Time Track

Projektkonzept

Enterprise Web Development

Claas Möhlmann
113366
tgtp6192@bht-berlin.de

Inhaltsverzeichnis

Projektkonzept

Die in diesem Projektkonzept beschriebene Anwendung ist als digitale Zeiterfassung für Werkstudenten konzipiert. Für die Priorisierung der Entwicklung wird zwischen einer primären und einer erweiterten Zielgruppe unterschieden. Zur primären Zielgruppe zählen Studierende, die als Werkstudenten tätig sind und ihre Arbeitszeiten außerhalb eines von Arbeitgebern bereitgestellten Zeiterfassungssystems dokumentieren möchten. Die Relevanz dieser Zielgruppe ergibt sich daraus, dass Studierende im Nebenerwerb im Jahr 2021 durchschnittlich etwa 12,5 Stunden pro Woche arbeiteten und ihre Arbeitszeit seit 2013 um 1,4 Stunden gestiegen ist [1]. Die erweiterte Zielgruppe umfasst selbstständig Tätige mit vergleichbaren Anforderungen an die Erfassung von Arbeitszeiten.

Die Wahl von Passkeys als Anmeldeverfahren orientiert sich an WebAuthn und an FIDO-Passkeys, die auf kryptografischen Anmeldungen ohne klassische Passwörter beruhen und als phishing-resistente Alternative zu passwortbasierten Verfahren etabliert sind [2], [3].

Für das Projekt wird sich in erster Linie auf die primäre Zielgruppe konzentriert; die erweiterte Zielgruppe wird als sekundärer Anwendungsfall berücksichtigt.

Die Anwendung soll es Nutzenden ermöglichen, ihre Arbeitszeiten zu erfassen, zu exportieren und dabei die Kontrolle über die eigenen Daten zu behalten, ohne auf ein externes, für sie nicht einsehbares System angewiesen zu sein. Dadurch erhalten sie eine eigenständige Dokumentation der geleisteten Arbeitszeit und reduzieren die Abhängigkeit von der korrekten Verarbeitung durch Systeme Dritter.

Ziel des Projekts ist nicht eine möglichst breit einsetzbare Lösung, sondern eine in ihrem Funktionsumfang bewusst begrenzte Anwendung mit hoher funktionaler Dichte und guter Gebrauchstauglichkeit. Dieses Vorgehen folgt dem Grundsatz eines reduzierten Systems mit geringer Interaktionskomplexität.

Anforderungen

Im Folgenden werden die funktionalen und nicht-funktionalen Anforderungen aufgeführt. Die wesentlichen Use-Cases der Anwendung werden im Kapitel Use-Cases erläutert. Einige Anforderungen erscheinen in mehreren Kategorien, etwa Barrierefreiheit, weil sie sowohl formale Konformitätskriterien als auch qualitative Anforderungen an die Nutzung betreffen.

Funktionale Anforderungen

1. Arbeitszeiterfassung
    1. Nutzende können ihre Arbeitszeiten aufzeichnen.
    2. Nutzende können ihre Arbeitszeiten exportieren.
    3. Nutzende können ihre Arbeitszeiten importieren.
    4. Nutzende können ihre Arbeitszeiten löschen.
    5. Nutzende können ihre Arbeitszeiten bearbeiten.
2. Projekte
    1. Nutzende können Projekte erstellen.
    2. Nutzende können Projekte bearbeiten.
    3. Nutzende können Projekte löschen.
    4. Nutzende können Zeiten Projekten zuordnen.
3. Organisationen
    1. Nutzende können Organisationen erstellen.
    2. Nutzende können Organisationen bearbeiten.
    3. Nutzende können Organisationen löschen.
    4. Nutzende können Projekte zu Organisationen zuordnen.

