pronto
Wir verstehen mehr
als Reinigung.

OST
Ostschweizer
Fachhochschule

Mitarbeiter-Informationsapp für die Pronto AG

Studienarbeit

Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Herbstsemester 2021

Autoren: Damian Kalberer, Gian Flütsch

Betreuer: Mirko Stocker

Projektpartner: Pronto AG

Abstract

In der heutigen Zeit werden digitalisierte Arbeitsprozesse immer wichtiger und bringen einer Firma und deren Mitarbeiter:innen einen grossen Mehrwert. Aus diesem Grund ist es auch für KMUs wichtig, die technischen Fortschritte nicht auszulassen und den ganzen Arbeitsalltag zu optimieren.

In einer vorangegangenen Bachelorarbeit wurde die Grundlage für die Mitarbeiter Informations App der Pronto AG, kurz Pronto-MIA, mit noch geringem Funktionsumfang entwickelt. Im Rahmen dieser Arbeit wurde die Pronto-MIA App, welche die Pronto AG zur Verteilung von Informationen an ihre Mitarbeitenden einsetzen, weiterentwickelt und um zusätzliche Funktionalitäten erweitert.

Die mit dieser Studienarbeit weiterentwickelte Applikation unterstützt die Pronto AG, um wichtige Informationen allen Mitarbeiter:innen möglichst einfach zur Verfügung zu stellen. Die Administration kann neue Einsatzpläne, News, Schulungsunterlagen sowie wichtige Termine erfassen und die Mitarbeiter:innen haben über ihr privates Smartphone Zugriff auf alle für sie relevanten Informationen.

Ein wichtiger Teil der Applikation ist die einfache sowie zielführende Bedienung und Navigation der App. Damit erhält die Pronto AG einen Mehrwert für die Verteilung von Informationen. Um dies zur bestmöglichen Zufriedenheit der späteren Benutzer umzusetzen, wurde ein Usability-Test mit diversen Mitarbeiter:innen durchgeführt und die App an die Bedürfnisse dieser angepasst. Mit diesem Schritt kann eine hohe Zufriedenheit und Akzeptanz für die App gewährleistet werden.

Die Pronto-MIA App kann über den AppStore für iOS sowie über den PlayStore für Android Geräte installiert werden.

Für die Realisierung wurde die Programmiersprache Dart mit dem Flutter-Framework fürs Frontend verwendet. Flutter ist ein Cross-Plattform Framework, welches es ermöglicht aus einer Codebase ein Android-App, ein iOS-App sowie eine Webapplikation zu betreiben. Das Backend ist in ASP.NET geschrieben und über eine GraphQL-API kommuniziert das Frontend mit dem Backend. Die Applikation wurde mit Blick in die Zukunft so entwickelt, dass zusätzliche Funktionalitäten möglichst gut in die App integriert werden können. Zudem wurde darauf geachtet, dass die App einen modernen Eindruck hinterlässt und die Mitarbeiter:innen bestmöglichst unterstützt.

Danksagung

Wir möchten uns bei unserem Betreuer, Mirko Stocker, herzlich bedanken. Er hat uns durch das gesamte Projekt stets unterstützt und war auch ausserhalb der vereinbarten Meetings schnell und unkompliziert erreichbar. Aufgrund seiner Erfahrung konnte er uns mit wertvollen Inputs, welche auch die administrativen Punkte umfassten, stets weiterhelfen.

Weiter möchten wir uns bei Ramon Herzog und Heidi Herzog von der Pronto AG bedanken. Wir konnten während dem ganzen Projekt auf eine sehr gute Zusammenarbeit zählen. Speziell waren Sie beim Usability-Test, welcher bei der Pronto AG vor Ort durchgeführt wurde, sehr zuvorkommend und unterstützend. Somit konnten wir ein aussagekräftiges Feedback von Mitarbeiter:innen einholen, welche die App schlussendlich auch benutzen werden.

Inhaltsverzeichnis

1 Management Summary

In diesem Kapitel wird mithilfe eines Management Summary das Projekt kurz erläutert und einen Überblick verschaffen.

1.1 Ausgangslage

Die Pronto AG ist eine Reinigungsfirma mit über 360 Mitarbeitenden. Diese begeben sich regelmässig zu Kunden um diverse Reinigungs- und Unterhaltsarbeiten durchzuführen. Um diese Einsätze und Aufträge zu koordinieren, nutzt die Pronto AG Einsatzpläne, welche an Aushängen in der Firma für die Mitarbeitenden einsehbar sind. Diese Einsatzpläne bilden die Grundlage für die Verteilung der Arbeitskräfte am darauffolgenden Tag.

Diese Studienarbeit befasst sich mit der Weiterentwicklung einer vorangegangen Bachelorarbeit [1]. Dabei sollen die Arbeitsabläufe digitalisiert werden, damit Mitarbeitende wichtige Informationen wie Einsatzpläne, News, Schulungsvideos und Termine über ihr Smartphone abrufen können.

1.2 Vorgehen

Im Vorfeld der Implementation wurde mit der Pronto AG ein Usability Test mit der entwickelten App aus der vorgängigen Bachelorarbeit durchgeführt. Mit dem Feedback aus dem Usability-Test und den weiteren gewünschten Funktionalitäten haben wir zusammen mit der Pronto AG die Punkte priorisiert und einen Releaseplan daraus erstellt.

Während der Implementation konnten wir uns am zuvor erstellten Releaseplan orientieren und haben iterativ die gewünschten Features umgesetzt.

Nach jeder Iteration wurde die Applikation auf den Plattformen (Web, iOS, Android) deployed und dem Kunden vorgestellt. Anhand der neuen Feedbacks konnten weitere Optimierungen in die nächste Iteration aufgenommen und zur Zufriedenheit des Kunden umgesetzt werden.

1.3 Ergebnisse

Mit dem Abschluss dieser Arbeit wurde die „Pronto MIA“ Applikation optimiert und um zusätzliche Features erweitert. Die Applikation besteht aus einem C#-Backend, GraphQL-API und einem Dart-Frontend. Mit dieser Architektur kann auf die Applikation über eine Website sowie auch über ein Android respektive iOS App zugegriffen werden. Somit werden alle gängigen Plattformen bedient und die Mitarbeiter:innen haben Zugriff auf die für sie relevanten Informationen. Bei einer neuen Veröffentlichung oder Anpassung eines Einsatzplanes werden die betroffenen Mitarbeiter:innen über eine Push-Nachricht informiert.

Die Pronto AG kann mit dieser Applikation Einsatzpläne veröffentlichen, interne und externe News publizieren, Schulungsvideos veröffentlichen sowie wichtige Termine im integrierten Kalender eintragen. Weiter ist es möglich externe News anzusehen, Kundenanfragen über die App einzureichen und Kontaktdaten der Pronto AG abzurufen.

Die iOS-App kann aufgrund von strengen Restriktionen seitens Apple nicht aus dem öffentlichen App Store heruntergeladen werden. Dafür musste auf den Business Manager zurückgegriffen werden. Über Redemption-Tokens können die Mitarbeiter:innen die App aber trotzdem auf ihren privaten Geräten ohne Probleme installieren.

1.4 Ausblick

Die Pronto-MIA Applikation umfasst nun die für den täglichen Betrieb wichtigsten Funktionen, sodass mit der produktiven Einführung und Schulung bei den Mitarbeiter:innen begonnen werden kann. Wir sind überzeugt, die Applikation der Pronto AG in einem guten Zustand übergeben zu können und denken, dass sie eine Bereicherung für den täglichen Betrieb bei der Administration aber auch für die Mitarbeiter:innen ist.

Teil I

Projektplan

2 Vorgängige Bachelorarbeit

Die Studienarbeit baut auf einer Bachelorarbeit aus dem Frühjahrssemester 2021 auf.

Im Kapitel 5 sowie 6 haben wir Teilabschnitte aus der erwähnten Arbeit übernommen, da sich die Ausgangslage sowie die Organisation für uns nicht geändert hat. Die bereits definierten Qualitätsmassnahmen erachten wir als sinnvoll und werden diese deshalb auch in unserer Arbeit so weiterführen.

3 Übersicht

Im Abschnitt „Projektplan“ wird die Aufgabenstellung erläutert, der Zeitplan definiert und das Projektvorgehen geplant. Ausserdem werden Projektprozesse sowie das Risiko und Qualitätsmanagement spezifiziert. Zusammengefasst geht es in diesem Teil um organisatorische Aspekte, welche vor dem technischen Teil des Projektes definiert werden müssen.

3.1 Aufgabenstellung

Die Pronto AG ist ein Komplettanbieter für Reinigungen aller Art in der Ost- und Innerschweiz. Um die Kommunikation mit den rund 360 Mitarbeitenden zu optimieren, soll eine bestehende Cross-Plattform Mobile App weiterentwickelt werden.

Im letzten Semester wurden bereits die Grundlagen einer Cross-Plattform (Android, iOS, Web) App mit Flutter (Backend mit .NET, GraphQL zum Datenaustausch) geschaffen und ein erstes Feature (Verwaltung von Einsatzplänen) end-to-end implementiert. Auf dieser Basis sollen nun folgende Features umgesetzt werden:

- Einreichung Frei- und Ferienanträge
- Bereich für Schulungsvideos
- Bereich für News / Kalender / Informationen

Die bereits beschlossenen Architekturentscheidungen sollen hinterfragt und falls nötig angepasst werden.
Auch das User Interface soll initial einem Usability-Test unterzogen werden.

Diese Aufgabenstellung wurde aus der Original-Aufgabenstellung im Anhang entnommen.

3.2 Erwartete Resultate

- Software in Form einer Cross-Plattform Mobile App, welche die in der vorgegebenen Zeit möglichen Features in einem dem Kunden zufriedenstellenden Umfang beinhaltet.
- Dokumentation der Arbeit (inklusive Vorgehen und kritischer Bewertung der getroffenen Entscheide).

3.3 Einschränkungen

Das Modul „Studienarbeit“ ist zeitlich begrenzt. Das Team hat gemeinsam 480 Stunden Zeit, die Projektarbeit in 14 Wochen durchzuführen. Diese zeitliche Begrenzung setzt sich folgendermassen zusammen:

- 480h = 2 Teammitglieder * 8 ECTS Punkte * 30h pro ECTS
- 14 Semesterwochen

Die Studienarbeit beginnt offiziell am 20.09.2021 und endet mit der Abgabe am 24.12.2021 um 17:00 Uhr.

3.4 Persönliche Ziele

Neben dem Endprodukt umfasst das Projekt folgende persönliche Ziele des Teams:

- Erfolgreiche Weiterführung eines vorgegebenen Projektes in der zur Verfügung gestellten Zeit.
- Weiterentwicklung eines Endproduktes, welches einen realen Nutzen darstellt und ein Bedürfnis befriedigt.
- Erfahrungen sammeln für weitere schulische, private und berufliche Projekte.
- Kennenlernen der Entwicklung von Cross-Plattform Mobile Apps.
- Kompetenzen im Bereich Kundeninteraktion erweitern.

4 Zeitplan

In diesem Kapitel wird näher auf den zeitlichen Aspekt und die geplanten Schritte eingegangen. Ebenfalls werden Meilensteine definiert und beschrieben. Zusätzlich wird auf die verschiedenen Phasen nach Scrum+ eingegangen und näher erläutert.

4.1 Übersicht

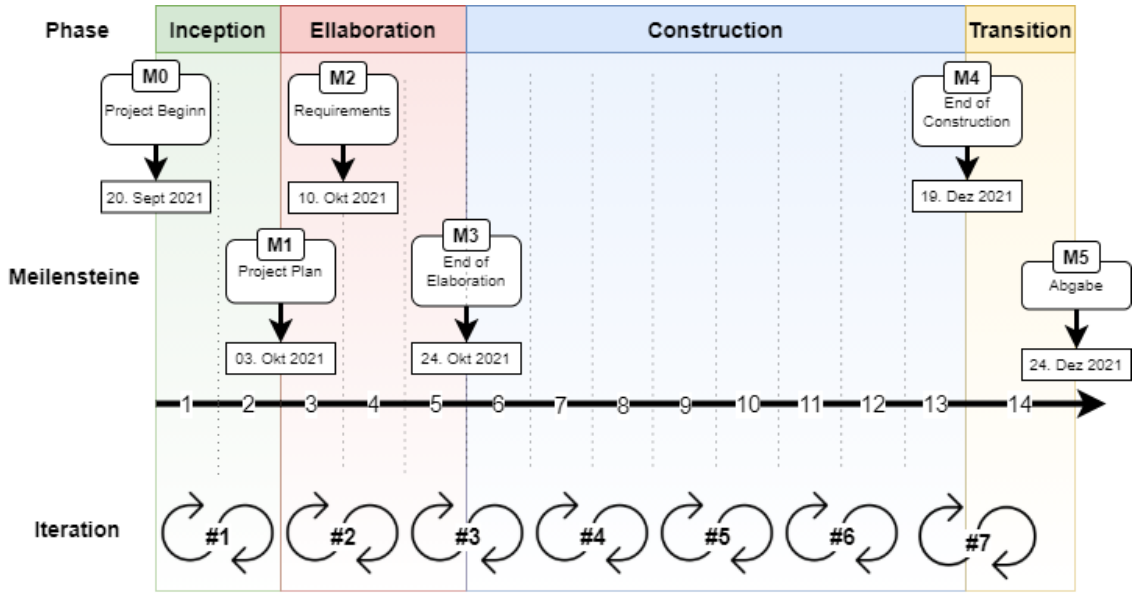

Abbildung 4.1: Übersicht Zeitplan

4.2 Phasen

Das Projekt wird nachfolgend, gemäss Projektmethodik aus Kapitel 5.3, in Phasen unterteilt. Das bedeutet konkret:

- Die Arbeit wird in 4 Phasen geplant.
- In jeder Phase gibt es Iterationen.
- Meilensteine werden auf das Ende von Iterationen gelegt.
- Es wird bei jeder Iteration (in der „Construction“-Phase) ein potentially shippable product angestrebt.
- Nach jeder Iteration findet eine Sprint-Review-Sitzung statt, um den Backlog mit Arbeitspaketen zu pflegen und das weitere Vorgehen zu besprechen.
- Nach jeder Iteration findet ein Austausch über das agile Vorgehen statt (Sprint Retrospective), um Probleme im Projektmanagement zu vermeiden.
- Es wird Wert auf ein sauberes End of Elaboration gelegt. Dazu wird die End of Elaboration Checklist verwendet.
- Am Ende der „Construction“-Phase gibt es einen Feature-Freeze.

4.2.1 Inception

- **Dauer:** 2 Wochen
- **Zeitraum:** 20.09.21 - 03.10.21

Obwohl ein agiles Vorgehen für das Projekt angestrebt wird, wird das Projekt in dieser ersten Phase grob durchgeplant. Dies dient in erster Linie zur Aufteilung der begrenzten Projektzeit, um spätere Zeitengpässe zu vermeiden. Ausserdem ermöglicht der Projektplan, für alle Beteiligten einen gemeinsamen Ausgangspunkt zu schaffen und die Rahmenbedingungen zu definieren.

Des Weiteren werden die Anforderungen des Kunden aufgenommen und erstmalig spezifiziert. Die Spezifikation wird in einem weiteren Schritt mit dem Kunden besprochen.

4.2.2 Elaboration

- **Dauer:** 3 Wochen
- **Zeitraum:** 04.10.21 - 24.10.21

In einem ersten Schritt werden in dieser Projektphase die Anforderungen mit dem Kunden ausgearbeitet. Durch die agile Natur des Projektes können sich diese Anforderungen im Laufe dieser Arbeit noch ändern. Es ist jedoch sinnvoll die Bedürfnisse des Kunden möglichst genau zu erfassen, damit diese bei der anschliessenden Planung der Architektur auch berücksichtigt werden können.

Weiter wird die technische Machbarkeit des Projektes überprüft. Dies geschieht mit einem minimalen Prototypen, welcher die wichtigsten Aspekte der geplanten Architektur abbildet. So sind am Ende der „Elaboration-Phase“ alle bereits bekannten Unklarheiten beseitigt, was einen gut vorbereiteten Start der Implementation ermöglicht.

4.2.3 Construction

- **Dauer:** 8 Wochen
- **Zeitraum:** 25.10.21 - 19.12.21

Während dieser dritten und längsten Phase wird hauptsächlich das geplante Produkt umgesetzt. Parallel dazu werden zur Gewährleistung der Code-Qualität Tests geschrieben und die Dokumentation nachgeführt.

In dieser Phase ist das Feedback des Kunden besonders wichtig, um das Endprodukt möglichst nach dessen Vorstellungen gestalten zu können. Um dies zusätzlich zu überprüfen werden Usability-Tests durchgeführt.

Am Ende der „Construction“-Phase findet ein Feature-Freeze statt. Dies bedeutet, dass keine neue Funktionalität mehr zum Produkt hinzugefügt wird.

4.2.4 Transition

- **Dauer:** 1 Woche
- **Zeitraum:** 20.12.21 - 24.12.21

Diese letzte Phase dient zur Behebung verbleibender Fehler, zur Qualitätsprüfung und zur Übergabe des Produkts an dessen weiteren Betreiber. Ausserdem wird das Projekt ausgewertet und letzte Verbesserungen an der Dokumentation vorgenommen, damit diese zur Abgabe bereit ist.

4.3 Iterationen / Sprints

Ein Sprint dauert stets zwei Wochen. Die Sprints beginnen am Montag und werden am Sonntag abgeschlossen. Ausgewertet werden die Sprints am zweiten Freitag mit anschliessender Planung des darauffolgenden Sprints. Die Auswertung am Freitag erlaubt es, noch nicht abgeschlossene Tätigkeiten bis Sonntag nachzuliefern.

Insgesamt werden in diesem Projekt sieben Sprints durchgeführt.

4.4 Meilensteine

Nachstehend werden die einzelnen Meilensteine spezifiziert. Jeder dieser Meilensteine enthält Arbeitsprodukte, welche in die folgenden Kategorien eingeteilt werden:

- **Priorität 1:** Arbeitsprodukt, welches zwingend vor dem entsprechenden Meilenstein fertiggestellt werden muss.
- **Priorität 2:** Arbeitsprodukt, welches auch in der darauffolgenden Iteration beendet werden kann.

4.4.1 Projektplan

Ziel:	Projektplanung abgeschlossen
Zeitpunkt:	Ende Sprint 1 am 03.09.21
Arbeitsprodukte:	• Priorität 1 – Der Projektplan ist inklusive Projektorganisation, Zeitplan, Qualitätsmanagement und Risikoanalyse fertiggestellt. • Priorität 2 – keine

Tabelle 4.1: Meilenstein: Projektplan

4.4.2 Requirements

Ziel:	Anforderungsanalyse abgeschlossen
Zeitpunkt:	Halbzeit Sprint 2 am 10.10.21
Arbeitsprodukte:	• Priorität 1 – Requirements definiert – Usability-Test ausgearbeitet • Priorität 2 – keine

Tabelle 4.2: Meilenstein: Requirements

4.4.3 End of Elaboration

Ziel:	Alle bekannten Unsicherheiten beseitigt
Zeitpunkt:	Halbzeit Sprint 3 am 24.10.21
Arbeitsprodukte:	• Priorität 1 – End of Elaboration Checklist überprüft – Usability-Test durchgeführt – Architektur überprüft • Priorität 2 – Dokumentation ist auf dem neusten Stand

Tabelle 4.3: Meilenstein: End of Elaboration

4.4.4 End Of Construction

Ziel:	Geplante Funktionalität implementiert
Zeitpunkt:	Halbzeit Sprint 7 am 19.12.21
Arbeitsprodukte:	• Priorität 1 – Alle geplanten Funktionalitäten implementiert – Nach jedem Sprint das App im PlayStore/ AppStore veröffentlicht • Priorität 2 – Dokumentation ist auf dem neusten Stand

Tabelle 4.4: Meilenstein: End of Construction

4.4.5 Abgabe

Ziel:	Projekt abgeschlossen
Zeitpunkt:	Abgabetermin der Studienarbeit am 24.12.21
Arbeitsprodukte:	• Priorität 1 – Dokumentation fertiggestellt – Dokumente auf https://archiv.i.ost.ch/ hochgeladen • Priorität 2 – Das Projekt wurde dem Kunden zum Betrieb übergeben.

Tabelle 4.5: Meilenstein: Abgabe

5 Organisation

In diesem Kapitel wird die Organisation innerhalb des Projektes und die Verteilung der Aufgaben beschrieben. Auch werden Vorgehensweisen, wie die Projektmethodik und die Projektabläufe, definiert. Erwähnt werden ausserdem die Werkzeuge, welche für das Projektmanagement eingesetzt wurden.

5.1 Rollenverteilung

Das Projektteam besteht aus zwei Personen – Damian Kalberer und Gian Flütsch. Mirko Stocker übernimmt die Betreuung sowie die Bewertung. Der Kunde, die Pronto AG, wird durch Ramon Herzog (Leiter Innovation und Entwicklung) sowie Heidi Herzog (Leiterin Administration) vertreten. Heidi Herzog steht ausserdem bei Fragen zur IT Infrastruktur zur Verfügung.

Während der Arbeit wird es keine fixe Aufteilung der Aufgaben geben.

- **OST – Ostschweizer Fachhochschule**
  - Mirko Stocker – Betreuung / Bewertung
  - Damian Kalberer – Studierender
  - Gian Flütsch – Studierender
- **Pronto AG**
  - Ramon Herzog – Leiter Innovation und Entwicklung
  - Heidi Herzog – Leiterin Administration

5.2 Besprechungen

Besprechungen mit dem Betreuer werden voraussichtlich wöchentlich abgehalten, jedoch wird dies nach Bedarf geregelt. Mit dem Kunden wird versucht, sich so oft wie möglich und nötig auszutauschen, um ein zufriedenstellendes Endresultat zu erzielen. In beiden Fällen wird jedoch auf fixe Termine verzichtet. Innerhalb des Projektteams gibt es am Ende jedes Sprints ein Review- und Planungsmeeting. Weitere Besprechungen werden, dank des kleinen Teams, spontan und der Notwendigkeit angepasst einberufen.

5.3 Projektmethodik

Es wurden diverse Projektmethodiken evaluiert. Genauer gesagt Rational Unified Process. (RUP), Wasserfall, Scrum und Scrum+. Anschliessend wurden Faktoren eruiert, welche die Wahl der Methodik beeinflussen:

1. Das Projektteam hat noch keine Erfahrungen in der Entwicklung von Cross-Plattform Mobile Apps und kann sich dementsprechend nicht auf historische Daten beziehen. Da es sich um das Weiterführen eines bestehenden Projektes handelt, liegt der agile Ansatz nahe.
2. Die Arbeit erfordert einiges an Exploration, was ebenfalls für einen agilen Ansatz spricht.
3. Das Projektteam konnte bereits im Engineeringprojekt Erfahrungen mit der Methodik Scrum+ sammeln und fühlt sich daher mit dieser am sichersten.

Da die oben genannten Punkte einen agilen Ansatz nahelegen, ein End of Elaboration aber trotzdem als notwendig empfunden wird, fällt die Wahl auf Scrum+.

5.4 Projektabläufe

Während des ganzen Projektes finden zwei zentrale Abläufe statt. Zum Einen ist dies das Sprint-Review und -Planning nach jeder Iteration und zum Anderen das Abarbeiten der Arbeitspakete.

5.4.1 Sprint-Review und Planning

Am Ende jeder Iteration erfolgt ein Sprint Review Meeting, bei welchem die Arbeitspakete des letzten Sprints überprüft werden. Konnten gewisse Arbeitspakete nicht abgeschlossen werden, oder wurden diese nach einem erneuten Review als nicht abgeschlossen angesehen, werden sie in den nächsten Sprint übernommen.

Sind alle Arbeitspakete besprochen, werden die Neuen evaluiert, aus dem Backlog entnommen und für den nächsten Sprint eingeplant.

5.4.2 Arbeitspakete

Der Entwicklungs- sowie der Dokumentationsprozess basieren auf dem Feature Branch Workflow. Dabei werden wir uns an der Atlassian Feature Branch Workflow Dokumentation orientieren [2]. Es wird allerdings auf den Branch *develop* verzichtet. Der letzte stabile Release wird stattdessen durch einen Tag markiert.

Für den Workflow gelten folgende Regeln:

- Es existiert ein Branch *master*, welcher als Integrationsbranch aller Feature-Branches fungiert.
  - Pro Version wird auf dem Branch *master* ein Tag gesetzt und es findet ein neuer Release statt.
- Für jedes Feature wird ein eigener Feature-Branch erstellt.

- Ein Feature-Branch wird ausgehend von einem Issue erstellt.
- Ein Feature-Branch wird per Pull-Request in den Branch *master* integriert.

Ausserdem gelten spezifisch für Arbeitspakete folgende Regeln:

- Arbeitspakete werden als Issues erstellt.
- Die Zeitschätzung und Rapportierung erfolgt über Clockify.
- Nach Abschluss der zu erwartenden Arbeit im Issue wird ein Review durchgeführt.
- Es wird erst ein Tag gesetzt, wenn die Definition of Done für alle Arbeitspakete im Release gemäss Qualitätsmanagement erreicht wurde.

Der folgende Ablauf gilt für Issues jeglicher Art, unabhängig davon, ob es sich dabei um Dokumentation oder Implementation handelt. Der Arbeitsschritt „Tests schreiben“ fällt für die Dokumentation weg:

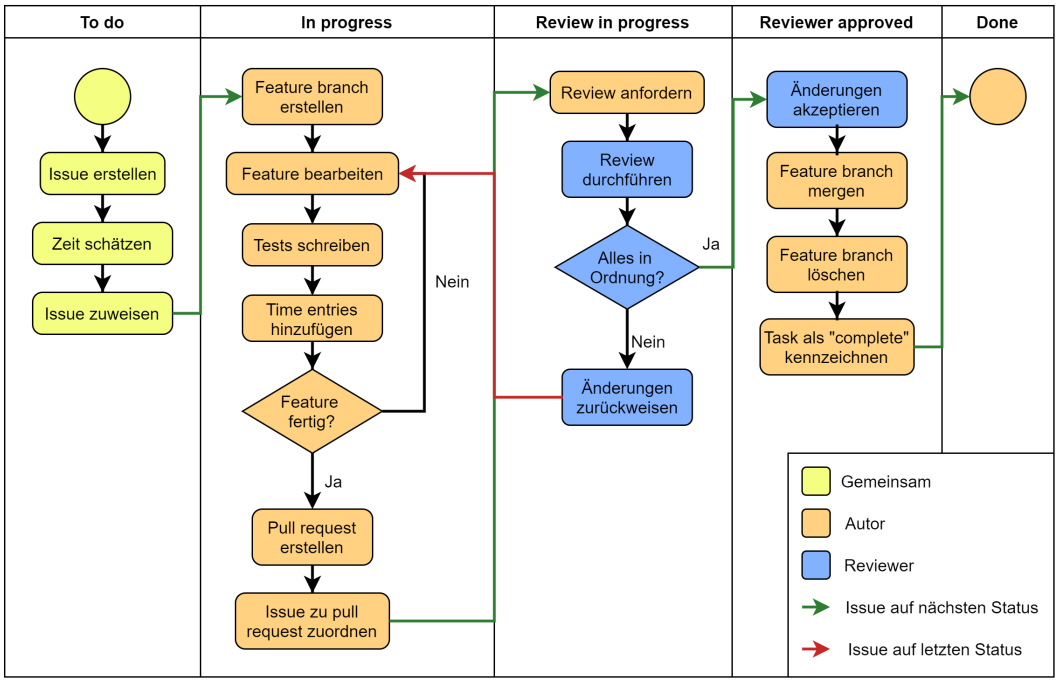

Abbildung 5.1: Lebenszyklus eines Arbeitspakets, übernommen aus [1]

5.5 Tools

Um das Projektmanagement zu erleichtern werden folgende Tools eingesetzt:

- GitHub:
  - Zur sauberen Aufteilung wird mit verschiedenen Repositories für Code und Dokumentation gearbeitet.
  - GitHub-Projects ermöglicht eine Übersicht über alle Issues in Form eines Kanban-Boards, aufgeteilt nach Status.
- LaTeX:
  - LaTeX wird verwendet, um die Dokumentation zu verfassen.
  - Der textbasierte LaTeX-Quellcode ermöglicht es, auch die Dokumentation sauber über Git zu versionieren.
- Clockify:
  - Clockify ermöglicht die Schätzung von Zeitaufwänden, deren Rapportierung und Auswertung.
  - Mithilfe einer Browsererweiterung lassen sich GitHub Issues direkt als Tasks in Clockify übertragen.
- Microsoft Teams:
  - Microsoft Teams wird als primärer Kommunikationskanal innerhalb, aber auch ausserhalb des Projektteams genutzt.
  - Es dient als Ersatz für persönliche Meetings während der Covid-19 Pandemie.

