HTL Leonding
next level

HÖHERE TECHNISCHE
BUNDESLEHRANSTALT
LEONDING

AberGymMobile

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Eingereicht von:
Antonio Peric
Antonio Kuvac

Betreuer:
Prof. Mag. Ing. Hans Christian Hammer

Projektpartner:
DI. Christian Aberger

Leonding, April 2023

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

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

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2023 Peric Antonio & Kuvac Antonio

Abstract

This diploma thesis aims to modernize the training plan process at the LionFit gym. Currently, training plans are created on a web application and printed as a PDF, and training data is recorded on the printed PDF. The project involves developing a native app using Android Studio and Flutter to provide a digital solution for managing and tracking training plans. The app will read data from the training plan database and update it with training data after each session, streamlining the entire process and making it more efficient and convenient for the gym's customers.

Zusammenfassung

Diese Diplomarbeit zielt darauf ab, den Trainingsplanprozess im Fitnessstudio LionFit zu modernisieren. Derzeit werden Trainingspläne in einer Web-Applikation erstellt und als PDF ausgedruckt, während die Trainingsdaten auf dem ausgedruckten PDF erfasst werden. Das Projekt beinhaltet die Entwicklung einer nativen App mit Android Studio und Flutter, um eine digitale Lösung für die Verwaltung und Überwachung von Trainingsplänen zu bieten. Die App wird Daten aus der Trainingsplandatenbank lesen und sie nach jeder Sitzung mit Trainingsdaten aktualisieren, um den gesamten Prozess zu vereinfachen und für die Kunden des Fitnessstudios effizienter und bequemer zu gestalten.

Danksagung

An dieser Stelle möchten wir uns bei den Personen bedanken, die uns während dieser Diplomarbeit unterstützt und motiviert haben. Gleich zu Beginn möchten wir uns gebührend bei Herrn Prof. Ing. Mag. Hans Christian Hammer bedanken, welcher uns immer mit seiner konstruktiven Kritik zur Seite stand.

Wir bedanken uns auch bei unseren Kollegen der 5CHIF, die uns stets mit ihrer Hilfsbereitschaft, und mit ihrem Wissen weitergeholfen haben.

Außerdem möchten wir uns bei unserem Projektpartner, DI. Christian Aberger bedanken, für die Bereitstellung von Ressourcen, und für die Beratung bei Fragen über unsere Diplomarbeit, und wie wir diese umsetzen sollen.

Abschließend möchten wir uns noch bei unseren Eltern bedanken, die uns während unserer Schulkarriere in der Höheren Technischen Bundeslehranstalt Leonding begleitet haben, und immer ein offenes Ohr für unsere Belange hatten.

Inhaltsverzeichnis

1 Pflichtenheft **1**

1.1 Über das Team 1

1.2 Bertreuer und Partner 1

1.3 Ausgangssituation und Zielstellung 1

1.4 Zielgruppe 4

1.5 Funktionale Anforderungen 4

1.6 App 5

2 Planung **7**

2.1 Projektpartnermeetings 7

2.2 Use-Case Diagramm 7

2.3 Entity Relationship Diagram - ERD 9

2.4 Design Thinking 12

2.5 Brain Storming 13

2.6 Kopfstand Methode 14

2.7 Mockups 15

3 Technologien **26**

3.1 Flutter 26

3.2 Dart als Programmiersprache 29

3.3 Visual Studio Code 32

3.4 Docker 33

3.5 Android Studio 34

3.6 AdobeXD 37

3.7 IntelliJ IDEA 38

4 Flutter-App **42**

4.1 Start-Screen 42

4.2 Login 47

4.3 Hauptmenü 55
4.4 To-Do-Liste 59
4.5 Satzzähler 62
4.6 Trainingplan bearbeiten 65

5 Backend **71**

5.1 Temporäres Backend für die Mobile Anwendung 71
5.2 Integration des echten Backends 87

Literaturverzeichnis **VII**

Abbildungsverzeichnis **VIII**

Tabellenverzeichnis **IX**

Quellcodeverzeichnis **X**

1 Pflichtenheft

1.1 Über das Team

Tabelle 1: *Tabelle 1.1: Informationen über den Bertreuer und Partner.*

Bertreuer	Prof. Mag. Ing. Hans Christian Hammer
Partner	DI. Christian Aberger

1.2 Bertreuer und Partner

Tabelle 2: *Tabelle 1.2: Informationen über das Projekt und das Team.*

Projektname	AGM - Abergymmobile
Teamleiter	Antonio Kuvac
Teammitglieder	Antonio Kuvac, Antonio Peric
Erstellt am	12.7.2023

1.3 Ausgangssituation und Zielstellung

1.3.1 Ausgangssituation

Aktuell ist der Prozess der Trainingsplangenerierung und -verwaltung im Fitnessstudio LionFit ineffizient und umständlich für Kundinnen und Kunden. Daher möchte das Fitnessstudio eine digitale Lösung implementieren, um den Prozess zu vereinfachen und die Effizienz zu steigern. Derzeit werden die Trainingspläne in einer Web-Applikation erstellt und als PDF ausgedruckt, während die Trainingsdaten auf dem ausgedruckten PDF erfasst werden. Dieser Prozess ist unpraktisch, da die Kundinnen und Kunden bei jeder Trainingseinheit ein ausgedrucktes PDF mitnehmen und die Daten manuell eingeben müssen.

1.3.2 Zieldefinition

Die Entwicklung einer nativen App für Android-Geräte zur Umsetzung von Trainingsplänen in einem Fitnessstudio gewinnt zunehmend an Bedeutung. Diese App ermöglicht es den Nutzer*innen, ihre individuellen Trainingspläne effektiv und effizient abzuarbeiten,

Die Trainingsplanverwaltung stellt sicher, dass die Daten des Trainingsplans aktuell und auf die Bedürfnisse der*die Nutzer*innen zugeschnitten sind. Durch die Anbindung der App an die Trainingsplanverwaltung wird es ermöglicht, die relevanten Daten direkt auf dem Android-Gerät der*die Nutzer*innen abzurufen. Dies erleichtert die Organisation und Durchführung des Trainings, indem es den Nutzer*innen erlaubt, auf ihre individuellen Trainingspläne in Echtzeit zuzugreifen.

Am Ende einer Trainingssession werden die Trainingsdaten, wie beispielsweise die Anzahl der Wiederholungen, die Gewichte oder die Trainingsdauer, in die Trainingsdatenbank übertragen und gespeichert. Dies ermöglicht eine kontinuierliche Analyse und Anpassung der Trainingspläne, um eine optimale Unterstützung der*die Nutzer*innen in ihrer sportlichen Entwicklung zu gewährleisten.

Durch die Integration von Training und Datenverwaltung in einer nativen Android-App werden Fitnessstudios in die Lage versetzt, ein benutzerfreundliches, modernes und zielgerichtetes Trainingsumfeld für ihre Mitglieder*innen zu schaffen. Das trägt zur Steigerung der Motivation und der Erfolgschancen bei, da individuelle Trainingsziele leichter erreicht werden können.

1.3.3 Nicht Ziele

Die Entwicklung einer App für Trainingspläne in Fitnessstudios birgt auch gewisse Herausforderungen und Risiken. Eine davon ist die Gefahr, eine zu komplex gestaltete App zu entwickeln, die den Nutzer*innen Schwierigkeiten bereitet und sie dazu veranlasst, stattdessen auf den traditionellen Zettel-Trainingsplan zurückzugreifen. Um dem entgegenzuwirken, sollte die App intuitiv und benutzerfreundlich gestaltet sein, sodass sie die Bedürfnisse der*die Nutzer*innen erfüllt und gleichzeitig den Trainingsprozess vereinfacht.

Ein weiteres Risiko besteht in der Entwicklung einer fehlerhaften App, die den Nutzer*innen Unannehmlichkeiten bereitet und ihre Trainingserfahrung beeinträchtigt. Um dies zu vermeiden, ist es wichtig, die App sorgfältig zu testen und mögliche Fehlerquellen frühzeitig zu identifizieren. Die Qualitätssicherung und regelmäßige Aktualisierung der App sind entscheidend für ihren Erfolg.

Schließlich kann auch das Design der App einen bedeutenden Einfluss auf die Akzeptanz bei den Nutzer*innen haben. Eine App, deren Design die Zielgruppe nicht anspricht, könnte weniger erfolgreich sein und das Potenzial der digitalen Trainingsplanunterstützung ungenutzt lassen. Daher ist es ratsam, bei der Gestaltung der App auf ansprechende und funktionale Designelemente zu achten, die die Nutzer*innen ansprechen und zum wiederholten Gebrauch motivieren.

Die Berücksichtigung dieser Herausforderungen bei der Entwicklung einer Trainingsplan-App ist essentiell, um eine positive Benutzererfahrung zu gewährleisten und den Nutzer*innen eine effektive und ansprechende Alternative zum traditionellen Zettel-Trainingsplan zu bieten.

1.4 Zielgruppe

Diese Zielgruppe umfasst sowohl regelmäßige Fitnessstudio-Besucher*innen als auch Sportler*innen, die unabhängig von einem Fitnessstudio trainieren und eine digitale Lösung für die Verwaltung ihrer Trainingspläne suchen. Besonders praktisch für diese Zielgruppe ist, dass sie jederzeit Zugang zu ihren Trainingsplänen und -daten auf ihrem Smartphone haben. Dies bietet mehr Flexibilität und Übersicht bei der Gestaltung und Überwachung des Trainings. Außerdem müssen die Kunden*innen nicht mehr auf ausgedruckte Trainingspläne zurückgreifen und können stattdessen auf eine sichere und zuverlässige digitale Lösung setzen.

1.5 Funktionale Anforderungen

Die Benutzeroberfläche der nativen App sollte einfach und ansprechend gestaltet sein, um auch Personen, die nicht besonders computeraffin sind, eine leichte Handhabung zu ermöglichen. Darüber hinaus soll die Modernisierung des Trainingsplanprozesses die Kommunikation zwischen der Verwaltung und den Koordinatoren des Fitnessstudios LionFit vereinfachen und automatisieren.

1.5.1 An die App

Die effektive Nutzung einer Trainingsplan-App erfordert eine sorgfältige Planung und Umsetzung verschiedener Funktionen. Eine Möglichkeit, die Nutzer*innen in ihrem Trainingsprozess zu unterstützen, besteht darin, die Durcharbeitung der Trainingspläne als To-Do-Liste zu gestalten. Dies ermöglicht es den Nutzer*innen, ihre Fortschritte während des Trainings klar zu erkennen und ihre Motivation aufrechtzuerhalten.

Nach Abschluss jeder Übung sollte die App die Möglichkeit bieten, den Trainingsplan zu überprüfen und gegebenenfalls anzupassen. Dies ermöglicht eine individuelle Anpassung des Trainings und stellt sicher, dass die Nutzer*innen stets auf dem aktuellen Stand ihrer Trainingsziele sind. Die flexible Anpassung des Trainingsplans trägt dazu bei, die Effektivität des Trainings zu erhöhen und den Bedürfnissen der*die Nutzer*innen gerecht zu werden.

Die Verfügbarkeit des überarbeiteten Trainingsplans zu jeder Zeit ist ein weiterer wichtiger Aspekt, der es den Nutzer*innen ermöglicht, ihre Trainingspläne erneut durchzuarbeiten und an ihren Zielen kontinuierlich zu arbeiten. Durch die ständige Verfügbarkeit der aktualisierten Trainingspläne können die Nutzer*innen ihre Trainingsfortschritte effizient verfolgen und die erforderlichen Anpassungen vornehmen.

Schließlich sollte die App auch eine Historie der alten Trainingspläne aufbewahren, um den Nutzer*innen einen Überblick über ihre Trainingsentwicklung und die Möglichkeit, auf vergangene Trainingspläne für zukünftige Referenzen zuzugreifen, zu bieten. Dies kann als wertvolles Instrument für die Selbstreflexion und Analyse des Trainingsfortschritts dienen.

1.6 App

1.6.1 Allgemeine Beschreibung

Mit unserer innovativen mobilen Anwendung bieten wir Ihnen die Möglichkeit, Ihre individuellen Trainingspläne bequem und effizient über Ihr Smartphone abzuarbeiten. Die Anwendung präsentiert den Nutzer*innen die Trainingspläne in Form einer leicht verständlichen To-Do-Liste, die eine strukturierte und systematische Durchführung jeder Übung gewährleistet. Dies fördert die Trainingsdisziplin und unterstützt die Nutzer*innen dabei, ihre persönlichen Trainingsziele zu erreichen.

Nach Abschluss jeder Übung haben die Nutzer*innen die Möglichkeit, ihren Trainingsplan zu überprüfen und gegebenenfalls anzupassen. Diese Flexibilität ermöglicht es, das Training kontinuierlich an die individuellen Bedürfnisse und Fortschritte der*die Nutzer*innen anzupassen, um eine optimale Trainingsgestaltung sicherzustellen.

Der überarbeitete Trainingsplan ist jederzeit für die Nutzer*innen zugänglich, sodass sie ihn bei Bedarf erneut durcharbeiten können. Dies gewährleistet eine hohe Verfügbarkeit der Trainingsinformationen und erleichtert die Planung und Durchführung des Trainings im Alltag der*die Nutzer*innen.

Darüber hinaus werden alte Trainingspläne in der Anwendungshistorie gespeichert, sodass sie bei Bedarf als Referenz herangezogen werden können. Die Archivierung der Trainingshistorie ermöglicht den Nutzer*innen, ihren Trainingsfortschritt im Laufe der Zeit zu analysieren und eventuelle Anpassungen oder Modifikationen ihrer Trainingsziele vorzunehmen.

2 Planung

Die erste und auch längste Phase dieses Projekts war die Planung. In der Planungsphase wurden verschiedenste Kreativitätstechniken angewandt und Dokumente, die für die Planung unerlässlich sind, erstellt.

2.1 Projektpartnermeetings

Der Grundstein für diese Arbeit waren die abgehaltenen Meetings mit dem Projektpartner dieses Projekts. Bei den ersten Meetings wurde definiert, was die App können muss und wie die App auszusehen hat um Missverständnisse im voraus zu behandeln und um sich ein Bild darüber zu machen, welche Anforderungen das Projekt hat und welche Herausforderungen zu bewältigen sind. Mit diesen Meetings wurde der Projektpartner auch auf aktuellen Stand gehalten und auftretende Probleme und Anmerkungen, die bei der Arbeit am Projekt entstanden sind, wurden besprochen und abgehandelt.

2.2 Use-Case Diagramm

Ein Use Case Diagramm oder auch Anwendungsfalldiagramm ist ein Diagramm von Anwendungsfällen und ihren Beziehungen zu ihrer Umgebung und zu anderen Anwendungsfällen. Damit beschreibt es, welche Funktionen und Dienste ein System für einen Anwender bereitstellt. Mit einem Use Case Diagramm werden weder die Abläufe des Systems beschrieben, noch die Reihenfolge der Funktionen oder Dienste dargestellt. Ein Anwendungsfalldiagramm visualisiert lediglich die Zusammenhänge zwischen einer Menge von Use Cases und den involvierten Akteuren. Damit eignet es sich sehr gut zur Anforderungsanalyse, also zur Ermittlung oder Verfeinerung von Anforderungen.

2.2.1 Elemente im Use-Case Diagramm

Folgende Elemente sind in einem Use-Case Diagramm üblicherweise enthalten:

- System: Das System ist an sich kein logisches Modellelement, sondern grenzt den Kontext ab. Das System wird durch eine Box im Diagramm dargestellt, in dem das System seine Funktionen und Dienste zur Verfügung stellt.
- Akteur: Der Akteur befindet sich außerhalb des Systems. Er wird als Strichmännchen gezeichnet und kann eine konkrete Person aber auch ein abstraktes Element wie zum Beispiel ein Sensor sein. Wird ein Akteur definiert, muss dieser auch immer mit mindestens einem Use Case in Verbindung stehen.
- Anwendungsfall: Ein Anwendungsfall wird meistens als Ellipse visualisiert. Ein Diagramm kann mehrere Anwendungsfälle besitzen. Ist ein Use Case nur durch andere Anwendungsfälle ausführbar, wird er als abstrakt bezeichnet.
- Beziehungen: Zwischen Akteuren und Anwendungsfällen oder zwischen Anwendungsfällen selbst bestehen Beziehungen. Sie werden meistens als Linien oder Pfeile zwischen den jeweiligen Objekten eingezeichnet.
- Notizen: Mit Notizen lassen sich Informationen hinzufügen, um das Verständnis zu erhöhen. Sie werden mit einem Rechteck dargestellt, dessen obere rechte Ecke eingeknickt ist. Eine gestichelte Linie verbindet die Notiz mit dem zu erklärenden Element.

2.2.2 Verwendung des Use-Case Diagramms in der Diplomarbeit

Das Use Case Diagramm wurde verwendet um Klarheit über die Funktionen und den Akteuren zu schaffen und für Vergewisserung zu sorgen, dass keine wichtigen Verbindungen vergessen wurden. Am Ende der Planungsphase angelangt sieht das Use-Case Diagramm wie folgt aus:

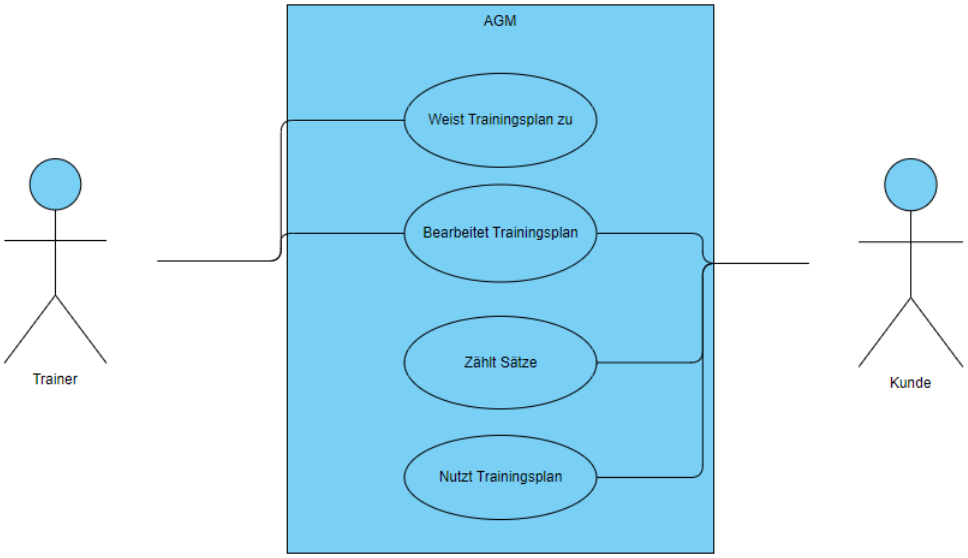

Abbildung 1: Use-Case Diagramm

2.3 Entity Relationship Diagram - ERD

Ein ERD ist ein Diagramm der Beziehungen (Relationship) zwischen Entitäten (Entity) in einem Datenbankschema darstellt. Es wird verwendet, um das Design einer Datenbank zu modellieren und darzustellen und wie Daten in der Datenbank organisiert und miteinander verbunden sind.

2.3.1 Komponenten eines ERDs

Üblicherweise bestehen ERD-Diagramme aus drei Hauptkomponenten, und zwar aus Entitäten, Attributen und Beziehungen.

