

HTBLuVA Graz Gösting
Ibererstraße 15-21
8051 Graz

ABTEILUNG FÜR ELEKTRONIK & TECHNISCHE
INFORMATIK

E X C E

Excerpt

DIPLOMARBEIT

Moritz PAIERL
5BHEL

BETREUER: PROF. DI Michael ZELLE

1 Eidestattliche Erklärung

Ich erkläre eidesstattlich, dass ich die Arbeit selbständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und alle aus ungedruckten Quellen, gedruckter Literatur oder aus dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte gemäß den Richtlinien wissenschaftlicher Arbeiten zitiert, durch Fußnoten gekennzeichnet beziehungsweise mit genauer Quellenangabe kenntlich gemacht habe.

_____, am _____
Ort, Datum

Moritz Paierl

2 Gender Erklärung

Aus Gründen der besseren Lesbarkeit wird in folgender Arbeit die Sprachform des generischen Maskulinums angewendet. Es wird darauf hingewiesen, dass die ausschließliche Verwendung des generischen Maskulinums als geschlechtsunabhängig verstanden werden soll.

_____, am _____

Ort, Datum

Moritz Paierl

3 Inhaltsverzeichnis

| | |
|-----------------------------------------------|----|
| 1 Eidesstattliche Erklärung..... | 2 |
| 2 Gender Erklärung..... | 3 |
| 3 Inhaltsverzeichnis | 4 |
| 4 Kurzfassung – Abstract..... | 6 |
| 4.1 Deutsch | 6 |
| 4.2 English | 6 |
| 5 Projektteam | 7 |
| 6 Danksagungen | 8 |
| 7. Technologie & Methodik | 9 |
| 7.1 Agile Software Entwicklungs-Methodik..... | 9 |
| 7.2 Eingesetzte Technologie | 9 |
| 8 Infrastruktur | 12 |
| 8.1 GitHub & Git | 12 |
| 8.1.1 Git vs. GitHub | 12 |
| 8.1.2 Git Befehle..... | 12 |
| 8.1.3 Erstellen des Repo | 14 |
| 8.1.4 Generieren des SSH-Keys | 16 |
| 8.1.5 SSH-Key in GitHub hinzufügen | 17 |
| 8.1.6 Repository klonen | 19 |
| 8.2 Aufsetzen des Servers..... | 20 |
| 8.2.1 Wahl des Servers..... | 20 |
| 8.2.2 Aufsetzten der EC2 Instanz | 21 |
| 8.2.1 Aufsetzen des LAMP Servers..... | 22 |
| 8.2.2 Testen des LAMP Servers | 24 |
| 8.3 Aufsetzen des Simulators | 25 |
| 9 Frontend | 28 |
| 9.1 Aufsetzen des Frontends | 28 |
| 9.2 Zusammenspiel der Pages | 30 |
| 9.3 Flutter Basics | 32 |
| 9.3.1 Bausteine des Frontends:..... | 32 |
| 9.4 StartUpPage: | 35 |

| | |
|------------------------------------|----|
| 9.5 Login & Registration Page..... | 36 |
| 9.5.1 LoginPage..... | 36 |
| 9.5.2 RegistrationPage..... | 38 |
| 9.6 MainPage und Pagehandler | 40 |
| 9.7 ProfilePage | 43 |
| 9.8 MapPage | 45 |
| 9.9 UploadPage | 48 |
| 10 Backend | 52 |
| 10.1 Allgemeines | 52 |
| 10.1.1 Tokens | 52 |
| 10.1.2 PHP vs Node..... | 53 |
| 10.2 SQL Datenbank | 54 |
| 10.3 REST API..... | 59 |
| 10.4 Routen..... | 64 |
| 10.4.1 Register Route: | 64 |
| 10.4.2 Login Route: | 66 |
| 10.4.3 Logoff Route: | 66 |
| 10.4.4 Upload Picture Route: | 67 |
| 11 Fazit..... | 69 |
| 12 Verzeichnisse | 70 |
| 12.1 Quellenverzeichnis..... | 70 |
| 12.1.1 Kapitel 8:..... | 70 |
| 12.1.2 Kapitel 9 | 71 |
| 12.2 Bilderverzeichnis | 72 |

4 Kurzfassung – Abstract

4.1 Deutsch

Die Vorliegende Diplomarbeit setzt sich mit der Entwicklung eines POCs (prove of concepts) für eine Social Media App für Android sowie iOS auseinander. Da eine vollständige App den Rahmen dieser Diplomarbeit sprengen würde, wird in dieser Arbeit der Fokus auf die ausgewählten Technologien, Programmiersprachen und ein solides und effizientes Fundament der App gelegt. Um eine vollständige Social Media App zu entwickeln, wäre ein Softwareteam notwendig, doch die Grundfunktionen wie Login und Registrierung, Aufnehmen von Fotos und Speichern der aufgenommenen Fotos sowie Coordinateen werden in dieser Diplomarbeit demonstriert.

4.2 English

The aim of this diploma thesis is to develop a POC (prove of concept) for a Social Media App supporting both, android, and iOS devices. Since the complexity of this platform would exceed the scope of this work by far, the focus is on the prove of concept. The diploma thesis describes the technologies, programming languages, frameworks and the setup of a foundation for the platform. To fully implement this platform a team of developers would be required, but the core functionality like registering, signing in, taking pictures with geo coordinates and storing them should be demonstrated.

5 Projektteam

Betreuungslehrer:

Prof. DI Michael ZELLE

Projektteam:

Moritz PAIERL

Projektleiter

Software - DevOps

Software – Frontend

Software – Backend

Software – Database

6 Danksagungen

Ein besonderer Dank gilt Herrn Professor Dipl.-Ing. Michael Zelle für seine Unterstützung. Weiters möchte ich meinem Vater Dipl.-Ing. Stephan Zimmermann für diverse Hilfestellungen beim Entwickeln des Backend, meiner Mutter Mag. Pia Paierl sowie meiner Freundin Emilia Lorenz für die vielen Ratschläge bezüglich des Verfassens der Arbeit und die aufbauenden Worte danken.

7. Technologie & Methodik

Bei einem Softwareprojekt dieser Größenordnung sind die Auswahl der Technologie und die Methodik der Entwicklung von entscheidender Bedeutung. Obwohl die Arbeit als Einzelarbeit durchgeführt wird, soll der Entwicklungsprozess so aufgesetzt werden, dass er auch in einem Team funktionieren würde.

7.1 Agile Software Entwicklungs-Methodik

Es wird agil entwickelt, das heißt, es werden kurze Sprints mit klar definierten Zielen abgearbeitet. Jeder Sprint enthält mehrere Einzelaufgaben. Jede dieser Einzelaufgaben (TASKS) wird auf einem eigenen Branch in Git (Mehr zu Git in [Kapitel 8.1 GitHub und Git](#)) erarbeitet. Nachdem die Task implementiert wurde, wird ein PR (Pull Request) erstellt. Dieser kann von einem anderen Entwickler überprüft und freigegeben werden. Die Code Reviews dienen dazu, Schlampigkeitsfehler zu entdecken und auf Code Smells („übelriechender Code“) hinzuweisen. Die aus dem Stand des Pull Requests gebaute Software wird getestet. Funktioniert die Software und wurden die Ziele des PR korrekt implementiert, wird der PR auf dem Main Branch gemerged. Diese Methodik mag kompliziert klingen, sie führt aber in der Praxis dazu, dass man einen fehlerfreien Main Stand hat und dass klar nachvollziehbar bleibt, welche Funktionalität in welchen Branch implementiert wurde. In einem Softwareprojekt würde man die Tickets noch in einer Software wie JIRA abbilden und ein Projektmanager wäre zuständig für die Verwaltung, Aktualisierung und fristgerechte Abarbeitung der Tickets.

7.2 Eingesetzte Technologie

Folgende Technologien werden eingesetzt. Für die Implementierung des Backend wird eine AWS Micro Instanz mit Amazon Linux 2023 aufgesetzt.

Es wird ein sogenannter LAMP Server aufgesetzt. LAMP steht für Linux, Apache, MySQL, PHP. Mit einem LAMP Server hat man einen Apache Server, der http Requests entgegen nimmt. Dieser in Kombination mit PHP ermöglicht es, ein einfaches REST API zu implementieren.

REST-API steht für „Representational State Transfer - Application Programming Interface“. Dieses macht den Austausch von Informationen möglich, wenn diese sich auf unterschiedlichen Systemen befinden. Der Vorteil eines REST APIs besteht vor allem darin, dass keine dauerhafte Verbindung zwischen Client und Backend notwendig ist, sondern bei Bedarf einzelne Requests geschickt werden können.

Das REST API ist die Grundlage des Backend. Sie ermöglicht den Clients sich zu registrieren, anzumelden und Daten abzufragen bzw. Daten zu speichern.

Die SQL Datenbank ist eine relationale Datenbank. Sie wird verwendet, um die Benutzer-Daten und die Bildinformation abzuspeichern. Die Bilder selbst werden aber aus Performance-Gründen nicht in der Datenbank abgespeichert, sondern man benötigt dafür ein CDN (Content Delivery Network). Amazon stellt dafür mit S3 Buckets einen hervorragenden Dienst zur Verfügung.

Speichert ein Benutzer ein Bild, so schickt er das Bild base64 codiert an das Backend. Das Backend speichert das Bild im S3 und bekommt einen eindeutigen Namen und den Download Link. Dieser Name und der Link werden jetzt in der SQL Datenbank gespeichert, gemeinsam mit der Benutzer ID. Die ID des Datensatzes wird zurückgegeben.

Möchte man nun ein Bild laden, so sucht man den Datensatz mit der Bild ID und gibt den Bild Link zurück. Das Bild kann dann vom Client direkt aus dem CDN geladen werden, ohne das Backend zu belasten. Auf diese

Weise wird die Verantwortung für die Skalierung an AWS abgegeben und Amazon ist eindeutig in der Lage, die Skalierbarkeit garantieren zu können.

Für die Implementierung der Clients wurde das sehr populäre Flutter Framework ausgewählt. Diese ermöglicht einem Code, den man einmal schreibt, für mehrere Plattformen zu deployen. Man kann also native iOS und Android Anwendungen generieren lassen und erspart sich so den Aufwand, jeden Client extra implementieren zu müssen.

Flutter verwendet die Programmiersprache Dart, welche modern und objektorientiert ist und eine einfache Syntax hat.

In dem Widget basierten UI Interface von Flutter wird ein User Interface durch einen Baum von Widgets dargestellt.

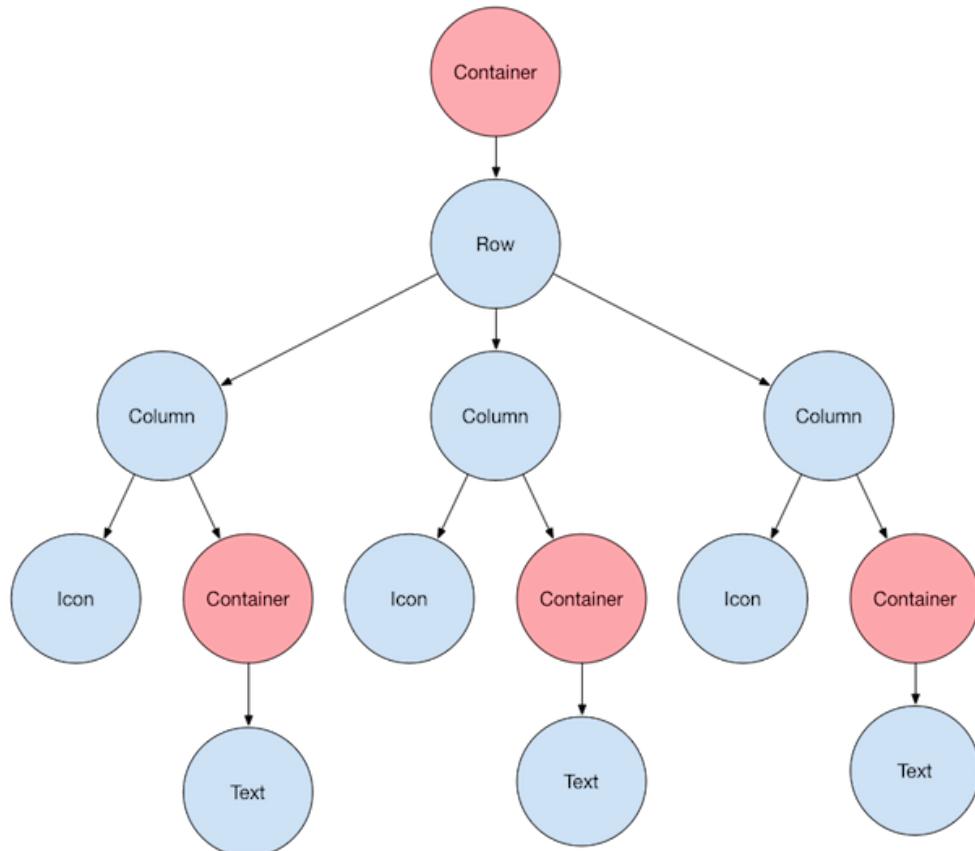


Abbildung 7.1

8 Infrastruktur

Im folgenden Kapitel geht es um die Infrastruktur, die für das Projekt geschaffen werden muss.

8.1 GitHub & Git

8.1.1 Git vs. GitHub

Git ist eine Versionskontrollsoftware und ermöglicht es den Usern, ihren Sourcecode zu versionieren und ältere Stände des Codes wieder abzurufen.

GitHub hingegen ist eine Webanwendung, welche alle Funktionen von Git beinhaltet und es mehreren Personen ermöglicht, gleichzeitig an einem Projekt beziehungsweise Code zu arbeiten. Software wie GitHub oder auch GitLab ermöglichen es, kollaborativ an der Software zu arbeiten, ist state of the art und wird in vielen Softwareunternehmen verwendet.

8.1.2 Git Befehle

Im folgenden Kapitel sind die wichtigsten Git Befehle aufgelistet, um mit Git sowie GitHub arbeiten zu können.

git add:

„git add“ wird verwendet, um alle Änderungen zu stagern und für einen Commit (eine neue Version) vorzubereiten.

git commit:

Übergibt alle auf der Stage liegenden Änderungen und trägt sie als neue Version in den Verlauf des Projektes.

git branch:

Mit „git branch“ kann ein neuer Branch abgespaltet werden. Es ist üblich, einen Development Branch für jeden PR zu haben. Auf den Development Branches wird das Projekt weiterentwickelt und auf dem Master oder Main

Branch befindet sich die aktuelle funktionierende Version. Es ist üblich, den Namen Main für den Haupt Branch zu wählen, da das Wort Master von manchen Personen mit Unterdrückung in Verbindung gebracht werden kann.