6 Qualitätsmanagement

In diesem Kapitel wird beschrieben, welche Massnahmen ergriffen werden um die Qualität der Dokumentation als auch des Endproduktes zu garantieren.

6.1 Qualitätsmassnahmen

Zeitraum	**Massnahme**	**Ziel**
Freitags	Sprint Review	Austausch und Review über den Stand der Arbeitspakete.
Freitags	Sprint Planning	Neuen Sprint mit Arbeitspaketen aus Backlog planen. Falls nötig Prioritäten anpassen.
Donnerstags	Meeting mit Projektbetreuung	Allfällige Unklarheiten klären, Projektestand besprechen.
Kontinuierlich	Continuous Integration	Durchführen von automatisierten Tests und statischer Code-Analyse.
Kontinuierlich	Code Review	Gegenseitige Codereviews bevor Features/Bugs in den Master/Development-Branch integriert werden.
Kontinuierlich	Dokumentationsreview	Gegenseitige Dokumentationsreviews bevor Änderungen in den Master/Development-Branch integriert werden.

Tabelle 6.1: Eingeplante Qualitätsmassnahmen

6.1.1 Projektmanagement

Die Basis für das Projektmanagement bildet ein Projektboard, welches über GitHub realisiert wird. Arbeitspakete werden als Issues angelegt und dem entsprechenden Meilenstein im Repository zugeordnet. Dies ermöglicht uns eine Übersicht über sämtliche Arbeitspakete und deren aktuellen Status. Die Zeiterfassung über das gesamte Projekt erfolgt über Clockify. Weitere Tools werden für das Projektmanagement vorerst nicht eingesetzt. Die Projektbetreuung hat Zugriff auf die GitHub Organisation und somit Zugriff auf sämtliche Repositories und den Stand der Arbeiten.

6.1.2 Dokumentation

Die Dokumentation befindet sich in einem eigenen Repository auf GitHub. Die Qualität bezüglich Inhalt, Formatierung und Grammatik wird mittels Peer-Reviews vor dem integrieren der Pull Requests gewährleistet.

6.1.3 Entwicklung

Der Source Code wird mit Git versioniert und befindet sich getrennt von der Dokumentation auf einem separaten Git-Repository auf GitHub.

Die Qualität des Source Codes wird, in einem ersten Schritt, automatisiert geprüft. Dies erfolgt im Rahmen einer Continuous Integration durch einen Linter, gegen ein vordefiniertes Stylesheet. Des Weiteren werden in dieser Continuous Integration auch Unit und Integration Tests durchgeführt. In einem letzten Schritt wird die Codequalität zusätzlich durch ein Peer-Review der Pull Requests auf GitHub geprüft.

6.2 Definition of Done

Um Unklarheiten bezüglich der Vollständigkeit von Arbeitspaketen, Sprints und Meilensteine zu beseitigen, wird für diese in den folgenden Abschnitten eine Definition of Done festgelegt.

6.2.1 Arbeitspaket

- Automatisierte Tests in der Continuous Integration zeigen keine Fehler an, allfällige Warnungen wurden überprüft und abgearbeitet.
- Die GitHub Action-Pipeline ist erfolgreich durchlaufen.
- Sämtliche Unit Tests wurden fehlerfrei ausgeführt.
- Die nicht-funktionalen Anforderungen sind/bleiben erfüllt.
- Es wurde ein Review des Codes beziehungsweise der Dokumentationsänderungen durchgeführt.

6.2.2 Sprint

- Nicht abgeschlossene Arbeitspakete in den nächsten Sprint verschoben
- Definition of Done für alle verbleibenden Arbeitspakete im Sprint erreicht
- Sprint Review durchgeführt

6.2.3 Meilenstein

- Definition of Done für alle Sprints im Meilenstein erreicht
- Die geplanten Resultate des Meilensteins fertiggestellt

6.3 Testing

In diesem Kapitel wird beschrieben, welche Arten von Tests durchgeführt werden. Auch wird definiert, wann diese Tests erstellt und wie diese dokumentiert werden.

6.3.1 Unit und Integration Tests

Um eine gute Testabdeckung zu garantieren, zum Schreiben der Tests genügend Zeit eingeplant. Diese werden im Review angeschaut und auf ihre Tauglichkeit geprüft. Priorisiert wird der anwendungskritische Code, konkret die Businesslogik.

Trivialer Code, wie beispielsweise Getter und Setter, wird nicht getestet. Es kann davon ausgegangen werden, dass bei Fehlern in diesem Code Folgefehler auftreten. Diese werden in weiteren Tests abgefangen. Ausserdem wird generell auf Tests für Code verzichtet, der lediglich als Bindeglied zwischen zwei Bibliotheken fungiert.

6.3.2 Systemtests

Anhand der definierten Use-Cases werden die Systemtests durchgeführt und ausgewertet. Dies wird regelmässig manuell von den Teammitgliedern erledigt.

6.3.3 Nichtfunktionale Tests

Basierend auf den nicht-funktionalen Anforderungen werden geeignete nicht-funktionale Tests durchgeführt. Bereits geplant ist ein Usability-Test zu Beginn des Projektes.

6.3.4 Abnahmetests

Es ist ein Abnahmetest mit dem Endkunden vorgesehen. Dieser wird voraussichtlich an einer der letzten Sitzungen mit dem Kunden durchgeführt.

7 Risikomanagement

Arbeitspakete werden gemäss ihrem Risiko priorisiert. Je grösser das Risiko, dass ein Arbeitspaket scheitert, desto früher wird es erledigt. So soll möglichst früh ein „Durchstich“ durch alle technischen Schichten erfolgen, damit technische Probleme frühzeitig erkannt werden. Am Schluss eingeplant sind „Nice to have“ - Funktionalitäten, wie das Aufteilen der Einsatzpläne nach Abteilung oder der Bereich für Schulungsvideos, da diese bei Zeitknappheit weggelassen werden können.

Es werden nur wenige zeitliche Reserven gebildet – stattdessen ist das Projekt so geplant, dass möglichst früh ein funktionierender Prototyp vorhanden ist, der im weiteren Verlauf optimiert und ausgebaut wird. So kann auch dann ein lauffähiges Produkt ausgeliefert werden, wenn zeitlich bedingt nicht alle Arbeitspakete abgeschlossen werden konnten.

Als minimaler Puffer ist die Zeit zwischen dem Meilenstein „Qualitätsmassnahmen“ und dem Meilenstein „End of Construction“ vorgesehen. Ausserdem ist die letzte Woche der Studienarbeit für abschliessende Arbeiten vorgesehen und kann somit als zusätzlicher Zeitpuffer dienen.

7.1 Risiken

In diesem Kapitel geht es um die evaluierten Risiken und die Abschätzung, welchen Einfluss diese auf den Verlauf der Arbeit haben könnten.

7.1.1 Risiko: Ausfall eines Teammitglieds

Da das Team nur aus zwei Personen besteht, hat der Ausfall eines Mitglieds einen starken Einfluss auf den Verlauf des Projektes. Gerade zu Zeiten von Covid ist es möglich, dass ein Mitglied krankheitshalber ausfällt.

- **Auswirkungen:** Der Zeitplan kann je nach Schweregrad der Erkrankung nicht mehr eingehalten werden.
- **Maximaler Schaden:** 36h
- **Eintrittswahrscheinlichkeit:** 30%
- **Gewichteter Schaden:** 10.8h
- **Vorbeugung:** Durch regelmässige Absprachen im Team wird sichergestellt, dass alle Teammitglieder auf demselben Informationsstand sind. Dadurch sind bei einem Ausfall alle nötigen Informationen zur Forsetzung der Arbeit vorhanden.

- **Verhalten beim Eintreten:** Die erkrankte Person arbeitet in dem Mass weiter, wie es ihr möglich ist. Die andere Person arbeitet normal weiter. Gegebenenfalls muss mit dem Kunden eine Reduktion des Umfangs angeschaut werden, um den Ausfall zu kompensieren.

7.1.2 Risiko: Missverständnisse in der Kommunikation

In diesem Projekt arbeiten drei Parteien zusammen. Aufgrund dessen, kann es zu Missverständnissen und Fehlinterpretationen kommen, welche Zeit kosten.

- **Auswirkungen:** Je nach Missverständnis muss im schlimmsten Fall ein ganzer Anwendungsfall neu implementiert werden.
- **Maximaler Schaden:** 48h
- **Eintrittswahrscheinlichkeit:** 20%
- **Gewichteter Schaden:** 9.6h
- **Vorbeugung:** Die Anwendungsfälle werden früh spezifiziert und mit dem Kunden abgesprochen. Während der Umsetzung wird regelmässig Feedback vom Kunden eingeholt.
- **Verhalten beim Eintreten:** Die Betreuungsperson sowie der Kunde werden über das Missverständnis informiert. Anschliessend wird besprochen wie weiter vorgegangen werden soll.

7.1.3 Risiko: Wartezeiten beim Deployment Prozess

Um die App auf dem PlayStore oder AppStore zu veröffentlichen, muss ein Prozess von Google/Apple durchlaufen werden. Dieser beinhaltet einige Stolpersteine, aufgrund welcher es möglich sein kann, dass ein Deployment länger geht als erwartet oder zurückgewiesen wird und wieder von neu begonnen werden muss.

- **Auswirkungen:** Anpassungen am Code oder am Deployment
- **Maximaler Schaden:** 16h
- **Eintrittswahrscheinlichkeit:** 50%
- **Gewichteter Schaden:** 8h
- **Vorbeugung:** Das Deployment wird so früh wie möglich lanciert, damit auf Probleme rechtzeitig reagiert werden kann.
- **Verhalten beim Eintreten:** Feedback vom Deployment Prozess umsetzen und wenn nötig Kunden und Betreuer informieren, wann mit der Veröffentlichung gerechnet werden kann. Bei Sprint Review lokale Lösung vorstellen.

7.1.4 Risiko: Probleme bei Einarbeitung ins Projekt

Da es sich um ein bereits bestehendes Projekt handelt, könnten Schwierigkeiten oder Probleme bei der Einarbeitung auftreten. Die meisten Technologien kennen wir nicht und die Dokumentation ist möglicherweise ungenügend.

- **Auswirkungen:** Sehr viel Mehraufwand um sich zu informieren. Im schlimmsten Fall, Lösung von selbst neu aufbauen.
- **Maximaler Schaden:** 60h
- **Eintrittswahrscheinlichkeit:** 10%
- **Gewichteter Schaden:** 6h
- **Vorbeugung:** Die Dokumentation und die Applikation werden zu Beginn des Projektes einem Review unterzogen, damit geprüft werden kann, ob unser Team sich damit zurechtfindet.
- **Verhalten beim Eintreten:** Betreuer muss informiert werden und es werden Massnahmen miteinander besprochen.

7.1.5 Risiko: Bestehende Architektur nicht passend für neue Anforderungen

Die Architektur wurde von einem abgeschlossenen Projekt übernommen. Dies kann Probleme verursachen, da wir für neue Anforderungen möglicherweise die Architektur anpassen müssen.

- **Auswirkungen:** Architektur anpassen
- **Maximaler Schaden:** 24h
- **Eintrittswahrscheinlichkeit:** 20%
- **Gewichteter Schaden:** 4.8h
- **Vorbeugung:** Die Architektur wird in der Elaboration Phase überprüft und mit unseren Anforderungen abgeglichen.
- **Verhalten beim Eintreten:** Architektur neu planen und anpassen. Möglicherweise müssen Arbeitspakete nach hinten verschoben werden.

7.1.6 Risiko: Applikation kann nicht im AppStore aufgeschalten werden

Apple hat sehr harte Kriterien, welche Applikationen in ihrem AppStore aufgeschaltet werden. Es ist schwierig alle Anforderungen abzudecken, damit die App im öffentlichen AppStore aufgeschalten wird. Da wir keine Erfahrungen damit haben, kann es dadurch aufwändiger werden.

- **Auswirkungen:** Applikation anpassen
- **Maximaler Schaden:** 16h
- **Eintrittswahrscheinlichkeit:** 40%
- **Gewichteter Schaden:** 6.4h
- **Vorbeugung:** Zu Beginn des Projektes wird dieses Thema gleich aufgegriffen, damit wir mögliche Änderungen an der Applikation einfach implementieren können.
- **Verhalten beim Eintreten:** Eine Lösung finden, wie es möglich ist die App aufzuschalten und diese mit den Stakeholdern besprechen.

7.2 Erkenntnisse

Der maximal gewichtete Schaden beträgt 45.6h. Aufgrund der frühen Erkennung von Problemen, sollte es möglich sein, diesen Schaden zu minimieren. Sollten im schlimmsten Fall mehrere/alle Risiken eintreffen, muss ein Treffen mit der Betreuungsperson und dem Kunden gehalten werden um das Projekt neu zu evaluieren.

Teil II

Requirements

8 Übersicht

Im Abschnitt *Vorstudie* werden die für die Umsetzung der Arbeit notwendigen Informationen zusammengetragen. Ausserdem werden Anforderungen, welche an das Endergebnis gestellt werden, definiert. Zu guter Letzt werden die erkannten Herausforderungen spezifiziert, analysiert und die daraus entstandenen Entscheidungen dokumentiert.

8.1 Vorgängige Bachelorarbeit

Die Studienarbeit baut auf einer Bachelorarbeit aus dem Frühjahrssemester 2021 auf.

Im Kapitel 9 haben wir einen grossen Teilabschnitt von [1] übernommen, da sich die Ausgangslage für uns nicht geändert hat und die konzeptionelle Planung für alle Anforderungen bereits erstellt wurde.

9 Anforderungsanalyse

In diesem Kapitel geht es darum, die Anforderungen an die Applikation aufzunehmen und diese anschliessend zu spezifizieren sowie zu dokumentieren. Hierbei ist eine gute Zusammenarbeit mit dem Kunden sehr wichtig, damit Missverständnisse und daraus resultierende Anpassungen vermieden werden können.

9.1 Lizenzierung

Aufgrund der Tatsache, dass der Kunde keine Anforderungen an die Lizenzierung stellt, ist das Entwicklerteam in deren Wahl frei. Die Entscheidung viel dabei auf die MIT-Lizenz. Diese Lizenz ist sehr offen und entwicklerfreundlich. Dadurch wird verhindert, dass Lizenzbedingungen allfällig verwendeter Software-Bibliotheken verletzt werden.

Es wurde ebenfalls entschieden, den Quellcode öffentlich einsehbar zu machen, damit andere Firmen oder Privatpersonen auch von diesem profitieren können. Auch können dadurch allfällige Fehler mithilfe der Community schneller erkannt werden und es stehen den Entwicklern mehr Tools ohne Zusatzkosten zur Verfügung.

9.2 Funktionale Anforderungen (Use Cases)

In diesem Kapitel werden die funktionalen Anforderungen an die Applikation definiert. Diese wurden zusammen mit dem Kunden erarbeitet. In der Abbildung 9.1 sind diese grafisch dargestellt. Die in der Grafik dargestellten Aktoren, werden in Kapitel 9.2.1 noch genauer spezifiziert.

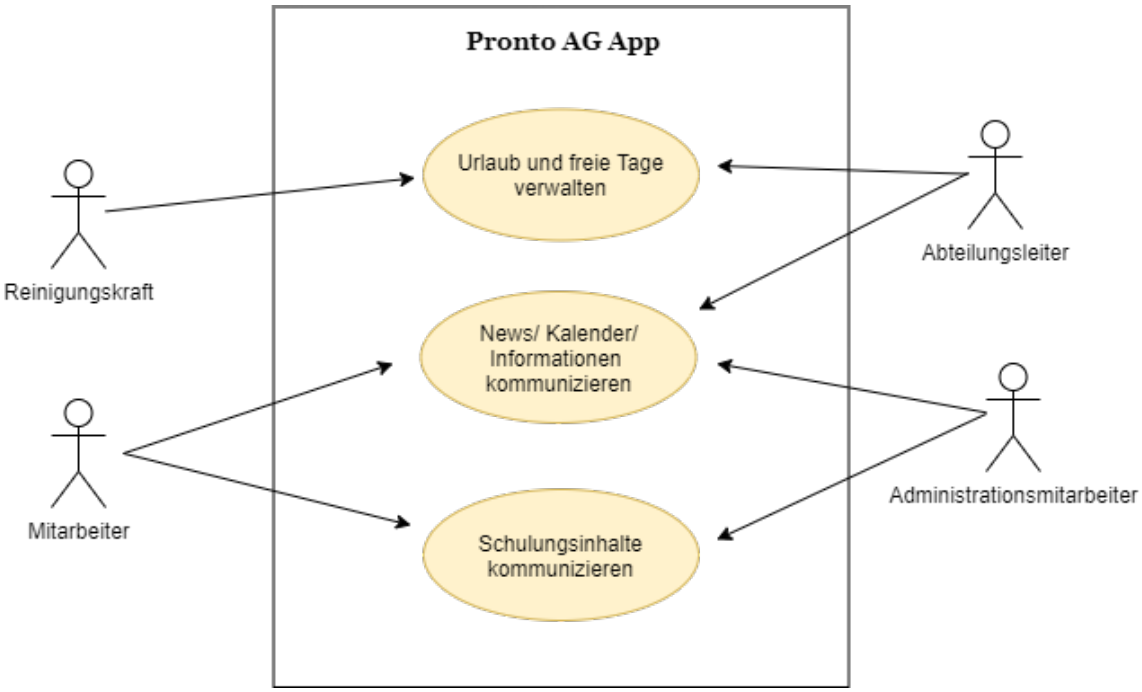

Abbildung 9.1: Übersicht Use Cases

9.2.1 Aktoren

Mitarbeiter	Der:die Mitarbeiter:in fungiert als Überbegriff für alle anderen genannten Aktoren. Zu seinen applikationsrelevanten Tätigkeiten gehören die Beantragung von Urlaub und freien Tagen sowie der Konsum von Schulungsinhalten und News.
Reinigungskraft	Die Reinigungskraft ist zuständig für die Abarbeitung der Termine gemäss Einsatzplan. Zu den applikationsrelevanten Tätigkeiten gehört der Abruf von Einsatzplänen.
Abteilungsleiter	Der:die Abteilungsleiter:in leitet ein Team von Reinigungskräften. Zu den applikationsrelevanten Tätigkeiten des:der Abteilungsleiter:in gehört die Verwaltung von Einsatzplänen, das Genehmigen von Urlaub und freien Tagen sowie das Publizieren von abteilungsspezifischen News.
Administration	Der:die Administrationsmitarbeiter:in ist für die Verwaltung von News, Schulungsinhalten und Mitarbeitern zuständig. Zu den applikationsrelevanten Tätigkeiten des Administrationsmitarbeiters gehört die Verwaltung eben dieser Daten.

Tabelle 9.1: Aktoren

9.2.2 UC01 – Urlaub und freie Tage verwalten

Mithilfe der Applikation soll es möglich sein, Urlaub sowie freie Tage zu verwalten und zu beantragen.

9.2.2.1 US01.01 – Jahresurlaub erfassen

Als Mitarbeiter:in möchte ich meinen Jahresurlaub erfassen und beantragen können, um anschliessend meine private Ferienplanung zu erstellen.

Ausbaustufe – Einfach
Ein Mitarbeiter:in kann unlimitiert Urlaub erfassen.

Ausbaustufe – Begrenzt
Die Anzahl Urlaubstage sowie die Länge zusammenhängender Urlaube können definiert werden.

9.2.2.2 US01.02 – Jahresurlaub genehmigen

Als Abteilungsleiter:in möchte ich den Urlaub meiner Mitarbeiter:innen genehmigen können, um den Urlaubsplan der Abteilung zu koordinieren.

Ausbaustufe – Einfach
Der:die Abteilungsleiter:in kann die Urlaube der Mitarbeiter:innen genehmigen.

9.2.2.3 US01.03 – freie Tage erfassen

Als Mitarbeiter:in möchte ich freie Tage erfassen und beantragen können, um anschliessend diese Tage in Anspruch zu nehmen.

Ausbaustufe – Einfach
Ein Mitarbeiter:in kann unlimitiert freie Tage erfassen.

Ausbaustufe – Begrenzt
Die Beantragung ist nur bis zu einem definierten Zeitraum vor dem Datum des freien Tages möglich.

Ausbaustufe – Zeitbasiert
Überstunden sowie verbleibender Urlaub werden in die Beantragung integriert.

9.2.2.4 US01.04 – freie Tage genehmigen

Als Abteilungsleiter:in möchte ich freie Tage für meine Mitarbeiter:innen genehmigen können, um die Einsatzplanung entsprechend anzupassen.

Ausbaustufe – Einfach
Der:die Abteilungsleiter:in kann die freien Tage der Mitarbeiter:innen genehmigen.

9.2.2.5 US01.05 – Persönliche Übersicht freier Tage einsehen

Als Mitarbeiter:in möchte ich meine beantragten und bewilligten Urlaube sowie freien Tage einsehen können.

Ausbaustufe – Liste
Die Einträge werden in einer Liste angezeigt.

Ausbaustufe – Kalender
Die Einträge werden in Kalenderform dargestellt.

Ausbaustufe – Integriert
Die Einträge werden in den Kalender des Geräts integriert.

9.2.2.6 US01.06 – Abteilungs-Übersicht freie Tage einsehen

Als Abteilungsleiter:in möchte ich die beantragten und bewilligten Urlaube sowie freien Tage meiner Abteilung einsehen können.

Ausbaustufe – Liste
Die Einträge werden in einer Liste angezeigt.

Ausbaustufe – Kalender
Die Einträge werden in Kalenderform dargestellt.

Ausbaustufe – Integriert
Die Einträge werden in den Kalender des Geräts integriert.

9.2.3 UC02 – Schulungsinhalte kommunizieren

Innerhalb der Applikation soll ein Schulungsbereich für Mitarbeiter:innen entstehen, welcher ermöglicht, sich eigenständig in gewisse Themenbereiche zu vertiefen.

9.2.3.1 UC02.01 – Schulungsvideos abrufen

Als Mitarbeiter:in möchte ich Schulungsvideos abrufen können, um mich über Neuerungen auf dem Laufenden zu halten.

Ausbaustufe – Dateien
Es stehen Videodateien zum Download über die Applikation bereit.

Ausbaustufe – Integriert
Videos können direkt innerhalb der Applikation abgerufen werden.

Ausbaustufe – Ausführlich
Zusätzlich zu den Videos werden noch Erklärungen und ergänzende Informationen angezeigt.

9.2.3.2 US02.02 – Schulungsvideos verwalten

Als Administrationsmitarbeiter:in möchte ich Schulungsvideos erstellen, bearbeiten und entfernen können, um den Mitarbeiter:innen stets aktuelles Weiterbildungsmaterial zu liefern.

Ausbaustufe – Dateien
Videos können als Datei hochgeladen werden.

Ausbaustufe – Ausführlich
Zusätzlich zu den Videos können noch Erklärungen und ergänzende Informationen definiert werden.

9.2.3.3 US02.03 – Anleitungen abrufen

Als Reinigungskraft möchte ich Anleitungen abrufen können, um mich über die auszuführenden Arbeitsschritte zu informieren.

Ausbaustufe – Dateien
Es stehen PDF-Dateien zum Download über die Applikation bereit.

Ausbaustufe – Integriert
Anleitungen können direkt innerhalb der Applikation abgerufen werden.

9.2.3.4 US02.04 – Anleitungen verwalten

Als Administrationsmitarbeiter:in möchte ich Anleitungen erstellen, bearbeiten und entfernen können, um den Mitarbeiter:innen die notwendigen Arbeitsschritte verschiedener Arbeiten einfach zugänglich zu machen.

Ausbaustufe – Dateien
Anleitungen können als PDF-Dateien hochgeladen werden.

Ausbaustufe – Integriert
Anleitungen können direkt innerhalb der Applikation mittels WYSIWYG-Editor verwaltet werden.

9.2.4 UC03 – Bereich für News / Kalender / Informationen

Im Rahmen der Applikation soll ein Newsportal entwickelt werden, über welches Neuigkeiten und wichtige Informationen geteilt und abgerufen werden können.

9.2.4.1 US03.01 – News und Informationen abrufen

Als Reinigungskraft möchte ich Neuigkeiten und Informationen der Firma abrufen können, um stets auf dem neusten Stand zu sein und entsprechend reagieren zu können.

Ausbaustufe – Integriert
News können direkt innerhalb der Applikation angeschaut werden.

Ausbaustufe – Firma
News können auf Firmenebene abgerufen werden.

Ausbaustufe – Abteilungsspezifisch
News können auf Abteilungsstufe abgerufen werden.

Ausbaustufe – Hierarchie
News können von Mitarbeiter:innen gemäss der definierten Hierarchie abgerufen und angeschaut werden.

9.2.4.2 US03.02 – News und Informationen verwalten

Als Administrationsmitarbeiter:in möchte ich Neuigkeiten erstellen, bearbeiten und entfernen können, um diese anschliessend den Mitarbeiter:innen zur Verfügung zu stellen.

Ausbaustufe – Dateien
News können als PDF-Dateien hochgeladen werden.

Ausbaustufe – Integriert
News können direkt innerhalb der Applikation mittels WYSIWYG-Editor verwaltet werden.

Ausbaustufe – Firma
News können auf Firmenebene erstellt werden.

Ausbaustufe – Abteilungsspezifisch
News können auf Abteilungsstufe erstellt werden.

Ausbaustufe – Abteilungsleiter
News auf Abteilungsstufe können von dem:der Abteilungsleiter:in erstellt werden.

Ausbaustufe – Hierarchie
News können an die in der Hierarchie definierten Funktionen zugewiesen werden.

9.2.4.3 US03.03 – News und Informationen publizieren

Als Administrationsmitarbeiter:in möchte ich Neuigkeiten publizieren können, um diese den Mitarbeiter:innen zugänglich zu machen.

Ausbaustufe – Firma
News können auf Firmenebene publiziert werden.

Ausbaustufe – Abteilungsspezifisch
News können auf Abteilungsstufe publiziert werden.

Ausbaustufe – Abteilungsleiter
News auf Abteilungsstufe können von dem:der Abteilungsleiter:in publiziert werden.

