Clockwise

Implementierung

Intuitive Time Tracking

Betreuer

Maximilian Li

Team

Dominik Pollok
Phil Gengenbach
Alina Petri
José Ayala
Johann Kohl

19. Juli 2024

Intuitive Robots Lab

Inhaltsverzeichnis

1 Einleitung

Das primäre Ziel unserer Anwendung ist die Entwicklung einer intuitiven und effizienten Zeiterfassungsanwendung für wissenschaftliche Hilfskräfte. Diese Implementierungsphase bildet einen entscheidenden Abschnitt in unserem Projekt, der auf den detaillierten Vorgaben des Pflichtenhefts und den technischen Spezifikationen unseres Entwurfsdokuments aufbaut.
Durch die Einführung der Model-View-Controller-Service (MVCS) Architektur konnten wir die Entwicklung unserer Anwendung systematisch strukturieren. Dies hat die Implementierung, das Testing effizienter gestaltet und ermöglicht es uns, die Benutzerfreundlichkeit und praktische Anwendbarkeit während der gesamten Entwicklung im Auge zu behalten.
In dieser Phase konzentrieren wir uns darauf, die geplanten Funktionen zu realisieren und dabei stets die Benutzerfreundlichkeit und die praktische Anwendbarkeit im Auge zu behalten.
Die nachstehenden Abschnitte beschreiben ausführlich den Verlauf der Implementierungsphase, darunter die zeitliche Planung, die Verteilung der Aufgaben innerhalb des Teams und die Ergebnisse unserer umfangreichen Tests.

2 Planung der Implementierung

2.1 Zeitlicher Ablauf

Die Implementierungsphase unseres Projekts war klar strukturiert und auf vier Wochen vom 23.06. bis zum 19.07. terminiert. Die folgende Beschreibung skizziert den geplanten zeitlichen Ablauf und die Hauptaktivitäten, die in dieser Zeit von den Backend- und Frontend-Teams parallel durchgeführt wurden. Die aufgeführten Aufgaben erheben keinen Anspruch auf Vollständigkeit und werden durch die im Gant-Diagramm aufgeführten Punkte ergänzt.

Woche 1 (23.06. - 29.06.): Initialisierung und Erweiterung

- **Backend-Team:**
  - Überprüfung und Optimierung der bestehenden Backend-Architektur.
  - Testen der Repository-Klassen mit Unit-Tests.
  - Weiterentwicklung spezifischer Funktionen wie Authentifizierung und Timesheet-Verwaltung.
- **Frontend-Team:**
  - Erste Anbindung an bestehende Backend-Services.
  - Entwicklung der Basis-UI-Komponenten zur Interaktion mit den Backend-Endpoints.

Woche 2 (30.06. - 06.07.): Entwicklung und Integration

- **Backend-Team:**
  - Implementierung neuer und Erweiterung bestehender Funktionen.
  - Einrichtung fortgeschrittener Datenvalidierung und Fehlerbehandlung.
  - Testen der Service- und Controller-Klassen mit Unit-Tests.
- **Frontend-Team:**
  - Entwicklung der Pages für alle Nutzergruppen.
  - Erstellung von Popups und verschiedenen Webapp-Funktionalitäten.

- Implementierung von Widgets und Filtern sowie Anbindung ans Backend.

