Benedikt Reuter[1]*, Gottfried Zimmermann, Tobias Ableitner and Sebastian Koch*

Universal Design & Personalization for Smart Homes – Implementation

Abstract

This unit aims to provide a technical understanding of how accessibility can be implemented during application development. The personalization framework OpenAPE is used for this purpose. In addition, basic knowledge in the area of smart homes will be taught, for this purpose, the smart home framework OpenHAB will be used.

Keywords

Personalization, OpenAPE, OpenHAB

1 Preface

1.1 Overview

In this chapter, a practical study unit is provided. The unit is about personalization and accessibility, on the one hand, and a possible interface to smart homes, on the other.

First, an attempt is made to get a sense of why personalization might be helpful. Then we describe how OpenAPE, the personalization framework we use in the chapter, works and which functionalities are needed for this study unit. These functionalities are described in technical detail and are also supported with short videos.

In the following part, OpenHAB is explained, a framework that is used in smart homes. Basic knowledge of this framework is presented and the functionalities that are needed in this unit are explained in more detail. This is again supported with short videos.

1 (1) Stuttgart Media University / E-Mail: reuter@hdm-stuttgart.de

Then the OpenAPETutorial app is presented, an example application where missing source code must be filled in. When filling in this source code, students must apply knowledge from the previous parts of the tutorial.

1.2 Didactic fundamentals

Didactical aspect	Description
Target group	Computer science students with prior knowledge of programming.
Effort in lecture hours	60 minutes theory, 180 minutes practice
Effort for self-study materials	~ 20 minutes
Prerequisites	– Computer with an IDE for Java development; eclipse should work best
Needed background	– Basic knowledge of programming, at best of Java – Study Unit "Universal Design & Personalization for Smart Homes—Concepts"
Additional information	– The students need an account with OpenAPE – An OpenHAB server must be available; this can also be provided by the VPuxLab.

1.3 Learning Objectives and Competence

The goals of this chapter are that the students:

1. pay more attention to accessibility problems
2. learn how to improve accessibility through personalization
3. know what OpenAPE is
4. know how to use OpenAPE
5. know what OpenHAB is
6. know how to use OpenHAB

2 Why Personalization

An example now demonstrates why personalization can be useful. In the example, visual impairment is demonstrated because a visual impairment is more tangible and comparable to other barriers.

Imagine not being able to see a specific color. You want to book a seat in the cinema, and you get one of the following views on your screen. Above it, it says: "Select a free seat; the free seats are marked in green."

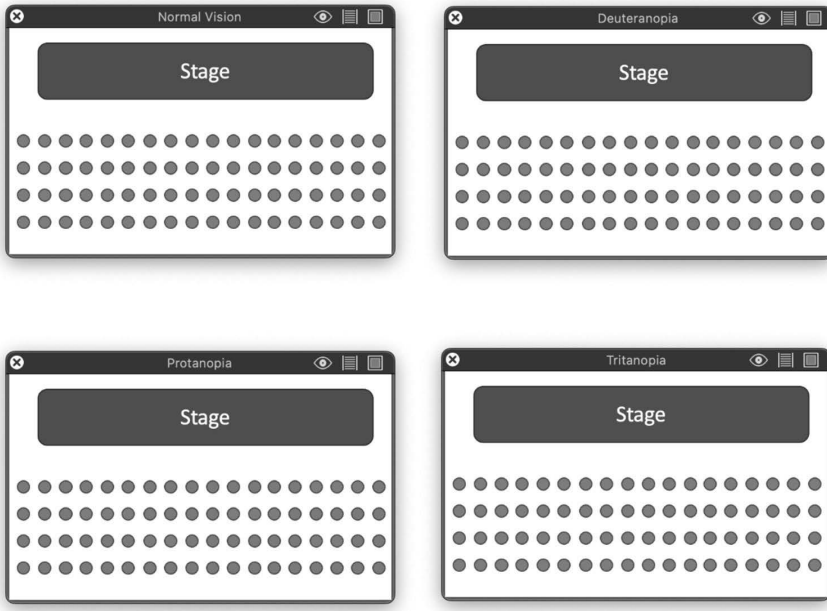

Figure 1: Seat selections with different visual impairments

The images above were created using the application Sim Daltonism: (https://apps.apple.com/de/app/sim-daltonism/id693112260?mt=12)

The picture on the top left shows how a person without a visual impairment sees the selection. All others are from the point of view of a person with a visual impairment. The picture on the top right shows deuteranopia, i.e., a red–green weakness. The image bottom left demonstrates protanopia, a red weakness. The last picture shows tritanopia, a blue–yellow weakness.

