Разработчик

St. Vincenz-Hospital Abteilungsleitung

des Krankenhauses Softwaretechnik

Herr Prof. Dr. Med. Menne

Danziger Str. 17 33034 Brakel

Analyse- und Entwurfsdokument

Im Rahmen des Softwareunterrichts

Mittwoch, 20. November 2024

Verantwortlicher: Kai Topp

Team

Projektleiterin: Jacqueline S. Brzezinska

Mitglieder: Jason Janzen

Kai Topp

Marius Reiffer

1 Analyse

In der Analyse werden, die bereits im Pflichtenheft genannten Aufgabenbereiche der zu entwickelnde Software genauer untersucht. Zunächst wird die Softwarearchitektur, der sich die einzelnen Programmmodule bedienen sollen, festgelegt und die Konsequenzen der Benutzung dieser erläutert. Anschließend werden effiziente Design-Pattern vorgestellt, die sich in der Implementierungsphase als durchaus nützlich erweisen werden. Darauffolgend werden die Werkzeuge, mit denen die Softwareimplementierung umgesetzt werden soll, vorgestellt und genau dargelegt, warum diese sich für den Aufgabenbereich am besten eignen. Um einen visuellen Überblick über das Projekt zu erhalten, werden wir die Struktur einzelner Pakete in einem UML-Komponentendiagramm darstellen, sowie benötigte Klassen und ihre Beziehungen untereinander in einem UML-Analyseklassendiagramm festhalten.

1.1 Architektur

In diesem Kapitel wir die Softwarestruktur des Projekts skizziert. Dabei beschreibt eine Softwarearchitektur das Grundgerüst bzw. das Fundament, auf dem die einzelnen Klassen aufbauen, und zu funktionstüchtigen Komponenten werden.

1.1.1 Modularer Aufbau

Modularer Aufbau bedeutet, dass das Programm in einzelnen Modulen programmiert wird, die, nach dem sie erfolgreich getestet wurden, logisch verknüpft werden und so mehrfach wiederverwendet werden können. Da die einzelnen Module auf diese Weise nur an wenigen Punkten Abhängigkeiten aufweisen, wird die Wartung deutlich vereinfacht und das Programm kann ohne große Probleme erweitert werden. Wie im Pflichtenheft erläutert, besteht unser Projekt aus insgesamt sechs Bestandteilen, die sich entweder auf den Benutzer oder auf die Routenfindung beziehen. Für uns bedeutet das, dass z.B. Räume oder Benutzer einfach abgeändert oder hinzugefügt werden können und das Programm weiterhin wie erwartet funktioniert.

1.1.2 Model-View-Controller

Die Anwendung folgt dem MVC-Pattern, um eine klare Trennung zwischen Daten, Benutzeroberfläche und Steuerung zu gewährleisten.

• Models

Die **Model-Schicht** ist für die Datenhaltung und die zugrunde liegende Logik zuständig.

• View

Die **View-Schicht** präsentiert die Daten und stellt Interaktionsmöglichkeiten für den Benutzer bereit.

• Controller

Die **Controller-Schicht** vermittelt zwischen View und Model, verarbeitet Benutzereingaben und steuert die Anwendungslogik.

Wir haben uns bei der Benutzung des MVC-Patterns entschieden, das wir die Anwendungslogik klar von der dazugehörigen Darstellung und Benutzerinteraktionen trennen. Durch diese Verfeinerung können Komponenten im bestehenden System leicht erweiterbar oder wiederverwendet werden.