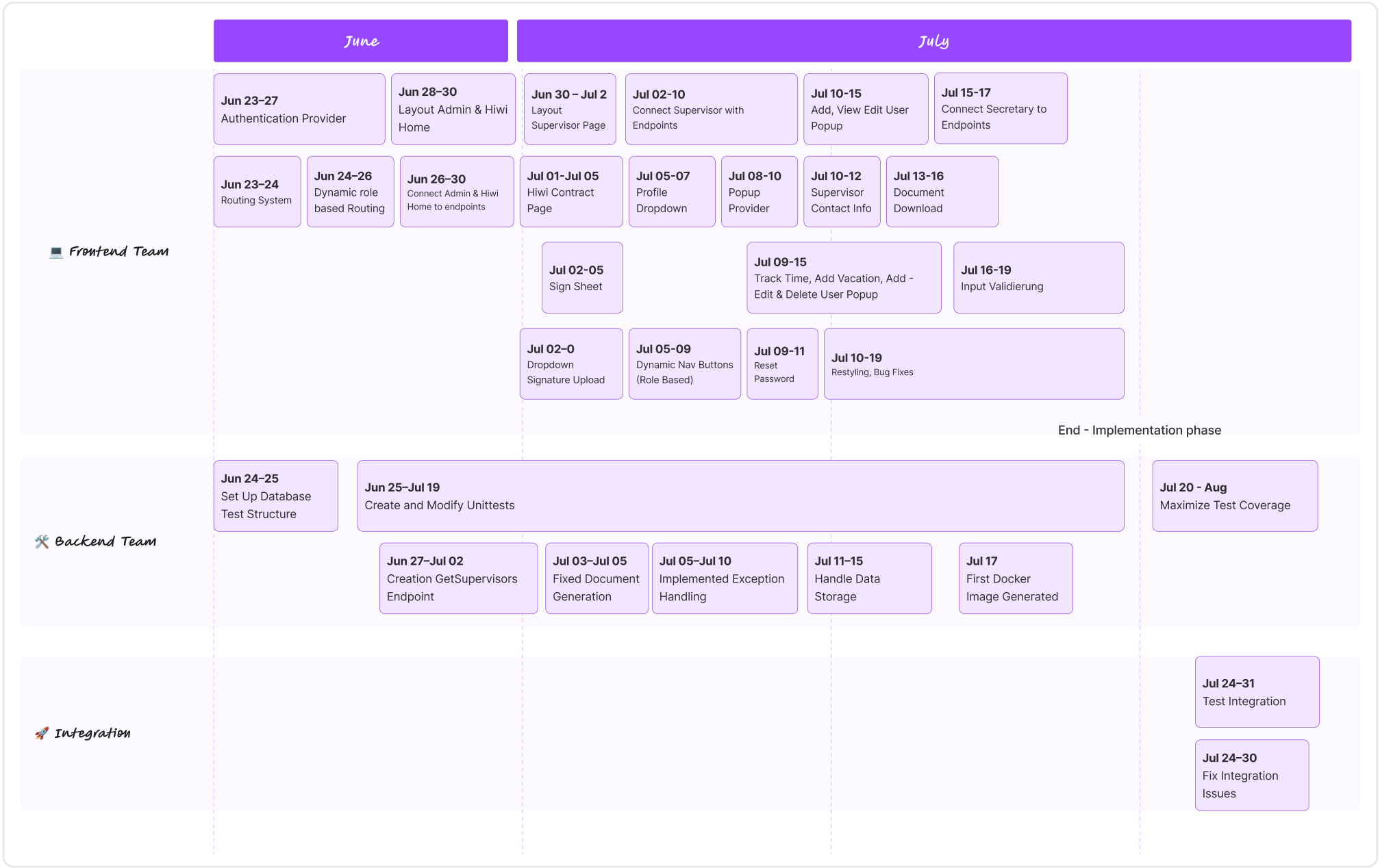
Woche 3 (07.07. - 13.07.): Feinabstimmung und Test

- **Backend-Team:**
  - Abschluss der Backend-Entwicklungen und Vorbereitung auf finale Phase.
- **Frontend-Team:**
  - Fortführung der Benutzerinteraktionstests und Anpassungen basierend auf Testergebnissen.
  - Intensive Tests der integrierten Funktionen und UI-Elemente.
  - Entwicklung letzter Elemente.

Woche 4 (14.07. - 19.07.): Abschluss

- **Backend-Team:**
  - Durchführung der finalen Optimierungen und Bug Fixes.
- **Frontend-Team:**
  - Abschluss aller Benutzeroberflächenarbeiten, Fine Tuning, Styling und Bugfixing.