As you can see, the selection of colors can be a big problem, especially if the color is the only characteristic used for the selection. But this is not recommended either way. The example should nevertheless clearly illustrate how such problems can arise.

With personalization, you could customize these colors. Users could choose the colors that suit them best. This can be applied to many settings like font, font size, contrasts, or even language. Additionally, these settings could then be saved and transferred to other devices and applications.

3 OpenAPE

The basic knowledge of OpenAPE should already have been covered in study unit 1 "Universal Design & Personalization for Smart Homes—Concepts". Therefore, this unit will focus more on technical details.

OpenAPE differs in four context types: User Contexts, Task Contexts, Equipment Contexts und Environment Contexts. Each context type is used specifically for certain use cases. In this unit, only the User Context matters, so in examples, this context type is always used.

OpenAPE: https://openape.gpii.eu/

1.1 OpenAPE Rest-API

Besides the call for authentication, there are five rest calls for each context type:

Description	Rest-Method	Path
Authentication	POST	/api/token
Get list of all contexts	GET	/api/<type>-contexts
Get single context	GET	/api/<type>-contexts/<context-id>
Create context	POST	/api/<type>-contexts
Update context	PUT	/api/<type>-contexts/<context-id>
Delete context	DELETE	/api/<type>-contexts/<context-id>

Rest-Documentation from OpenAPE via Swagger: https://docs.openape.gpii.eu/

Examples of all rest calls can be found in the following playlist. The link to each individual example is additionally directly next to the example. The videos with the numbers 1–6 belong thematically to this section:

https://www.youtube.com/watch?v=gDIyaC3eC_Q&list=PLNVCxdMrGvQkmUjCS8gHd31qXWUbEjXDV

All the examples were made with the help of the tool Insomnia (https://insomnia.rest/). In the appendix of this unit, there is an exported file containing the examples. This file can be imported into Insomnia to try out the interfaces independently using the examples. A tutorial on how to use Insomnia can be found here: https://docs.insomnia.rest/

3.1.1 Authentication

Aspect of authentication	Description
Method	POST
Path	/api/token
Header	content-type: multipart/form-data

Aspect of authentication	Description
Multipart/form-data	– grant_type: password – username: <username> – password: <password>

Example response body	{
	"access_token": "eyJhb…",
	"expires_in": "1440"
	}

Example error body	{
	"error": "<error>",
	"error_description": "<errorText>"
	}

Example execution	https://youtu.be/gDIyaC3eC_Q

3.1.2 Get list of contexts

Aspect of get contexts	Description
Method	GET
Path	/api/user-contexts
Header	authorization: <token>

Example response body	{
	"totalContexts": 1,
	"user-context-uris": ["6130c06503604740e28eb27a"]
	}

Example errors	–Status 401 – Unauthorized oNo or wrong Token?
Example execution	https://youtu.be/zwWqU7LsqsU

3.1.3 Get single context

Aspect of get context	Description
Method	GET
Path	/api/user-contexts/<context-id>
Header	– authorization: <token> – content-type: application/json
Response	Context in json format

Example errors	– Status 401 – Unauthorized
	  – No or wrong Token?
	– Status 404 – Not found
	  – No Content-Type?
	  – "no object with that id"
	  – "invalid hexadecimal representation of an ObjectID"

Aspect of get context	Description
Example execution	https://youtu.be/HkuBlghoxgs

3.1.4 Create context

Aspect of create context	Description
Method	POST
Path	/api/user-contexts
Header	– authorization: <token> – content-type: application/json
Body	Valid json of context
Response	No response body
Example errors	– Status 400 – Bad Request •Not a valid User Context? – Status 401 – Unauthorized •No or wrong Token? – Status 404 – Not found •No Content-Type?
Example execution	https://youtu.be/Fkn7KVboqHg

3.1.5 Update context

Aspect of update context	Description
Method	PUT
Path	/api/user-contexts/<context-id>
Header	– authorization: <token> – content-type: application/json
Body	Valid json of context
Response	No response body
Example errors	–Status 400 – Bad Request • Not a valid User Context? –Status 401 – Unauthorized • No or wrong Token? –Status 404 – Not found • No Content-Type?
Example execution	https://youtu.be/HRy8_RsBuVU

Be careful with this call. When you update a context, it will be completely overwritten on the OpenAPE server. So, the full context must be sent along with it.

