ClockWise

Project Oriented Time Management Application

Software-Entwicklungspraktikum (SEP)
Sommersemester 2025

Fachentwurf

Auftraggeber
Technische Universität Braunschweig
Peter L. Reichertz Institute for Medical Informatics
Prof. Dr. Thomas M. Deserno
Mühlenpfordtstraße 23
38106 Braunschweig

Betreuer: Corinna Thoben

Auftragnehmer:

Name	E-Mail-Adresse
Joud Mawad	j.mawad@tu-braunschweig.de
Laden Zeynep Erkenci	l.erkenci@tu-braunschweig.de
Sophie Gebauer	sophie.gebauer@tu-braunschweig.de
Luis Jair Gutierrez Pacheco	l.gutierrez-pacheco@tu-braunschweig.de
Chantal Ebben	c.ebben@tu-braunschweig.de
Merle Lüer	m.lueer@tu-braunschweig.de
Moetez Belleh Hosni	m.hosni@tu-braunschweig.de

Braunschweig, 25. Juni 2025

Bearbeiterübersicht

Kapitel	Autoren	Kommentare
1	Luis Jair Gutierrez Pacheco	
1.1	Luis Jair Gutierrez Pacheco	
1.2	Luis Jair Gutierrez Pacheco	
1.3	Luis Jair Gutierrez Pacheco	
2	Chantal Ebben	
2.1	Chantal Ebben	
2.2	Chantal Ebben	
2.3	Chantal Ebben	
2.3.1	Chantal Ebben	
2.3.2	Chantal Ebben	
2.4	Chantal Ebben	
2.5	Laden Zeynep Erkenci	
2.5.1	Laden Zeynep Erkenci	
2.5.2	Laden Zeynep Erkenci	
2.5.3	Laden Zeynep Erkenci	
2.6	Laden Zeynep Erkenci	
2.6.1	Laden Zeynep Erkenci	
2.6.2	Laden Zeynep Erkenci	
2.6.3	Laden Zeynep Erkenci	
2.7	Laden Zeynep Erkenci	
2.8	Laden Zeynep Erkenci	
2.9	Joud Mawad	
2.10	Joud Mawad	
2.11	Joud Mawad	
2.12	Joud Mawad	
2.13	Chantal Ebben	
3	Merle Lüer, Sophie Gebauer	
3.1	Merle Lüer	
3.2	Sophie Gebauer	

4	Sophie Gebauer	
5	Alle Teammitglieder	

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In einer zunehmend projektbasierten und digitalisierten Arbeitswelt wird strukturierte Zeiterfassung zu einem entscheidenden Faktor für effiziente Planung und transparente Zusammenarbeit. Die Anwendung ClockWise adressiert dieses Bedürfnis mit einer webbasierten Lösung zur projektorientierten Zeiterfassung und Analyse.

Dieses Dokument beschreibt die fachliche Struktur und das erwartete Verhalten des Systems ClockWise. Anhand von typischen Nutzungsszenarien und modellbasierten Diagrammen wird dargestellt, wie Nutzer:innen mit dem System interagieren und welche Abläufe dabei im Vordergrund stehen. Ziel ist es, einen systematischen Überblick über die Funktionalität der Anwendung zu geben, aus einer fachlichen Perspektive.

Das vorliegende Dokument ist wie folgt aufgebaut: Kapitel 1 stellt die fachliche Grundlage sowie typische Nutzungsszenarien vor. Kapitel 2 analysiert die Funktionen anhand konkreter Abläufe und Interaktionen. Kapitel 3 beschreibt das Datenmodell und Kapitel 4 enthält Konfigurationshinweise für den Einsatz der Anwendung.

1.1 Projektübersicht

ClockWise ist eine browserbasierte Webanwendung zur projektorientierten Zeiterfassung und Auswertung. Die Anwendung ermöglicht es Nutzer:innen, Zeitdaten strukturiert zu erfassen, sowohl live per Start/Stopp Funktion als auch manuell und diese in Kalender und Diagrammansichten zu analysieren. Zusätzlich erlaubt das System die Definition von Zeitlimits für Projekte und gibt bei deren Überschreitung automatisierte Hinweise.

Zu den Kernfunktionen zählen unter anderem die Verwaltung von Projekten und Aufgaben, die zeitliche Kategorisierung von Aktivitäten, eine Teamfunktion mit Rollenverteilung sowie Exportmöglichkeiten der erfassten Daten. Die Bedienung erfolgt über eine intuitive grafische Oberfläche im Webbrowser, wodurch keine lokale Installation erforderlich ist.

ClockWise richtet sich an Einzelpersonen und Teams, die ihre Zeitplanung systematisch und nachvollziehbar gestalten möchten. Die Anwendung ist vielseitig einsetzbar, z. B. im universitären Umfeld, im Arbeitskontext oder bei der persönlichen Projektorganisation.

1.2 Projektdetails

In diesem Abschnitt werden die zentralen Funktionen der Anwendung ClockWise aus fachlicher Sicht beschrieben. Dabei steht im Vordergrund, wie der:die Nutzer:in mit dem System interagiert, welche Abläufe vorgesehen sind und welche fachlichen Anforderungen diesen Prozessen zugrunde liegen. Zur besseren Verständlichkeit werden zentrale Funktionsabläufe — wo sinnvoll — zusätzlich durch Aktivitätsdiagramme visualisiert.

1.2.1 Registrierung und Login

Um ClockWise nutzen zu können, ist zunächst eine Registrierung erforderlich. Der:Die Nutzer:in legt ein persönliches Konto an, das durch eine Kombination aus E-Mail Adresse und Passwort gesichert ist. Nach erfolgreicher Registrierung erfolgt die Anmeldung über die Login-Seite.

Nach dem Login wird der:die Nutzer:in direkt zum Dashboard weitergeleitet, das den Einstiegspunkt für alle weiteren Funktionen bildet. Die Authentifizierung ermöglicht eine individuelle Nutzung der Anwendung und gewährleistet die sichere Speicherung persönlicher Zeitdaten.

Alternative Abläufe wie OAuth-Login oder Passwort-Zurücksetzung sind ebenfalls berücksichtigt (vgl. Abbildungen 1.1 und 1.2).

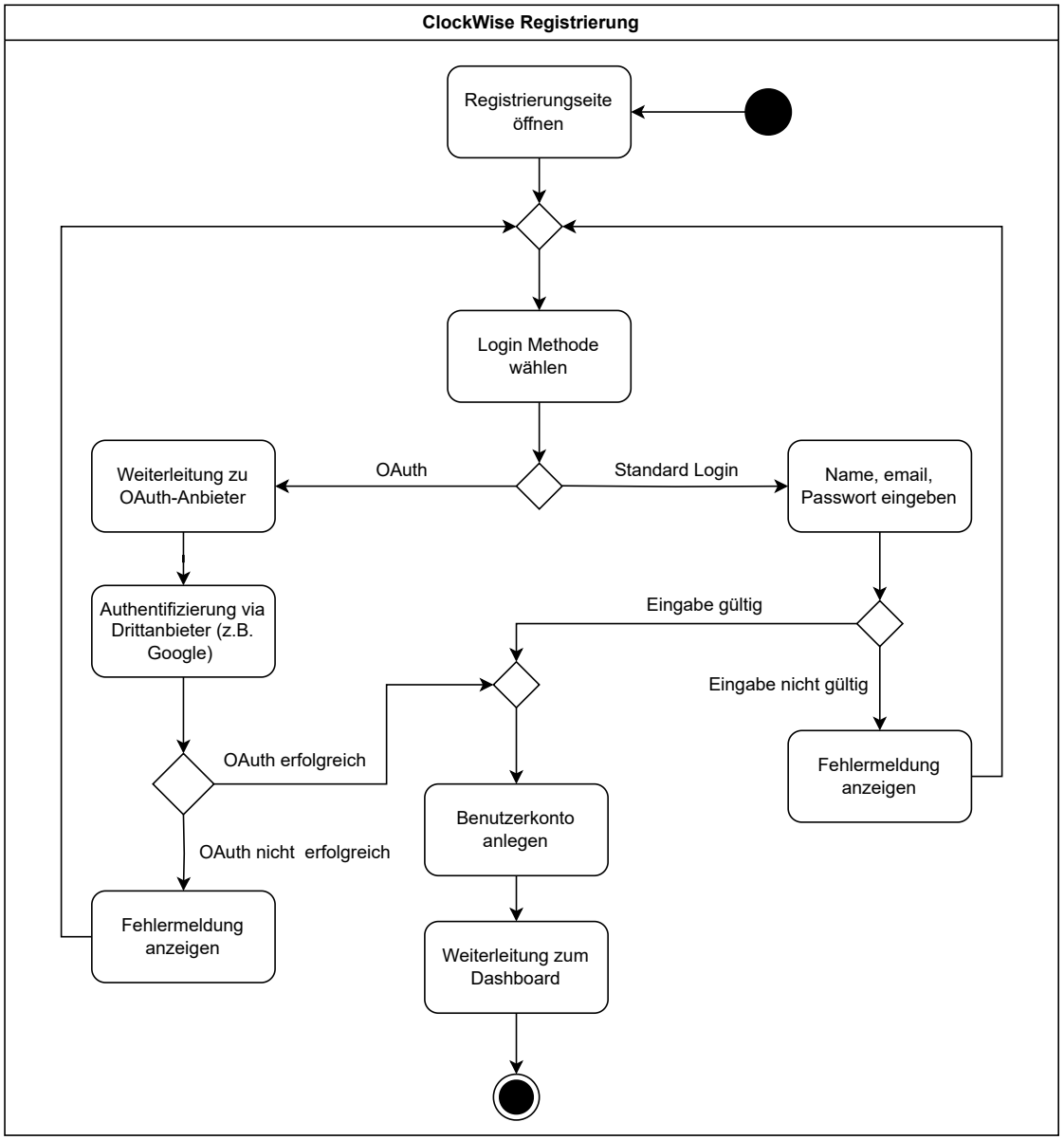

Abbildung 1.1: Aktivitätsdiagramm F10: Registrierung eines Nutzerkontos

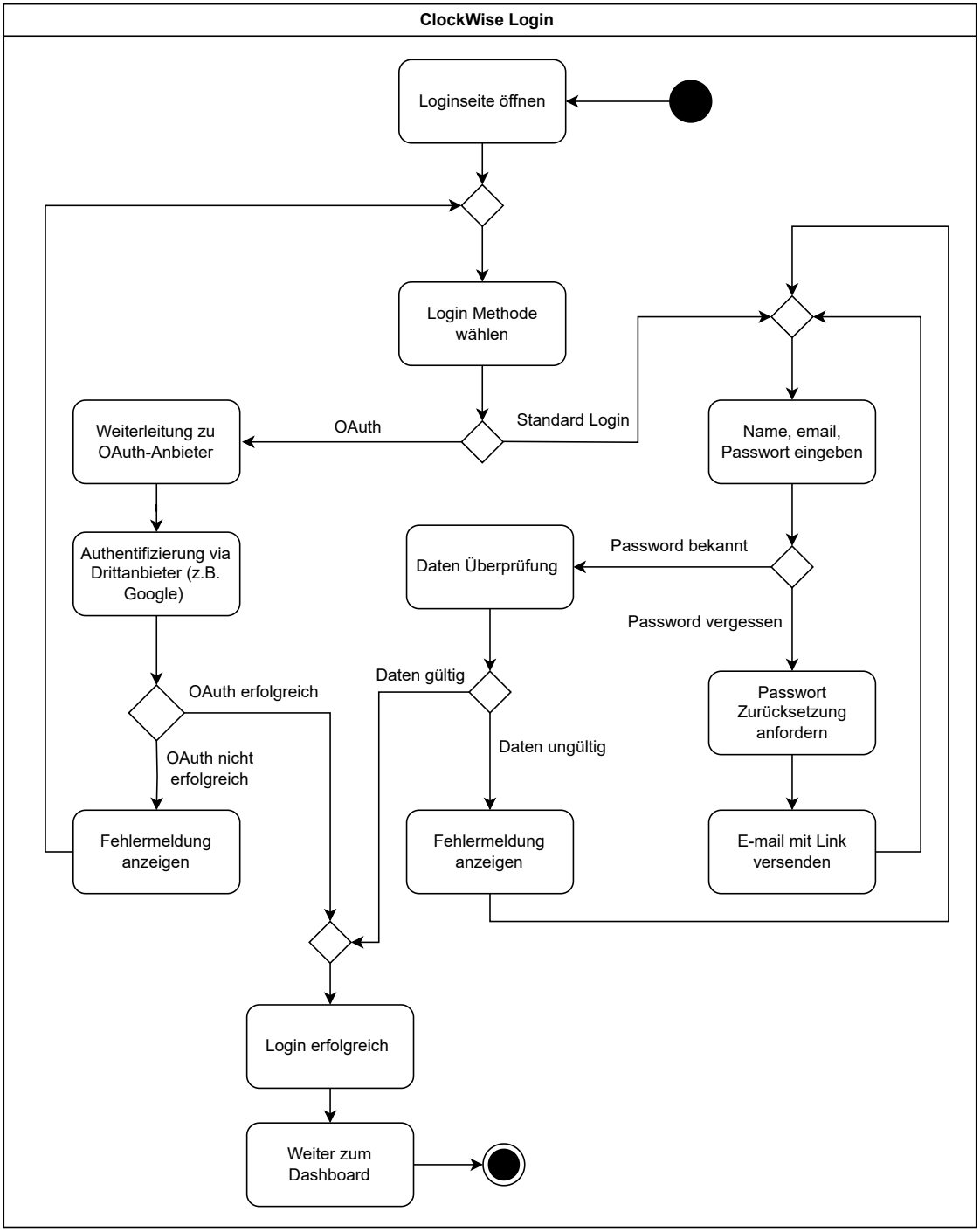

Abbildung 1.2: Aktivitätsdiagramm F20: Login eines Nutzers

1.2.2 Zeiterfassung starten und bearbeiten

Die zentrale Funktion von ClockWise ist die projektbezogene Erfassung von Zeit. Der:Die Nutzer:in kann entweder eine laufende Erfassung über eine Start/Stopp-Funktion aktivieren oder bestehende Zeiteinträge manuell hinzufügen und bearbeiten. Jeder Zeiteintrag wird einem spezifischen Projekt zugeordnet und kann bei Bedarf nachträglich angepasst werden.

Die Anwendung stellt sicher, dass laufende Zeiteinträge eindeutig sind, sodass nicht mehrere Aktivitäten gleichzeitig erfasst werden können. Durch die intuitive Benutzeroberfläche kann die Zeitdokumentation flexibel an den individuellen Arbeitsstil angepasst werden – etwa bei kontinuierlichem Arbeiten oder fragmentierten Aufgabenblöcken.

Der vollständige Ablauf der Zeiterfassung inklusive Start, Pause, Stoppen und Speichern wird in Abbildung 1.3 dargestellt.

