

PROJEKT DOKUMENTATION



Smart Room Anwendung

Team 1

Ilker Akceylan (K01627088)
Gabriel Mohamed-Hofmayr (K11802871)
Florian Ecker-Eckhofen (K11914711)

Institut für Wirtschaftsinformatik –
SE Data & Knowledge Engineering



Beurteiler:
Dr. Johannes Sametinger
DI Daniel Lehner

Table of Contents

1. Introduction.....	3
2. Management Perspective	3
3. User Perspective	6
4. Developer Perspective.....	10
4.1. Design	10
4.1.1. Overview of the System.....	10
4.1.2. Important Design Decisions	11
4.2. Implementation.....	11
4.3. Code Quality.....	15
4.4. Testing	16
5. Installation guide.....	20

Version history

Version	Date	Creator	Changes
1	14.01.2023	Ilker Akceylan	Introduction
2	16.01.2023	Ilker Akceylan	Introduction
3	07.02.2023	Ilker Akceylan	User Perspective
4	08.02.2023	Ilker Akceylan	Implementation, Management Perspective
5	10.02.2023	Ilker Akceylan	User Perspective, Implementation, Developer Perspective, Implementation, Code Quality
5	10.02.2023	Florian Ecker-Eckhofen	Testing
5	10.02.2023	Gabriel Mohamed-Hofmayr	Management Perspective, Installation Guide

Table 1: Version history

1. Introduction

Ziel dieses Projektes ist es einen Digitalen Zwilling eines Raumes inklusive der Objekte wie Fenster, Türen, Lichter und Ventilatoren darzustellen. Dem Benutzer soll durch eine Applikation ermöglicht werden, diese Räume selbstständig und individuell zu erstellen. Dadurch kann der User für einen Raum eine eindeutige ID, einen Raumnamen, die Personenzahl des Raums und jeweils für die verschiedenen Objekte, die Anzahl der vorhandenen Objekte in einem Raum festlegen.

Diese Applikation bringt aber auch einige nützliche Funktionen mit sich. Nach Anlegen eines Raumes, wird diese in einer Liste dargestellt sowie auch in der Datenbank gespeichert. Der Benutzer hat hier die Möglichkeit, die erstellten Räume zu bearbeiten bzw. zu löschen. Die Applikation bietet auch die Möglichkeit, dass der User die Liste der Räume samt den Objekten als eine Excel-Datei exportiert.

Andersherum ist es auch möglich, dass der User in Form von einer Excel-Datei eine vorhandene Liste der Räume importiert. Mit dieser Applikation ist es dem User gestattet, die einzelnen Objekte eines Raumes per Remote Control zu kontrollieren. Dadurch können Objekte wie beispielsweise Fenster geöffnet und geschlossen werden. Weitere Vorteile dieser Applikation besteht darin, dass sämtliche Security- und Alarmfunktionen mit sich bringt.

Dank den erstellten Charts, hat hier der Benutzer die Möglichkeit, die Raum-Temperaturen, Anzahl der Personen in einem Raum und die CO2-Werte innerhalb der Räume, als graphische Darstellungen zu beobachten, welches die Übersicht deutlich angenehmer macht.

Durch dieses Dokument sollen die Leserinnen und Leser eine Übersicht über den Aufbau des Projektes, implementierte bzw. nicht implementierte Funktionen und die Zuständigkeiten der Entwickler und deren Aufwand für das Projekt bekommen. Darüber hinaus befindet sich in diesem Dokument eine Installationsguide und eine Benutzeranleitung der Applikation für die daran interessierten Leserinnen und Leser, die eventuell diese Applikation benutzen wollen.

Auch die Designentscheidung der Entwickler, die Qualität des Codes und das Testen des Quellcodes sind ein Teil des Dokumentes.

2. Management Perspective

Basic: Import data from CSV to an Entity-Relationship database

- Gabriel (17h)

Basic: Create/Update Rooms (id, size, available doors, windows, lights and fans)

- Ilker (7h)
- Florian (15h)
- Gabriel (8h)

Basic: Update and remove rooms

- Florian (12h)
- Illker (3,5)
- Gabriel (6h)

Basic: Visualize available rooms

- Ilker (3,5h)
- Basic:** Visualize static info. for each room (id, size, available doors, windows, lights and fans)
 - Gabriel (2h)
 - Ilker (2h)
 - Florian (8h)
- Basic:** Develop a line chart that shows real-time data regarding light/fan/window/door
 - Gabriel (3h)
 - Ilker (3h)

