



**Hochschule
Kaiserslautern**
University of
Applied Sciences

Projekt 2

Konzeption und Entwicklung einer Arbeitszeiterfassung: Konzeption, Design und Implementierung

"Welche Auswirkungen hat die Einführung eines digitalen Arbeitszeiterfassungssystems auf die Produktivität und Mitarbeiterzufriedenheit in mittelständischen Betrieben und welche technischen Schwierigkeiten sind bei der Entwicklung und Einführung eines solchen Systems zu bewältigen?"

Studiengang

Wirtschaftsinformatik

Autor/Student

Name	Vorname	Matrikelnummer	Adresse
Tsane Tsafack	Billy Max	882814	Virginias Straße 13, 66482 Zweibrücken

Abgabedatum

16.07.2024

Unter der Leitung von

Dipl.-Handelslehrer Andreas Heß

Inhaltsverzeichnis

Inhaltsverzeichnis.....	I
Abbildungsverzeichnis	III
Abkürzungsverzeichnis	IV
1. Einleitung.....	- 1 -
2. Technologien und Tools	- 2 -
2.1 Programmiersprachen und Frameworks	- 2 -
2.1.1 Frontend	- 2 -
2.1.2 Backend	- 3 -
2.1.3 Datenbank.....	- 4 -
3. Use Case und Anwendungsfälle	- 5 -
3.1 Beschreibung des Haupt-Use-Cases: Zeiterfassung durch Mitarbeiter	- 5 -
3.2 Detaillierte Schritte des Use-Case	- 5 -
3.2.1 Registrieren:	- 5 -
3.2.2 Login:	- 5 -
3.2.3 Arbeitsbeginn erfassen (Check-in):.....	- 6 -
3.2.4 Arbeitsende erfassen (Check-out):	- 6 -
4. Dashboard Design und Funktionen	- 8 -
4.1 Konzept des Dashboards.....	- 8 -
4.2 Wichtige Funktionen des Dashboards	- 8 -
4.3 Prototyp.....	- 9 -
5. Technische Umsetzung	- 10 -
5.1 Frontend-Entwicklung	- 10 -
5.2 Backend-Entwicklung.....	- 12 -
6. Zusammenfassung und Ausblick	- 15 -

6.1 Zusammenfassung der Ergebnisse	- 15 -
6.2 Ausblick auf zukünftige Entwicklungen und Forschungsbedarf	- 15 -
Literaturverzeichnis	V
Anhang	VI
1. Login.jsx	VI
2. pageHomeTerminal.jsx	VII
3. PageSidebar.jsx	VIII
4 Time_EntriesController.java	IX
5 Time_EntriesServiceImpl.js	IX
6. AuthEntryPointJwt.java	XI
7. AuthTokenFilter.java	XII
8. JwtUtils.java	XIII
9. CrossSecurity.java	XIV
10. UserDetailsServiceImpl.java	XV
11. WebSecurityConfig.java	XVI
Eidesstattliche Erklärung	XIX

Abbildungsverzeichnis

Abbildung 1: UML-Diagramm für das Use Case	- 7 -
Abbildung 2: Erstellung des Dashboards (Screenshot).....	- 9 -
Abbildung 3: Komponentenstruktur von React (Screenshot aus Vscod).....	- 11 -
Abbildung 4: Login Page (Screenshot)	- 11 -

Abkürzungsverzeichnis

API	<i>Application Programming Interface</i>
CSS	<i>Cascading Style Sheets</i>
HTML	<i>Hyper Text Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
JWT.....	<i>JSON Web Tokens</i>
KI	<i>Künstliche Intelligenz</i>
SQL.....	<i>Structured Query Language</i>
UI	<i>User Interface</i>

1. Einleitung

Heutzutage gewinnen flexible Arbeitszeiten, Homeoffice Arbeit und projektbasierte Arbeit in der Arbeitswelt immer mehr an Bedeutung. Diese Entwicklungen erfordern präzise und flexible Systeme zur Erfassung der Arbeitszeiten. Traditionelle Methoden, wie manuelle Stundenzettel oder einfache elektronische Tabellen, stoßen hierbei an ihre Grenzen. Sie sind oft fehleranfällig, zeitaufwändig und bieten nur begrenzte Möglichkeiten zur Analyse und Berichterstattung.¹

Moderne Arbeitszeiterfassungssysteme bieten hier entscheidende Vorteile. Sie ermöglichen nicht nur eine genaue Erfassung der Arbeitszeiten, sondern auch die Verwaltung von Zeit Art, Teamauskunft, Salden und Urlaubsanträgen. Darüber hinaus unterstützen sie die Einhaltung gesetzlicher Vorgaben, wie beispielsweise Arbeitszeitgesetze und Datenschutzbestimmungen. Ein gut gestaltetes Zeiterfassungssystem kann somit die Effizienz und Transparenz in Unternehmen erheblich steigern.

Die Motivation für diese Arbeit ergibt sich aus dem Bedarf an einer benutzerfreundlichen, flexiblen und zuverlässigen Lösung zur Arbeitszeiterfassung. Ziel ist es, ein System zu entwickeln, das sowohl den Anforderungen der Benutzer als auch der Unternehmen gerecht wird und dabei die Vorteile moderner Technologien nutzt.

Das Hauptziel dieser Arbeit ist die Konzeption, Gestaltung und Implementierung einer Arbeitszeiterfassungsanwendung, die eine einfache und genaue Erfassung der Arbeitszeiten ermöglicht. Die Anwendung soll benutzerfreundlich sein, sodass Mitarbeiter ihre Arbeitszeiten problemlos eintragen können, und gleichzeitig den Unternehmen umfassende Auswertungs- und Verwaltungsfunktionen bieten.

Im Einzelnen verfolgt die Arbeit folgende Ziele:

- **Technologieauswahl:** Auswahl geeigneter Technologien und Tools, die eine effiziente und skalierbare Entwicklung ermöglichen.
- **Design und Prototypen:** Entwicklung eines benutzerfreundlichen Interface-Designs und Erstellung von Prototypen zur Veranschaulichung der Anwendung.
- **Implementierung:** Umsetzung der Anwendung unter Verwendung moderner Technologien für Frontend und Backend.
- **Qualitätssicherung:** Durchführung umfassender Tests, um die Funktionalität und Zuverlässigkeit der Anwendung sicherzustellen

¹ Hellert, U. (2018). Arbeitszeitmodelle der Zukunft - inkl. Arbeitshilfen online. <https://doi.org/10.34157/9783648119969>

2. Technologien und Tools

In diesem Kapitel geht es um die verschiedenen Technologien und Tools, die bei der Implementierung der Arbeitszeiterfassung verwendet werden. Dazu gehören insbesondere die Programmiersprachen und die Frameworks für Frontend, Backend und Datenbanken.

2.1 Programmiersprachen und Frameworks

Die Wahl der richtigen Programmiersprachen und Frameworks ist entscheidend für die Entwicklung einer effizienten Arbeitszeiterfassungsanwendung. Für das Frontend wurden HTML, CSS und JavaScript mit dem React-Framework gewählt, um eine dynamische und benutzerfreundliche Oberfläche zu schaffen. Im Backend kommen Java und Spring Boot zum Einsatz, die für ihre Robustheit und umfangreiche Funktionalitäten bekannt sind. Als Datenbankmanagementsystem wird MySQL verwendet, um eine zuverlässige und performante Datenspeicherung sicherzustellen. Für die API-Entwicklung und -Tests wird Postman eingesetzt, um eine reibungslose Kommunikation zwischen Frontend und Backend zu gewährleisten. Diese Technologien wurden ausgewählt, um eine optimale Balance zwischen Benutzerfreundlichkeit, Leistungsfähigkeit und Erweiterbarkeit zu gewährleisten.

2.1.1 Frontend

Bei diesem ersten Punkt geht es einfach darum, die Rolle des Frontend zu erklären, die Funktionen der verschiedenen Programmiersprachen und Frameworks des Frontend zu nennen und zu erklären, insbesondere HTML, CSS, JavaScript und React. Also das Frontend ist ein Begriff aus der Softwareentwicklung. Das Frontend umfasst alles, was der Benutzer einer Software oder einer Webseite sieht, berührt und erlebt. Bei einer Website umfasst das Frontend die Inhalte - Posts, Seiten, Medien und Kommentare -, das Design sowie die Navigation. Ein weiteres Beispiel wäre das Frontend einer Datenbank-Software, wo Nutzer Daten eingeben und anzeigen lassen können.²

Nun folgen die Erklärungen zu den Programmiersprachen und den erwähnten Frameworks:

HTML: Also HTML ist die Abkürzung für den englischen Begriff Hypertext Markup Language. Übersetzt steht dies für Hypertext-Auszeichnungssprache. Hierbei handelt es sich um eine textbasierte Auszeichnungssprache, die seit einigen Jahrzehnten für die Strukturierung elektronischer Dokumente genutzt wird.³ Es ist wichtig zu verstehen, dass HTML keine Pro-

² Seobility. (o. D.). Frontend (vs. Backend) - Was ist das? - Seobility Wiki. <https://www.seobility.net/de/wiki/Frontend>

³ Akademie, D. (2024, 26. Juni). HTML-Grundlagen im Überblick. Developer Akademie. <https://developerakademie.com/blog/all/html-grundlagen-im-ueberblick>

grammiersprache ist, sondern eine Auszeichnungssprache. Das bedeutet, dass HTML nicht dazu dient, interaktive Funktionen oder komplexe Logik auf einer Website zu programmieren. Stattdessen legt man mit HTML die Struktur und den Inhalt einer Webseite fest.