Ausbaustufe – Hierarchie
News können für die in der Hierarchie definierten Hierarchiestufen publiziert werden.

9.2.4.4 US03.04 – Kalender abrufen

Als Reinigungskraft möchte ich einen Kalender mit wichtigen Terminen abrufen können, um stets auf dem neusten Stand zu sein und entsprechend reagieren zu können.

Ausbaustufe – Integriert
Termine können direkt innerhalb der Applikation angeschaut werden.

Ausbaustufe – Firma
Termine können auf Firmenebene abgerufen werden.

Ausbaustufe – Abteilungsspezifisch
Termine können auf Abteilungsstufe abgerufen werden.

Ausbaustufe – Hierarchie
Termine können von Mitarbeiter:innen gemäss der definierten Hierarchie abgerufen und angeschaut werden.

9.2.4.5 US03.05 – Kalender verwalten

Als Administrationsmitarbeiter:in möchte ich Kalendereinträge erstellen, bearbeiten und entfernen können, um diese anschliessend den Mitarbeiter:innen zur Verfügung zu stellen.

Ausbaustufe – Dateien
Termine können als PDF-Dateien hochgeladen werden.

Ausbaustufe – Integriert
Termine können direkt innerhalb der Applikation mittels integriertem Kalender verwaltet werden.

Ausbaustufe – Firma
Termine können auf Firmenebene erstellt werden.

Ausbaustufe – Abteilungsspezifisch
Termine können auf Abteilungsstufe erstellt werden.

Ausbaustufe – Abteilungsleiter
Termine auf Abteilungsstufe können von dem:der Abteilungsleiter:in erstellt werden.

Ausbaustufe – Hierarchie
Termine können an die in der Hierarchie definierten Funktionen zugewiesen werden.

9.2.4.6 US03.06 – Kalender publizieren

Als Administrationsmitarbeiter:in möchte ich wichtige Termine publizieren können, um diese den Mitarbeiter:innen zugänglich zu machen.

Ausbaustufe – Firma
Termine können auf Firmenebene publiziert werden.

Ausbaustufe – Abteilungsspezifisch
Termine können auf Abteilungsstufe publiziert werden.

Ausbaustufe – Abteilungsleiter
Termine auf Abteilungsstufe können von dem:der Abteilungsleiter:in publiziert werden.

Ausbaustufe – Hierarchie
Termine können für die definierten Hierarchiestufen publiziert werden.

9.3 Nicht-funktionale Anforderungen

Die Anforderungen heissen „nicht funktional“, da diese nicht direkt mit der Erfüllung des Auftrages der Software zu tun haben, sondern mit deren Qualität. Die nicht funktionalen Anforderungen wurden mithilfe des FURPS-Model definiert. FURPS selbst ist ein Akronym aus dem Englischen und steht für die Softwarequalitäts-Merkmale Funktionalität, Benutzbarkeit, Zuverlässigkeit, Effizienz und Änderbarkeit. Es ist daher bei jedem NFR angegeben, in welche Kategorie des Akronyms es fällt. Die deutschen Begriffe der Kategorien wurden aus der ISO/IEC 9126-1:2001 Norm entnommen [3].

9.3.1 NFR01

- **Kategorie:** Funktionalität
- **Beschreibung:** Daten können erst nach erfolgreicher Anmeldung ausgelesen werden.

9.3.2 NFR02

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Ein Benutzer soll alle Aufgaben welche, gemäss den Use Cases aus Kapitel 9.2, dem Aktor „Mitarbeiter“ oder „Reinigungskraft“ zugeordnet sind, selbständig ausführen können. Hierbei soll keine zusätzliche Hilfestellung von aussen notwendig sein.

9.3.3 NFR03

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Informationen die farbcodiert sind, sollen durch eine zusätzliche Methode dargestellt werden, um farbenblinde Personen zu unterstützen.

9.3.4 NFR04

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Da nicht alle Benutzer über genügend gute Deutschkenntnisse verfügen, müssen Informationen auch visuell und nicht nur textuell dargestellt werden. Ausnahmefälle sind zu begründen.

9.3.5 NFR05

- **Kategorie:** Zuverlässigkeit
- **Beschreibung:** Tritt unerwartetes Verhalten auf, so wird dem Benutzer eine Fehlermeldung angezeigt. Dies wird mithilfe einer globalen Fehlerbehandlung bewerkstelligt.

9.3.6 NFR06

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Klassen sind nicht länger als 300 Zeilen.

9.3.7 NFR07

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Methoden sind nicht länger als 30 Zeilen.

9.3.8 NFR08

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Zeilen sind nicht länger als 80 Zeichen.

9.3.9 NFR09

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Der McCabe-Wert jeder Methode in der Codebase liegt unter 10.

9.3.10 NFR10

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Keine Methode in der Codebase hat mehr als 5 Argumente.

9.3.11 NFR11

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Die Testabdeckung des Codes beträgt mindestens 80%.

9.3.12 NFR12

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Alle möglichen Aufrufszenarien der API werden dokumentiert.

9.3.13 NFR13

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Es wird eine Installationsanleitung sowie eine Wartungsanleitung erstellt.

9.4 Modellüberlegungen

In diesem Kapitel wird versucht ein erstes Modell zu erstellen, welches die benötigten Funktionalitäten, welche vom Kunden gewünscht wurden, abbilden kann. Hierzu wurde eine erste Skizze des Domain-Models erstellt, welche alle benötigten Business-Objekte abbildet. Auch werden in diesem Kapitel erste Überlegungen zur Benutzeroberfläche gemacht und mit Wireframes grafisch abgebildet.

9.4.1 Domain-Model

Im Domain-Model der Problem-Domain wurden alle für die vom Kunden gewünschte Funktionalität benötigten Business-Objekte abgebildet. Es wurde ausserdem jedes Objekt noch textuell erläutert.

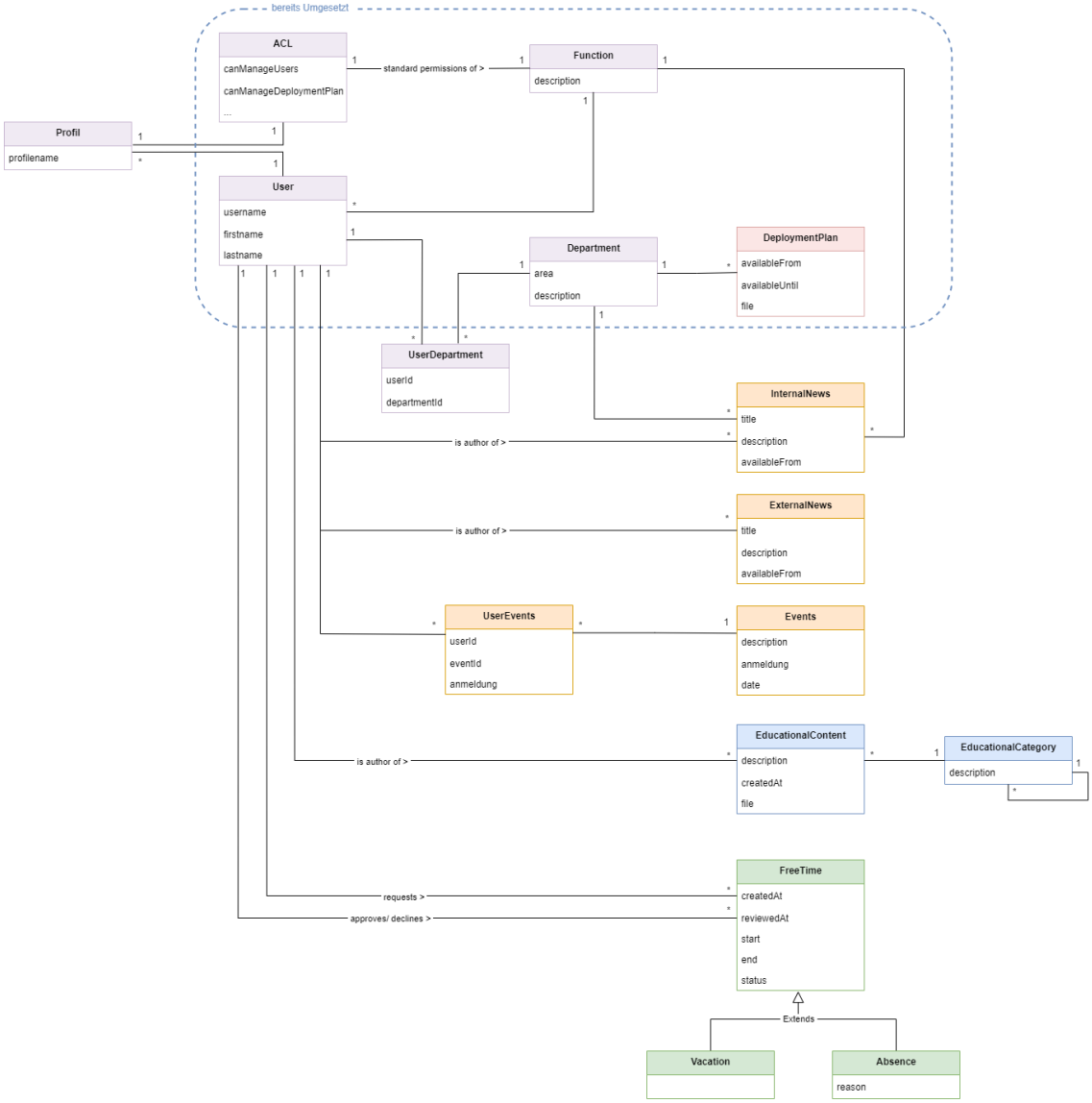

Abbildung 9.2: Übersicht über die Problem-Domain, ursprünglich übernommen aus [1]

ACL	Beschreibt eine Liste von Berechtigungen, welche einem Benutzer zugewiesen sind.
Function	Beschreibt eine Funktion, welche der Benutzer innehat und seine Berechtigungen beeinflusst z.B. Reinigungskraft.
Profil	Beschreibt eine Zwischentabelle, um Benutzern eine Rolle und nicht mehr direkt die ACL zuweisen zu müssen.
User	Beschreibt einen Benutzer, welcher die Applikation nutzt.
Department	Beschreibt eine Abteilung, welcher Benutzer, Einsatzpläne und News zugeordnet sind.
UserDepartment	Beschreibt die Zwischentabelle, welche es ermöglicht einem Mitarbeiter mehrere Abteilungen hinzuzufügen.
DeploymentPlan	Beschreibt einen Einsatzplan, welcher von Benutzern abgerufen werden kann.
InternalNews	Beschreibt einen News-Eintrag, welcher von angemeldeten Benutzern abgerufen werden kann.
ExternalNews	Beschreibt einen öffentlichen News-Eintrag, welcher auch ohne Authentifizierung abgerufen werden kann.
UserEvents	Beschreibt eine Zwischentabelle, über welche sich Benutzer zu einem Event anmelden oder abmelden können.
Events	Beschreibt einen Event, welchen die Benutzer ansehen und sich darauf anmelden oder abmelden können.
EducationalContent	Beschreibt eine Schulungsunterlage, welche einer bestimmten Kategorie angehört.
EducationalCategory	Beschreibt eine Kategorie, welcher Schulungsunterlagen zugeordnet werden können.
FreeTime	Beschreibt freie Zeit, welche ein Benutzer beantragt hat.
Vacation	Beschreibt einen Urlaub, welcher ein Benutzer beantragt hat.
Absence	Beschreibt eine Absenz, welche ein Benutzer beantragt hat.

Tabelle 9.2: Beschreibung Problem-Domain

9.4.2 Benutzeroberfläche

Die Benutzeroberfläche ist für den Kunden wahrscheinlich das wichtigste Element der Applikation. Daher muss diese gut geplant und sorgfältig aufgebaut werden. Nachfolgend sind die Punkte dokumentiert, welche während der Planung der Benutzeroberfläche relevant waren.

9.4.2.1 Design

Massgebend für das Design der Benutzeroberfläche sind die nicht-funktionalen Anforderungen in der Kategorie Usability, sowie das Corporate Design der Pronto AG gemäss Appendix Kapitel **??**.

Sollten zur Hervorhebung oder Kategorisierung (z.B. Info- / Warn- und Fehlermeldungen) weitere Farben benötigt werden, wird auf die Farbpalette des verwendeten UI-Frameworks zurückgegriffen.

9.4.2.2 Wireframes

Zur verbesserten Veranschaulichung der funktionalen Anforderungen für den Kunden, wurden Wireframes erstellt. Des Weiteren fungieren die Wireframes als Implementationsvorlage für die Benutzeroberfläche.

Die wichtigsten Funktionalitäten für Reinigungskräfte und allgemeine Mitarbeiter:innen sind im Format eines Smartphone Apps abgebildet. Für die administrativen Funktionalitäten, welche primär von Abteilungsleiter:innen und Administrationsmitarbeiter:innen verwendet werden, wurden Wireframes im Desktop-Format erstellt.

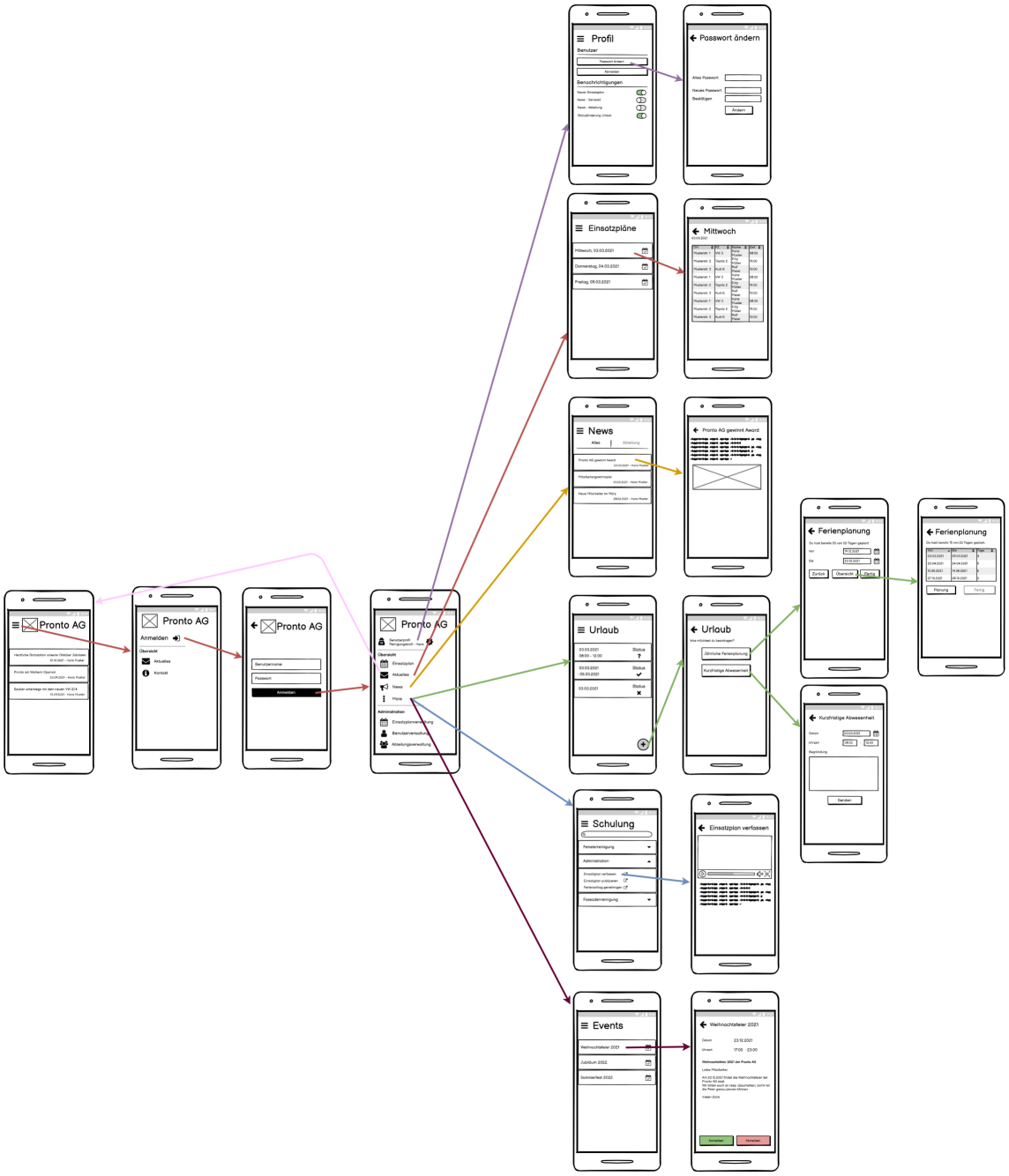

Abbildung 9.3: Übersicht Wireframes App

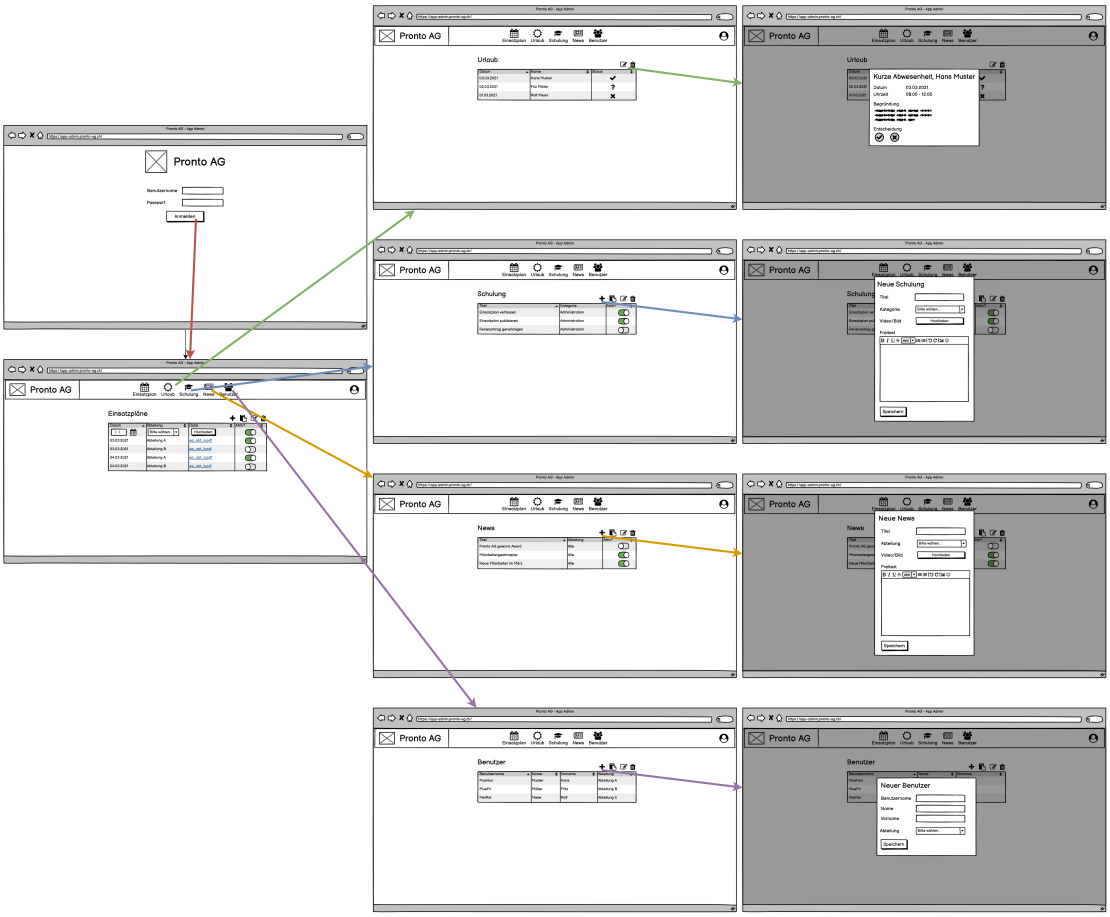

Abbildung 9.4: Übersicht Wireframes Administrations-Webseite, übernommen aus [1]

Eine Übersicht aller Wireframes ist unter Kapitel G.1 zu finden.

10 Usability-Test

In diesem Kapitel geht es darum, die Usability-Tests der Applikation zu dokumentieren und Schlüsse daraus zu ziehen. Es ist ungewohnt, einen Usability-Test so früh im Projekt durchzuführen. Da unsere Vorgänger nicht mehr dazugekommen sind, wurde entschieden diesen so früh im Projekt einzuplanen. Damit ist gewährleistet, dass genügend Zeit vorhanden ist um allfällige Anpassungen planen und umsetzen zu können. Der Usability-Test prüft die Wahrnehmung, Oberfläche, Nutzbarkeit und Funktionalität aus Sicht der Benutzer.

Um den Usability-Test zu planen, haben wir uns an den gelernten Punkten aus dem Modul Software Engineering 2 sowie an Internet-Recherchen orientiert [4].

10.1 Ziele

Unser Ziel ist es, mit dem Usability-Test herauszufinden wie gut die Benutzer mit der App klar kommen und ob für sie direkt ersichtlich ist, was sich wo befindet. Dabei stellen wir den Testpersonen kleine Fragen und Aufgaben. Somit können wir beobachten, wo noch mögliche Schwierigkeiten vorhanden sind und auch was sich bewährt. Die genauen Aufgaben sind unter 10.3 ersichtlich.

Für den Usability-Test setzen wir unsere Hauptziele darauf, dass die Aufgaben *spezifisch* sowie *messbar* sind und nach *Priorität* geordnet werden können. Weiter möchten wir den Usability-Test auf dem Betriebssystem, welches der Benutzer besitzt (iOS oder Android), durchführen. Somit kann sich der Benutzer besser auf die App konzentrieren und eine mögliche Umgewöhnung von der Darstellung zwischen den beiden Betriebssystemen hat keinen Einfluss.

10.2 Teilnehmer

Den Usability-Test führen wir direkt mit Mitarbeiter:innen der Pronto AG durch. Dabei ist es uns wichtig, dass wir nicht nur mit Mitarbeiter:innen, welche bei der Entwicklung schon dabei waren den Test durchführen, sondern mit mit Personen welche die App noch gar nicht kennen. Damit wir die verschiedenen Rollen und deren vorhandenen Funktionen innerhalb der App testen, möchten wir den Test mit *Reinigungskräften*, *Abteilungsleiter:innen* sowie Mitarbeiter:innen aus der *Administration* durchführen.

Die genauen Beschreibungen und deren Funktionen sind unter 9.2.1 ersichtlich.

Ein grosser Vorteil eines Usability-Test vor Ort ist es, dass viel mehr Rückschlüsse aus dem Test gezogen werden können. Dazu gehört nicht nur die direkte Rückmeldung des Testers, sondern auch seine Mimik, Vorgehensweise sowie mögliche Stolpersteine. Zudem kann der Test dynamisch angepasst und erweitert oder der Testbenutzer bei Problemen unterstützt werden.

Zum Schluss können wir mit einem persönlichen Gespräch auch noch zusätzliche wünschenswerte Features aufnehmen und auf ihre Umsetzbarkeit prüfen.

10.3 Planung

Nummer	Beschreibung	Testperson	Erwartetes Ergebnis
UT01	App wird auf iOS und Android Geräten korrekt dargestellt und ist nutzbar	Pronto AG - Mitarbeiter	Cross-Plattform Darstellung ist korrekt und Funktionen sind alle nutzbar
UT02	Die Mobile-First optimierte App ist auch im Browser nutzbar	dk	Darstellung sauber
UT03	Farbenblinde können die App nutzen	Farbenblinde:r	Alle Funktionen sind vollständig durch sehbehinderte Person nutzbar
UT04	Sehbeeinträchtigte können die App ohne Probleme nutzen	Sehbeinträchtigte:r	Alle Funktionen und Informationen können genutzt werden. Verschiedene Hilfsmittel können in der App eingesetzt werden, wie zum Beispiel Zoom
UT05	Die Funktionsweise der Oberfläche ist den Benutzern ohne Einführung klar	Pronto AG - Mitarbeiter	Die Benutzer finden sich zurecht
UT06	Die Funktionsweise der Oberfläche ist zur Nutzung der App ausreichend	Pronto AG - Mitarbeiter	Die Benutzer sind zufrieden mit den bestehenden Funktionen und würden die App im Berufsalltag benutzen
UT07	Die App ist einfach gehalten und man kommt mit wenigen Schritten zum Ziel	Pronto AG - Mitarbeiter	Die Benutzer müssen keine überflüssigen Eingaben tätigen
UT08	Die App vereinfacht den Arbeitsalltag	Pronto AG - Mitarbeiter	Die Benutzer haben durch die Nutzung der App einen Mehrwert
UT09	Fehler in der App werden korrekt dargestellt	Pronto AG - Mitarbeiter	Die Benutzer erhalten bei einem Fehler aussagekräftige Meldungen
UT10	Zeitplanung ist gut umgesetzt	Pronto AG - Mitarbeiter	Die Benutzer empfinden die Umsetzung des Zeitplan-Features als angemessen und alle wichtigen Funktionen sind vorhanden

Tabelle 10.1: Planung Usability-Test

10.4 Ablauf

Der Usability-Test wird mit einem Moderator durchgeführt, damit eine bessere Interaktion, direktes Feedback eingeholt sowie die Gedanken der Testbenutzer während den verschiedenen Aufgaben nachvollzogen werden können. Dabei ist es wichtig, dass der Moderator dem Testbenutzer hilft, aber nicht die Führung übernimmt.

Um die Gedankengänge des Testbenutzers möglichst gut zu kennen wir dieser vom Moderator animiert:

- laut zu denken
- sagen, wenn etwas gesucht aber nicht gefunden wird
- eine Rückmeldung zu geben, ob das Resultat der Erwartung entspricht

Der genaue Ablauf des Usability-Test sowie die jeweiligen Funktionen der Mitarbeiter:innen sind nachfolgend ersichtlich.

Funktion	**Beschreibung**
Reinigungskraft	1. Login 2. Einsatzpläne anschauen 3. eigenes Profil anschauen (allenfalls Passwort zurücksetzen)
Abteilungsleiter	1. Login 2. Einsatzpläne für bestimmte Gruppen in App aufschalten 3. Einsatzpläne anschauen 4. eigenes Profil anschauen (allenfalls Passwort zurücksetzen)

Tabelle 10.2: Ablauf Usability-Test

10.5 Resultate

In diesem Abschnitt werden die Resultate des Usability-Tests ausgewertet. Der Usability-Test konnte mit verschiedenen Testbenutzern der Pronto AG durchgeführt werden. Dazu gehörten zwei Reinigungskräfte sowie zwei Mitarbeiter:innen aus der Administration, welche auch die Funktion der Abteilungsleiter:in durchspielten. Wir konnten dabei viele *qualitative Daten* sammeln, um Einblicke in die von den Teilnehmern eingeschlagenen Wege und die aufgetretenen Probleme zu erhalten. In einem kurzen Abschlussgespräch konnten wir noch über zusätzliche Ideen für Funktionen oder sonstige Verbesserungen mit den Mitarbeiter:innen der Pronto AG brainstormen.

10.5.1 Auswertung

Die Usability-Tests wurden im Kapitel 10.3 bereits genau beschrieben und aufgelistet. Aus diesem Grund werden im Folgenden nur die Bewertung sowie die erstellten Notizen zusammengefasst aufgelistet. Die kompletten Ergebnisse des Usability-Test sind im Anhang H ersichtlich.

