JOHANNES KEPLER
UNIVERSITÄT LINZ

Digital Twin of a Smart Room Release 2 Team 2

Elma Buljina, Nuray Seker, Abir Sikder, Stefan Pilgerstorfer

Praktikum Software Engineering – WS2022/23

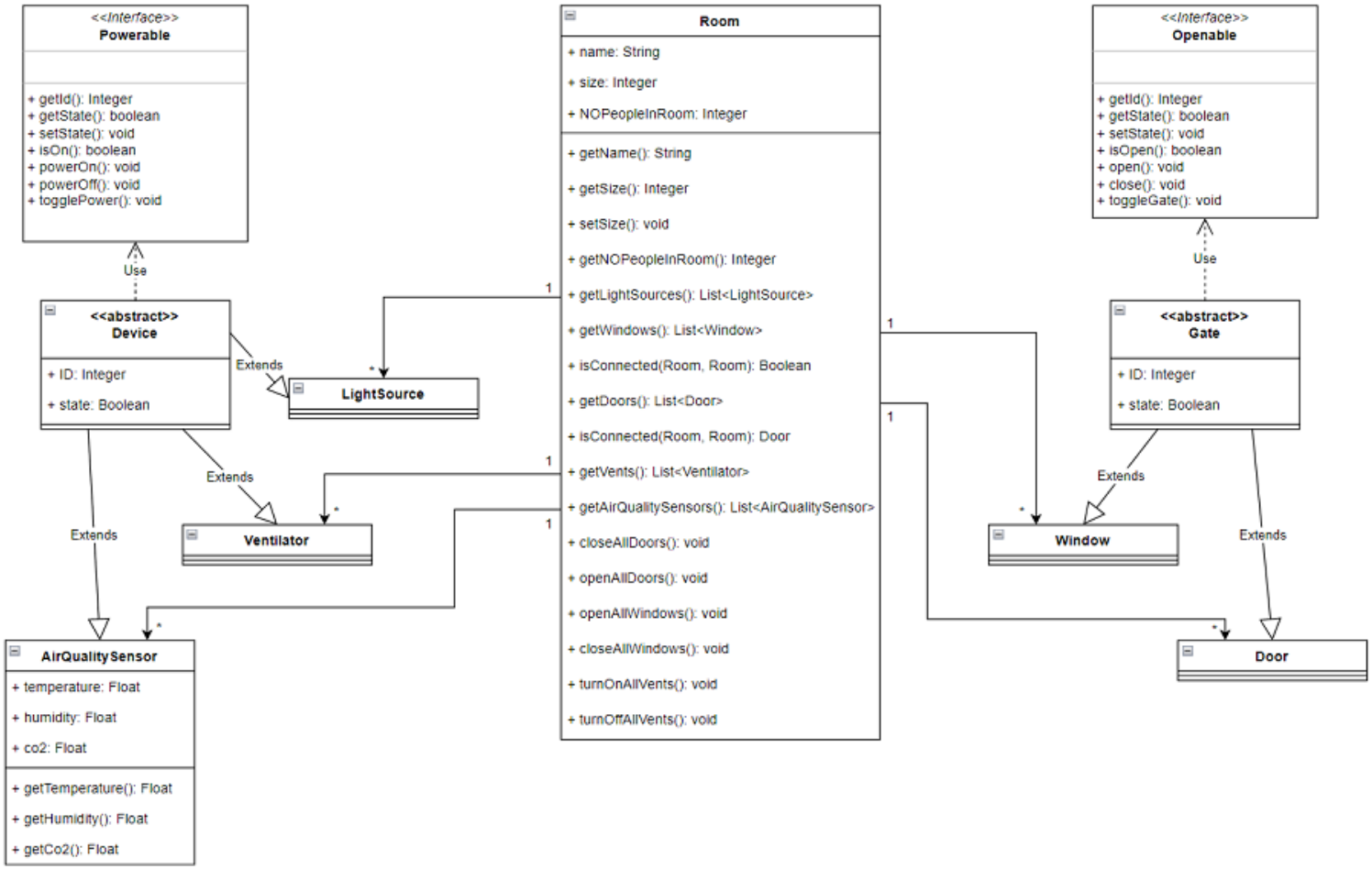
JOHANNES KEPLER UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
www.jku.at