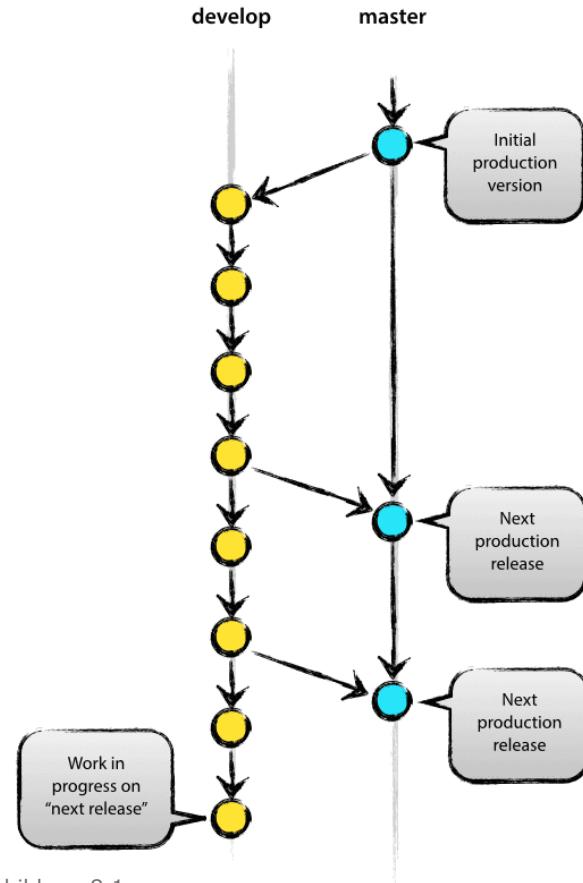


Abbildung 8.1

git merge:

Mit „git merge“ werden zwei Branches zusammengeführt. Dafür wird auf den Branch, auf den gemerged wird, mit „git checkout“ navigiert und dann mit „git merge <Name>“ gemerged. Als Name wird der zu mergende Branch angegeben.

git checkout <Name>:

Mit „git checkout“ kann der Branch gewechselt werden. Wird zum Beispiel der Befehl „git checkout main“ ausgeführt, wird auf den Main Branch navigiert.

git push:

„git push“ lädt den Inhalt eines lokalen Repository auf ein remote Repository.

8.1.3 Erstellen des Repo

Um ein neues Repository zu erstellen, wird zuerst das Plus in der rechten oberen Ecke der GitHub Webanwendung gedrückt.

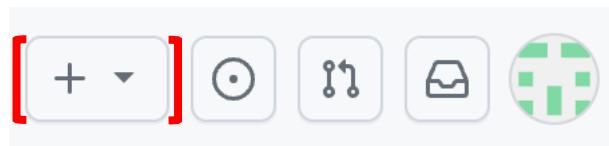
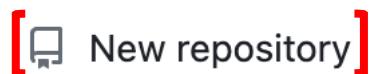


Abbildung 8.2

Danach kann unter mehreren Möglichkeiten die Option „New repository“ ausgewählt werden, um ein neues Repository anzulegen.



Import repository

New codespace

New gist

New organization

New project

Abbildung 8.3

Im letzten Schritt wird dem Repository ein Name und eine Beschreibung hinzugefügt. Namen für Repositories sollten kurz und leicht memorisierbar sein. Die Beschreibung ist im Gegensatz zum Namen ein optionales Feld und muss nicht befüllt werden. Wird die Beschreibung allerdings befüllt, sollte sie nicht länger als ein bis zwei Sätze sein. Das Repository kann auf

Public oder Private gestellt werden. Soll jeder GitHub Benutzer Einsicht in das Repository haben, wird die Option Public gewählt. Auch wenn das Repository Public ist, können nur ausgewählte Personen zum Repository commiten. Wird Private gewählt, entscheidet der Repository Admin, wer Einsicht hat und commiten kann. Wenn eine längere Beschreibung des Projektes nötig ist, kann ein README File hinzugefügt werden. Final können noch ein gitignore File und eine Lizenz hinzugefügt werden. Das gitignore bestimmt, welche Files nicht von git getracked werden sollen. Eine Lizenz sagt anderen Usern, was sie mit dem Code machen dürfen und was nicht.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Owner * Repository name *



mopaierl



Great repository names are short and memorable. Need inspiration? How about **supreme-octo-spoon** ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore



Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license



A license tells others what they can and can't do with your code. [Learn more about licenses](#).

You are creating a public repository in your personal account.

Create repository

Abbildung 8.4

8.1.4 Generieren des SSH-Keys

Nachdem das Repository in GitHub erstellt worden ist, verwendet man einen SSH-Key, um das Repository klonen zu können. Zum Erstellen des SSH-Keys wird ein Terminal geöffnet und folgender Befehl eingegeben:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Das Terminal wird folgende Zeile ausgeben, welche einen Speicherort für den SSH-Key abfragt.

```
> Enter a file in which to save the key  
(/Users/YOU/.ssh/id_ALGORITHM):
```

Jetzt wird ein Speicherort und der Name für den File, welcher den SSH Key beinhaltet, angegeben. Als Beispiel würde die Eingabe `/Users/maxMustermann/Testkey` den Key im Benutzerverzeichnis `maxMustermann` unter dem Namen `Testkey` speichern. Wird dieser Schritt übersprungen, wird der Standard File Speicherort und ein automatisch generierter Name zum Speichern der Datei verwendet. Als Nächstes werden folgende zwei Zeilen im Terminal angezeigt, welche die Möglichkeit bieten, eine Passphrase anzugeben.

```
> Enter passphrase (empty for no passphrase):  
> Enter same passphrase again:
```

Dieser Schritt ist optional und kann mit zweimaliger Eingabe von Enter übersprungen werden.

8.1.5 SSH-Key in GitHub hinzufügen

Um den SSH-Key in GitHub hinzuzufügen, wird zuerst der Inhalt des generierten Schlüssel Files kopiert. Danach wird in GitHub in der rechten oberen Ecke auf das Profilbild gedrückt.

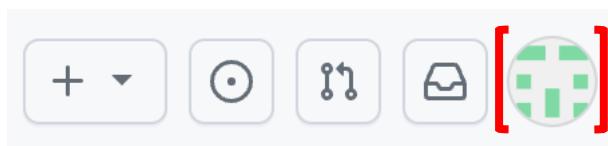


Abbildung 8.5

Anschließend wird zu den Settings navigiert.

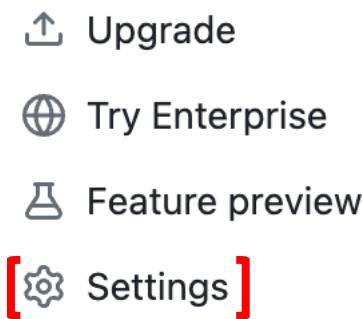


Abbildung 8.6

In den Settings kann auf der linken Seite im Raster Access unter SSH- und GPG-Keys alles rund um SSH- und GPG-Keys gefunden werden.

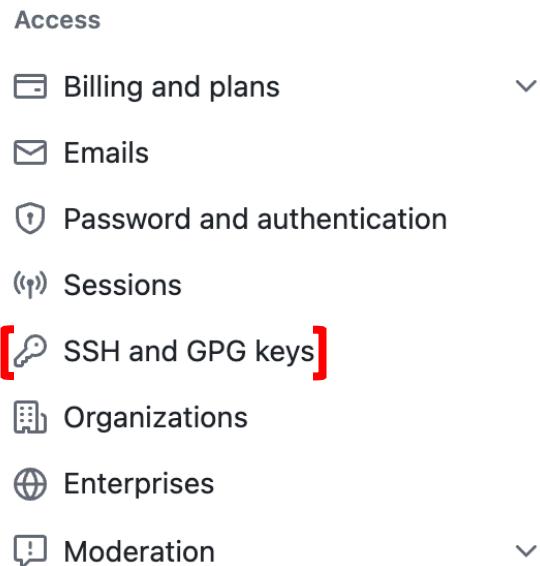


Abbildung 8.7

Dann wird „new SSH key“ gedrückt, um einen neuen Schlüssel hinzuzufügen.



Abbildung 8.8

Der Inhalt des in Kapitel 8.1.4 generierten SSH-Keys wird in das Fenster „Key“ oder „Schlüssel“ eingefügt und der Key wird betitelt. Schlussendlich wird mit „Add SSH Key“ der Key gespeichert.

Add new SSH Key

Title

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

[Add SSH key]

Abbildung 8.9

8.1.6 Repository klonen

Wird das Repository geklont, wird eine vollständige Kopie vom Repository von Github.com auf den lokalen Speicher des Computers gelegt. Das Klonen hilft, das Hinzufügen und Entfernen von Code und das Durchführen von großen Commits zu vereinfachen. Um ein Repository zu klonen, wird zuerst in das zu klonende Repository navigiert. Danach wird auf den Code Button gedrückt.

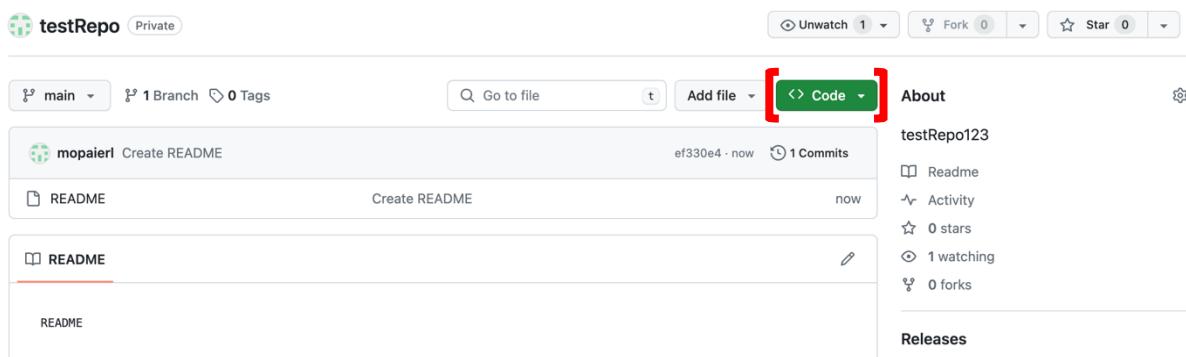


Abbildung 8.10

Im Fenster, welches anschließend aufgeht, wird der SSH Link kopiert.

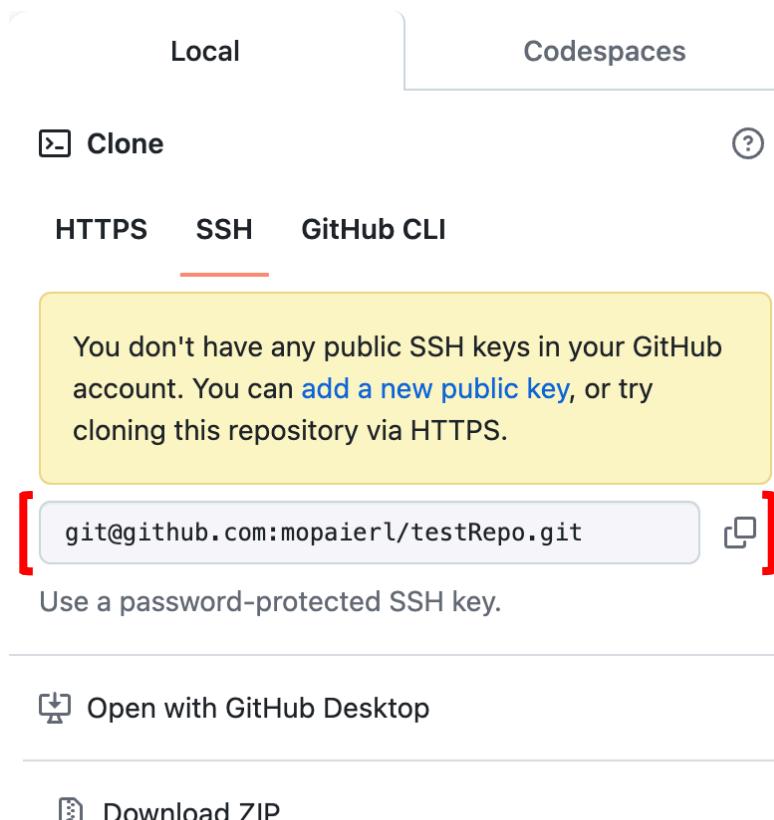


Abbildung 8.11

Schlussendlich kann im Zielordner ein Terminal geöffnet und mit folgendem Befehl das Repository geklont werden.

```
git clone git@github.com:mopaierl/testRepo.git
```

8.2 Aufsetzen des Servers

8.2.1 Wahl des Servers

Als Server wird eine EC2 Instanz vom Anbieter Amazon Web Services (kurz AWS) verwendet. Diese ist von überall erreichbar, was für das Entwickeln einer serverbasierten App notwendig ist. Ein weiterer Vorteil der EC2 Instanz ist die ausführliche Dokumentation, welche es ermöglicht, eine EC2 Instanz schnell und effektiv aufzusetzen.

8.2.2 Aufsetzen der EC2 Instanz

Um eine EC2 Instanz aufzusetzen, ist es zuerst notwendig, eine Region auszuwählen, in der sich die Instanz befinden soll. Es wurde die Region Frankfurt gewählt, damit die europäischen Datenschutzrichtlinien erfüllt sind. Auf der Homepage der AWS Konsole kann zum EC2 Dashboard navigiert werden, indem in der Suchleiste nach EC2 gesucht wird.



Abbildung 8.12

Auf dem EC2 Dashboard navigiert man zum „Instanzen“ Raster, um eine neue Instanz aufzusetzen zu können.

▼ Instances

[Instances]

Instance-Typen

Startvorlagen

Spot-Anfragen

Savings Plans

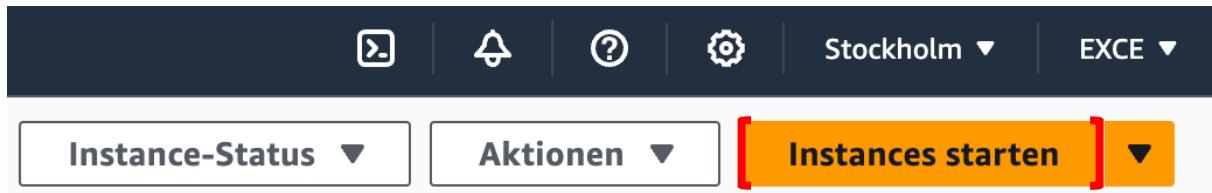
Reserved Instances

Dedicated Hosts

Kapazitätsreservierungen

Abbildung 8.13

Um die Instanz zu erstellen, muss der „Instanz Starten“ Button gedrückt werden.



Damit die Instanz läuft, muss ein Name für die Instanz eingegeben und danach ein RSA (.pem File zur Verwendung mit OpenSSH) Schlüsselpaar erstellt und/oder eingetragen werden.