➔ *Befüllen der Charts konnte nicht rechtzeitig umgesetzt werden da der Hauptverantwortliche für die Logging Daten der Room Control (Florian Ecker-Eckhofen) diese erst am Tag der Abgabe liefern konnte. Anpassung der bestehenden Methoden zum Befüllen der Charts konnten nicht mehr rechtzeitig von Gabriel oder Ilker umgesetzt werden*
- Basic:** Develop a line chart that shows the co2/temp. v. and the NoP for each room over time
 - Ilker (4h)
 - Gabriel (10h)
- Basic:** Save rooms structure (rooms + static information) in a .csv file
 - Ilker (6h)
- Basic:** Automat. add random values of co2/temperature/number of people for a specific room
 - Ilker (4h)
 - Gabriel (5h)
- Basic:** Live update of visualizations for co2, temperature, and lights/ventilators/windows/doors status for each room
 - Ilker (3h)
 - Gabriel (3h)
 - Florian(6h)
- Remote Control:** Allow to lock/unlock doors via the user interface
 - Florian(7h)
- Remote Control:** Allow to turn on/off lights via the user interface
 - Florian(7h)
- Remote Control:** Allow to open/close windows via the user interface
 - Florian(7h)
- Remote Control:** Allow to turn on/off fans via the user interface
 - Florian(7h)
- Security:** Send alarm if temperature is above 70 degrees Celsius
 - Ilker (2h)
 - Gabriel (2h)
- Security:** Unlock all doors if temperature is above 70 degrees Celsius
 - Ilker (2h)
 - Gabriel (2h)
- Energy Saving:** Turn lights on if there are people in the room
 - Ilker (1h)
 - Gabriel (1h)
- Energy Saving:** Lights should be turned off if the room is empty

- Ilker (1h)
- Gabriel (1h)

Energy Saving: Turn off running devices if the room is empty

- Ilker (1h)
- Gabriel (1h)

Air Quality: Open window + activate fan if co2 values are > 1000 parts per million (ppm)

- Ilker (2h)
- Gabriel (2h)

Air Quality: Change room color in user interface based on co2 values.

- Ilker (2h)
- Gabriel (2h)

Gesamtaufwand für das Projekt:

Ilker: 179h

Gabriel: 172h

Florian: 191h

3. User Perspective

Die Anforderungen für die Benutzeroberfläche wurde mit dem Tool „Scene Builder“ umgesetzt, welches mit dem JavaFX-Ökosystem zusammenarbeitet. Es ist ein visuelles-Tool und ermöglicht den Benutzern eine JavaFX-Benutzeroberfläche ohne Programmieraufwand zu erstellen. Durch ein einfaches Drag & Drop können Komponenten erstellt und ihre Eigenschaften geändert werden. Es wird automatisch im Hintergrund der FXML-Code für das Layout erzeugt. Danach wird eine FXML-Datei mit einem JAVA-Projekt verknüpft.

Szenario 1: Anlegen eines Raumes inklusive der Objekte, exportieren und importieren von Dateien.

Nach Start der Applikation erscheint die Hauptseite der Anwendung. Am linken Rand der Anwendung befindet sich unter dem Punkt „Room-Creator“ die statischen Informationen zu einem Raum, die der Benutzer befüllen muss. Nachdem alle Textfelder der Objekte ausgefüllt wurden, kann der Benutzer durch einen Klick auf das Button „Add Room“ einen Raum erstellen, welches dann sofort in der Liste („List of added rooms“) angelegt und angezeigt wird.

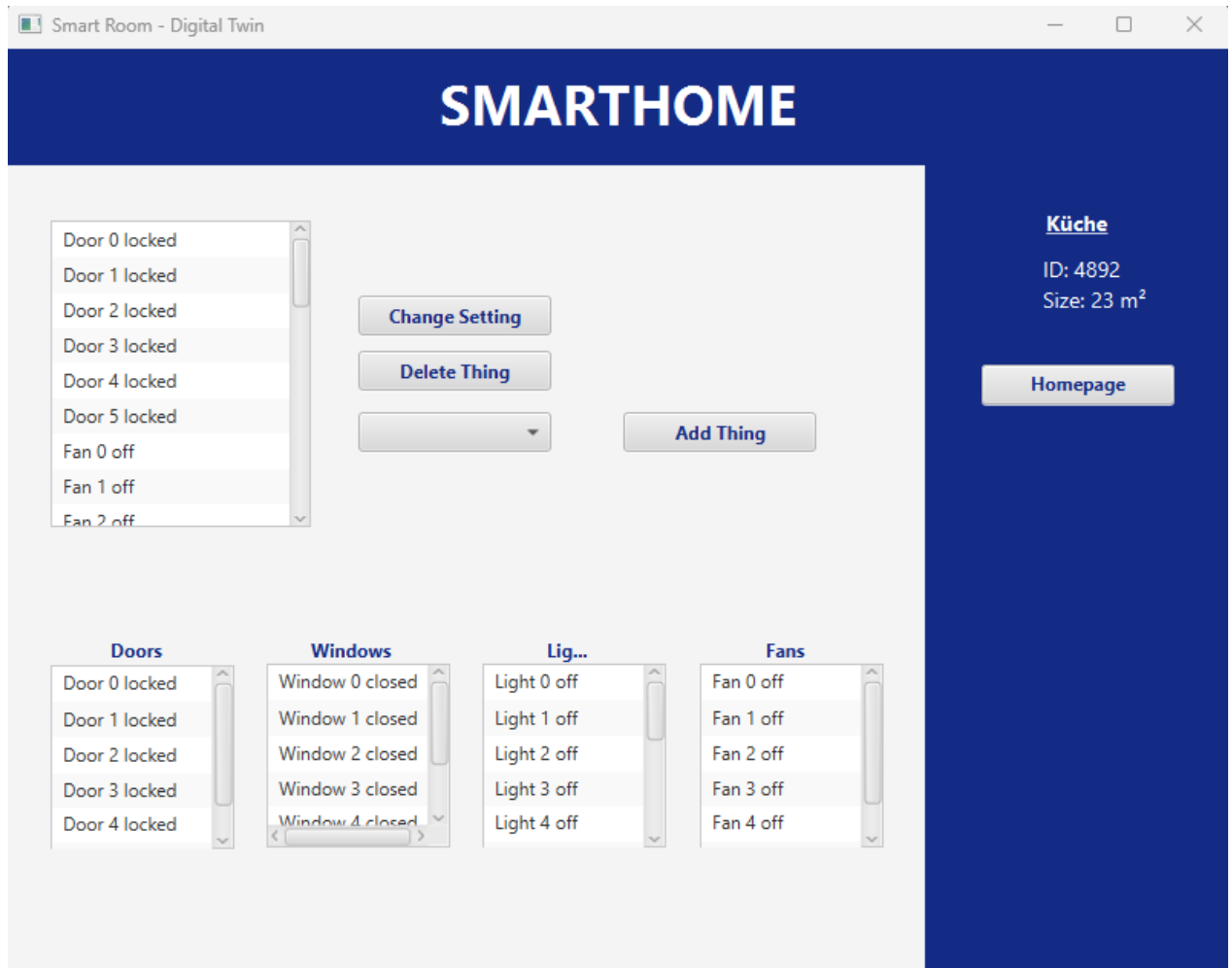
Am rechten Rand der Anwendung befinden sich weitere Funktionen.

