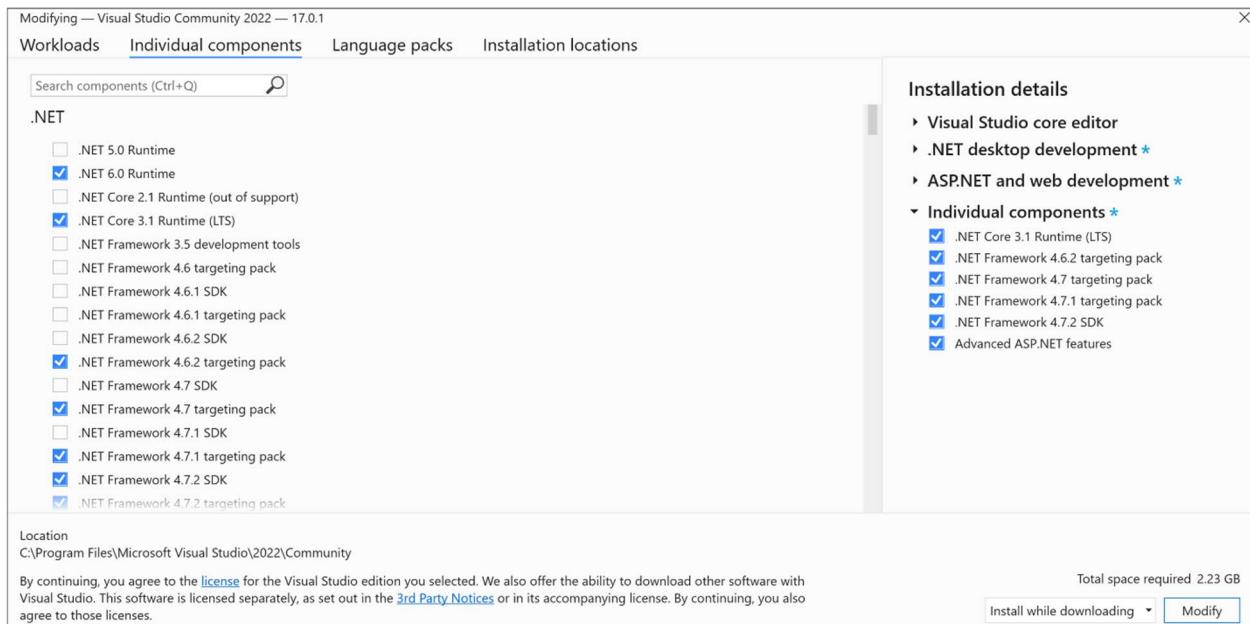


Abbildung 20 Weitere Komponenten

Nach dem Installieren kann ein neues MAUI-Projekt erstellt werden. Sobald alles aufgesetzt ist, kann in der Toolbar ein neues Android Gerät hinterlegt werden. Um die Performance vom Emulator zu verbessern kann Hyper-V aktiviert werden. Wenn ein Mac im Netzwerk ist, dann kann dieser als Build-Server missbraucht werden:

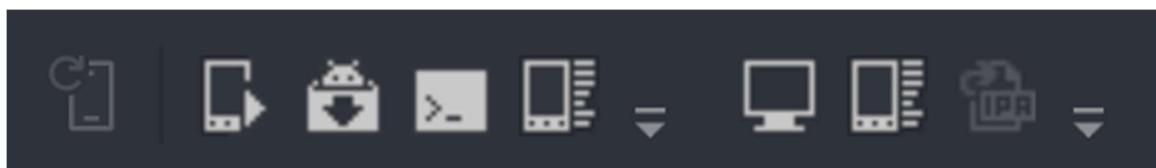


Abbildung 21 Android Emulator erstellen

### 3.1.3 Versionen

Folgende Versionen werden für die Entwicklung für Cordova oder MAUI benötigt:

SERVICE	VERSION
JAVA	1.8 <sup>38</sup>
NODE	16.17.1 (LTS / Current) <sup>39</sup>
NPM	8.19.2 <sup>40</sup>
VISUAL STUDIO	2022 – Enterprise (Community reicht) <sup>41</sup>
.NET	6 (LTS) <sup>42</sup>
ANDROID	13 <sup>43</sup>

Was für Versionen aktuell installiert sind, können über folgende Befehle abgefragt werden:



```

powershell
java -version
>> node -v
>> npm -v

```

Abbildung 22 Installation überprüfen

<sup>38</sup> <https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html>

<sup>39</sup> <https://nodejs.org/en/>

<sup>40</sup> <https://www.npmjs.com/>

<sup>41</sup> <https://visualstudio.microsoft.com/vs/>

<sup>42</sup> <https://dotnet.microsoft.com/en-us/download/dotnet/6.0>

<sup>43</sup> <https://www.android.com/android-13/>

### 3.1.4 Umgebungsvariablen

Damit Cordova weiss, wo die benötigten Dateien sind, müssen Umgebungsvariablen definiert werden. Wichtig ist, dass danach der Rechner neu gestartet wird, damit die Änderungen ihre Wirkung haben.

VARIABLE	WERT
<b>ANDROID_HOME</b>	C:\Users\M335\AppData\Local\Android\Sdk
<b>ANDROID_PLATFORM_TOOLS</b>	C:\Users\M335\AppData\Local\Android\Sdk\platform-tools
<b>ANDROID_SDK_ROOT</b>	C:\Users\M335\AppData\Local\Android\Sdk
<b>ANDROID_TOOLS</b>	C:\Users\M335\AppData\Local\Android\Sdk\tools
<b>PATH</b>	C:\gradle-7.2\bin

## 3.2 Umsetzung Sensor

Hauptsächlich verwende ich den Media Picker, welcher mir Zugriff auf die Kamera und das Dateisystem erlaubt. Vor dem Zugreifen wird überprüft, ob die benötigten Berechtigungen vorhanden sind oder ob der Sensor überhaupt existiert. Dies ist wichtig, um einen Absturz zu vermeiden. Mit dem Aufruf der Methode «`CapturePhotoAsync`» kann die Kamera geöffnet werden und der Benutzer kann ein Foto aufnehmen. Wenn der Benutzer ein Foto aufnimmt, ist der Rückgabewert der Methode ein Nicht-Null-Wert. Das folgende Codebeispiel verwendet den Media Picker, um ein Foto aufzunehmen und es im Cache-Verzeichnis zu speichern:

```
C# Copy

public async void TakePhoto()
{
    if (MediaPicker.Default.IsCaptureSupported)
    {
        FileResult photo = await MediaPicker.Default.CapturePhotoAsync();

        if (photo != null)
        {
            // save the file into local storage
            string localFilePath = Path.Combine(FileSystem.CacheDirectory, photo.FileName);

            using Stream sourceStream = await photo.OpenReadAsync();
            using FileStream localFileStream = File.OpenWrite(localFilePath);

            await sourceStream.CopyToAsync(localFileStream);
        }
    }
}
```

Abbildung 23 Codeausschnitt Kamera öffnen<sup>44</sup>

Folgende Methoden sind auf dem IMediaPicker-Interface verfügbar:

METHODE	BESCHREIBUNG
<b>PICKPHOTOASYNC</b>	Öffnet die Medienauswahl, um ein Foto auszuwählen
<b>CAPTUREPHOTOASYNC</b>	Öffnet die Kamera, um ein Foto zu schiessen
<b>PICKVIDEOASYNC</b>	Öffnet die Medienauswahl, um ein Video auszuwählen
<b>CAPTUREVIDEOASYNC</b>	Öffnet die Kamera, um ein Video zu schiessen

<sup>44</sup> <https://learn.microsoft.com/en-us/dotnet/maui/platform-integration/device-media/picker?tabs=windows>

Alle geben als Rückgabewert einen `FileResult` zurück, welcher das Lesen des Streams erlaubt. Ein Stream sind die Bytes einer Datei oder eines anderen Mediums. Wenn ein Bild ausgewählt wird, dann kann auch der Pfad dazu ausgewertet werden.

### 3.3 Datenbank

Die Datenbank verwende ich als Speicher der Nachrichten. Für .NET MAUI verwende ich C#, was eine objektorientierte Programmiersprache ist. Um diese Entitäten auf die Datenbank zu übertragen, gibt es sogenannte ORM (Object-Relational-Mapper), welche das SQL schreiben übernehmen. Es hat sich eingebürgert, dass alle NET-Applikationen Entity Framework (EF)-Core<sup>45</sup> verwenden. Durch die Plattformunabhängigkeit und hohe Unterstützung eignet es sich auch für diese App.

Durch die Erstellung eines DB-Kontextes wird eine Verbindungsklasse erstellt. Diese kann sich dann über eine URL mit der Datenbank verbinden. Die URL enthält die Anmeldedaten, Datenbank und den Server, mit welchem verbunden werden sollte. Durch die `DbSet` Klasse wird gekennzeichnet, was für Klassen in die Datenbank übernommen werden sollen. Sobald eine Änderung an der Klasse gemacht wird, kann über eine automatische Migration diese Änderung auch auf die Datenbank gebracht werden. Durch Annotationen auf den Properties wird die Validation eingerichtet. Ein Kontext könnte wie folgt aussehen:

```
5 references | Manuel Schumacher, 7 hours ago | 1 author, 1 change
public class ServerDbContext : DbContext
{
    0 references | Manuel Schumacher, 7 hours ago | 1 author, 1 change
    public ServerDbContext(DbContextOptions<ServerDbContext> options) : base(options)
    {
        0 references | Manuel Schumacher, 7 hours ago | 1 author, 1 change
        public DbSet<User> Users { get; set; }
        0 references | Manuel Schumacher, 7 hours ago | 1 author, 1 change
        public DbSet<Message> Messages { get; set; }
        0 references | Manuel Schumacher, 7 hours ago | 1 author, 1 change
        public DbSet<Chat> Chats { get; set; }
    }
}
```

Abbildung 24 Beispiel `DbContext`

Um die Datenbank über Dependency Injection<sup>46</sup> zu verwenden muss dies beim Erstellen der App eingerichtet werden. Mithilfe des Builders kann ein Service hinterlegt werden. Im nächsten Bild wird gezeigt wie ein Kontext hinterlegt werden kann und was für Einstellungen gemacht werden müssen.

---

<sup>45</sup> <https://learn.microsoft.com/en-us/ef/core/>

<sup>46</sup> <https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection-usage>

```
// Datenbanken mit Dependency Injection hinterlegen
var builder = MauiApp.CreateBuilder();
builder.Services.AddDbContext<ServerDbContext>(options =>
    options.UseMySql(Constants.ServerDbConnectionString, ServerVersion.Create(10, 5, 12, ServerType.MariaDb),
        m => m.EnableRetryOnFailure(5, TimeSpan.FromSeconds(10), null)));
```

Abbildung 25 DbContext einstellen

Wenn in einer Komponente ein Property mit «Inject» versehen ist, dann wird während der Laufzeit versucht eine Instanz über DI zu erstellen. Abhängig von der Lebenszeit des Services wird ein bestehender oder neuer verwendet.

### 3.3.1 Installation

Um das Entity Framework zu verwenden, werden einige NuGet-Pakete benötigt, welche über Visual Studio oder die Konsole hinzugefügt werden können. Zunächst wird der SQLite-Provider über die Konsole installiert:

PowerShell

 Copy

```
Install-Package Microsoft.EntityFrameworkCore.SQLite
```

Wenn eine andere Datenbank verwendet werden sollte, dann kann einer der unterstützten Datenbanken anstatt SQLite geschrieben werden. Für das verwenden von Annotationen und des DB-Kontextes wird noch dieses Packet benötigt.

PowerShell

 Copy

```
Install-Package Microsoft.EntityFrameworkCore.Tools
```

### 3.3.2 Unterstützte Datenbanken

Das Entity Framework unterstützt sehr viele Datenbankproviders<sup>47</sup>. Das Einzige, was gemacht werden muss, damit die Datenbank funktioniert, ist, dass das NuGet-Package<sup>48</sup> zu installieren.