A screenshot of the AWS Lambda instance creation form. It starts with a section titled 'Name und Tags' with an 'Info' link. A 'Name' field contains the placeholder 'z. B. Mein Webserver' and a 'Weitere Tags hinzufügen' link. Below this is a section titled '▼ Schlüsselpaar (Anmeldung)' with an 'Info' link. It contains a note: 'Sie können ein Schlüsselpaar verwenden, um eine sichere Verbindung zu Ihrer Instance herzustellen. Stellen Sie sicher, dass Sie Zugriff auf das ausgewählte Schlüsselpaar haben, bevor Sie die Instance starten.' Underneath is a 'Schlüsselpaarname - Pflichtfeld' section with a dropdown menu showing 'Auswählen' and a 'Neues Schlüsselpaar erstellen' button with a plus sign icon.

Abbildung 8.15

8.2.1 Aufsetzen des LAMP Servers

Zum Aufsetzen des LAMP Servers werden als Erstes alle Pakete mit folgendem Befehl auf den neuesten Stand gebracht:

```
sudo dnf update -y
```

Danach werden mit folgendem Befehl die aktuellen Versionen von Apache und PHP für Amazon Linux 2023 heruntergeladen:

```
sudo dnf install -y httpd wget php-fpm php-mysqli php-json php  
php-devel
```

Um MariaDB herunterzuladen, wird folgender Befehl ins Terminal eingegeben:

```
sudo dnf install mariadb105-server
```

Um die aktuelle Version eines Paketes zu überprüfen, wird folgender Befehl ins Terminal eingegeben:

```
sudo dnf info package_name
```

Um den Apache Server zu starten, wird folgender Befehl ins Terminal eingegeben:

```
sudo systemctl start httpd
```

Um den Apache Webserver so zu konfigurieren, dass er bei jedem System Start auch hochfahren wird, wird folgender Befehl ins Terminal eingegeben:

```
sudo systemctl enable httpd
```

Um zu verifizieren, dass httpd angeschaltet ist, wird folgender Befehl ins Terminal eingegeben:

```
sudo systemctl is-enabled httpd
```

Als Nächstes muss das http Protokoll zu unseren Sicherheitsregeln hinzugefügt werden. Dafür muss in der Amazon Web Services Konsole auf dem EC2 Dashboard auf Instanzen und dann auf unsere Instanz geklickt werden. Danach wird unter dem Raster Sicherheit auf die aktuelle Sicherheitsgruppe geklickt. Das Menü, welches nun aufgeht, bietet die Möglichkeit, eine neue Sicherheitsregel hinzuzufügen. Unsere neue Regel ist vom Typen http, das Protokoll ist TCP und der Port ist 80. Die Quelle ist Benutzer definiert.

Der Server wird getestet, indem `http://{IP Adresse}/` in einen Browser eingegeben wird. Wichtig ist es, dass nicht die sichere Version von HTTP, also HTTPS, verwendet wird, da sonst eine Fehlermeldung aufscheinen wird, da unser Server nur für HTTP konfiguriert ist.

8.2.2 Testen des LAMP Servers

Um den Server zu testen, erstellen wir ein `phpinfo.php` File mit folgendem Befehl:

```
echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
```

Wird nun in einem Webbrowser folgender Link eingegeben, wird ein Informationsfenster mit Informationen rund um PHP aufgehen.

<http://{ip Adresse der EC2 Instanz}/phpinfo.php>



| | |
|------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| System | Linux ip-172-31-16-77.ec2.internal 5.15.57-28.127.amzn2022.aarch64 #1 SMP Thu Aug 4 17:06:57 UTC 2022 aarch64 |
| Build Date | Jun 7 2022 18:21:38 |
| Build System | Linux |
| Build Provider | Amazon Linux |
| Compiler | gcc (GCC) 11.3.1 20220421 (Red Hat 11.3.1-2) |
| Architecture | aarch64 |
| Server API | FPM/FastCGI |
| Virtual Directory Support | disabled |
| Configuration File (php.ini) Path | /etc |
| Loaded Configuration File | /etc/php.ini |
| Scan this dir for additional .ini files | /etc/php.d |
| Additional .ini files parsed | /etc/php.d/10-opcache.ini, /etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-finfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mysqlind.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmllwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-xmlreader.ini |
| PHP API | 20210902 |
| PHP Extension | 20210902 |
| Zend Extension | 420210902 |
| Zend Extension Build | API420210902,NTS |
| PHP Extension Build | API20210902,NTS |
| Debug Build | no |
| Thread Safety | disabled |
| Zend Signal Handling | enabled |
| Zend Memory Manager | enabled |
| Zend Multibyte Support | provided by mbstring |
| IPv6 Support | enabled |
| DTrace Support | available, disabled |
| Registered PHP Streams | https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar |
| Registered Stream Socket Transports | tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3 |
| Registered Stream Filters | zlib.* , string.rot13, string.toupper, string.tolower, convert.* , consumed, dechunk, bzip2.* , convert.iconv.* |

This program makes use of the Zend Scripting Language Engine:
 Copyright (c) Zend Technologies
 Abbildung 8.16

zend engine

8.3 Aufsetzen des Simulators

Der Simulator wird verwendet, um den Code und die App zu testen. Er wird über XCode aufgesetzt. Dafür wird XCode installiert und gestartet. Dann wird über *XCode -> Open Developer Tools -> Simulator* der Simulator gestartet.

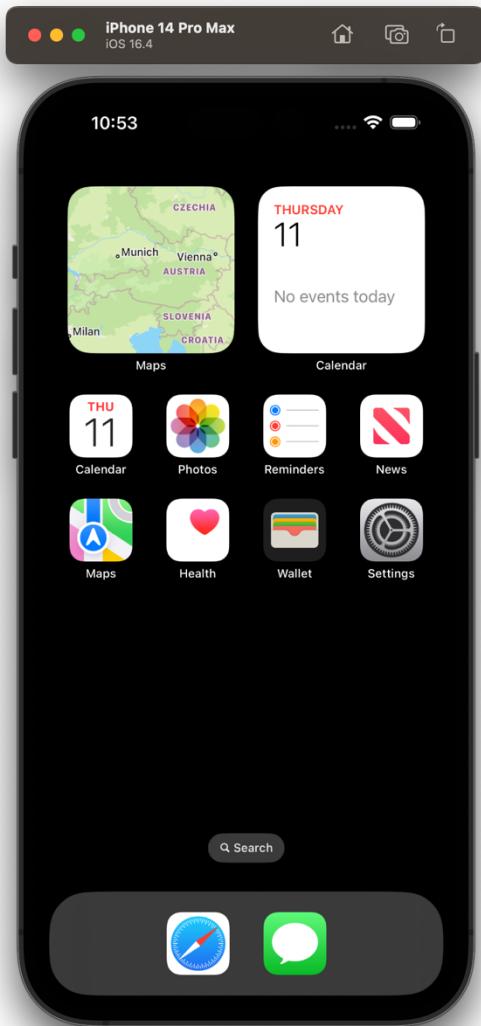


Abbildung 8.17

In VSCode muss der Simulator als Device ausgewählt werden, damit das Programm getestet und kompiliert werden kann. Dafür wird in der rechten unteren Ecke auf das aktuelle Device geklickt.

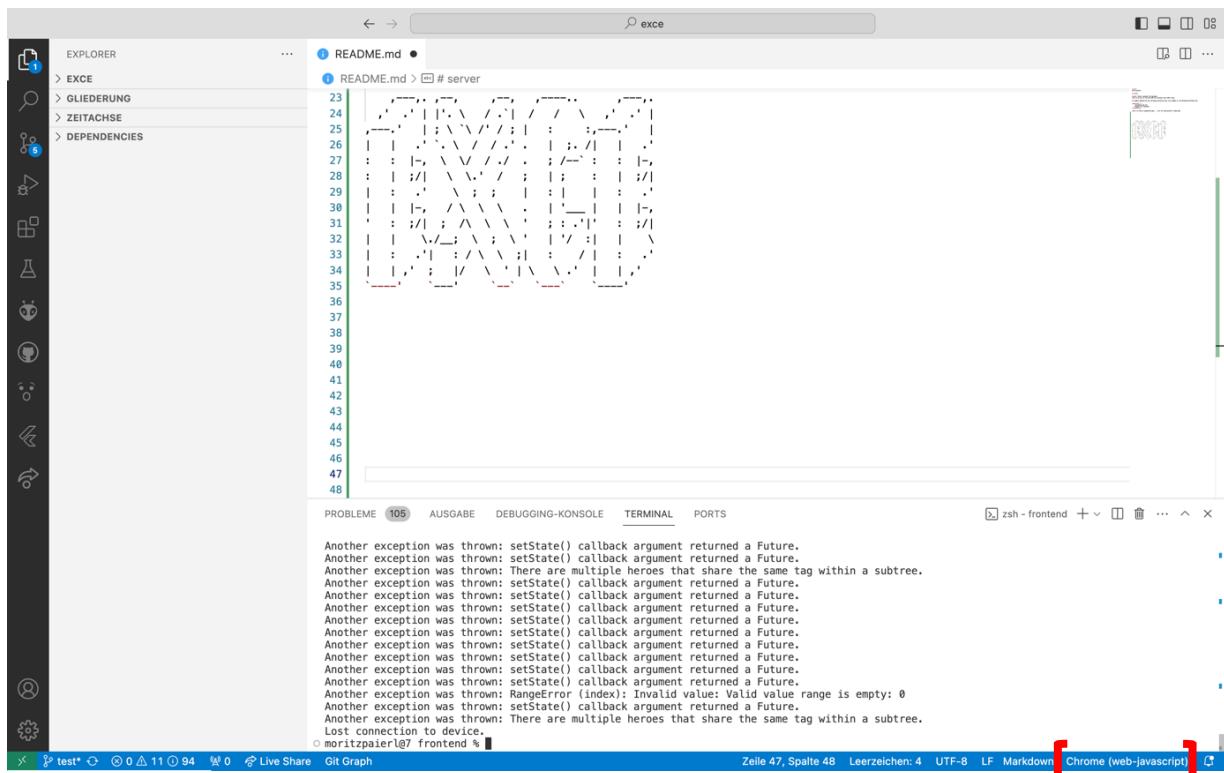


Abbildung 8.18

Im Menü, welches oben in der Mitte des Screens aufgeht, wird nun der Simulator ausgewählt.

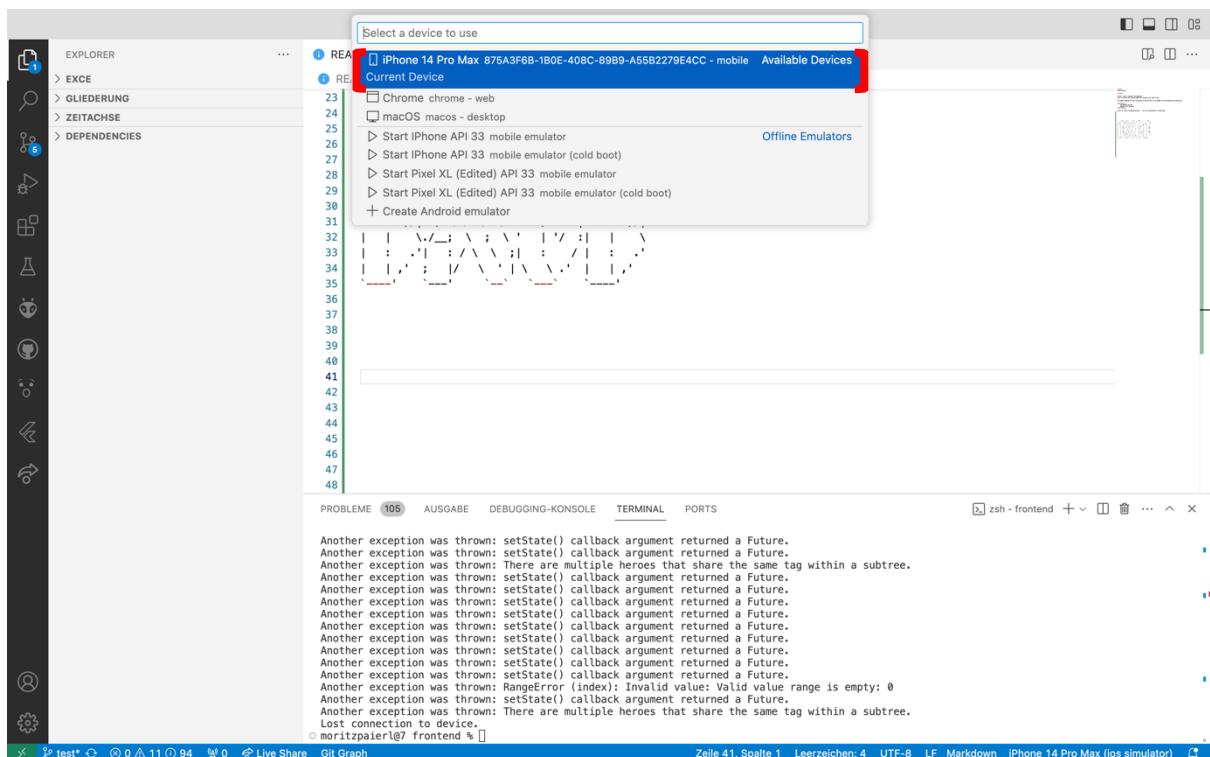


Abbildung 8.19

9 Frontend

Im Kapitel Frontend wird das Setup des Frontends sowie das Zusammenspiel der verschiedenen Klassen (Pages) beschrieben. Außerdem werden einige wichtigere Aspekte des Frontends sowie gewisse Klassen oder Funktionen genauer definiert.

9.1 Aufsetzen des Frontends

Für das Frontend wird Flutter verwendet. Der Hauptgrund, warum Flutter und kein anderes Framework verwendet wird, liegt in der Kompatibilität von Flutter. Der geschriebene Code kann sowohl auf Android, iOS als auch im Web laufen. Als Programmierumgebung wird Visual Studio Code verwendet. Flutter sowie Dart werden direkt in Visual Studio Code als Extension installiert. Zum Überprüfen, ob das Aufsetzen von Flutter erfolgreich war, wird folgender Befehl im Ordner des Projektes ausgeführt:

```
flutter doctor
```

Die Struktur des Projektes wird mit folgendem Befehl automatisch von Flutter generiert:

```
flutter create "name"
```

Die *main.dart* Datei befindet sich im */lib* Ordner und stellt die Basis der App dar. Es befindet sich ein Hello World Programm in der *main.dart* Datei. Dieses Programm wird, bevor entwickelt werden kann, auf einem Handy getestet. Bei dem Handy handelt es sich um ein Samsung Galaxy S9+. Um das Programm auf dem Handy debuggen und laden zu können, muss im *build.gradle* file, welches sich im Pfad *frontend/android/app/*

befindet, die compileSdkVersion auf 34 und die minSdkVersion auf 21 gesetzt werden.

```
android {  
  
    compileSdkVersion 34  
  
    defaultConfig {  
  
        minSdkVersion 21  
    }  
  
}
```

Außerdem wird das Programm auf verschiedenen simulierten Geräten getestet, um die Kompatibilität zu testen und aufzuzeigen. Darunter sind unter anderem das iPhone 14 Pro mit dem Betriebssystem iOS 16.4 und das Google Pixel 3a XL mit dem Betriebssystem Android 13.