Agenda

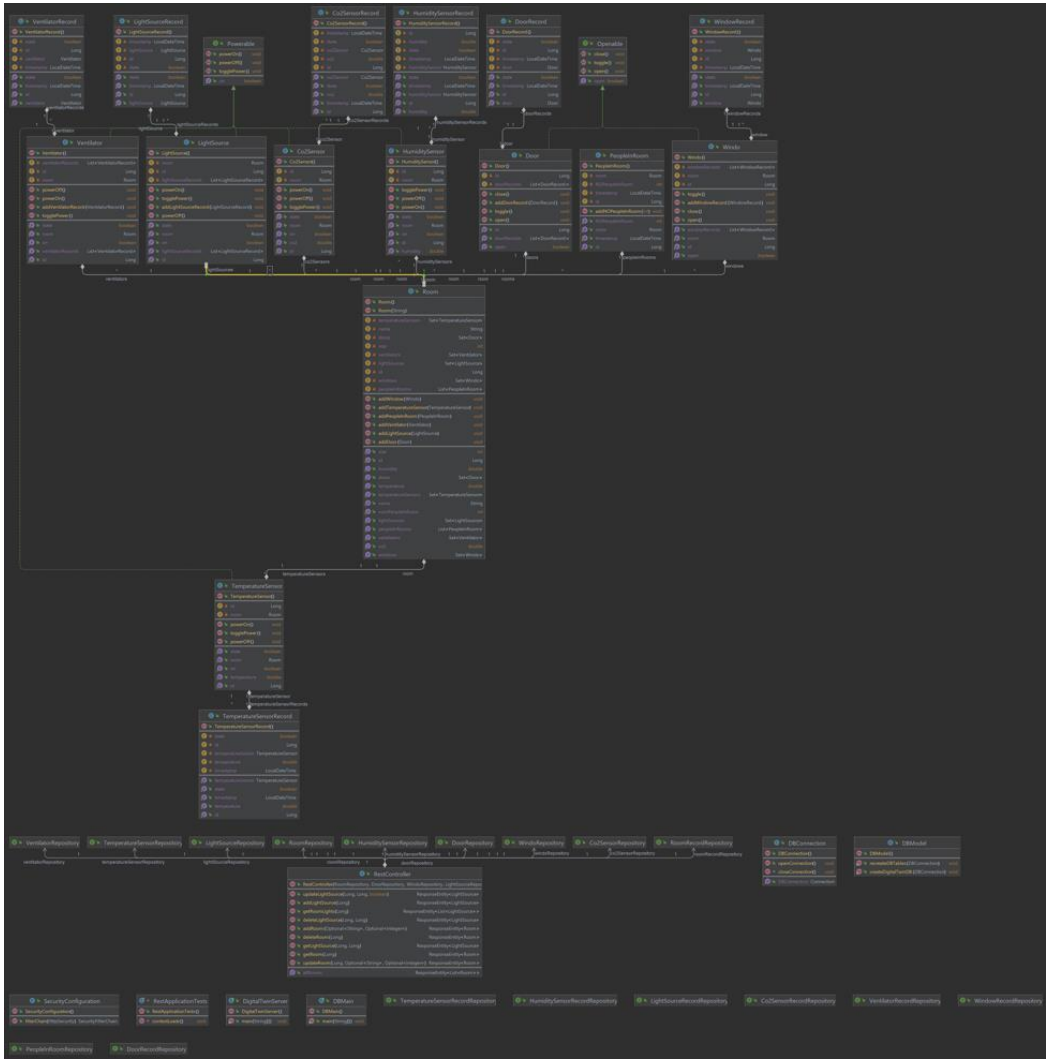
- *UML*
  - *UML (old)*
  - *Server-UML, Simulator-UML, Client-UML*
- *Demo*
- *Project Structure*
- *Software Quality*
- *Current Status + progress in Release 2*
- *Problems in Release 2*
- *Next Steps*
- *Release 3 - Strategy*