CSS: Cascading Style Sheets (CSS) ist eine Programmiersprache, die es Ihnen ermöglicht, das Design von elektronischen Dokumenten zu bestimmen. Anhand einfacher Anweisungen – dargestellt in übersichtlichen Quellcodes – lassen sich so Webseiten-Elemente wie Layout, Farbe und Typografie nach Belieben anpassen.⁴

JAVASCRIPT: JavaScript ist eine Programmiersprache, die Entwickler verwenden, um interaktive Webseiten zu erstellen. Von der Aktualisierung von Social Media Feeds bis hin zur Anzeige von Animationen und interaktiven Karten können JavaScript-Funktionen die Benutzerfreundlichkeit einer Website verbessern. Als clientseitige Skriptsprache ist sie eine der Kerntechnologien des World Wide Web. Wenn Sie zum Beispiel im Internet surfen und ein Bildkarussell, ein Dropdown-Menü zum Anklicken oder dynamisch wechselnde Elementfarben auf einer Webseite sehen, sehen Sie die Auswirkungen von JavaScript.⁵

REACT(Framework): React ist eine JavaScript-Programmbibliothek zur Erstellung von webbasierten Benutzeroberflächen. Komponenten werden in React hierarchisch aufgebaut und können in dessen Syntax als selbst definierte JSX-Tags repräsentiert werden. Das Modell von React verspricht durch die Konzepte des unidirektionalen Datenflusses und des Virtual DOM den einfachen, aber trotzdem performanten Aufbau auch komplexer Anwendungen. React bildet typischerweise die Basis für Single-Page-Webanwendungen, kann jedoch auch mit Node.js serverseitig (vor-)gerendert werden.⁶

2.1.2 Backend

Das Backend dieser Arbeit basiert sich auf Java und Spring Boot als Programmiersprachen und Framework. Als das Frontend ist das Backend auch ein Begriff aus der Softwareentwicklung. Das Backend bezeichnet in der Regel den funktionalen Teil eines digitalen Produkts wie beispielsweise Webseiten oder Apps und bezieht sich meist auf Client Server. Während das Frontend näher am Benutzer ist, ist das Backend näher am System. Programmierer arbeiten im Backend, Webdesigner im Frontend. Das Backend stellt dabei den administrativen Teil dar.⁷

⁴ Redaktion, I. (2022, 21. Juni). Was ist CSS? Definition und Anwendung. IONOS Digital Guide. <https://www.ionos.de/digitalguide/websites/webdesign/was-ist-css>

⁵ Was ist JavaScript? – JavaScript (JS) erklärt – AWS. (o. D.). Amazon Web Services, Inc. <https://aws.amazon.com/de/what-is/javascript>

⁶ Wikipedia-Autoren. (2016, 6. September). React. <https://de.wikipedia.org/wiki/React>

⁷ IT-SERVICE.NETWORK. (2024, 15. März). Backend. <https://it-service.network/it-lexikon/backend>

folgen sind die Definitionen von Java und Spring Boot:

JAVA: Java ist eine weitverbreitete objektorientierte Programmiersprache und Softwareplattform, die auf Milliarden von Geräten läuft, darunter Notebooks, mobile Geräte, Spielkonsolen, medizinische Geräte und viele andere. Java ist eine vielseitige Sprache, die weit über traditionelle Anwendungen hinausgeht. Sie bildet das Herzstück des Android-Betriebssystems, ist beliebt für maschinelles Lernen und Data-Science und wird wegen ihrer Robustheit, Benutzerfreundlichkeit, plattformübergreifenden Fähigkeiten und Sicherheit häufig für Internetlösungen verwendet. Java-Technologie ist ideal für die Entwicklung von Webanwendungen und bietet mit Java-Anwendungsservern eine stabile Umgebung für Unternehmensanwendungen. Diese Server unterstützen Funktionen wie Transaktionsmanagement, Sicherheit, Clustering, Leistung, Verfügbarkeit, Konnektivität und Skalierbarkeit, was sie zur bevorzugten Wahl für digitale Unternehmen macht.⁸

SPRING BOOT(Framework): Java Spring Boot ist ein Open-Source-Tool, das die Verwendung von Java-basierten Frameworks zur Erstellung von Microservices und Webanwendungen erleichtert. Für jede Definition von Spring Boot muss das Gespräch mit Java beginnen - eine der beliebtesten und am weitesten verbreiteten Entwicklungssprachen und Computerplattformen für die App-Entwicklung. Entwickler auf der ganzen Welt beginnen ihre Reise in die Programmierung mit Java. Java ist flexibel und benutzerfreundlich und wird von Entwicklern gerne für eine Vielzahl von Anwendungen eingesetzt - von Social Media-, Web- und Spieleanwendungen bis hin zu Netzwerk- und Unternehmensanwendungen.⁹

2.1.3 Datenbank

Als die Datenbank, spielt MySQL für die Arbeit eine große Rolle. MySQL ist ein weit verbreitetes Open-Source-SQL-Datenbankmanagementsystem, das von Oracle Corporation entwickelt und unterstützt wird. Es handelt sich um eine relationale Datenbank, die Daten in separaten, optimierten Tabellen speichert und mithilfe der standardisierten Sprache SQL verwaltet wird. MySQL bietet hohe Geschwindigkeit, Zuverlässigkeit und Skalierbarkeit, wodurch es sowohl für kleine Anwendungen als auch für große, anspruchsvolle Produktionsumgebungen geeignet ist. Es kann als Client/Server-System oder eingebettetes System genutzt werden und ist für seine einfache Handhabung und umfassende Funktionalität bekannt. MySQL ist plattformunabhängig und bietet durch die Open-Source-Lizenzierung große Flexibilität und Anpassungsfähigkeit.¹⁰

⁸ Was ist Java? | IBM. (o. D.). <https://www.ibm.com/de-de/topics/java>

⁹ What is Java Spring Boot?—Intro to Spring Boot | Microsoft Azure. (o. D.). <https://azure.microsoft.com/en-gb/resources/cloud-computing-dictionary/what-is-java-spring-boot>

¹⁰ <https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.htm>

3. Use Case und Anwendungsfälle

Einfach ausgedrückt wird mit einem Use-Case die nach außen sichtbarer Interaktion eines Nutzers mit einem System dokumentiert. Dieser Nutzer kann wie bereits erwähnt entweder eine Person, eine Organisation oder ein anderes System sein. Dabei besteht am Anfang vom Anwendungsfall ein Auslöser, der den Nutzer dazu veranlasst, in Interaktion mit dem System zu treten. Der Nutzer tritt hierbei also als Akteur auf, der mit der Verwendung des Systems ein bestimmtes Ziel bzw. ein Ergebnis erreichen möchte. Dieses Ziel erreicht der Akteur, indem er eine gewissen Reihe an Aktionen durchführt.

Im Rahmen dieser Arbeit werden wir einen Anwendungsfall haben, in dem die Akteure die Mitarbeiter sind und das System das Zeiterfassungssystem ist.¹¹

3.1 Beschreibung des Haupt-Use-Cases: Zeiterfassung durch Mitarbeiter

Mitarbeiter haben die Möglichkeit, ihre Arbeitszeiten mithilfe des Webterminals zu erfassen, ihre Tagesübersicht einzusehen und Informationen zu ihren Salden und dem Team zu erhalten. Darüber hinaus beinhaltet der Use-Case die Ergänzung von Kommentaren und die Änderung des Status (Anwesend/Abwesend).

3.2 Detaillierte Schritte des Use-Case

3.2.1 Registrieren:

- **Beschreibung:** Nur der Admin (Mitarbeiter) registriert den Mitarbeitern im System, um Zugang zu den Funktionen zu erhalten.
- **Akteur:** Admin
- **Schritte:**
 1. Admin öffnet die Registrierungsseite.
 2. Admin gibt erforderliche Informationen ein (Name, E-Mail, Passwort, etc.).
 3. Admin bestätigt die Registrierung.

3.2.2 Login:

- **Beschreibung:** Der Mitarbeiter loggt sich in das System ein.
- **Akteur:** Mitarbeiter und Admin

¹¹Steubel, P. (2024, 3. März). Was ist ein Use-Case? Definition und Nutzen im Überblick! [2024] • Asana. Asana. <https://asana.com/de/resources/what-is-a-use-case>

- **Schritte:**

1. Mitarbeiter oder Admin öffnet die Login-Seite.
2. Mitarbeiter oder Admin gibt seine Anmeldedaten ein (Username, Passwort).
3. Mitarbeiter oder Admin klickt auf „Anmelden“.

3.2.3 Arbeitsbeginn erfassen (Check-in):

- **Beschreibung:** Der Mitarbeiter und Admin erfassen den Beginn seiner Arbeitszeit.
- **Akteur:** Mitarbeiter und Admin
- **Schritte:**

1. Mitarbeiter oder Admin wählt im Webterminal den Status „Anwesend“ aus.
2. Mitarbeiter oder Admin fügt ggf. Kommentare hinzu.
3. Mitarbeiter oder Admin klickt auf „Kommen“.
4. System speichert die Startzeit und aktualisiert die Tagesübersicht.

3.2.4 Arbeitsende erfassen (Check-out):

4. **Beschreibung:** Der Mitarbeiter und Admin erfassen das Ende seiner Arbeitszeit.
5. **Akteur:** Mitarbeiter und Admin
6. **Schritte:**
 1. Mitarbeiter oder Admin wählt im Webterminal den Status „Abwesend“ aus.
 2. Mitarbeiter oder Admin fügt ggf. Kommentare hinzu.
 3. Mitarbeiter oder Admin klickt auf „Gehen“.
 4. System speichert die Endzeit und aktualisiert die Tagesübersicht.

Folgend wird die Darstellung des UML-Diagramms und seine verschiedenen Funktionen beziehungsweise der Zusammenhang zwischen die Akteure und die Use Cases.