Jede Page des Frontends ist eine eigene Klasse, aber nicht jede Klasse im Frontend ist eine Page. Die Pages werden mit Widgets gefüllt, um die App zum Leben zu erwecken.

Im Frontend Ordner wird ein Terminal geöffnet und mit folgendem Befehl wird das Programm auf das Handy geladen und gestartet:

```
flutter run
```

Wird nun im Terminal von VS Code der Buchstabe „r“ eingegeben, wird ein hot reload ausgeführt und im Code geänderte Zeilen werden auf das Handy übertragen. Mit „R“ kann ein hot restart ausgelöst werden. Schreibt man „q“ in das Terminal, kann man die Applikation beenden.

9.2 Zusammenspiel der Pages

Der Login Prozess wird in folgendem Diagramm vereinfacht veranschaulicht. Die schwarzen Kästchen sind dabei Klassen, also Pages, und die grauen Kästchen Buttons der App. Der Start der App und die Ziel Page sind in Grün markiert.

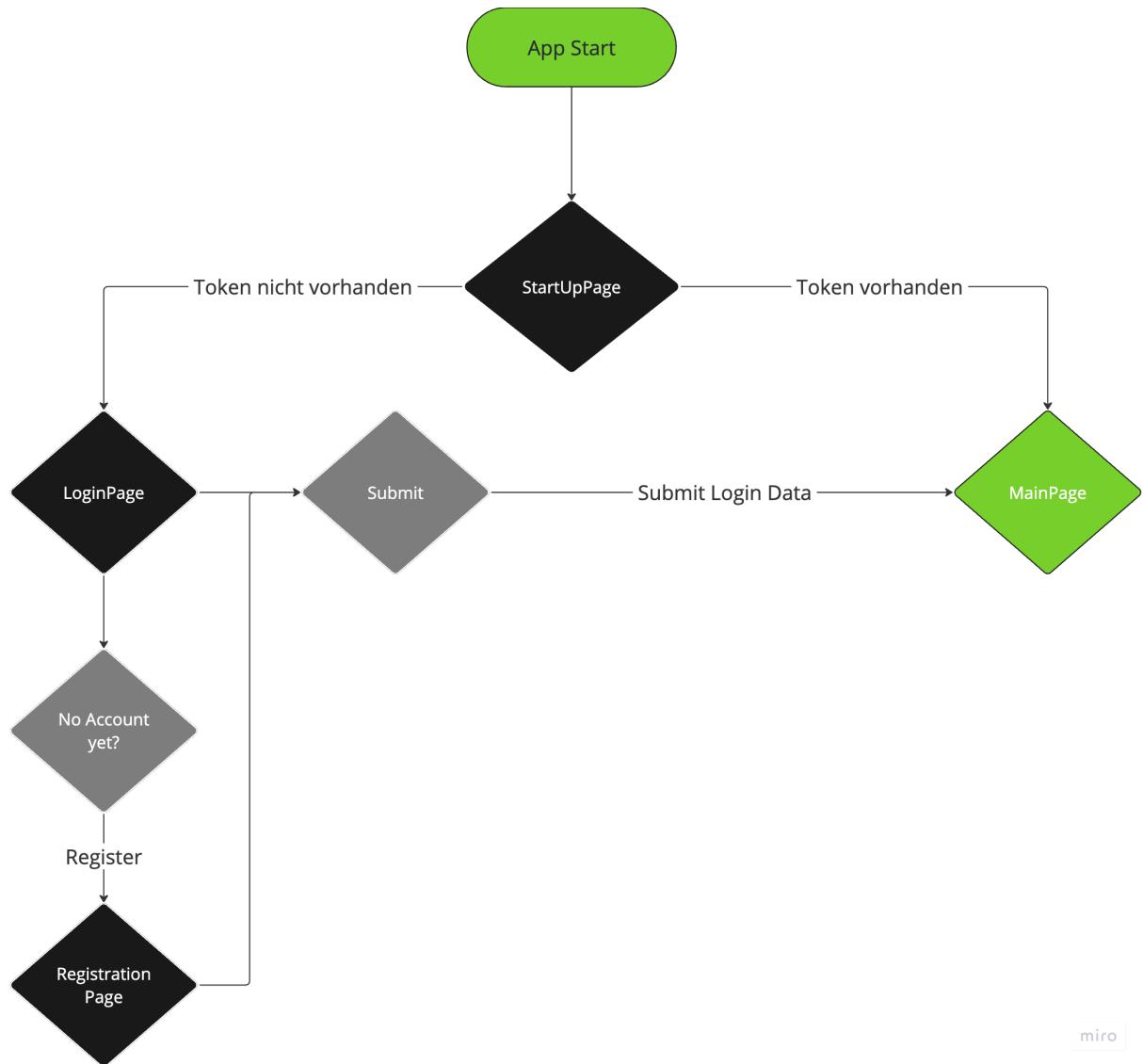


Abbildung 9.1

Die StartUpPage kontrolliert, ob bereits ein Usertoken vorhanden ist. Falls ein Token vorhanden ist, wird direkt auf die MainPage navigiert. Falls kein Token gefunden wird, gibt es die Möglichkeit, einen Token zu generieren. Dies passiert entweder durch eine Anmeldung oder eine Registrierung.

Beim Anmelden werden die Benutzer-Daten, also E-Mail-Adresse und Passwort, in die entsprechenden Eingabefenster eingegeben und der „Submit“ Button gedrückt. Ist noch kein Account vorhanden, kann man über einen Knopfdruck auf den „No Account Yet“ Button zur RegistrationPage kommen. Hier kann ein User, durch das Ausfüllen der Benötigten Felder und das Übermitteln der Daten, erstellt werden. Danach wird, wie auch beim Login zur MainPage, navigiert.

Ist der Login-Prozess abgeschlossen, kann das Verhalten der App durch folgendes Diagramm dargestellt werden.

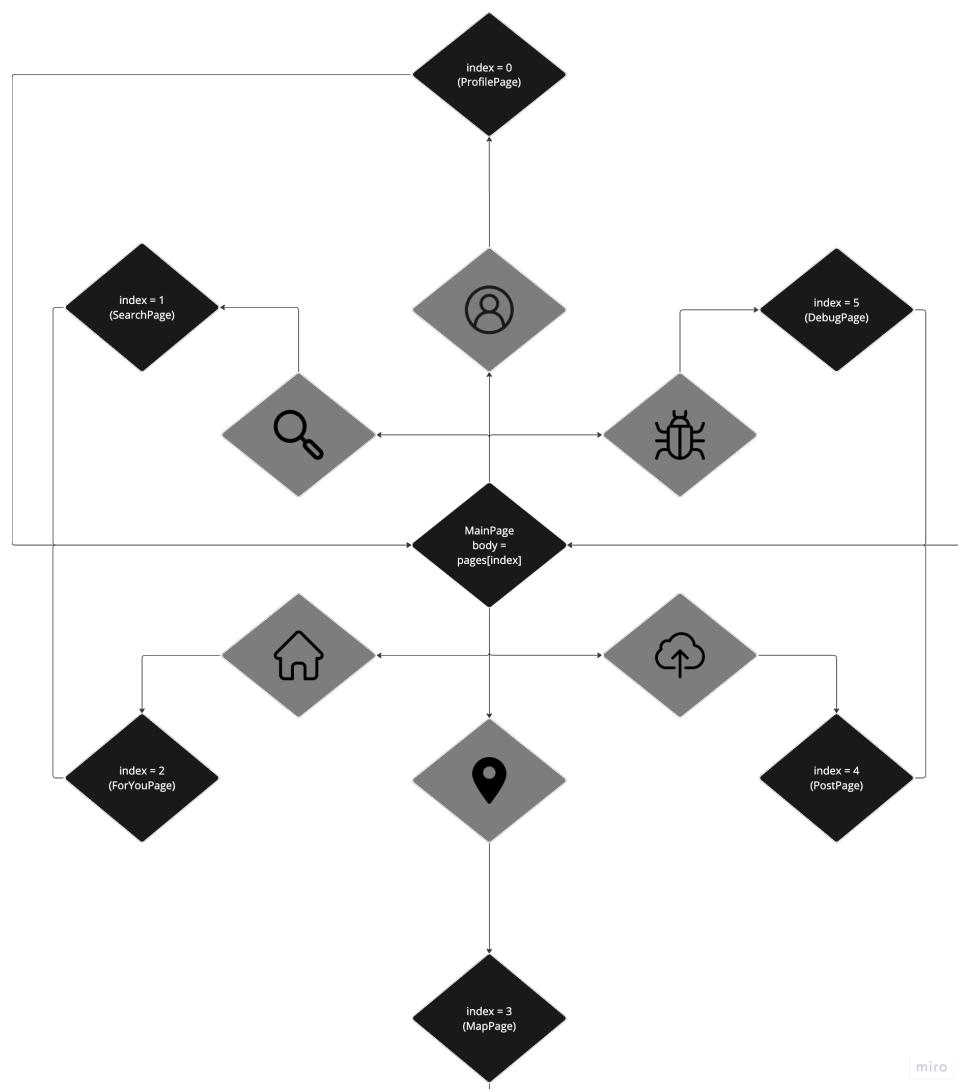


Abbildung 9.2

Wird auf einen der Buttons gedrückt, wird der Index vom array pages[] neu gesetzt und dementsprechend eine neue Page in den Body der MainPage geladen. Die MainPage und der Pagehandler werden in Kapitel 9.6 genauer beschrieben.

9.3 Flutter Basics

In Flutter kann ein Frontend gebaut werden, indem Seiten und Widgets verwendet werden. Die Seiten werden mit Widgets gefüllt und die Widgets können auf vielfältige Art und Weise verwendet werden.

Zum Navigieren zwischen den Seiten und zum Erstellen von Widgets werden stets die gleichen Funktionen und Codeblöcke verwendet. Da diese Funktionen im Code vermehrt vorkommen, werden hier zur Übersichtlichkeit dieser Diplomarbeit die wichtigsten Funktionen beschrieben.

9.3.1 Bausteine des Frontends:

Navigation:

Wird in der Diplomarbeit von Navigation gesprochen, wird folgende Funktion gemeint:

```
Navigator.push(  
    context,  
    MaterialPageRoute(builder: (context) => MainPage()));  
,
```

Die Navigation der sich auf der MainPage befindenden Seiten funktioniert auf eine andere Art und Weise. Diese ist im Kapitel MainPage und Pagehandler genauer beschrieben.

Die sich auf der MainPage befindenden Seiten, für die die Navigation anders funktioniert, sind:

- ProfilePage
- ForYouPage

- MapPage
- UploadPage
- DebugPage

Folgende Seiten navigieren über die oben aufgezeigte Navigator-push-Funktion:

- StartUpPage
- LoginPage
- RegistrationPage
- DisplayPicturePage
- Settingspage
- EmiplusMo

Buttons:

Wird ein Button auf dem Bildschirm angezeigt, wurde er durch folgenden Code erstellt.

```
FloatingActionButton(onPressed: () async {}, child:),
```

Mit dem Floating Action Button wird der Button erstellt. Das child Attribut wird mit einem Icon oder Text befüllt. Wenn ein Text eingefüllt werden soll, wird dem child ein Text Widget übergeben.

```
child: Text("text of button"),
```

Wird ein Icon eingefügt, wird dem child Attribut ein Icon Widget hinzugefügt.

```
child: Icon(Icons.upload,  
color: Color.fromARGB(255, 255, 255, 255)),  
,
```

Eingabefelder:

Wird ein Eingabefeld auf dem Bildschirm angezeigt, wurde es durch folgenden Code erstellt.

```
TextField(controller:, keyboardType:, decoration:)
```

Das TextFormField bekommt einen Controller, welcher dafür benötigt wird, den Text im Eingabefeld zu verwalten. Hier ein Beispiel für einen TextEditingController mit dem Namen _email.

```
TextEditingController _email = TextEditingController();
```

Der TextInputType gibt an, was für eine Eingabe getätigt wird. Zum Beispiel in diesem Fall eine E-Mail Adresse.

```
keyboardType: TextInputType.emailAddress
```

Die Decoration bietet die Möglichkeit, einen Label Text und ein Icon hinzuzufügen.

```
decoration: InputDecoration(  
    labelText: "Email",  
    icon: Icon(Icons.email),  
,  
,
```

9.4 StartUpPage:

Die StartupPage wird bei jedem Start der App ausgeführt. Sie überprüft, ob bereits ein Token vorhanden ist. Ist das der Fall, wird sofort auf die MainPage navigiert. Ist kein Token vorhanden, wird der Login Prozess durchgeführt. Die Überprüfung erfolgt mit der `_loadUserFile()` Funktion.

```
_loadUserFile() async {
    String userInfo = await readUserFile();

    if (userInfo == '') {
        if (!context.mounted) return;
        Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => LoginPage()),
        );
    } else {
        if (!context.mounted) return;
        Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => MainPage()),
        );
    }
}
```

9.5 Login & Registration Page

Die StartUpPage, LoginPage und RegistrationPage spielen, wie in 9.2 beschrieben, zusammen. Für den Login sowie die Registrierung müssen eine E-Mail Adresse sowie ein dazu passendes Passwort eingegeben werden.

9.5.1 LoginPage

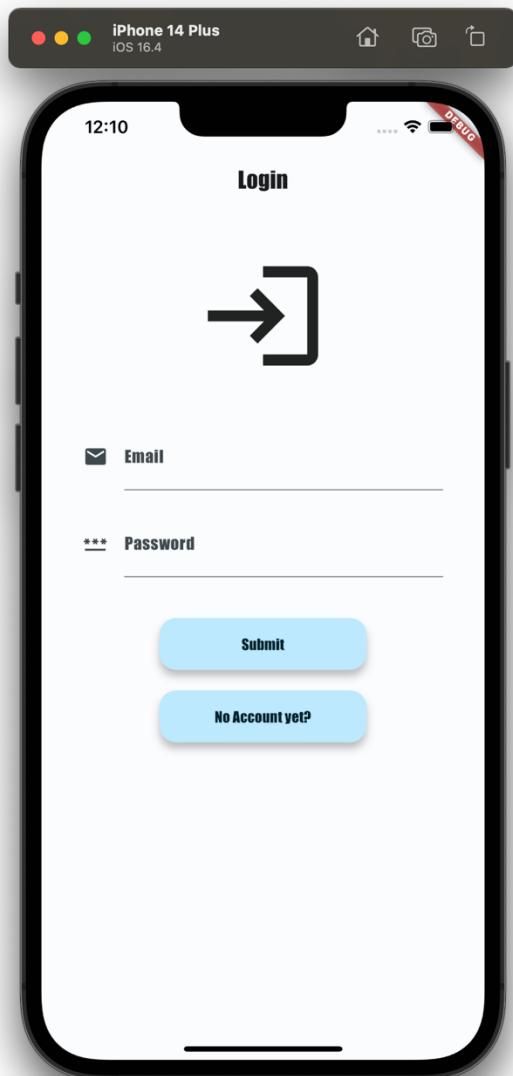


Abbildung 9.3

Die LoginPage ermöglicht es dem User, sich in der App anzumelden. Sie besteht aus zwei Eingabefeldern: eines für die E-Mail Adresse und eines für das Passwort. Außerdem gibt es einen Button, welcher den Anmeldevorgang durchführt und einen, welcher den User zur RegistrationPage weiterleitet.

