



M223-Todo-App

By Leon

Leon Baltensperger

Inhalt

Intro	2
User Story 1: Aufgaben erstellen	3
User Story: Aufgabenstatus ändern	3
User Story 3: Benutzerkonten verwalten.....	4
Backend-Technologien und -Architektur.....	5
Transactional.....	6
Frontend-Technologie und -Architektur	7
Sicherheitskonzept für das Backend:	9
Sicherheitskonzept für das Frontend:	9
Git-Commits.....	10
Beschreibung der Abläufe beim Login	11
Arbeitsjournal.....	12

Intro

Willkommen zur Dokumentation unserer To-Do-Anwendung! Diese App soll Ihnen dabei helfen, Ihre Aufgaben ganz einfach zu organisieren und zu verwalten. Egal, ob Sie persönliche Ziele erreichen, berufliche Projekte managen oder einfach den Alltag im Griff behalten wollen – unsere To-Do-App bietet Ihnen die richtigen Werkzeuge, um produktiver zu sein und den Überblick zu behalten.

Users

Admin User:

Name Admin

Password: 12345678

Das Projekt ist in Github.

[leon23132/M223-IFZ-013-Leon-ToDoApp \(github.com\)](https://github.com/leon23132/M223-IFZ-013-Leon-ToDoApp)

User Story 1: Aufgaben erstellen

Als Nutzer

möchte **ich** neue Aufgaben hinzufügen können

damit ich meine zu erledigenden Aufgaben organisieren und verfolgen kann.

Akzeptanzkriterien:

1. Der Nutzer kann eine neue Aufgabe mit Titel und Beschreibung hinzufügen.
2. Der Nutzer kann ein Fälligkeitsdatum für die Aufgabe festlegen.
3. Die Aufgabe wird in der Liste der anstehenden Aufgaben angezeigt.
4. Der Nutzer erhält eine Bestätigung, dass die Aufgabe erfolgreich hinzugefügt wurde.

User Story: Aufgabenstatus ändern

Als Nutzer

möchte ich den Status meiner Aufgaben ändern können

damit ich den Fortschritt meiner Aufgaben verfolgen und klar erkennen kann, welche Aufgaben noch offen, in Bearbeitung oder erledigt sind.

Akzeptanzkriterien:

5. Der Nutzer kann den Status einer Aufgabe als "Offen", "In Bearbeitung" oder "Erledigt" markieren.
6. Der aktuelle Status jeder Aufgabe wird in der Aufgabenliste angezeigt.
7. Der Nutzer kann den Status einer Aufgabe jederzeit ändern.
8. Änderungen am Aufgabenstatus werden sofort in der Aufgabenliste reflektiert.
9. Der Nutzer kann Aufgaben nach ihrem Status filtern und sortieren.

User Story 3: Benutzerkonten verwalten

Als Nutzer

möchte ich ein Benutzerkonto erstellen und mich anmelden können
damit ich meine Aufgaben sicher speichern und von verschiedenen Geräten darauf zugreifen kann.

Akzeptanzkriterien:

1. Der Nutzer kann ein neues Benutzerkonto mit E-Mail und Passwort erstellen.
2. Der Nutzer kann sich mit seinen Anmeldedaten in sein Konto einloggen.
3. Der Nutzer kann sein Passwort zurücksetzen, falls er es vergisst.
4. Der Nutzer kann auf seine Aufgabenliste zugreifen und diese bearbeiten, nachdem er sich eingeloggt hat.
5. Die Aufgabenliste des Nutzers bleibt über verschiedene Geräte hinweg synchronisiert.

Backend-Technologien und -Architektur

Unsere To-Do-Anwendung verwendet eine RESTful API-Architektur, die auf dem Spring Boot-Framework basiert. Dieses ermöglicht eine schnelle Entwicklung von robusten und skalierbaren Java-Anwendungen. Die API bietet Endpunkte für das Erstellen, Lesen, Aktualisieren und Löschen (CRUD) von Aufgaben und die Verwaltung von Benutzerkonten. Hier ist ein Überblick über die verwendeten Technologien und die Architektur.