Löschen: Der Benutzer kann je nach Wahl ein Raum aus der Liste auswählen und diese mit Hilfe eines „Delete Room“-Button, das ausgewählte Objekt von der Liste entfernen lassen.

Export: Räume, die sich in der Liste befinden, können durch das Button „Export Data“ als eine Excel-Datei exportiert werden. Darin befinden sich alle statischen Informationen zu einem Raum, die, während dem Anlegen eines Raumes gespeichert wurden.

Import: Vorhandene Excel-Dateien, welche statische Informationen eines Raumes beinhalten, können durch einen Klick auf das Button „Import Data“ in die Liste importiert werden.

Szenario 2: Bearbeitung des zuvor angelegten Objekts eines Raumes.



Damit der Benutzer auf diese Ansicht der Applikation landet, muss dieser zuerst auf der Hauptseite aus der Liste einen Raum auswählen und danach das Button „Room Info/Control“ anklicken.

Nun sieht der Benutzer auf der linken Seite der Ansicht eine Liste mit Objekten eines Raumes.

Status: Der User kann ein Objekt aus der Liste auswählen und dessen Status mit der Funktion „Change Setting“ beliebig oft ändern.

Löschen/Hinzufügen: Falls sich der User vorhin beim Anlegen eines Raumes vertippt hat, kann dieser aus der Liste ausgewählt und durch einen Klick auf „Delete Thing“-Button von der Liste entfernt werden. Aber es können auch Objekte hinzugefügt werden, indem die Funktion „Choice Box“ und „Add Thing“-Button gemeinsam benutzt werden. Hier muss der Benutzer dafür die

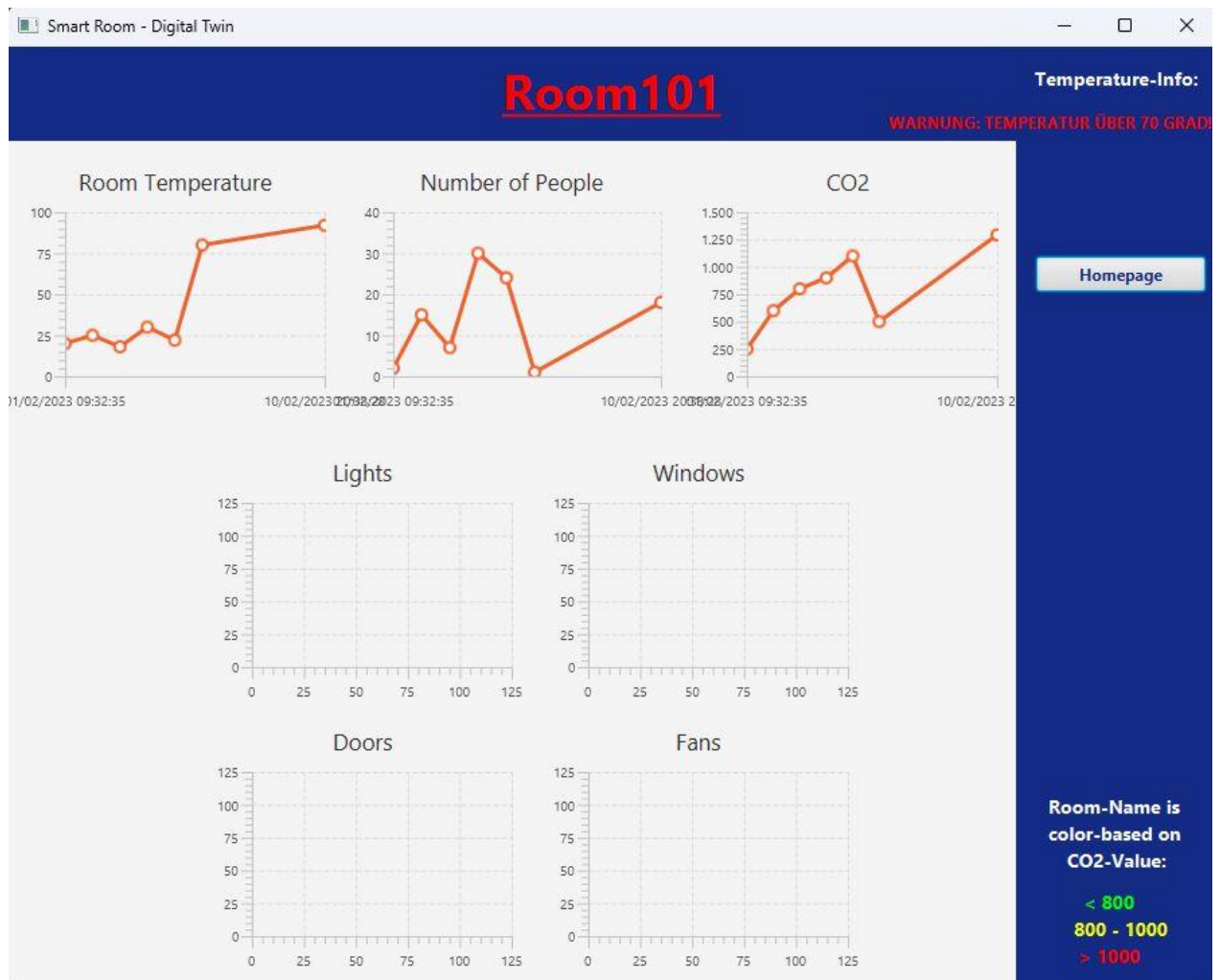
Choice Box, welche die 4 unterschiedlichen Objekte beinhaltet, anklicken, ein Objekt nach Wunsch auswählen und letztlich auf das Button „Add Thing“ klicken. Das hinzugefügte Objekt wird sofort in der Liste, wo sich alle Objekte zu einem Raum befinden, hinzugefügt.