    5. Nutzende können Zeiten zu Organisationen über Projekte zuordnen.
    6. Nutzende können Nutzende zu Organisationen einladen.
    7. Nutzende können Nutzende aus Organisationen entfernen.
    8. Nutzende können Nutzende für Organisationen anlegen.
4. Stopp-Start-Funktionalität
    1. Nutzende müssen Arbeitszeiten starten und stoppen können; die Arbeitszeit wird dabei automatisch aufgezeichnet.
5. Benachrichtigungen
    1. Nutzende können regelmäßige Benachrichtigungen aktivieren, um an das Eintragen der Arbeitszeiten erinnert zu werden.
6. Die Anwendung soll in Deutsch und Englisch verfügbar sein.
7. Rechtliches (Compliance)
    1. Barrierefreiheit (Accessibility): Die Anwendung soll WCAG 2.2 AA-konform sein. [4]
8. WebAuthn/Sign-in
    1. Nutzende müssen sich mit einem Passkey registrieren können [2].
    2. Nutzende müssen sich über einen Passkey authentifizieren können [2].
    3. Nutzende können weitere Passkeys hinzufügen.
    4. Nutzende können Passkeys löschen.
    5. Hinzufügen, Editieren von Recovery-Email

Nicht-Funktionale Anforderungen

1. Benutzerfreundlichkeit (Usability)
    1. Die Anwendung soll intuitiv und einfach zu bedienen sein. Dazu soll die Benutzeroberfläche eine klare visuelle Hierarchie aufweisen und typische Aufgaben mit möglichst geringem Interaktionsaufwand unterstützen.
    2. Die Eingabe von Zeiten sollte mit möglichst wenigen Tastatureingaben erfolgen.
2. Sicherheit (Security)
    1. Daten werden verschlüsselt zwischen Client und Server gesendet.
    2. Daten von Nutzenden sind sicher vor unauthorisierten Zugriffen geschützt.
3. Barrierefreiheit (Accessibility)
    1. Dies ist sowohl eine funktionale als auch eine nicht-funktionale Anforderung. Funktional: Konformität mit den geforderten Standards. Nicht-funktional: eine über Mindestanforderungen hinausgehende Zugänglichkeit.
    2. In funktionaler Hinsicht soll die Anwendung die rechtlichen Anforderungen erfüllen.
4. Fokus
    1. Die Anwendung soll sich auf die primären Ziele konzentrieren und diese qualitativ hochwertig umsetzen, anstatt einen möglichst breiten Funktionsumfang anzustreben.
5. Plattformunabhängigkeit
    1. Die Anwendung soll auf verschiedenen Plattformen lauffähig sein. Dazu gehören Windows, macOS, Linux, iOS und Android.

- Die serverseitigen Komponenten der Anwendung sollen weitestgehend horizontal skalierbar sein, auch wenn im Rahmen des Projekts keine horizontale Skalierung umgesetzt wird.

1. Zuverlässigkeit (Reliability)
    1. Die Anwendung wird für Demonstrationszwecke bereitgestellt und soll in diesem Umfang verfügbar, leistungsfähig und zuverlässig sein.

Anforderungen außerhalb des Projektziels (Non-Goals)

Die Anwendung ist nicht für die Zeiterfassung innerhalb von Organisationen mit mehr als einer arbeitenden Person vorgesehen. Eine Verwaltung von Nutzenden innerhalb von Organisationen ist daher nicht Teil des Projektumfangs. Während des Projekts wird die Anwendung für Demonstrationszwecke bereitgestellt. Aus Kostengründen ist keine Produktionsumgebung vorgesehen; entsprechend sind hohe Last, langfristige Datenpersistenz und maximale Verfügbarkeit nicht Teil des Zielsystems.

Erweiterungen

Die Anwendung soll in Zukunft erweiterbar sein, um Nutzenden das nachträgliche Zuordnen von Arbeitszeiten zu Projekten zu ermöglichen. Dies ist jedoch nicht das primäre Ziel der Anwendung und wird gegebenenfalls in einem späteren Projekt ergänzt.

Anwendungsfälle (Use-Cases)

Nutzende können mit der Anwendung ihre Arbeitszeiten aufzeichnen, exportieren, importieren, löschen und bearbeiten. Eine Arbeitszeit ist einem Projekt zugeordnet; Nutzende können Projekte erstellen, bearbeiten und löschen. Zur Vereinfachung der Zeiterfassung können Nutzende einen Timer verwenden. Für diesen können sie optional Benachrichtigungen zur Erinnerung aktivieren.