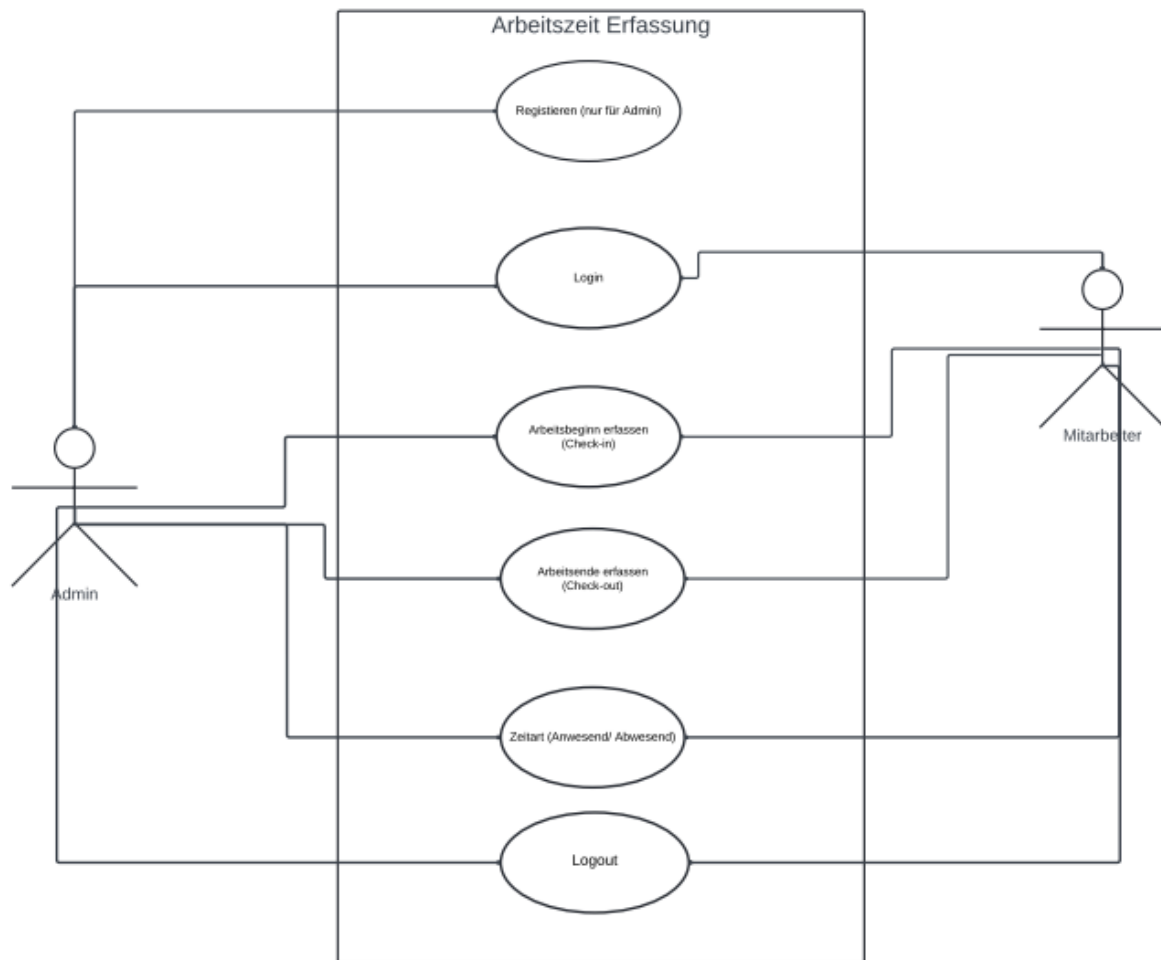


Abbildung 1: UML-Diagramm für das Use Case

4. Dashboard Design und Funktionen

Ein gut gestaltetes Dashboard ist das Herzstück einer Arbeitszeiterfassungsanwendung. Es ermöglicht den Benutzern, ihre Arbeitszeiten effizient zu verwalten, ihre Aktivitäten zu überwachen und Berichte zu generieren. Das Dashboard dient als zentrale Anlaufstelle, die eine intuitive Navigation und eine klare Darstellung der wesentlichen Informationen gewährleistet. In den folgenden Abschnitten werden das Konzept des Dashboards sowie die wichtigsten Funktionen detailliert erläutert.

4.1 Konzept des Dashboards

Das Konzept des Dashboards basiert auf Benutzerfreundlichkeit und Effizienz. Es bietet eine übersichtliche und leicht verständliche Benutzeroberfläche, die den Zugriff auf alle wesentlichen Funktionen wie Zeiterfassung und Berichterstellung ermöglicht. Das Design berücksichtigt die Bedürfnisse der Benutzer und stellt sicher, dass alle relevanten Informationen schnell und unkompliziert zugänglich sind.

4.2 Wichtige Funktionen des Dashboards

In diesem Punkt werden alle wichtigen Funktionen des Dashboards ausführlich erklärt. Die Funktionsweise des Dashboards zu verstehen, ist für alle Benutzer sehr wichtig, und vor allem vermittelt es ein echtes Verständnis des Dashboards und seiner Nutzung. Es ist wichtig zu betonen, dass das Dashboard erst erscheint, nachdem sich der Benutzer, der entweder ein einfacher Mitarbeiter oder ein Administrator ist, angemeldet hat.

Die wichtigen Funktionen des Dashboards sind;

Webterminal: Hier können die Benutzern Ihre „Kommen“ oder „Gehen“ Buchung erfassen. Für eine „Kommen“ Buchung wählen den Benutzern dazu die Zeitart aus. Schreiben sie evtl. einen Kommentar und klicken den Benutzern auf die Schaltfläche „Kommen“.

Für eine „Gehen“ Buchung klicken den Benutzern auf die Schaltfläche „Gehen“.

Tagesübersicht: In der Tagesübersicht sehen den Benutzern Ihre heute getätigten Buchungen.

Salden: In den Salden bekommen den Benutzern die Werte des Urlaubes und der Istzeit angezeigt.

Teamauskunft: Hier sehen den Benutzern immer aktuell welche/r Mitarbeiter/in in Ihrem Unternehmen anwesend ist.

Urlaub: Hier ist schnell ersichtlich wie viel Urlaub bereits genommen oder geplant wurde. Ebenfalls wird der verbleibende Resturlaub angezeigt.

So sieht das Dashboard im Grunde aus:

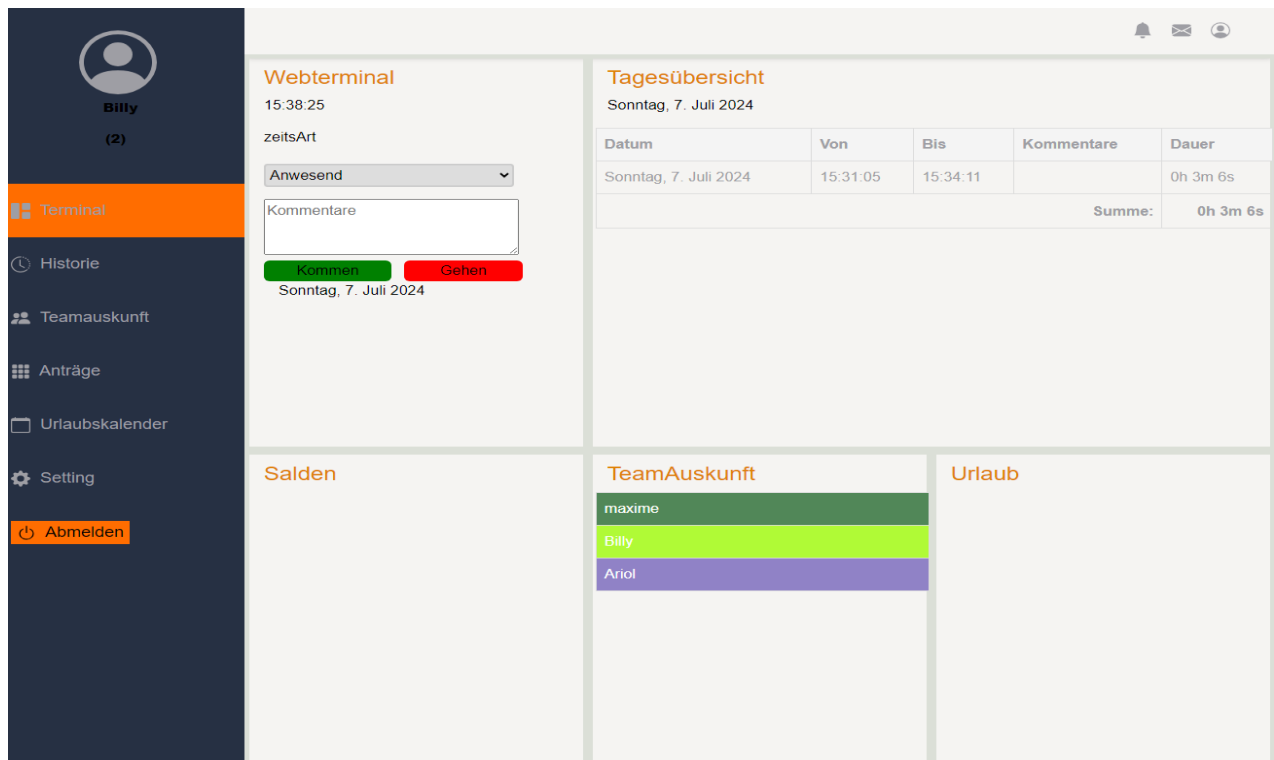


Abbildung 2: Erstellung des Dashboards (Screenshot)

Zusammenfassend lässt sich sagen, dass das Dashboard die erste Seite ist, auf der die Nutzer Zugriff auf all diese Funktionen haben.