Damit man die Übersicht nicht verliert, falls viele Objekte angelegt wurden, wurde unten auf der Ansicht 4 Listen mit den jeweiligen Objektnamen erstellt.

Auf der rechten Seite der Ansicht wird dem Benutzer der Name des Raumes angezeigt, indem sich der Benutzer derzeit befindet. Gleich darunter befindet sich ein „Homepage“-Button, mit dem man auf die Hauptseite gelangt.

Szenario 3: Anzeigen der Charts zu den Objekten und zusätzlich die Raum-Temperatur, CO2-Wert des Raumes und die Anzahl der Personen in einem Raum, des ausgewählten Raumes.

Damit der Benutzer auf diese Ansicht der Applikation landet, muss dieser zuerst auf der Hauptseite aus der Liste einen Raum auswählen und danach das Button „Room Charts“ anklicken.



Auf dieser Ansicht sind zu den 4 Objekten (Licht, Fenster, Tür, Ventilator) und zur Raum-Temperatur, CO2-Wert und Anzahl der Personen in einem Raum je ein Chart zu sehen. Diese werden als „Line-Charts“ dargestellt.

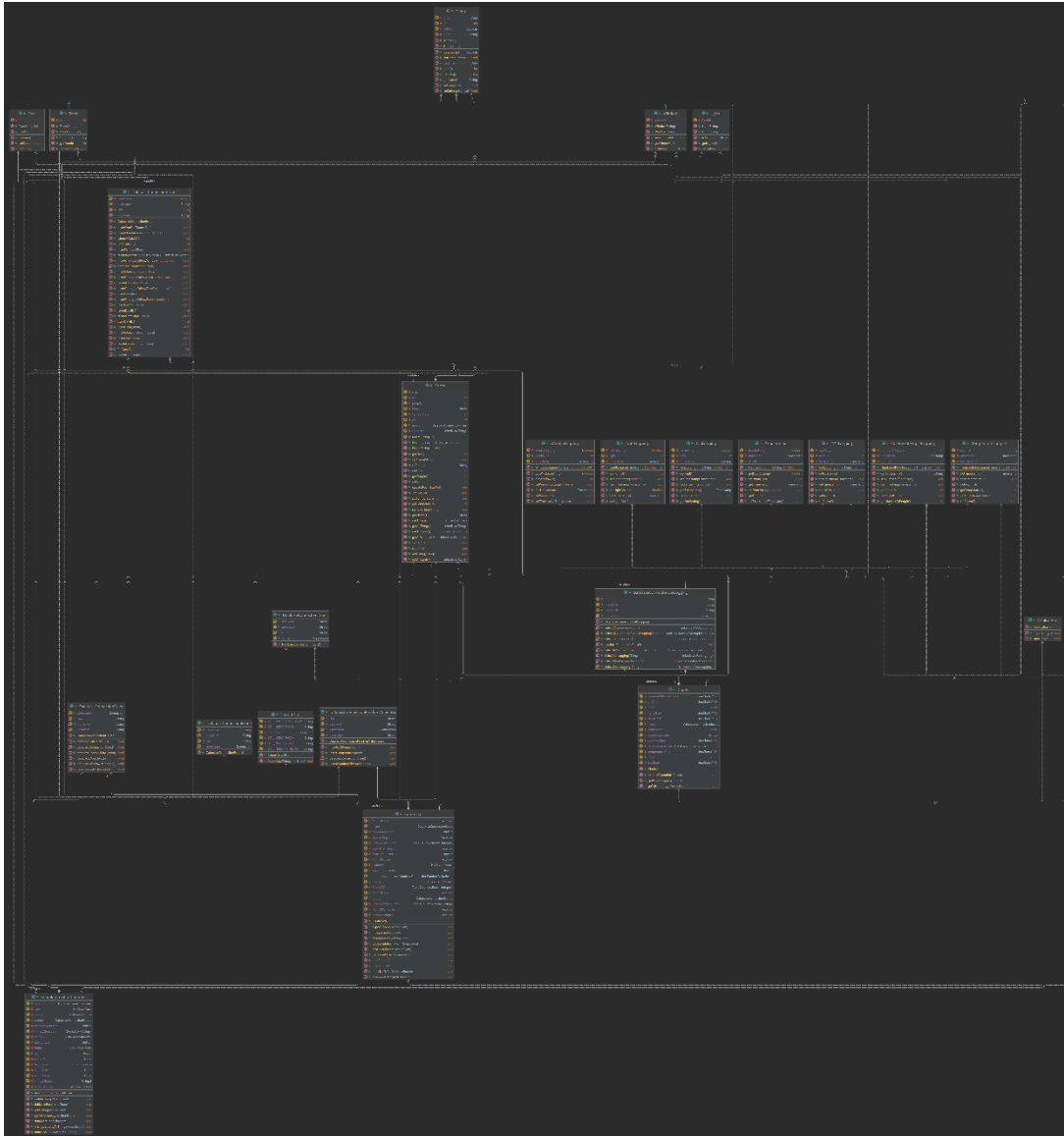
Rechts oben befindet sich die Temperatur-Info. Diese ändert sich je nach Raum-Temperatur. Falls sich die Raum-Temperatur unter 70 Grad Celsius befindet, wird die Meldung „Keine Temperaturgefahr“ in weißer Schrift angezeigt. Falls sich jedoch die Raum-Temperatur über 70 Grad Celsius befindet, wird die Meldung auf „Warnung: Temperatur über 70 Grad Celsius“ in roter Farbe umgeändert.