Nummer	Bewertung (1-5)	Bemerkungen
UT05	4	• Menu-Icon oben links statt unten links • Einstellungs-Icon für Profilübersicht (statt auf Benutzerprofil drücken) • Ausloggen schwierig (bis Einstellung gefunden wurde)
UT06	3	• Neues Passwort bestätigen (2x eingeben) • Passwortwechsel «logisch» über Benutzerprofil
UT07	4	• Passwort ändern schwierig (bis gefunden)
UT08	3	• Aktuell über WhatsApp-Gruppe (App bringt mehrere Verbesserungen) • Einsatzplan bringt aktuell nicht viel Vorteil • Ferienanträge wären wichtig
UT09	5	• Bei falschem Passwort (beim Login) und bei falschen Angaben bei der Benutzererstellung werden gute Fehlermeldung dargestellt
UT10	5	• PDF nach Seite scrollen • PDF-Seitenanzahl anzeigen

Tabelle 10.3: Auswertung Usability-Test

10.5.1.1 Weitere mögliche Funktionen

- Teamevents integrieren (an- & abmelden, Informationen, etc.)
- Geburtstagsliste der Mitarbeiter:innen (direkt in Kalender der App oder sonst als Liste darstellen)

10.5.1.2 Verbesserungen Administration

- Einsatzplan löschen verbessern
  - Mit Rechts-Swipe (auf Mobile)
  - Einsatzplan markieren + Delete Button
  - Zusätzliche Nachfrage (ob man wirklich löschen möchte) ist sehr gut
- Einsatzplan hochladen + veröffentlichen ist sehr gut
  - Allenfalls Datumfeld besser ersichtlich machen (dass User nicht versucht Text einzugeben)
    - direkt mit Default-Werten abfüllen
  - Ablauf + Veröffentlichkeitsdatum vorab abfüllen (Default Wert)
    - Ab Erstellung 1 Monat gültig
    - Zeit nicht nötig
- Bei Benutzerverwaltung die Berechtigungen besser angeben
  - Einzelne Schalter mit Rechten verbergen -> nur Rollen den Benutzern zuweisen und über Rollen die effektiven Berechtigungen definieren
- Abteilungen erstellen sehr gut und einfach
- Benutzerverwaltung mit Abteilungen sehr gut
- Push-Benachrichtigungen, wenn Einsatzplan aktualisiert wurde
  - Damit Mitarbeiter:innen benachrichtigt werden, wenn es eine Anpassung gab (Anpassungen kommen sehr häufig vor)
  - Allenfalls auch Titel des Einsatzplanes fett markieren, wenn Plan noch nicht angesehen wurde (seit letzter Aktualisierung)
- Push-Benachrichtigungen im App aktivieren/ deaktivieren
  - Bei Ferien, Unfall, etc.
  - Falls nicht möglich über Handy-Einstellungen
- Benutzern zu mehreren Abteilungen hinzufügen
  - Bei normalen Mitarbeiter:innen und Abteilungsleiter:innen nötig
- Beim Passwortwechsel und Login soll Passwort im Klartext angesehen werden können
- Suchfunktion oder Filter bei Benutzerverwaltung

10.5.2 Priorisierung für Construction-Phase

Die Priorisierung wurde zusammen mit der Pronto AG aufgestellt und wird im folgenden aufgelistet. Für die Planung der Arbeitspakete während der Construction-Phase orientieren wir uns stark an dieser Priorisierung.

10.5.2.1 Priorisierung

1. News als neue Landing-Page implementieren (für Deployment in Public App Store)
    - Externe News (allenfalls direkt von Pronto Website integrieren)
2. Rückmeldungen Usability-Test umsetzen
    - Passwort bestätigen (auch bei Passwortwechsel)
    - Ausloggen "versteckt"
    - Icons oben links statt unten links
    - Hinzufügen-Button unten links (Floating Action Button)
    - Bei Benutzerprofil Einstellungsbutton statt auf Namen klicken
    - Veröffentlichkeitsdatum von Einsatzplänen vorab mit Default-Wert abfüllen
    - Einsatzplan löschen verbessern
    - Benutzerprofile mit spezifischen Rechten erstellen und diese zuweisen
        - Aber nicht bei jedem zu erstellenden User alle Möglichkeiten anzeigen
    - Abteilungsleiter:in kann noch "nichts"verwalten
    - Push-Benachrichtigungen bei nachträglichem Update von Einsatzplan
    - Mitarbeiter:in mehreren Abteilugen hinzufügen
        - Aktuell gemäss Architektur nicht vorgesehen -> in Architekturreview anschauen
    - Passwort in Klartext anzeigen unterstützen (Browser + App)
    - Benutzerverwaltung mit Suchfunktion oder Filter (Alphabetisch)
    - Seitenanzahl bei PDF (Einsatzplan) anzeigen
3. Bereich für News/ Schulungsvideos

11 Architektur Review

In diesem Kapitel geht es darum, die Architektur der Applikation aus [1] einem Review durchzuziehen und anhand der neu entstandenen Anforderungen beziehungsweise des Usability-Tests die Architektur gegebenfalls neu zu planen.

11.1 Ergebnisse

Was	Beschreibung	Wie
Systeme und Technologien	Flutter, .NET, Firebase, usw. sollen nicht hinterfragt werden, sondern weiter damit gearbeitet werden	-
Domain-Model	Abteilungen und Benutzer aktuell eine 1:* Beziehungen, Pronto AG will das Mitarbeiter:innen in mehreren Abteilungen sein können	Zwischentabelle wird benötigt für *:*
Domain-Model	Es sollen nicht nur interne News in der App ersichtlich sein, sondern auch externe	Tabelle umbennen und neue für externe News hinzufügen
App Design	Beim Start der Applikation sollen die externen News als Einstieg angezeigt werden, somit ist die App im AppStore ohne Probleme zu veröffentlichen	Design umstellen - externe News als Startseite und Login soll als Button verfügbar sein. Erst nach Login die anderen Funktionen zur Verfügung

Tabelle 11.1: Ergebnisse Architektur Review

Die Architektur wird wie oben beschrieben angepasst. Bis auf diese Änderungen sind zu diesem Zeitpunkt keine weiteren geplant. Die Architektur sollte damit für die Weiterführung des Projektes ausreichen. Das Domain-Modell 9.4.1 und die Wireframes 9.4.2.2 sind entsprechend angepasst worden.

Teil III

Umsetzung

12 Releases

In diesem Abschnitt werden die erstellten Releases mit den umgesetzten Funktionalitäten beschrieben. Die NFRs welche in diesem Projekt definiert sind, wurden über die gesamte Laufzeit des Projektes beachtet und sind daher nicht in den Releases aufgeführt. Für die Versionierung haben wir uns an den Semantic Versioning 2.0 Standard gehalten und ist auf [1] aufbauend.
Die Releases wurden so aufgebaut, dass nach jedem Release ein fertiges Produkt verfügbar ist.

12.1 v0.4.0

- Upgrade Packages und Dependencies auf aktuellste Version

12.2 v0.4.1

- Optimierungen gemäss Usability-Test
- Upload von Bildern
- Passwort in Klartext anzeigen lassen

12.3 v0.5.0

- Optimierungen gemäss Usability-Test
- Anfrageformular
- User mehreren Abteilungen hinzufügen
- US03.01 - News und Informationen abrufen
- US03.02 - News und Informationen verwalten
- US03.03 - News und Informationen publizieren

12.4 v0.6.0

- US02.01 - Schulungsvideos abrufen
- US02.02 - Schulungsvideos verwalten
- US02.03 - Anleitungen abrufen
- US02.04 - Anleitungen verwalten
- US03.04 - Kalender abrufen
- US03.05 - Kalender verwalten
- US03.06 - Kalender publizieren

12.5 v1.0.0

- Bug fixes
- Produktiver Release auf allen Plattformen

Alle Releases sind auf GitHub ersichtlich.

13 Herausforderungen

In diesem Kapitel wird beschrieben, wie die geplante Softwarearchitektur umgesetzt und erweitert wurde. Es soll ein Einblick in die Strukturen und Abläufe der Softwarekomponenten gewährt werden. Während der Umsetzungsphase können oft unerwartete Situationen eintreten, welche in zusätzlichen Herausforderungen enden. Aus diesem Grund werden in diesem Kapitel die Key Points der Umsetzungphase sowie spezielle Implementationsdetails genauer erläutert.

13.1 Allgemein

In diesem Abschnitt gehen wir auf die Herausforderungen ein, welche nicht direkt mit dem Frontend oder dem Backend zusammenhängen, aber dennoch wichtig sind.

13.1.1 Einarbeitung

Ein wichtiger Punkt, welcher wir nochmals erwähnen möchten, ist die Einarbeitung in das Projekt. Da wir nicht auf der grünen Wiese begonnen haben, sondern das Projekt von einer vorgängigen Bachelorarbeit weitergeführt haben, ist diese entscheidend. Wir kannten die eingesetzten Technologien, Flutter respektive Dart fürs Frontend und ASP.NET fürs Backend, nicht oder fast nicht. Dies hatte für uns eine steile Lernkurve zur Folge. Aufgrund der guten Qualität der Software und der Dokumentation aus dem Vorgängerprojekt [1] konnten wir uns schnell ins Projekt einarbeiten sowie mit dem Code auseinandersetzen. Wie so oft kann man sich mit *Learning by Doing* sehr schnell in neue Technologien einarbeiten. Somit konnten wir auch schon nach kurzer Zeit mit dem Implementieren von neuen Features beginnen.

13.1.2 Apple Deployment

Da ein Grossteil der Mitarbeitenden der Pronto AG ein iPhone besitzen, ist es wichtig, dass die App aus dem AppStore installiert werden kann.

Zu Beginn wurde versucht mit der Integrierung von zusätzlichen Funktionen, welche auch für externe Benutzer vorhanden waren, die App im öffentlichen AppStore zur Verfügung zu stellen. Dazu wurde die Landing-Page der App umgestellt. Neu zeigt diese die externen News an, welche öffentlich für alle Benutzer zugänglich sind. Über eine Side-Navigation hat man nun die Möglichkeit die Kontaktdaten der Pronto AG anzusehen und kann auch direkt ein Anfrageformular ausfüllen und über die App absenden. Über das `Login` haben Mitarbeiter:innen nun die Möglichkeit sich in der App anzumelden und Zugriff auf interne Informationen zu erhalten.

Da es sich bei der *Pronto-MIA* App um eine Business Applikation handelt, kann die App aufgrund von Limitationen seitens Apple nicht in den öffentlichen AppStore deployed werden. Aus diesem Grund muss die App über den Business Manager verteilt werden. Dabei wird die App grundsätzlich auch im AppStore aufgeschaltet, ist aber versteckt für die Öffentlichkeit. Im Business Manager können nun *Redemption Tokens* erstellt werden, über welche die Mitarbeiter:innen der Pronto AG die App aus dem AppStore installieren können.

13.1.3 Funktionsumfang

Um zu zeigen, was wir alles im Rahmen unserer Studienarbeit umsetzen konnten, möchten wir nochmals auf das UML-Diagramm der Problem Domain aus Kapitel 9.4.1 eingehen.

Wie im aktualisierten UML ersichtlich ist, konnten wir einige zusätzliche Features umsetzen. Es ist aber auch ersichtlich, dass sich das UML seit der Anforderungsanalyse nochmals verändert hat, da wir es den neuen Anforderungen jeweils wieder anpassen mussten.

Im Folgenden gehen wir nun auf die neu implementierten Funktionalitäten noch etwas genauer ein.

13.1.3.1 DepartmentUsers

Damit ein Benutzer mehreren Abteilungen zugehören kann, wurde eine Zwischentabelle `DepartmentUsers` erstellt. Durch diese Zwischentabelle kann dem User eine Liste von `Departments` zugewiesen werden und nicht mehr nur ein einzelnes.

13.1.3.2 InternalNews

Die internen News können dazu benutzt werden, interne Informationen der Pronto AG den Mitarbeiter:innen zur Verfügung zu stellen. Diese Informationen sind nur für authentifizierte und autorisierte Benutzer ersichtlich. Ein `InternalNews`-Objekt beinhaltet die Attribute `Title`, `Description`, `AvailableFrom`, `File` und `Published`.

13.1.3.3 ExternalNews

Die externen News können dauz benutzt werden, Informationen der Pronto AG der Öffentlichkeit zur Verfügung zu stellen. Diese Informationen sind für alle Benutzer ohne

Authentifizierung und Autorisierung ersichtlich. Ein `ExternalNews`-Objekt beinhaltet die Attribute `Title`, `Description`, `AvailableFrom`, `File` sowie `Published`.

13.1.3.4 Appointment

Beim Appointment handelt es sich um ein Ereignis, welches im Kalender dargestellt wird. Die Einträge im Kalender sind nur für authentifizierte und autorisierte Benutzer ersichtlich. Ein `Appointment`-Objekt beinhaltet die Attribute `Title`, `Location`, `From`, `To`, `isAllDay` und `isYearly`.

13.1.3.5 EducationalContent

Bei der Tabelle `EducationalContent` handelt es sich um Schulungsvideos mit zusätzlichem Inhalt oder PDF-Dateien. Diese Informationen sind nur für angemeldete Benutzer ersichtlich. Ein `EducationalContent`-Objekt beinhaltet die Attribute `Title`, `Description`, `File` sowie `Published`.

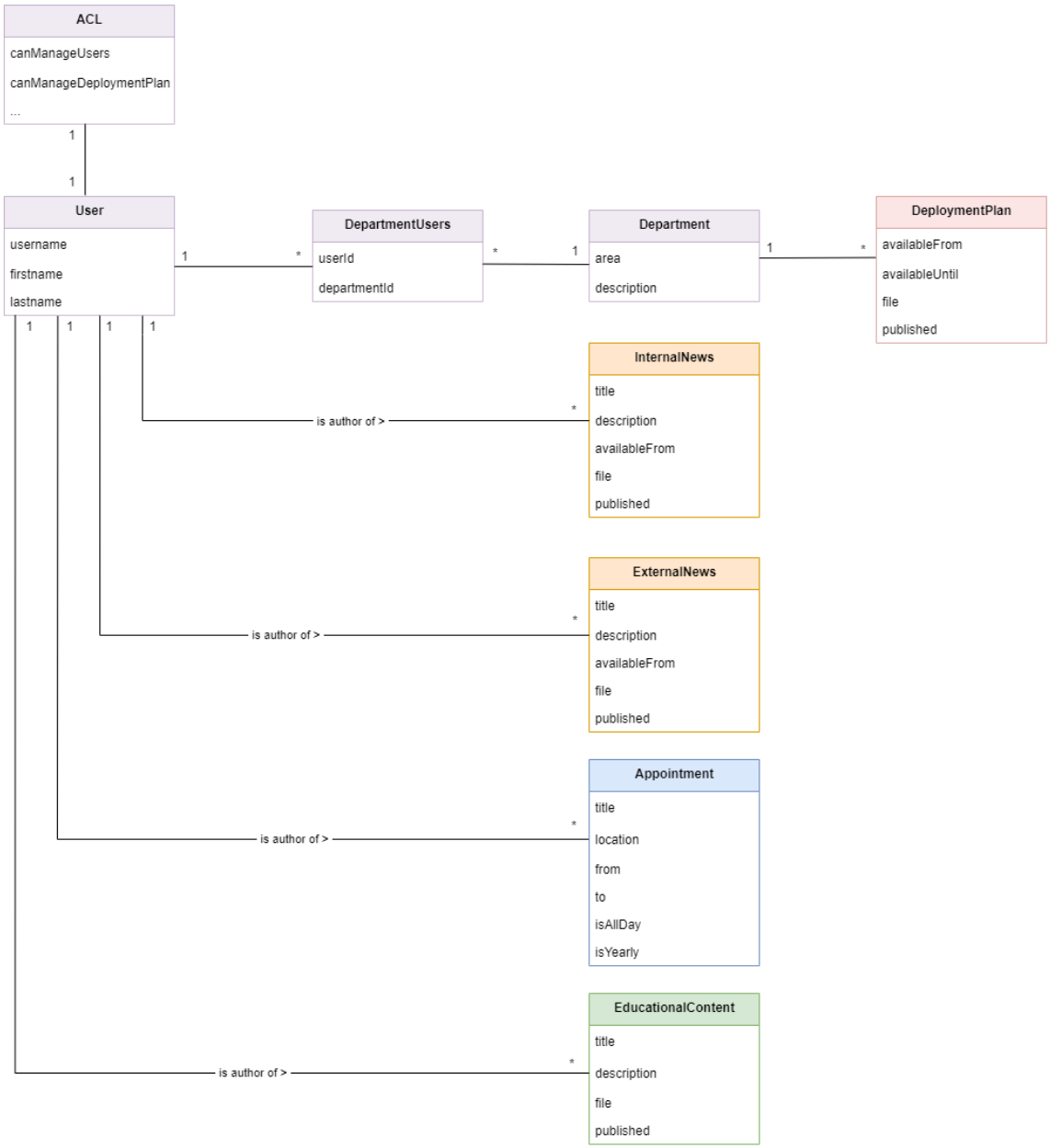

Abbildung 13.1: Übersicht über implementierte Problem Domain

13.2 Frontend

In diesem Kapitel werden die Herausforderungen beschrieben, welche hauptsächlich im Frontend stattgefunden haben oder zumindest die grösste Herausforderung bei der Umsetzung stattfand.

13.2.1 Inquiry / Anfrageformular

Mit dem Ziel, die Pronto-MIA App im normalen AppStore zu veröffentlichen, wurde ein Anfrageformular integriert. Dieses ist für alle externen Benutzer:innen ersichtlich und damit kann direkt über die App eine Anfrage übermittelt werden. Die grösste Herausforderung bei diesem Formular war die Übermittlung der Informationen an die Pronto AG.

Da über die Website bereits ein Kontaktformular existiert, welches die Informationen via E-Mail an ein Postfach übermittelt, wollten wir dies in der Applikation auch so umsetzen.

13.2.1.1 Post-Request

Das Formular in der Applikation entspricht mehr oder weniger dem Anfrageformular auf der Website der Pronto AG. Damit Anfragen aus beiden Applikationen im gleichen Mail-Postfach eintreffen und gleich formatiert sind, wollten wir das Formular der Website überprüfen. Das Ziel war es, den gleichen `POST`-Request mit den entsprechenden Informationen über die App abzusenden. Damit könnte die Implementierung einfach und sauber gelöst werden und die Pronto AG erhält Anfragen aus beiden Quellen im gleichen Format.

Leider sendet das aktuelle Kontaktformular Anfragen an eine `Typo3`-Schnittstelle mit `powermail`.

```
1  ------WebKitFormBoundarylp2iWlgmT6W80e0C
2  Content-Disposition: form-data;
       name="tx_powermail_pi1[field][vorname]"
3  Gian
4  ------WebKitFormBoundarylp2iWlgmT6W80e0C
5  Content-Disposition: form-data;
       name="tx_powermail_pi1[field][nachname]"
6  Fluetsch
7  ------WebKitFormBoundarylp2iWlgmT6W80e0C
8  Content-Disposition: form-data;
       name="tx_powermail_pi1[field][strasse]"
9  Oberseestrasse 101
10 ------WebKitFormBoundarylp2iWlgmT6W80e0C
11 Content-Disposition: form-data;
       name="tx_powermail_pi1[field][postleitzahl]"
12 8645
13 ------WebKitFormBoundarylp2iWlgmT6W80e0C
14 Content-Disposition: form-data;
       name="tx_powermail_pi1[field][ort]"
15 Rapperswil-Jona
```

Auflistung 13.1: Ausschnitt Body des Post-Requests via Website

Wie im Request ersichtlich ist, werden verschiedene IDs generiert und der Request speziell formatiert. Da leider keine Library gefunden werden konnte, welche solche Anfragen generiert und das Ganze manuell zu implementieren sehr aufwändig geworden wäre, wurde entschieden, einen anderen Ansatz zu wählen.

13.2.1.2 SMTP Client Frontend

Wir haben nun versucht statt im Frontend einen `POST`-Request abzusetzen, einen `SMTP`-Client zu instanziieren. Der `SMTP`-Client authentifiziert sich beim Mailserver der Pronto AG und kann so aus der Applikation selbst ein Mail an das gewünschte Postfach senden. Der Vorteil bei dieser Variante ist, dass keine zusätzliche Schnittstelle mehr benötigt wird und alles innerhalb der Applikation eigenständig funktioniert. Mit dem `smtp`-Package gibt es eine Flutter-Library, welche dieses Vorhaben unterstützen sollte. Das Problem bei der Frontend-seitigen Implementation des `SMTP`-Client war aber, dass die Library keine

`Socket`-Objekte im Web erzeugen kann, da diese von `dart.io` stammen. Da die Webapplikation in JavaScript kompiliert wird, kann auf diese Standard Dart-Library nicht mehr zugegriffen werden. Die weiteren gefunden Libraries konnten eigenständig keinen `SMTP`-Client instanziieren und waren auf weitere Drittanbieter-Komponenten angewiesen.

13.2.1.3 SMTP Client Backend

Schlussendlich wurde die gesamte Logik für das Senden des Mails ins Backend ausgelagert. C# bietet mit dem `MailKit`-NuGet Package eine Library an, welche alle Anforderungen erfüllt. Das Frontend validiert nun das Formular und generiert daraus den gewünschten HTML-String. Dieser sendet das Frontend nun über die `GraphQL`-API ans Backend, welches schlussendlich das Mail ans gewünschte Postfach sendet.

```
1         MimeMessage message = new ();
2             message.From.Add(
3                 MailboxAddress.Parse(this.GetSmtpSender));
4             message.To.Add(
5                 MailboxAddress.Parse(this.GetSmtpRecipient));
6             message.Subject = subject;
7             message.Body =
8                 new TextPart("html") { Text = content };
```

Auflistung 13.2: Generiung Message

```
1         client.Connect(
2                 this.GetSmtpServer,
3                 this.GetSmtpPort,
4                 SecureSocketOptions.SslOnConnect);
5
6         client.Authenticate(
7                 this.GetSmtpUsername, this.GetSmtpPassword);
```

Auflistung 13.3: Authentifizierung und Instanziierung SMTP-Client

Abschliessend kann gesagt werden, dass die nun implementierte Lösung mit der Logik im Backend sicher auch die Richtige ist.

13.2.2 Kalender

Aus dem Usability-Test enstanden zwei Anforderungen, welche wir mit einem Kalender abdecken wollten. Eine Anforderung von der Pronto AG war, dass sie Veranstaltungen übersichtlich darstellen können und das sich Mitarbeiter:innen über das App an diesen Veranstaltungen an- bzw. abmelden können. Die zweite Anforderung war, dass in der App eine Geburtstagsliste ersichtlich sein soll, bei welcher alle Geburtstage der Mitarbeiter:innen eingetragen sind. Da diese beiden Anforderungen keine grosse Priorität hatten, wir das Feedback aber trotzdem einfliessen lassen wollten, haben wir uns dazu entschieden, eine möglichst einfache Lösung mit diesem Kalender zu entwickeln. Damit können wir beide Anforderungen zu einem grossen Teil abdecken. Wir haben uns bei der Implementation des Kalenders für ein Packet namens `sfCalendar` entschieden. Bei dieser gab es eine grosse Herausforderung. Beim Package konnte man die Datenquelle angeben, damit die Einträge korrekt ersichtlich sind. Allerdings müssen vererbte Funktionen überschrieben werden, damit alle vom Package benötigten Informationen auch korrekt sind. Mit dem Überschreiben der `StartZeit` und `EndZeit` Methode ist uns ein Fehler aufgefallen. Jegliche wiederkehrende Einträge, wie in unserem Beispiel Geburtstage, wurden erst im darauffolgenden Jahr angezeigt. Sobald ein wiederkehrender Termin erstellt wird, welcher ab heute ersichtlich sein sollte, wurde dieser erst im nächsten Jahr angezeigt. Wir konnten uns das Problem nicht erklären, da aus unserer Sicht alles soweit gepasst hat. Wir haben verschiedene Optionen ausprobiert und schlussendlich sind wir auf eine Lösung gestossen, welche nicht ganz zufriedenstellend ist. Wir haben das `DateTime`-Objekt, welches in der Variable `appointment` gespeichert war, nur noch mit den wichtigsten Angaben neu erstellt. Irgendwie hat danach die Funktion, wie von Anfang an zu erwarten gewesen wäre, wieder funktioniert.

Unsere Beurteilung war, dass beim Package `SfCalendar` der Bug sein muss, da es auch mit der Lokaliserungsangabe teilweise zu Problemen kam. Da wir diese Informationen mittels Neuerstellung des `DateTime`-Objekts sozusagen verloren haben, kann das Package seine default Einstellungen verwenden. Allerdings konnten wir diese Vermutung nicht abschliessend bestätigen lassen. Es wurde deswegen der Code an der entsprechenden Stelle kommentiert, damit dieser Workaround nicht entfernt wird solange das Package die zu erwartende Funktionalität nicht korrekt umsetzt.

```
1     @override
2     DateTime getStartTime(int index) {
3         final local.Appointment appointment =
4             appointments[index] as local.Appointment;
5
6         return appointment.from;
7     }
```

Auflistung 13.4: Überschreiben der Methode getStartTime ohne Workaround

```
    @override
    DateTime getStartTime(int index) {
        final local.Appointment appointment =
            appointments[index] as local.Appointment;

        // Bug in syncfusion, do not return DateTime of
           appointment.from object
        return DateTime(
            appointment.from.year,
            appointment.from.month,
            appointment.from.day,
            appointment.from.hour,
            appointment.from.minute,
        );
    }
```

Auflistung 13.5: Überschreiben der Methode getStartTime mit Workaround

13.3 Backend

In diesem Kapitel werden die Herausforderungen beschrieben, welche hauptsächlich im Backend stattgefunden haben oder zumindest die grösste Herausforderung bei der Umsetzung darstellten.