3.1.6 Delete context

Aspect of delete context	Description
Method	DELETE
Path	/api/user-contexts/<context-id>
Header	authorization: <token>
Response	No response body
Example errors	– Status 401 – Unauthorized • No or wrong Token? - Status 404 – Not found • No Content-Type? • "no object with that id" • "invalid hexadecimal representation of an ObjectID"
Example execution	https://youtu.be/3uld25UeXQc

3.4 OpenAPE Java Client

In the OpenAPE Java client, the following methods are available:

Method Call	Link to Example Execution
login(..)	https://youtu.be/HQjCGhvk4Ms
isLoggedIn()	Same video as for the login method
getUserContexts()	https://youtu.be/WbJO1vZw500
getUserContext(..)	https://youtu.be/4trZ1HsWSgo
createUserContext(..)	https://youtu.be/sgflu_hikxM
updateUserContext(..)	https://youtu.be/h9emwfFYH1A
deleteUserContext(..)	https://youtu.be/9AV3O1VNYsc

Besides these methods the constructor is needed, i.e. OpenAPEClient(). Besides calling it up without parameters, it can also be called up with a URL as a string parameter. If this is done, the application will try to initialize the client with the corresponding URL as the OpenAPE Server. If this is not done, the default URL https://openape.gpii.eu/ will be used.

You can watch example videos of the methods in the following playlist. Videos 9–15 are about the OpenAPEClient: https://www.youtube.com/watch?v=gDIyaC3eC_Q&list=PLNVCxdMrGvQkmUjCS8gHd31qXWUbEjXDV

The documentation and further information about the client can also be found in the README.md file in the client's repository. You can find it here: https://gitlab.bf-hdm.de/benedikt.reuter/openapeclient
Among other things, it describes how to build the jar file of the OpenAPE-Client or how to include it in existing projects.

Since the client uses the Rest API of OpenAPE, the methods mentioned are of course based on it.

The following image describes an example workflow using the OpenAPEClient. In the example, a user logs in via the interface and saves their settings for the first time. The example is simplified.

First, the OpenAPEClient logs the user into the OpenAPE interface. The token it receives is then stored. Then the OpenAPEClient creates a new context with the user's saved settings. The ID of the new context is sent in response. What is done with it varies depending on the use case.

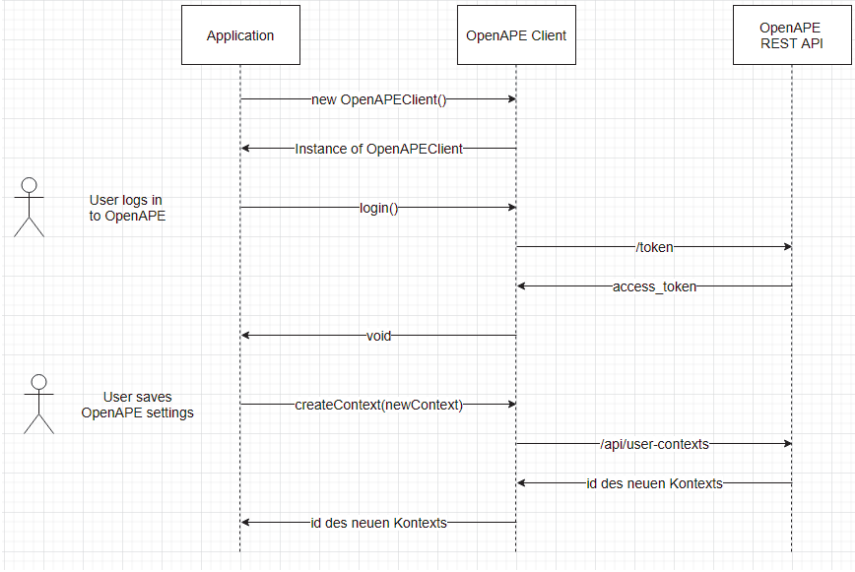

Figure 2: Sequence diagram for an example workflow

Careful! In real applications, it should first be checked whether a context already exists which can be updated.

Instructions on how to import the .jar file into a project can be found here: https://youtu.be/oDOloJaHtoE

1.1.1 Data structure

The data types used in the client all end with the abbreviation DTO, which stands for Data Transfer Object. Using these data types, the data is de-serialized from the rest interface, i.e., from a JSON format into a suitable format and the other way round.