- Entitäten: Eine Entität repräsentiert eine Klasse von Objekten, die bestimmte Eigenschaften oder Attribute gemeinsam haben. Ein Beispiel für eine Entität wäre zum Beispiel ein Kunde der als Attribute einen Namen und eine Wohnadresse hat. Entitäten werden meistens als Rechtecke dargestellt
- Attribute: Attribute sind die Eigenschaften, die die Entitäten beschreiben, wie beispielsweise der Name eines Kunden oder die Menge eines Produkts. Attribute werden meistens durch Ellipsen dargestellt können aber auch in der Entität selbst drinnen stehen.
- Beziehungen: Beziehungen beschreiben die Art und Weise, wie Entitäten miteinander in Verbindung stehen. Man unterscheidet zwischen drei Arten von Beziehungen. Die erste ist One-to-One. Diese Beziehung tritt auf, wenn jeder Datensatz in der ersten Tabelle nur einen entsprechenden Datensatz in der zweiten Tabelle hat und umgekehrt. Ein Beispiel dazu wäre eine Person und sein Reisepass, denn jede Person hat nur einen Reisepass und ein Reisepass gehört nur zu einer Person. Die zweite Art ist One-to-Many. Diese Beziehung tritt auf, wenn ein Datensatz in der ersten Tabelle mehrere Datensätze in der zweiten Tabelle hat. Ein Beispiel dafür ist eine Mutter die mehrere Kinder hat, aber jedes Kind hat nur eine leibliche Mutter. Die Letzte Art von Beziehung ist Many-to-Many.Diese Beziehung tritt auf, wenn ein Datensatz in der ersten Tabelle mehrere Datensätze in der zweiten Tabelle hat, aber auch umgekehrt. Ein Beispiel für diese Beziehung ist ein Student der mehrere Kurse hat und ein Kurs der mehrere Studenten hat.

2.3.2 Verwendung des Entity Relationship Diagrams in der Diplomarbeit

Um eine funktionierende Datenbank für das Backend zu gewährleisten wurde in Zusammenarbeit mit den Mitgliedern vom Projekt AberGym mithilfe eines ERDs eine Datenbankstruktur erstellt die sowohl für die Anforderungen von AberGym als auch für die Anforderungen von AberGymMobile angepasst ist.

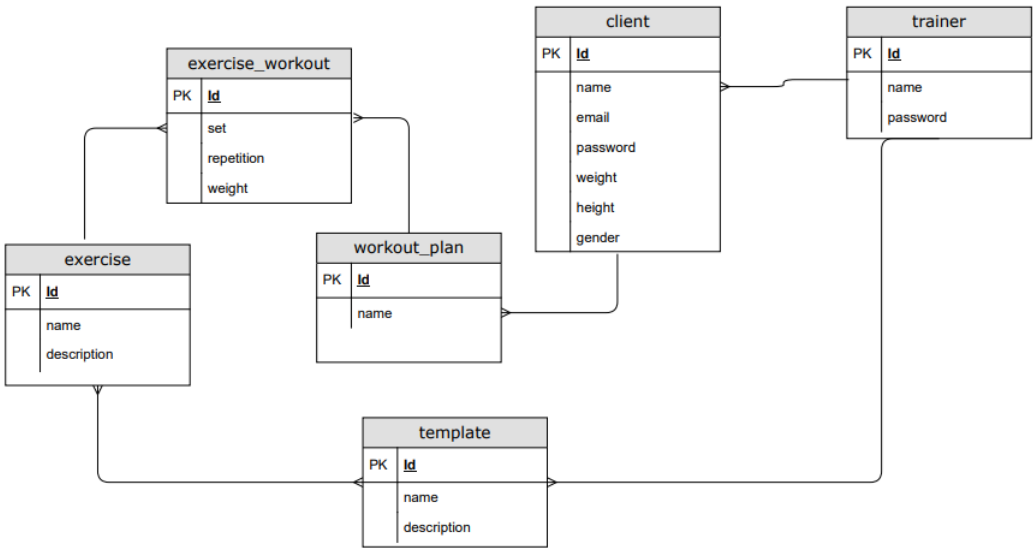

Abbildung 2: ERD Diagramm in PlantUml

2.3.3 PlantUml

PlantUML ist ein Open-Source-Tool zur Erstellung von UML-Diagrammen, das dabei unterstützt, Software-Architekturen, Systemdesigns und Prozessabläufe auf einfache Weise zu visualisieren. Es verwendet eine textbasierte Syntax, um Diagramme zu erstellen, die leicht verständlich, modifizierbar und in verschiedenen Formaten exportierbar sind. In diesem Text werden die wichtigsten Merkmale von PlantUML, seine Anwendungsbereiche sowie einige nützliche Ressourcen vorgestellt.

2.4 Design Thinking

Design Thinking ist eine innovative Kreativitätstechnik, die in den letzten Jahren immer mehr an Bedeutung gewonnen hat. Diese Methode verbindet kreative und analytische Ansätze, um komplexe Probleme zu lösen und innovative Lösungen zu entwickeln. Design Thinking fördert dabei ein tiefes Verständnis für die Bedürfnisse der Nutzer*innen und stellt diese in den Mittelpunkt des Entwicklungsprozesses.

2.4.1 Phasen des Design Thinking

Der Design-Thinking-Prozess besteht aus mehreren Phasen, die iterativ und flexibel durchlaufen werden. Diese Phasen sind:

- **Verstehen:** In dieser Phase geht es darum, das Problem, die Herausforderung oder den Kontext zu erfassen und ein tiefes Verständnis für die Bedürfnisse der Nutzer*innen zu entwickeln.
- **Beobachten:** Durch direkte Beobachtungen, Interviews und andere qualitative Methoden Informationen und Empathie für die Zielgruppe sammeln.
- **Sichten:** Die gesammelten Informationen werden analysiert, um Muster und Zusammenhänge zu erkennen. Hier werden auch Problemstellungen und Fragestellungen definiert.
- **Ideenfindung:** In dieser Phase werden verschiedene Kreativitätstechniken eingesetzt, um möglichst viele und vielfältige Ideen zu generieren.
- **Prototyping:** Die entwickelten Ideen werden in Form von einfachen und kostengünstigen Prototypen umgesetzt, um sie anschließend testen zu können.
- **Testen:** Die Prototypen werden mit der Zielgruppe getestet, um Feedback zu erhalten und die Lösungen kontinuierlich zu verbessern.

2.4.2 Verwendung von Design Thinking in der Diplomarbeit

Design Thinking wurde während der Planungsphase und darüber hinaus kontinuierlich verwendet. Durch häufige Meetings mit dem Betreuer und dem Projektpartner dieser Diplomarbeit haben Ideen und die Prototypen (Mockups) permanent Feedback erhalten und wurden dementsprechend angepasst und verbessert.

2.5 Brain Storming

Brainstorming ist eine weit verbreitete und populäre Kreativitätstechnik, die darauf abzielt, eine Vielzahl von Ideen in kurzer Zeit zu generieren.

2.5.1 Prinzipien von Brainstorming

Das Brainstorming basiert auf einigen grundlegenden Prinzipien, die darauf abzielen, die kreative Zusammenarbeit zu fördern und die Entstehung neuer Ideen zu begünstigen. Zu diesen Prinzipien gehören:

- **Keine Kritik:** Während der Brainstorming-Sitzung sollte jegliche Kritik und Bewertung von Ideen vermieden werden.
- **Quantität geht vor Qualität:** In der Ideengenerierungsphase des Brainstormings ist das Hauptziel, möglichst viele Ideen zu sammeln, unabhängig von ihrer Qualität. Die Auswahl und Bewertung der Ideen erfolgt in einem späteren Schritt.
- **Kombination und Verbesserung von Ideen:** Man wird ermutigt auf bereits geäußerte Ideen aufzubauen und sie zu verbinden oder zu verbessern.
- **Freies Denken:** Brainstorming ist auf unkonventionelle und unerwartete Ideen ausgelegt um keine Ideen als unnötig einzustufen.

2.5.2 Verwendung von Brainstorming in der Diplomarbeit

Nach jedem Meeting wurde Brainstorming angewandt um auf alle möglichen Ideen zu kommen. Wenn sich auf etwas geeinigt wurde, wurde es in die Tat umgesetzt, um Feedback dafür zu bekommen.

2.6 Kopfstand Methode

Die Kopfstand-Methode, ist eine Kreativitätstechnik, die darauf abzielt, sich anstatt direkt auf die Lösung eines Problems zu konzentrieren, Probleme aus einer negativen Perspektive zu betrachten.

- **Problemumkehr**: Das ursprüngliche Problem oder die Fragestellung wird in ihr Gegenteil umgeformt, um eine neue Perspektive zu gewinnen.
- **Ideengenerierung**: Es werden Ideen entwickelt, die auf die umgekehrte Problemstellung oder Fragestellung abzielen, und konzentrieren sich dabei auf die negativen Aspekte des Themas.
- **Umkehr der Lösungsansätze**: Die negativen Ideen werden zurück in positive Lösungsansätze umgewandelt.
- **Bewertung und Auswahl**: Die entwickelten Ideen und Lösungsansätze werden hinsichtlich ihrer Anwendbarkeit, Machbarkeit und Relevanz bewertet. Die vielversprechendsten Lösungen können dann weiter verfeinert und in die Praxis umgesetzt werden.

2.6.1 Verwendung der Kopfstand Methode in der Diplomarbeit

Durch eine Kombination von anderen Kreativitätstechniken kamen immer Gedanken von Problemen wo sich die frage stellte , welche Probleme man mit gewissen Vorangehensweisen hätte. Somit wurde im Laufe der Planung mit der Kopfstand Methode abgewägt was in naher Zukunft an Problemen bereiten könnte und wie mit diesen Problemen umgegangen wird.

2.7 Mockups

Im Rahmen des Entwicklungsprozesses unserer mobilen Anwendung haben wir vier verschiedene Mockups erstellt, um das beste Design für die Nutzer*innen zu ermitteln und es optimal an die Bedürfnisse verschiedener Zielgruppen anzupassen. Die Gestaltung dieser Mockups berücksichtigt sowohl ästhetische als auch funktionale Aspekte, um eine ansprechende und effiziente Benutzeroberfläche zu schaffen, die den unterschiedlichen Anforderungen der*die Nutzer*innen gerecht wird.

Die vier Mockups wurden sorgfältig entworfen und auf Basis von Designprinzipien und Benutzeranforderungen entwickelt, um sicherzustellen, dass sie alle relevanten Funktionen und Interaktionen der Anwendung abbilden. Dabei wurde darauf geachtet, unterschiedliche Designansätze und Interaktionsmöglichkeiten zu berücksichtigen, um ein breites Spektrum an Nutzerpräferenzen abzudecken.

Nach einer umfassenden Analyse und Bewertung der vier Mockups haben unser Berater und Partner das vierte Mockup als das am besten geeignete Design ausgewählt. Dieses Design wurde aufgrund seiner benutzerfreundlichen Gestaltung, der klaren Struktur und der ansprechenden visuellen Elemente bevorzugt. Zudem wurde es als am ehesten in der Lage erachtet, die Bedürfnisse und Anforderungen der verschiedenen Zielgruppen zu erfüllen.

Beim Öffnen der mobilen Anwendung gelangen die Nutzer*innen zunächst in den Log-in-Bereich (siehe Abbildung 1, siehe Abbildung 2). Hier besteht die Möglichkeit, sich mithilfe eines QR-Codes oder NFC-Technologie schnell und unkompliziert einzuloggen. Durch die Integration dieser modernen und benutzerfreundlichen Authentifizierungsmethoden wird der Zugang zur Anwendung für die Nutzer*innen erleichtert und die Sicherheit der persönlichen Daten gewährleistet.

Abbildung 3: Mockup 1 | Log-in-Bereich der mobilen Anwendung

Abbildung 4: Mockup 2 | Log-in-Bereich der mobilen Anwendung

Nach erfolgreichem Einloggen in die mobile Anwendung werden die Nutzer*innen automatisch zum heutigen Trainingsplan weitergeleitet. Dort besteht die Möglichkeit, den Trainingsplan jederzeit durch Berührung der Touchfläche “Trainingsplan starten” zu starten. Diese benutzerfreundliche Gestaltung ermöglicht einen schnellen Zugriff auf die wichtigsten Funktionen und erleichtert den Einstieg in das Training.

In der unteren Leiste, die in Abbildung 3 und siehe Abbildung 4 dargestellt ist, können die Nutzer*innen bequem zwischen dem letzten und dem heutigen Trainingsplan navigieren. Diese Funktion erleichtert den Zugriff auf vergangene und aktuelle Trainingspläne und bietet den Nutzer*innen eine flexible Möglichkeit, ihre Trainingshistorie und Fortschritte einzusehen und zu verwalten.

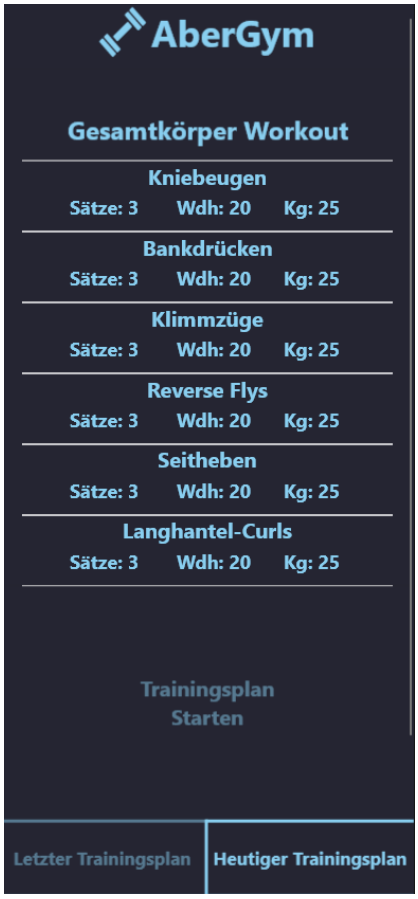

Abbildung 5: Mockup 3 | Hauptbereich der mobilen Anwendung

Abbildung 6: Mockup 4 | Hauptbereich der mobilen Anwendung

Sobald die Nutzer*innen die Touchfläche “Trainingsplan starten” betätigen, wird der heutige Trainingsplan auf einer neuen Seite geöffnet und in einer To-Do-Ansicht präsentiert. In dieser Ansicht können die Nutzer*innen eine beliebige Übung auswählen und durchgehen. Zudem besteht die Möglichkeit, eine Übung zu bearbeiten, indem die Nutzer*innen die gewünschte Übung längere Zeit gedrückt halten. Diese flexible und benutzerfreundliche Gestaltung ermöglicht es den Nutzer*innen, ihren Trainingsplan individuell anzupassen und effizient durchzuarbeiten (siehe Abbildung 5, siehe Abbildung 6, siehe Abbildung 7, siehe Abbildung 8).

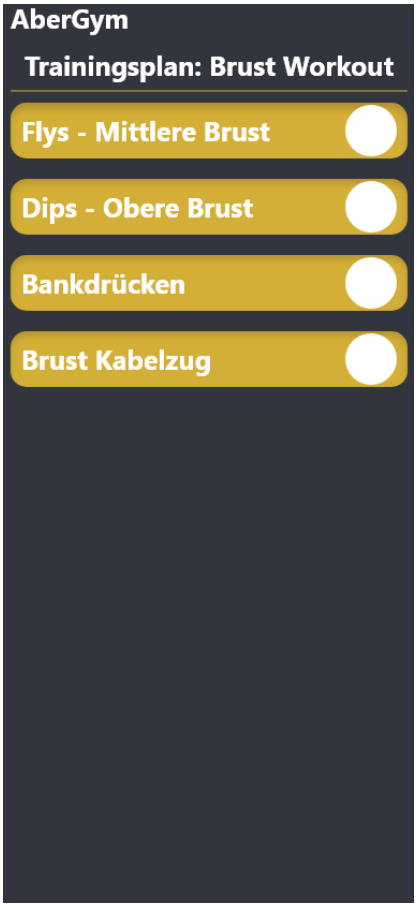

Abbildung 7: Mockup 1 | To-Do-Bereich der mobilen Anwendung

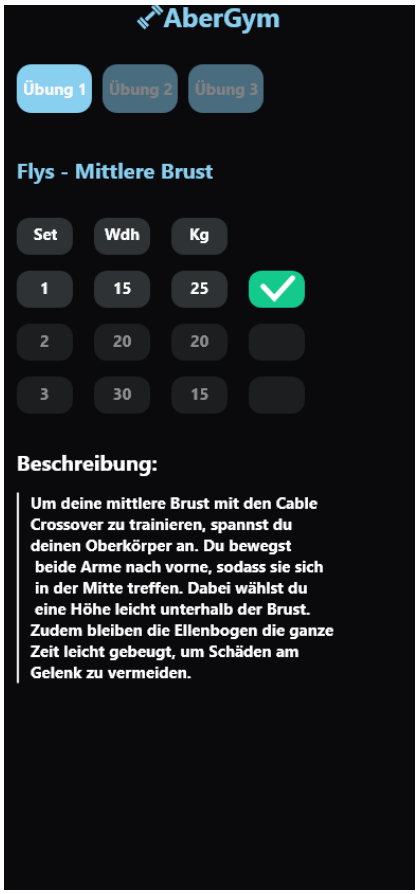

Abbildung 8: Mockup 2 | To-Do-Bereich der mobilen Anwendung

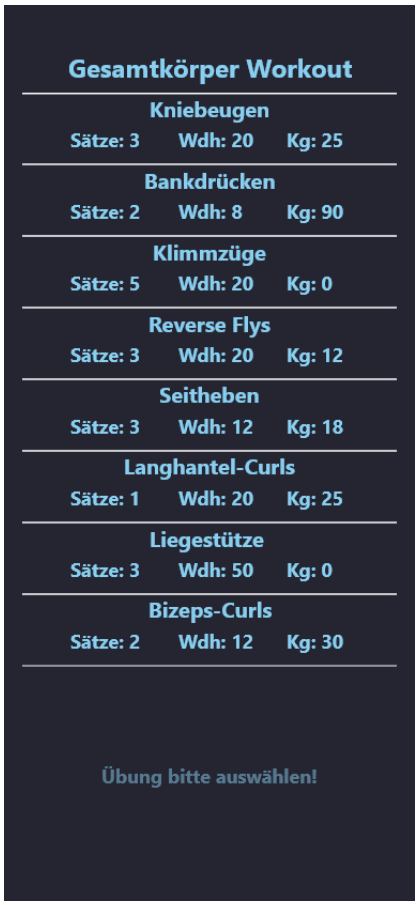

Abbildung 9: Mockup 3 | To-Do-Bereich der mobilen Anwendung

Abbildung 10: Mockup 4 | To-Do-Bereich der mobilen Anwendung

Wenn die Nutzer*innen eine Übung auswählen, wird die entsprechende Übungsinformation erneut angezeigt. In jedem Mockup ist ein Zähler integriert, der bei Berührung des Bildschirms erhöht wird. Der gesamte Bildschirm dient als Druckfläche, um die Bedienung auch für Nutzer*innen mit Handschuhen zu erleichtern, da das Drücken von kleinen Buttons in solchen Situationen schwierig sein kann.

Sobald der Zähler den gleichen Wert wie die Satzanzahl der Übung erreicht hat, werden die Nutzer*innen automatisch zur To-Do-Ansicht zurückgeleitet. Die abgeschlossene Übung wird dann grau markiert und ans Ende der Liste verschoben. Dieses intuitive Design ermöglicht es den Nutzer*innen, ihren Fortschritt im Trainingsplan einfach nachzuvollziehen und sich auf die verbleibenden Übungen zu konzentrieren (siehe Abbildung 9, siehe Abbildung 6, siehe Abbildung 11, siehe Abbildung 12).

