HOCHSCHULE
ESSLINGEN

Facharbeit

Progressive Web App für Studierende: Eine App zur Verwaltung von Vorlesungen, Notizen und Prüfungen

im Studiengang
Softwaretechnik und Medieninformatik (B. Eng)

vorgelegt von

Esra Balci
Matr.-Nr.: 766561
E-Mail: esbait01@hs-esslingen.de
am 24.06.2025
an der Hochschule Esslingen

Prüferin: Prof. Dipl.-Informatikerin Astrid Beck

Zweitprüfer: Prof. Dr.-Ing. Harald Melcher

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation und Zielsetzung

In einer Zeit, in der das digitale Studium stetig an Bedeutung gewinnt, stehen Studierende vor der Herausforderung, eine Vielzahl an Terminen und Materialien effizient zu verwalten. Handschriftliche Notizen sowie verstreute Kalendereinträge verursachen häufig Unübersichtlichkeit und Stress.
Mit der zunehmenden Digitalisierung steigt der Bedarf an eigenverantwortlicher Organisation des Lernalltags. Studien zeigen, dass Selbstorganisations- und Motivationsfähigkeiten zentrale Voraussetzungen für erfolgreiches Arbeiten und Lernen sind - insbesondere in agilen und digitalen Umgebungen (Busse u. a. 2022). Gerade für Studierende bedeutet das, geeignete Werkzeuge zu nutzen, um Termine, Inhalte und Notizen effizient zu strukturieren. Die im Rahmen dieser Arbeit entwickelte Web App adressiert genau diese Bedürfnisse.

Ziel ist es, eine Progressive Web App (PWA) zu entwickeln, die nicht nur Vorlesungspläne und Termine in einer übersichtlichen Oberfläche bündelt, sondern auch das Anlegen und Wiederfinden persönlicher Notizen erleichtert. Dabei stehen eine gebrauchstaugliche Bedienung, Offline-Funktionalität sowie eine nahtlose Synchronisierung über verschiedene Endgeräte hinweg im Fokus. Die Anwendung soll Studierende dabei unterstützen, ihren Studienalltag effizienter zu strukturieren und sich auf das Wesentliche - das Lernen - zu konzentrieren.

1.2 Projektgegenstand

Der Gegenstand dieses Projekts ist die Konzeption und Umsetzung einer PWA für Studierende, die drei Kernmodule integriert:

1. **Vorlesungsmanagement**, das Stundenpläne visuell darstellt.
2. **Kalender**, der Termine anzeigt.

3. **Notizmodul**, das das Erstellen, Kategorisieren und Durchsuchen von persönlichen Lernnotizen ermöglicht.

Die technologische Basis bildet ein ReactJS-Frontend, das PWA-Features nutzt, sowie ein in Node.js entwickeltes Backend, das in Docker-Containern läuft und eine MySQL-Datenbank verwaltet. Durch diese Architektur gewährleistet die Web App Skalierbarkeit, einfache Erweiterbarkeit und einen reibungslosen Betrieb in unterschiedlichen Hosting-Umgebungen.

2 Anforderungen und Konzept

2.1 Zielgruppenanalyse

Die Zielgruppe der Web App sind Studierende technischer und medienorientierter Studiengänge, die einen hohen Bedarf an Selbstorganisation haben. Viele verwenden mehrere Tools wie Papierkalendar, Notizapps, Moodle oder Google Kalender. Die Web App soll eine zentrale und strukturierte Plattform zur Verfügung stellen.

Digital unterstützte Selbstorganisation durch Studierende wird zunehmend international adressiert. Obexer und Giardina (2016) zeigen, dass die Rolle sogenannter Learning Designers - als Brücke zwischen Technologie und didaktischem Einsatz - nicht nur Lehrende unterstützt, sondern auch Studierende in ihrem Lernprozess stärkt. Sie betonen, dass ohne gezielte Support-Strukturen digitale Tools oft fragmentiert und ineffizeint genutzt werden.

2.2 Funktionale Anforderungen

Die Progressive Web App soll die folgenden zentralen Funktionen bereitstellen:

- **Vorlesungsverwaltung:** Nutzer:innen sollen Vorlesungen hinzufügen, bearbeiten und löschen können.
- **Kalenderintegration:** Prüfungstermine, Abgabefristen und Veranstaltungen sollen im Kalender sichtbar und verwaltbar sein.
- **Notizmodul:** Studierende sollen in der Lage sein, Notizen zu Vorlesungen zu speichern, zu verschlagworten und gezielt zu durchsuchen.
- **Offline-Funktionalität:** Die Anwendung soll auch ohne Internetzugang nutzbar bleiben (PWA-Standard).

- **Synchronisation:** Daten sollen geräteübergreifend synchronisiert werden.
- **Gebrauchstaugliche Oberfläche:** Die Web App soll eine intuitive, barrierearme Oberfläche bieten.

2.3 User Stories

Als Student:in möchte ich meine Vorlesungen eintragen, um meinen Stundenplan digital zu organisieren.
Als Student:in möchte ich Notizen zu Vorlesungen speichern, um diese später leicht und schnell zufinden.
Als Student:in ohne Internetverbindung möchte ich meine Notizen offline einsehen können, damit ich auch unterwegs lernen kann.
Als Student:in möchte ich Prüfungstermine in einem Kalender verwalten, um keine Fristen zu verpassen.

2.4 Grobkonzept und Mock-ups

Die Abblindung 2.1 zeigt die Startseite der Web App, die eine Übersicht über alle Vorlesungen und anstehenden Termine bietet. Die Navigation ermöglicht den Zugriff auf den Stundenplan, Notizen und den Kalender.

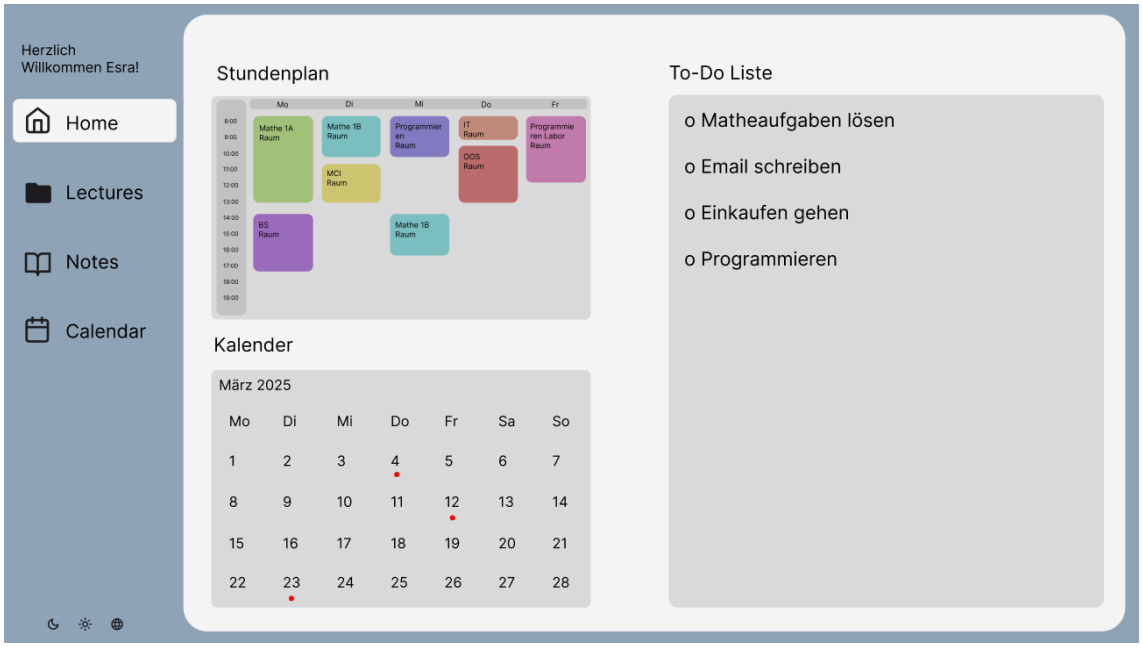

Abbildung 2.1: Startseite

In Abbildung 2.2 ist die Seite für die Verwaltung vom Stundenplan dargestellt. Hier können Nutzer:innen Vorlesungen hinzufügen, bearbeiten und löschen.

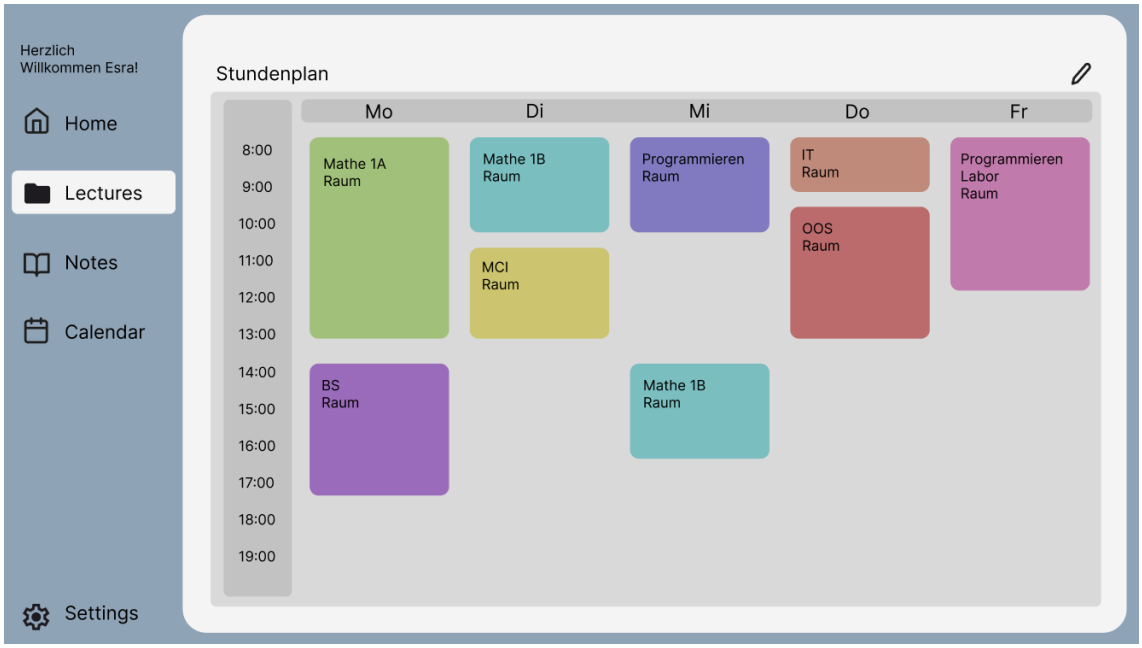

Abbildung 2.2: Stundenplan

Die Abbildung 2.3 zeigt das Notizmodul, in dem Studierende ihre Lernmaterialien organisieren können.

