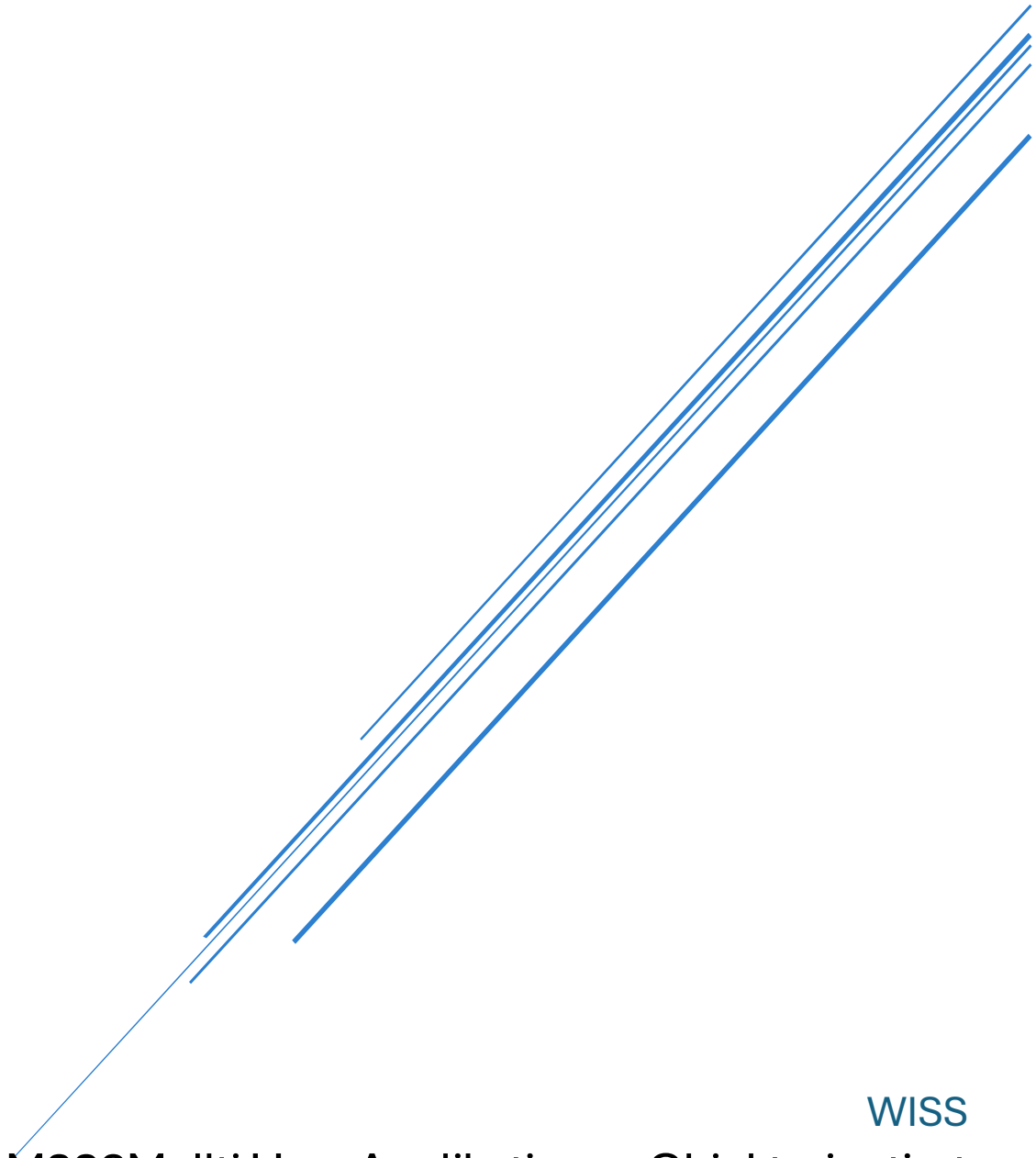
LB Dokumentation

Sasha Fellmann



WISS

M223Mullti User Applikationen Objektorientiert Realisieren

Inhaltsverzeichnis

Einleitung

Diese Dokumentation beinhaltet eine umfassende Übersicht zum Inhalt und Verlauf des Abschlussprojektes vom Modul 223. Im Projekt geht es um die Login/Registrierungs-funktion und die Gewährleistung von Sicherheit mithilfe von JWT Tokens und Route Protection.