- SQL-Server
- SQLite
- In-Memory
- Azure Cosmos DB
- PostgreSQL
- MySQL
- Oracle DB
- Und noch viele weitere

## 3.4 Persistenz

Damit die Daten auch bei einem Neustart oder Offline vorhanden bleiben werden sie in die Datenbanken gespeichert. Wenn eine Internetverbindung besteht, dann werden die neuen Nachrichten auf die lokale SQLite-Datenbank geladen. Sobald eine Nachricht versendet wird, wird diese auf beide Datenbanken geschrieben. So bleiben die Nachrichten des Servers den lokalen synchronisiert. Aus dem lässt sich schliessen, dass das Synchronisationsverfahren verwendet wird. Durch diese Art ist es möglich, dass die App auch mit mehreren Accounts verwendet werden könnte.

## 3.5 Tests

Die App wird mithilfe von Unit Tests automatisch getestet. So wird sichergestellt, dass die Methoden die gewünschte Funktionalität erfüllen. Während dem Programmieren werden diese im Hintergrund ausgeführt, um sicherzustellen, dass auch bei Änderungen noch alles funktioniert. Ob die Methoden richtig eingebunden sind und auch richtig verwendet werden wird von Hand getestet. Sobald die Applikation fertiggestellt ist, wird sie anhand der vordefinierten Testfälle, welche im nächsten Kapitel folgen, getestet.

Ich persönlich kenne das UI und weiss, was für Knöpfe ich drücken muss. Aus diesem Grund fragte ich einen Schulkamerad, ob er die App testen könnte. Er machte die Testfälle durch und führte die gängigsten Aktionen durch. Die daraus erhaltenen Erkenntnisse baute ich anschliessen in das App ein und liess ihn erneut testen.

<sup>47</sup> <https://learn.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli>

<sup>48</sup> <https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Sqlite>

Es wird somit Unit Tests und Manuelles als Verfahren verwendet. Das händische Testen könnte auch automatisiert werden. Dazu reicht aber die Zeit und der Fokus dieses ÜKs nicht.

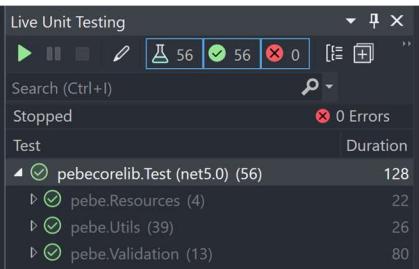
## 3.6 Simulationsumgebungen

Beim Testen ist wichtig, dass die Version getestet wird, welche die Kunden am Schluss anwenden. In jedem Framework gibt es eine Entwicklungs- und Produktionseinstellung, welche sich beim Verhalten unterscheiden. Die finale Version wird optimiert und enthält keine debugg Informationen. In meiner Entwicklungsumgebung gibt es die Möglichkeit, dass Unit Tests während der Entwicklung ständig im Hintergrund ausgeführt werden. Wichtig anzumerken ist, dass dieses Feature kostenpflichtig und einen guten Computer benötigt.

### 3.6.1 Unit Test

Die Unit Tests werden im Hintergrund durch Visual Studio ausgeführt. Dabei wird eine kleinere Applikation mit nur den benötigten Klassen zusammengebaut und die entsprechenden Komponenten ausgeführt. Bei den Tests kommt es nicht auf die Anzahl, sondern vielmehr auf die Isolation und Qualität an, damit ein Bereich auf alle möglichen Extremfälle getestet ist. Im Test Explorer kann dann der Test ausgeführt und evaluiert werden. Falls ein Test fehlschlägt, wird dies visuell dargestellt.

Aus meinem Betrieb ([pebe AG](#)) habe ich ein Unit Test gesucht, welcher ich geschrieben habe und alle Anforderungen erfüllt. Dabei wird verhindert, dass ein Passwort in ein Log oder sonstiges ausgegeben wird. Der Test sieht wie folgt aus:



```
[TestMethod]
[0 references | 0 changes | 0 authors, 0 changes]
public void ToSecureStringTestOk()
{
    string secure = "TOP_SECRET";
    var secureString = secure.ToSecureString();
    Assert.AreNotEqual(secure, secureString.ToString());
}
```

Test	Duration
pebecorelib.Test (net5.0) (56)	128
pebe.Resources (4)	22
pebe.Utils (39)	26
pebe.Validation (13)	80

Abbildung 26 pebe Live Produktionstests<sup>49</sup>

### 3.6.2 Emulator

Die Funktionalität des Apps wird auf einem Emulator getestet, welcher sich fast gleich wie ein richtiges Mobiltelefon verhält. Der Unterschied ist, dass die Sensoren gesteuert und modifiziert werden können. Da es eben nur fast gleich ist müsste es unbedingt auch noch auf einem physischen

---

<sup>49</sup> <https://pebelive.ch/>

Mobiltelefon ausgiebig getestet werden, da es sich dort wiederum anders verhalten könnte. Der Android Emulator verfügt über erweiterte Einstellungsmöglichkeiten:

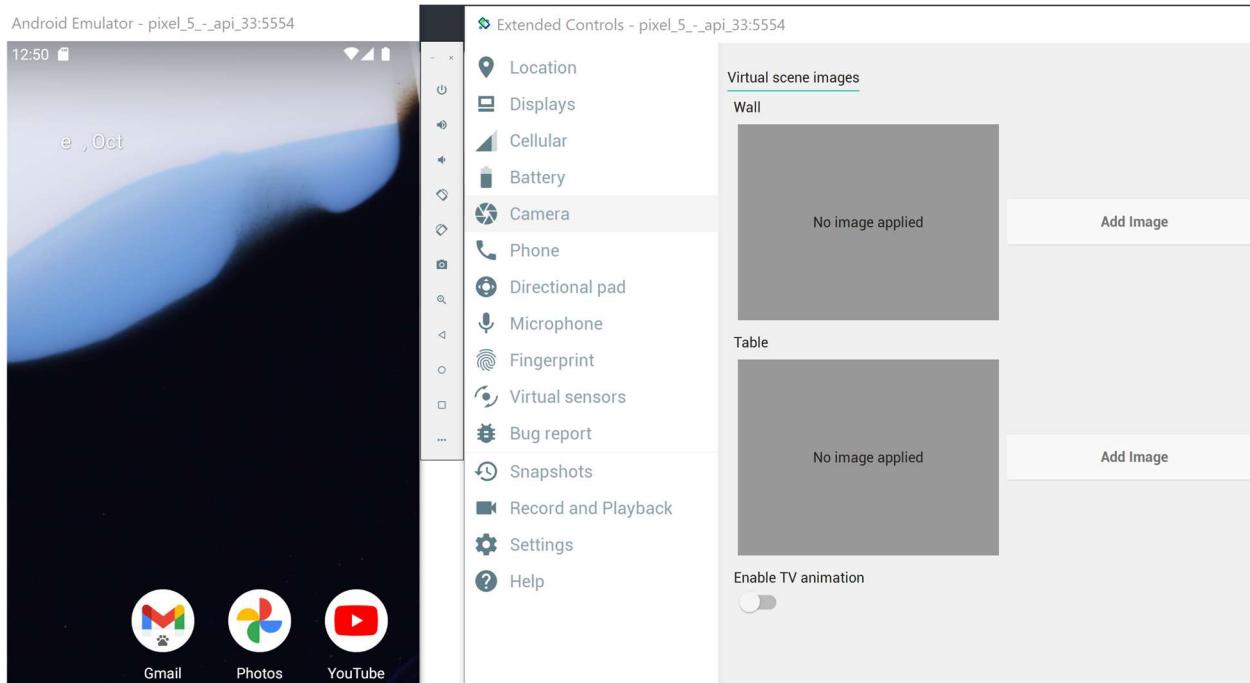


Abbildung 27 Android Emulator

## 3.7 Probleme

Beim Installieren und Programmieren von Visual Studio und MAUI gab es keine Probleme. Es war im Grund nur ein Durchklicken mit wenig potenzial etwas falsch zu machen. Auch während der Entwicklung konnte ich von meinem Vorwissen profitieren und bekam nie grössere Fehler, bei welchen ich Hilfe holen musste. Meist waren es CSS-Kleinigkeiten, welche sich anders als erwartet verhielten.

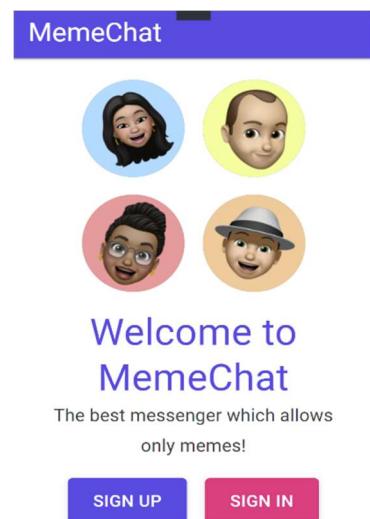
Die Einrichtung von Cordova war im Gegenteil fast nur mit Problemen versehrt. Ich hatte durch die Schule die neuste Version von Java auf meinem Computer. Es musste jedoch eine ältere verwendet werden, was ich zu Beginn nicht wusste. Beim De- und Neuinstallieren musste ich die Umgebungsvariablen von Hand neu setzen. In der Installation von Visual Studio, was ich für meine Arbeit im Betrieb verwende, wird auch eine modifizierte Version von Node.js installiert. Diese Ausführung konnte keine Befehle direkt in der Kommandozeile ausführen. Durch das AD, in welchem ich bin, ist mein Benutzername durch einen Punkt getrennt. Bei anderen Schülern war dies nicht der Fall und sie bekamen Probleme mit Sonder- und Leerzeichen.

### 3.8 Bedienungsanleitung

Die Applikation ist so einfach und verständlich aufgebaut, dass es eigentlich keine Anleitung benötigt. Trotzdem ist hier eine kurze Beschreibung der Funktionen vorhanden.

Wenn das App gestartet wird und noch kein Konto hinterlegt ist, dann kommt man zu dieser schönen Ansicht. Hier sieht man eine kurze Beschreibung des Projektes und einen Willkommenstext. Über die Optionen «Sign Up» kann ein neues Konto erstellt werden. Wenn ein Konto bereits vorhanden ist, dann kann man sich mit diesem über «Sign In» anmelden. Zu beachten gilt, dass für die Anmeldung und Registrierung eine Internetverbindung benötigt wird. Wenn dies nicht der Fall ist, dann kommt eine Fehlermeldung.

Bei der Registrierung werden gewisse Informationen verlangt, welche später von anderen Benutzern eingesehen werden können. Über «Fill Your Profile» kann zurück zur Startseite navigiert werden. Der Klick auf den Avatar öffnet die Kamera, welche das Hinterlegen eines Profilbildes erlaubt. Dies ist nur optional aber empfohlen, da ansonsten ein generiertes Bild angezeigt wird. Ein leicht zu merkender Nickname ist wichtig, da über diesen ein neuer Kontakt hinterlegt wird und die Anmeldung erfolgt. Alle Felder werden auf die Länge und das geforderte Format validiert. Wenn das Formular nicht gültig ist, dann erfolgt auch nicht die erstellung.



MemeChat

← FILL YOUR PROFILE

Full Name

Nickname

Date of Birth  
07.10.2006

Email

Password

About

0 / 250

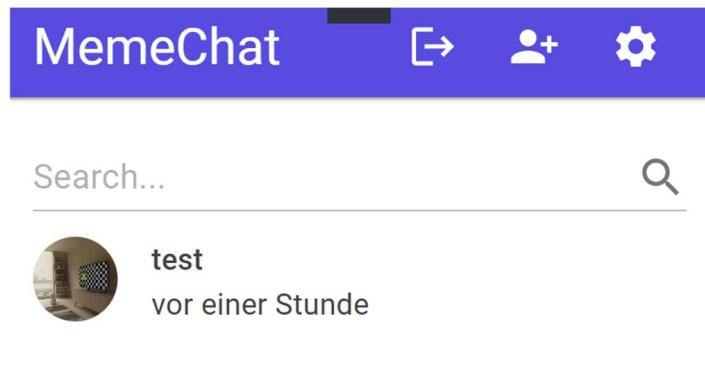
CONTINUE

Abbildung 28 Registrierung

## MemeChat

Wenn das Anmelden ausgewählt wird, dann muss der Nickname und das Passwort eingegeben werden. Auch hier kann wieder zur Startseite zurückgegangen werden. Wenn auf «Continue» geklickt wird, dann wird überprüft, ob die Anmeldung gültig ist. Wenn ja, dann wird auf die Home-Seite navigiert.