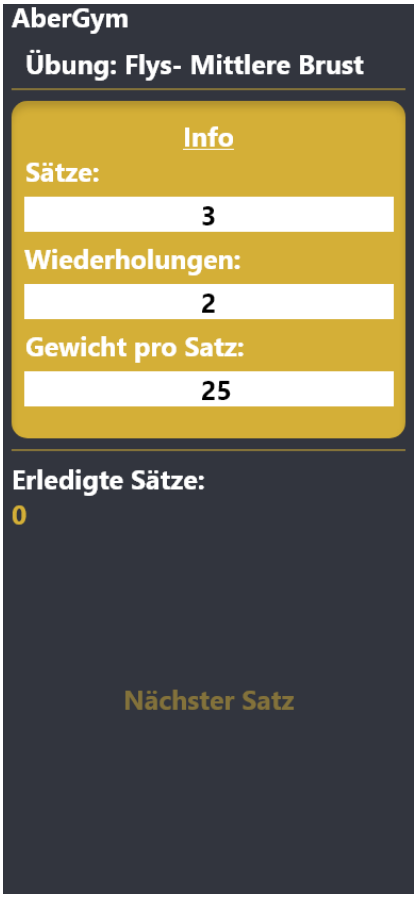

Abbildung 11: Mockup 1 | Zähl-Bereich der mobilen Anwendung

Abbildung 12: Mockup 3 | Zähl-Bereich der mobilen Anwendung

Abbildung 13: Mockup 4 | Zähl-Bereich der mobilen Anwendung

Sobald die Nutzer*innen alle Übungen abgeschlossen haben, werden sie zur Hauptseite zurückgeleitet, auf der sie die vorgenommenen Änderungen während der Durchführung des Trainingsplans einsehen können. Der aktualisierte Trainingsplan, der die neuen Werte der jeweils bearbeiteten Übungen enthält, wird unter der Navigation "Heutiger Trainingsplanängezeigt. Die Nutze*rinnen können den überarbeiteten Trainingsplan erneut durcharbeiten, indem sie die Touchfläche “Trainingsplan starten” betätigen.

In der Navigation "Letzter Trainingsplan"wird hingegen der ursprüngliche Trainingsplan mit den alten Daten der jeweiligen bearbeiteten Übungen dargestellt. Dieser kann jedoch nicht erneut gestartet werden. Diese Funktion ermöglicht den Nutzer*innen, die Fortschritte und Veränderungen im Trainingsplan nachzuvollziehen und die Entwicklung ihrer Trainingsergebnisse im Zeitverlauf zu verfolgen (siehe Abbildung 13, siehe Abbildung 14).

Abbildung 14: Mockup 3 | Hauptbereich nach der Durcharbeitung des Trainingplans

Abbildung 15: Mockup 4 | Hauptbereich nach der Durcharbeitung des Trainingplans

Obwohl das Mockup 4 ursprünglich ausgewählt wurde, wurde während des Entwicklungsprozesses das Design weiter angepasst, optimiert und verschönert. Diese Änderungen wurden vorgenommen, um den Nutzer*innen die Navigation innerhalb der App zu erleichtern und ihnen dabei zu helfen, sich besser zurechtzufinden. Durch diese Designanpassungen wurde eine benutzerfreundliche und ansprechende Oberfläche geschaffen, die den Bedürfnissen und Erwartungen der-die Nutzer-innen entspricht.

3 Technologien

3.1 Flutter

Flutter ist ein Open-Source UI-Toolkit von Google zur Entwicklung von Anwendungen für mobile Geräte, Desktop-Computer und das Web. Es basiert auf der Programmiersprache Dart und verwendet eine Vielzahl von Widgets und Tools, um eine schnelle Entwicklung und hohe Qualität zu ermöglichen. Flutter ermöglicht es Entwicklern, attraktive und benutzerfreundliche Anwendungen zu erstellen, die auf verschiedenen Plattformen laufen können [1].

3.1.1 Entwicklungs-Ergonomie

Flutter bietet eine sehr gute Entwicklungs-Ergonomie durch eine intuitive Benutzeroberfläche und eine umfassende Dokumentation. Die Widgets und Tools von Flutter sind gut strukturiert und können einfach angepasst werden. Darüber hinaus ermöglicht Flutter eine schnelle Erstellung von Prototypen und eine schnelle Iteration, was die Entwicklung von Anwendungen beschleunigt.

3.1.2 Tiefgehende Codeanalyse

Flutter bietet tiefgehende Codeanalysefunktionen, die Entwicklern helfen, Fehler schnell zu finden und zu beheben. Die Analysefunktionen umfassen eine statische Analyse, die Probleme wie Typfehler, fehlende Importe und andere Fehler aufdeckt, bevor die Anwendung ausgeführt wird. Flutter bietet auch eine dynamische Analyse, die Probleme während der Laufzeit aufdeckt.

3.1.3 Effiziente Navigation und Suche

Flutter bietet eine effiziente Navigation und Suche, die es Entwicklern ermöglicht, schnell durch den Code zu navigieren und Probleme zu finden. Die IDEs, die Flutter unterstützen, wie beispielsweise IntelliJ IDEA, Visual Studio Code und Android Studio, bieten eine schnelle Navigation durch den Code sowie eine leistungsstarke Suche nach Dateien und Zeilen.

3.1.4 Entwicklung: Ausführen, Testen, Debuggen

Flutter ermöglicht Entwicklern eine effiziente Entwicklung, indem es ein integriertes Debugging- und Test-Framework bietet. Entwickler können ihre Anwendungen in Echtzeit testen und debuggen, ohne die Anwendung manuell neu starten zu müssen. Das Testen von Anwendungen ist ebenfalls einfach und kann durch den Einsatz von Frameworks wie Flutter Test und Mockito unterstützt werden.

3.1.5 Funktionen

Flutter bietet zahlreiche Funktionen und Tools, um Entwicklern bei der Entwicklung von Anwendungen zu helfen. Hier sind einige der wichtigsten Funktionen:

- Widgets: Flutter bietet eine große Auswahl an vorgefertigten Widgets, die Entwickler verwenden können, um benutzerdefinierte Benutzeroberflächen zu erstellen.
- Hot Reload: Mit Hot Reload können Entwickler Änderungen an der Anwendung vornehmen und diese Änderungen sofort überprüfen, ohne die Anwendung neu starten zu müssen.
- State Management: Flutter bietet eine Vielzahl von State-Management-Lösungen, einschließlich Provider und Redux, die es Entwicklern ermöglichen, den Zustand ihrer Anwendung effizient zu verwalten.
- Plattformunabhängigkeit: Flutter ist plattformunabhängig und ermöglicht es Entwicklern, eine Anwendung zu erstellen, die auf verschiedenen Plattformen ausgeführt werden kann, einschließlich Android, iOS, Web und Desktop.

- Performance: Flutter ist schnell und leistungsfähig und bietet eine schnelle Animation und flüssige Benutzeroberflächen, ohne Einbußen bei der Performance. Durch die Verwendung von Skia, einer 2D-Grafik-Engine, können Entwickler hochwertige Animationen und visuell ansprechende Benutzeroberflächen erstellen, die auf verschiedenen Geräten einheitlich und flüssig funktionieren.

3.1.6 Vorteile

Flutter bietet mehrere Vorteile:

- **Plattformübergreifende Entwicklung**: Mit Flutter können Entwickler plattformübergreifende Anwendungen erstellen, die auf verschiedenen Betriebssystemen wie iOS, Android, Web und Desktop laufen. Dies reduziert die Entwicklungskosten und spart Zeit und Ressourcen.
- **Schnelle Entwicklung**: Flutter bietet die Funktion "Hot Reload", die es Entwickler*innen ermöglicht, Änderungen in Echtzeit zu sehen, ohne die Anwendung neu starten zu müssen. Dadurch wird die Entwicklung von Flutter-Anwendungen schneller und effizienter.
- **Reaktionsfähigkeit**: Flutter-Anwendungen sind schnell und reaktionsfähig, da sie auf der leistungsstarken Grafik-Engine Skia basieren. Dies ermöglicht es Entwickler*innen, reibungslose Benutzererfahrungen mit flüssigen Animationen und Grafiken zu schaffen.
- **Einfache UI-Erstellung**: Mit der eigenen Widget-Bibliothek von Flutter können Entwickler*innen schnell und einfach ansprechende Benutzeroberflächen erstellen. Die Bibliothek enthält viele vorgefertigte Widgets, die einfach angepasst werden können.
- **Native Performance**: Flutter-Anwendungen werden in nativem Code ausgeführt, was zu einer höheren Leistung und Geschwindigkeit führt als bei Hybrid- oder webbasierten Anwendungen.

3.2 Dart als Programmiersprache

Dart ist eine objektorientierte Programmiersprache, die von Google entwickelt wurde und erstmals im Jahr 2011 vorgestellt wurde. Im Vergleich zu anderen Sprachen wie Java, Python und C++ ist Dart eine vergleichsweise neue Programmiersprache. Sie wurde entwickelt, um die Herausforderungen bei der Entwicklung von Webanwendungen zu bewältigen und ist auch für die Entwicklung von plattformübergreifenden mobilen Anwendungen geeignet [1].

3.2.1 Statische Typisierung

Eine der wichtigsten Eigenschaften von Dart ist die statische Typisierung. Statische Typisierung bedeutet, dass Variablen und Funktionen vor der Laufzeit überprüft werden. Dadurch können Entwickler*innen Fehler frühzeitig erkennen und vermeiden. Statische Typisierung verbessert auch die Lesbarkeit und Wartbarkeit des Codes.

3.2.2 Dynamische Typisierung

Dart unterstützt auch dynamische Typisierung. Diese Funktion erleichtert die Entwicklung von Anwendungen, die auf sich ändernden Datenstrukturen basieren. Mit der dynamischen Typisierung kann der Code flexibler gestaltet werden und Entwickler*innen können schneller auf Änderungen reagieren.

3.2.3 Funktionale Programmierung

Dart unterstützt auch funktionale Programmierung, die eine saubere und lesbare Codebasis fördert. Die Verwendung von Funktionen als Parameter erhöht die Flexibilität und Wiederverwendbarkeit von Code.

3.2.4 Asynchrone Programmierung

Dart unterstützt asynchrone Programmierung, um die Leistung bei der Verarbeitung von Netzwerk- und E/A-Operationen zu verbessern. Mit der asynchronen Programmierung können Entwickler*innen Anwendungen entwickeln, die schnell und effizient arbeiten und auf die Bedürfnisse der*die Benutzer*innen reagieren.

3.2.5 Widgets

Ein zentraler Bestandteil von Flutter sind Widgets. Widgets sind die Bausteine für die Erstellung von Benutzeroberflächen in Flutter. Sie sind leichtgewichtig und anpassbar und ermöglichen es Entwicklern, komplexe Benutzeroberflächen mit wenig Code zu erstellen. Es gibt eine große Auswahl an vorgefertigten Widgets in Flutter, die Entwickler verwenden können, um schnell und einfach eine benutzerdefinierte Benutzeroberfläche zu erstellen.

Flutter bietet auch die Möglichkeit, eigene Widgets zu erstellen. Dies gibt Entwicklern mehr Flexibilität und Kontrolle über die Benutzeroberfläche ihrer Anwendung. Darüber hinaus können Widgets in Flutter einfach angepasst werden, um sie an die Anforderungen der Anwendung anzupassen.

3.2.6 State Management

Die Verwaltung des Zustands ist ein wichtiger Aspekt bei der Entwicklung von Anwendungen. In Flutter gibt es verschiedene Möglichkeiten, den Zustand einer Anwendung effizient zu verwalten. Die am häufigsten verwendeten Lösungen für das State Management in Flutter sind Provider und Redux.

Provider ist eine einfachere Lösung für das State Management, die es Entwicklern ermöglicht, den Zustand ihrer Anwendung zu verwalten, ohne zusätzliche Bibliotheken hinzufügen zu müssen. Es ist eine leichtgewichtige Lösung, die die Verwaltung des Zustands vereinfacht und die Leistung der Anwendung verbessert.

Redux ist eine robuste Lösung für das State Management in Flutter, die Entwicklern eine hohe Kontrolle über den Zustand ihrer Anwendung bietet. Redux ist eine Bibliothek, die es Entwicklern ermöglicht, den Zustand ihrer Anwendung zentralisiert zu verwalten. Es bietet auch eine Vielzahl von Werkzeugen und Funktionen, um die Verwaltung des Zustands zu erleichtern.

3.2.7 Vorteile von Dart

Dart bietet eine Reihe von Vorteilen für Entwickler, darunter:

- **Effiziente Ausführung**: Dart ist eine schnelle und leistungsfähige Sprache, die eine effiziente Ausführung von Anwendungen ermöglicht.
- **Statische Typisierung**: Dart ist eine statisch typisierte Sprache, die Entwicklern hilft, Fehler frühzeitig zu erkennen und zu beheben.
- **Flexibilität**: Dart unterstützt dynamische Typisierung, was Entwicklern Flexibilität bei der Entwicklung von Anwendungen bietet, die auf sich ändernden Datenstrukturen basieren.
- **Asynchrone Programmierung**: Dart unterstützt asynchrone Programmierung, um die Leistung bei der Verarbeitung von Netzwerk- und E/A-Operationen zu verbessern.
- **Wiederverwendbarkeit von Code**: Dart bietet die Möglichkeit, Funktionen als Parameter zu übergeben, um die Wiederverwendbarkeit von Code zu erhöhen.
- **Einfach zu erlernen**: Dart ist eine relativ einfache Sprache, die schnell erlernt werden kann.
- **Integration mit Flutter**: Dart wird von Flutter unterstützt und bietet Entwicklern eine leistungsstarke und effiziente Möglichkeit, plattformübergreifende Anwendungen zu entwickeln.

3.3 Visual Studio Code

Visual Studio Code ist ein kostenloses, plattformübergreifendes Code-Editor-Tool von Microsoft. Es ist ein beliebtes Tool für Entwickler*innen, da es eine Vielzahl von Funktionen und Erweiterungen bietet, um die Produktivität und Effizienz zu verbessern.

Visual Studio Code bietet eine intuitive Benutzeroberfläche, die es Entwickler*innen erleichtert, schnell und einfach zu navigieren und Code zu schreiben. Es bietet auch integrierte Debugging-Tools, die Entwickler*innen helfen, Fehler zu finden und zu beheben, sowie integrierte Versionskontrolltools für Git, um Änderungen am Code effektiv zu verwalten.

3.3.1 Extensions

Eine der wesentlichsten Funktionen in VS Code sind Extensions (Erweiterungen). VS Code-Extensions können verschiedene Funktionen bieten, wie z.B. Syntax-Hervorhebung, Autovervollständigung, Debugging-Tools, Git-Integration und vieles mehr. Es gibt eine breite Palette von Erweiterungen, die für verschiedene Programmiersprachen und Frameworks verfügbar sind, um die Entwicklungserfahrung zu verbessern und die Produktivität zu steigern.

Die Installation von VS Code-Extensions ist einfach und unkompliziert. Sie können über den Visual Studio Code-Marktplatz oder direkt aus der Editor-Benutzeroberfläche installiert werden. Nach der Installation stehen die neuen Funktionen und Tools sofort zur Verfügung.

3.4 Docker

Docker ist eine Open-Source-Plattform, die es Entwicklern ermöglicht, Anwendungen in isolierten Containern zu erstellen, bereitzustellen und auszuführen. Docker-Container sind leichtgewichtig und portabel und bieten eine effektive Möglichkeit, Anwendungen in verschiedenen Umgebungen und Infrastrukturen auszuführen.

3.4.1 Funktionsweise

Docker verwendet Container, um Anwendungen zu isolieren und eine konsistente Umgebung für ihre Ausführung zu schaffen. Container sind ähnlich wie virtuelle Maschinen, jedoch leichter und schneller zu erstellen, da sie den Kernel des Host-Betriebssystems nutzen. Jeder Container enthält alles, was eine Anwendung zum Ausführen benötigt, einschließlich des Codes, der Abhängigkeiten und der Konfiguration.

Die Funktionsweise von Docker basiert auf einem Schichtmodell, das aus drei Komponenten besteht:

1. **Docker Engine**: Dies ist das Kernstück von Docker und besteht aus dem Docker-Daemon und der Docker-CLI (Command Line Interface). Der Docker-Daemon ist ein Hintergrundprozess, der die Verwaltung und Ausführung von Containern übernimmt, während die Docker-CLI als Schnittstelle für den Benutzer dient, um mit dem Docker-Daemon zu interagieren.

2. **Images**: Ein Docker-Image ist eine Vorlage oder Blaupause für die Erstellung von Containern. Es enthält den Code, die Abhängigkeiten und Konfigurationen einer Anwendung sowie alle anderen erforderlichen Komponenten, die zur Ausführung der Anwendung benötigt werden. Docker-Images werden über Docker Files erstellt, die eine Liste von Anweisungen enthalten, um das Image zu konfigurieren und zu erstellen.

3. **Container**: Ein Docker-Container ist eine Instanz eines Docker-Images, die ausgeführt wird. Ein Container kann gestartet, gestoppt und gelöscht werden. Jeder Container ist isoliert und hat seine eigene Dateisystemumgebung, Netzwerkschnittstellen und Ressourcenlimits.

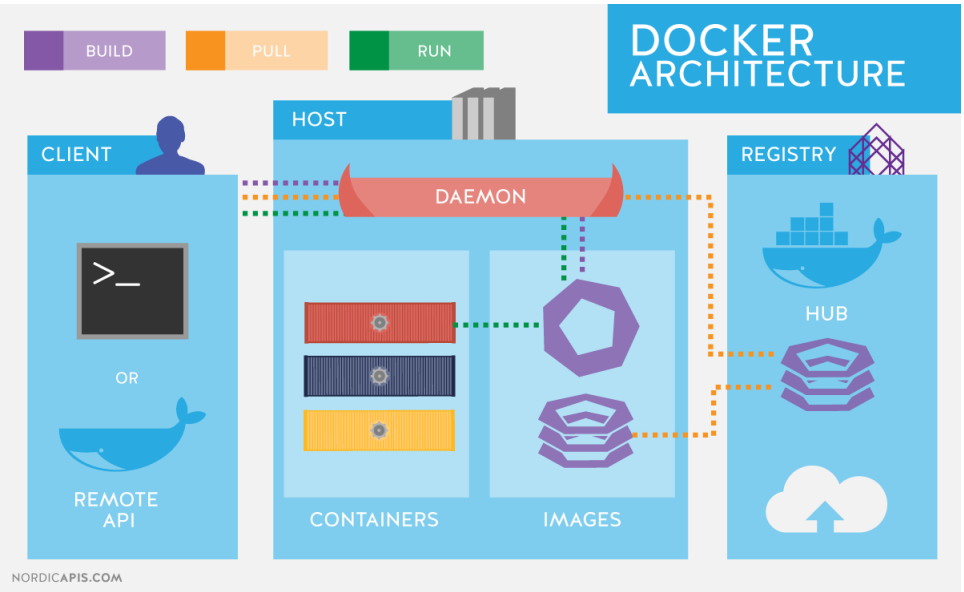

Abbildung 16: Docker Architektur

3.4.2 Vorteile

- **Portabilität**: Docker-Container sind plattformunabhängig und können auf verschiedenen Betriebssystemen und Infrastrukturen ausgeführt werden.
- **Flexibilität**: Mit Docker können Anwendungen schnell erstellt, abgeändert und bereitgestellt werden, ohne die zugrunde liegende Infrastruktur ändern zu müssen.
- **Skalierbarkeit**: Docker ermöglicht eine einfache horizontale Skalierung von Anwendungen, indem es das Erstellen und Bereitstellen von Containern automatisiert.
- **Sicherheit**: Docker bietet Sicherheitsfunktionen wie Isolation und eingeschränkte Ressourcenkontrolle, um eine sicherere Ausführung von Anwendungen zu gewährleisten.
- **Effizienz**: Docker-Container sind leicht und benötigen weniger Ressourcen als virtuelle Maschinen, was zu einer höheren Effizienz und Leistung führt.