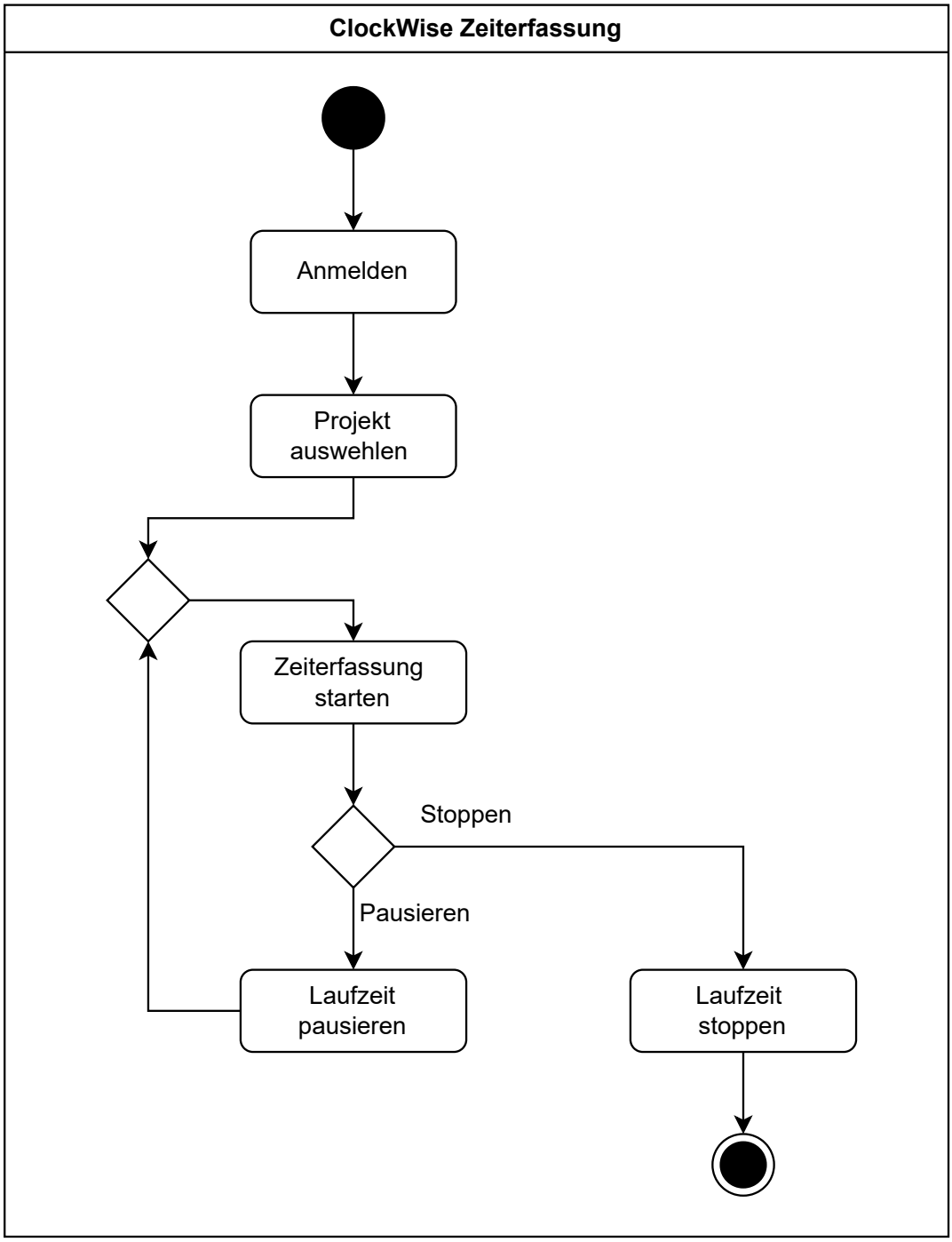

Abbildung 1.3: Aktivitätsdiagramm 3.3 des Pflichtenhefts F30 und F40: Zeiterfassung

1.2.3 Projekte und Aufgaben verwalten

ClockWise ermöglicht es Nutzer:innen, Projekte anzulegen, Aufgaben zu definieren und diese individuell oder innerhalb eines Teams zu strukturieren. Jedes Projekt dient als übergeordnete Einheit zur Sammlung von Zeiteinträgen, die thematisch oder zeitlich zusammengehören. Innerhalb eines Projekts können konkrete Aufgaben erstellt werden, um Arbeitsabschnitte feiner zu unterteilen.

Die Organisation in Projekte und Aufgaben erlaubt eine gezielte Planung und Nachverfolgung von Tätigkeiten. Zeiteinträge lassen sich eindeutig zuordnen, sodass die spätere Analyse nicht nur auf Projektebene, sondern auch auf Task Ebene erfolgen kann.

Beim Erstellen oder Bearbeiten von Projekten und Aufgaben erfolgt zunächst eine Validierung der eingegebenen Daten. Danach werden die Informationen in der Datenbank gespeichert. Aufgaben können optional einem bestimmten Teammitglied zugewiesen werden, wobei nur Projektverantwortliche entsprechende Berechtigungen besitzen. Neben der Erstellung sind auch das Aktualisieren oder Löschen von Aufgaben möglich. Dieser gesamte Ablauf ist in Abbildung 1.4 dargestellt.

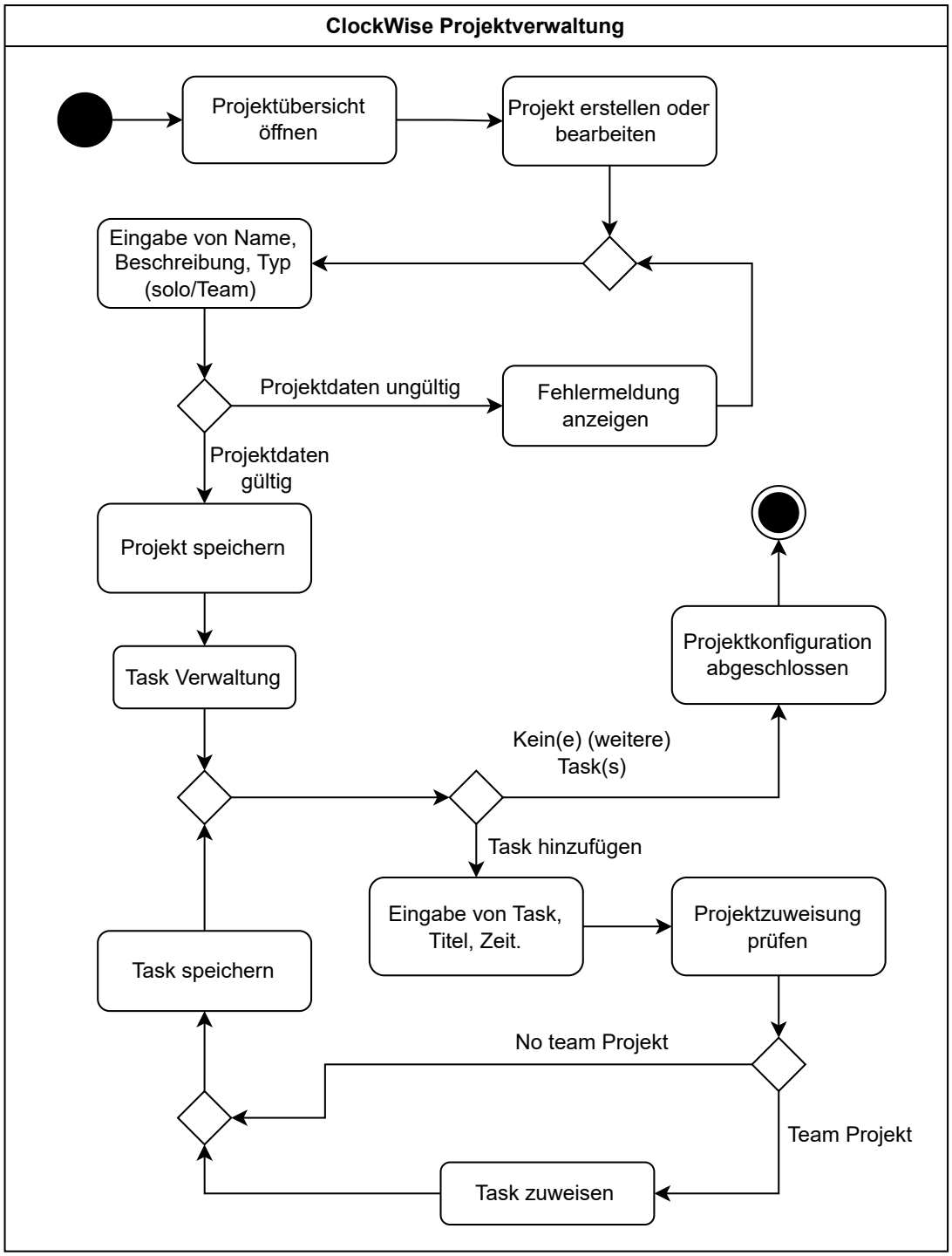

Abbildung 1.4: Aktivitätsdiagramm F50 und F60: Projektverwaltung

1.2.4 Analyse und Kalenderansicht

Zur Unterstützung der Selbstorganisation stellt ClockWise verschiedene Möglichkeiten zur Auswertung der erfassten Zeitdaten bereit. Nutzer:innen können ihre Zeitverteilung in Form von Diagrammen visualisieren oder über eine Kalenderansicht nachvollziehen, wann welche Aktivitäten stattgefunden haben.

Die Kalenderansicht zeigt alle Zeiteinträge tagesweise strukturiert und erleichtert so die zeitliche Orientierung. Zusätzlich werden projektbezogene Übersichten generiert, die eine Bewertung des Fortschritts im Vergleich zu vorher definierten Zielzeiten ermöglichen. Ein Fortschrittsbalken visualisiert dabei das Verhältnis von geleisteter zu geplanter Zeit.

Diese Funktionen dienen nicht nur der Reflexion, sondern auch der Motivation, indem sie individuelle Arbeitsmuster sichtbar machen und zur besseren Zeitplanung beitragen.

Da es sich bei den Analyse und Kalenderfunktionen nicht um klassische Prozessabläufe handelt, sondern um die Darstellung bereits gespeicherter Zeitdaten, wurde auf ein Aktivitätsdiagramm verzichtet. Die Nutzung dieser Funktionen erfolgt primär lesend und datengetrieben, sodass eine modellbasierte Visualisierung hier keinen zusätzlichen Erkenntnisgewinn bietet.

1.2.5 Teamfunktion und Rollen

ClockWise bietet eine integrierte Teamfunktion, die es mehreren Nutzer:innen ermöglicht, gemeinsam an Projekten zu arbeiten. Dabei können Teammitglieder einzelnen Projekten zugeordnet und unterschiedliche Rollen vergeben werden, z. B. zur Unterscheidung zwischen Administrator:innen und regulären Mitgliedern.

Administrator:innen haben die Möglichkeit, Projekte zu erstellen, Teammitglieder zu verwalten und Rollen zu vergeben. Reguläre Mitglieder können Zeiteinträge hinzufügen und bestehende Aufgaben bearbeiten, jedoch keine grundlegenden Strukturen verändern. Diese Rollenverteilung schafft klare Zuständigkeiten und vereinfacht die Koordination innerhalb des Teams.

Zusätzlich wird die Gesamtzeitverteilung im Team sichtbar gemacht, wodurch eine transparente Aufgabenverteilung gefördert und Überlastungen frühzeitig erkannt werden können.

Die Teamfunktion umfasst mehrere kontextabhängige Teilprozesse wie die Verwaltung von Mitgliedern, die Vergabe von Rollen und die Zuweisung von Aufgaben. Da diese Prozesse stark von der jeweiligen Nutzerrolle (z. B. Administrator:in oder reguläres Mitglied) und dem aktuellen Projektstatus abhängen, wurde auf ein einzelnes Aktivitätsdiagramm verzichtet. Eine schematische Darstellung hätte entweder zu einer unzureichenden Generalisierung oder zu einer übermäßig komplexen Visualisierung geführt. Stattdessen wurde der Ablauf verbal differenziert beschrieben.

1.3 Systemübersicht

Die folgende Tabelle bietet einen systematischen Überblick über die funktionalen Hauptbereiche der Anwendung ClockWise. Für jede Funktion wird die technische Umsetzung kurz beschrieben und die beteiligten Systemkomponenten benannt.

Funktion	Technische Umsetzung	Komponenten
Zeiterfassung (F30, F40)	Start/Stopp oder manuelle Eingabe, ein aktiver Eintrag pro Nutzer:in. Dauerberechnung automatisch. Untitled Task bei leerem Taskfeld. Pausieren/Fortsetzen möglich.	C10 (UI), C20 (Logik), C30 (TimeEntry Modell)
Projektverwaltung (F50)	CRUD Funktionalität mit Zeitlimits, Deadlines, Projekttypen, Teamzuweisung und optionalen Kreditpunkten. Validierung erfolgt serverseitig.	C10 (Projektformular), C20 (API Logik), C30 (Project Modell)
Taskverwaltung (F60)	Aufgaben können Projekten zugewiesen oder unabhängig davon erstellt werden. Sie beinhalten Titel, Beschreibung, Kategorie und Fälligkeitsdatum. Bei leerem Titel wird „Untitled Task“ gesetzt.	C10 (Taskformular), C20 (API F60), C30 (Task Modell)
Exportfunktion (F70)	Export der erfassten Zeitdaten als CSV Datei. Export erfolgt projektbezogen und beinhaltet Zeit, Dauer, Nutzer:in, Task und optional Teamrolle.	C10 (Downloadbutton), C20 (Routen F70), C30 (TimeEntry Modell)
Analyse (F80, F90)	Auswertung der erfassten Zeiten pro Tag und Projekt. Beinhaltet Wochenübersichten, Fortschrittsbalken, Kalenderansicht sowie den Vergleich zwischen Soll- und Ist-Zeit pro Projekt.	C10 (Analyse UI), C20 (Funktionen F80/F90)
Benachrichtigungen (Teil von F90)	Automatische Hinweise bei systemrelevanten Ereignissen. Speicherung in der Datenbank, Filterung nach Nutzer:in, Anzeige nach Erstellungszeit. Möglichkeit zum Löschen durch Nutzer:in.	C10 (Benachrichtigungsanzeige), C20 (Trigger), C30 (Notification Modell), C50
Teamfunktion (F100)	Erstellung von Teams, Rollenvergabe (Admin/Member), Mitgliederverwaltung (Hinzufügen/Entfernen), Rechteprüfung durch Authentifizierung. Benachrichtigung bei Teamerstellung.	C10 (Team-UI), C20 (Logik F100), C30 (Team-, UserTeam-Modell), C40
Authentifizierung (F10, F20)	Registrierung, Login, OAuth-Anmeldung, Passwort Zurücksetzen, Sessionverwaltung.	C10 (Formulare), C20 (Flask Auth), C30 (User Modell)

Tabelle 1.1: Systemübersicht der Kernfunktionen von ClockWise

2 Analyse der Produktfunktionen

In diesem Kapitel wird das Verhalten der im Pflichtenheft definierten Produktfunktionen untersucht und in Form von Sequenzdiagrammen veranschaulicht. Ziel ist es, ein besseres Verständnis über das Zusammenspiel der beteiligten Systemkomponenten zu gewinnen und aufzuzeigen, wie bestimmte Abläufe deren Verhalten und Funktionalität beeinflussen.

2.1 Analyse von Funktionalität <F10>: Registrierung

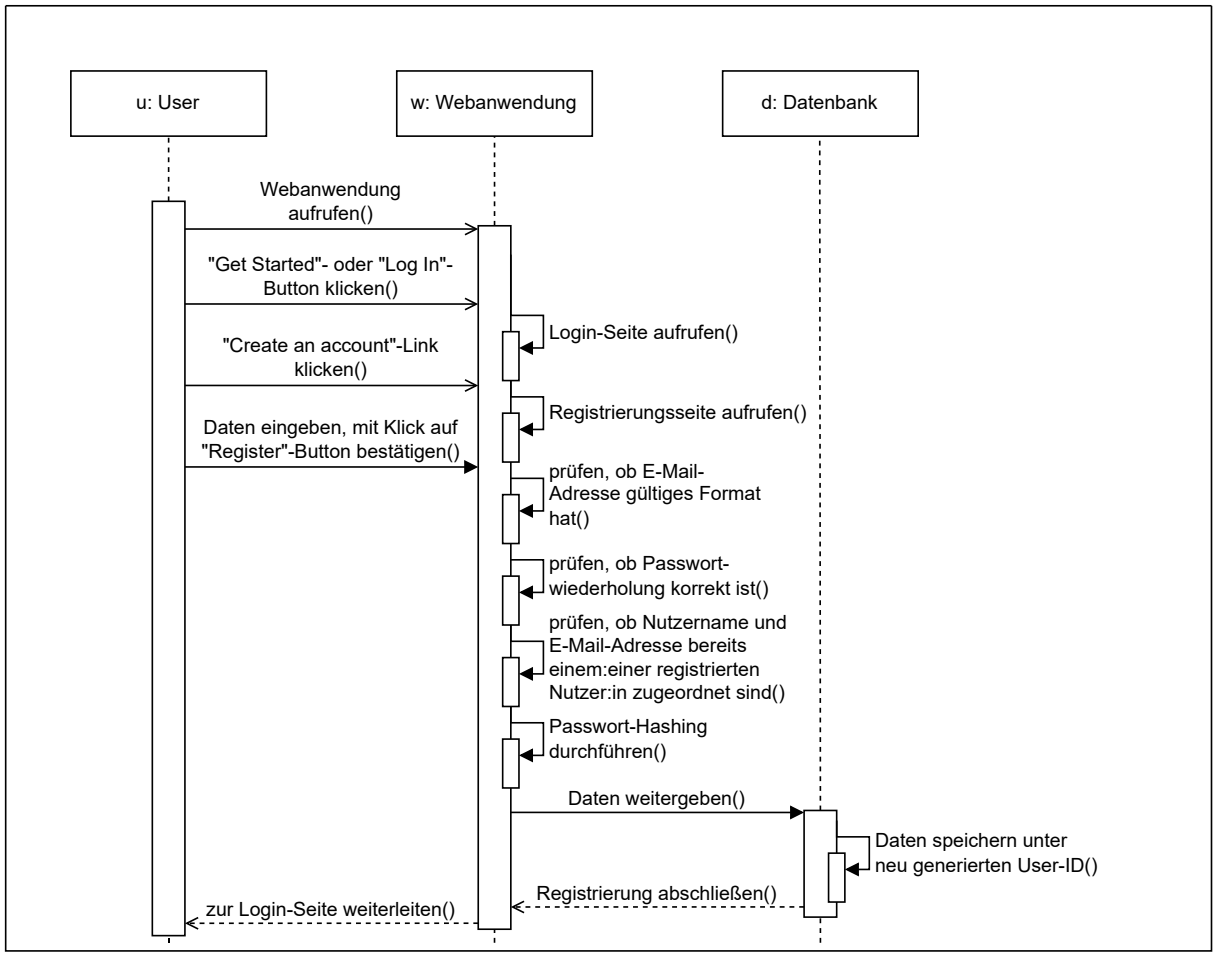

Abbildung 2.1: Sequenzdiagramm F10: Registrierung

Im Sequenzdiagramm zu der Funktion <F10> wird der Prozess der Erstellung eines Nutzerkontos zur erstmaligen Nutzung der Webanwendung ClockWise dargestellt. Zu Beginn wird die Webanwendung durch das Ausführen von app.py (lokal auf dem Endgerät) aufgerufen. Durch einen Klick auf den „Get Started“- oder „Log In“-Button gelangt der:die Nutzer:in zur Login-Seite, von wo aus über den Link „Create an account“ die Weiterleitung zur Registrierungsseite erfolgt. Dort gibt der:die Nutzer:in die für die Kontoerstellung erforderlichen Daten ein: Vorname, Nachname, Nutzername, E-Mail-Adresse, Passwort und Passwortwiederholung. Optional kann ein Profilbild ausgewählt werden.

