

Beta-Release washly

Diethelm Mike (diethmik), Fuchs Patric (fupat002), Gallati Martina (gallamar), Kassem Ismail (kasseism), Kolattukudy Benssy (kolatben), Nambiar Raphael (nambirap)

Zürcher Hochschule
der angewandten
Wissenschaften,
15.05.2023

Abstract

This technical report presents the washly web application, a laundry management system designed to allow users to reserve washing machines and dryers from anywhere, at any time. The digital washing schedule offers a real-time overview of machine availability and occupancy, allowing users to coordinate their washing times effectively and avoid long waiting times. Users can easily plan and organize their laundry needs using the web app, which provides a convenient and efficient way to manage communal laundry rooms.

By adopting washly, administrators gain a streamlined and efficient solution for managing laundry facilities within their residential complexes. The application simplifies various administrative tasks, such as monitoring and scheduling laundry usage, tracking maintenance needs, and generating usage reports.

This report includes an overview of the code architecture and design of the washly web application. The backend of washly is built using Java and follows a layered architecture pattern. The use of JUnit for unit testing ensures high code quality and reliability. On the frontend, washly utilizes the Vue.js framework. The code is organized following a component-based architecture, allowing for reusability and modularity. The frontend prioritizes other forms of testing, such as end-to-end tests and manual testing.

Further developments for washly are outlined, including additional requirements and open points. These include features such as the integration of payment systems, notification enhancements, integration with smart devices, and improvements to the user interface.

Overall, this technical report provides a comprehensive overview of the implementation of the washly web application, its testing strategies, code architecture, code design, achieved goals, and potential areas for future enhancements. It serves to enhance the understanding of the development and functionality of washly.

Inhaltsverzeichnis

| | | |
|---------------------|---|-----------|
| 1 | Projektidee | 1 |
| 1.1 | Ausgangslage | 1 |
| 1.2 | Idee..... | 1 |
| 1.3 | Kundennutzen | 1 |
| 1.4 | Stand der Technik / Konkurrenzanalyse | 1 |
| 1.5 | Hauptablauf / Kontextszenario..... | 2 |
| 1.6 | Wirtschaftlichkeit..... | 3 |
| 2 | Analyse..... | 4 |
| 2.1 | Use-Case-Modell | 4 |
| 2.2 | Use-Case-Diagramm | 4 |
| 2.3 | Use-Cases | 5 |
| 2.4 | Zusätzliche Anforderungen..... | 10 |
| 2.5 | Domänenmodell | 11 |
| 3 | Design | 12 |
| 3.1 | Softwarearchitektur..... | 12 |
| 3.2 | Design-Klassendiagramm | 15 |
| 3.3 | Interaktionsdiagramme | 16 |
| 3.4 | Systemoperationen..... | 17 |
| 3.5 | Designentscheide | 21 |
| 3.6 | Wichtige Entscheidung GUI | 22 |
| 3.7 | Design Patterns..... | 23 |
| 3.8 | Authentifizierung..... | 25 |
| 4 | Implementation | 25 |
| 4.1 | Lieferergebnisse..... | 25 |
| 4.2 | Ausführbare Dateien | 26 |
| 4.3 | Automatisiertes Testing..... | 26 |
| 4.4 | Manuelles Testing..... | 27 |
| 5 | Resultate | 28 |
| 5.1 | Erreichte Ziele..... | 28 |
| 5.2 | Offene Punkte..... | 28 |
| 5.3 | Weiterentwicklungen | 28 |
| 6 | Selbstständigkeitserklärung..... | 29 |
| 7 | Quellenverzeichnis..... | 30 |
| 8 | Abbildungsverzeichnis..... | 31 |
| 9 | Tabellenverzeichnis | 31 |
| 10 | Glossar | 32 |
| Anhang | 34 | |
| A | Manuelle Testingresultate..... | 34 |

| | |
|-----------------------------|----|
| B Projektmanagement..... | 38 |
| B.1 Organigramm..... | 38 |
| B.2 Iterationsplanung | 38 |
| B.3 Projektrisiken..... | 39 |

1 Projektidee

In diesem Kapitel wird das Projekt washly vorgestellt.

1.1 Ausgangslage

Gemeinschaftswaschküchen sind ein weit verbreitetes Merkmal in Wohnanlagen, Wohnungen und Mehrfamilienhäusern. Sie bieten den Bewohner:innen eine Möglichkeit ihre Wäsche zu waschen und zu trocknen, ohne eine eigene Waschmaschine oder einen eigenen Trockner zu besitzen. Um die Nutzung von Gemeinschaftswaschküchen zu erleichtern und Konflikte zu vermeiden, werden häufig Waschpläne in Papierform erstellt. Die Bewohner:innen müssen diese Informationen manuell ablesen und ihre Waschzeiten entsprechend planen. Der Waschplan wird oft an einem zentralen Ort, wie zum Beispiel in der Waschküche, aufgehängt und kann nur vor Ort eingesehen werden. Ein Waschplan in Papierform bietet daher nur begrenzte Möglichkeiten zur effizienten Organisation des Waschvorgangs.

1.2 Idee

Die grundlegende Idee war es, eine Web-App zur Reservierung von Waschmaschinen / Trocknern in Gemeinschaftswaschküchen zu entwickeln. Mitbenutzende der Gemeinschaftswaschküche können jederzeit via Browser (Desktop- oder Mobilgerät) den gewünschten Tag und die gewünschte Uhrzeit auswählen und die Maschine für sich reservieren.

Ein digitaler Waschplan bietet den Bewohner:innen die Möglichkeit, ihre Wäsche bequem und effizient zu planen und zu organisieren. Durch die digitale Erfassung der Waschmaschinenbelegung und -verfügbarkeit können Bewohner:innen ihre Waschzeiten besser koordinieren und lange Wartezeiten vermeiden. Zudem ermöglicht ein digitaler Waschplan eine bessere Planung und Überwachung des Waschprozesses sowie eine einfachere Verwaltung von Maschinen. Dadurch sparen die Bewohner:innen Zeit und die Effizienz ihres Waschvorgangs wird erhöht.

1.3 Kundennutzen

Mit der Einführung von washly erhalten Administrator:innen eine optimierte und effiziente Lösung für die Verwaltung von Gemeinschaftswaschküchen in ihren Wohnanlagen. Die Web-App vereinfacht verschiedene Verwaltungsaufgaben, wie zum Beispiel die Überwachung und Planung der Waschkapazitäten, detaillierte Nutzungsstatistiken, schnelle Benachrichtigungen über Probleme und Zeiteinsparungen bei der Verwaltung des Waschplans.

Für Waschende ermöglicht die Organisation via washly eine jederzeitige Einsicht in den Waschplan, erleichtert spontanes, unkompliziertes Waschen und reduziert potenzielle Konflikte sowie Frustrationen, die bei der Verwendung eines konventionellen Waschplans in Papierform oder bei einem fehlenden Plan auftreten könnten.

1.4 Stand der Technik / Konkurrenzanalyse

Das Konzept eines digitalen Waschplans dieser Form existiert noch nicht als unabhängige, nutzerorientierte Lösung. Verschiedene Marken und Unternehmen haben bereits eine Anzahl von Lösungen veröffentlicht. Bekannte Beispiele hierfür sind die Folgenden:

1.4.1 WeWash von Bosch

Mit einer externen Box, entwickelt und produziert von Bosch, können Waschgeräte unabhängig von Marke oder Alter aufgerüstet werden. Diese Box ist bereits mit einer redundanten

Netzwerkanbindung ausgestattet. Zudem wird direkt über eine dazugehörige App bezahlt. Diese Lösung wurde in erster Linie im Interesse von Immobilienverwaltungen für die bargeldlose Abrechnung des Waschens der verschiedenen Bewohner:innen entwickelt. [1]

1.4.2 AppWash von Miele

Die App, entwickelt und betrieben von Miele, ist abhängig von Miele-Geräten mit Kassiersystem und ist, ähnlich wie die Lösung von Bosch, ausgelegt auf das kontaktlose Bezahlungssystem und weniger nutzerorientiert. [2]

1.4.3 Terresta Waschküchen-App

Die Lösung von Terresta, einer Immobilienverwaltung mit Sitz in Winterthur, ist nutzerorientierter als die anderen Anwendungen; ist allerdings nur Bewohner:innen einer Liegenschaft von Terresta zugänglich. Die Nutzerberichte der App zeigen eine negative Resonanz [3], [4]. [5]

Lösung – washly

Die washly Web-App hat keine Abhängigkeit zu den jeweiligen Geräten und kann daher «*stand alone*» genutzt werden, egal welches Waschgrossgerät im Einsatz ist. washly ist nutzerorientierter als vergleichbare Lösungen und legt den Fokus nicht auf das kontaktlose Bezahlen. Zudem wird keine externe Hardware benötigt.

1.5 Hauptablauf / Kontextszenario

In einer modernen Wohnanlage mit mehreren Wohneinheiten gibt es oft gemeinschaftliche Waschküchen, in der die Bewohner:innen ihre Wäsche waschen können. Die Nutzung und Verwaltung einer gemeinsamen Waschküche können jedoch zu Herausforderungen führen, wie unklare Zeitpläne, Konflikte um die Nutzung und mangelnde Transparenz. Diese Probleme werden im Kontextszenario mit den fiktiven Personen «Andreas Admin» und «Marta Mutter» dargestellt und verdeutlicht. Sie repräsentieren die Nutzergruppen und werden verwendet, um das Verständnis für die Bedürfnisse, Ziele und Verhaltensweisen der Benutzer zu vertiefen.

Andreas Admin arbeitet in der Immobilienbranche und verwaltet einige Wohnblöcke. Er möchte die Waschmaschinenverwaltung erleichtern und setzt deshalb bei seinen Immobilien washly ein. Er kennt die Altersgruppen der Bewohner:innen und kann somit gut einschätzen, ob sie mit washly zurechtkommen werden. Andreas Admin verwaltet somit auch die Gemeinschaftswaschküchen auf washly. Er kann Gemeinschaftswaschküchen erstellen und bearbeiten. Er hat Zugriff auf eine Übersicht über alle Waschmaschinen und kann Bewohner:innen hinzufügen oder entfernen.

Marta Mutter, eine alleinerziehende Mutter eines Kleinkindes, die in einer Wohnung im vierten Stock lebt, steht vor der Herausforderung, ihre Wäsche zu waschen, ohne ihr Kind unbeaufsichtigt zu lassen und ohne sich auf ungewisse Verfügbarkeiten von Waschmaschinen in der Gemeinschaftsküche verlassen zu müssen. Glücklicherweise kann sie auf washly zurückgreifen, das in ihrer Gemeinschaftswaschküche verwendet wird. Deshalb kann Marta Mutter sich auf washly mit ihrem Account anmelden und die Verfügbarkeit, der für sie zugänglichen Waschmaschinen einsehen und reservieren. Für eine Reservation muss sie nur das Zeitfenster angeben, in dem sie eine Waschmaschine belegt. Nach Ablauf des Zeitfensters wird die Waschmaschine automatisch freigegeben oder Marta Mutter kann die Maschine auch manuell vorzeitig freigegeben.

1.6 Wirtschaftlichkeit

Der geschätzte Aufwand (Personalaufwand rein für die Entwicklung) liegt bei total 220 Stunden mit einem Stundenansatz von 110 CHF, dies ergibt somit eine Summe von 24'000 CHF.

Der Vertrieb der Web-App soll durch ein klassisches *SaaS*-Abo-Modell [6] realisiert werden. Hierbei liegt der Kostenpunkt auf der einzelnen Waschmaschine/dem einzelnen Trockner. Preisvorschlag ist hier bei *5 CHF pro Maschine im Monat*. Eine durchschnittliche Gemeinschaftswaschküche besitzt schätzungsweise 5 Maschinen. Dies resultiert in einem Kostenpunkt von 300 CHF pro Jahr pro Waschküche.

Eine erste Markteinschätzung lässt auf 40 User im ersten Jahr schliessen. Es ist mit einem jährlichen Wachstum von 20% zu rechnen. Die Einnahmen sind in Tabelle 1 ersichtlich.

Tabelle 1: Markteinschätzung der Einnahmen

| Jahr | Anzahl Kund:innen | Jährliche Einnahmen |
|------|-------------------|---------------------|
| 1 | 40 | 12'000 CHF |
| 2 | 48 | 14'400 CHF |
| 3 | 58 | 17'400 CHF |
| 4 | 70 | 21'000 CHF |
| 5 | 83 | 24'900 CHF |

Nach dem zweiten Jahr wird das *Return on Investment* erreicht. Nach 5 Jahren ist ein Gewinn von 65'700 CHF zu erwarten.

2 Analyse

In folgender Analyse werden Teile der Architekturentscheide genauer erläutert.

2.1 Use-Case-Modell

Die folgende Aufzählung in Tabelle 2 beschreibt die verschiedenen Funktionen, die von den zwei unterschiedenen Nutzergruppen, Waschenden und Administrator:innen, genutzt werden können. Waschende sind hierbei Bewohner:innen, die die Waschküche als solches nutzen, Administrator:innen sind alle verwaltende Instanzen der Waschküchen.

Tabelle 2: Auflistung der Use-Cases pro Akteur:in

| Waschende | Administrator:innen |
|---|--|
| <ul style="list-style-type: none">- Waschtermin buchen- Waschplan einsehen | <ul style="list-style-type: none">- Maschine erfassen- Maschine bearbeiten / sperren- Waschende einladen / entfernen |

2.2 Use-Case-Diagramm

In Abbildung 1 ist die Visualisierung der Nutzungsmöglichkeiten der Akteur:innen innerhalb von washly aufgezeigt.

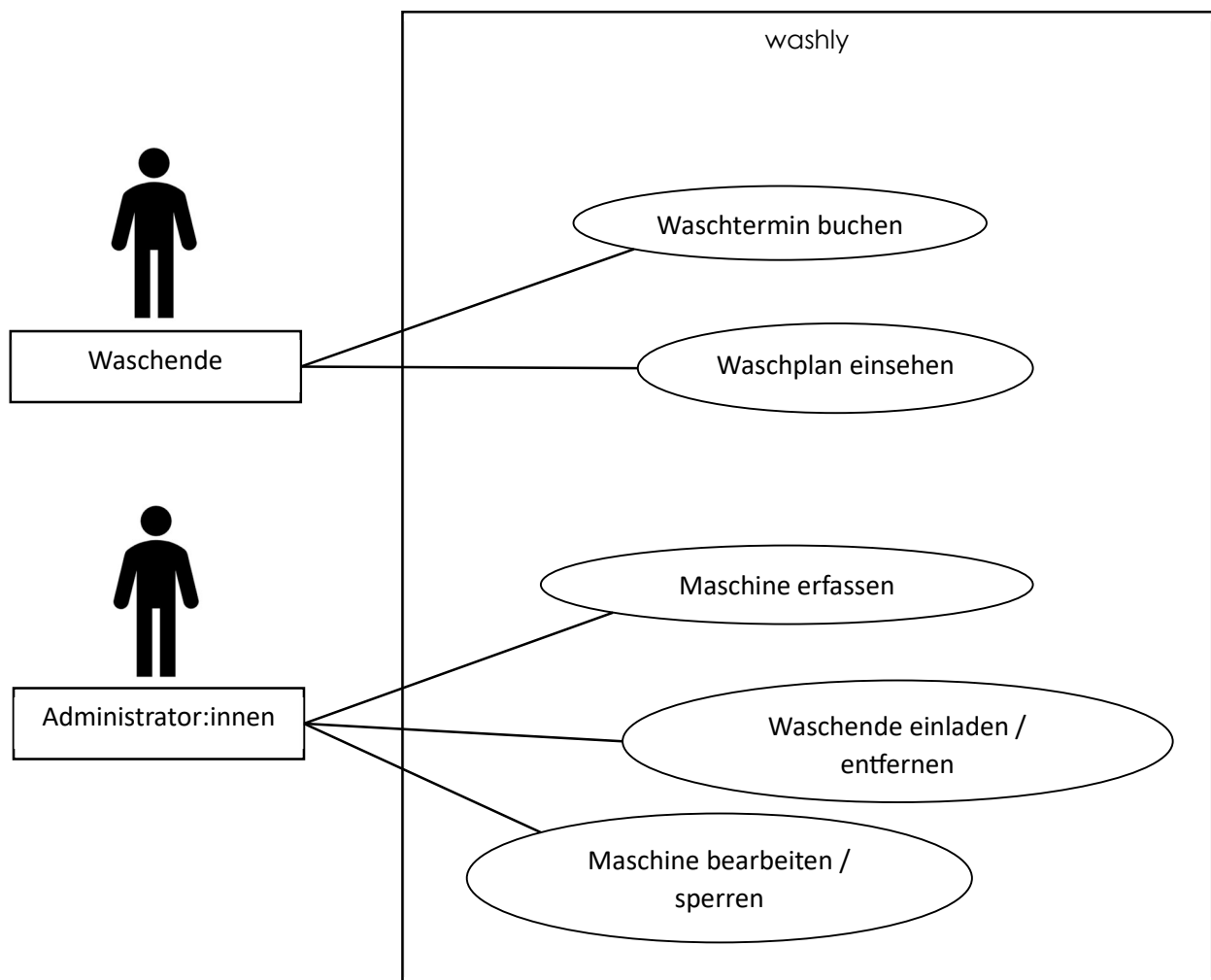


Abbildung 1: Use-Case-Diagramm

2.3 Use-Cases

In diesem Kapitel werden die verschiedenen Use-Cases der Applikation in unterschiedlicher Ausführlichkeit beschrieben.