Technologien

- **Spring Boot:** Ein Framework für die Entwicklung von Java-Anwendungen, das auf dem Spring Framework basiert und die Erstellung von Produktionsanwendungen vereinfacht.
- **Spring Data JPA:** Ein Modul, das die Implementierung von Datenzugriffsschichten erleichtert und JPA (Java Persistence API) zur Kommunikation mit relationalen Datenbanken nutzt.
- **Spring Security:** Ein Framework für Authentifizierungs- und Autorisierungsdienste.
- **JSON Web Tokens (JWT):** Eine Methode zur sicheren Übertragung von Informationen zwischen Parteien als JSON-Objekt.
- **MySQL:** Eine relationale Datenbank, die für die Speicherung der Anwendungsdaten verwendet wird.
- **Maven:** Ein Build-Management-Tool, das verwendet wird, um die Projektabhängigkeiten und den Build-Prozess zu verwalten.

Architektur

Die Anwendung ist in mehrere Schichten unterteilt, die jeweils eine spezifische Rolle in der Gesamtarchitektur spielen:

1. **Controller-Schicht:** Beinhaltet die REST-Controller, die die Endpunkte der API definieren. Diese Schicht nimmt Anfragen von den Clients entgegen und gibt Antworten zurück.
2. **Service-Schicht:** Enthält die Geschäftslogik der Anwendung. Diese Schicht verarbeitet die Daten, die von der Controller-Schicht übergeben werden, und führt die erforderlichen Operationen aus.
3. **Repository-Schicht:** Diese Schicht ist für die Datenzugriffsebene verantwortlich. Sie kommuniziert mit der Datenbank über JPA und führt CRUD-Operationen aus.
4. **Sicherheits-Schicht:** Verwaltet die Authentifizierungs- und Autorisierungsprozesse mithilfe von Spring Security und JWT.

Transactional

Die @Transactional Annotation wird in Spring Framework verwendet, um die Transaktionsunterstützung für Methoden oder Klassen bereitzustellen. Wenn eine Methode mit @Transactional annotiert ist, wird eine Transaktion automatisch um den Aufruf dieser Methode herum gestartet.

Eine Transaktion ist eine Gruppe von Datenbankoperationen, die als eine einzige Einheit ausgeführt werden. Dies bedeutet, dass alle Operationen entweder vollständig durchgeführt werden oder dass keine von ihnen durchgeführt wird. Wenn eine Operation in der Transaktion fehlschlägt, werden alle Änderungen rückgängig gemacht (gerollt), um die Datenbank in einen konsistenten Zustand zurückzubringen.

Die Verwendung der @Transactional Annotation bietet mehrere Vorteile:

1. **Automatische Transaktionsverwaltung:** Spring verwaltet automatisch den Beginn, das Commit und das Rollback von Transaktionen, was die Entwickler von manuellen Transaktionsverwaltungen befreit.
2. **Konsistenz der Datenbank:** Durch die Transaktionsunterstützung wird sichergestellt, dass die Datenbank in einem konsistenten Zustand bleibt, auch wenn es zu Fehlern während der Operationen kommt.
3. **Sicherheit:** Die @Transactional Annotation hilft, sicherzustellen, dass Operationen atomar ausgeführt werden, was bedeutet, dass sie entweder vollständig oder gar nicht ausgeführt werden.

Frontend-Technologie und -Architektur

Meine Frontend-Anwendung verwendet eine moderne Architektur und eine Reihe von Technologien, um eine benutzerfreundliche und reaktionsschnelle Benutzeroberfläche zu ermöglichen.

Technologien

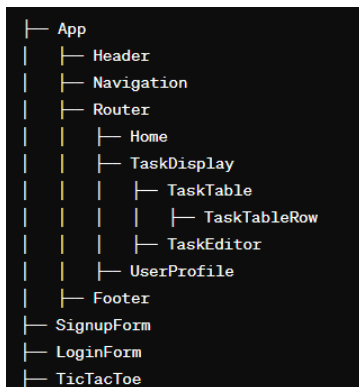
- **React:** Eine JavaScript-Bibliothek zur Entwicklung von Benutzeroberflächen, die eine deklarative und komponentenbasierte Herangehensweise bietet.
- **React Router:** Eine Bibliothek zur Navigation und zum Routing in React-Anwendungen, die eine strukturierte Navigation zwischen verschiedenen Ansichten ermöglicht.
- **Axios:** Eine Bibliothek zur Durchführung von HTTP-Anfragen aus dem Browser, die für die Kommunikation mit dem Backend verwendet wird.
- **Testing-Libraries:** Enthält Jest für das Testen von JavaScript-Code sowie Testing Library für das Testen von React-Komponenten und Benutzerinteraktionen.

