



Michael Rossmann, BSc

Facility Access Regulation inside an Academic Makerspace

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Patrick Herstätter

Institut für Innovation und Industrie Management
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Christian Ramsauer

Graz, Mai 2021

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Abstract

Makerspaces are places where diverse groups of people come together for sharing knowledge, learning, but also for prototyping and realizing projects. Having so many different user groups in a makerspace also raises certain issues. Not everyone should have access to all resources in a makerspace. Specific areas in a makerspace should only be accessible to authorized user groups. This thesis focuses on the challenge of managing access regulation in an academic makerspace on the example of the Schumpeter Laboratory for Innovation (SLFI). Different access regulation technologies and systems are currently in use within the SLFI. However, each of these technologies must be managed individually, creating additional effort for administrators. The goal of this thesis is to reduce this additional effort for administrators by creating a platform that allows them to centrally manage all different technologies. To create such a central platform, all relevant systems within the SLFI are analyzed. Based on this analysis, various concepts are developed and evaluated. Finally, the concept that best fits the requirements is to be realized within the SLFI.

Contents

Abstract	iii
1 Introduction	1
1.1 Problem Description	1
1.2 Aim of this Thesis	2
2 Theory	3
2.1 The Internet of Things	3
2.2 The Web of Things	5
2.3 CRUD Operations	6
2.4 Application Programming Interface	8
2.5 Web Service	8
2.5.1 Difference between API and WS	9
2.6 Simple Object Access Protocol	10
2.7 Representational State Transfer	11
2.8 OpenAPI Specification	15
2.8.1 Swagger UI	15
2.9 Content Management Systems	15
2.9.1 WordPress	17
2.9.2 Custom WordPress Plugins	17
2.9.3 WordPress REST API	17
3 Analysis	19
3.1 Schumpeter Laboratory for Innovation	20
3.2 System Overview	22
3.2.1 Current System Architecture	22
3.2.2 Desired System Architecture	24
3.3 System Analysis	26
3.3.1 System Types	26

Contents

3.3.2	Data Carriers	26
3.3.3	Primion Access Control System	28
3.3.4	How is the Primion System used inside the SLFI?	29
3.3.5	ZID Access Control Management	31
3.3.6	Primion System Limitation and Possibilities	33
3.3.7	Primion Summary	35
3.3.8	Gantner Access Control System	35
3.3.9	How is the Gantner System used inside the SLFI?	35
3.3.10	Gantner System Limitation and Possibilities	37
3.3.11	Gantner Summary	39
3.3.12	Hoffmann Group Access Control System	39
3.3.13	GARANT Electronic Lock System (G-ELS)	39
3.3.14	Hoffmann System Limitation and Possibilities	44
3.3.15	Hoffmann Summary	44
3.3.16	SLFI WordPress System	45
3.3.17	SLFI Card Reader	45
3.3.18	Systems Summary	46
4	Concepts	48
4.1	System Boundaries	48
4.2	Hoffmann G-ELS Adaptation	49
4.3	Concept Analysis	49
4.3.1	Concept I	50
4.3.2	Concept II	54
4.3.3	Concept III	57
4.4	Concept Decision	59
5	Results	62
5.1	Concept Architecture	62
5.1.1	WordPress System	66
5.1.2	WordPress Database	68
5.1.3	Custom WordPress plugin	70
5.1.4	Python flask	71
5.1.5	Gantner GAT Relaxx	72
5.1.6	Python CSV Module	76
5.1.7	Python JSON Module	76

Contents

5.2	Development Environment Setup	77
5.2.1	XAMPP WordPress Setup	80
5.3	Setting up the Python web application	83
5.3.1	WordPress - Middleware - Gantner Interaction	83
5.3.2	WordPress - Middleware - Primion Interaction	93
5.3.3	WordPress - Middleware - Hoffmann Interaction	94
5.4	Setting up the WordPress plugin	94
5.4.1	Manage Gantner Resources	98
5.4.2	Manage Primion Resources	103
5.4.3	Manage Hoffmann Resources	107
6	Conclusion	110
	Bibliography	115

List of Figures

2.1	Application Design using RESTful principles. Own representation based on [Wil07]	5
2.2	Internet of Things and Web of Things Comparison. Own representation based on [Gui16]	7
2.3	An application that is using multiple APIs. Own representation based on [Red11]	9
2.4	SOAP architecture. Own representation based on [HK11]	10
2.5	REST architecture. Own representation based on [Rel19]	12
3.1	Schumpeter Laboratory for Innovation Layout	19
3.2	Schumpeter Laboratory for Innovation Areas	21
3.3	SLFI Current System Architecture	23
3.4	SLFI Desired System Architecture	25
3.5	Simplified own representation of a smart card data carrier containing multiple applications.	27
3.6	Primion Prime Key Technology, Own representation based on [Tec20]	30
3.7	Primion PKT master reader, [Pri21]	31
3.8	Primion PKT offline reader, [Pri21]	31
3.9	Primion PKT offline electronic door, [Pri21]	31
3.10	ZID Access Control Management	32
3.11	Overview of Primion Locks in the SLFI	34
3.12	Gantner Technology, Own representation based on [Gmb20]	36
3.13	SLFI Gantner System Overview	38
3.14	Android App Setup with Master Card. Own representation based on [Gro21a]	40
3.15	Register a lock in the Android App. Own representation based on [Gro21a]	40

List of Figures

3.16	Register a TAG in the Android App. Own representation based on [Gro21a]	41
3.17	Assign multiple TAGs to one Lock. Own representation based on [Gro21a]	41
3.18	Synchronize changes with the lock. Own representation based on [Gro21a]	42
3.19	Overview of Hoffmann locks in the SLFI	43
4.1	Concept I: A custom WordPress plugin. The Gantner system is accessed by REST endpoints. The plugin creates a CSV file that can be imported into the Primion system. The plugin provides a data endpoint for the mobile application from Hoffmann.	53
4.2	Concept II: A custom WordPress plugin. The Hoffmann system is integrated into the Primion system.	55
4.3	Concept III: Gantner and Hoffmann integration into Primion system	58
5.1	Overview of the core components that are used in concept I.	65
5.2	The WordPress dashboard is the control panel for the administrators.	66
5.3	The WordPress database structure and the relations between the tables. This figure is fully adopted from [Wor21a]	69
5.4	Plugins folder structure in WordPress.	70
5.5	Gantner REST API endpoints.	73
5.6	The main window of the GAT Relaxx desktop client.	75
5.7	An example Primion CSV file.	76
5.8	XAMPP control panel.	81
5.9	PHPMyAdmin dashboard used to create new databases.	82
5.10	Local WordPress installation running on <i>http://localhost</i>	82
5.11	Simplified sequence diagram of the interaction between WordPress, middleware and the access control systems.	83
5.12	Gantner authentication sequence.	84
5.13	Several different REST endpoints are used to create the state that is sent back to WordPress.	86
5.14	Updating Gantner authorizations	90
5.15	Delete or update a Gantner authorization	92

List of Figures

5.16	Middleware providing the Primion authorizations in CSV format.	93
5.17	Middleware REST endpoint that is providing the Hoffmann authorizations.	94
5.18	The main menu extended with the <i>Resources</i> options.	95
5.19	Custom WordPress <i>shortcode</i> used to load the dynamic content for the Gantner resources.	96
5.20	Overview of the front-end interaction	97
5.21	The left side represents an example organisation view created within the GAT Relaxx desktop client application. The Gantner state represented as HTML output on the <i>Gantner Resources</i> page is shown on the right side.	98
5.22	Locker group holding 12 lockers.	99
5.23	Plugin representation of the example locker group.	100
5.24	Gantner authorizations administration with the plugin.	101
5.25	Assign three user authorizations to the selected locker group.	102
5.26	Updated locker information	103
5.27	The simulated Primion state loaded from the WordPress database.	104
5.28	Assign new authorizations to the selected Primion resource.	105
5.29	The CSV file will be created and updated by the middleware. The file contains all Primion authorizations that are stored in the WordPress database.	106
5.30	Assign new authorizations to the selected Hoffmann resource.	107
5.31	The simulated Hoffmann state loaded from the WordPress database.	108
5.32	The REST endpoint provided by the middleware to get the Hoffmann state.	109
6.1	The implemented concept I. The red triangles indicate where additional work is needed.	113

1 Introduction

Makerspaces are shared workplaces that can be used by members for prototyping, but also realizing projects of any kind. Makerspaces that are equipped with similar types of maker tools including laser cutters, CNC milling machines, 3D printers, and more are called FabLabs or Techshops. These places have in common that they are communal and are designed for sharing information, learning, and collaborating. Makerspaces can be used by any member of any age reaching from kids to students and adults as well as potential partners of the industry. However, having a variety of user groups that are constantly changing does also raise challenges. These challenges include access regulation to certain facilities for example. Not everyone should be able to use a CNC milling machine without a proper training session. A wardrobe on the other side should be usable by anyone. And even here, certain lockers in the wardrobe might be wanted to stay reserved for in-house staff. [Mak]

1.1 Problem Description

Depending on the makerspace there might be areas that should only be accessible for certain user groups. Areas with expensive high-tech equipment should only be used by trained staff whereas the wardrobe can be used by everyone for example. Constantly switching user groups inside a makerspace requires a proper setup for access regulation. This thesis focuses on the challenge of managing access regulation in an academic makerspace on the example of the Schumpeter Laboratory for Innovation (SLFI) that uses multiple different software solutions. Using different software solutions for access control management causes additional overhead and problems. Administrators need to operate on multiple programs. Keeping track of

the access rights can get confusing because there is no uniform overview. The effort of access regulation administration in makerspaces with highly fluctuating user groups should therefore be reduced to a minimum within this thesis.

1.2 Aim of this Thesis

The primary goal of this thesis is to find out how to:

- Simplify the access control management to different areas of an academic makerspace for highly fluctuating user groups on the example of the SLFI.

Combining the important and most used functionalities of the different software solutions into one central platform makes the administration more transparent and reduces the overall complexity. Often, only a subset of the provided functionality of every single application is needed.

To provide a uniform platform it is necessary to:

- Identify potential interfaces to combine heterogeneous software solutions for access control.

To understand how the relevant information can be made available on one common platform, this thesis is giving an overview of modern approaches in the theory chapter 2. Chapter 3 analyses all relevant systems that are used in the academic makerspace. Relevant systems include the different software solutions for access management as well as other systems that are currently used and are important to fulfill the goal of this thesis. Based on the analysis, different concepts are developed in chapter 4. The concepts show how the different software solutions can be made accessible through one common platform. The results in chapter 5 show the implementation of the most prominent concept.

2 Theory

This chapter explains how the internet has evolved over the past years to provide a common platform for applications that are used today. The most important components include Application Programming Interfaces (APIs, section 2.4) and Web Services (WS, section 2.5). This chapter describes the prominent approaches and patterns that are used today.

2.1 The Internet of Things

Over the past decades, the internet has become more than the collection of multimedia pages. The immense progress in embedded devices brought a new form of objects into the world: Smart Things. A Smart Thing is a digitally enhanced physical object with one or more of the following attributes: [GT16], [Mad15]

- Sensors (temperature, location, light, etc.),
- Actuators (displays etc.),
- Processing (able to run programs and logic),
- Network/Communication interfaces (wired or wireless).

Smart Things range from machines and home appliances to wireless sensor and actuator networks as well as tagged everyday objects. [GT16]

Guinard et al. [GT16] define the term IoT as: “The Internet of Things is a system of physical objects that can be discovered, monitored, controlled, or interacted with by electronic devices that communicate over various networking interfaces and eventually can be connected to the wider internet.”

Things must not be directly connected to the internet itself. The term internet in IoT means that the services and data from a Thing can be accessed by the already existing infrastructure of the internet. [GT16]

The IoT follows the approach of using the web as a transport system. The idea is to use Hypertext Transfer Protocol (HTTP) as the transport protocol for Application Programming Interface (API, section 2.4) calls that can be used through a Web Service (WS, section 2.5). WS allows applications to communicate. Software applications that are written in different languages and running on different platforms can use WS to exchange data over the computer network such as the internet.

In the IoT approach, API calls are completely hiding the resources that are handled by the application. This approach is the exact opposite of a web architecture. In a web architecture, the operations on resources that are identified by Uniform Resource Identifiers (URIs) are modeled as HTTP interactions (section 2.2).

WS revolving around the Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL) technologies are the most prominent examples (section 2.6). Applications can be extended to the web by applying the extensions of the web API-oriented SOAP design. The transport-oriented approach is anchored in the world of building tightly coupled distributed systems. The problem is that only peers that support a set of technologies can use these systems. [Wilo7]

With so many Smart Things trying to communicate with one another in different ways, it is challenging to build one uniform communication platform in which all Smart Things can speak to each other effectively. The big drawback of IoT is that there is no universal protocol or standard that can work across the many networking interfaces available. In the IoT, many incompatible protocols co-exist that makes the integration of devices extremely complex and often not possible. Today, the IoT is a collection of isolated intranets that can not be connected. The limitations of the IoT can be seen when devices from various manufacturers should be integrated into a single application. [GT16]

The Web of Things (WoT) follows a different approach. Instead of creating another WS protocol, the WoT is using a platform that already exists, the

World Wide Web (WWW). The approach of the WoT is to integrate Smart Things into the web. [Gui11]

2.2 The Web of Things

While the IoT establishes the transport capabilities that allow Smart Things to interact with the physical world, the WoT proposes a different approach. The WoT aims to integrate Smart Things into one single application, the web. By integrating Smart Things into the web, they use the same standards and techniques as traditional websites. This leads to the big advantage that Smart Things can be reused in different contexts and applications, especially when they are designed using RESTful principles. Figure 2.1 demonstrates how a RESTful design of resources allows multiple applications to interact with the same set of resources without the need for API-based interactions between the applications. [Gui11], [Wil07]

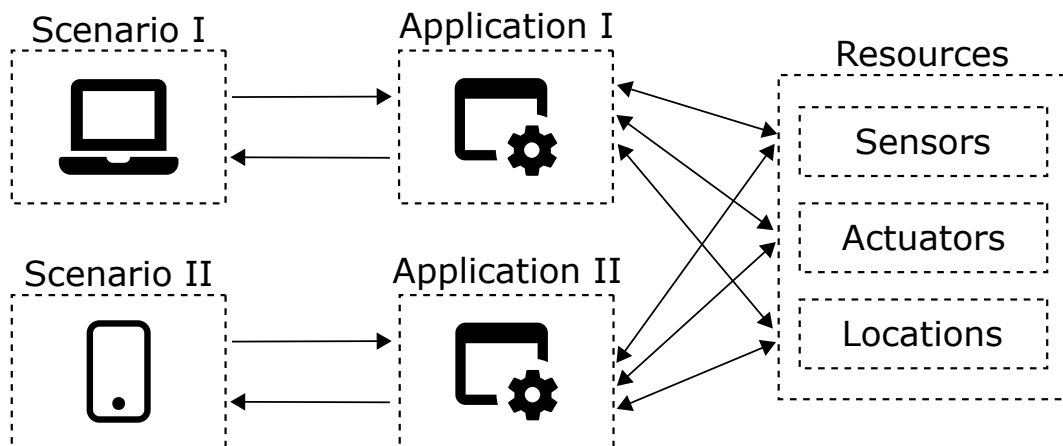


Figure 2.1: Application Design using RESTful principles. Own representation based on [Wil07]

The goal is to make Smart Things available through the fundamental mechanisms of the web. These fundamentals include: [Wil07]

- Uniform Resource Identifier (URI),
- Hypertext Transfer Protocol (HTTP),
- Hypertext Markup Language (HTML).

Uniform Resource Identifier: A URI is a globally scoped string of characters to identify resources. In the WoT the URI enables web clients to interact with the resource. [BFM05]

Hypertext Transfer Protocol: The main protocol for interacting with resources in a lightweight and loosely coupled way. HTTP provides the following main methods to interact with resources: [Wil07], [FR14]

- GET: retrieve a resource,
- POST: create a new resource,
- PUT: update an existing resource,
- DELETE: remove a resource.

Hypertext Markup Language: The used markup language to view resources in a human-readable form. XML is another important resource representation language mainly used for machine-readable resource formats. [BC95], [Wil07], [J3S02]

The next section explains the Representational State Transfer (REST) design and how it fits into the concept of WoT. Figure 2.2 demonstrates the major difference between the explained concepts IoT and WoT.

2.3 CRUD Operations

CRUD (Create, Read, Update and Delete) are the elemental functions to implement a persistent storage management system. In relational databases, the four CRUD operations refer to the SQL commands INSERT, SELECT, UPDATE and DELETE. Depending on the system users might use different CRUD operations.

Looking at an online store for example. Users can create an account, update

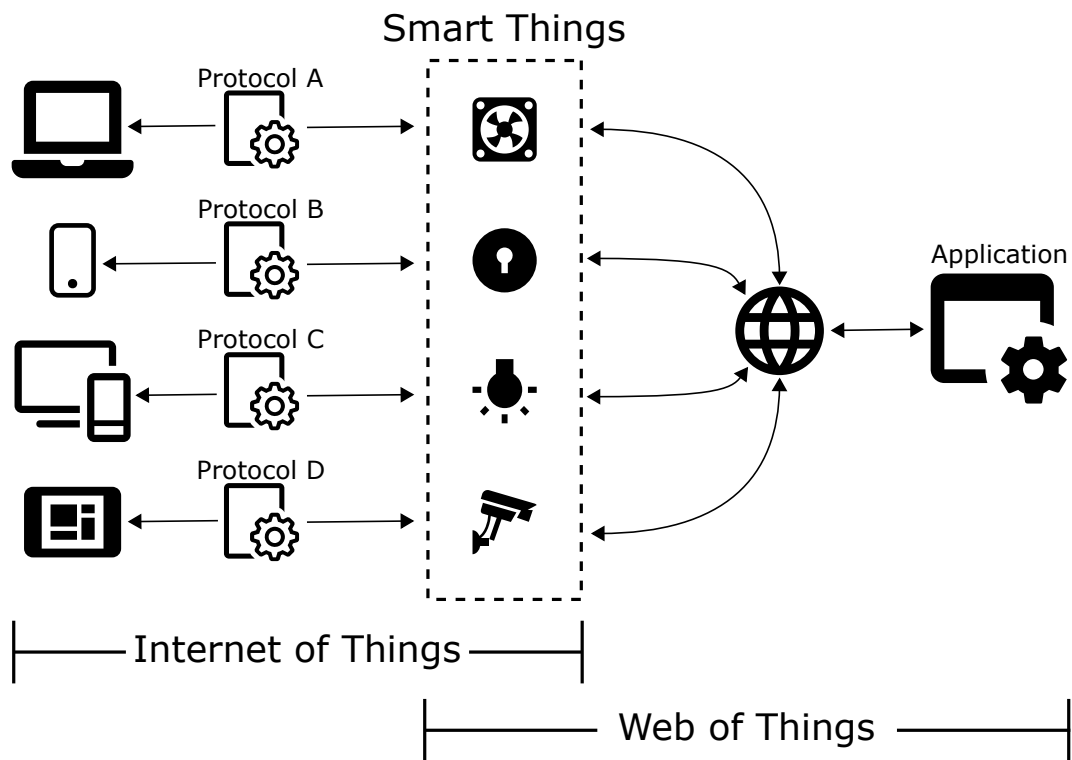


Figure 2.2: Internet of Things and Web of Things Comparison. Own representation based on [Gui16]

or delete their user information. In contrast, an operations manager might create product records, update or delete them if necessary.

A common way of creating good APIs is to follow a certain design pattern. CRUD is one of those design patterns and it is working well with APIs and WS. [Alt17]

2.4 Application Programming Interface

Application Programming Interfaces (APIs) allow applications of different types and implementations to communicate and exchange data with each other. APIs allow applications to leverage each other's data and functionality through a standardized and documented interface. Working with APIs is ubiquitous in the modern software development process. Modern applications are built on top of many APIs where some of these can depend on further APIs. Figure 2.3 illustrates an example application that directly depends on two APIs(1-2) where one of those APIs depend on a further API(3). [Red11]

The example application could be an image viewer where one API is used for loading images, and that same API is built upon a lower-level API for data compression and decompression. [Red11]

2.5 Web Service

Today's software applications are written in various languages. A Web Service (WS) allows these applications to communicate with each other over the internet. A WS provides resource-oriented interfaces to a database server that can be used by web clients (Figure 2.5).

Many technologies make this communication possible. However, when it comes to compatibility and security, two modern web API paradigms have been established: SOAP and REST. SOAP is a service-oriented architecture (SOA) whereas REST is a resource-oriented architecture (ROA). Both have

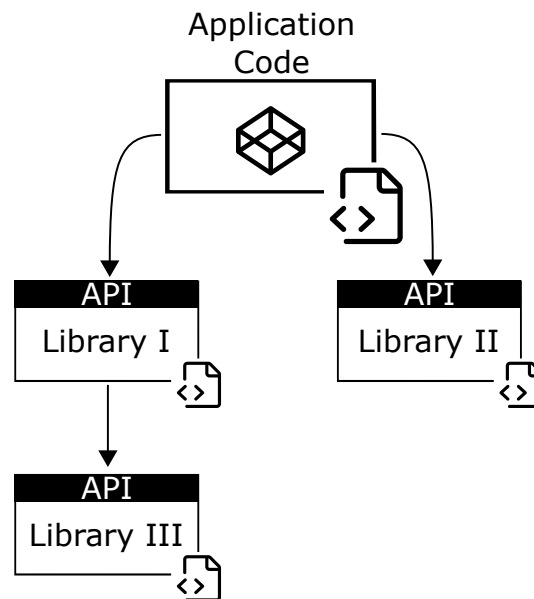


Figure 2.3: An application that is using multiple APIs. Own representation based on [Red11]

in common that they make an applications business logic or data available over a network. REST services represent about 70% of today's public APIs. [Rel19]

2.5.1 Difference between API and WS

The two terms are not mutually exclusive. Every WS is an API since it exposes the data or functionality of an application, but not every API is a WS.

- WS require a network, while APIs can be online or offline.
- APIs can use any design style or protocol. WS use established protocols and standards (REST, SOAP).

The design concept of CRUD operations can also be applied to the World Wide Web (WWW) and its underlying HTTP protocol and is a very common pattern when creating web APIs. Data is made available through URLs

and can be accessed by the different HTTP methods GET, POST, PUT and DELETE that refer to the aforementioned CRUD operations. [Bus19], [Alt17]

2.6 Simple Object Access Protocol

Simple Object Access Protocol (SOAP) is a web service communication protocol. The protocol specifies guidelines that define how messages are sent over the network. The Extensible Markup Language (XML) is used to exchange data. XML is a markup language. A markup language is a set of codes that are used to structure the contents of digital documents. SOAP uses transport protocols (HTTP, SMTP, etc.) to ensure data transport. [Rel19]

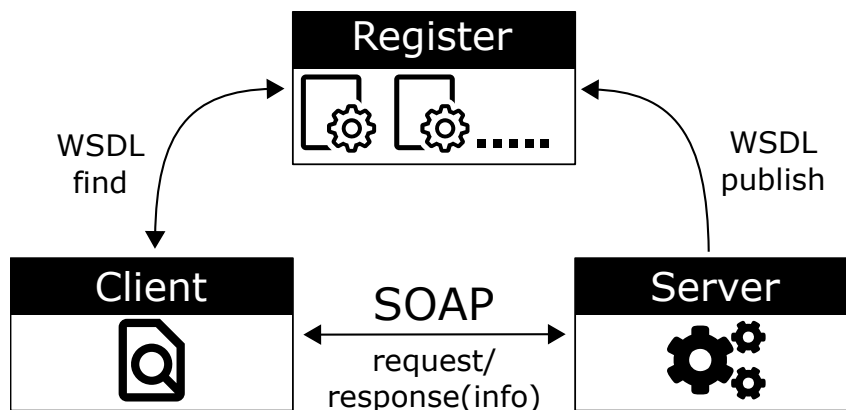


Figure 2.4: SOAP architecture. Own representation based on [HK11]

Figure 2.4 shows the SOAP web service architecture that consists of the three components: service requester, service provider, and service registry. Compared to REST, which is exposing the data of an application, SOAP exposes the functionality or logic of an application as a service.

The service requester is any type of software module or application that requires a service. The service provider accepts and executes the request from the service requester. WSDL is the XML file that describes the service. The service provider pushes the service description into the service registry.

The service requester finds the service description in the registry and can start to use the service. [HK11],

SOAP is also including and supporting other technologies and protocols such as WS-Security. WS-Security is a standard that specifies how security features are implemented in a WS to protect them from external attacks. In general, SOAP web services are more resource-intensive than REST services. SOAP services need more bandwidth and have lower performance compared to REST services. But there are areas where SOAP is preferred over REST. SOAP is used in applications that need higher security standards and have complex interactions (financial services, telecommunications services, etc.).

A comparison between REST and SOAP is illustrated in table 2.1. [TG16]

2.7 Representational State Transfer

Representational State Transfer (REST) is an architecture style that is designed to create loosely coupled applications over HTTP. RESTful applications make sure that important data of an application are represented as URI-identified resources. The main interaction methods with resources are the aforementioned HTTP methods GET, POST, PUT and DELETE. All interactions with a client through a server are stateless. This allows high scalability since the server does not have to maintain session states with clients. A full RESTful architectural design is defined by six constraints: [Fieoo]

- Uniform Interface
- Client-Server architecture
- Stateless
- Cacheable
- Layered System
- Code on demand (optional)

Uniform Interface: The central difference between REST architectural style and other styles is the importance of providing uniform interfaces between

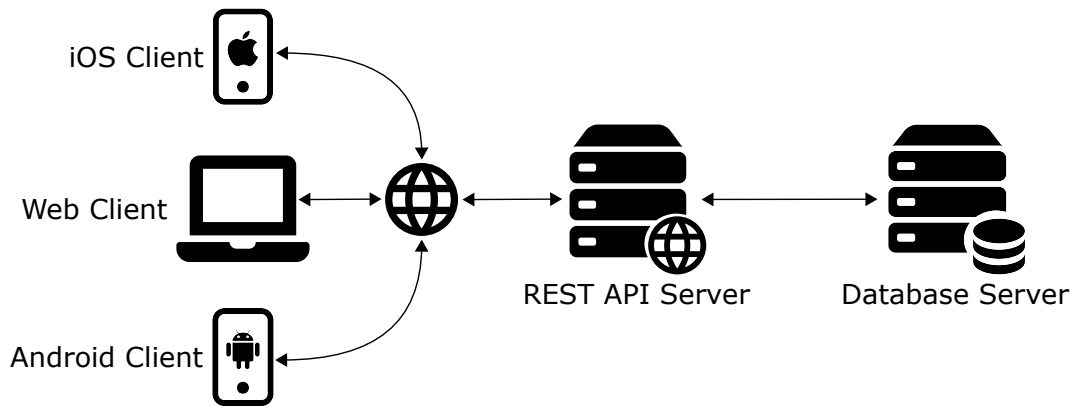


Figure 2.5: REST architecture. Own representation based on [Rel19]

components. A resource in a system has one logical URI, and that should provide a way to fetch and manipulate data. The most common HTTP functions in RESTful services to fetch and manipulate data are: GET, POST, DELETE, PUT. [Fie00], [Cos+14]

Client-Server architecture: REST applications should have a client-server architecture. A client should only request resources and should not be concerned about data storage. A server should be responsible for storing application data along with all the business logic required to interact with it. Business logic tasks include e.g. user authentication, authorization, and data validation. A server should not be concerned about the user interface. By decoupling the client and the server from each other they can evolve independently. The user interface can be transferred across multiple applications while scalability can be improved by simplifying server components. [Fie00], [Cos+14]

Stateless: Every request from the client must contain all the information that the server might need to properly execute it. The client can not take advantage of any stored session context on the server. Maintaining the session state is kept entirely on the client-side. [Fie00]

Cacheable: The data within a response to a request can be labeled as cachable or not cachable. If a response is cachable, the client can reuse this data for later, equivalent requests. Client-side caching can be used

to improve performance by reducing the average latency of interactions. [Fieoo]

Layered System: A client can not tell if he is directly connected to the server that returns the resources in response to a request. Authentication can be implemented on server A, APIs can be deployed on server B and the actual resource data can be stored on server C for example. A client can not tell whether it is connected directly to the end server or an intermediate server. [Fieoo]

Code on demand (Optional): This optional constraint describes that a client can also request executable code from the server. [Fieoo]

	SOAP	REST
Meaning	SOAP stands for Simple Object Access Protocol	REST stands for Representational State Transfer
Design	SOAP is a protocol with standards and specifications. SOAP includes a WSDL file that provides the client with the necessary information about the available web services	REST is an architectural style. A web service can be considered RESTful if it follows the constraints: <ul style="list-style-type: none"> - Uniform Interface - Client-Server Architecture - Stateless - Cacheable - Layered System - Code on demand (optional)
Message Transfer	Transport protocols, e.g.: HTTP, SMTP and more	HTTP only
Message Format	XML only	Many formats, e.g.: JSON, YAML and more
Procedure	Function-driven. Data is made available as a service	Data-driven. Data is made available as resource
Statefulness	Stateless by default. SOAP APIs can be made stateful	Stateless
Caching	Does not support caching for API calls.	API calls can be cached
Security	Supports standards, e.g: WS-Security	HTTPS and SSL
Performance	SOAP has more overhead and requires more bandwidth and resources <ul style="list-style-type: none"> - Standardized - High Security - Harder to develop 	Requests are lightweight and do not require much bandwidth and resources <ul style="list-style-type: none"> - Flexibility - Scalability - Developer friendly (easier to develop than SOAP)
Advantages	<ul style="list-style-type: none"> - Less performance - Complex - Less flexible 	<ul style="list-style-type: none"> - Lack of standards - Harder development of services with sophisticated requirements
Disadvantage		

Table 2.1: Comparison SOAP and REST. [HR18], [Monzo]

2.8 OpenAPI Specification

Many different API documentation standards have been proposed to make it easier for developers to create and work with APIs. The OpenAPI Specification (OAS) is the most prominent standard for developing REST-based WS. The standard describes an API in JSON or YAML markup language. Humans and machines are able to understand the capabilities of a service without the need to access the source code or documentation. OpenAPI documents describe which resources are available in the REST API and what operations can be called on those resources. The document also specifies a list of parameters that can be used to perform an operation on a resource as well as the type of those parameters and if they are optional or not. OpenAPI is providing additional tools for developers and testers to make API development more enjoyable. Swagger UI is one of them. [Iniz0], [KK18], [Sur16], [Sof20]

2.8.1 Swagger UI

Swagger UI is an open software tool that allows anyone to visually interact with the APIs resources on an HTML page. The HTML page will be automatically generated from the OpenAPI specification file. This tool is very convenient for developers to test if the APIs are working correctly. Developers can use the built-in testing functions to simulate client requests and inspect the resulting outputs of the operations [Sof20].