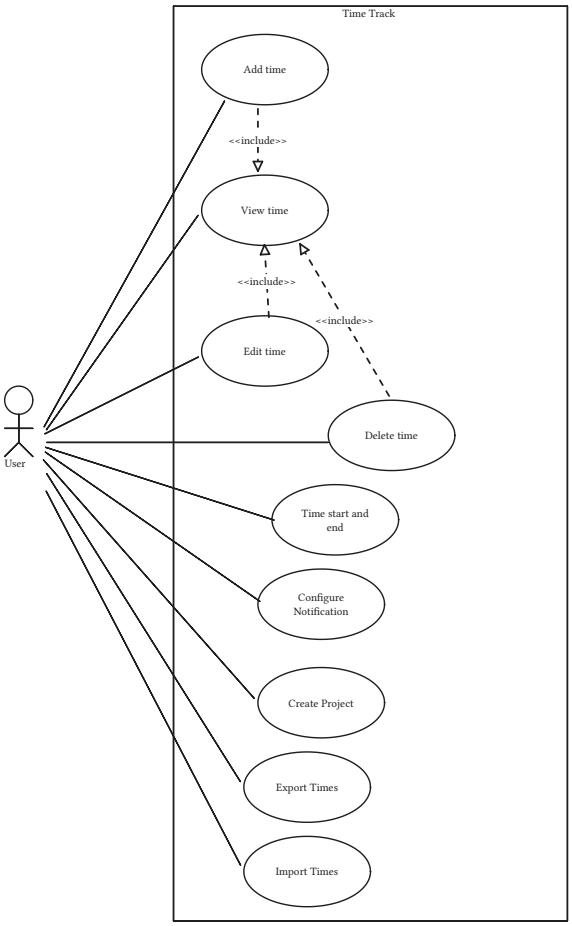

Abbildung 1: Time Tracking Use Case Diagram

Nutzende können sich mit der Anwendung authentifizieren und mit einem Passkey registrieren. Passkeys können mit einem Label versehen werden, um die Wiedererkennung zu erleichtern. Zudem können Nutzende Details zu ihren Passkeys einsehen, etwa den verwendeten Authenticator.