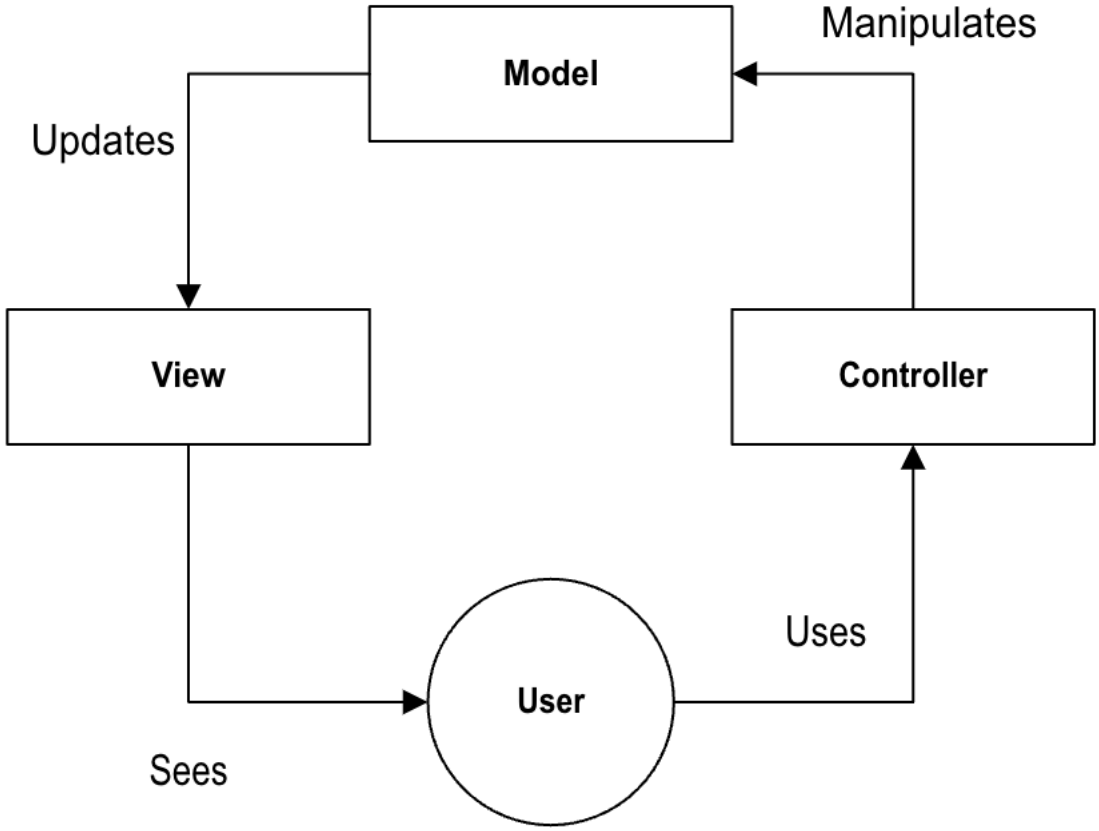

Abbildung 1: Model-View-Controller

1.2 Observer-Pattern

Das Observer-Pattern ist ein effizientes Design-Pattern aus der Softwareentwicklung und gehört zum Typ der Verhaltensmuster. Die damit gelöste Problematik bezieht sich auf die Kommunikation zwischen View und Model Klassen der dazugehörigen Architektur. Dabei benachrichtigt eine Model Klasse, sobald sie modifiziert wurde, die auf ihr registrierten Viewer, welche sich entsprechend aktualisieren. Dieses Pattern eignet sich besonders gut für die Wegfindungsapplikation, da die Karte auf Änderungen reagieren muss, die von der Route vorgenommen werden. In Abbildung 2 wird das beschriebene Verhalten noch einmal visualisiert.

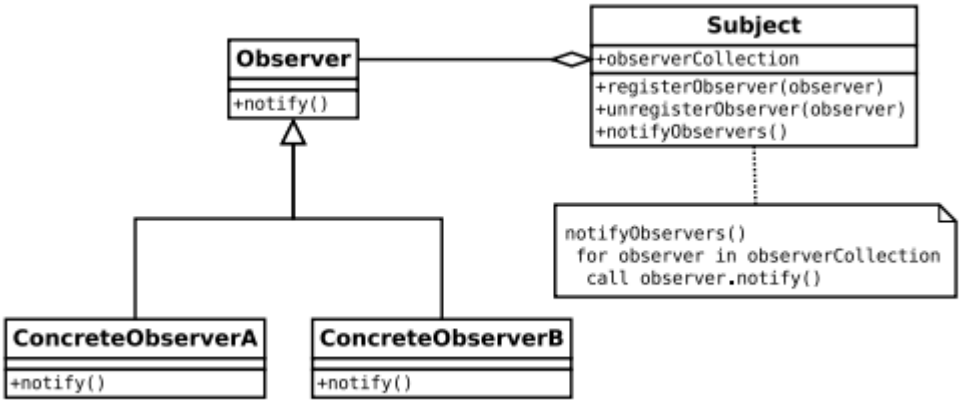

Abbildung 2: Observer-Pattern

https://upload.wikimedia.org/wikipedia/commons/8/8d/Observer.svg,

Abbildung 4 zeigt ein UML-Klassendiagramm des Observer-Patterns. Hier steht der Observer für die View Klassen, welche auf den Model Klassen also Subject registriert werden können. Es kann mehr als ein Observer gleichzeitig für ein Subject registriert sein. Diese werden beim Aufruf von notifyObserver () aufgefordert, die modifizierten Daten korrekt anzuzeigen bzw. sich zu aktualisieren.

1.3 Werkzeuge

Im weiteren Verlauf werden die Werkzeuge vorgestellt, die in Übereinstimmung mit dem Pflichtenheft für die Wegfindungsapplikation benutzt werden.

1.3.1 Eclipse

Eclipse ist eine Open-Source IDE, die für Java entwickelt wurde. Wir werden innerhalb von Eclipse den WindowBuilder für die Gestaltung der Benutzeroberfläche verwenden. Für Java benutzen wir das Java Development Kit (JDK) Version 1.8 und die dazugehörige Java Runtime Environment (JRE) Version 1.8 als Laufzeit-Umgebung. Arbeiten werden wir unter der Eclipse-Version: 2020-09 (4.17.0).

1.3.2 Windows

Windows 10 ist ein weit verbreitetes Betriebssystem von Microsoft, das eine stabile und benutzerfreundliche Entwicklungsumgebung bietet. Für die Entwicklung eines Java-basierten Programms verwenden wir die **Eclipse IDE für Java Developers** und setzen **Java 1.8 oder höher** als Programmiersprache ein. Die Benutzeroberfläche wird mit **Java WindowBuilder** erstellt, um eine moderne und interaktive Anwendung zu ermöglichen.

1.3.3 SWT / WindowBuilder

Wir verwenden das Standard Widget Toolkit (SWT) das im WindowBuilder ist, um die Grafische Oberfläche des Programms zu erstellen. SWT verwendet die nativen grafischen

Elemente des Betriebssystems, um reaktionsschnell und kompakt zu bleiben und eignet sich gut für die einfache Oberfläche des Programms.

1.4 Analyse-Klassendiagramm

Im Folgenden werden die Aufgaben der Klassen erläutert.

1.4.1 Boundary

1.4.1.1 AdminGUI

In der AdminView können Benutzer andere Benutzer hinzufügen sowie Routen und Räume Sperren aber auch Raumnummern ändern.

1.4.1.2 MapGUI

In der MapGUI kann man den Mitarbeiter Login Button sowie die Karte und das Eingabefeld für die Räume sehen.

1.4.1.3 LoginGUI

In der LoginGUI können sich Benutzer bei dem Terminal anmelden.

1.4.2 Control

1.4.2.1 Routecal

Die Model Klasse Routecal berechnet die Route, die dann an die Karte also View Klasse gesendet wird.

1.4.2.2 Logincal

Der Logincal sorgt dafür das sich Benutzer an dem Terminal anmelden können.

1.4.2.3 Admincal

Der Admincal verändert die GUIs, die alle Benutzer betreffen.