3.5 Android Studio

Android Studio ist eine integrierte Entwicklungsumgebung (IDE), die speziell für die Entwicklung von Android-Apps entwickelt wurde. Es wurde von Google entwickelt und ist kostenlos für Entwickler verfügbar, um Android-Apps zu erstellen und zu bearbeiten.

3.5.1 Funktionen

Android Studio verfügt über eine Vielzahl von Funktionen, die dabei helfen, schneller und effizienter Android-Apps zu entwickeln. Zu den wichtigsten Funktionen gehören:

- **Intelligentes Code-Editing:** Android Studio bietet intelligentes Code-Editing mit automatischen Vorschlägen, Fehlererkennung und Refactoring-Funktionen.
- **Emulator:** Entwickler können den Android-Emulator nutzen, um ihre Apps auf verschiedenen Android-Geräten zu testen, ohne physische Geräte besitzen zu müssen.
- **Layout-Editor:** Android Studio verfügt über einen Layout-Editor, mit dem Entwickler die Benutzeroberfläche ihrer Apps visuell gestalten können.
- **Gradle Build-System:** Android Studio verwendet das Gradle Build-System, das Entwicklern ermöglicht, komplexe Abhängigkeiten und Builds zu verwalten.
- **Integration mit anderen Tools:** Android Studio ist nahtlos in andere Google-Tools wie Firebase und Google Cloud Platform integriert.

3.5.2 Emulator

Der Android Studio Emulator ist ein wichtiges Tool für Android-Entwickler, das es ihnen ermöglicht, ihre Apps auf verschiedenen Android-Geräten zu testen, ohne physische Geräte besitzen zu müssen. Der Emulator wird mit Android Studio mitgeliefert und kann einfach über die IDE gestartet werden.

Einer der größten Vorteile des Emulators ist, dass Entwickler ihre Apps auf verschiedenen Android-Versionen und Gerätekonfigurationen testen können, um sicherzustellen, dass ihre Apps auf allen unterstützten Geräten reibungslos funktionieren. Der Emulator kann eine Vielzahl von Android-Versionen und -Gerätekonfigurationen emulieren, einschließlich verschiedener Bildschirmauflösungen und -größen, Prozessortypen und Speicherkapazitäten.

Ein weiterer Vorteil des Emulators ist, dass er den Entwicklungsprozess beschleunigen kann, indem er den Build- und Bereitstellungsprozess verkürzt. Anstatt jedes Mal eine neue Version der App auf einem physischen Gerät zu testen, können Entwickler die App einfach im Emulator starten und testen, was Zeit spart und die Entwicklungszeit verkürzt.

Die Einrichtung des Emulators in Android Studio ist einfach und erfordert nur wenige Schritte. Entwickler müssen zunächst sicherstellen, dass sie die neueste Version von Android Studio heruntergeladen und installiert haben. Sobald sie Android Studio geöffnet haben, können sie den Emulator über das AVD Manager-Tool starten, das im Menü "Werkzeugeßu finden ist.

Es gibt jedoch auch einige Nachteile beim Verwenden des Emulators. Einer der größten Nachteile ist die Geschwindigkeit. Da der Emulator ein virtuelles Gerät ist, kann er langsamer sein als ein physisches Gerät. Dies kann dazu führen, dass Entwickler länger warten müssen, um ihre Apps im Emulator zu testen.

Ein weiterer Nachteil ist, dass der Emulator nicht alle Funktionen eines physischen Geräts emulieren kann. Beispielsweise kann der Emulator keine Anrufe oder Textnachrichten empfangen und senden, da er kein physisches Mobilfunkmodem hat. Dies bedeutet, dass Entwickler nicht alle Aspekte ihrer App im Emulator testen können und gegebenenfalls auf physische Geräte zurückgreifen müssen.

Zusammenfassend lässt sich sagen, dass der Android Studio Emulator ein wertvolles Tool für Android-Entwickler ist, um ihre Apps auf verschiedenen Geräten und Android-Versionen zu testen. Obwohl der Emulator einige Nachteile hat, überwiegen die Vorteile in den meisten Fällen und er ist ein unverzichtbares Werkzeug für die App-Entwicklung.

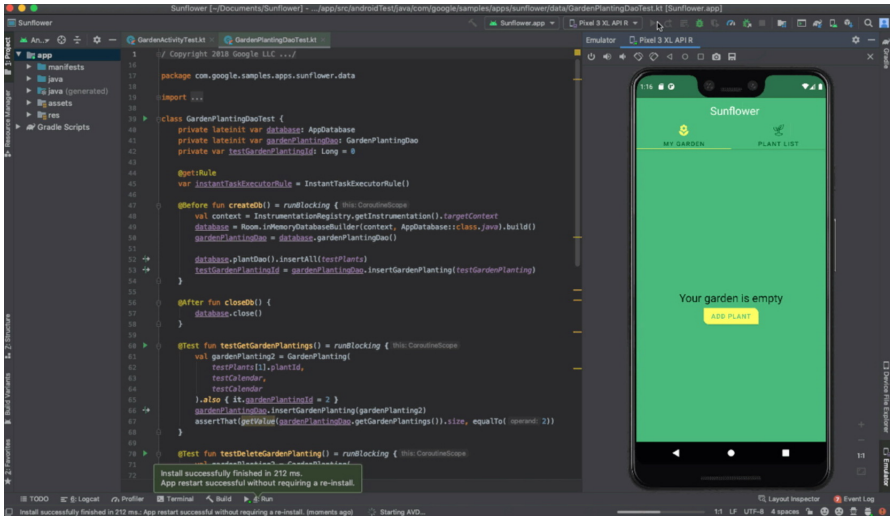

Abbildung 17: Android Studio Emulator

3.6 AdobeXD

AdobeXD ist eine Design-Software, die speziell für die Erstellung von Benutzeroberflächen und Interaktionen für Mobile Apps, Webseiten und andere digitale Plattformen entwickelt wurde.

Funktionen

Die wichtigsten Funktionen sind intuitive Layout-Tools, um Entwürfe schnell zu erstellen und zu bearbeiten, Vektor-Tools, um hochwertige Grafiken zu erstellen, Prototyping-Funktionen, um interaktive Prototypen zu erstellen und zu testen und die Möglichkeit, Designs in Echtzeit zu teilen und Feedback von Stakeholdern zu erhalten.

Plattformübergreifendes Design

AdobeXD unterstützt plattformübergreifendes Design, was bedeutet, dass man nur eine einzige Design-Datei erstellen muss, um diese dann auf AdobeXD auf verschiedenen Geräten und Plattformen verwenden zu können. Das ermöglicht, schnell und effizient Designs für verschiedene Geräte und Plattformen zu erstellen.

Integration mit anderen Tools

AdobeXD ist nahtlos in andere Adobe-Tools wie Photoshop und Illustrator integriert. Das ermöglicht, Designs nahtlos zwischen verschiedenen Adobe-Tools zu übertragen. Es ist auch in andere Tools wie Slack und Microsoft Teams integriert, um die Zusammenarbeit zu erleichtern.

Cloud-basierte Zusammenarbeit

AdobeXD bietet eine Cloud-basierte Zusammenarbeit an, mit der es möglich ist, Designs in Echtzeit zu teilen und Feedback von Stakeholdern zu erhalten. Designer*innen können Links zu ihren Designs freigeben und Stakeholder können Kommentare und Feedback direkt in die Designs geben.

3.7 IntelliJ IDEA

IntelliJ IDEA ist eine leistungsstarke und vielseitige Entwicklungsumgebung, die von Entwicklern auf der ganzen Welt genutzt wird. Mit der Fähigkeit, eine Vielzahl von Programmiersprachen zu unterstützen, hat es sich zu einem wichtigen Werkzeug für die Softwareentwicklung entwickelt.

Die Free-Version von IntelliJ IDEA bietet Unterstützung für Programmiersprachen wie Java, Groovy, Kotlin und Scala sowie Standardsyntaxsprachen wie XML. Mit dieser Version können Entwickler grundlegende Projekte erstellen und debuggen. Die Ultimate-Version von IntelliJ IDEA bietet jedoch erweiterte Funktionen, wie beispielsweise die Integration mit Frameworks, eine größere Auswahl an Plugins, Code-Refactoring-Tools und viele weitere Funktionen, die die Produktivität von Entwicklern erhöhen.

Für Schüler*innen und Studenten*innen steht die Ultimate-Version von IntelliJ IDEA zu einem reduzierten Preis zur Verfügung, was ihnen den Zugang zu leistungsstarken Werkzeugen zur Entwicklung von Anwendungen ermöglicht. Zudem ist es möglich, durch kostenlose Community- oder offizielle Jetbrains-Plugins, weitere Programmiersprachen in IntelliJ IDEA zu integrieren, wie Go, Python, SQL, Ruby und PHP [2].

3.7.1 Entwicklungs-Ergonomie

Die IntelliJ IDEA Ultimate bietet zahlreiche Optionen zur Erstellung neuer Projekte und ermöglicht so einen einfachen Einstieg in die Entwicklung. Individuelle Anpassungen wie das Ändern des Designs oder der Tastenkürzel sind möglich, um die IDE an die Bedürfnisse des Entwicklers anzupassen. Zusätzliche Funktionalitäten können durch Plugins hinzugefügt werden, um spezifische Anforderungen zu erfüllen.

3.7.2 Tiefgehende Codeanalyse

Obwohl die IntelliJ Ultimate IDE hauptsächlich für die Java-Entwicklung entwickelt wurde, unterstützt sie auch eine Vielzahl anderer Programmiersprachen wie Groovy, Kotlin, Scala, JavaScript, TypeScript und SQL. Für jede dieser Sprachen bietet die IntelliJ IDEA eine spezielle intelligente Programmierhilfe. Durch die Indexierung des Projektcodes beim Start der IDE können Fehler während der Entwicklung in Echtzeit erkannt und Code-Vervollständigungsvorschläge angezeigt werden. Zusätzlich bietet die IDE Funktionen zur Durchführung von Refactorings, wie zum Beispiel Umbenennung von Dateinamen.

3.7.3 Effiziente Navigation und Suche

Die IntelliJ IDEA ermöglicht es dem Benutzer, schnell und einfach nach bestimmten Wörtern im gesamten Projekt zu suchen, indem eine Globale Suche verwendet wird. Alternativ kann auch eine Lokale Suche in der aktuellen Datei durchgeführt werden. Wenn ein Wort in einer Datei ausgewählt wird, werden alle anderen Vorkommen dieses Wortes in der Datei hervorgehoben. Zusätzlich können Funktionen und Klassen in ihrem Kontext betrachtet werden, indem jede Verwendung dieser Elemente dargestellt wird. Dies erleichtert die Navigation und Verwaltung von Code in großen Projekten.

3.7.4 Entwicklung: Ausführen, Testen, Debuggen

Für das Ausführen von Projekten kann in IntelliJ IDEA das integrierte Terminal genutzt werden. Alternativ dazu bietet die IDE auch die Möglichkeit, Projekte direkt innerhalb der Entwicklungsumgebung auszuführen. Hierfür können verschiedene Zielplattformen definiert werden, auf denen das Projekt laufen soll. Des Weiteren bietet die IDE eine umfangreiche Unterstützung für Code-Tests und Debugging-Funktionen an.

3.7.5 Effektive Zusammenarbeit im Team

Durch Code With Me von Jetbrains können mehrere Personen in Echtzeit an einer gemeinsamen Datei arbeiten. Zusätzlich können Video- und Sprachanrufe über die Plattform getätigt werden, um sich mit anderen Teammitgliedern zu besprechen. Des Weiteren hat Jetbrains kürzlich ein Remote-Development-System eingeführt, welches es ermöglicht, die IntelliJ Ultimate IDEA auf einem leistungsstarken Host-System zu betreiben. Über dieses neue Tool können Nutzer von überall auf der Welt auf ihre Projekte zugreifen und von verschiedenen Orten aus gemeinsam arbeiten.

3.7.6 Funktionen

Die Entwicklungsumgebung IntelliJ bietet zahlreiche Funktionen und Tools, um Entwickler*innen bei der Entwicklung von Softwareprojekten zu unterstützen. Nachfolgend sind einige der wichtigsten Funktionen aufgeführt:

- Code-Editing: IntelliJ bietet intelligentes Code-Editing mit Code-Vervollständigung, Syntax-Hervorhebung, Refactoring und Code-Analyse-Funktionen.
- Debugger: Der integrierte Debugger ermöglicht es Entwicklerinnen, Code zu debuggen und Fehler schnell zu finden.
- Build-Tools: IntelliJ unterstützt eine Vielzahl von Build-Tools wie Gradle, Maven und Ant.
- Integration mit anderen Tools: IntelliJ ist nahtlos in andere Tools und Frameworks wie Git, JUnit und Spring integriert.
- Version-Control-System: IntelliJ unterstützt verschiedene Version-Control-Systeme wie Git, Subversion und Mercurial.
- Code-Qualität: IntelliJ bietet Code-Qualität-Tools, um Entwicklerinnen dabei zu helfen, fehlerfreien Code zu schreiben und Code-Qualitätsstandards einzuhalten.
- Frameworks: IntelliJ unterstützt eine Vielzahl von Frameworks wie Spring, Hibernate, Struts und mehr.

3.7.7 Vorteile

IntelliJ bietet eine Vielzahl von Vorteilen für Entwickler, darunter:

- Code-Refactoring: IntelliJ bietet eine Vielzahl von Code-Refactoring-Tools, mit denen Entwickler Code effizienter und strukturierter gestalten können.
- Automatisierte Tests: IntelliJ unterstützt automatisierte Tests, die Entwicklern helfen, Fehler frühzeitig zu erkennen und zu beheben.
- Plugins: IntelliJ hat eine große Auswahl an Plugins, die Entwickler verwenden können, um ihre Arbeitsumgebung zu erweitern und ihre Produktivität zu steigern.
- Gute Performance: IntelliJ ist bekannt für seine gute Performance und Stabilität, was Entwicklern ermöglicht, effizienter zu arbeiten.
- Unterstützung für Cloud-Computing: IntelliJ ist auch in der Lage, mit Cloud-Computing-Plattformen wie Amazon Web Services, Google Cloud Platform und Microsoft Azure zu arbeiten.
- Mobile App-Entwicklung: IntelliJ bietet Unterstützung für die Entwicklung von mobilen Apps für Android und iOS, einschließlich Integrationen mit Android Studio und Xcode.

4 Flutter-App

4.1 Start-Screen

Wenn die App geöffnet wird, erscheint zunächst der Startbildschirm.

Abbildung 18: Start-Screen

Dieser wurde wie folgt implementiert:

Listing 1: Start-Screen Packages

```
1       import 'package:abergymmobile/AGM.Animations/FadeAnimation.dart';
2       import 'package:abergymmobile/AGM.Login/NFC.dart';
3       import 'package:abergymmobile/AGM.Login/QrCode.dart';
4       import 'package:google_fonts/google_fonts.dart';
5       import 'package:flutter/foundation.dart';
6       import 'package:flutter/material.dart';
7       import 'package:flutter/services.dart';
```

Der Code importiert verschiedene Pakete, um das Layout und die Funktionen der Anwendung zu erstellen. Dazu gehören Google Fonts für benutzerdefinierte Schriftarten und Flutter-Pakete für Widgets und das Material Design.

Listing 2: main-Funktion

```
1       Future<void> main() async {
2           WidgetsFlutterBinding.ensureInitialized();
3           SystemChrome.setEnabledSystemUIMode(SystemUiMode.manual, overlays: []);
4           debugDefaultTargetPlatformOverride = null;
5           runApp(const MaterialApp(
6           debugShowCheckedModeBanner: false,
7           home: LoginPage(),
8           ));
9           }
```

Die main-Funktion initialisiert die Flutter-App und startet sie. Sie setzt den System-UI-Modus und entfernt das Debug-Banner. Schließlich wird die LoginPage als Startseite festgelegt.

Listing 3: Login-Seite

```
1       class LoginPage extends StatelessWidget {
2   const LoginPage({super.key});
3
4   @override
5   Widget build(BuildContext context) {
6   return Scaffold(
7   ...
8   );
9   }
10  }
```

Das LoginPage-Widget erstellt die Login-Seite. Diese Seite ist in mehrere Abschnitte unterteilt, die im Folgenden erläutert werden.

Listing 4: Layout und Hintergrundfarbe

```
1       decoration: BoxDecoration(
2           gradient: LinearGradient(
3           begin: Alignment.topCenter,
4           colors: [
5           Colors.lightBlue[900]!,
6           Colors.lightBlue[800]!,
7           Colors.lightBlue[400]!,
8           ],
9           ),
10          ),
```

Mit LinearGradient wird ein linearer Farbverlauf von Hellblau bis Dunkelblau erstellt der von oben nach unten verläuft(begin: Allignment.topCenter).

Listing 5: Logo und Willkommensnachricht

```
children: <Widget>[
    FadeAnimation(
      1,
      Row(
        children: <Widget>[
          Text(
            "AberGym",
            style: GoogleFonts.montserrat(
              color: Colors.white,
              fontSize: 50,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(width: 10),
          const Icon(
            Icons.fitness_center,
            color: Colors.white,
            size: 50,
            shadows: <Shadow>[
              Shadow(
                color: Colors.black,
                blurRadius: 20.0,
              )
            ],
          )
        ],
      ),
    ),
    const SizedBox(height: 10),
    FadeAnimation(
      1.3,
      Text(
        "Willkommen zurueck!",
        style: GoogleFonts.montserrat(
          color: Colors.white,
          fontSize: 23,
        ),
      ),
    ),
  ],
```

Der Code erstellt das AberGym-Logo und die Willkommensnachricht mit Hilfe von `FadeAnimation`, `Text`- und `Icon`-Widgets.

Listing 6: QR-Code Schaltfläche

```
FadeAnimation(
1.6,
GestureDetector(
onTap: () {
Navigator.push(
context,
MaterialPageRoute(
builder: (context) => const QRCodePage(),
),
);
},
child: Container(
height: 50,
margin: const EdgeInsets.symmetric(horizontal: 20),
decoration: BoxDecoration(
gradient: LinearGradient(
begin: Alignment.centerLeft,
colors: [
Colors.lightBlue[900]!,
Colors.lightBlue[800]!,
Colors.lightBlue[400]!,
],
),
color: Colors.lightBlue,
borderRadius: BorderRadius.circular(15),
),
child: Row(
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Icon(
Icons.camera_alt,
color: Colors.white,
size: 20,
),
const SizedBox(width: 10),
Text(
"Einloggen per QR-Code",
style: GoogleFonts.montserrat(
color: Colors.white,
fontWeight: FontWeight.bold,
),
),
],
),
),
),
),
```

Diese Schaltfläche ermöglicht dem Benutzer*der Benutzerin, sich per QR-Code einzuloggen. Sie verwendet `FadeAnimation`, `GestureDetector`, `Container` und `Row`-Widgets, um das Layout und die Funktionen der Schaltfläche zu erstellen. Beim Tippen auf die Schaltfläche wird der Benutzer*die Benutzerin zur `QRCodePage` weitergeleitet.