4.3 Prototyp

Bei dem Prototyp stammt die Referenz aus den verschiedenen Templates der Firma Gdi Software GmbH. Für diese Arbeit wurden die Templates der GDI Zeit Compact: Zeiterfassung als Prototyp verwendet.¹²

¹² [GDI Webterminal: Webbasierte Zeiterfassung leicht gemacht](#)

5. Technische Umsetzung

Die technische Umsetzung der Arbeitszeiterfassungsanwendung umfasst die Entwicklung sowohl des Frontend als auch des Backend. Hierbei werden moderne Technologien und bewährte Methoden eingesetzt, um eine zuverlässige, performante und skalierbare Anwendung zu gewährleisten. Im Frontend liegt der Fokus auf einer reaktionsschnellen Benutzeroberfläche mit HTML, CSS, JavaScript und dem React-Framework. Das Backend wird mit Java und Spring Boot realisiert, um robuste und sichere Serverfunktionen zu bieten. Die Daten werden in einer MySQL-Datenbank gespeichert, und die APIs werden mit Postman getestet und dokumentiert. In den folgenden Abschnitten werden die einzelnen Komponenten und deren Implementierung detailliert beschrieben.

5.1 Frontend-Entwicklung

Die technische Umsetzung des Frontend dieser Arbeitszeiterfassungsanwendung basiert auf dem React-Framework, das aufgrund seiner Effizienz und Flexibilität weit verbreitet ist. React ermöglicht die Erstellung von wiederverwendbaren UI-Komponenten, was die Entwicklung und Wartung großer Anwendungen erleichtert. Im Folgenden wird der Aufbau und die Umsetzung des React-Frontend im Detail erläutert. Die Coden für den wichtigen UI-Komponenten werden im Anhang gezeigt. Nur die Komponenten, Services und das Styling für das Dashboard und Login Page werden erklärt.

▪ Komponenten Struktur

Die Komponentenstruktur von React ermöglicht die Aufteilung der Benutzeroberfläche in wiederverwendbare, isolierte Komponenten. Jede Komponente kümmert sich um einen bestimmten Teil der UI und kann bei Bedarf erneut verwendet werden.

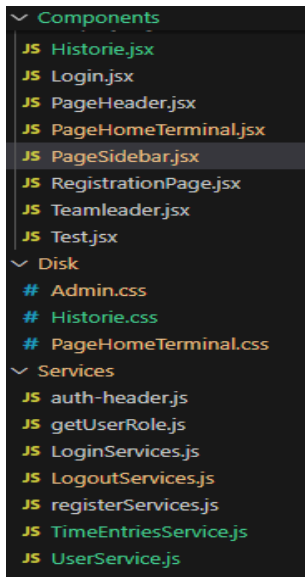


Abbildung 3: Komponentenstruktur von React (Screenshot aus Vscod)

Login.jsx: Ermöglicht den Benutzern das Einloggen in die Anwendung. Das Login Page sieht so aus.

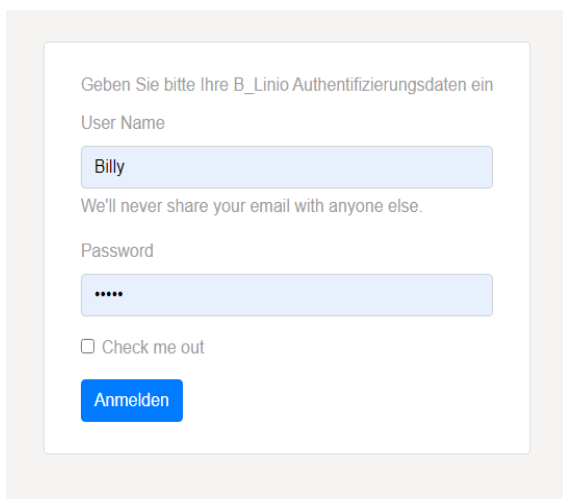


Abbildung 4: Login Page (Screenshot)

PageHomeTerminal.jsx: Die Hauptseite, die nach dem Einloggen angezeigt wird, mit den wichtigsten Funktionen der Anwendung.

PageSidebar.jsx: Die Seitenleiste, die Navigationslinks und weitere wichtige Informationen enthält.

▪ Services

Services in React sind separate Module, die API-Aufrufe und andere logische Opera-

tionen handhaben. Dies fördert die Trennung von Logik und Darstellung und erleichtert die Wartung und Erweiterung der Anwendung.

auth-header.js: Fügt Autorisierungsheader zu API-Anfragen hinzu.

getUserRole.js: Überprüft die Rolle des angemeldeten Benutzers.

LoginServices.js: Handhabt die Login-Operationen.

TimeEntriesService.js: Handhabt API-Aufrufe im Zusammenhang mit der Zeiterfassung.

UserService.js: Handhabt allgemeine Benutzeroperationen wie das Abrufen von Benutzerinformationen.

- **Styling**

Das Styling in React erfolgt häufig über CSS-Dateien, die spezifisch für die jeweiligen Komponenten sind. Dies ermöglicht eine saubere Trennung von Stil und Funktionalität.

PageHomeTerminal.css: Enthält spezifische Stile für die Hauptseite nach dem Einloggen.

5.2 Backend-Entwicklung

In Bezug auf die Entwicklung des Backends der Arbeitszeiterfassungssoftware verwendet Java und das Spring Boot Framework, das für seine Robustheit, Sicherheit und einfache Integration bekannt sind. Es folgt eine detaillierte Erläuterung des Aufbaus und der Implementierung des Backends, einschließlich der Struktur und der wesentlichen Bestandteile. Wie bei der Frontend werden auch die Implementierung Coden für die Backend Entwicklung im Anhang präsentiert.

- **Projektstruktur**

Die Struktur des Backend ist in verschiedene Pakete und Klassen unterteilt, die jeweils eine spezifische Funktionalität erfüllen. Dies fördern die Modularität und Wartbarkeit der Anwendung.

- **Controller**

Controller sind für die Verarbeitung von HTTP-Anfragen und die Definition der API-Endpunkte verantwortlich. Sie orchestrieren die Kommunikation zwischen dem Frontend und den Services.

PagesController.java: Kümmt sich um die Navigation und Seitendarstellung.

RoleController.java: Verarbeitet Anfragen im Zusammenhang mit Benutzerrollen.

StatusController.java: Handhabt Statusanfragen.

Time_EntriesController.java: Verarbeitet Anfragen zur Arbeitszeiterfassung.

▪ Modell

Modelle repräsentieren die Datenstruktur der Anwendung und sind oft mit Datenbanktabellen verknüpft.

LoginRequest.java: Datenmodell für Login-Anfragen.

Role.java: Datenmodell für Benutzerrollen.

Time_Entries.java: Datenmodell für Arbeitszeiteinträge.

User.java: Datenmodell für Benutzerinformationen.

UserStatus.java: Datenmodell für Benutzerstatus.

▪ Services

Services enthalten die Geschäftslogik der Anwendung und kommunizieren mit den Repositories zur Datenverarbeitung.

auth-header.js: Fügt Autorisierungsheader zu API-Anfragen hinzu.

getUserRole.js: Überprüft die Rolle des angemeldeten Benutzers.

LoginServices.js: Handhabt die Login-Operationen.

TimeEntriesService.js: Handhabt API-Aufrufe im Zusammenhang mit der Zeiterfassung.

UserService.js: Handhabt allgemeine Benutzeroperationen wie das Abrufen von Benutzerinformationen.

▪ Sicherheit

Das Sicherheitsmodul der Anwendung ist ein zentraler Bestandteil, um den Schutz sensibler Daten und die Zugriffskontrolle zu gewährleisten. Es verwendet JSON Web Tokens (JWT) für die Authentifizierung und Autorisierung, um sicherzustellen, dass nur autorisierte Benut-

zer auf bestimmte Ressourcen zugreifen können. Das Sicherheitsmodul ist in verschiedene Pakete und Klassen unterteilt, die jeweils spezifische Sicherheitsfunktionen erfüllen.

➤ **JWT (JSON Web Tokens)**

AuthEntryPointJwt.java: Diese Klasse handelt unautorisierte Zugriffsversuche und gibt entsprechende Fehlerantworten zurück.

AuthTokenFilter.java: Filtert eingehende Anfragen und überprüft die Gültigkeit des JWT-Tokens.

GenerateToken.java: Generiert JWT-Tokens für authentifizierte Benutzer.

JwtUtils.java: Dienstprogrammklasse zum Verarbeiten und Analysieren von JWT-Tokens.

➤ **SecurityService**

CrossSecurity.java: Konfiguriert CORS (Cross-Origin Resource Sharing) für die Anwendung.

UserDetailsImpl.java: Implementiert die Benutzerdetails für die Authentifizierung.

UserDetailsService.java: Schnittstelle für die Bereitstellung von Benutzerdetails aus der Datenquelle.

UserDetailsServiceImpl.java: Implementierung der Benutzerdetails-Service-Schnittstelle.

WebSecurityConfig.java: Konfiguriert die Sicherheitsaspekte der Anwendung, einschließlich Authentifizierung und Autorisierung, sowie die Sicherheitsfilterkette.

▪ **Anwendungseintrittspunkt**

BLinioTimerApplication.java: Dies ist die Hauptklasse der Spring Boot-Anwendung und dient als Einstiegspunkt, um die Anwendung zu starten.

6. Zusammenfassung und Ausblick

Dieser Abschnitt fasst das Ergebnis der Entwicklung der Anwendung zur Arbeitszeiterfassung zusammen und gibt einen Ausblick auf kommende Entwicklungen und den Bedarf an Forschung. Ziel ist es, die wichtigsten Erkenntnisse und Erfolge des Projekts zu beleuchten und mögliche Weiterentwicklungen und Optimierungen aufzuzeigen.

6.1 Zusammenfassung der Ergebnisse

Das Ziel der Erstellung der Zeiterfassungsanwendung war es, eine nutzerfreundliche und effektive Lösung für die Erfassung und Verwaltung von Arbeitszeiten zu entwickeln. Hier sind die wichtigsten Ergebnisse:

Die Auswahl der Technologie: Die Wahl von React im Frontend und Java mit Spring Boot im Backend gewährleistete eine stabile, skalierbare und sichere Anwendung.

Design und Prototypen: Die Optimierung des Benutzererlebnisses wurde durch ein nutzerfreundliches Interface-Design und die Erstellung von Prototypen unterstützt.

