# Architektur

## Einleitung / Mission Statement nach Abstract

Aus Fachkonzept klauen und umschreiben

## Problemstellung

Fachkonzept zusammenfassen und Problemstellung herausarbeiten

## Lösungsstrategie

Architekturbeschreibung (nächste Woche)

## Architekturentscheidungen

* Frontend:
  + React mit TypeScript und Vite Server für Hosting
  + MUI Theme wenn möglich benutzen, wenn nicht dann SCSS
* Backend:
  + TypeScript mit Express-Server und Sequelize in einer NodeJS-Umgebung mit Anbindung an den Docker-Container mit der Datenbank
* Datenbank:
  + ER-Modell einfügen und beschreiben
  + MySQL-Datenbank in einem Docker-Container
  + Tabellen werden automatisch bei Starten des Containers angelegt und mit Testdaten befüllt

## Randbedingungen

### Technische Randbedingungen

* VS Code als Entwicklungs-Umgebung
* Testing:
  + Unit-Tests für jede Frontend-Komponente mit Vitest und dem React-testing-Framework
  + Unit-Tests für Backend-Endpunkte mit Vitest und möglicherweise mit der Postman-Anwendung
  + Datenbank wird mit Testdaten befüllt zum Testen
  + Gegenseitiges Testen des Codes der Anderen, damit kein Autor seinen eigenen Code testet
* Material-UI als Komponenten-Bibliothek
* Material Icons benutzen[: https://fonts.google.com/icons?selected=Material+Icons](https://fonts.google.com/icons?selected=Material+Icons)

### Organisatorische Randbedingungen

* Besprechung im Plenum jede Woche Montag
* Bilden von Mentor und Mentee Beziehungen, um die weniger erfahrenen Mitglieder in React zu schulen
* Peer Programming, wenn möglich

### Codier-Richtlinien

* Format zur Benennung der Branches: <Issue\_Nummer>-<Name\_des\_Issues>
* Format-Einstellungen für den Programmcode sind in der Datei „.prettierrc.json“ festgehalten und werden automatisch von der VS Code Erweiterung „Prettier“ genutzt
* Programmcode wird in Englisch geschrieben
* Text in der Benutzeroberfläche auf Deutsch
* Möglichst spezifische Typen (kein any!)
* Funktionen/Variablen sollten mit ihren Namen selbstverständlich und nachvollziehbar sein
* Wenn Namen der Funktionen/Variablen nicht sich selbst gut genug dokumentieren, dann Kommentare ergänzen
* Umfang einer Methode auf ein Minimum beschränken
* Methoden sollten nicht mehr machen als ihr Name vermuten lässt
* camelCase für Methoden, Variablen und Ordnernamen
* PascalCase für Typen, Klassen und React-Komponenten
* Kein var in TypeScript (sondern let oder const)

## Fazit und Ausblick auf Verbesserungen (nächste Schritte)

Kommt noch