2.9 Content Management Systems

A Content Management System (CMS) is a server program that focuses on managing and providing content. In contrast to traditional websites where content is mixed with HTML elements, a CMS is separating content from the structural and technical basis of the system. Most CMS are organized in a Model View Controller (MVC) concept. This concept enables flexible updates of content but also layout and design since the content layer is

separated from the presentation layer. Content creators can focus on publishing content without the need for technical background knowledge. [Ste16], [PRP11]

Most CMS also provide a built-in user management system. Users of the system can be assigned to different roles. Each role is defined by a set of capabilities. Content creators are assigned to the role editor whereas consumers of the content are assigned to the role visitor for example. Administrators can extend the functionality of the system by installing plug-ins that can go live immediately. [Ste16]

A CMS comes up with many advantages compared to traditional created web sites. The following list will highlight the most important features of a CMS: [Ste16]

- Fast and easy setup for a web presentation
- Existing content can be updated/deleted fast and easy
- Different content can be provided to different users depending on their language or role for example
- Content can be created without programming skills
- System functionality can be extended by installing plug-ins
- Numerous free-to-use plug-ins available that accomplish a wide variety of tasks
- Creating and using customized plug-ins
- Corporate design through the whole system
- Free to use design templates are available that can be integrated without a huge effort
- MVC concept that is separating content, design, and program code
- Separation of the website into an area for visitors (frontend) and an administrative area (backend) that are both accessible via the same address

There are a lot of different CMS available on the market. The next section provides an overview of WordPress which is one of the most prominent CMS on the market.

2.9.1 WordPress

WordPress was initially designed as a Weblog Publishing System (WPS) which evolved into one of the most used CMS on the web. WordPress is an open-source project that is released under the General Public License (GPL). Users are free to modify and distribute the software however their needs call for. Some key arguments for choosing WordPress are: [Ste16]

- **Easy installation:** WordPress is often advertised with the famous five minute installation process
- **Open-Source software:** Commercial use of WordPress without restrictions
- **Using themes:** A huge selection of free-to-use design themes are available to customize the design of WordPress
- **Using plug-ins:** A huge selection of free-to-use plug-ins are available to extend the functionality of WordPress
- **Documentation:** A large community has formed over the past years that makes a lot of documentation available.

2.9.2 Custom WordPress Plugins

A WordPress plugin is a standalone set of code that extends the functionality of WordPress. New features can be added to any part of the existing website, including the administration area. Custom plugins can be created that are communicating with the WordPress environment using the API that is provided by WordPress. This API is ideal for internal processes that do not leave the WordPress environment. Creating plugins that are integrating external software systems into WordPress can be achieved by using the WordPress REST API that was introduced in 2016. [Uza16]

2.9.3 WordPress REST API

The WordPress REST API enables developers to interact with third-party applications that are outside of the WordPress environment. These applications might be coded in a different programming language and might

also run on a different machine. A custom WordPress plugin can use the REST API to load content from external services dynamically to extend the functionality of WordPress. Enabling cross-platform communication in WordPress helps developers to built new and more useful applications in WordPress. [Uza16]

3 Analysis

This chapter is analyzing the different systems that may be used inside an academic makerspace on the example of the SLFI. A layout of the Schumpeter Labor für Innovation (SLFI) is illustrated in figure 3.1. The SLFI is using two offline and one online system for access administration (section 3.3.1). Gantner is an online system that is explained in section 3.3.8. Primion and Hoffmann are offline solutions that are analyzed in section 3.3.3 and 3.3.12. The SLFI uses a WordPress system for content creation and other administrative tasks. This system is explained in section 2.9.

In the situation analysis (section 3.2) the current system architecture and the future system architecture are illustrated. The current system architecture shows how the systems are currently being used. The future system architecture shows how the systems should be used in the future.

By the end of this section, all necessary information to identify potential interfaces is available. This information includes the possibilities and limitations of the systems that are used to develop various concepts in chapter 4.

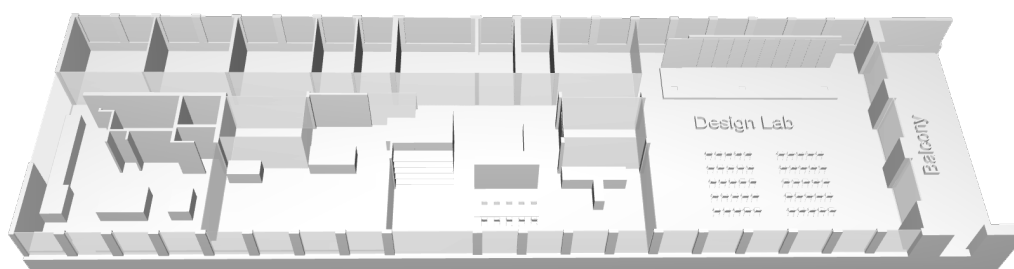


Figure 3.1: Schumpeter Laboratory for Innovation Layout

3.1 Schumpeter Laboratory for Innovation

The SLFI provides a platform to connect students, researchers, start-ups, and established industrial companies. The different user groups can use the SLFI for learning, prototyping, and exchanging ideas. The SLFI offers access to modern infrastructure with modern production machines. Students have the opportunity to participate in practically oriented courses such as design thinking and rapid prototyping. In this course, students will be encouraged to use the high-performance production facilities and infrastructure of the SLFI to build prototypes based on their acquired technical skills. [Inn2ob]

Having so many different user groups in the SLFI also raises challenges. Not everyone should have access to the expansive equipment. Only members with proper training may use the machines. The wardrobe, however, can be used by anyone. Depending on the user group, they should only have access to certain premises and storage options. The accesses to the premises and storage facilities within the SLFI are equipped with electronic locks from various manufacturers. Each manufacturer has its specification of how the electronic locks must be operated. In order to offer users the best possible stay at the SLFI, these electronic locks should be as uncomplicated to use as possible. Users should be able to use the different locks with a uniform data carrier. Administrators should be able to manage the various locks on a uniform platform. To provide such a unified platform, this section analyzes all relevant systems in more detail.

The SLFI is organized in several different premises (figure 3.2) and include:

1. FabLab I
2. FabLab II
3. Lobby
4. Office Rooms
5. Tesla Conference Room
6. Schumpeter Conference Room
7. Design Lab
8. Wardrobe
9. Terrace

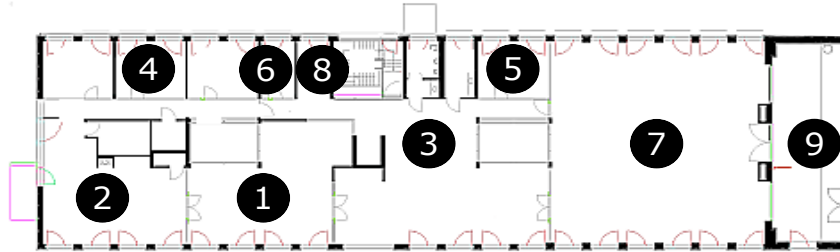


Figure 3.2: Schumpeter Laboratory for Innovation Areas

FabLab I/II

FabLab I and II are equipped with modern production machines such as laser cutters, waterjet cutters, CNC-milling machines, 3D printers and scanners, a PCB printer, and electronic workstations. These areas are designed for rapid prototyping. [Inn20a], [Inn20b]

Lobby and Office Rooms

Apart from the reception area, the lobby is designed to relax and share knowledge. The office rooms will only be used by the employees of the institute. [Inn20b]

Conference Rooms

The meeting rooms are equipped with touchscreen monitors and video conferencing material to support online meetings. [Inn20b]

Design Lab

The Design Lab is the biggest area of the SLFI. This multifunctional area can accommodate up to 120 people and is used for various events including panel discussions, workshops, and lectures. Equipped with state-of-the-art

technique (4K LED video wall, sound, video conferencing material), the Design Lab is also used for non-university events such as conferences and product presentations. [Inn2ob]

Wardrobe

The wardrobe provides storage options for all guests, members, and employees of the SLFI. The wardrobe offers 85 lockers in different sizes to store personal equipment. [WSR20]

3.2 System Overview

This section illustrates the current system architecture and the future system architecture that should be achieved. The current system architecture with all the important systems is summarized in figure 3.3. The desired future system architecture is shown in figure 3.4.

3.2.1 Current System Architecture

The current system architecture shows all the relevant systems that are described in the system analysis and how they are interacting in the SLFI.

The computer in the reception area of the SLFI is the only device that has access to the Gantner system. The Gantner system is managed by the GAT Relaxx Desktop Client that is installed on the computer. The client is communicating with the GAT Relaxx Service as shown in figure 3.12. This service is running on an external server managed by the ZID. The server is accessed by the computer with a preconfigured static IP address.

The Primion system is managed by the ZID. There is no direct API that allows interaction with that system. TUGOnline provides a GUI that allows access administration for authorized personnel as shown in figure 3.10.

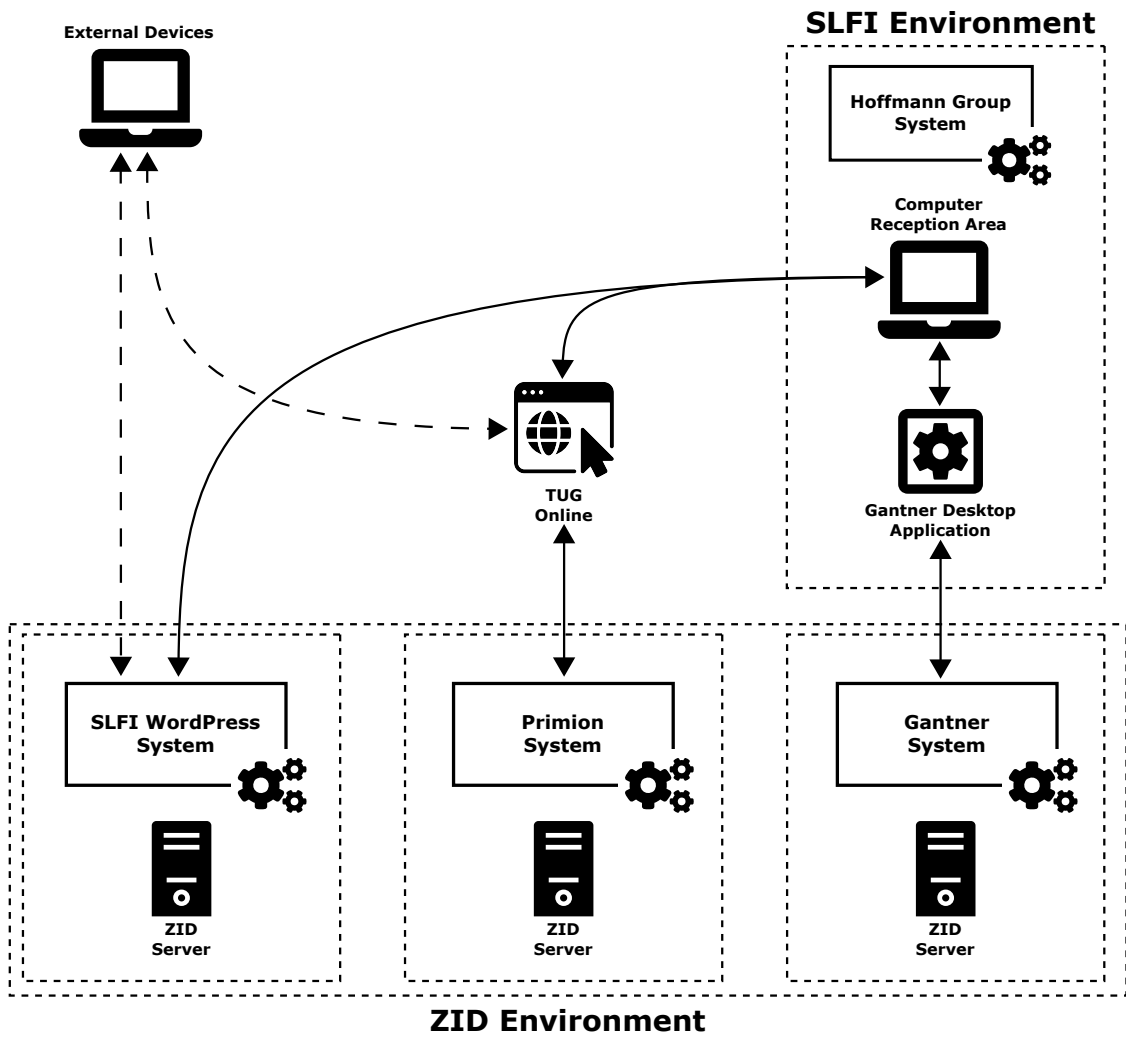


Figure 3.3: SLFI Current System Architecture

The Hoffmann Group system is not connected at all and provides no accessible interfaces.

The WordPress system is responsible for publishing content (e.g.: upcoming events), hosting the website (<https://fablab.tugraz.at/>), and other administrative tasks. Various projects in the past have extended the functionality of this system. The WordPress system is hosted on a ZID server. This server is accessible with the File Transfer Protocol (FTP). New data can be uploaded to this server without any external restrictions.

Figure 3.3 shows that all systems are not well connected with each other. The Primion system can only be accessed within the TUGOnline website. GAT Relaxx Desktop Client is used to manage Gantner lockers and can only be used by the central computer of the institute. The Hoffmann system is not accessible with any interfaces.

3.2.2 Desired System Architecture

The future state shows how the different software solutions should be made accessible by using one common platform. This platform should communicate with all other systems and can be accessed by external devices that do not reside inside the SLFI. This makes it possible to administrate the different systems more flexibly.

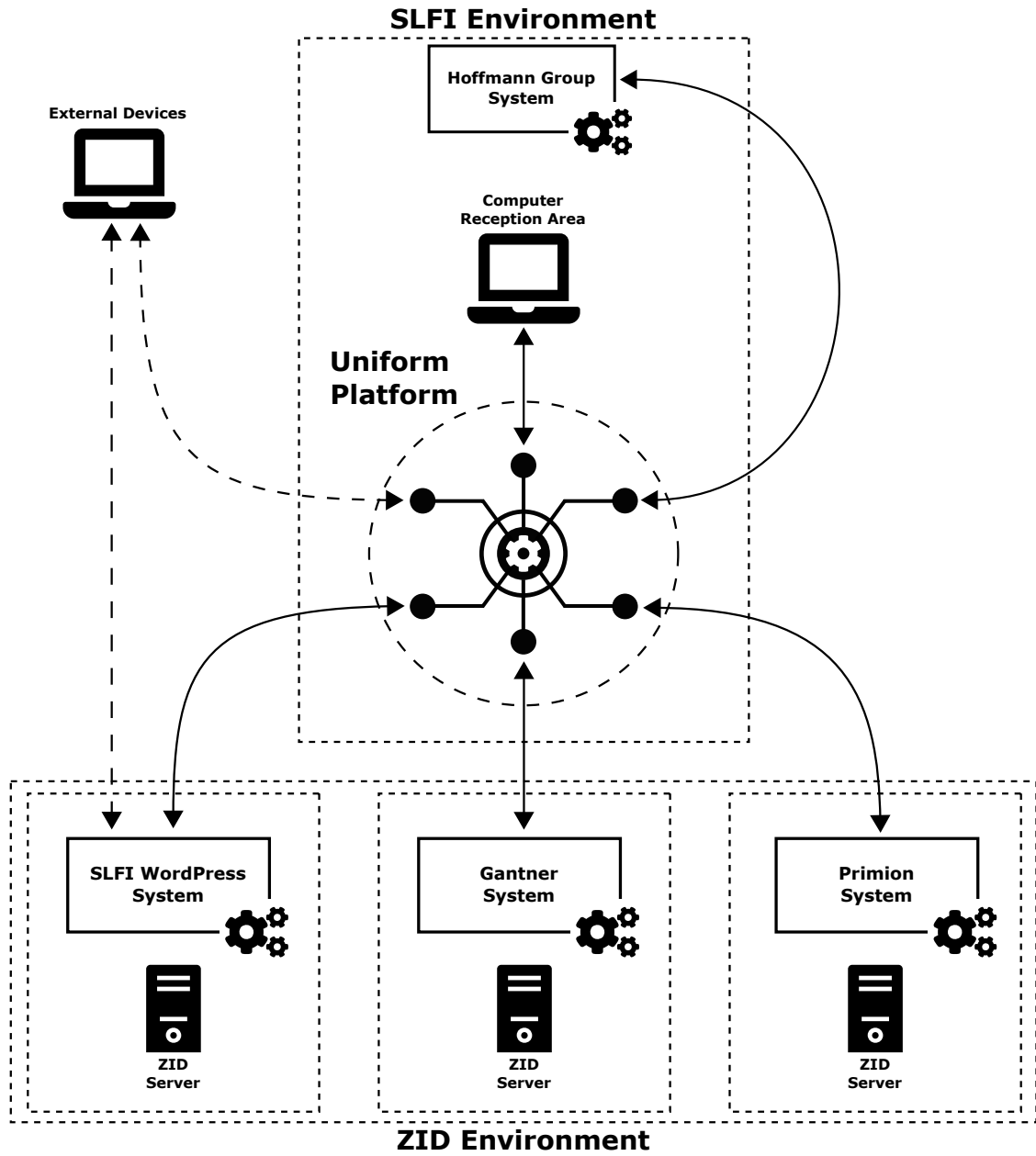


Figure 3.4: SLFI Desired System Architecture

3.3 System Analysis

In a universal place such as the SLFI, organization and management is an important factor. Access rights are required to enter the building and the SLFI. In-house training sessions are mandatory to use the high-quality equipment in FabLab I and II. Participating in training sessions is only possible after registration. Offering all the services of the SLFI does therefore need a proper level of organization and management. The management decided to integrate new hardware and software solutions. The next sections explain the most important systems that are used in the SLFI.

3.3.1 System Types

Two types of access control solutions are used in the SLFI: online and offline solutions. Access rights in an online system are stored in a database. The locks have access to the database and can read out the access rights directly. Gantner (section 3.3.8) is an online solution and is used for storage options in the wardrobe, lobby, and reception area of the SLFI.

Offline solutions follow the *Data on Card* approach. Access rights are written and stored directly on the data carrier. The locks are not connected (wired) to the system. Instead, they function autonomously and are powered by a battery. The locks read the access rights from the data carrier and will remain locked or unlocked accordingly. Primion (section 3.3.3) and Hoffmann (section 3.3.12) are two offline solutions. Primion is used by the ZID to manage authorizations on the campus of TU Graz. Hoffmann is used for storage options inside FabLab I and II. [Ass21]

3.3.2 Data Carriers

The students and employees at TU Graz are using MIFARE Classic and MIFARE DESFire EV₁ smart cards. MIFARE is a product line of contactless smart cards provided by NXP Semiconductors. This product line includes various products that differ in security, memory capacity, functionality, and

price. Multiple different applications can be stored on a MIFARE smart card. Applications include access control, school and campus cards, public transportation, and more. As explained in section 3.3.3, the ZID of the TU Graz provides the infrastructure to use MIFARE smart cards for access regulation. [WSR20], [Sem21]

Applications are stored and managed in sectors on a MIFARE smart card. The number of applications that can be stored depends on the available sectors on the smart card. Higher memory capacity provides more sectors. A MIFARE DESFire EV1 smart card can hold up to 28 different applications. New applications must be registered on the smart card. NXP offers an official registration form for new applications. Developers of a new application are responsible to determine if the NXP products are suitable for their product, as well as for future third-party customers. Developers are also self-responsible for the correct design, implementation, and operation of the new application. [Sem]

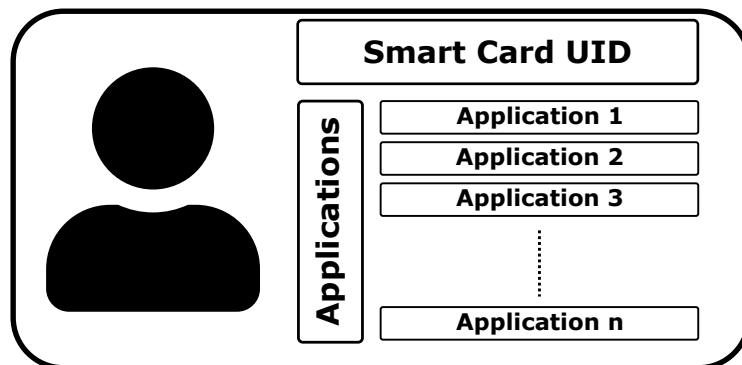


Figure 3.5: Simplified own representation of a smart card data carrier containing multiple applications.

The UID of the smart card is readable and can be used for Gantner authorizations as explained in section 3.3.8. Manufacturers of offline locks are responsible for ensuring that the locks can read the data correctly from a smart card. This raises the problem that different manufacturers have different solutions for interacting with the data carrier. The *OSS Standard*

Offline provides a solution for this problem as described in this section. [Sem21]

OSS Standard Offline

Offline solutions have one common problem. Manufacturers have their own approaches to write access rights to a smart card. Locks of different brands can not interpret and read the access rights from the card. The organization OSS Association provides a common standard to face this problem. The OSS Standard Offline defines how information is written on the card and read from it. This common standard allows locks of different brands to read and interpret the access rights from the card. There are already software solutions available that support this standard, including Primion. [Ass21]

3.3.3 Primion Access Control System

Primion is providing a wide variety of access control solutions that can reach from simple locking systems to highly complex security systems. Complete buildings, specific areas, or individual doors can be protected. Therefore, Primion offers a wide range of in-house hardware and software products. The Graz University Computer Center (ZID) is using Primion solutions to regulate access to buildings. Access regulation of almost every building that belongs to the campus, including the SLFI, will be managed by primion solutions. [Tec20]

Prime Key Technology

The prime key technology (PKT) provided by Primion is an offline solution for access control regulations. Sensible areas are equipped with mecha-tronic components such as digital cylinders. These components require no additional equipment or wiring. Central points of access to a building are equipped with online readers (master readers) that are connected to a control unit. All components and access rights will be configured and managed with custom software. Users need to update their identification

medium (chip, card) at the online readers regularly to get their access rights. The mechatronic components can then be opened with the updated identification medium. Figure 3.6 shows the PKT concept. [Tec20]

3.3.4 How is the Primion System used inside the SLFI?

The ZID of the TU Graz is using Primions PKT solution to manage access control on the campus. The SLFI is located on the third floor of a building on the TU Graz campus. A PKT master reader (figure 3.7) is placed in front of the main entrance of the building. The master reader is connected (wired) to a control unit and is used to update the access authorizations on the data carriers. The TU Graz uses the contactless smart card products MIFARE Classic and MIFARE DESFire EV1 that are provided by NXP Semiconductors. The main entrance of the building is equipped with a mechatronic lock. The lock opens if a valid authorization is stored on the data carrier. [Tec20], [WSR20], [Sem21]

The entrance to the SLFI on the third floor and the elevator inside the building are equipped with PKT offline readers (figure 3.8). These readers are capable of reading access rights from a data carrier. The door to the SLFI can be opened if a valid authorization is stored on the data card. Getting access to the SLFI by using the elevator is also requiring a valid authorization on the data carrier. [WSR20]

Electronic offline door locks (figure 3.9) are used inside the SLFI. The door locks support the standards MIFARE Classic and MIFARE DESFire EV1 and are equipped with long-lasting battery packs. The door lock can be opened if a valid authorization is stored on the data carrier. Figure 3.11 shows the mechatronic door locks that are installed inside the SLFI. [WSR20]

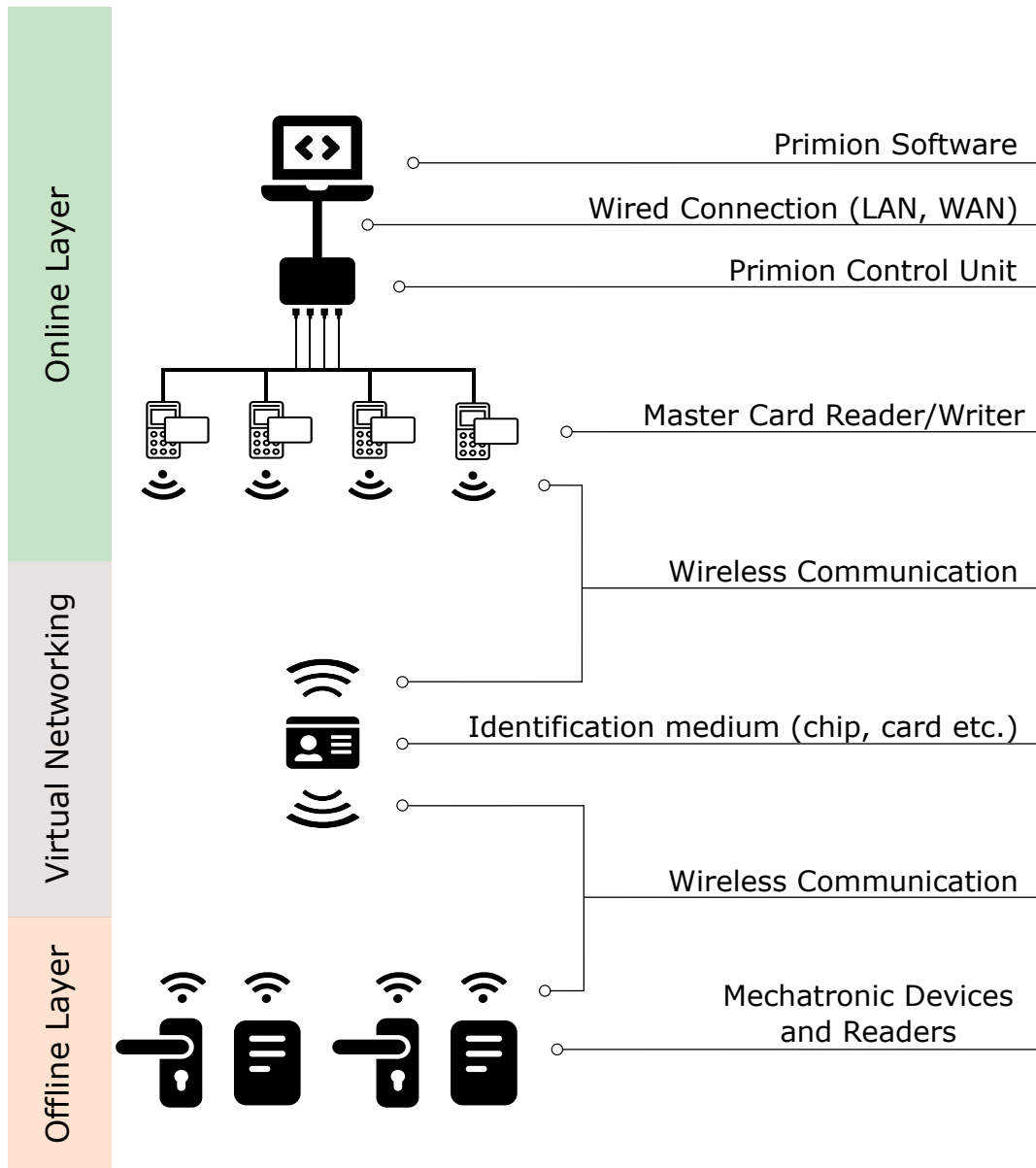


Figure 3.6: Primion Prime Key Technology, Own representation based on [Tec20]



Figure 3.7: Primion PKT master reader, [Pri21]



Figure 3.8: Primion PKT offline reader, [Pri21]



Figure 3.9: Primion PKT offline electronic door, [Pri21]

3.3.5 ZID Access Control Management

Access management on the premises of the TU Graz is performed regularly. Primions custom software solution is not directly accessible for everyone. A customized graphical user interface (GUI) is provided by the Campus Online Management software (TUGOnline). Authorized individuals of the TU Graz are granted rights to use this interface to manage access controls. Relevant information will be provided through this interface and send to the Primion software. Once per day all newly added authorizations will be imported into the Primion software. A CSV import is used to get the

data from TUGOnline into Primions management software. Figure 3.10 summarizes this procedure.