Auf der Home-Seite gibt es zunächst die Task Bar, mit welcher sich abgemeldet werden kann, ein neues Konto erstellt wird oder ein bestehender Account bearbeitet werden kann. Dort wird die gleiche Registrierungsseite wie verwendet. Mit der Suchleiste kann nach einem Benutzer gesucht werden. Hier



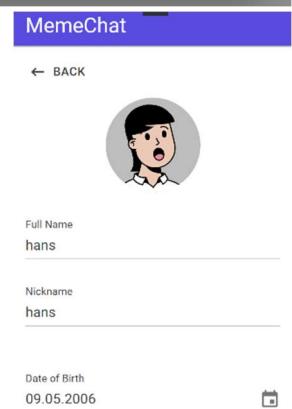
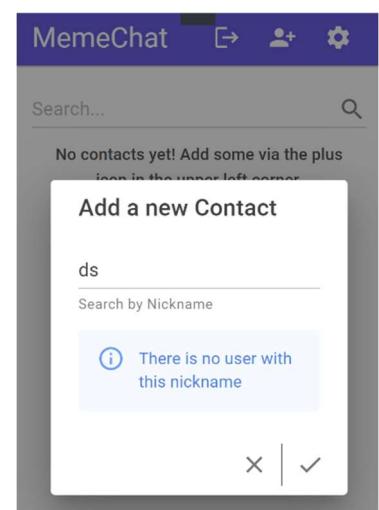
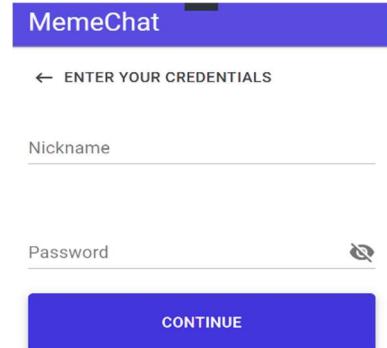
wird geschaut, ob der Name eine der Zeichen enthält. Die Benutzer, mit welchen ein Chat besteht, werden darunter in einer Liste dargestellt. Man sieht das Profilbild, den Namen und wann die letzte Nachricht versendet wurde. Durch den Klick auf den Benutzer wird dann der Chat geöffnet.

Ein neuer Kontakt kann über das Kontakticon mit dem Plus hinzugefügt werden. Hier kommt ein Popup, in welchem der Nickname angegeben werden muss. Wenn er gefunden wird, dann kann können Nachrichten versendet werden. Ansonsten kommt eine Fehlermeldung.

Die Chatansicht kann entweder über das Kontaktinzufügen oder Klicken auf den Chat geöffnet werden. In dieser kann in der Navigation zurück oder die Kontaktinformationen angezeigt werden. Die



Kontaktansicht ist die gleiche wie bei der Registrierung. Die Nachrichten werden hier schön der Reihe aufgelistet. Links sind die empfangenen und rechts die gesendeten. Über den Plusknöpf öffnet sich die Dateiauswahl, mit welcher ein Foto versendet werden kann. Das Versenden ist nur mit Internet möglich. Den Chat anschauen kann aber auch ohne Internet geschehen. Der Benutzer merkt davon nicht ausser, dass die Auswahl nicht auftaucht.



## 4 Testdokumentation

Damit eine Applikation als vollendet und funktionstüchtig angesehen wird, braucht es Tests. Diese stellen sicher, dass die Funktionalitäten richtig umgesetzt wurden und der Kunde die gewünschten Aktionen ausführen kann. Wenn ein Fehler auftaucht, dann ist der Kunde sehr schnell unzufrieden und gibt eine schlechte Bewertung.

Ein Testfall beschreibt eine elementare Funktionalität, welche getestet werden soll. Er hat eine Beschreibung und eine Voraussetzung. Die Voraussetzung ist eine Bedingung, die erfüllt sein muss, damit der Testfall ausgeführt werden kann. Die Schritte zum Ausführen werden auch mitgegeben, damit es zu keinen Missverständnissen kommt. Aus diesem Grund müssen diese Schritte exakt ausgeführt werden.

Beim Testen der App holte ich mir Hilfe von einem meiner Schulkameraden. Dieser testete die Bedienung und gängigsten Funktionen. So konnte ich sichergehen, dass auch eine Person, welche das App noch nie gesehen hat, die App benutzen kann. Das Ergebnis war, dass die Benutzerführung sehr gut gelungen ist.

**Die Testfälle befinden sich zur Übersichtlichkeit in einem anderen Dokument.**

## 5 Implementation

Nachdem die Planung vollendet ist und alle Anforderungen abgeklärt sind kann es zur Entwicklung kommen. Bei der Implementation zeigt sich wie gut geplant wurde und wie mit Fehler umgegangen werden kann. Da das Projekt eher klein ist und wenige Funktionalitäten besitzt, sind die Möglichkeiten für die Implementation sehr eingeschränkt. Es wurde trotzdem versucht ein möglichst qualitatives Projekt zu erstellen.

In den nächsten Kapiteln wird erläutert wie der Sensor und die Datenbank angesprochen werden und was für Probleme es dabei gab. Ohne zu wissen, wie ein Projekt aufgesetzt wird und wo die Einstellungen gemacht werden können, kommt man nicht weit. Dementsprechend habe diesen Vorgang ausgiebig mit Bildern und Text dokumentiert.

### 5.1 Sensor

Hauptsächlich wird der Media Picker Sensor verwendet, welcher mir erlaubt ein Foto zu schiessen oder auszuwählen. Wenn in der Anmeldung auf das Icon geklickt wird, dann wird ein Foto geschossen. Zuvor wird jedoch überprüft, ob das Gerät überhaupt über eine Kamera verfügt. Beim Auswählen eines Bildes funktioniert es exakt gleich. Der Stream, welcher zurückgegeben wird, wird in das Base64-Format gebracht, damit es in die Datenbank gespeichert werden kann.

```

0 references | 0 changes | 0 authors, 0 changes
public static class SensorViewModel
{
    0 references | 0 changes | 0 authors, 0 changes
    public static async Task<string> TakePhoto()
    {
        try
        {
            if (MediaPicker.Default.IsCaptureSupported)
            {
                var photo = await MediaPicker.Default.CapturePhotoAsync();
                return await CreateBase64String(photo);
            }
        }
        catch { /* Handle permission denied */ }

        return string.Empty;
    }

    0 references | 0 changes | 0 authors, 0 changes
    public static async Task<string> PickPhoto()...
}

2 references | 0 changes | 0 authors, 0 changes
private static async Task<string> CreateBase64String(FileResult photo)
{
    if (photo != null)
    {
        using var stream = await photo.OpenReadAsync();
        using var ms = new MemoryStream();

        stream.CopyTo(ms);
        var bytes = ms.ToArray();

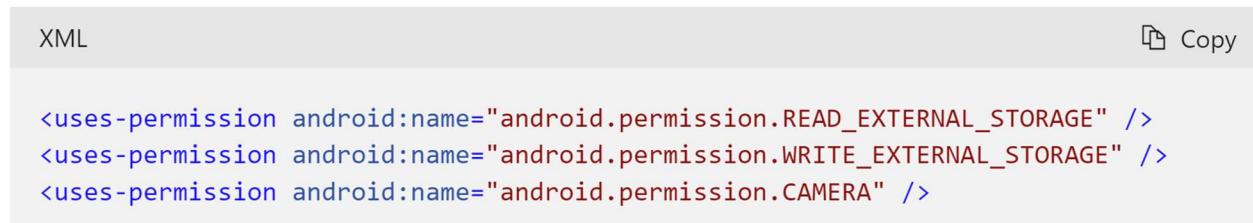
        return Convert.ToBase64String(bytes);
    }

    return string.Empty;
}

```

Abbildung 29 Foto machen

Damit ein Bild gemacht oder ausgewählt werden kann, müssen die Berechtigungen definiert werden. Diese sind dann auch im App Store sichtbar. Sobald im App eine Methode benutzt wird, welche die Berechtigungen benötigt wird ein Popup zur Bestätigung angezeigt. Sie müssen für die Plattformen jeweils spezifisch in den Konfigurationsdateien, welche sich unter «Platforms» befinden, definiert werden. Folglich werden folgende Zeilen eingefügt:



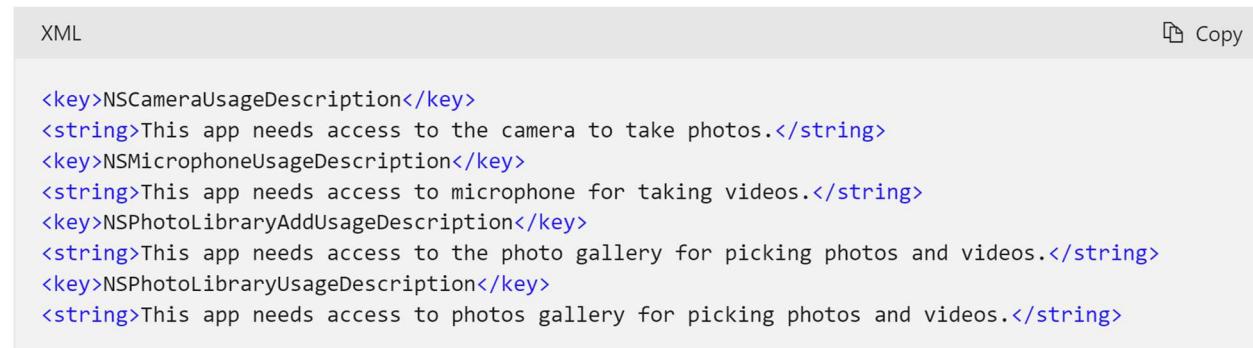
```

XML
Copy

<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />

```

Abbildung 30 Android Foto Berechtigungen



```

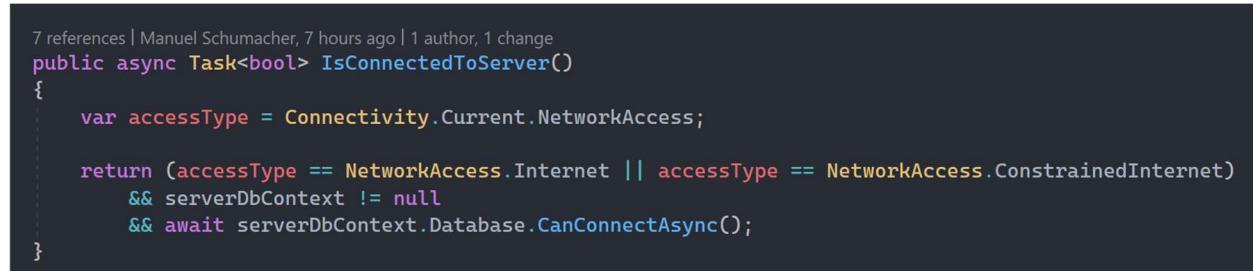
XML
Copy

<key>NSCameraUsageDescription</key>
<string>This app needs access to the camera to take photos.</string>
<key>NSMicrophoneUsageDescription</key>
<string>This app needs access to microphone for taking videos.</string>
<key>NSPhotoLibraryAddUsageDescription</key>
<string>This app needs access to the photo gallery for picking photos and videos.</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>This app needs access to photos gallery for picking photos and videos.</string>

```

Abbildung 31 iOS Foto Berechtigungen

Als nebensächlicher Sensor wird die Netzwerkverbindung<sup>50</sup> getestet. Nur wenn eine Verbindung besteht, dann können die Daten auf die Datenbank auf dem Server geschrieben werden. Über die Netzwerkklasse kann die aktuelle Verbindung ausgelesen werden. Anhand eines Enums wird dann der Status überprüft. Entweder es besteht voller Zugang zum Internet oder mit Einschränkungen über ein Webportal.



```

7 references | Manuel Schumacher, 7 hours ago | 1 author, 1 change
public async Task<bool> IsConnectedToServer()
{
    var accessType = Connectivity.Current.NetworkAccess;

    return (accessType == NetworkAccess.Internet || accessType == NetworkAccess.ConstrainedInternet)
        && serverDbContext != null
        && await serverDbContext.Database.CanConnectAsync();
}

```

Abbildung 32 Netzwerkkonnektivität testen

---

<sup>50</sup> <https://learn.microsoft.com/en-us/dotnet/maui/platform-integration/communication/networking?tabs=windows>

### 5.1.1 Probleme

Beim Verwenden des Media Pickers gab es keine Probleme. Die Dokumentation erklärte mir genau was ich machen muss und wo die Einstellungen zu betätigen sind. Die Bilder dann darzustellen war eher die Schwierigkeit, da ich sie als URL angeben musste. Ich griff somit auf die Base64 Repräsentation zurück.