Listing 7: NFC Schaltfläche

```
FadeAnimation(
1.7,
GestureDetector(
onTap: () {
Navigator.push(
context,
MaterialPageRoute(
builder: (context) => const NFC(),
),
);
},
child: Container(
height: 50,
margin: const EdgeInsets.symmetric(horizontal: 20),
decoration: BoxDecoration(
gradient: LinearGradient(
begin: Alignment.topCenter,
colors: [
Colors.lightBlue[900]!,
Colors.lightBlue[800]!,
Colors.lightBlue[400]!,
],
),
color: Colors.lightBlue,
borderRadius: BorderRadius.circular(15),
),
child: Row(
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Icon(
Icons.nfc,
color: Colors.white,
size: 20,
),
const SizedBox(width: 10),
Text(
"Einloggen per NFC",
style: GoogleFonts.montserrat(
color: Colors.white,
fontWeight: FontWeight.bold,
),
),
],
),
),
),
),
```

Diese Schaltfläche ermöglicht dem Benutzer*der Benutzerin, sich per NFC einzuloggen. Ähnlich wie bei der QR-Code-Schaltfläche verwendet sie `FadeAnimation`, `GestureDetector`, `Container` und `Row`-Widgets, um das Layout und die Funktionen der Schaltfläche zu erstellen. Beim Tippen auf die Schaltfläche wird der Benutzer*die Benutzerin zur `NFC`-Seite weitergeleitet.

4.2 Login

Um sich einzuloggen gibt es zwei Varianten. Entweder man loggt sich mithilfe eines QR-Codes ein oder mit NFC.

4.2.1 QR-Code Login

Die QR-Code-Funktion ermöglicht es Benutzer*innen, sich mit einem personalisierten QR-Code einzuloggen.

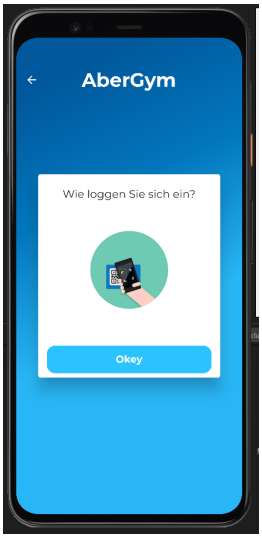

Abbildung 19: QR-CodeTutorial

Abbildung 20: QR-Code auf Kamera

Listing 8: QR-Code Seite erstellen

```
import 'package:qr_code_scanner/qr_code_scanner.dart';
import 'package:shared_preferences/shared_preferences.dart';

class QRCodePage extends StatefulWidget {
const QRCodePage({super.key});

@override
State<QRCodePage> createState() => _QRCodePageState();
}

class _QRCodePageState extends State<QRCodePage> {
final Color darkgrey = const Color.fromRGBO(37, 37, 50, 1);
final Color lightblue = const Color.fromARGB(255, 42, 195, 255);
bool showScanRect = false;
final GlobalKey qrKey = GlobalKey(debugLabel: 'QR');
Barcode? result;
QRViewController? controller;
bool logintrue = false;
String name = "";
bool _shouldNavigate = false;
```

Der Code beginnt mit dem Importieren der erforderlichen Pakete und dem Erstellen der `QRCodePage` StatefulWidget-Klasse. Der Zustand der Seite wird in der `QRCodePageState` Klasse verwaltet. Hier werden einige Variablen deklariert, wie zum Beispiel Farben, Zustände und der QRViewController.

Listing 9: QR-Code Seitenstruktur

```
@override
Widget build(BuildContext context) {
return Scaffold(
extendBodyBehindAppBar: true,
appBar: AppBar(
title: Text(
'AberGym',
style: GoogleFonts.montserrat(
fontSize: 35,
color: Colors.white,
fontWeight: FontWeight.bold,
),
),
backgroundColor: Colors.transparent,
centerTitle: true,
elevation: 0,
),
body: Container(
width: double.infinity,
decoration: BoxDecoration(
gradient: LinearGradient(
begin: Alignment.topCenter,
colors: [
Colors.lightBlue[900]!,
Colors.lightBlue[800]!,
Colors.lightBlue[400]!,
],
),
),
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
!showScanRect
? AlertDialog(
title: Text('Wie loggen Sie sich ein?',
textAlign: TextAlign.center,
style: GoogleFonts.getFont('Montserrat')),
content: Image.asset('assets/images/qrcode.gif'),
actions: [
GestureDetector(
child: Center(
child: Container(
width: 300,
height: 50,
decoration: BoxDecoration(
color: const Color.fromARGB(255, 42, 195, 255),
borderRadius: BorderRadius.circular(15),
),
child: Row(
mainAxisAlignment: MainAxisAlignment.center,
children: [
Text(
'Okey',
style: GoogleFonts.montserrat(
fontSize: 18,
color: Colors.white,
fontWeight: FontWeight.bold,
),
),
],
),
),
),
onTap: () {
setState(() {
showScanRect = true;
});
},
),
],
)
: SizedBox(
height: 400,
width: double.infinity,
child: _buildQrView(context),
),
],
),
),
```

```
80  );
81  }
```

In der `build`-Methode wird die Seitenstruktur der `QRCodePage` erstellt. Hier wird ein `Scaffold`-Widget verwendet, um die Hauptstruktur der Seite zu erzeugen. In der Mitte der Seite wird eine Bedingung geprüft, um entweder einen AlertDialog anzuzeigen, der Benutzer*innen darüber informiert, wie sie sich einloggen können, oder den QR-Code-Scanner anzuzeigen.

Listing 10: QR-Code Scanner erstellen

```
1  Widget _buildQrView(BuildContext context) {
2  var scanArea = (MediaQuery.of(context).size.width < 400 ||
3  MediaQuery.of(context).size.height < 400)
4  ? 150.0
5  : 250.0;
6
7  return QRView(
8  key: qrKey,
9  onQRViewCreated: _onQRViewCreated,
10 overlay: QrScannerOverlayShape(
11 borderColor: lightblue,
12 borderRadius: 10,
13 borderLength: 30,
14 borderWidth: 10,
15 cutOutSize: scanArea),
16 );
17 }
```

Die `buildQrView`-Methode erstellt das QR-Code-Scanner-Widget. Das QRView-Widget erhält die Größe des Scanbereichs basierend auf der Bildschirmgröße und erstellt das Widget mit den angegebenen Design-Parametern.

Listing 11: QR-Code-Scanner Controller

```
1  void _onQRViewCreated(QRViewController controller) {
2  this.controller = controller;
3  controller.scannedDataStream.listen((scanData) {
4  setState(() {
5  result = scanData;
6  checkUser(result?.code);
7  });
8  if (_shouldNavigate) {
9  controller.dispose();
10 Navigator.of(context).pushReplacement(
11 MaterialPageRoute(builder: (context) => WelcomeSplashPage(name)));
12 }
13 });
14 }
```

Die `onQRViewCreated`-Methode wird aufgerufen, wenn das QRView-Widget erstellt wurde. Diese Methode initialisiert den `QRViewController` und hört auf den Datenstrom des gescannten QR-Codes. Wenn ein QR-Code gescannt wurde, wird die `checkUser` Methode aufgerufen, um die Benutzer*innen-Informationen abzurufen und die Seite zu navigieren, falls der Benutzer*die Benutzerin erfolgreich identifiziert wurde.

Listing 12: Datenbankabfrage QR-Code

```
Future<void> checkUser(String? qrCardId) async {
IResultSet result;

final conn = await MySQLConnection.createConnection(

);

await conn.connect();
result = await conn.execute(
"SELECT first_name, last_name FROM Person WHERE card_id = :card_id",
{"card_id": qrCardId},
);

for (final row in result.rows) {
setState(
() {
String? firstName = "";
String? lastName = "";
firstName = row.colAt(0);
lastName = row.colAt(1);
name = "$firstName $lastName";
},
);
}
final prefs = await SharedPreferences.getInstance();
if (name.isEmpty == false) {
setState(() {
_shouldNavigate = true;
logintrue = true;
prefs.setBool('login', logintrue);
prefs.setString('key', name);
});
}

await conn.close();
}
```

Die `checkUser`-Methode überprüft, ob der gescannte QR-Code mit einer gültigen Benutzerinnen-ID übereinstimmt, indem sie eine Datenbankabfrage durchführt. Sie verwendet die `MySQLConnection`-Klasse, um eine Verbindung zur Datenbank herzustellen, und führt eine SQL-Abfrage aus, um den Vornamen und Nachnamen des Benutzersin basierend auf der QR-Code-ID abzurufen. Wenn die Abfrage erfolgreich ist, wird der Name des Benutzers*in gespeichert und der Anmeldestatus in den `SharedPreferences` aktualisiert. Schließlich wird die Verbindung zur Datenbank geschlossen.

4.2.2 NFC Login

Die NFC-Funktion ermöglicht es Benutzer*innen, sich mit einer Chipkarte wie einer Fitness-Mitgliedskarte einzuloggen.

Listing 13: NFC Package und State

```
import 'package:nfc_manager/nfc_manager.dart';

class NFC extends StatefulWidget {
const NFC({super.key});

@override
State<NFC> createState() => _NFCState();
}

class _NFCState extends State<NFC> {
bool _nfcEnabled = false;
ValueNotifier<dynamic> result = ValueNotifier(null);
final Color darkgrey = const Color.fromRGBO(37, 37, 50, 1);
final Color lightblue = const Color.fromARGB(255, 42, 195, 255);

@override
void initState() {
super.initState();
_checkNfcStatus();
}
}
```

Zunächst werden die erforderlichen Pakete importiert, insbesondere das `nfcmanager` Paket. Die `NFC`-Klasse erbt von `StatefulWidget`, und der zugehörige `State` wird in der `NFCState`-Klasse definiert. In `NFCState` werden die initialen Zustände und Farben gesetzt, und die Methode `initState` wird überschrieben, um den NFC-Status beim Start der Anwendung zu überprüfen.

Listing 14: NFC Status überprüfen

```
void _checkNfcStatus() async {
bool nfcEnabled = await NfcManager.instance.isAvailable();
setState(() {
_nfcEnabled = nfcEnabled;
});
}
```

Die Methode `checkNfcStatus` überprüft, ob NFC auf dem Gerät verfügbar und aktiviert ist. Mithilfe der `NfcManager`-Instanz wird die Methode `isAvailable` aufgerufen, um den NFC-Status abzurufen. Der ermittelte Status wird dann im `nfcEnabled` Attribut gespeichert.

Listing 15: NFC Tag scannen

```
void _startScan() {
NfcManager.instance.startSession(onDiscovered: (NfcTag tag) async {
result.value = tag.data;
NfcManager.instance.stopSession();
});
}
```

Die Methode `startScan` startet die NFC-Session und wartet auf das Entdecken eines NFC-Tags. Sobald ein Tag entdeckt wird, wird die `onDiscovered`-Funktion aufgerufen, die die Daten des NFC-Tags in der `result`-Variable speichert und die Session beendet.

Listing 16: NFC Widget erstellen

```
    @override
    Widget build(BuildContext context) {
    return Scaffold(
    extendBodyBehindAppBar: true,
    appBar: AppBar(
    title: Text(
    'AberGym',
    style: GoogleFonts.montserrat(
    fontSize: 35,
    color: Colors.white,
    fontWeight: FontWeight.bold,
    ),
    ),
    backgroundColor: Colors.transparent,
    centerTitle: true,
    elevation: 0,
    ),
    backgroundColor: darkgrey,
    body: Container(
    decoration: BoxDecoration(
    gradient: LinearGradient(
    begin: Alignment.topCenter,
    colors: [
    Colors.lightBlue[900]!,
    Colors.lightBlue[800]!,
    Colors.lightBlue[400]!,
    ],
    ),
    ),
    child: _nfcEnabled
    ? Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
    Text(
    "Bitte halten Sie Ihre Fitnesskarte zum Handy, um die Karte zu scannen!",
    style: GoogleFonts.montserrat(
    color: Colors.white,
    fontWeight: FontWeight.bold,
    fontSize: 25,
    ),
    ),
    GestureDetector(
    onTap: _startScan,
    child: const Text("Karte scannen"),
    ),
    if (result.value != null) ...[Text(result.value)]
    ],
    )
    : Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
    Text(
    "Bitte schalten Sie NFC ein, um die Karte zu scannen!",
    style: GoogleFonts.montserrat(
    color: Colors.white,
    fontWeight: FontWeight.bold,
    fontSize: 25,
    ),
```

```
59        textAlign: TextAlign.center,
60        ),
61        ],
62        ),
63        ),
64        );
65        }
```

In der `build`-Methode wird das Haupt-Widget der NFC-Seite erstellt. Ein `Scaffold`-Widget wird verwendet, um das grundlegende Layout der Seite zu definieren, einschließlich der AppBar und des Hintergrundfarbverlaufs. Im Hauptteil der Seite wird überprüft, ob NFC aktiviert ist. Wenn NFC aktiviert ist, wird der Benutzer*die Benutzerin aufgefordert, die Fitnesskarte zum Gerät zu halten, und ein GestureDetector-Widget erlaubt das Starten des Scanvorgangs. Andernfalls wird der Benutzer*die Benutzerin aufgefordert, NFC zu aktivieren.

Abbildung 21: NFC

4.3 Hauptmenü

Im Hauptmenü kann der Benutzer*die Benutzerin sich den aktuellen Trainingsplan ansehen, den letzten Trainingsplan ansehen oder den Trainingsplan starten.

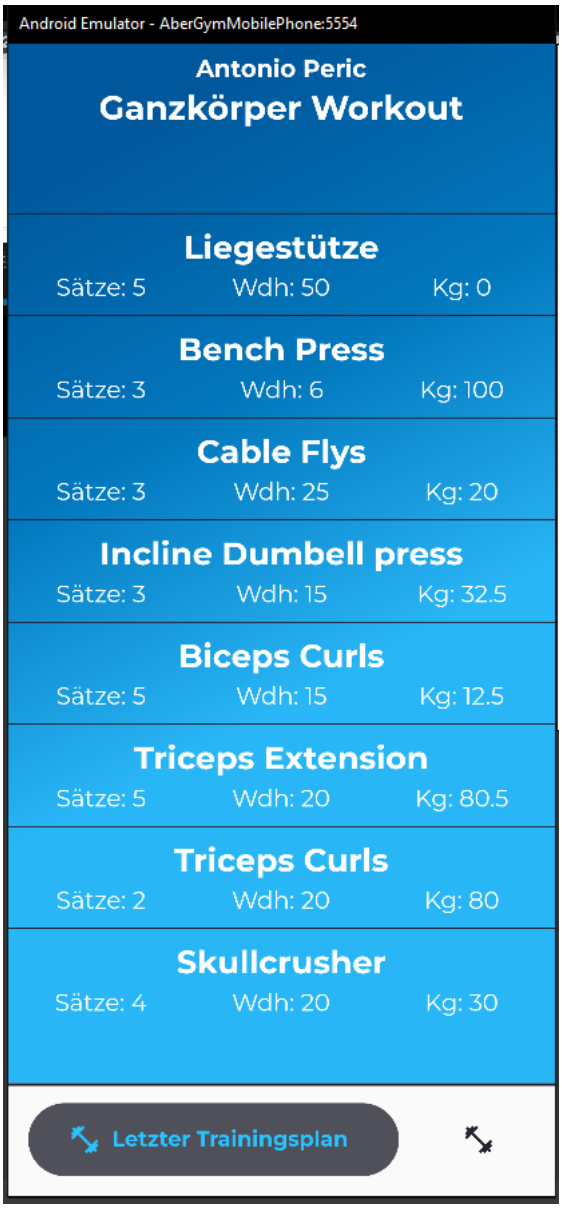

Abbildung 22: Hauptmenü

4.3.1 Layout

Listing 17: Layout Packages

```
import 'package:google_fonts/google_fonts.dart';
import 'package:google_nav_bar/google_nav_bar.dart';
```

Das Paket `googlefonts` ermöglicht die Verwendung von Google Fonts, während das Paket `googlenavbar` für die Erstellung der Navigationsleiste verwendet wird.

Listing 18: LayoutState

```
class _LayoutState extends State<Layout> {
final bodies = [const SecondPage(), const HomePage()];
int currentIndex = 1;
Color lightblue = const Color.fromARGB(255, 42, 195, 255);
Color darkgrey = const Color.fromRGBO(37, 37, 50, 1);
```

Die Klasse `LayoutState` erbt von der `State<Layout>` Basisklasse und definiert einige Eigenschaften, wie die Liste der Seiten, den aktuellen ausgewählten Index und Farben für das Design der Navigationsleiste.

Listing 19: Layout Scaffold-Widget

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

Das `Scaffold`-Widget stellt das Grundgerüst für das Layout der App bereit. Der `body` wird durch den aktuellen ausgewählten Index in der `bodies`-Liste bestimmt. Die `bottomNavigationBar` ist ein `Stack`-Widget, das die Navigationsleiste und eine Linie oben auf der Navigationsleiste enthält.

Listing 20: Navbar

```
children: [
Padding(
padding: const EdgeInsets.symmetric(horizontal: 15.0, vertical: 15),
child: GNav(
curve: Curves.easeOutSine,
color: darkgrey,
gap: 8,
iconSize: 25,
activeColor: lightblue,
tabBackgroundColor: darkgrey.withOpacity(0.8),
padding: const EdgeInsets.symmetric(vertical: 15, horizontal: 30),
duration: const Duration(milliseconds: 500),
tabs: [
GButton(
icon: Icons.fitness_center,
text: "Letzter Trainingsplan",
textStyle: GoogleFonts.montserrat(
color: lightblue,
```

```
19  fontSize: 16,
20  fontWeight: FontWeight.bold,
21  ),
22  ),
23  GButton(
24  icon: Icons.fitness_center,
25  text: "Heutiger Trainingsplan",
26  textStyle: GoogleFonts.montserrat(
27  color: lightblue,
28  fontSize: 16,
29  fontWeight: FontWeight.bold,
30  ),
31  ),
32  ],
33  selectedIndex: currentIndex,
34  onTabChange: (index) {
35      setState(() {
36      currentIndex = index;
37      });
38      },
39      ),
40      ),
41      Positioned(
42      top: 0,
43      left: 0,
44      right: 0,
45      child: Container(
46      height: 2,
47      color: darkgrey,
48      ),
49      ),
50      ],
51      ),
52      );
53      }
54      }
55  }
```

Die `GNav`-Komponente erstellt die Navigationsleiste und definiert deren Aussehen und Verhalten. Die Navigationsleiste besteht aus zwei `GButton`-Widgets, die jeweils ein Icon und einen Text enthalten. Der Stil des Textes wird mit dem GoogleFonts-Paket festgelegt. Die `selectedIndex`-Eigenschaft gibt den ausgewählten Index der Navigationsleiste an, und die `onTabChange`-Eigenschaft ist eine Funktion, die den aktuellen Index aktualisiert, wenn der Benutzer*die Benutzerin eine andere Registerkarte auswählt.