Umsetzung: Durch den Einsatz moderner Technologien wird eine reibungslose Funktionalität und Interaktion zwischen Frontend und Backend sichergestellt.

Qualitätssicherung: Umfassende Prüfungen sorgten dafür, dass die Anwendung zuverlässig und leistungsfähig ist.

Die Anwendung erlaubt den Nutzern eine genaue Erfassung von Arbeitszeiten und die Erstellung von Berichten, was zu einer gesteigerten Effizienz und Präzision bei der Erfassung von Arbeitszeiten führt.

6.2 Ausblick auf zukünftige Entwicklungen und Forschungsbedarf

Auch wenn die bereits entwickelte Anwendung viele Vorzüge mit sich bringt, besteht nach wie vor Bedarf an Verbesserungen und Erweiterungen. Möglicherweise könnten zukünftige Entwicklungen folgendes beinhalten:

Erweiterte Berichtsfunktionen: Die Implementierung von ausführlicheren Berichten und Analysewerkzeugen könnte Unternehmen dabei unterstützen, noch genauere Informationen über die Arbeitszeiten und die Produktivität ihrer Mitarbeiter zu erhalten.

Entwicklung einer mobilen App: Mit der Erstellung einer mobilen Version der Anwendung könnten die Nutzer ihre Arbeitszeiten von jedem Ort aus erfassen und verwalten.

Der Einsatz von KI-Technologien könnte dazu beitragen, Arbeitsmuster vorherzusagen und Abweichungen in den erfassten Zeiten automatisch zu erkennen.¹³

Anpassung an gesetzliche Änderungen: Die Anwendung sollte regelmäßig aktualisiert werden, um sicherzustellen, dass sie den neuesten gesetzlichen Anforderungen in Bezug auf Arbeitszeiten und Datenschutz entspricht.¹⁴

Benutzerfeedback und kontinuierliche Verbesserung: Durch regelmäßiges Einholen von Benutzerfeedback können kontinuierliche Verbesserungen und Anpassungen vorgenommen werden, um die Benutzerfreundlichkeit weiter zu steigern.

Die entwickelte Arbeitszeiterfassungsanwendung stellt eine stabile Grundlage für eine effektive Arbeitszeitverwaltung dar. Zukünftige Fortschritte und Forschungen werden einen Beitrag zur weiteren Verbesserung der Funktionalität und der Benutzererfahrung leisten und sie den sich wandelnden Benutzeranforderungen anpassen.

¹³ Guckert, M. & Taentzer, G. (2015). PLATTFORMUNABHÄNGIGE ENTWICKLUNG MOBILER ANWENDUNGEN MIT AUGMENTED REALITY-FUNKTIONALITÄT. *Anwendungen und Konzepte der Wirtschaftsinformatik*, 3, 5. <https://doi.org/10.26034/lu.akwi.2015.3161>

¹⁴ Hoff, A. (2021). Arbeitszeit-Aufzeichnung. In *Essentials* (S. 35–36). https://doi.org/10.1007/978-3-658-34424-5_8

Literaturverzeichnis

Hellert, U. (2018). Arbeitszeitmodelle der Zukunft - inkl. Arbeitshilfen online. <https://doi.org/10.34157/9783648119969>

Seobility. (o. D.). Frontend (vs. Backend) - Was ist das? - Seobility Wiki. <https://www.seobility.net/de/wiki/Frontend>

Akademie, D. (2024, 26. Juni). HTML-Grundlagen im Überblick. Developer Akademie. <https://developerakademie.com/blog/all/html-grundlagen-im-ueberblick>

Redaktion, I. (2022, 21. Juni). Was ist CSS? Definition und Anwendung. IONOS Digital Guide. <https://www.ionos.de/digitalguide/websites/webdesign/was-ist-css>

Was ist JavaScript? – JavaScript (JS) erklärt – AWS. (o. D.). Amazon Web Services, Inc. <https://aws.amazon.com/de/what-is/javascript>

Wikipedia-Autoren. (2016, 6. September). React. <https://de.wikipedia.org/wiki/React>

IT-SERVICE.NETWORK. (2024, 15. März). Backend. <https://it-service.network/it-lexikon/backend>

Was ist Java? | IBM. (o. D.). <https://www.ibm.com/de-de/topics/java>

What is Java Spring Boot?—Intro to Spring Boot | Microsoft Azure. (o. D.). <https://azure.microsoft.com/en-gb/resources/cloud-computing-dictionary/what-is-java-spring-boot>

<https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.htm>

Steubel, P. (2024, 3. März). Was ist ein Use-Case? Definition und Nutzen im Überblick! [2024] • Asana. Asana. <https://asana.com/de/resources/what-is-a-use-case>

[GDI Webterminal: Webbasierte Zeiterfassung leicht gemacht](#)

Guckert, M. & Taentzer, G. (2015). PLATTFORMUNABHÄNGIGE ENTWICKLUNG MOBILER ANWENDUNGEN MIT AUGMENTED REALITY-FUNKTIONALITÄT. *Anwendungen und Konzepte der Wirtschaftsinformatik*, 3, 5. <https://doi.org/10.26034/lu.akwi.2015.3161>

Hoff, A. (2021). Arbeitszeit-Aufzeichnung. In *Essentials* (S. 35–36). https://doi.org/10.1007/978-3-658-34424-5_8

Anhang

1. Login.jsx

```
JS PageHeader.jsx JS PageHomeTerminal.jsx M README.md M JS Login.jsx X JS Employee.jsx JS TimeEntriesService.js U ...
TimerClient > src > Components > JS Login.jsx > Login > handleSubmit
7 function Login() {
21   const handleSubmit = async (event) => {
29     const userRole = getUserRole(jwtToken);
30
31     if (Array.isArray(userRole) && userRole.length > 0) {
32       // Extrahiere die erste Rolle
33       const firstRole = userRole[0];
34
35       // Fügen Sie den Token dem Authorization-Header für zukünftige Anfragen hinzu
36       axios.defaults.headers.common['Authorization'] = `Bearer ${jwtToken}`;
37
38       // Speichern Sie das JWT-Token im localStorage
39       localStorage.setItem('accessToken', jwtToken);
40
41       // Weiterleiten zur richtigen Dashboard-Seite basierend auf der Rolle
42       if (firstRole === 'ADMIN') {
43         navigate('/admin');
44       } else if (firstRole === 'EMPLOYEE') {
45         navigate('/employee');
46       } else if (firstRole === 'TEAMLEADER') {
47         navigate('/teamleader');
48       } else {
49         alert('Unbekannte Benutzerrolle');
50       }
51     } else {
52       alert('Benutzerrolle nicht gefunden');
53     }
54   } else {
55     setLoginError('Passwort oder Benutzername ist falsch');
56   }
57 }
58
59 catch (error) {
```

```
JS PageHeader.jsx JS PageHomeTerminal.jsx M README.md M JS Login.jsx X JS Employee.jsx JS TimeEntriesService.js U ...
TimerClient > src > Components > JS Login.jsx > Login > handleSubmit
7 function Login() {
65   return (
66     <div className="d-flex justify-content-center align-items-center vh-100">
67       <div className="card">
68         <div className="card-body">
69           <h6 className="card-title text-center">Geben Sie bitte Ihre B_Linio Authentifizierungsdaten ein
70           <form onSubmit={handleSubmit}>
71             {loginError} && <div className="alert alert-danger">{loginError}</div> { /* Anzeige von Fehler
72             <div className="mb-3">
73               <label htmlFor="exampleInputEmail1" className="form-label">
74                 User Name
75               </label>
76               <input
77                 type="text"
78                 className="form-control"
79                 id="exampleInputEmail1"
80                 aria-describedby="emailHelp"
81                 onChange={handleUsernameChange}
82               />
83               <div id="emailHelp" className="form-text">
84                 We'll never share your email with anyone else.
85               </div>
86             </div>
87             <div className="mb-3">
88               <label htmlFor="exampleInputPassword1" className="form-label">
89                 Password
90               </label>
91               <input
92                 type="password"
93                 className="form-control"
94                 id="exampleInputPassword1"
95                 onChange={handlePasswordChange}
96               />
97             </div>
98           </form>
99         </div>
100       </div>
101     </div>
102   );
103 }
```

2. pageHomeTerminal.jsx

```

152 return (
153   <>
154     <div className='home'>
155       <div className='terminal'>
156         <div className='date-time'>
157           <h4>Webterminal</h4>
158           <p>{currentTime.toLocaleTimeString()}</p>
159         </div>
160
161         <form action="/action_zeitsArt">
162           <p>zeitsArt</p>
163           <select id="zeitsArt" name="zeitsArt">
164             <option value="Anwesend">Anwesend</option>
165             <option value="Abwesend">Abwesend</option>
166           </select>
167         </form>
168
169         <div className='timeEntriesSection'>
170           <div className='comments'>
171             <textarea
172               className='textarea-comment'
173               placeholder="Kommentare"
174               value={comment}
175               onChange={(e) => setComment(e.target.value)}
176             />
177           </div>
178
179           <div className='comeGobtn'>
180             <button className='comeBtn' onClick={handleKommenClick}>Kommen</button>
181             <button className='goBtn' onClick={handleGoClick}>Gehen</button>
182           </div>

```

```

183     <div><p>{currentTime.toLocaleDateString(undefined, options)}</p></div>
184   </div>
185 </div>
186
187 <div className='tagesÜbersicht'>
188   <div className="date-time">
189     <h4>Tagesübersicht</h4>
190     <p>{currentTime.toLocaleDateString(undefined, options)}</p>
191   </div>
192   <div className='tableForTagesÜbersicht'>
193     <table>
194       <thead>
195         <tr>
196           <th>Datum</th>
197           <th>Von</th>
198           <th>Bis</th>
199           <th>Kommentare</th>
200           <th>Dauer</th>
201         </tr>
202       </thead>
203       <tbody>
204         {timeEntries.map((entry, index) => (
205           <tr key={index}>
206             <td>{entry.date}</td>
207             <td>{entry.from}</td>
208             <td>{entry.to}</td>
209             <td>{entry.comment}</td>
210             <td>{entry.duration}</td>
211           </tr>
212         ))}
213       </tbody>
214     </table>