Clockwise GANT Chart



3 Implementierung

3.1 Organisation

Unser Entwicklungsprojekt ist in zwei Hauptteams gegliedert: das Frontend- und das Backend-Team. Diese Struktur ermöglicht eine spezialisierte und effiziente Bearbeitung der unterschiedlichen Aspekte unserer Anwendung.

- **Backend-Team:** Die Backend-Entwicklung hat bereits vor der offiziellen Implementierungsphase begonnen. Dies hat es dem Frontend-Team ermöglicht, nahtlos mit der Entwicklung zu beginnen, sobald das Backend ausreichend vorbereitet war. Das Backend-Team konzentriert sich auf die Erstellung robuster API-Endpunkte und die Sicherstellung der Datenintegrität.
- **Frontend-Team:** Das Frontend-Team beginnt mit der Implementierung der Benutzeroberfläche und die Anbindung an die Endpunkte des Backends. Dieses Team ist dafür verantwortlich, eine intuitive und reaktionsfähige Benutzererfahrung zu gewährleisten.

Die gesamte Entwicklung ist issue-getrieben und wird über GitHub organisiert. Jedes aufkommende Problem, sei es ein fehlender Endpunkt oder ein Bug, wird als Issue erfasst. Diese Issues werden mit Prioritäten und Tags versehen, um ihre Dringlichkeit und Art zu klassifizieren und eine schnelle Zuweisung und Bearbeitung zu ermöglichen.

3.1.1 Kommunikation und Kollaboration

Die Kommunikation innerhalb und zwischen den Teams erfolgt regelmäßig und strukturiert. Wir setzen auf folgende Methoden zur Förderung der Teamarbeit und Effizienz:

1. **Pair Programming:** Besonders komplexe oder kritische Aufgaben werden oft im Pair Programming bearbeitet. Diese Methode verbessert die Codequalität und fördert den Wissenstransfer innerhalb des Teams. Außerdem können so alle Teammitglieder vom Wissen ihrer Kollegen profitieren, was zu einer steileren Lernkurve führt.
2. **Regelmäßige Meetings:** Um sicherzustellen, dass alle Teammitglieder auf dem gleichen Stand sind und um die interdisziplinäre Kommunikation zu fördern, finden regelmäßige Meetings statt. Diese dienen auch dazu, den Fortschritt zu überprüfen und kommende Aufgaben

zu planen. Konkret heißt das drei wöchentliche Treffen im gesamten Team, davon einmal pro Woche zusammen mit dem genialen Betreuer Maximilian Li, dessen Fachwissen und Engagement das Projekt maßgeblich voranbringt. Zudem werden je nach Bedarf innerhalb der zwei Teams (Frontend / Backend) flexibel Meetings zum Pair Programming, oder allgemeinen Absprache abgehalten.

3. **Code Reviews:** Jeder Code, der in den ‘develop‘ Branch gemerged wird, unterliegt einem Review-Prozess durch mindestens ein anderes Teammitglied. Dies wird normalerweise innerhalb des Frontend- bzw. Backend-Teams gemacht, da diese Mitglieder schon vertrauter mit dem zu überprüfenden Code sind. Das Überprüfen stellt nicht nur die Codequalität sicher, sondern fördert auch eine gemeinschaftliche Verantwortung für das Projekt.

Durch diese strukturierten Prozesse und die klare Aufgabenverteilung kann unser Team effizient arbeiten und gleichzeitig sicherstellen, dass die Entwicklung unserer Anwendung reibungslos und ohne signifikante Verzögerungen voranschreitet.

3.2 Version Control System

Wir verwenden Git und GitHub als unsere Plattformen für das Versionskontrollsystem. Diese Tools ermöglichen es uns, den Entwicklungsprozess effizient zu organisieren, Änderungen zu verfolgen und die Zusammenarbeit in einem team- und issue-getriebenen Umfeld zu erleichtern.

3.2.1 Git Branch Strategie

Unsere Branching-Strategie spielt eine entscheidende Rolle in der Strukturierung unseres Entwicklungsprozesses. Sie hilft uns, eine klare Trennung zwischen stabilen und in Entwicklung befindlichen Versionen des Codes zu gewährleisten.