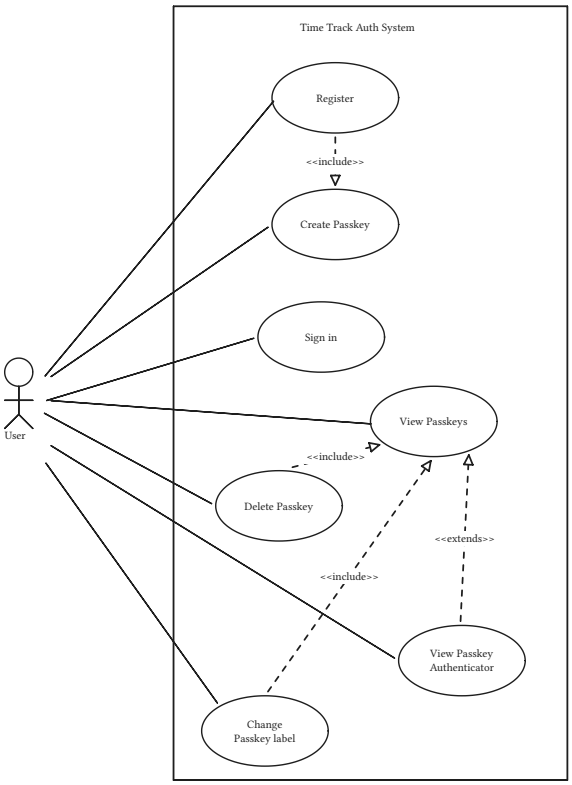

Abbildung 2: Auth Use Case Diagram

Entitäten

Im Zentrum der Anwendung steht die User-Entität. Sie dient dazu, sämtliche Daten einer Person zuzuordnen. In der aktuellen Version besitzt eine Person keinen Namen, da dieser an keiner Stelle benötigt wird. In einem System mit mehreren Nutzenden wäre ein Name zur gegenseitigen Identifikation sinnvoll; da Nutzende jedoch nicht miteinander interagieren, wird eine Minimierung der erfassten Daten bevorzugt. Nutzende können ein oder mehrere Projekte erstellen, unter denen sie Zeiten erfassen können. Zu jedem Zeitpunkt existiert mindestens ein Standardprojekt, das nicht gesondert angelegt werden muss, damit die Zeiterfassung unmittelbar begonnen werden kann. Für Timer existiert eine eigene Entität, damit laufende Timer auch über Neustarts der Anwendung hinweg erhalten bleiben. Sie speichern lediglich eine Startzeit, da sie bei Beendigung in die erfassten Zeiten überführt und anschließend entfernt werden. Separat zur User-Entität existieren WebAuthn-User über eine 1:1-Beziehung, die mit einem Passkey (WebAuthn-Credential) verknüpft sind. Diese Trennung ermöglicht es, Authentifizierung und Autorisierung modular aus dem System auszugliedern oder das gesamte Modul später einfacher zu entfernen. Zur besseren Wiedererkennung von Passkeys werden außerdem Metadaten wie Name und Icons der verwendeten Authenticatoren gespeichert.

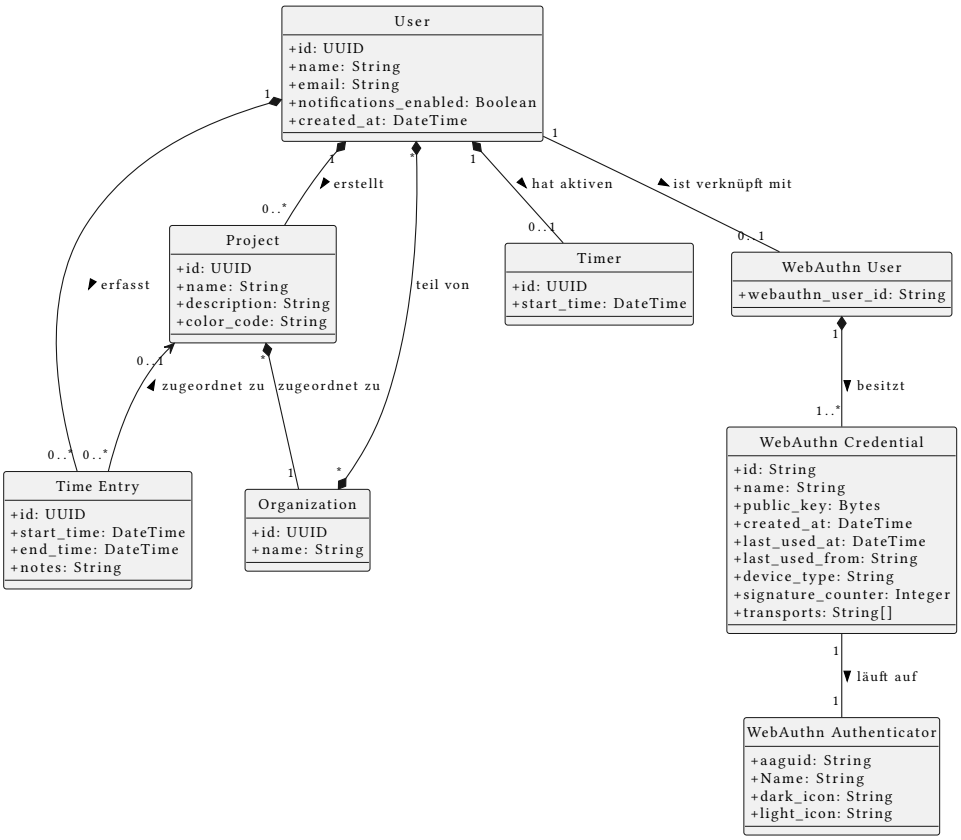

Abbildung 3: Entities Class Diagram

Software-Architektur

Durch die Anforderungen des Moduls an dieses Projekt soll für diese Software zwei Komponenten entwickelt werden, die in einer zentralisierten Server-Client-Architektur arrangiert sind. Die erste Komponente ist eine HTTP-JSON-API, die den REST-Prinzipien nach Fielding folgt [5]. Für Endpunkte zur Authentifizierung, Autorisierung und zum Ausliefern des Clients kann davon abgewichen werden, da der API-Server zugleich die Rolle des Authentifizierungsservers übernimmt.

