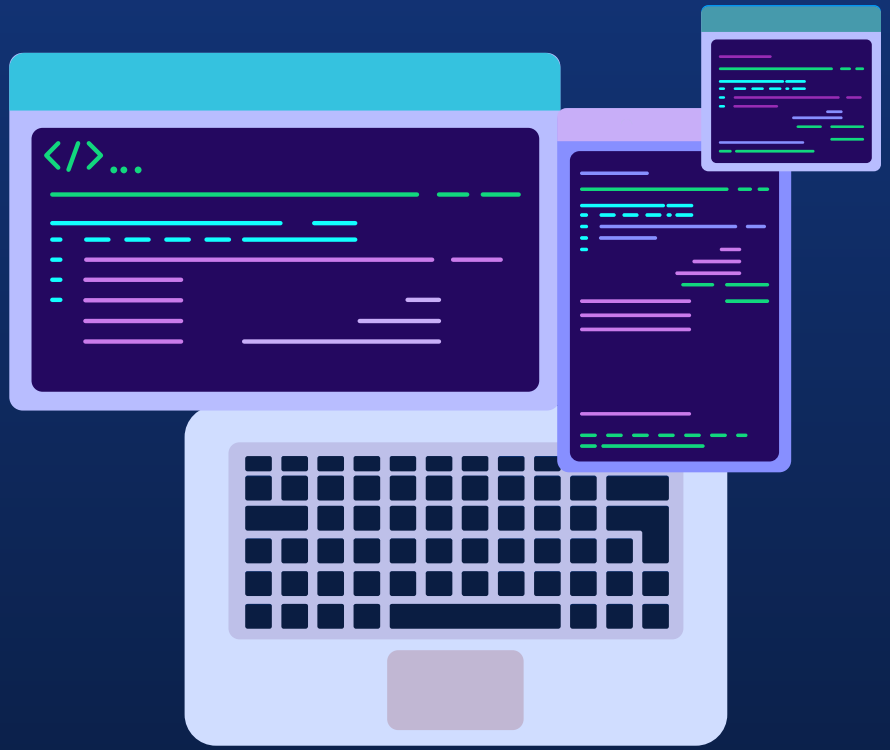
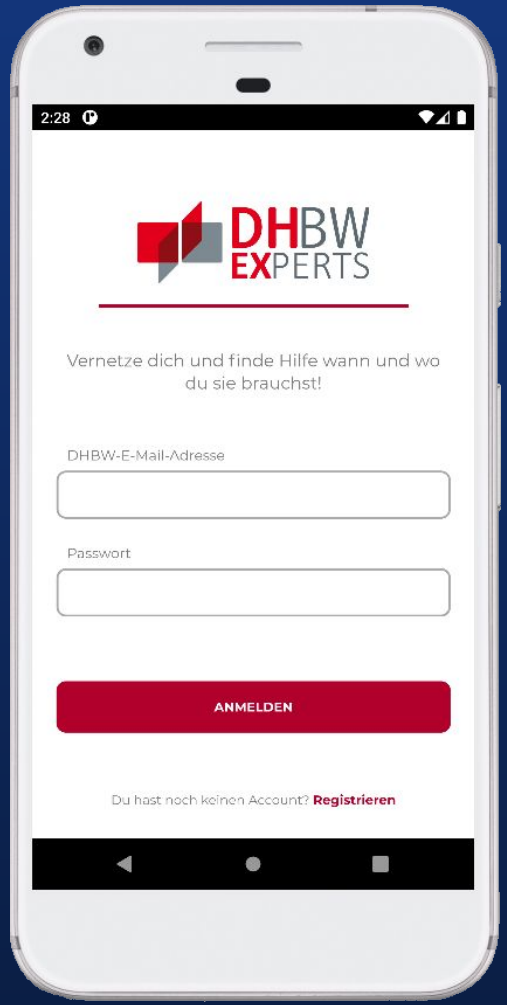
DHBW Experts

-All you need are the right connections-



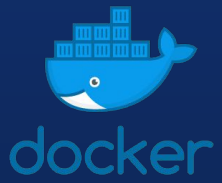
Vision

- Experten finden
- DHBW Ausweis als digitale Visitenkarte
- Kontakte pflegen
- Neue Leute finden