Der Login erfolgt nach einem Knopfdruck auf den Submit Button mit folgender Funktion.

```
await Shared().login(_email.text, _password.text);
```

Die login Funktion im Shared postet dann email und den MD5 hash des Passwortes ins Backend.

```
Future<String> login(email, password) async {
    final url = 'http://13.48.177.242/api/login';
    Map<String, String> headers = {"Content-Type": "text/plain"};
    final uri = Uri.parse(url);
    final response =
        await http.post(uri, headers: headers, body: email + "\n" + password);

    final content = response.body;
```

Der Statuscode wird mittels einer „if“ Abfrage überprüft. Ist der Statuscode 200 (http code ok), wird der body in das userFile geschrieben.

```
if (response.statusCode == 200) {
    final file = await userFile;
    file.writeAsString(content);
```

Die user-Daten werden dann aus dem Json String (content) gelesen und in der Shared Klasse in Variablen gespeichert, damit sie in der App verwendet werden können. Zum decoden wird die jsonDecode() Funktion verwendet.

```
final user = jsonDecode(content);
_email = user['email'];
_password = user['password'];
_token = user['token'];
return content;
```

Wird ein anderer Statuscode returniert, wird ein Error ausgegeben.

```
} else {
    return "Error 2";
}
```

9.5.2 RegistrationPage

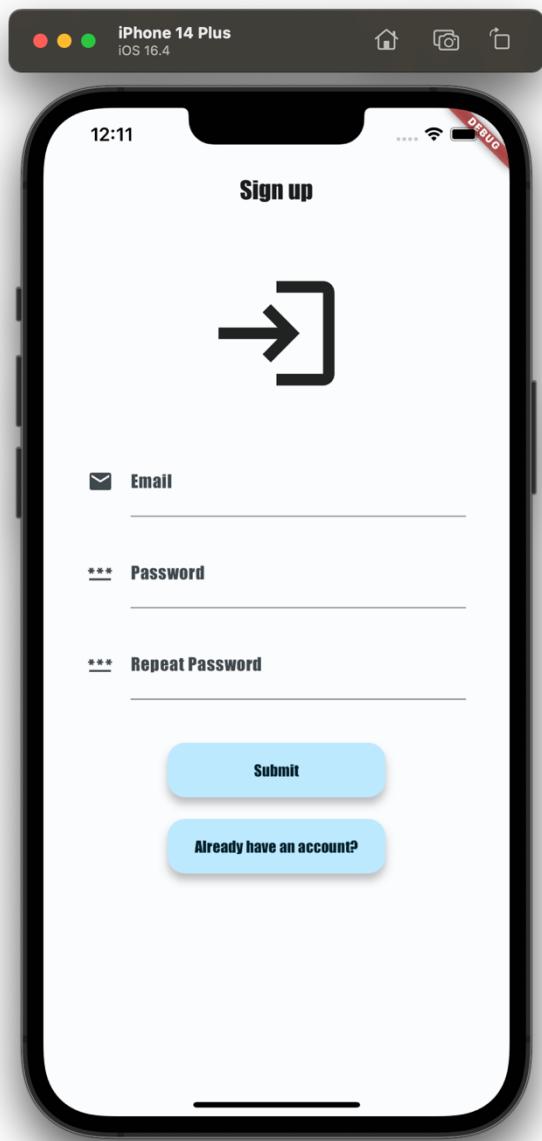


Abbildung 9.4

Die RegistrationPage kann von der LoginPage aus mit einem Klick auf den „No Account Yet“ Button aufgerufen werden. Sie ist gleich aufgebaut wie die LoginPage, nur dass ein weiteres Eingabefeld für eine Passwortbestätigung vorhanden ist

Die Registrierung erfolgt nach Knopfdruck auf den Submit Button. Zuerst wird überprüft, ob das Passwort zweimal richtig eingegeben wurde.

```
if (_passConfirm.text == _password.text) {
```

Ist das der Fall, führt folgende Funktion den Registrierungsvorgang durch.

```
await Shared().register(_email.text, _password.text);
```

Die Funktion register() postet die Email-Adresse und den MD5 hash des Passwortes zur Route „<http://13.48.177.242/api/register>“.

```
Future<String> register(email, password) async {
    final url = 'http://13.48.177.242/api/register';
    Shared().log('LOGIN: $url $email ');
    Map<String, String> headers = {"Content-Type": "text/plain"};
    final uri = Uri.parse(url);
    final response =
        await http.post(uri, headers: headers, body: email + "\n" + password);

    final content = response.body;
```

Es wird auf den Statuscode 200 (http code ok) überprüft und dann werden aus dem content des response body wieder die Daten ausgelesen.

```
if (response.statusCode == 200) {
    final file = await userFile;
    file.writeAsString(content);

    final user = jsonDecode(content);
    _email = user['email'];
    _password = user['password'];
    _token = user['token'];

    return content;
```

Wird ein anderer als Code 200 zurückgegeben, wird wieder ein Error Code ausgegeben.

```
} else {
    return "Error 2";
}
```

9.6 MainPage und Pagehandler

Die MainPage liegt im main.dart File und bildet das Grundgerüst der App.

Die Imports der main.dart Datei sind wie folgt:

```
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';

//Pages import
import "SearchPage.dart";
import "MapPage.dart";
import "ProfilePage.dart";
import "DebugPage.dart";
import "ForYouPage.dart";
import "UploadPage.dart";
import "StartUpPage.dart";
```

Die MainPage ist ein StatefulWidget, da sich der Zustand der Page ändert. In unserem Fall wird die MainPage ihren eigenen Zustand managen. Es wird mit dem @override Befehl ein State Objekt erstellt. Wenn ein Widget gebildet werden soll, ruft das Framework createState() auf. Der Befehl createState() erstellt dann eine Instanz der Klasse _MainPageState.

```
class MainPage extends StatefulWidget {
  @override
  _MainPageState createState() => _MainPageState();
}
```

Die _MainPageState Klasse beinhaltet die Daten, die über die Lebensdauer des Widgets verändert werden können.

```
class _MainPageState extends State<MainPage> {
```

Auf der MainPage wird zwischen den verschiedenen Pages gewechselt. Dafür werden ein currentIndex und eine Liste von Widgets, welche mit unseren Pages gefüllt wird, erstellt. Die Liste der Pages heißt pages.

```
int currentIndex = 0; // Aktueller Index für die Bottom Navigation Bar

//Liste an pages für Bottom Navbar
List<Widget> pages = [
  ProfilePage(),
```

```
SearchPage(),
ForYouPage(),
MapPage(),
CameraScreen(),
DebugPage(),
];
```

Das Widget wird gebuilled und gibt ein Scaffold zurück. Im body des Widgets liegt pages[currentIndex], also immer die Seite, welche in der pages liste auf dem currentIndex liegt.

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        body: pages[currentIndex],
```

Die BottomNavigationBar wird von links nach rechts mit 6 Items befüllt. Die Items auf der BottomNavigationBar heißen Profile, Search, For You, Map, Post und Debug. Die Elemente fungieren als Buttons und haben einen Index. Wird nun beispielsweise der vierte Button in der BottomNavigationBar, also der Map Button, gedrückt, wird über die onTap Funktion der currentIndex auf den Index der Map Page, also 3, gesetzt. Dadurch wird der body des Scaffolds in der MainPage auf pages[3] geändert und zeigt somit die Map an.

```
bottomNavigationBar: BottomNavigationBar(
    type: BottomNavigationBarType.fixed, // Festes Design für mehr als 3 Tabs

    currentIndex: currentIndex, // Aktuell ausgewählter Index

    onTap: (index) {
        setState(() {
            currentIndex = index;
        });
    },

    items: [
        BottomNavigationBarItem( //liegt auf index 0
            icon: Icon(Icons.account_circle_rounded),
            label: 'Profile', // Label der Profile Page
        ),
    ],
);
```

```
BottomNavigationBarItem( //liegt auf index 1
    icon: Icon(Icons.search_rounded),
    label: 'Explore', // Label der Search Page
),

BottomNavigationBarItem( //liegt auf index 2
    icon: Icon(Icons.home_rounded),
    label: 'For You', // Label der For You Page
),

BottomNavigationBarItem( //liegt auf index 3
    icon: Icon(Icons.map_rounded),
    label: 'Map', // Label der Map Page
),

BottomNavigationBarItem( //liegt auf index 4
    icon: Icon(Icons.add_a_photo_rounded),
    label: 'Post', // Label der Upload Page
),

BottomNavigationBarItem( //liegt auf index 5
    icon: Icon(Icons.admin_panel_settings_rounded),
    label: "Debug" // Label der Debug Page
),
],  
,
```

9.7 ProfilePage



Abbildung 9.5

Auf der ProfilePage kann der User sich ausloggen und seine Userdaten (Geburtstag und Biografie) ändern. Der Username wird in der Shared Klasse automatisch aus der Emailadresse extrahiert. Der String, welcher sich in der Email Adresse vor dem „@“ befindet, ist der Benutzername.

```
get username {  
    return _email.split('@');  
}
```

Durch das Splitten wird ein String Array erstellt. Das erste Element des String Arrays wird dann auf dem Bildschirm wiedergegeben.

```
Shared().username[0]
```

Die E-Mail Adresse wird ebenfalls aus der Shared Klasse genommen. Die Daten werden, wie oben bereits beschrieben, vorher mithilfe der Login oder der Register Funktion aus dem userfile in den Variablen abgespeichert.

```
if (response.statusCode == 200) {  
    final file = await userFile;  
    file.writeAsString(content);  
  
    final user = jsonDecode(content);  
    _email = user['email'];  
    _password = user['password'];  
    _token = user['token'];  
  
    return content;  
}
```

Wird auf den Settings Button gedrückt, gelangt der User zur SettingsPage. Auf dieser kann er seine Daten in zwei Input-Felder eingeben und mit einem Klick auf den Speichern-Button in die Profilepage übernehmen.

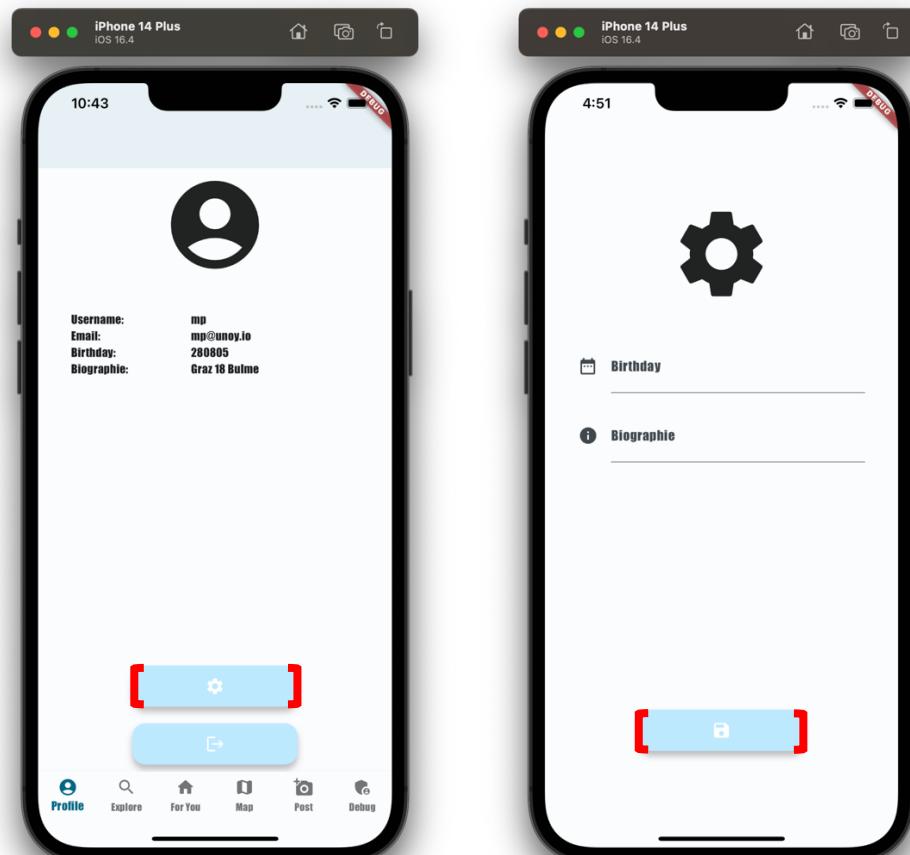


Abbildung 9.6

9.8 MapPage

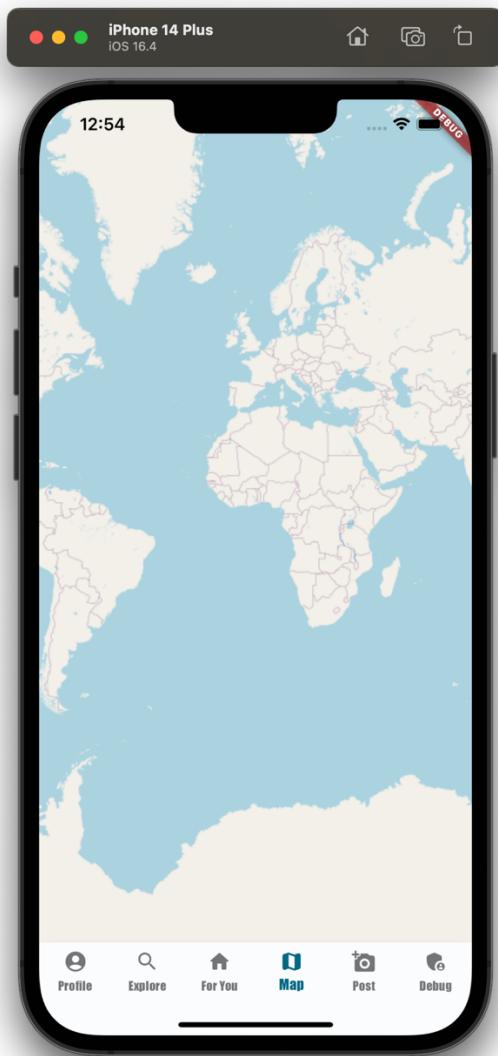


Abbildung 9.7

Für die Weltkarte wird die Bibliothek syncfusion_flutter_maps verwendet. Zum Verwenden dieser Bibliothek muss die Version der http Bibliothek im pubspec.yaml File auf kleiner als 0.13.3 gestellt sein. Um die Weltkarte zu erstellen, sind folgende packages benötigt:

```
import 'package:exce/Shared.dart';
import 'package:flutter/material.dart';
import 'package:syncfusion_flutter_maps/maps.dart';
```

Die Weltkarte wird mithilfe eines MapTileLayer Widgets erstellt. Der Attribut urlTemplate wird mit einem Link befüllt, welcher die Tiles ladet.

Es werden beim Zoomen oder Bewegen der Karte immer nur die benötigten Tiles geladen, was eine TileMap so effizient macht.