Architektur

Unser Frontend ist nach dem Komponentenmodell strukturiert, das eine klare Trennung der verschiedenen Teile der Benutzeroberfläche ermöglicht:

1. **Komponenten:** Die Anwendung ist in wiederverwendbare Komponenten unterteilt, die jeweils eine spezifische Funktionalität oder Ansicht repräsentieren. Diese Komponenten können sowohl Zustandskomponenten als auch reine Funktionen sein.
2. **Routing:** Das React Router-Modul ermöglicht die Definition von Routen und das Navigieren zwischen verschiedenen Ansichten innerhalb der Anwendung. Dies erleichtert die Strukturierung und Organisation des Codes.
3. **Datenzugriff:** Axios wird für den Datenzugriff verwendet, um HTTP-Anfragen an das Backend zu senden und Daten abzurufen oder zu aktualisieren. Dies ermöglicht eine effiziente Kommunikation mit dem Server.

Illustration der React-Komponenten



Erklärung der Hierarchie

- **App:** Die Hauptkomponente, die die gesamte Anwendung umschließt und die Header-, Navigation-, Router- und Footer-Komponenten enthält.
- **Header:** Die Kopfzeile der Anwendung mit dem Logo und anderen Header-Elementen.
- **Navigation:** Die Navigationsleiste, die dem Benutzer die Navigation zwischen verschiedenen Ansichten ermöglicht.
- **Router:** Die React Router-Komponente, die die verschiedenen Routen definiert und die entsprechenden Ansichten rendert.
- **Home:** Die Startseite der Anwendung mit einer Begrüßungsnachricht und möglicherweise anderen Inhalten wie dem TicTacToe-Spiel.
- **TaskDisplay:** Eine Ansicht zur Anzeige und Verwaltung von Aufgaben.
- **TaskTable:** Eine Tabelle zur Anzeige einer Liste von Aufgaben.
- **TaskTableRow:** Eine Zeile in der Aufgabenliste mit Informationen zu einer einzelnen Aufgabe.
- **TaskEditor:** Ein Formular zum Bearbeiten von Aufgaben oder zum Hinzufügen neuer Aufgaben.
- **UserProfile:** Eine Ansicht zur Anzeige und Bearbeitung des Benutzerprofils.
- **Footer:** Die Fußzeile der Anwendung mit Links und anderen Informationen.
- **SignupForm:** Ein Formular zum Registrieren neuer Benutzer.
- **LoginForm:** Ein Formular zum Einloggen registrierter Benutzer.
- **TicTacToe:** Ein Spiel zur Unterhaltung des Benutzers auf der Startseite.

Sicherheitskonzept für das Backend:

Authentifizierung und Autorisierung:

- Implementierung eines Token-basierten Authentifizierungsmechanismus, z. B. JWT (JSON Web Token).
- Überprüfung der Authentizität des Tokens bei jeder Anfrage und Berechtigungsprüfung, um sicherzustellen, dass der Benutzer die erforderlichen Berechtigungen für die angeforderten Aktionen hat.

Passwortsicherheit:

- Verwendung sicherer Passwortrichtlinien, z. B. Mindestlänge, Komplexität und Hashing der Passwörter vor der Speicherung in der Datenbank.
- Verwendung von Salt beim Hashen von Passwörtern, um Angriffe wie Rainbow-Tabellen zu erschweren.

Schutz vor Injection-Angriffen:

- Verwendung von Parameterized Queries oder Prepared Statements, um SQL-Injection-Angriffe zu verhindern.
- Validierung und Bereinigung von Benutzereingaben, um andere Arten von Injection-Angriffen zu verhindern, z. B. NoSQL-Injection.