Die Übermittlung der Daten an die Webanwendung erfolgt nach einem Klick auf den „Register“-Button. Die Anwendung prüft zunächst, ob das Format der E-Mail-Adresse gültig ist und ob die Passwortwiederholung korrekt ist. Wird dabei festgestellt, dass das E-Mail-Format oder die Passwortwiederholung inkorrekt sind, wird dem:der Nutzer:in eine entsprechende Fehlermeldung

angezeigt. Zusätzlich wird überprüft, ob der Nutzername oder die E-Mail-Adresse bereits einem:einer registrierten Nutzer:in zugeordnet sind, was ebenfalls die Ausgabe einer Fehlermeldung bewirken würde. Auf die Darstellung der Ausgabe der Fehlermeldungen an den:die Nutzer:in im Falle eines festgestellten Fehlers bei einer der Überprüfungen wird in diesem Diagramm aus Platzgründen verzichtet. Anschließend erfolgt das Hashing des Passworts und dann werden die vollständigen und validierten Daten an die Datenbank weitergegeben. In der Datenbank werden daraufhin die Nutzerdaten unter einer neu generierten User-ID gespeichert. Nach erfolgreicher Registrierung wird der:die Nutzer:in auf die Login-Seite weitergeleitet.

2.2 Analyse von Funktionalität <F20>: Login

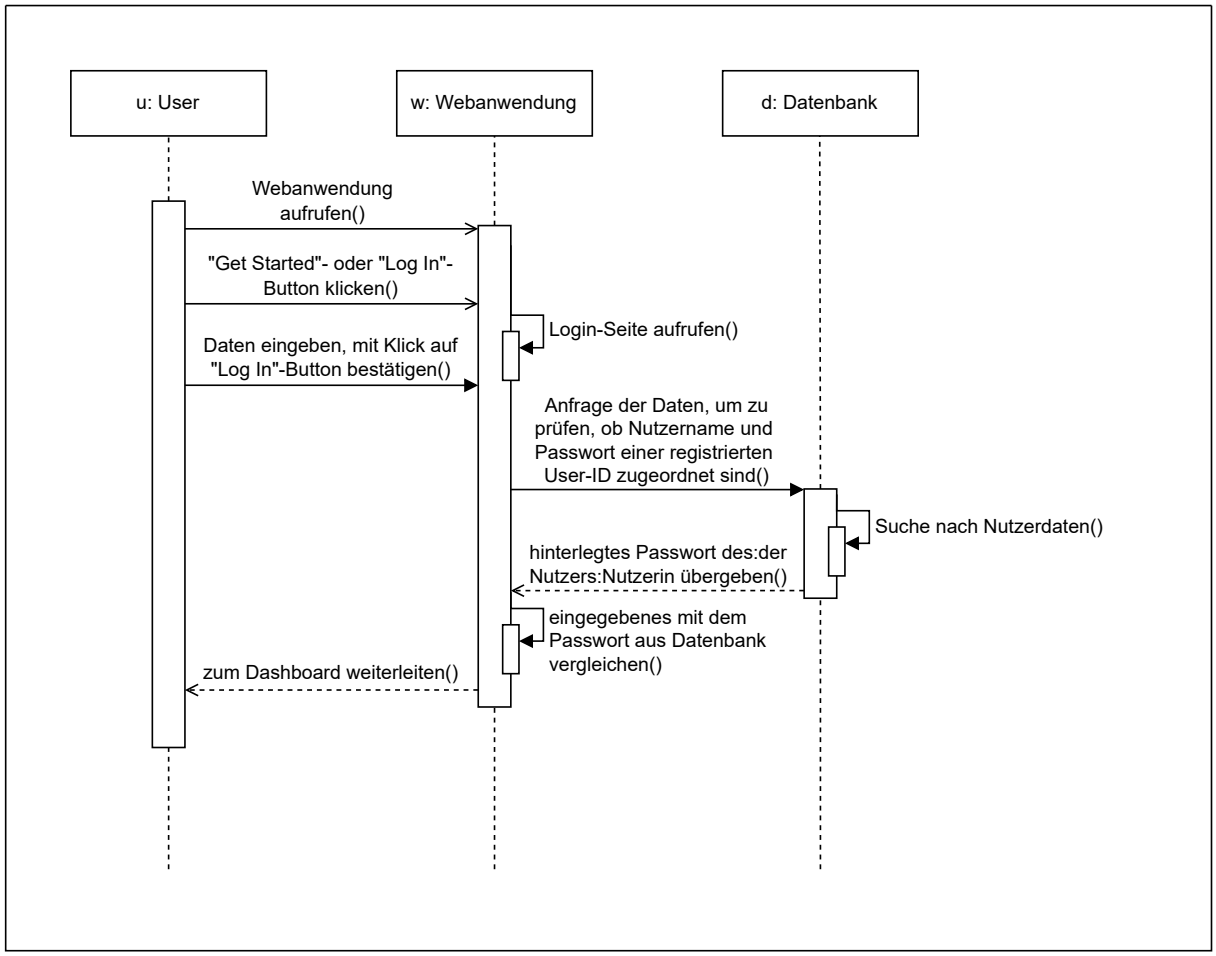

Abbildung 2.2: Sequenzdiagramm F20: Login

Im Sequenzdiagramm zur Funktion <F20> wird der Prozess der Anmeldung mit bereits bestehenden Zugangsdaten dargestellt. Nach dem Aufrufen der Webanwendung gelangt der:die Nutzer:in durch einen Klick auf den „Get Started“- oder „Log In“-Button zur Login-Seite. Dort gibt der:die Nutzer:in die für die Anmeldung erforderlichen Daten ein: Nutzername und Passwort. Mit einem Klick auf den „Log In“-Button werden die Daten an die Webanwendung übermittelt, die anschließend eine Datenbankabfrage durchführt, um zu prüfen, ob diese Daten einer bereits registrierten Nutzer-ID zugeordnet sind. Sind die Daten nicht in der Datenbank vorhanden, erhält der:die Nutzer:in eine Fehlermeldung, dass die eingegebenen Anmeldedaten inkorrekt sind. Werden die Daten gefunden, übermittelt die Datenbank das gespeicherte Passwort an die Webanwendung, die das eingegebene Passwort mit dem gespeicherten vergleicht. Stimmen beide überein, wird die Anmeldung erfolgreich ausgeführt und der:die Nutzer:in zum persönlichen Dashboard weitergeleitet. Stimmen sie nicht überein, erhält der:die Nutzer:in eine Fehlermeldung und der Zugriff wird verweigert. Auf die Darstellung der Fehlermeldungen, die dem:der

Nutzer:in im Falle eines auftretenden Fehlers beim Abrufen der Daten aus der Datenbank oder beim Passwortvergleich angezeigt werden, wird in diesem Diagramm aus Platzgründen verzichtet.

2.3 Analyse von Funktionalität <F30>: Zeiterfassung (Start/Stop)

2.3.1 Funktionalität <F30.1>: Zeiterfassung neu angelegter Aufgaben

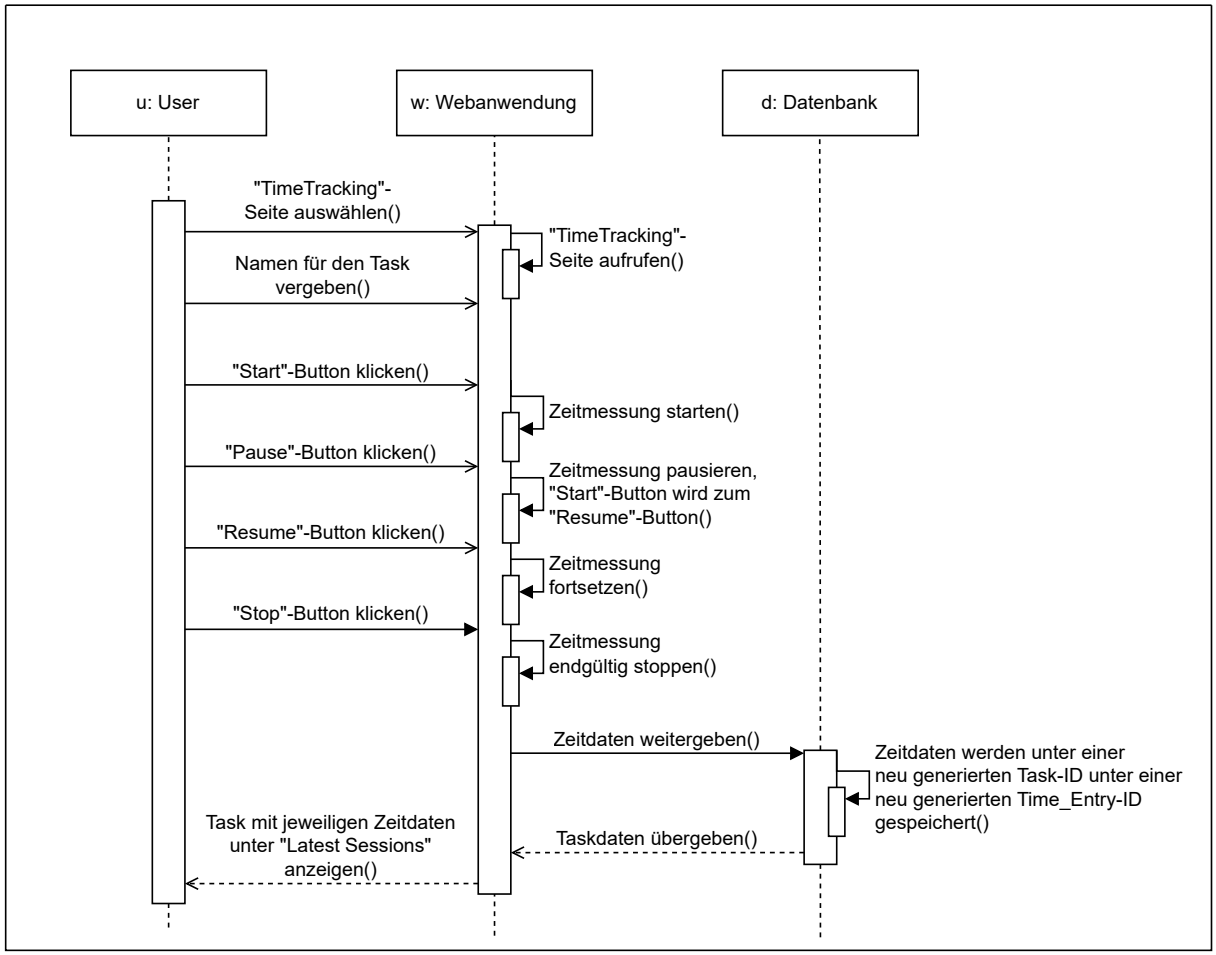

Abbildung 2.3: Sequenzdiagramm F30.1: Zeiterfassung (Start/Stop) neu angelegter Aufgaben

Im Sequenzdiagramm zu der Funktion <F30.1> wird der Prozess des Erfassens von Arbeitszeiten neu angelegter Aufgaben durch Starten, Pausieren und Stoppen eines Timers dargestellt. Auf der „TimeTracking“-Seite kann ein Name für den neuen Task eingegeben werden. Durch einen Klick auf den „Start“-Button wird die Zeitmessung begonnen, durch „Pause“ unterbrochen. Wird die Zeitmessung pausiert, erscheint anstelle des „Pause“-Buttons der „Resume“-Button, mit dem die Zeitmessung fortgesetzt werden kann.

Ein Klick auf den „Stop“-Button beendet die Zeitmessung endgültig. Die erfassten Daten werden anschließend an die Datenbank übermittelt und dort unter einer neu generierten Task-ID sowie unter einer neu generierten Time_Entry-ID gespeichert. Nach erfolgreicher Speicherung werden

die Daten an die Webanwendung zurückgegeben, sodass die aktualisierte Benutzeroberfläche den neuen Task auf der „TimeTracking“-Seite unter „Latest Sessions“ anzeigen kann.

2.3.2 Funktionalität <F30.2>: Zeiterfassung bereits existierender Aufgaben

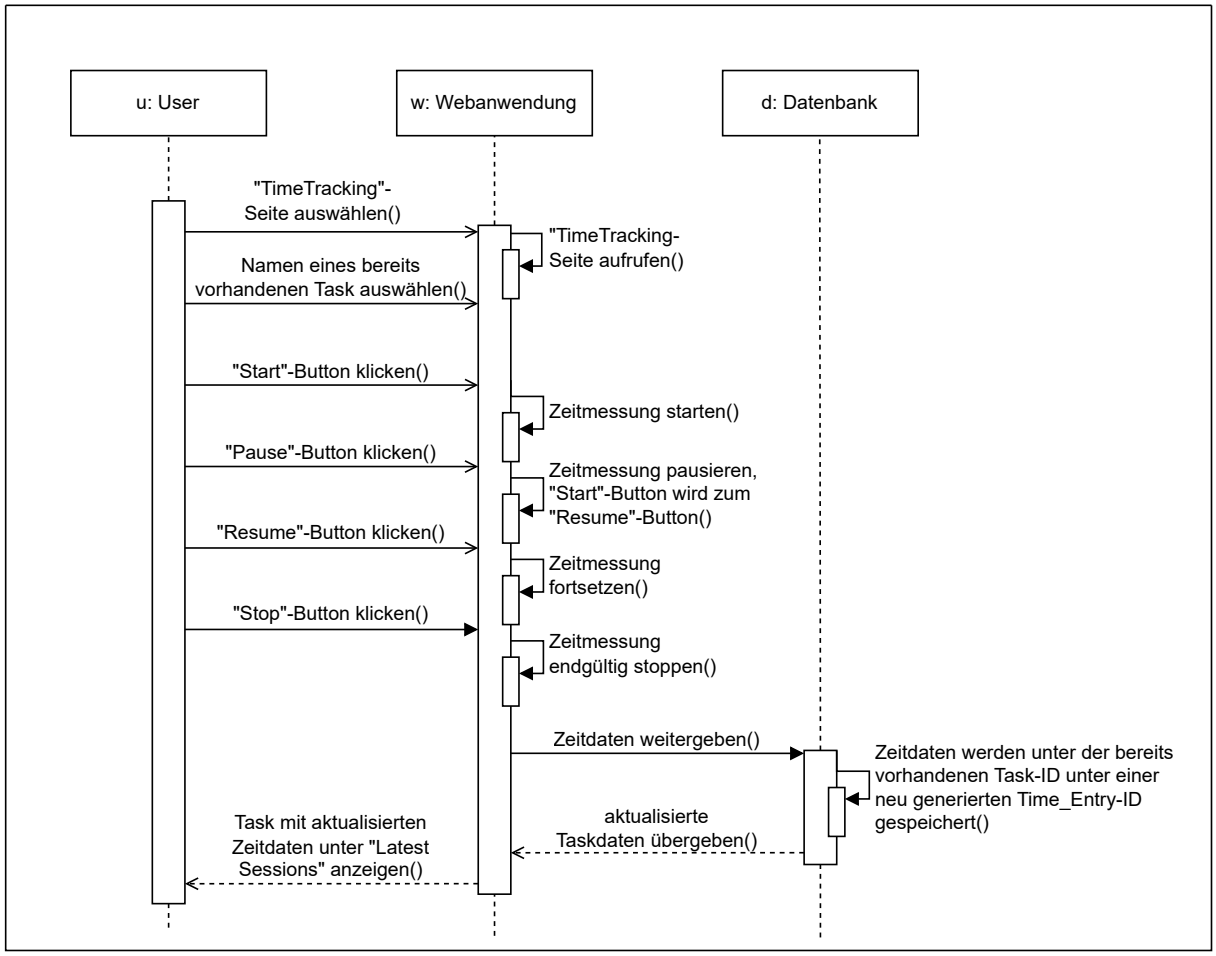

Abbildung 2.4: Sequenzdiagramm F30.2: Zeiterfassung (Start/Stop) bereits existierender Aufgaben

Im Sequenzdiagramm zu der Funktion <F30.2> wird der Prozess des Erfassens von Arbeitszeiten bereits existierender Aufgaben durch Starten, Pausieren und Stoppen eines Timers dargestellt. Auf der „TimeTracking“-Seite kann ein bereits angelegter Task ausgewählt werden. Die Zeitmessung erfolgt genauso wie bei neu angelegten Aufgaben.