4.3.2 HomePage

Listing 21: Trainingsplan Logik

```
1  final prefs = await SharedPreferences.getInstance();
2  int? amountrows = prefs.getInt('amountrows');
3  int countTrue = 0;
4
5          if (amountrows != null) {
6            for (int i = 0; i < amountrows; i++) {
7              if (prefs.getBool('finishedExcersice_$i') == true) {
8                countTrue++;
9              }
10           }
11           if (countTrue == amountrows) {
12             for (int i = 0; i < amountrows; i++) {
13               prefs.remove('finishedExcersice_$i');
14             }
15           }
16         }
17
18         _navigateToNextScreen(context);
```

Innerhalb der `onPressed`-Funktion wird die Logik für das Starten des Trainingsplans und das Verwalten der SharedPreferences implementiert. Die SharedPreferences-Instanz wird abgerufen, und die Anzahl der Zeilen und die Anzahl der abgeschlossenen Übungen werden verwaltet. Wenn alle Übungen abgeschlossen sind, werden die SharedPreferences-Einträge entfernt. Schließlich wird die Funktion `navigateToNextScreen` aufgerufen, um zur nächsten Seite zu navigieren.

Listing 22: Trainingsplan starten Knopf

```
style: ElevatedButton.styleFrom(
backgroundColor: darkgrey.withOpacity(0.8),
shape: const RoundedRectangleBorder(
borderRadius: BorderRadius.only(
topRight: Radius.circular(0.0),
topLeft: Radius.circular(0.0),
),
),
),
child: SizedBox(
height: 216,
child: Column(
),
),
```

Für den Trainingsplan Starten Knopf wurde der Hintergrund des `ElevatedButton`-Widgets mit einer leicht transparenten dunkelgrauen Farbe festgelegt. Die Form des Buttons ist ein rechteckiges Widget mit abgerundeten Ecken an der Oberseite. Der Inhalt des Buttons besteht aus einer `SizedBox` mit einer festen Höhe und einer `Column`, die die Text- und Icon-Widgets enthält.

Listing 23: HomePage Navigation

```
void _navigateToNextScreen(BuildContext context) {
Navigator.of(context)
.push(MaterialPageRoute(builder: (context) => const LayoutTDL()));
}
```

Die Navigationsfunktion `navigateToNextScreen` wird definiert, um zur nächsten Seite zu navigieren.Die Funktion `navigateToNextScreen` akzeptiert ein `BuildContext`-Objekt und verwendet den `Navigator` zum Navigieren zur nächsten Seite, die durch das `LayoutTDL`-Widget dargestellt wird.

4.3.3 SecondPage

Listing 24: Second Page

```
1          class SecondPage extends StatelessWidget {
2    const SecondPage({super.key});
3
4    @override
5    Widget build(BuildContext context) {
6      return SqlTable(version: 0);
7    }
8  }
```

Die SecondPage Methode lädt, wenn die "Letzter TrainingsplanSSeite ausgewählt wird, aus der Datenbank den letzten Trainingsplan.

4.4 To-Do-Liste

Die To-Do-Liste ist eine interaktive Liste aus Übungen die im Trainingsplan enthalten sind. Durch das Auswählen einer Übung in dieser Liste, wird man an die Satzzähler Seite weitergeleitet. Wenn man eine Übung abgeschlossen hat, wird die Übung in der Liste ausgegraut und an letzter Stelle der Liste verschoben.

Abbildung 23: To-Do-Liste

Listing 25: To-Do-Liste Trainingsplan Variablen

```
late String wname = "";
late List<String> wereps = [];
late List<String> wesets = [];
late List<String> weweight = [];
late List<String> ename = [];
late int amountrows = 0;
final Color lightblue = const Color.fromARGB(255, 42, 195, 255);
final Color darkgrey = const Color.fromRGBO(37, 37, 50, 1);
List<bool> finishedExcersiceList = [];
int counter = 0;
bool finished = false;
```

Diese Variablen enthalten Informationen über die Übungen, die Anzahl der Zeilen, Farben und ob eine Übung abgeschlossen ist oder nicht.

Listing 26: To-Do-Liste get Methode

```
void getData() async {
    final prefs = await SharedPreferences.getInstance();
    wname = prefs.getString('wname')!;
    wereps = prefs.getStringList('wereps')!;
    wesets = prefs.getStringList('wesets')!;
    weweight = prefs.getStringList('weweight')!;
    ename = prefs.getStringList('ename')!;
    amountrows = prefs.getInt('amountrows')!;

    for (int i = 0; i < amountrows; i++) {
      if (prefs.get('finishedExcersice_$i') == true) {
        finishedExcersiceList.add(true);
        setState(() {
          counter++;
        });
      } else {
        finishedExcersiceList.add(false);
      }
    }
    if (counter == amountrows) {
      setState(() {
        finished = true;
      });
    }
    setState(() {
      wname = wname;
      wereps = wereps;
      wesets = wesets;
      wesets = wesets;
      weweight = weweight;
      ename = ename;
      amountrows = amountrows;
    });
  }
```

Die Methode `getData` liest die Daten aus SharedPreferences und aktualisiert die Zustände der ToDoList-Klasse.

Listing 27: To-Do-Liste Item Methode

```
Container toDoListItem(int i, bool isFinished) {
    return Container(
      height: 78,
      margin: const EdgeInsets.only(
        top: 3.8,
        right: 12,
        left: 12,
        bottom: 3.8,
      ),
      decoration: isFinished
          ? BoxDecoration(
              borderRadius: BorderRadius.circular(15.0),
              color: const Color.fromARGB(255, 25, 68, 85),
            )
          : BoxDecoration(
              borderRadius: BorderRadius.circular(15.0),
              color: lightblue,
            ),
      child: Column(
        children: [
          Container(
            padding: const EdgeInsets.all(8.0),
            child: Text(
              (ename.isNotEmpty ? ename[i].toString() : ""),
              textScaleFactor: 2,
              style: isFinished
                  ? GoogleFonts.montserrat(
                      fontSize: 11,
                      color: darkgrey,
                      fontWeight: FontWeight.bold,
                    )
                  : GoogleFonts.montserrat(
                      fontSize: 11,
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                    ),
              textAlign: TextAlign.center,
            ),
          ),
          Table(
            children: [
              TableRow(
                children: [
                  Center(
                    child: Text(
                      (wesets.isNotEmpty
                          ? 'Saetze: ${wesets[i].toString()}'
                          : ""),
                      textScaleFactor: 1.5,
                      style: GoogleFonts.montserrat(
                        fontSize: 11,
                        color: darkgrey,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                  Center(
                    child: Text(
                      (wereps.isNotEmpty ? 'Wdh: ${wereps[i].toString()}'
                          : ""),
                      textScaleFactor: 1.5,
                      style: GoogleFonts.montserrat(
                        fontSize: 11,
                        color: darkgrey,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                  Center(
                    child: Text(
                      (weweight.isNotEmpty
                          ? 'Kg: ${weweight[i].toString()}'
                          : ""),
                      textScaleFactor: 1.5,
                      style: GoogleFonts.montserrat(
                        fontSize: 11,
                        color: darkgrey,
                        fontWeight: FontWeight.bold,
                      ),
```

```
79                                ),
80                              ),
81                            ],
82                          ),
83                        ],
84                      ),
85                    ],
86                  ),
87                );
88              }
```

Die `toDoListItem`-Methode erstellt ein individuelles Aufgabenlisten-Element basierend auf einem gegebenen Index und ob es abgeschlossen ist oder nicht.

4.5 Satzzähler

Der Satzzähler ermöglicht es beim Trainieren, die Sätze der aktuell ausgewählten Übung mitzuzählen und somit die Übung abzuschließen.

Abbildung 24: Satzzähler

Listing 28: SetCounter

```
1         class SetCounter extends StatefulWidget {
2         int index = 0;
3         SetCounter({super.key, required this.index});
4
5         @override
6         State<SetCounter> createState() => _SetCounterState(index);
7         }
```

Die Klasse SetCounter erbt von StatefulWidget und nimmt den Parameter index entgegen. Dieser Index gibt die Position der aktuellen Übung in der Liste an. Der Konstruktor

der Klasse speichert den Index und übergibt ihn an den State der Klasse, SetCounterState.

Listing 29: SetCounterState

```
class _SetCounterState extends State<SetCounter> {
int index = 0;
late List<String> werepsList = [];
late List<String> wesetsList = [];
late List<String> weweightList = [];
late List<String> enameList = [];
String wereps = "";
String wesets = "";
String weweight = "";
String ename = "";
Color lightblue = const Color.fromARGB(255, 42, 195, 255);
Color darkgrey = const Color.fromRGBO(37, 37, 50, 1);
int counter = 0;
late bool finishedExcersice;
double? fontSizeRows = 13;
double? fontPixelsRows = 1.5;

_SetCounterState(this.index);
}
```

Die Klasse SetCounterState speichert den index und verwaltet die Zustände der App. Es gibt verschiedene Zustandsvariablen für die Anzahl der Wiederholungen, Sätze, das Gewicht und den Namen der Übung.

Listing 30: Increase Counter Funktion

```
void increaseCounter() async {
final prefs = await SharedPreferences.getInstance();
setState(() => counter++);
if (counter > int.parse(wesets)) {
finishedExcersice = true;
prefs.setBool('finishedExcersice$index', finishedExcersice);
}
}
```

Die increaseCounter-Funktion erhöht den Satzzähler und aktualisiert die Benutzeroberfläche. Wenn die Anzahl der abgeschlossenen Sätze größer ist als die vorgegebene Anzahl der Sätze, wird die Variable finishedExcersice auf true gesetzt und in den SharedPreferences gespeichert.

Listing 31: initState

```
@override
void initState() {
super.initState();
getData();
}
```

Die initState()-Funktion wird aufgerufen, wenn der Zustand des Widgets initialisiert wird. Hier wird die Funktion getData() aufgerufen, um die Daten für die aktuelle Übung zu laden.

Listing 32: GestureDetector

```
GestureDetector(
    onTap: _increaseCounter,
    child: Container(
      padding: const EdgeInsets.only(top: 65),
      margin: const EdgeInsets.all(50.0),
      width: 300.0,
      height: 300.0,
      decoration: BoxDecoration(
        gradient: LinearGradient(
          begin: Alignment.centerLeft,
          colors: [
            Colors.lightBlue[900]!,
            Colors.lightBlue[800]!,
            Colors.lightBlue[400]!,
          ],
        ),
        shape: BoxShape.circle,
      ),
      child: Column(
        children: [
          Text(
            '$counter/$wesets',
            style: GoogleFonts.montserrat(
              color: Colors.white,
              fontSize: 100,
            ),
          ),
          Text(
            "Erledigte Saetze",
            style: GoogleFonts.montserrat(
              color: Colors.white,
              fontSize: 25,
              fontWeight: FontWeight.bold,
            ),
          ),
        ],
      ),
    ),
  ),
```

Ein GestureDetector wird verwendet, um auf Berührungen des Benutzers zu reagieren und den Satzzähler entsprechend zu erhöhen. Wenn der Benutzer auf den Container tippt, wird die increaseCounter-Funktion aufgerufen.

Listing 33: GestureDetector zum Schluss

```
GestureDetector(
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => const ToDoList(),
        ),
      );
    },
    child: Container(
      padding: const EdgeInsets.only(top: 65),
      margin: const EdgeInsets.all(50.0),
      width: 300.0,
      height: 300.0,
      decoration: BoxDecoration(
```

```
16                          gradient: LinearGradient(
17                            begin: Alignment.centerLeft,
18                            colors: [
19                              Colors.lightBlue[900]!,
20                              Colors.lightBlue[800]!,
21                              Colors.lightBlue[400]!,
22                            ],
23                          ),
24                          shape: BoxShape.circle,
25                        ),
26                        child: Column(
27                          children: [
28                            InkWell(
29                              onTap: () {
30                                Navigator.push(
31                                  context,
32                                  MaterialPageRoute(
33                                    builder: (context) => const ToDoList(),
34                                  ),
35                                );
36                              },
37                              child: Column(
38                                children: <Widget>[
39                                  Text(
40                                    "Uebung",
41                                    style: GoogleFonts.montserrat(
42                                      color: Colors.white,
43                                      fontSize: 70,
44                                    ),
45                                  ),
46                                  Text(
47                                    "Fertig",
48                                    style: GoogleFonts.montserrat(
49                                      color: Colors.white,
50                                      fontSize: 65,
51                                    ),
52                                  ),
53                                ],
54                              ),
55                            ),
56                          ],
57                        ),
58                      ),
59                    ),
```

Wenn die Übung abgeschlossen ist (wenn die Anzahl der abgeschlossenen Sätze gleich oder größer als die vorgegebene Anzahl ist), wird ein anderer GestureDetector und Container angezeigt, der dem Benutzer signalisiert, dass die Übung beendet ist. Bei einem Tippen auf den Container wird der Benutzer zur ToDoList-Seite weitergeleitet.

4.6 Trainingplan bearbeiten

Durch langes gedrückt halten einer Übung in der To-Do-Liste wird die Bearbeitungsseite geöffnet und die ausgewählte Übung kann bearbeitet werden. Es können die Wiederholungen, die Sätze und das Gewicht geändert werden, um während des Trainings flexibel auf eigene Wünsche das Training anpassen zu können.

Abbildung 25: Trainingplan bearbeiten

Listing 34: Bearbeiten Index

```
class UpdateExcersice extends StatefulWidget {
int index = 0;
UpdateExcersice({super.key, required this.index});

@override
State<UpdateExcersice> createState() => _UpdateExcersiceState(index);
}
```

Die UpdateExcersice-Klasse erbt von StatefulWidget und speichert den index der aktuellen Übung. Der Konstruktor nimmt den index als erforderliches Argument an und übergibt diesen an die UpdateExcersiceState-Klasse.

Listing 35: Bearbeiten UpdateExcersiceState

```
class _UpdateExcersiceState extends State<UpdateExcersice> {
  int index = 0;
  late List<String> werepsList = [];
  late List<String> wesetsList = [];
  late List<String> weweightList = [];
  late List<String> enameList = [];
  String wereps = "";
  String wesets = "";
  String weweight = "";
  String ename = "";
  Color lightblue = const Color.fromARGB(255, 42, 195, 255);
  Color darkgrey = const Color.fromRGBO(37, 37, 50, 1);
  String _sets = '';
  String _reps = '';
  String _kg = '';
  final setsController = TextEditingController();
  final repsController = TextEditingController();
  final kgController = TextEditingController();

  _UpdateExcersiceState(this.index);

```

```
22                  void getData() async {
23                    final prefs = await SharedPreferences.getInstance();
24                    werepsList = prefs.getStringList('wereps')!;
25                    wesetsList = prefs.getStringList('wesets')!;
26                    weweightList = prefs.getStringList('weweight')!;
27                    enameList = prefs.getStringList('ename')!;
28
29                    setState(() {
30                      wereps = werepsList[index];
31                      wesets = wesetsList[index];
32                      weweight = weweightList[index];
33                      ename = enameList[index];
34                      setsController.text = wesets;
35                      repsController.text = wereps;
36                      kgController.text = weweight;
37                    });
38                  }}
```

Die UpdateExcersiceState-Klasse erweitert die State-Klasse für das UpdateExcersice-Widget. Innerhalb dieser Klasse werden verschiedene Variablen und Methoden definiert, um Daten zu laden, zu speichern und die Benutzeroberfläche aufzubauen.

Listing 36: Bearbeiten initState getData

```
1         void initState() {
2         super.initState();
3         getData();
4         }
```

Die initState-Methode wird aufgerufen, wenn der Zustand des Widgets initialisiert wird. Hier wird die getData-Methode aufgerufen, um die aktuellen Daten der Übung zu laden.

Listing 37: Bearbeiten Build

```
1         @override
2         Widget build(BuildContext context) {
3             @override
4             Widget build(BuildContext context) {
5               return Scaffold(
6                 extendBodyBehindAppBar: true,
7                 appBar: AppBar(
8                   title: Text(
9                     'AberGym',
10                    style: GoogleFonts.montserrat(
11                      fontSize: 35,
12                      color: Colors.white,
13                      fontWeight: FontWeight.bold,
14                    ),
15                  ),
16                  backgroundColor: Colors.transparent,
17                  centerTitle: true,
18                  elevation: 0,
19                ),
20                body: Container(
21                  padding: const EdgeInsets.only(top: 110),
22                  decoration: BoxDecoration(
23                    gradient: LinearGradient(
24                      begin: Alignment.centerLeft,
25                      colors: [
26                        Colors.lightBlue[200]!,
27                        Colors.lightBlue[500]!,
28                        Colors.lightBlue[900]!,
29                      ],
30                    ),
31                  ),
32                  child: Center(
33                    child: Column(
```

```
34                          children: [
35                            Container(
36                              padding: const EdgeInsets.all(10.0),
37                              child: Text(
38                                "Aenderungen bitte Eingeben:",
39                                style: GoogleFonts.montserrat(
40                                  fontSize: 24,
41                                  color: Colors.white,
42                                  fontWeight: FontWeight.bold,
43                                ),
44                                textAlign: TextAlign.center,
45                              ),
46                            ),
47                            Container(
48                              padding: const EdgeInsets.only(bottom: 15.0, top: 25),
49                              child: Text(
50                                'Uebung: $ename',
51                                style: GoogleFonts.montserrat(
52                                  fontSize: 24,
53                                  color: Colors.white,
54                                  fontWeight: FontWeight.bold,
55                                ),
56                                textAlign: TextAlign.center,
57                              ),
58                            ),
59                            // Textfeld 1
60                            Container(
61                              margin: const EdgeInsets.all(5),
62                              decoration: BoxDecoration(
63                                borderRadius: BorderRadius.circular(5),
64                                border: Border.all(color: Colors.white),
65                              ),
66                              child: TextField(
67                                keyboardType: TextInputType.number,
68                                decoration: InputDecoration(
69                                  labelText: 'Saetze',
70                                  labelStyle: GoogleFonts.montserrat(
71                                    fontSize: 24,
72                                    color: Colors.white,
73                                    fontWeight: FontWeight.bold,
74                                  ),
75                                ),
76                                controller: setsController,
77                                onChanged: (value) {
78                                  _saveInput();
79                                },
80                                onTap: () {
81                                  setsController.clear();
82                                },
83                                textAlign: TextAlign.center,
84                                style: GoogleFonts.montserrat(
85                                  fontSize: 24,
86                                  color: Colors.white,
87                                  fontWeight: FontWeight.bold,
88                                ),
89                              ),
90                            ),
91                            // Textfeld 2
92                            Container(
93                              margin: const EdgeInsets.all(5),
94                              decoration: BoxDecoration(
95                                borderRadius: BorderRadius.circular(5),
96                                border: Border.all(color: Colors.white),
97                              ),
98                              child: TextField(
99                                keyboardType: TextInputType.number,
100                               decoration: InputDecoration(
101                                 labelText: 'Wiederholungen',
102                                 labelStyle: GoogleFonts.montserrat(
103                                   fontSize: 24,
104                                   color: Colors.white,
105                                   fontWeight: FontWeight.bold,
106                                 ),
107                               ),
108                               controller: repsController,
109                               onChanged: (value) {
110                                 _saveInput();
111                               },
112                               onTap: () {
113                                 repsController.clear();
114                               },
115                               textAlign: TextAlign.center,
```

```
116                             style: GoogleFonts.montserrat(
117                               fontSize: 24,
118                               color: Colors.white,
119                               fontWeight: FontWeight.bold,
120                             ),
121                           ),
122                         ),
123                         // Textfeld 3
124                         Container(
125                           margin: const EdgeInsets.all(5),
126                           decoration: BoxDecoration(
127                             borderRadius: BorderRadius.circular(5),
128                             border: Border.all(color: Colors.white),
129                           ),
130                           child: TextField(
131                             keyboardType: TextInputType.number,
132                             decoration: InputDecoration(
133                               labelText: 'Kg',
134                               labelStyle: GoogleFonts.montserrat(
135                                 fontSize: 24,
136                                 color: Colors.white,
137                                 fontWeight: FontWeight.bold,
138                               ),
139                             ),
140                             controller: kgController,
141                             onChanged: (value) {
142                               _saveInput();
143                             },
144                             onTap: () {
145                               kgController.clear();
146                             },
147                             textAlign: TextAlign.center,
148                             style: GoogleFonts.montserrat(
149                               fontSize: 24,
150                               color: Colors.white,
151                               fontWeight: FontWeight.bold,
152                             ),
153                           ),
154                         ),
155                         GestureDetector(
156                           onTap: () async {
157                             await saveInputInCache();
158                             Navigator.push(
159                               context,
160                               PageTransition(
161                                 type: PageTransitionType.leftToRight,
162                                 child: const LayoutTDL(),
163                               ),
164                             );
165                           },
166                           child: Container(
167                             margin: const EdgeInsets.only(top: 25),
168                             width: 300,
169                             height: 50,
170                             decoration: BoxDecoration(
171                               color: Colors.lightBlue[900]!,
172                               borderRadius: BorderRadius.circular(15),
173                             ),
174                             child: Row(
175                               mainAxisAlignment: MainAxisAlignment.center,
176                               children: [
177                                 Text(
178                                   'Speichern',
179                                   style: GoogleFonts.montserrat(
180                                     fontSize: 16,
181                                     color: Colors.white,
182                                     fontWeight: FontWeight.bold,
183                                   ),
184                                 ),
185                               ],
186                             ),
187                           ),
188                         ),
189                       ],
190                     ),
191                   ),
192                 ),
193               );
194             }
195           }
```

Die build-Methode erstellt die Benutzeroberfläche für das Widget. In dieser Methode wird ein Scaffold-Widget verwendet, um die grundlegende Struktur der App zu definieren. Die Benutzeroberfläche besteht aus mehreren Eingabefeldern und Schaltflächen, die den Benutzer*innen ermöglichen, ihre Änderungen einzugeben und zu speichern.