Primion Resources

The aforementioned Primion offline readers and door locks are represented as *resources* in the Primion software. TUGOnline allows managing this resources in a human-readable form. A CSV entry is created whenever a person gets access to a specific room or area. The CSV entry holds information about:

- The person such as name, data carrier unique identification number
- The mapped resource name that represents the room or area the person gets access to
- The period of validity of the authorization

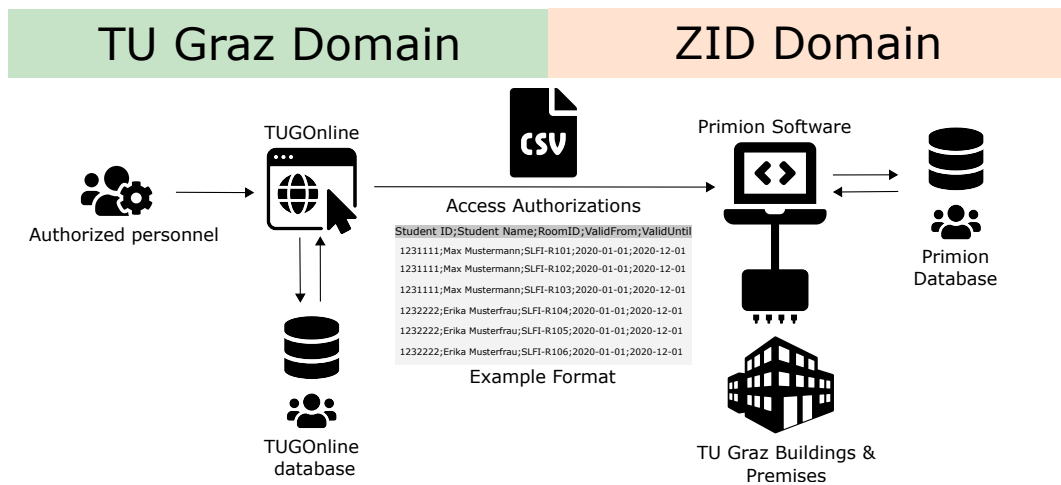


Figure 3.10: ZID Access Control Management

3.3.6 Primion System Limitation and Possibilities

The Primion system is used by the ZID to manage resources all over the campus including the SLFI. The major limitation of this system is that there are no APIs available that allow direct interaction with the Primion software. The ZID is running this system encapsulated from the rest of the campus. The TUGOnline platform must be used to manage Primion resources. Another huge limitation is data security. The ZID is not allowed to give third-party applications access to any user information. This would violate the legal requirements specified in the General Data Protection Regulation (GDPR). The ZID also ensures that all authorization updates are secure. Fake entries could lead to giving unauthorized persons access to premises they should not have access to. Therefore, the ZID is keeping the authorization management in a secure environment that is encapsulated from any third-party application. Direct communication with the Primion software is not possible. [WSR20]

Although there are no APIs available that allow direct communication with the Primion software, a CSV import can be used to manage authorizations. This approach was also discussed directly with the responsible persons from the ZID. A simplified representation of the data flow of the Primion system is summarized in figure 3.6. The most promising approach to access this system is a CSV import. A CSV import is already being used by the ZID. The Primion system is getting updated daily with CSV data to manage the user authorizations on the campus.

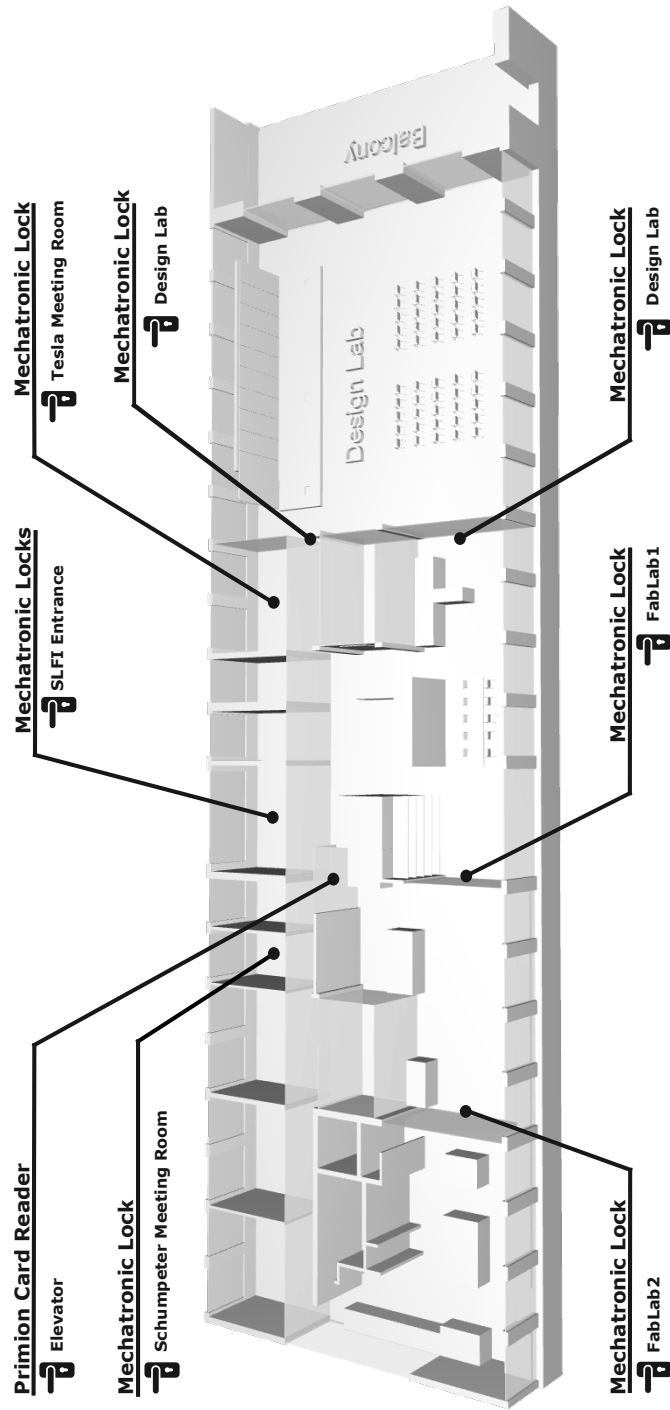


Figure 3.11: Overview of Primion Locks in the SLFI

3.3.7 Primion Summary

To summarize the most important points:

- The Primion system is managed by the ZID
- Authorized individuals can grant access permissions within TUGOnline
- Student cards are used as data carriers (MIFARE Classic and MIFARE DESFire EV₁)
- The ZID is not allowed to grant access to user information (name, card id)
- No APIs are available to access the Primion software
- CSV imports are used to update authorizations

3.3.8 Gantner Access Control System

Similar to Primion, Gantner is another big company providing locking solutions in various sectors including access control, cabinet locking systems, and accounting systems. The locks are compatible with various data carriers including bracelets, key rings, or employee identity cards. The uniform identifier of a data carrier will be used to assign a locker to a person. All lockers are directly connected to a management service provided by Gantner. Online solutions have the advantage of managing the lockers in real-time compared to systems that are not wired. Gantner is providing their own in-house software solution to manage the lockers. The management software can either be used as a standalone application on a working station or using a web interface.

A complete system description of the locker management solution from Gantner is shown in figure 3.12. [Gmb20]

3.3.9 How is the Gantner System used inside the SLFI?

The SLFI uses the Gantner system to manage lockers in the wardrobe and lobby area. A total of 85 lockers are used in the wardrobe area. These lockers provide storage options for guests of the SLFI. A total of 12 lockers are used in the lobby area. Six of the lockers in the lobby area are used for projects.

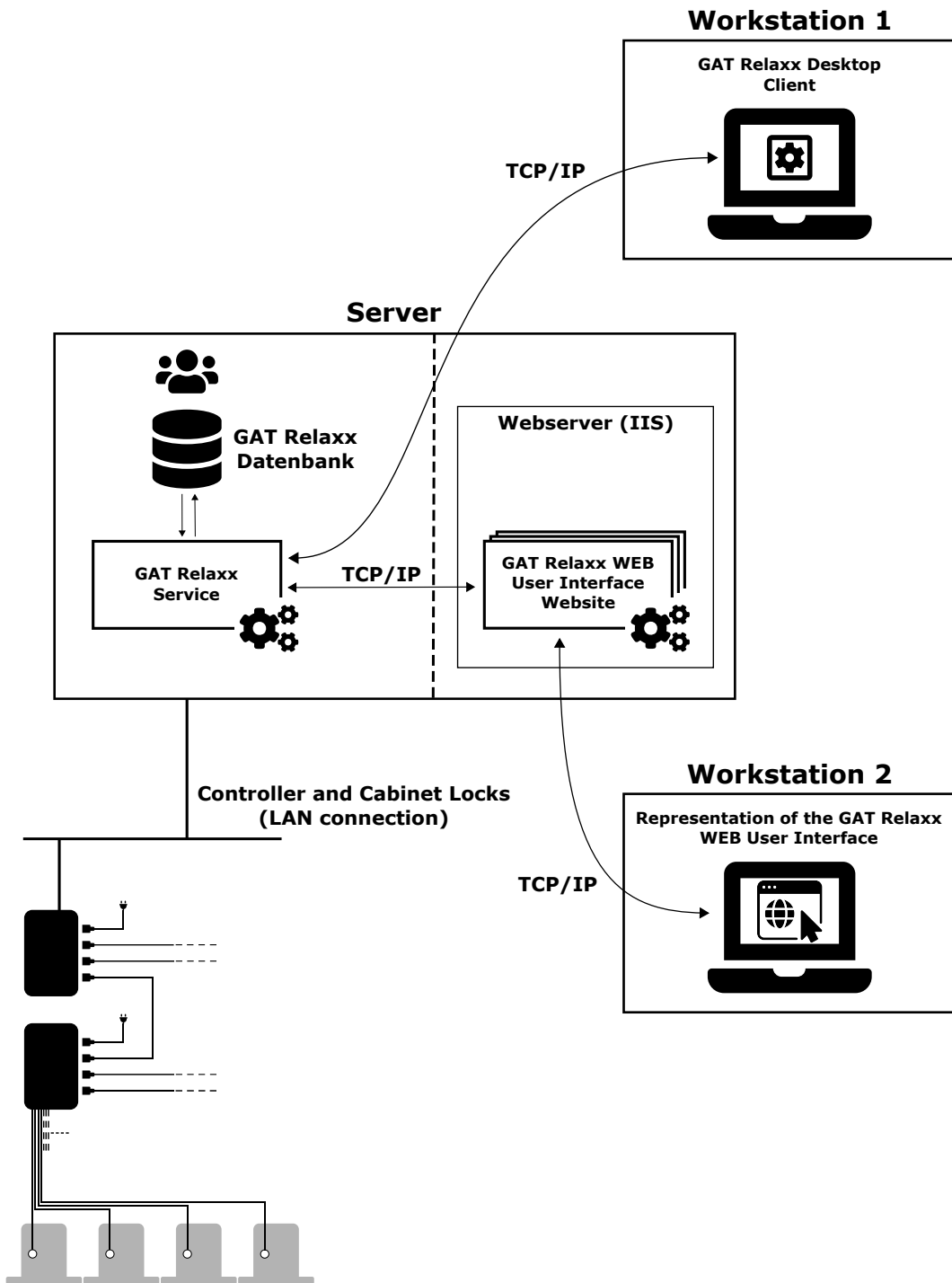


Figure 3.12: Gantner Technology, Own representation based on [Gmb20]

Guests who are working on long-term projects can use these lockers for stowing. The other half of the lockers in the lobby area are used by the reception. To manage all those lockers the SLFI is using the GAT Relaxx desktop client. This client is installed on the working station in the reception area. The Gantner server (GAT Relaxx service, database, etc.) is managed and operated by the ZID. The working station in the reception area is accessing the services of Gantner with a preconfigured static IP address. Figure 3.13 shows the Gantner locks that are used in the SLFI. [Gmb20], [WSR20]

3.3.10 Gantner System Limitation and Possibilities

The computer in the reception area is the only device that has access to the Gantner services. Gantner is providing additional interfaces. One interface is TCP-based network communication. Another way to interact with the system is to use the web interface. The web interface provides REST endpoints that can be used by software developers. The REST endpoints allow interaction with common authorization management tasks. These tasks include reading, creating, updating, and deleting authorizations in the Gantner system. Gantner is also supporting CSV imports.

Minor limitations are that the REST API is still in development. This means that some information about the lockers and authorizations is currently not available. Another limitation is the hardware configuration. Adding additional Gantner lockers to the SLFI still requires the GAT Relaxx client software to configure the new hardware components. The new lockers can be managed with the REST API endpoints after the configuration.

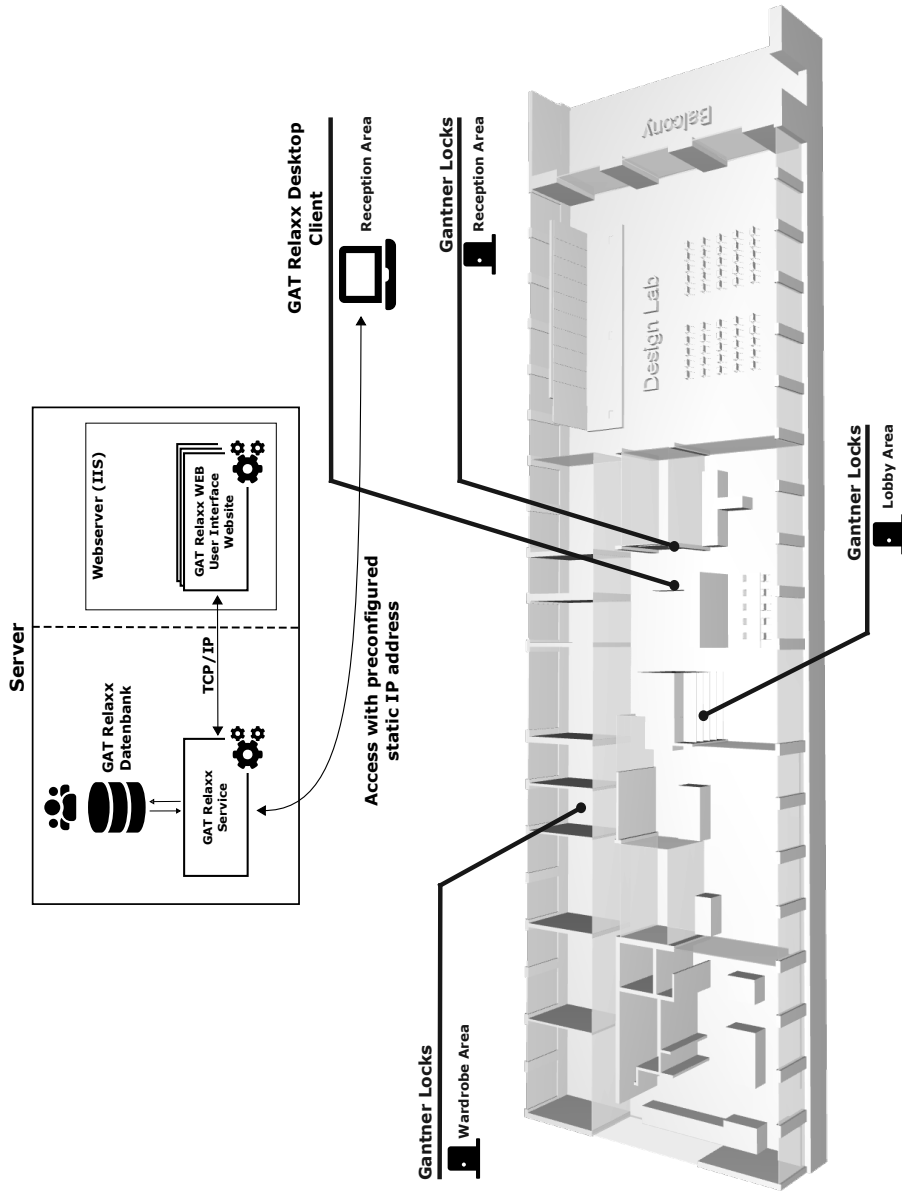


Figure 3.13: SLFI Gantner System Overview

3.3.11 Gantner Summary

To summarize the most important points:

- GAT Relaxx desktop client software is installed on the computer in the reception area
- GAT Relaxx desktop client must be used for hardware configuration
- Multiple APIs available: TCP-based network communication and REST API endpoints
- Student cards can be used as data carriers (MIFARE Classic and MIFARE DESFire EV1)
- CSV import is possible

3.3.12 Hoffmann Group Access Control System

Hoffmann Group is a provider of a wide range of factory equipment including electronic locks. The SLFI is planning to use the GARANT Electronic Lock System (G-ELS). G-ELS locks are installed in the various storage options inside FabLab I and FabLab II (Figure 3.19). [Gro21a]

3.3.13 GARANT Electronic Lock System (G-ELS)

The G-ELS system is an offline system that can be used in two different modes: [Gro21b]

- PIN mode
- TAG mode

PIN Mode

First, the lock must be initialized with the master card. The lock can then be configured in this mode. The configuration requires to specify a four or six-digit pin code. The lock can then be manually opened and closed with the pin code.

TAG Mode

In this mode, the configuration will be done with an Android app. The app is developed by Hoffmann and can be downloaded for free. The TAG mode can be summarized as followed:

Android App Setup: In order to enable the TAG mode it is necessary to initialize the master card with the Android app. The initialization will be done using the NFC interface of the mobile device.

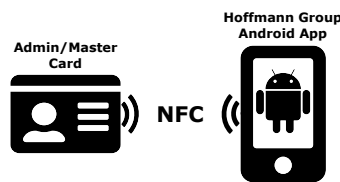


Figure 3.14: Android App Setup with Master Card. Own representation based on [Gro21a]

Lock Setup: After successful initialization, locks and users can be registered in the Android app. The NFC interface will be used by holding the mobile device close to the lock.

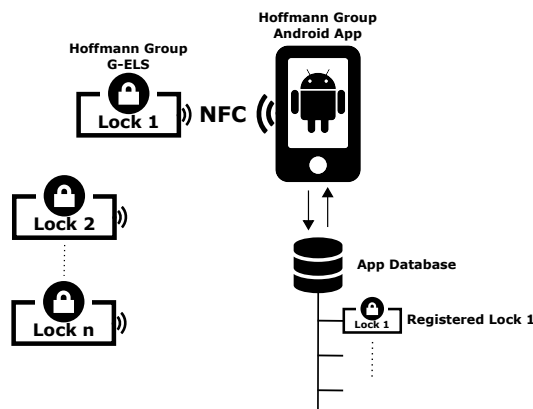


Figure 3.15: Register a lock in the Android App. Own representation based on [Gro21a]

User Setup: Users are also registered with the NFC interface. Holding the mobile device close to the identification device (tag) will register a user in

the app. Tags are using the MIFARE DESFire technology. Therefore it is also possible to use the TU Graz card as an identification device (tag).

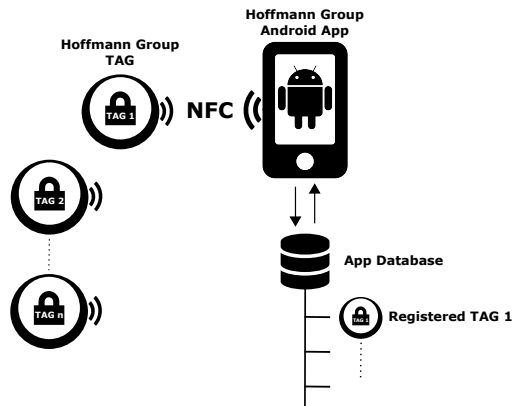


Figure 3.16: Register a TAG in the Android App. Own representation based on [Gro21a]

Assign User to a lock: Locker and user management will be done in the Android app. Users can be assigned to available locks in the app. Additional lock-specific configurations can also be performed in the app.

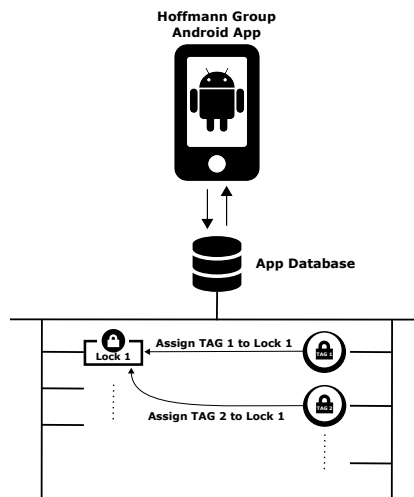


Figure 3.17: Assign multiple TAGs to one Lock. Own representation based on [Gro21a]

Synchronization: To apply the changes to a lock it must be synchronized. This will be done again with the NFC interface. All lock-related changes will be synchronized by holding the mobile device close to the lock. After successful synchronization, the lock will be able to use.

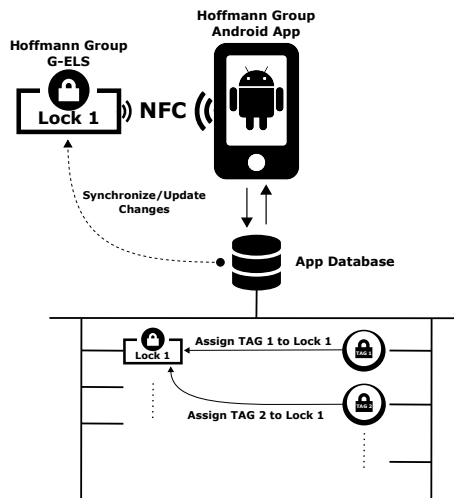


Figure 3.18: Synchronize changes with the lock. Own representation based on [Gro21a]

All locks are operating using battery power. The lifetime of the batteries is about 3 years when using the TAG mode and about 1,5 years when using the PIN mode.

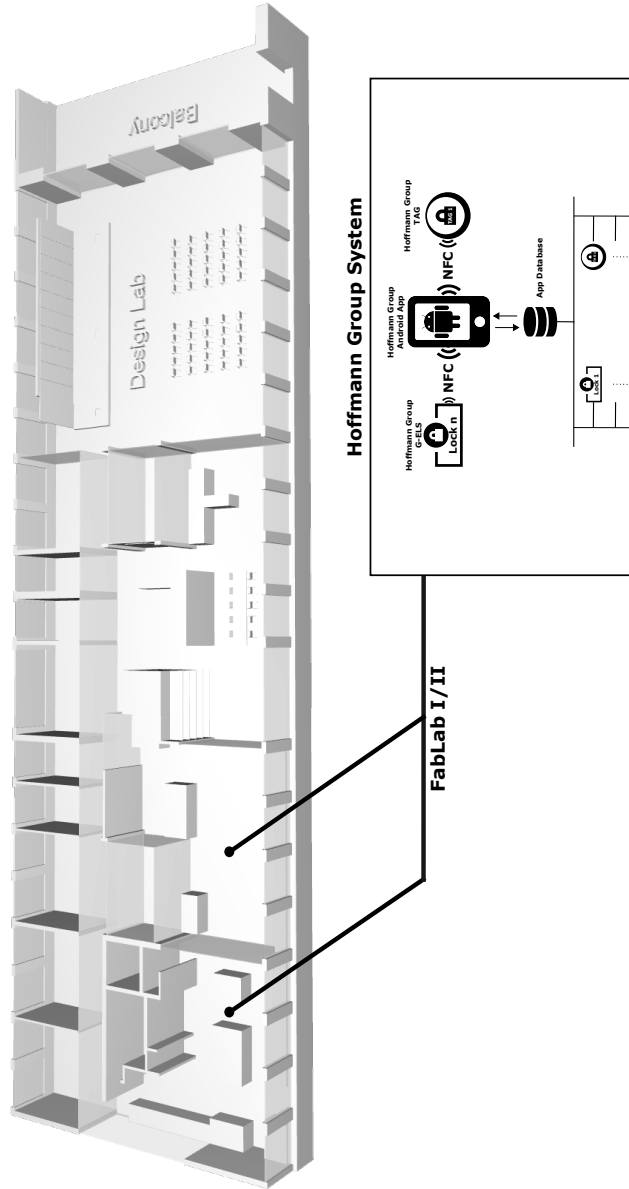


Figure 3.19: Overview of Hoffmann locks in the SLFI

3.3.14 Hoffmann System Limitation and Possibilities

One major limitation of this system is that Hoffmann locks can only be managed with the mobile Android application. This app does not provide any APIs. First, it was only possible to manage the locks with data carriers (tokens) that are also provided by Hoffmann. This limitation was solved by upgrading the Hoffmann locks in expansion phase I (section 4.2). The locks can now be managed with data carriers that support the MIFARE Classic and MIFARE DESFire EV₁ standards. Another limitation is that the locks of the Hoffmann system do not support the *Data on Card* technology. This upgrade needs additional external programming effort. This upgrade would create the possibility to integrate the Hoffmann system into the Primion system. Additional resources need to be created by the ZID. These resources can then be administrated inside TUGOnline as described in figure 3.10. Providing an interface directly to the mobile application would also allow the integration of the system. A third-party application would be able to update the authorization status on the mobile app. A direct interface to the mobile application would be an additional requirement to expansion phase I. This additional requirement is feasible according to an interview with responsible persons from Hoffmann.

3.3.15 Hoffmann Summary

To summarize the most important points:

- G-ELS locks can only be managed with the mobile application provided by Hoffmann
- The mobile application does not provide any APIs
- G-ELS locks do not support the *Data on Card* technology
- Custom Hoffmann tokens must be used as data carriers
- Upgrading the locks to support *Data on Card* allows the integration into the Primion system

3.3.16 SLFI WordPress System

A WordPress system is used for content management at the SLFI. Content management includes hosting the SLFI home page (<https://fablab.tugraz.at/>), adding information about upcoming events, custom user management, and much more. Content about new events is uploaded regularly. These events include open days, training sessions, makerthons and more. To be able to register for these events a user must have an account. WordPress provides a custom user management system as mentioned in section 2.9. The SLFI is using this system for its custom user management. People can sign up and create an account at <https://fablab.tugraz.at/>. With an account users can register for training sessions (e.g.: basic training). These training sessions are needed to get access to the tools and equipment in the FabLabs.