Rechts unten befindet sich für den User eine Information zum CO2 Wert des Raumes. Ist der Wert unter 800ppm, so ändert sich die Raum Name auf grün. Liegt der Wert zwischen 800-1000ppm so wird die Schrift in Gelb dargestellt. Und zuletzt verändert es sich auf Rot, wenn es einen Wert über 1000ppm hat.

4. Developer Perspective

4.1. Design

4.1.1. Overview of the System



Unter dem "Controller"-Paket befindet sich drei verschiedene Controller-Klassen. Die erste Klasse ist unser Hauptkontroller mit dem Namen „Controlle“, indem wir all unsere wichtigen Funktionen implementiert haben. Diese Funktionen wären bspw. AddRoom, DeleteRoom, etc. In dem zweiten Controller, das ist unser „RoomInformationController“, der für die Things zuständig ist. Diese „Things“ können hiermit bearbeitet werden. Beim dritten Controller handelt es sich um unser „Chart“-Controller. In diesem Controller werden die Daten, die für die Befüllung von Charts nötig sind, verwendet.

Im „Database“-Paket befinden sich die Klassen, die für die Datenbankconnection zuständig sind, die auch jeweils eine andere Funktion beinhalten.

Im „Loggings“-Paket haben wir die Logging-Objekte, welche die notwendigen Informationen, die für die Charts nötig sind. Diese Logging-Objekte beinhalten die Timestamps, ID's und Values der Objekte „Door“, „Window“, „Lights“ und „Fans“ sowie die Raum-Temperatur-Werte, CO2-Werte und die Anzahl der Personen in einem Raum.

Im „Objects“-Paket befinden sich beispielsweise die Getter- und Setter-Methoden für die jeweiligen Objekte. Diese sind „Door“, „Window“, „Lights“, „Fans“ und die abstrakte Klasse Thing.

4.1.2. Important Design Decisions

- **Datenbank**

Decision: PostgreSQL Datenbank

Reason: Unbegrenzte skalierbarer Datenbank, vorh. Erfahrung einiger Projektmitglieder, kostenlos (OpenSource-Community)

Considered Alternatives: MySQL Datenbank

Assumptions: leicht zu installieren bzw. aufzusetzen

Effect: Schnittstelle zu zahlreichen Programmiersprachen

- **Benutzeroberfläche**

Decision: Scene Builder

Reason: Kostenlos, Web UI ohne Programmcode, manuell WebUi graphisch gestalten,

Considered Pencil Project

Assumptions: Leichte Integration in IntelliJ

Effect: Effizient, da man Komponenten per Drag&Drop erstellen und die Eigenschaften verändern kann. Die graphische Darstellung kann je nach Wunsch schnell geändert werden.