2.3.1 Maschine erfassen

Dieser Use-Case, ersichtlich in Abbildung 2, zeigt die Erfassung einer Maschine aus der Verwaltungsperspektive. Eine Maschine kann eine Waschmaschine oder ein Trockner sein.

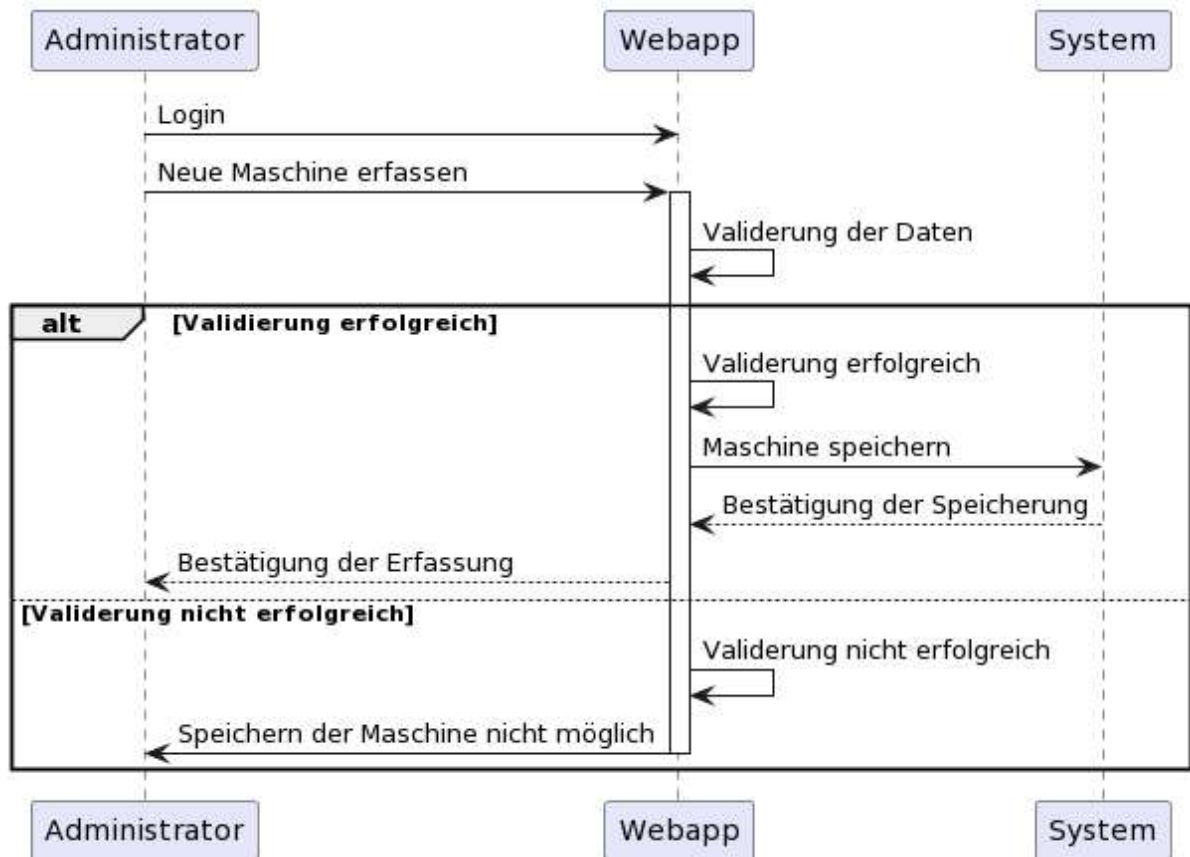


Abbildung 2: Use-Case «Maschine erfassen»

Umfang

Erfassen einer Waschmaschine/eines Trockners im digitalen Waschplan

Ebene

Administrator

Primärakteur:innen

Administrator:innen der Gemeinschaftswaschküchen der Wohnanlagen

Stakeholders und Interessen

Administrator:innen

- Wollen eine neue Maschine in washly einfügen, um den Waschenden die Möglichkeit zu geben, diese zu reservieren

Waschende

- Wollen die hinzugefügte Maschine reservieren

Vorbedingungen

- Administrator:in hat Zugang zu washly
- Administrator:in verfügt über ein washly-Account mit administrativen Berechtigungen

Erfolgsgarantie/Nachbedingungen

Die neu hinzugefügte Maschine kann über washly von den Waschenden reserviert werden.

Standardablauf

1. Administrator:in loggt sich in washly ein
2. Administrator:in navigiert zur Seite für die Erfassung einer neuen Maschine
3. Administrator:in gibt die folgenden Informationen für die neue Maschine ein:
 - Name der Maschine
 - Typ der Maschine (Waschmaschine/Trockner)
 - Status (Aktiv/Inaktiv)
4. Administrator:in bestätigt die Eingabe der Informationen
5. washly zeigt eine Bestätigung an, dass die neue Maschine erfolgreich erfasst wurde

Erweiterungen

4a: Administrator:in gibt fehlende oder unvollständige Informationen ein.

1. washly zeigt dem/der Administrator:in eine Fehlermeldung an, dass fehlende oder unvollständige Informationen eingegeben wurden
2. washly zeigt die betroffenen Felder an, die vervollständigt oder korrigiert werden müssen
3. Administrator:in gibt die fehlenden oder korrigierten Informationen ein
4. Administrator:in bestätigt die Eingabe oder korrigierten Informationen
5. washly zeigt eine Bestätigung an, dass die Informationen erfolgreich vervollständigt oder korrigiert wurden

Spezielle Anforderungen

- Der/die Administrator:in muss über ein Gerät mit Internetzugang verfügen, um sich auf der washly-Web App einzuloggen
- Möglicherweise müssen bestimmte Details wie zum Beispiel Wartungshinweise erfasst werden, um die Verwaltung und Wartung zu erleichtern

Liste der Technik und Datavariationen

Wird über die Web-App erfasst.

Häufigkeit des Auftretens

Je nachdem, wie oft neue Maschinen installiert oder alte ausgetauscht werden müssen, kann dies häufig oder selten auftreten.

Verschiedenes

Eine Funktion zur Setzung des Gerätestatus auf «inaktiv», falls ein Gerät defekt ist oder bei Wartungsarbeiten sowie eine zusätzlicher Nachrichtenfeed, um die Waschenden über den Status des entsprechenden Geräts zu informieren und sie zu benachrichtigen.

2.3.2 Waschtermin buchen

Dieser Use-Case, ersichtlich in Abbildung 3, zeigt die Buchung eines Waschtermins aus der Userperspektive.

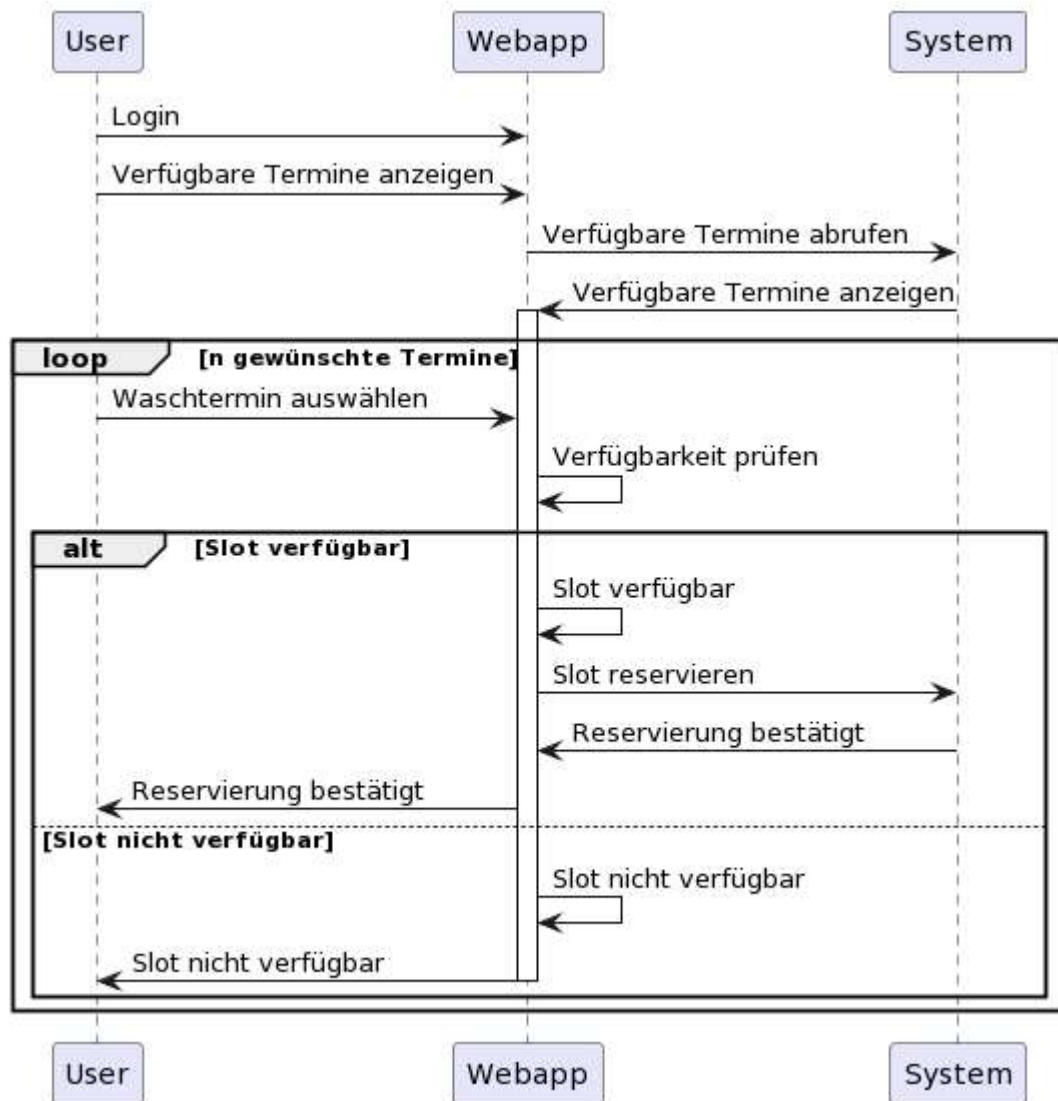


Abbildung 3: Use-Case «Waschtermin buchen»

Umfang

Reservierung eines Waschtermins im digitalen Waschplan.

Ebene

Anwenderziel

Primärakteur:innen

Waschende der Gemeinschaftswaschküchen.

Stakeholders und Interessen

Administrator:innen

- Wollen einfache und direkte Verwaltung (Zeitersparnis)
- Wollen keine Probleme in der Waschküche (App soll Problemen vorbeugen)

- Wollen Prozesse optimieren und Kosten einsparen

Waschende

- Wollen direkt und unkompliziert Waschtermin buchen
- Wollen jederzeit von überall Waschplan einsehen
- Wollen wissen, wann die Waschküche am meisten ausgelastet ist

Vorbedingungen

- Waschende müssen einen Account besitzen und einer Waschküche mit den dazu gehörigen Geräten zugewiesen sein
- Waschende müssen mit ihrem Account in der Web-App angemeldet sein

Erfolgsgarantie/Nachbedingungen

Erfolgreiche Buchung eines Waschsots ohne Überschneidungen mit anderen Slots. Der gegebene Zeitslot wird für andere Waschende als besetzt angezeigt.

Standardablauf (Happy Path)

1. Waschende melden sich in washly an
2. washly zeigt den Waschplan an
3. Waschende wählen einen freien Waschsot im Waschplan aus
4. Waschende überprüfen Datum, Zeit und Maschine des Waschsots
5. Waschende bestätigen den gewählten Slot
6. Das System zeigt eine Bestätigung der Reservierung an

Erweiterungen

5a: Falls die gewünschte Zeit bereits von anderen Usern reserviert wurde, wird eine Fehlermeldung angezeigt.

1. Das System prüft, ob der ausgewählte Waschtermin bereits von anderen Usern reserviert wurde
2. Falls der Waschtermin bereits reserviert ist, wird eine Fehlermeldung angezeigt, dass der gewählte Waschsot bereits belegt ist
3. Waschende können nach der Fehlermeldung wieder auf die Seite mit der Waschplanübersicht zurückkehren

Spezielle Anforderungen

- Das System muss sicherstellen, dass Waschende keine doppelten Reservierungen vornehmen können
- Das System muss sicherstellen, dass Waschende nur Maschinen reservieren können, die zu ihrer Gemeinschaftsküche gehören

Liste der Technik und Datavariationen

Buchung über verschiedene Systeme; Mobilgerät oder Desktop

Häufigkeit des Auftretens

Zwei bis vier Mal pro Waschende pro Monat.

Verschiedenes

- Einführen eines Benachrichtigungssystems, um die Waschenden entsprechend über den Waschstatus zu informieren
- Als mögliche Ergänzung für Waschende, die keinen Zugang zu einem Computer oder Mobilgerät haben oder nicht über ausreichende Kenntnisse mit digitalen Geräten verfügen, können beispielsweise wiederkehrende Reservierungen für einen bestimmten Tag und eine bestimmte Zeit ermöglicht werden

2.3.3 Waschmaschine bearbeiten

Der/die Administrator:in navigiert zu der Waschgeräteansicht in washly und wählt das zu bearbeitende Gerät aus. Das Waschgerät kann zu denselben Punkten bearbeitet werden, die bereits bei der Erfassung, näher erläutert in Kapitel 2.3.1 Maschine erfassen, gespeichert wurden, nämlich:

- Name der Maschine
- Typ der Maschine (Waschmaschine/Trockner)
- Status (Aktiv/Inaktiv)

Nach Bearbeitung der Daten speichert der/die Administrator:in die Änderungen. Das Waschgerät wird aktualisiert und die neuen Informationen werden für die Administrator:innen sowie für die User dargestellt.

2.3.4 User einladen

In der Benutzerverwaltungsansicht kann der/die Administrator:in für eine bestimmte Waschküche User mittels eines geteilten Links einladen. Die eingeladenen User nutzen diesen Link, um sich bei washly zu registrieren oder anzumelden und sich in der gegebenen Waschküche hinzuzufügen.

Der/die Administrator:in kann die eingeladenen und registrierten User in der Benutzerverwaltungsansicht einsehen. Nach Eintragung in der Waschküche kann der eingeladene User washly nutzen.

2.3.5 User entfernen

In der Benutzerverwaltungsansicht der Administrator:innen kann der zu entfernende User ausgewählt und gelöscht werden. Der entfernte User kann danach nicht mehr auf washly zugreifen.

2.3.6 Waschplan einsehen

Der User loggt sich in washly ein und navigiert zum Waschplan. Die verschiedenen Zeitfenster werden hier als frei oder besetzt angezeigt. Der User sieht verfügbare und besetzte Time-Slots und nutzt diese Informationen, um den nächsten Waschtermin zu planen oder bereits gebuchte, eigene Waschtermine einzusehen.

2.4 Zusätzliche Anforderungen

Neben den in Kapitel 2.3 Use-Cases definierten Anforderungen an die verschiedenen Use-Cases, sind in Tabelle 3 zusätzliche funktionale Anforderungen, und in Tabelle 4 zusätzliche nicht-funktionale Anforderungen erläutert; diese wurden auf die essenziellen beschränkt, um im Rahmen des Projektes zu bleiben.