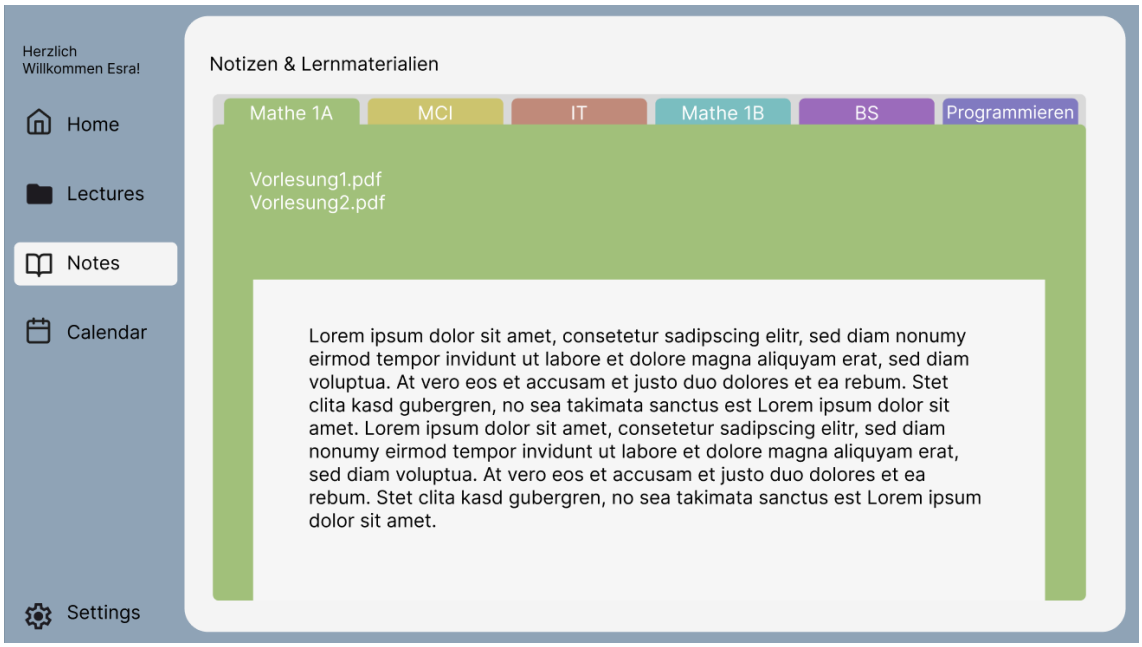

Abbildung 2.3: Notizen und Lernmaterialien

Abbildung 2.4 und 2.5 zeigen den Kalender, in dem Prüfungstermine und Fris-

ten eingetragen werden können. Nutzer:innen können neue Einträge hinzufügen und bestehende verwalten.

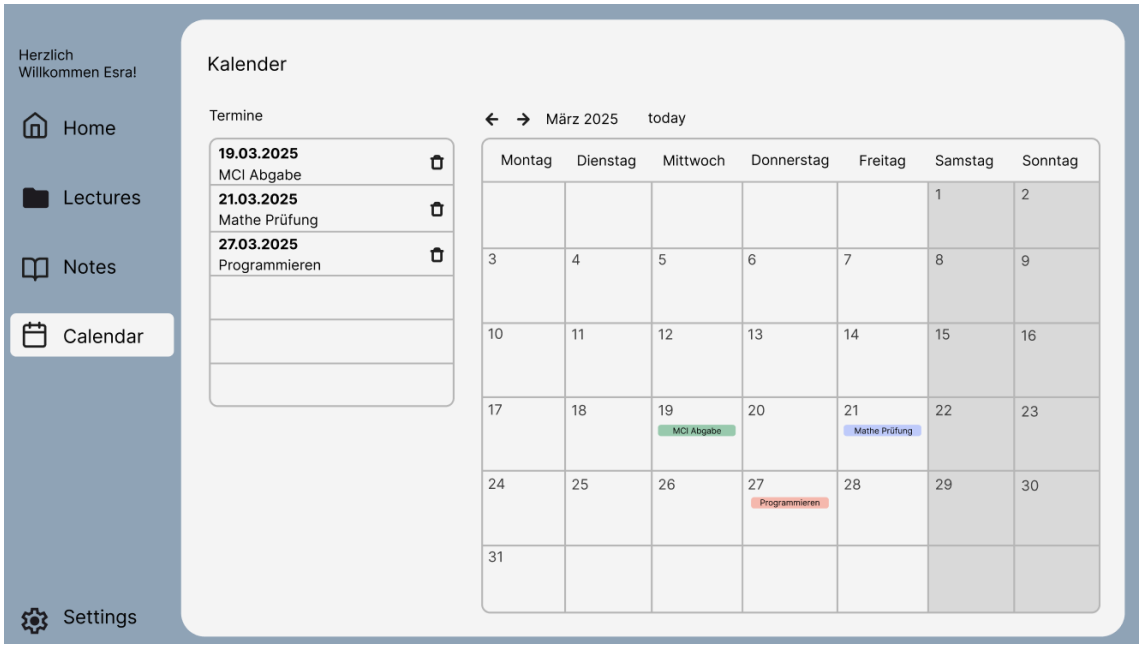

Abbildung 2.4: Kalender

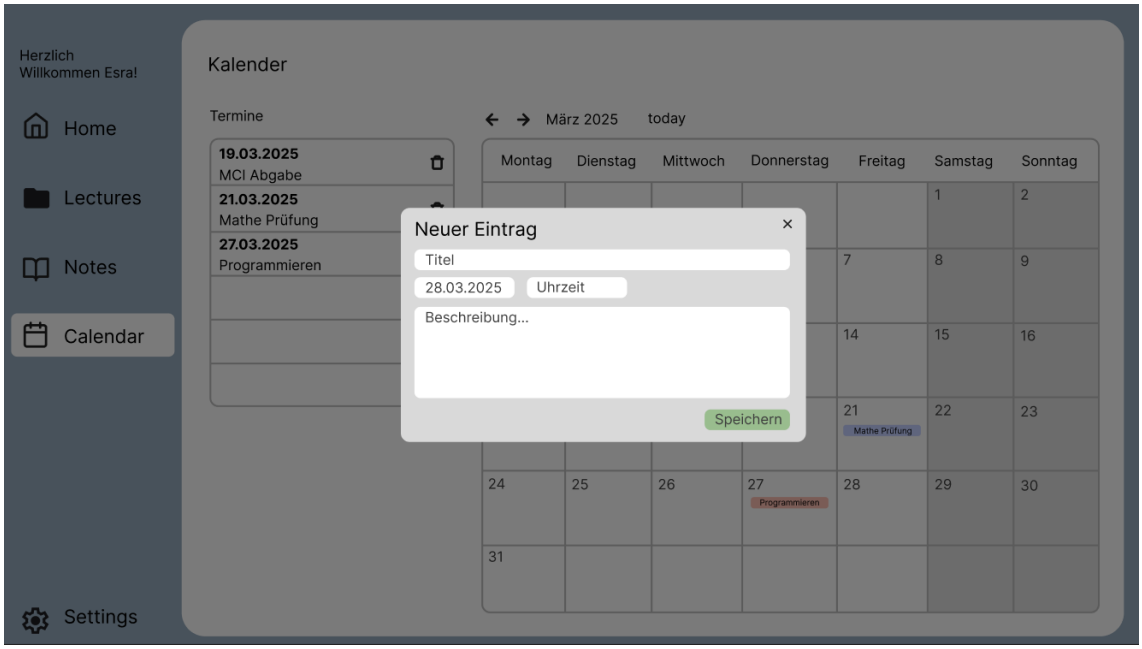

Abbildung 2.5: Neuen Eintrag in den Kalender einfügen

3 Architekturübersicht

3.1 Gesamtsystem und Komponenten

Das entwickelte System basiert auf einer modularen client-server-Architektur, bei der verschiedene Komponenten logisch getrennt, aber technisch eng miteinander verknüpft agieren. Ziel ist es, eine wartbare, skalierbare und plattformunabhängige Anwendung bereitzustellen, die sich durch hohe Gebrauchstauglichkeit auszeichnet.