```
MapTileLayer(  
    urlTemplate:  
        'https://tile.openstreetmap.org/{z}/{x}/{y}.png',  
)
```

Das initialFocalLatLng Attribut des MapTileLayer Widgets setzt den ersten Mittelpunkt der Map und das initialZoomLevel setzt den Zoom beim Starten der Map

```
initialFocalLatLng: MapLatLng(0, 0),  
initialZoomLevel: 3,
```

zoomPanBehavior ist das Attribut zum Verwalten aller Zoom bezogenen Eigenschaften sowie das ZoomLevel, den max- sowie minZoomLevel und was für eine Zoom Methode man verwenden soll. In diesem Fall wird pinching als Zoom Methode verwendet.

```
zoomPanBehavior: MapZoomPanBehavior(  
    zoomLevel: 2,  
    maxZoomLevel: 17,  
    minZoomLevel: 2,  
    enablePinching: true),
```

Der initialMarkersCount bestimmt, wieviele Marker auf der Map angezeigt werden können. Die Anzahl wird mit der Länge der markerData vergrößert. Die markerData liegt in der Shared Klasse und ist eine Liste, welche mit Objekten, bestehend aus latitude und longitude, befüllt ist.

```
initialMarkersCount: Shared().markerData.length,
```

Die Marker werden mithilfe eines markerBuilders gebaut. Dieser gibt für jedes Objekt in der markerData, welche sich im Shared File befindet, einen Marker zurück. Beim Uploaden eines Bildes werden die Handy-Koordinaten zur markerData hinzugefügt und so wird die Anzahl der Marker erweitert (mehr dazu im Kapitel 9.9 UploadPage). Die Marker sind kleine farbige Punkte auf der Karte, die Icon Color der Marker ist für jeden Marker zufällig gesetzt.

```
markerBuilder: (BuildContext context, int index) {
    return MapMarker(
        latitude: Shared().markerData[index].latitude,
        longitude: Shared().markerData[index].longitude,
        iconColor: Color(
            (math.Random().nextDouble() * 0xFFFFFFFF).toInt()
        ).withOpacity(1.0),
    );
};
```

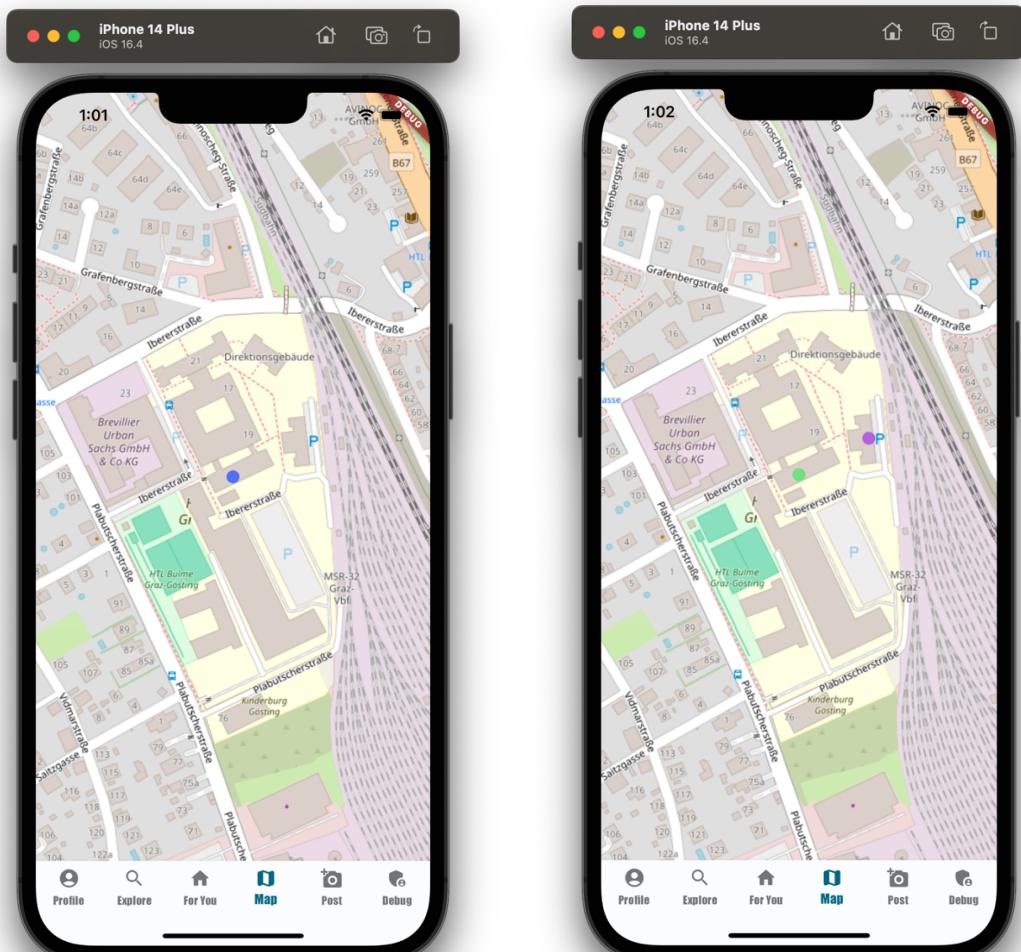


Abbildung 9.8

9.9 UploadPage

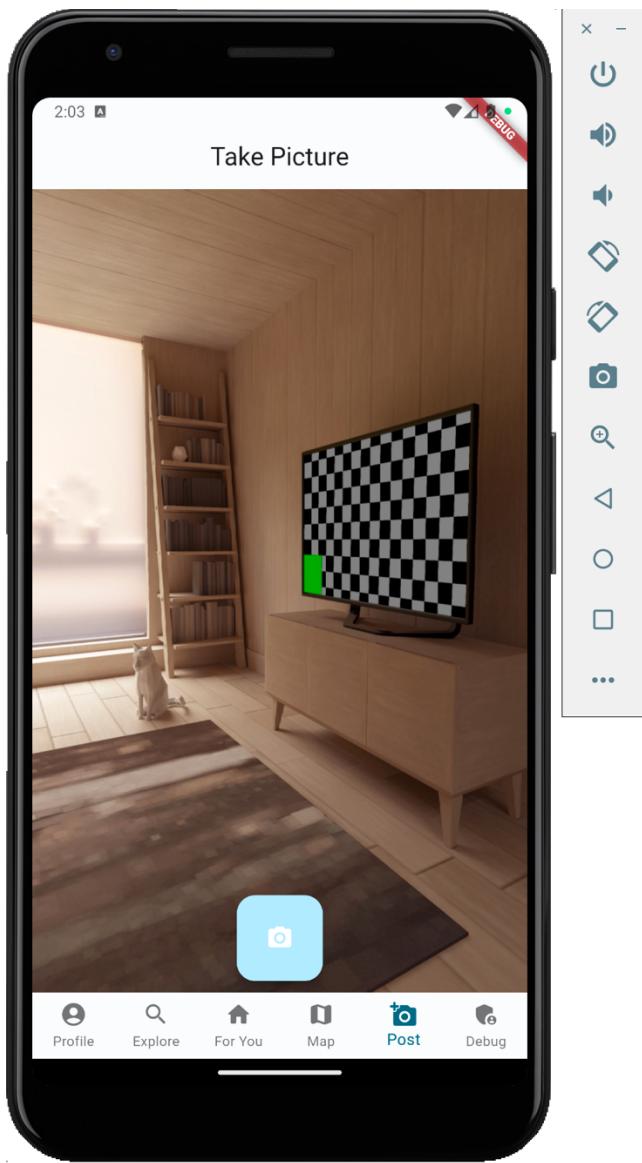


Abbildung 9.9

Auf der UploadPage ist es dem Benutzer möglich, ein Bild aufzunehmen und zu posten. Dafür wird zuerst eine Liste von CameraDescriptions mit dem Namen cameras erstellt.

```
late List<CameraDescription> cameras;
```

Diese wird dann mit allen verfügbaren Kameras befüllt.

```
cameras = await availableCameras();
```

Dann wird ein Kamera Controller erstellt und mit dem ersten Element der cameras-Liste verknüpft.

```
late CameraController controller;  
controller = CameraController(cameras[0], ResolutionPreset.max);
```

Der Controller wird mit der initialize Methode initialisiert.

```
controller.initialize()
```

Zum Aufnehmen eines Fotos wird die takePicture() Methode verwendet. Sie kontrolliert zuerst, ob bereits ein Bild aufgenommen wird, um nicht zweimal gleichzeitig ausgeführt werden zu können.

```
Future<XFile?> takePicture() async {  
    if (controller.value.isTakingPicture) {  
        return null;  
    }  
}
```

Danach wird mithilfe des initialisierten Kontrollers ein Bild aufgenommen. Der ganze Prozess wird in ein try / catch gegeben, um Fehler abzufangen.

```
try {  
    XFile file = await controller.takePicture();  
    return file;  
} on CameraException catch (e) {  
    print(e);  
    return null;  
}  
}
```

Wird nun der Hellblaue takePicture Button gedrückt, wird das Bild aufgenommen und es wird zur DisplayPicturePage Seite navigiert.

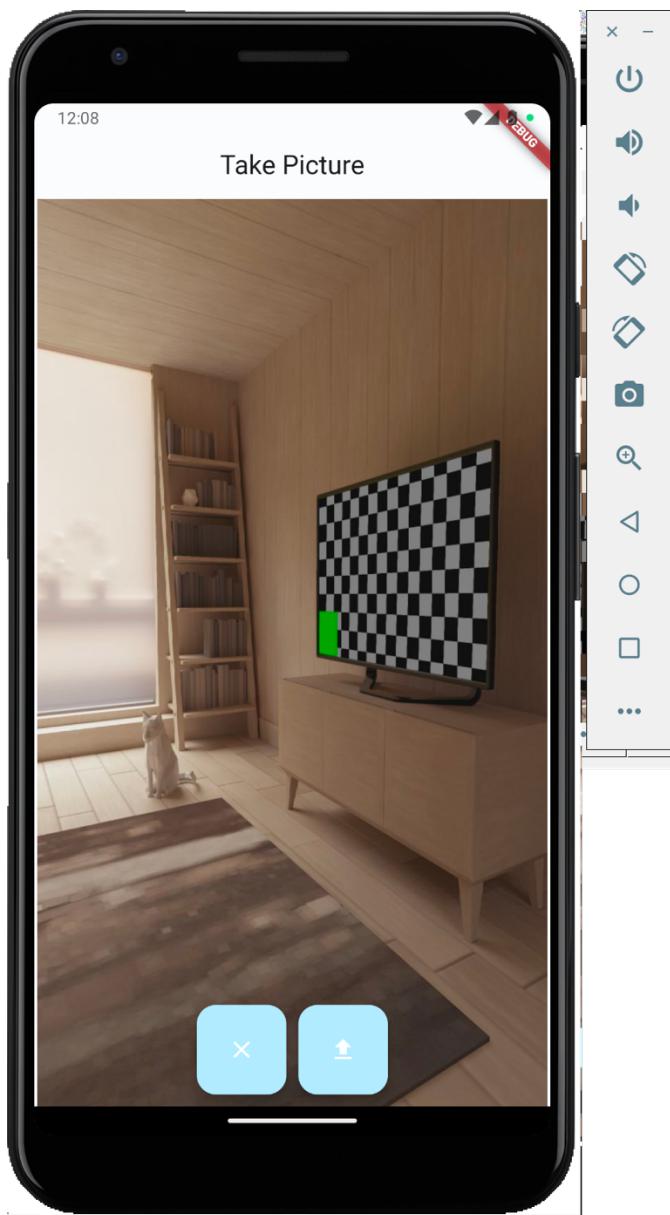


Abbildung 9.10

Auf dieser wird das Bild, welches aufgenommen wurde, angezeigt und es gibt die Option, das Bild hochzuladen oder es erneut aufzunehmen. Die DisplayPicturePage Seite ist ein Stateless Widget und bekommt den Pfad des Bildes beim Navigieren übergeben.

```
class DisplayPicturePage extends StatelessWidget {  
    final String imagePath;  
  
    const DisplayPicturePage({super.key, required this.imagePath});
```

Das File wird einem Container übergeben und dann in der Page angezeigt.

```
Container(child: Image.file(File(imagePath))),
```

Wird nun der Upload Button gedrückt, wird das Bild hochgeladen (mehr Informationen in Kapitel 10 Backend). Außerdem werden mithilfe des Geolocator packages die Koordinaten, an denen das Bild aufgenommen wurde, zur markerData hinzugefügt (wie in Kapitel MapPage beschrieben).