Die erfassten Daten werden anschließend an die Datenbank übermittelt, wo der Task anhand seiner Task-ID aufgerufen wird und der neue Zeiteintrag unter einer neu generierten Time_Entry-ID gespeichert wird. Nach erfolgreicher Speicherung werden die Daten an die Webanwendung zurückgegeben, sodass die aktualisierte Benutzeroberfläche den überarbeiteten Task auf der „TimeTracking“-Seite unter „Latest Sessions“ anzeigen kann.

2.4 Analyse von Funktionalität <F40>: Zeiteinträge bearbeiten

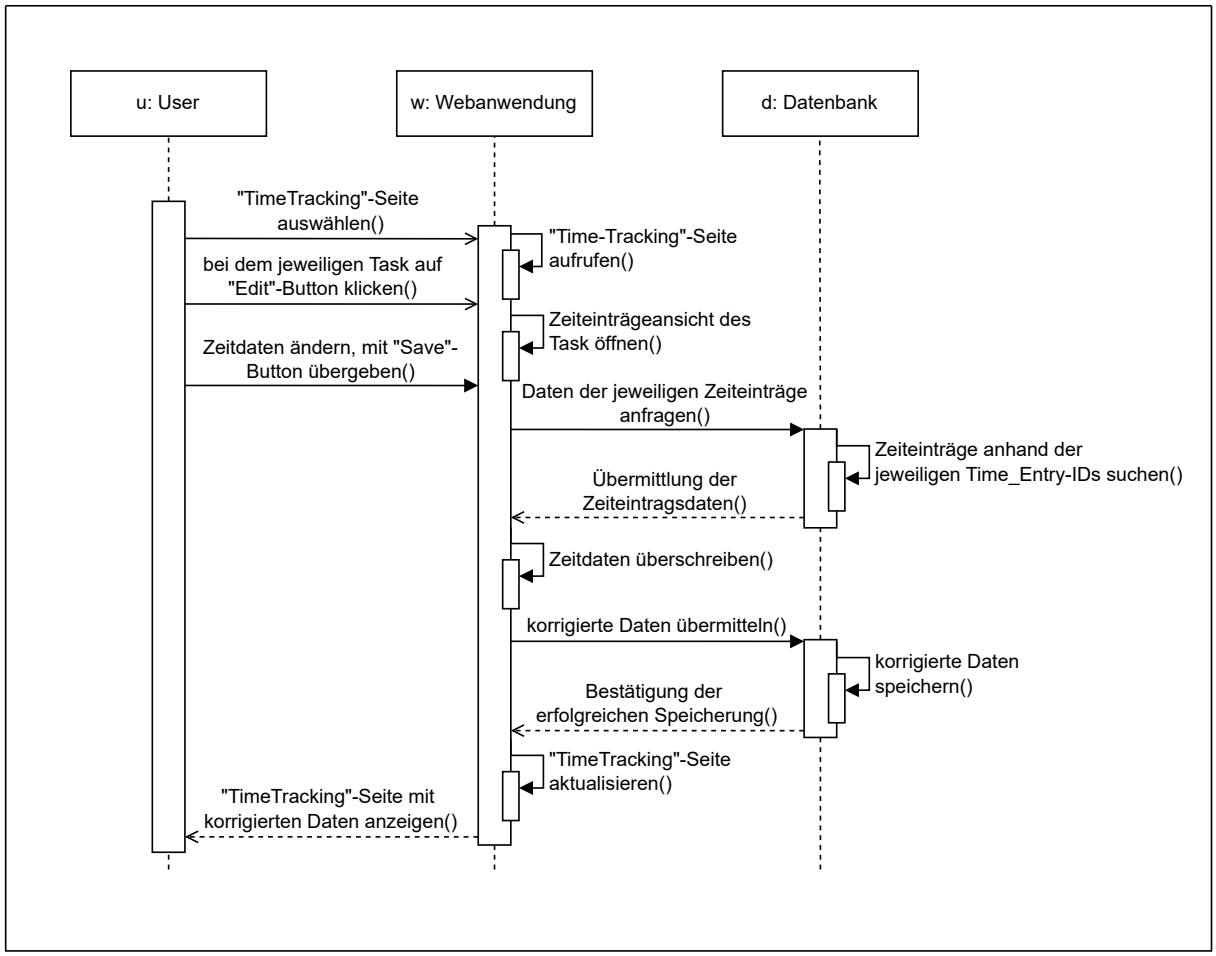

Abbildung 2.5: Sequenzdiagramm F40: Zeiteinträge bearbeiten

Im Sequenzdiagramm zur Funktion <F40> wird der Prozess der manuellen Anpassung oder Korrektur von bereits erfassten Zeitdaten durch den:die Nutzer:in dargestellt. Über die „TimeTracking“-Seite der Webanwendung wird zunächst eine Übersicht aller Tasks angezeigt. Der:Die Nutzer:in kann dort bei einem beliebigen Eintrag über den Button „Edit“ die Zeiteinträgeansicht öffnen. Dort lassen sich die erfassten Zeiteinträge des jeweiligen Tasks durch Veränderung des Datums sowie der Start- und Stopp-Zeit anpassen und neue Zeiteinträge erstellen. Über den „Save“-Button werden diese Änderungen übermittelt.

Die Webanwendung stellt daraufhin eine Anfrage der Zeitdaten der Zeiteinträge, an denen Änderungen vorgenommen werden sollen. Wenn die Datenbank diese Daten anhand der Time_Entry-IDs gefunden hat, übermittelt sie sie an die Webanwendung, die die Daten überschreibt. Die aktualisierten Daten werden anschließend zur Speicherung erneut an die Datenbank übergeben, welche bei erfolgreicher Speicherung eine Bestätigung an die Webanwendung zurückgibt.

Die „TimeTracking“-Seite wird daraufhin aktualisiert und dem:der Nutzer:in mit den korrigierten Daten angezeigt. Sollte es während der Speicherung oder beim Laden der aktualisierten Daten zu Fehlern kommen, werden entsprechende Fehlermeldungen ausgegeben. Auf deren Darstellung im Diagramm wird aus Platzgründen verzichtet.

2.5 Analyse von Funktionalität <F50>: Projektverwaltung

2.5.1 Funktionalität <F50.1>: Projekt erstellen

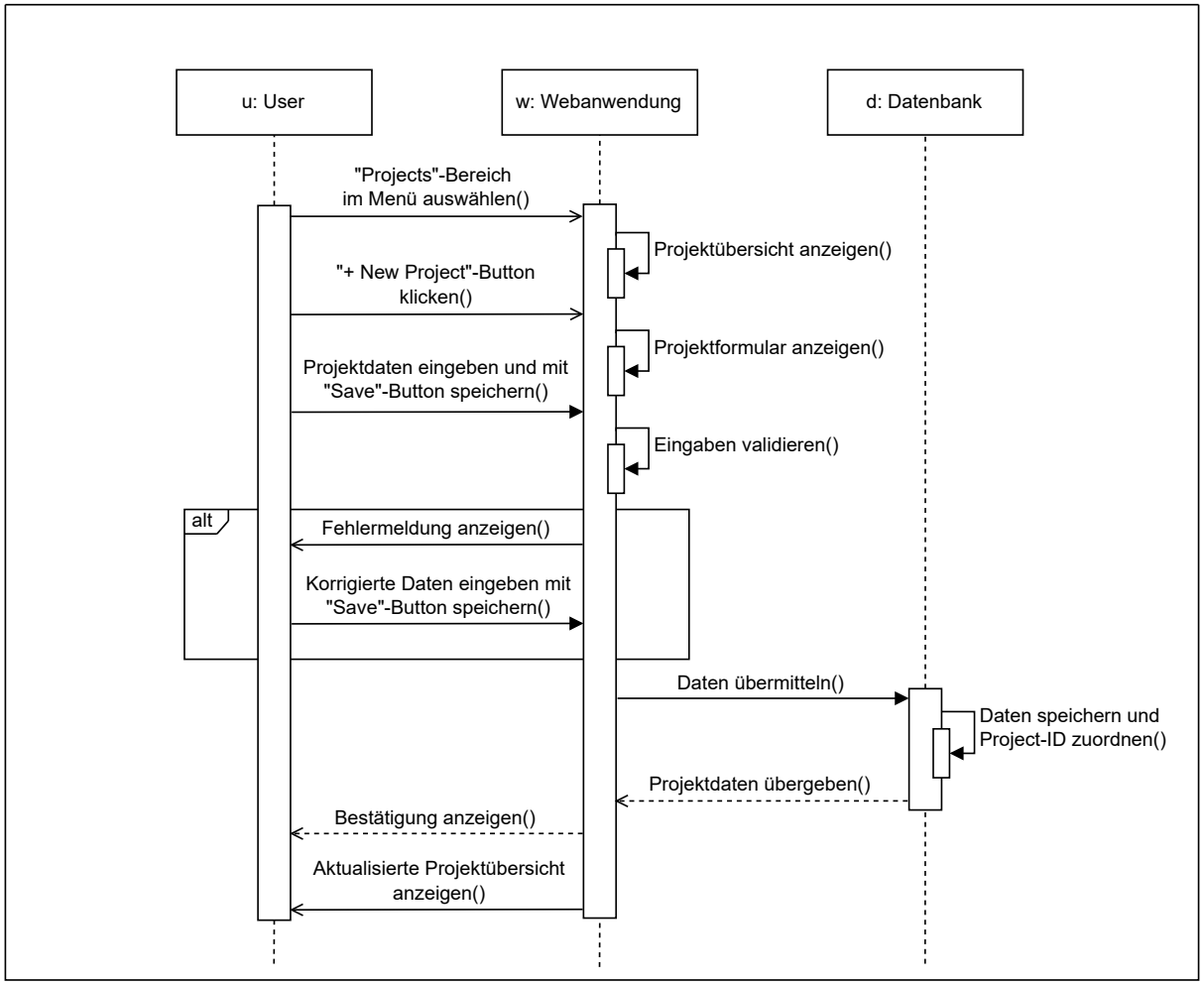

Abbildung 2.6: Sequenzdiagramm F50.1: Projekt erstellen

Die Funktion F50: Projektverwaltung wurde zur besseren Übersichtlichkeit in drei Teilfunktionen unterteilt: Projekte erstellen, Projekte bearbeiten und Projekte löschen.

Das Sequenzdiagramm zu F50.1 zeigt den Ablauf zur Erstellung eines neuen Projekts durch den:die Nutzer:in.

Nach dem Öffnen des Bereichs „Projects“ in der Webanwendung wird die Projektübersicht angezeigt. Über den Button „+ New Project“ kann ein Formular aufgerufen werden, in dem Projektdaten eingegeben werden. Nach dem Speichern werden die Eingaben zunächst von der Anwendung validiert.

Falls die Eingaben ungültig sind, zeigt die Anwendung eine Fehlermeldung an und der:die Nutzer:in kann die Daten korrigieren und erneut speichern (alt-Block). Sind die Eingaben gültig, werden die Projektdaten an die Datenbank übermittelt, dort gespeichert und mit einer eindeutigen ID versehen.

Anschließend zeigt die Anwendung eine Bestätigung und aktualisiert die Projektübersicht.

2.5.2 Analyse von Funktionalität <F50.2>: Projekt bearbeiten

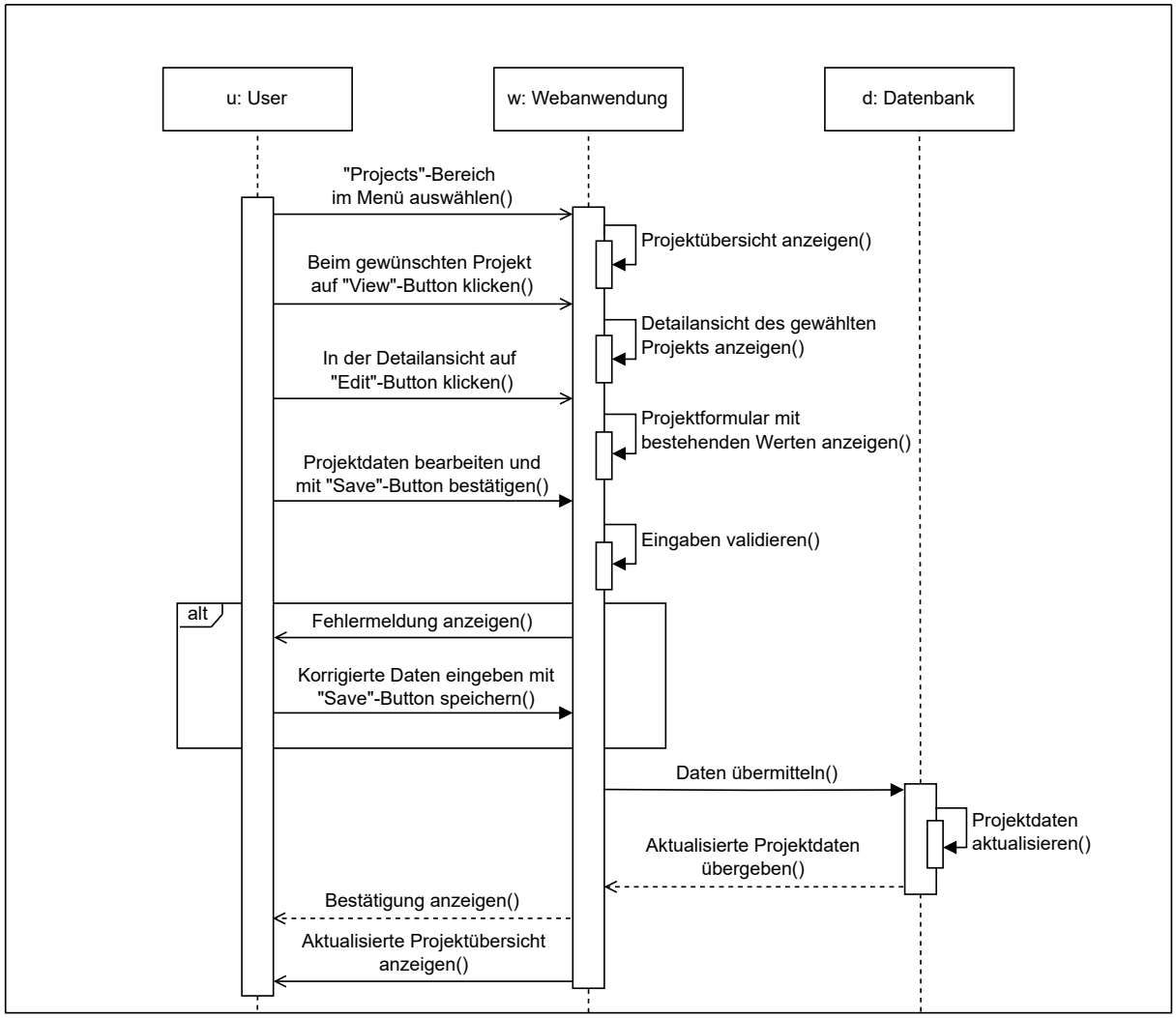

Abbildung 2.7: Sequenzdiagramm F50.2: Projekt bearbeiten

Das Sequenzdiagramm zu F50.2 zeigt den Ablauf zur Bearbeitung eines bestehenden Projekts durch den:die Nutzer:in..