The most important and complex are UserContextMapDTO and UserContextDTO. The UserContextMapDTO contains a map which is returned when all contexts are retrieved. The key in the map is the ID of a context and the value is then the context itself. The context is a UserContextDTO. To understand the basics of these classes, corresponding UML diagrams are shown. These UML diagrams do not represent the entire data structure of the OpenAPEClient, but the most important parts.

In the following UML diagram, you can see all the publicly visible methods of the UserContextMapDTO. With it you can add, update, and remove contexts.

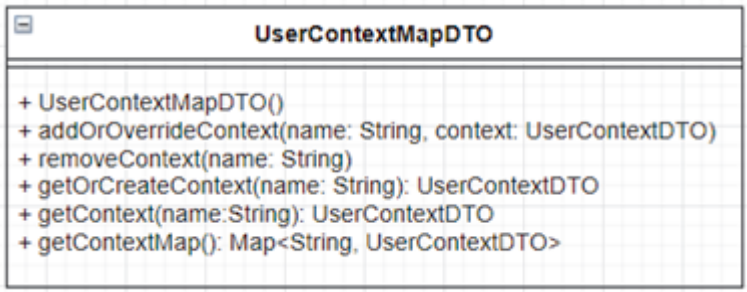

Figure 3: UML Diagram UserContextMapDTO

The next UML diagram shows the class UserContextDTO. Here, again, all the important publicly visible methods are described. With this class, you have the chance to change the name, the preferences, and the conditions of a single context.

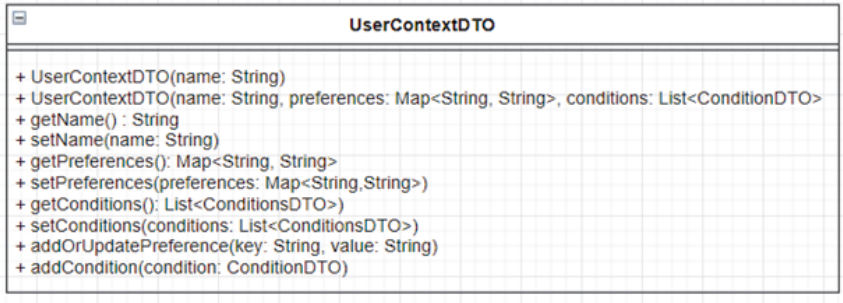

Figure 4: UML Diagram UserContextDTO

The ConditionDTO mentioned in the diagram will not be discussed further in this unit, as it is not required for the time being.

3.4.7 Error Handling

There are six exceptions, which arise depending on the use case. The name of the exception should already indicate when it is thrown up.

Name of Exception	Short description
OpenAPEAuthenticationException	thrown up when an authentication error happens
OpenAPECreateContextException	thrown up when an error happens during context creation
OpenAPEDeleteContextException	thrown up when an error happens during context deletion
OpenAPEUpdateContextException	thrown when an error happens during context update
OpenAPEResourceNotFoundException	thrown up when a requested context is not available
OpenAPEServerConntectionException	thrown up when the server is not available

All exceptions derive from the OpenAPEException. This exception can be caught to handle all error messages from this client.

4 OpenHAB

4.1 Background

The open-source project OpenHAB stands for open Home Automation Bus. It is used in smart homes, where it can do different jobs. Many devices, systems, or technologies can be connected to the OpenHAB system and OpenHAB can become the center of a smart home. Moreover, automation

can be achieved in the area of smart home with OpenHAB. A cloud connection is not necessary with OpenHAB, but if desired, it can be connected to common cloud-based systems. You can host OpenHAB on your own local system or via a cloud connector directly in the cloud.

To connect OpenHAB with other devices, you can install bindings in your OpenHAB system. These bindings create a connection to the devices and then create so-called things. These things are the corresponding representations of the devices.

Things can then be used to generate items. Items represent functionalities of the things. They have a status which can be changed if necessary. For example, there is the item type "Switch", which can represent the on/off switch of a lamp.

For automation, rules can be created that can execute different commands in different situations.