Anforderungsanalyse und User Stories

User Stories

Persona Nr. 1	Paul
Bedürfnisse/Erwartungen	Paul möchte dass seine Daten sicher sind. Er will, dass nur er auf seinen Account zugreifen kann.
Pain Points/Befürchtungen	Er befürchtet, dass mehrere Accounts den selben Namen haben können und sich somit jemand in seinen Account einloggen kann.

Persona Nr. 2	John
Bedürfnisse/Erwartungen	John möchte zugriff auf die Editors Lounge, ohne die Standard User.
Pain Points/Befürchtungen	Er befürchtet, dass die Routes dorthin nicht richtig geschützt sind, und normal Nutzer auch darauf zugreifen können.

Persona Nr. 3	Ringo
Bedürfnisse/Erwartungen	Ringo möchte sich ausloggen können, da er seinen Computer gerne anlässt und nicht möchte, dass fremde in seinen Account können.
Pain Points/Befürchtungen	Er befürchtet, dass die Accountsicherheit nicht gegeben ist.

Akzeptanzkriterien

1. **Registrierung:** Nutzer sollen neue Accounts erstellen können. Es soll klar sein, was von den Usern erwartet wird, was Nutzernamen und Passwörter angeht.
2. **Accounts:** Nutzer sollten nur Individuelle Accounts erstellen können. Es sollten keine 2 Accounts mit dem gleichen Namen möglich sein.
3. **Login:** Nutzer sollen sich mit Ihren bestehenden Accounts wieder anmelden können.
4. **Rollen:** Nutzer sollen Ihre Rollen (User, Admin, Editor) auswählen können.
5. **Zugriff:** Nutzer sollen je nach ausgewählter Rolle beschränkt Zugriff auf Daten bzw. Seiten der Webapp haben.
6. **Abmelden:** Nutzer sollen sich wieder abmelden können, um Ihre Daten zu schützen.

Sicherheitskonzept

Sicherheitsmassnahme	Beschreibung
Formularvalidierung	Benutzernamen und Passwörter werden vor dem Absenden validiert, um sicherzustellen, dass sie den festgelegten Regeln entsprechen.
Eingabevalidierung	Validierung aller Eingaben, um sicherzustellen, dass sie den erwarteten Formaten entsprechen und keine schädlichen Daten enthalten.
Passwort-Hashing	Passwörter werden vor der Speicherung in der Datenbank gehasht (z.B. mit bcrypt), um sicherzustellen, dass sie im Falle eines Datenlecks nicht im Klartext verfügbar sind.
Rollenbasierte Zugriffskontrolle (RBAC)	Benutzerrollen wie "User", "Admin" und "Editor" werden implementiert, um sicherzustellen, dass Benutzer nur auf die für sie vorgesehenen Ressourcen und Aktionen zugreifen können.
Token-basierte Authentifizierung	Verwendung von JSON Web Tokens zur Authentifizierung, um sicherzustellen, dass nur autorisierte Benutzer Zugriff auf geschützte Ressourcen haben.
Umgang mit Fehlern	Sorgfältige Fehlerbehandlung und -protokollierung, um sicherzustellen, dass sensible Informationen nicht in Fehlermeldungen preisgegeben werden. (Nicht wirklich implementiert, da relevante Informationen in der Konsole angezeigt werden.)
CORS	Konfigurieren von Cross-Origin Resource Sharing (CORS), um sicherzustellen, dass nur autorisierte Domains Zugriff auf die API haben.
Rate Limiting	Implementierung von Rate Limiting, um die Anzahl der Anfragen pro Benutzer/IP-Adresse zu begrenzen und DDoS-Angriffe zu verhindern.
Sichere Konfigurationen	Verwendung von sicheren Standardeinstellungen und Umgebungsvariablen zur Verwaltung sensibler Informationen wie Datenbankverbindungsdetails und API-Schlüssel.
SQL-Injection-Schutz	Verwendung von vorbereiteten Statements (Prepared Statements) und ORM (Sequelize), um SQL-Injection-Angriffe zu verhindern.