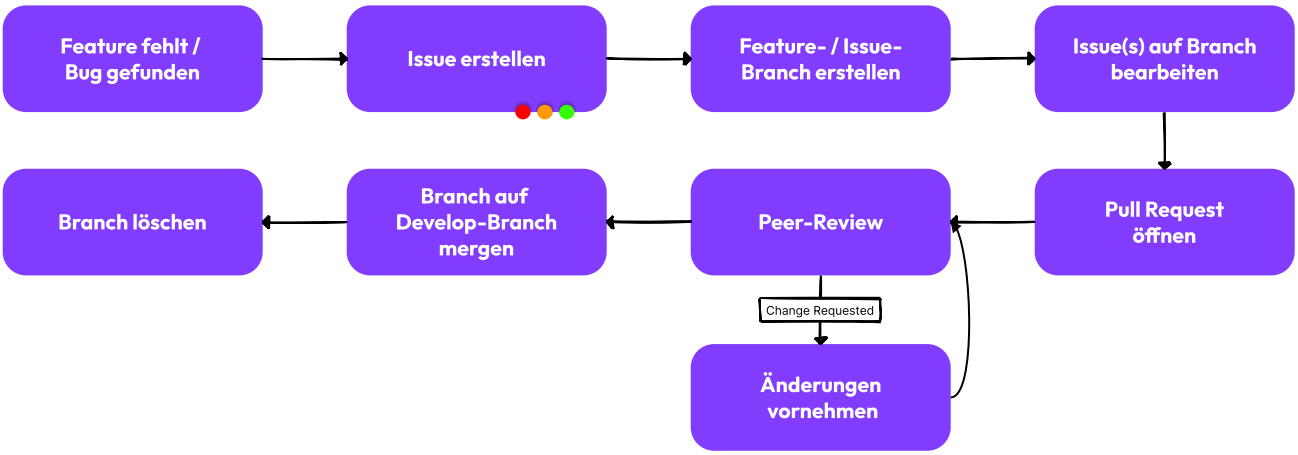
- **Main Branch:** Der `main` Branch repräsentiert die stabilste Version des Projekts und wird nur für Deployments in die Produktionsumgebung verwendet.
- **Develop Branch:** Der `develop` Branch dient als Hauptentwicklungsbranch, der alle laufenden Änderungen enthält, die noch nicht für die Produktion freigegeben sind.

3.2.2 Feature und Fix Branches

Von `develop` ausgehend werden spezifische Branches für neue Features oder Bugfixes erstellt. Nachfolgend ist der typische Ablauf skizziert:

1. Ein neues Issue wird identifiziert, z.B. ein fehlender Endpunkt, der vom Frontend-Team entdeckt und gemeldet wird.
2. Ein Branch wird von `develop` erstellt, um das Problem zu adressieren.
3. Nach der Implementierung wird der Branch in `develop` gemerged, wo er umfassend getestet wird.
4. Nach erfolgreicher Testphase und Konfliktlösung wird der Branch schließlich in den `main` gemerged, um die Stabilität für das Deployment zu garantieren.

3.2.3 Übersicht - Entwicklungsprozess



Diese Abbildung zeigt den schrittweisen Ablauf unseres Entwicklungsprozesses. Das wichtigste hierbei ist das Peer-Review, welches vor jedem Merge auf den `develop` Branch durchlaufen werden muss. Sollten im Zuge der Reviews Fehler auffallen, oder der Code nicht den gewünschten Qualitätsstandards entsprechen, so wird ein "Change Request" angefordert. Nachdem die angemerkten Änderungen vorgenommen wurden, wird dies im Peer-Review überprüft und erst bei erfolgreicher Abnahme in den `develop` Branch gemerged.

4 Implementierungsergebnisse

4.1 Implementierte Kriterien (noch nicht angepasst)

Im Folgenden werden noch nicht implementierte Kriterien mit einem Stern (*) markiert. Diese werden im Rahmen der Qualitätssicherung nachgereicht.