In der Projektübersicht wird das gewünschte Projekt ausgewählt und über den „View“-Button geöffnet. In der Detailansicht kann es über den „Edit“-Button zur Bearbeitung aufgerufen werden. Die Webanwendung zeigt ein Formular mit den vorhandenen Projektdaten an, die angepasst werden können. Nach dem Klick auf den „Save“-Button werden die Eingaben zunächst validiert.

Falls dabei ungültige Eingaben erkannt werden, zeigt die Anwendung eine Fehlermeldung an. Der:die Nutzer:in kann daraufhin die Daten korrigieren und den Speichervorgang erneut auslö-

sen (alt-Block). Sind die Eingaben gültig, werden die überarbeiteten Daten an die Datenbank übermittelt und dort gespeichert.

Anschließend zeigt die Webanwendung eine Bestätigung sowie die aktualisierte Projektübersicht.

2.5.3 Analyse von Funktionalität <F50.3>: Projekt löschen

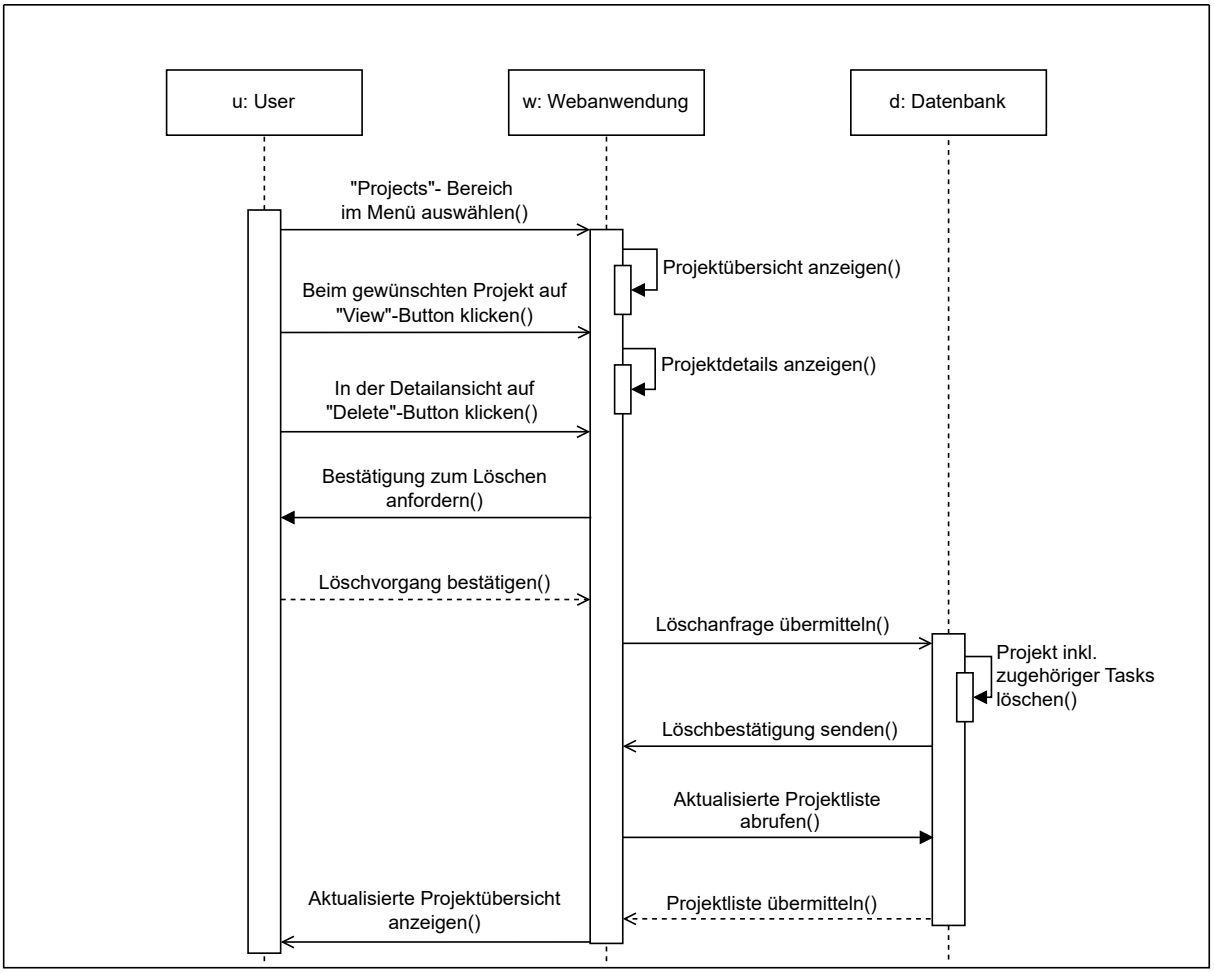

Abbildung 2.8: Sequenzdiagramm F50.3: Projekt löschen

Das Sequenzdiagramm zu F50.3 zeigt den Ablauf zum Löschen eines Projekts durch den:die Nutzer:in.

Nach Auswahl des gewünschten Projekts in der Übersicht wird in der Detailansicht der Löschvorgang über den „Delete“-Button gestartet. Die Anwendung fordert eine Bestätigung an, bevor die Löschanfrage an die Datenbank übermittelt wird. Dort werden das Projekt und die zugehörigen Tasks entfernt.

Abschließend wird die Projektübersicht aktualisiert angezeigt.

2.6 Analyse von Funktionalität <F60>: Taskverwaltung

2.6.1 Analyse von Funktionalität <F60.1>: Task erstellen auf der Projektseite

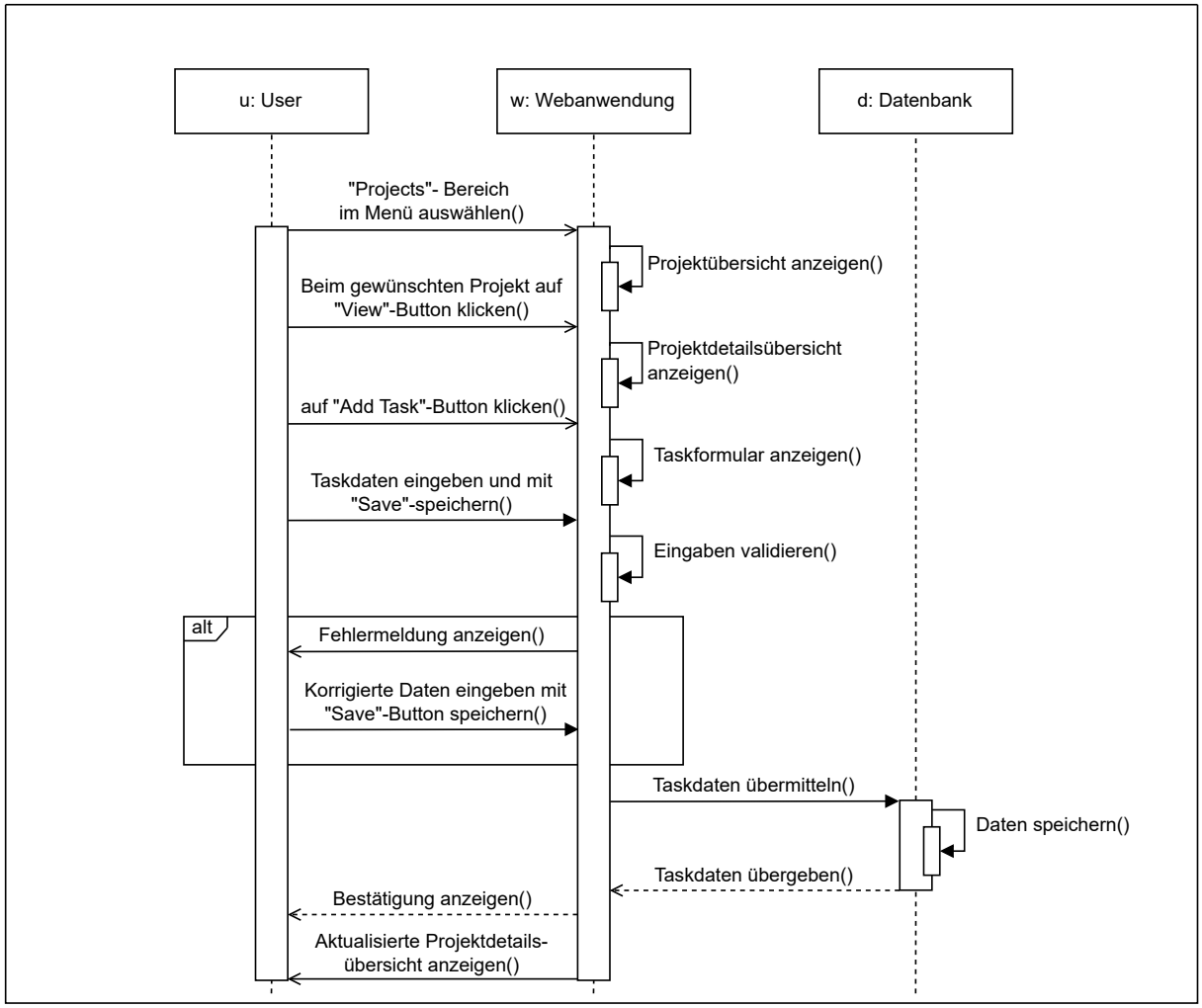

Abbildung 2.9: Sequenzdiagramm F60.1: Task erstellen auf der Projektseite

Die Funktion F60: Taskverwaltung wurde zur besseren Übersichtlichkeit in drei Teilfunktionen unterteilt: Tasks erstellen, Tasks bearbeiten und Tasks löschen.

Das Sequenzdiagramm zu F60.1 zeigt den Ablauf der Erstellung eines neuen Tasks innerhalb eines bestehenden Projekts durch den:die Nutzer:in.

Nach dem Öffnen des Bereichs „Projects“ klickt der:die Nutzer:in beim gewünschten Projekt auf „Add Task“. Ein Eingabeformular erscheint, in dem die Taskdaten (z.B. Name, Beschreibung, Kategorie, Fälligkeitsdatum) eingegeben und mit „Save“ bestätigt werden. Nach dem Speichern werden die Eingaben zunächst von der Anwendung validiert.

Wenn die Eingaben ungültig sind, zeigt die Anwendung eine Fehlermeldung an. Der:die Nutzer:in kann die Daten daraufhin korrigieren und den Speichervorgang erneut auslösen (alt-Block). Bei gültigen Eingaben werden die Projektdaten an die Datenbank übermittelt, dort gespeichert und mit einer eindeutigen ID versehen.

Anschließend zeigt die Anwendung eine Bestätigung sowie die aktualisierte Projektübersicht.

2.6.2 Analyse von Funktionalität <F60.2>: Task bearbeiten

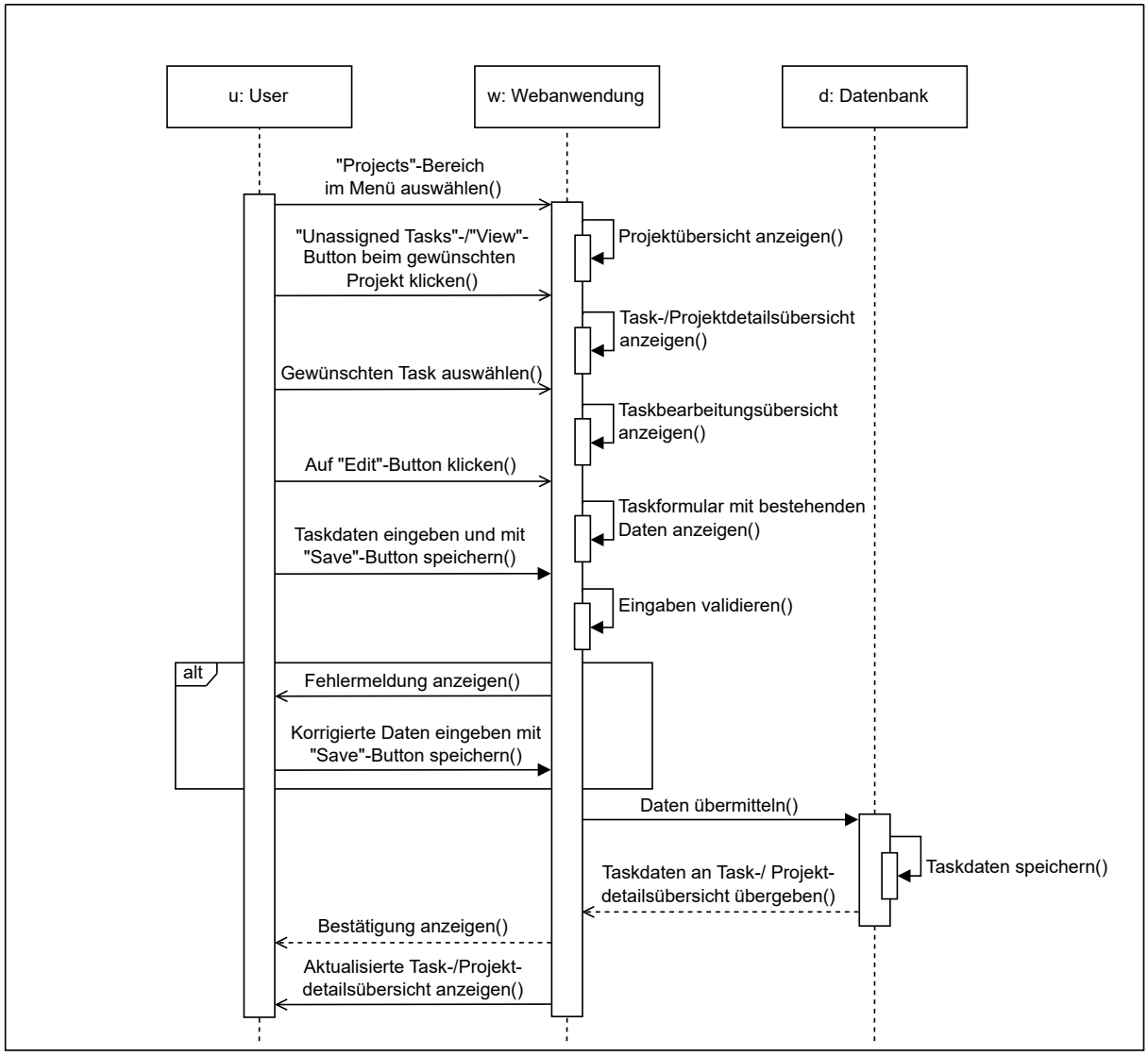

Abbildung 2.10: Sequenzdiagramm F60.2: Task bearbeiten

Das Sequenzdiagramm zu F60.2 beschreibt den Ablauf der Bearbeitung eines bestehenden Tasks durch den:die Nutzer:in.

Die Bearbeitung von Tasks erfolgt ausschließlich über die Projektseite im Bereich „Projects“ der Webanwendung. Der:die Nutzer:in wählt entweder einen nicht zugeordneten Task aus der Übersicht oder öffnet die Detailansicht eines Projekts über den „View“-Button, um einen zugehörigen Task zu bearbeiten.

Nach dem Klick auf „Edit“ erscheint ein Formular mit den vorhandenen Taskdaten, die angepasst werden können (z.B. Name, Beschreibung, Kategorie, Fälligkeitsdatum oder Projektzuordnung).

Nach dem Speichern werden die Eingaben validiert:

Wenn die Eingaben ungültig sind, zeigt die Anwendung eine Fehlermeldung an. Der:die Nutzer:in kann die Angaben daraufhin korrigieren und den Speichervorgang erneut auslösen (alt-Block). Bei gültigen Eingaben werden die Taskdaten an die Datenbank übermittelt und gespeichert.

Wenn eine Projektzuordnung geändert wurde, wird auch der zugehörige Projektdatensatz aktualisiert.