Über die von der API bereitgestellten Endpunkte wird eine Client-Server-Architektur realisiert. Der Client ist eine Webanwendung, die die API-Endpunkte aufruft und Daten aus der API darstellt. Der Server verarbeitet und speichert die Daten. Für die persistente Speicherung kommen wahlweise PostgreSQL oder SQLite in Betracht. SQLite ist für einen schlanken Projekt- und Deployment-Setup vorteilhaft, da es als dateibasierte Datenbank ohne separates Datenbankserversystem betrieben werden kann [6]. PostgreSQL stellt demgegenüber ein vollwertiges Datenbanksystem mit separater Serverinstanz bereit [7]. Zusätzlich wird für die WebAuthn-Authentifizierung ein temporärer Speicher für Challenges benötigt. Zur Vereinfachung des Deployments kann dieser Speicher in einer Servertabelle umgesetzt werden, sofern ein externer Redis-Dienst vermieden werden soll. In diesem Fall ist ein regelmäßiger Hintergrundjob erforderlich, um abgelaufene Challenges zu entfernen und ein unkontrolliertes Anwachsen der Tabelle zu verhindern.

Damit Nutzende die Anwendung verwenden können, wird eine Single-Page-Application (SPA) implementiert. Sie greift über die API auf die Daten zu und bildet die Benutzungsoberfläche der Anwendung. Die Anwendung kann in modernen Webbrowsern auf unterschiedlichen Plattformen ausgeführt werden und erfüllt damit die Anforderung an Plattformunabhängigkeit. Das Hosting der SPA erfolgt zusammen mit der API, da dies cookie-basierte Authentifizierung vereinfacht und Cross-Origin Resource Sharing (CORS) vermeidet, weil SPA und API unter derselben Domain erreichbar sind.

Wireframes

Der erste Screen, den Nutzende sehen, ist der Sign-in-Screen. Dort können sie sich mit einem vorhandenen Passkey anmelden oder einen neuen Passkey registrieren.

Die beiden Optionen sind am unteren Rand des Screens angeordnet, damit sie auf mobilen Geräten leicht erreichbar sind. Auf Desktop-Geräten werden sie mittiger platziert, damit sie schneller wahrgenommen werden.

Nach dem Sign-in sehen Nutzende ihre aktuellen Tage der Woche und erfasste Zeiten. Dort können sie Zeiten erfassen, bearbeiten und löschen.

Über ein Menü oben rechts können Nutzende sich abmelden oder zu ihren Passkey-Einstellungen wechseln.

Um neue Zeiten schnell zu erfassen oder einen Timer zu starten, befindet sich unten rechts ein Floating-Action-Button in Daumenreichweite.