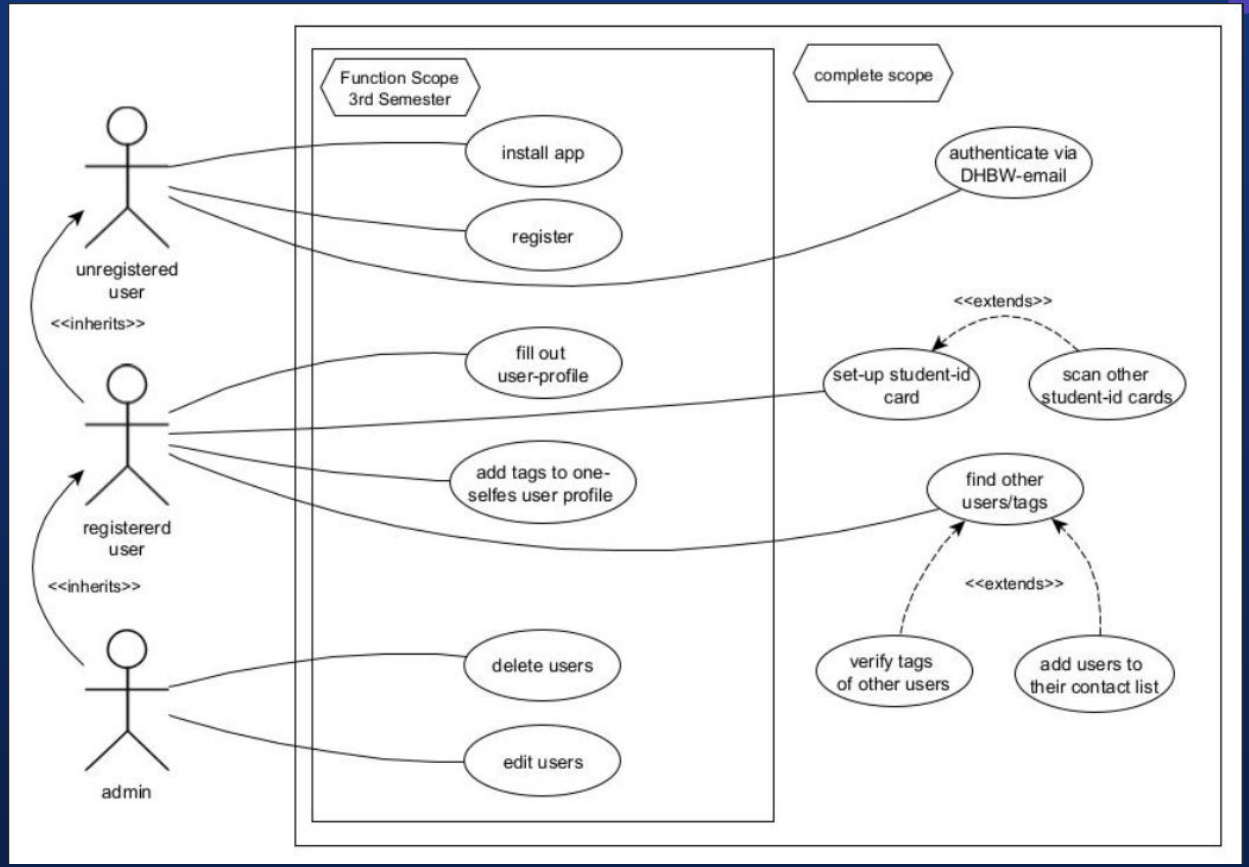


Tools

- Visual Studio Code
- Docker
- GitHub Actions
- Azure
- Ionic Jasmine



Use Case Diagram



Rollenverteilung im Team

Ralph

Integrator
Implementer

Noah

Software Architect
Code Reviewer

Tim

Systems Analyst
Project Manager

Lukas

Configuration Manager
Graphic Artist
Deployment Manager

Project Management

Scrumming, RUP & mehr

Tools

YouTrack

Unser genutztes Kanban Board ist von YouTrack

Discord

Für eine bessere Kommunikation

GitHub

Mehrere Projekt-Repositories

GitHub - Blog

Ein simpler Weg, um Blogs zu veröffentlichen

Scrumming

Kanban Boards

Für eine gute Übersicht während dem Arbeiten

Iterativer Prozess

Sprints (2-wöchig) und Meetings (2x/Woche) finden regelmäßig statt

Pair Programming

Gemeinsames Erarbeiten einer Lösung für mehr Durchsicht

Retrospektive

Hinterfragen nicht fachlicher Themen, die sich auf das Projektmanagement beziehen