13.3.1 User in mehreren Departments

Bei dieser Herausforderung haben wir den Aufwand massiv unterschätzt. Wie im folgenden Codeausschnitt ersichtlich ist, enstand aus der eigentlich sehr kleinen Änderung auf der `User`-Klasse ein massiver Änderungsbedarf. Nun kann ein Benutzer mehreren Abteilungen angehören. Durch diese Änderung mussten wir so ziemlich alle bestehenden `Services`, `ViewModels` und `Views` anpassen, da alles vom Department abhängig war. Dies reichte von der `Autorisation` bis hin zu den Darstellungen der Verwaltungen. Ausserdem wurden durch diese Änderungen die alten Versionen nicht mehr unterstützt. Nun besitzt ein Benutzer eine Liste von Departments und nicht mehr einen einzelnen Department-Eintrag. Der Aufwand dieser Änderung war definitv sehr hoch und das Ergebnis sieht man eigentlich kaum. Diese Anforderung kam allerdings aus dem Usability-Test und hatte eine hohe Priorität für die Pronto AG. Es gibt immer wieder Fälle, bei welchen Mitarbeitende zu mehreren Abteilungen gehören und somit auch die Informationen aus beiden Abteilungen benötigen.

```
    public virtual Department Department { get; set; }
```

Auflistung 13.6: User Klasse mit Abhängigkeit zu einem Department

```
    public virtual ICollection<Department> Departments { get;
       set; }
```

Auflistung 13.7: User Klasse mit Abhängigkeit zu mehreren Departments

13.3.2 Autorisation Abteilungsleiter

Bei der Funktion Abteilungsleiter enstand ein Problem, als wir alle Abhängigkeiten im Backend auf den aktuellsten Stand gebracht haben. Und zwar funktionierte die Autorisierung nicht mehr wie zuvor. Ein:e Abteilungsleiter:in hatte keine Rechte mehr, die Inhalte seiner Abteilung zu bearbeiten. Dies war zurückzuschliessen auf die Methode `GetAccessObjectId`, welche unten aufgelistet ist. Die Methode gab nicht mehr die `Id` zurück, welche ursprünglich angefragt wurde. Im Beispiel Einsatzplan kam nicht mehr die `Id` des Einsatzplanes zurück, welche benötigt wird um zu überprüfen, ob der Benutzer den Einsatzplan anpassen darf. Stattdessen wurde eine Exception geworfen. Wir haben dann entdeckt, dass die Methode, welche genutzt wird um die Werte der Argumente auszulesen, einen String zurückgibt bei welchem `Id` steht, statt der Wert der `Id`. Somit sind wir auf die Suche nach einer neuen Möglichkeit gegangen und sind in der Methode `ArgumentValue` fündig geworden. Mit dieser Methode hat alles wieder funktioniert wie zuvor und der:die Abteilungsleiter:in wurde wieder korrekt autorisiert.

```
1     private static int? GetAccessObjectId(
2         IResolverContext resource,
3         string argumentName)
4     {
5         var idArgumentNode =
              resource.Selection.SyntaxNode.Arguments
6             .SingleOrDefault(
7                 a => a.Name.Value == argumentName);
8
9         if (idArgumentNode == null)
10        {
11            return null;
12        }
13
14        return int.Parse((string)idArgumentNode.Value.Value!);
15    }
```

Auflistung 13.8: Methode GetAccessObjectId mit alten Funktionen

```
1     private static int? GetAccessObjectId(
2         IResolverContext resource,
3         string argumentName)
4     {
5         return resource.ArgumentValue<int?>(argumentName);
6     }
```

Auflistung 13.9: Methode GetAccessObjectId mit neuen Funktionen

13.3.3 Externe News ohne Autorisierung ersichtlich

Diese Herausforderung war in diesem Sinne keine technische Herausforderung, sondern vielmehr eine Wissenslücke. Da wir beide, wie schon erwähnt, wenig bis keine Kentnisse in ASP.NET und Flutter hatten, kam uns dies bei diesem Feature ein wenig in die Quere. Mit dem Frontend Framework Flutter haben wir uns schnell zurechtgefunden, da wir bereits Erfahrungen mit Frontend Frameworks wie React und Angular hatten und auch viel Code von unseren Vorgängern übernehmen konnten. Beim Backend hatten wir mehr Mühe. Wir kannten uns schon mit C# aus, somit war die Sprache kein Problem, dafür war aber die Funktionsweise des Frameworks noch unbekannt. In gewissem Sinne die „Magie“, welche vom Framework gemacht wird. Als eines der ersten Features haben wir die externen News geplant, da diese wichtig waren und die Möglichkeit die App im AppStore zu veröffentlichen erhöht wurde. Allerdings war dies kein einfaches Feature um mit der Implementation zu beginnen, da es keine Autorisierung bzw. Authentisierung benötigt. Alle bisherigen Komponenten haben diese benötigt und somit konnten wir nichts von der bestehenden Logik übernehmen.

Wir mussten uns also in ASP.NET schlau machen und Möglichkeiten finden, die externen News so zu implementieren wie wir uns das vorgestellt haben. Eigentlich funktionierte alles wie gewünscht und wir konnten sehr schnell Fortschritte erzielen in unserem ASP.NET-Wissen. Als wir soweit fertig waren mit der Implementation, fiel uns ein Problem auf, mit welchem wir ein bisschen länger beschäftigt waren. Dies steht wie so oft in keinem Verhältnis dazu, wie schnell es eigentlich gelöst sein könnte. Und zwar gibt es die Möglichkeit bei den externen News ein Bild hochzuladen. Dieses Bild wird auf dem Backend-Server gespeichert und zur Verfügung gestellt. Das Backend war aber so konfiguriert, dass jeder Zugriff auf die `StaticFiles` mit einem authentisierten Benutzer stattfinden muss. Wir haben uns deshalb nochmals schlau gemacht wie diese `StaticFiles`, also das zur Verfügungstellen der Dateien, funktioniert. Dabei haben wir in der Datei `Startup.cs`, welche von ASP.NET gebraucht wird, gesehen, dass der Status der Benutzeranmeldung abgefragt wird. Somit mussten wir nur noch bei der `if` Abfrage einen `Case` einbauen, welcher überprüft ob auf Dateien der externen News zugegriffen werden soll und dies in dem Fall zulässt. Eine einfache Lösung, welche viele Stunden Einarbeitung mit sich brachte.

```
private void ConfigureStaticFiles(IApplicationBuilder app)
{
    app.UseStaticFiles(new StaticFileOptions
    {
        OnPrepareResponse = ctx =>
        {
            if (ctx.Context.User.Identity != null ||
                ctx.Context.User.Identity.IsAuthenticated ||
                ctx.Context.Request.Path
                .StartsWithSegments("/StaticFiles/external_news"))
            {
                return;
            }

            ctx.Context.Response.StatusCode =
                (int)HttpStatusCode.UnAutorized;
            ctx.Context.Response.ContentLength = 0;
            ctx.Context.Response.Body = Stream.Null;
        },
        FileProvider = new PhysicalFileProvider(
            this.Cfg.GetValue<string>("StaticFiles:ROOT_DIRECTORY")),
        RequestPath = "/" +
                        this.Cfg.GetValue<string>(
                            "StaticFiles:ENDPOINT"),
    });
}
```

Auflistung 13.10: Authentisierung wird beim zur Verfügung stellen von Dateien überprüft

14 Qualitätssicherung

In diesem Kapitel gehen wir auf die in Kapitel 7 analysierten Risiken ein. Der Fokus dabei liegt darauf, ob ein Risiko eingetreten ist, wie mit einem allfällig entstandenen Schaden umgegangen wurde oder wie die Risiken mitigiert und minimiert werden konnten.

Abschliessend möchten wir auch auf die Nicht-funktionalen Anforderungen, welche in Kapitel 9.3 definiert wurden und deren Einhaltung nach Abschluss des Projektes eingehen.

14.1 Risikomanagement

Aufgrund der guten Vorarbeit aus [1] konnten schon viele technische Risiken abgedeckt oder mitigiert werden. Trotzdem gab es aufgrund neuer Anforderungen und deren Integration in die bestehende Architektur diverse Risiken, welche mit einer guten Analyse abgefangen werden konnten und somit nicht eingetreten sind.

Wir möchten im Folgenden nun etwas genauer auf ein Risiko, welches eingetroffen und uns viel Zeit gekostet hat, eingehen.

14.1.1 Applikation kann nicht im AppStore aufgeschalten werden

Aufgrund von strengen Limitationen für den öffentlichen AppStore ist das Risiko „Applikation kann nicht im AppStore aufgeschalten werden“ aus Kapitel 7.1.6 eingetreten. Gemäss der Rückmeldung von Apple haben wir versucht verschiedene zusätzliche öffentliche Funktionen in die App einzubauen, damit auch für externe Benutzer ohne Login ein gewisser Funktionsumfang vorhanden ist. Trotzdem wurde die App aber leider erneut abgelehnt, ohne ein wirklich aussagekräftiges Feedback zu erhalten. Sie verwiesen uns wiederholt auf den „Apple Business Manager“.

Warten auf Prüfung	damian.kalberer@outlook.com	11.11.2021 um 11:31 Uhr
In Prüfung	Apple	11.11.2021 um 17:24 Uhr
Abgelehnt	Apple	12.11.2021 um 00:32 Uhr
In Vorbereitung zur Übermittlung	damian.kalberer@outlook.com	30.11.2021 um 01:55 Uhr
Warten auf Prüfung	damian.kalberer@outlook.com	30.11.2021 um 01:55 Uhr
In Prüfung	Apple	01.12.2021 um 20:24 Uhr
Abgelehnt	Apple	01.12.2021 um 20:50 Uhr

Abbildung 14.1: Übersicht über Verlauf der eingereichten iOS builds

Nach einer Einsprache und erneutem Review haben wir das Feedback erhalten, dass der Grossteil der App nur für Mitarbeiter:innen verfügbar ist und deshalb als „Business App“ deklariert und über den Business Manager verteilt werden muss.

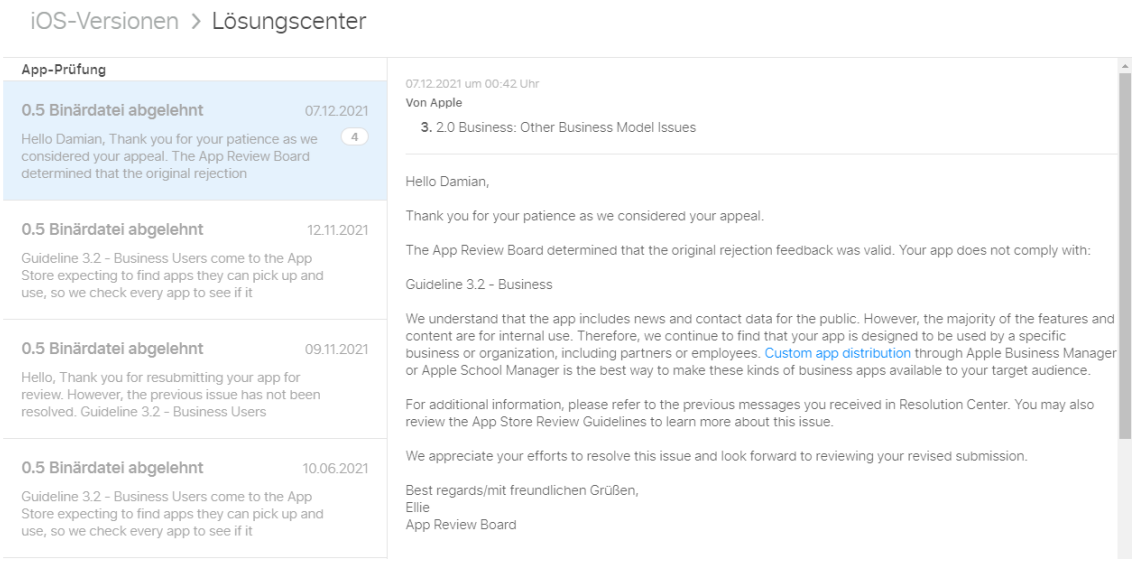

Abbildung 14.2: Übersicht über Rückmeldung auf eingereichten Antrag

Grundsätzlich hat uns das Eintreffen dieses Risikos einen Schaden von 5 Stunden verursacht. Dieser Wert liegt unter dem von uns maximal geschätzten Schaden von 16 Stunden und hatte somit praktisch keinen negativen Einfluss auf das Projekt. Diese 5 Stunden sind die effektiv verlorenen Stunden, welche wir zusätzlich nur für das Deployment aufwenden mussten.

Da wir aber aufgrund des erhaltenen Feedbacks von Apple zusätzliche Features implementierten, welche auch für externe Benutzer verfügbar sind, kostete uns das bedeutend mehr Zeit. Mit dem Aufwand für die Implementation der Kontaktdaten sowie des Kontaktformulares entstand ein Schaden von 27 Stunden.

Wenn wir die externen News auch noch dazu zählen, würde sich der Schaden auf insgesamt 57 Stunden erhöhen. Da diese Funktion aber sowieso gewünscht wurde, beachten wir diese Funktionalität nicht als Reaktion auf ein Risiko und zählen diese zusätzlichen Stunden nicht zum Schaden des Risikos dazu.

Durch den zusätzlichen Einarbeitungsaufwand in dem „Apple Business Manager“ und dem initialen Problem des darin fehlenden Apps, entstanden nochmals 2 Stunden Mehraufwand.

Durch zusätzlichen Aufwand unter der Woche und am Wochenende konnte der eingetretene Schaden aber grösstenteils wieder kompensiert werden.

14.1.2 Flutter Package Inkompatibilität

Ein unerwartetes Risiko, welches eingetreten ist, ist die Inkompatibilität von verschiedenen Flutter Packages. Dabei waren die Packages selbst nicht inkompatibel zueinander, jedoch waren beide auf ein weiteres Package in unterschiedlichen Versionen angewiesen. Da die

Packages alle Open-Source sind, kann es eine gewisse Zeit dauern, bis die neuste Version der abhängigen Packages unterstützt wird.

Wir hatten das Problem, dass das Package für den Video Player (Chewie) nicht mit dem Package für den HTML-Viewer (flutter_html) kompatibel war. Beide Packages waren auf das Package *provider* angewiesen. Chewie allerdings in der Version 5.0.0 und flutter_html in der Version 6.0.0. Unter folgendem Link wurde bereits ein Issue bei Chewie eröffnet, um den Provider in der Version 6.0.0 zu unterstützen. Zum Zeitpunkt dieser Arbeit wird dies aber noch nicht unterstützt und deshalb mussten wir dieses Problem mit einem Workaround lösen.

14.2 Nicht-funktionale Anforderungen

Wie schon im Kapitel 9.3 haben wir die Nicht-funktionalen Anforderungen in die jeweilige Kategorie gruppiert. Dies ermöglicht eine bessere Übersicht und zusammengehörende NFR sind besser erkennbar.

Da sich viele NFRs seit [1] nicht verändert haben und mit den erstellten Linterrules weitergearbeitet wurden, wurden gewisse Teile von dieser Arbeit übernommen.

14.2.1 Funktionalität

In dieser Kategorie befinden sich alle NFRs, welche mit der Funktionalität der Applikation zu tun haben.

14.2.1.1 NFR01

Dieses NFR besagt, dass interne Daten erst nach erfolgreicher Authentifizierung und Authorisierung angesehen oder bearbeitet werden dürfen.

Die Authentifizierung und Authorisierung wurde bereits in [1] umgesetzt und nun noch weiter optimiert.

14.2.2 Benutzbarkeit

In dieser Kategorie befinden sich alle NFRs, welche mit der Benutzbarkeit der App zusammenhängen.

14.2.2.1 NFR02

Die Mitarbeiter:innen sollen sich schnell im App zurechtfinden und ohne zusätzliche Erklärung die für sie relevanten Information abrufen können.

Dieses NFR wurde erfüllt, da ein Usability-Test durchgeführt und die Applikation gemäss der Rückmeldung optimiert wurde.

14.2.2.2 NFR03

Gemäss diesem NFR müssen farbcodierte Informationen auch für farbenblinde Personen durch eine weitere Methode ersichtlich sein. Alle Informationen in der App sind textuell

oder grafisch ersichtlich und wurden teilweise mit Farbe zusätzlich ergänzt. Somit kann auch eine farbenblinde Person die App ohne Probleme benutzen.

14.2.2.3 NFR04

In diesem NFR geht es darum, dass die Informationen nicht nur textuell, sondern auch grafisch dargestellt werden. Da nicht alle Mitarbeiter:innen über gute Deutschkenntnisse verfügen, können sie sich über die grafischen Elemente trotzdem zurechtfinden und zu den nötigen Informationen gelangen. Für den grössten Teil der Funktionalität ist dies gelungen. Einzig bei der Administration werden gewisse Informationen nur textuell dargestellt. Da in der Administration aber grundsätzlich gute Deutschkenntnisse vorausgesetzt werden und auch vorhanden sind, stellt dies ein geringes Problem dar.

14.2.3 Zuverlässigkeit

In dieser Kategorie befinden sich alle NFRs, welche die Zuverlässigkeit der Applikation definieren.

14.2.3.1 NFR05

Dieses NFR besagt, dass bei unerwartetem Verhalten der Applikation eine globale Fehlerbehandlung vorhanden ist. Die Fehlerbehandlung ist global im Frontend sowie Backend implementiert über Input Validation und Error Messages und geben dem Benutzer Rückmeldungen über unvorgesehenes Verhalten.

14.2.4 Änderbarkeit (Wartbarkeit)

In dieser Kategorie befinden sich alle NFRs, welche mit der Änderbarkeit respektive der Wartbarkeit der Applikation zu tun haben.

14.2.4.1 NFR06

Dieses NFR definiert, dass Klassen innerhalb der Applikation nicht länger als 300 Zeilen sein dürfen. Damit kann eine bessere Lesbarkeit garantiert werden.

Frontend
Im Frontend wurde dieses NFR manuell geprüft, da kein Tool gefunden werden konnte, um dies zu automatisieren. Der Test-Code wurde bei dieser Analyse ignoriert. Dieser Wert wurde nur in drei Klassen überschritten. Bei diesen Klassen handelt es sich um komplexe `Views`. Da aber die verschiedenen Funktionen eng miteinander gekoppelt sind, machte ein Refactoring keinen Sinn und wurde aus diesem Grund nicht durchgeführt.

Backend
Dieses NFR wurde im Backend mithilfe eines Linter überprüft und weitestgehend eingehalten. Wo es nicht eingehalten werden konnte, wurde dies mit einem Kommentar signalisiert und begründet. Meistens wurden Klassen mit starker Kohäsion durch Dokumentationskommentare aufgeblasen und waren somit grösser als 300 Zeilen. Für solche Fälle wurde eine Ausnahme definiert.

14.2.4.2 NFR07

Bei diesem NFR geht es darum, dass Methoden innerhalb einer Klasse nicht länger als 30 Zeilen sein dürfen. Damit wird eine bessere Lesbarkeit der Methode garantiert.

Frontend
Im Frontend wurde dieses NFR per Linter geprüft. In fünf Fällen konnte die Regel nicht eingehalten werden. Bei vier davon handelt es sich um Methoden, die UI-Elemente generieren. Diese lassen sich nicht sinnvoll aufteilen, da der Grund für die erhöhte Anzahl an Zeilen auf die vielen Optionen zurückzuführen ist, welche einige der verwendeten Elemente benötigen. Im letzten Fall, bei welchem die Regel nicht eingehalten werden konnte, handelt es sich um die Methode `UserEditViewModel.modifyAccessControlList`. Diese Methode beherbergt ein Switch-Case-Statement, welches sich aufgrund der vielen verschiedenen Fälle beim Ändern der AccessControlList nicht trennen lässt. Es wurde daher entschieden, alle erwähnten Methoden zu belassen.

Backend
Diese Anforderung wurde im Backend mithilfe eines Linter geprüft und eingehalten. Vereinzelt wurden bei starker Kohäsion innerhalb der Methode oder einfachen Konstrukten wie `Switch`, welche lange Methoden zur Folge haben, Ausnahmen definiert und dokumentiert.

14.2.4.3 NFR08

Dieses NFR besagt, dass Zeilen nicht länger als 80 Zeichen sein dürfen. Dies fördert die Lesbarkeit der einzelnen Zeilen.

Frontend
Im Frontend wurde dieses NFR per Linter geprüft und konnte über den gesamten Code eingehalten werden. Zusätzlich wird der Code aber auch noch durch das Flutter-eigene Format-Tool in das Einhalten der Regel gezwungen, da dieses den Code entsprechend formatiert.

Backend
Dieses NFR wurde im Backend mithilfe eines Linter geprüft und eingehalten. Es wurden wenige Ausnahmen, primär im Test-Code definiert, um lange Strings auf einer Zeile platzieren zu können.

14.2.4.4 NFR09

Mit diesem NFR ist definiert, dass der McCabe-Wert jeder Methode unter zehn liegen muss.

Frontend
Im Frontend wurde dieses NFR per Linter geprüft. In vier Fällen konnte die Regel nicht eingehalten werden. Bei drei davon handelt es sich um Methoden, deren alleinige Aufgabe es ist, über ein Switch-Case-Statement zwischen mehreren Fällen zu unterscheiden. Hierbei ist es nicht möglich, zu einer Einhaltung der Regel zu kommen, da der Switch-Case nicht aufgeteilt werden soll beziehungsweise kann. Beim letzten Fall handelt es sich um die Methode `AccessControlList.isEqual`, welche den Wertevergleich zweier `AccessControlList`-Objekte ermöglicht. Da Dart unglücklicherweise keine Alternative

zum schrittweisen Vergleich der elf Attribute bietet, ist ein Refactoring nicht möglich. Es wurde daher entschieden, alle erwähnten Methoden zu belassen.

Backend
Der McCabe-Wert der Methoden im Backend wurde mithilfe einer Erweiterung in der IDE überprüft und analysiert. Bis auf fünf Methoden ist der Wert sogar unter oder gleich fünf. Von diesen fünf Methoden, welche einen McCabe-Wert über fünf haben, sind lediglich zwei über dem vorgegebenen Limit von zehn.

Dies sind die `HasControl`-Methode in der Klasse `AccessControlListExtension` (McCabe-Wert 22) und die `Message`-Methode in der Klasse `ErrorExtensions` (McCabe-Wert 21). Da beide Methoden nur triviale Logik beinhalten, werden keine weiteren Massnahmen bezüglich dieser Methoden ergriffen. Der McCabe-Wert wurde in beiden Fällen aufgrund des enthaltenen `Switch`-Statement in die Höhe getrieben.

14.2.4.5 NFR10

Über dieses NFR wird spezifiziert, dass Methoden innerhalb der Applikation höchstens fünf Argumente haben dürfen.

Frontend
Im Frontend wurde dieses NFR per Linter geprüft. In drei Fällen konnte die Regel nicht eingehalten werden. Die drei Methoden sind alles `Service`-Methoden, welche für die GraphQL-API benötigt werden. Theoretisch wäre es möglich, die Parameter jeweils zu einem Objekt zu konvertieren und dieses zu übergeben, jedoch müsste dies innerhalb der Methode, aufgrund der darin enthaltenen GraphQL-Query direkt wieder auseinandergenommen werden. Aufgrund der Tatsache, dass die Limite nur um ein Argument überschritten wird, wird ein Refactoring als nicht sinnvoll errachtet.

Backend
Im Backend konnte dieses NFR bei 15 Methoden nicht eingehalten werden. Hierbei handelt es sich bei allen betroffenen Methoden um solche, welche direkt von der GraphQL-API aufgerufen werden. Ausserdem arbeiten alle von ihnen intern mit mehreren Managern, um ihre Aufgabe zu erfüllen. Die Manager selbst erfüllen alle dieses NFR.

Da es sich bei den betroffenen Methoden um API-Methoden handelt, wird die Argumentliste zusätzlich durch die Dependency Injection aufgebläht. Aufgrund dieses Umstandes und aufgrund der Tatsache, dass alle übergebenen Argumente für einen erfolgreichen API-Call gebraucht werden, wurde beschlossen, dass diese Methoden so bestehen bleiben können.

14.2.4.6 NFR11

Dieses NFR besagt, dass die Testabdeckung des Codes mindestens 80% betragen muss.

Frontend
Im Frontend beträgt die Testabdeckung des testbaren, nicht automatisch generierten Codes rund 85.2%. Somit wurde der gewünschte Wert erreicht und sogar übertroffen.

Backend
Im Backend beträgt die Testabdeckung des testbaren, nicht automatisch generierten Codes 91.4%. Somit wurde der gewünschte Wert auch hier erreicht. Die Files `ProntoMiaDbContextModelSnapshot`

und `AccessControlListInputType` wurden excluded, da es sich um automatisch generierten Code respektive das Setzen von Default Values handelt. Damit diese Files die Code-Coverage nicht beeiflussen, haben wir uns zu diesem Schritt entschieden.

14.2.4.7 NFR12

Bei diesem NFR geht es darum, dass alle Aufrufszenarien der API dokumentiert sein müssen.

Im Backend wurde für jeden möglichen API-Aufruf ein Dokumentations-Kommentar angelegt, welcher den Aufruf und dessen Argumente beschreibt.

14.2.4.8 NFR13

Dieses NFR besagt, dass eine Installations- sowie eine Wartungsanleitung erstellt werden soll.

Es wurde sowohl im Frontend als auch im Backend eine Installationsanleitung in der Form einer README-Datei angelegt:

- Frontend:
  - https://github.com/Pronto-AG/Pronto-MIA-App/blob/master/README.md
- Backend:
  - https://github.com/Pronto-AG/Pronto-MIA-Server/blob/master/README.md

Auf das Erstellen einer Wartungsanleitung wurde verzichtet, da diese vom Kunden nicht gewünscht war.

15 Zeitauswertung

Für das Projekt wurden insgesamt 480h budgetiert, was pro Person 240h ergibt und mit dem vorgesehenen Aufwand der acht ECTS Punkte des Moduls „Studienarbeit“ entspricht.