```
Position position = await Geolocator.getCurrentPosition(desiredAccuracy:  
LocationAccuracy.high);  
Shared().markerData.add(Model(position.latitude, position.longitude));
```

10 Backend

Im Kapitel Backend wird die Funktionalität des Backends beschrieben. Das Backend ist der funktionale Teil eines Software-Systems und bezieht sich auf eine „Client Server Architektur“. Dabei nützen Clients (die sich autorisieren müssen) das Backend (=Server), um Daten abzurufen und Daten zu speichern.

10.1 Allgemeines

Üblicherweise werden die Zugriffe auf das Backend in einem REST API zur Verfügung gestellt. REST-API steht dabei für „Representational State Transfer - Application Programming Interface“.

Die Datenbank des Systems ist direkt mit dem Backend verbunden und wird nur von diesem benutzt. Die Ports, die zur Kommunikation mit der Datenbank verwendet werden, können also nach außen hin geschlossen werden, was die Sicherheit erhöht, da die Datenbank nach außen isoliert ist.

10.1.1 Tokens

Die Berechtigungen für die Zugriffe auf das API werden durch Zugriffs Tokens geregelt. Der Client, der einen Token „besitzt“, kann gewisse Funktionen im API nutzen, je nachdem, welche Funktionalität an den Token gebunden ist. Wir sprechen hier vom „Bearer Token Authentication“ Verfahren.

Das Bearer Token Verfahren ist eines der Standard HTTP Authentifizierungsverfahren. Es eignet sich deshalb sehr gut für APIs, weil weder Benutzername noch Kennwort erforderlich sind. Ein Token wird generiert und zur Verfügung gestellt und kann bei Bedarf einfach widerrufen werden.

Die Tokens werden in der Datenbank für jeden User gespeichert und können mit zusätzlichen Attributen wie Berechtigungen oder Funktionen verknüpft werden.

Um einen Bearer Token zu verwenden, schickt man diesen einfach im http header mit:

„Authorization“ : „Bearer {TOKEN}“

Der Server überprüft das Vorhandensein des Authorization Feldes, liest den Token und kann

- den Benutzer ermitteln
- die Gültigkeitsdauer des Tokens ermitteln
- spezifische Berechtigungen ermitteln (Nur Lesezugriff / Lese Schreib Zugriff / Demo Modus / Vollversions Modus ...)

10.1.2 PHP vs Node

Das Backend wurde in PHP implementiert. Um die Diskussion, ob PHP oder NodeJS, kurz zu halten: Es wurde hier deshalb PHP verwendet, weil es

- der Syntax von C sehr ähnlich ist und wir im Verlauf der letzten fünf Jahre viel in C programmiert haben, was es leichter macht, die Sprache und Syntax zu verstehen.
- eine interpretative Sprache ist und Änderungen im Code ohne neues Compilieren oder Neustart der Anwendung durchgeführt werden können. Deshalb eignet sich PHP sehr gut zum raschen effektiven Erstellen von funktionalen Prototypen.

Nachteile von PHP liegen in der Performance. Eine Node-Anwendung läuft multi-threaded, während eine PHP-Anwendung stets auf einem Thread läuft. Das bedeutet, dass eine Server-Anfrage in PHP stets abgearbeitet

werden muss, bevor die nächste angenommen werden kann. Beim Bau eines Prototyps ist dies aber nicht weiter von Bedeutung.

10.2 SQL Datenbank

Als Datenbank wurde MariaDB verwendet. MariaDB ist ein freies relationales Open Source Datenbank Managementsystem, welches durch Abspaltung aus MySQL entstanden ist.

Als Administrations-Oberfläche wurde PhpMyAdmin installiert.

Die Datenbank ist ein essenzieller Bestandteil dieses Systems.

Sie erfüllt folgende Kernaufgaben:

1. Benutzer können angelegt und mit Berechtigungen ausgestattet werden.
2. Benutzer können sich an- und abmelden. Beim Anmelden werden Tokens generiert, welche vom API verwendet werden, um Benutzer zu identifizieren. Beim Abmelden werden diese Tokens wieder gelöscht.
3. In diesem System werden Dateien (Bilder) in einem CDN (Content delivery network) gespeichert. Um Bilder komfortabel suchen und abrufen zu können, werden die Links zu den Dateien in der Datenbank gespeichert. So ist es möglich, per SQL Abfragen zu suchen. Zum Beispiel:

```
„SELECT * from images WHERE user`=55 AND `created` > „2024 01 01T00:00:00.000Z“ AND `created` < „2024 01 02T00:00:00.000Z“
```

Diese Abfrage liefert alle Bilder, die von Benutzern mit id 55 am 01.01.2024 hochgeladen wurden. Würde man die Bilder direkt auf dem Server speichern, hätte man folgende Probleme:

1. Das Dateisystem des Servers würde irgendwann voll werden.
2. Man müsste sich selbst um das Backup der Bilder kümmern.
3. Es wäre sehr aufwendig, Bilder zu suchen und finden.

Deshalb wurde der Weg gewählt, die Bilder zuerst in die S3 bei AWS hochzuladen und dann den erhaltenen Abruf URL samt dem ursprünglichen Dateinamen, Benutzer ID und einen Zeitstempel hinzuzufügen.

Dadurch wird es möglich, wie oben beschrieben, komfortabel in den Bildern suchen zu können und diese dann von CDN laden zu können. Das Laden kann auch direkt vom Frontend aus erfolgen, so dass keine Belastung durch Bild- Downloads am Server entsteht.

Bei großen Softwareprojekten spielt vor allem die Weiterentwicklung der Datenbank eine große Rolle. Man steht oft vor der Herausforderung, dass sich die Anforderungen an die Datenbank ebenso wie die Anforderungen an die Software ständig ändern. Das bedeutet, dass sich die Datenbank ebenfalls weiterentwickelt. Damit dies nicht unkontrolliert geschieht, erzeugt man normalerweise „Migrations-Skripte“ und stellt dies unter Versions-Kontrolle. Jedes Migrations-Skript beinhaltet eine up() und eine down() Funktion, welche in der Lage ist, die Datenbank auf den nächsten Level anzuheben oder auf den vorherigen Level zurückzusetzen. Bei einem Softwareupdate wird die neue Software eingespielt und das Migrationsskript ausgeführt. Dadurch erhält man die nächste Datenbank-Version, die genau zur freigegebenen Software passt.

Das erste dieser Migrations Skripte erzeugt die initiale (erste) Version der Datenbank.

Durch dieses Skript und die darauffolgenden Migrations-Skripte kann jede beliebige Version der Datenbank zumindest von der Struktur wiederhergestellt werden. Natürlich kann es bei einem Downgrade vorkommen, dass Daten verloren gehen, wenn man z.B. Spalten entfernt. Deshalb muss man mit der Datenbank sehr verantwortungsvoll umgehen - Fehler in Migrationsskripten oder Verlust der Datenbank können komplexe Systeme komplett zum Stillstand bringen.

Dieses Projekt befindet sich noch in der ersten Version. Es gibt also noch keine Migrationsskripte, außer dem initialen Skript zum Erstellen der Datenbank.

Im Folgenden soll das initiale Datenbank-Skript hier beschrieben werden:

```
START TRANSACTION;
```

Hier wird eine Transaktion gestartet. Eine Transaktion beschreibt eine Folge von Befehlen, welche entweder vollständig bei Fehlerfreiheit oder sonst gar nicht ausgeführt wird. Abgeschlossen wird die Transaktion mit einem Commit Kommando (siehe unten):

```
SET time_zone = "+00:00";
```

Die Zeitzone wird auf UTC gesetzt. Es hat sich bewährt, immer in Software mit UTC Zeiten (Zulu Time) zu arbeiten und nur bei Ausgaben diese in die lokale Zeit umzuwandeln:

```
CREATE TABLE `s3` (
  `id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `name` varchar(32) NOT NULL,
  `s3key` text NOT NULL,
  `created` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COLLATE=latin1_swedish_ci;
```

Hier wird die Tabelle S3 angelegt. In dieser Tabelle speichern wir unsere Bilder ab. Zusätzlich speichern wir die Benutzer ID den Namen und den s3 key und einen created Zeitstempel ab:

```
INSERT INTO `s3` (`id`, `user_id`, `name`, `s3key`, `created`)
VALUES
(1, 8, '1707763900.8.png', 'https://mopa-exce.s3.eu-west-1.amazonaws.com/1707763900.8.png', '2024-02-12 18:51:41'),
```

```
(2, 8, '1707763903.8.png', 'https://mopa-exce.s3.eu-west-1.amazonaws.com/1707763903.8.png', '2024-02-12 18:51:43'),
(3, 8, '1707764111.8.png', 'https://mopa-exce.s3.eu-west-1.amazonaws.com/1707764111.8.png', '2024-02-12 18:55:11');
```

Hier werden die Daten für die Tabelle als Zeilen hinzugefügt.

```
CREATE TABLE `users` (
  `id` bigint(20) NOT NULL,
  `username` varchar(16) NOT NULL,
  `email` varchar(128) NOT NULL DEFAULT '',
  `password` varchar(32) NOT NULL,
  `token` text DEFAULT NULL,
  `expires` varchar(24) DEFAULT NULL,
  `last_login` varchar(24) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COLLATE=latin1_swedish_ci;
```

Hier wird die Tabelle Users erzeugt. Jeder Benutzer hat eine eindeutige ID, einen Benutzernamen, eine E-Mail Adresse, ein Kennwort, und einen Zugriffstoken, welcher einen Gültigkeits-Zeitstempel trägt, sowie ein last_login Zeitstempel Feld:

```
INSERT INTO `users` (`id`, `username`, `email`, `password`,
`token`, `expires`, `last_login`) VALUES
(1, 'mopa', 'mp@unoy.io', 'e2fc714c4727ee9395f324cd2e7f331f',
', 'ca8d94c3-2460-4bca-9824-2d8a443a04f1', NULL, '2024-03-
08T09:34:24.000Z'),
(2, 'zimstep', 'sz@unoy.io ',
'81dc9bdb52d04dc20036dbd8313ed055
', '06c0a855-85ce-4c6f-b220-7f05e7e93ee7', NULL, '2023-11-
27T14:22:25.000Z');
```

Hier werden zwei Benutzer angelegt. Wie man deutlich erkennen kann, werden die Kennworte nicht in Klartext gespeichert, sondern es werden die MD5 Hashes der Kennworte gespeichert. So wird das Kennwort eines Benutzers nie preisgegeben.

```
ALTER TABLE `s3`  
    ADD PRIMARY KEY (`id`);
```

Die Spalte ID in der Tabelle s3 wird zum Primärschlüssel. Primärschlüssel müssen eindeutig sein und dürfen nicht NULL sein. Ein Primärschlüssel kann über eine oder mehrere Spalten angelegt werden. Ein Primärschlüssel ist in einer relationalen Datenbank wichtig, da über die Primärschlüssel Tabellen miteinander in Beziehung stehen können. So gibt es bei EXCE zum Beispiel die 1:N Beziehung **user.id – S3.user_id**, was bedeutet, dass ein Benutzer N Bilder anlegen kann.

```
ALTER TABLE `s3`  
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,  
    AUTO_INCREMENT=8;
```

Die Spalte ID in der Tabelle s3 erhält das Auto Increment Attribut. Jedes Mal, wenn jetzt eine neue Zeile eingefügt wird, wird automatisch die nächsthöhere ID vergeben. Wir können also jetzt Benutzer per INSERT einfügen, ohne das ID Feld setzen zu müssen.

```
ALTER TABLE `users`  
    ADD PRIMARY KEY (`id`, `email`),  
    ADD UNIQUE KEY `email` (`email`);  
  
ALTER TABLE `users`  
    MODIFY `id` bigint(20) NOT NULL AUTO_INCREMENT,  
    AUTO_INCREMENT=13;
```

Die Spalten ID und E-Mail werden zu Primary Keys. Die Spalte **email** wird mit UNIQUE versehen. Von jetzt an darf eine E-Mail-Adresse nur einmal

verwendet werden. Jeder weitere INSERT mit derselben E-Mail Adresse schlägt fehl.

```
COMMIT;
```

Der zugehörige COMMIT zu START TRANSACTION in Zeile 1. Konnten alle Befehle bis hierher durchgeführt werden, werden die Änderungen tatsächlich in die Datenbank übernommen, sonst passiert nichts. Dadurch ist sichergestellt, dass es entweder einen definierten Endzustand gibt oder der Ausgangszustand unverändert geblieben ist.

10.3 REST API

Für das REST API wurde die Rewrite Engine des Apache Servers benutzt, um sämtliche Anfragen auf das index.php file umzuleiten.

Die notwendigen Direktiven stehen in einer .htaccess Datei, welche im root Verzeichnis der Server-Anwendung liegen muss:

```
RewriteEngine on
RewriteCond %{REQUEST_URI} !^/api/index.php$
RewriteRule ^(.+)$ /api/index.php?url=$1 [NC,L]
```

Das ist insofern erwünschenswert, als man die API Pfade verwenden möchte, um übergebene Parameter zum Teil des Pfades zu machen.

Das index.php file ist die Landingpage des Backends. Der Apache Webserver registriert jeglichen Datenverkehr auf Port 80 und dieser wird durch das htaccess file auf das index.php File weitergeleitet. Wenn jetzt Datenverkehr registriert wird, erstellt das index.php File ein Objekt der Klasse api.php

```
$api = new Api();
```

Außerdem werden alle unsere Routes definiert. Dies geschieht durch einen Aufruf der get, put oder post methode der api Klasse. Man übergibt

der Methode den Namen und anschließend die anonyme Funktion, welche die Route ausführen soll. Der Name kann auch einen Pfad enthalten und mit Doppelpunkten können Parameter gekennzeichnet werden.

Route ohne Parameter:

```
$api->get('/hello', function ($route) {
    echo "hello world." . $route->info();
});
```

Route mit Parameter:

```
$api->get('/hello/:world', function ($route, $world) {
    echo "hello $world." . $route->info();
});
```

Die get, put und post Methoden lauft immer gleich ab. Es wird dafür immer die addRoute Funktion aufgerufen, nur dass andere Parameter übergeben werden. Durch addRoute wird ein Route-Objekt erstellt

```
private function addRoute($method, $def, $function)
{
    $route = new Route($method, $def, $function);
    array_push($this->routen, $route);
}
```

Hier ein Beispiel anhand der get und put Methode:

GET http Methode:

```
public function get($routeDef, $function)
{
    $this->addRoute('GET', $this->curPrefix . $routeDef, $function);
}
```

PUT http Methode:

```
public function put($routeDef, $function)
{
    $this->addRoute('PUT', $this->curPrefix . $routeDef, $function);
}
```

Es ist die gleiche Methode, nur dass die http Methode GET mit PUT ausgetauscht wurde.

Alle Routes, die definiert werden, werden im routes array des api Objektes abgespeichert und für alle der Routes wird ein Route-Objekt erstellt. Nach dem Erstellen des api Objektes und der Routes wird die run-Methode des api Objektes ausgeführt. In der run-Methode wird die eingegangene route, welche als requestUri gespeichert und vom Server aus einer Superglobalen Variable geliefert wurde, mit den definierten Routen verglichen.

```
$this->requestUri = $_SERVER['REQUEST_URI'];
```

Für das Vergleichen wird die matches Methode des Route-Objekts verwendet. Jede erstellte Route wird mit der vom Apache Webserver registrierten Route verglichen. Wenn die Methode nicht übereinstimmt, das heißt, wenn das Route-Objekt von der Methode PUT ist und der Request zum Beispiel von der Methode GET, wird sofort false retuniert. Das heißt, dass es sich nicht um die richtige Route handeln kann.

```
function matches()
{
    if ($this->method !== $this->requestMethod) {
        return false;
}
```

Ist die definition (der Name) gleich wie der request URL, wird sofort true returniert. Das trifft zum Beispiel zu, wenn eine Route ohne Parameter aufgerufen wurde.

```
// instant match e.g /api/users <=> /api/users
if ($this->def === $this->requestUri) {
    return true;
}
```

Danach wird die Route auf Parameter überprüft, um diese im Falle des Vorhandenseins zu speichern. Die Route wird in ein Array gesplittet. Danach wird ein neues Array namens Params erstellt, in dem im Falle des Vorhandenseins eines Parameters alles gespeichert werden kann. Mit einer for Schleife werden alle Teile der zu vergleichenden Routen

überprüft. Ist ein Teil ungleich, wird überprüft, ob ein „:“ am Anfang des Strings steht, da dieser einen Parameter angeibt. Ist dies nicht der Fall, wird false returniert. Ist der Doppelpunkt vorhanden, wird der Wert ins param Array gespeichert.