Sprints und Burndown Charts

Idealer Burndown

Verbleibender Aufwand

Zeitaufwand

Beispiel Burndown Chart vom Sprint KW46-47

RUP-Phasen

Inception

Festlegen auf Projekt, Techstack, Umfang abklären

Elaboration

Projekt aufsetzen, erste Mock-Ups erstellen

Construction

Implementierung, Arbeit an Front- und Backend

Transition

Noch nicht erreicht; Abnahme des Produkts

RUP Workflows

Project Management

Alles nicht fachliche, was sich um das Projekt dreht

Implementation

Die Implementierung der App

Requirements

Festlegen von Anforderungen an unsere Anwendung

Analysis & Design

Designstrukturen der App

Deployment

Deployen des Projekts und API-Anbindungen

Environment

Die Umgebung, auf der das Projekt läuft

Tech Stack

Azure SQL Cloud Database

C# API

Mail Server

Backend

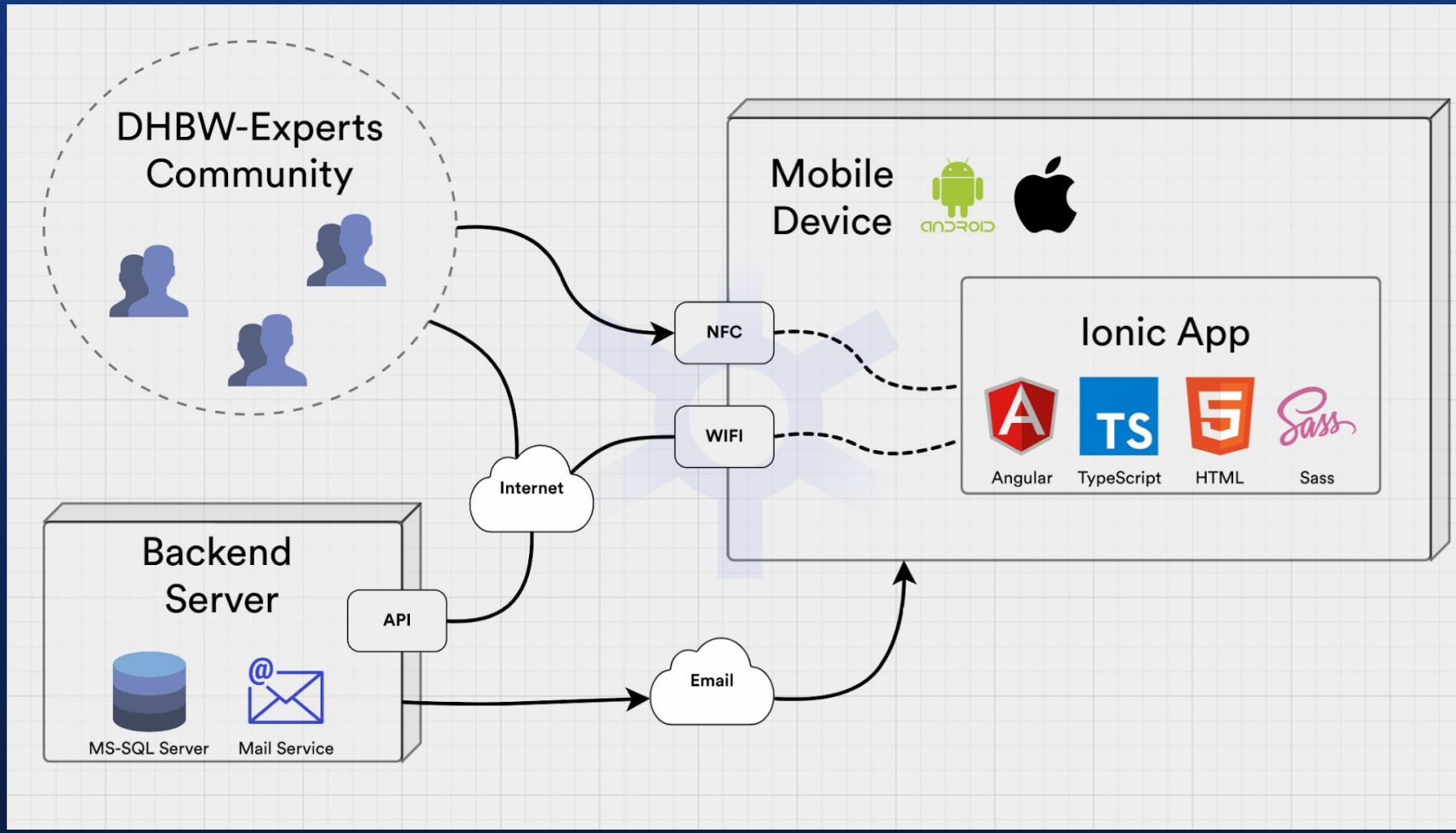
TypeScript

Ionic + Angular

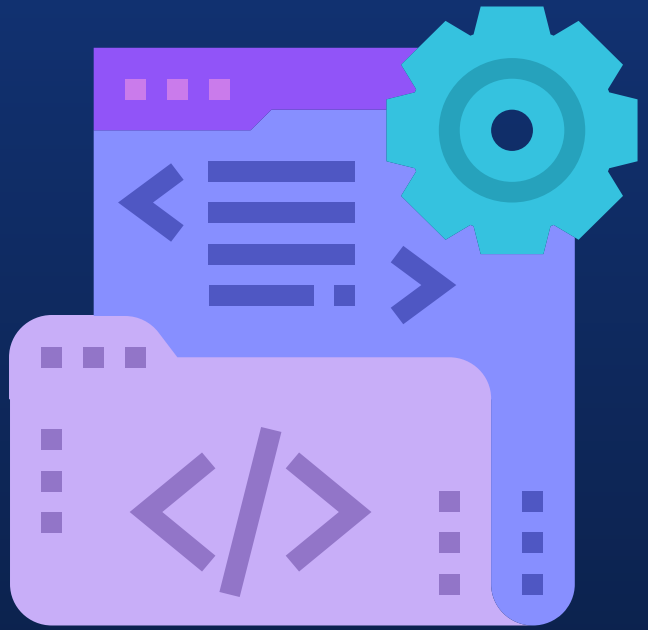
HTML

Frontend

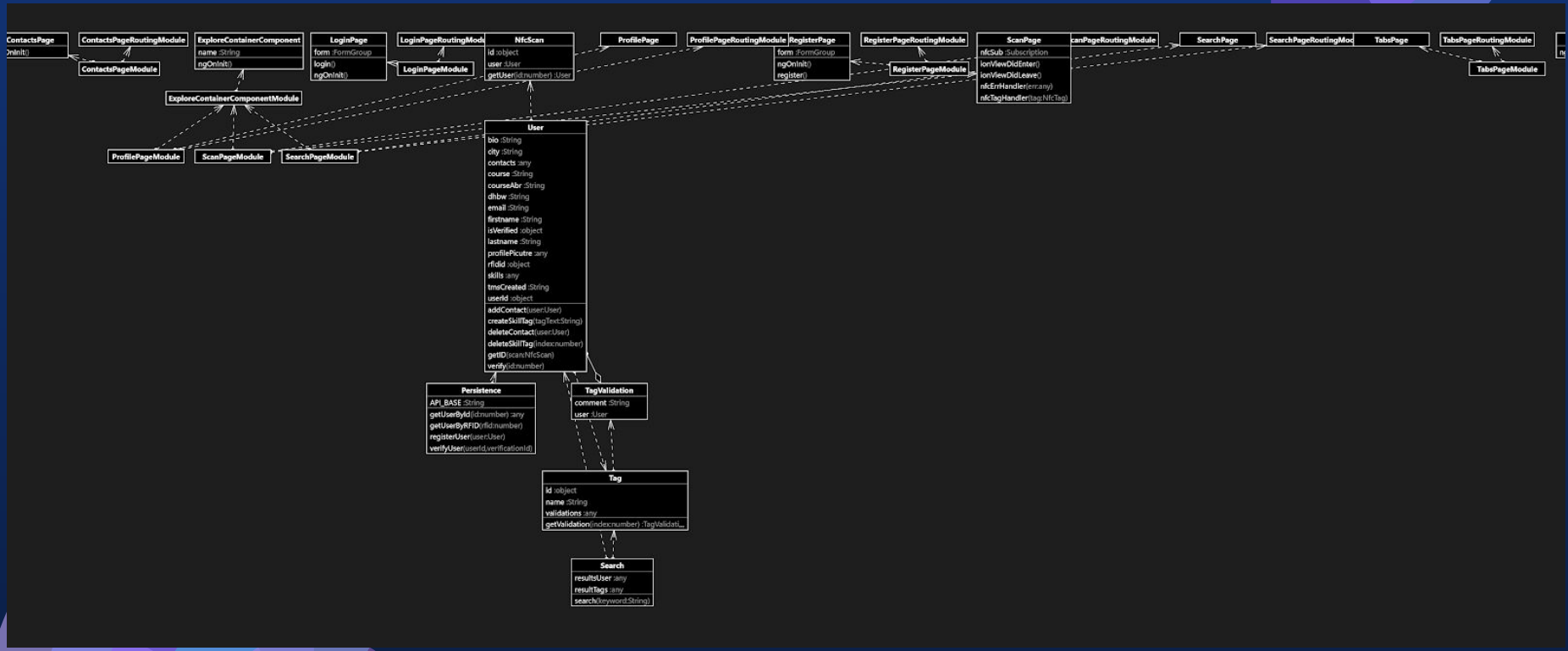
Sass


DHBW-Experts
Community
Mobile
Device