Tatsächlich aufgewendet wurden im Rahmen der Studienarbeit 535h, welche sich wie folgt auf die Projektmitglieder, Aufwandarten und Projektwochen aufteilen:

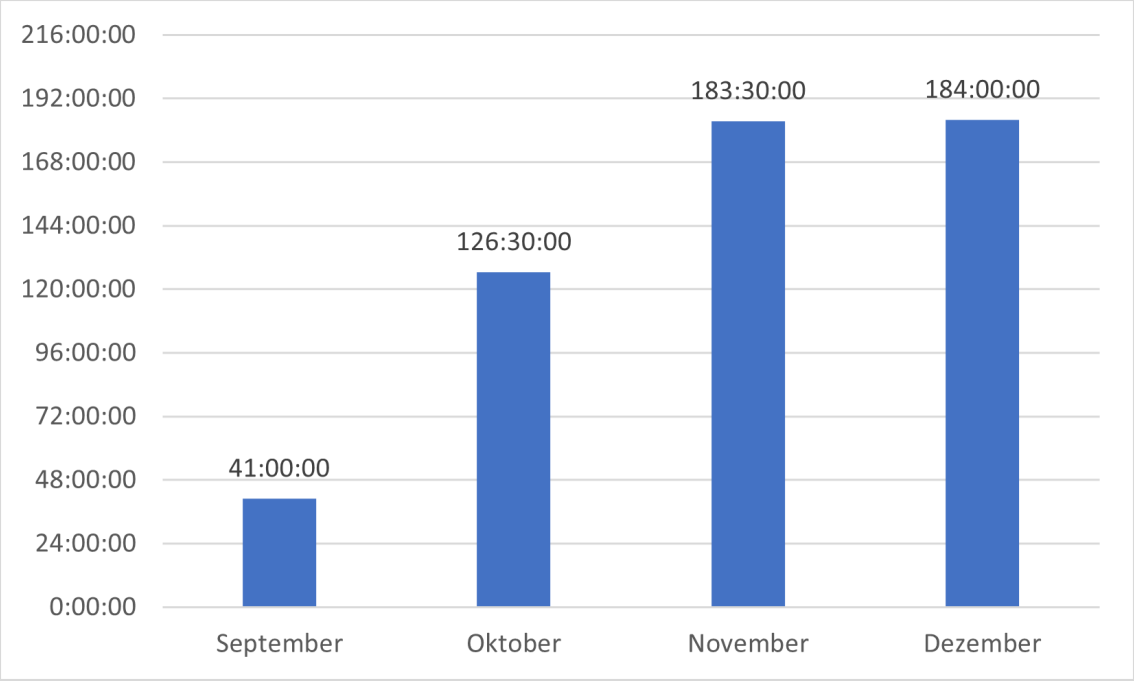

Abbildung 15.1: Zeitrapport, ausgewertet nach Monat

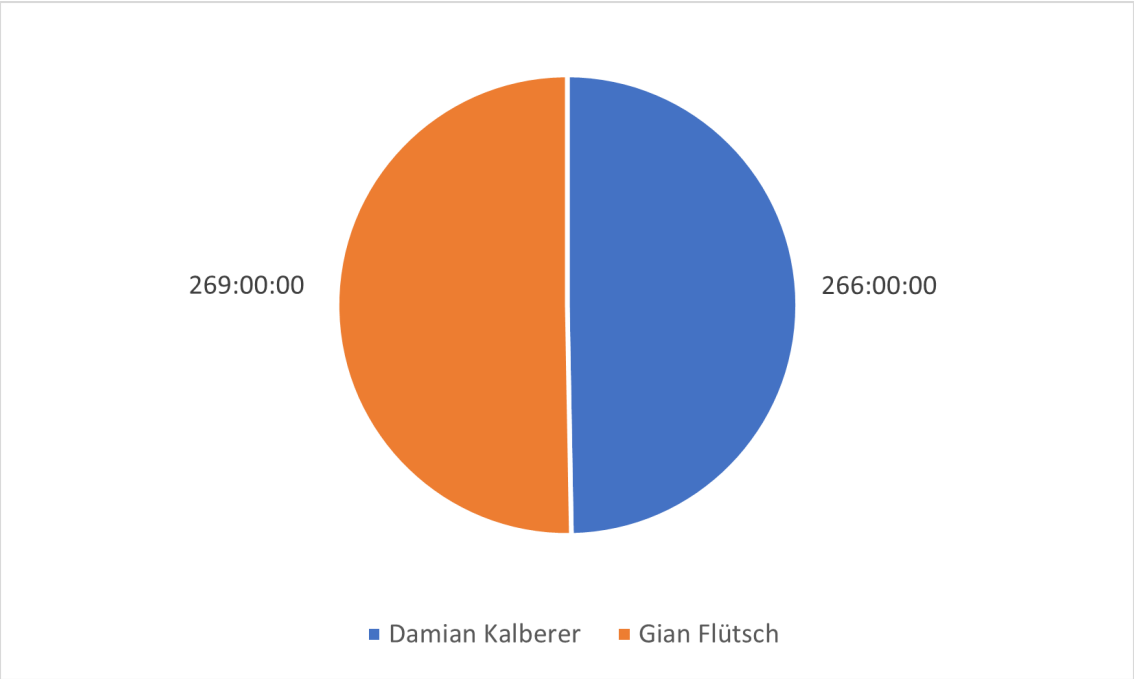

Abbildung 15.2: Zeitrapport, ausgewertet nach Personen

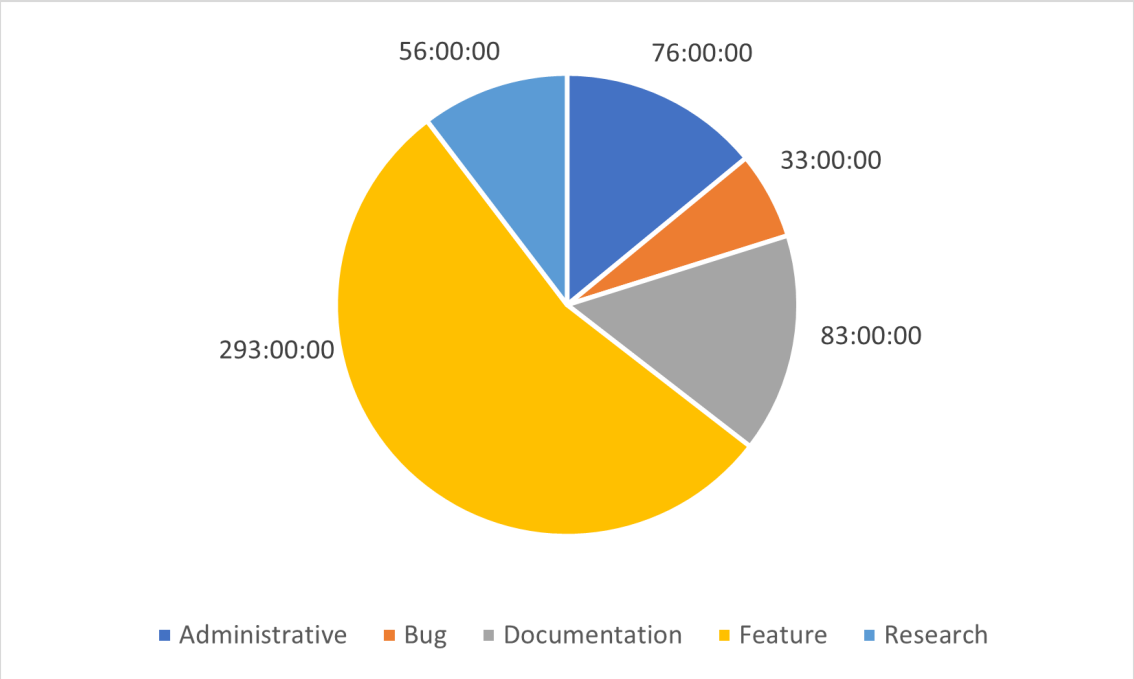

Abbildung 15.3: Zeitrapport, ausgewertet nach Kategorie

16 Schlussfolgerungen

In diesem Kapitel geben wir abschliessend unser Fazit über das gesamte Projekt ab und möchten noch ein paar Worte zum Ausblick und weiteren Vorgehen für die entwickelte Applikation verlieren.

16.1 Fazit

In der ursprünglichen Bachelorarbeit [1] wurde eine sehr gute Grundlage für die Weiterentwicklung der Applikation gelegt. Die erstellte Architektur musste nur geringfügig angepasst werden, um den neuen Anforderungen gerecht werden zu können. Im Rahmen dieser Studienarbeit konnten wir nun die „Pronto-MIA“ Applikation um wichtige und gewünschte Funktionalitäten problemlos erweitern.

Leider konnten wir aufgrund der beschränkten Zeit nicht ganz alle geplanten Funktionen umsetzen. Im Gegensatz dazu konnten wir mit der Kalenderintegration jedoch direkt zwei Anforderungen fast vollständig abdecken.

Besonders zu erwähnen ist auch, dass die App aus dem PlayStore sowie dem AppStore installiert werden kann. Bei Google's PlayStore hat dies problemlos funktioniert. Dafür trafen wir bei Apple's AppStore auf umso grössere Herausforderungen. Aufgrund der Limitationen für Business Apps mussten wir auf den Business Manager ausweichen. Damit ist es möglich, die App trotzdem über den AppStore installieren zu können. Die App muss nun über einen *Redemption Token* bezogen werden und ist nicht für jeden öffentlich ersichtlich. Da die App aber hauptsächlich an die Mitarbeiter:innen der Pronto AG gerichtet ist, erachten wir dies nicht als grossen Nachteil und insgesamt überwiegen die Vorteile durch die App.

Weiter wurde während der Umsetzung der Fokus auf eine gute Wartungsfähigkeit, Testabdeckung sowie Erweiterbarkeit gelegt. Damit sollten allfällige Erweiterungen in Zukunft problemlos integriert und die App den steigenden Bedürfnissen und Entwicklungen angepasst werden können.

Wir sind überzeugt, dass der aktuelle Funktionsumfang der Applikation einen sehr guten Stand aufweist und der Pronto AG einen grossen Mehrwert bringt.

16.2 Ausblick

Mit dem Release v1.0.0 wurde die Applikation auf den Plattformen Android, iOS und Web ausgerollt. Nach Abschluss dieser Arbeit kann ein angemessener Funktionsumfang innerhalb der App angeboten werden.

Die Pronto AG ist sehr zufrieden mit dem aktuellen Stand der App und möchte diese einführen. Dazu wird zuerst ein Pilot mit einer kleinen Testgruppe gestartet und wenn dieser erfolgreich ist, wird die Applikation produktiv eingeführt.

Für den Betrieb sowie Fragen dazu wird sich die Pronto AG bei Mirko Stocker melden und anhand der Anfrage kann über das weitere Vorgehen entschieden werden.

Teil IV

Appendix

A Abschlussberichte

A.1 Damian Kalberer

Bei der Themenwahl im Frühling war für mich diese Arbeit bereits sehr zutreffend. Mich interessierte die App Entwicklung sehr und ich befasse mich gerne mit für mich neuen Technologien, wie in diesem Beispiel Flutter. Deshalb habe ich mich auch für diese Arbeit beworben und zum Glück auch bekommen. Zusammen mit Gian Flütsch diese Arbeit zu bewältigen, fand ich sehr hilfreich, da wir uns beide sehr gut in unserem Wissen ergänzen. Dadurch können wir sehr viele Anforderung abdecken und sind für viele Herausforderungen gut vorbereitet.

Die Arbeit war zu Beginn wie zu erwarten sehr herausfordernd. Es trafen unbekannte Technologien und bereits eine vorangegange Bachelorarbeit zusammen. Letzteres war für mich neu, da ich sonst immer Projekte von der "grünen Wiese"durchgeführt habe. Mit einer gut dokumentierten Arbeit und vielen Informationen im Web über die Technologien konnten wir das jedoch gut meistern.

Die Zusammenarbeit mit unserem Betreuer Mirko und den beiden Industriepartner Heidi und Ramon von der Pronto AG schätzte ich sehr. Sie waren stets hilfsbereit und unkompliziert. Dadurch war man automatisch noch motivierter ein gutes Ergebnis zu erzielen, was mit der Tatsache, dass das Endprodukt danach im Berufsalltag eingesetzt werden soll, noch bestärkt wurde.

Für mich stach das Abschlussmeeting bei dieser Arbeit heraus. Man spürte, dass Heidi und Ramon sehr zufrieden mit dem Ergebnis sind und voller Tatendrang die App im Unternehmen auszurollen. Das find ich für mich persönlich sehr wichtig, da ich zufrieden mit meiner Arbeit sein kann und gerne auch wieder so eine Arbeit durchführen will.

Zum Abschluss kann ich sagen, dass ich die Studienarbeit als eine sehr interessante und herausfordernde Arbeit empfand, welche wir aus meiner Sicht gut gemeistert haben. Ich konnte mir neues Wissen aneignen und bestehendes Wissen festigen, was ich bestimmt in der kommenden Bachelorarbeit gut nutzen kann.

A.2 Gian Flütsch

Zu Beginn der Studienarbeit konnte ich mir noch nicht vorstellen, was alles auf uns zukommen wird. Dieses Projekt war für mich zugleich auch das grösste Softwareprojekt, welches ich bisher absolviert habe. Wir wussten, dass unsere Arbeit auf einer Bachelorarbeit aus dem letzten Semester aufbaut, in der die Grundlagen für die Applikation gelegt wurden. In unserer SA geht es vorallem darum, noch fehlende Funktionalitäten zu implementieren.

Von bereits absolvierten Modulen kannte ich C# sowie die Entwicklung für Android bereits. Dabei hat mich vor allem die Mobile-Entwicklung fasziniert und ich habe mich deshalb sehr gefreut in der Studienarbeit wieder diese Thematik aufgreifen zu können. Mit den verwendeten Frameworks, Flutter und ASP.NET, habe ich vor dem Projekt noch nicht gearbeitet und konnte somit viel Neues lernen. Dass wir dieses Projekt zugleich mit einem Industriepartner durchführen konnten, war ein weiteres Argument für mich.

Mit der im Engineering Projekt bereits erfolgreich eingesetzten Projektmanagementmethode Scrum+ konnten wir unser Wissen damit während der Studienarbeit nochmals vertiefen. Mit der gewählten agilen Vorgehensweise konnten wir sehr flexibel und effizient am Projekt arbeiten. Weiter wurde das Increment nach jedem Sprint dem Product Owner vorgestellt und es konnte direkt ein Feedback eingeholt werden. Dieses Feedback wurde wiederum direkt in den nächsten Sprint eingeplant und umgesetzt. Somit war der Kunde stets über den aktuellsten Stand informiert und die Entwicklung ging stets in die richtige Richtung.

Die in den späteren Projektphasen aufgetretenen Probleme konnten durch gute Teamarbeit und grossen Arbeitseinsatz erfolgreich gemeistert werden. Eines der Probleme betraf die Verteilung des Apps in Apple's AppStore. Zu Beginn konnte ich mir nicht vorstellen, dass dies ein so schwieriges Unterfangen werden könnte und Apple sehr strikt bei der Veröffentlichung von Apps agiert. Trotz unseres zusätzlichen Aufwandes mit weiteren Funktionalitäten mussten wir schlussendlich die App über den Business Manager verteilen. Dies ist nicht weiter schlimm, da die App schlussendlich trotzdem auf iPhones installiert werden kann.

Ein unterschätztes Problem war die Inkompatibilität von Flutter-Packages untereinander sowie auch mit gewissen Plattformen. Dieses Problem konnten wir mit anderen Packages oder eigenen Implementationen ohne grossen zeitlichen Verlust lösen.

Abschliessend gilt es zu sagen, dass ich die Arbeit als eine durchaus herausfordernde, aber auch als sehr spannende Aufgabe empfand. Ich konnte das im Studium erlernte Wissen anwenden, vertiefen und viele neue Erfahrungen sammeln. Mit dem Endprodukt bin ich sehr zufrieden und überzeugt, dass es der Pronto AG einen Mehrwert bringen wird. Ich freue mich, die kommende Bachelorarbeit wiederum mit Damian Kalberer bestreiten zu dürfen und die neu gewonnenen Erfahrungen dort einfliessen zu lassen.

B Literaturverzeichnis

[1] C. Kirchmeier und Y. Vogt. „Mitarbeiter-Informationsapp für die Pronto AG,“ Studiengang Informatik, OST. (2021).

[2] „Git Feature Branch Workflow.“ Englisch, Atlassian. (2021), Adresse: https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow (besucht am 01. 10. 2021).

[3] „ISO/IEC 9126-1:2001.“ Englisch, International Organization for Standardization. (2001), Adresse: https://www.iso.org/standard/22749.html.

[4] „Usability Testing Of Mobile Applications: A Step-By-Step Guide.“ Englisch, Usability Geek. (2021-10-10), Adresse: https://usabilitygeek.com/usability-testing-mobile-applications/#testing.

[5] „LaTeX– A document preparation system.“ Englisch, The LaTeXProject. (2021-12-10), Adresse: https://www.latex-project.org/.

[6] „Programmierschnittstelle.“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://de.wikipedia.org/wiki/Programmierschnittstelle.

[7] „Clockify Time Tracking.“ Englisch, Clockify. (2021-12-01), Adresse: https://clockify.me/.

[8] „Dependency Injection.“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://de.wikipedia.org/wiki/Dependency_Injection.

[9] „Framework.“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://de.wikipedia.org/wiki/Framework.

[10] „FURPS.“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://en.wikipedia.org/wiki/FURPS.

[11] „GitHub.“ Englisch, GitHub. (2021-12-20), Adresse: https://github.com/.

[12] „Kohäsion (Informatik).“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://de.wikipedia.org/wiki/Koh%C3%A4sion_(Informatik).

[13] „Lint (Programmierwerkzeug).“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://de.wikipedia.org/wiki/Lint_(Programmierwerkzeug).

[14] „McCabe-Metrik.“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://de.wikipedia.org/wiki/McCabe-Metrik.

[15] „Microsoft Teams.“ Deutsch, Microsoft. (2021-12-20), Adresse: https://www.microsoft.com/de-ch/microsoft-teams/group-chat-software.

[16] „Open Source Initiative.“ Englisch, opensource. (2021-12-20), Adresse: https://opensource.org/licenses/MIT.

[17] „Open Source Initiative.“ Englisch, opensource. (2021-12-20), Adresse: https://opensource.org/osd.

[18] „Semantic Versioning 2.0.0.“ Deutsch. (2021-12-20), Adresse: https://semver.org/lang/de/.

[19] „Single-Responsibility-Prinzip.“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://de.wikipedia.org/wiki/Single-Responsibility-Prinzip.
[20] „Usability-Test.“ Deutsch, Wikipedia. (2021-12-20), Adresse: https://de.wikipedia.org/wiki/Usability-Test.

C Abbildungsverzeichnis

D Tabellenverzeichnis

E Auflistungsverzeichnis

F | Glossar

LaTeX

Textsystem zur Erstellung von technischen und wissenschaftlichen Dokumenten. Verfügbar unter [5]. 22

API

API steht für „application programming interface “ was auf Deutsch übersetzt Programmierschnittstelle bedeutet. Die API fungiert als Anbindungsstelle, über welche andere Softwares auf die angebotenen Funktionen zugreifen kann. Mehr dazu: [6]. 41, 74, 75

Clockify

Tool für die Zeitschätzung und Erfassung. Verfügbar unter [7]. 21, 22, 24

Closed-Source

Als Closed-Source wird Software bezeichnet, bei welcher der Quellcode nicht öffentlich zugänglich gemacht wird. Diese Variante wird häufig durch Firmen verwendet, welche mit dieser Massnahme ihr intellektuelles Eigentum schützen wollen. 91

Cross-Plattform

Cross-Plattform ist ein Begriff aus der Informatik, welcher Softwares beschreibt, die auf mehreren Plattformen (Geräten, Betriebssystemen) lauffähig ist. 1, 11, 12, 20

Definition of Done

Eine Liste von Aufgaben, welche erledigt sein müssen, bevor ein Projekt oder Arbeitspaket als abgeschlossen angesehen werden kann. 21, 24, 25

Dependency Injection

Beschreibt eine Software-Engineering-Technik, in welcher ein Objekt andere Objekte, welche es selbst verwendet, übergeben bekommt. Somit kann eine globale Instanz die Abhängigkeiten verwalten und an die entsprechenden Objekte übergeben. Mehr dazu: [8]. 74

End of Elaboration

Meilenstein in der RUP-Methodik, bei welchem die grössten Risiken behoben sein müssen. Bei diesem Meilenstein wird entschieden, ob mit dem Projekt so fortgefahren werden kann oder nicht. 14, 20, 90, 91

End of Elaboration Checklist

Eine Liste von Bedingungen, welche zum End of Elaboration erfüllt worden sein müssen. Die Bedingungen sind wie folgt:

- Wir haben den Kunden verstanden (Requirements so vollständig wie nur möglich). Scope (=grober Funktionsumfang) ist vereinbart.
- Wir haben alle Werkzeuge im Griff (IDE, Versionskontrolle, Build Server, Deployment, Unit Testing, Workflow Tools, Wiki, ...).
- Wir wissen, wie die Architektur aussehen wird (Architektur skizziert, die grossen Interfaces festgelegt, Architektur-Prototypen gemacht).
- Für das User Interface gibt es einen ersten Entwurf (Grafiken,Wireframes, klickbarer Prototyp) und dem Kunden gefällt es.
- Wir haben die Marschrichtung (grobe Meilensteine) für die nächsten zwei Iterationen festgelegt, viele Arbeitspakete erstellt und dem Kunden eine genauere Zeitschätzung geliefert.
- Alle grossen Risiken, alle grossen Fragezeichen sind weg.

14, 17

Feature Branch Workflow

Workflow in Git, bei welchem die Entwicklung von Features in separaten Branches erfolgt. Dokumentiert unter [2]. 20

Feature-Freeze

Zeitpunkt in einem Projekt, ab welchem keine neuen Features mehr implementiert werden. 14, 15

Framework

Ein Framework ist ein Programmiergerüst und stellt einen Rahmen zur Verfügung, in welchem ein Entwickler seine Anwendung programmieren kann. Das Framework gibt hierbei die Rahmenbedinungen vor und stellt Funktionalitäten zur Verfügung. Mehr dazu: [9]. 44

FURPS

FURPS ist ein Modell aus dem Bereich der Softwarequalität. Es entstammt dem Englischen und fasst Qualitätsmerkmale von einer Software zusammen. Siehe auch [10]. 40

GitHub

Dienst zur Versionsverwaltung von Softwareprojekten. Verfügbar unter [11]. 22, 24

Kohäsion

Als „Kohäsion“ wird in der Programmierung die Eigenschaft von Code bezeichnet, genau eine wohldefinierte Aufgabe abzubilden. Klassen/Methoden mit starker

Kohäsion bilden entsprechend genau eine Aufgabe ab, so wie dies vom Single-Responsibility-Prinzip vorgegeben wird. Mehr dazu: [12]. 73

Linter

Software zur statischen Code-Analyse. Wird oft zur Überprüfung der Code-Formatierung verwendet. Mehr dazu: [13]. 24, 73, 74

McCabe-Wert

Der McCabe-Wert, auch „Cyclomatic Complexity“ genannt, ist eine Messgrösse in der Softwareentwicklung, mithilfe wessen die Komplexität einer Software angegeben werden kann. Siehe auch: [14]. 41, 73, 74

Microsoft Teams

Plattform für Kommunikation in Form von Chats und Anrufen. Verfügbar unter [15]. 22

MIT-Lizenz

Die MIT-Lizenz ist eine Open-Source-Softwarelizenz. Sie ermöglicht die Wiederverwendung der Software in Open-Source und Closed-Source Projekten. Der Lizenztext ist verfügbar unter: [16]. 32

NFR

Abkürzung für den englischen Begriff ”non-functional requirement”, was auf Deutsch nichtfunktionale Anforderung heisst. plural 40, 56, 71–75

Open-Source

Als Open-Source wird grundsätzlich eine Software bezeichnet, bei welcher der Quellcode öffentlich verfügbar ist. Zu Open-Source im eigentlichen Sinne gehört allerdings noch mehr. Die vollständige Definition kann hier eingesehen werden: [17]. 91

potentially shippable product

Softwareprodukt, welches in dieser Form an den Kunden ausgeliefert werden könnte. Alle implementierten Features sind komplett funktionsfähig. 14

RUP

Rational Unified Process. 20, 89, 91

Scrum+

Projektmethodik welche Scrum mit dem End of Elaboration von RUP verbindet. 13, 20, 82

Semantic Versioning

Standard zur Versionierung von Software, dokumentiert unter [18] 56

Single-Responsibility-Prinzip

Ein Prinzip aus der Softwareentwicklung welches besagt, dass Softwarekomponenten nur eine fest definierte Aufgabe zu erfüllen haben. Mehr dazu: [19]. 91

Software Engineering 2

Software Engineering 2 (neu SE Practices 2) ist ein Modul der OST, welches sich mit verschiedenen Methoden, Prinzipien, Design-Pattern und Best-Practices im Software Engineering auseinander setzt. 47

Usability-Test

Ein Usability-Test ist ein Test, welcher mit Benutzern einer Applikation durchgeführt wird, um deren Bedienbarkeit für den Endbenutzer zu evaluieren. Mehr dazu: [20]. 1, 15, 17, 25, 47, 48, 50, 51, 54, 64, 65, 71

WYSIWYG

WYSIWYG steht für "What you see is what you get" und wird häufig auch Echtzeitdarstellung genannt. Der Begriff wird häufig bei Programmen verwendet, welche dem Benutzer während der Bearbeitung bereits anzeigt, wie das Endprodukt aussieht. 37