Listing 38: Bearbeiten saveInput

```
1          void _saveInput() {
2          setState(() {
3          _sets = setsController.text;
4          _reps = repsController.text;
5          _kg = kgController.text;
6          });
7          }
```

Die saveInput-Methode speichert die Benutzereingaben in den entsprechenden Variablen. Diese Methode wird aufgerufen, wenn der Inhalt eines der Eingabefelder geändert wird.

Listing 39: Bearbeiten saveInputInCache

```
1              Future<void> saveInputInCache() async {
2                    await conn.connect();
3                    await conn.execute(
4                      'UPDATE WorkoutExercise we JOIN Exercise e ON we.exercise_id =
                           e.id SET we.‘sets‘ = :sets, we.weight = :kg, we.reps =
                           :reps WHERE e.name = :name AND we.workoutplan_id = (SELECT
                           MAX(id) FROM Workoutplan)',
5                      {"sets": _sets, "kg": _kg, "reps": _reps, "name": ename},
6                    );
7                    wesetsList[index] = _sets;
8                    weweightList[index] = _kg;
9                    werepsList[index] = _reps;
10                   final prefs = await SharedPreferences.getInstance();
11
12                   await prefs.remove('wesets');
13                   await prefs.remove('weweight');
14                   await prefs.remove('wereps');
15
16                   await prefs.setStringList('wesets', wesetsList);
17                   await prefs.setStringList('weweight', weweightList);
18                   await prefs.setStringList('wereps', werepsList);
19
20                   conn.close();
21               }
```

Die saveInputInCache-Methode speichert die Benutzereingaben in den SharedPreferences und aktualisiert die Datenbank. Diese Methode wird aufgerufen, wenn der Benutzer auf die Schaltfläche SSpeichern"tippt.

5 Backend

5.1 Temporäres Backend für die Mobile Anwendung

5.1.1 IntelliJ Projekt

In der vorliegenden Arbeit wurde ein temporäres Backend mithilfe der Technologieumgebung IntelliJ [3] und der Programmiersprache Java [4] entwickelt. Bei der Umsetzung wurde besonderes Augenmerk auf eine präzise Definition der einzelnen Entitäten, eine klare Setzung der Attribute sowie eine sorgfältige Pflege der Beziehungen zwischen den Entitäten gelegt.

Zu Beginn des Entwicklungsprozesses wurde die Funktionalität des temporären Backends genau spezifiziert und in mehrere Entitäten aufgeteilt. Anschließend wurden für jede Entität ihre Eigenschaften und Attribute definiert, um eine klare Struktur und einheitliche Verarbeitung innerhalb des Systems zu gewährleisten.

Im weiteren Verlauf der Entwicklung wurden die Beziehungen zwischen den einzelnen Entitäten gepflegt und mit Hilfe von Referenzschlüsseln miteinander verknüpft. Hierbei wurde darauf geachtet, dass die Verknüpfungen korrekt und eindeutig sind, um eine reibungslose Funktionalität des temporären Backends zu gewährleisten.

Zusammenfassend lässt sich feststellen, dass das temporäre Backend erfolgreich unter Verwendung der Technologieumgebung IntelliJ und der Programmiersprache Java entwickelt wurde. Durch eine präzise Definition der Entitäten, eine klare Setzung der Attribute und eine sorgfältige Pflege der Beziehungen zwischen den Entitäten konnte eine hohe Qualität und Zuverlässigkeit des Systems erreicht werden.

Listing 40: Entity | Person

```
@Entity
public class Exercise {
    //Attributes
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    @Column
    public String name;
    @Column(name = "muscle_group")
    public String muscleGroup;
    //Navigation
    @OneToMany(mappedBy = "exercise", fetch = FetchType.EAGER)
    public Set<WorkoutExercise> workoutExcersices = new HashSet<>();
}
```

Listing 41: Entity | Trainee

```
@Entity
public class Trainee extends Person{
    //Navigation
    @OneToMany(mappedBy = "trainee", fetch = FetchType.EAGER)
    public Set<Workoutplan> workoutPlanList = new HashSet<>();
}
```

Listing 42: Entity | Trainer

```
@Entity
public class Trainer extends Person{
    //Navigation
    @OneToMany(mappedBy = "trainer", fetch = FetchType.EAGER)
    public List<Template> templateList;
}
```

Listing 43: Entity | Template

```
@Entity
public class Template {
    //Attributes
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    @Column
    public String name;
    //Navigation
    @ManyToOne
    @JoinColumn(name="trainer_id", nullable=false)
    public Trainer trainer;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(
            name = "Template_Exercise", // name of the association table
            joinColumns = @JoinColumn(name = "template_id"), // foreign key
                columns
            inverseJoinColumns = @JoinColumn(name = "exercise_id"))
    private Set<Exercise> exercise = new HashSet<>();
}
```

Listing 44: Entity | Workoutplan

```
@Entity
public class Workoutplan {
    //Attributes
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    @Column
    public String name;
    //Navigation
    @ManyToOne
    @JoinColumn(name="trainee_id", nullable=false)
    public Trainee trainee;
    @OneToMany(mappedBy = "workoutplan", fetch = FetchType.EAGER)
    public Set<WorkoutExercise> workoutExcersices = new HashSet<>();
}
```

Listing 45: Entity | WorkoutExersice

```
@Entity
public class WorkoutExercise {
    //Attributes
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    @Column
    public Integer sets;
    @Column
    public Double weight;
    @Column
    public Integer reps;
    @Column
    public Double time;
    //Navigation
    @ManyToOne
    @JoinColumn(name="workoutplan_id", nullable=false)
    public Workoutplan workoutplan;
    @ManyToOne
    @JoinColumn(name="exercise_id", nullable=false)
    public Exercise exercise;
}
```

Listing 46: Entity | Exersice

```
@Entity
public class Exercise {
    //Attributes
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    @Column
    public String name;
    @Column(name = "muscle_group")
    public String muscleGroup;
    //Navigation
    @OneToMany(mappedBy = "exercise", fetch = FetchType.EAGER)
    public Set<WorkoutExercise> workoutExcersices = new HashSet<>();
}
```

5.1.2 Packages des IntelliJ Projekts

Listing 47: Dependency | reactive-mysql-client

```
1       <dependency>
2         <groupId>io.quarkus</groupId>
3         <artifactId>quarkus-reactive-mysql-client</artifactId>
4       </dependency>
```

Das Package "quarkus-reactive-mysql-client"[5] ist eine Abhängigkeit, die von der Software-Entwicklungsumgebung IntelliJ bereitgestellt wird. Diese Abhängigkeit wird für die Verbindung zu einer MySQL-Datenbank verwendet und ermöglicht es, auf eine reaktive Weise auf die Datenbank zuzugreifen.

Das Package basiert auf dem Quarkus-Framework, welches für die Entwicklung von Java-basierten Anwendungen verwendet wird. Es ermöglicht eine schnelle und effiziente Entwicklung von Microservices und bietet dabei eine hohe Flexibilität in der Wahl der verwendeten Technologien.

Die Verwendung des "quarkus-reactive-mysql-clientPackages bietet eine Vielzahl von Vorteilen. Durch die Verwendung von Reactive-Streams können Daten asynchron verarbeitet werden, was zu einer besseren Skalierbarkeit und Leistung der Anwendung führt. Darüber hinaus ermöglicht es die Verwendung von SQL-Abfragen, um auf die Daten in der MySQL-Datenbank zuzugreifen und diese zu manipulieren.

Listing 48: Dependency | smallrye-openapi

```
1       <dependency>
2         <groupId>io.quarkus</groupId>
3         <artifactId>quarkus-smallrye-openapi</artifactId>
4       </dependency>
```

Das oben genannte Package "quarkus-smallrye-openapi"[6] ist eine Abhängigkeit, die von der Software-Entwicklungsumgebung IntelliJ bereitgestellt wird. Diese Abhängigkeit wird für die Generierung von OpenAPI-Dokumentationen in einer Quarkus-basierten Anwendung verwendet.

Das Package basiert auf dem Quarkus-Framework, welches für die Entwicklung von Java-basierten Anwendungen verwendet wird. Es ermöglicht eine schnelle und effiziente Entwicklung von Microservices und bietet dabei eine hohe Flexibilität in der Wahl der verwendeten Technologien.

Die Verwendung des "quarkus-smallrye-openapiPackages bietet eine Vielzahl von Vorteilen. Es erleichtert die Dokumentation von APIs und bietet eine automatisierte Möglichkeit, eine OpenAPI-Dokumentation zu generieren. Dadurch können Entwicklerinnen und Entwickler schnell und einfach eine Dokumentation erstellen, die es anderen Entwicklerinnen und Entwicklern erleichtert, die API zu verstehen und zu verwenden.

Darüber hinaus ermöglicht das Package die Verwendung von Annotations, um die API-Endpunkte und deren Parameter zu dokumentieren. Dies erleichtert die Integration mit anderen Tools wie Swagger UI, um die API-Dokumentationen zu visualisieren.

Listing 49: Dependency | jdbc-mysql

```
1       <dependency>
2         <groupId>io.quarkus</groupId>
3         <artifactId>quarkus-jdbc-mysql</artifactId>
4       </dependency>
```

"quarkus-jdbc-mysql"[7] ist eine Abhängigkeit, die von der Software-Entwicklungsumgebung IntelliJ bereitgestellt wird. Diese Abhängigkeit wird verwendet, um eine Verbindung zu einer MySQL-Datenbank herzustellen und ermöglicht es, auf eine standardmäßige Weise auf die Datenbank zuzugreifen.

Das Package basiert auf dem Quarkus-Framework, welches für die Entwicklung von Java-basierten Anwendungen verwendet wird. Es bietet dabei eine hohe Flexibilität in der Wahl der verwendeten Technologien und ermöglicht eine schnelle und effiziente Entwicklung von Microservices.

Die Verwendung des "quarkus-jdbc-mysqlPackages bietet eine Vielzahl von Vorteilen. Es ermöglicht die Verwendung von JDBC, um auf die Daten in der MySQL-Datenbank zuzugreifen und diese zu manipulieren. Darüber hinaus bietet es eine standardmäßige Möglichkeit, eine Verbindung zur Datenbank herzustellen und Abfragen auszuführen.

Listing 50: Dependency | hibernate-orm-panache

```
1       <dependency>
2       <groupId>io.quarkus</groupId>
3       <artifactId>quarkus-hibernate-orm-panache</artifactId>
4       <version>2.9.2.Final</version>
5   </dependency>
```

Dieses Package "quarkus-hibernate-orm-panache"[8] ist eine Abhängigkeit, die von der Software-Entwicklungsumgebung IntelliJ bereitgestellt wird. Dieses Package basiert auf dem Quarkus-Framework und bietet eine Implementierung des Hibernate Object Relational Mapping (ORM) mit Panache, einem vereinfachten und ausdrucksstarken Ansatz für die Datenbankanbindung in Java-basierten Anwendungen.

Die Verwendung des "quarkus-hibernate-orm-panachePackages bietet eine Vielzahl von Vorteilen. Es ermöglicht die einfache Verbindung mit einer Datenbank, indem es eine Abstraktionsschicht bereitstellt, die es ermöglicht, Datenbankabfragen in einer vereinfachten Weise zu formulieren. Durch die Verwendung von Panache-Entitäten können die Entwickler schnell und einfach die Verbindung mit der Datenbank herstellen und CRUD-Operationen (Create, Read, Update, Delete) durchführen.

Das Package bietet auch eine einfache Möglichkeit, um effektiv mit der Datenbank zu arbeiten und sichere und zuverlässige Abfragen zu formulieren. Durch die Verwendung von Annotationen können Entitäten schnell und einfach definiert und mit den Tabellen in der Datenbank verbunden werden.

Listing 51: Dependency | lombok

```
1       <dependency>
2         <groupId>org.projectlombok</groupId>
3         <artifactId>lombok</artifactId>
4       </dependency>
```

Das Package "lombokïst eine Abhängigkeit, die von der Software-Entwicklungsumgebung IntelliJ bereitgestellt wird. Diese Abhängigkeit ermöglicht es, boilerplate-Code in Java-Anwendungen zu reduzieren und die Entwicklungszeit zu verkürzen.

Das Package basiert auf dem Prinzip der Annotationen und bietet eine Vielzahl von Annotationen, die verwendet werden können, um wiederkehrende Aufgaben wie das Erstellen von Getter- und Setter-Methoden oder das Implementieren von Equals- und Hashcode-Methoden automatisch zu erledigen.

Die Verwendung des "lombokPackages bietet eine Vielzahl von Vorteilen. Zum einen reduziert es den Codeumfang und erhöht dadurch die Lesbarkeit und Wartbarkeit der Anwendung. Zum anderen verkürzt es die Entwicklungszeit, da der Entwickler sich nicht um das Schreiben von boilerplate-Code kümmern muss und sich stattdessen auf die Implementierung der tatsächlichen Funktionalität konzentrieren kann.

5.1.3 MySql Datenbank

MySQL [9] ist eine relationale Datenbank, die als Open-Source-System für die Verwaltung und Speicherung von Daten eingesetzt wird. Sie ist besonders für Webanwendungen und Online-Datenbanken geeignet und bietet eine breite Palette von Funktionen.

Zu den Vorteilen von MySQL gehört, dass es kostenlos und einfach zu verwenden ist. Es kann auf verschiedenen Betriebssystemen wie Windows, Linux und MacOS betrieben werden und ist sehr flexibel und skalierbar. Die Datenbank kann schnell und einfach installiert und eingerichtet werden und bietet eine Vielzahl von Funktionen für die effiziente Verwaltung von Daten. MySQL ist auch sehr zuverlässig und bietet eine hohe Performance. Es unterstützt mehrere gleichzeitige Verbindungen und kann daher von vielen Benutzern gleichzeitig genutzt werden.

Darüber hinaus bietet MySQL eine gute Sicherheit durch verschiedene Methoden zur Authentifizierung und Verschlüsselung von Daten. Die Datenbank ist auch sehr flexibel und kann für verschiedene Anwendungsfälle angepasst werden. Es gibt auch eine aktive Community von Entwicklern und Benutzern, die helfen, Probleme zu lösen und neue Funktionen zu implementieren.

Zu den Nachteilen von MySQL gehört, dass es einige Einschränkungen bei der Skalierung gibt. Es kann schwierig sein, MySQL auf große Datenmengen zu skalieren, und es kann auch langsam werden, wenn die Datenbank überlastet ist. Zudem ist es nicht so robust wie andere Datenbanksysteme und bietet nicht so viele Funktionen wie einige der kommerziellen Datenbanksysteme.

5.1.4 Erstellung der Datenbank

Listing 52: Erstellung der Datenbank mittels Docker Befehl

```
1       docker run --name abergymmobile -p 3306:3306 -e
            MYSQL_ROOT_PASSWORD=abergymmobile_kp -d mysql:latest
```

Der Befehl "docker runïst ein Befehl der Containerisierungstechnologie Docker, mit dem ein neuer Container gestartet werden kann. Der Parameter -nameërmöglicht es dem Benutzer, dem Container einen eindeutigen Namen zu geben. Im vorliegenden Beispiel wird der Container äbergymmobile"genannt [10].

Der Parameter p"gibt an, dass der Container auf dem Host-Port 3306 gehostet wird. Der Host-Port wird auf den Container-Port 3306 weitergeleitet. Dies ist notwendig, um auf den MySQL-Server innerhalb des Containers zugreifen zu können.

Der Parameter `-e` gibt an, dass eine Umgebungsvariable gesetzt werden soll. Im vorliegenden Beispiel wird die Umgebungsvariable `MYSQL_ROOT_PASSWORD` auf den Wert `"**********"` gesetzt. Dies ist das Passwort für den Root-Benutzer des MySQL-Servers innerhalb des Containers.

Der Parameter d"gibt an, dass der Container im Hintergrund ausgeführt werden soll. Der Parameter "mysql:latest"gibt an, dass der Container auf der neuesten Version des offiziellen MySQL-Images basiert.

5.1.5 ERD Erklärung

Die Datenbank des Trainingsplan-Verwaltungssystems ist ein wesentlicher Bestandteil des gesamten Systems, da sie die zentrale Plattform für die Speicherung, Verwaltung und Abfrage von Trainingsplänen und damit verbundenen Daten darstellt. Die korrekte Verwendung und Optimierung dieser Datenbank ist entscheidend für die Effizienz und Leistung des Systems.

Eine der Hauptfunktionen der Datenbank ist es, sicherzustellen, dass jeder*jede Benutzer*inn des Systems über eine eindeutige Identifikation verfügt. Die "PersonËntity ist die Grundlage für diese Identifikation, da sie alle Benutzer*innen des Systems repräsentiert und Informationen wie den Benutzernamen und die Karten-ID enthält. Diese Informationen werden bei der Anmeldung des Benutzers verwendet, um sicherzustellen, dass sie auf den richtigen Trainingsplan zugreifen.

Die "TraineeËntity ist eng mit der "PersonËntity verbunden und repräsentiert die Benutzer des Systems, die Trainingspläne verwenden. Die "TrainerËntity repräsentiert hingegen die Benutzer*innen, die Trainingspläne erstellen und betreuen. Die Verknüpfung dieser beiden Entities ermöglicht es den Trainern, spezifische Trainingspläne für Trainees zu erstellen und zuzuweisen, während die Trainees in der Lage sind, auf ihre eigenen Trainingspläne zuzugreifen und diese durchzuarbeiten.