Tabelle 3: Zusätzliche funktionale Anforderungen

| Funktionale Anforderung | Beschreibung |
|--|--|
| Benachrichtigungen | Benachrichtigungen per E-Mail oder Push-Benachrichtigungen auf dem Smartphone, um die Waschenden an bevorstehende Waschtermine zu erinnern oder über Änderungen im Waschplan zu informieren. |
| Bewertungssystem | Eine Möglichkeit für die Waschenden, ihre Wascherfahrungen zu bewerten und Feedback zu geben, um die Qualität der Waschmaschinen und den Service insgesamt zu verbessern. |
| Integration von Zahlungsdienstleistern | Eine Integration von Zahlungsdienstleistern wie PayPal oder Kreditkarten, um eine einfache und sichere Bezahlung zur Nutzung des Waschplans zu ermöglichen. |
| Waschstatistiken | Der Admin kann Statistiken zur Benutzung der Waschküche und den Maschinen einsehen. Durch die Einsicht der Waschstatistiken kann der Unterhalt der Waschküche optimiert werden. |

Tabelle 4: Zusätzliche nicht-funktionale Anforderungen

| Nicht-funktionale Anforderung | Beschreibung |
|-------------------------------|---|
| Mehrsprachigkeit | Die Möglichkeit, die App in verschiedenen Sprachen anzubieten, um Waschende mit unterschiedlichen Sprachkenntnissen zu erreichen. |
| Barrierefreiheit | Eine barrierefreie Gestaltung der App, um Menschen mit Behinderungen und Einschränkungen den Zugang und die Nutzung der App zu erleichtern. |
| Responsives Design | Die App soll auf verschiedenen Geräten und Bildschirmgrößen optimal dargestellt werden, um eine gute Benutzererfahrung zu gewährleisten. |
| Datenschutz | Ein besonderes Augenmerk auf den Datenschutz, um die Privatsphäre der User zu schützen und die Einhaltung der geltenden Datenschutzgesetze sicherzustellen. |

2.5 Domänenmodell

Im Domänenmodell werden die wichtigsten Entitäten, Attribute und Beziehungen durch ein konzeptuelles UML-Klassendiagramm dargestellt, um ein mentales Modell der Zusammenhänge zu detaillieren. Das nachfolgende Domänenmodell (Abbildung 4) ist für den MVP im Rahmen des PM3-Projekts.

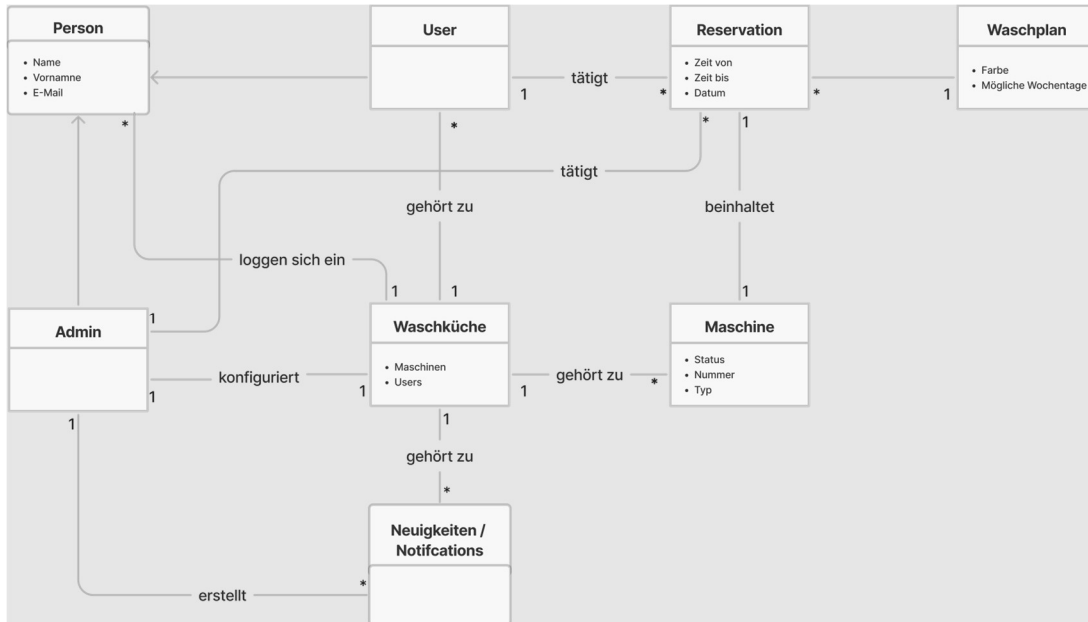


Abbildung 4: Domänenmodell MVP

Das Domänenmodell in Abbildung 5 wurde für das marktfertige Produkt entworfen.

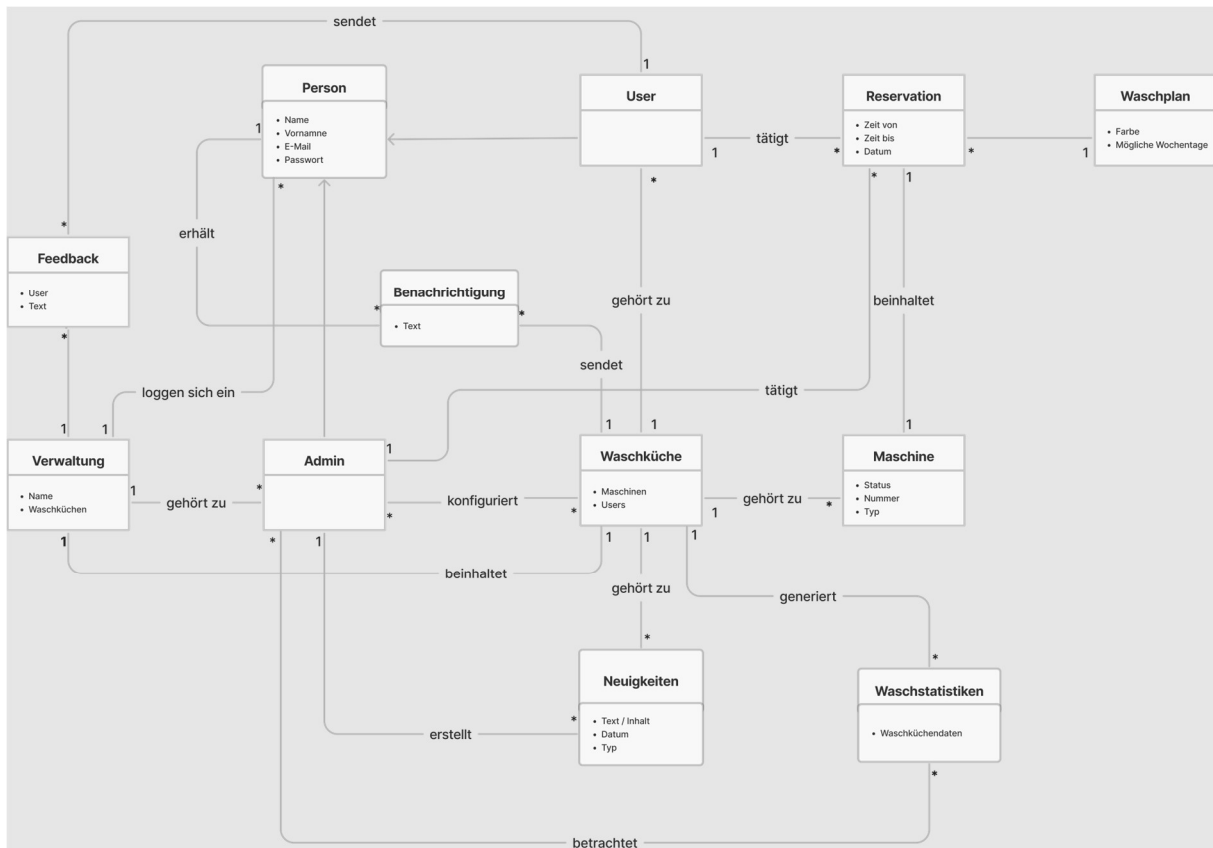


Abbildung 5: Domänenmodell für marktfertiges Produkt

3 Design

Dieses Kapitel zeigt einen umfassenden Überblick über das Design der washly-Software und dokumentiert die angewendeten Prinzipien und Design-Patterns.

3.1 Softwarearchitektur

Im Rahmen des Projekts washly wurde eine umfassende Softwarearchitektur, bestehend aus einem Java-Backend und einem Web-Frontend, entwickelt.

Das Ziel des architektonischen Ansatzes war es, eine modulare und erweiterbare Softwarearchitektur zu schaffen, welche die Online-Buchung und -Verwaltung von Waschräumen ermöglicht.

washly besteht aus einem Java-Backend, das die Geschäftslogik enthält, und einem Web-Frontend, mit welchem ein User Reservierungen vornehmen oder administrative Aufgaben erledigen kann.

3.1.1 Backend

Das Backend besteht aus mehreren Modulen, welche in Tabelle 5 aufgelistet sind und näher erläutert werden.

Tabelle 5: Backend-Module

| Modul | Beschreibung |
|------------------------|---|
| Reservations-Service | Verwaltet Buchungen und Aufträge. User können Buchungen erstellen, ändern und stornieren. |
| User-Service | Verwaltet User. Der Service ermöglicht es, User zu verwalten. |
| Waschmaschinen-Service | Verwaltet Waschmaschinen. Der Service ermöglicht es Waschmaschinen zu verwalten. |
| News-Feed-Service | Zeigt dem User diverse, von Administrator:innen erfasste, Informationen an. |

Das Backend verarbeitet Anfragen des Frontends über RESTful-APIs. Dies ist in Abbildung 6 näher ersichtlich.

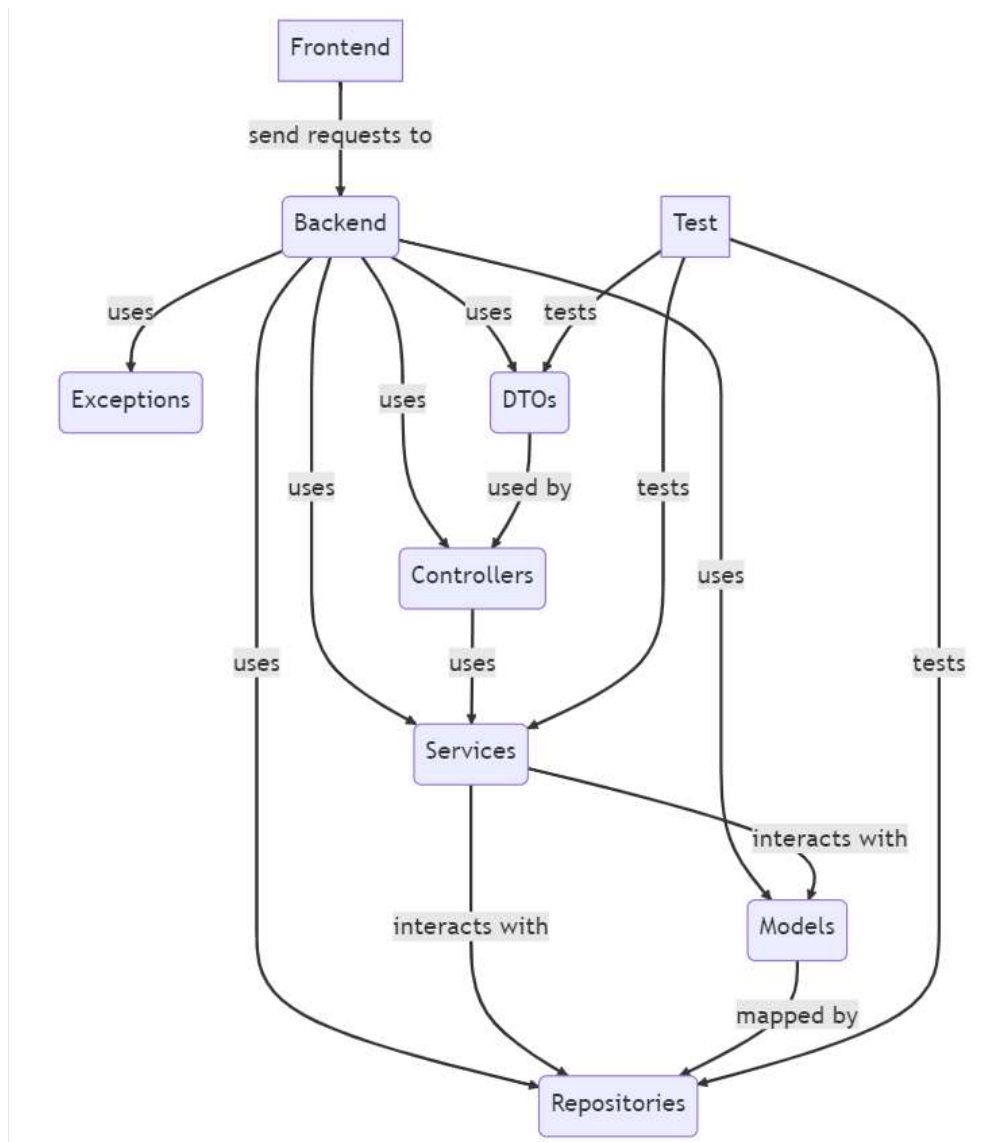


Abbildung 6: Systemübersichtsdiagramm

3.1.2 Frontend

Das Frontend ist durch folgende, in Tabelle 6 beschriebene Struktur möglichst modular aufgebaut:

Tabelle 6: Frontend-Module

| Modul | Beschreibung |
|-------------|--|
| Assets | Beinhaltet alle statischen Media-Inhalte wie Bilder, Videos, etc. |
| Components | Alle isolierten, modularen Vue-Komponenten |
| Composables | Enthält alle Composables |
| Forms | Die Formulare, welche zur Validierung und Übermittlung benötigt werden |
| Router | Standardordner für den Vue-Router |
| Views | Beinhaltet alle Views, welche als eigenständige Seite funktionieren |

3.1.3 Datenbank

Die Datenbank, dargestellt in Abbildung 7, ist einfach aufgebaut. Sie besitzt User, Waschmaschinen und Reservationen, wobei eine Reservation User und Waschmaschinen beinhaltet. Daneben gibt es auch Notifikationen, welche von Administrator:innen erstellt werden können. Dieses Modell deckt die Datenbank, die im Rahmen des PM3-Projektes aufgesetzt wurde, ab.

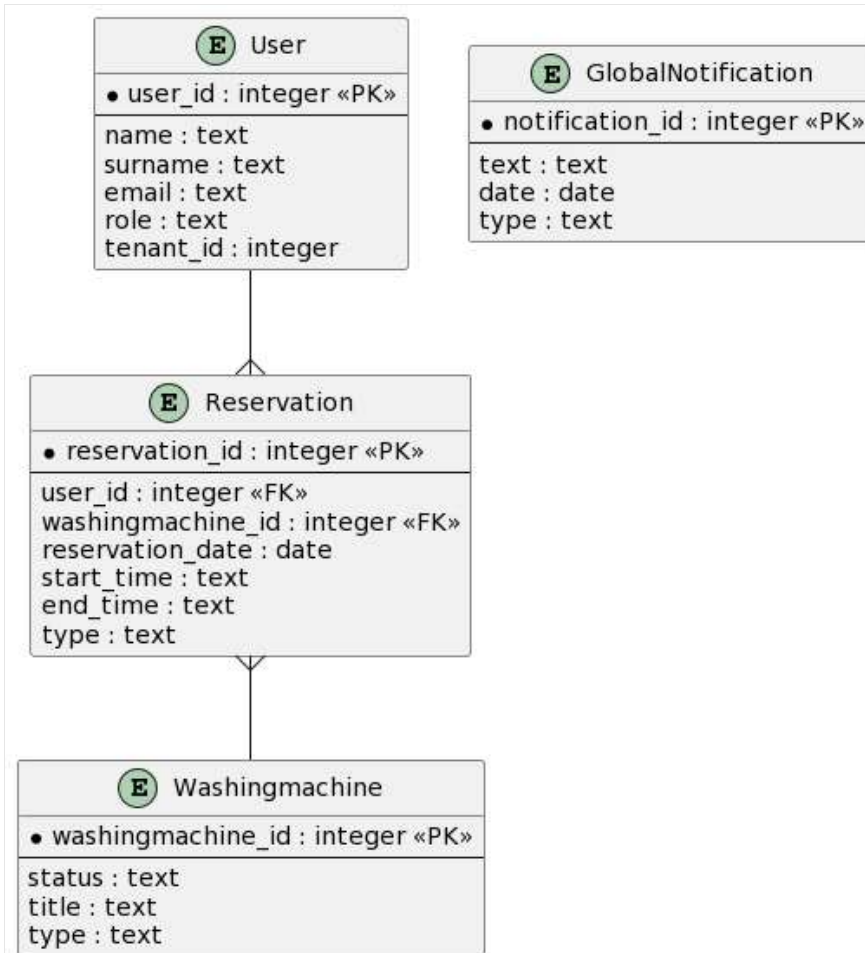


Abbildung 7: Datenbankmodell

3.2 Design-Klassendiagramm

Das Klassendiagramm in Abbildung 8 zeigt die implementierte Modularisierung auf.