As explained in section 2.9, WordPress provides a lot of different ways to extend the system. A powerful tool for developers is the provided REST API (see 2.9.3). The REST API allows developers to integrate third-party applications into WordPress.

3.3.17 SLFI Card Reader

A card reader is located in the SLFI. The card reader can read the IDs of data carriers including MIFARE Classic and MIFARE DESFire EV1 data cards. The reader displays the IDs in hexadecimal format. Depending on the system a decimal format might be used for managing authorizations. The transformation from hexadecimal format to decimal format needs to be performed if needed. There is currently no option to automatically store the data carrier IDs into the WordPress system. Additionally, there is no option available to map a card ID to a user.

3.3.18 Systems Summary

The following table 3.1 summarizes the key factors of the three systems. These key factors were taken into account for the concept development. In addition to the key factors of every management system, the SLFI WordPress system (see 3.3.16) and the SLFI card reader (see 3.3.17) are also important components to consider. The WordPress system is already providing a central platform for content management in the SLFI. The card reader can read student card IDs (MIFARE Classic and MIFARE DESFire EV₁).

System Type	Gantner	Primion	Hoffmann
Supported data carriers	online various including MIFARE Classic and MIFARE DESFire EV1 cards	offline MIFARE Classic and MIFARE DESFire EV1 cards	offline custom tokens, MIFARE Classic and MIFARE DESFire EV1 cards
Current Management Tool	GAT Relaxx Desktop Client	TUGOnline	Hoffmann custom Android application
Application Programming Interface	TCP-based network communication and REST endpoints	none	none
Alternative Interface	CSV import supported	CSV import	none
Future expansions	REST API is being constantly extended	CSV merge and import. Adding new resources for external systems.	API for mobile application. Expansion phase II (see 4.2)

Table 3.1: Key factors of all access regulation systems that are used for the concept development.

4 Concepts

This chapter explains the developed concepts that meet the defined requirements. The defined requirements are:

- a central platform for administrators to manage the different locking systems
- a uniform data carrier that can be used to operate the various locking systems

The concepts were developed based on the system analysis described in section 3.3. The Hoffmann expansions (phase I and II) explained in section 4.2 are also included in the concept development process. Concept I and II explains how the already used WordPress platform can be used to fulfill the goal of a central administration platform. Concept III shows how the systems could be managed within the existing TUGOnline platform. The last part of this chapter explains the decision for one of the concepts.

4.1 System Boundaries

For all concepts, it is required that a uniform data carrier is used through all systems to manage authorizations. The uniform data carriers are the MIFARE Classic and MIFARE DESFire EV1 university ID cards that are used by students and employees (section 3.3.2). All concepts are developed to work with these university ID cards. The concepts assume that relevant student card ID information is available for authorization management. Therefore, the concepts do not provide a solution on how a card ID should be assigned to a system user. The concepts suggest possible approaches to make the card ID information available.

4.2 Hoffmann G-ELS Adaptation

Two adaptations were planned to use the G-ELS locks in the FabLabs:

- Expansion phase I: Upgrade the locks to work with data carriers that support the standards MIFARE Classic and MIFARE DESFire EV₁
- Expansion phase II: Upgrade the locks to read access rights from a data carrier (support *Data on Card* technology)

Expansion phase I is already completed. Data carriers that support the standards MIFARE Classic and MIFARE DESFire EV₁ can be used as tags (see section 3.16). The remaining steps: assigning the tags to the locks (see 3.17) and synchronizing the changes (see 3.18) must still be performed. The locks can be opened with valid authorizations on the data carrier after synchronizing the status on the Android app with all locks.

Expansion phase II is not completed yet. This phase focuses on upgrading the locks to work with the *Data on Card* technology (section 3.3.2). The locks should be able to read access rights directly from the data carrier. Reading access rights from a data carrier would allow this system to be integrated into the Primion software. New Primion *resources* must be created that represent the Hoffmann locks. These newly created *resources* can then be managed within the TUGOnline platform the same way Primion *resources* are managed. Creating new resources for the Hoffmann system is confirmed by the responsible persons from the ZID. Detailed integration steps of the Hoffmann system into the ZID environment are to be discussed after completion of expansion phase II.

4.3 Concept Analysis

Concepts I and II are based on extending the running WordPress application. WordPress is frequently used by the employees of the SLFI. Announcements to upcoming events and bookings for training sessions are administrated on this platform. Extending the existing WordPress application meets the requirement of a single access point for all three applications. Concept I

and II describe a WordPress extension that takes into account the limitations and possibilities of the systems described in section 3.3. Another approach is to integrate the systems into the ZID domain. Concept III describes the procedure of integrating Hoffmann and Gantner into the ZID environment.

- **Concept I:** WordPress integration: Gantner REST API integration. CSV export for Primion. Providing REST API for Hoffmann
- **Concept II:** WordPress integration: Gantner REST API integration. CSV export for Primion and Hoffmann
- **Concept III:** ZID integration: Integrate Gantner and Hoffmann system into ZID environment

4.3.1 Concept I

The WordPress REST API (section 2.9.3) makes it possible to integrate third-party applications into the WordPress environment. Custom plugins are already used in the SLFI. Registering for training sessions as well as a ticketing system are plugins that were created in the past. Built-in plugins to create content for the website are also used. Upcoming events (makerthons etc.) are published with the built-in functionality of WordPress. Figure 4.1 illustrates how the three systems can be integrated into a custom WordPress plugin.

The Gantner system (section 3.3.8) provides REST API endpoints that can be used by third-party applications. These endpoints include the creation, deletion, and modification of access rights. Because Gantner is an online system, authorization updates will be performed immediately.

The two offline solutions Primion and Hoffmann do not provide any APIs. Primion is using a CSV import to update the access rights (section 3.10). Even though there are currently no APIs available, the plugin can create a CSV file that can be imported.

To import the CSV file in the Primion system the following specifications must be clarified with the responsible persons from the ZID:

- CSV file transfer
- CSV file format

A secure file transfer must be established. It must be ensured that no unauthorized entity can modify the CSV file. In addition, the process of transferring the file to the ZID must be automated.

The format of the CSV file needs to be specified. This includes all data that needs to be present in the CSV file for a valid import operation.

The ZID is importing the CSV file once a day. Updates on the authorizations are not performed immediately. The data carrier must be updated regularly on a master reader.

The existing WordPress database can be extended to simulate information about the Primion authorizations. Authorizations for all Primion components used in the SLFI can be mapped in the WordPress database. The Primion authorizations can then be provided as a CSV file that can be imported by the ZID.

The completed expansion phase I of Hoffmann allows MIFARE Classic and MIFARE DESFire EV1 cards to be used as data carriers. The existing WordPress database can be extended to simulate information about the Hoffmann authorizations. This information can be retrieved from a provided REST endpoint and imported into the mobile application from Hoffmann. The Hoffmann system administers the access rights using their custom mobile application (section 3.3.12). The plugin provides a REST endpoint to update access rights in the mobile application. An API to the mobile application allows updating the authorizations from a third-party application. Providing an API to the mobile application is feasible according to an interview with responsible persons from Hoffmann and needs to be further specified.

Using an API to update the user authorizations in the mobile application still requires the synchronization step explained in figure 3.18.

The existing user management system of WordPress can be extended. Card IDs from students and employees can be read with the card reader (section 3.3.17). The existing WordPress database table *wp_users* can be extended by an additional *card_id* entry. This entry stores and connects the data carrier information to a user and can be used for managing authorizations.

4 Concepts

Only authorized staff should have access to the central management system. The available WordPress *roles and capabilities* (section 2.9) can be used to ensure that specific content is only shown to specific users. New capabilities can be created and assigned to a role. Only users with this new capability can use the central authorization management system.

Pros	Cons
Already used WordPress system can be used as the central platform for managing all three locking systems	Primion CSV format and file transfer must be provided
MIFARE Classic and MIFARE DESFire EV1 cards as a uniform authentication medium	Hoffmann mobile application API must be provided
Gantner fully integrated	Hoffmann lock synchronization is still required
Primion partly integrated	
Hoffmann partly integrated	

Table 4.1: Pros and Cons of concept I

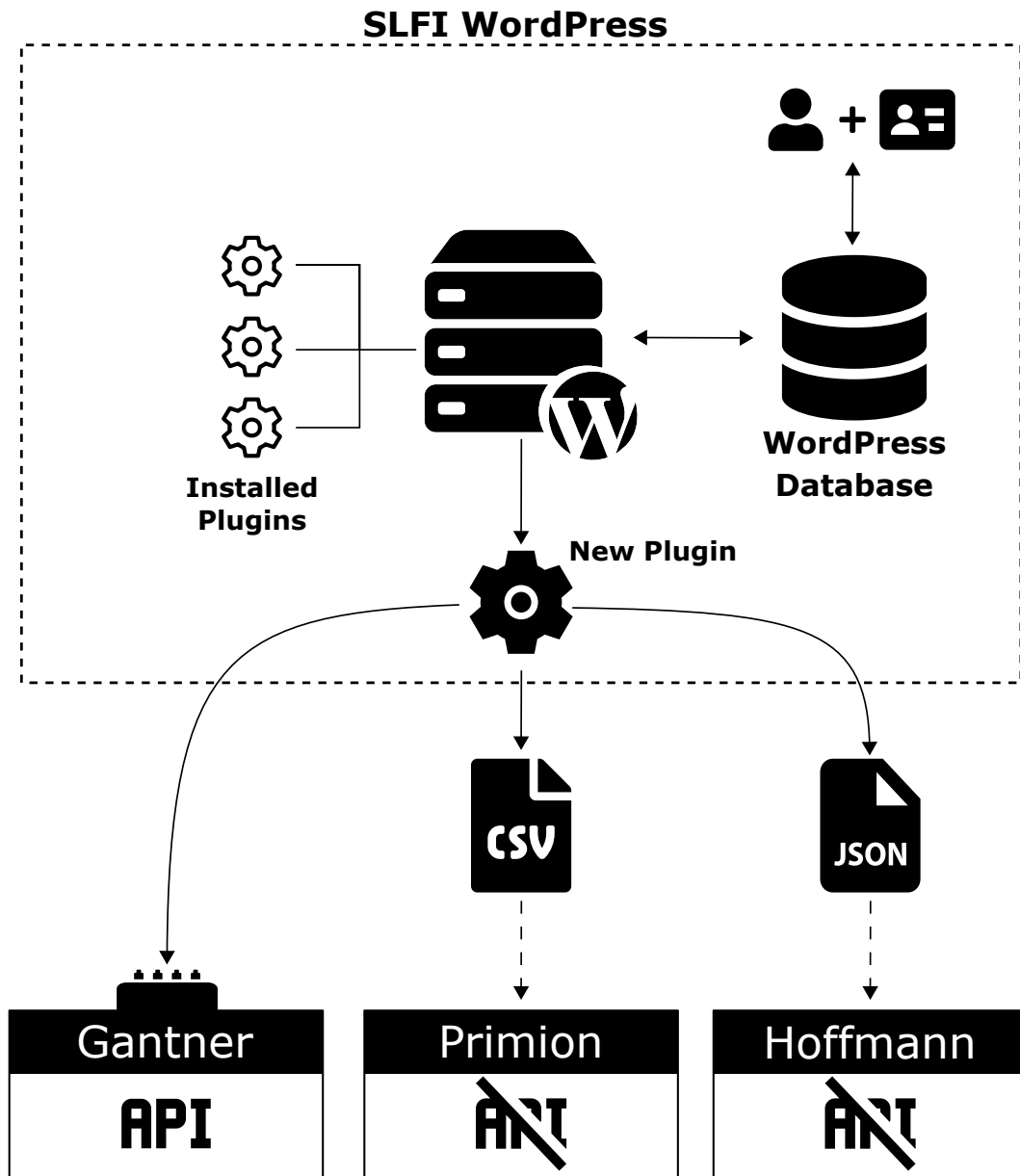


Figure 4.1: Concept I: A custom WordPress plugin. The Gantner system is accessed by REST endpoints. The plugin creates a CSV file that can be imported into the Primion system. The plugin provides a data endpoint for the mobile application from Hoffmann.

4.3.2 Concept II

Concept II (figure 4.2) is similar to Concept I. This concept is also creating a custom WordPress plugin to integrate the systems into WordPress. This concept shows how the three systems can be managed with a completed expansion phase II from Hoffmann. As explained in section 4.2 two major adaptations must be finished:

- Hoffmann locks must support the *Data on Card* technology
- Hoffmann system must be integrated into the Primion system

Hoffmann needs to finish expansion phase II as explained in section 4.2 and make sure that the locks can read authorizations from university ID cards (MIFARE Classic and MIFARE DESFire EV1). The Hoffmann system can then be integrated into the Primion system. This integration requires additional external work from the ZID. The ZID needs to create Primion *resources* for the Hoffmann system. These resources can then be managed the same way Primion resources are managed. Hoffmann locks could be administered as explained in figure 3.10. An API for the combined offline systems is not available. To integrate the offline systems only one CSV file needs to be generated. The CSV file contains information about the Primion and Hoffmann authorizations and can be imported by the ZID.

The extensions to the WordPress database for the Primion and Hoffmann authorizations are the same as described in concept I. The extensions to the existing user management system of WordPress are also the same as explained in concept I.

A major advantage of this concept is that the mobile application from Hoffmann no longer needs to be used. The *TAG mode* explained in section 3.3.13 is not required anymore.

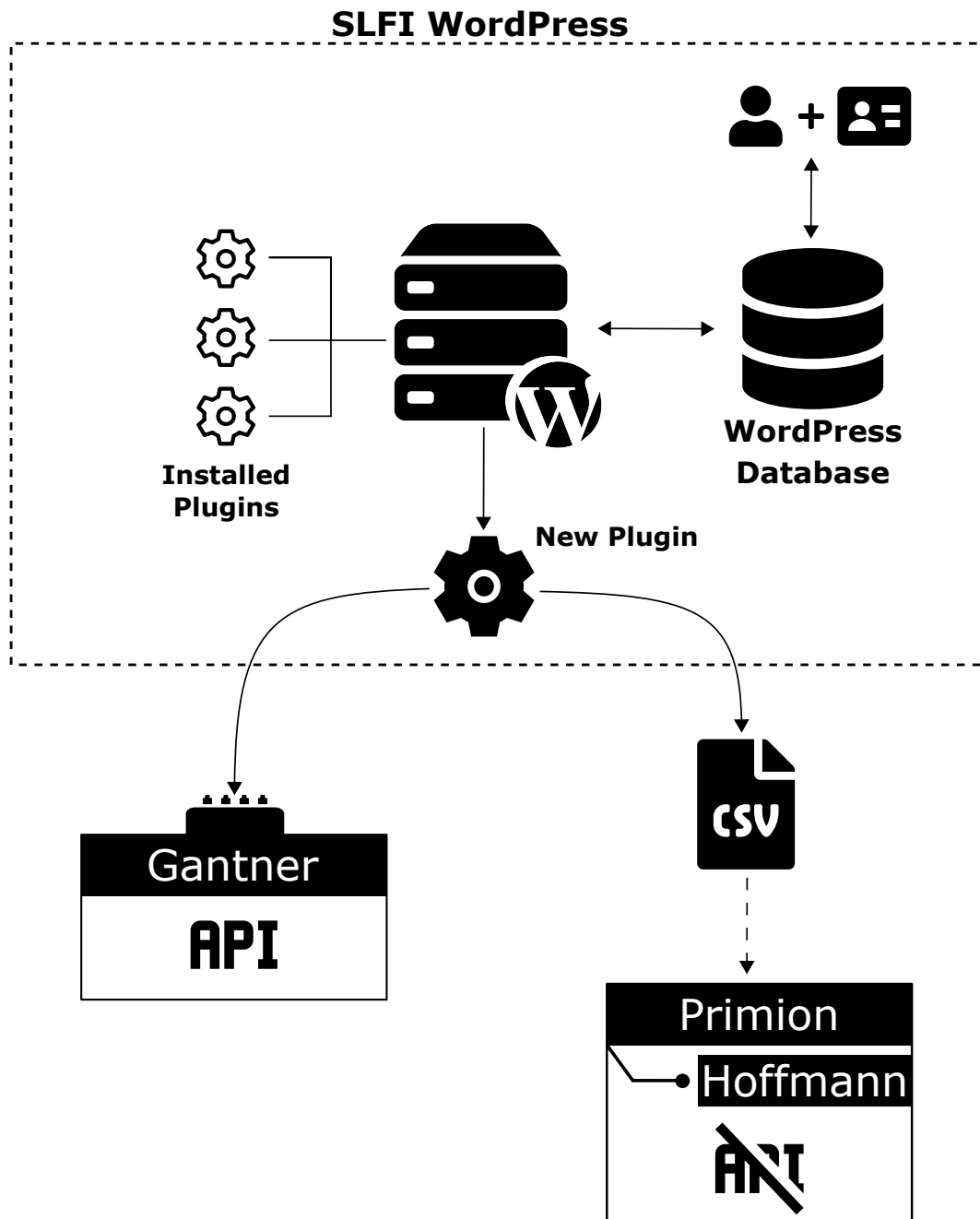


Figure 4.2: Concept II: A custom WordPress plugin. The Hoffmann system is integrated into the Primion system.

4 Concepts

Pros	Cons
Already used WordPress system can be used as the central platform for managing all three locking systems	Primion/Hoffmann CSV format and file transfer must be provided
MIFARE Classic and MIFARE DESFire EV1 cards as a uniform authentication medium	ZID needs to integrate the Hoffmann system into the Primion system
Gantner fully integrated	Hoffmann expansion phase II must be completed
Primion partly integrated	Hoffmann expansion phase II is cost-intensive and will not be finished soon
Hoffmann partly integrated	
Hoffmann mobile application is no longer needed	

Table 4.2: Pros and Cons of concept II

4.3.3 Concept III

Concept III (figure 4.3) shows how Gantner and Hoffmann can be integrated into the Primion system. The requirements for the Hoffmann system are already explained in Concept II.

The Gantner system is also supporting CSV import. Gantner could be administrated inside the TUGOnline as described in figure 3.10. The requirement is to develop a customized GUI for the Gantner system that can be used by authorized members of the TU Graz. Concept III shows the adaptation of the ZID access control management including the integration of the Gantner and Hoffmann system.

Pros	Cons
TUGOnline as the central managing platform for all three systems	The central platform is not available within the already used WordPress system
Gantner fully integrated	The ZID needs to develop the integration of the Gantner system into the TUGOnline platform
Primion fully integrated	No clarification if it is legally possible to outsource the Gantner system. The Gantner product is licensed to the IIM institute
Hoffmann fully integrated	Possibly high costs
Hoffmann mobile application is no longer needed	Possibly long development time
MIFARE Classic and MIFARE DESFire EV1 cards as a uniform authentication medium	ZID needs to integrate the Hoffmann system into the Primion system

Table 4.3: Pros and Cons of concept III

This concept fulfills the requirement of a central management platform. The TUGOnline is already used to manage Primion resources. The Hoffmann system can be integrated into Primion as described in concept II. The Gantner system would be maintained and managed by the ZID and can co-exist beside the Primion system.

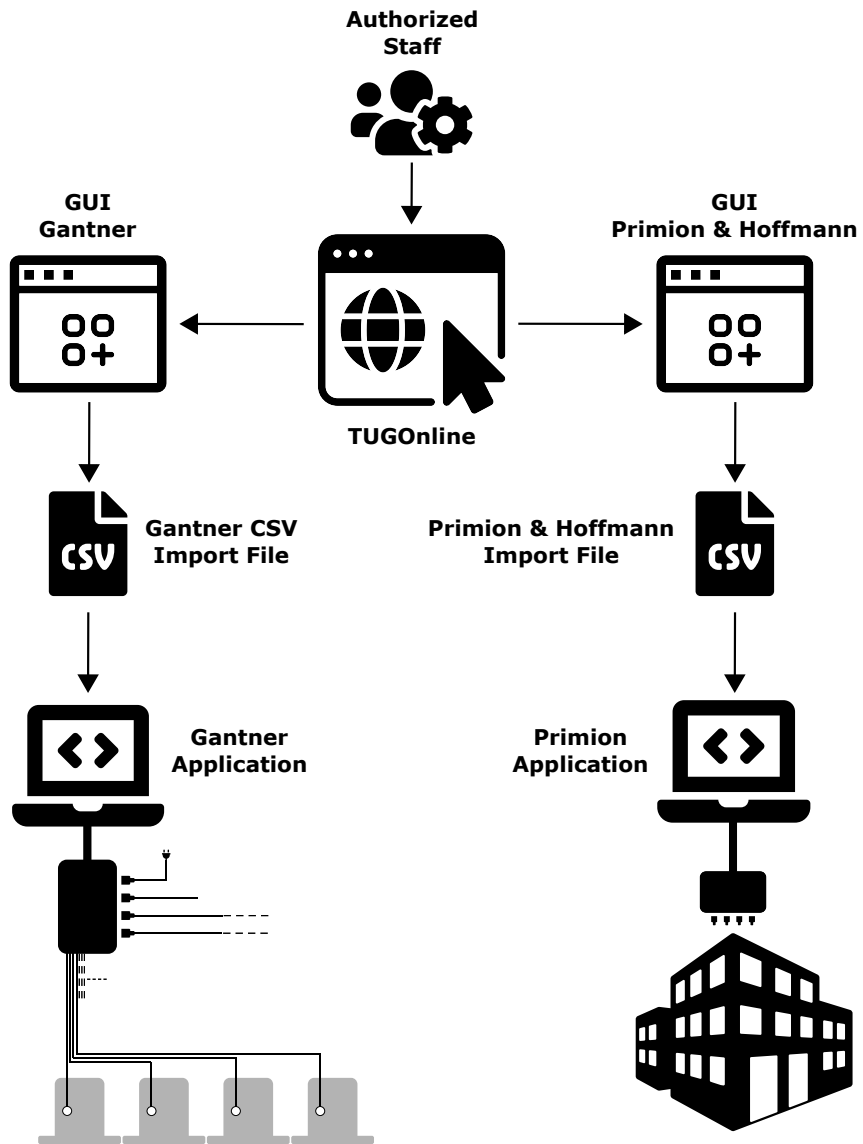


Figure 4.3: Concept III: Gantner and Hoffmann integration into Primion system

4.4 Concept Decision

All concepts fulfill the defined requirements. Table 4.4 shows the cons of each concept that was considered for the concept decision.

Concept III shows a solution that could also benefit premises outside the makerspace. External institutions of the TU Graz can upgrade their storage options with lockers from Gantner and Hoffmann and manage them within TUGOnline. This concept needs further feasibility checks. The Gantner system is owned by the IIM institute. The possibility of outsourcing the Gantner solution to the ZID must be decided by the management. The Hoffmann locks must be upgraded to support the *Data on Card* technology. Only then the locks can technically be integrated into the Primion system by the ZID. The main reason to not choose this concept is the lack of information. It is not clarified if the ZID would allow this approach. It is unsure how much time Hoffmann will need to upgrade their locks to support the *Data on Card* technology. There is no information about outsourcing the in-house Gantner solution to the ZID. It is unclear what additional costs would be incurred and how long the development time would be.

Concept I and II are very similar. Both concepts build on extending the already used WordPress platform. WordPress is the central platform for all systems. The key difference is that concept II requires the Hoffmann system to be integrated into Primion. To manage the Primion and Hoffmann locks only one CSV file is generated by the plugin. This file can then be imported by the ZID. Concept I is providing a CSV file for the Primion system and a REST endpoint for the Hoffmann system. The REST endpoint contains all necessary information in JSON format. This endpoint can be used to update the authorizations stored in the database of the mobile application from Hoffmann.

Upgrading the Hoffmann locks to support the *Data on Card* technology is expensive and definitely will take more time to be finally completed. Concept I is therefore the better approach. The mobile app of Hoffmann is currently not able to use an external interface to load information and update their locks. Providing such an interface is far less costly than the expansion phase II. Furthermore, providing the information in a CSV format as described in concept II requires relatively little effort.

Concept I	Concept II	Concept III
Primion CSV format and file transfer must be provided	Primion and Hoffmann CSV format and file transfer must be provided	The central platform is not available within the already used WordPress system
Hoffmann mobile application API must be provided	ZID needs to integrate the Hoffmann system into the Primion system	The ZID needs to develop the integration of the Gantner system into the TUGOnline platform
Hoffmann lock synchronization is still required	Hoffmann expansion phase II must be completed	ZID needs to integrate the Hoffmann system into the Primion system
	Hoffmann expansion phase II is cost-intensive and will not be finished soon	No clarification if it is legally possible to outsource the Gantner system. The Gantner product is licensed to the IIM institute
		Possible long development time and additional high costs

Table 4.4: The cons of the developed concepts summarized.

5 Results

This chapter describes the implementation of concept I. Figure 5.1 highlights the core components that are used in the concept. Section 5.1 explains the architecture of the concept. The structure of the custom WordPress plugin is also explained in this section.

Section 5.1.1 explains the relevant parts of WordPress that were used in the concept development. Section 5.1.2 describes the WordPress database structure and the tables that were used. The installation process for a custom WordPress plugin will be explained in section 5.1.3. Section 5.1.4 explains the Python flask framework that was used to develop the middleware. The relation between the Gantner GAT Relaxx system and the Gantner REST API endpoints will be described in section 5.1.5. The Python CSV module is explained in section 5.1.6. The JSON data format used for data exchange between the systems will be explained in section 5.1.7.

Section 5.2 describes the setup of the local development environment. All tools and technologies that were used to develop the concept will be explained in this section. The development of the middleware is described in section 5.3. The user interaction with the custom plugin will be explained in section 5.4.

5.1 Concept Architecture

The concept architecture can be divided into three layers:

- Frontend: Responsible for how the user interacts with the user interface.
- Backend: Deals with business logic, data storage, and processing.
- Middleware: Communication layer between the front- and backend.

The frontend layer is responsible for the data representation. User requests are handled by the server. The server response includes all information that is displayed to the user. Information about the access control systems is provided by the middleware. The server requests this information from the middleware by using the provided REST API endpoints. With the response of the middleware, the server dynamically builds the content that is presented to the user.

The middleware is the bridge between the front- and backend. Server requests are processed by this layer. Requests can include any of the CRUD operations that are explained in section 2.3. All the required information is gathered here to execute the operations. This layer is also communicating with the backend systems. Authentication to these systems is also handled by this layer.

The backend layer consists of the access control systems: Gantner, Primion and Hoffmann. The backend layer is responsible for executing the requests from the middleware. As explained in section 4.3.1, REST API endpoints are used for the Gantner system. The remaining systems provide no APIs. The middleware creates a CSV file for the Primion system. The Hoffmann system can request the updated authorizations from the middleware. The middleware provides a REST API endpoint to get this data.

Figure 5.1 shows the core components that are used in the concept:

- WordPress system
- WordPress database
- Custom WordPress plugin
- Python flask server
- Gantner swagger UI
- Primion CSV file
- Hoffmann JSON data

Plugin Architecture

Figure 5.1 shows that the plugin is divided into two parts. The first part is the custom WordPress plugin that is explained in section 5.1.3. This plugin is an extension to the existing WordPress system and operates on the server. The second part is the Python web server. The server is a standalone application that handles the communication between WordPress and the access control systems.

It is also possible to access the different locking systems directly from WordPress. Addressing the individual systems directly from WordPress can lead to problems for future extensions. If WordPress is replaced by another management system in the future, not only the graphical user interface but also the complete logic for accessing the individual locking systems must be reprogrammed. To prevent this situation, the GUI and the programming logic responsible for addressing the individual locking systems are separated from each other. Only the GUI needs to be re-implemented. The GUI and the programming logic for accessing different systems can therefore be developed separately. The chosen software architecture should enable future changes and extensions to be implemented efficiently.