1.4.2.4 Mapdata

In der Mapdata wird der aktuelle Stand der Karte gespeichert und aktualisiert.

1.4.3 Entity

1.4.3.1 User

Ein User ist ein Rollen Basierter Account der mit Nutzername und Password bestückt ist.

2 Entwurf

Im Abschnitt Entwurf soll die Struktur der in der Analyse eingeführten Komponenten näher erläutert und deren Abläufe und Zusammenhänge mit Hilfe von Sequenzdiagrammen detailliert veranschaulicht werden. Dabei werden insbesondere die technischen Komponenten

einbezogen, welche für die Lösung der auf den Entwurf folgenden Implementierung von Relevanz sein.

2.1 Qualitätskriterien

Bei der Implementierung aller Komponenten wird besonders wird Großer Wert auf die Wartbarkeit, Korrektheit und Benutzerfreundlichkeit gelegt. Im Weiteren werden die drei Themen genauer erläutert.

2.1.1 Wartbarkeit

Die Software wird so konzipiert, dass sie leicht zu warten und anzupassen ist, um zukünftige Änderungen effizient umsetzen zu können. Dazu wird ein modularer Aufbau nach dem MVC-Pattern verwendet, wodurch die Trennung zwischen Logik, Darstellung und Steuerung klare Verantwortlichkeiten schafft. Änderungen in einer Schicht, wie beispielsweise die Anpassung des Kartenlayouts im Model, wirken sich minimal auf andere Schichten aus. Der Quellcode wird umfassend dokumentiert, und zwar im Javadoc-Format, damit Entwickler die Funktionen und Änderungen nachvollziehen können. Ergänzend dazu werden alle Designentscheidungen und die Architektur der Software in einem separaten Entwurfsdokument festgehalten.

2.1.2 Korrektheit

Die Applikation wird so entwickelt, dass sie die Anforderungen fehlerfrei erfüllt und zuverlässig funktioniert. Dies wird durch automatisierte Tests sichergestellt, die jede Hauptkomponente der Anwendung prüfen. Zum Beispiel wird die Routenberechnung im Model und die Berechtigungsprüfung im UserModel in isolierten Unit-Tests validiert. Integrationstests stellen sicher, dass die Zusammenarbeit zwischen den verschiedenen Schichten und Komponenten reibungslos abläuft. Zusätzlich zu den automatisierten Tests werden manuelle Tests durchgeführt, um reale Anwendungsfälle zu überprüfen. Beispielsweise wird getestet, ob die Applikation gesperrte Bereiche korrekt vermeidet oder Admin-Funktionen wie das Umbenennen von Räumen ordnungsgemäß funktionieren. Um mögliche Fehler im Betrieb zu behandeln, zeigt die Applikation klare und verständliche Fehlermeldungen in der Benutzeroberfläche an, etwa „Keine Route gefunden“. Ungültige Eingaben werden abgefangen, damit die Anwendung nicht abstürzt.

2.1.3 Benutzerfreundlichkeit

Die Anwendung wird so gestaltet, dass sie für technisch unerfahrene Nutzer einfach und intuitiv bedienbar ist. Die Benutzeroberfläche wird übersichtlich gestaltet, insbesondere die Kartenansicht, die Start- und Zielpunkte sowie die berechnete Route klar hervorhebt. Interaktive Funktionen wie das Setzen von Start- und Zielpunkten direkt auf der Karte oder die visuelle Auswahl von gesperrten Bereichen in der Administratorebene erleichtern die Bedienung zusätzlich. Alle Elemente der Oberfläche, wie Schriftgrößen und Symbole, sind so gestaltet, dass sie gut lesbar und verständlich sind. Die Applikation gibt hilfreiche Rückmeldungen, beispielsweise wenn ein Zielpunkt fehlt, und verhindert so unnötige Frustrationen.

2.2.1 Entwurfsklassendiagramm

Das System für die Wegfindung im Krankenhaus basiert auf dem MVC-Pattern, das die Architektur in drei Hauptkomponenten unterteilt.

2.2.2 Die Datenhaltungsschicht com.wegapplikation.config.model

Das Paket com.wegapplikation.model dient dazu, die Einstellung der Wegfindung zu speichern. Dazu gehören die Werte der Klassen Roomdata und Userdata. Diese können bei Bedarf abgerufen oder modifiziert werden.

2.2.3 Die Programmsteuerung com.wegapplikation.config.controller

In diesem Paket com.wegapplikation.config.controller ist die Steuerung der Aktionen des Programmes. Das Paket enthält die Klassen mapdata, routecal, logincal und admincal, welche die Berechnungen und Datenverwaltungen übernehmen und weiterleiten.

2.2.4 Die Präsentationschicht com.wegapplikation.config.view

Das Paket com.wegapplikation.config.view enthält alle Oberflächen, die in dem Programm beteiligt sind. Dazu gehören die mapGUI und loginGui und adminGUI.

2.2.4.1 MapGui

In der mapGui können Benutzer Star und Zielpunkt auswählen und sich eine Route berechnen lassen.

2.2.4.2 LoginGui

In der loginGui können sich Mitarbeiter oder Administratoren anmelden um spezielle funktionen zu erhalten.

2.2.4.3 AdminGui

Die AdminGui ermöglicht es Administratoren, Änderungen an Räumen oder Gängen usw. Vorzunehmen.

In Abbildung 3 ist ein Klassendiagramm angegeben, welches die Struktur der Wegfindung Applikation beschreibt. Nähere Erläuterung schließen sich in den darauffolgenden Absätzen an.