Bei der Verbindung zur Datenbank wird abgewartet, bis die Datenbank antwortet. Wenn diese nicht erreichbar ist, dann wird eine bestimmte Zeit abgewartet. Während dieser Zeit sieht der Benutzer nicht, was passiert und muss gegebenenfalls länger warten. Um dem entgegenzuwirken, teste ich zuvor ob überhaupt eine Verbindung zum Internet gemacht werden kann.

## 5.2 Datenbank

Bei der Datenbankkommunikation wurde mit dem Repository Pattern gearbeitet. So ist das Testen mit einer Datenbank sehr einfach, da die Klasse durch einen Mock ersetzt werden kann. Mit der Datenbank wird über das Entity Framework gesprochen. Als Abfragesprache wird Linq<sup>51</sup> verwendet, was eine Ansammlung lesbaren Methoden ist. So ist der Code nachvollziehbar. SQL wird nicht direkt geschrieben, sondern über die Parameter generiert. Dies erhöht die Wartbarkeit drastisch, da die SQL-Abfragen bei Änderungen nicht manuell angepasst werden müssen. SQLite-Datenbank wird über eine Dateiverbindung verwaltet. Die Serverdatenbank, an welche die Nachrichten gesendet werden, wird über HTTPS abgefragt. So ist die Verschlüsselung der Daten gewährleistet. Natürlich könnte die Abfrage über Tabellen hinweg geschehen und Aggregationen enthalten.

Für eine Abfrage wird zunächst der Datenbankkontext benötigt. Dieser kann entweder durch das Instanziieren der Klasse oder Dependency Injection<sup>52</sup> erstellt werden. Wichtig ist, dass nach dem Öffnen einer Verbindung diese auch wieder geschlossen wird. Ansonsten kann es bei einer weiten Abfrage zu einem Fehler kommen. Wenn das App mehrere Threads besitzt, dann sollte immer nur einer für die Datenbankkommunikation zuständig sein, um Änderungen über Threads hinweg zu vermeiden. Durch meine ausgiebige Arbeit mit EF-Core habe ich ein enormes Wissen aufgebaut. Aus diesem Grund gab es keine Probleme bei der Verbindung oder der Verwaltung.

<sup>51</sup> <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>

<sup>52</sup> <https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection>

### 5.2.1 Abfragen

Eine Abfrage mit dem Entity Framework und Linq könnte wie folgt aussehen:

```
C# Copy

using (var db = new BloggingContext())
{
    var blogs = db.Blogs
        .Where(b => b.Rating > 3)
        .OrderBy(b => b.Url)
        .ToList();
}
```

Der Code ist gut leserlich und auch verständlich für jemanden, der nicht viel mit Datenbanken zu tun hat. Es wird aus der Tabelle Blogs nach denjenigen, die ein Rating über drei besitzen gefiltert. Diese Liste wird dann anhand der URL sortiert und übergeben.

### 5.2.2 Speichern

Das Speichern verhält sich gleich wie das Abfragen von Daten, mit der Sonderheit, dass gespeichert bleibt, was für Entitäten bereits in der Datenbank vorhanden sind. Abhängig vom Status wird ein Insert oder Update ausgeführt. Erst wenn «SaveChanges» aufgerufen wird, dann werden die Daten auf die Datenbank übertragen.

```
C# Copy

using (var db = new BloggingContext())
{
    var blog = new Blog { Url = "http://sample.com" };
    db.Blogs.Add(blog);
    db.SaveChanges();
}
```

### 5.2.3 Backup

Ein Backup wird automatisch von Hostinger jeden Tag gemacht, weswegen ich diese Aufgabe nicht explizit übernehmen muss. Über die CLI, welche über folgenden Befehl installiert werden kann, können Migrationen und Backups der aktuellen Datenbank gemacht werden.

```
.NET CLI Copy

dotnet tool install --global dotnet-ef
```

Wie das Format des Backups aussieht, ist jedoch abhängig von der Datenbank, mit welcher EF verbunden ist. PostgreSQL wird die Datenbank als ein SQL-Skript ausgeben, welches dann versiert werden kann. Der SQL-Server hingegen speichert die Daten in einer verschlüsselten Datei, welche nur wieder durch einen Server eingesehen werden kann.

## 5.2.4 Aufsetzen

Die verwendete Datenbank wird über Hostinger gehostet. Bei einem einfachen Plan können bis zu sechs Datenbanken erstellt werden. Unter dem Datenbank-Tab können neue Datenbanken erstellt werden. Dazu muss ein Name und der dazugehörige Benutzer erstellt werden:

The screenshot shows a web-based interface for creating a new MySQL database. At the top, it says "MySQL Databases". Below that, there's a section titled "+ Create a New MySQL Database And Database User". It contains three input fields: "MySQL database name" with the value "u166616309\_MemeChat", "MySQL username" with the value "u166616309\_MemeChat", and "Password\*" with the value "\*\*\*\*\*". To the right of the password field are two small icons: a keyhole and an eye. At the bottom of the form is a green "Create" button with a white checkmark icon.

Abbildung 33 Datenbank erstellen

Über phpMyAdmin kann dann auf die Datenbank zugegriffen werden. Dort wird auch die URL angezeigt, welche dann für die Verbindung verwendet wird. Das Passwort und die Berechtigungen können auch in diesem Bereich neu vergeben werden.

## 5.2.5 Probleme

Beim Einrichten und Verwenden der Datenbank gab es keine Probleme. Die Verbindungen konnten ohne Probleme lokal und über das Internet gemacht werden. Bisher habe ich immer mit SQL-Server oder PostgreSQL<sup>53</sup> gearbeitet, weswegen ich zuerst herausfinden musste wie MySQL integriert wird. Die Verbindung ist etwas anders, da die Version der Datenbank angegeben werden muss. Der Grund dafür ist, dass der Syntax davon abhängig ist. Mithilfe der Dokumentation fand ich aber schnell heraus, wo mein Denkfehler war und konnte das Problem in Windeseile beheben.

---

<sup>53</sup> <https://www.postgresql.org/>

## 5.3 Projekt

Um ein App zu erstellen, wird zunächst ein Projekt benötigt. Dazu wurde in den vorherigen Kapiteln auf die Varianten und die Installation eingegangen. Aus den daraus gewonnenen Erkenntnissen wurde dann das Projekt mit den benötigten Frameworks erstellt. Das Projekt wird mit MAUI gemacht, weswegen die Einrichtung hier erläutert wird. Zur Vollständigkeit des Moduls wurde auch die Einrichtung von Cordova dokumentiert. Für eine moderne Struktur wird mit dem Design-Pattern MVVM gearbeitet, welches auch beschrieben wird. Ein Projekt allein ist noch nicht sehr hilfreich, weswegen Frameworks zur Vollständigkeit benötigt werden. Ich greife hier auf mir bekannte Tools zurück, in welchen ich bereits Kenntnisse habe.

### 5.3.1 Cordova Projekt

Cordova ist in der Lage ein Projekt mit allen benötigten Dateien zu erstellen. Beim Namen muss beachtet werden, dass alles klein und ohne Sonderzeichen oder Leerzeichen geschrieben ist. Dies gilt auch für den Pfad, da es ansonsten Fehler gibt. Wie ein Projekt erstellt werden kann im folgenden Bild gesehen werden:



```

    ➤ 15:25:50 80ms
    cordova create helloworld
    >> cordova create helloworld com.example.helloworld HelloWorld
  
```

Abbildung 34 Projekt erstellen

Über «`cordova run browser`» kann das Projekt gestartet und im Browser dargestellt werden. Da alles in HTML und JavaScript geschrieben ist funktioniert auch alles im Browser. Wenn nun der Emulator verwendet werden möchte, dann kann die Plattform über «`cordova platform add android`» hinzugefügt werden.

Android Applikationen laufen im Hintergrund mit Java. Damit Android Studio läuft und Apps erstellen kann wird noch Gradle<sup>54</sup> benötigt. Damit Cordova darauf zugreifen kann muss es wieder keine Sonderzeichen im Pfad haben. Das Root-Verzeichnis eignet sich hier auch. Schlussendlich kann mit «`cordova build android`» eine App erstellt werden. Was nach der Installation gemacht werden sollte, kann [auf Moodle](#) nachgeschaut werden.

### 5.3.2 MAUI-Projekt

Um ein MAUI-Projekt zu erstellen, muss zunächst Visual Studio gestartet werden. Auf der rechten Seite befindet sich die Kategorie «Get started» ein neues Projekt erstellt werden. Mit dem Filter

---

<sup>54</sup> <https://gradle.org/>

## MemeChat

«MAUI» werden alle möglichen Projekte aufgelistet. Um HTML für das UI zu verwenden, muss das Projekt mit Blazor ausgewählt werden:

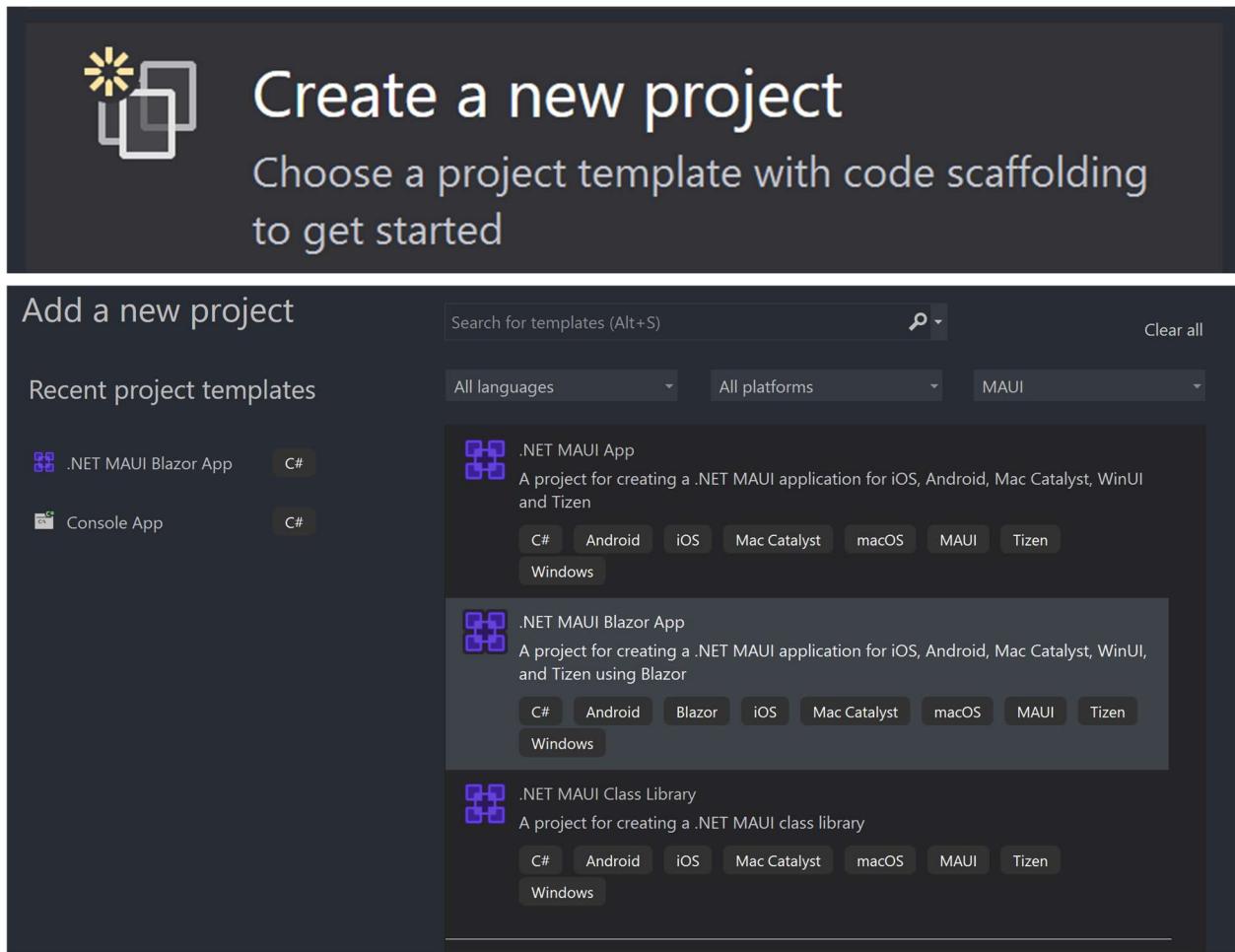


Abbildung 35 Neues Projekt erstellen

Auf der nächsten Seite muss der Projektname und Speicherort angegeben werden. Anhand der Tags sieht man auf welchen Plattformen das App laufen wird. Es macht Sinn, dass bei weiteren Projekten der Name nach einem Domainmodel, also über Punkte getrennt, geschrieben wird. Beim Namen und dem Pfad gibt es grundsätzlich nichts zu beachten.

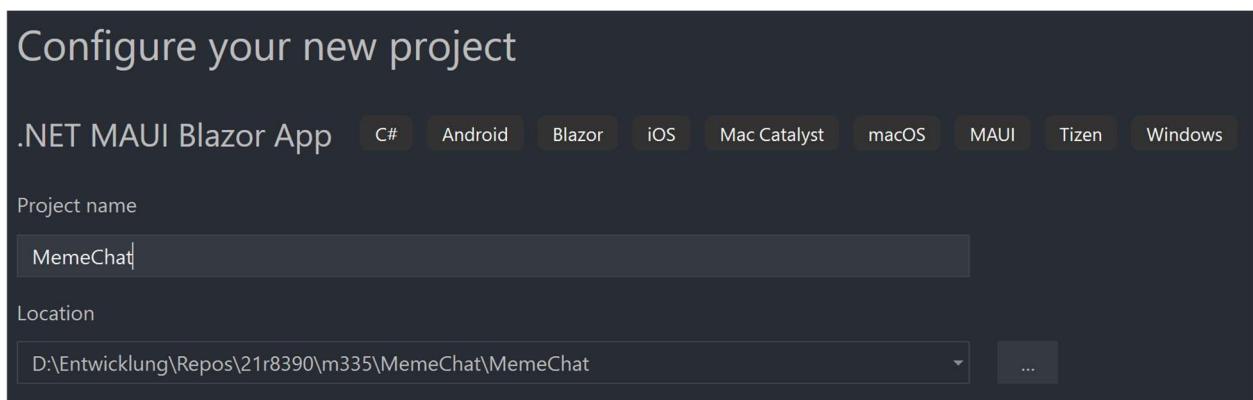


Abbildung 36 Projektname eingeben

Vor dem Erstellen der App muss noch die .NET-Version angegeben werden. Hier wird die Long-Term Support (LTS) Version empfohlen, da diese am stabilsten läuft und mit den meisten Bibliotheken zusammenarbeiten kann.

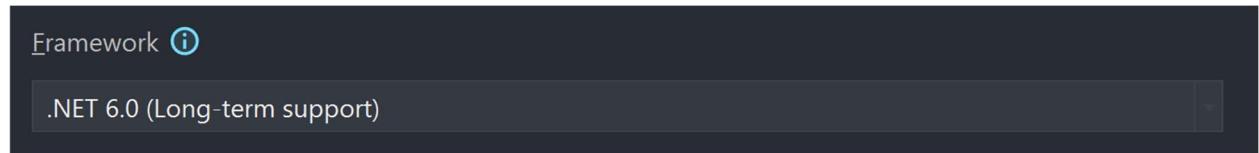


Abbildung 37 .NET-Version konfigurieren

### 5.3.3 MVVM

MVVM ist ein Design-Pattern, welches die Strukturierung von Anwendungen erleichtert. Es ist ein Ansatz, um die verschiedenen Komponenten einer Anwendung zu trennen. Das Ziel von MVVM ist es, die einzelnen Komponenten des Programms voneinander zu trennen. Dadurch ist es möglich, dass die einzelnen Komponenten unabhängig voneinander entwickelt werden können. Es ist möglich, dass die View von einem anderen Programm verwendet wird, ohne dass das Model verwendet wird. Dasselbe gilt umgekehrt für das Model, welches unabhängig sein sollte. Aus diesem Grund sind die Datenbanken und das UI in zwei unterschiedlichen Projekten.

Als Einstieg bei MVVM gilt der Controller. Dieser ist für die Verarbeitung der Anfragen zuständig. Er ruft die entsprechenden Model-Methoden auf und gibt die Daten an die View weiter. Die View ist für die Darstellung der Daten zuständig. Sie ruft die entsprechenden Controller-Methoden auf, wenn der Benutzer auf einen Button klickt.

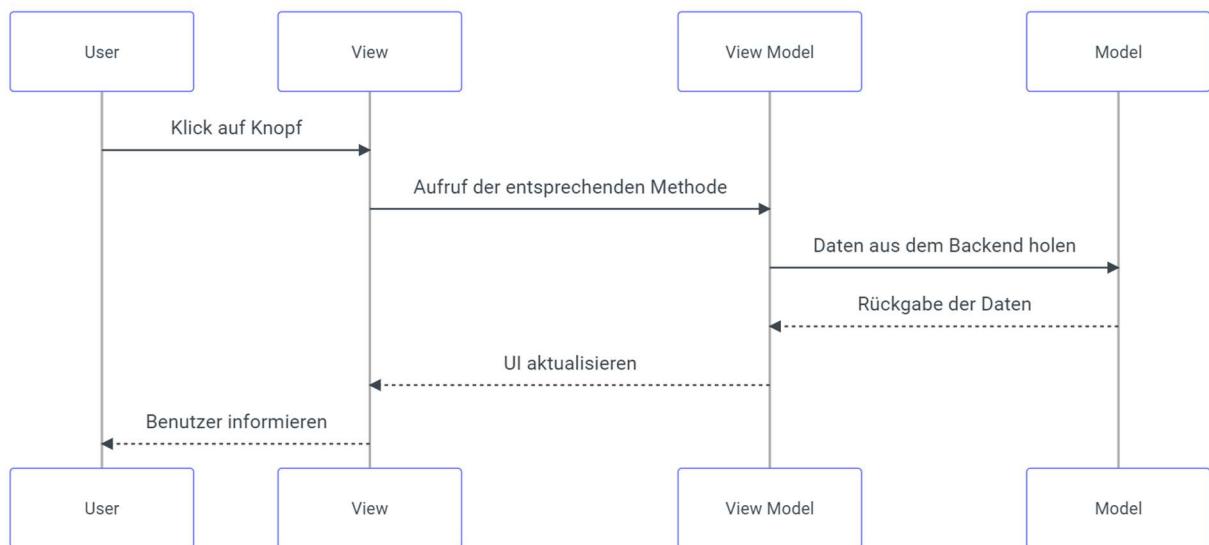


Abbildung 38 MVVM-Ablauf<sup>55</sup>

<sup>55</sup> <https://bztfinformatik.github.io/lernportfolio-21r8390-php/Appendix/DesignPatterns/MVC/>

Das Model ist die Datenhaltung. Hier werden die Daten gespeichert und verarbeitet. Es ist unabhängig von der View und dem Controller. Bei richtigem Design ist es möglich, dass das Model auch von anderen Programmen verwendet werden kann, ohne die View oder den Controller zu verwenden.

Die View ist die Benutzeroberfläche. Hier werden die Daten angezeigt und der Benutzer kann mit dem Programm interagieren. Die View sendet Events an den Controller, wenn der Benutzer auf einen Button klickt.

Der Controller ist die Verbindung zwischen Model und View. Er verarbeitet die Benutzereingaben und gibt diese an das Model weiter. Er holt sich die Daten aus dem Model und gibt sie an die View weiter.

### 5.3.4 .NET

.NET<sup>56</sup> ist eine Open-Source Entwicklungsplattform, welche von Microsoft erstellt ist. Es ist Plattformunabhängig und kann für sehr viele unterschiedliche Arten von Applikationen verwendet werden. NET ist nur ein Framework und keine Programmiersprache. Es kann in C#, F# oder Visual Basic eingebunden werden. Der häufigste Anwendungsfall sind Webserver und Desktopapplikationen, welche unter Windows laufen. Mithilfe von NuGet besteht eine Auswahl von mehr als 100'000 Bibliotheken. Bei der Installation von Visual Studio und MAUI wird NET 6, was in diesem Projekt verwendet wird, automatisch heruntergeladen.

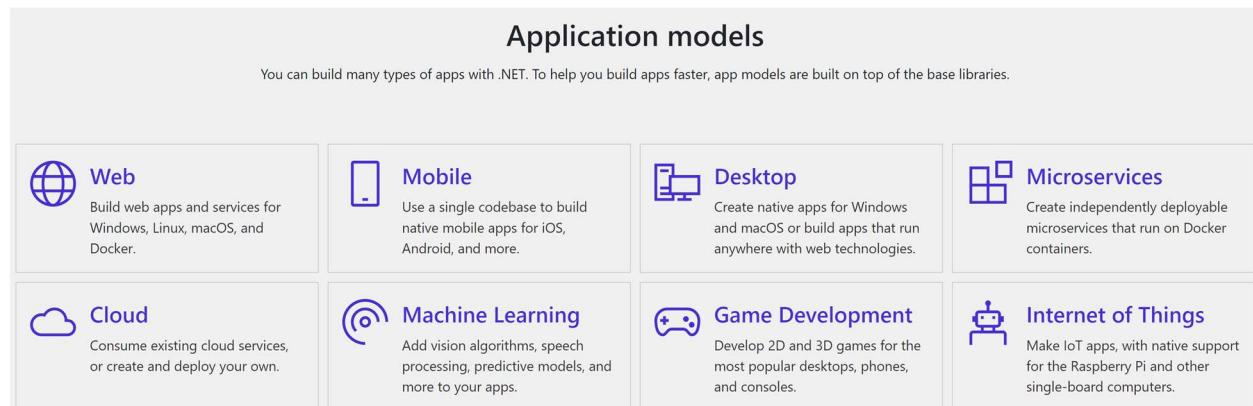


Abbildung 39 Anwendungsfälle

### 5.3.5 MudBlazor

Für eine schnellere und einfachere Entwicklung des UIs mithilfe von Blazor wird MudBlazor<sup>57</sup> verwendet. MudBlazor ist ein ambitioniertes Material-Design-Komponenten-Framework für Bla-

<sup>56</sup> <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet>

<sup>57</sup> <https://mudblazor.com/>

zor mit Schwerpunkt auf Benutzerfreundlichkeit und klarer Struktur. Es ist perfekt für .NET-Entwickler, die schnell Webanwendungen erstellen wollen, ohne sich mit CSS und Javascript herumschlagen zu müssen. Da MudBlazor vollständig in C# geschrieben ist, können Sie das Framework anpassen, korrigieren oder erweitern. Es gibt viele Beispiele<sup>58</sup> in der Dokumentation, die das Verstehen und Lernen von MudBlazor sehr einfach machen.