Die "WorkoutplanËntity enthält den Namen des spezifischen Trainingsplans, auf den sich die Trainees beziehen. Die "TemplateËntity dient als Vorlage für die Erstellung von Trainingsplänen und enthält eine Liste von Übungen sowie Informationen zur Dauer, Intensität und Anzahl der Sätze. Trainer können Templates erstellen und als Grundlage für neue Trainingspläne verwenden.

Die "WorkoutExerciseËntity enthält alle Informationen zu den Übungen innerhalb eines Trainingsplans, wie das Gewicht, die Anzahl der Wiederholungen pro Übung und Anzahl der Sätze. Die ËxerciseËntity enthält hingegen Informationen zu den verfügbaren Übungen im System und ist somit eine umfassende Sammlung von Ressourcen für Trainer und Trainees.

Die Verknüpfung dieser Entities ermöglicht es dem Trainingsplan-Verwaltungssystem, effektiv Trainingspläne für Trainees zu erstellen und zu verwalten, während Trainer Zugang zu Ressourcen und Vorlagen haben, um Trainingspläne schnell und effizient zu erstellen. Die Datenbank spielt somit eine zentrale Rolle bei der Verwaltung und Organisation von Trainingsplänen und ist entscheidend für die Leistung des gesamten Systems (siehe Abbildung 17).

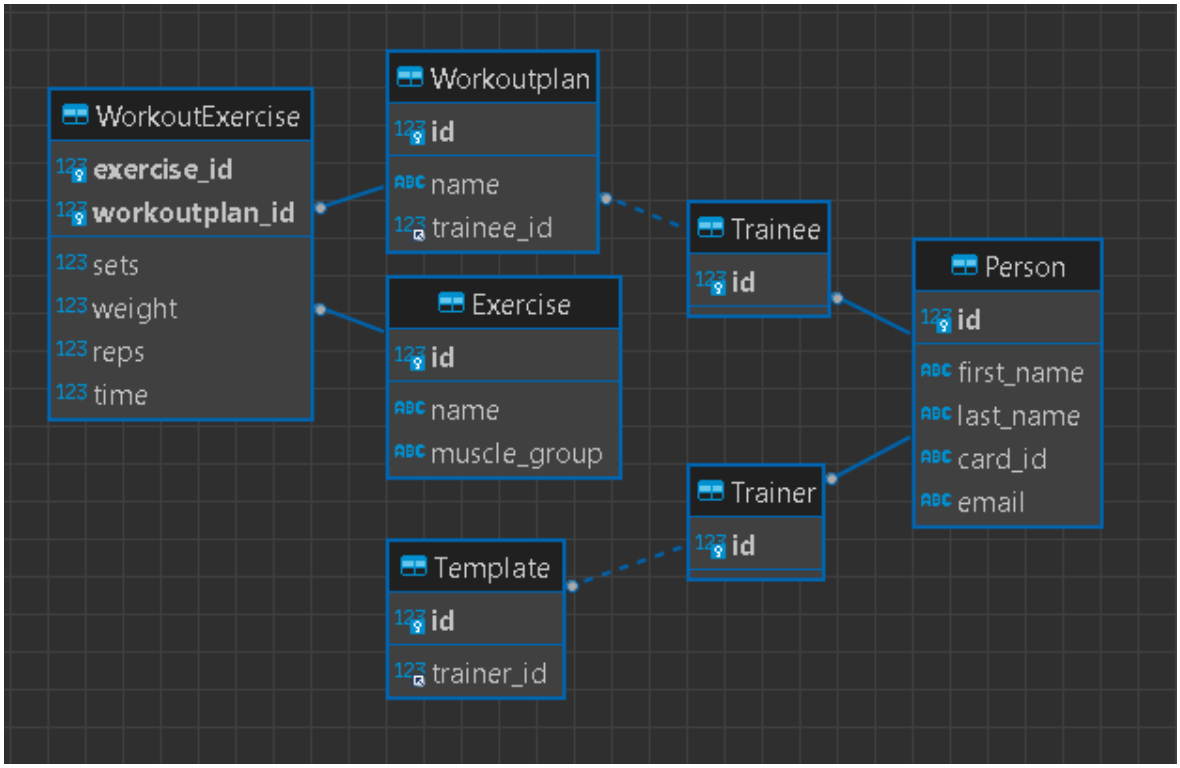

Abbildung 26: ERD der Datenbankstruktur

5.1.6 Datenbankverbindung für IntelliJ Projekt

Listing 53: Datenbank verbindung

```
# Configuration file

quarkus.swagger-ui.path=/swagger
quarkus.swagger-ui.always-include=true

# MS SQL DB connection
quarkus.datasource.db-kind=mysql
quarkus.datasource.username=****
quarkus.datasource.password=***************
quarkus.datasource.jdbc.url=jdbc:mysql://localhost:3306/AberGymDb
#quarkus.hibernate-orm.database.generation=drop-and-create
quarkus.hibernate-orm.database.generation=create
#quarkus.hibernate-orm.database.generation=update
#quarkus.hibernate-orm.sql-load-script=import.sql
```

Dieser Code zeigt die Konfigurationsdatei [11] des Trainingsplan-Verwaltungssystems. Die Konfigurationsdatei legt die Einstellungen und Parameter fest, die das System beeinflussen. Die ersten beiden Einträge definieren den Pfad und die Verfügbarkeit der Swagger-UI, die eine grafische Benutzeroberfläche bietet, um das System zu testen und zu dokumentieren.

Die nächsten Einträge definieren die Verbindung zu einer MySQL-Datenbank. Der Befehl "quarkus.datasource.db-kind"gibt an, welcher Typ von Datenbank verwendet wird, in diesem Fall MySQL. "quarkus.datasource.usernameünd "quarkus.datasource.password"legen die Anmeldeinformationen für den Zugriff auf die Datenbank fest. "quarkus.datasource.jdbc.url"gibt die URL an, unter der die Datenbank erreichbar ist.

Der nächste Eintrag "quarkus.hibernate-orm.database.generation"gibt an, wie die Datenbank erstellt werden soll. In diesem Fall wird die Datenbank jedes Mal, wenn die Anwendung gestartet wird, erstellt, indem der Befehl "create"verwendet wird. Alternativ könnte der Befehl üpdate"verwendet werden, um die Datenbank zu aktualisieren, oder "drop-and-create", um die Datenbank vollständig zu löschen und neu zu erstellen.

Der letzte Eintrag "quarkus.hibernate-orm.sql-load-script"gibt an, welches SQL-Skript verwendet werden soll, um Daten in die Datenbank zu importieren. In diesem Fall wird kein Skript verwendet.

Die Konfigurationsdatei ist ein wichtiger Bestandteil des Trainingsplan-Verwaltungssystems, da sie die Verbindung zur Datenbank und die Erstellung der Datenbank definiert. Durch die Verwendung von Quarkus und Hibernate können Entwickler*innen schnell und einfach Datenbankanwendungen erstellen und konfigurieren.

5.1.7 Demodaten für den Entwicklungsprozess der Mobile Anwendung

Der Befehl ÏNSERT INTOïst ein grundlegender SQL-Befehl [**?**], der verwendet wird, um neue Datensätze in eine Datenbanktabelle einzufügen. Mit diesem Befehl können Daten in eine Tabelle eingefügt werden, indem der Befehl "VALUES"verwendet wird, um die Werte für die einzelnen Spalten anzugeben.

Die Syntax des Befehls ÏNSERT INTO"lautet wie folgt:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Der Befehl beginnt mit dem Namen der Tabelle, in die die Daten eingefügt werden sollen. Anschließend werden in Klammern die Namen der Spalten angegeben, in die die Daten eingefügt werden sollen. Wenn der Befehl ÏNSERT INTO"verwendet wird, um in alle Spalten einer Tabelle einzufügen, kann die Angabe von Spaltennamen entfallen.

Die Werte, die in die Tabelle eingefügt werden sollen, werden nach dem Befehl "VALUESäufgelistet und durch Kommas getrennt. Die Reihenfolge der Werte muss mit der Reihenfolge der Spaltennamen übereinstimmen.

Ein Beispiel für die Verwendung des Befehls ÏNSERT INTO"lautet wie folgt:

```
INSERT INTO customers (name, address, city, state, zip)
VALUES ('Max Mustermann', 'Musterstraße 1', 'Musterstadt', 'Musterland', '123
```

In diesem Beispiel wird ein neuer Datensatz in die Tabelle "customersëingefügt. Der Datensatz enthält die Werte "Max Mustermann"für den Namen, "Musterstraße 1"für die Adresse, "Musterstadt"für die Stadt, "Musterland"für den Staat und "12345"für die Postleitzahl.

Der Befehl ÏNSERT INTOïst ein wichtiger Bestandteil der Datenmanipulation in SQL und wird häufig in Kombination mit anderen Befehlen wie SSELECT", ÜPDATEünd "DELETE"verwendet, um Daten in einer Datenbank zu verwalten.

5.1.8 Import Datei für die Demo Daten

```
-----------------------------------------------------
--  First Customer | Test data
-----------------------------------------------------
Insert into Person (first_name,last_name,email,cardId)
    values ('Antonio','Peric','test@gmail.com','**************');
Insert into Trainee (id) values (1);

-----------------------------------------------------
--  First Workoutplan
-----------------------------------------------------

Insert into Workoutplan (name,trainee_id)
    values ('Ganzkörper Workout', 1);

--  Exercise

Insert into Exercise (name) values ('Liegestütze');
Insert into Exercise (name) values ('Bench Press');
Insert into Exercise (name) values ('Cable Flys');
Insert into Exercise (name) values ('Incline Dumbell press');
Insert into Exercise (name) values ('Biceps Curls');
Insert into Exercise (name) values ('Triceps Extension');
Insert into Exercise (name) values ('Triceps Curls');
Insert into Exercise (name) values ('Skullcrusher');

-- WorkoutExercise

Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (5,0.0,50,1,1);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (3,100.0,6,1,2);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (3,20.0,25,1,3);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
```

```
    values (3,32.5,15,1,4);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (5,12.5,15,1,5);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (5,80.5,20,1,6);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (2,80.0,20,1,7);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (4,30.0,20,1,8);

-----------------------------------------------------
--  Second Workoutplan
-----------------------------------------------------

Insert into Workoutplan (name,trainee_id) values ('Ganzkörper Workout', 1);

-- WorkoutExercise

Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (5,10,50,2,1);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (3,100.0,6,2,2);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (3,20.0,25,2,3);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (3,32.5,15,2,4);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (5,12.5,15,2,5);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (5,80.5,20,2,6);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (2,80.0,20,2,7);
Insert into WorkoutExercise (sets,weight,reps,workoutplan_id, exercise_id)
    values (4,30.0,20,2,8);
```

Der bereitgestellte Code enthält SQL-Anweisungen, die in eine MySQL-Datenbank eingefügt werden sollen. Der erste Teil des Codes bezieht sich auf die Testdaten des ersten Kunden*inn und besteht aus zwei Insert-Anweisungen, die Daten in die Tabellen Person und Trainee einfügen.

Der zweite Teil des Codes bezieht sich auf den ersten Trainingsplan, der erstellt wird. Die Insert-Anweisung fügt einen neuen Eintrag in die Workoutplan-Tabelle ein und weist ihn dem Trainee mit der ID 1 zu.

Der dritte Teil des Codes bezieht sich auf die einzelnen Übungen, die Teil des Trainingsplans sind. Die Insert-Anweisungen fügen neue Einträge in die Exercise-Tabelle ein.

Der letzte Teil des Codes fügt die einzelnen Übungen mit den entsprechenden Sets, Gewichten und Wiederholungen in die WorkoutExercise-Tabelle ein und weist sie dem ersten Trainingsplan zu. Es werden auch Insert-Anweisungen für den zweiten Trainingsplan erstellt, die die gleichen Übungen wie der erste Trainingsplan enthalten, jedoch unterschiedliche Set-, Gewichts- und Wiederholungswerte aufweisen.

5.2 Integration des echten Backends

5.2.1 LeoCloud

LeoCloud [12] ist ein Kubernetes Cluster [13], der von der HTBLA Leonding betrieben wird und Schülern zur Verfügung steht. Kubernetes ist eine Container-Orchestrierungsplattform, die es Entwicklern ermöglicht, Containeranwendungen effizient zu erstellen, zu verwalten und zu skalieren. LeoCloud stellt Schülern eine Umgebung zur Verfügung, um praktische Erfahrungen mit der Container-Technologie und der Entwicklung von Cloud-Anwendungen zu sammeln.

Das Backend des Projekts AberGym-Syp wird auf LeoCloud gehostet, was den Entwicklern eine zuverlässige und skalierbare Plattform bietet, um die Anwendung bereitzustellen. Die Nutzung von Kubernetes ermöglicht eine automatisierte Skalierung von Ressourcen, die die Anforderungen der Anwendung erfüllen. So kann die Leistungsfähigkeit des Backends bei Bedarf erhöht werden, ohne manuelle Eingriffe von Seiten des Entwicklers.

Die Integration von Rest-Schnittstellen ermöglicht es, auf die Datenbank des Projekts AberGym-Syp zuzugreifen. Die MySql-Datenbank enthält alle relevanten Informationen, die von der Anwendung benötigt werden, um die Trainingspläne zu erstellen und zu verwalten. Das Einrichten von Rest-Schnittstellen erleichtert den Datenaustausch zwischen verschiedenen Anwendungen und ermöglicht eine einfache Integration in andere Systeme.

Die Nutzung von LeoCloud und Kubernetes ist ein Beispiel für den Einsatz moderner Cloud-Technologien in der Bildung. Es bietet Schülern eine Plattform, um praktische Erfahrungen mit den neuesten Technologien zu sammeln und ihre Fähigkeiten in der Entwicklung von Cloud-Anwendungen zu verbessern.

Literaturverzeichnis

[1] Amazon Web Services, „What is Flutter?" https://aws.amazon.com/de/what-is/flutter/#:~:text=Flutter%20ist%20ein%20Open%2DSource,auf%20Basis%20eines%20einzigen%20Codes, 2023, Online verfügbar:.

[2] IntelliJ, „IntelliJ," https://www.jetbrains.com/dede/idea/features/, 2023.

[3] Red Hat, „Quarkus Code Generator," https://code.quarkus.io/, 2023.

[4] Oracle Corporation, „What is Java?" https://www.java.com/en/download/help/whatis_java.html, 2023.

[5] Eclipse Foundation, „Vert.x MySQL Client Documentation," https://vertx.io/docs/vertx-mysql-client/java/, 2023.

[6] SmallRye, „SmallRye Documentation," https://smallrye.io/docs/index/index/index.html, 2023.

[7] Red Hat, „Quarkus MySQL Datasource Guide," https://quarkus.io/guides/datasource#mysql, 2023.

[8] ——, „Quarkus Hibernate ORM with Panache Guide," https://quarkus.io/guides/hibernate-orm-panache, 2023.

[9] Kinsta, „Was ist MySQL?" https://kinsta.com/de/wissensdatenbank/was-ist-mysql/, 2022.

[10] Docker, „MySQL Docker Image Documentation," https://hub.docker.com/_/mysql, 2022.

[11] Red Hat, „Quarkus Hibernate ORM Guide," https://quarkus.io/guides/hibernate-orm, 2022.

[12] Christian Aberger, „Canonical Application," https://aberger.at/blog/design/2021/11/06/canonical-application.html, November 2021.

[13] VMware, „Kubernetes Cluster," https://www.vmware.com/topics/glossary/content/kubernetes-cluster.html, 2022.

Abbildungsverzeichnis

1 Use-Case Diagramm 9
2 ERD Diagramm in PlantUml 11
3 Mockup 1 | Log-in-Bereich der mobilen Anwendung 16
4 Mockup 2 | Log-in-Bereich der mobilen Anwendung 17
5 Mockup 3 | Hauptbereich der mobilen Anwendung 18
6 Mockup 4 | Hauptbereich der mobilen Anwendung 18
7 Mockup 1 | To-Do-Bereich der mobilen Anwendung 19
8 Mockup 2 | To-Do-Bereich der mobilen Anwendung 20
9 Mockup 3 | To-Do-Bereich der mobilen Anwendung 20
10 Mockup 4 | To-Do-Bereich der mobilen Anwendung 21
11 Mockup 1 | Zähl-Bereich der mobilen Anwendung 22
12 Mockup 3 | Zähl-Bereich der mobilen Anwendung 23
13 Mockup 4 | Zähl-Bereich der mobilen Anwendung 23
14 Mockup 3 | Hauptbereich nach der Durcharbeitung des Trainingplans . 24
15 Mockup 4 | Hauptbereich nach der Durcharbeitung des Trainingplans . 25
16 Docker Architektur 34
17 Android Studio Emulator 36
18 Start-Screen 42
19 QR-CodeTutorial 47
20 QR-Code auf Kamera 47
21 NFC 54
22 Hauptmenü 55
23 To-Do-Liste 59
24 Satzzähler 62
25 Trainingplan bearbeiten 66
26 ERD der Datenbankstruktur 81

Tabellenverzeichnis

1 *Tabelle 1.1: Informationen über den Bertreuer und Partner.* 1
2 *Tabelle 1.2: Informationen über das Projekt und das Team.* 1

Quellcodeverzeichnis

1 Start-Screen Packages . . . 42
2 main-Funktion . . . 43
3 Login-Seite . . . 43
4 Layout und Hintergrundfarbe . . . 43
5 Logo und Willkommensnachricht . . . 44
6 QR-Code Schaltfläche . . . 45
7 NFC Schaltfläche . . . 46
8 QR-Code Seite erstellen . . . 48
9 QR-Code Seitenstruktur . . . 49
10 QR-Code Scanner erstellen . . . 50
11 QR-Code-Scanner Controller . . . 50
12 Datenbankabfrage QR-Code . . . 50
13 NFC Package und State . . . 52
14 NFC Status überprüfen . . . 52
15 NFC Tag scannen . . . 53
16 NFC Widget erstellen . . . 53
17 Layout Packages . . . 56
18 LayoutState . . . 56
19 Layout Scaffold-Widget . . . 56
20 Navbar . . . 56
21 Trainingsplan Logik . . . 57
22 Trainingsplan starten Knopf . . . 58
23 HomePage Navigation . . . 58
24 Second Page . . . 59
25 To-Do-Liste Trainingsplan Variablen . . . 60
26 To-Do-Liste get Methode . . . 60
27 To-Do-Liste Item Methode . . . 61
28 SetCounter . . . 62
29 SetCounterState . . . 63
30 Increase Counter Funktion . . . 63
31 initState . . . 63
32 GestureDetector . . . 64
33 GestureDetector zum Schluss . . . 64
34 Bearbeiten Index . . . 66
35 Bearbeiten UpdateExcersiceState . . . 66
36 Bearbeiten initState getData . . . 67
37 Bearbeiten Build . . . 67
38 Bearbeiten saveInput . . . 70
39 Bearbeiten saveInputInCache . . . 70
40 Entity | Person . . . 72
41 Entity | Trainee . . . 72
42 Entity | Trainer . . . 72
43 Entity | Template . . . 72

44 Entity | Workoutplan 73
45 Entity | WorkoutExersice 73
46 Entity | Exersice 73
47 Dependency | reactive-mysql-client 74
48 Dependency | smallrye-openapi 74
49 Dependency | jdbc-mysql 75
50 Dependency | hibernate-orm-panache 76
51 Dependency | lombok 77
52 Erstellung der Datenbank mittels Docker Befehl 79
53 Datenbank verbindung 82