android
NFC
WIFI
Internet
Ionic App
Angular
TypeScript
HTML
Sass
Backend
Server
API
Email
MS-SQL Server
Mail Service

Architecture



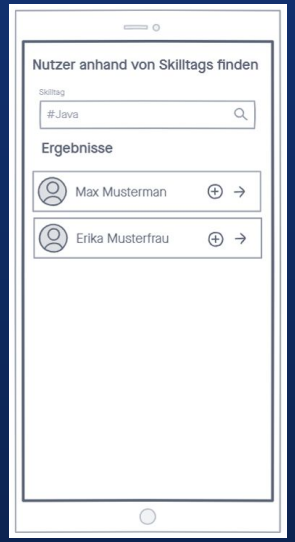
Class Diagram

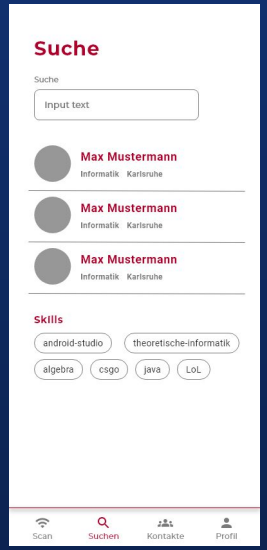


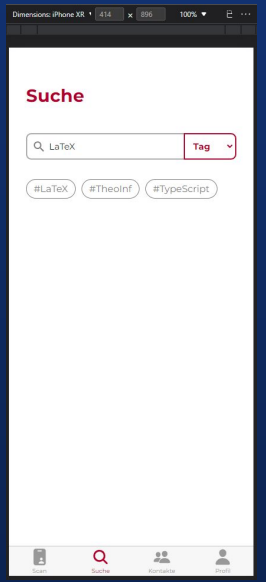
APP

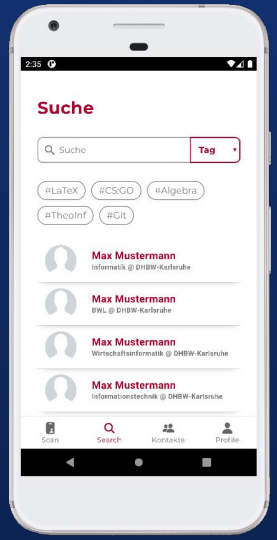
QUALITY

APP-UI-DESIGN-PROZESS









WIREFRAMES

IONIC

Testing - Feature Files

Es existieren Feature Files zu 2 Use Cases, die jedoch (noch) nicht zum Laufen gebracht wurden