```


3. PageSidebar.jsx

```
70   return (  
71     <aside id="sidebar" className={openSidebarToggle ? 'sidebar-responsive' : ''}>  
72       <div className='sidebar-title'>  
73         <div className='sidebar-brand'>  
74           <BsPersonCircle className='icon_header' />  
75           <p className='sub'>{sub}</p>  
76           <p className='employeeNbr'>{employeeNbr}</p>  
77         </div>  
78         <span className='icon close_icon' onClick={OpenSidebar}>X</span>  
79       </div>  
80  
81       <ul className='sidebar-list'>  
82         <li className={`sidebar-list-item ${selectedMenuItem === 'Terminal' ? 'selected' : ''}`}>  
83           <Link to="/pageHome" onClick={() => handleMenuItemClick('Terminal')}>  
84             <BsGrid1X2Fill className='icon' /> Terminal  
85           </Link>  
86         </li>  
87         <li className={`sidebar-list-item ${selectedMenuItem === 'Historie' ? 'selected' : ''}`}>  
88           <Link to="/historie" onClick={() => handleMenuItemClick('Historie')}>  
89             <BsClockHistory className='icon' /> Historie  
90           </Link>  
91         </li>  
92         <li className={`sidebar-list-item ${selectedMenuItem === 'Teamauskunft' ? 'selected' : ''}`}>  
93           <Link to="/teamauskunft" onClick={() => handleMenuItemClick('Teamauskunft')}>  
94             <BsPeopleFill className='icon' /> Teamauskunft  
95           </Link>  
96         </li>  
97         <li className={`sidebar-list-item ${selectedMenuItem === 'Anträge' ? 'selected' : ''}`}>  
98           <Link to="/antraege" onClick={() => handleMenuItemClick('Anträge')}>  
99             <BsFillGrid3X3GapFill className='icon' /> Anträge  
100          </Link>  
101        </li>  
102      </ul>  
103    </aside>  
104  )  
105 }
```

4 Time_EntriesController.java

```
76
77
78 @PreAuthorize("hasAnyAuthority('ADMIN', 'TEAMLEADER', 'EMPLOYEE')")
79 @GetMapping("/getentries")
80 public ResponseEntity<List<TimeEntriesProjection>> getTimeEntries() {
81     try {
82
83         Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
84
85         if (authentication != null && authentication.getPrincipal() instanceof UserDetailsImpl) {
86             UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
87             Integer employeeNbr = userDetails.getEmployeeNbr();
88
89
90             if (employeeNbr != null) {
91                 List<TimeEntriesProjection> entries = time_EntriesService.getTimeEntriesByEmployeeNbr(emp
92                 return ResponseEntity.ok(entries);
93             } else {
94
95                 return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
96             }
97         } else {
98
99             return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
100         }
101     } catch (Exception e) {
102
103         return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
104     }
105 }
106
```

5 Time_EntriesServiceImpl.js

```
24 @Service
25 public class Time_EntriesServiceImpl implements Time_EntriesService {
26
27     private final Time_EntriesRepository timeEntriesRepository;
28     private final UserRepository userRepository;
29
30     public Time_EntriesServiceImpl(Time_EntriesRepository timeEntriesRepository, UserRepository userRepos
31         this.timeEntriesRepository = timeEntriesRepository;
32         this.userRepository = userRepository;
33     }
34
35     @Override
36     public Time_EntriesDtoComeTime createEntry(Time_EntriesDtoComeTime time_EntriesDtoComeTime) {
37         Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
38         if (authentication != null && authentication.getPrincipal() instanceof UserDetailsImpl) {
39             UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
40             Integer employeeNbr = userDetails.getEmployeeNbr();
41
42             List<Time_Entries> openComeTimes = timeEntriesRepository.findOpenEntriesByEmployee(employeeNb
43             if (!openComeTimes.isEmpty()) {
44                 throw new IllegalStateException(s:"Sie haben bereits eine offene 'Kommen'-Zeit. Bitte sch
45             }
46
47             Optional<User> userOptional = userRepository.findByEmployeeNbr(employeeNbr);
48             if (userOptional.isPresent()) {

```

```
45     }
46
47     Optional<User> userOptional = userRepository.findByEmployeeNbr(employeeNbr);
48     if (userOptional.isPresent()) {
49         User user = userOptional.get();
50         Time_Entries newEntry = new Time_Entries();
51         newEntry.setEmployeeNbr(user);
52         newEntry.setComment(time_EntriesDtoComeTime.getComment());
53         newEntry.setEntriesDateTime(java.sql.Date.valueOf(LocalDate.now()));
54         newEntry.setStartTime(LocalDateTime.now());
55
56         Time_Entries savedEntry = timeEntriesRepository.save(newEntry);
57         return convertToDto(savedEntry);
58     } else {
59         throw new UserNotFoundException(message:"Benutzer nicht gefunden");
60     }
61 } else {
62     throw new UnauthorizedException(message:"Unauthorized");
63 }
64 }
65
66 @Override
67 public Time_EntriesDtoGoTime updateGoTimeAndCalculateDuration() {
68     Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
69     if (authentication != null && authentication.getPrincipal() instanceof UserDetailsImpl) {
70         UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
```

```
71         Integer employeeNbr = userDetails.getEmployeeNbr();
72
73         List<Time_Entries> openComeTimes = timeEntriesRepository.findOpenEntriesByEmployee(employeeNbr);
74         if (openComeTimes.isEmpty()) {
75             throw new IllegalStateException(s:"Sie haben keine offene 'Kommen'-Zeit. Bitte erstellen");
76         }
77
78         Time_Entries openEntry = openComeTimes.get(index:0);
79         LocalDateTime comeTime = openEntry.getStartTime();
80         LocalDateTime goTime = LocalDateTime.now();
81         double duration = calculateDurationInHoursAndMinutes(comeTime, goTime);
82
83         openEntry.setEndTime(goTime);
84         openEntry.setDuration(duration);
85         timeEntriesRepository.save(openEntry);
86
87         Time_EntriesDtoGoTime dto = new Time_EntriesDtoGoTime();
88         dto.setEmployeeNbr(employeeNbr);
89         dto.setEndTime(goTime);
90         dto.setDuration(duration);
91         return dto;
92     } else {
93         throw new UnauthorizedException(message:"Unauthorized");
94     }
95 }
96 }
```

```

102     @Override
103     public List<TimeEntriesProjection> getTimeEntriesByEmployeeNbr(Integer employeeNbr) {
104         List<Object[]> timeEntriesDataList = timeEntriesRepository.findTimeEntriesByEmployeeNbr(employeeNbr);
105         return mapObjectArrayToTimeEntriesProjection(timeEntriesDataList);
106     }
107
108     private List<TimeEntriesProjection> mapObjectArrayToTimeEntriesProjection(List<Object[]> timeEntriesDataList) {
109         return timeEntriesDataList.stream()
110             .map(this::mapObjectArrayToTimeEntriesProjection)
111             .collect(Collectors.toList());
112     }
113
114     private TimeEntriesProjection mapObjectArrayToTimeEntriesProjection(Object[] timeEntriesData) {
115         TimeEntriesProjection timeEntriesProjection = new TimeEntriesProjection();
116         timeEntriesProjection.setEntriesDateTime((Date) timeEntriesData[0]);
117         timeEntriesProjection.setStartTime((LocalDateTime) timeEntriesData[1]);
118         timeEntriesProjection.setEndTime((LocalDateTime) timeEntriesData[2]);
119         timeEntriesProjection.setDuration((double) timeEntriesData[3]);
120         return timeEntriesProjection;
121     }
122
123     private Time_EntriesDtoComeTime convertToDto(Time_Entries entry) {
124         Time_EntriesDtoComeTime dto = new Time_EntriesDtoComeTime();
125         dto.setEmployeeNbr(entry.getEmployeeNbr().getEmployeeNbr());
126         dto.setComment(entry.getComment());
127         dto.setStarTime(entry.getStartTime());
128         return dto;

```

6. AuthEntryPointJwt.java

```

19
20 @Component
21 public class AuthEntryPointJwt implements AuthenticationEntryPoint {
22
23     private static final Logger logger = LoggerFactory.getLogger(clazz:AuthEntryPointJwt.class);
24
25     @Override
26     public void commence(HttpServletRequest request, HttpServletResponse response, AuthenticationException
27         throws IOException, ServletException {
28         logger.error(format:"Unauthorized error: {}", authException.getMessage());
29
30         response.setContentType(MediaType.APPLICATION_JSON_VALUE);
31         response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
32
33         final Map<String, Object> body = new HashMap<>();
34         body.put(key:"status", HttpServletResponse.SC_UNAUTHORIZED);
35         body.put(key:"error", value:"Unauthorized");
36         body.put(key:"message", authException.getMessage());
37         body.put(key:"path", request.getServletPath());
38
39         final ObjectMapper mapper = new ObjectMapper();
40         mapper.writeValue(response.getOutputStream(), body);
41     }
42
43 }
44