```
$partsDef = explode('/', $this->def);
// try to find match like /api/user/1 <=> /api/user/:id
if (count($this->curRoute) === count($partsDef)) {
    $this->params = array();
    $routeMatches = true;
    for ($l = 0; $l < count($this->curRoute); $l++) {
        $p0 = $this->curRoute[$l];
        $p1 = $partsDef[$l];
        if ($p0 === $p1) {
            //match
            continue;
        } else if (str_starts_with($p1, ':')) {
            // part is a parameter
            array_push($this->params, $p0);
        } else {
            // mismatch
            $routeMatches = false;
        }
    }
    if ($routeMatches === true) {
        return true;
    }
}
return false;
```

Wenn eine matchende Route gefunden wird, wird die Funktion, welche zur matchenden Route passt, von der run Funktion aufgerufen. So kommunizieren das Frontend und das Backend miteinander.
Die PHP Anwendung verwendet den Composer Dependency Manager, um zusätzliche Pakete zu inkludieren.

Vlucas/phpdotenv:

```
vvlucas/phpdotenv
```

Dieses Paket erlaubt es .env Dateien zur Laufzeit zu lesen und die Werte in superglobalen Variablen zur Verfügung zu stellen.

Auf diese Weise ist sichergestellt, dass keine Verbindungsdaten oder Kennworte jemals im Source code eingecheckt werden. Es wird im Git ein .env.schema File angelegt, welches vorgibt, welche Schlüssel im .env file vorhanden sein müssen. Dann wird ein .env file mit diesen Schlüsseln angelegt und mit den notwendigen Werten befüllt. Dieses .env file kommt nicht in die Versionsverwaltung (Siehe Kapitel Git & GitHub):

```
S3_BUCKET="mopa-exce"  
S3_KEY="AKIA2JP....."  
S3_SECRET="3DWhR....."  
S3_REGION="eu-west-1"  
SECRET_KEY="souper_seekret_key"  
DB_SERVER="localhost"  
DB_USER="exce"  
DB_PASS="zrnac5v!"  
DB_NAME="exce"
```

Im Code können wir schreiben:

```
$this->mysqli = new mysqli($_ENV['DB_SERVER'], $_ENV['DB_USER'],  
$_ENV['DB_PASS'], $_ENV['DB_NAME']);
```

Um uns mit der Datenbank zu verbinden, installiert man die Anwendung auf einem anderen Server. Es müssen nur die .env Files aktualisiert werden. So erscheinen die Passwörter nicht im Code, weil nur das Schema File und nicht das .env File in der Versionskontrolle ist.

[aws/aws-sdk-php](#):

```
aws/aws-sdk-php
```

Amazon stellt uns hier ein fertiges Paket für Zugriffe auf AWS zur Verfügung. Wir verwenden es, um Bilder in der S3 zu speichern und von dort bei Bedarf wieder zu laden.

10.4 Routen

Die oben Bereits angesprochenen definierten Routen lauten wie folgt:

10.4.1 Register Route:

Die Register Route erstellt zuerst eine Hilfsvariable der Klasse-User. Dann wird die register Methode der Klasse-User verwendet.

```
$api->post('/register', function ($route) {
    $users = new Users();
    $body = $route->postBody();
    $parts = explode(PHP_EOL, $body);
    $username = $parts[0];
    $password = $parts[1];
    $user = $users->register($username, $password);
    if ($user !== null) {
        echo json_encode($user);
    }
});
```

In der Register Funktion wird, falls die angegebene E-Mailadresse noch nicht vergeben ist, ein neuer User angelegt.

```
function register($email, $pass)
{
    if ($this->userByEmail($email)) {
        return ErrorCode::USER_ALREADY_TAKEN;
    } else {
        return $this->createUserByEmail($email, $pass);
    }
}
```

Die Funktion userByEmail überprüft hierbei, ob die E-Mailadresse des Users schon vergeben ist.

```
private function userByEmail($email)
{
    $rows = array();

    $query = "SELECT * FROM `users` WHERE `email`='$email'";

    $result = mysqli_query($this->mysqli, $query);
    $num_results = mysqli_num_rows($result);

    If (num_results > 0) {
        Return mysqli_fetch_assoc($result);
    }
}
```

```

        return null;
}

```

Wenn die E-Mail noch nicht vorhanden ist, wird die createUserByEmail Funktion verwendet. Der User Name des Accounts wird bei dieser App automatisch aus der E-Mailadresse generiert.

```

private function createUserByEmail($email, $pass)
{
    $parts = explode("@", $email);
    $username = $parts[0];
}

```

Danach wird ein User in der Datenbank hinzugefügt und automatisch eingeloggt.

```

$query = "INSERT INTO users (`email`, `password`, `username`) VALUES ('$email',
'$pass', '$username')";
if (mysqli_query($this->mysqli, $query)) {
    return $this->login($email, $pass);
}
return ErrorCode::UNABLE_TO_CREATE_USER;
}

```

Das Einloggen passiert hier mithilfe der Login Funktion. Diese bekommt die E-Mail und den User vom Frontend übergeben, danach wird überprüft, ob das Passwort des Users mit dem übergebenen Passwort übereinstimmt. Ist das nicht der Fall, wird ein Fehler zurückgegeben.

Auch wenn die E-Mail Adresse nicht gefunden wird - die Datenbank wird auf diese wieder mit der userByEmail Funktion überprüft - wird ein Fehler retourniert.

```

function login($email, $pass)
{
    $user = $this->userByEmail($email);
    if ($user) {
        if ($user['password'] !== $pass) {
            return ErrorCode::WRONG_PASSWORD;
        }
        return $this->updateUserToken($user);
    } else {
        return ErrorCode::USER_NOT_FOUND;
    }
}

```

10.4.2 Login Route:

Die Login Route postet einen Benutzernamen und das Passwort zum Server. Dafür wird wieder die login Funktion, welche auch schon bei der Register Route verwendet wurde, ausgeführt.

```
$api->post('/login', function ($route) {
    $users = new Users();
    $body = $route->postBody();
    $parts = explode(PHP_EOL, $body);
    $username = $parts[0];
    $password = $parts[1];
    $user = $users->login($username, $password);
    if ($user !== null) {
        echo json_encode($user);
    }
});
```

10.4.3 Logoff Route:

Die Logoff Route verwendet den User Token, um den User abzumelden.

```
$api->get('/logoff/:token', function ($route, $token) {
    $users = new Users();
    $user = $users->logoff($token);
    if ($user !== null) {
        echo json_encode($user);
    }
});
```

Dafür wird die logoff Funktion verwendet. Diese bekommt den token übergeben und wählt in der Datenbank den dazu passenden User aus.

```
function logoff($token)
{
    $rows = array();

    $query = "SELECT * FROM `users` WHERE `token`='$token'";
```

Dieser wird dann abgemeldet. Das heißt, der Token wird aus der Datenbank gelöscht.

```
$result = mysqli_query($this->mysqli, $query);
$num_results = mysqli_num_rows($result);

if ($num_results === 1) {
```

```

        $row = mysqli_fetch_assoc($result);
        $id = $user["id"];
        $query = "UPDATE users SET last_login=NULL,token=NULL WHERE id=$id";
        mysqli_query($this->mysql, $query);
        return $user;
    }

    return null;
}

```

10.4.4 Upload Picture Route:

Die Upload Picture Route ist dafür zuständig, Bilder in einen AWS s3 Bucket hochzuladen. Der Vorteil an s3 ist, dass die Bilder nicht direkt im Backend gespeichert werden und somit der Server nicht übermäßig viele Daten speichern muss. Es wird zuerst der User, welcher die Route aufruft, über den Usertoken evaluiert, wird er nicht gefunden, wird ein Fehler geworfen.

```

$api->post('/s3/:token/picture', function ($route, $token) {
    $users = new Users();
    $user = $users->getUserOrDie($token);
}

```

Danach wird ein key erstellt. Dieser ist immer einzigartig, weil er die User ID und den Zeitstempel beinhaltet. Eine Hilfsobjekt der Klasse Files wird erstellt, um die Funktion uploadToBucket auszuführen.

```

// files will be like
// 1707744078.8.png
// UNIX TIMESTAMP | USER ID | EXTENSION
$key = sprintf("%d.%d.png", time(), $user['id']);
$files = new Files();
$files->uploadToBucket($user, $key, $route->postBody());
});

```

Die uploadToBucket Funktion bekommt den user (User ID) den key(Dateiname) und den postbody (Datei Inhalt in base64) übergeben. Im Moment werden nur Bilder vom Format .png unterstützt. Das erkennt man an der hardcodierten Variable \$signature, welche für base64 codierte

Bilder vorbereitet ist. Mit der Funktion str_starts_with wird der start des base64 codierten Bodys mit der Signatur verglichen.

```
function uploadToBucket($user, $key, $body)
{
    $signature = 'data:image/png;base64,';

    if (str_starts_with($body, $signature)) {
        $body_b64 = str_replace($signature, '', $body);
```

Die Signatur wird vom body entfernt, damit nur die base64 Daten übrigbleiben. Diese werden in Binär decodiert und mit s3client->putObject der AWS Librar in den Bucket hochgeladen. Durch Deklaration des ContentType als Image weiß s3, dass es sich um ein Bild handelt.

```
$binary = base64_decode($body_b64);
$contents = $this->s3client->putObject([
    'Bucket' => $_ENV['S3_BUCKET'],
    'Key' => $key,
    'ContentType' => 'image',
    'Body' => $binary
]);
```

Der Rückgabewert des Aufrufes wird in contents gespeichert und im Feld ObjectURL des Ergebnis-Objektes findet man den URL des s3 Objektes (+Bild). Mit diesem URL kann das Bild heruntergeladen werden. In der Datenbank werden nun der URL, die User ID und der Filename gespeichert. Zusätzlich wird ein Zeitstempel generiert, damit nachvollziehbar ist, wer wann etwas gepostet hat. Die Spalte created wird also mit current_timestamp() deklariert und kann somit leer bleiben.

```
$s3url = $contents['ObjectURL'];
$userId = $user['id'];

$query = "INSERT INTO s3 (`name`, `user_id`, `s3key` ) VALUES ('$key',
$userId, '$s3url')";
if (mysqli_query($this->mysql, $query)) {
    $id = mysqli_insert_id($this->mysql);
    echo ($id);
}
}
```

11 Fazit

Abschließend möchte ich Folgendes festhalten. Durch die Ausführung der Arbeit wurde mir bewusst, wie komplex das Entwickeln von Social Media-Plattformen ist. In der Arbeit wurden bei weitem nicht alle Themen behandelt. Es fehlen wichtige Themen für Plattformen, wie z.B. die Skalierung, die Wartung und die Bewerbung.

Durch die vielen frei zugänglichen Entwicklungstools am Markt wird es Entwicklern leicht gemacht, an neuen Ideen zu arbeiten. Es geht sehr schnell, einen funktionierenden Prototypen zu entwickeln. Die Herausforderung steckt in der neuartigen Idee, der Umsetzung und Weiterentwicklung im laufenden Betrieb, um am Markt zu bestehen. Um dies leisten zu können, wird ein Team an Softwareentwicklern benötigt.

Ich möchte an dieser Stelle das Paretoprinzip, benannt nach Vifredo Pareto, auch Pareto Effekt oder 80 zu 20 –Regel genannt, erwähnen. Es besagt, dass 80% der Ergebnisse mit 20% des Gesamtaufwands erreicht werden. Die verbleibende 20% der Ergebnisse erfordern mit 80% des Gesamtaufwands die meiste Arbeit.

Dieser Effekt führt oft zu Selbstüberschätzung in Projekten, da durch die modernen Entwicklungsumgebung schnelle Ergebnisse sichtbar sind, die aber noch lange nicht die Marktreife erfüllen können.

12 Verzeichnisse

12.1 Quellenverzeichnis

12.1.1 Kapitel 8:

- [https://www.computerweekly.com/de/definition/GitHub#:~:text=Es%20erm%C3%B6glicht%20eine%20effektivere%20Zusammenarbeit,passen%20sie%20an%20und%20verbessernsie.](https://www.computerweekly.com/de/definition/GitHub#:~:text=Es%20erm%C3%B6glicht%20eine%20effektivere%20Zusammenarbeit,passen%20sie%20an%20und%20verbessernsie)
- <https://www.ionos.at/digitalguide/websites/web-entwicklung/git-push/#:~:text=Der%20Befehl%20Git%20Push%20l%C3%A4dt,Branche%20als%20Ziel%20ausgef%C3%BCrtenwerden>.
- <https://www.atlassian.com/git/tutorials/syncing/git-push>
- <https://nvie.com/posts/a-successful-git-branching-model/>
- <https://www.dhiwise.com/post/how-to-remove-debug-banner-in-flutter-a-comprehensive-guide>
- <https://docs.github.com/de/repositories/creating-and-managing-repositories/cloning-a-repository>
- <https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/github-clone-with-ssh-keys>
- <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>
- <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>
- <https://docs.aws.amazon.com/linux/al2023/ug/ec2-lamp-amazon-linux-2023.html>

12.1.2 Kapitel 9

- <https://www.objectbay.com/blog/flutter-die-zukunft-der-mobile-app-entwicklung#:~:text=Einer%20der%20gr%C3%B6%C3%9Ften%20Vorteile%20von,somit%20ideal%20f%C3%BCr%20hybride%20Appentwicklung.>
- <https://docs.flutter.dev/ui/interactivity#:~:text=Stateful%20and%20stateless%20widgets,-A%20widget%20is&text=If%20a%20widget%20can%20change,Stateless%20widgets%20subclass%20 StatelessWidget%20.>
- <https://docs.flutter.dev/>

12.1.3 Kapitel 10

- <https://www.php.net/docs.php>
- <https://nodejs.org/en/docs/>

12.2 Bilderverzeichnis

- Abbildung 7.1: Flutter Widget Tree
- Abbildung 8.1: Git Branches
- Abbildung 8.2 - 8.4: Repo erstellen Schritt 1-3
- Abbildung 8.5 - 8.9: SSH Key in GitHub hinzufügen Schritt 1-5
- Abbildung 8.10 - 8.11: Repository klonen Schritt 1-2
- Abbildung 8.12 - 8.15: Aufsetzen der EC2 Instanz Schritt 1-4
- Abbildung 8.16: Testen des LAMP Servers, PHP Infoscreen
- Abbildung 8.17: iPhone Simulator
- Abbildung 8.18 - 8.19: Device zum compilieren auswählen Schritt 1-2
- Abbildung 9.1: Zusammenspiel der Klassen Login/Registrierungs Prozess
- Abbildung 9.2: Zusammenspiel der Klassen MainPage & Handler
- Abbildung 9.3: Login Page
- Abbildung 9.4: Registration Page
- Abbildung 9.5: Profile Page
- Abbildung 9.6: Settings Page
- Abbildung 9.7: Map Page
- Abbildung 9.8: Map Page Markers
- Abbildung 9.9: Upload Page
- Abbildung 9.10: Display Picture Page