Der Vorteil davon ist, dass das Team dahinter sehr hilfsbereit ist und gerne auf Fragen eingeht. Zusätzlich wird es ständig weiterentwickelt und erhält so immer wieder neue Features. Mithilfe von Clean-Code und Unit Tests wird gewährleistet, dass der Code zu 90% getestet ist.

Der Hauptgrund für diesen Entscheid war, dass ich ein breit gefächertes Wissen darüber besitze, was mir Zeit und Aufwand ersparen sollte. In diversen Hobbyprojekten verwendete ich es und wurde bisher noch nie enttäuscht.

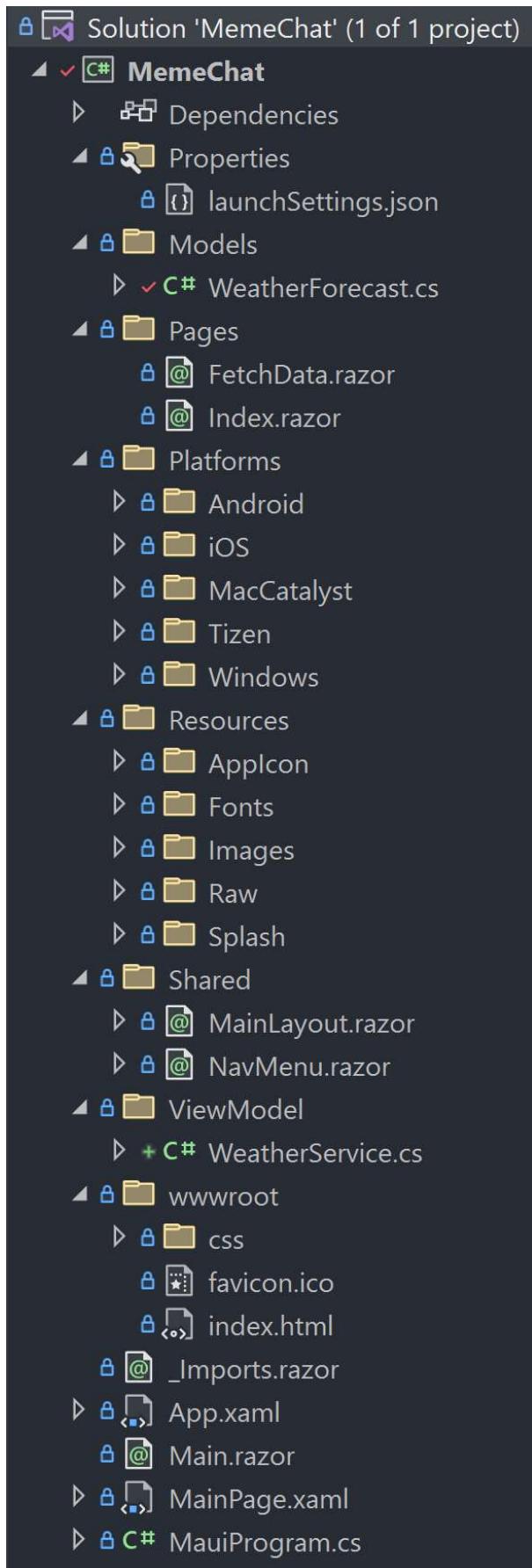
Zur Installation<sup>59</sup> muss nur das NuGet-Package «MudBlazor» zum Projekt und den globalen Imports (`_Imports.razor`) hinzugefügt werden. Damit die Komponenten richtig gestylet sind muss das CSS und JavaScript im «Index.html» eingebunden sein. Gleich wie die Datenbank gibt es Services, welche für das UI zuständig sind. Auch diese müssen hinterlegt werden. Wenn die Installation anhand der Onlineanleitung gemacht wird, dann ist es sehr einfach nachzuvollziehen.

---

<sup>58</sup> <https://github.com/MudBlazor/TryMudBlazor>

<sup>59</sup> <https://dev.mudblazor.com/getting-started/installation#manual-install-add-script-reference>

## 5.4 Ordnerstruktur



Die Ordnerstruktur einer C#-Applikation ist sehr simple und komponentenorientiert aufgebaut.

Unter den Properties werden allgemeine Einstellungen wie die Starteinstellungen gespeichert.

Das Projekt ist nach dem MVVM-Pattern aufgebaut und hat dementsprechend die Ordnerstruktur.

Die Models sind die Entitäten, welche in die Datenbank gespeichert werden. Sie sind Klassen mit nur Properties ohne wirkliche Funktionalität.

In den View Models hingegen werden die Daten aus der Datenbank gelesen und in die Models geladen. Durch diverse Methoden wird mit dem Backend gesprochen.

Die View sind im Pages Ordner und sind mit Blazor geschrieben. Sie stellen die Daten aus dem View Model in einem ansprechenden Format dar.

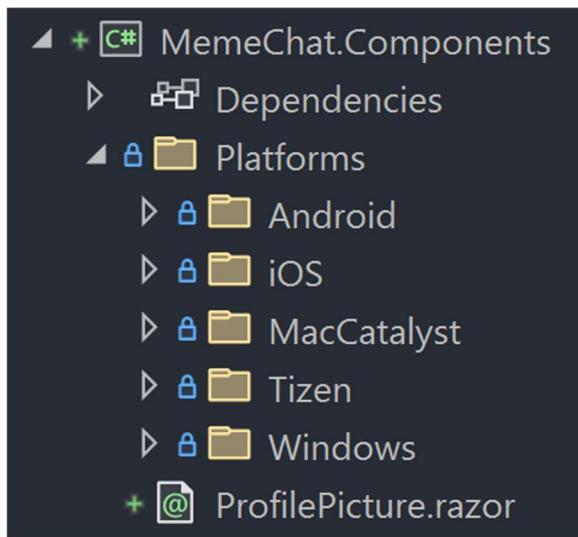
Unter den Plattformen befindet sich der Spezifische Code, welcher nur auf den spezifischen Geräten ausgeführt werden sollte. So können Features benutzt werden, die nur auf einer Plattform existieren.

Unter den Resourcen befinden sich die wichtigen Dateien für die Applikation. Dazu gehört das App Icon und die Bilder.

Das wwwroot ist ein ähnliches Verzeichnis. Es enthält jedoch die HTML spezifischen Einstellungen wie CSS und JavaScript.

Im Shared Ordner befinden sich Komponenten, welche auf jeder Seite angezeigt werden.

### 5.4.1 Komponenten



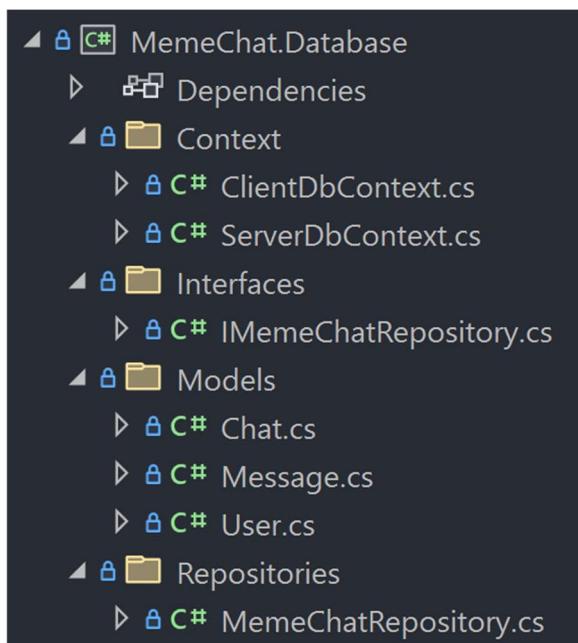
Die einzelnen Komponenten der Seiten befinden sich in einem einzelnen Projekt. Das sollte der Struktur helfen und die Übersicht verbessern. Natürlich können auch dort wieder plattformspezifische Einstellungen gemacht werden

Eine Komponente kann, wie ein HTML-Tag verwendet werden und nicht Attribute an. Sobald sich die Werte ändern wird, das UI automatisch mit den neuen Daten gerendert:

```

<FocusOnNavigate
  RouteData="@routeData"
  Selector="h1" />
  
```

### 5.4.2 Datenbank



Die Models für die Datenbank sind auch nochmals in einem einzelnen Projekt. Die Kontexte werden für die tatsächliche Verbindung mit der Datenbank verwendet. Die Applikation spricht jedoch durch das Repository indirekt damit. Das Repository ist auch zuständig, dass die Daten auf dem Server und Lokal gespeichert werden.

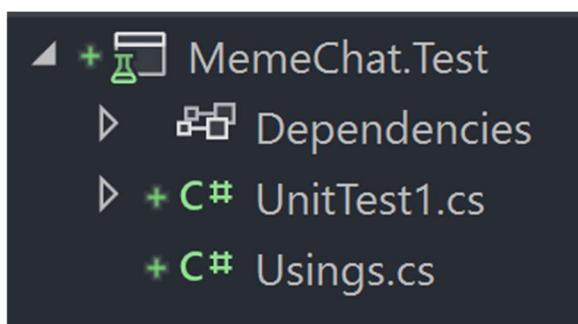
Für die Tests werden Mocks verwendet, weswegen Interface definiert wurden. Diese ermöglichen eine einfache Überschreibung der Methode.

Die Models sind die Entitäten, die auf die Datenbank übertragen werden. Die Verlinkung zwischen ihnen wird automatisch gemacht.

Die Unit Tests befinden sich isoliert in einem einzelnen Ordner, damit sie von den anderen Einstellungen nicht betroffen sind. Die Tests bilden dieselbe Struktur wie das richtige Projekt ab mit dem Unterschied, dass die Dateien mit Test enden.

Die genannten Projekte sind alles nur Bibliotheken und können somit nicht ausgeführt werden.

### 5.4.3 Tests



## 5.5 Einstellungen

In den vorherigen Kapiteln wurde über die Icons und der Name der App gesprochen. Hier wird beschrieben, wie diese Erkenntnisse in einer MAUI-Applikation umgesetzt werden. Dies ist sehr einfach, da das Ziel von NET ist, dass möglichst viele Aufgaben automatisiert oder sehr einfach sind.

### 5.5.1 Informationen

In den Projekteinstellungen, welche über **Projekt > Appname Properties** geöffnet werden können, befinden sich alle nötigen Einstellungsmöglichkeiten, wie die Möglichkeit den Namen oder die Version zu konfigurieren. Die Version wird dann auch im Store angezeigt. Eines der wichtigsten Einstellung neben dem Titel ist, dass die Applikation ID vergeben werden kann. Sie wird zur eindeutigen Identifikation einer App verwendet und muss einzigartig sein. Sie setzt sich aus der umgekehrten Anordnung der Domain zusammen.