- **Frontend (Client):** Das Frontend ist als Single Page Application (SPA) mit ReactJS umgesetzt. Es bietet die Benutzeroberfläche für Studierende und ermöglicht das Erstellen und Anzeigen von Vorlesungen, Notizen, Kalendereereignissen und To-Dos. Das Frontend greift über HTTP auf die REST-API des Backends zu.
- **Backend:** Das Backend ist in Node.js realisiert und stellt eine RESTful API bereit, die die Geschäftslogik der Anwendung kapselt. Es verarbeitet Anfragen vom Frontend, kommuniziert mit der Datenbank und führt die erforderlichen Operationen aus. Durch die Containerisierung in Docker ist eine konsistente Ausführung in verschiedenen Umgebungen gewährleistet.
- **Datenbank:** Die persistenten Daten werden in einer MySQL-Datenbank gespeichert. Diese läuft als Container innerhalb der Entwicklungs- und Produktionsumgebung. Die Datenbank enthält strukturierte Informationen zu Vorlesungen, Notizen, Kalendereereignissen und Stundenplaneinträgen. Das relationale Datenmodell erlaubt eine logische Trennung der Entitäten und eine effiziente Verarbeitung auch großer Datenmengen.
- **Dateispeicherung:** Hochgeladene Datein (z.B. PDF-Dokumente im Notizmodul) werden im lokalen Uploads-Verzeichnis des Servers gespeichert. Dieses Verzeichnis wird über einen statischen Pfad (/uploads) öffentlich bereitgestellt.

3.2 Technologiestack

In diesem Abschnitt werden die verwendeten Technologien beschrieben.

3.2.1 ReactJS

React ist eine Open-Source-JavaScript-Bibliothek, die dem User ermöglicht, Benutzeroberflächen aus einzelnen, wiederverwendbaren Komponenten zu erstellen. Mit React lassen sich sowohl klassische Web-UIs als auch native Benutzeroberflächen für Mobil- und Desktop-Apps realisieren, indem die gleichen Konzepte und das gleiche API-Design genutzt werden, so bleibt der Workflow plattformübergreifend konsistent und effizient. (React 2025)

3.2.2 PWA

Eine Progressive Web App (PWA) ist eine Webanwendung, die moderne Webtechnologien wie HTML, CSS und JavaScript nutzt, um das Nutzungserlebnis einer nativen mobilen App nachzuahmen. Dabei kann sie über einen gewöhnlichen Webbrowser aufgerufen werden, lässt sich jedoch auch wie eine klassische App auf dem Startbildschirm eines Geräts installieren. PWAs zeichnen sich durch Eigenschaften wie Offline-Funktionalität, schnelle Ladezeiten, Push-Benachrichtigungen und eine für mobile Endgeräte optimierte Benutzeroberfläche aus. Ziel ist es, die Vorteile von Websites mit denen nativer Apps zu kombinieren und somit eine plattformübergreifende, gebrauchstaugliche Lösung bereitzustellen. (Sana 2025)

3.2.3 Node.js

Node.js ist eine offene, plattformübergreifende JavaScript-Laufzeitumgebung, die es ermöglicht, JavaScript nicht nur im Browser, sondern auch auf Servern, in Kommandozeilen-Tools und Skripten einzusetzen. Node.js bietet Entwicklern eine moderne, ereignisorientierte Plattform zur Erstellung hochskalierbarer Netzwerkapplikationen, ohne komplexe Thread-Verwaltung oder Blocking-I/O-Probleme. (Node.js 2025)

3.2.4 MySQL

MySQL ist ein weit verbreitetes, quelloffenes relationales Datenbankmanagementsystem (RDBMS). Es speichert strukturierte Daten in Tabellen mit expliziten Zeilen und Spalten, die durch Schemata definiert sind. Abfragen und Manipulation erfolgen über die standardisierte Sprache SQL (Structured Query Language). MySQL unterstützt ACID-konforme Transaktionen, was für sichere und zuverlässige Datenverarbeitung sorgt. Dank seiner hohen Leistung, Skalierbarkeit und der großen Entwickler-Community eignet sich MySQL sowohl für kleine Anwendungen als auch für internationale Webseiten, Webdienste und Unternehmenssysteme. (MySQL 2025)

3.2.5 Docker

Docker ist eine Plattform für Betriebssystemvirtualisierung, die Anwendungen in sogenannten Containern verpackt. Diese Container enthalten nicht nur die Anwendung selbst, sondern auch alle benötigten Abhängigkeiten. Im Gegensatz zu VMs teilen sich Container den Kernel des Host-Systems, was sie deutlich leichter und schneller macht. Images dienen dabei als Bauvorlagen und Container laufen als deren Instanzen. Docker unterstützt eine konsistente Portabilität über verschiedene Umgebungen hinweg und ermöglicht skalierbare, isolierte Anwendungsbereitstellung. Registry-Services wie Docker Hub erlauben das Teilen und Verwalten von Images. Zusätzlich bieten Tools wie Docker Compose die Orchestrierung mehrerer Container innerhalb komplexerer Anwendungen. (Docker 2025)

3.3 Datenmodell und Datenbank

Zur strukturierten und konsistenten Verwaltung der Anwendungsdaten wird ein relationales Datenmodell verwendet. Die Datenbank bildet das Rückgrat für die Speicherung von Informationen zu Vorlesungen, Notizen und Kalendereinträgen.
Zum Einsatz kommt eine MySQL-Datenbank, da dieses System weit verbreitet, gut dokumentiert und durch seine stabile Performance besonders geeignet für webbasierte Anwendungen ist. Die Datenbank läuft als eigenständiger Docker-Container und ist über eine REST-API mit dem Backend verbunden.

Für die Verwaltung und Speicherung der Anwendungsdaten wird eine relationale Datenbankstruktur verwendet, die mit MySQL umgesetzt wurde. Das Datenmodell ist in mehrere logisch getrennte Entitäten gegliedert und orientiert sich an den funktionalen Anforderungen der Webanwendung. Die folgende Grafik zeigt den schematischen Aufbau des Datenbanksystems:

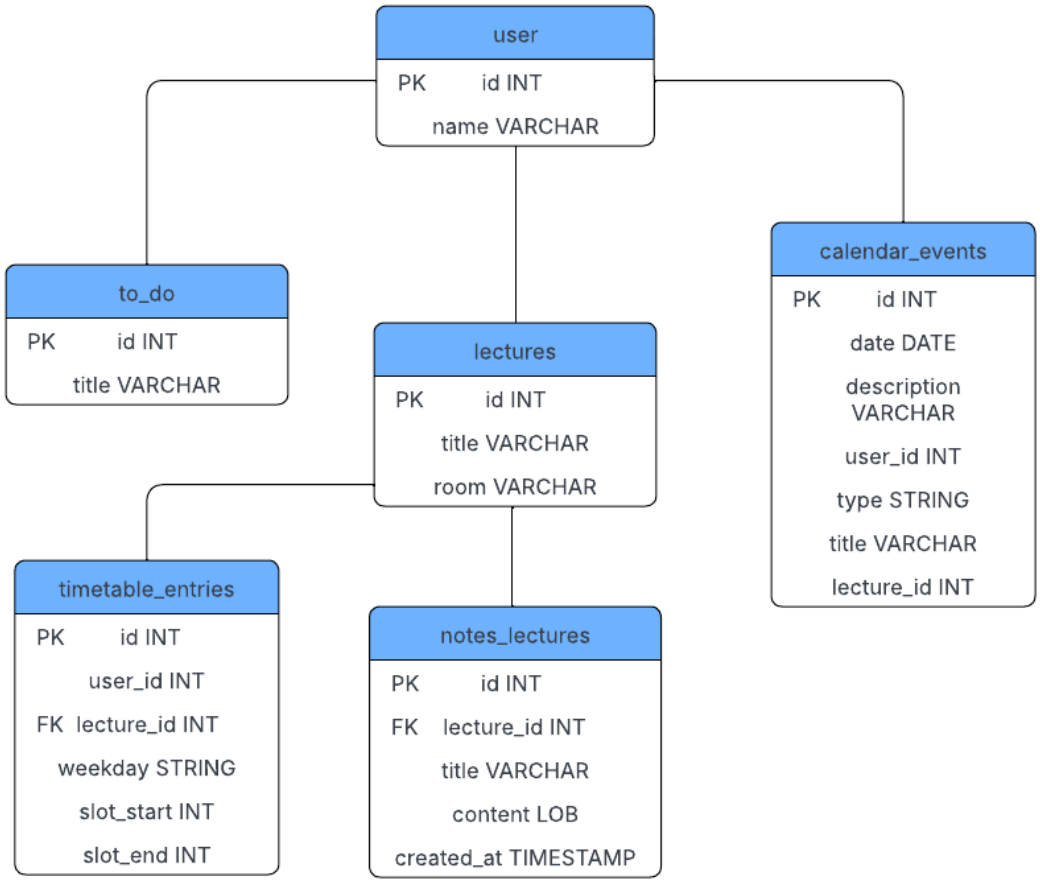

Abbildung 3.1: Datenbankmodell

Entitäten und Beziehungen:

- **user:** Enthält Informationen zu den registrierten Nutzer:innen. Nutzer:innen können mehrere Vorlesungen, Notizen, Kalenderereignisse und To-Do-Einträge besitzen.
- **lectures:** Repräsentiert einzelne Vorlesungen. Jede Vorlesung kann mit mehreren Notizen und Stundenplaneinträgen verknüpft sein.
- **notes_lectures:** Speichert zu einer bestimmten Vorlesung gehörende Notizen mit Titel, Inhalt und Zeitstempel. Die Beziehung zu `lectures` erfolgt über einen Foreign Key.

- **calendar_events:** Enthält Ereignisse wie Prüfungen, Abgaben oder Termine.
- **timetable_entries:** Bildet konkrete Stundenplaneinträge ab. Neben der Vorlesungs-ID werden auch Wochentag, Start- und Endzeit gespeichert.
- **to_do:** Dient der Verwaltung einfacher Aufgabenlisten, die einem Nutzer:innen zugeordnet sind.