```

7. AuthTokenFilter.java

```
22 @Component
23 public class AuthTokenFilter extends OncePerRequestFilter {
24
25     @Autowired
26     private JwtUtils jwtUtils;
27
28     @Autowired
29     private UserDetailsServiceImpl userDetailsService;
30
31     private static final Logger logger = LoggerFactory.getLogger(clazz:AuthTokenFilter.class);
32
33     @Override
34     protected void doFilterInternal(@SuppressWarnings("null") HttpServletRequest request, @SuppressWarnings
35         throws ServletException, IOException {
36         try {
37             String jwt = parseJwt(request);
38             if (jwt != null && jwtUtils.validateJwtToken(jwt)) {
39                 String username = jwtUtils.getUserNameFromJwtToken(jwt);
40
41                 UserDetails userDetails = userDetailsService.loadUserByUsername(username);
42                 UsernamePasswordAuthenticationToken authentication =
43                     new UsernamePasswordAuthenticationToken(
44                         userDetails,
45                         credentials:null,
46                         userDetails.getAuthorities());
47                 authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
48
49                 SecurityContextHolder.getContext().setAuthentication(authentication);
50
51                 SecurityContextHolder.getContext().setAuthentication(authentication);
52             }
53         } catch (Exception e) {
54             logger.error(msg:"Cannot set user authentication: {}", e);
55         }
56         filterChain.doFilter(request, response);
57     }
58
59     private String parseJwt(HttpServletRequest request) {
60         String headerAuth = request.getHeader(name:"Authorization");
61
62         if (StringUtils.hasText(headerAuth) && headerAuth.startsWith(prefix:"Bearer ")) {
63             return headerAuth.substring(beginIndex:7);
64         }
65
66         return null;
67     }
68 }
```

8. JwtUtils.java

```
26 @Component
27 public class JwtUtils {
28
29     private static final Logger logger = LoggerFactory.getLogger(clazz:JwtUtils.class);
30
31     private Set<String> invalidTokens = new HashSet<>(); // Set zur Verfolgung ungültiger Tokens
32
33     @Value("${BLinioTimer.app.jwtSecret}")
34     private String jwtSecret;
35
36     @Value("${BLinioTimer.app.jwtExpirationMs}")
37     private int jwtExpirationMs;
38
39     public String generateJwtToken(Authentication authentication, Integer employeeNbr) {
40         UserDetailsImpl userPrincipal = (UserDetailsImpl) authentication.getPrincipal();
41         List<String> roles = userPrincipal.getAuthorities().stream()
42             .map(item -> item.getAuthority())
43             .collect(Collectors.toList());
44
45         // Füge die Rolle zu den Claims hinzu
46         Claims claims = Jwts.claims().setSubject(userPrincipal.getUsername());
47         claims.put(key:"roles", roles);
48         claims.put(key:"employeeNbr", employeeNbr);
49
50         return Jwts.builder()
51             .setClaims(claims) // Setze die Claims mit der Rolle
52             .setIssuedAt(new Date())
53             .setExpiration(new Date(System.currentTimeMillis() + jwtExpirationMs))
```

```
54             .signWith(key(), SignatureAlgorithm.HS256)
55             .compact();
56     }
57
58     private Key key() {
59         return Keys.hmacShaKeyFor(Decoders.BASE64.decode(jwtSecret));
60     }
61
62     public String getUserNbrFromJwtToken(String token) {
63         return Jwts.parserBuilder().setSigningKey(key()).build()
64             .parseClaimsJws(token).getBody().getSubject();
65     }
66
67     public boolean validateJwtToken(String authToken) {
68         try {
69             // Überprüfen, ob das Token ungültig ist
70             if (isTokenInvalid(authToken)) {
71                 logger.error(msg:"Invalid JWT token: Token is invalidated");
72                 return false;
73             }
74
75             Jwts.parserBuilder().setSigningKey(key()).build().parse(authToken);
76             return true;
77         } catch (MalformedJwtException e) {
78             logger.error(format:"Invalid JWT token: {}", e.getMessage());
79         } catch (ExpiredJwtException e) {
80             logger.error(format:"JWT token is expired: {}", e.getMessage());
```

```

80     logger.error(format:"JWT token is expired: {}", e.getMessage());
81 } catch (UnsupportedJwtException e) {
82     logger.error(format:"JWT token is unsupported: {}", e.getMessage());
83 } catch (IllegalArgumentException e) {
84     logger.error(format:"JWT claims string is empty: {}", e.getMessage());
85 }
86
87     return false;
88 }
89
90
91 // Methode zum Überprüfen, ob ein Token ungültig ist
92 private boolean isTokenInvalid(String token) {
93     return invalidTokens.contains(token);
94 }
95
96 // Methode zum Ungültigmachen eines Tokens beim Ausloggen
97 public void invalidateToken(String token) {
98     // Fügen Sie das Token zur Liste der ungültigen Tokens hinzu
99     invalidTokens.add(token);
100 }
101
102 }
103
104

```

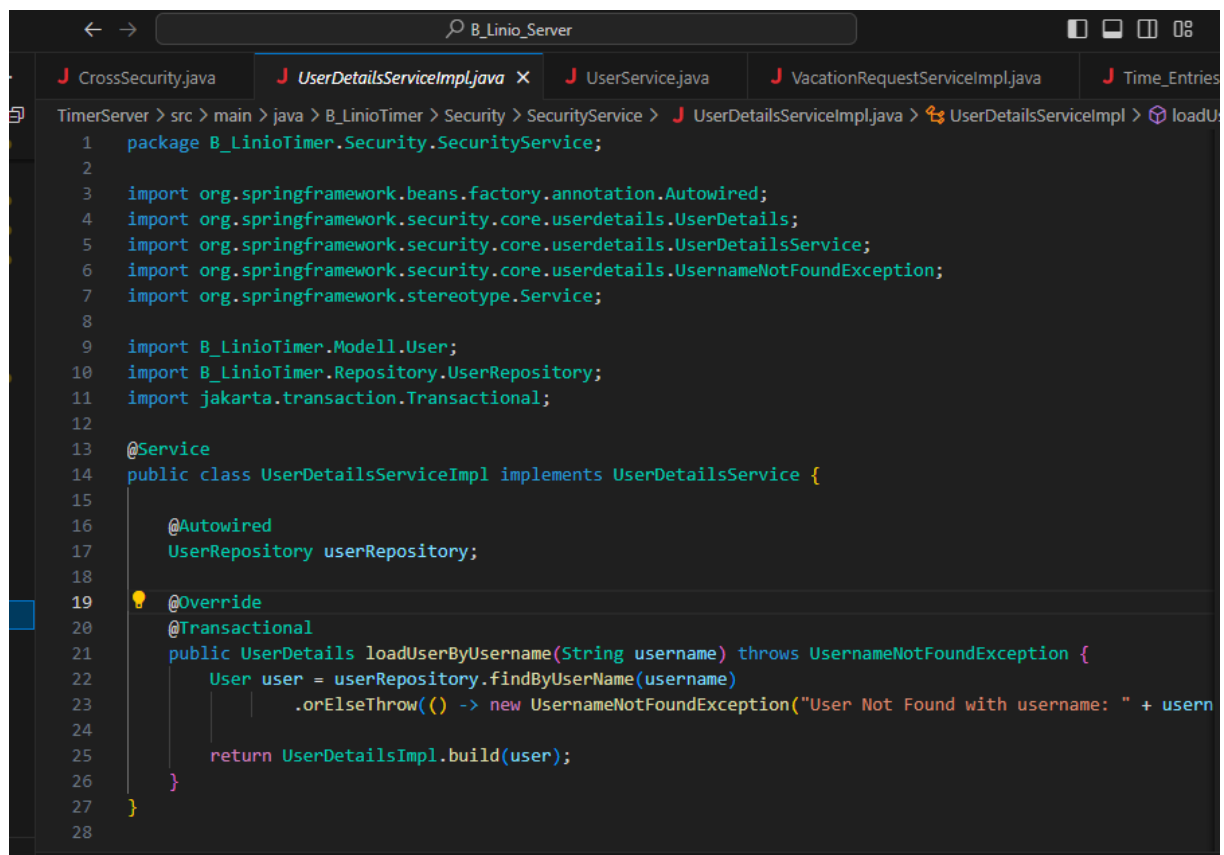
9. CrossSecurity.java

```

CrossSecurity.java X UserService.java VacationRequestServiceImpl.java Time_EntriesController.java UserControlle
TimerServer > src > main > java > B_LinioTimer > Security > SecurityService > CrossSecurity.java > CrossSecurity > corsConfigurer() > new
8  @Configuration
9  public class CrossSecurity {
10
11      @Bean
12      public WebMvcConfigurer corsConfigurer() {
13          return new WebMvcConfigurer() {
14              @SuppressWarnings("null")
15              @Override
16              public void addCorsMappings(CorsRegistry registry) {
17                  registry.addMapping(pathPattern:"/api/auth/loginUser")
18                      .allowedOrigins(...origins:"http://localhost:3000")
19                      .allowedMethods(...methods:"GET", "POST", "PUT", "DELETE")
20                      .allowCredentials(allowCredentials:false);
21
22                  registry.addMapping(pathPattern:"/api/auth/logout")
23                      .allowedOrigins(...origins:"http://localhost:3000")
24                      .allowedMethods(...methods:"GET", "POST", "PUT", "DELETE")
25                      .allowCredentials(allowCredentials:false);
26
27                  registry.addMapping(pathPattern:"/register")
28                      .allowedOrigins(...origins:"http://localhost:3000")
29                      .allowedMethods(...methods:"GET", "POST", "PUT", "DELETE")
30                      .allowCredentials(allowCredentials:false);
31
32                  registry.addMapping(pathPattern:"/comingTime")
33                      .allowedOrigins(...origins:"http://localhost:3000")
34                      .allowedMethods(...methods:"GET", "POST", "PUT", "DELETE")
35                      .allowCredentials(allowCredentials:false);

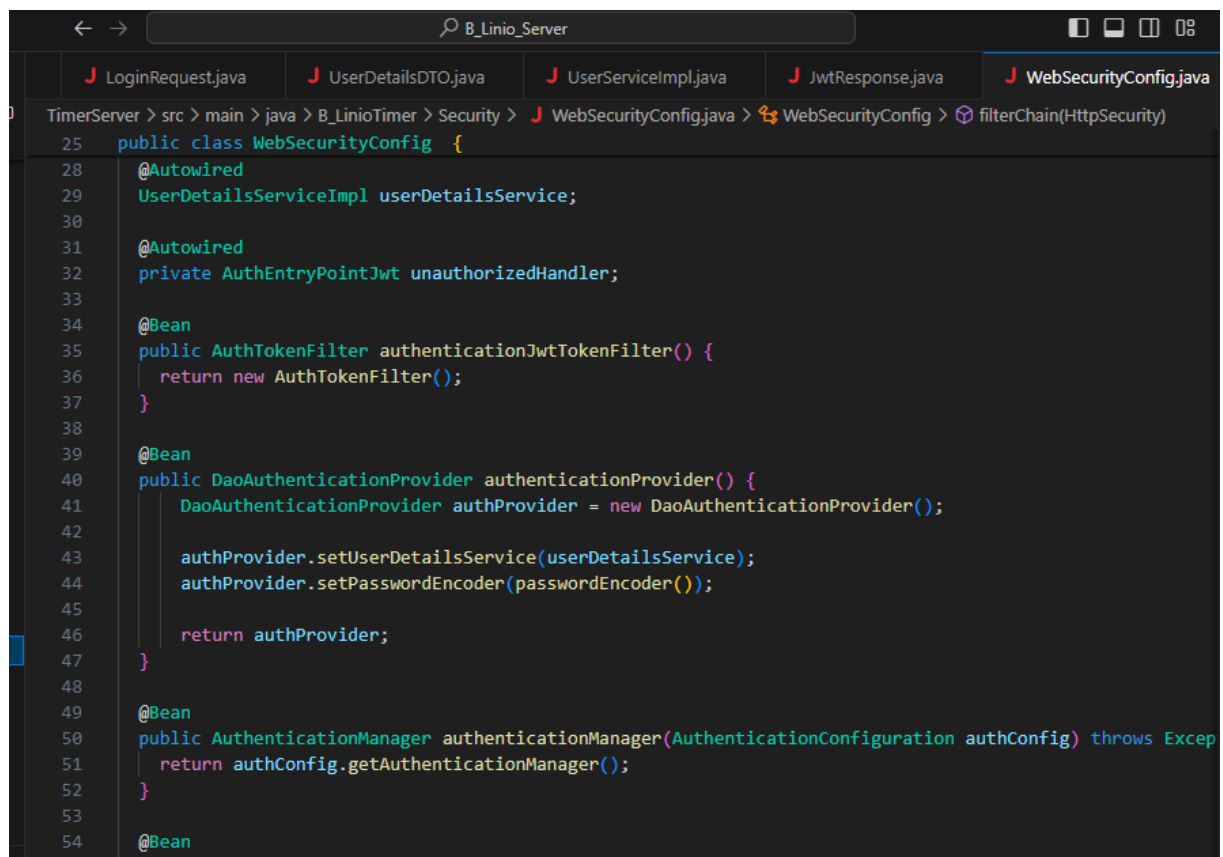
```