API

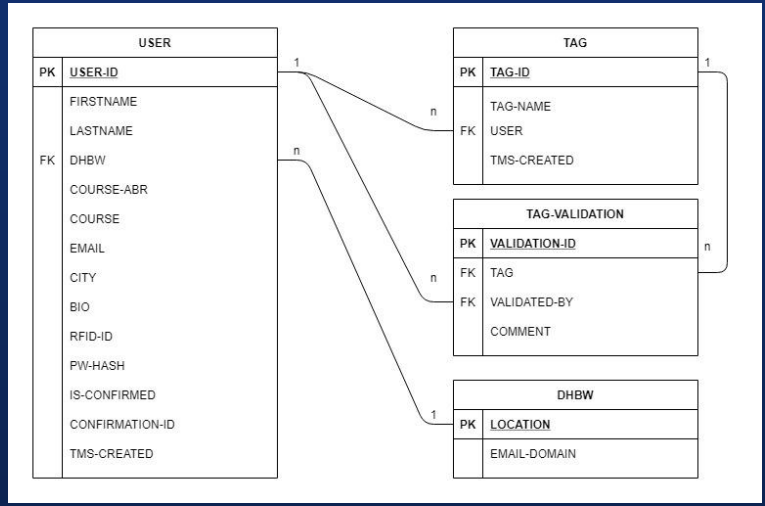
Wo alle Daten
zusammenkommen

Gedankenprozess

Schritt 1	Welche Daten fallen an?
Schritt 2	Wie setzen wir unsere Anforderungen in ein Datenbankmodell um?
Schritt 3	Welche Datenbank verwenden wir?
Schritt 4	Wie kommt unsere App sicher an diese Daten?

Welche Daten fallen an? Wie speichern wir sie?

- Orientierung an Use-Cases
- Relationales Datenbankmodell
- Leicht umzusetzen



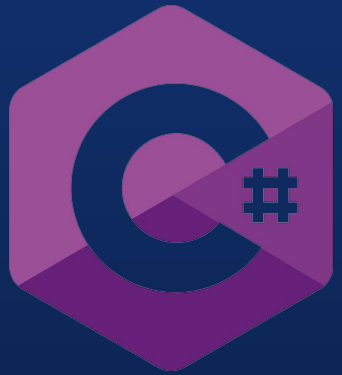
Microsoft Azure

Welche Datenbank verwenden wir?

- Kostenlos für Studenten
- Erfahrung aus dem letzten Projekt
- 24/7 verfügbar
- Alle arbeiten auf den gleichen Datensätzen
- Spiegelt realistisches Szenario wider

Wie machen wir unsere Daten verfügbar?

- C# (C Sharp) Backend
- Kontrollierter Zugriff auf Daten
- Code/Database First Coding möglich
- Authentifiziert Benutzer
- Verifiziert Nutzer
- (E-Mail Verifizierung vorgeplant)



DotNet C#

21 API Schnittstellen

Davon 19 bereits im ersten Semester implementiert

API-Endpunkte

Login, Register, Verify

GET Login User

POST Register User

PUT Verify User

API-Endpunkte

Users

- **GET** Get User by User-ID
- **GET** Get User by RFID-ID
- **POST** Edit User
- **DELETE** Delete User by User-ID

API-Endpunkte

Tags

- **GET** Get Tag by Tag-ID
- **GET** Get Tags by User-ID
- **GET** Get Tag-Validation by Val-ID
- **GET** Get Tag-Validations by Tag-ID
- **POST** Add Tag to User
- **DELETE** Delete Tag by Tag-ID
- **DELETE** Delete Tag-Validation by Val-ID

API-Endpunkte

Contacts

GET	Get Contacts of User by User-ID
POST	Add User to Contact List
DELETE	Delete Contact of User by User-ID

API-Endpunkte

Search

GET GET User by User-ID

GET GET Distinct Tags by Text

API

Qualitätssicherung

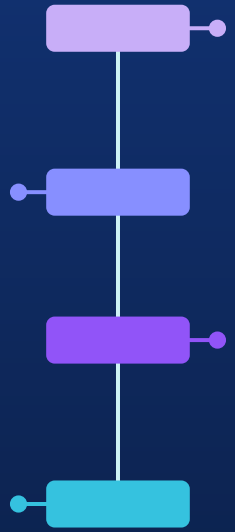
Datenbank Constraints

Nur DHBW-E-Mail-Adressen

Anders ist eine Registrierung nicht möglich

Keine eigenen Tags validieren

Reputation muss sich erarbeitet werden



Nur verifizierte User können teilnehmen

Andere blendet die Such-API aus

Sämtliche unlogischen Aktionen wurden eingeschränkt

z.B. sich selbst als Kontakt hinzufügen

CODE FIRST

Jede Änderung an der Datenbank sorgt für Bugs in der API

Mit C# und Code First kann anhand der Datenbank aber in Sekunden ein neues funktionales Datenbankmodell generiert werden

CI-CD

Push auf den
main-Branch

Compile & Build durch
Github Actions

Deploy auf einen
Azure Dev-Server

DANKE

Fragen? Immer her damit!

DHBW-Experts
Ralph, Tim, Lukas, Noah