JKU

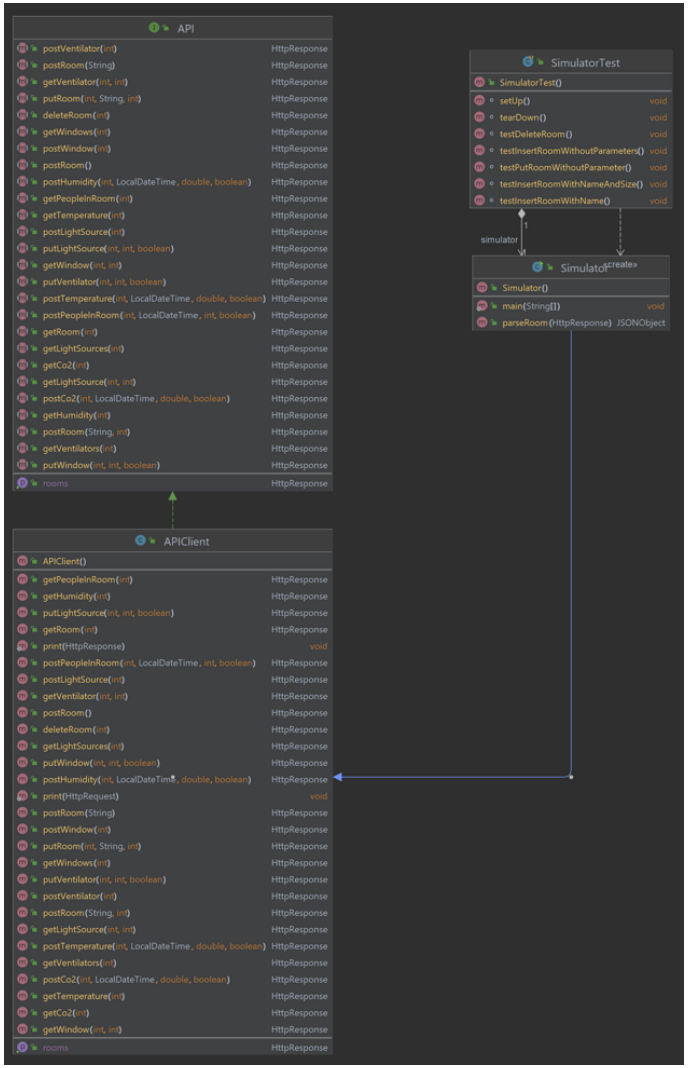
UML (old)


<<Interface>>
Powerable
+ getId(): Integer
+ getState(): boolean
+ setState(): void
+ isOn(): boolean
+ powerOn(): void
+ powerOff(): void
+ togglePower(): void
Use
<<abstract>>
Device
+ ID: Integer
+ state: Boolean
Extends
LightSource
Extends
Ventilator
Extends
AirQualitySensor
+ temperature: Float
+ humidity: Float
+ co2: Float
+ getTemperature(): Float
+ getHumidity(): Float
+ getCo2(): Float
Room
+ name: String
+ size: Integer
+ NOPeopleInRoom: Integer
+ getName(): String
+ getSize(): Integer
+ setSize(): void
+ getNOPeopleInRoom(): Integer
+ getLightSources(): List<LightSource>
+ getWindows(): List<Window>
+ isConnected(Room, Room): Boolean
+ getDoors(): List<Door>
+ isConnected(Room, Room): Door
+ getVents(): List<Ventilator>
+ getAirQualitySensors(): List<AirQualitySensor>
+ closeAllDoors(): void
+ openAllDoors(): void
+ openAllWindows(): void
+ closeAllWindows(): void
+ turnOnAllVents(): void
+ turnOffAllVents(): void
<<Interface>>
Openable
+ getId(): Integer
+ getState(): boolean
+ setState(): void
+ isOpen(): boolean
+ open(): void
+ close(): void
+ toggleGate(): void
Use
<<abstract>>
Gate
+ ID: Integer
+ state: Boolean
Extends
Window
Extends
Door
1
*
1
*
1
*
1
*
1
*