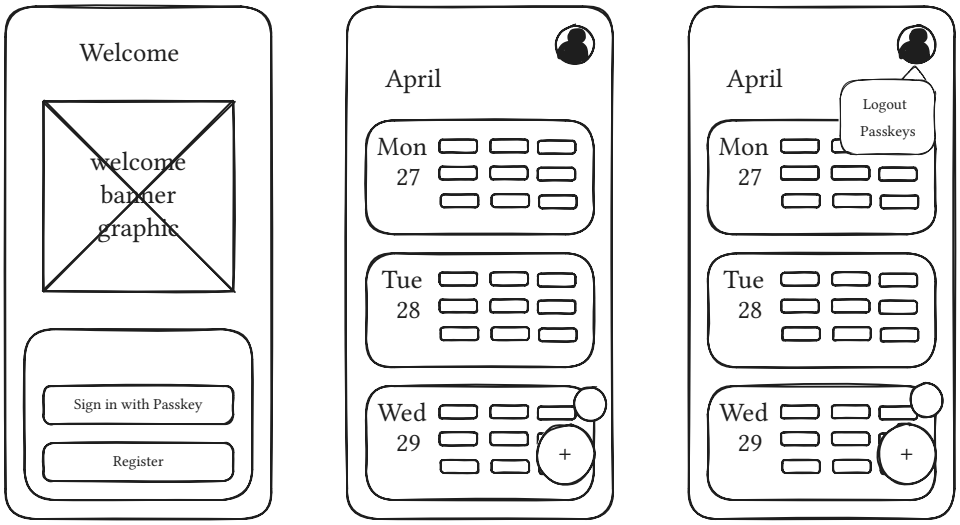

Abbildung 4: Wireframes Mobile

Auf Desktop-Geräten wird die Darstellung breiter und kann mehr Informationen nebeneinander anzeigen. Zudem befinden sich der Floating-Action-Button und das Menü oben links, da dies der Leserichtung folgt und mit einer Maus schneller erreichbar ist.