Anschließend wird die aktualisierte Task-/Projektdetailansicht angezeigt.

2.6.3 Analyse von Funktionalität <F60.3>: Task löschen

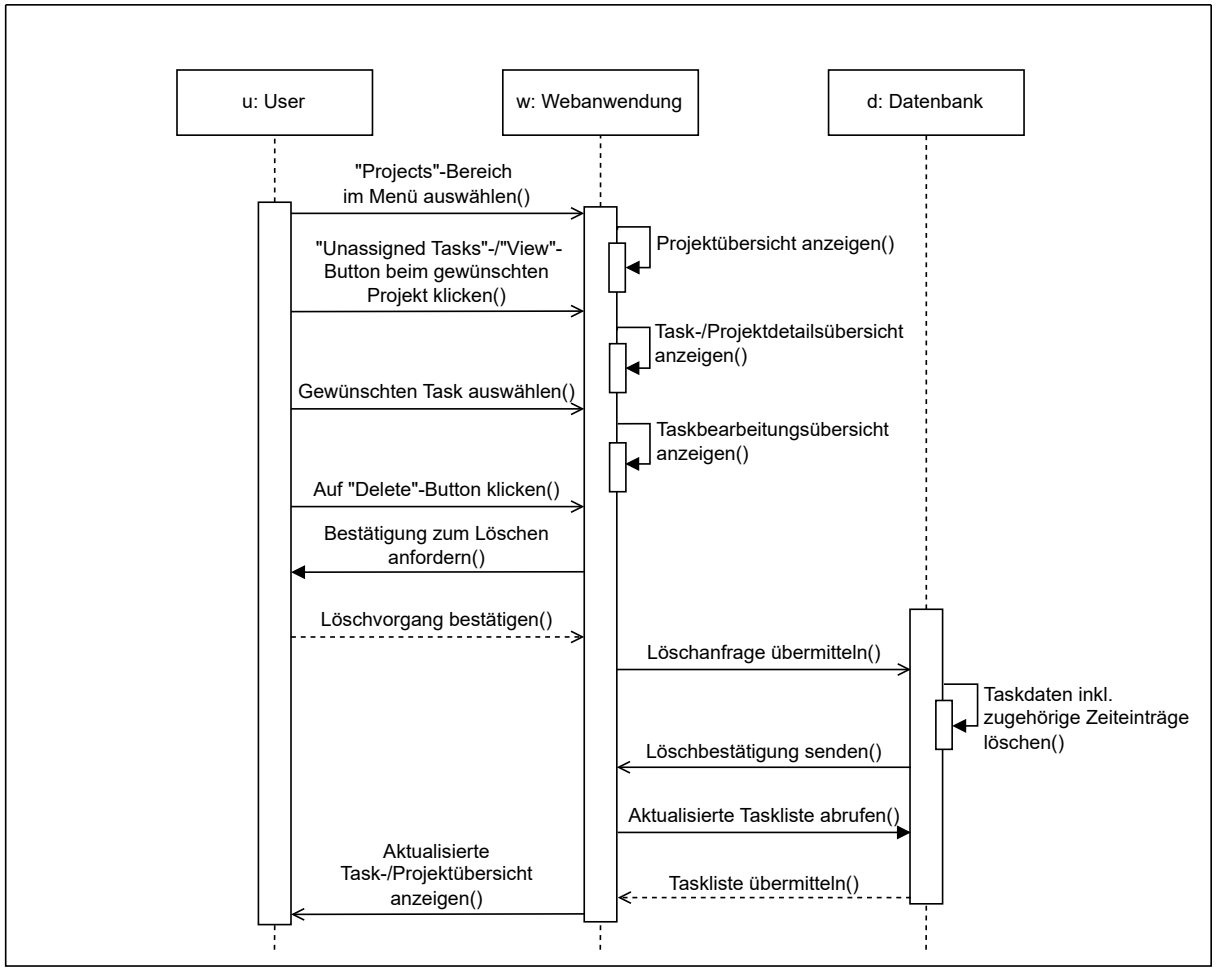

Abbildung 2.11: Sequenzdiagramm F60.3: Task löschen

Das Sequenzdiagramm zu F60.3 beschreibt den Ablauf des Löschens eines bestehenden Tasks durch den:die Nutzer:in.

Die Löschung erfolgt über die Projektseite. Nach dem Öffnen des Bereichs „Projects“ wird die Projektübersicht angezeigt. Dort hat der:die Nutzer:in zwei Möglichkeiten: Entweder wird der Task direkt im Abschnitt „Unassigned Tasks“ gefunden oder durch Klick auf den „View“-Button eines Projekts die zugehörige Projektdetailansicht geöffnet, in der ein projektbezogener Task ausgewählt werden kann. Beim gewünschten Task klickt der:die Nutzer:in auf „Delete“. Daraufhin wird eine Bestätigung angefordert. Nach Zustimmung übermittelt die Webanwendung die Löschanfrage an das Backend. Die Datenbank entfernt den Task sowie alle zugehörigen Zeiteinträge. Nach erfolgreicher Löschung wird eine Bestätigung gesendet, und die aktualisierte Taskliste wird angezeigt.

2.7 Analyse von Funktionalität <70>: Export von Zeitdaten

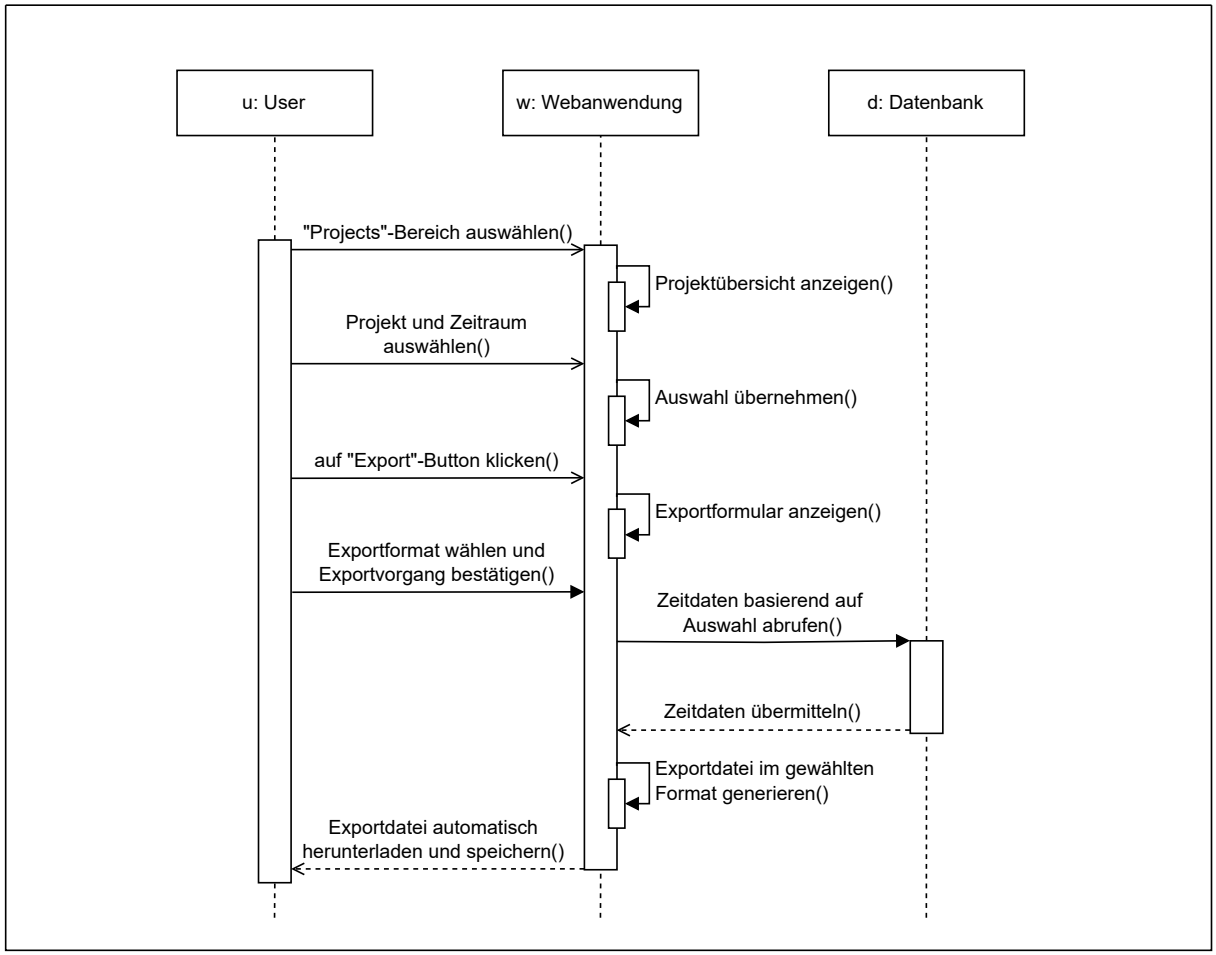

Abbildung 2.12: Sequenzdiagramm F70: Export von Zeitdaten

Das Sequenzdiagramm zu F70 zeigt den Ablauf zum Exportieren von erfassten Zeitdaten durch den:die Nutzer:in.

Zu Beginn wählt der:die Nutzer:in den Bereich „Projects“ in der Anwendung aus, woraufhin die Webanwendung die Projektübersicht lädt. Dort wird ein bestimmtes Projekt sowie ein Zeitraum ausgewählt. Diese Auswahl wird gespeichert und durch einen Klick auf den Button „Export“ wird der Exportvorgang gestartet. Die Webanwendung sendet automatisch eine Anfrage an das Backend, das die relevanten Zeitdaten abruft und eine Datei im gewählten Format (z.B. CSV oder PDF) generiert. Diese Datei wird direkt durch den Browser als Download ausgelöst und im Dateisystem des:der Nutzer:in gespeichert.

Diese Funktion ermöglicht es, Zeiteinträge strukturiert außerhalb der Anwendung zu sichern oder weiterzuverarbeiten.

2.8 Analyse von Funktionalität <80>: Kalenderansicht zur Anzeige projektbezogener Zeiteinträge

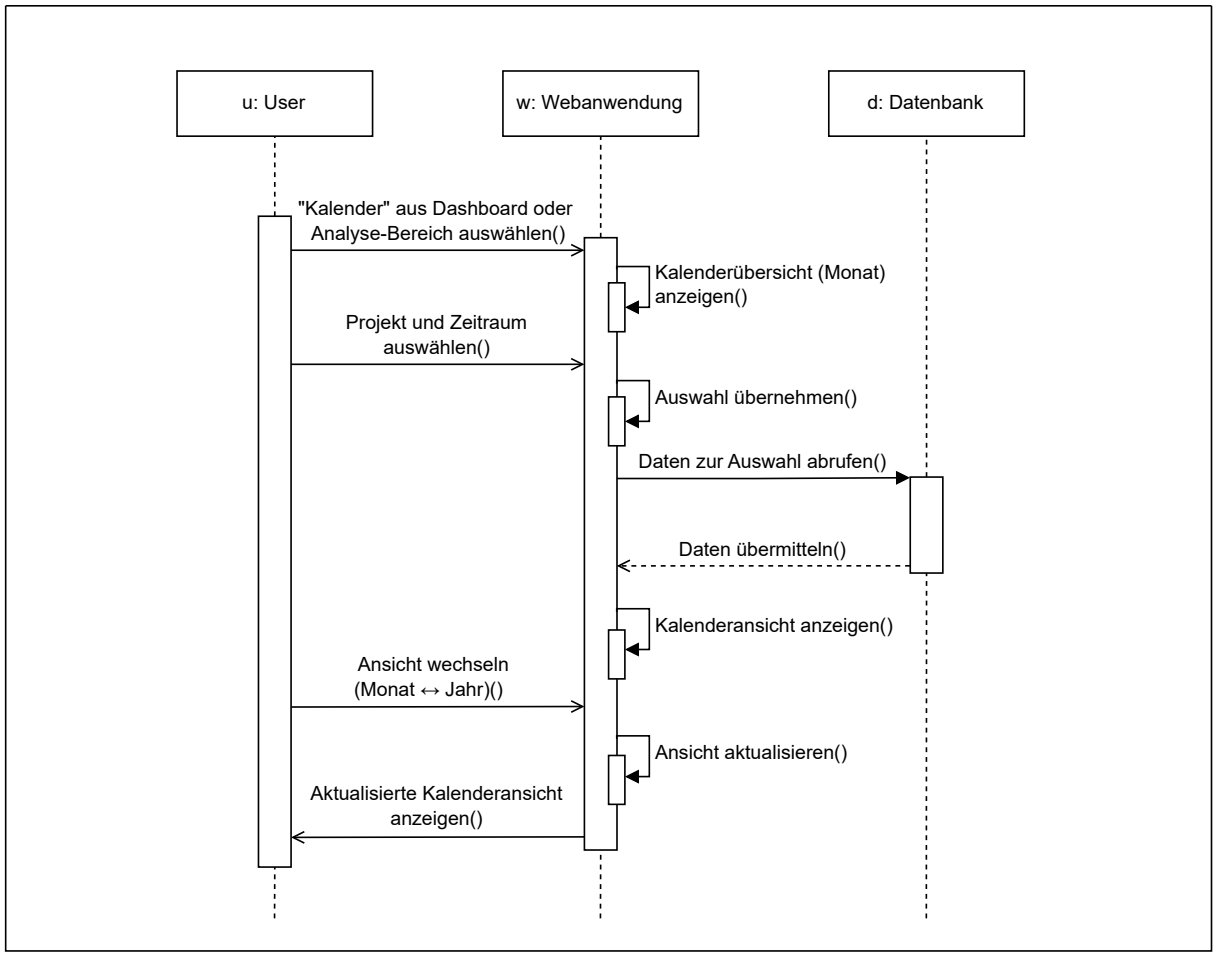

Abbildung 2.13: Sequenzdiagramm F80: Kalenderansicht zur Anzeige projektbezogener Zeiteinträge

Das Sequenzdiagramm zu F80 beschreibt den Ablauf beim Aufrufen und Nutzen der Kalenderansicht durch den:die Nutzer:in.

Der:Die Nutzer:in öffnet die Kalenderansicht über das Dashboard oder den Analysebereich. Die Webanwendung lädt automatisch die Monatsübersicht des Kalenders. Anschließend kann ein bestimmtes Projekt sowie ein gewünschter Zeitraum ausgewählt werden. Die Anwendung übernimmt die Auswahl und ruft die entsprechenden Daten aus der Datenbank ab. Nach erfolgreicher Übermittlung werden die relevanten Inhalte in der Kalenderansicht dargestellt. Optional kann der:die Nutzer:in in die Jahresansicht wechseln, woraufhin die Darstellung aktualisiert wird.

Diese Funktion dient der übersichtlichen Visualisierung dokumentierter Aktivitäten und unterstützt die zeitbezogene Auswertung und Planung.