Zugriffskontrolle	Begrenzung der Datenbankzugriffe auf nur die notwendigen Berechtigungen für den Anwendungsbenutzer.

Arbeitsplanung

Phase 1: Planung und Anforderungsanalyse (1/2 Woche)

Ziele und Anforderungen festlegen

- Hauptziele der Anwendung identifizieren/bestimmen.
- Liste der funktionalen und nicht-funktionalen Anforderungen erstellen.

Technologie-Stack und Architektur definieren

Projektplan und Zeitrahmen erstellen

- Projektplan erstellen.

Phase 2: Entwurf (1/2 Woche)

Datenbankentwurf

- Datenbankschema für die MySQL-Datenbank entwerfen(Tabellen, Beziehungen, etc.).

API-Entwurf

- Endpunkte für die RESTful API definieren(z.B. /register, /login, /employees).

Frontend-Entwurf

- Struktur und Navigation der React-Komponenten definieren.

Phase 3: Implementierung (2 Wochen)

Woche 1: Backend-Implementierung

Projektsetup

- Node.js-Projekt initialisieren und die erforderlichen Abhängigkeiten instalieren (Express, Sequelize, bcrypt, etc.).

Datenbankverbindung

- Verbindung zur MySQL-Datenbank einrichten.

Modelle

- Sequelize-Modelle für User und Employee Definieren.

API-Endpunkte

- Authentifizierungs- und Autorisierungslogik implementieren.
- CRUD-Endpunkte für die Employee-Ressource erstellen.
- Registrierungs- und Login-Endpunkte implementieren.

Middleware und Sicherheit

- Middleware für die Authentifizierung und Fehlerbehandlung implementieren.
- Sicherstellen, dass sensible Daten (Passwörter) sicher gespeichert werden.

Woche 2: Frontend-Implementierung

Projektsetup

- React-Projekt Initialisieren und Abhängigkeiten installieren Axios, React Router, etc.).

Komponentenentwicklung

- Hauptkomponenten erstellen(Register, Login, Employee Management).
- Formulare mit Validierung Implementieren(Username, Passwort, etc.).

API-Integration

- Kommunikation mit dem Backend implementieren (Axios).
- Benutzer authentifizierung und autorisierung sicherstellen.

Styling und Benutzererfahrung

- CSS-Styling anpassen.

Phase 4: Testen (1-2 Wochen)

Automatisierte Tests

- Automatisierte Tests schreiben und durchführen.

Manuelles Testen

- Manuele Tets durchführen.

Fehlerbehebung und Optimierung

- Gefundene Fehler beheben.

Phase 5: Dokumentation (1 Woche)

Bereitstellungsvorbereitung

- Dokumentation fertigstellen.

Der Tatsächliche verlauf des Projekts ist leider nicht dieser Arbeitsplanung gefolgt.

Testkonzept

Testziele

- Sicherstellen, dass Benutzer sich erfolgreich registrieren können.
- Sicherstellen, dass Benutzer mit bereits existierenden Benutzernamen nicht erneut registriert werden können.
- Sicherstellen, dass Benutzer die richtige Rolle zugewiesen bekommen.
- Sicherstellen, dass alle Validierungsregeln für Benutzernamen und Passwörter korrekt umgesetzt sind.

Frontend-Tests

Registrierungsformular Validierung

- Sicherstellen, dass Benutzernamen und Passwörter den festgelegten Regeln entsprechen.
- Sicherstellen, dass das Passwort und das bestätigte Passwort übereinstimmen.