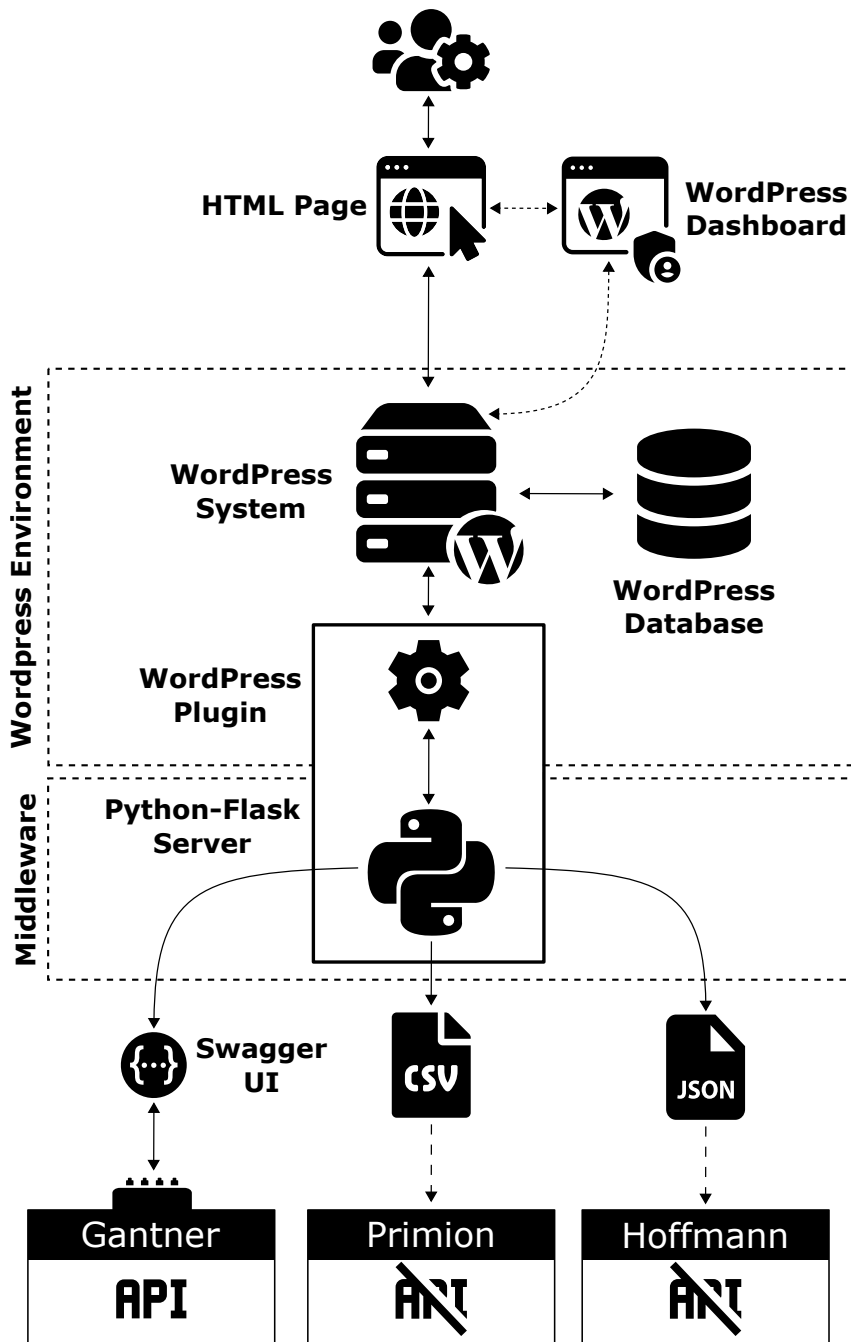


Figure 5.1: Overview of the core components that are used in concept I.

5.1.1 WordPress System

Section 2.9.1 gives a brief overview of the WordPress system. This section explains the important parts that are used in the concept in more detail. The WordPress system consists of a lot of different files that ensure that the system is running correctly. Modification of these files is dangerous and can lead to a software fault or cause the whole system to crash. The following folders contain the core files of WordPress: [Sto21]

- wp-admin
- wp-content
- wp-includes

wp-admin

This folder contains all the files that are required for the WordPress dashboard to function correctly. The dashboard (figure 5.2) allows administrators to create new content, perform updates, install new themes, and much more. [Sto21]

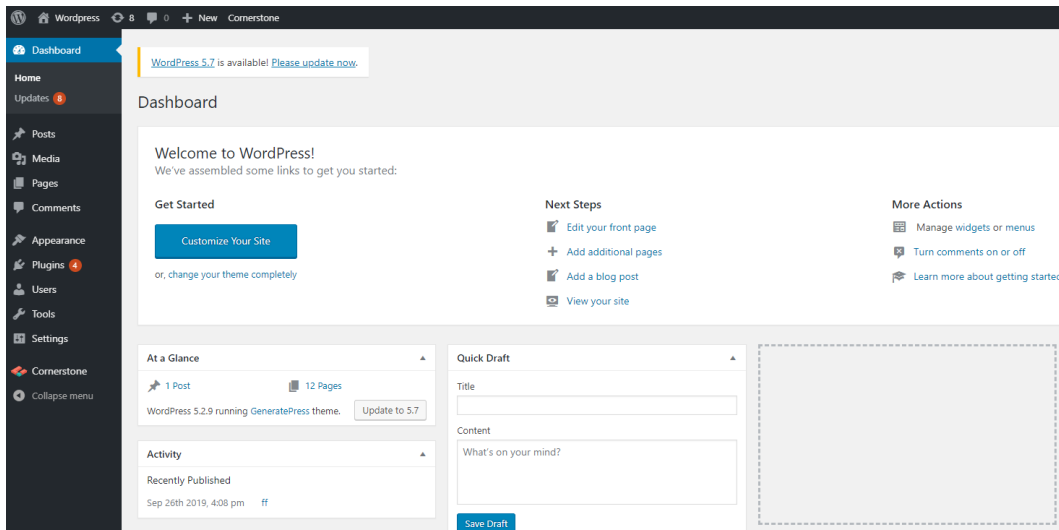


Figure 5.2: The WordPress dashboard is the control panel for the administrators.

The three most important options that are used for developing the concept are:

- Pages
- Plugins
- Users

In WordPress, pages, and blogs are used to provide content. Creating blog entries are time-dependent and appear in reverse chronological order on a WordPress home page. Pages are used to create timeless information that is always relevant (e.g.: *About* page). WordPress templates can be used to create a new page without the need for coding. [Wor21b]

The dashboard option *Plugins* shows all the plugins that are stored in the `wp-content\plugins` folder. The plugins can be activated, deactivated, updated and deleted by the administrator.

The *Users* option allows administrators to manage the user accounts. WordPress provides their custom role system. Every user in WordPress is assigned to at least one role. Every role has a set of capabilities that specify what a user is allowed to do on the system. The default roles are: Subscriber, Contributor, Author, Editor and Administrator. WordPress allows the creation of new roles and capabilities that can be assigned to users.

wp-content

This folder contains all the plugins, themes, upgrades and uploads that have been used in the WordPress system. The *Plugins* option of the dashboard shows all plugins that are located in this folder. The custom plugin that was developed (figure 5.1) is also stored in this directory.

wp-includes

Besides the `wp-admin` folder, this directory contains all the remaining files that are important for the system to run properly. The most important file

in this directory is the *functions.php* file. It is considered the main WordPress API. [Sto21]

5.1.2 WordPress Database

A new WordPress installation requires a database that is holding all the relevant information. WordPress only supports the database types: MySQL version 5 or greater, or any version of MariaDB. When creating a custom plugin it might become relevant to interact with this database. WordPress is therefore providing the *wpdb* class to make this interaction easier. This class should be used to interact with the database without the need of raw SQL statements.

Figure 5.3 illustrates the WordPress database and the relations between the tables. The *wp_users* and *wp_usermeta* are the two important tables for the concept development. The *wp_users* table contains relevant information about the users (e.g.: user login name). Additional information about a user is stored in the *wp_usermeta* table (e.g.: roles and capabilities). [Wor21a], [Wor21h]

5 Results

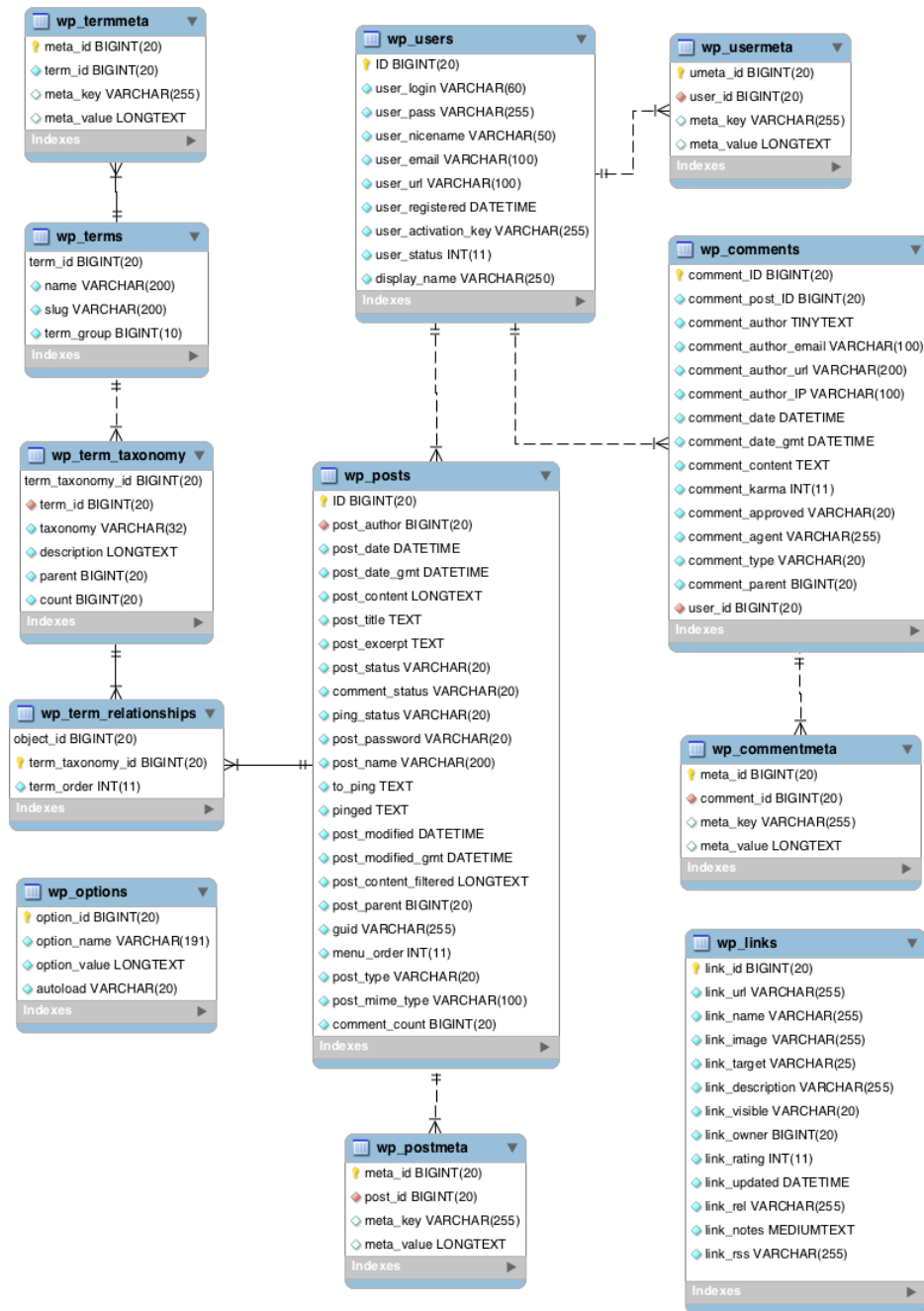


Figure 5.3: The WordPress database structure and the relations between the tables. This figure is fully adopted from [Wor21a]

5.1.3 Custom WordPress plugin

To create a new plugin it is necessary to create a new folder inside `wp-content\plugins` subdirectory. This new folder contains all plugin related files. The folder must have a unique name to prevent conflicts with the other plugins. It is a common practice to put a `.php` file inside the plugin directory that is also named after the plugin. This file is the main function of the plugin and should contain a plugin header comment. The comment is a PHP block that contains metadata about the plugin (e.g.: name, author, license, and more). The plugin will be listed in the *Plugins* option in the WordPress dashboard. Figure 5.4 summarizes this procedure. [Wor21c]

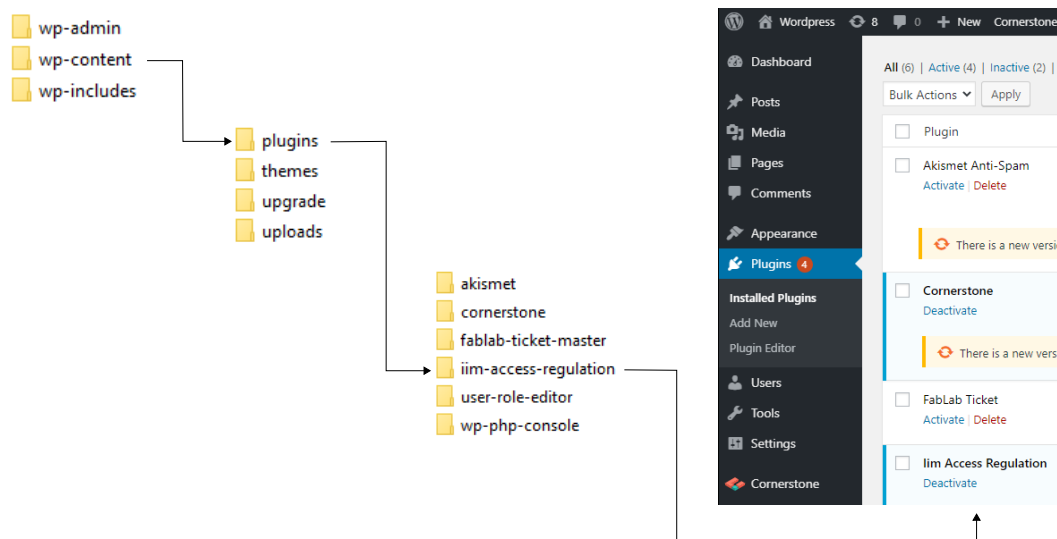


Figure 5.4: Plugins folder structure in WordPress.

WordPress Hooks

Hooks allow to tap into WordPress at specific points and change the default functionality. WordPress has two types of hooks: *Actions* and *Filters*. *Actions* are used to change or add new functionality. *Filters* are used to modify content before it is displayed to the user. There are three basic hooks for

plugin development: `register_activation_hook()`, `register_deactivation_hook()` and `register_uninstall_hook()`. Activation and deactivation of a plugin trigger the `register_activation_hook()` and `register_deactivation_hook()` respectively. The `register_uninstall_hook()` allows specifying clean-up functions that are executed when a plugin gets deleted. [Wor21c]

5.1.4 Python flask

Flask is a Python web framework that is used for developing web applications. Web frameworks make development easier and faster by providing common patterns (e.g.: HTTP operations) in the web development process. Only a few lines of code are needed to set up a running web application. [Mak21a], [Mak21b]

```
from flask import Flask
simple_web_application = Flask(__name__)

@simple_web_application.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    simple_web_application.run()
```

Listing 5.1: Set-up of a simple web application with Python flask. Own representation based on [Mak21a]

The few lines of code start the web application on the local port 5000 and display the text *Hello World!*. [Mak21a]

Flask-RESTPlus

Flask-RESTPlus is an extension of the Flask web framework. This extension is adding support for building RESTful APIs efficiently. It provides a collection of tools and decorators to develop and describe API endpoints. It also enables proper documentation of the APIs by using *swagger 2.8.1*. [Hau21]

Requests Module

The *requests* module is an HTTP python library. It provides a simple to use API that abstracts the complexity of creating HTTP requests. [Doc21]

5.1.5 Gantner GAT Relaxx

A brief overview of the Gantner system and how it is used in the SLFI is described in section 3.3.8. A complete GAT Relaxx system overview is shown in figure 3.12. The GAT Relaxx server is installed as a Windows service that runs in the background. The server has no user interface and is responsible for the communication with the locker system. The GAT Relaxx client is a desktop application that is used to configure and manage all the lockers in the system. Figure 5.6 shows the main dashboard of the GAT Relaxx desktop client.

- Area 1: Modify or add new areas and/or locker groups. Locker groups can be assigned to areas. Lockers are assigned to locker groups.
- Area 2: Visual representation of all areas and locker groups.
- Area 3: Visual representation of all lockers.
- Area 4: Switch between different views: Lockers, Authorizations and more.
- Area 5: Additional information about a locker is shown here. A locker must be selected in Area 3.

The GAT Relaxx client provides a lot more functions such as the hardware configuration of the lockers to the controllers, database configuration, locker history protocol, and more. The Gantner REST API is the most important extension of Gantner that is used for concept development.

GAT Relaxx API

Gantner summarizes the most common locker management tasks and makes them available by using the GAT Relaxx API. These REST API endpoints can be used by third-party applications. The endpoints are specified and visualized with *Swagger* (figure 5.5).

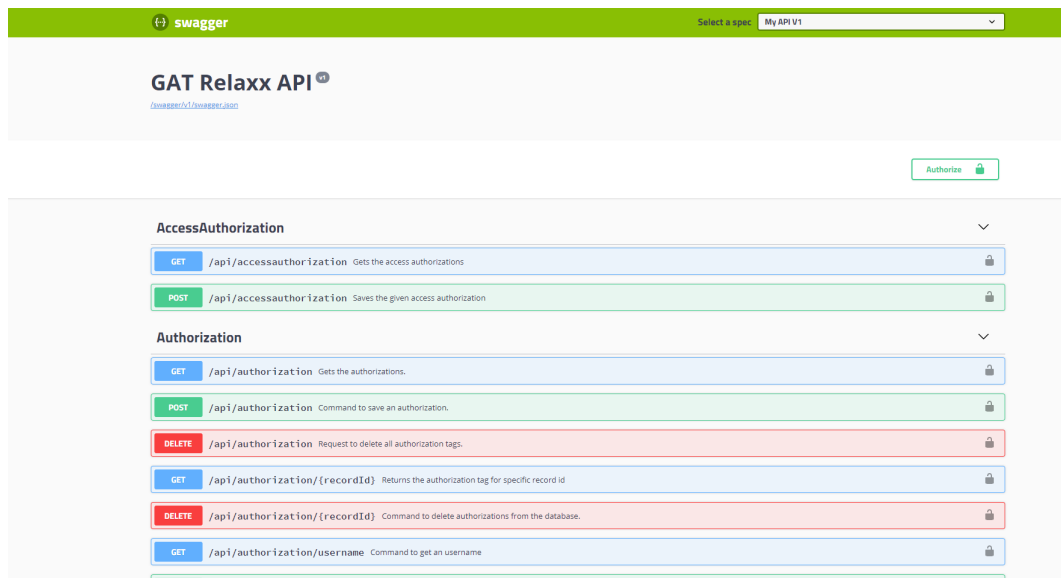


Figure 5.5: Gantner REST API endpoints.

Authentication

The REST API endpoints can only be used after successful authentication. OpenAPI specifies different authentication schemes to protect the API

endpoints. Gantner is using the *bearer* HTTP authentication scheme. This scheme can also be called a token authentication scheme and can be described as "grant access to the bearer of this token". The token is a cryptic string of characters. The string is generated by the server in response to a successful client login request. [Sof21]

5 Results

The screenshot displays the GAT Relaxx 4.0.1 desktop client interface. The main window shows a reservation calendar with a grid of dates and reservation counts. The interface includes a top navigation bar with 'Start', 'Hilfe', and 'Ansicht' options. A sidebar on the left contains a search bar and a list of categories: 'Organisation', 'Labor für Innovation (LFI)', 'Garderobe', 'Garderobe Freie Locker', 'Garderobe Private Locker', 'Garderobe Spezial', 'Projekte', 'Private Private Locker', 'Rezeption', and 'Rezeption Private Locker'. A red box labeled '1' highlights the top navigation bar. A red box labeled '2' highlights the sidebar menu. The main calendar area is a grid with dates 1 through 11 in the first row, 2 through 10 in the second, 3 through 9 in the third, 4 through 12 in the fourth, 5 through 11 in the fifth, and 6 in the sixth. A red box labeled '3' highlights the entire calendar grid. To the right of the calendar is a 'Filter' section with dropdown menus for 'Suche', 'Schrankmodus', 'Status', 'Wartung', and 'Reserviert'. Below the filter is a 'Statistiken' section with a list of items and their counts. A red box labeled '4' highlights the bottom navigation bar, which includes 'Schränke', 'Berechtigungen', 'Reservierungen', 'Zeitplanung', 'Protokoll', 'Berichte', and 'Extras'. A red box labeled '5' highlights the 'Statistiken' section. The bottom status bar shows '50 Elemente' and '50/100'.

Es gibt nicht genügend Lizenzen (0) für alle in Relax verwalteten Schlüssel (50). Alle Schränke sind nur im Notfallmodus.

Statistiken

- Gedrehte Schränke 0/50
- Schranke in Verwendung 0/50
- Alarmierte Schränke 0/50
- Unbekannte Schränke 30/50
- Nicht gefundene Schränke 0/50
- Deaktivierte Schränke 0/50
- Reservierte Schränke 0/50
- Schranke in Wartung 0/50

Schranke Berechtigungen Reservierungen Zeitplanung Protokoll Berichte Extras

Figure 5.6: The main window of the GAT Relaxx desktop client.

5.1.6 Python CSV Module

The Primion system provides no APIs as explained in section 3.3.3. Authorizations are updated by importing data in CSV (Comma Separated Values) format. CSV is a very common import and export format for spreadsheets and databases and is standardized in RFC 4180. The first line can be used as an optional header line. The header line describes the fields of the records in the file. It should also have the same amount of fields as the records. Records are located on separate lines, delimited by a line break.

The Python *csv module* provides classes to read and write data in CSV format. It also allows developers to create their own CSV format. [Lib21], [Shao5]

```
FirstName;LastName;CardUID;PrimionResource;ValidFrom;ValitUntil;
Michael;Rossmann;2917354981;Main Building Entrance;12.01.2021;15.05.2021;
Michael;Rossmann;2917354981;SLFI Entrance;12.01.2021;15.05.2021;
Michael;Rossmann;2917354981;Door FabLab I;12.01.2021;15.05.2021;
Max;Mustermann;9876543210;Main Building Entrance;01.01.2021;10.01.2021;
Max;Mustermann;9876543210;SLFI Entrance;01.01.2021;10.01.2021;
Max;Mustermann;9876543210;Door Tesla Meeting Room;01.01.2021;10.01.2021
```

Figure 5.7: An example Primion CSV file.

5.1.7 Python JSON Module

JSON (JavaScript Object Notation) is a text-based format to serialize structured data and is specified in the RFC 8259. JSON is one of the most used data formats to transmit and receive data between a server and a web application. JSON supports primitive types such as *strings* and *numbers* as well as nested *arrays* and *objects*. The syntax rules of JSON can be summarized as:

- Data is represented as key/value pairs and is separated by commas
- Curly brackets hold objects, square brackets hold arrays

Python supports the JSON data format natively. A built-in module called *json* allows to encode and decode JSON data. [Bra17], [ORG21]

```
1 {
2   "Authorizations": [
3     {
4       "FirstName": "Michael",
5       "LastName": "Rossmann",
6       "ValidFrom": "2021-01-01",
7       "ValidUntil": "2021-01-30"
8     },
9     {
10      "FirstName": "Max",
11      "LastName": "Mustermann",
12      "ValidFrom": "2021-01-01",
13      "ValidUntil": "2021-01-15"
14    }
15  ],
16  "Lock": "Hoffmann Locker 3D Printer",
17  "Lock Number": "1",
18  "Lock ID": "983c5b8f-f760-4968-8f9c-4030040ec85f"
19 }
```

Listing 5.2: Example JSON file containing authorizations for a Hoffmann lock.

5.2 Development Environment Setup

For the realization of the concept many different areas needed to be covered as described in the concept architecture 5.1. Setting up a local environment for the plugin development has many benefits. The most important one is testing. Local development and testing ensure that nothing breaks on the live version of WordPress.

This section explains the technologies and tools (table 5.1) that were used to develop the concept.

XAMPP

WordPress needs three components to run: A server-side programming language (e.g.: PHP), a web server (e.g.: Apache) and a database management system (e.g.: MySQL). One option is to install all these components separately. Another option is to install XAMPP. It provides the complete environment that WordPress needs to function and is available on their official website (<https://www.apachefriends.org/index.html>). [Ste16]

WordPress

Section 2.9 gives a general overview of WordPress. WordPress version 5.2.9 was used for the concept development. A list of all releases can be found here: <https://wordpress.org/download/releases/>. The WordPress theme that was used for development and testing is *GeneratePress*. It is a lightweight theme with a focus on performance. The theme can be downloaded here: <https://de.wordpress.org/themes/generatepress/>. [Org21b], [Gen21]

Python

Python is an object-oriented, high-level programming language. Its simple and easy-to-learn code syntax does not only highlight the readability but does also reduce the maintenance of Python applications. A wide range of packages and modules in the most diverse areas are supported and enhance this programming language. This makes Python very attractive for rapid prototyping, as well as for use as a scripting language to connect existing applications or components. [Org21a]

PHP

PHP is a server-side programming and scripting language. It can be embedded into HTML code to build dynamic and interactive websites. HTML output is generated on the server and sent back as a response to a client

request. WordPress is explicitly using PHP as the server-side programming language. [Ste16], [al21]

JavaScript and AJAX

JavaScript is a client-side programming and scripting language. It is a very important extension to HTML and provides the client-side logic of a web application. This logic allows building more dynamic websites. Code can be included in HTML that interacts with the user. For example, a script that is validating user input or creates HTML content dynamically.

AJAX (Asynchronous JavaScript and XML) is a technology to improve the responsiveness of interactive web applications. Content of web pages can be updated asynchronously. Data is exchanged with the server in the background. This allows updating only specific parts of a web page without the need to reload the whole page. Users can interact in almost real-time with web applications. [Inf21], [Ste16]

PyCharm

PyCharm is a development tool for building Python applications provided by JetBrains. It was used to develop the middleware application explained in the concept architecture 5.1. PyCharm can be downloaded here: <https://www.jetbrains.com/de-de/pycharm/>. [Jet21]

PhpStorm

This development tool is also provided by JetBrains. It was used for developing the front-end of the concept including the custom WordPress plugin described in the concept architecture 5.1. PhpStorm can be downloaded here: <https://www.jetbrains.com/de-de/phpstorm/>. [Jet21]

GAT Relaxx

The following components were installed for the local development:

- SQL Database
- SQL Server Management Studio
- Relaxx Server
- Relaxx Client
- Relaxx REST API

GAT Relaxx requires a database. If there is no pre-installed database, Microsoft SQL Server Express can be installed with the installation wizard of GAT Relaxx. SQL Server Management Studio was installed for managing the database.

Relaxx Server is the main component for the configuration and management of the locker system. It runs as a background Windows service.

Relaxx Client is the graphical user interface 5.6 that is used for configuring the locker system.

To be able to use the Gantner REST API the following components must be installed:

- .NET Core Runtime & Hosting Bundle
- Microsoft Internet Information Service (IIS)

The hosting bundle is needed to run .NET Core applications. The IIS is an optional feature of the Windows operating system. It is a web server that allows hosting ASP.NET web applications. [Mic21b], [Mic21c], [Mic21d], [Mic21a]

5.2.1 XAMPP WordPress Setup

Setting up a working local WordPress system can be done with XAMPP in a few steps. XAMPP needs to be installed with the components MySQL and PHPMyAdmin. Figure 5.8 shows the XAMPP control panel with the running components. The downloaded WordPress files need to be placed into the *htdocs* folder that is located in the XAMPP installation directory. If

5 Results

Technology/Tool Name	Version
XAMPP	3.2.4
WordPress	5.2.9
Python	3.7.4
PHP	7.3.8
PyCharm (Community Edition)	2019.1.3
PhpStorm	2019.2.1
GAT Relaxx	4.0.1

Table 5.1: All tools and their versions that were used for developing the concept.

there is no existing database a new one can be created with the admin panel of PHPMyAdmin (figure 5.9).