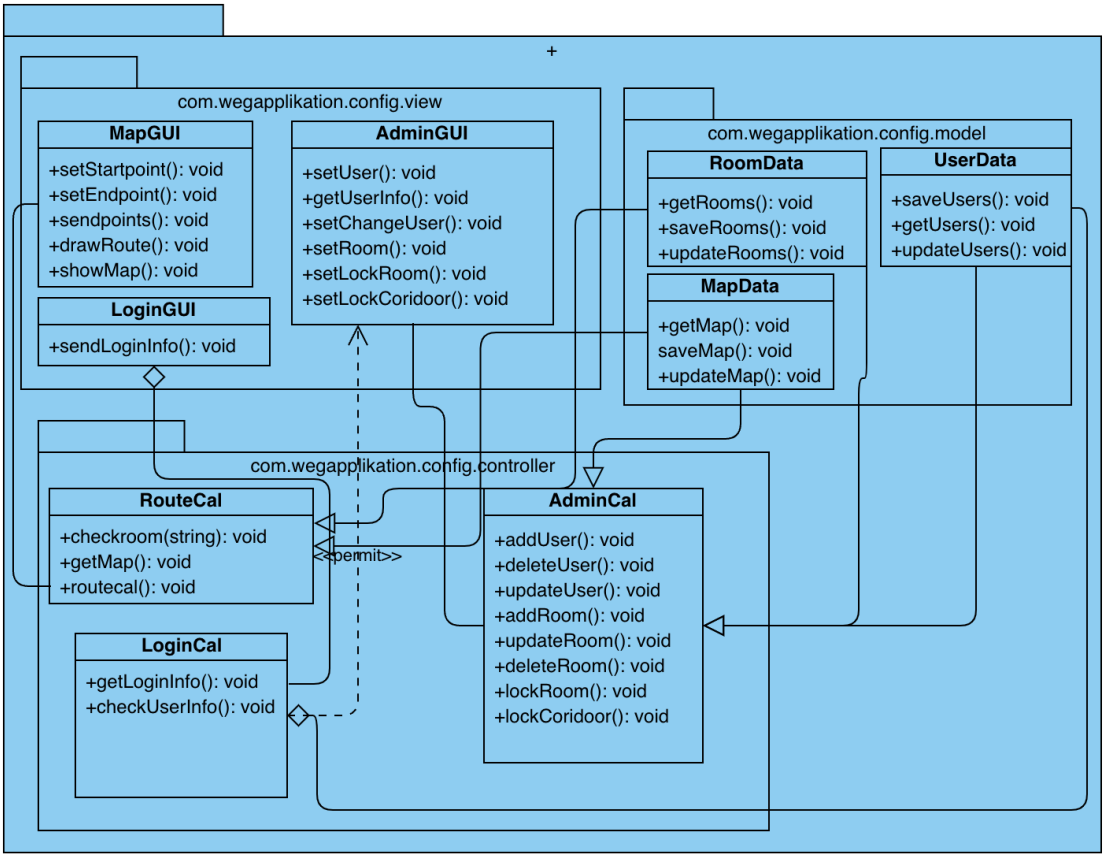

Abbildung 3: Entwurfsklassendiagramm

2.3 Sequenzdiagramm

Im folgenden Unterkapitel wird über UML-Sequenzdiagrammen die Kommunikation zwischen den verschiedenen Klassen bei jeder Aktivität dargestellt. Wichtig sind hier, die Nachrichten, deren Ablauf im System und den Klassen, auf denen sie ausgeführt werden. Klassen werden als ein großer Kreis dargestellt, Nachrichten sind als Pfeile mit den Funktionsnamen dargestellt. Die vertikal gestrichelten Linien zeigen den Ablauf und wann sie aktiv sind, erkennbar wenn diese in dicke Balken wechseln. Nachrichten werden von einer Klasse oder einem Nutzer zu einer Klasse gesendet.

2.3.1 Route erstellen

Das Sequenzdiagramm “create route” (Abbildung 4) zeigt das Erstellen einer Route über das Terminal. Nach dem Eingeben von dem Startpunkt wird er mit der Karte verglichen, wenn der Startpunkt existiert, wird er auf der Karte angezeigt, sonst wird eine Fehlermeldung aufgerufen. Danach wird beim Angeben des Zielpunktes derselbe Ablauf ausgelöst wie beim Startpunkt, nur das mit allen Räumen verglichen wird. Wurden die Daten gesendet, wird mit den beiden Punkten die Route berechnet und zur GUI übergeben, auf der sie dann angezeigt wird.