G Zusatzinformationen

G.1 Wireframes

G.1.1 App

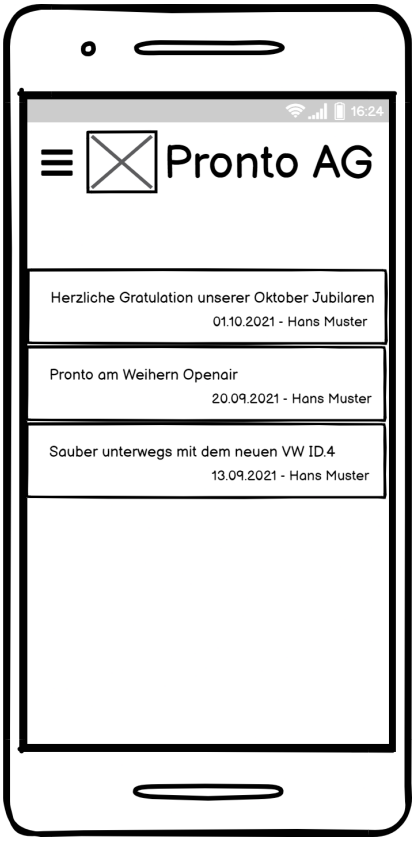

Abbildung G.1: Wireframe Landing Page

Abbildung G.2: Wireframe Menü Logged out

Abbildung G.3: Wireframe Login

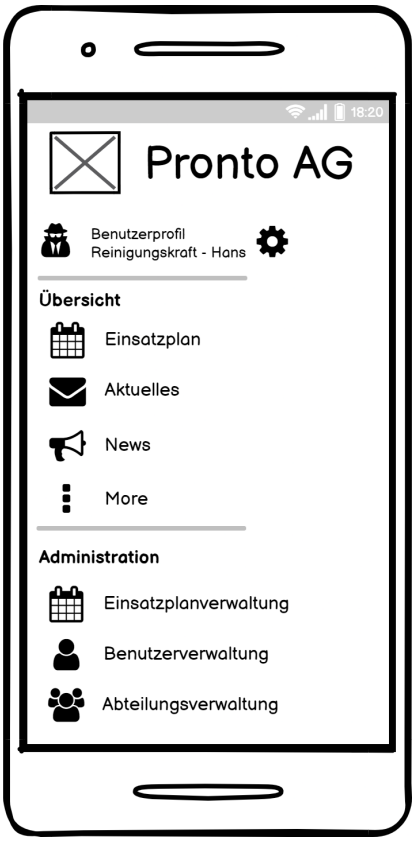

Abbildung G.4: Wireframe Menü Logged in

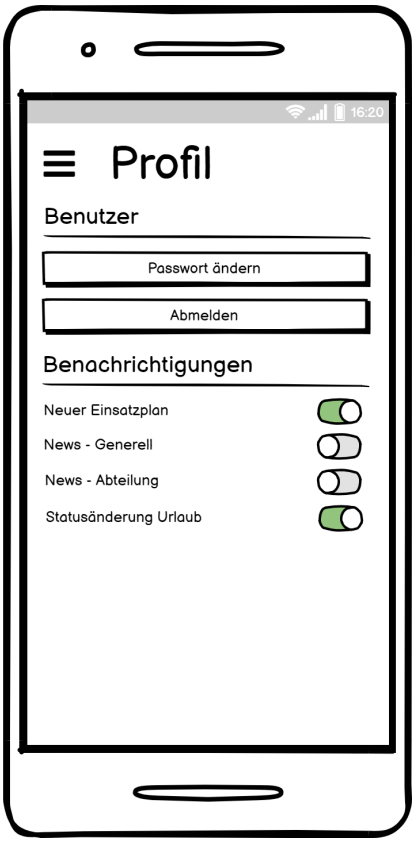

Abbildung G.5: Wireframe Profilübersicht

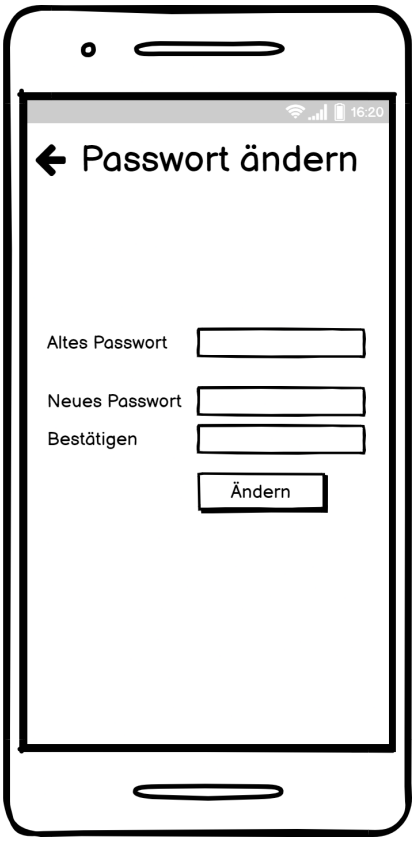

Abbildung G.6: Wireframe Passwort ändern

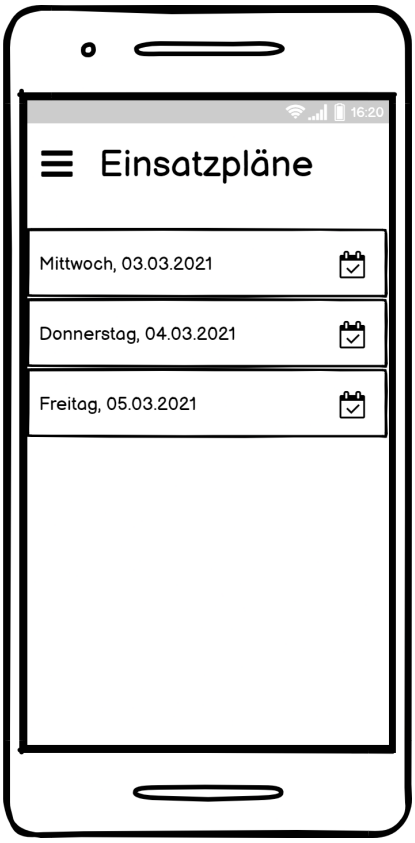

Abbildung G.7: Wireframe Einsatzplanübersicht

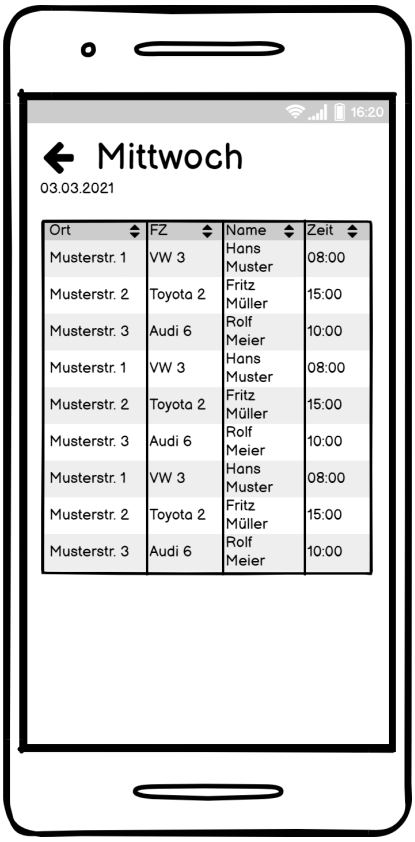

Abbildung G.8: Wireframe Einsatzplandetails

Abbildung G.9: Wireframe Newsübersicht

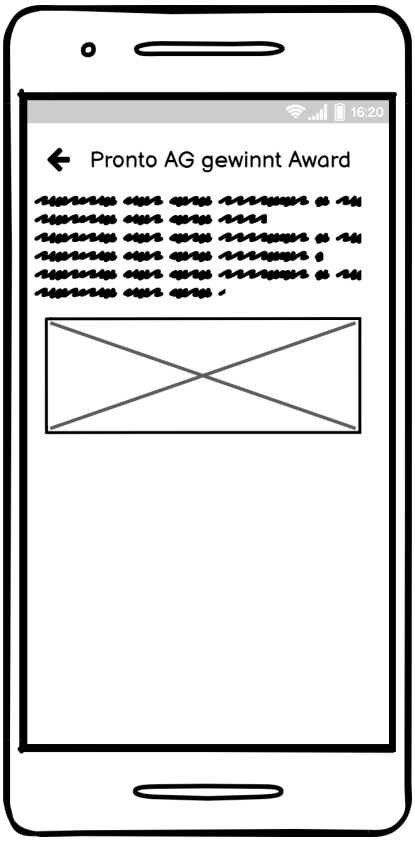

Abbildung G.10: Wireframe Newsdetails

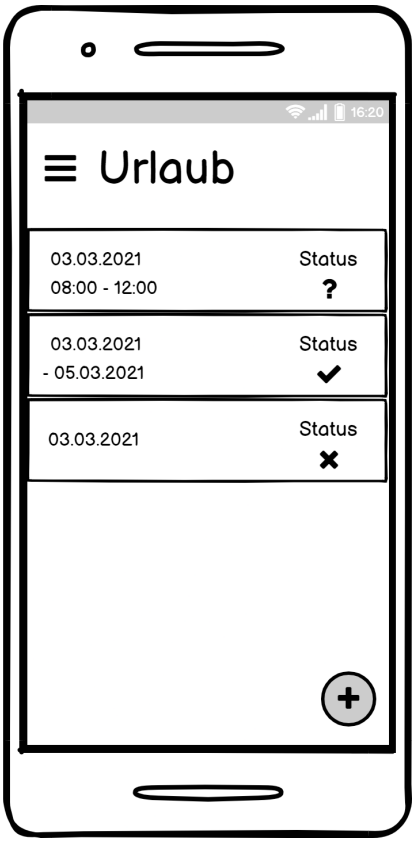

Abbildung G.11: Wireframe Urlaubsübersicht

Abbildung G.12: Wireframe Urlaub Antragstyp

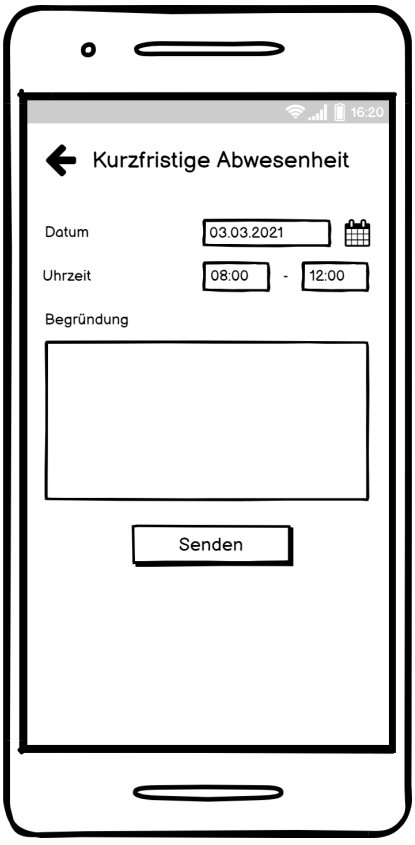

Abbildung G.13: Wireframe Kurzfristige Abwesenheit

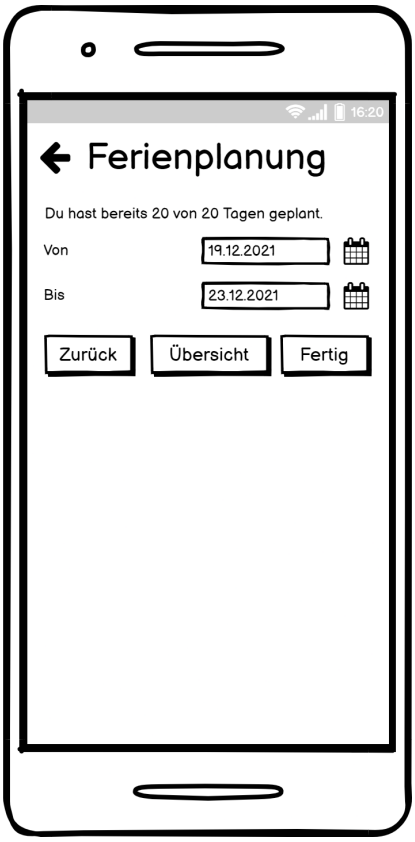

Abbildung G.14: Wireframe Jährliche Urlaubsplanung

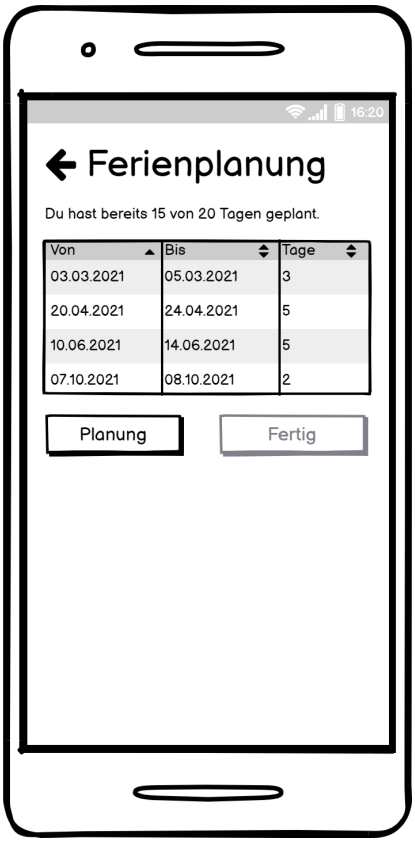

Abbildung G.15: Wireframe Jährliche Urlaubsübersicht

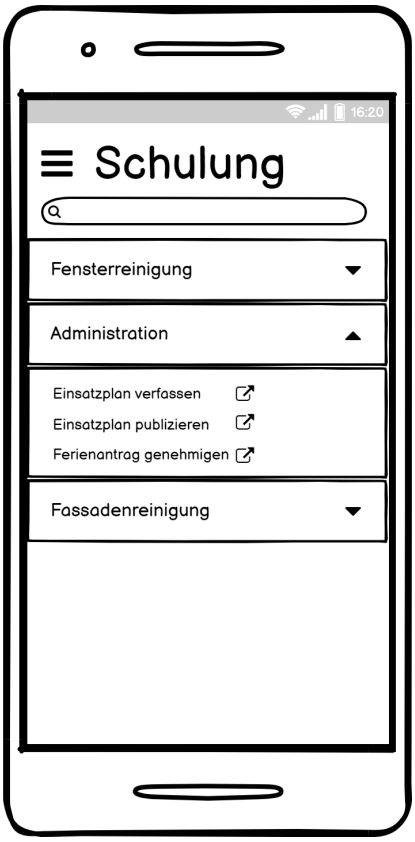

Abbildung G.16: Wireframe Schulungsübersicht

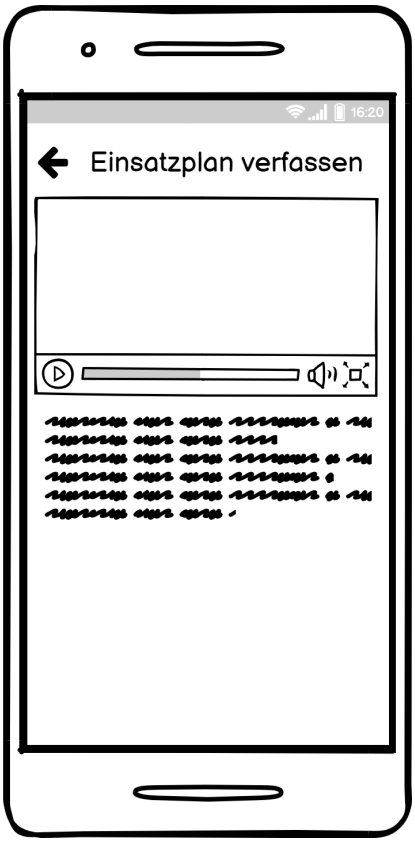

Abbildung G.17: Wireframe Schulungsdetails

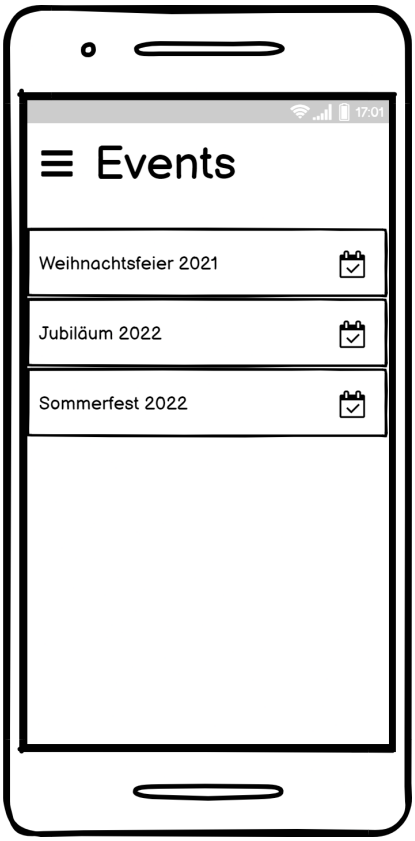

Abbildung G.18: Wireframe Eventübersicht

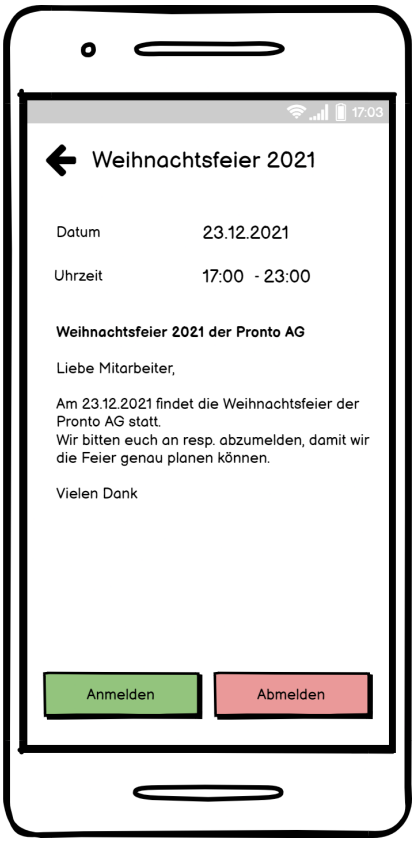

Abbildung G.19: Wireframe Eventdetails

G.1.2 Administrations-Webseite

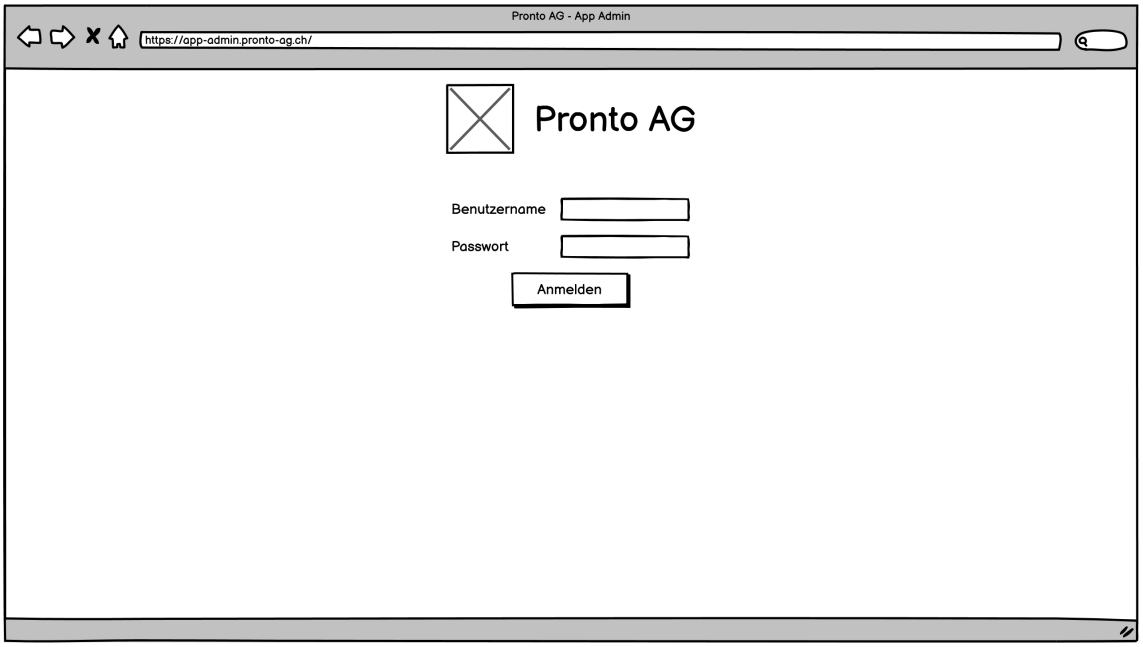

Abbildung G.20: Wireframe Login

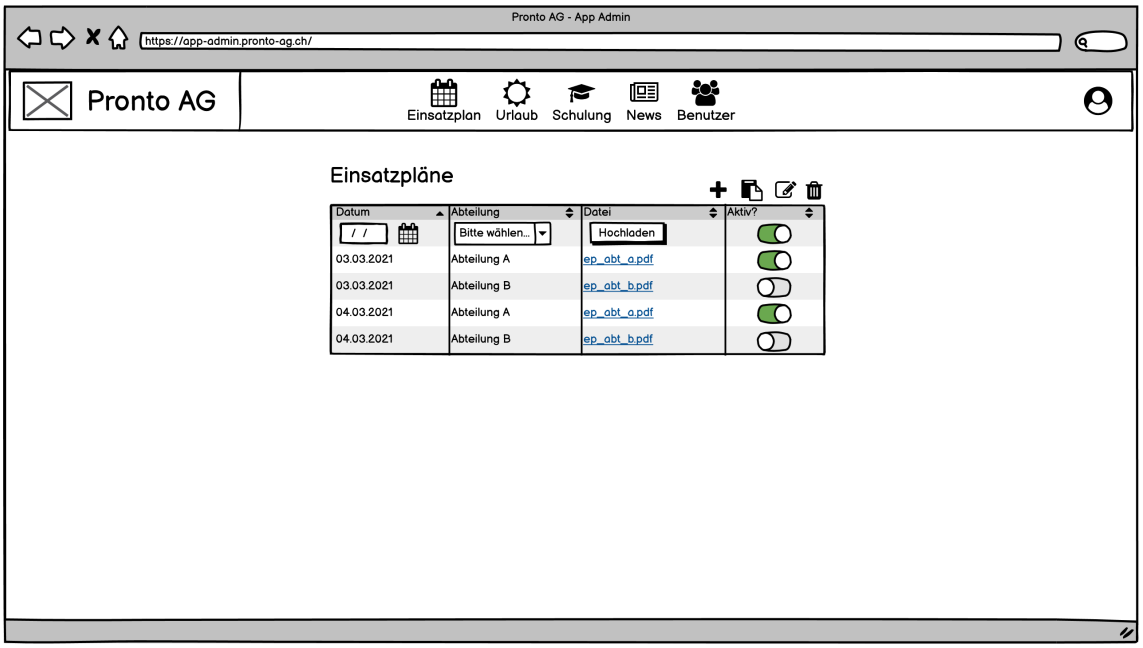

Abbildung G.21: Wireframe Einsatzpläne Ausbaustufe Dateien

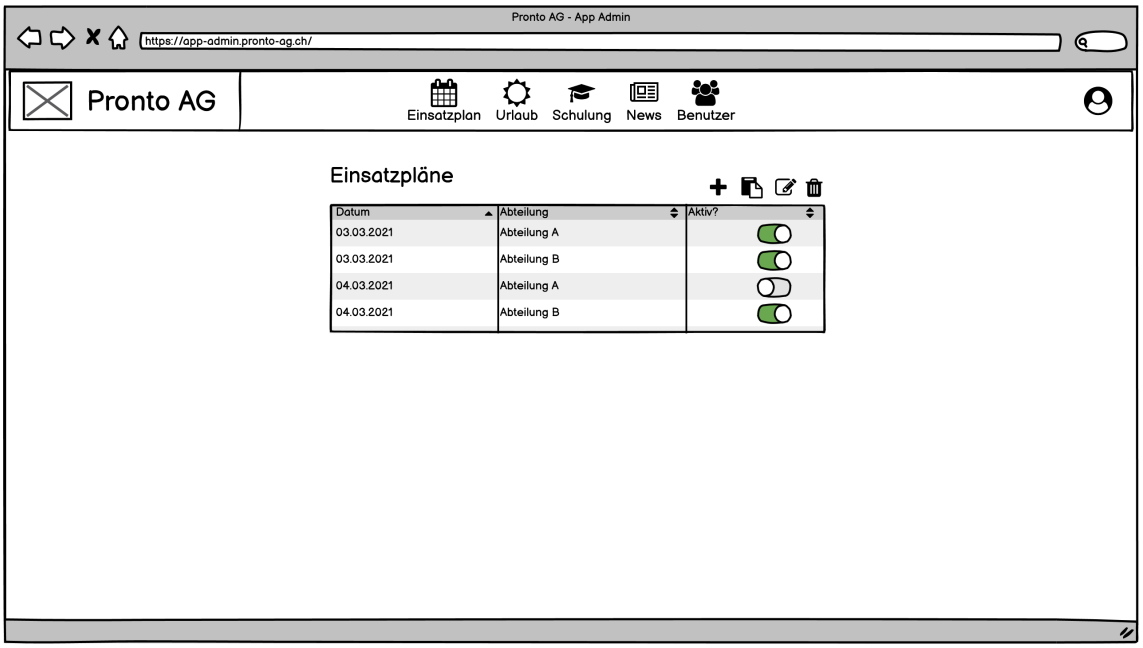

Abbildung G.22: Wireframe Einsatzpläne Ausbaustufe Integriert

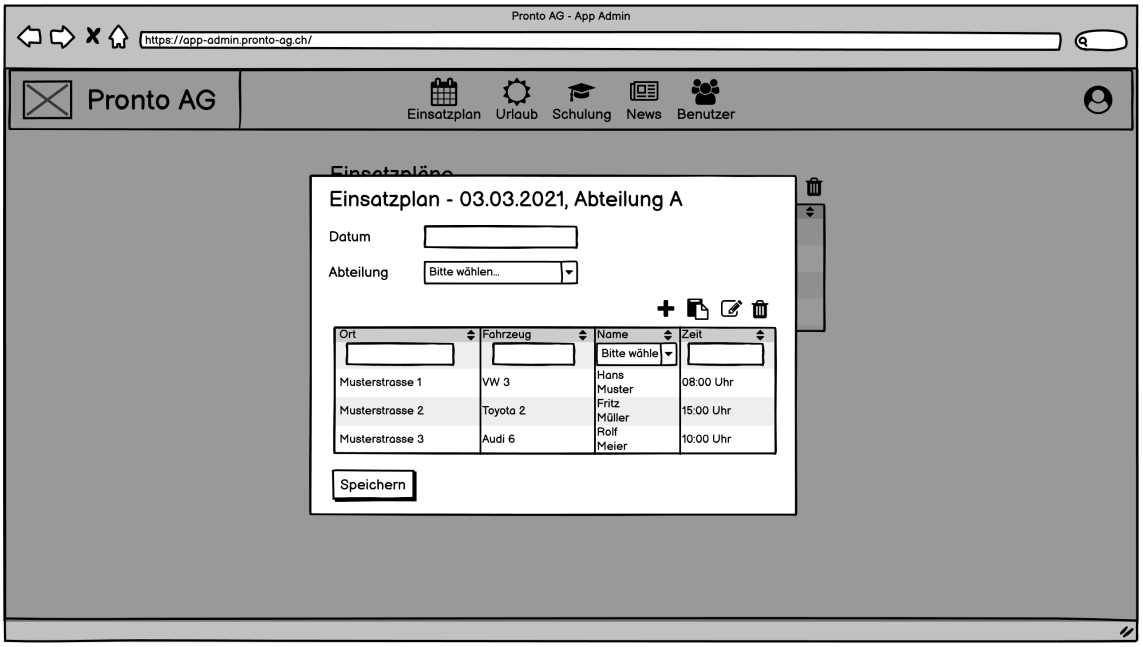

Abbildung G.23: Wireframe Einsatzplan erstellen

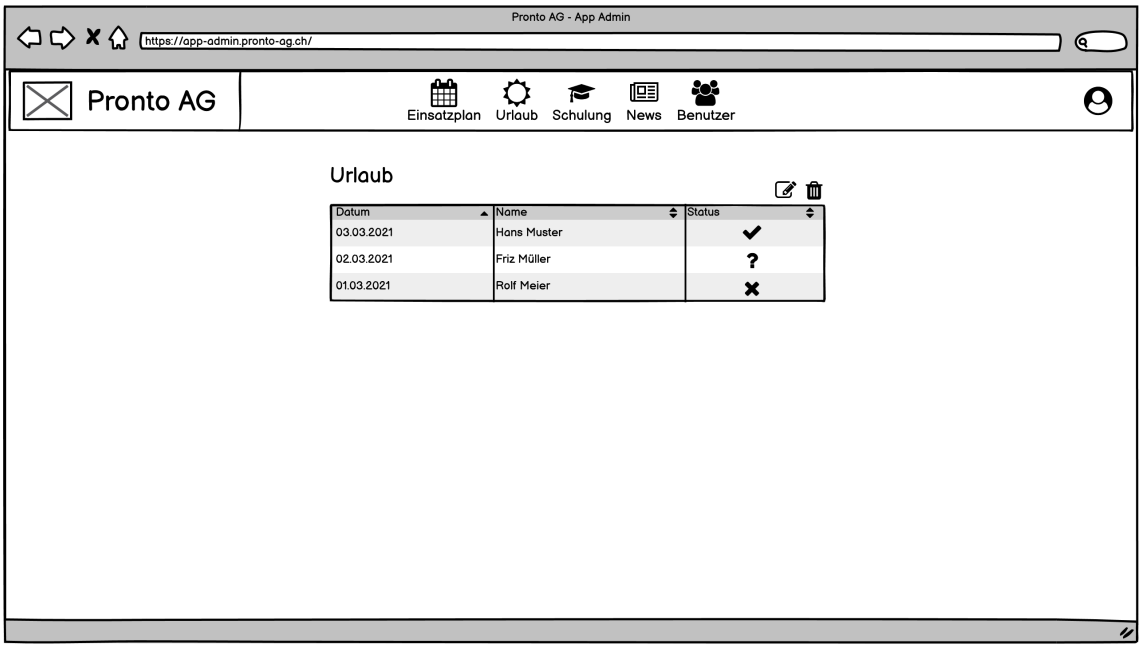

Abbildung G.24: Wireframe Urlaubsverwaltung

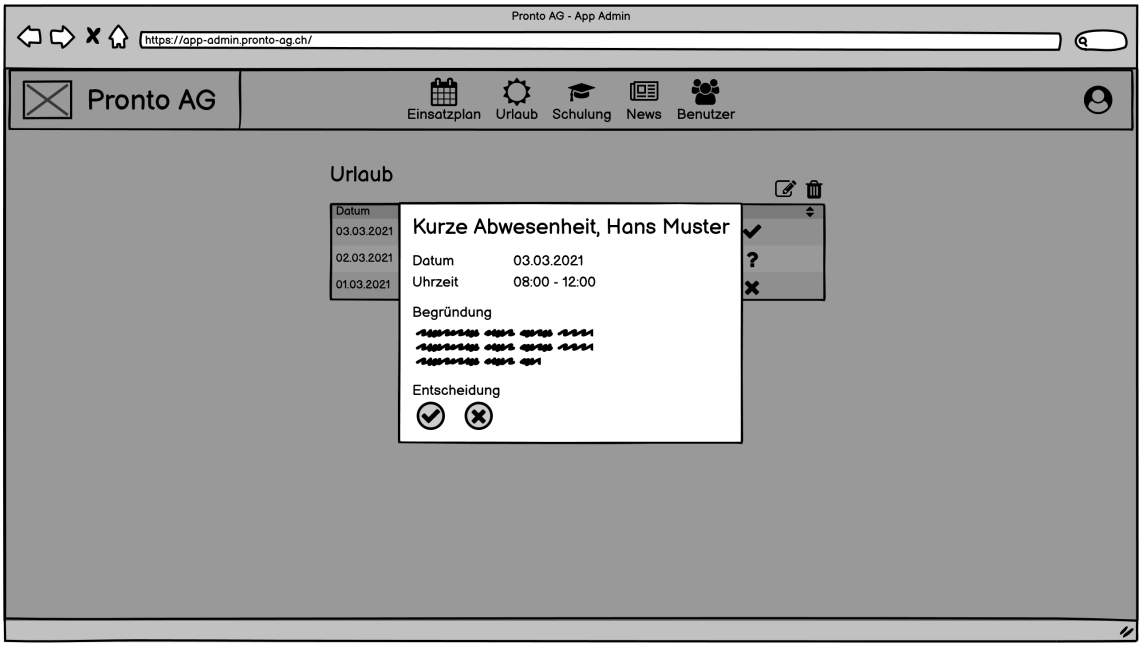

Abbildung G.25: Wireframe Urlaubsantrag

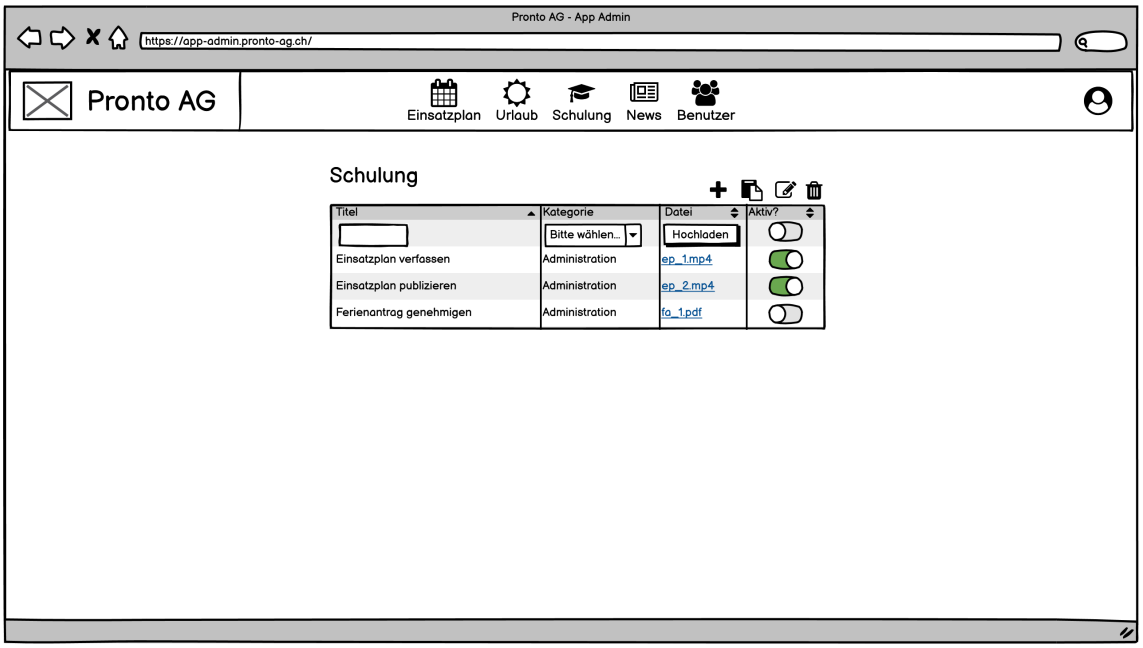

Abbildung G.26: Wireframe Schulungsverwaltung Ausbaustufe Dateien

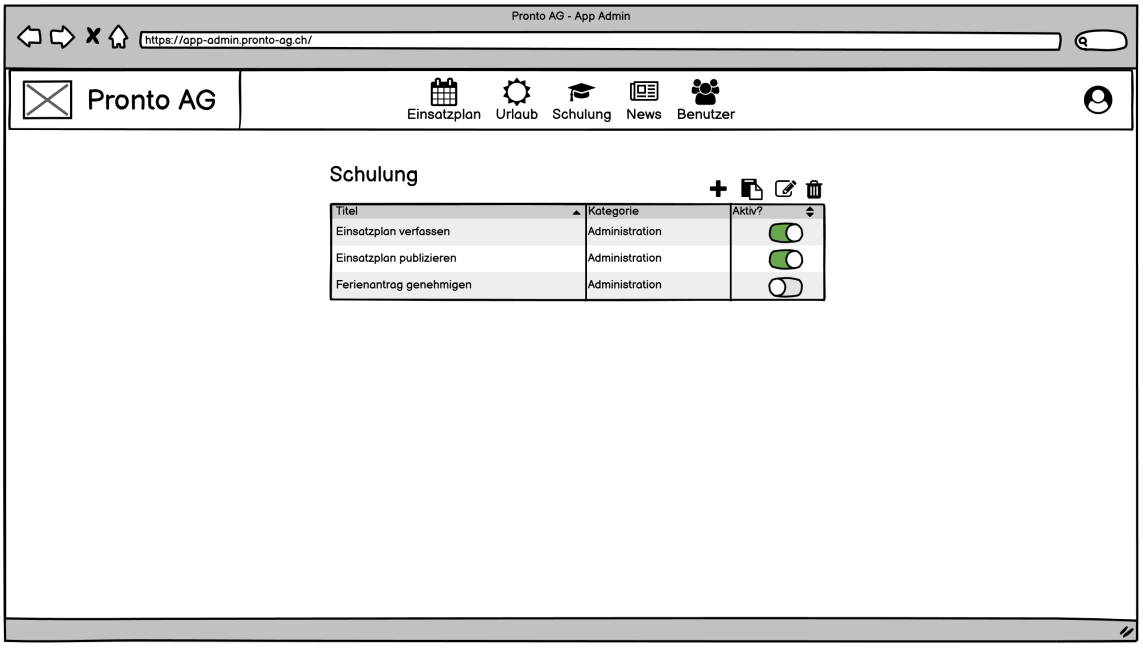

Abbildung G.27: Wireframe Schulungsverwaltung Ausbaustufe Integriert

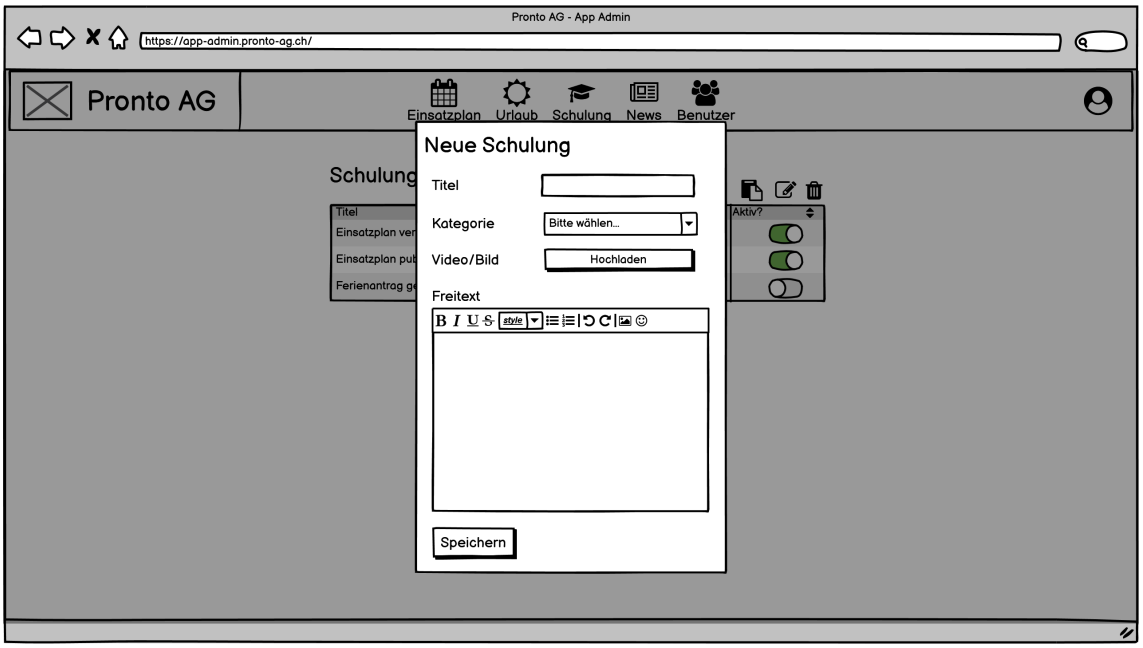

Abbildung G.28: Wireframe Schulungsinhalt erstellen

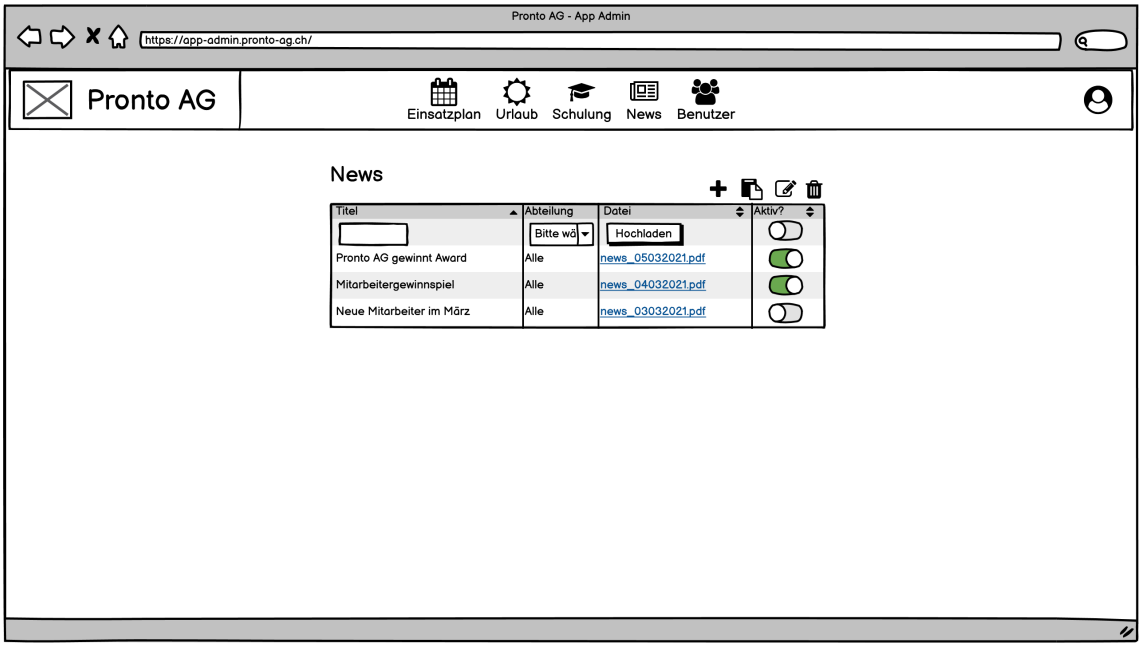

Abbildung G.29: Wireframe Newsverwaltung Ausbaustufe Dateien

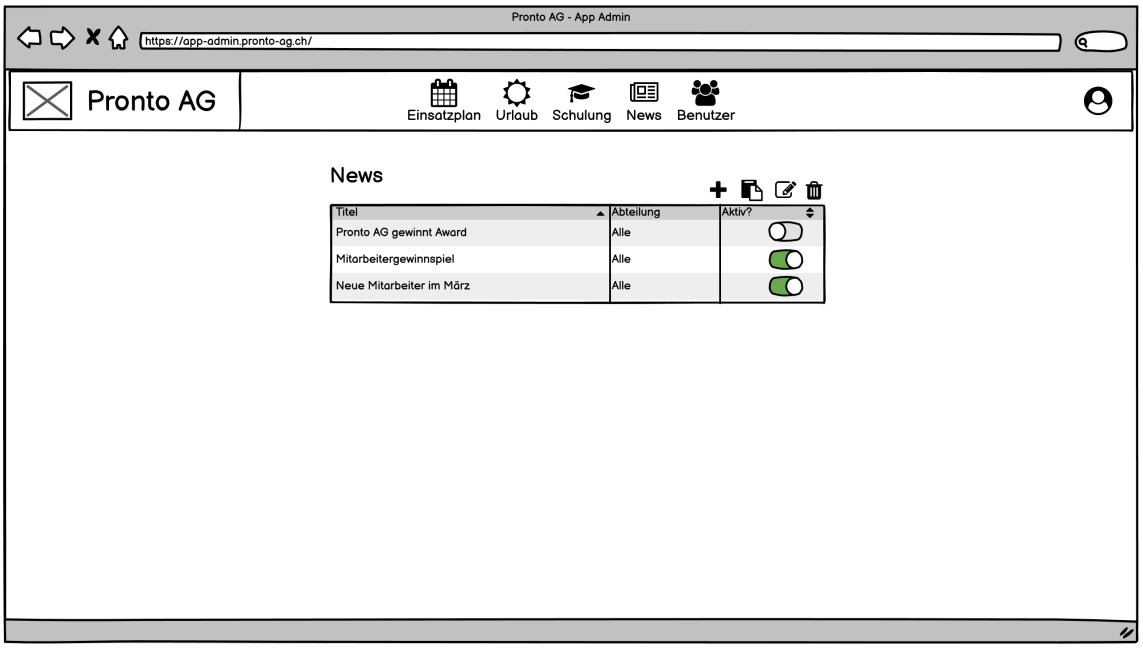

Abbildung G.30: Wireframe Newsverwaltung Ausbaustufe Integriert

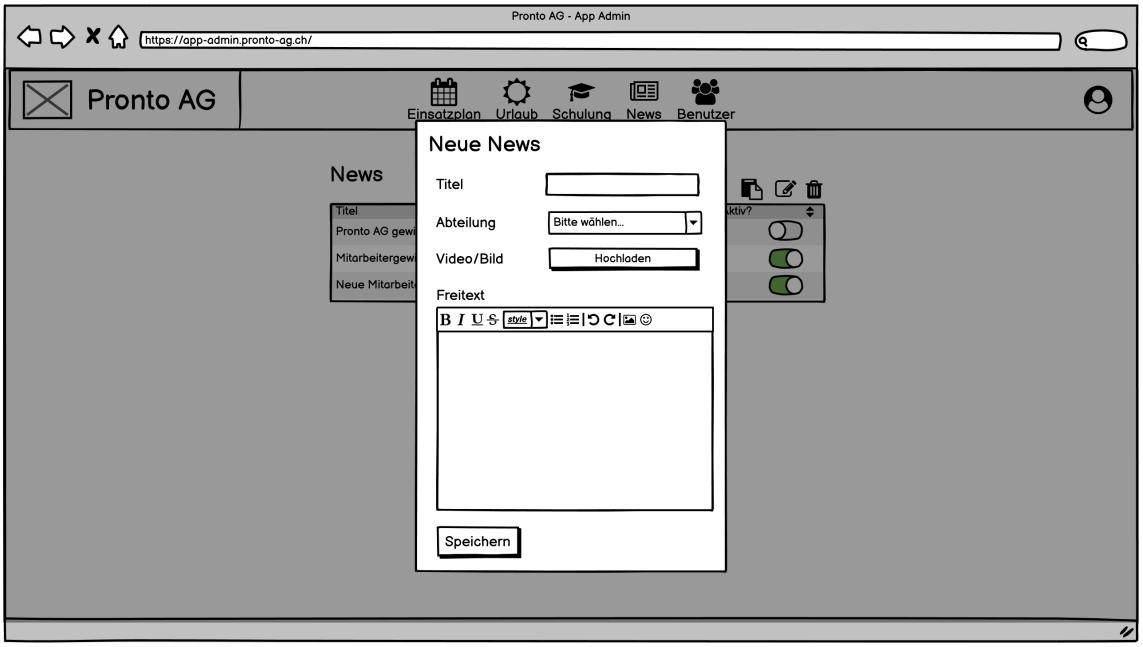

Abbildung G.31: Wireframe Newsartikel erstellen

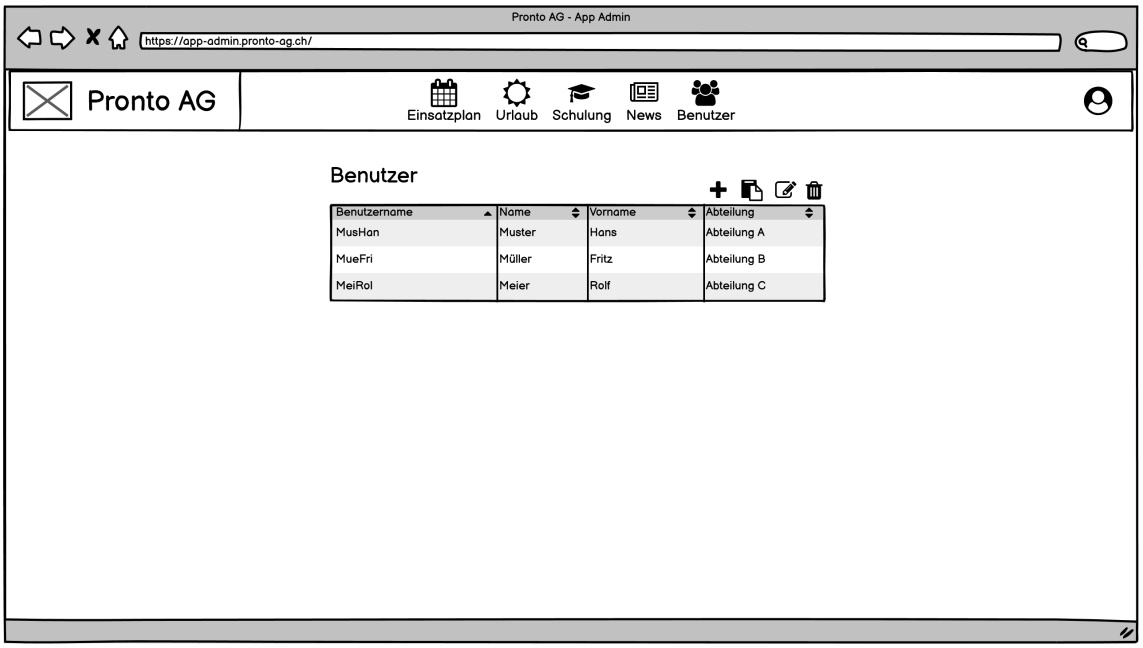

Abbildung G.32: Wireframe Benutzerverwaltung

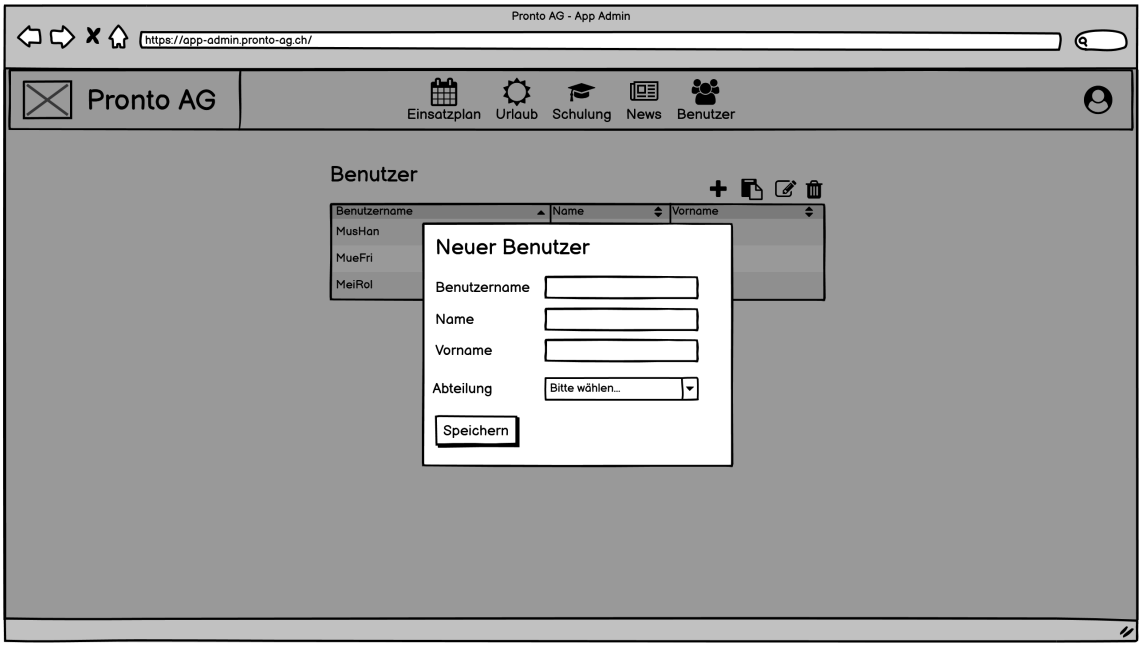

Abbildung G.33: Wireframe Benutzer erstellen

H Resultate Usability Test

Bei zus. Funktionen
besser deklarieren (wo Abmelden, wo PW-wechseln)

Einstellungsicon wäre besser
≡ → besser oben rechts als unten links

Name: Nina

Nummer	Beschreibung	Erwartetes Ergebnis	Bewertung (1-5)	Notizen
UT05	Die Funktionsweise der Oberfläche ist den Benutzern ohne Einführung klar	Die Benutzer finden sich zurecht.	4	PDF nach Seite scrollen → Alle Seiten beim Einsatzplan anzeigen oder Seitenanzahl anzeigen
UT06	Die Funktionsweise der Oberfläche ist ausreichend zur Nutzung der App	Die Benutzer sind zufrieden mit den bestehenden Funktionen und würden die App im Berufs-Alltag benutzen.	3	Passwort (neues) 2x eingeben Passwort-Wechsel "logisch" über Benutzerprofil
UT07	Die App ist einfach gehalten und man kommt mit wenigen Schritten zum Ziel.	Die Benutzer müssen keine überflüssigen Eingaben tätigen.	4	
UT08	Die App vereinfacht den Arbeits-Alltag	Die Benutzer haben durch die Nutzung der App einen Mehrwert.	3	
UT09	Fehler in der App werden korrekt dargestellt	Die Benutzer bekommen bei einem Fehler aussagekräftige Meldungen.	-	
UT10	Einsatzplan ist gut umgesetzt	Die Benutzer empfinden die Umsetzung des Einsatzplan-Features als angemessen und alle wichtigen Funktionen sind vorhanden.	4	

Reinigungskraft	Abteilungsleiter