Read more:

- Website: https://www.openhab.org/
- Documentation: https://www.openhab.org/docs/
- Documentation Rest-API: https://www.openhab.org/docs/configuration/restdocs.html
- Install Instructions: https://www.openhab.org/docs/installation/
- Tutorial: https://www.openhab.org/docs/tutorial/

4.2 OpenHAB Rest-API

For the task in this study unit, two rest calls are needed: a call to get all the items of the interface and a call to change the state of an item.

Examples of all rest calls can be found in the following playlist. Videos 7 and 8 show the examples from this chapter: https://www.youtube.com/watch?v=gDIyaC3eC_Q&list=PLNVCxdMrGvQkmUjCS8gHd31qXWUbEjXDV

4.2.1 Get all Items

Aspect of get all items	Description
Method	GET
Path	/rest/items
Basic Authorization	– username = <api-key> – password = empty
Response	List of all items with some information
Example execution	https://youtu.be/8rwBLBFSU1k

4.2.2 send Command

Aspect of send command	Description
Method	POST
Path	/rest/items/<item_name>
Basic Authorization	– username = <api-key> – password = empty
Header	content-type: text/plain
Body	Command as string, e.g., “ON”
Response	No response body
Example execution	https://youtu.be/CmhP-veYAZ4

To use the Basic Authorization, the Authorization field must be set in the header. This must be set with the value "Basic <credentials>", where the credentials must contain the following string encoded as base64: “username:password” (https://www.openhab.org/docs/configuration/restdocs.html#authentication)

4.3 OpenHAB Server

To implement the task from the tutorial, you need an OpenHAB server. You have two possibilities:

1. Set up a local OpenHAB server and connect your own devices

You need at least one item in your OpenHAB configuration.
Tutorial: https://www.openhab.org/docs/installation/

2. Use the virtual OpenHAB interface of the VPuxLab

A tutorial on how to use the VPuxLab will be provided by the instructor.

5 OpenAPETutorial Application

The application is a simple control application for lamps in a smart home. The lamps are connected via the OpenHAB interface; their personalization should be able to be synchronized via OpenAPE.

The application could be used, for example, on a tablet in a building where different people come and go. OpenAPE can be used effectively so that all people can use the lamp controls. To do so, the people can log in and their own settings will be adopted.

App functionalities:

- OpenHAB
  - Fetch & display items
  - Send commands (change device state)
- OpenAPE
  - Log in
  - Save Settings in Context
  - Load Settings from Context

The application is implemented using maven. Therefore, you start the project using maven. How this works can be found in the README.md file of the project. When maven is used, the project's configuration is done through the pom.xml file. When you download the project, you will have to enter the path to the .jar file of the OpenAPEClient there.

Only certain classes have to be filled in for the task. In these classes, empty methods, which are called up from other parts of the source code, already exist. So, if the functionality is implemented in these methods, the application should work properly.

In order to be able to save or retrieve settings, there is another method in the class for the OpenAPE functionalities. This method returns the object AppSettings. This object can be used to retrieve or save the settings of the application.

For more Information about the app, see:
https://gitlab.bf-hdm.de/benedikt.reuter/openapetutorialassignment

5.1 HTTP Client retrofit

To implement the functionalities for the rest interface of OpenHAB, an http client is needed. In this application, retrofit is used for this purpose. With retrofit an interface is created which implements appropriate wrappers for rest calls. These wrappers can then be used to execute the corresponding calls.

To implement the functionalities, you will need to know more about the retrofit client. You can find instructions on how to implement retrofit here: https://square.github.io/retrofit/

Note: The de-serialization of the items from the OpenHAB interface is already given. For this, you can use the ScalarsConverter and the JacksonConverter in retrofit. The ScalarsConverter is used to convert strings into JSON data. With the JacksonConverter retrofit can automatically convert the JSON data into the OpenHABItemDTO class. To use the ScalarsConver-

ter and the JacksonConverter you need the ScalarsConverterFactory and the JacksonConverterFactory. When the retrofit client is built with these converters, the required resources will be automatically converted.

You can find more detailed instructions on how this works in the documentation already linked. Example:

… .addConverterFactory(ScalarsConverterFactory.create()) …

6 Assignment

Now you should know how OpenAPE and OpenHAB can be used. In the exercise, you must now complete the smart home application mentioned in Chapter 4. If you get stuck, you can always refer to the information in this document.