- **Abstrake Klasse**

Decision: Abstrake Klasse "Thing"

Reason: Dynamischere Objekterstellung

Considered Alternatives: Mehr Aufwand für Objektklassen

Assumptions: Steigerung der Code-Qualität

Effect: Code effizienter

- **JDBC**

Decision: JDBC-Methoden

Reason: Aufgrund Erfahrung der Projektmitarbeiter

Considered Alternatives: Spring-Framework/Hibernate

Assumptions: Stabile Datenbankabfragen möglich

Effect: Intergration von SQL-Abfragen in Java

4.2. Implementation

Dependencies:

- JUnit, auch Java Unit genannt, ist ein Unit-Testing-Framework zum Schreiben und Ausführen automatisierter Tests von Klassen und Methoden in Java-Programmen. Im nachfolgenden Screenshot ist ein JUnit 4 zu sehen, welches in unserem Maven-Projekt eingerichtet wurde.

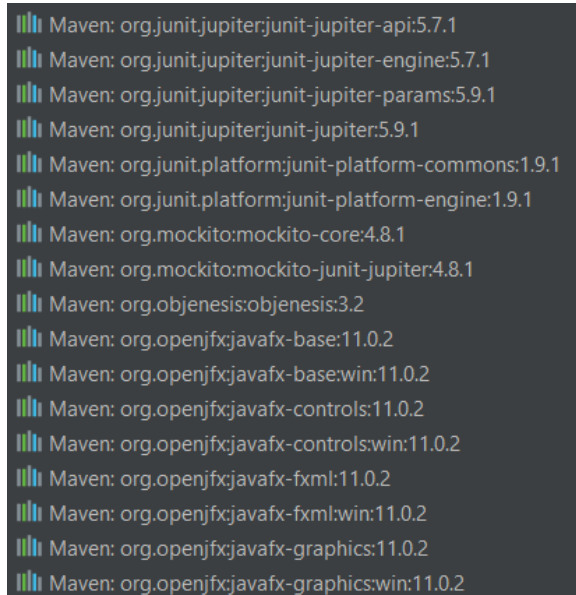
```
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-api</artifactId>
  <version>${junit.version}</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-engine</artifactId>
  <version>${junit.version}</version>
  <scope>test</scope>
```

- Da wir eine JavaFX-Anwendung basierend auf Maven entwickeln, welches für die Erstellung von Benutzeroberflächen notwendig ist, sind auch folgende dependencies nötig:

```
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-controls</artifactId>
  <version>11.0.2</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-fxml</artifactId>
  <version>11.0.2</version>
</dependency>
```

Libraries:

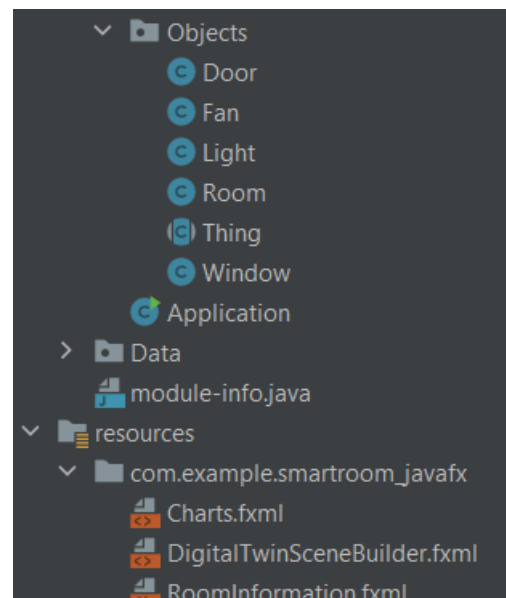
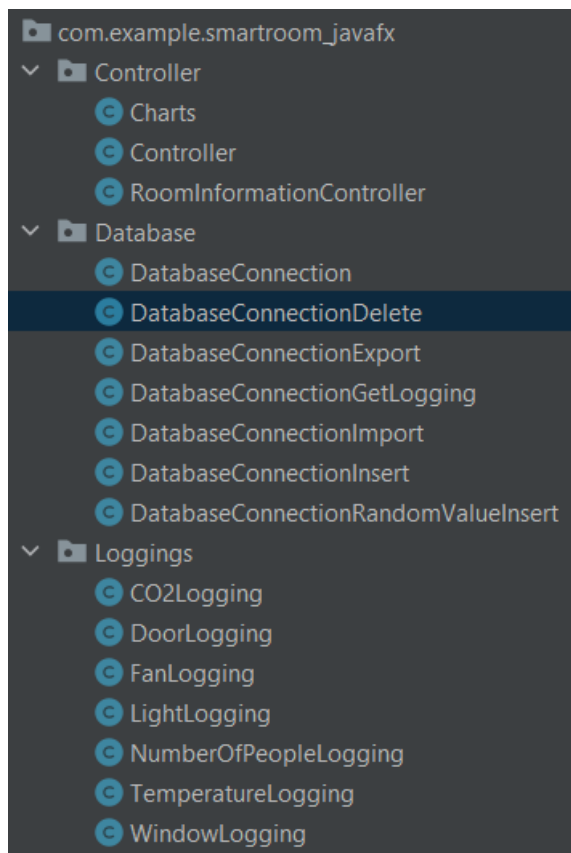
- Das folgende Bild ist ein Ausschnitt der Libraries unseres Projektes. Hier ist zu sehen, dass JUnit, Mockito und JavaFX-Libraries verwendet werden.



```
Maven: org.junit.jupiter:junit-jupiter-api:5.7.1
Maven: org.junit.jupiter:junit-jupiter-engine:5.7.1
Maven: org.junit.jupiter:junit-jupiter-params:5.9.1
Maven: org.junit.jupiter:junit-jupiter:5.9.1
Maven: org.junit.platform:junit-platform-commons:1.9.1
Maven: org.junit.platform:junit-platform-engine:1.9.1
Maven: org.mockito:mockito-core:4.8.1
Maven: org.mockito:mockito-junit-jupiter:4.8.1
Maven: org.objenesis:objenesis:3.2
Maven: org.openjfx:javafx-base:11.0.2
Maven: org.openjfx:javafx-base:win:11.0.2
Maven: org.openjfx:javafx-controls:11.0.2
Maven: org.openjfx:javafx-controls:win:11.0.2
Maven: org.openjfx:javafx-fxml:11.0.2
Maven: org.openjfx:javafx-fxml:win:11.0.2
Maven: org.openjfx:javafx-graphics:11.0.2
Maven: org.openjfx:javafx-graphics:win:11.0.2
```