Vorteile der Struktur:

- **Erweiterbarkeit:** Neue Funktionalitäten (z.B. Gruppen oder Tags) lassen sich durch zusätzliche Entitäten ohne größere Änderungen realisieren.
- **Skalierbarkeit:** Die klaren Beziehungen ermöglichen effiziente Abfragen, auch bei großen Datenmengen.
- **Wartbarkeit:** Die logische Trennung der Daten erhöht die Übersichtlichkeit und erleichtert Änderungen am Datenmodell.

3.4 API-Design

Die Kommunikation zwischen Frontend und Backend erfolgt über eine RESTful API, die auf dem HTTP-Protokoll basiert. Die Daten werden im JSON-Format übertragen. Die API stellt Endpunkte für die zentralen Funktionen der Anwendung bereit, wie das Erstellen und Abrufen von Vorlesungen, Notizen, Kalendereignissen und Aufgaben.

Endpoint	**Methode**	**Beschreibung**
/api/events	GET	Lädt alle Kalendereinträge, optional gefiltert nach `user_id` und `type`.
/api/events	POST	Erstellt einen neuen Kalendereintrag mit Datum, Titel, Beschreibung usw.
/api/events/:id	DELETE	Löscht einen Kalendereintrag anhand seiner ID.
/api/lectures	GET	Gibt eine Liste aller Vorlesungen zurück.
/api/lectures	POST	Fügt eine neue Vorlesung mit Titel und Raum hinzu.

/api/lectures/:id	GET	Gibt eine bestimmte Vorlesung anhand ihrer ID zurück.
/api/lectures/:id	DELETE	Löscht eine Vorlesung anhand ihrer ID.
/api/timetable	POST	Fügt einen Stundenplaneintrag für einen bestimmten Nutzer hinzu.
/api/timetable/:user_id	GET	Gibt den vollständigen Stundenplan eines Nutzers zurück.
/api/timetable/:id	DELETE	Löscht einen Stundenplaneintrag anhand seiner ID.
/api/to_do	POST	Fügt einen neuen To-Do-Eintrag hinzu.
/api/to_do	GET	Lädt alle To-Do-Einträge.
/api/to_do/:id	DELETE	Löscht einen bestimmten To-Do-Eintrag.
/api/notes_lectures	POST	Lädt eine neue Notiz zu einer Vorlesung hoch (PDF + Titel).
/api/notes_lectures	GET	Lädt alle Notizen, sortiert nach Erstellungsdatum.
/api/notes_lectures/:lecture_id	GET	Lädt alle Notizen zu einer bestimmten Vorlesung.
/api/notes_lectures/:id	DELETE	Löscht eine hochgeladene Notiz anhand ihrer ID.

Die grundsätzliche Systemarchitektur ist in Abbildung 3.2 dargestellt. Sie zeigt die Interatkion zwischen den drei Hauptkomponenten der Anwendung: dem ReactJS-basierten Frontend, dem containerisierten Node.js-Backend sowie der MySQL-Datenbank. Die Kommunikation erfolgt dabei über standardisierte HTTP- und SQL-Schnittstellen. Auch der Prozess des Datei-Uploads ist visualisiert: Hochgeladene Dokumente werden vom Frontend an das Backend übermittelt und dort im Verzeichnis `/uploads` gespeichert, auf das wiederum das Frontend zur Anzeige zugreift.

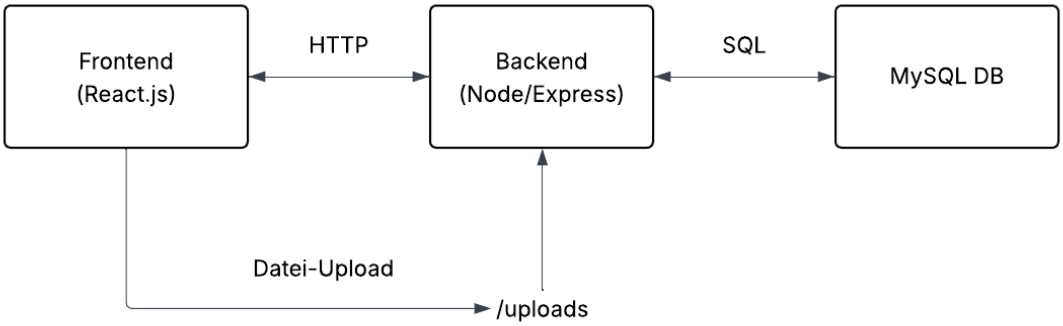

Abbildung 3.2: Systemarchitektur

4 Ergebnisse und Ausblick

4.1 Erreichte Funktionalitäten

In Abbildung 4.1 ist die Startseite der WebApp zu sehen. Sie bietet eine Übersicht über die wichtigsten Funktionen und ermöglicht den schnellen Zugriff auf den Stundenplan, Notizen, Kalender und Einstellungen. Die Startseite ist so gestaltet, dass sie eine klare Navigation und eine ansprechende Benutzeroberfläche bietet.

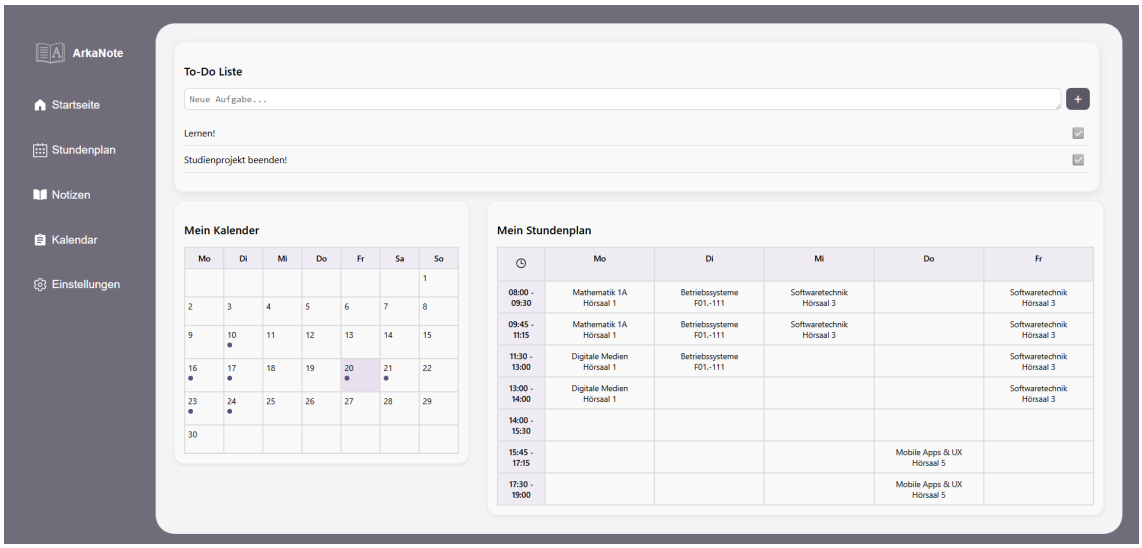

Abbildung 4.1: Startseite

Die WebApp ermöglicht es den Nutzer:innen, Vorlesungen zu verwalten und in ihren Stundenplan einzutragen. In Abbildung 4.2 ist der Stundenplan zu sehen, der eine visuelle Darstellung der Vorlesungen bietet. Nutzer:innen können neue Vorlesungen hinzufügen, wie in Abbildung 4.3 gezeigt, und diese dann in ihren Stundenplan eintragen (Abbildung 4.4).