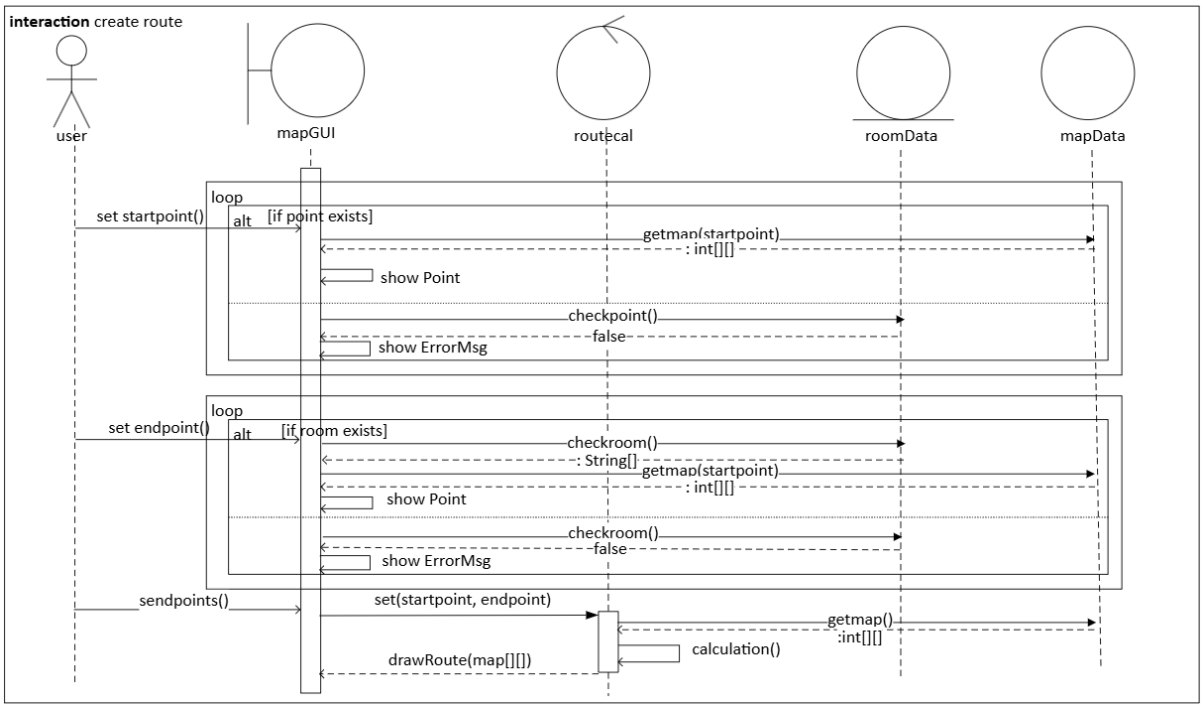

Abbildung 4: Sequenzdiagramm ”create route”

2.3.2 Raumnummer ändern

In Abbildung 5 “change roominformation” wird dargestellt, wie eine Raumnummer geändert werden kann. Der Administrator gibt zuerst die momentane Raumnummer an, wenn diese nicht existiert, wird eine Pop-Up Fehlermeldung ausgegeben. Nach einer existierenden Eingabe wird als nächstes die neue Raumnummer eingegeben, welche nach dem Übergeben “send change” die Namenskonventionen überprüft. Bei nicht eingehaltenen Konventionen wird ein Fehler ausgegeben und die Raumnummer bleibt. Sonst wird die Nummer geändert mit einem anschließenden Pop-Up Fenster zur Bestätigung.

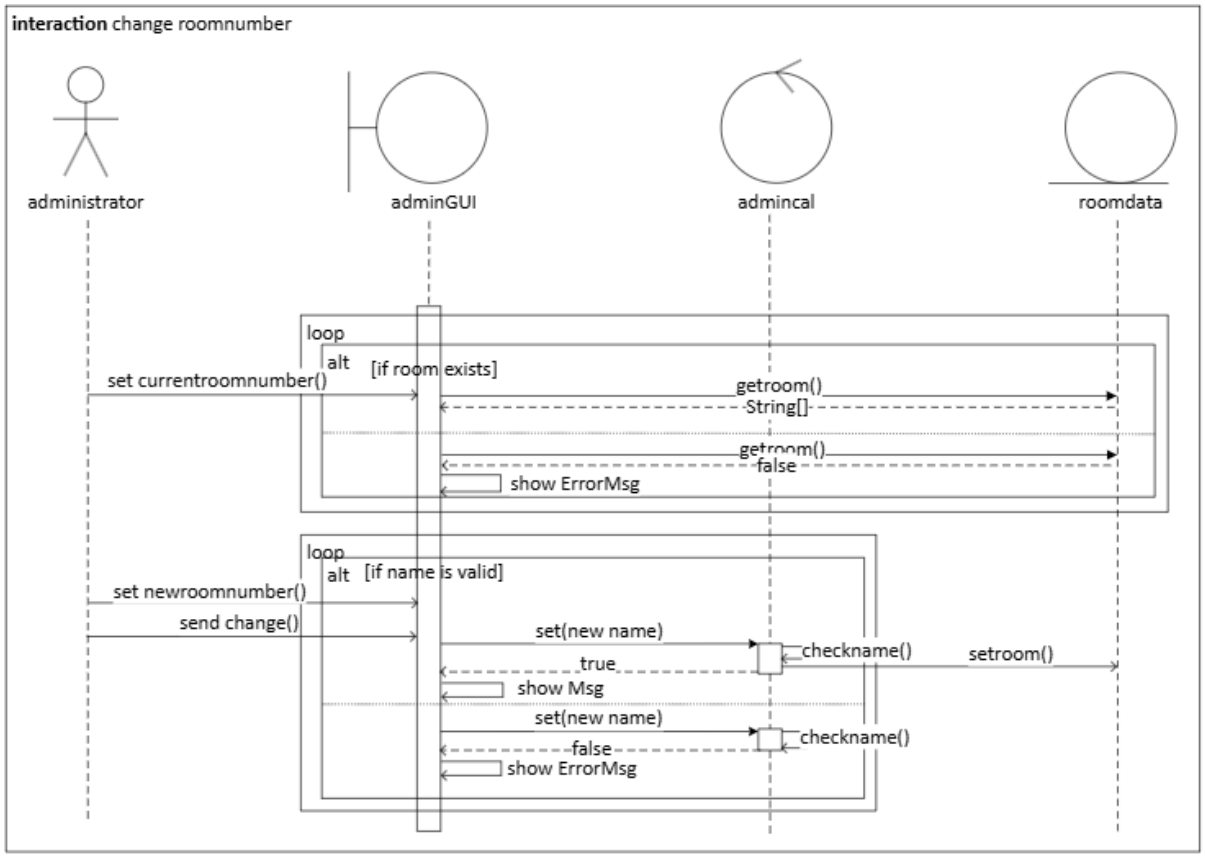

Abbildung 5: Sequenzdiagramm ”change roomnumber”

2.3.2 Nutzerinformation ändern

Das Sequenzdiagramm “change userinformation” (Abbildung 6) zeigt den Ablauf den ein Administrator beim Ändern eines Nutzers durchläuft. Dafür muss der momentane Name des Nutzers eingegeben werden. Wenn der Nutzer nicht als ein Eintrag vorhanden ist, erscheint eine Fehlermeldung. Als nächstes muss der Administrator einen neuen Nutzernamen oder einen neuen Pin für den Nutzer setzen, wobei gleich beides gemacht werden kann. Die neuen Daten werden über “send change” aktualisiert und die Änderung wird durch ein Pop-Up bestätigt. Wenn der Name oder der Pin nicht zu den Konventionen passen, wird stattdessen ein Fehler ausgegeben.