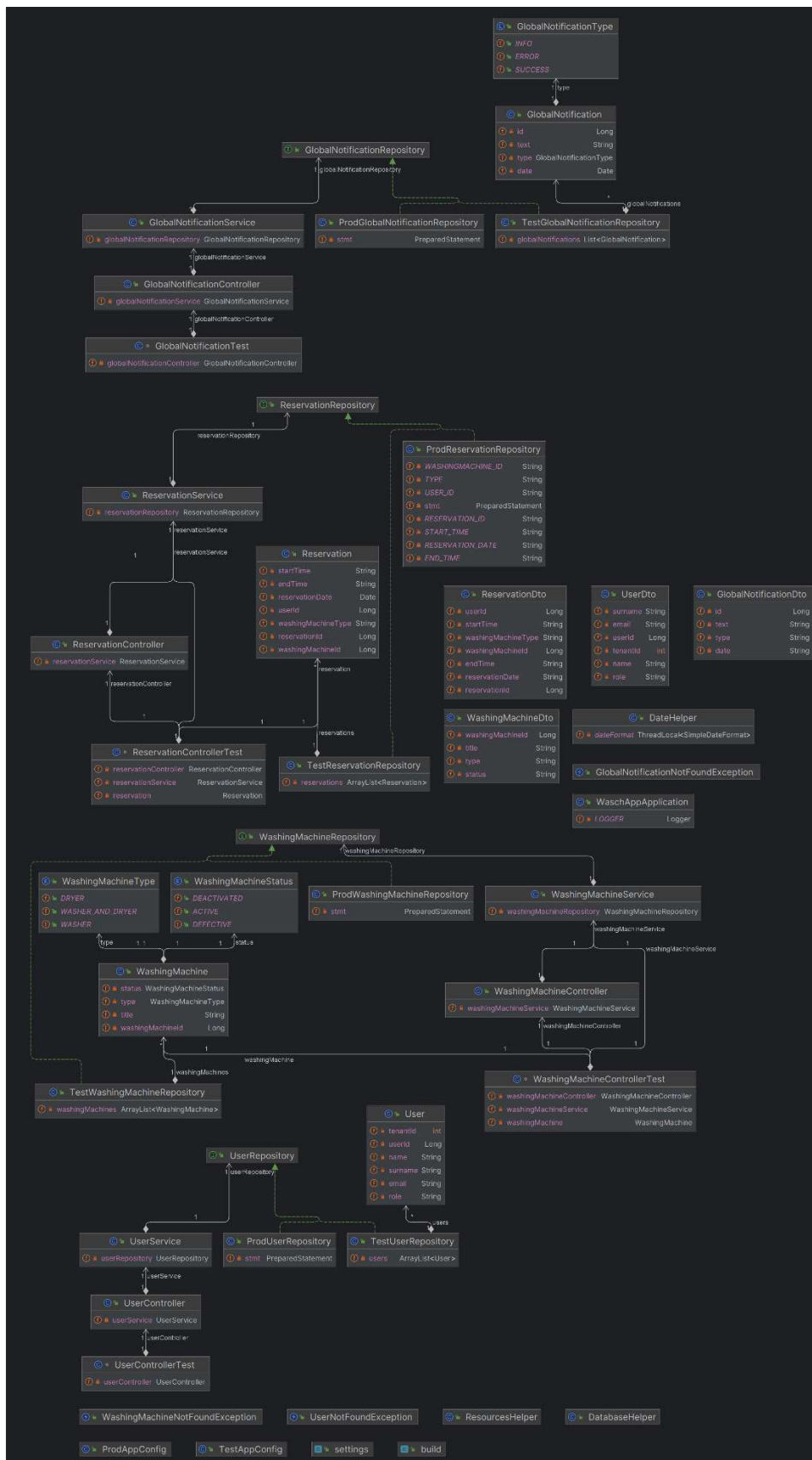


Abbildung 8: Klassendiagramm

3.3 Interaktionsdiagramme

Die Interaktion der User mit der App wird durch das Sequenzdiagramm in Abbildung 9 veranschaulicht, indem es zeigt, welche Akteur:innen beteiligt sind und welche Nachrichten zwischen ihnen ausgetauscht werden. Die Interaktion beginnt mit der Anmeldung des Users und endet mit der Abmeldung. Die einzelnen Schritte sind in Tabelle 7 notiert.

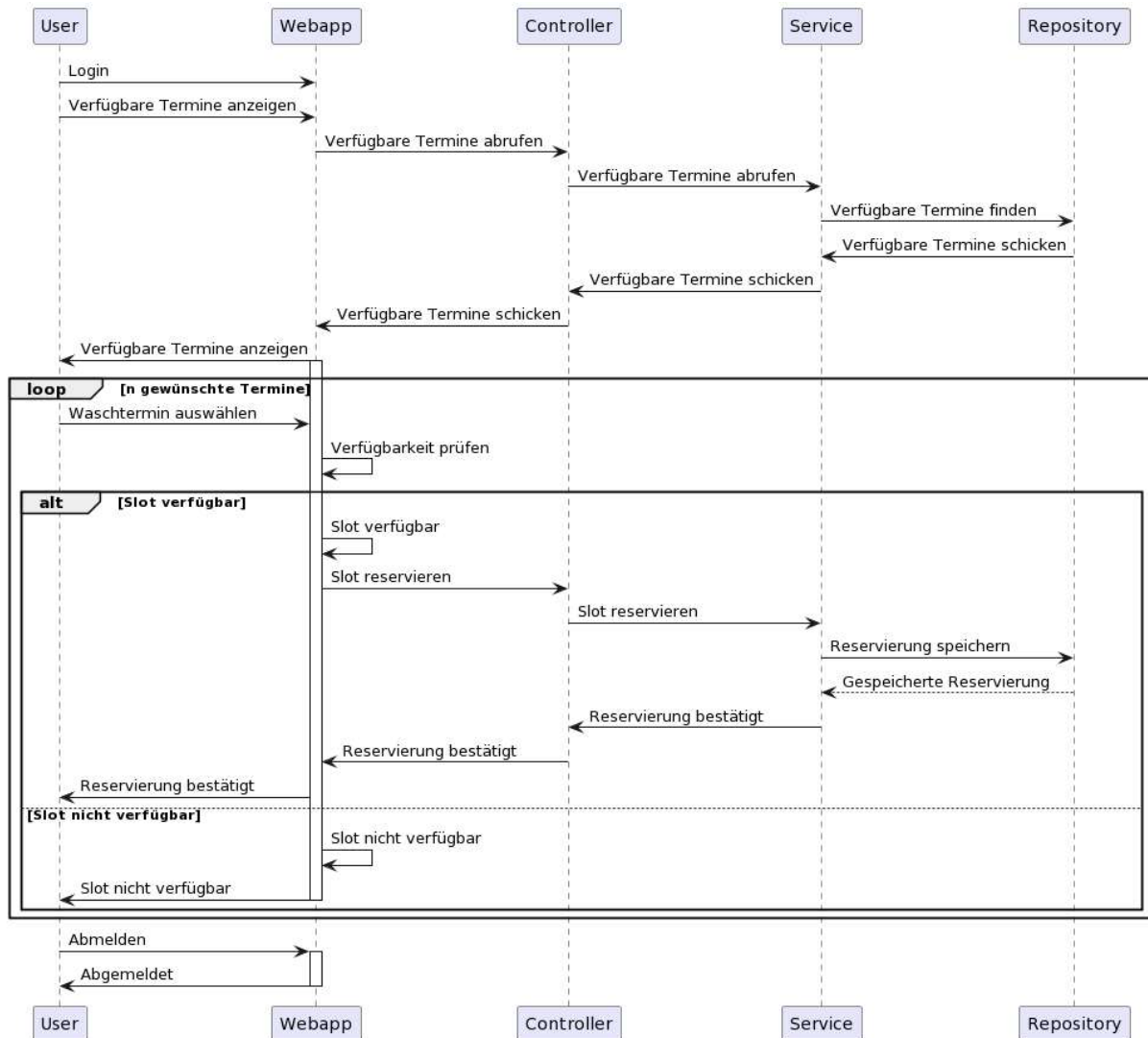


Abbildung 9: Datenflussdiagramm

Tabelle 7: Datenflussschritte

| Nr. | Schritt |
|-----|---|
| 1. | Der User loggt sich in die Web-App ein. |
| 2. | Der User fordert die Anzeige der verfügbaren Termine an. |
| 3. | Die Web-App ruft über den Controller die verfügbaren Termine ab. |
| 4. | Der Controller ruft über den Service die verfügbaren Termine ab. |
| 5. | Der Service sucht über das Repository nach den verfügbaren Terminen. |
| 6. | Das Repository schickt die gefundenen, verfügbaren Termine zurück an den Service. |
| 7. | Der Service schickt die verfügbaren Termine zurück an den Controller. |
| 8. | Der Controller schickt die verfügbaren Termine zurück an die Web-App. |

| Nr. | Schritt |
|-----|---|
| 9. | Die Web-App zeigt dem User die verfügbaren Termine an. |
| 10. | Der User wählt einen Waschtermin aus. |
| 11. | Die Web-App prüft die Verfügbarkeit des ausgewählten Termins. |
| 12. | Falls der Termin verfügbar ist, reserviert die Web-App den Slot. |
| 13. | Der Controller reserviert über den Service den ausgewählten Slot. |
| 14. | Der Service speichert die Reservierung im Repository. |
| 15. | Das Repository schickt die gespeicherte Reservierung zurück an den Service. |
| 16. | Der Service bestätigt dem Controller die Reservierung. |
| 17. | Der Controller bestätigt die Reservierung gegenüber der Web-App. |
| 18. | Die Web-App bestätigt die Reservierung gegenüber dem User. |
| 19. | Der User meldet sich von der Web-App ab. |
| 20. | Die Web-App bestätigt die Abmeldung des Users. |

3.4 Systemoperationen

In diesem Kapitel werden die wichtigsten Systemoperationen des washly-Systems genauer betrachtet und durch Sequenzdiagramme dargestellt.

3.4.1 Authentifizierung

Das Sequenzdiagramm in Abbildung 10 visualisiert die Schritte, die ein User ausführen kann, um sich bei washly zu authentifizieren, sowie die Interaktionen mit dem System, um die E-Mail-Adresse und das Passwort des Users zu validieren und ihm Zugang zum Waschplan-System zu gewähren. Die Schritte sind in Tabelle 8 genauer erläutert.

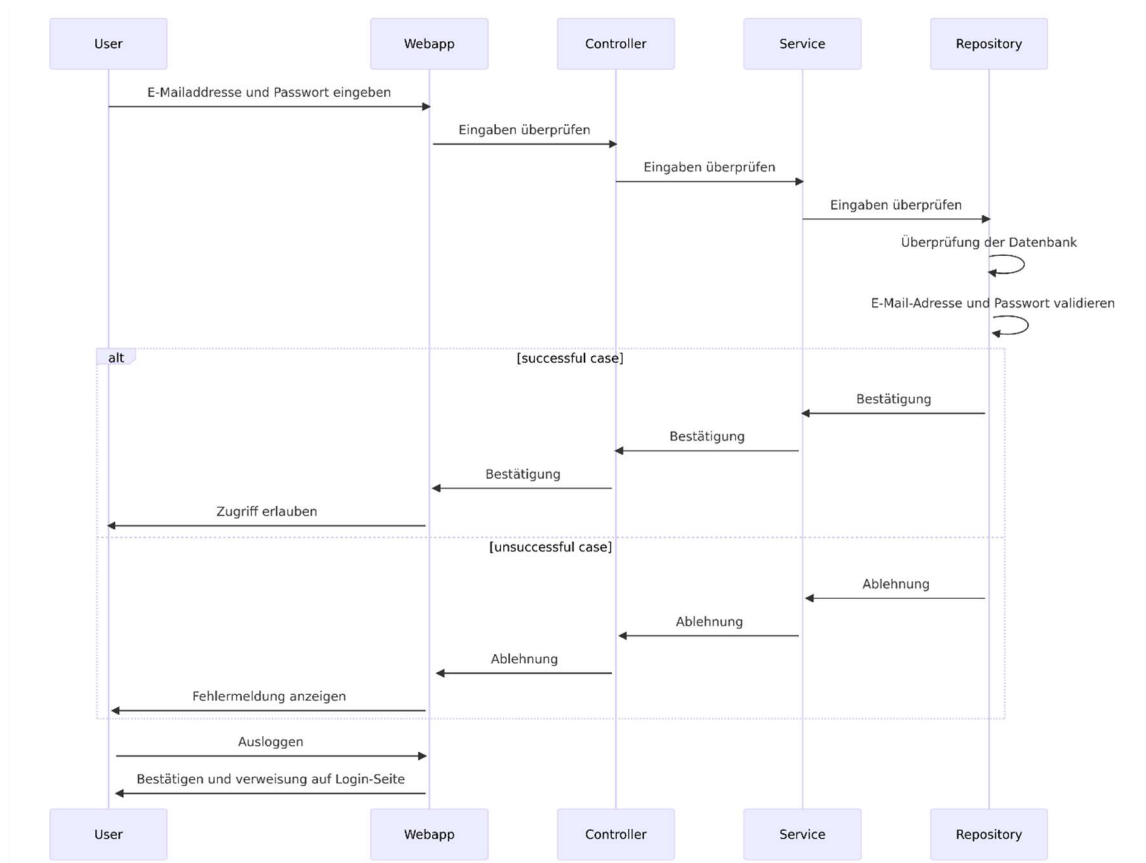


Abbildung 10: Sequenzdiagramm Authentifizierung

Tabelle 8: Authentifizierungsschritte

| Nr. | Schritt |
|-----|--|
| 1. | Das System zeigt dem User die Login-Seite an. |
| 2. | Der User gibt seine E-Mail-Adresse und sein Passwort ein. |
| 3. | Das System überprüft die Eingaben des Users auf Korrektheit und Validität. |
| 4. | Das System validiert die E-Mail-Adresse und das Passwort des Users gegen die in der Datenbank gespeicherten Informationen. Im Rahmen des PM3-Projektes geschieht dies auf der Web-App. Das Passwort ist hartkodiert («123») und dient rein der Veranschaulichung. |
| 5. | Wenn die E-Mail-Adresse und das Passwort korrekt sind, gibt das System dem User Zugriff auf das Waschplan-System. |
| 6. | Wenn die E-Mail-Adresse und das Passwort nicht korrekt sind, zeigt das System dem User eine Fehlermeldung an und gibt ihm die Möglichkeit, es erneut zu versuchen. |
| 7. | Der User drückt auf seinem Profil «Abmelden». |
| 8. | Das System loggt den User aus und zeigt ihm die Login-Seite erneut an. |

3.4.2 Reservierung

Das Reservierungs-Sequenzdiagramm in Abbildung 11 mit den Schritten in Tabelle 9 zeigt den Ablauf einer Reservierung einer Waschmaschine oder eines Trockners.