4.1.1 Musskriterien

- Das gesamte System muss vollständig in einer Docker-Containerumgebung implementiert und betrieben werden können.
- Das System muss ein Login-System umfassen, das die Folgenden vier verschiedenen Benutzergruppen unterstützt:
  1. Administrator (Admin)
  2. Sekretariat
  3. Betreuer
  4. Wissenschaftliche Hilfskraft (Hiwi)
- Das System muss innerhalb des lokalen Netzwerks (LAN) vollumfänglich erreichbar und funktionsfähig sein.
- Im Docker-Container muss ein vordefiniertes Notfallpasswort hinterlegt sein, das den Zugriff auf das System ermöglicht, falls sämtliche Administratoren ihr Passwort vergessen sollten.

4.1.2 Musskriterien nach Benutzergruppe

Admin:

- Muss Benutzer erstellen, löschen, bearbeiten, sperren und freigeben können. *(Das Sperren und Freigeben von Nutzern ist noch nicht implementiert)
- Muss ein Passwort vergeben können, das nach dem ersten Login durch den Benutzer geändert werden kann.
- Muss das Passwort eines Nutzers auf ein neu generiertes Passwort zurücksetzen können. *
- Muss wissenschaftliche Hilfskräfte (Hiwis) einem Betreuer zuordnen können.

- Muss die Nutzergruppen 'Hiwi', 'Sekretariat', 'Betreuer' und 'Admin' Nutzern zuweisen können.

Sekretariat:

- Muss eingereichte Timesheets einsehen, eine Archiv-PDF erstellen und diese herunterladen können.
- Muss einsehen können, welche Hiwis bereits einen Stundenzettel eingereicht haben und von welchen noch Einreichungen ausstehen.

Betreuer:

- Muss eingereichte Timesheets seiner Hiwis einsehen und signieren können.
- Muss eine Änderung an einem Stundenzettel anfordern können, falls Unstimmigkeiten auftreten.
- Muss seine Unterschrift zur Signatur der Timesheets im Profil hinterlegen können.
- Muss einsehen können, wenn ein ihm zugeordneter Hiwi einen Stundenzettel signiert und zur Unterschrift freigibt.

Hiwi:

- Muss Arbeitsstunden und Pausen für Arbeitstage des aktuellen Monats eintragen können.
- Muss eingetragenen Arbeitszeiten ein Projekt zuweisen.
- Muss eingetragene Arbeitszeiten am Monatsende signieren und diese damit dem Betreuer zur Signierung freigeben können.
- Muss seine Unterschrift zur Signatur der Timesheets im Profil hinterlegen können.
- Muss den Übertrag der Minus- und Überstunden des Vormonats einsehen können.
- Muss die gesamte Historie seiner eingetragenen Arbeitszeit einsehen können.
- Muss in Anspruch genommenen Urlaub einsehen können.
- Muss Urlaubstage im System erfassen können.

4.2 Implementierte Wunschkriterien

- Ein Betreuer kann beim Anfordern von Änderungen in einem kurzen Text beschreiben, welche Änderungen gewünscht sind. Diese Beschreibungen werden dann an den Hiwi weitergeleitet.
- Die Zeiterfassung verhindert das Eintragen von Arbeitszeiten an gesetzlichen Feiertagen.
- Das System warnt den Nutzer, falls die erfassten Arbeitszeiten nicht mit den institutsinternen Richtlinien für Arbeit übereinstimmen, z.B.: keine Arbeit am Wochenende oder in der Nacht.
- Hiwis können ihre erfassten Arbeitszeiten als PDF-Dokument herunterladen.