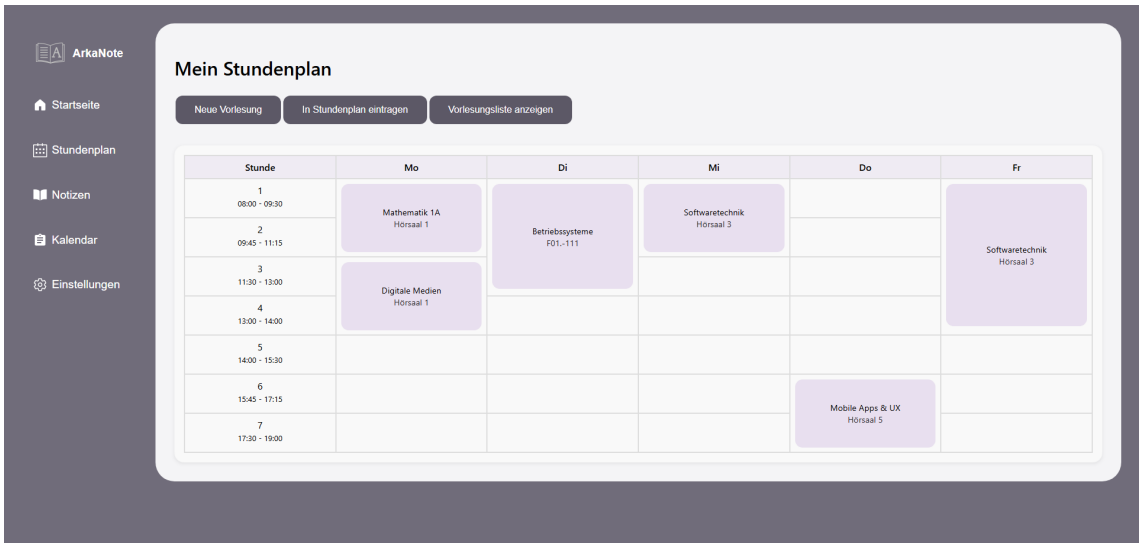

Abbildung 4.2: Stundenplan

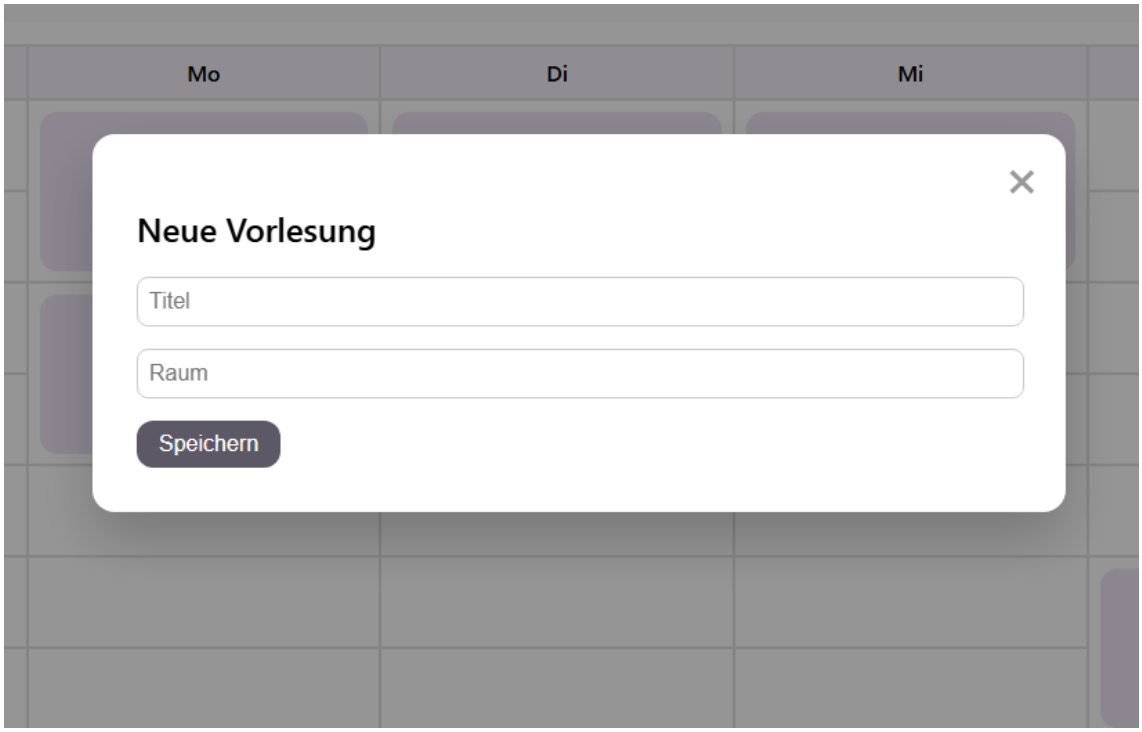

Abbildung 4.3: Neue Vorlesung hinzufügen

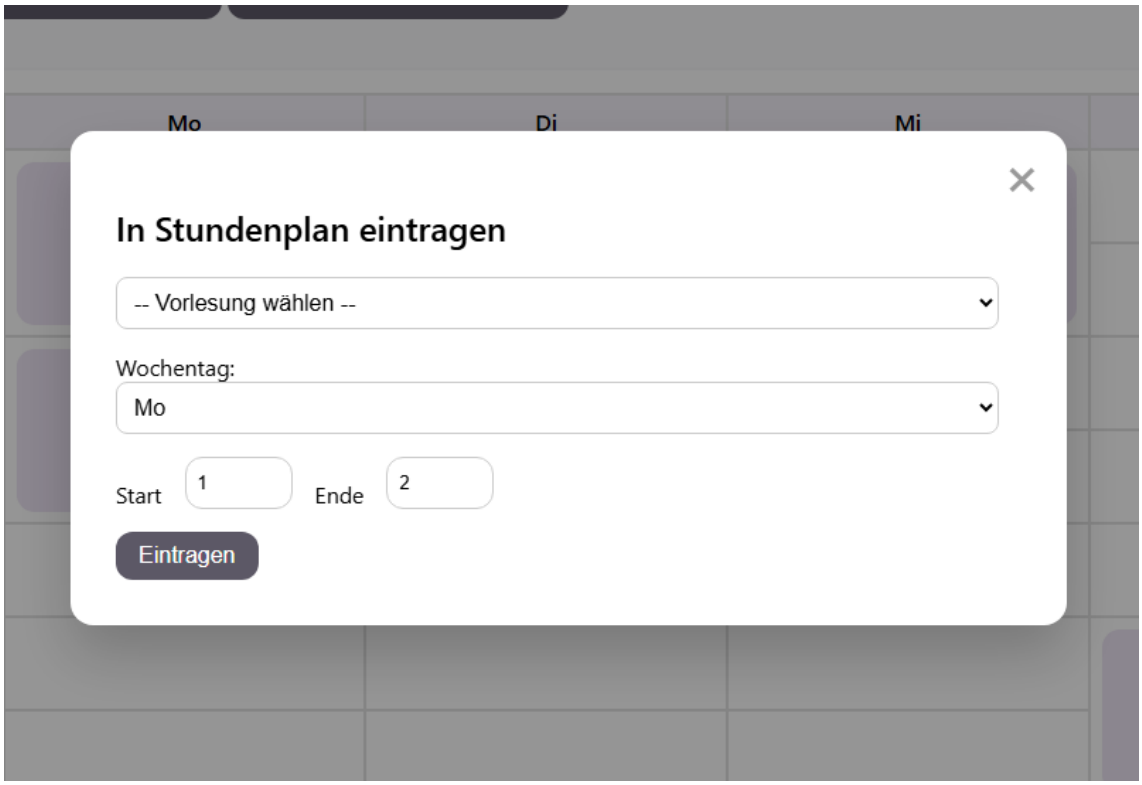

Abbildung 4.4: Vorlesung in den Stundenplan eintragen

Abbildung 4.5 zeigt die Vorlesungsliste, in der alle verfügbaren Vorlesungen aufgelistet sind. Nutzer:innen können aus dieser Liste Vorlesungen löschen.

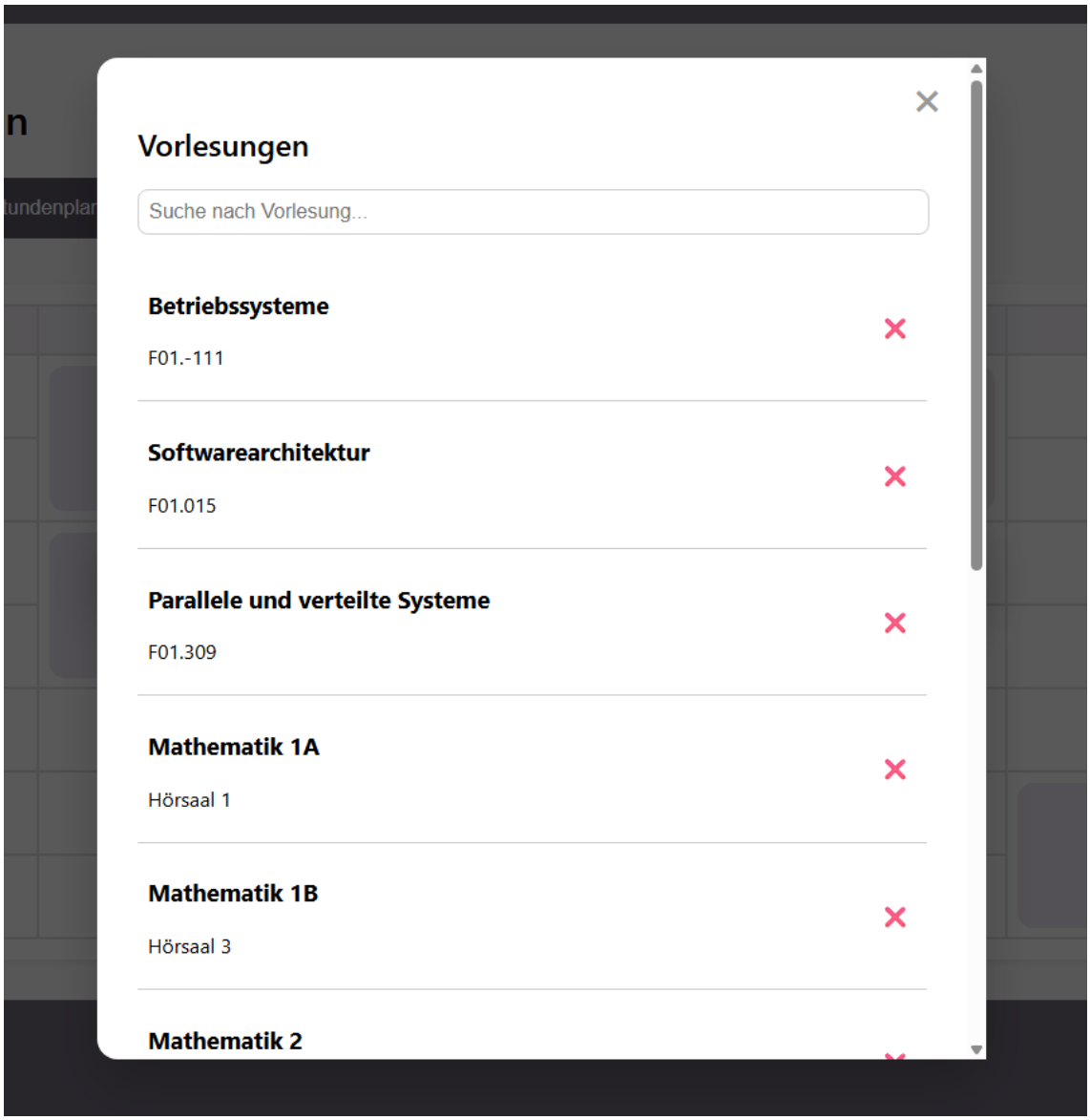

Abbildung 4.5: Vorlesungsliste

Die Web App bietet auch eine Notizfunktion, die es den Nutzer:innen ermöglicht, Notizen zu Vorlesungen zu erstellen und zu verwalten. In Abbildung 4.6 sind die Notizen und Lernmaterialien zu sehen, die hochgeladen und kategorisiert werden können. Nutzer:innen können Notizen zu spezifischen Vorlesungen hinzufügen und diese nach Bedarf organisieren.