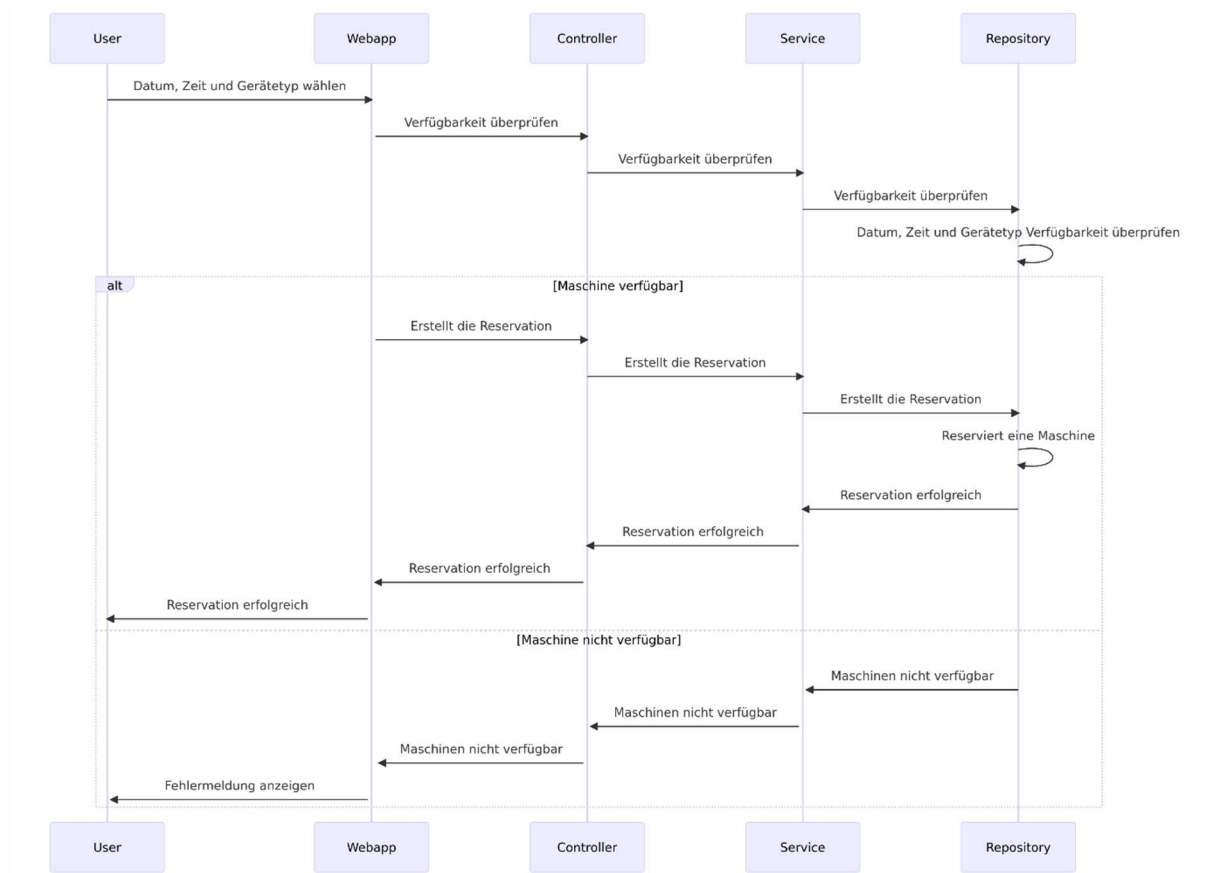


Abbildung 11: Sequenzdiagramm Reservierung

Tabelle 9: Reservierungsschritte

| Nr. | Schritt |
|-----|--|
| 1. | Der User möchte eine Maschine zu einem bestimmten Zeitpunkt reservieren. |
| 2. | Das System prüft die Verfügbarkeit. |
| 3. | Wenn eine Maschine frei ist, erstellt das System die Reservation. |
| 4. | Der User wird über die erfolgreiche Reservation informiert. |
| 5. | Wenn keine Maschine frei ist, kann die Reservation nicht durchgeführt werden und der User wird mit einer Fehlermeldung benachrichtigt. |

3.4.3 Belegungsüberwachung

Das Sequenzdiagramm (Abbildung 12) zur Belegungsüberwachung zeigt den Ablauf, wie überprüft wird wie viele Maschinen zu einem bestimmten Zeitpunkt frei sind. Die Schritte sind in Tabelle 10 erläutert.

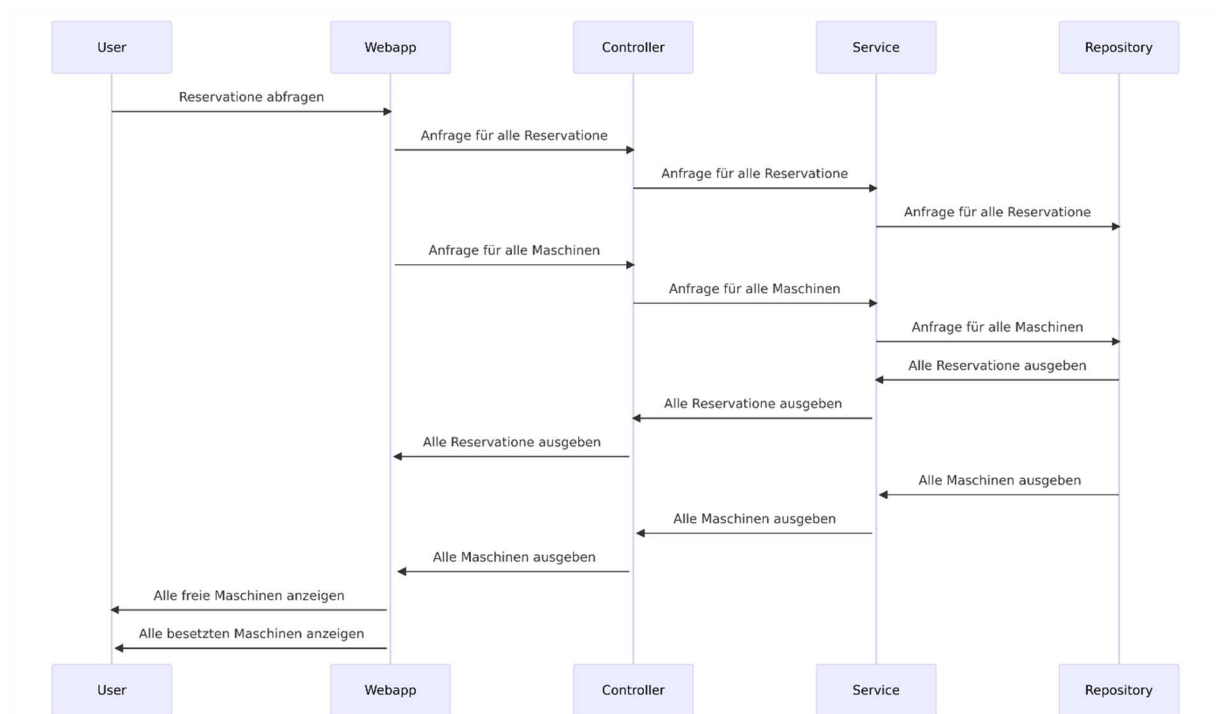


Abbildung 12: Sequenzdiagramm Belegungsüberwachung

Tabelle 10: Belegungsüberwachungsschritte

| Nr. | Schritt |
|-----|---|
| 1. | Der User möchte alle Slots sehen. |
| 2. | Das System ruft alle Maschinen und alle Reservationen ab und generiert daraus die freien und besetzten Slots. |
| 3. | Das System zeigt dem User alle Slots an. |

3.4.4 Maschine hinzufügen

Das Sequenzdiagramm in Abbildung 13 mit den beschriebenen Schritten in Tabelle 11 für das Hinzufügen von Maschinen beschreibt den Prozess, wie ein Administrator eine neue Waschmaschine oder einen neuen Trockner zur Gemeinschaftswaschküche hinzufügt.

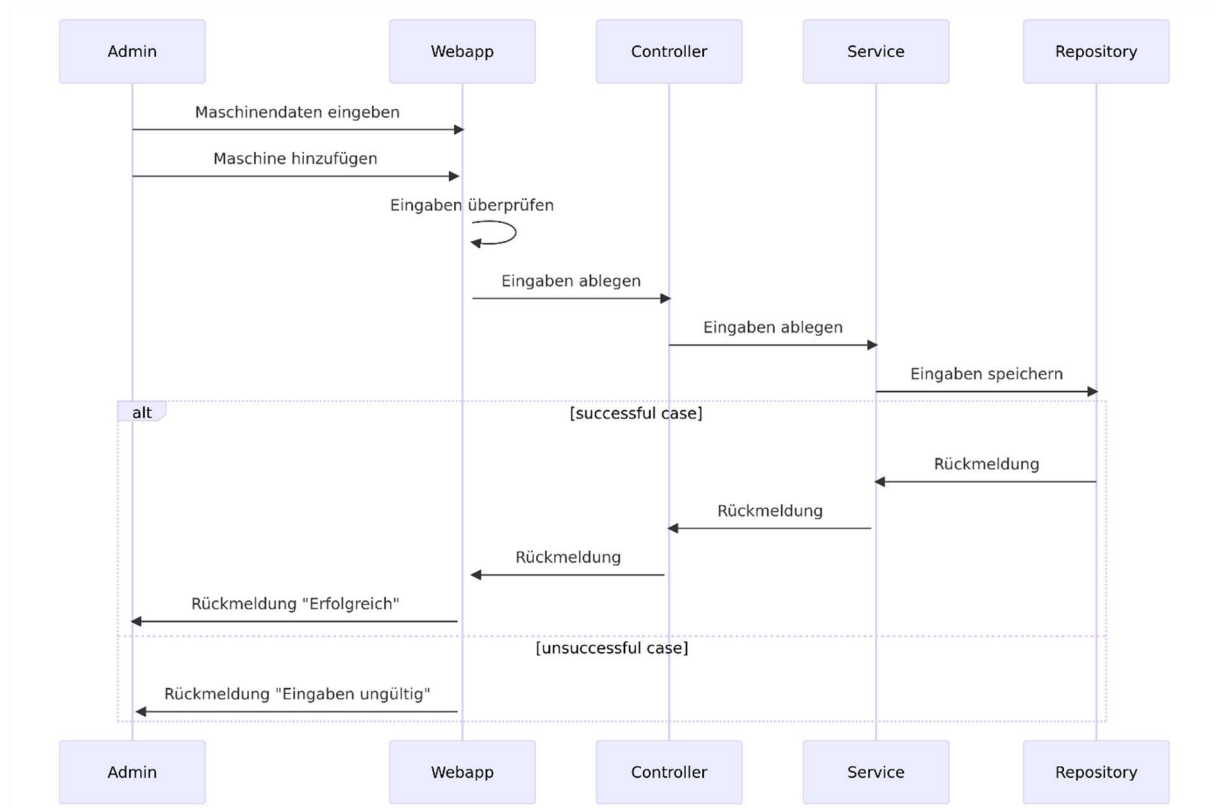


Abbildung 13: Sequenzdiagramm Hinzufügen einer Maschine

Tabelle 11: Schritte des Hinzufügens einer Maschine

| Nr. | Schritt |
|-----|--|
| 1. | Der Admin gibt die Maschinendaten ein. |
| 2. | Wenn die Daten eingeben wurden, fügt er sie hinzu. |
| 3. | Das System überprüft die Eingaben des Admins auf Korrektheit und Validität. |
| 4. | Das System validiert die Eingaben und speichert die neue Maschine. |
| 5. | Wenn die Eingaben valide sind, gibt das System dem Admin positive Rückmeldung. |
| 6. | Wenn die Eingaben nicht korrekt sind, erscheint eine Fehlermeldung. |

3.4.5 Neuigkeiten abrufen

Das Sequenzdiagramm (Abbildung 14) für das Abrufen von Neuigkeiten beschreibt den Prozess, wie ein User die Neuigkeiten betreffend der Gemeinschaftswaschküche abrufen kann. Die Schritte in Tabelle 12 erläutern den genauen Verlauf.

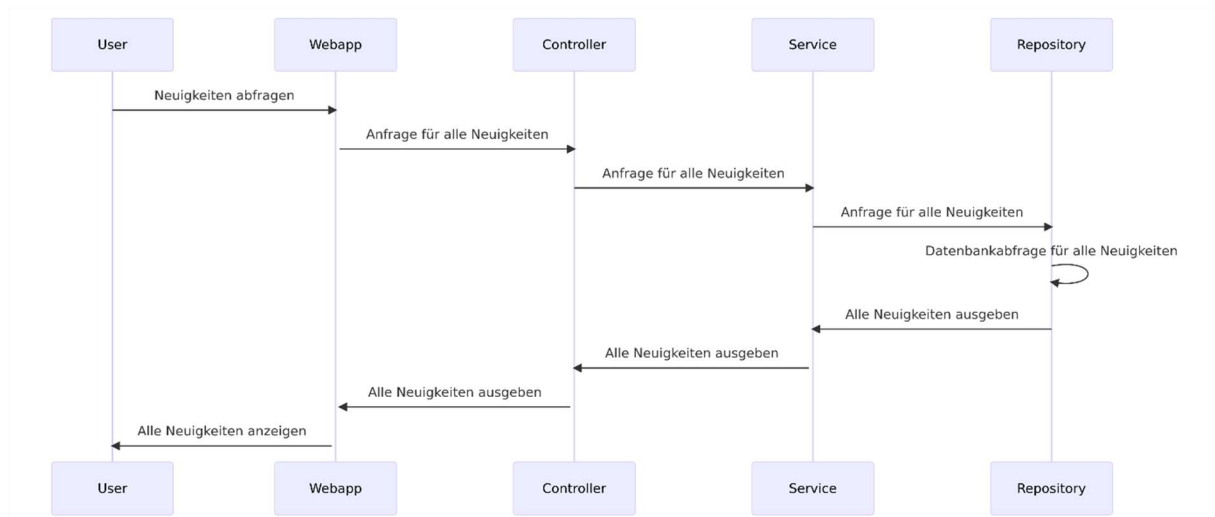


Abbildung 14: Sequenzdiagramm Abrufen von Neuigkeiten

Tabelle 12: Schritte des Abrufens von Neuigkeiten

| Nr. | Schritt |
|-----|--|
| 1. | Der User drückt auf das Glockensymbol, um alle Neuigkeiten anzuzeigen. |
| 2. | Das System ruft alle Neuigkeiten ab. |
| 3. | Das System zeigt dem User alle Neuigkeiten an. |

3.5 Designentscheide

Die Entscheidungen bezüglich der Technologie, die für das Projekt washly verwendet wurden, sind von grosser Bedeutung. Hier wird beschrieben, welche Technologien und Frameworks ausgewählt wurden und warum diese Entscheidungen getroffen wurden.

3.5.1 Design-Entscheide Backend-Komponenten

Die Backend-Technologien sind entscheidend für die Entwicklung und Funktionsweise einer Webanwendung. Sie umfassen Programmiersprachen und Frameworks, die für die Entwicklung und Bereitstellung des Backends verwendet werden.

Java

Java wurde als Hauptsprache für die Entwicklung des Backends von washly gewählt. Java ist eine robuste Sprache, die eine hohe Stabilität und Sicherheit bietet, was für einen Online-Service wie washly von grosser Bedeutung ist. Auch die Skalierbarkeit von Java ist ein Vorteil für das Projekt, da es in der Lage ist, grosse Anwendungen zu handhaben und problemlos auf eine grössere Userzahl zu skalieren.

Ein weiterer wichtiger Faktor, der zu der Wahl von Java beigetragen hat, ist die Erfahrung und Kenntnisse des Entwicklungsteams. Das Team verfügt über umfangreiche Erfahrung in der Java-Entwicklung und ist mit den Spracheigenheiten und -funktionen bestens vertraut. Dies ermöglicht es dem Team, effizient und schnell zu arbeiten und gleichzeitig eine hohe Code-Qualität zu gewährleisten.

Spring

Spring [7] bietet eine Vielzahl von Funktionen, wie z. B. Transaktionsmanagement und eine breite Unterstützung für Datenbanken. Für washly wurde Spring ausgewählt, da es sich als stabile und

zuverlässige Technologie erwiesen hat und eine grosse Community hat, was die Wartbarkeit und Skalierbarkeit des Projekts erleichtert.

SQLite

SQLite [8] wurde ausgewählt, da die Datenbank leichtgewichtig und einfach zu bedienen ist und damit den Anforderungen von washly entspricht. SQLite ist plattformunabhängig und kann daher problemlos auf verschiedenen Betriebssystemen eingesetzt werden.

Eine SQLite-Datenbank kann als Cluster konfiguriert werden, um die Verfügbarkeit und Skalierbarkeit durch die Verteilung von Daten und Vermeidung von Engpässen zu verbessern, aber es ist nicht zwingend notwendig, solange Leistungs- und Verfügbarkeitsanforderungen erfüllt sind.

3.5.2 Design-Entscheidung Frontend-Komponenten

Nachfolgend werden kurz die gewählten Bausteine für das Frontend erklärt, und deren Wahl begründet.

Vue.js

Vue.js [9] ist ein leichtgewichtiges JavaScript-Framework und zeichnet sich durch seine Einfachheit und Flexibilität aus. Vue.js ermöglicht es Entwickler:innen, Komponenten-basierte Anwendungen zu erstellen. Diese ermöglichen es, HTML-Code in die JavaScript-Logik zu integrieren, was eine einfachere Wartung und Erweiterbarkeit der Anwendung ermöglicht. Eine gute Dokumentation und eine grosse Community zeichnen Vue.js aus. Diese genannten Gründe zusammen mit der Tatsache, dass bereits Vorwissen zu Vue.js im Team vorhanden ist, haben die Wahl Vue.js einzusetzen bestärkt und schlussendlich ausgemacht.

Tailwind CSS

Tailwind-CSS [10] ist ein CSS-Framework, das es Entwickler:innen ermöglicht, schneller und effizienter responsive Webanwendungen zu erstellen. Für diese Eigenschaften und für seine optimale Kompatibilität mit Vue.js wurde Tailwind gewählt.