2.9 Analyse von Funktionalität <F90>: Fortschritts- und Vergleichsdiagramme

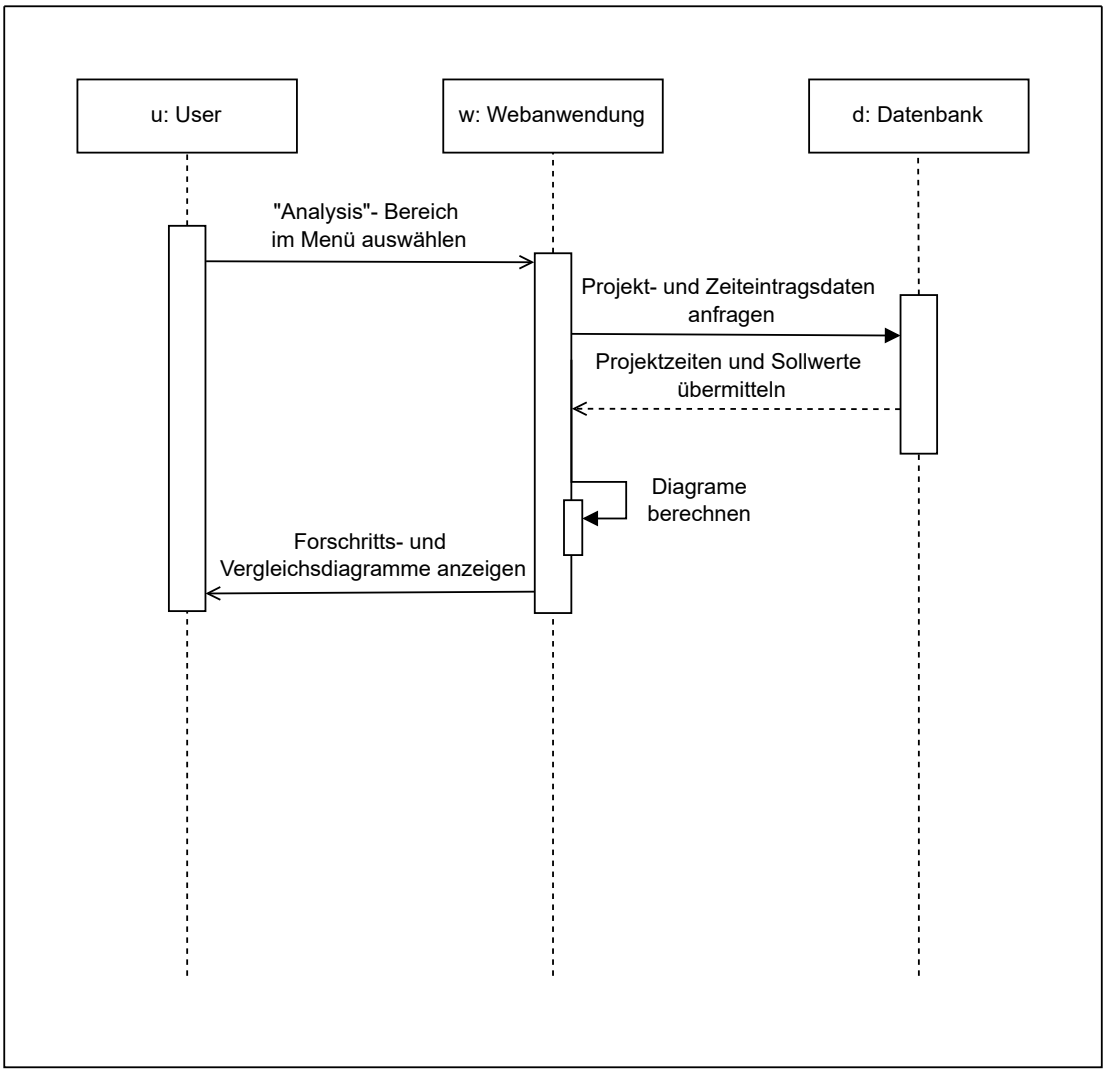

Abbildung 2.14: Sequenzdiagramm F90: Fortschritts- und Vergleichsdiagramme

Im Sequenzdiagramm zur Funktion <F90> wird der Ablauf zur Visualisierung von Projektfortschritt und Ist-Soll-Vergleich gezeigt. Nach Anmeldung und Aufruf des Analysebereichs wählt der:die Nutzer:in ein oder mehrere Projekte aus. Das System berechnet relevante Prozentwerte und erstellt Fortschritts- sowie Vergleichsdiagramme. Diese zeigen den Fortschritt einzelner Projekte und deren Zielerreichung im zeitlichen Vergleich. Bei fehlenden Sollwerten erscheint ein entsprechender Hinweis.

2.10 Analyse von Funktionalität <F100>: Teamverwaltung

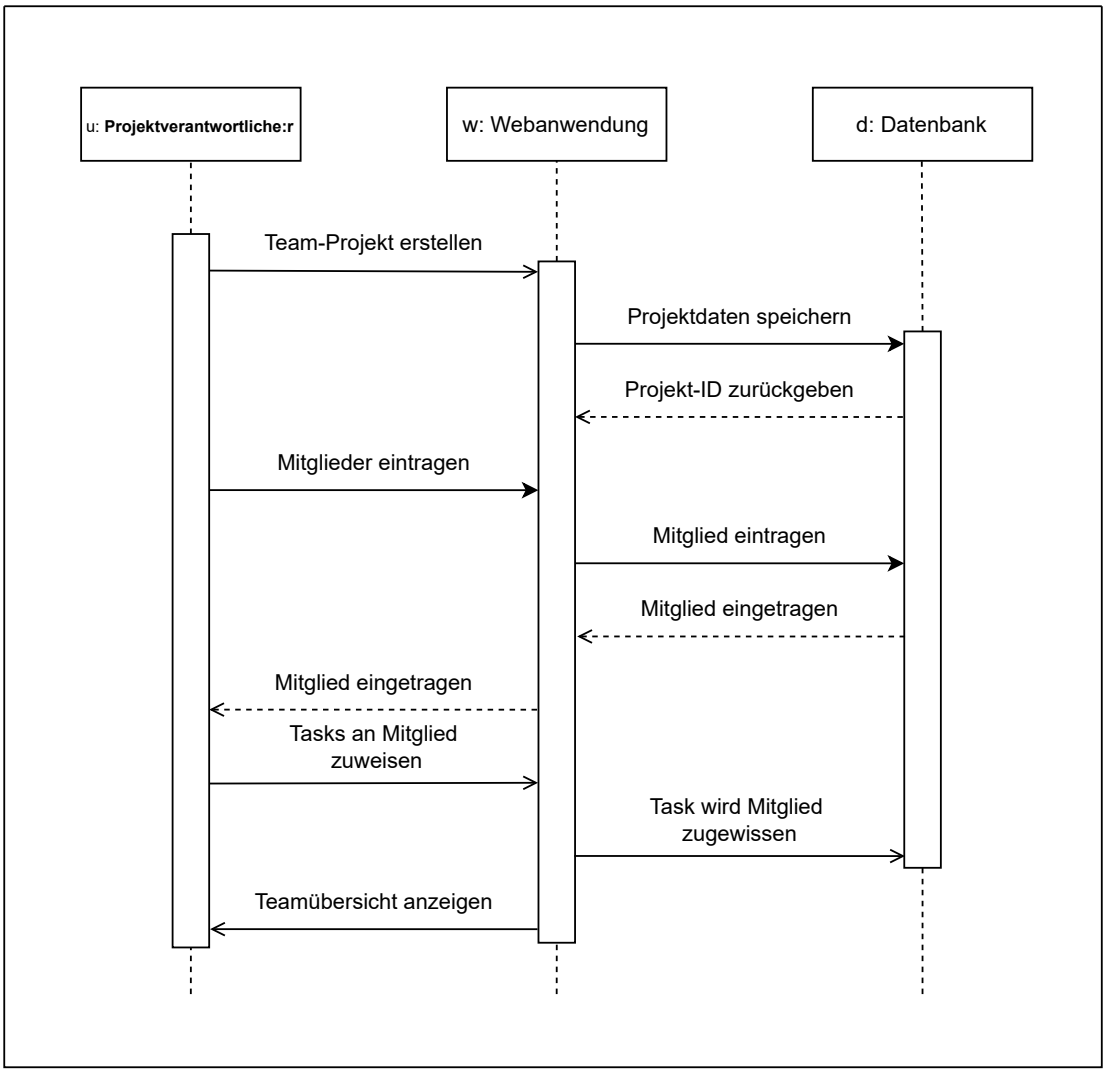

Abbildung 2.15: Sequenzdiagramm F100: Teamverwaltung

Im Sequenzdiagramm zur Funktion <F100> wird die Verwaltung von Teamprojekten dargestellt. Der:die Projektverantwortliche erstellt ein Projekt vom Typ „Team“ und weist anschließend Aufgaben gezielt einzelnen Teammitgliedern zu. Wird versucht, Aufgaben an nicht vorhandene oder nicht berechtigte Nutzer:innen zuzuweisen, erscheint eine entsprechende Fehlermeldung.

2.11 Analyse von Funktionalität <F110>: Benutzeroberfläche – Sprache

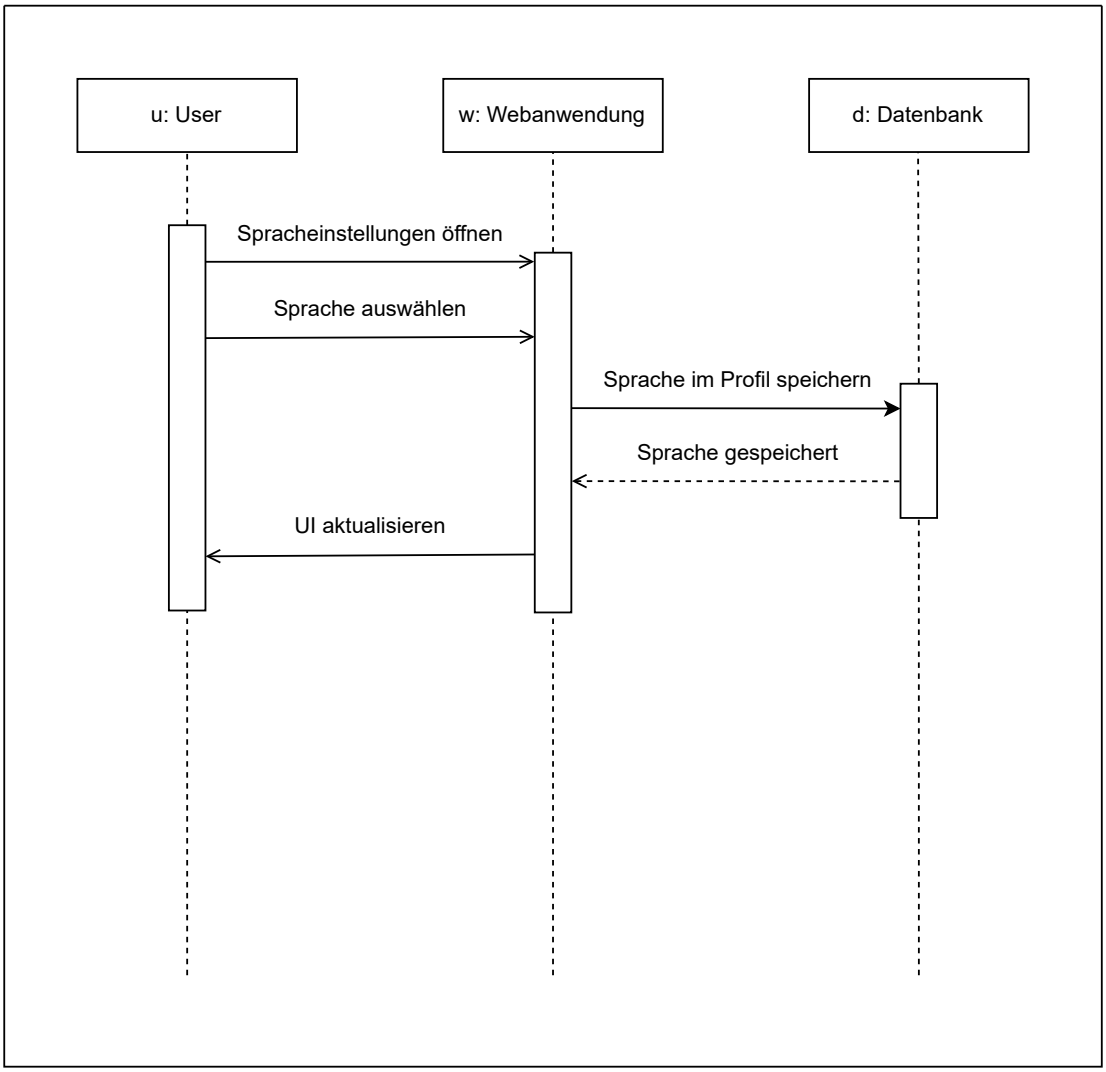

Abbildung 2.16: Sequenzdiagramm F110: Benutzeroberfläche – Sprache

Im Sequenzdiagramm zur Funktion <F110> wird der Ablauf zur Änderung der Spracheinstellung der Benutzeroberfläche dargestellt. Der:die Nutzer:in öffnet entweder das Einstellungsmenü oder nutzt einen sichtbaren Sprachumschalter. Nach Auswahl der gewünschten Sprache (standardmäßig Englisch) wird diese Einstellung im Profil der:des Nutzer:in gespeichert. Bei technischen Problemen bleibt die Standardsprache erhalten oder es erscheint eine Fehlermeldung. Zukünftig sind weitere Sprachen (z.B. Deutsch oder Spanisch) vorgesehen.

2.12 Analyse von Funktionalität <F120>: Plattformkompatibilität (Chrome)

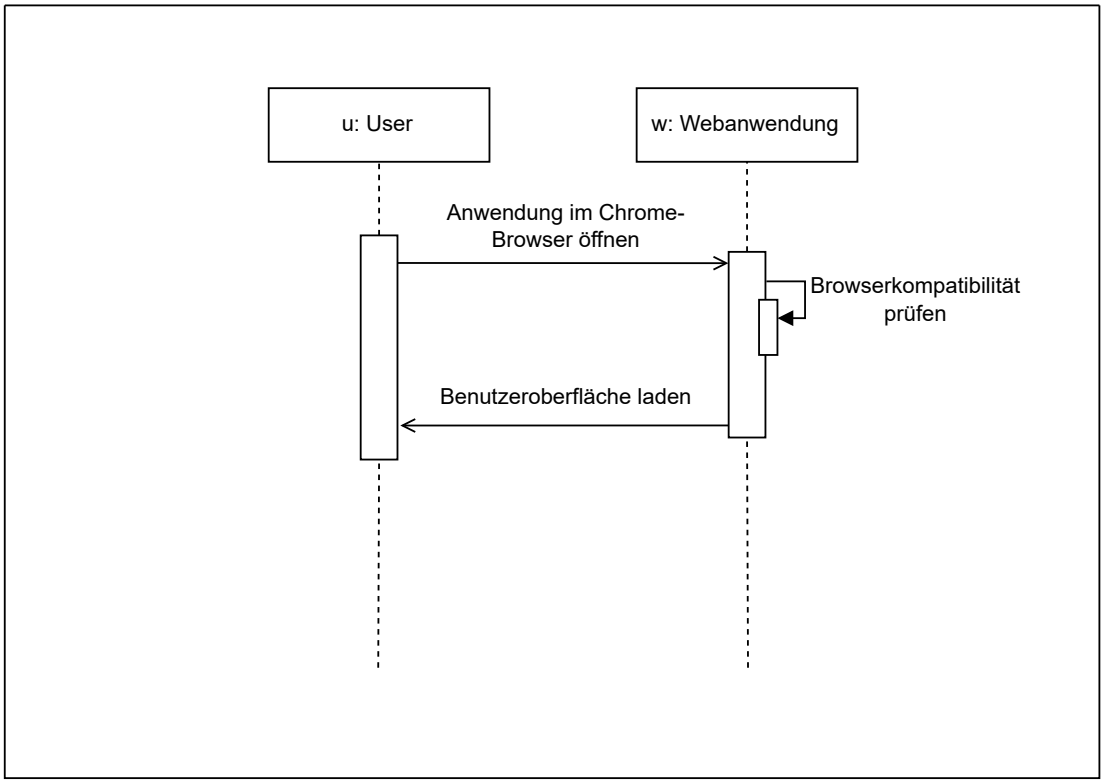

Abbildung 2.17: Sequenzdiagramm F120: Plattformkompatibilität (Chrome)

Im Sequenzdiagramm zu der Funktion <F120> wird die plattformunabhängige Nutzung der Webanwendung in Google Chrome dargestellt. Der:Die Nutzer:in öffnet die Anwendung im Chrome-Browser. Die Benutzeroberfläche wird daraufhin korrekt gerendert und alle Funktionen stehen uneingeschränkt zur Verfügung. Bei veralteten oder inkompatiblen Browsern kann es zu Einschränkungen kommen. Eine Erweiterung zur Unterstützung weiterer Browser (z.B. Firefox, Safari, Edge) ist möglich.