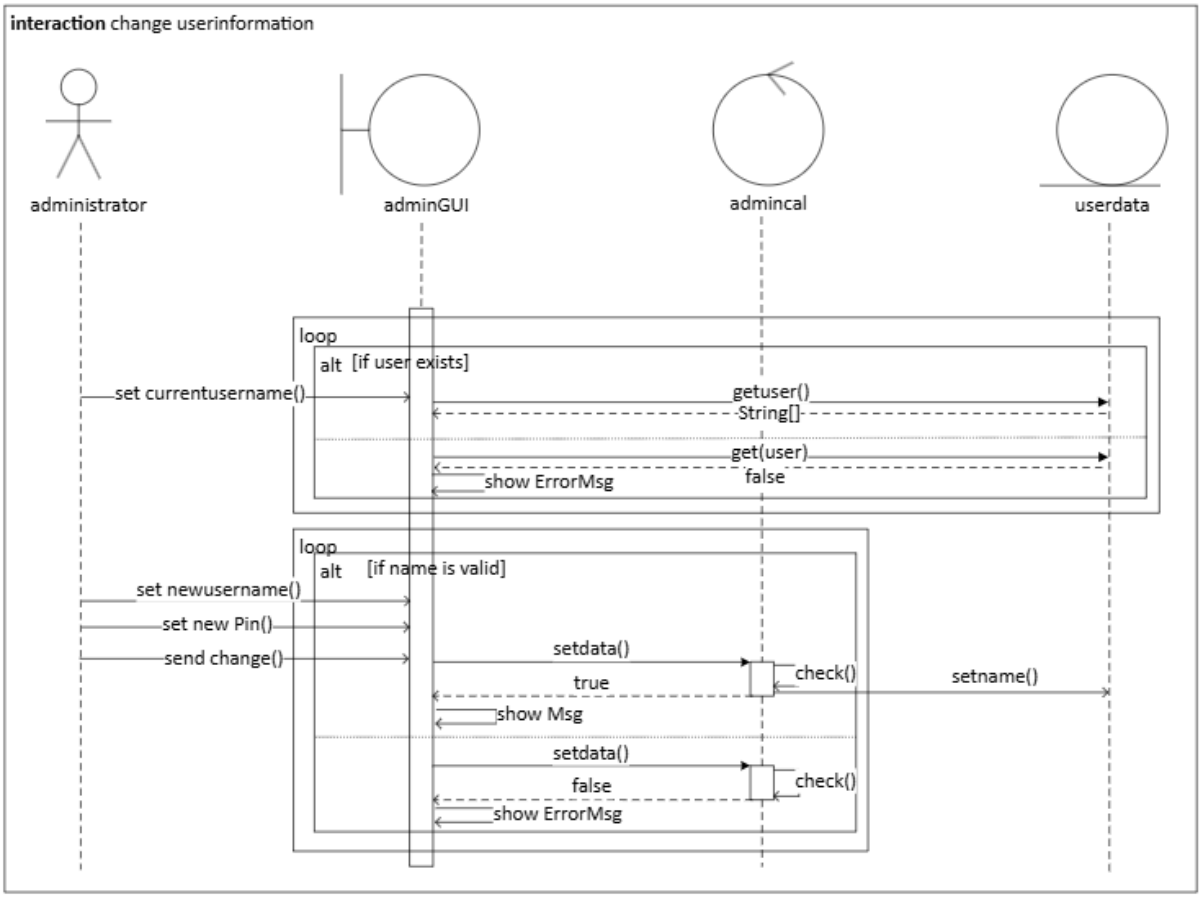

Abbildung 6: Sequenzdiagramm ”change userinformation”

2.3.3 Nutzer hinzufügen

Das Sequenzdiagramm “add user” (Abbildung 7) zeigt die Erstellung eines neuen Benutzereintrags. Zuerst muss der Administrator einen Namen, Pin und Rolle eingeben und übergeben. Nach der Übergabe wird verglichen, ob dieser Nutzer schon existiert, wenn nicht wird ein neuer Eintrag erstellt und ein Pop-Up geöffnet. Falls ein Eintrag schon existiert, wird eine Fehlermeldung ausgegeben.