DaisyUI

DaisyUI [11] ist ein zusätzliches UI-Kit, das auf Tailwind-CSS aufbaut und es Entwickler:innen ermöglicht, noch schneller und effizienter ansprechende UI-Komponenten zu erstellen. Durch Vorwissen im Team und den Entscheid Tailwind-CSS einzusetzen, wurde auch die Entscheidung für den Einsatz von DaisyUI gefällt.

3.6 Wichtige Entscheidung GUI

Die Reservations-Ansicht in Form einer reduzierten Kachelansicht in Abbildung 16 (dient rein zur Veranschaulichung des gewählten Designs) wurde gewählt, um sicherzustellen, dass die Anwendung so benutzerfreundlich wie möglich ist. Im Vergleich zur klassischen Kalenderansicht (beispielhaftes Design-Konzept, nicht repräsentativ für washly), wie in Abbildung 15, bietet die Kachelansicht eine einfachere Navigation und eine schnellere Terminbuchung. Insbesondere für User, die auf mobilen Geräten arbeiten, ist die Kachelansicht praktischer und einfacher zu bedienen.

Eine klassische Kalenderansicht wäre zwar besser für Desktop-Anwendungen geeignet, jedoch würde sie auf mobilen Geräten komplizierter und unpraktischer sein. Um sicherzustellen, dass die Anwendung von einer breiteren Nutzerbasis genutzt werden kann, wurde die Kachelansicht als die beste Option gewählt.

Die Kachelansicht ermöglicht es Usern, Termine schnell und einfach zu buchen, ohne dass sie eine lange Reihe von Schritten durchführen müssen. Darüber hinaus ist die Kachelansicht übersichtlich gestaltet und bietet eine klare Darstellung von freien Maschinen.

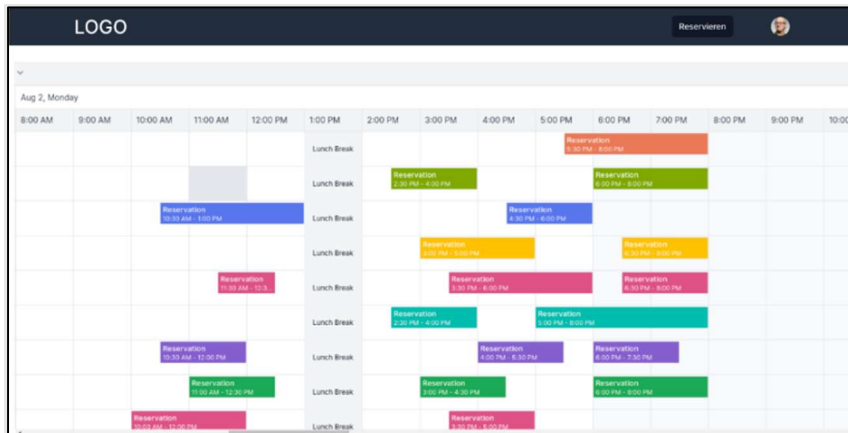


Abbildung 15: Kalender-Ansicht



Abbildung 16: Kachel-Ansicht

3.7 Design Patterns

In diesem Kapitel werden die verschiedenen Architekturmuster beschrieben, die bei der Entwicklung der Softwarearchitektur des Projekts washly zum Einsatz gekommen sind. Architekturmuster stellen bewährte Lösungsansätze für häufig auftretende Entwurfsprobleme dar und können dazu beitragen, die Struktur, Lesbarkeit, Wartbarkeit und Skalierbarkeit der Software zu verbessern.

3.7.1 Model-View-Controller

Das Architekturmuster Model-View-Controller (MVC) [12] dient der Trennung von Geschäftslogik, Datenhaltung und Präsentationsschicht, um eine lose Kopplung zwischen den verschiedenen Komponenten zu ermöglichen.

MVC hat Vorteile wie Wiederverwendbarkeit und einfache Wartung durch lose Kopplung der Schichten sowie verbesserte Testbarkeit durch die Trennung in Model, View und Controller.

In washly ist das MVC-Muster implementiert, um eine klare Trennung von Geschäftslogik, Datenhaltung und Präsentation zu ermöglichen. Der Backend-Code ist in verschiedene Ordner aufgeteilt, wie "controller" für die Entgegennahme von HTTP-Anfragen, "dto" für die Datenübertragung zwischen Client und Server, "model" für die Definition von Entitäten, "repository" für die Datenhaltung sowie "service" für die Implementierung von Geschäftslogik.

3.7.2 Dependency Injection

Im Projekt washly wird das Architekturmuster *Dependency Injection* (DI) [13] angewendet, um eine lose Kopplung zwischen den verschiedenen Komponenten zu gewährleisten und die Testbarkeit zu verbessern. Dabei wird das Spring interne *Dependency Injection Framework* [14] verwendet.

Im Projekt washly werden beispielsweise die Service-Klassen durch Spring-DI verwaltet und bereitgestellt. In der Konfiguration der Spring-DI-Komponenten werden die Abhängigkeiten zwischen den Service-Klassen und den Repository-Klassen definiert. Durch die Verwendung von Spring-DI können diese Abhängigkeiten automatisch erfüllt werden, ohne dass jede Service-Klasse einzeln initialisiert werden muss.

3.7.3 Repository-Pattern

Das Repository-Pattern [15] ist ein Architekturmuster, das im Zusammenhang mit Datenzugriffen und Datenpersistenz verwendet wird. In der Anwendung von washly wird das Repository-Pattern eingesetzt, um eine Abstraktionsschicht zwischen der Datenbank und der restlichen Anwendung zu schaffen.

Im Repository-Pattern werden alle Datenzugriffe auf ein Repository-Objekt abstrahiert, die sich um die Speicherung und Verwaltung der Daten kümmern. Dadurch wird die Abhängigkeit zwischen der Anwendung und der Datenbank reduziert, was die Wartbarkeit und Testbarkeit der Anwendung verbessern.

In der washly-Anwendung werden verschiedene Repository-Klassen verwendet, um auf die verschiedenen Tabellen der Datenbank zuzugreifen. So gibt es beispielsweise das UserRepository, das das User-Model kapselt und die CRUD-Operationen auf der user-Tabelle der Datenbank bereitstellt. Ebenso gibt es das WashingMachineRepository, das auf die washing_machine-Tabelle zugreift und auf das ReservationRepository, das die reservation-Tabelle kapselt.

Durch die Verwendung des Repository-Patterns wird auch die Möglichkeit geschaffen, die Anwendung mit verschiedenen Datenbanken oder Datenspeichern zu betreiben, ohne dass hierfür Änderungen an der restlichen Anwendung notwendig sind.

3.7.4 Service-Layer-Pattern

In washly wird das Service-Layer-Pattern [16] eingesetzt, um eine klare Trennung zwischen der Präsentationsschicht (Controller) und der Datenhaltungsschicht (Repository) zu ermöglichen. Der Service-Layer stellt dabei die Geschäftslogik der Anwendung dar und bietet Schnittstellen an, um Datenverarbeitung und -zugriff zu koordinieren.

Im Ordner "service" des Backend-Teils des Projekts werden verschiedene Service-Klassen implementiert, wie z. B. ReservationService, UserService und WashingMachineService. Diese Klassen enthalten die spezifische Geschäftslogik für die jeweiligen Entitäten, wie z. B. die Erstellung und Verwaltung von Reservierungen, Usern und Waschmaschinen.

3.7.5 Builder-Pattern

In washly wird das Builder-Pattern [17] verwendet, um eine einfache und intuitive Erstellung von Objekten zu ermöglichen. Insbesondere wird das Pattern in der Erstellung von Datenübertragungsobjekten (DTOs) verwendet.

Das Builder-Pattern ist eine Alternative zu Konstruktoren mit vielen Parametern, die unübersichtlich und fehleranfällig sein können. Stattdessen erstellt das Builder-Pattern ein separates Objekt, das die Erstellung des gewünschten Objekts abwickelt. Dieses Objekt stellt Methoden bereit, um jeden Parameter separat zu setzen und kann dann das endgültige Objekt erstellen.

In washly sind die DTO-Klassen auf diese Weise konstruiert. Sie enthalten ein öffentliches statisches Builder-Objekt, das mit den verschiedenen Eigenschaften des DTOs aufgerufen werden kann. Jede Methode des Builders setzt einen Parameter des DTOs und gibt das Builder-Objekt selbst zurück, um die Methodenaufrufe zu verketteten. Wenn alle gewünschten Parameter gesetzt sind, wird die build()-Methode aufgerufen, um das endgültige DTO-Objekt zurückzugeben.

Dies ermöglicht es Entwickler:innen, die DTOs einfach und intuitiv zu erstellen, indem sie nur die gewünschten Eigenschaften setzen und sich nicht um die Reihenfolge der Parameter oder ihre Bedeutung sorgen müssen.

3.7.6 DTO-Pattern

In washly wird das DTO-Pattern (Data Transfer Object) [18] verwendet, um Datenobjekte zwischen der Anwendungsschicht und der Präsentationsschicht zu übertragen. Konkret werden DTO-Klassen verwendet, um komplexe Objekte aus der Geschäftslogik in einfachere Objekte zu transformieren, die leicht von der Präsentationsschicht konsumiert werden können.

Im Backend-Teil des Projekts gibt es spezielle DTO-Klassen wie zum Beispiel ReservationDto und WashingMachineDto, die verwendet werden, um Daten von der Präsentationsschicht zu empfangen oder an die Präsentationsschicht zu senden. Sie enthalten nur die notwendigen Attribute, um die Datenübertragung zwischen verschiedenen Schichten der Anwendung zu erleichtern.

Das DTO-Pattern hilft dabei, die Datenübertragung zwischen den Schichten zu vereinfachen und die Datenstrukturen klarer und leichter verständlich zu machen.

3.8 Authentifizierung

Nachfolgend wird die Authentifizierung anhand vom gelieferten MVP-Stand erklärt und ein Ausblick für den zukünftigen Produktiveinsatz gegeben.

3.8.1 Authentifizierung MVP

Die Authentifizierung bzw. das Login wurde lediglich zu Demonstrationszwecken implementiert. Der Login-Mechanismus prüft clientseitig, ob die eingegebene Mail-Adresse (in der Datenbank) vorhanden ist und ob das statisch gesetzte Passwort übereinstimmt. Dabei handelt es sich um ein allgemeines Passwort, das für alle Benutzer gleich ist. Dieses einfache Vorgehen ermöglicht es, den Login-Prozess schnell und unkompliziert zu gestalten, um den Fokus auf andere Projektbereiche zu legen. Es ist in keiner Weise für den Einsatz in einem produktiven Umfeld geeignet, da es keine angemessene Sicherheit bietet.

3.8.2 Ausblick Produktiveinsatz

Die Authentifizierung ist ein kritischer Bereich, bei dem Sicherheitsaspekte nicht vernachlässigt werden sollten. Das Team verfügt nicht über das erforderliche Know-how, um eine eigene Implementation durchzuführen, was den Kauf von externem Know-how erforderlich machen würde. Eine Alternative besteht darin, die Architektur minimal umzubauen und einen Login-Provider wie Auth0 [19] zu verwenden, um die Sicherheit zu erhöhen und gleichzeitig den Aufwand und die Kosten für eine eigene Implementation zu reduzieren.

4 Implementation

In diesem Kapitel werden die Testkonzepte für das automatisierte und manuelle Testing erläutert, sowie die Lieferergebnisse und ausführbare Dateien genannt. Informationen zu der technischen Implementation der Architektur finden sich in Kapitel 3 Design.

4.1 Lieferergebnisse

Die Lieferergebnisse bestehen aus einem [Github-Projekt](#) mit jeweiligen Unterprojekten für das Frontend ([washly-vue](#)) und das Backend ([washly-api](#)). Darin befinden sich die ganze Code-Base sowie die Dokumentation zum erarbeiteten Code.

4.2 Ausführbare Dateien

Da washly eine Web-App ist, können keine ausführbare Dateien angegeben werden. Zur Ausführung von sowohl Frontend wie auch Backend wird eine Server-Instanz benötigt. Die generelle Installations-Anleitung befindet sich jeweils im dazugehörigen Readme im Github-Repository:

- [Frontend](#)
- [Backend](#)

4.3 Automatisiertes Testing

Diverse Ansätze für die automatisierte Qualitätssicherung der Web-App wurden gewählt; unterschieden wird zwischen Backend- und Frontend-Testing.

4.3.1 Unit-Tests Backend

Automatisierte Tests führen zu einer hohen Code-Qualität und Zuverlässigkeit. Um dies sicherzustellen, wird im Backend das Framework JUnit [20] eingesetzt.

Es wurden Tests für die Service-Klassen geschrieben, um die Geschäftslogik und die Integration mit den Repositories zu überprüfen. Auch die Controller-Klassen wurden getestet, um zu verifizieren, dass die HTTP-Endpunkte korrekt funktionieren und die erwarteten Antworten zurückgeben.

4.3.2 Unit-Tests Frontend

Nach sorgfältiger Abwägung der Bedürfnisse und Anforderungen der Vue.js-Anwendung wurde beschlossen, keine Unit-Tests zu schreiben. In Anbetracht der begrenzten Menge an Business-Logik in der Anwendung ist es nicht kosteneffektiv, die Zeit und die Ressourcen aufzuwenden, die für das Schreiben von Unit-Tests erforderlich sind. Stattdessen werden andere Formen von Tests bevorzugt, wie z. B. End-to-End-Tests oder manuelle Tests, um zu verifizieren, dass die Anwendung wie vorgesehen funktioniert.

Diese Entscheidung ist der effizienteste und effektivste Ansatz für das Projekt, um sich so auf das Testen der kritischen Anwendungsfälle der Anwendung konzentrieren zu können und gleichzeitig Zeit und Ressourcen zu sparen. Jedoch wurde die Teststrategie im Laufe des Projekts weiter evaluiert und es werden bei Bedarf Anpassungen vorgenommen, um sicherzustellen, dass die Anforderungen aller Interessengruppen erfüllt werden.

4.3.3 End-to-End-Testing Frontend

Wenn die Applikation weiter anwächst und weitere Features hinzukommen (nach dem MVP für PM3), werden E2E-Tests für das Vue.js Frontend sinnvoll. Hierbei würde auf das *Playwright Framework* [21] von Microsoft gesetzt werden.

Playwright ist ein von Microsoft entwickeltes Open-Source-Tool für browserbasiertes End-to-End-Testing von Webanwendungen. Es ermöglicht das Testen von Webanwendungen in verschiedenen Zuständen und Betriebssystemen und bietet dabei eine hohe Flexibilität und Wartbarkeit.

Durch das automatisierte Testing können potenzielle Fehler frühzeitig erkannt und behoben werden, bevor die Anwendung in Produktion geht. Playwright bietet die Möglichkeit, Tests in verschiedenen Browsern auszuführen und somit sicherzustellen, dass die Anwendung auf allen gängigen Browsern einwandfrei läuft. Dies spart Zeit und Ressourcen und trägt zur Verbesserung der Qualität und Zuverlässigkeit der Anwendung bei.

4.4 Manuelles Testing

Durch manuelles, exploratives Testing wurde die Web-App zusätzlich zu den automatisierten Tests auf ihre Qualität überprüft. Hierbei wurde auf Black-Box-Testing gesetzt; es wurden Use-Case-Testing, Quick-Testing, Failure-Testing und Scenario-Testing genutzt.

Das Testing variierte pro Use-Case zwischen einem explorativen Ansatz oder reinem Akzeptanz-Testing. Durch einen abschliessenden Testbericht, ersichtlich in Anhang A Manuelle Testingsresultate, wurde die Applikation nach bestem Ermessen verifiziert.

5 Resultate

Die Projektergebnisse werden in diesem Kapitel zusammengefasst und in Bezug auf die ursprüngliche Planung des Projektes abgebildet. Der geplante sowie reelle Arbeitsaufwand wurde im Anhang unter B.2 Iterationsplanung aufgeführt.

5.1 Erreichte Ziele

Die definierten und während des Projektverlaufs veränderten Ziele wurden erreicht - es wurden allerdings einige Anpassungen vorgenommen. Das Projektergebnis ist in Kapitel 4.1 Lieferergebnisse erläutert.

Use-Cases

Die Use-Cases wurden gemäss dem Use-Case-Modell in Kapitel 2.1 vollends implementiert und getestet.

Login und Newsfeed

Während der Implementation wurden zwei Funktionalitäten zu den Iterationen hinzugefügt, welche ursprünglich nicht geplant waren. Nach anfänglichem Entscheid, dass das Login und der Newsfeed nicht in den Rahmen des PM3-Projekts passen, wurde dieser revidiert und diese zwei Funktionen neu priorisiert, damit der MVP an essenziellem Umfang gewinnt. Wichtig ist hier zu erwähnen, dass das Login nur zu Demonstrationszwecken implementiert wurde.