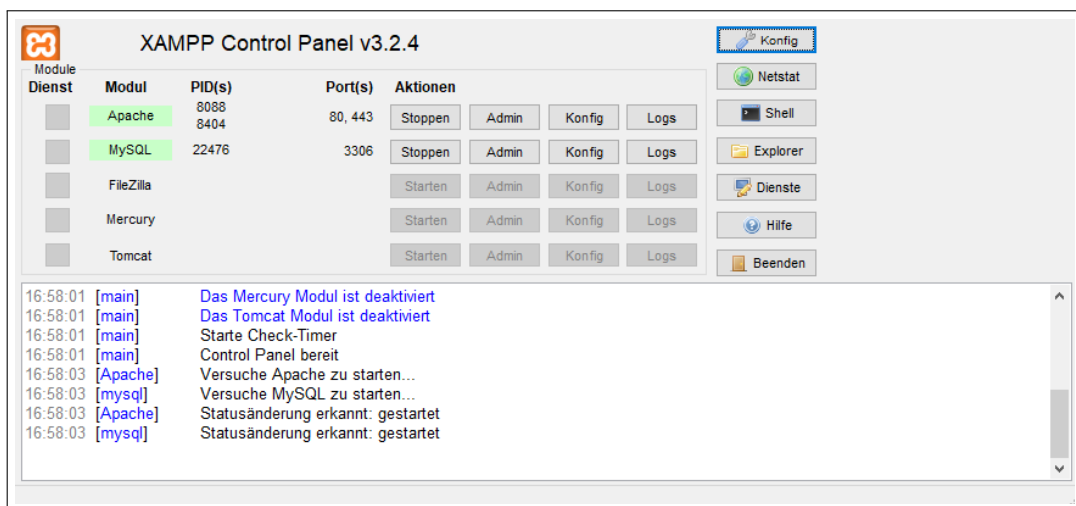


Figure 5.8: XAMPP control panel.

WordPress is accessible on the local machine at *http://localhost* (figure 5.10). After the completion of the installation the XAMPP control panel is used to turn the local WordPress instance off and on.

5 Results

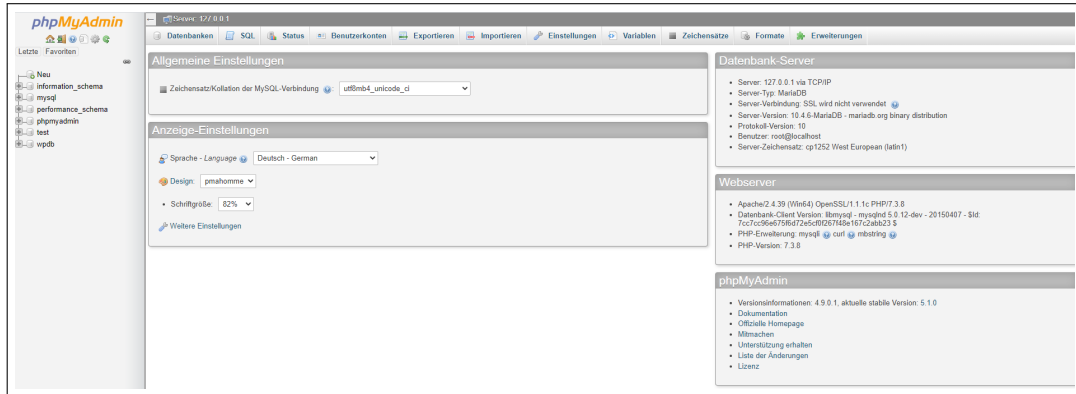


Figure 5.9: PHPMYAdmin dashboard used to create new databases.

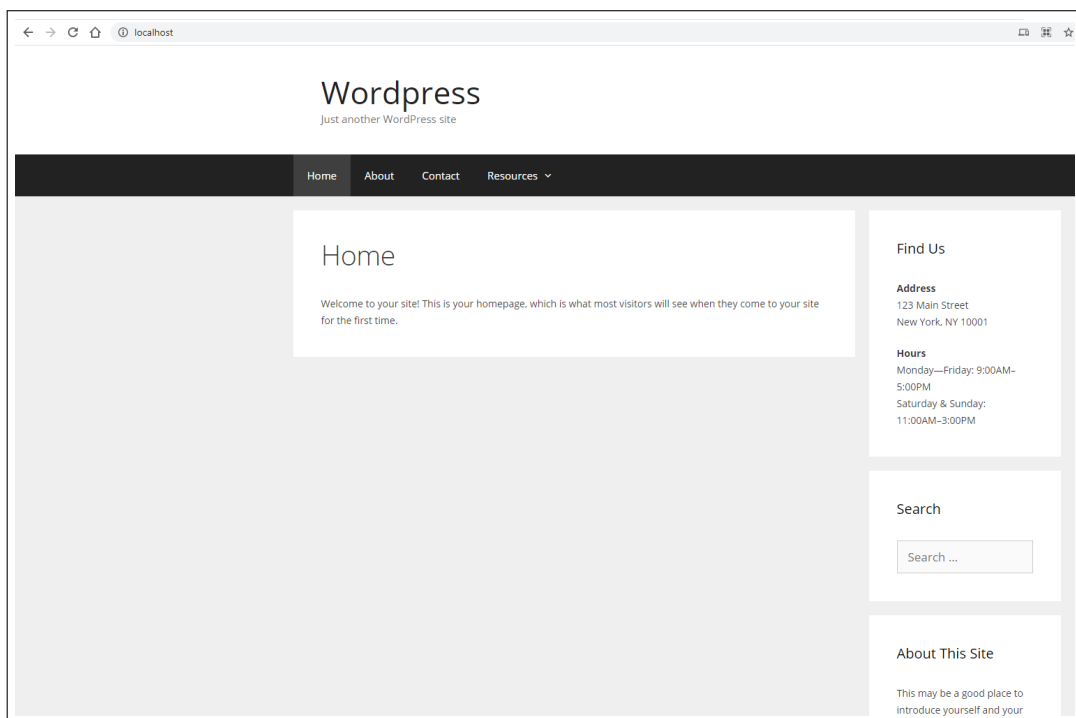


Figure 5.10: Local WordPress installation running on *http://localhost*.

5.3 Setting up the Python web application

This section demonstrates the development of the middleware. It is a Python web application that works between WordPress and the access control systems. The web application is built with the Python flask web framework as described in section 5.1.4. The flask app was extended to provide REST API endpoints for the WordPress and Hoffmann system. The endpoints are created with the Python *Flask-RESTPlus* module 5.1.4. To communicate with the REST endpoints from WordPress and Gantner the *requests* module 5.1.4 is used.

5.3.1 WordPress - Middleware - Gantner Interaction

Figure 5.11 shows a simplified sequence diagram of the interaction between WordPress, middleware, and the Gantner system. The middleware receives requests from the WordPress system. The requests are checked first. The check makes sure that:

- The target system can be reached
- All data is available for further execution

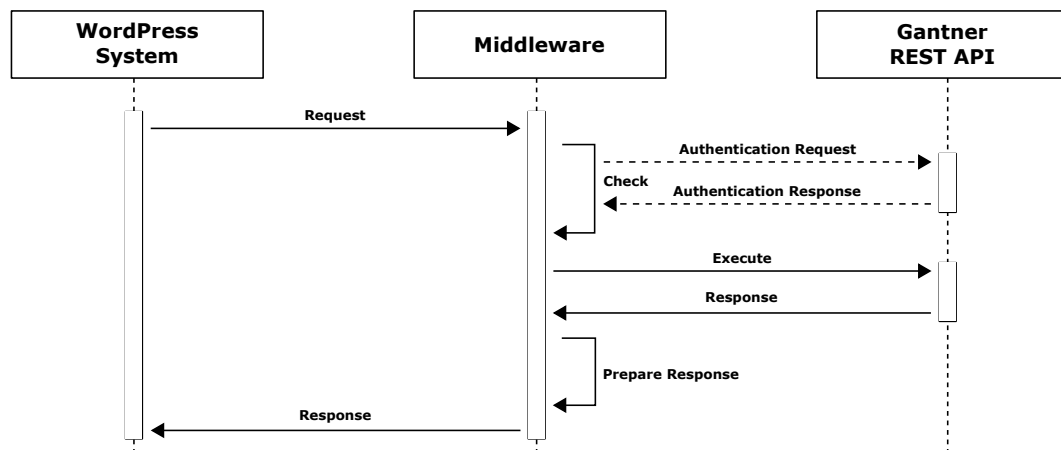


Figure 5.11: Simplified sequence diagram of the interaction between WordPress, middleware and the access control systems.

Gantner is the only system that provides an API. The REST endpoints from Gantner can be used by the middleware after successful authentication 5.1.5. The data provided by WordPress must be complete for execution. An error message is sent back to WordPress otherwise. After a successful check, the request will be executed. The following commands can be executed:

- Authentication
- Gantner State
- Update Gantner Authorizations
- Update Gantner Authorization

Authentication

Resources can only be used after a successful authorization. Gantner is using the *bearer* HTTP authentication scheme. The Gantner administrator credentials *username* and *password* are sent to the `/api/user/login_api` endpoint with an HTTP *post* request. A *bearer* token is created by the Gantner system if the credentials are correct. An error message is sent back otherwise. The token is a cryptic string that is sent back to the requester. The token needs to be included in all further requests.

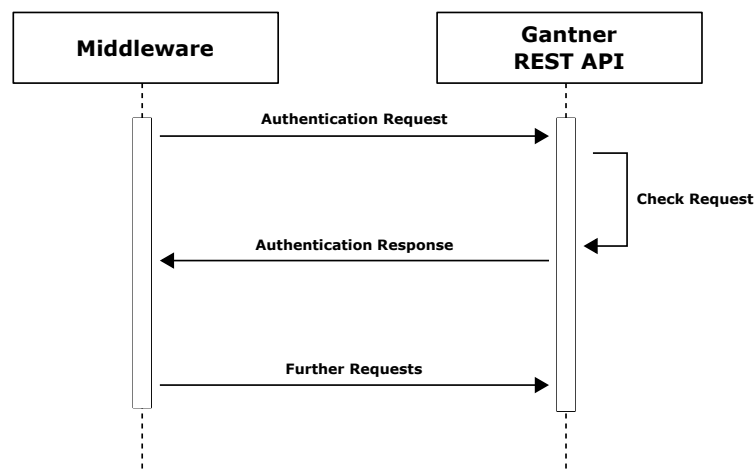


Figure 5.12: Gantner authentication sequence.

Gantner State

The WordPress GUI is visualizing the complete Gantner state. This is achieved by getting all the necessary information from the REST endpoints. The information needed to visualize the state includes:

- All locker areas, sub-areas, and locker groups
- All lockers
- All authorizations

Gantner provides no endpoint that represents the complete state. This endpoint is created by the middleware. Table 5.2 shows that several different Gantner endpoints are used to create the state.

REST endpoint	Returns
<i>/api/locker_area</i>	Areas, sub-areas and locker groups
<i>/api/locker</i>	List of lockers
<i>/api/locker_authorization/get_all</i>	Locker authorizations excluding user information
<i>/api/authorization</i>	Authorization including user information

Table 5.2: The REST endpoints that are used to create the Gantner state.

The state is returned to WordPress in JSON format as described in section 5.1.7. The state is represented in a tree structure. The locker groups represent the root of this structure. Each locker group contains a list of lockers and each locker contains a list of authorizations. Figure 5.3 shows the structure in JSON format.

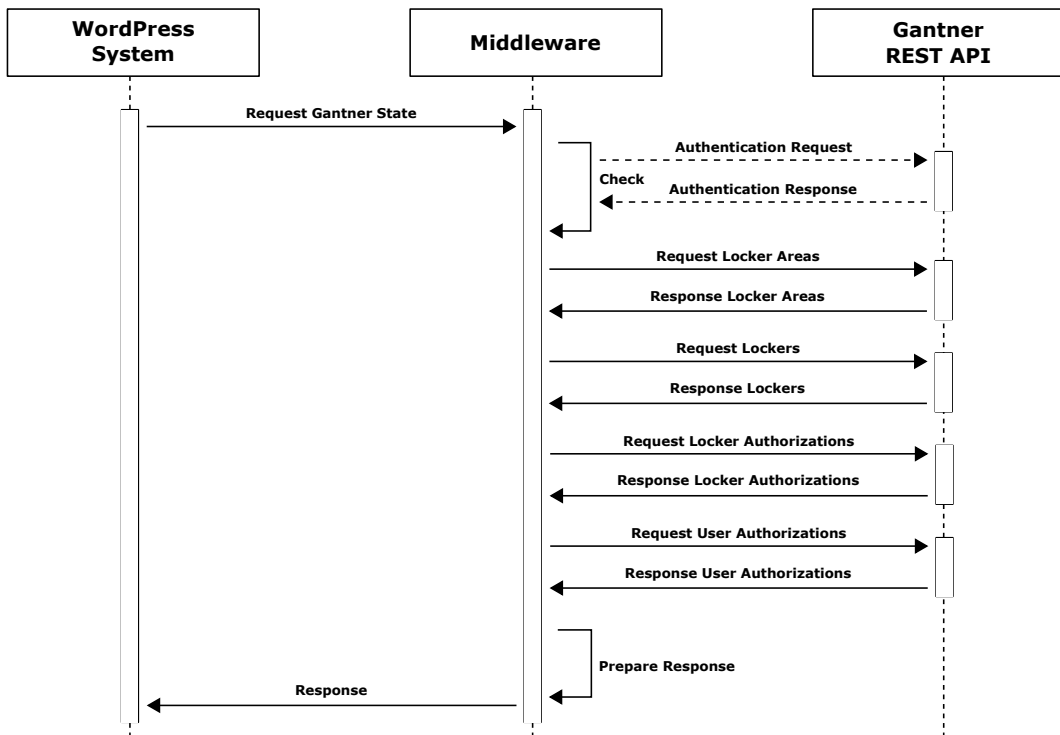


Figure 5.13: Several different REST endpoints are used to create the state that is sent back to WordPress.

```
1 {
2   "lockerGroups": [
3     {
4       "name": "",
5       "id": "",
6       "lockers": [
7         {
8           "id": "",
9           "lockerGroupId": "",
10          "lockerGroupName": "",
11          "lockerNumber": "",
12          "authorizations": [
13            {
14              "firstName": "",
15              "lastName": "",
16              "cardUID": "",
17              "validFrom": "",
18              "validUntil": ""
19            }
20          ]
21        }
22      ]
23    }
24  ]
25 }
```

Listing 5.3: JSON structure containing all fields for WordPress to represent the Gantner state

Update Gantner Authorizations

This endpoint can be used by WordPress to create new or update existing user authorizations. The WordPress request includes all information needed for the update in the Gantner system. The JSON format of the request is shown in figure 5.4.

```
1 {
2   "persons:" [
3     {
4       "first_name": "",
5       "last_name": "",
6       "card_id": ""
7     }
8   ],
9   "locker_area": "",
10  "locker_number": "",
11  "start_date": "",
12  "end_date": ""
13 }
```

Listing 5.4: JSON structure containing all fields for WordPress to update Gantner authorizations

The Gantner endpoint used for the update is */api/authorization*. A simplified JSON format of the most important fields is shown in figure 5.5. Gantner authorizations contain a list of users. Each user holds a list of locker authorizations.

The middleware gets all authorizations from Gantner with an HTTP GET request. An existing user authorization in the Gantner system is needed for an update. A new user authorization object must be created if it does not exist. The middleware is creating a new Gantner user authorization object if needed. The required authorization fields are updated otherwise. Updating or creating a new authorization is performed with an HTTP POST request.

5 Results

```
1  [
2    {
3      "cardUID": "",
4      "firstName": "",
5      "lastName": "",
6      "lockerAuthorizations": [
7        {
8          "validFrom": "",
9          "validUntil": "",
10         "lockerNumber": "",
11         "lockerGroupName": "",
12         "lockerGroupRecordId": "",
13         "isDeleted": "",
14         "isModified": ""
15       }
16     ]
17   }
18 ]
```

Listing 5.5: Simplified JSON structure of the Gantner endpoint for updating user authorizations

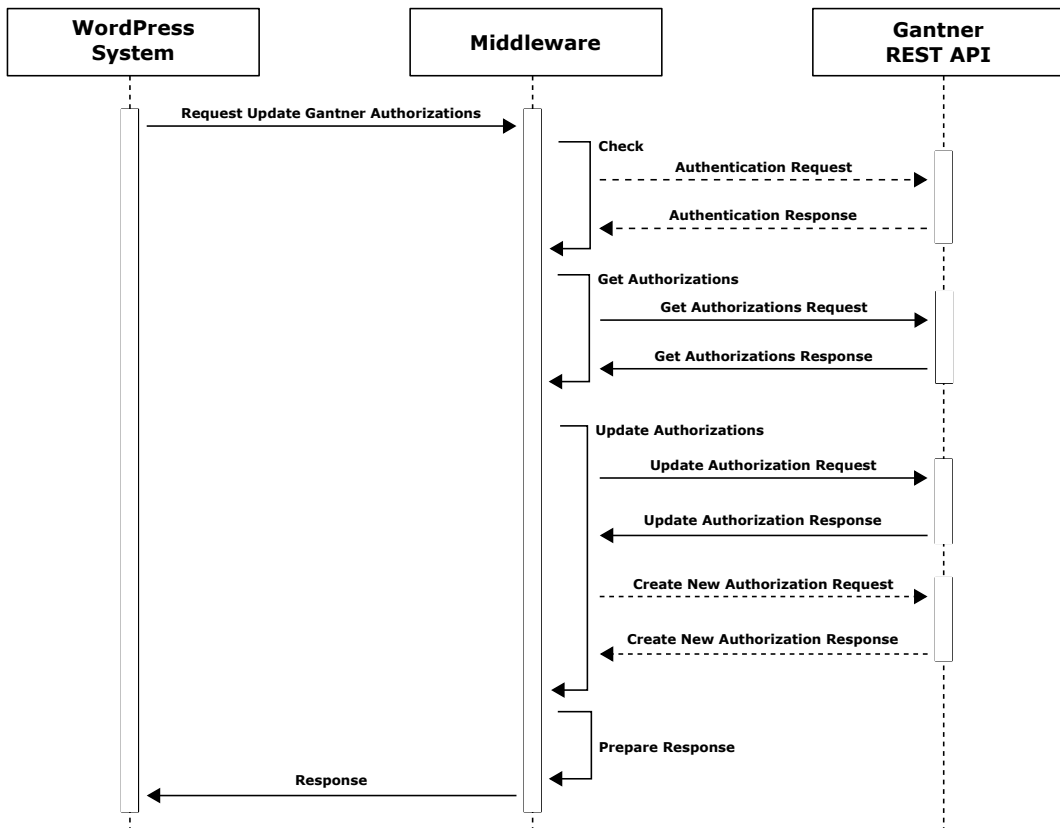


Figure 5.14: Updating Gantner authorizations

Update Gantner Authorization

This endpoint can be used to delete an authorization or to update the *date* attribute. The Gantner endpoint used for this operation is */api/authorization*. The *isDeleted* and *isModified* fields of the Gantner authorization object (figure 5.5) must be used. The WordPress request includes all information needed to delete or update the Gantner authorization. The JSON format of the request is shown in figure 5.6.

```
1 {
2   "person": {
3     "first_name": "",
4     "last_name": "",
5     "card_id": ""
6   },
7   "start_date": "",
8   "end_date": "",
9   "locker_area": "",
10  "locker_number": "",
11  "isModified": "",
12  "isDeleted": ""
13 }
```

Listing 5.6: JSON structure containing all fields for WordPress to delete or update a Gantner authorization

The middleware gets all authorizations from Gantner with an HTTP GET request. The Gantner authorization will be deleted if the *isDeleted* field in the WordPress request is set to *true*. The *validFrom* and *validTo* fields of the Gantner authorization will be updated if the *isModified* field in the WordPress request is set to *true*. The delete and update operation for an Gantner authorization is performed with an HTTP POST request.

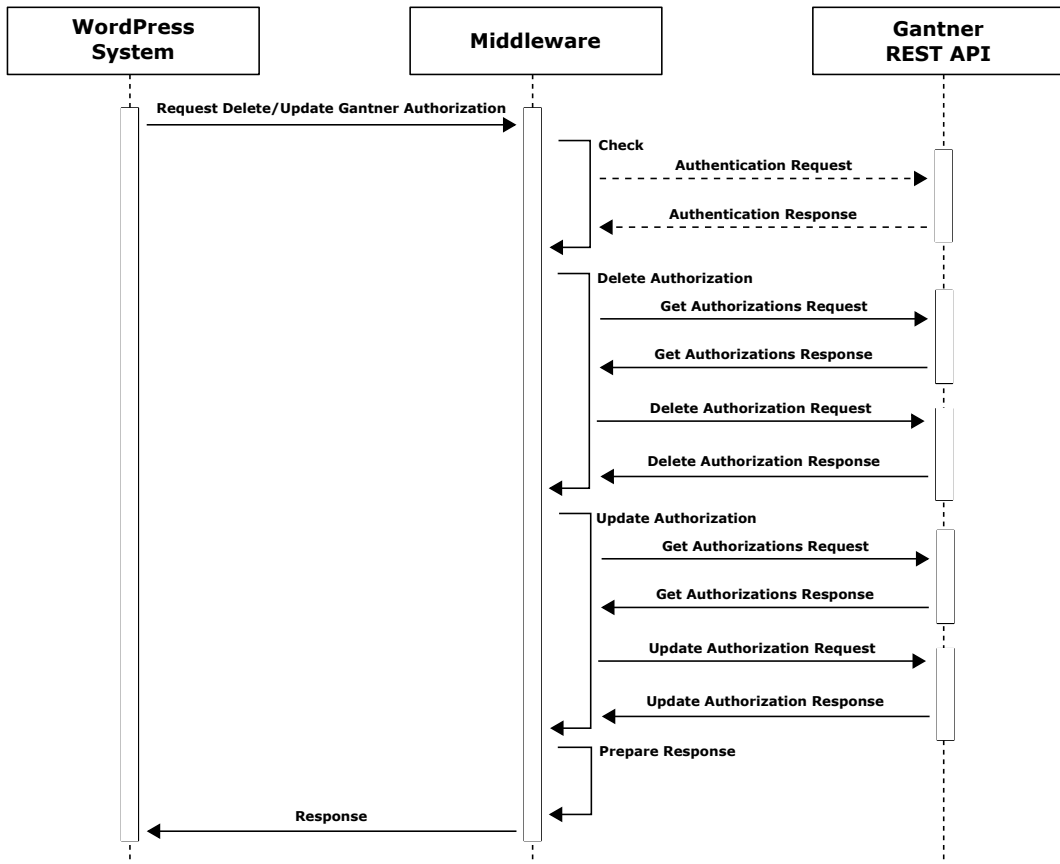


Figure 5.15: Delete or update a Gantner authorization

5.3.2 WordPress - Middleware - Primion Interaction

Figure 5.16 shows a sequence diagram of the interaction between WordPress and the middleware. The WordPress system requests the middleware to create a CSV file with the updated Primion authorizations. The middleware checks the request and saves the updated authorizations in a CSV file. To create the CSV file the *csv* Python module is used. A success message is sent back to WordPress if everything worked fine. An error message is sent otherwise.

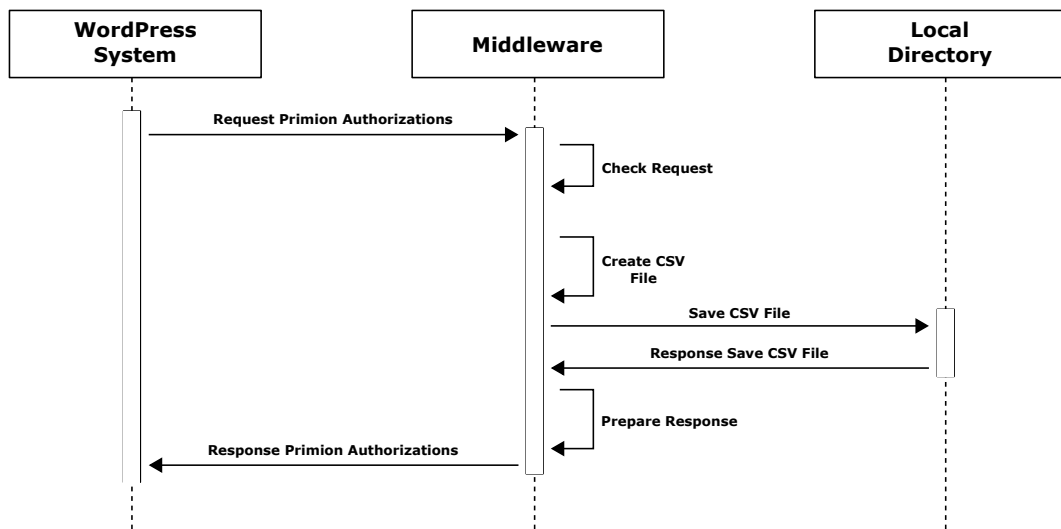


Figure 5.16: Middleware providing the Primion authorizations in CSV format.

5.3.3 WordPress - Middleware - Hoffmann Interaction

Figure 5.17 shows a sequence diagram of the interaction between WordPress, middleware and the Hoffmann system. The middleware provides a REST endpoint for the Hoffmann system. This endpoint requests all Hoffmann authorizations from WordPress. The authorizations are provided by the middleware endpoint in JSON format.

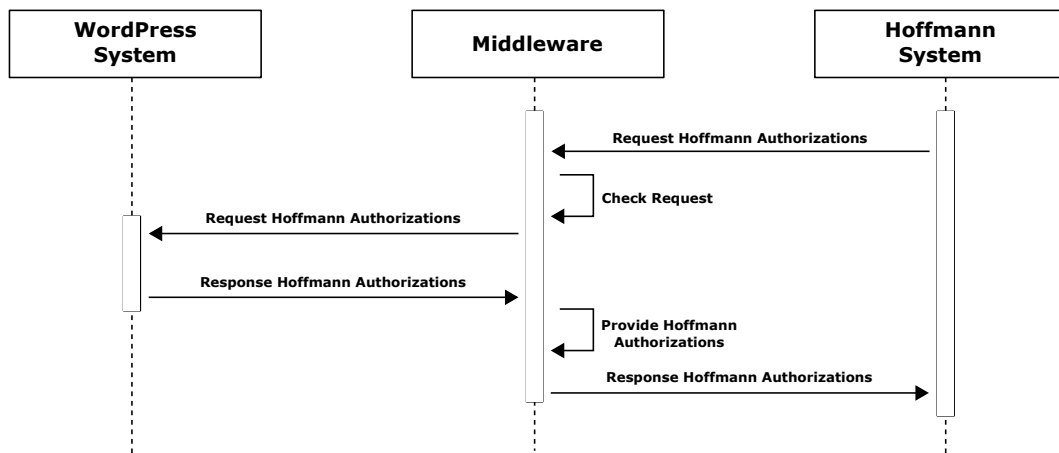


Figure 5.17: Middleware REST endpoint that is providing the Hoffmann authorizations.

5.4 Setting up the WordPress plugin

The custom plugin adds new functionality to the menu bar on the main WordPress page. The menu bar is extended with the *Resources* option. The *Resources* menu contains the following sub-menus:

- Gantner Resources
- Primion Resources
- Hoffmann Resources

The sub-menu pages were created by using the WordPress administrator dashboard (figure 5.2). WordPress *shortcodes* are used to create custom content on the pages. A *shortcode* is a piece of code that can be embedded into WordPress pages. Custom functions can be created by developers to load dynamic content (e.g.: content from external sites). *Shortcodes* are written inside two square brackets as shown in figure 5.19. [Wor21g]

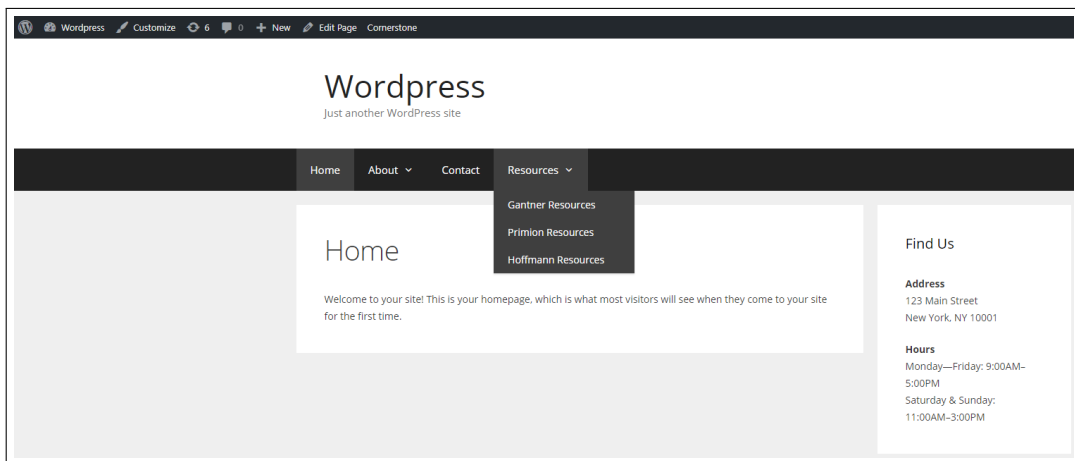


Figure 5.18: The main menu extended with the *Resources* options.