Server - UML



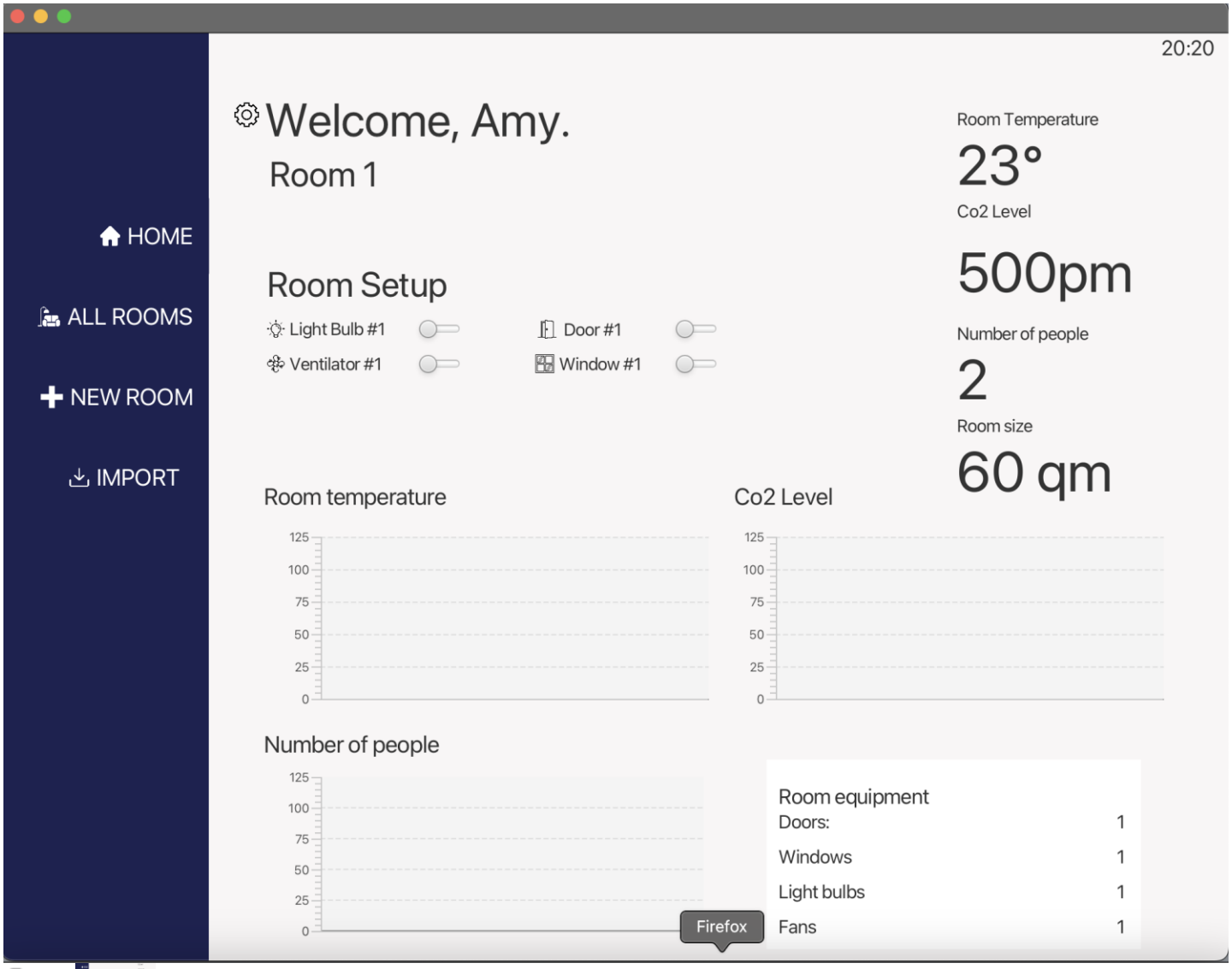
Simulator - UML



Client - UML


API
AllRoomsController
DigitalTwinApp
ImportController
PrimaryController
APIClient
NewRoomController

Live-Demo


20:20
Welcome, Amy.
Room 1
HOME
ALL ROOMS
NEW ROOM
IMPORT
Room Setup
Light Bulb #1
Door #1
Ventilator #1
Window #1
Room Temperature
23°
Co2 Level
500pm
Number of people
2
Room size
60 qm
Room temperature
Co2 Level
Number of people
125
100
75
50
25
0
Room equipment
Doors: 1
Windows 1
Light bulbs 1
Fans 1
Firefox

Code Quality

- *SonarLint (Live Demo, per Java-Class)*

Code Quality Suggestion 1 (SonarLint)

```
@RequestMapping(value = "/co2sensors/{co2sensor_id:.*}", method = RequestMethod.PUT)
public ResponseEntity<Co2Sensor> updateCo2Sensor(@PathVariable Long co2sensor_id,
                                                 @RequestParam Optional<Room> room) {
```

```
@PutMapping(value = "/rooms/{room_id:.*}/lights/{light_id:.*}")
public ResponseEntity<LightSource> updateLightSource(@PathVariable Long room_id,
                                                     @PathVariable Long light_id,
                                                     @RequestParam boolean state) {
    final Room room = roomRepository.getById(room_id);
    final Optional<LightSource> ls = room.getLightSources().stream().filter(l -> l.getId().equals(light_id)).findFirst();
    if (ls.isPresent()) {
        ls.get().setState(state);
    }
    return ResponseEntity.ok(ls.orElse( other: null));
}
```