5.2 Offene Punkte

Nachfolgend ist definiert, welche Funktionalitäten für dieses MVP in PM3 nicht implementiert wurden. Durch diese Abgrenzungen wurde ein machbarer Rahmen für das Projekt geschaffen, indem der Fokus auf die wichtigsten Funktionen gelegt wurde, um den Zeitaufwand und die Komplexität in einem realistischen Bereich zu halten.

Benutzer-Rollen

Die Benutzer-Rollen (User/Admin) sind eng mit der Login-Funktionalität gekoppelt. Da diese Funktionalität erst während der Implementation neu priorisiert und implementiert wurde, konnten die Benutzer-Rollen nicht implementiert werden.

Verwaltung von multiplen Waschküchen

Es ist noch nicht möglich, mehrere verschiedene Waschküchen zu verwalten. Nur eine einzelne Waschküche ist verwaltbar.

Waschstatistiken

Der Use-Case für Administrator:innen, Waschstatistiken der Waschküche einzusehen, wurde während des Projektverlaufs aus den Iterationsplanungen entfernt, um Zeit für andere, höher priorisierte Funktionen zu nutzen.

5.3 Weiterentwicklungen

Mögliche Weiterentwicklungen für washly sind bereits definiert; Diese sind im Kapitel 2.4 Zusätzliche Anforderungen in Tabelle 3 sowie Tabelle 4 und im Kapitel 5.2 Offene Punkte erläutert.

6 Selbstständigkeitserklärung

Hiermit erklären wir, dass wir den vorliegenden Technischen Bericht selbstständig verfasst haben. Sämtliche verwendete Quellen und Hilfsmittel sind ordnungsgemäß und vollständig angegeben. Alle direkten oder indirekten Zitate sind als solche gekennzeichnet und die Verwendung von Texten, Daten und Bildern Dritter erfolgte unter Beachtung der Urheberrechte.

Winterthur 15.05.2023

Gruppe 1, washly

7 Quellenverzeichnis

- [1] Bosch Global. (D. u.). *WeWash* [Online]. <https://www.bosch.com/stories/wewash/> [Stand: 27.02.2023]
- [2] appWash. (D. u.). *appWash* [Online]. <https://appwash.com/de/> [Stand: 25.02.2023]
- [3] Apple App Store. (D. u.). *Terresta App - Apple App Store* [Online]. <https://apps.apple.com/ch/app/terresta/id1585209644> [Stand: 27.02.2023]
- [4] Google Play. (D. u.). *Terresta App - Google Play Store* [Online]. <https://play.google.com/store/apps/details?id=ch.infrasupport.terresta> [Stand: 27.02.2023]
- [5] Terresta. (D. u.). *Terresta Waschküchen-App* [Online]. <https://www.terresta.ch/apps> [Stand: 27.02.2023]
- [6] Wikipedia. (02.02.2023). *Software as a Service* [Online]. https://de.wikipedia.org/w/index.php?title=Software_as_a_Service&oldid=230467740 [Stand: 01.03.2023]
- [7] Spring. (D. u.). *Spring Framework* [Online]. <https://spring.io> [Stand: 15.04.2023]
- [8] SQLite. (D. u.). *SQLite Home Page* [Online]. <https://www.sqlite.org/index.html> [Stand: 15.04.2023]
- [9] Vue.js. (D. u.). *Vue.js - The Progressive JavaScript Framework | Vue.js* [Online]. <https://vuejs.org/> [Stand: 17.04.2023]
- [10] Tailwind CSS. (D. u.). *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. [Online]. <https://tailwindcss.com/> [Stand: 15.04.2023]
- [11] daisyUI. (D. u.). *daisyUI — Tailwind CSS Components* [Online]. <https://daisyui.com/> [Stand: 15.04.2023]
- [12] W3Schools. (D. u.). *What is MVC Architecture?* [Online]. <https://www.w3schools.in/mvc-architecture> [Stand: 15.04.2023]
- [13] T. Janssen. (19.06.2018). *Design Patterns Explained – Dependency Injection with Code Examples* [Online]. <https://stackify.com/dependency-injection/> [Stand: 15.04.2023]
- [14] Spring Docs. (D. u.). *Core Technologies* [Online]. <https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-dependencies> [Stand: 17.04.2023]
- [15] T. Siddiqui. (27.01.2022). *The Repository Pattern in C# | CodeGuru.com* [Online]. <https://www.codeguru.com/csharp/repository-pattern-c-sharp/> [Stand: 15.04.2023]
- [16] Java Design Patterns. (D. u.). *Service Layer* [Online]. <https://java-design-patterns.com/patterns/service-layer/> [Stand: 15.04.2023]
- [17] Dofactory. (D. u.). *C# Builder Design Pattern - Dofactory* [Online]. <https://www.dofactory.com/net/builder-design-pattern> [Stand: 15.04.2023]
- [18] Baeldung. (29.08.2021). *The DTO Pattern (Data Transfer Object) | Baeldung* [Online]. <https://www.baeldung.com/java-dto-pattern> [Stand: 15.04.2023]
- [19] Auth0. (D. u.). *Auth0: Secure access for everyone. But not just anyone.* [Online]. <https://auth0.com/> [Stand: 14.05.2023]
- [20] JUnit. (D. u.). *JUnit 5* [Online]. <https://junit.org/junit5/> [Stand: 06.05.2023]
- [21] Playwright. (D. u.). *Fast and reliable end-to-end testing for modern web apps | Playwright* [Online]. <https://playwright.dev/> [Stand: 06.05.2023]

8 Abbildungsverzeichnis

| | |
|---|----|
| Abbildung 1: Use-Case-Diagramm..... | 4 |
| Abbildung 2: Use-Case «Maschine erfassen» | 5 |
| Abbildung 3: Use-Case «Waschtermin buchen» | 7 |
| Abbildung 4: Domänenmodell MVP..... | 11 |
| Abbildung 5: Domänenmodell für marktfertiges Produkt..... | 11 |
| Abbildung 6: Systemübersichtsdiagramm | 13 |
| Abbildung 7: Datenbankmodell..... | 14 |
| Abbildung 8: Klassendiagramm | 15 |
| Abbildung 9: Datenflussdiagramm | 16 |
| Abbildung 10: Sequenzdiagramm Authentifizierung..... | 17 |
| Abbildung 11: Sequenzdiagramm Reservierung | 18 |
| Abbildung 12: Sequenzdiagramm Belegungsüberwachung | 19 |
| Abbildung 13: Sequenzdiagramm Hinzufügen einer Maschine | 20 |
| Abbildung 14: Sequenzdiagramm Abrufen von Neuigkeiten | 21 |
| Abbildung 15: Kalender-Ansicht..... | 23 |
| Abbildung 16: Kachel-Ansicht..... | 23 |
| Abbildung 17: Organigramm | 38 |

9 Tabellenverzeichnis

| | |
|--|----|
| Tabelle 1: Markteinschätzung der Einnahmen | 3 |
| Tabelle 2: Auflistung der Use-Cases pro Akteur:in | 4 |
| Tabelle 3: Zusätzliche funktionale Anforderungen..... | 10 |
| Tabelle 4: Zusätzliche nicht-funktionale Anforderungen..... | 10 |
| Tabelle 5: Backend-Module | 12 |
| Tabelle 6: Frontend-Module | 13 |
| Tabelle 7: Datenflussschritte | 16 |
| Tabelle 8: Authentifizierungsschritte..... | 18 |
| Tabelle 9: Reservierungsschritte..... | 19 |
| Tabelle 10: Belegungsüberwachungsschritte | 19 |
| Tabelle 11: Schritte des Hinzufügens einer Maschine | 20 |
| Tabelle 12: Schritte des Abrufens von Neuigkeiten | 21 |
| Tabelle 13: Manuelle Testingresultate «Maschine erfassen»..... | 34 |
| Tabelle 14: Manuelle Testingresultate «Waschmaschine bearbeiten» | 35 |
| Tabelle 15: Manuelle Testingresultate «Waschtermin buchen» | 35 |
| Tabelle 16: Manuelle Testingresultate «User einladen» | 36 |
| Tabelle 17: Manuelle Testingresultate «User entfernen» | 37 |
| Tabelle 18: Manuelle Testingresultate «Waschplan einsehen» | 37 |
| Tabelle 19: Iterationsplanung | 38 |
| Tabelle 20: Projektrisiken | 40 |

10 Glossar

Das folgende Glossar listet die für das Verständnis notwendigen Begriffe auf und erläutert diese.

| Begriff | Beschreibung |
|---------------------------|--|
| Administrator:in | Die Person oder Verwaltung, die die Waschküche und somit auch den Waschplan verwaltet. |
| API | (Application Programming Interface) Ist die Schnittstelle, die es verschiedenen Anwendungen ermöglicht, miteinander zu kommunizieren. |
| Backend | Teil einer Softwareanwendung, der sich mit der Verarbeitung von Daten und der Logik der Anwendung befasst. Es besteht aus verschiedenen Komponenten wie der Datenbank, APIs und Anwendungslogik. |
| Backups | Ist eine Kopie von Daten, die erstellt wird, um im Falle eines Datenverlusts eine Wiederherstellung zu ermöglichen. |
| Black-Box-Testing | Die Software wird als Black-Box betrachtet, somit wird das Verhalten der Software aus der Perspektive eines Users ohne internes Code-Wissen getestet. |
| Cloud | Ermöglicht die Nutzung von Rechenleistung und Speicherplatz von entfernten Servern, anstatt lokale Server oder Computer zu verwenden. |
| Composable | Composables sind Funktionen in Vue.js, die spezifische Funktionalitäten implementieren und in Komponenten wiederverwendet werden können, um Code modularer und einfacher wartbar zu machen. Sie können als eigenständige Funktionen oder Teil von Plugins definiert werden. |
| Create-REST-Anfrage | (POST-Anfrage) Ist eine Anfrage, um neue Daten in Datenbank zu erstellen. |
| CRUD-Operationen | Eine Abkürzung für "Create, Read, Update, Delete"-Operationen |
| Datenbank-Cluster | Eine Gruppe von miteinander verbundenen und koordinierten Datenbank-Servern, die zusammenarbeiten. |
| Dependency Injection (DI) | Ist ein Entwurfsmuster in der Softwareentwicklung, dass die Abhängigkeiten zwischen verschiedenen Komponenten einer Anwendung vereinfacht und flexibler gestaltet. |
| DTO | (Data Transfer Object) Überträgt Daten zwischen verschiedenen Schichten einer Anwendung. |
| Exception | Ist ein Ereignis, das während der Ausführung eines Programms auftritt und das normale Verhalten der Anwendung unterbricht. |
| Exception Handling | Bezieht sich auf die Behandlung von Exceptions. |
| Failure-Testing | Es werden gezielt Fehler in der Software provoziert, um das Verhalten und den Umgang mit Fehlern innerhalb der Software zu testen. |
| Frontend | Der Teil einer Softwareanwendung, der für die Darstellung von Daten und die Interaktion mit dem/der Benutzer:in verantwortlich ist. |
| Geschäftslogik | Enthält die Regeln, die bestimmen, wie die Anwendung Daten verarbeitet und manipuliert. Die Geschäftslogik ist in der Regel unabhängig von der Implementierung der Benutzeroberfläche und der Datenhaltungsschicht. |
| High Availability | High Availability bezieht sich auf die Fähigkeit eines Systems, kontinuierlich und ohne Unterbrechungen verfügbar zu sein, auch wenn einzelne Komponenten ausfallen oder Wartungsarbeiten durchgeführt werden müssen. |
| HTTPS | HTTPS (Hypertext Transfer Protocol Secure) Ist ein verschlüsseltes Kommunikationsprotokoll im Internet, das eine sichere Verbindung zwischen einem Webserver und einem Client herstellt, um die Vertraulichkeit und Integrität von Daten während der Übertragung zu gewährleisten. |

| Begriff | Beschreibung |
|--------------|---|
| JSON-Objekte | (JavaScript Object Notation) Ist ein Datenformat, das zur Übertragung strukturierter Daten zwischen verschiedenen Anwendungen verwendet wird |
| MVP | (Minimal Viable Product) Definiert das funktionierende, noch nicht fertiggestellte Softwareprodukt mit einem Minimum an Funktionen, dass bereits geliefert werden kann. |
| Stand Alone | Ohne Zusätze nutzbar |
| User | Personen, die washly nutzen. |
| Waschende | Personen, die die Waschküche nutzen. |

Anhang

A Manuelle Testingresultate

Tabelle 13: Manuelle Testingresultate «Maschine erfassen»

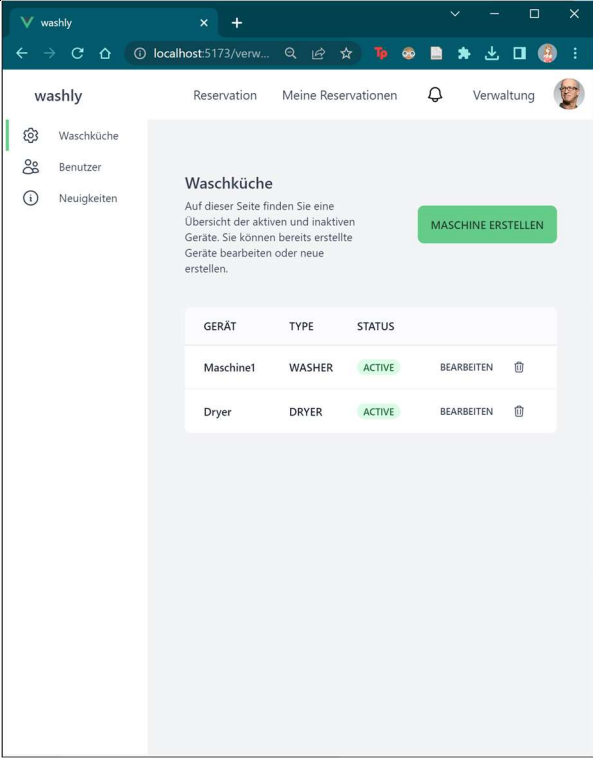
| Testschritte | Beobachtungen | Fazit |
|---|---|-------|
| <p>Admin:</p> <ul style="list-style-type: none">✓ Verwaltungsansicht✓ Waschküche → Maschine erstellen <hr/> <p>Titel:</p> <ul style="list-style-type: none">- ✓ Keine Eingabe (nicht möglich)- ✓ Normale Eingabe (möglich)- ✓ <50 Zeichen (möglich) → vernachlässigbar <p>Type:</p> <ul style="list-style-type: none">- ✓ Dropdown <p>Status:</p> <ul style="list-style-type: none">- ✓ Dropdown <hr/> <p>Nach Eingabe:</p> <ul style="list-style-type: none">✓ Maschine ist erstellt und bearbeit- und löschar✓ Für User sofort buchbar |  | ✓ |

Tabelle 14: Manuelle Testingresultate «Waschmaschine bearbeiten»

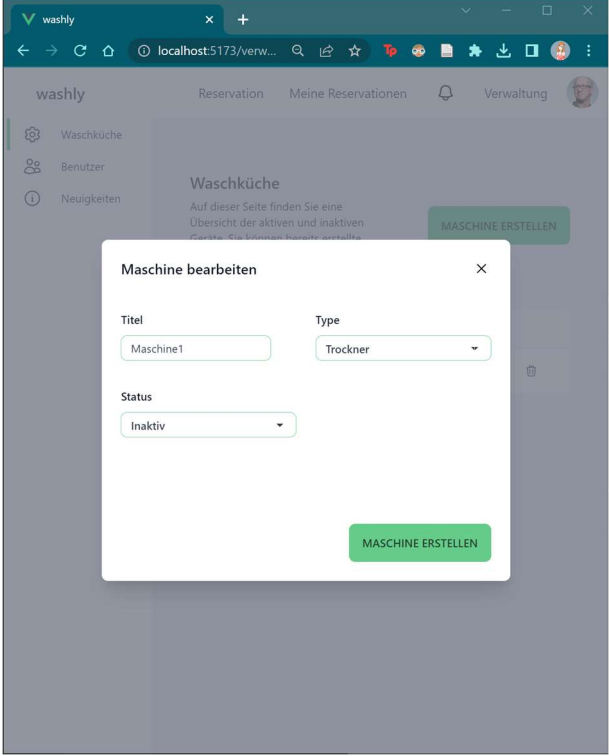
| Testschritte | Beobachtungen | Fazit |
|---|---|--|
| <p>Admin:</p> <ul style="list-style-type: none"> ✓ Verwaltungsansicht ✓ Waschküche → Maschine bearbeiten |  | <p>✗ Button heisst noch «Maschine erstellen»</p> |
| <ul style="list-style-type: none"> ✓ Bearbeitungen analog Maschine erstellen | | <p>-> Aktualisiert:</p> <ul style="list-style-type: none"> ✓ Button heisst «Maschine bearbeiten» |
| <p>✗ Button heisst noch «Maschine erstellen»</p> <p>-> Aktualisiert:</p> <ul style="list-style-type: none"> ✓ Button heisst «Maschine bearbeiten» <ul style="list-style-type: none"> ✓ Bearbeitungen sofort ersichtlich für User | | |