The functions that are executed by the *shortcodes* are implemented in the custom plugin. Activating the plugin performs the following actions:

- Load scripts
- Create custom shortcodes
- Create REST endpoints

Load scripts The plugin uses external libraries that are needed to create dynamic content (e.g.: style sheets, date picker). External resources must be registered before they can be used by the plugin. WordPress *Action* hooks (section 5.1.3) are used to register scripts. The proper hook name is *wp_enqueue_scripts*. [Wor21e]

Create custom shortcodes The callback functions for the created *shortcodes* on the resource pages are registered by the plugin. WordPress provides the *add_shortcode()* function to register a *shortcode*. [Wor21f]

Create REST endpoints The REST endpoints that are used to communicate with external sites (middleware) are registered by the plugin. WordPress *Action* hooks (section 5.1.3) are used to register REST endpoints. The proper hook name is *rest_api_init*. [Wor21d]

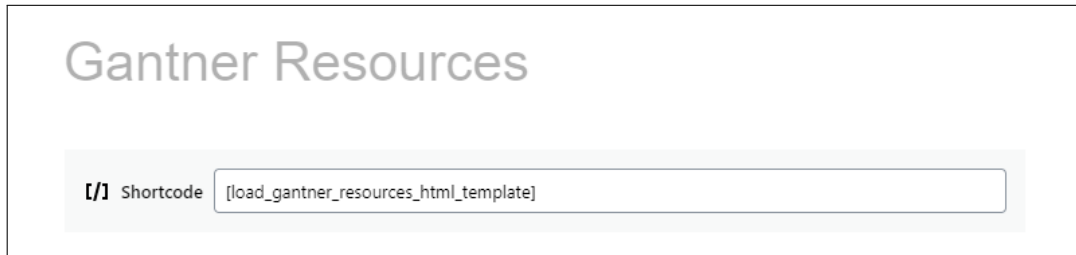


Figure 5.19: Custom WordPress *shortcode* used to load the dynamic content for the Gantner resources.

The *Resources* pages load the content that is displayed to the user dynamically. A JavaScript event is triggered when the page gets loaded. This event uses the AJAX technology and performs an HTTP request to the REST APIs provided by WordPress. The server executes the requests and sends back a server response to the AJAX call. The response data is processed and displayed to the user. Figure 5.20 shows an overview of the front-end interaction of the concept.

WordPress User Management

The *Resources* extension is only available for authorized users. Authorized users are assigned the capability to administrate the Gantner, Primion and Hoffmann resources. The capabilities were created within the WordPress administrator dashboard 5.2. The *User Role Editor* plugin was used to create new capabilities. The capabilities are prefixed with *iim_name_of_capability*. The *shortcodes* embedded into the *Resources* pages are only loaded if the user has the required capability. [Gar21]

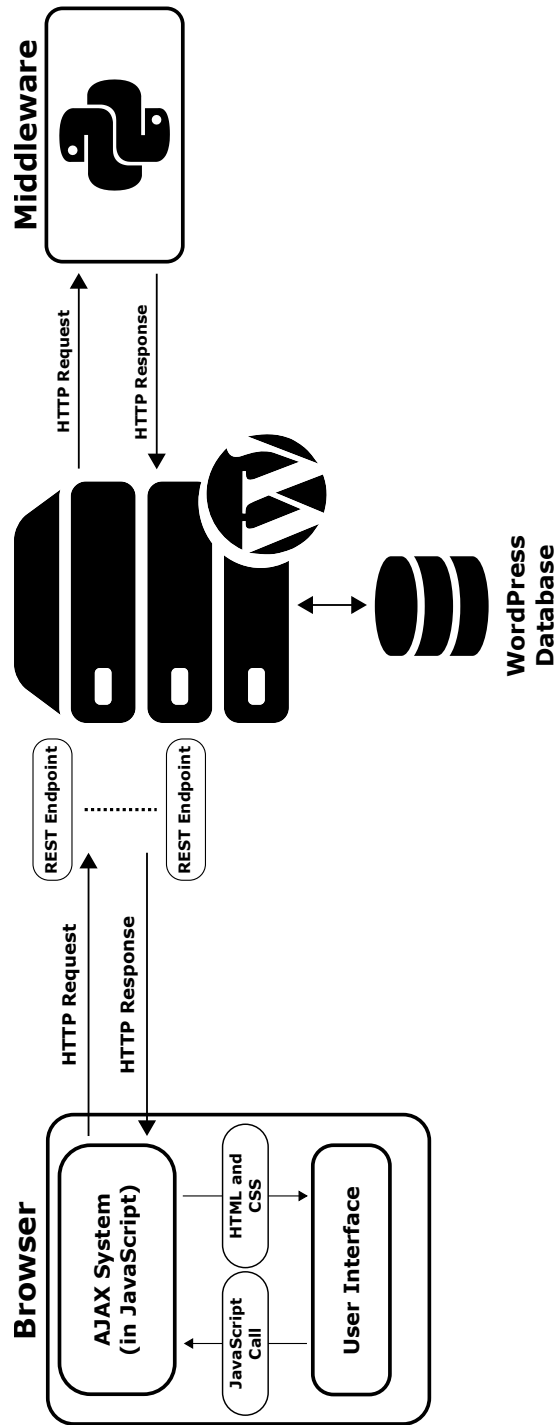


Figure 5.20: Overview of the front-end interaction

5.4.1 Manage Gantner Resources

Authorized users can administrate the authorizations for the Gantner system within the *Gantner Resources* sub-menu 5.18. The page uses the REST endpoint provided by WordPress to request the Gantner state as shown in figure 5.20. WordPress requests the Gantner state information from the middleware as explained in section 5.3.1. The response is processed and shown to the user. Figure 5.21 shows the created HTML output that represents the Gantner state.

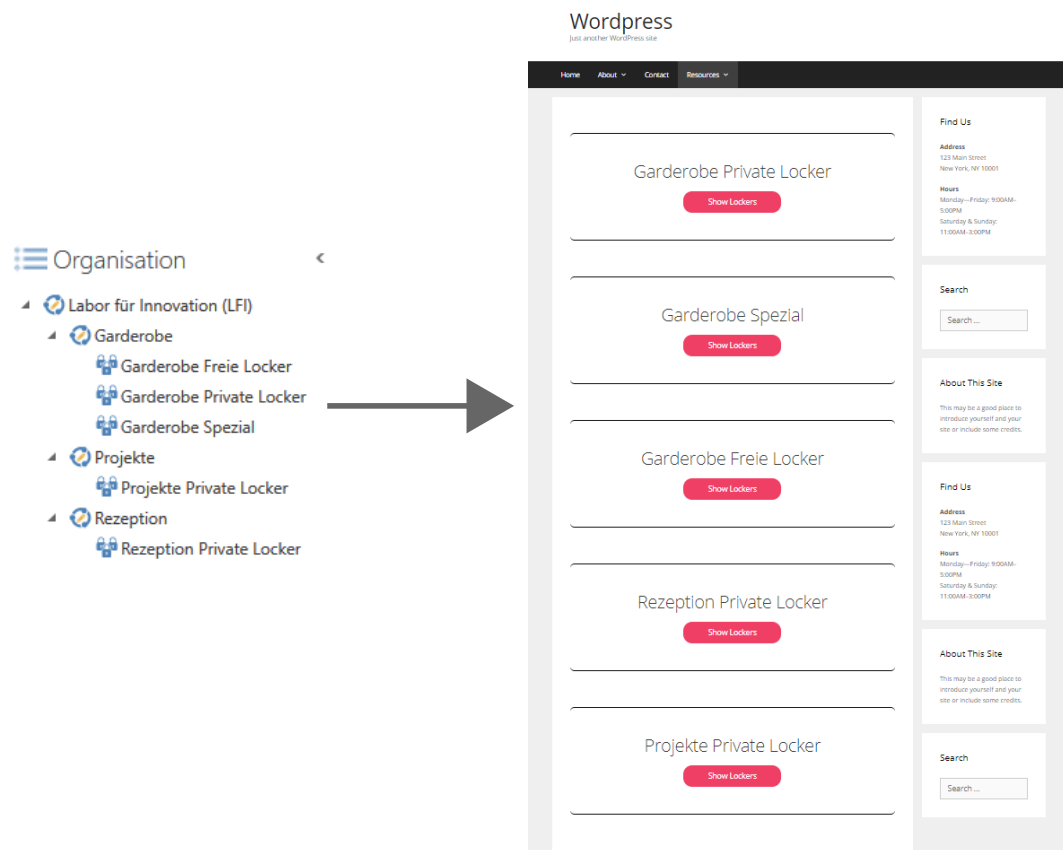


Figure 5.21: The left side represents an example organisation view created within the GAT Relaxx desktop client application. The Gantner state represented as HTML output on the *Gantner Resources* page is shown on the right side.

5 Results

Figure 5.22 shows an example locker group. This locker group holds 12 lockers without any authorizations.

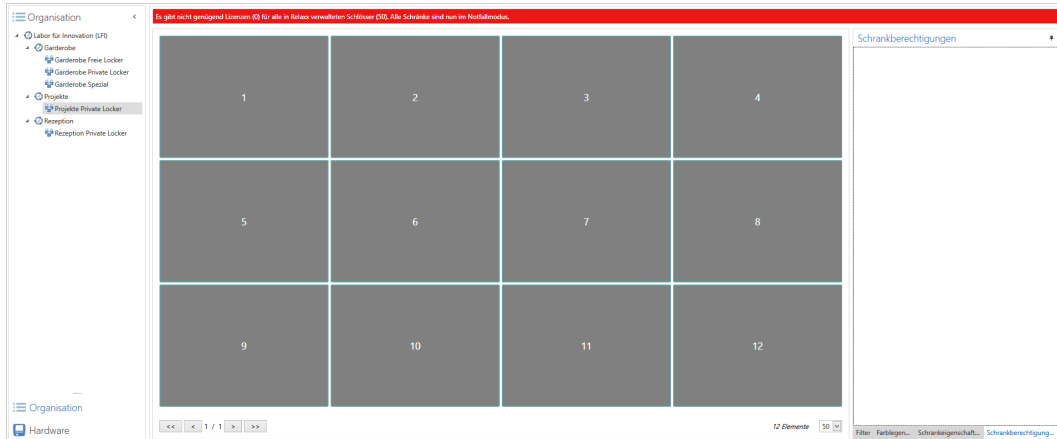


Figure 5.22: Locker group holding 12 lockers.

Figure 5.23 shows how the custom plugin presents the example locker group for administrators. The *Show Lockers* event expands the view and shows all lockers that are assigned to the locker group. Lockers can also be expanded to manage user authorizations. On the right side of each locker, a number is displayed. This number shows the amount of authorizations that a locker holds.

A click on *Manage Authorizations* allows administrators to assign new user authorizations to the selected locker. An event is triggered that gets all users from the WordPress database. Users that have an authorization for the selected locker will not be shown.

A *modal* form allows assigning one or more users to the selected locker. The modal displays the first name, last name, card identification number and the state of the users. New authorizations will be assigned by:

- Switching the state of one or more users
- Selecting the start date and end date

5 Results

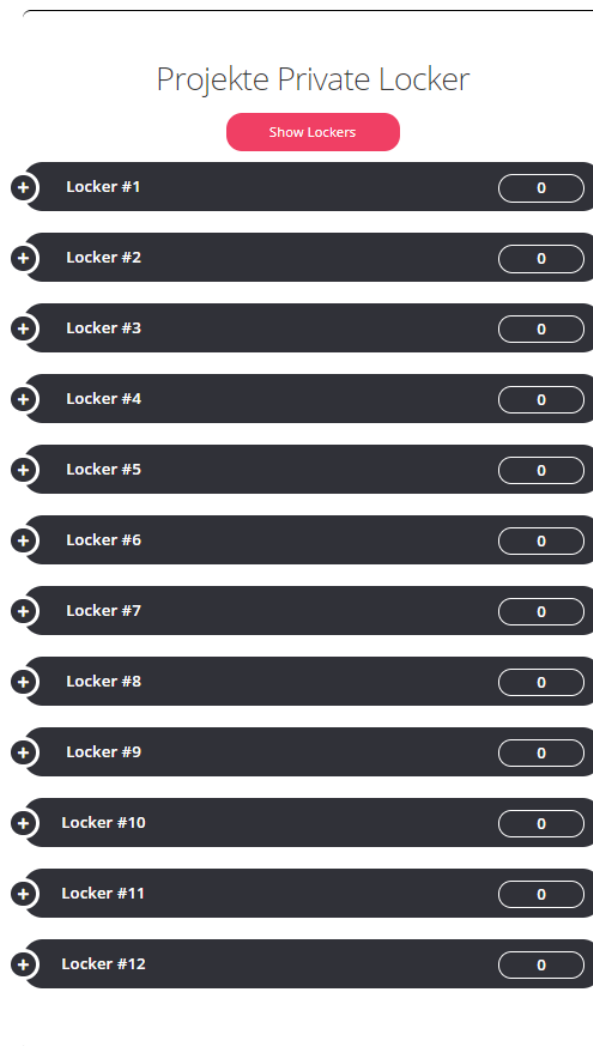


Figure 5.23: Plugin representation of the example locker group.

5 Results

One or more users can be assigned to a locker at once by switching the state flag to *on*. The start date and end date for the authorization(s) must be selected. A date picker is used to configure the date information. By clicking the save button an event is triggered. This event sends the date to the corresponding WordPress REST endpoint. WordPress forwards the request to the middleware. The middleware updates the user authorizations as explained in section 5.3.1.

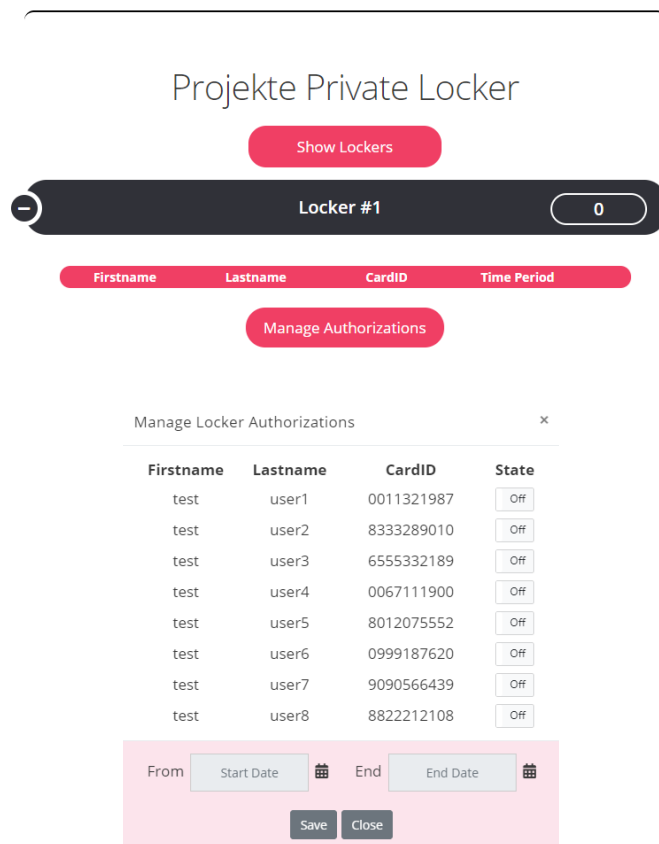
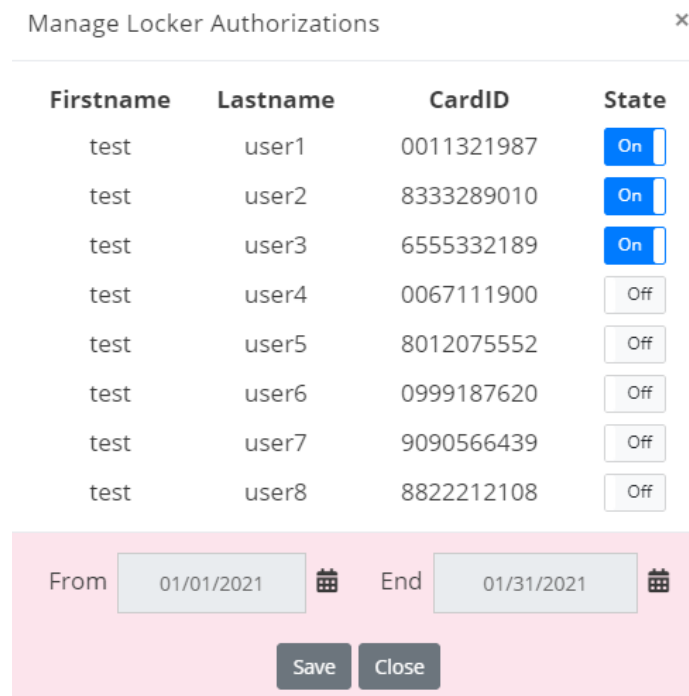


Figure 5.24: Gantner authorizations administration with the plugin.

Figure 5.25 shows how three users are assigned to locker one in the selected

locker group *Projekte Private Locker*. After saving the authorizations the user interface will be updated with the new information.



Firstname	Lastname	CardID	State
test	user1	0011321987	<input checked="" type="checkbox"/> On
test	user2	8333289010	<input checked="" type="checkbox"/> On
test	user3	6555332189	<input checked="" type="checkbox"/> On
test	user4	0067111900	<input type="checkbox"/> Off
test	user5	8012075552	<input type="checkbox"/> Off
test	user6	0999187620	<input type="checkbox"/> Off
test	user7	9090566439	<input type="checkbox"/> Off
test	user8	8822212108	<input type="checkbox"/> Off

From End

Figure 5.25: Assign three user authorizations to the selected locker group.

The newly added user authorizations are shown in figure 5.26. The locker contains three authorizations which are indicated by the number on the right side of the locker information. Authorizations can be deleted by clicking on the trash icon. This triggers an event that will be sent to the associated WordPress REST endpoint. The update will be performed by the middleware as explained in section 5.3.1. Updating the validity can be managed by clicking on the start or end date of the authorization. New dates can be selected with the date picker. Selecting a new date triggers an event to the corresponding WordPress REST endpoint. The update will be executed by the middleware as explained in section 5.3.1.

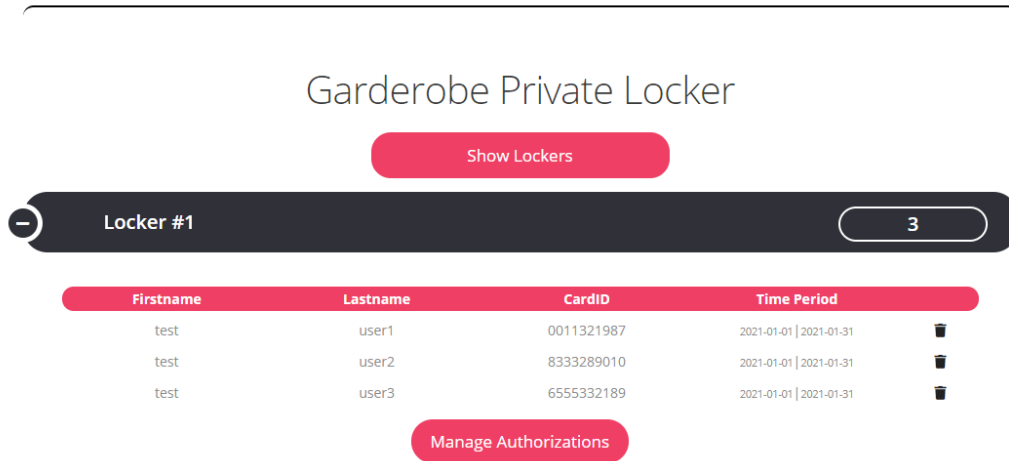


Figure 5.26: Updated locker information

5.4.2 Manage Primion Resources

The Primion authorizations can be managed within the *Primion Resources* sub-menu 5.18. The page uses a REST endpoint provided by WordPress to request the Primion state. The simulated state is loaded from the WordPress database and displayed as shown in figure 5.27. The state simulates the Primion locks: *Main Building Entrance*, *SLFI Entrance*, *Elevator*, *Schumpeter Meeting Room*, *Tesla Meeting Room*, *Design Lab Door I*, *Design Lab Door II*, *FabLab I* and *FabLab II* that are used in the SLFI as shown in figure 3.11.

The *Show Authorizations* event expands the view (figure 5.28) and shows all active authorizations for the selected resource. The *Manage Authorizations* event allows administrators to assign new authorizations. Allocating new authorizations is performed the same way as assigning Gantner resources explained in section 5.4.1.

By saving new authorizations an event is triggered. This event sends the data to a REST endpoint provided by WordPress. WordPress creates the new authorizations in the WordPress database. By clicking the trash icon the selected authorization will be deleted. An event is triggered that sends the

5 Results

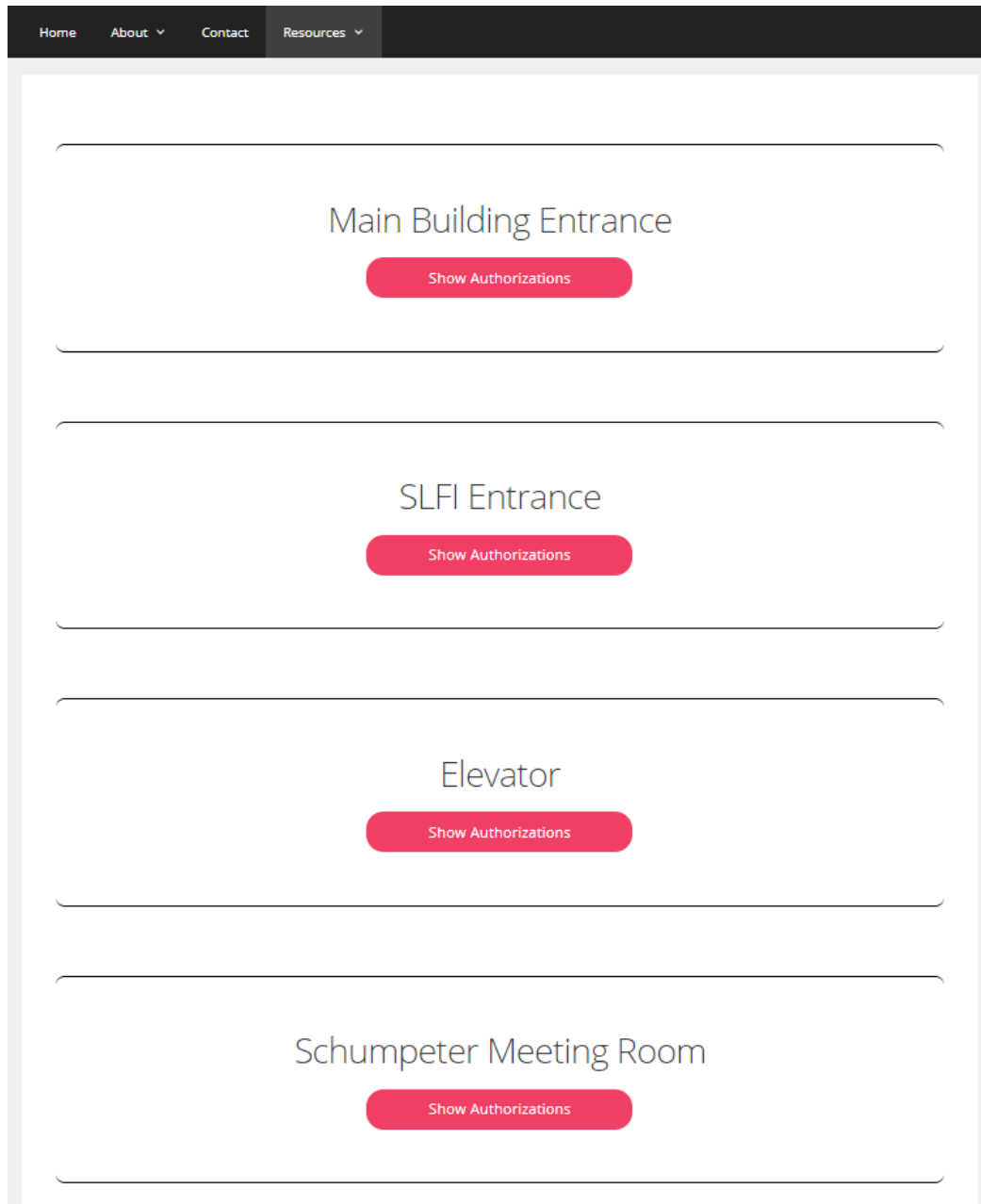


Figure 5.27: The simulated Primion state loaded from the WordPress database.

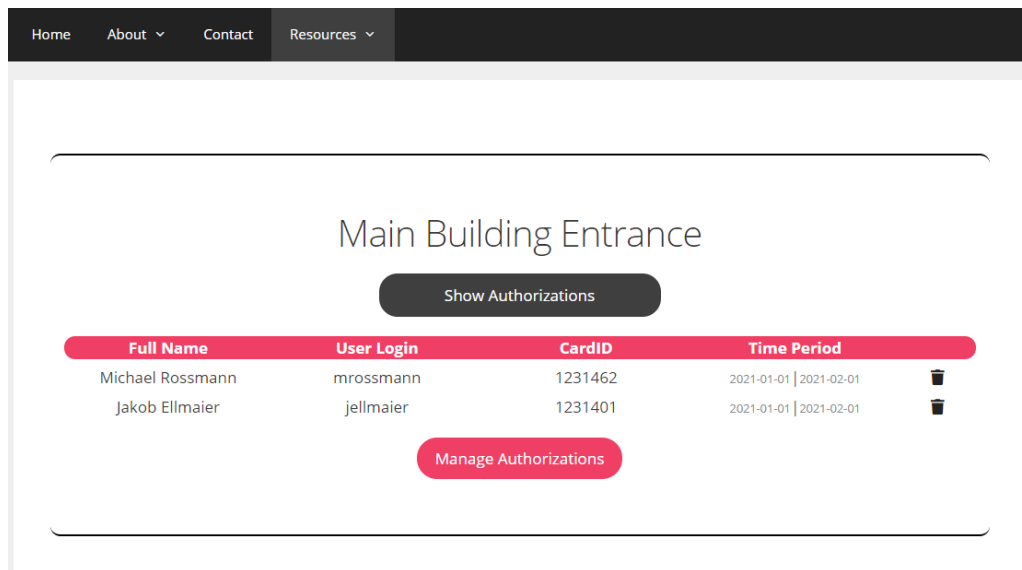


Figure 5.28: Assign new authorizations to the selected Primion resource.

information to the WordPress server. WordPress deletes the authorization from the database. Updating the *Time Period* information of an authorization also triggers an event that will be handled by WordPress. An update operation is performed on the selected user authorization.

Primion CSV File

Changes that are applied to the Primion authorizations are handled by WordPress. When the changes are applied to the database, WordPress is also requesting the middleware to update the Primion CSV file. The file is created and updated by the middleware. WordPress sends the Primion state information to the provided REST endpoint. The Python *csv* module is used to update the CSV file. The default file path for the CSV file is the custom plugin folder. Figure 5.29 shows an example CSV file. The Primion state contains four authorizations for two different users. The CSV file represents this information in CSV format.