Login Einsatzpläne anschauen eigenes Profil anschauen (allenfalls PW zurücksetzen)	Login Einsatzpläne in App aufschalten für bestimmte gruppe Einsatzpläne anschauen eigenes Profil anschauen (allenfalls PW zurücksetzen) neues Profil erstellen

Verbesserungsvorschläge/Notizen:

Aktuell Whatsapp-Gruppe → App besser

- Bei Fest/Grillabend (Events) anmelden/abmelden unterstützen
- Geburtstagsliste der MA → Mit Kalender oder eigener Funktion

Prio:
1. Schulungsvideos
2. Ferienanträge

Name: Selina

Nummer	Beschreibung	Erwartetes Ergebnis	Bewertung (1-5)	Notizen
UT05	Die Funktionsweise der Oberfläche ist den Benutzern ohne Einführung klar	Die Benutzer finden sich zurecht.	4	
UT06	Die Funktionsweise der Oberfläche ist ausreichend zur Nutzung der App	Die Benutzer sind zufrieden mit den bestehenden Funktionen und würden die App im Berufs-Alltag benutzen.	2	
UT07	Die App ist einfach gehalten und man kommt mit wenigen Schritten zum Ziel.	Die Benutzer müssen keine überflüssigen Eingaben tätigen.	4	Passwort ändern schwierig, ~~bei~~ navigation oben besser
UT08	Die App vereinfacht den Arbeits-Alltag	Die Benutzer haben durch die Nutzung der App einen Mehrwert.	3	Einsatzplan nicht viel vorteil Ferienantrag wäre wichtig
UT09	Fehler in der App werden korrekt dargestellt	Die Benutzer bekommen bei einem Fehler aussagekräftige Meldungen.	-	
UT10	Einsatzplan ist gut umgesetzt	Die Benutzer empfinden die Umsetzung des Einsatzplan-Features als angemessen und alle wichtigen Funktionen sind vorhanden.	5	

Reinigungskraft	Abteilungsleiter
Login Einsatzpläne anschauen eigenes Profil anschauen (allenfalls PW zurücksetzen)	Login Einsatzpläne in App aufschalten für bestimmte gruppe Einsatzpläne anschauen eigenes Profil anschauen (allenfalls PW zurücksetzen) neues Profil erstellen

Verbesserungsvorschläge/Notizen:

Passwort zweimal eintippen → Mein verbesserung

Auslogen schwierig

Einsatzplan gut , Navigation läuft gut ☺

Ablauf & Veröffentlichsdatum vorab abfüllen (Default Wert)
-> ab "jetzt" bis ca. 1 Monat
-> Zeit nicht nötig

Name: Heidi + Raman

Nummer	Beschreibung	Erwartetes Ergebnis	Bewertung (1-5)	Notizen
UT05	Die Funktionsweise der Oberfläche ist den Benutzern ohne Einführung klar	Die Benutzer finden sich zurecht.	5	
UT06	Die Funktionsweise der Oberfläche ist ausreichend zur Nutzung der App	Die Benutzer sind zufrieden mit den bestehenden Funktionen und würden die App im Berufs-Alltag benutzen.	4	
UT07	Die App ist einfach gehalten und man kommt mit wenigen Schritten zum Ziel.	Die Benutzer müssen keine überflüssigen Eingaben tätigen.	4	
UT08	Die App vereinfacht den Arbeits-Alltag	Die Benutzer haben durch die Nutzung der App einen Mehrwert.	4	
UT09	Fehler in der App werden korrekt dargestellt	Die Benutzer bekommen bei einem Fehler aussagekräftige Meldungen.	5	
UT10	Einsatzplan ist gut umgesetzt	Die Benutzer empfinden die Umsetzung des Einsatzplan-Features als angemessen und alle wichtigen Funktionen sind vorhanden.	4/5	

Reinigungskraft	Abteilungsleiter
Login Einsatzpläne anschauen eigenes Profil anschauen (allenfalls PW zurücksetzen)	Login Einsatzpläne in App aufschalten für bestimmte gruppe Einsatzpläne anschauen eigenes Profil anschauen (allenfalls PW zurücksetzen) neues Profil erstellen

Verbesserungsvorschläge/Notizen:

Einsatzplan löschen verbessern
-> mit Rechtsswipe (auf Mobile)
-> oder Rechtsklick oder auswählen + Delete Button rechts
-> mit zus. Nachfrage sehr gut

- Einsatzplan hochladen + veröffentlichen sehr gut
- allenfalls für Datum besser ersichtlich, dass DatePicker verfügbar
- Benutzerverwaltung Rechte besser angeben
- Abteilungen erstellen sehr gut
- Benutzerverwaltung mit Abteilungen sehr gut

Push-Benachrichtigung, wenn Einsatzplan aktualisiert wurde
→ damit MA benachrichtigt wird wenn es ein Update gab
→ Allenfalls Titel Fett markieren, wenn Plan noch nicht angesehen wurde resp. wieder auch Fett wenn es ein Update gab

MA mehreren Abteilungen zuweisen können
→ normale MA + Abteilungsleiter

PW wechseln → PW in Klartext anzeigen geht nicht, anmelden auch nicht

Benutzerverwaltung mit Suchfunktion oder Filtern

Pushbenachrichtigungen im App aktivieren/deaktivieren
→ für Ferien, Unfall, etc.

Einsatzplan in Querformat darstellen wenn Handy gedreht wird