Tabelle 15: Manuelle Testingresultate «Waschtermin buchen»

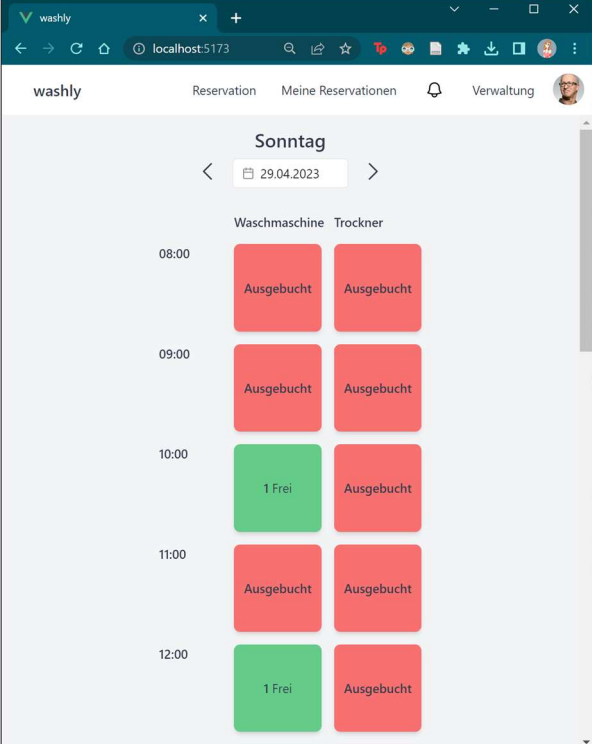
| Testschritte | Beobachtungen | Fazit |
|--|--|---|
| <p>User:</p> <ul style="list-style-type: none"> ✓ Reservation ✓ Freien Slot mit Zeit auswählen und reservieren ✓ Bestätigung angezeigt ✓ Reservation sofort ersichtlich ✓ Ausgebuchte Slots nicht buchbar ✗ Wochentag nicht richtig <p>-> Aktualisiert:</p> <ul style="list-style-type: none"> ✓ Wochentag stimmt mit Datum übereint |  | <p>✗ Wochentag nicht richtig</p> |
| | | <p>-> Aktualisiert:</p> <ul style="list-style-type: none"> ✓ Wochentag stimmt mit Datum überein |

Tabelle 16: Manuelle Testingresultate «User einladen»

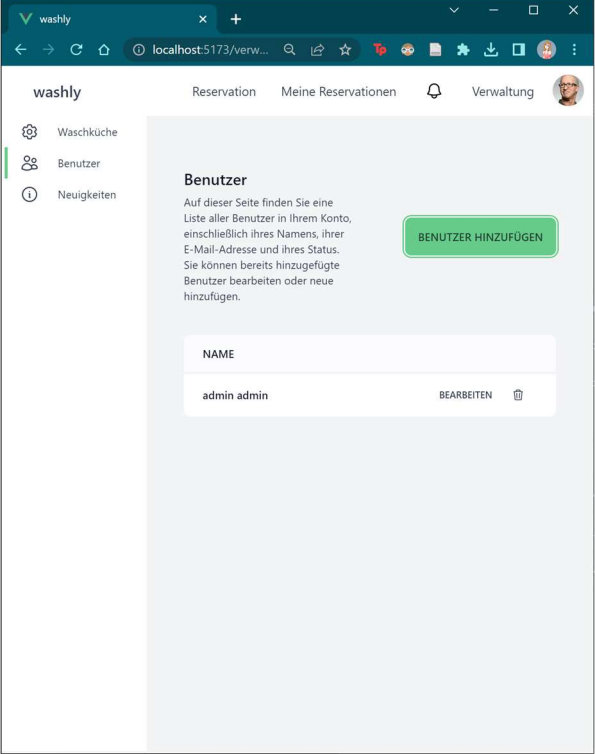
| Testschritte | Beobachtungen | Fazit |
|---|---|---|
| <p>Admin:</p> <ul style="list-style-type: none"> ✓ Verwaltungsansicht ✓ Benutzer → Benutzer hinzufügen <hr/> <p>Name/Nachname:</p> <ul style="list-style-type: none"> ✓ Keine Eingabe (nicht möglich) ✓ Normale Eingabe (möglich) ✓ <50 Zeichen (möglich) → vernachlässigbar <p>Email:</p> <ul style="list-style-type: none"> ✓ Email-Syntax muss erfüllt sein ✗ Typo: Bitte einen Email eingeben. Aktualisiert: ✓ Neu: Bitte eine Email eingeben <hr/> <p>✗ Benutzer bearbeiten: Button falsch</p> <p>-> Aktualisiert: ✓ Benutzer bearbeiten: Button «Benutzer bearbeiten»</p> |  | <p>✗ Typo: Bitte einen Email eingeben.</p> <p>✗ Benutzer bearbeiten: Button falsch</p> <p>-> Aktualisiert: ✓ Neu: Bitte eine Email eingeben</p> <p>✓ Benutzer bearbeiten: Button «Benutzer bearbeiten»</p> |

Tabelle 17: Manuelle Testresultate «User entfernen»

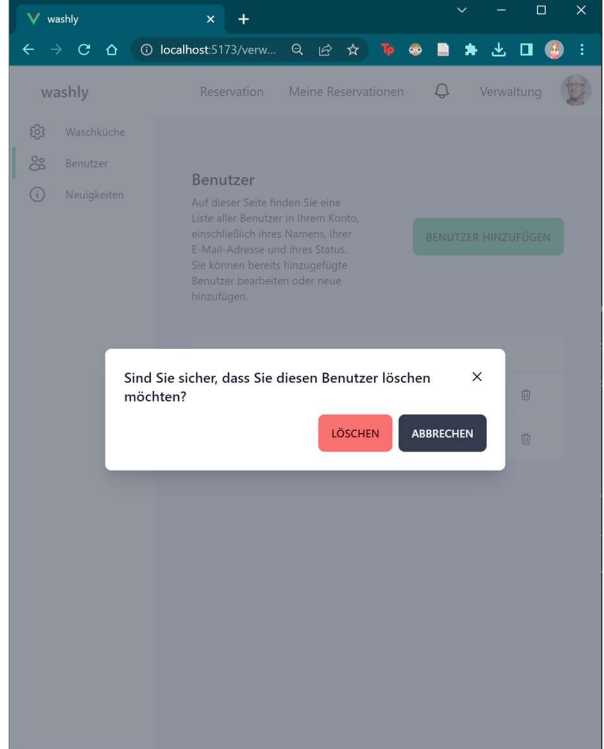
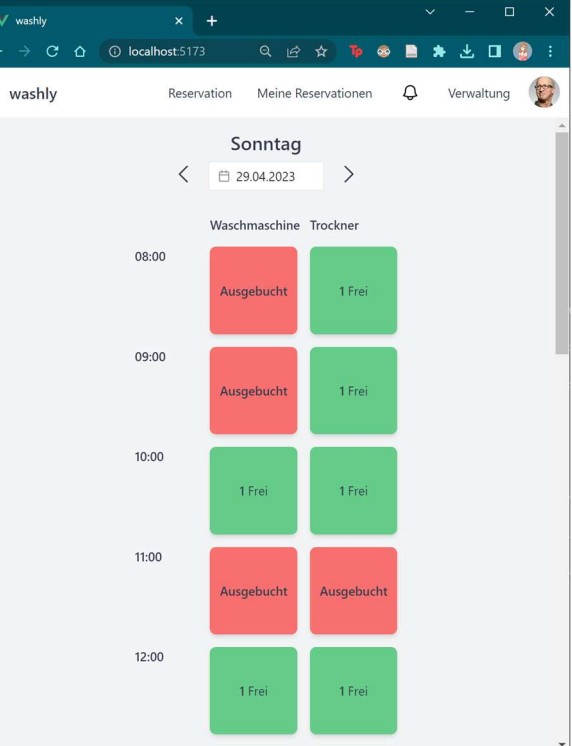
| Testschritte | Beobachtungen | Fazit |
|---|---|-------|
| Admin: ✓ Verwaltungsansicht ✓ Benutzer → Benutzer entfernen ✓ Benutzer sofort entfernt |  | ✓ |

Tabelle 18: Manuelle Testresultate «Waschplan einsehen»

| Testschritte | Beobachtungen | Fazit |
|--|--|-------|
| User: ✓ Reservation ✓ Wechsel der Tage durch Pfeiltasten möglich ✓ Datumswahl über Kalender möglich |  | ✓ |

B Projektmanagement

Folgend werden ein Organigramm der verschiedenen Rollen gezeigt, sowie die Zeitplanung des Projektes und die Risikoliste aufgezeigt und erläutert.

B.1 Organigramm

Das Organigramm in Abbildung 17 zeigt die interne Projektstruktur und Arbeitsverteilung der unterschiedlichen Rollen.

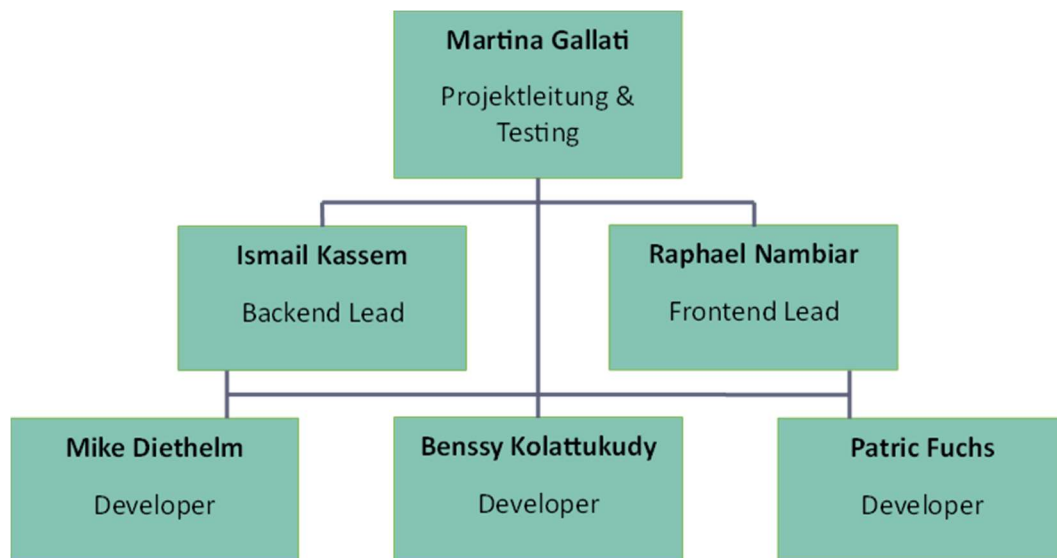


Abbildung 17: Organigramm

B.2 Iterationsplanung

Das Projekt wurde in Iterationen angegangen, nachfolgend in Tabelle 19 aufgelistet. Die Diskrepanzen belegen sich durch Unterschätzungen des Arbeitsaufwandes in der ersten Projekthälfte, sowie Zeitverzögerungen durch Verschiebungen der Abgabetermine.

Tabelle 19: Iterationsplanung

| Iteration bis: | Artefakte und Reviews | Iterationsziele | Ziele erreicht? Wenn nein: Erklärung! | Aufwand geschätzt und real in h | |
|----------------|-----------------------|---|---------------------------------------|---------------------------------|-----|
| 06.03.2023 | M1: Projektskizze | <ul style="list-style-type: none">- Projektskizze abschliessen und abgeben- Präsentation vorbereiten und durchführen- PoC der Technologien aufsetzen, technischen Prototyp entwickeln | ✓ | 120 | 100 |
| 20.03.2023 | Iterationsreview #2 | <ul style="list-style-type: none">- Use-Case-Modell für technischen Bericht konkretisieren- Zeitplan schätzen- Workflow in GitHub festlegen- Manuellen Testplan erarbeiten- Vorbereitung Iterationsreview | ✓ | 120 | 110 |

| Iteration bis: | Artefakte und Reviews | Iterationsziele | Ziele erreicht? Wenn nein: Erklärung! | Aufwand geschätzt und real in h | |
|----------------|--|--|--|---------------------------------|-----|
| 03.04.2023 | Iterationsreview #3 | <ul style="list-style-type: none"> - Technische Issues für Code erstellen - Sprachliches Feedback für M2 einholen - Use-Case-Modell fertigstellen - Vorbereitung Iterationsreview | ✓ | 120 | 100 |
| 17.04.2023 | M2: Technischer Bericht I – Lösungsarchitektur | <ul style="list-style-type: none"> - Technischen Bericht der Lösungsarchitektur abschliessen und abgeben - Präsentation vorbereiten und durchführen - App-Navigation implementieren - Screen-Mockups erstellen - Reservationsprototyp erstellen | Technischer Bericht: In Arbeit aufgrund Feedbacks durch Dozierende, ansonsten: ✓ | 120 | 140 |
| 01.05.2023 | Iterationsreview #5 | <ul style="list-style-type: none"> - Technischen Bericht der Lösungsarchitektur abschliessen und abgeben - Frontend-Backend-Anbindung - Reservations-Feature komplettieren - Design-Entscheid Kalenderansicht Desktop - Manuelles Testing starten - Login implementieren - Projektbericht M3 aufsetzen und sprachliches Feedback einholen | ✓ | 120 | 130 |
| 15.05.2023 | M3: Technischer Bericht II – Beta-Release | <ul style="list-style-type: none"> - Technischen Bericht des Beta-Releases abschliessen und abgeben - Projekt-Rampdown: Code-Basis abschliessen | ✓ | 120 | 140 |

B.3 Projektrisiken

In der folgenden Tabelle 20 der Projektrisiken werden einzelne Gefährdungen des Projekts aufgelistet, nach Priorität absteigend sortiert und versucht, eine Mitigation zu finden.

Tabelle 20: Projektrisiken

| Risiko | Beschreibung | Mitigation | I | P |
|-------------------------|---|--|-----------|---------|
| Nutzerakzeptanz | Waschende können sich gegen die Digitalisierung Ihres Waschplans wehren. | Um auch Waschende abzuholen, die diesen Digitalisierungsschritt nicht machen möchten oder können, kann die Verwaltung einen anonymen, nicht nutzerbezogenen wiederkehrenden Termin für diejenigen Personen erstellen. | Hoch | Mittel |
| Datenschutz | Die Applikation muss alle Datenschutzgesetze und -verordnungen ausnahmslos befolgen und den Usern und Administrator:innen mitteilen, welche personenbezogenen Daten gesammelt und gespeichert werden. | Die Applikation kommuniziert den Usern und Administrator:innen alle nötigen Informationen. Personen, die Ihre Daten gänzlich nicht teilen möchten, können von Ihrer Verwaltung eine anonyme Terminserie erstellen lassen (siehe auch: Mitigation des Nutzerakzeptanzrisiko). | Sehr hoch | Niedrig |
| Sicherheit | Die Applikation darf nicht angreifbar sein und insbesondere Nutzerdaten müssen mit höchster Sorgfalt behandelt werden. | Die Applikation setzt die neusten Sicherheitsstandards ein. | Hoch | Niedrig |
| Verfügbarkeit | Die Applikation ist höchstverfügbar. Die High Availability für User und Administrator:innen ist gegeben. | Die Applikation wird mit genügend Bandbreite betrieben, um den Usern und Administrator:innen höchste Verfügbarkeit zu gewährleisten. | Mittel | Niedrig |
| Technologieabhängigkeit | Die Applikation ist abhängig von Technologien wie Vue.js, Spring, und SQLite. | Wenn eine Inkompatibilität der Technologien festgestellt wird, werden diese durch kompatible, ähnliche Technologien ausgetauscht. | Niedrig | Niedrig |

| Risiko | Beschreibung | Mitigation | I | P |
|-----------------------|--|--|--------|-----------------------|
| Wissenslücken im Team | Im Team bestehen Wissenslücken für bestimmte Technologien. | Die Wissenslücken im Projektteam werden durch interne Schulungen und Wissensaustausch gefüllt. | Mittel | Vollständig mitigiert |

(I = Impact / Schweregrad des Eintritts; P = Probability / Wahrscheinlichkeit des Eintritts)