Behandlung von doppelten Benutzernamen

- Sicherstellen, dass bei einem Versuch, einen bereits existierenden Benutzernamen zu registrieren, eine entsprechende Fehlermeldung zurückgegeben wird.

Backend-Tests

Erstellen eines neuen Benutzers

- Sicherstellen, dass ein neuer Benutzer erfolgreich in der Datenbank erstellt wird.
- Sicherstellen, dass die Rolle des Benutzers korrekt in der Datenbank gespeichert wird.

Behandlung von doppelten Benutzernamen

- Sicherstellen, dass bei einem Versuch, einen bereits existierenden Benutzernamen zu registrieren, eine entsprechende Fehlermeldung zurückgegeben wird.

Teststrategie

Testmethoden

- **Automatisierte Tests**: Verwendung von Jest für Frontend-Tests und Supertest für Backend-Tests.
- **Manuelle Tests**: Ergänzende Tests durch manuelle Überprüfung der Benutzeroberfläche und der Backend-APIs.

Testwerkzeuge

- **Frontend-Tests**: Jest, Testing Library (React)
- **Backend-Tests**: Jest, Supertest

Testfälle

Frontend-Testfälle

Erstellen eines neuen Benutzers

- **Beschreibung**: Überprüft, ob ein neuer Benutzer erfolgreich in der Datenbank erstellt wird.
- **Erwartetes Ergebnis**: Benutzer wird erfolgreich erstellt, und eine Erfolgsmeldung wird zurückgegeben.
- **Tatsächliches Ergebnis**: Benutzer wird erfolgreich erstellt, und eine Erfolgsmeldung wird zurückgegeben.

Behandlung von doppelten Benutzernamen

- **Beschreibung**: Überprüft, ob bei einem Versuch, einen bereits existierenden Benutzernamen zu registrieren, eine Fehlermeldung zurückgegeben wird.
- **Erwartetes Ergebnis**: Fehlercode 409 und Fehlermeldung werden zurückgegeben.
- **Tatsächliches Ergebnis**: Fehlercode 409 und Fehlermeldung werden zurückgegeben.

Backend-Testfälle

Erstellen eines neuen Benutzers

- **Beschreibung**: Überprüft, ob ein neuer Benutzer erfolgreich in der Datenbank erstellt wird.
- **Erwartetes Ergebnis**: Benutzer wird erfolgreich erstellt, und seine Rolle wird korrekt in die Datenbank übernommen.
- **Tatsächliches Ergebnis**: Benutzer wird erfolgreich erstellt, die Rolle wird jedoch nicht übernommen.

Behandlung von doppelten Benutzernamen

- **Beschreibung**: Überprüft, ob bei einem Versuch, einen bereits existierenden Benutzernamen zu registrieren, eine Fehlermeldung zurückgegeben wird.
- **Erwartetes Ergebnis**: Fehlercode 409 und Fehlermeldung werden zurückgegeben.
- **Tatsächliches Ergebnis**: Fehlercode 409 und Fehlermeldung werden zurückgegeben.

Testausführung

Frontend-Tests:

- Führe die Tests aus mit: npm test

Backend-Tests:

- Führe die Tests aus mit: jest path/to/testfile.test.js

Technologie und Architektur

Verwendete Technologien

Frontend

React

- Ein JavaScript-Framework für den Aufbau von Benutzeroberflächen. Es ermöglicht die Erstellung wiederverwendbarer UI-Komponenten.

Axios

- Eine Bibliothek zum Durchführen von HTTP-Anfragen. Es wird verwendet, um Anfragen an das Backend-API zu senden.

Backend

Node.js

- Eine serverseitige JavaScript-Laufzeitumgebung, die auf Chrome's V8 JavaScript-Engine basiert.

Express.js

- Ein Web-Framework für Node.js, das eine schlanke und flexible Lösung für die Erstellung von Web- und API-Anwendungen bietet.