Projektstruktur

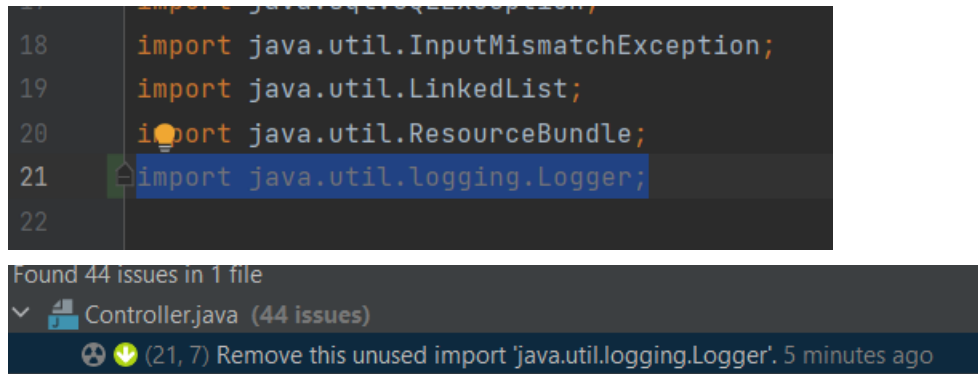
- In unserem Projekt haben wir mehrere Pakete. Diese sind unterteilt in „Controller-Paket“, „Database-Paket“, „Loggings-Paket“ und „Objects-Paket“. Unter dem Paket „com.example.smartroom_javafx“ befinden sich für die Benutzeroberfläche notwendigen FXML-Dateien, die mit Hilfe des Programms „Scene-Builder“ geöffnet werden können. Wir haben uns dazu entschieden, dass wir verschiedene Pakete erstellen, damit es übersichtlicher und geordneter wird. Ansonsten ist der Zeitaufwand, bis eine Klasse gefunden wird, viel höher. Es hat auch den Vorteil, dass zusammengehörige/ähnliche Klassen, die denselben Zweck haben, in demselben Paket enthalten sind. Dadurch wird eine effizientere Arbeitsweise ermöglicht.



4.3. Code Quality

SonarLint

- Unused Imports: Alle nicht verwendeten „import-Dateien“ wurden entfernt.



The screenshot shows a code editor with the following imports in `Controller.java`:

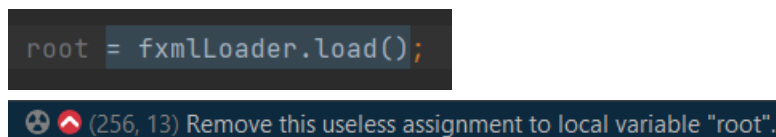
```

18 import java.util.InputMismatchException;
19 import java.util.LinkedList;
20 import java.util.ResourceBundle;
21 import java.util.logging.Logger;
22

```

Below the code, a SonarLint message states: "Found 44 issues in 1 file". A dropdown menu shows "Controller.java (44 issues)". A specific issue is highlighted: "(21, 7) Remove this unused import 'java.util.logging.Logger'. 5 minutes ago".

- Remove useless assignment: Nicht verwendete Zuweisungen wurden entfernt.



The screenshot shows a code editor with the following line of code:

```

root = fxmlloader.load();

```

Below the code, a SonarLint message states: "(256, 13) Remove this useless assignment to local variable 'root'".

4.4. Testing

Alle JUnit Tests wurden nach dem Arrange-Act-Assert Pattern umgesetzt. Folglich wurden 3 Test-Cases zur Veranschaulichung präsentiert. Die restlichen Tests können dem Code entnommen werden.

Test Case ID	1
Designed by	Florian
Execute on	DatabaseConnectionExport-Class
Carried out by	Florian
Test steps	Run the tests
Test data	Data of a room to export
Expected result	roomId is on the expected position of the exported file
Post condition	Exported file will be deleted afterwards
Status	All tests passed

```
public class DatabaseConnectionExportTest {

    @Test
    public void testExcelFileCreation() throws IOException, SQLException {
        //arrange + act
        DatabaseConnectionExport connection = new DatabaseConnectionExport();
        File file = new File( pathname: ".\\SmartRoomExportTEST.xlsx");

        //assert
        assertTrue(file.exists());
        file.delete();
    }

    @Test
    public void testExcelFileContent() throws IOException, SQLException, InvalidFormatException {

        //arrange + act
        DatabaseConnectionExport connection = new DatabaseConnectionExport();
        File file = new File( pathname: ".\\SmartRoomExportTEST.xlsx");
        Workbook workbook = new XSSFWorkbook(file);

        //assert
        assertTrue( condition: workbook.getNumberOfSheets() == 5);
        assertTrue(workbook.getSheet( s: "ROOM").getRow( i: 0).getCell( i: 0)
            .getStringCellValue().equals("roomId"));
        file.delete();
    }

}
```