4.2.1 Informationen und Benachrichtigungen

- Eine Hilfeseite informiert Hiwi-Mitarbeiter über Vorschriften zur Arbeitszeiterfassung, einschließlich der Regelung einer maximalen täglichen Arbeitszeit von 10 Stunden und die Einhaltung der verpflichtenden Pausezeiten.
- Hiwis können die Kontaktübersicht ihres Betreuers sowie den Signatur-Status des letzten Timesheets einsehen.
- Betreuer erhalten eine Benachrichtigung, sobald einer ihrer Hiwis ein Timesheet zur Signatur freigibt. *(geplant)
- Hiwis und Betreuer werden benachrichtigt, wenn eine Frist für die Abgabe naht oder überschritten wird. *(geplant)
- Hiwis erhalten eine Benachrichtigung, wenn der Betreuer Änderungen am Timesheet anfordert. Es wird dem Hiwi dabei ein Kommentar mit der gewünschten Änderungswunsch übermittelt. *(geplant)
- Hiwis werden gewarnt, wenn sie mehr als 8 Überstunden ansammeln. *(geplant)
- Die Web-App benutzt Slack, um Benachrichtigungen an entsprechende Personen des Instituts zu senden. *(geplant)

4.3 Abweichungen vom Pflichtenheft

- Admin muss die maximal zulässige Arbeitszeit eines Hiwis festlegen können. Gründe für die Nicht-Implementierung:
  - **Gesetzliche Regelungen:** Die Arbeitszeiten sind bereits durch Gesetze streng geregelt, was eine manuelle Anpassung durch Administratoren unnötig macht.
  - **Warning and Failure:** Das System gibt automatisch Warnungen aus und blockiert die Zeiterfassung, wenn gesetzliche Höchstarbeitszeiten erreicht werden, was eine zusätzliche manuelle Eingabe überflüssig macht.
- Anders als im Pflichtenheft vorgesehen hat der Admin nicht die Möglichkeit ein Einmalpasswort für einen Nutzer zu setzen, sondern lediglich ein Passwort festzulegen, welches vom Nutzer im Anschluss geändert werden kann. Dies sorgt für einen weniger restriktiven Prozess und somit eine flexiblere Anwendungsnutzung.

4.4 Verzögerungen während der Implementation

Während der Implementierungsphase unserers Projekts traten wiederholt kleinere Verzögerungen auf. Diese wurden hauptsächlich durch das Auftreten unvorhergesehener Bugs verursacht, die nicht nur zusätzliche Zeit zur Behebung erforderten, sondern auch gelegentliche Umstrukturierungen im Entwicklungsprozess erforderten.

Diese Vorfälle bestätigten uns in unserem systematischen Vorgehen, einer flexiblen Projektplanung und die Notwendigkeit, Ressourcen für unvorhergesehene Ereignisse einzuplanen. Sie zeigten uns auch die Wichtigkeit regelmäßiger Code-Reviews und einer robusten Teststrategie, um die Stabilität unserer Software langfristig zu sichern und Verzögerungen zu minimieren.

5 Testing

Um bereits einen Grundstein für die folgende Qualitätssicherungsphase zu legen, haben wir uns bereits zu Beginn der Implementierungsphase mit dem Thema Testing auseinandergesetzt. So konnten wir bereits in der Entwicklung frühzeitig Fehler erkennen und beheben, sodass die Anwendung von grundauf auf Qualität geprüft wurde.