Sequelize

- Ein ORM (Object-Relational Mapping) für Node.js, das mit verschiedenen Datenbanken, einschließlich MySQL, arbeitet. Es bietet eine einfache Möglichkeit, mit der Datenbank zu interagieren und Modelle zu definieren.

Datenbank

MySQL

- Ein relationales Datenbankmanagementsystem. Es wird verwendet, um die Anwendungsdaten sicher und strukturiert zu speichern.

Sicherheit und Authentifizierung

bcrypt

- Eine Bibliothek zum Hashen von Passwörtern. Es wird verwendet, um Passwörter vor der Speicherung sicher zu machen.

jsonwebtoken

- Eine Bibliothek zum Erstellen und Verifizieren von JSON Web Tokens (JWT), die für die Benutzerauthentifizierung verwendet werden.

Architektur des Projekts

Frontend-Architektur

Komponentenbasierte Struktur

- Die Anwendung ist in wiederverwendbare und eigenständige Komponenten unterteilt. Zum Beispiel: `Register.js` für das Registrierungsformular.

State Management

- Verwendung von React Hooks (`useState`, `useEffect`, etc.) zur Verwaltung des Komponentenzustands und der Seiteneffekte.

Backend-Architektur

Model-View-Controller (MVC) Architektur

- **Model**: Definiert die Datenstrukturen und interagiert mit der Datenbank (z.B. `User` und `Employee` Modelle).
- **View**: In diesem Projekt wird die View vom Frontend bereitgestellt.
- **Controller**: Enthält die Geschäftslogik und verarbeitet eingehende Anfragen. Beispiel: `handleLogin`, `getAllEmployees`, etc.

Routing

- Express.js wird verwendet, um verschiedene Routen für die API-Endpunkte zu definieren. Beispiel: POST `/register`, GET `/employees`, etc.

Middleware

- Verwendung von Middleware-Funktionen in Express, um Anfragen zu verarbeiten, bevor sie die Endpunkte erreichen. Beispiel: Authentifizierungs-Middleware, um geschützte Routen zu sichern.

Datenbank-Architektur

Relationale Datenbank

- Verwendung von MySQL, um Daten in tabellarischer Form zu speichern. Tabellen haben definierte Beziehungen untereinander.
- **User-Tabelle**: Speichert Benutzerdaten inklusive gehashter Passwörter und Rollen.
- **Employee-Tabelle**: Speichert Daten zu Mitarbeitern.

ORM (Sequelize)

- Verwaltet die Datenbankoperationen und stellt sicher, dass Datenbankabfragen sicher und effizient sind.
- Definiert Modelle für die Tabellen (z.B. User, Employee) und deren Beziehungen.

Zusätzliche Technologien und Praktiken

Environment Variables

- Verwendung von dotenv, um sensible Konfigurationsdaten wie Datenbankverbindungsdetails und geheime Schlüssel sicher zu verwalten.

Version Control

- Verwendung von Git für die Quellcodeverwaltung und Zusammenarbeit.

Testing

- Verwendung von Jest und React Testing Library für automatisierte Frontend-Tests.
- Verwendung von Mocha und Chai für Backend-Tests.

Testprotokoll

Leider musste ich während den Automatisierten und manuellen Testing feststellen, dass die App nicht wie gedacht funktioniert. Die mitgegebenen Rollen, werden nicht richtig in die Datenbank übernommen. Dort wird automatisch der User als Default eingegeben. Das heisst natürlich auch, dass neu erstellte Nutzer keinen Zugriff auf die verschiedenen Routes haben, da die meisten mit Routeprotection keine normalen User erlaubten.

Arbeitsjournal und Fazit

Da ich das Projekt viel zu späht fertiggestellt habe, fällt es mir schwer ein Arbeitsjournal zu schreiben, da sich die Arbeit über mehrere Wochen hingezogen hat. Trotz der längeren Arbeitszeit funktioniert die App nicht so wie sie sollte. Ich bin weder mit meiner Arbeitsplanung, noch mit der Umsetzung des Projekts zufrieden.