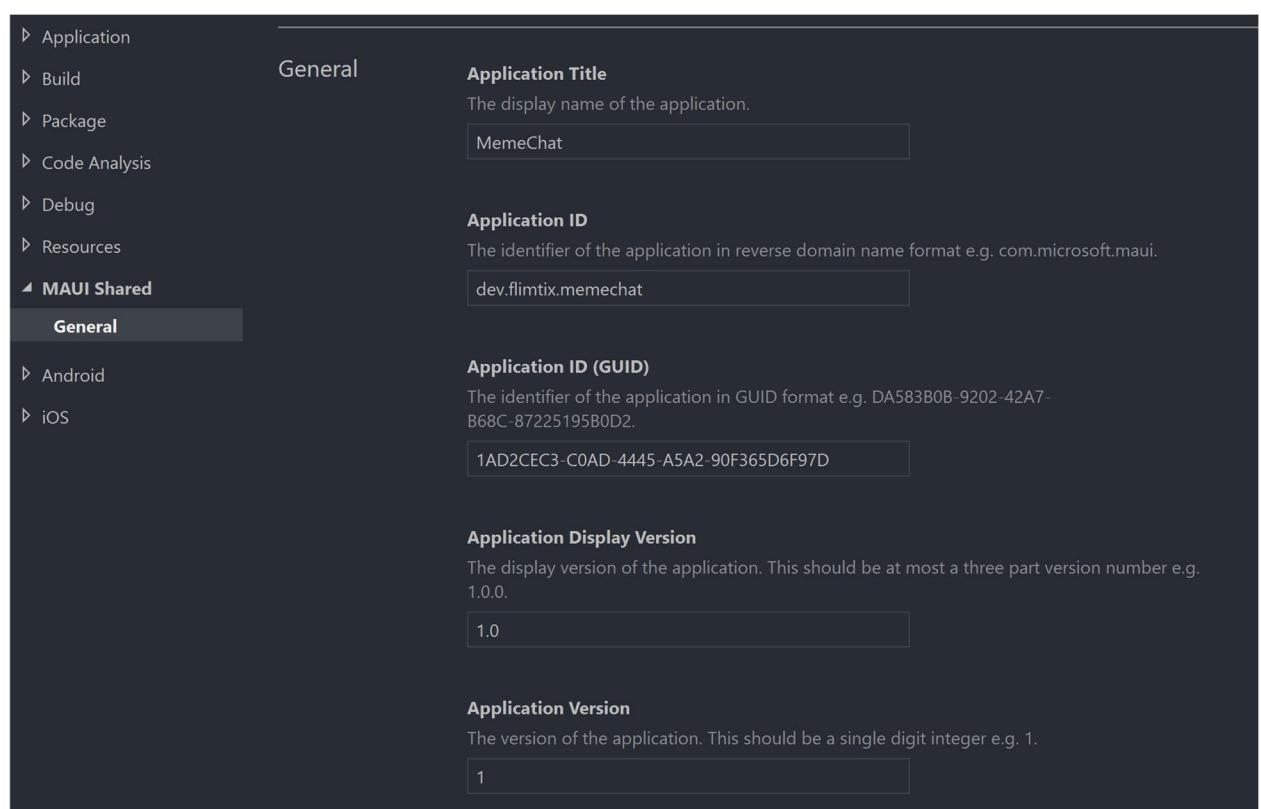


Abbildung 40 Projektproperties

Um für eine Plattform eine spezifische Einstellung zu machen, wie die Berechtigungen, gibt es im Ordner «Platforms» XML- und plist-Dateien. Diese ermöglichen das gezielte Einrichten der Informationen.

### 5.5.2 Icons & Startlogo

Das Icon und das Startlogo einer MAUI-Applikation zu ändern ist eine Leichtigkeit, da die Plattformspezifischen automatisch erstellt werden. Als Basis der Icons werden Vektorgrafiken (SVG) verwendet, da diese auf alle möglichen Größen skaliert werden können. Sobald das App veröffentlicht wird, werden die Spezifischen erstellt. Es gibt ein Vordergrund (appicon.svg), welches das Logo enthalten sollte. Der Hintergrund (appiconfg.svg) wird verwendet, um sicherzustellen, dass der ganze verfügbare Bereich mit Inhalt gefüllt ist. Dasselbe Verhalten wird auch für die Startseite verwendet, welche beim Start angezeigt wird. Dort gibt es ein Bild, welches zentriert dargestellt wird.

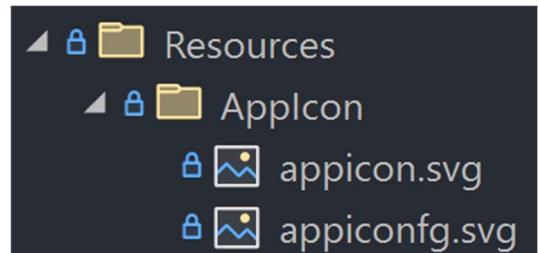


Abbildung 41 AppIcons Ordner

### 5.5.3 Same-Origin-Policy

Das Same-Origin-Policy ist ein wichtiger Sicherheitsmechanismus, der einschränkt, wie und auf welche Informationen zugegriffen werden kann. Es wird damit verlangt, dass die Ressourcen von der gleichen Domäne stammen. Es hilft potenzielle bösartige Abfragen zu isolieren und mögliche Sicherheitslücken zu reduzieren. Eine Anwendung davon ist das IFrame, welches nicht auf Daten ausserhalb seines Bereiches zugreifen darf. Der Ursprung ist gleich, wenn das Protokoll, der Port und der Host gleich sind. In C# sind grundsätzlich alle Abfragen an einen externen Server, welcher nicht dieselbe Domäne hat, verboten. Es können jedoch Cross-Origin Abfragen (CORS<sup>60</sup>) explizit erlaubt werden. In meiner eigenen Applikation wird nicht auf eine andere Domäne zugegriffen, weswegen der Codeausschnitt aus den Dokumentationen kommt:

---

<sup>60</sup> <https://learn.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-6.0>



```
C#
var MyAllowSpecificOrigins = "_myAllowSpecificOrigins";

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddCors(options =>
{
    options.AddPolicy(name: MyAllowSpecificOrigins,
                      policy =>
    {
        policy.WithOrigins("http://example.com",
                           "http://www.contoso.com");
    });
});
```

Abbildung 42 CORS einrichten

#### 5.5.4 Sicherheit

Auch wenn Same Origin Policy diverse Sicherheitsprobleme behebt, gibt es immer noch mögliche Lücken, welche beachtet werden müssen. Unter anderem ist dies Cross Site Request Forgery (CSRF) Cross-Site-Request-Forgery ist das Verfahren, um eine unbefugte Aktion auf einer anderen Seite auszuführen. Dabei wird die aktive Session eines Benutzers ausgenutzt, um ungewollte Aktionen auszuführen. Wenn CSRF verhindert werden möchte, dann muss ein One-Time-Token generiert werden. Nur wenn dieser korrekt ist, darf die Abfrage betätigt werden.

Cross-Site-Scripting ist das Verfahren, um JavaScript in eine Webseite einzubinden. Dies kann zum Beispiel dazu verwendet werden, um ein Cookie zu stehlen. Da JavaScript auf dem Client ausgeführt wird, kann der Angreifer bei alleinigem Betrachten der Webseite erfolgen. Um dies zu verhindern, muss der HTML-Code ersetzt werden.

Wenn eine Session gestartet wird, wird ein Session-ID generiert. Diese wird dann in einem Cookie gespeichert. Wenn ein Angreifer nun die Session-ID aus dem Cookie stiehlt, kann er sich als der Benutzer ausgeben. Die Auswirkung ist, dass der Angreifer Zugriff auf die vollen Daten des Benutzers hat und keine bis minimale Spuren hinterlässt.<sup>61</sup>

## 5.6 Veröffentlichung

Visual Studio bietet die Möglichkeit das App gleich über die Benutzeroberfläche zu veröffentlichen. Mithilfe eines Rechtsklicks auf das Projekt kann die Option «Publish» ausgewählt werden. Damit dies erfolgreich passiert muss auf «Release» geändert werden. Automatisch wird für die

---

<sup>61</sup> <https://bztfinformatik.github.io/lernportfolio-21r8390-php/PHP/Appendix/Sicherheit/>

## MemeChat

ausgewählte Plattform eine Veröffentlichung erstellt. Für Android wäre dies das AAB-Format, aus welchem dann die Versionen für die unterschiedlichen Plattformen erstellt werden:

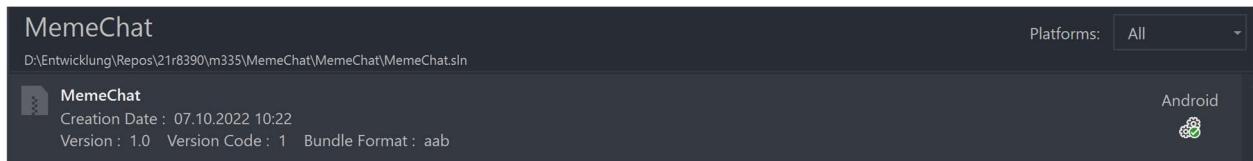


Abbildung 43 Bundle erstellen

Mithilfe eines AAB zu APK-Konvertierter<sup>62</sup> kann für das lokale Testen ein APK erstellt werden. Dabei ist wichtig zu bemerken, dass diese Version nicht debugg fähig ist. Um das App für iOS zu veröffentlichen wird ein Mac benötigt, was ich nicht zur Verfügung habe. Aus diesem Grund ist dies hier nicht aufgeführt.

### 5.6.1 App Store

Unter iOS kann ein App über den App Store veröffentlicht werden. Dazu muss ein Entwickleraccount angefordert werden, welcher jährlich 109 Franken kostet. Damit man dazu zugelassen wird muss die Seriosität und Qualität bestätigt werden. Im Gegensatz zum Play Store können nicht alle ein App veröffentlichen. Bis man angenommen wird kann es locker bis zu einem Monat dauern. Im Folgenden wird die Veröffentlichung von [pebe mobile](#) genauer beschrieben. Es ist ein App für Buchhalter, welche mit dem pebeFinance<sup>63</sup> arbeiten, damit diese bei Aussendiensten ihre Zeit und Arbeit erfassen können.

Membership details	
Entity name	pebe AG
Team ID	WX49EY4T9L
Program	Apple Developer Program
Enrolled as	Organization
Phone	41-5272301x01
Street address	Messenriet 16 Frauenfeld, Thurgau 8500 Switzerland
Account Holder	<a href="#">Manuel Schumacher</a>
Your role	Account Holder
Renewal date	July 23, 2023
Annual fee	CHF 109
Auto-renew	<input checked="" type="checkbox"/>

Sobald ein Konto erstellt und zugelassen wurde muss ein neues App eingerichtet werden. Beim Erstellen muss der Titel und die Beschreibung eingegeben werden. Zusätzlich müssen noch unzählige administrative Dateien hochgeladen und Einstellungen gemacht werden:

<sup>62</sup> <https://www.aabtoapkconverter.com/>

<sup>63</sup> <https://www.pebe.ch/de/Software/pebeFINANCE/Finanzen>

# MemeChat

## iOS App 1.4

[Save](#) [Add for Review](#)

### Version Information

German ?

The product page for this app version will be published on the App Store with the assets and metadata below.

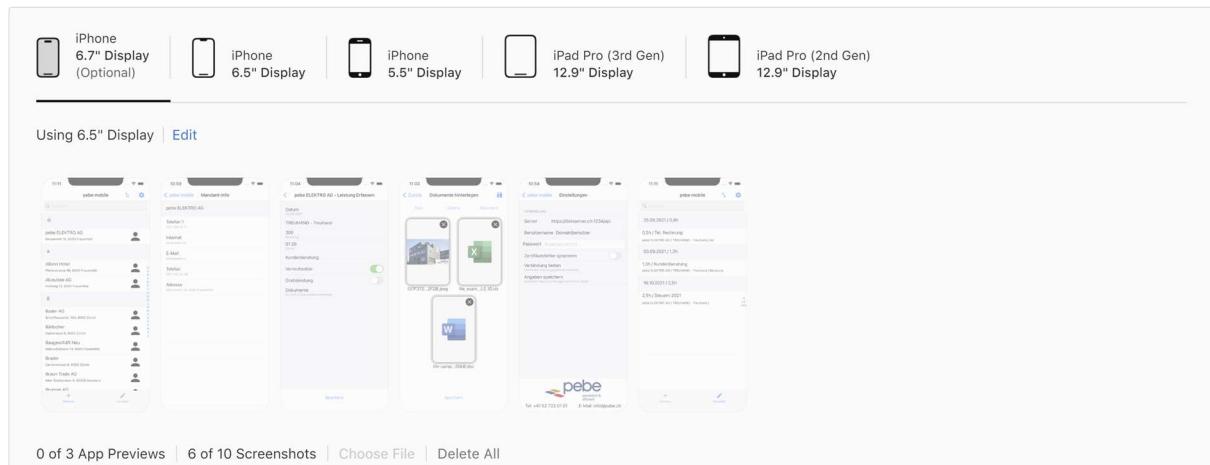


Abbildung 44 App Store Versionsinformation

Sicherheit und Datenschutz ist bei Apple ein grosses Thema, welches ernstzunehmend ist. Bevor ein App in den Store kann, muss angegeben werden, was für Daten gesammelt und wie diese verarbeitet werden. Wenn im Daten gesammelt werden, welche hier nicht aufgelistet sind, dann kann das App von Apple aus dem Store entfernt werden.