5.1 Unit Testing

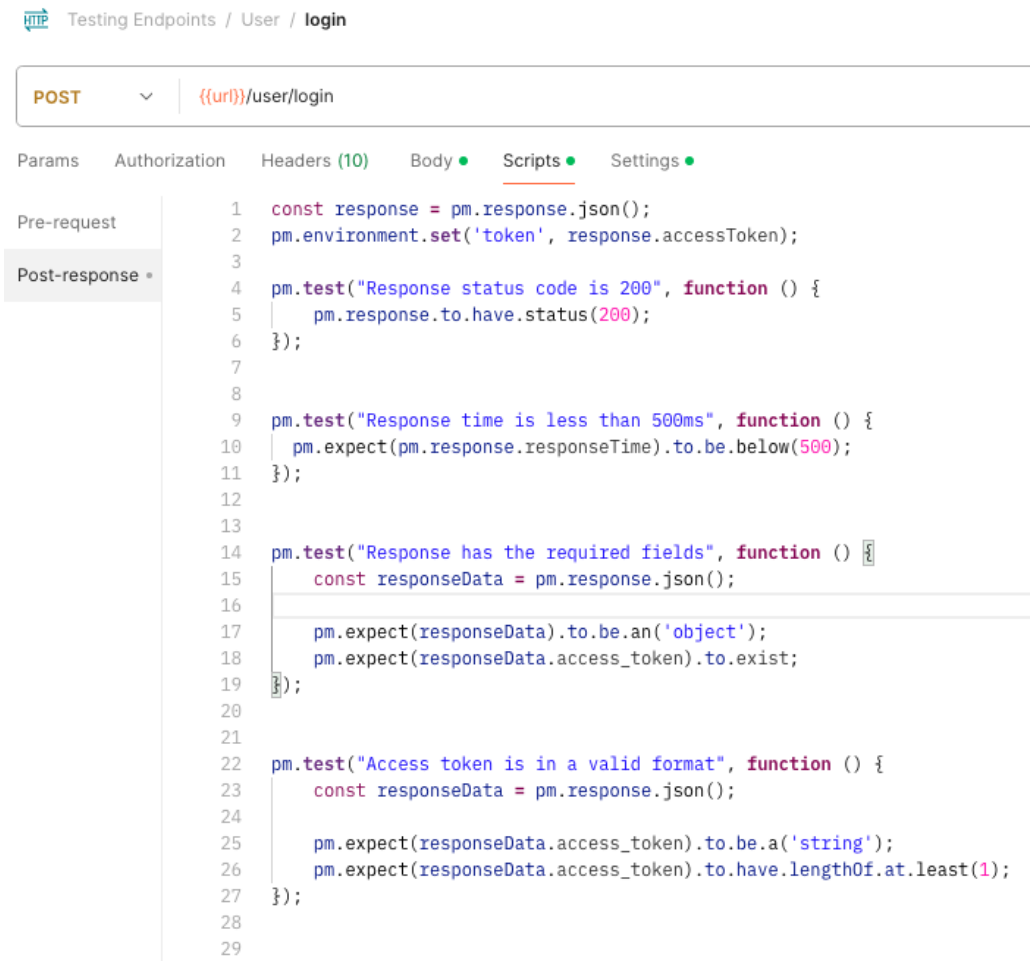
Unit Testing bildet den Kern unserer Teststrategie. Bereits zu Beginn der Implementierungsphase haben wir begonnen, schrittweise alle relevanten Methoden durch Unittests abzusichern. Diese frühzeitige Integration von Tests gewährleistete, dass jede neue Funktionalität oder jeder behobene Fehler die bestehende Funktionalität nicht negativ beeinflusst. Vor dem Zusammenführen von Code in den Entwicklungs- oder Hauptzweig wurden alle Unittests auf korrekte Ausführung geprüft, um sicherzustellen, dass kein fehlerhafter Code in die Produktionszweige gelangt.

Für die Implementierung der Unittests haben wir das Python-Paket `unittest` verwendet.

5.2 Endpoint Testing

Neben den Unittests haben wir auch das Endpoint Testing mithilfe der API-Testing Plattform Postman durchgeführt, um nicht nur einzelne Methoden, sondern auch Aspekte wie die Reaktionszeit zu testen. Innerhalb des Postman Tools haben wir für unser Intuitive Time Tracking Team eine eigene Collection erstellt, die es uns ermöglichte, verschiedene Endpunkte unserer WebApp schnell und effizient zu testen, unabhängig von einem funktionsfähigen Frontend. Dies war besonders hilfreich, um festzustellen, ob bestimmte Fehler durch das Frontend oder Backend verursacht wurden und um die Fehlerbehebung zu beschleunigen.

Die Script-Funktionalität von Postman ermöglichte es uns zudem, die Korrektheit der HTTP-Antworten sowie die Antwortzeiten zu überprüfen, was für ein reibungsloses Nutzererlebnis unerlässlich ist.