2.13 Analyse von Funktionalität <F130>: Datenpersistenz in SQLite

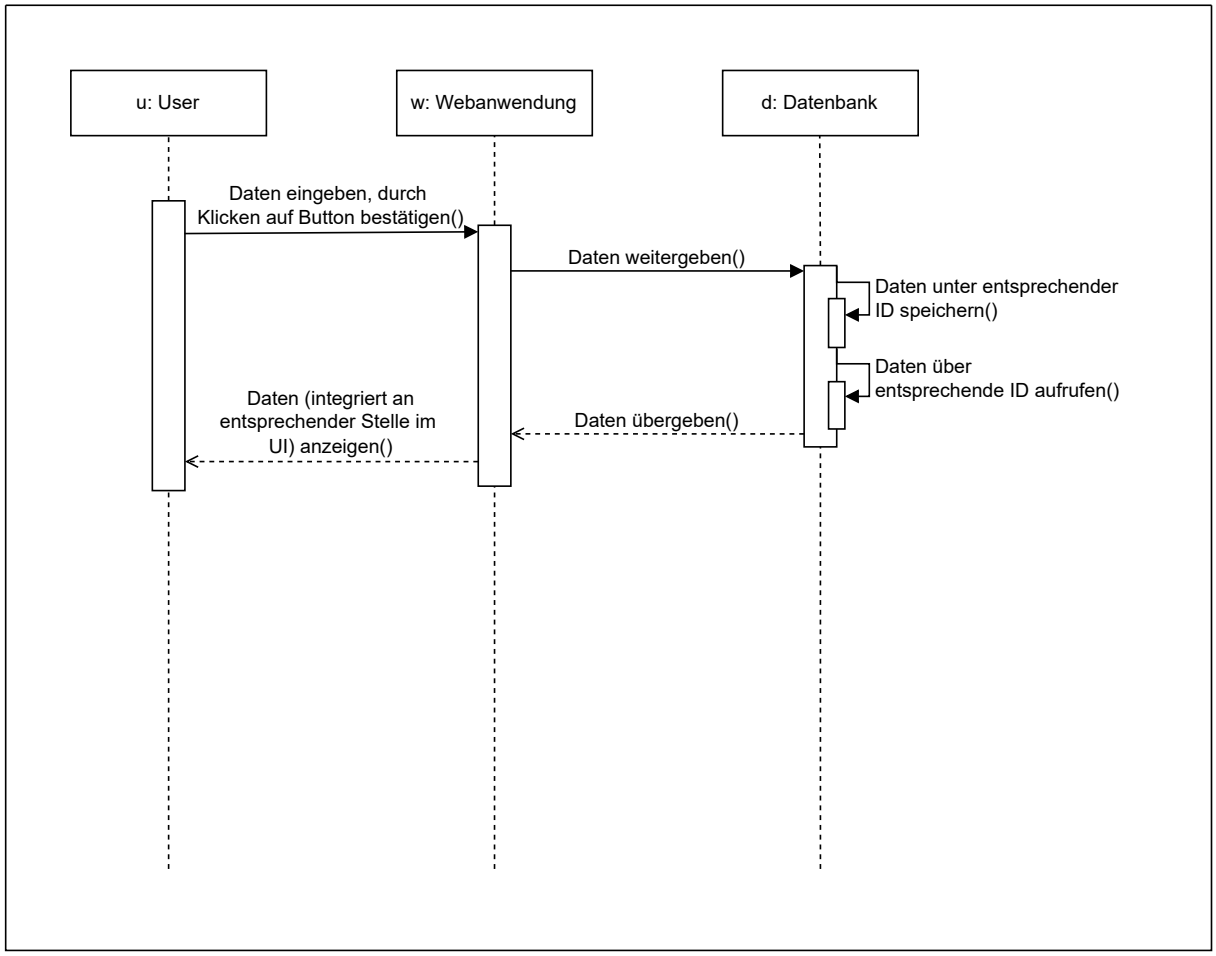

Abbildung 2.18: Sequenzdiagramm F130: Datenpersistenz in SQLite

Im Sequenzdiagramm zu der Funktion <F130> wird der Prozess der dauerhaften Speicherung aller erfassten Zeitdaten in einer zentralen SQLite-Datenbank dargestellt. Sobald Daten hinterlegt werden sollen, werden sie auf der Webseite eingetragen und durch das Klicken eines Buttons bestätigt. Daraufhin werden die Daten an die Datenbank übermittelt und im Anschluss unter einer eindeutigen ID gespeichert. Bereits vorhandene Einträge mit identischer ID werden in diesem Fall überschrieben, sodass Änderungen an bestehenden Daten zuverlässig berücksichtigt werden können. Die dauerhaft gespeicherten Daten können gezielt über ihre ID aus der Datenbank abgerufen werden und sind dementsprechend bei zukünftigen Zugriffen verfügbar. Sie werden anschließend an die Webanwendung zurückgegeben und dort in der Benutzeroberfläche an der vorgesehenen Stelle integriert. Durch diesen Ablauf wird eine konsistente und redundanzfreie Datenspeicherung gewährleistet.

3 Datenmodell

In diesem Kapitel wird der Aufbau und die Funktionsweise des Datenmodels genauer beschrieben. Die Speicherung der Daten erfolgt dabei in einer SQLite-Datenbank. Ziel ist es eine möglichst stabile und fehlerfrei Datenbank den Nutzer:innen zu bieten, um eine reibungslose Nutzung der Anwendung zu gewährleisten.

3.1 Diagramm

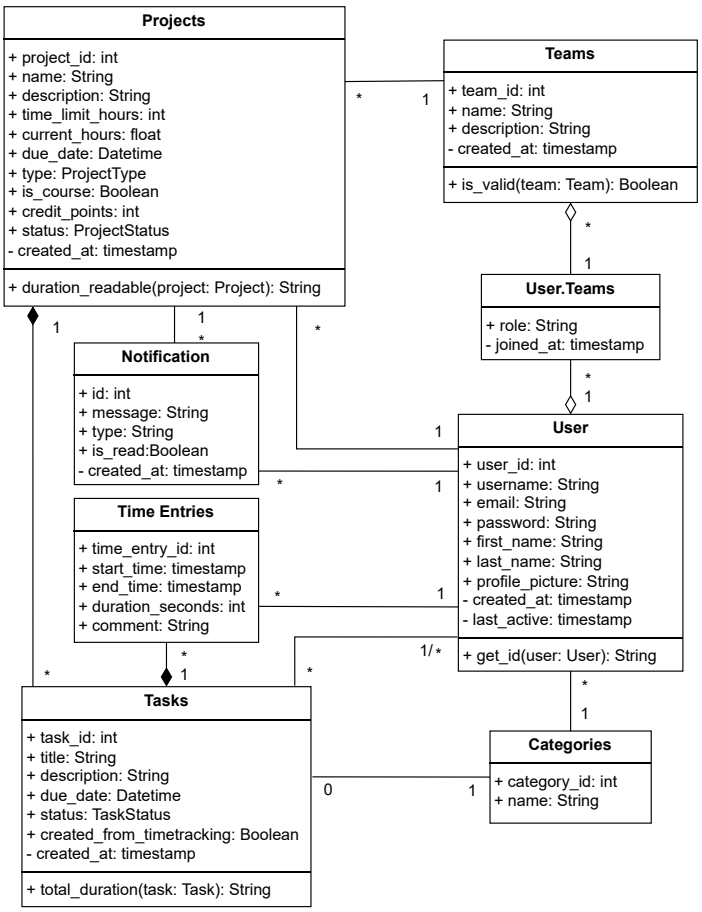

Abbildung 3.1: Klassendiagramm der langfristig zu speichernden Daten

3.2 Erläuterung

Die folgenden Tabelleneinträge dienen dazu, alle Entitäten im obigen Klassendiagramm weiter zu erläutern. Die grundlegenden Funktionalitäten werden von SQLAlchemy bereitgestellt, weitere Funktionalitäten wie to_dict( ) oder __repr__( ) wurden aufgrund von Redundanzen ausgelassen.

User $\langle E10 \rangle$

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
UserTeam <E20>	1:N	10–500 KB	Verknüpft Nutzer:in mit Teams über Zwischentabelle
TimeEntry <E30>	1:N	1–50 MB	Erfasste Arbeitszeiten des:der Nutzers:Nutzerin
Task <E50>	1/N:N	100–1.000 KB	Aufgaben, die dem:der Nutzer:in zugewiesen sind
Project <E60>	1:N	50–500 KB	Nutzer:in kann mehrere Projekte erstellen
Notification <E70>	1:N	50–500 KB	Benachrichtigungen für Aktionen von Nutzer:innen
Category <E80>	1:N	1–50 MB	Nutzer:in kann mehrere selbst definierte Projekte erstellen

UserTeam ⟨*E*20⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
User <E10>	N:1	10–500 KB	Verbindung zu genau einem:einer Nutzer:in
Team <E40>	N:1	10–500 KB	Verbindung zu genau einem Team

TimeEntry ⟨*E*30⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
User <E10>	N:1	10–500 KB	Arbeitszeit eines:einer Nutzer:in
Task <E50>	N:1	1–50 MB	Zeit wird konkreter Aufgabe zugeordnet

Team ⟨*E*40⟩

Beziehung	Kardinalität	Erwartete Datenmenge	Beschreibung
UserTeam <E20>	1:N	10–500 KB	Verknüpfung zu Mitgliedern
Project <E60>	1:N	50–500 KB	Team kann mehrere Projekte besitzen

Task ⟨*E*50⟩

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
TimeEntry <E30>	1:N	1–50 MB	Erfasste Zeiten zu Aufgabe
User <E10>	N:1	10–500 KB	Aufgabe ist einer Person zugeordnet, kann im Team auch mehreren Nutzer:innen zugwiesen sein.
Project <E60>	N:1	50–500 KB	Aufgabe gehört zu Projekt
Category <E80>	0:1	10–100 KB	Optionale Zuordnung zu Kategorie

Project ⟨*E*60⟩

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
Task <E50>	1:N	1–50 MB	Enthält zu erledigende Tasks
Team <E40>	N:1	50–500 KB	Projekt wird von Team betreut
User <E10>	N:1	10–500 KB	Projektverantwortliche:r
Notification <E70>	0:N	50–500 KB	Benachrichtigung zu Projektaktionen

Notification ⟨*E*70⟩

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
User <E10>	N:1	10–500 KB	Info über relevante Ereignisse
Project <E60>	N:1	10–500 KB	Verknüpfung zu Auslöser-Projekt

Category ⟨*E*80⟩

Beziehung	**Kardinalität**	**Erwartete Datenmenge**	**Beschreibung**
User <E10>	N: 1	50-100 KB	Nutzer:in kann mehrere selbst definierte Kategorien erstellen
Task <E50>	0:N	10–500 KB	Aufgaben können einer Kategorie zugeordnet sein

4 Konfiguration

In diesem Kapitel werden alle erforderlichen Konfigurationen beschrieben, welche für einen reibungslosen Betrieb der Anwendung notwendig sind. Die Anwendung basiert auf dem Python-Framework Flask und setzt sich aus verschiedenen Komponenten zusammen, die modular miteinander kommunizieren. Die Konfigurationsdateien dienen sowohl zur Anpassung an verschiedene Umgebungen als auch zur Festlegung von sicherheitsrelevanten und infrastrukturellen Parametern.

Zentraler Bestandteil der Konfiguration ist die Datei `.env`, welche sich im Hauptverzeichnis der Anwendung befindet. Diese Datei enthält eine Vielzahl von Umgebungsvariablen, die für den Betrieb der Applikation notwendig sind. Zu den in der `.env`-Datei enthaltenen Einstellungen gehören unter anderem der geheime Schlüssel der Anwendung, sowie Pfade zu Ressourcen wie zum Beispiel zu den Profilbildern. Darüber hinaus wird über diese Datei auch der Zugriff auf den verwendeten Mailserver geregelt. Hierzu gehören beispielsweise der Hostname des Maildienstes, die Portnummer, Authentifizierungsdaten sowie die Absenderadresse. Diese Angaben sind insbesondere für Funktionen wie Passwortzurücksetzung oder Benachrichtigungen relevant.

Ein weiterer wichtiger Aspekt der Konfiguration ist die Datenbankanbindung. Die Anwendung verwendet SQLAlchemy als ORM-Bibliothek. Der Pfad zur Datenbank wird dynamisch aus dem Projektverzeichnis heraus generiert und über eine Umgebungsvariable bereitgestellt. Die Migration von Datenbankschemata wird mithilfe von Alembic durchgeführt. Die hierfür erforderlichen Einstellungen sind ebenfalls an die Umgebungsvariablen gekoppelt und werden bei jeder Migration automatisch berücksichtigt.

Die installierten Abhängigkeiten der Anwendung sind in der Datei `requirements.txt` festgehalten. Diese Datei befindet sich im Hauptverzeichnis des Projekts und listet alle für die Ausführung notwendigen Python-Pakete mit genauer Versionsangabe auf. Dazu gehören neben Flask selbst auch Erweiterungen für Authentifizierung, Autorisierung, E-Mail-Versand, Datenbankintegration, Migration sowie Testautomatisierung (`pytest`). Durch die Angabe fester Versionen wird eine einheitliche Umgebung gewährleistet, was insbesondere für die Nachvollziehbarkeit und Wartbarkeit des Systems von Bedeutung ist. Die Pakete können typischerweise automatisiert über einen Paketmanager installiert werden.

Die Anwendung ist so aufgebaut, dass sie sowohl auf einem lokalen Entwicklungsrechner als auch auf einem Produktionsserver betrieben werden kann. Zur lokalen Entwicklung genügt ein

Rechner mit Python-Unterstützung, während für den produktiven Betrieb zusätzliche Konfigurationen wie beispielsweise eine sichere Transportverschlüsselung, ein externer Maildienst sowie persistente Datenbankverbindungen notwendig sind.

Neben der serverseitigen Anwendung kommen im Frontend kleinere JavaScript-Dateien zum Einsatz. Diese übernehmen grundlegende Aufgaben wie die Validierung von Formulareingaben. Die Dateien befinden sich im Verzeichnis `frontend/static/js` und werden direkt in die entsprechenden HTML-Templates eingebunden.

5 Glossar

.env Eine Datei, in der Umgebungsvariablen gespeichert werden, zum Beispiel Zugangsdaten oder Konfigurationen, die nicht im Quellcode hinterlegt werden sollen.

Alembic Ein Datenbank-Migrationswerkzeug für SQLAlchemy, das Änderungen am Datenbankschema versioniert und verwaltet (siehe `https://alembic.sqlalchemy.org/`).

Entwicklungsrechner Ein Computer, auf dem Software lokal programmiert, getestet und debuggt wird.

Flask Leichtgewichtiges Web-Framework für Python zur Entwicklung von Webanwendungen (siehe `https://flask.palletsprojects.com/`).

Frontend Grafische Benutzeroberfläche zur Bedienung einer Website oder App.

Hostname Der Name eines Computers im Netzwerk, der zur Adressierung genutzt wird (z. B. `localhost` oder `mail.example.com`).

HTML-Templates Vorlagen in HTML, die Platzhalter enthalten und mit dynamischen Inhalten (z. B. durch Flask) gefüllt werden, um Webseiten zu generieren.

JavaScript Eine Programmiersprache zur Umsetzung interaktiver Funktionen im Webbrowser.

Mailserver Ein Server, der den Versand, Empfang und die Weiterleitung von E-Mails übernimmt.

OAuth Offenes Protokoll zur Autorisierung, das es Nutzer:innen ermöglicht, sich über Drittanbieter wie Google oder GitHub bei einer Anwendung zu authentifizieren, ohne ihre Zugangsdaten direkt weiterzugeben.

ORM „Object-Relational Mapping“ – eine Technik, die Objekte in Programmiersprachen mit Datenbanktabellen verknüpft, um Datenbankoperationen objektorientiert durchzuführen.

Paketmanager Ein Werkzeug zur Verwaltung von Softwarepaketen, das Installation, Aktualisierung und Abhängigkeiten automatisiert (z. B. `pip` für Python oder `npm` für JavaScript).

Passwort-Hashing Verfahren, bei dem ein Passwort mithilfe einer kryptografischen Funktion in eine eindeutige Zeichenkette (den Hash) umgewandelt wird. Dieser Hash wird anstelle des Klartextpassworts gespeichert und schützt so die Passwörter vor unbefugtem Zugriff.

Portnummer Eine numerische Adresse zur Identifizierung eines bestimmten Dienstes auf einem Rechner (z. B. `5000` für Flask-Entwicklung oder `25` für SMTP).

Produktionsserver Ein Server, auf dem eine Anwendung live und für Endnutzer verfügbar läuft.

pytest Python-Framework zum Schreiben und Ausführen von Tests (siehe `https://docs.pytest.org/`).

Python Programmiersprache, die sich besonders gut für Webentwicklung, Datenanalyse, Automatisierung und viele weitere Bereiche eignet (siehe `https://www.python.org/`).

SQLAlchemy Ein in Python geschriebenes Toolkit zur Datenbankprogrammierung, das die Arbeit mit relationalen Datenbanken vereinfacht (siehe `https://www.sqlalchemy.org/`).

Webanwendung Eine über den Webbrowser nutzbare Software, die keine Installation auf dem eigenen Gerät erfordert.