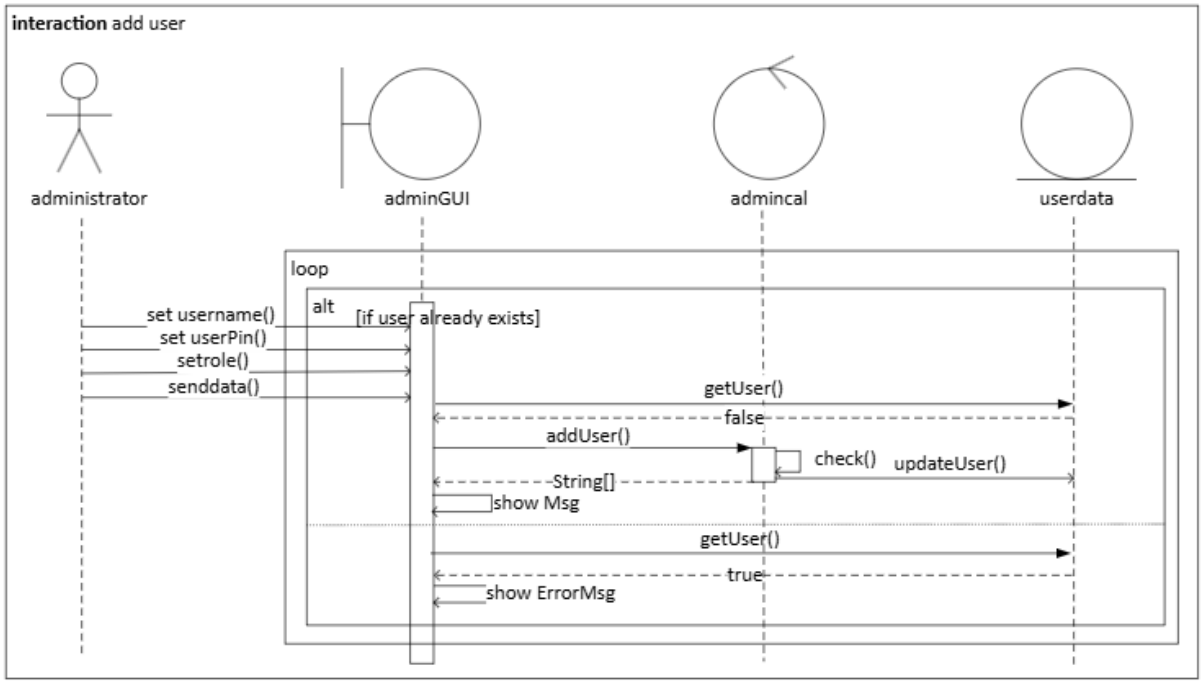

Abbildung 7: Sequenzdiagramm ”add user”

2.3.4 Nutzer entfernen

Die Abbildung 8 “remove user” stellt dar, wie ein Administrator einen Nutzer entfernt. Als erstes wird der Nutzername übergeben. Dann wird die Eingabe verglichen mit den Nutzereinträgen. Wenn der eingegebene Nutzer existiert, wird er entfernt, falls nicht dann wird eine Pop-Up Fehlermeldung geöffnet.

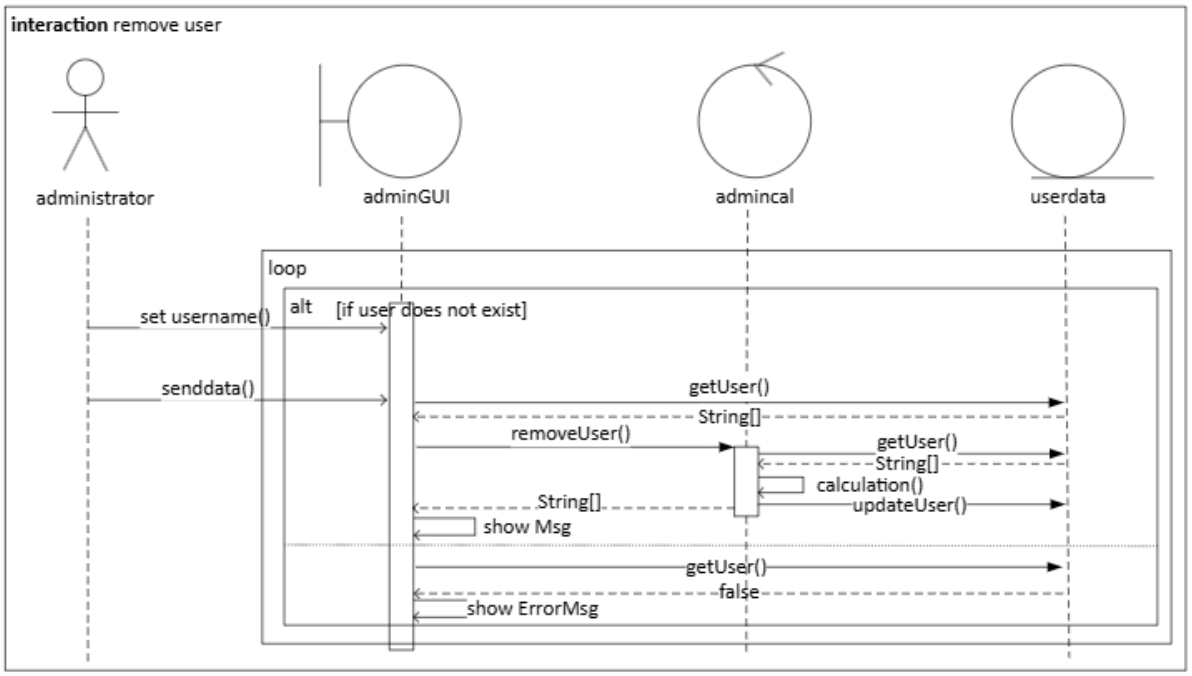

Abbildung 8: Sequenzdiagramm ”remove user”

2.3.5 Anmelden

Das Sequenzdiagramm “login” (Abbildung 9) zeigt wie ein Nutzer sich anmelden kann. Zuerst ist der Nutzer aufgefordert einen Namen und Pin einzugeben. Nach der Abgabe werden der Name und Pin mit Einträgen aus der Klasse “userData” verglichen. Wenn kein Eintrag existiert, wird eine Fehlernachricht erscheinen. Bei einem existenten Eintrag wird die Rolle verglichen. Ist die Rolle “Administrator” wird die “adminGUI” geöffnet, sonst wird die “mapGUI” geöffnet.