Die obige Grafik zeigt ein Beispiel für die von uns genutzte Script-Funktion von Postman am Login-Endpoint. In den ersten zwei Zeilen wird der zurückgegebene Token automatisch in die Umgebungsvariablen geschrieben, um die Weiterverwendung in anderen Requests zu vereinfachen. In den Zeilen vier bis sechs wird überprüft, ob der Statuscode der Response 200 ist, was auf eine erfolgreiche Anfrage hindeutet. Die folgenden Funktionen prüfen die Antwortzeit, die Vollständigkeit der erforderlichen Felder und die Korrektheit des Datenformats.

6 Ausblick

Die Qualitätssicherungsphase beginnt am 23. August und markiert den Abschluss der Implementierungsphase. Zu diesem Zeitpunkt wird eine solide Basis für die Einbindung weiterer Features und die Stabilisierung der Anwendung geschaffen sein. Bis zum Beginn der Qualitätssicherung liegt der Fokus auf mehreren Entwicklungsmeilensteinen:

- **Archivierung gelöschter Nutzer:** Statt Nutzer aus der Datenbank zu entfernen, sollen diese archiviert werden. Im deaktivierten Zustand ist zwar die Anmeldung und die Nutzung weiterer Funktionen blockiert, doch aus Gründen der Dokumentenarchivierung und Rechtssicherheit müssen die Daten erhalten bleiben.

- **In-App Notification Feature:** Implementierung eines Features, das die Interaktivität und Benutzerfreundlichkeit erhöht, indem es Nutzern ermöglicht, Echtzeit-Updates direkt innerhalb der App zu erhalten.

- **Slack Notification Feature:** Entwicklung eines Features, das die Anwendung in bestehende Kommunikationsflüsse einbindet und wichtige Benachrichtigungen direkt in Slack bereitstellt. Dies fördert die Effizienz und Effektivität der Teamkommunikation.

- **Integration von Profilbildern:** Geplant ist die Integration von Profilbildern, um die Personalisierung der Nutzererfahrung zu vertiefen. Dieses Feature ermöglicht die optische Anpassung der Plattform und stärkt die Bindung der Nutzer an die Anwendung.

- **Erhöhung der Testabdeckung:** Systematische Erhöhung der Testabdeckung, um die Zuverlässigkeit und Fehlerfreiheit der Anwendung zu garantieren. Diese Maßnahme ist entscheidend für die Funktionalität und Stabilität der Software.

- **Auswahlmöglichkeit bei Zeiteinträgen:** Bei der Erstellung neuer Zeiteinträge soll zwischen Projektarbeit und Projektmeeting unterschieden werden können, um die Datenerfassung zu präzisieren.

- **Optimierung der Validierungsfunktion:** Die Validierungsfunktionen der Anwendung sollen weiter optimiert werden, um die Datenintegrität und Benutzererfahrung zu verbessern.

- **Starten eines Timesheets am 15. Tag des Monats:** Da Hiwis auch zum 15. jeden Monats ihren Vertrag anfangen können, soll es

möglich sein die Web-App schon im ersten halben Monat zu nutzen, ohne Fehler zu verursachen, da die Software die Arbeitsstunden eines gesamten Monats erwartet.

Abschluss der Entwicklungsphase: Den Abschluss der intensiven Entwicklungsphase bildet die Erstellung eines Docker Images, welches vor dem 23. August geplant ist. Dieses Image wird es ermöglichen, die Anwendung effizient in verschiedenen Umgebungen zu deployen und zu verwalten, was die Skalierbarkeit und Wartbarkeit der Software der Anwendung erleicherten soll.

Mit dem Start der Qualitätssicherungsphase werden alle implementierten Features und Komponenten einer gründlichen Prüfung unterzogen, um sicherzustellen, dass sie den Anforderungen an Qualität und Leistungsfähigkeit gerecht werden. Diese Phase dient nicht nur der Fehlerbehebung, sondern auch der Optimierung der Anwendererfahrung, um die Software zum geplanten Launch in bestmöglichem Zustand zu präsentieren.