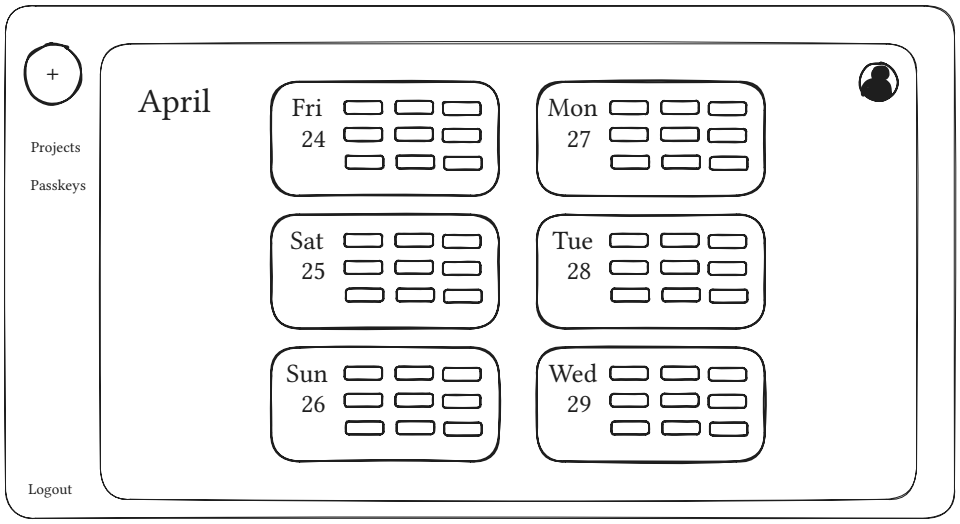

Abbildung 5: Wireframes Desktop

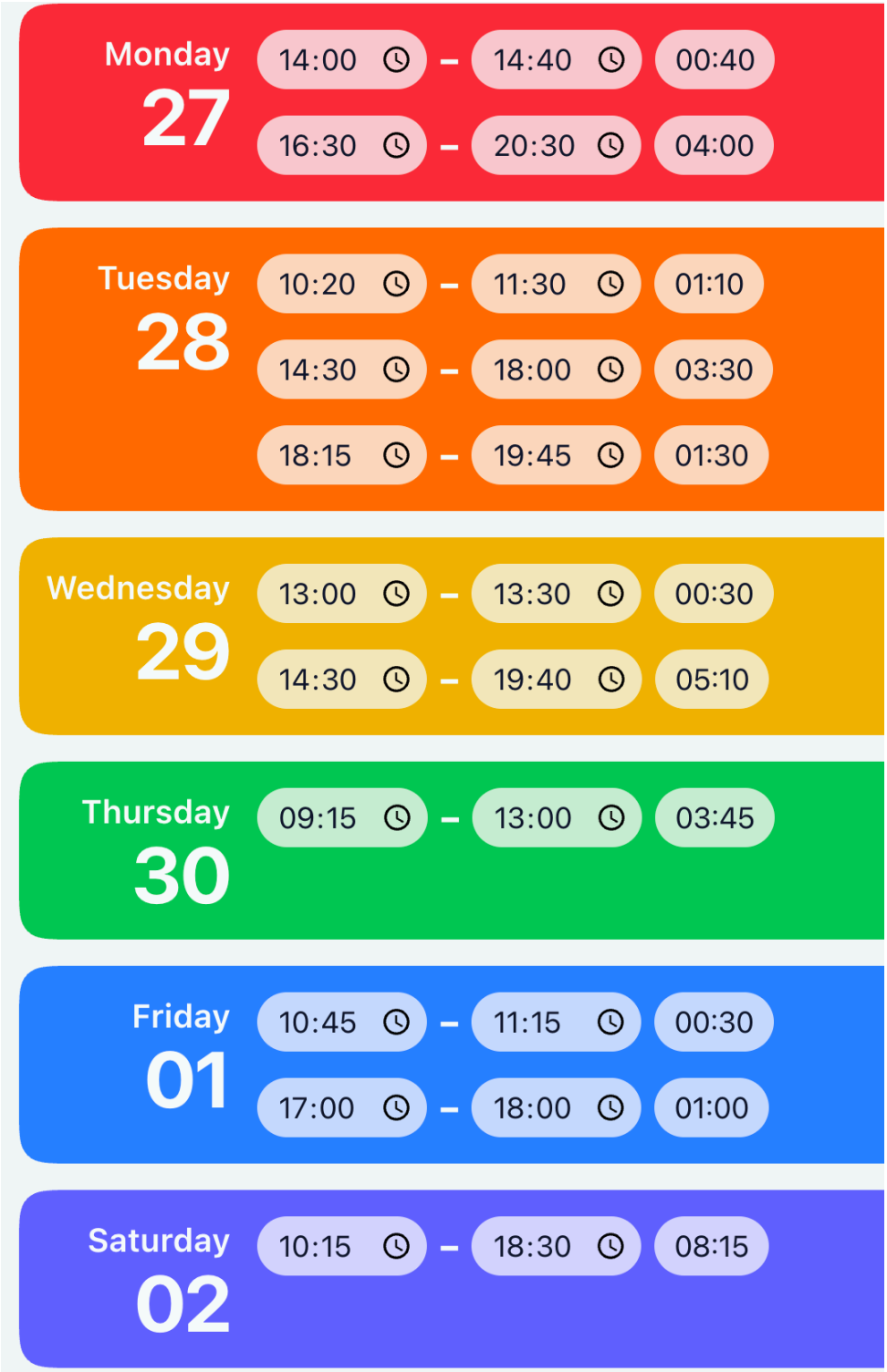

Abbildung 6: Prototype

Bibliografie

[1] jobvalley, „Durchschnittliche Arbeitsstunden im Nebenerwerb von Studierenden in Deutschland von 2013 bis 2021“. [Online]. Verfügbar unter: https://de.statista.com/statistik/daten/studie/1328490/umfrage/arbeitsstunden-im-nebenerwerb-von-studierenden/

[2] W3C, „Web Authentication: An API for accessing Public Key Credentials Level 2“. [Online]. Verfügbar unter: https://www.w3.org/TR/webauthn-2/

[3] F. Alliance, „Passkeys“. [Online]. Verfügbar unter: https://www.fidoalliance.org/passkeys/

[4] W3C, „Web Content Accessibility Guidelines (WCAG) 2.2“. [Online]. Verfügbar unter: https://www.w3.org/TR/WCAG22/

[5] R. T. Fielding, „Architectural Styles and the Design of Network-based Software Architectures“, *University of California, Irvine*, 2000.

[6] SQLite, „SQLite Documentation“. [Online]. Verfügbar unter: https://www.sqlite.org/docs.html

[7] P. G. D. Group, „PostgreSQL 18.3 Documentation“. [Online]. Verfügbar unter: https://www.postgresql.org/docs/current/