How you proceed with the exercise is described below:

a) Download source code
   - Download OpenAPETutorial
   - Download the .jar file of the OpenAPEClient and add the path to the pom.xml file
   - Check if you can run the application; you can find a launch guide in README.md
   - The basic functionalities should work, but due to the missing source code no items will be displayed and the OpenAPE synchronization will not work

b) Fill in the missing code
   - package: org.openAPETutorial.openAPETutorial.exercise
   - classes: OpenAPEService, OpenHABService, OpenHABRest
   - There are already empty methods in the classes; these methods are called from the rest of the source code

c) Check functionality
   - Lamps (OpenHAB Items) are shown
   - Commands on OpenHAB lamps work—a command will change the status of a lamp
   - A login to OpenAPE will download the saved config from OpenAPE
   - After a login, a change in the settings will sync to the OpenAPE Account—check this on the website

Authors

Benedikt Reuter
Hochschule der Medien
Nobelstraße 10,
70469 Stuttgart
https://www.hdm-stuttgart.de/remex
reuter@hdm-stuttgart.de
benedikt.reuter@posteo.de

Prof. Dr. Gottfried Zimmermann
Hochschule der Medien
Nobelstraße 10,
70469 Stuttgart
https://www.hdm-stuttgart.de/remex
zimmermanng@hdm-stuttgart.de

Tobias Ableitner
Hochschule der Medien
Nobelstraße 10,
70469 Stuttgart
https://www.hdm-stuttgart.de/remex
ableitner@hdm-stuttgart.de

Sebastian Koch
Hochschule der Medien
Nobelstraße 10,
70469 Stuttgart
https://www.hdm-stuttgart.de/remex
kochs@hdm-stuttgart.de

1 Didactical Concept—Handout for Teachers Universal Design & Personalization for Smart Homes—Implementation

OpenAPETutorial App

2 Didactical Analysis

Short Description

This study unit aims to provide a technical understanding of how accessibility can be implemented during application development. The personalization framework OpenAPE is used for this purpose. In addition, basic knowledge in the area of smart homes will be taught, for this purpose the smart home framework OpenHAB will be used.

Target Group

The unit is intended for students of computer science. Students should already have prior knowledge of programming and have already completed the study unit "Universal Design & Personalization for Smart Homes—Concepts".

Institutional Requirements

The required resources are all described in the unit itself. For questions regarding content, the teachers should be available and already have some understanding of OpenAPE, OpenHAB, and the development of Java applications. The students need an OpenHAB interface. This can be provided centrally or made available via a VPuxLab.

Learning Objectives

Students should learn how to pay attention to the needs of all people while developing an application. Personalization with OpenAPE as a personalization framework will be used to teach these skills. Furthermore, the students will learn how to use OpenHAB and thus get to know basic knowledge in the field of smart homes. After the unit, students should be able to use the OpenAPE and OpenHAB interfaces and implement them in applications.

Subject matter

This study unit consists of theory and practical exercises at the same time. Some short videos with practical examples are shown in the unit, while further information on documents and videos is provided as further resources. The theory unit is followed by the practical exercise, in which the students must apply the knowledge they have gained. To do this, they can always check the resources provided.

3 Didactical Concept

Methodical implementation

Aspects of problem-based learning are used in this unit. In this regard, the students will learn mostly in a self-directed way. However, they should always have the opportunity to ask the lecturer questions if there is a problem. The procedure:

- Theoretical unit with the help of
  - basic knowledge
  - sample videos
  - further third-party documentation
- Exercise
  - Download the source code
  - Fill in the missing source code
  - Check the functionalities

Media

The unit requires an OpenHAB server in addition to the students' development environment. This can either be provided via a real OpenHAB server, or the OpenHAB interface of the VPuxLab can be used. Besides that, the resources provided in the course and the third-party documentation are sufficient.

Learning Organization

The course unit is treated individually by each student. The lecturer should only be ready for possible questions at any time. A forum where students can discuss problems or solutions could also be useful. The unit can therefo-

re be conducted in an online course, especially if the VPuxLab is used as an OpenHAB server.

Feedback and Evaluation

Students can check their work themselves by checking the functionality of the application for correctness. If further questions arise, a sample solution can also be provided to students.

Learning analytics could be used in the development phase to find out where students have problems and where the tasks are too easy.

Expert Tips

Probably the best preparation for the unit is to perform the exercise once yourself. Furthermore, a little experience in the use of the respective interfaces is useful.

The pdf and word files for the study unit are mostly accessible. So, if new content is added, care should be taken to ensure accessibility. The videos created also have English subtitles for this purpose.