10. UserDetailsServiceImpl.java

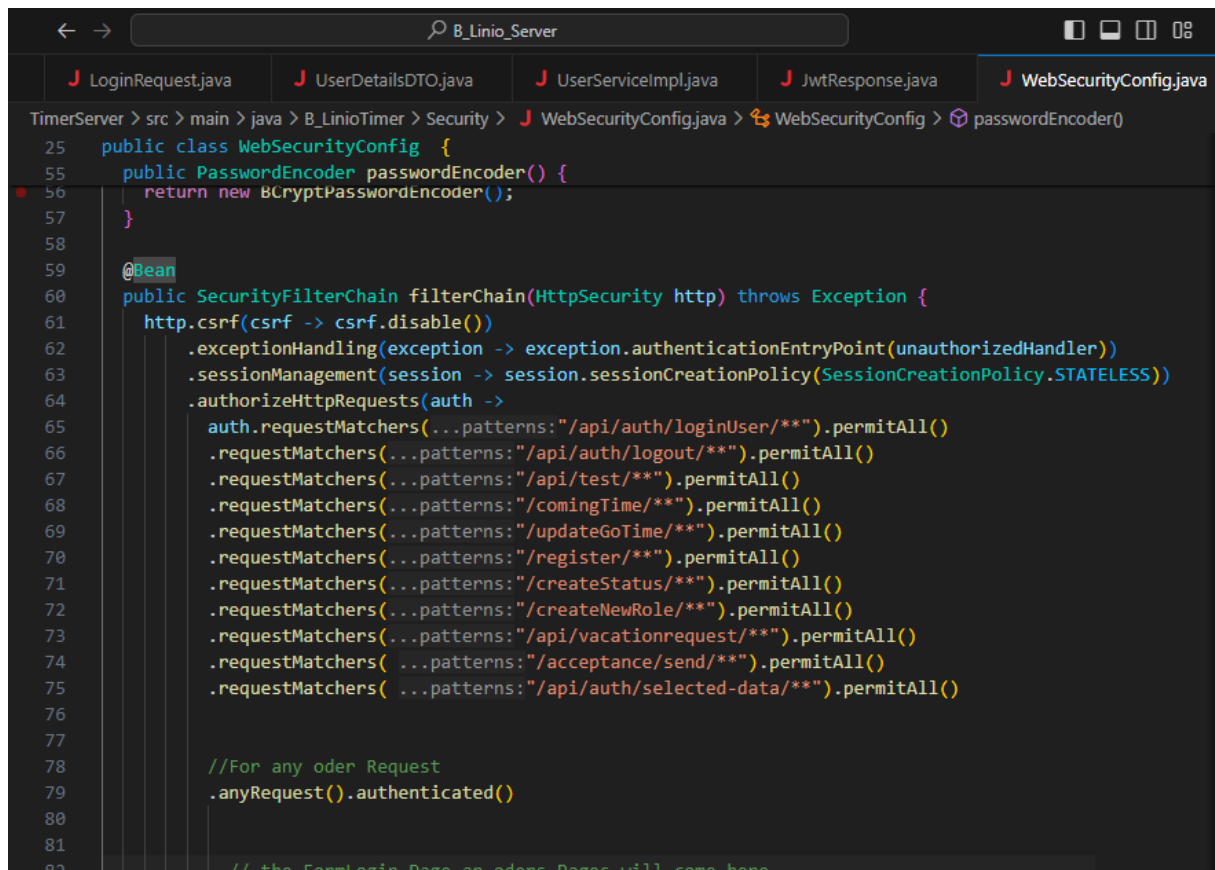


```
1 package B_LinioTimer.Security.SecurityService;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.security.core.userdetails.UserDetails;
5 import org.springframework.security.core.userdetails.UserDetailsService;
6 import org.springframework.security.core.userdetails.UsernameNotFoundException;
7 import org.springframework.stereotype.Service;
8
9 import B_LinioTimer.Modell.User;
10 import B_LinioTimer.Repository.UserRepository;
11 import jakarta.transaction.Transactional;
12
13 @Service
14 public class UserDetailsServiceImpl implements UserDetailsService {
15
16     @Autowired
17     UserRepository userRepository;
18
19     @Override
20     @Transactional
21     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
22         User user = userRepository.findByUserName(username)
23             .orElseThrow(() -> new UsernameNotFoundException("User Not Found with username: " + usern
24
25         return UserDetailsImpl.build(user);
26     }
27 }
28
```


11. WebSecurityConfig.java



```
TimerServer > src > main > java > B_LinioTimer > Security > WebSecurityConfig.java > WebSecurityConfig > filterChain(HttpSecurity)
25 public class WebSecurityConfig {
28     @Autowired
29     UserDetailsServiceImpl userDetailsService;
30
31     @Autowired
32     private AuthEntryPointJwt unauthorizedHandler;
33
34     @Bean
35     public AuthTokenFilter authenticationJwtTokenFilter() {
36         return new AuthTokenFilter();
37     }
38
39     @Bean
40     public DaoAuthenticationProvider authenticationProvider() {
41         DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
42
43         authProvider.setUserDetailsService(userDetailsService);
44         authProvider.setPasswordEncoder(passwordEncoder());
45
46         return authProvider;
47     }
48
49     @Bean
50     public AuthenticationManager authenticationManager(AuthenticationConfiguration authConfig) throws Exception {
51         return authConfig.getAuthenticationManager();
52     }
53
54     @Bean
```



```
TimerServer > src > main > java > B_LinioTimer > Security > WebSecurityConfig.java > WebSecurityConfig > passwordEncoder()
25 public class WebSecurityConfig {
55     public PasswordEncoder passwordEncoder() {
56         return new BCryptPasswordEncoder();
57     }
58
59     @Bean
60     public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
61         http.csrf(csrf -> csrf.disable())
62             .exceptionHandling(exception -> exception.authenticationEntryPoint(unauthorizedHandler))
63             .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
64             .authorizeHttpRequests(auth ->
65                 auth.requestMatchers(...patterns: "/api/auth/loginUser/**").permitAll()
66                 .requestMatchers(...patterns: "/api/auth/logout/**").permitAll()
67                 .requestMatchers(...patterns: "/api/test/**").permitAll()
68                 .requestMatchers(...patterns: "/comingTime/**").permitAll()
69                 .requestMatchers(...patterns: "/updateGoTime/**").permitAll()
70                 .requestMatchers(...patterns: "/register/**").permitAll()
71                 .requestMatchers(...patterns: "/createStatus/**").permitAll()
72                 .requestMatchers(...patterns: "/createNewRole/**").permitAll()
73                 .requestMatchers(...patterns: "/api/vacationrequest/**").permitAll()
74                 .requestMatchers(...patterns: "/acceptance/send/**").permitAll()
75                 .requestMatchers(...patterns: "/api/auth/selected-data/**").permitAll()
76
77
78                 //For any oder Request
79                 .anyRequest().authenticated()
80
81
82                 // the Forbidden Page or oder Page will come here
83             );
84     }
85 }
```

Eidesstattliche Erklärung

Ich versichere, dass ich diese Arbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel verwendet und alle wörtlichen oder sinngemäßen Entlehnungen deutlich, als solche gekennzeichnet habe.

Zweibrücken, 16.07.2024

Billy Max Tsane Tsafack

Ort, Datum

Vor- und Nachname