## Product Page Preview

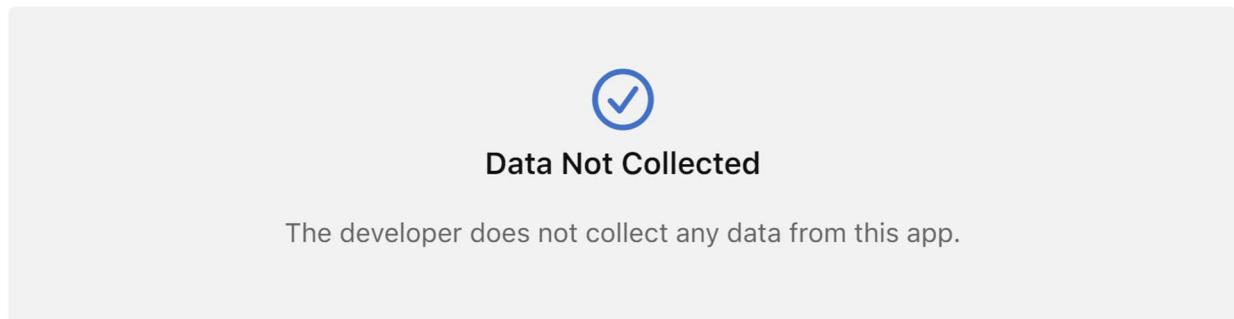


Abbildung 45 App Store Privacy

Die Entwicklung eines Apps ist mit Kosten versehen. Irgendwann möchte man einen Gewinn erzielen und die Weiterentwicklung finanzieren können. Dazu kann ein Preis oder Abo-Modell erstellt werden. Im App Store sind die In-App-Käufe auch für den Anwender vor dem Installieren ersichtlich. Seit wenigen Monaten ist es möglich, dass App Käufe ausserhalb des App Stores erlaubt sind. Zuvor war dies eine Verletzung der AGBs. Trotzdem müssen einige Prozente des Gewinns an Apple abgegeben werden.

# MemeChat

## Price Schedule +

All Prices and Currencies

PRICE ?	START DATE ?	END DATE ?
0,00 CHF (Free) <span style="float: right;">▼</span>	Other Currencies	Oct 7, 2022 No End Date

## Tax Category ? Edit

Category: App Store software

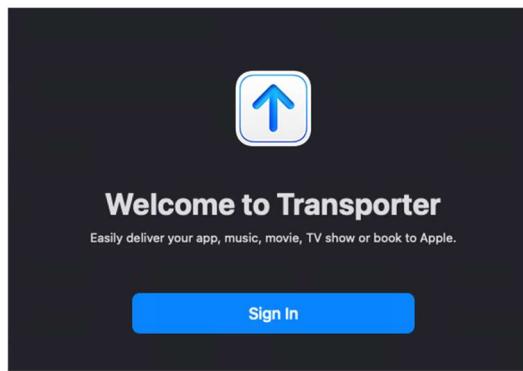
## Availability

1 of 175 countries or regions selected. [Edit](#)

Remove from sale

*Abbildung 46 App Preis*

Mithilfe des Apps «Transporter<sup>64</sup>» wird das App in den Store geladen. Wenn das App fehlerhaft oder nicht signiert ist, dann kommt eine Fehlermeldung und kann nicht hochgeladen werden. So wird schon beim Hochladen auf fehlerhafte Konfigurationen geachtet:



*Abbildung 47 Transporter für Apple*

Wenn das App von Apple angenommen wurde, dann kann es Intern über TestFlight getestet werden. Personen mit einem Apple Account können als Tester eingeladen werden. Sie bekommen eine Mitteilung, wenn eine neue Version verfügbar ist, welche sie testen müssen. Über Gruppen kann definiert werden welches Teams für das Testen zuständig ist.

<sup>64</sup> <https://apps.apple.com/us/app/transporter/id1450874784?mt=12>

The screenshot shows the pebe mobile dashboard under the 'Builds' section for iOS. It displays a table for 'Version 3.9' with one build entry: '1.3.13' (Status: Expired, Groups: AS, PE). The table includes columns for BUILD, STATUS, GROUPS, INVITES, INSTALLS, SESSIONS, CRASHES, and FEEDBACK.

BUILD	STATUS	GROUPS	INVITES	INSTALLS	SESSIONS	CRASHES	FEEDBACK
1.3.13	Expired	AS PE	4	-	-	-	-

Below this, there is a link to 'Version 3.8'.

*Abbildung 48 TestFlight testen*

Wenn alle Tester zufrieden sind, dann kann das App veröffentlicht werden. Bevor es jedoch im Store auftaucht, gibt es noch das schwierigste Hindernis. Von Apple wird automatisch getestet, ob es auf allen Geräten und bis zur angegebenen Version lauffähig ist. So wird sichergestellt, dass es nicht beim Start abstürzt oder sich anders verhält. Ein Apple Mitarbeiter wird von Hand schauen ob die Bilder mit der App übereinstimmen und kein bösartiger Code enthalten ist. Unter den Statistiken kann nach der Veröffentlichung das Verhalten und die Bewertung der Kunden eingesehen werden:

NAME	IMPRESSIONS	UNITS	PROCEEDS	SESSIONS	CRASHES
pebe mobile iOS	12040 +135%	4343 +300%	0	98437 +13%	1 0%

*Abbildung 49 pebe mobile Analytics*

## 5.6.2 Signieren

Wenn das App in den Google Play Store veröffentlicht werden sollte, dann muss es signiert werden. Dies ist mit Visual Studio sehr einfach und praktisch ohne Probleme zu machen. Es muss nur den Google Store beim Veröffentlichen ausgewählt werden. Als nächstes kann mein einen neuen Key-store oder einen bestehenden auswählen. Im Popup müssen alle Felder ausgefüllt werden. Wenn alles funktioniert hat, dann wird es nochmals neu kompiliert und signiert. Die Datei wird dann auch von Store angenommen.

## MemeChat

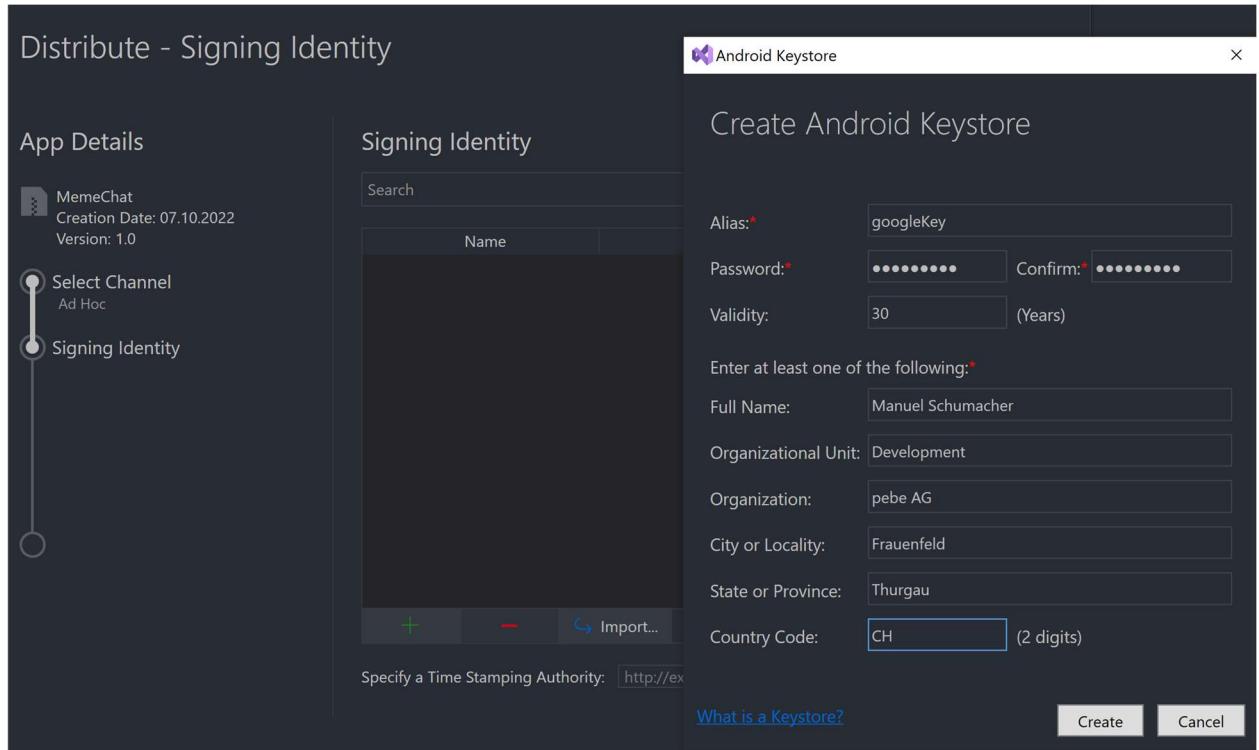


Abbildung 50 Android Signieren

## 6 Schlusswort

Das Projekt ist an sich beendet und erfüllt alle gewollten Ziele. Vollständig beendet ist ein Projekt nie, da es immer die Option für Verbesserungen oder neue Funktionen gibt. Mit dem erreichten Ergebnis bin ich sehr zufrieden und habe sogar meine Erwartungen etwas übertrffen. An einen Kunden würde ich es trotzdem nicht geben, da es noch viel mehr getestet und optimiert werden müsste.

Als Erweiterung könnte man Memes bereits zur Verfügung stellen, welche dann nicht aus einer Liste auserwählt werden. Im Moment sind Chats nur mit einem Nutzer möglich. Es so umzubauen, dass ein Gruppenchat möglich ist, wäre von der Datenstruktur kein grosser Aufwand. Ein weiterer Punkt wäre, dass Nachrichten auch offline versendet werden können.

In der App gibt es keine Lösch oder Bearbeitungsfunktion, da es nicht zum Kontext passt. Nach mehrfachen Nachfragen wurde mir versichert, dass das App lauffähig ist

Aus diesem ÜK habe ich sehr viel über die Entwicklung und Planung eines Apps gelernt. Ich weiss nun auf was ich beim Veröffentlichen beachten muss und wie dies gemacht wird. Zudem habe ich gelernt, wie ein App aufgebaut ist und was bei der Usability beachtet werden muss. Die Idee eine Webseite auf ein Mobilegerät anzuzeigen und so Plattformunabhängig zu entwickeln, finde ich genial. Das im Kurs nur mit Cordova gearbeitet wird finde ich etwas schade. Das Einrichten dauerte mir viel zu lange und war bei fast allen mit Fehlern verbunden. Ein fertiges System in Form einer Virtuellen Maschine wäre von Vorteil gewesen, da so ein Tag weniger für das Einrichten von Nöten gewesen wäre. Dieser ÜK wäre besser als Modul geeignet, in welchem mehrere Wochen zur Verfügung stehen, damit mehr auf die Codequalität und das Verständnis geachtet wird.

Dies ist eine Dokumentation von:

Manuel Schumacher

Mail. : [Manuel.schumacher@pebe.ch](mailto:Manuel.schumacher@pebe.ch)

Tel. : 076 458 36 66