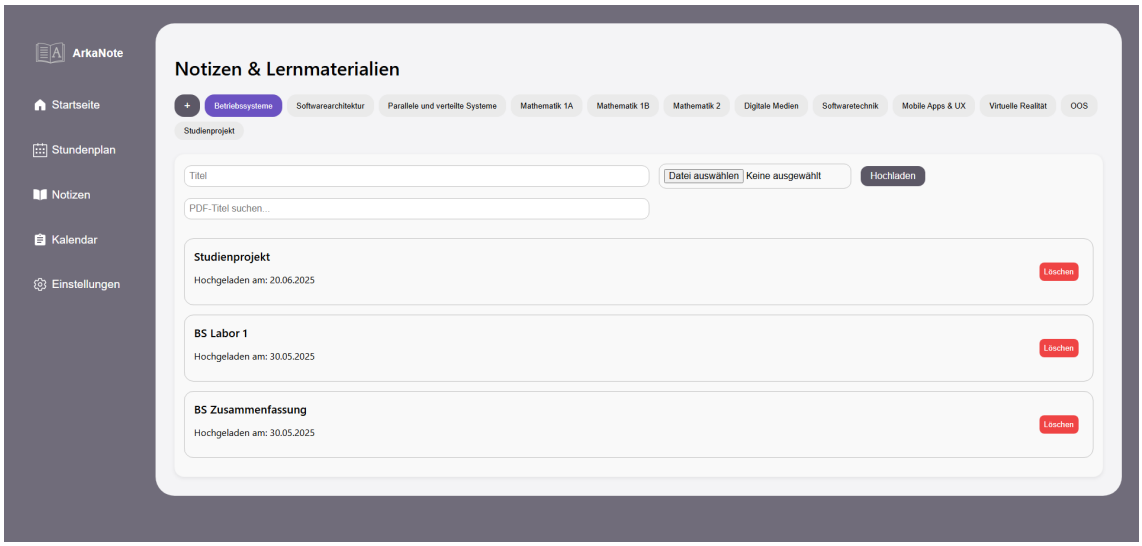

Abbildung 4.6: Notizen und Lernmaterialien

Die Kalenderfunktion der Web App ermöglicht es den Nutzer:innen, Termine und Fristen zu verwalten. In Abbildung 4.7 ist der Kalender zu sehen, in dem Nutzer:innen Ereignisse hinzufügen können, wie in Abbildung 4.8 gezeigt. Diese Funktionalität ist besonders nützlich, um wichtige Termine im Blick zu behalten und eine bessere Organisation des Studienalltags zu ermöglichen.

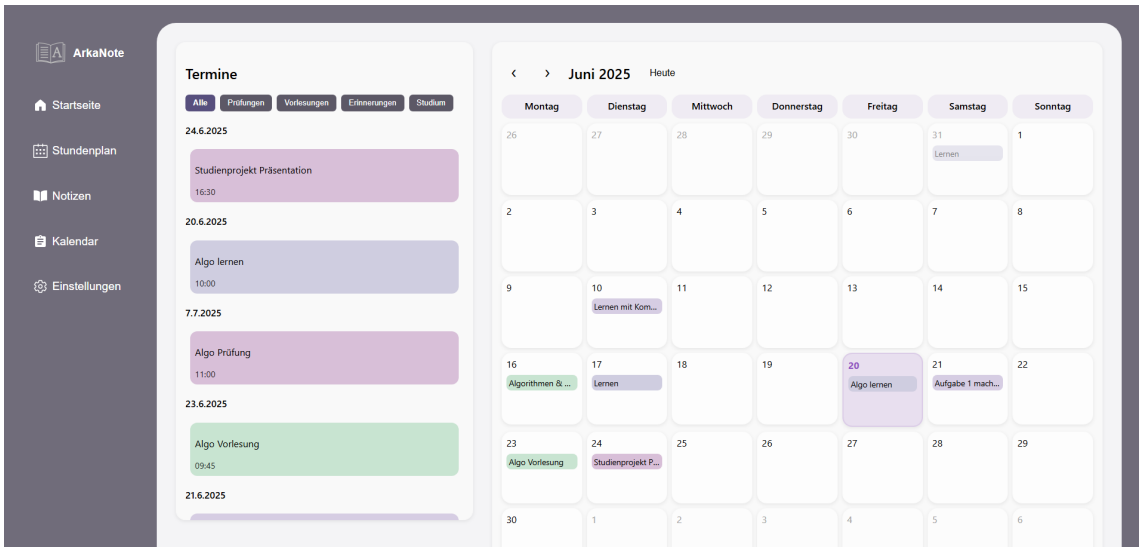

Abbildung 4.7: Kalender

Abbildung 4.8: Kalendereereignis hinzufügen

Abbildung 4.9 zeigt die Einstellungen der Web App, in denen Nutzer:innen ihre persönlichen Daten verwalten und das Design der Anwendung anpassen können.

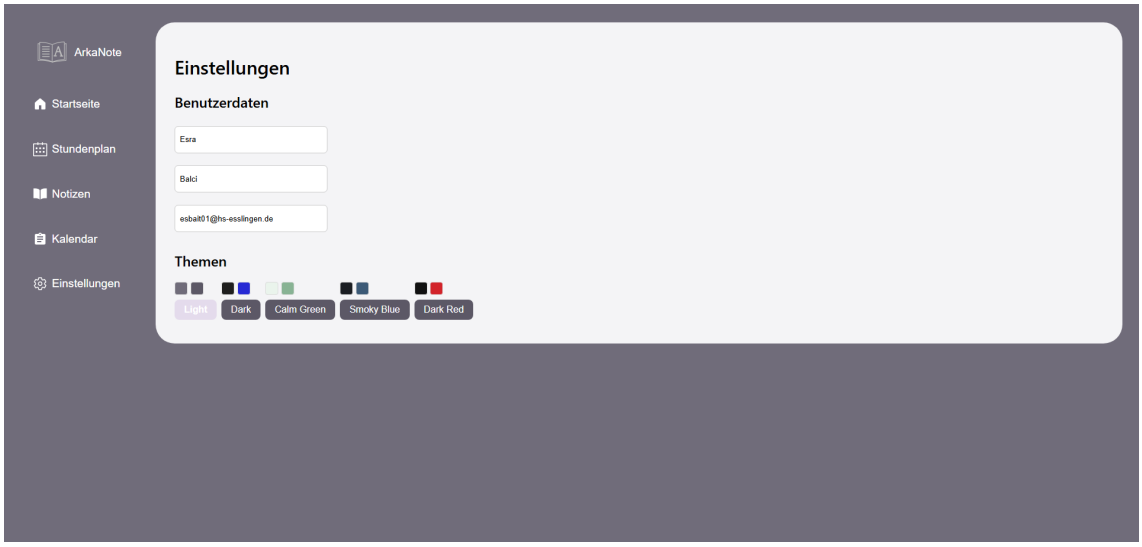

Abbildung 4.9: Einstellungen

Die Web App bietet verschiedene Design-Themes, die den Nutzer:innen eine individuelle Anpassung ermöglichen. In den Abbildungen 4.10, 4.11, 4.12, 4.13,

4.14, 4.15, 4.16 und 4.17 sind die verschiedenen Themes zu sehen, die den Nutzer:innen zur Verfügung stehen.