Code Quality Suggestion 2 (SonarLint)

```
assertTrue( condition: this.id != null);
```

```
assertNotNull(this.id);
```

Sample Unit Test

```
class SimulatorTest {

    10 usages
    private JSONObject json;
    8 usages
    private Simulator simulator = new Simulator();
    17 usages
    private HttpResponse response;
    8 usages
    private Long id;

    no usages  buegi
    @BeforeEach
    void setUp() {
        this.json = new JSONObject();
        this.simulator = new Simulator();
        this.response = simulator.postRoom();
    }

    no usages  buegi
    @AfterEach
    void tearDown() {
        this.json = new JSONObject(response.body().toString());
        this.id = json.getLong( key: "id");
        this.response = simulator.deleteRoom(this.id);
    }
```

```
@Test
void testPutRoomWithoutParameter() {
    this.response = simulator.postRoom();
    this.json = new JSONObject(response.body().toString());
    this.id = json.getLong( key: "id");
    this.response = this.simulator.putRoom(this.id, name: "SimUpdate", size: 45);
    this.json = new JSONObject(response.body().toString());
    this.id = json.getLong( key: "id");
    String name = json.getString( key: "name");
    Integer size = json.getInt( key: "size");
    assertEquals( expected: 200, this.response.statusCode());
    assertNotNull(this.id);
    assertEquals( expected: "SimUpdate", name);
    assertEquals( expected: 45, size);
}
```

Current Status + progress in Release 2

- *What was done, how was the distribution of the tasks?*
  - ☐ *API for Room, Light Source, Ventilator, Window*
  - ☐ *CRUD Room*
  - ☐ *Majority of GUI*
  - ☐ *Export functionality (room only)*
  - ☐ *(Simulator) HTTP Response for Room, Light Source, Ventilator, Window*
  - ☐ *UML Diagram (Update)*
- *What are tasks that were planned but not finished?*
  - ☐ *Remote control*
  - ☐ *Visualize Static information*
  - ☐ *Complete API*
  - ☐ *Unit tests*

Task Distribution

- **Mandatory:** *Distribution of tasks*
  - *Which person was working on which parts, and how much effort was put into these parts?*
  - *This is your chance to show us that everyone is contributing to each part of the software (project requirement)*

Member	Task	Details
Nuray Seker	Front-End: GUI Back-End: -Action Handles -Calling Rest Methods	Design of Main Pane (Mock Data), Menu Bar, Page: All Rooms Functions: - Export function with chosen name (into a .csv File) Action Handle: - Show existing rooms (currently sample size of 10) - Create/Update/Delete room - Select ALL
Elma Buljina	Front- End: GUI	New room Controller created New files Links Import Edit room function Update Save, Cancel button

Member	Task	Details
Stefan Pilgerstorfer	Back-End: -Project Structure -Modules -Database (AWS / MySQL) -DB Model -Hibernate, JPA, Spring-Boot, Spring-Security, Spring-Data -JPA Repositories -API Room -Simulator Client -API Client -Unit Test	API: (GET, POST, PUT, DELETE)
Abir Sikder	Back-End: -API LightSource -API Ventilator -API Window -API Client (LightSource, Ventilator, Window) -Modules -UML Diagram (Update)	API: (GET, POST, PUT, DELETE) API Client: HttpResponse -> Room, LightSource, Ventilator, Window

Problems in Release 2

- *Major Problems*
  - *Implementing Program parts that depend on unfinished other parts*
  - *Implementing Unit Tests while classes conceptually change*
  - *Effort estimation (high initial effort)*
- *What did the team learn?*
  - *Effort estimation (with knowledge comes speed)*

Next Steps

- *CSV Import functionality (Backend)*
- *Complete API (Questions!) (Backend)*
- *Unit Tests (Backend)*
- *Adding/editing devices (GUI)*
- *Line Charts (GUI)*
- *Remote Control (Backend & GUI)*
- *Automation Rules (Backend & GUI)*

Release 3 - Strategy

- *Project conceptually finished*
- *Working on class = finishing it*
  - ☐ *Correct Code Quality Issues*
  - ☐ *Implement TODOS*
  - ☐ *Implement missing functionalities*
  - ☐ *write Unit Tests*
- *Release 3 = Functionally finished product*
- *After Release 3 only minor bug fixing*