Figure 1: JUnit Tests 1

Test Case ID	2
Designed by	Florian
Execute on	DatabaseConncectionInsert-Class
Carried out by	Florian
Test steps	Run the tests
Test data	No extra data required
Expected result	Room is correctly inserted into the database
Status	All tests passed

```
class DatabaseConnectionInsertTest {

    private DatabaseConnectionInsert connectionInsert;

    @BeforeEach
    void setUp() throws SQLException {
        //arrange
        connectionInsert = new DatabaseConnectionInsert();
    }

    @AfterEach
    void tearDown() { DatabaseConnectionInsert.databaseConnectionClose(); }

    @Test
    void createRoomsTest() throws SQLException {
        //arrange
        LinkedList<Room> rooms = new LinkedList<>();

        //act
        rooms = connectionInsert.createRooms(rooms);

        //assert
        assertNotNull(rooms);
    }
}
```

Figure 2: JUnit Tests 2.1

```
@Test
void insertRoomTest() throws SQLException {

    //arrange
    Room room = new Room( name: "Test Room", size: 100);
    room.setId(1);
    connectionInsert.insertRoom(room);
    LinkedList<Room> rooms = new LinkedList<>();
    rooms = connectionInsert.createRooms(rooms);
    boolean inserted = false;

    //act
    for (Room r : rooms) {
        if (r.getName().equals(room.getName()) && r.getSize() == room.getSize()) {
            inserted = true;
            break;
        }
    }

    //assert
    assertTrue(inserted);
}
```

Figure 3: JUnit Tests 2.2

```
@Test
void insertFanTest() throws SQLException {

    //arrange
    Room room = new Room( name: "Test Room", size: 100);
    room.setId(1);
    Fan fan = new Fan( name: "dyson");
    room.addThing(fan);
    connectionInsert.insertOneFan(room, fan);
    LinkedList<Room> rooms = new LinkedList<>();
    rooms = connectionInsert.createRooms(rooms);
    boolean inserted = false;

    //act
    for (Room r : rooms) {
        if (r.getName().equals(room.getName()) && r.getSize() == room.getSize()) {
            inserted = true;
            break;
        }
    }

    //assert
    assertTrue(inserted);
}
```

Figure 4: JUnit Tests 2.3

Test Case ID	3
Designed by	Florian
Execute on	WindowLogging-Class
Carried out by	Florian
Test steps	Run the tests
Test data	No extra data required
Expected result	Window changes are correctly logged
Status	All tests passed

```
@Test
public void testConstructorAndGetters() {

    //arrange
    Timestamp timestamp = new Timestamp(System.currentTimeMillis());
    int windowID = 1;
    boolean windowSetting = true;

    //act
    WindowLogging windowLogging = new WindowLogging(timestamp, windowSetting, windowID);

    //assert
    assertEquals(windowID, windowLogging.getWindowID());
    assertEquals(windowSetting, windowLogging.getWindowSetting());
    assertEquals(timestamp, windowLogging.getTimestamp());
}
```

Figure 5: JUnit Tests 3.1

```
@Test
public void testSetters() {

    //arrange
    Timestamp timestamp = new Timestamp(System.currentTimeMillis());
    int windowID = 1;
    boolean windowSetting = true;
    WindowLogging windowLogging = new WindowLogging(timestamp, windowSetting, windowID);
    int newWindowID = 2;
    boolean newWindowSetting = false;
    Timestamp newTimestamp = new Timestamp(System.currentTimeMillis());

    //act
    windowLogging.setWindowID(newWindowID);
    windowLogging.setWindowSetting(newWindowSetting);
    windowLogging.setTimestamp(newTimestamp);

    //assert
    assertEquals(newWindowID, windowLogging.getWindowID());
    assertEquals(newWindowSetting, windowLogging.getWindowSetting());
    assertEquals(newTimestamp, windowLogging.getTimestamp());
}
```

Figure 6: JUnit Tests 3.2

5. Installation guide

- 1) Postgres Datenbank und PG4 Admin downloaden (Windows Installer)
 - a. <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>
 - b. Als User Namen „postgres“ angeben und als Passwort und Admin Passwort „user123“ angeben
 - c. Datenbank „postgres“ und dazugehöriges Schema „public“ wurden default mäßig erstellt
- 2) Java Programm mit dem Branch „Release_3_Final“ starten
 - a. GitHub Link:
 - i. <https://github.com/jku-win-se/teaching.ws22.prse.smartroom.Team1>
- 3) Im Java Programm „Logischer_Entwurf_Gruppe1“ öffnen, befindet sich im Folder „Data“
 - a. SQL-Code kopieren
 - b. PG4 Admin starten
 - c. Rechtsklick auf Schema „public“
 - i. Query Tool des PostgreSQL Servers öffnen
 - d. Code einfügen und ausführen
- 4) Klasse „Application“ ausführen