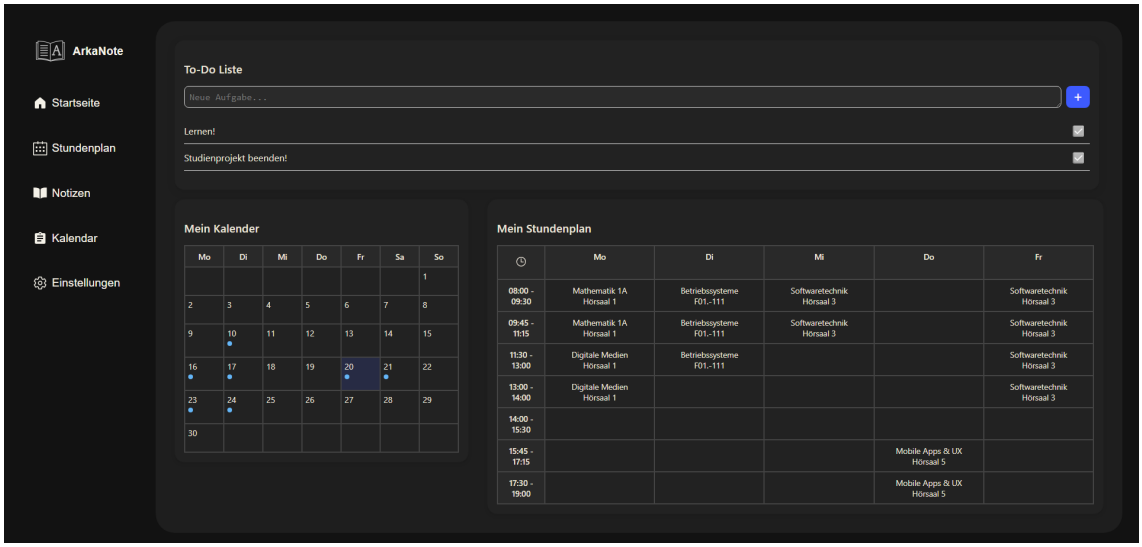

Abbildung 4.10: Startseite im Dark Theme

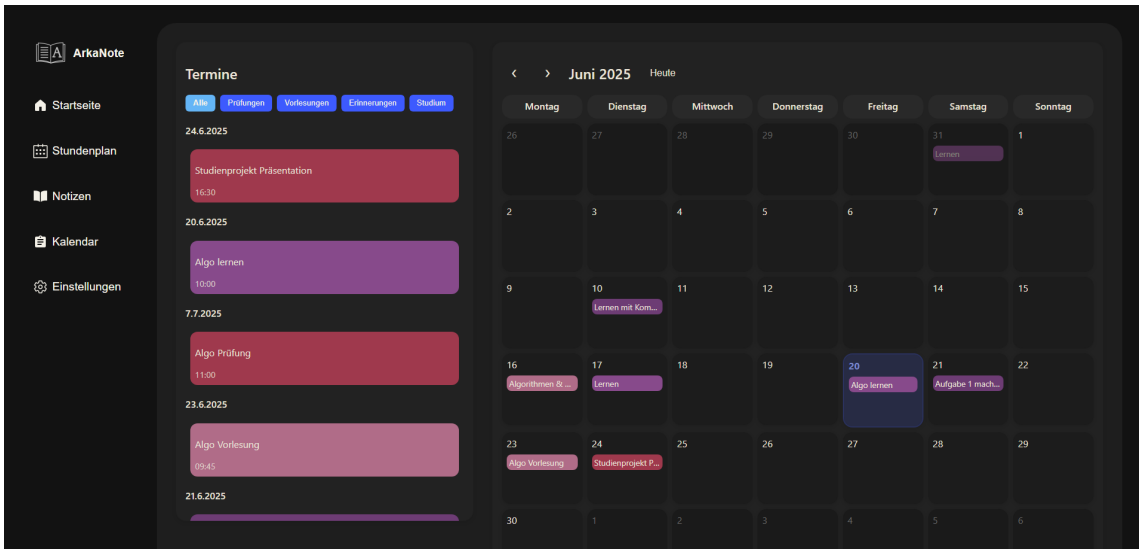

Abbildung 4.11: Kalender im Dark Theme

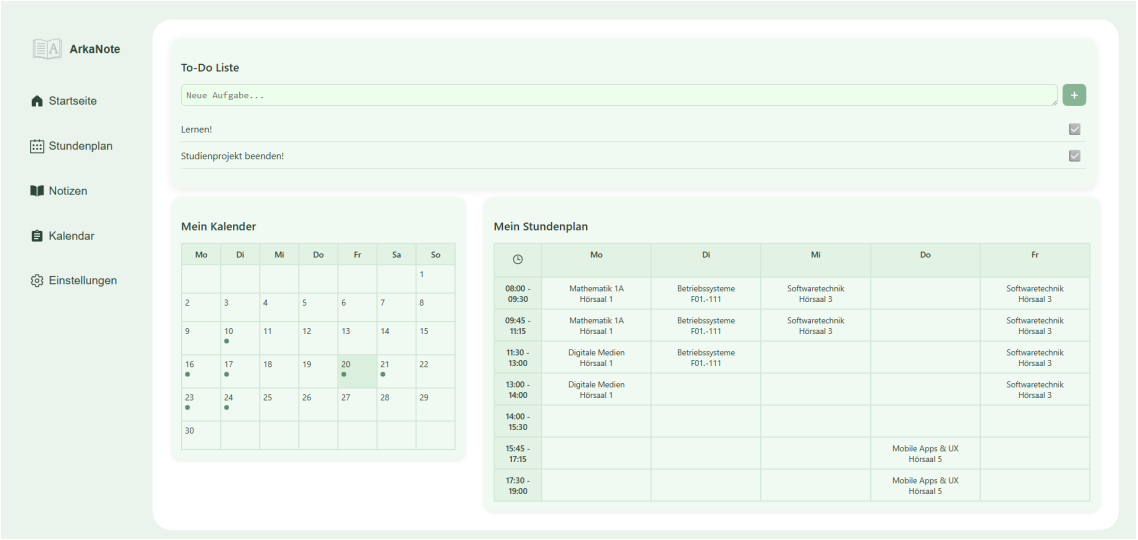

Abbildung 4.12: Startseite im Calm Green Theme

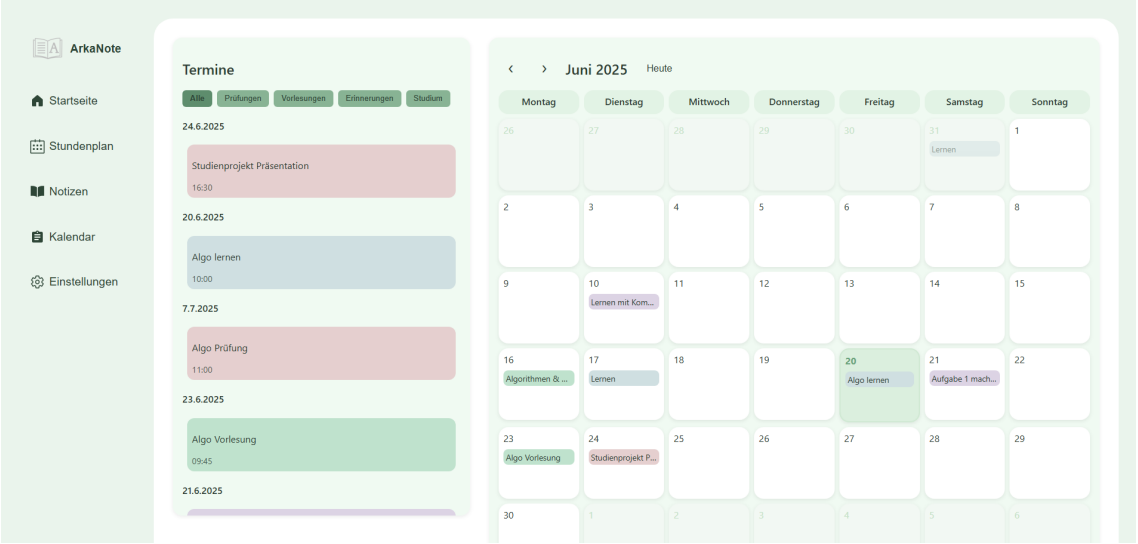

Abbildung 4.13: Kalender im Calm Green Theme

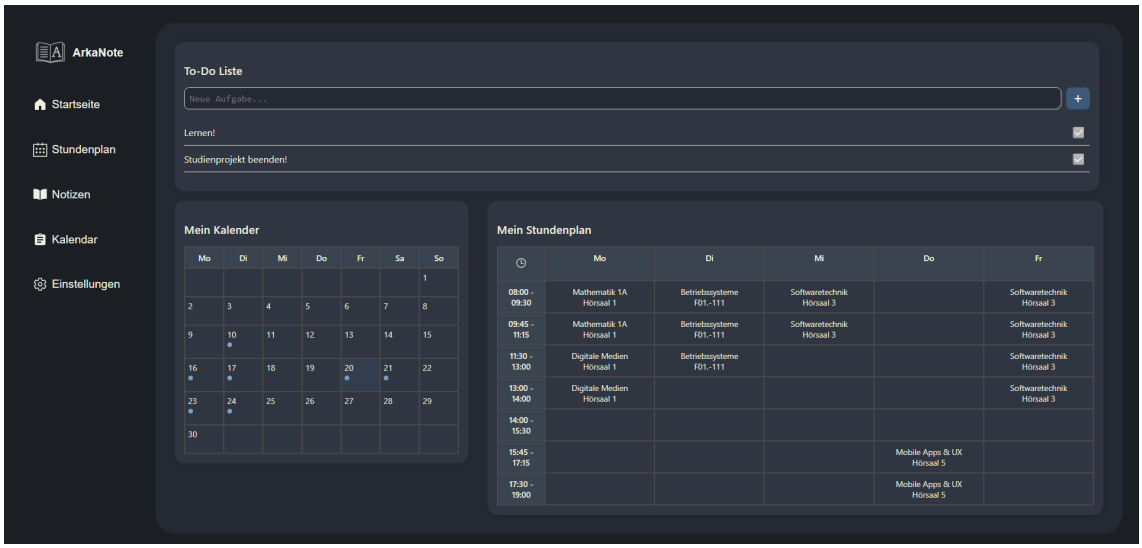

Abbildung 4.14: Startseite im Smoky Blue Theme

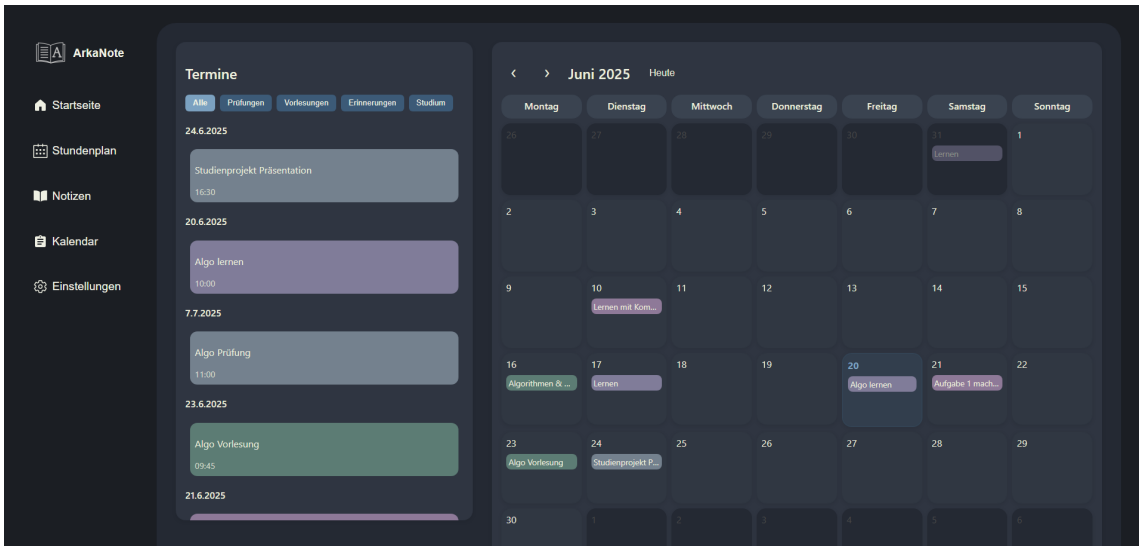

Abbildung 4.15: Kalender im Smoky Blue Theme

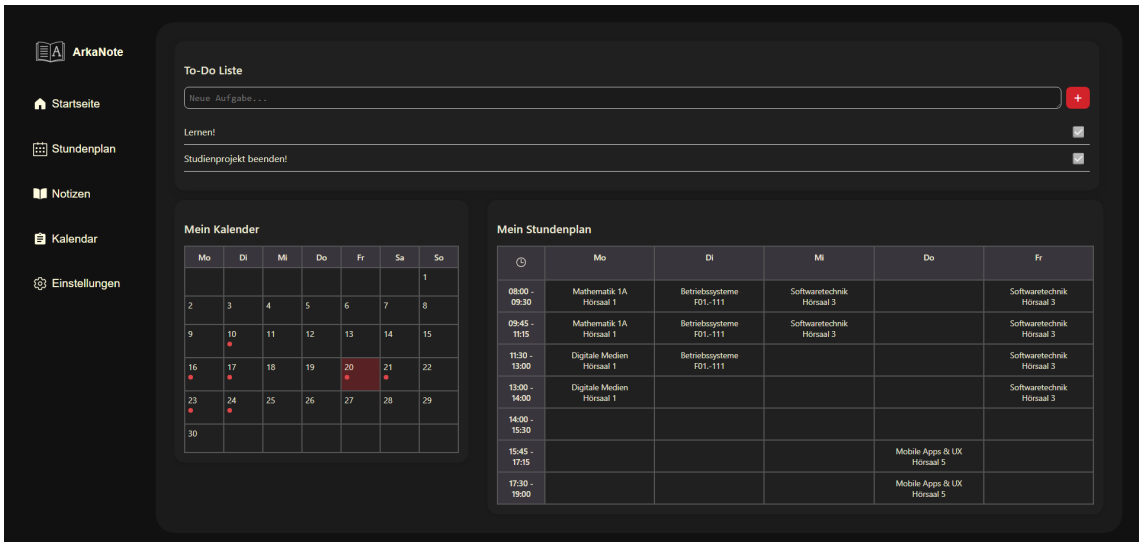

Abbildung 4.16: Startseite im Dark Red Theme

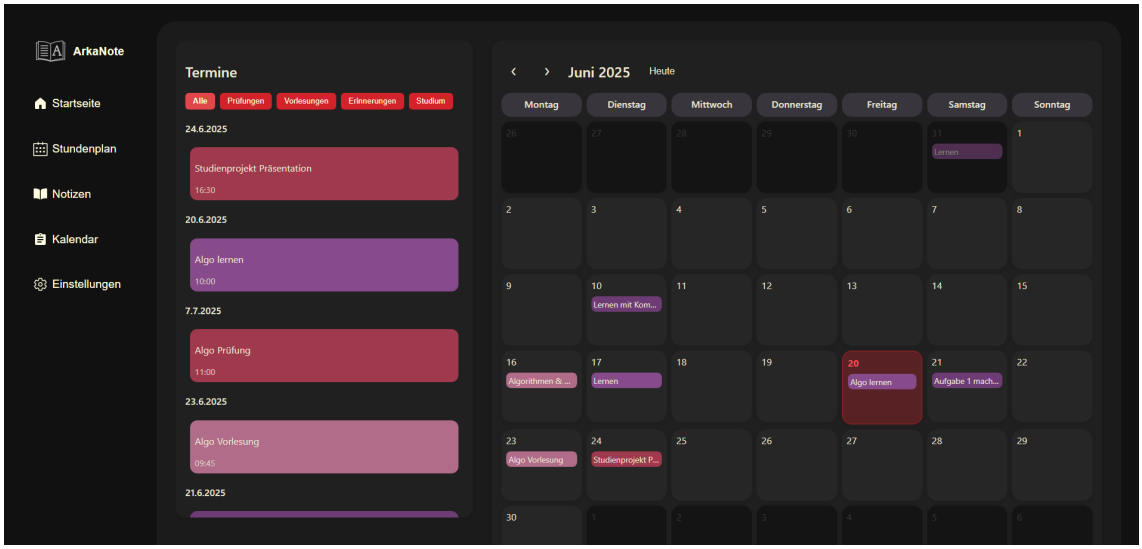

Abbildung 4.17: Kalender im Dark Red Theme

4.2 Vergleich zu bestehenden Lösungen

Kriterium / Tool	Projekt	Moodle	Google Kalender	Notion
Zielgruppe	Studierende, selbstorganisiert	Hochschulen, Lehrende + Studis	Allgemein, breites Publikum	Teams, Selbstorganisation
Stundenplan-Funktion	Visuell und interaktiv	Nicht enthalten	Nur Ereignisse	Manuell möglich
Kalender	Mit Deadlines und Kategorien	Mit Farbcodes + Kategorien	Eingeschränkt, wenig UX	Sehr leistungsfähig, aber unstrukturiert
Notizen mit Tags	Uploads	PDF-Upload + Tagging	Nur in Foren/Dateien	Sehr stark (Rich Content)
Offline-Funktion (PWA)	Nein	Nein	Nur mit Google Workspace	Nur mit App
Gebrauchstauglichkeit (UI)	Modern, Studierendenfokus	Veraltet, überladen	Klar und bekannt	Flexibel, aber komplex
Plattformunabhängigkeit	Web + App	Web + App	Web + App	Web + App
Open Source / Anpassbar	Ja (Eigenentwicklung möglich)	Teilweise	Nein	Nein
Integration in Lernumgebung	Manuell über Nutzung	Hochschulweit integriert	Nur externe Anbindung	Nur Copy/Paste

Tabelle 4.1: Vergleich vom Projekt mit bestehenden Tools für studentisches Selbstmanagement

4.3 Offene Punkte und mögliche Erweiterungen

Die entwickelte Web App erfüllt grundlegende Anforderungen an ein digitales Selbstorganisationswerkzeug, weist jedoch in ihrer aktuellen Version noch einige funktionale und technische Lücken auf, die in zukünftigen Entwicklungsschritten adressiert werden sollten.
Derzeit besteht keine Möglichkeit zur geräteübergreifenden Synchronisation von Daten. Dies bedeutet dass die Nutzung auf ein einzelnes Gerät beschränkt bleibt. Eine logische Weiterentwicklung stellt die Einführung eines cloudbasierten Backends mit Benutzerverwaltung dar.
Zudem fehlen Push-Benachrichtigungen, die die Nutzer:innen aktiv an bevorstehende Termine und Fristen erinnern könnten. Die Integration solcher Erinnerungsfunktionen würde den Nutzwert der App im Studienalltag deutlich steigern, insbesondere in Verbindung mit der Kalenderfunktion.