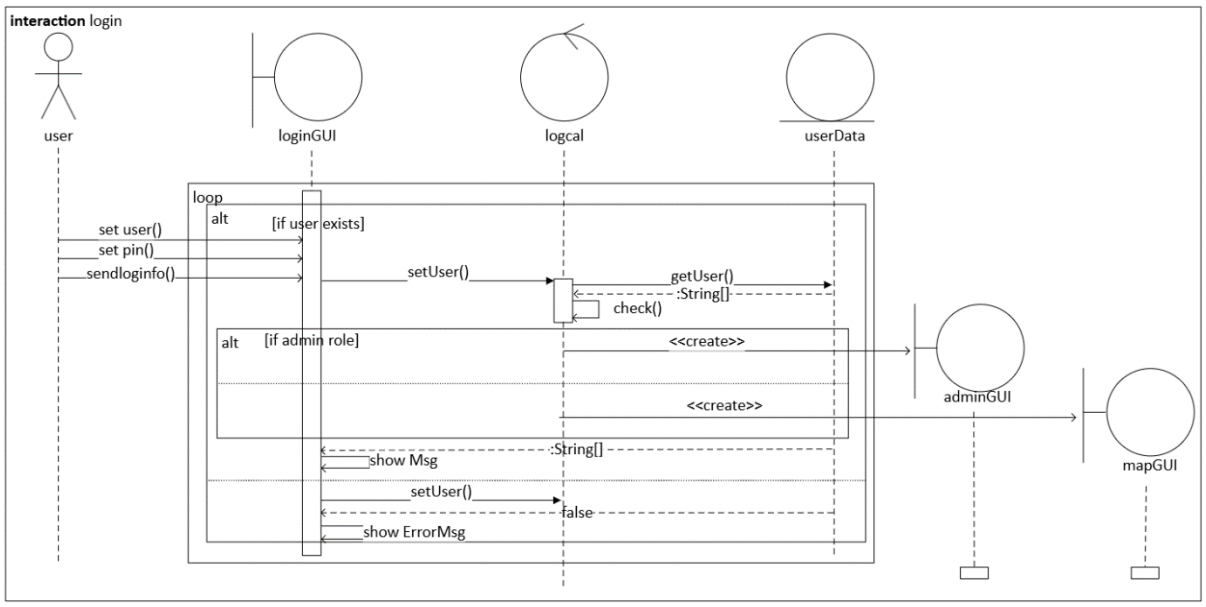

Abbildung 9: Sequenzdiagramm ”login”

2.3.6 Wege blockieren/Raumtür ändern

In der Abbildung 10 “block passage/roomdoor” zeigt den Ablauf für das Schließen von Gängen und das Umstellen von Raumtüren bzw. Wänden. Zuerst muss der Administrator den Modus wechseln zwischen blockiere Weg und ändere Raum Wand, welcher sofort umgestellt wird. Danach kann der Admin eine Fläche markieren, die nach Modus Wege oder Wände sind. Markierte Wege und Wände werden auf der Karte angezeigt. Als letztes muss der Administrator das Markierte akzeptieren, wodurch es auf der Karte geändert und für die GUI aktualisiert wird.

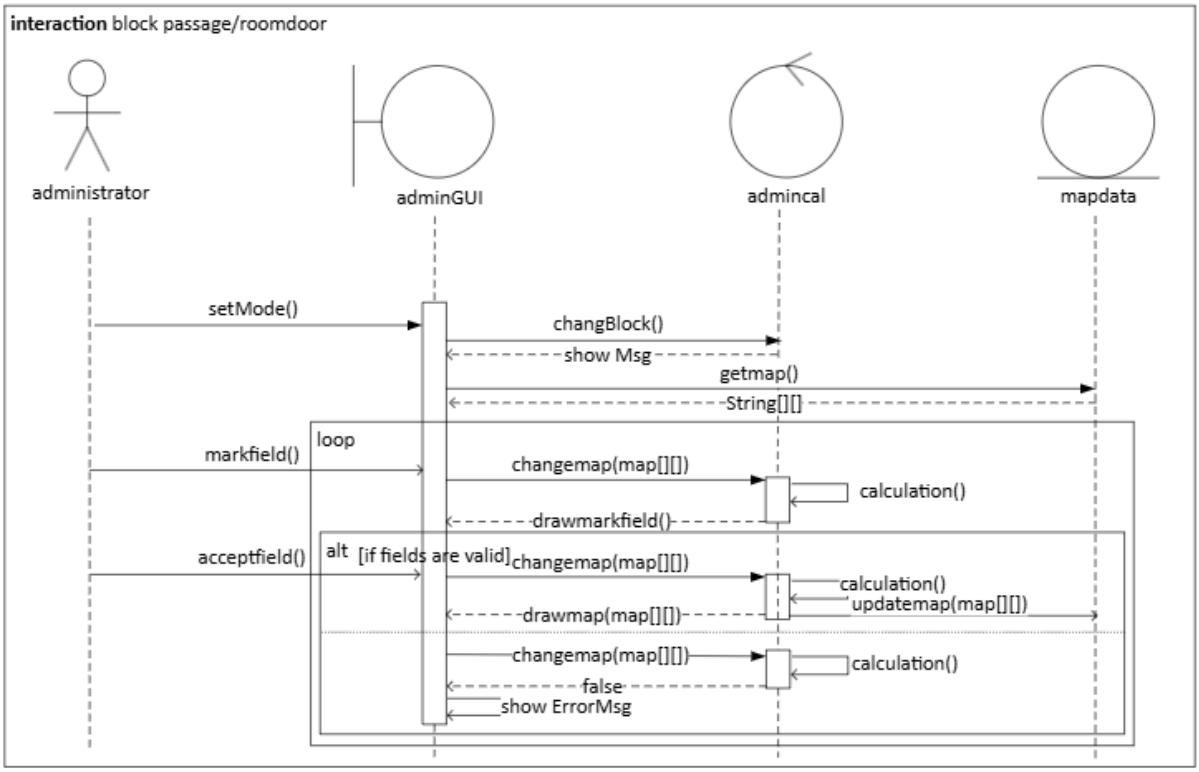

Abbildung 10: Sequenzdiagramm ”block passage/roomdoor