Sicherheitskonzept für das Frontend:

HTTPS-Verschlüsselung:

- Gewährleistung der Datenintegrität und -vertraulichkeit durch die Verwendung von HTTPS für die Kommunikation zwischen dem Frontend und dem Backend.

Benutzerauthentifizierung:

- Implementierung einer sicheren Benutzerauthentifizierung, die Benutzeranmeldeinformationen sicher überträgt und die Identität des Benutzers überprüft, z. B. mit JWT (JSON Web Token).
- Speichern von Tokens sicher im Local Storage und regelmäßiges Überprüfen auf deren Ablaufdatum, um sicherzustellen, dass sie gültig sind.

Schutz vor Cross-Site Scripting (XSS):








- Validierung und Bereinigung von Benutzereingaben, um XSS-Angriffe zu verhindern, die durch Einschleusen von böartigem JavaScript in die Anwendung erfolgen können.
- Verwendung von Content Security Policy (CSP), um die Ausführung von nicht autorisiertem JavaScript auf der Seite zu verhindern.

Rollen

Admin: Hat alle Berechtigungen.

User: Kann nur auf den Task zugreifen

Git-Commits

Commits on May 14, 2024		
LoginWithTokenFrontend	leon23132 committed last week	ca9196c  <>
Commits on May 7, 2024		
Login	leon23132 committed 2 weeks ago	5b8afa2  <>
Commits on Apr 16, 2024		
LoginSQ48	leon23132 committed last month	dae4ba2  <>
NEWModels	leon23132 committed last month	7e03338  <>
Commits on Apr 9, 2024		
FirstDay-Documentation	leon23132 committed last month	1ccc7f0  <>
ProjectStructure	leon23132 committed last month	352706e  <>
Initial commit	leon23132 committed last month	Verified 47d449d  <>

Beschreibung der Abläufe beim Login

1. Benutzeranmeldung:

- Der Benutzer navigiert zur Anmeldeseite der Anwendung.
- Dort gibt er seine Anmeldeinformationen ein, einschließlich E-Mail-Adresse und Passwort.
- Nachdem der Benutzer seine Anmeldeinformationen eingegeben hat, sendet die Anwendung eine Anfrage an den Backend-Server zur Überprüfung der Authentizität der Anmeldeinformationen.

2. Authentifizierung und Token-Erstellung:

- Der Backend-Server überprüft die Anmeldeinformationen anhand der in der Datenbank gespeicherten Benutzerdaten.
- Bei erfolgreicher Überprüfung generiert der Server ein JWT (JSON Web Token), das die Identität des Benutzers bestätigt und ihm Zugriff auf geschützte Ressourcen gewährt.
- Das JWT wird dem Frontend zurückgesendet und im Local Storage des Browsers gespeichert.

3. Zugriff auf geschützte Ressourcen:

- Nachdem der Benutzer erfolgreich angemeldet ist und ein gültiges JWT besitzt, kann er auf geschützte Ressourcen der Anwendung zugreifen, z. B. seine Aufgabenliste oder sein Benutzerprofil.
- Bei jeder Anfrage an den Backend-Server sendet das Frontend das JWT im Authentifizierungsheader mit, um seine Identität zu bestätigen und sicherzustellen, dass der Benutzer autorisiert ist.

4. Token-Verwaltung und Ablauf:

- Das Frontend überprüft regelmäßig das Ablaufdatum des JWT im Local Storage, um sicherzustellen, dass es noch gültig ist.
- Wenn das JWT abgelaufen ist, wird der Benutzer automatisch abgemeldet und zur erneuten Anmeldung aufgefordert.
- Beim Abmelden oder beim Ablauf des JWT wird das Token aus dem Local Storage entfernt, um die Sicherheit des Benutzerkontos zu gewährleisten.

Arbeitsjournal

1	Projektstart und Konzeption	01.05.2024 - 14.05.2024
2	Backend-Entwicklung	15.05.2024 - 28.05.2024
3	Frontend-Entwicklung	29.05.2024 - 11.06.2024
4	Feinschliff und Testing	12.06.2024 - 18.06.2024
5	Rollout und Bereitstellung	19.06.2024 - 25.06.2024

ERD MySQL Database