Neben der technischen Weiterentwicklung bestehen auch funktionale Erweiterungspotenziale. So wäre beispielsweise die Unterstützung von Gruppenfunktionen denkbar, bei der Studierende gemeinsame Notizen verwalten oder Termine in Lerngruppen organisieren können. Dies würde die Web App auch für kollaborative Lernszenarien attraktiver machen.
Langfristig ließe sich die Anwendung zudem durch den Einsatz Künstlicher Intelligenz erweitern. Möglich wären beispielsweise automatische Zusammenfas-

sungen von Notizen oder intelligente Vorschläge für die Terminplanung auf Basis individueller Lernmuster. Diese Erweiterungen könnten nicht nur die Effizienz der Web App steigern, sondern auch den indiviuellen Lernprozess der Studierenden nachhaltig fördern.

5 Fazit

In dieser Arbeit wurde eine Progressive Web App (PWA) entwickelt und umgesetzt, die Studierenden bei der Organisation ihres Studienalltags unerstützt. Ziel war es, eine gebrauchstaugliche Oberfläche für Vorlesungen, Notizen und Prüfungstermine bereitzustellen.
Die Implementierung zeigt, dass mit modernen Webtechnologien wie ReactJS ein hoher Grad an Gebrauchstauglichkeit und Plattformunabhängigkeit erreicht werden kann.
Für die Zukunft wären Features wie eine KI-gesstützte Zusammenfassung von Notizen oder eine Gruppenfunktion zur gemeinsamen Vorlesungsvorbereitung denkbar. Auch die Erweiterung auf eine native App oder die Integration in bestehende Hochschulplattformen bietet Potenzial.

Literatur

Busse, Julian u. a. (2022). „Didaktische Bedeutung der Digitalisierung für die kaufmännische Berufsausbildung“. In: *Digitale Transformation in der Berufsbildung*. wbv Publikation.

Docker (2025). *What is Docker?* URL: `https://docs.docker.com/get-started/docker-overview/` (besucht am 15. 06. 2025).

MySQL (2025). *MySQL*. URL: `https://www.mysql.com/de/` (besucht am 15. 06. 2025).

Node.js (2025). *Node.js*. URL: `https://nodejs.org/en/` (besucht am 27. 05. 2025).

Obexer, Regina und Natasha Giardina (2016). „What is a learning designer? Support roles and structures for collaborative E-Learning implementation“. In: *Digitale Medien: Zusammenarbeit in der Bildung*. Hrsg. von Josef Wachtler u. a. Waxmann, S. 137–146.

React (2025). *React*. URL: `https://react.dev/` (besucht am 26. 05. 2025).

Sana (2025). *PWA (Progressive Web App)*. URL: `https://www.sana-commerce.com/de/ecommerce-erklaert/was-ist-pwa/` (besucht am 15. 06. 2025).