5 Results

Main Building Entrance

Show Authorizations

Firstname	Lastname	CardID	Time Period	
Michael	Rossmann	8865332901	2021-01-01 2021-02-01	🗑️
test	user1	0011321987	2021-01-01 2021-02-01	🗑️


Manage Authorizations

SLFI Entrance

Show Authorizations

Firstname	Lastname	CardID	Time Period	
Michael	Rossmann	8865332901	2021-01-01 2021-02-01	🗑️
test	user1	0011321987	2021-01-01 2021-02-01	🗑️

Manage Authorizations



```
1 FullName;CardID;Resource;ValidFrom;ValidUntil
2 Michael Rossmann;8865332901;Main Building Entrance;2021-01-01;2021-02-01
3 test user1;0011321987;Main Building Entrance;2021-01-01;2021-02-01
4 Michael Rossmann;8865332901;SLFI Entrance;2021-01-01;2021-02-01
5 test user1;0011321987;SLFI Entrance;2021-01-01;2021-02-01
```

Figure 5.29: The CSV file will be created and updated by the middleware. The file contains all Primion authorizations that are stored in the WordPress database.

5.4.3 Manage Hoffmann Resources

Hoffmann authorizations can be managed within the *Hoffmann Resources* sub-menu 5.18. A REST endpoint provided by WordPress is used to gather the required information. WordPress loads the simulated data from the database and displays the hoffmann state as shown in figure 5.31. The resources: *Workstation I*, *Workstation II*, *3D Printer*, *Laser Cutter* and *CNC Milling Machines* are used to simulate the locks in FabLab I and FabLab II as shown in figure 3.19.

The *Show Authorizations* event expands the view (figure 5.30) and shows all active authorizations for the selected resource. New authorizations can be assigned with *Manage Authorizations*. Allocating new authorizations is handled the same way as assigning Gantner or Primion resources explained in section 5.4.1. The saved authorizations are sent to a WordPress REST endpoint. WordPress saves the new authorizations in the database. Updating the *Time Period* information and deleting an authorization is also handled by WordPress. The information is sent to provided REST endpoints. WordPress performs the update or delete operation respectively.

Workstation I

Show Authorizations

Firstname	Lastname	CardID	Time Period	
Michael	Rossmann	8865332901	2021-01-01 2021-02-01	
test	user1	0011321987	2021-01-01 2021-02-01	

Manage Authorizations

Figure 5.30: Assign new authorizations to the selected Hoffmann resource.

5 Results

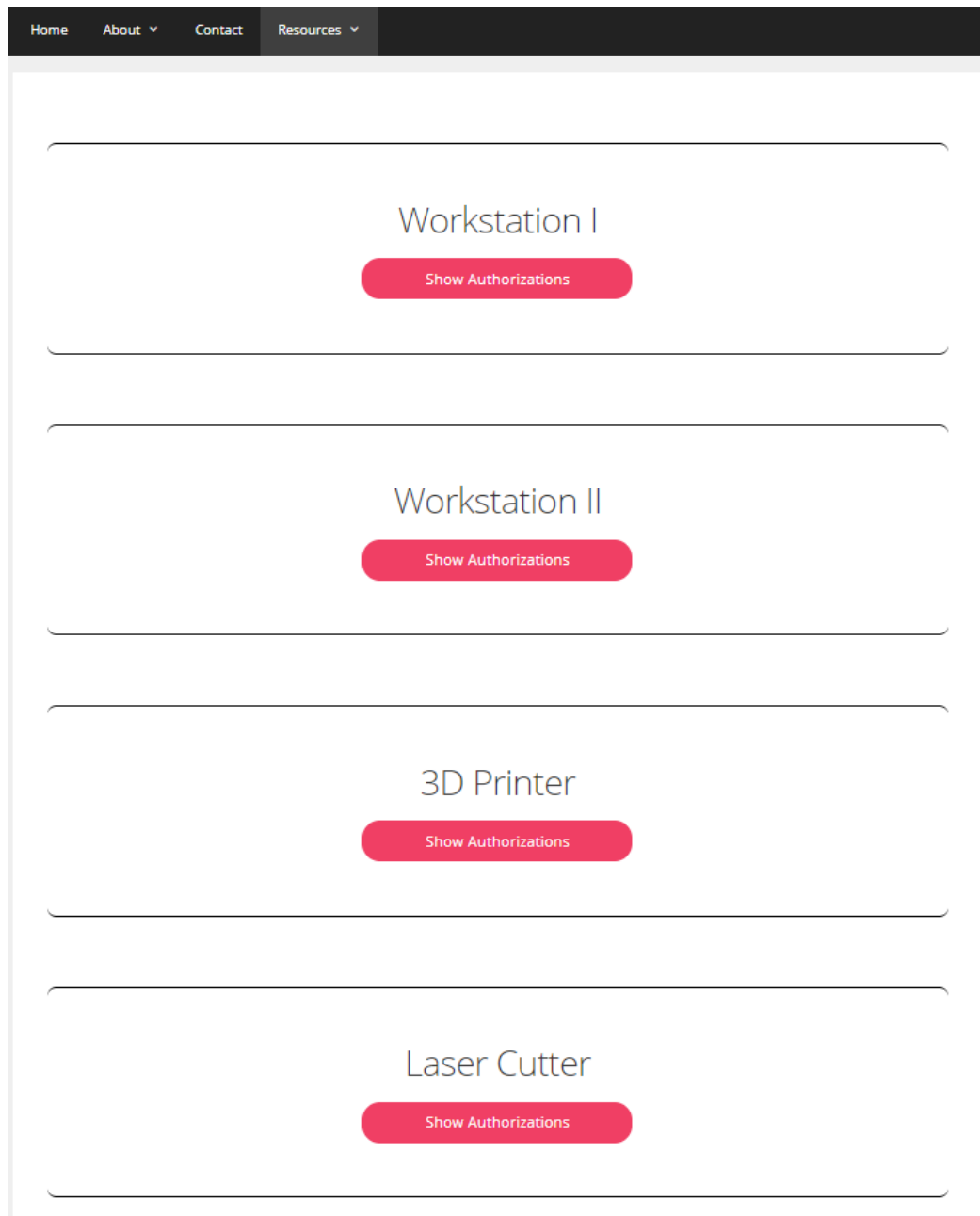


Figure 5.31: The simulated Hoffmann state loaded from the WordPress database.

Hoffmann State Endpoint

The simulated Hoffmann locks and authorizations are stored in the WordPress database. This information can be gathered from a REST endpoint provided by WordPress. The front end is using this endpoint to display the state to the administrators. The middleware is also using this endpoint to get the Hoffmann state information. A REST endpoint is provided that displays the Hoffmann state in JSON format 5.2. This endpoint can be used by the Hoffmann mobile app to update and synchronize the locks as explained in section 3.3.13.

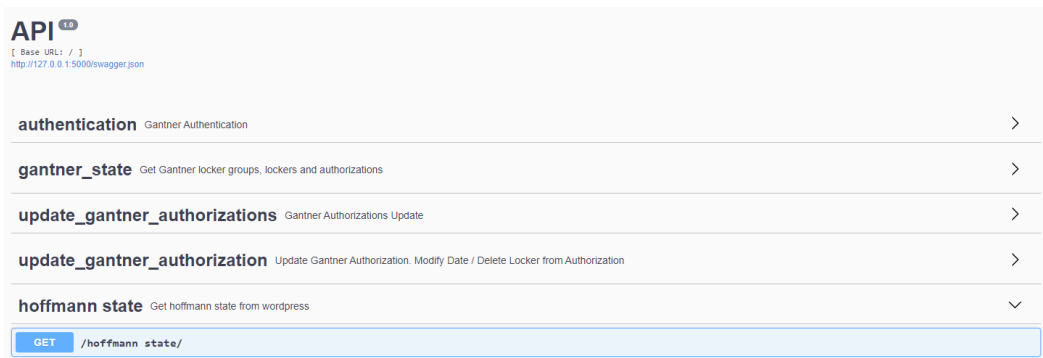


Figure 5.32: The REST endpoint provided by the middleware to get the Hoffmann state.

6 Conclusion

This master thesis showed how a central administration platform for electronic locking systems from different manufacturers can be implemented. Two important requirements were specified:

- a central platform for administrators to manage the different locking systems
- a uniform data carrier that can be used to operate the various locking systems

Both requirements can be fulfilled with the developed solution. The already used in-house WordPress system was extended to serve as a central administration platform for the different locking systems Gantner, Primion, and Hoffmann. MIFARE Classic and MIFARE DESFire EV1 cards can be used as a uniform data carrier for all systems. Table 6.1 summarizes what the system can do and what is still needed.

The extended WordPress system provides a unified GUI for the administrators of the SLFI to manage the different electronic locking systems. The Gantner system was successfully integrated. This system can be administered using the provided REST endpoints. The two offline solutions Primion and Hoffmann still require additional external work. Figure 6.1 shows the concept that was implemented in the SLFI. The red triangles indicate where additional work is needed.

Hoffmann is still working on expansion phase II that would allow the locks to read authorizations from MIFARE Classic and MIFARE DESFire EV1 cards. This expansion of Hoffmann will continue indefinitely and will involve additional costs.

Expansion phase I is already completed. This extension allows MIFARE Classic and MIFARE DESFire EV1 data cards to be used as access tags instead of the custom Hoffmann tags. The existing WordPress database was extended to simulate information about the Hoffmann authorizations. This information can be retrieved from a provided REST endpoint and imported into the mobile application from Hoffmann. To be able to import the changes into the mobile application, an interface must be provided. According to the responsible persons at Hoffmann, it is possible to provide such an interface. A possible interface for the mobile application from Hoffmann as an additional extension to the expansion phase one can be implemented more quickly is less expensive and requires less effort. This extension will have to be specified in more detail in the future.

The existing WordPress database was extended to simulate information about the Primion authorizations. Information about Primion authorizations is provided as a CSV file that can be imported by the ZID. The exact format of the CSV file has to be specified more precisely in the future. Only then it is possible to import the CSV file correctly into the Primion system. Additionally, it must be defined how the CSV file should be sent to the ZID. Currently, the CSV file is stored directly in the running environment of the WordPress system.

Real Environment

The plugin is installed on the server running the live version of WordPress of the SLFI. The middleware is also installed on this server. Necessary settings for the middleware, which allow communication with the Ganter REST endpoints, were configured with responsible persons of the ZID.

Future Systems

The solution described in this thesis can be used as a framework for integrating other systems in the future. The programming logic to integrate a new system can be programmed in the middleware using the Python programming language. The graphical representation can be programmed within the custom WordPress plugin. The flexibility of custom WordPress plugins also allows fast prototyping. The custom plugin can be easily deactivated, extended, and tested in a secure test environment. The software architecture chosen in this project should facilitate future integrations in the SLFI.

The implemented solution provides	The implemented solution still requires
Central platform to administer the various electronic locking systems.	Primion CSV format specification
Provides the option of using MIFARE Classic and MIFARE DESFIRE EV1 cards as uniform data carrier.	Primion CSV file transfer specification
Fully integrated Gantner system	Hoffmann mobile application API
Partly integrated Primion system. A CSV file will be created that can be imported by Primion	
Partly integrated Hoffmann system. Authorization updates for the Hoffmann system will be provided by a REST endpoint. This endpoint can be used by Hoffmann to update the authorizations stored in the mobile application database	

Table 6.1: Summary of what the system can do and what is still needed.

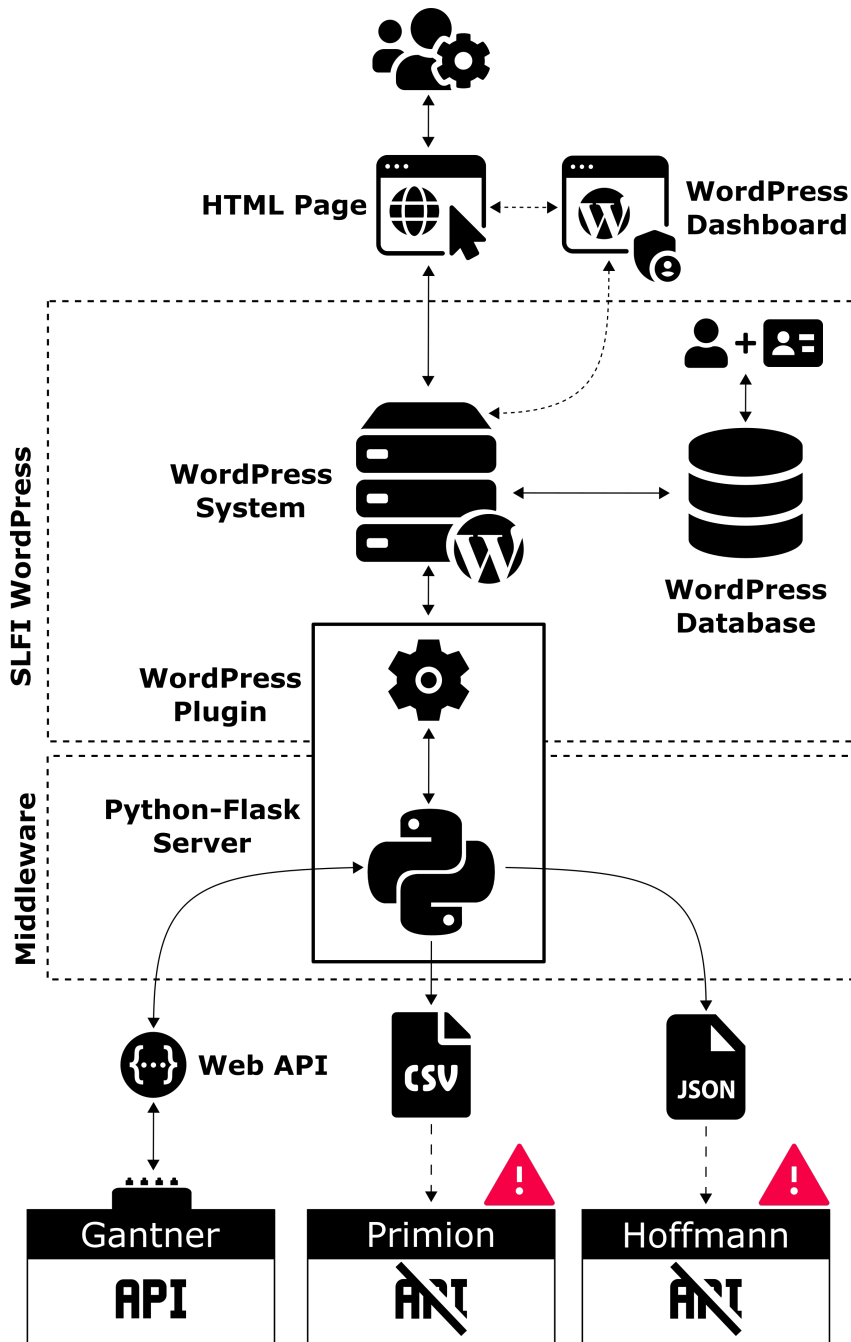


Figure 6.1: The implemented concept I. The red triangles indicate where additional work is needed.

Appendix

Bibliography

- [al21] Mehdi Achour et al. *PHP Manual*. 2021. URL: <https://www.php.net/manual/de/index.php> (visited on 04/05/2021) (cit. on p. 79).
- [Alt17] Alexandra Altvater. *What are CRUD Operations: How CRUD Operations Work, Examples, Tutorials & More*. 2017. URL: https://stackify.com/what-are-crud-operations/#wpautbox_about (visited on 03/09/2021) (cit. on pp. 8, 10).
- [Ass21] OSS Association. *OSS-Association e.V (Open Security Standards Association)*. 2021. URL: <https://www.oss-association.com/> (cit. on pp. 26, 28).
- [BC95] Tim Berners-Lee and Daniel W. Connolly. *Hypertext Markup Language - 2.0*. RFC 1866. Nov. 1995. DOI: 10.17487/RFC1866. URL: <https://rfc-editor.org/rfc/rfc1866.txt> (cit. on p. 6).
- [BFM05] T. Berners-Lee, R. Fielding, and L. Masinter. *RFC 3986, Uniform Resource Identifier (URI): Generic Syntax*. 2005. URL: <http://rfc.net/rfc3986.html> (cit. on p. 6).
- [Bra17] Tim Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC 8259. Dec. 2017. DOI: 10.17487/RFC8259. URL: <https://rfc-editor.org/rfc/rfc8259.txt> (cit. on p. 76).
- [Bus19] Thomas Bush. *What Is The Difference Between Web Services and APIs?* 2019. URL: <https://nordicapis.com/what-is-the-difference-between-web-services-and-apis/> (visited on 03/09/2021) (cit. on p. 10).
- [Cos+14] B. Costa et al. "Evaluating a Representational State Transfer (REST) Architecture: What is the Impact of REST in My Architecture?" In: *2014 IEEE/IFIP Conference on Software Architecture*. 2014, pp. 105–114 (cit. on p. 12).

Bibliography

- [Doc21] Python Docs. *Python Requests*. 2021. URL: <https://docs.python-requests.org/en/master/> (visited on 04/05/2021) (cit. on p. 72).
- [Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. 2000 (cit. on pp. 11–13).
- [FR14] Roy T. Fielding and Julian Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231. June 2014. DOI: 10.17487/RFC7231. URL: <https://rfc-editor.org/rfc/rfc7231.txt> (cit. on p. 6).
- [Gar21] Vladimir Garagulya. *User Role Editor*. 2021. URL: <https://de.wordpress.org/plugins/user-role-editor/> (visited on 04/05/2021) (cit. on p. 96).
- [Gen21] GeneratePress. *GeneratePress Theme*. 2021. URL: <https://generatepress.com/> (visited on 04/05/2021) (cit. on p. 78).
- [Gmb20] GANTNER Electronic GmbH. *GANTNER Electronic GmbH*. 2020 (accessed November 19, 2020). URL: <https://www.gantner.com/de/> (cit. on pp. 35–37).
- [Gro21a] Hoffmann Group. *Hoffmann Group*. 2021. URL: <https://www.hoffmann-group.com/DE/de/hom/> (visited on 03/10/2021) (cit. on pp. 39–42).
- [Gro21b] Hoffmann Group. *GARANT Elektronisches Schloss G-ELS*. 2020 (accessed March 10, 2021). URL: https://www.hoffmann-group.com/medias/sys_master/images/images/hfb/hd1/9276494151710/BA-GARANT-Elektronisches-Schloss-G-ELS-05656IN.pdf?attachment=true (cit. on p. 39).
- [GT16] D. Guinard and V. Trifa. *Building the Web of Things: With examples in Node.js and Raspberry Pi*. Manning Publications, 2016. ISBN: 9781617292682. URL: <https://books.google.at/books?id=Q1b2jwEACAAJ> (cit. on pp. 3, 4).
- [Gui11] Dominique Guinard. “A Web of Things Application Architecture – Integrating the Real-World into the Web.” PhD thesis. Zurich, Switzerland: ETH Zurich, Aug. 2011 (cit. on p. 5).

Bibliography

- [Gui16] Dominique Guinard. *Web of Things vs Internet of Things: 1/2*. 2016. URL: <https://webofthings.org/2016/01/23/wot-vs-iot-12/> (visited on 03/10/2021) (cit. on p. 7).
- [Hau21] Axel Haustant. *Flask-RESTPlus*. 2021. URL: <https://flask-restplus.readthedocs.io/en/stable/> (visited on 04/05/2021) (cit. on p. 72).
- [HK11] Festim Halili and Merita Kasa Halili. "Analysis and comparison of web service architectural styles, and business benefits of their use." In: June 2011 (cit. on pp. 10, 11).
- [HR18] Festim Halili and Erenis Ramadani. "Web Services: A Comparison of Soap and Rest Services." In: *Modern Applied Science* 12 (Feb. 2018), p. 175. DOI: 10.5539/mas.v12n3p175 (cit. on p. 14).
- [Inf21] JavaScript Info. *An Introduction to JavaScript*. 2021. URL: <https://javascript.info/intro> (visited on 04/05/2021) (cit. on p. 79).
- [Ini20] OpenAPI Initiative. *OpenAPI Specification*. 2020. URL: <https://www.openapis.org/> (cit. on p. 15).
- [Inn20a] Institut für Innovation und Industrie Management. *FabLab*. 2020 (accessed November 19, 2020). URL: <https://fablab.tugraz.at/> (cit. on p. 21).
- [Inn20b] Institut für Innovation und Industrie Management. *Schumpeter Labor für Innovation*. 2020 (accessed November 19, 2020). URL: <https://www.tugraz.at/institute/iim/infrastruktur/schumpeter-labor-fuer-innovation/> (cit. on pp. 20–22).
- [J3S02] Joseph M. Reagle Jr., Donald E. Eastlake 3rd, and David Solo. *(Extensible Markup Language) XML-Signature Syntax and Processing*. RFC 3275. Mar. 2002. DOI: 10.17487/RFC3275. URL: <https://rfc-editor.org/rfc/rfc3275.txt> (cit. on p. 6).
- [Jet21] JetBrains. *JetBrains development tools*. 2021. URL: <https://www.jetbrains.com> (visited on 04/05/2021) (cit. on p. 79).
- [KK18] István Koren and Ralf Klamma. "The Exploitation of OpenAPI Documentation for the Generation of Web Frontends." In: Apr. 2018, pp. 781–787. DOI: 10.1145/3184558.3188740 (cit. on p. 15).

Bibliography

- [Lib21] Python Standard Library. *CSV File Reading and Writing*. 2021. URL: <https://docs.python.org/3/library/csv.html> (visited on 04/05/2021) (cit. on p. 76).
- [Mad15] Somayya Madakam. *Internet of Things: Smart Things*. 2015. DOI: 10.7763/IJFCC.2015.V4.395 (cit. on p. 3).
- [Mak] Makerspaces.com. *What is a Makerspace?* <https://www.makerspaces.com/what-is-a-makerspace>. Accessed: 2021-03-02 (cit. on p. 1).
- [Mak21a] Matt Makai. *Flask*. 2021. URL: <https://www.fullstackpython.com/flask.html> (visited on 04/05/2021) (cit. on p. 71).
- [Mak21b] Matt Makai. *Web Frameworks*. 2021. URL: <https://www.fullstackpython.com/web-frameworks.html> (visited on 04/05/2021) (cit. on p. 71).
- [Mic21a] Microsoft. *.NET Core Hosting Bundle*. 2021. URL: <https://dotnet.microsoft.com/download/dotnet> (visited on 04/05/2021) (cit. on p. 80).
- [Mic21b] Microsoft. *IIS*. 2021. URL: <https://www.iis.net/> (visited on 04/05/2021) (cit. on p. 80).
- [Mic21c] Microsoft. *Microsoft SQL Server*. 2021. URL: <https://www.microsoft.com/de-de/sql-server/sql-server-downloads> (visited on 04/05/2021) (cit. on p. 80).
- [Mic21d] Microsoft. *Microsoft SQL Server Management Studio*. 2021. URL: <https://docs.microsoft.com/en-us/sql/ssms/> (visited on 04/05/2021) (cit. on p. 80).
- [Mon20] Anna Monus. *SOAP vs REST vs JSON - a 2020 comparison*. 2020. URL: <https://raygun.com/blog/soap-vs-rest-vs-json/> (visited on 03/10/2021) (cit. on p. 14).
- [ORG21] JSON ORG. *Einführung in JSON*. 2021. URL: <https://www.json.org/json-de.html> (visited on 04/05/2021) (cit. on p. 76).
- [Org21a] Python Org. *Python Executive Summary*. 2021. URL: <https://www.python.org/doc/essays/blurb/> (visited on 04/05/2021) (cit. on p. 78).
- [Org21b] WordPress Org. *Releases*. 2021. URL: <https://wordpress.org/download/releases/> (visited on 04/05/2021) (cit. on p. 78).

Bibliography

- [Pri21] Primion. *Komponenten der Zutrittskontrolle von primion*. 2021. URL: <https://www.primion.de/de/loesungen/zutrittskontrolle/weitere-komponenten-der-zutrittskontrolle/> (cit. on p. 31).
- [PRP11] Savan Patel, V. Rathod, and Jigna Prajapati. "Performance Analysis of Content Management Systems Joomla, Drupal and WordPress". In: *International Journal of Computer Applications* 21 (May 2011), pp. 39–43. DOI: 10.5120/2496-3373 (cit. on p. 16).
- [Red11] Martin Reddy. *API Design for C++*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN: 9780123850034 (cit. on pp. 8, 9).
- [Rel19] K. Relan. *Building REST APIs with Flask: Create Python Web Services with MySQL*. Apress, 2019. ISBN: 9781484250235. URL: <https://books.google.at/books?id=wM9GzQEACAAJ> (cit. on pp. 9, 10, 12).
- [Sem] NXP Semiconductors. *MIFARE Application Directory (MAD)*. <https://www.nxp.com/docs/en/application-note/AN10787.pdf>. Accessed: 2021-05-05 (cit. on p. 27).
- [Sem21] NXP Semiconductors. *Mifare Products*. 2021. URL: https://www.nxp.com/products/rfid-nfc/mifare-hf:MC_53422 (visited on 05/01/2021) (cit. on pp. 27–29).
- [Shao5] Yakov Shafranovich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. RFC 4180. Oct. 2005. DOI: 10.17487/RFC4180. URL: <https://rfc-editor.org/rfc/rfc4180.txt> (cit. on p. 76).
- [Sof20] SmartBear Software. *Swagger UI*. 2020. URL: <https://swagger.io/tools/swagger-ui/> (cit. on p. 15).
- [Sof21] SmartBear Software. *Authentication and Authorization*. 2021. URL: <https://swagger.io/docs/specification/authentication/> (visited on 04/05/2021) (cit. on p. 74).
- [Ste16] Ralph Steyer. *WordPress*. Springer Vieweg, 2016. ISBN: 978-3-658-12830-2 (cit. on pp. 16, 17, 78, 79).

Bibliography

- [Sto21] Sasa Stojanovic. *Understanding the WordPress File and Directory Structure*. 2021. URL: <https://qodeinteractive.com/magazine/wordpress-file-structure/> (visited on 04/05/2021) (cit. on pp. 66, 68).
- [Sur16] Vijay Surwase. "REST API Modeling Languages - A Developer's Perspective." In: *International Journal For Science Technology And Engineering* 2 (2016), pp. 634–637 (cit. on p. 15).
- [Tec20] primion Technology GmbH. *primion Technology GmbH*. 2020 (accessed November 19, 2020). URL: <https://www.primion.de/de/> (cit. on pp. 28–30).
- [TG16] Juris Tihomirovs and Janis Grabis. "Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics." In: *Information Technology and Management Science* 19 (Dec. 2016). DOI: 10.1515/itms-2016-0017 (cit. on p. 11).
- [Uza16] Sufyan bin Uzayr. *Learning WordPress REST API*. Packt Publishing, 2016. ISBN: 1786469243 (cit. on pp. 17, 18).
- [Wilo7] E Wilde. *Putting Things to REST*. 2007. URL: <https://escholarship.org/uc/item/1786t1dm> (cit. on pp. 4–6).
- [Wor21a] WordPress.org. *Database Description*. 2021. URL: https://codex.wordpress.org/Database_Description (visited on 04/05/2021) (cit. on pp. 68, 69).
- [Wor21b] WordPress.org. *Pages*. 2021. URL: <https://wordpress.org/support/article/pages/> (visited on 04/05/2021) (cit. on p. 67).
- [Wor21c] WordPress.org. *Plugin Basics*. 2021. URL: <https://developer.wordpress.org/plugins/plugin-basics/> (visited on 04/05/2021) (cit. on pp. 70, 71).
- [Wor21d] WordPress.org. *Wordpress Register Custom REST Endpoint*. 2021. URL: <https://developer.wordpress.org/rest-api/extending-the-rest-api/adding-custom-endpoints/> (visited on 04/05/2021) (cit. on p. 96).
- [Wor21e] WordPress.org. *Wordpress Register Script*. 2021. URL: https://developer.wordpress.org/reference/functions/wp_enqueue_script/ (visited on 04/05/2021) (cit. on p. 95).

Bibliography

- [Wor21f] WordPress.org. *Wordpress Register Shortcodes*. 2021. URL: https://developer.wordpress.org/reference/functions/add_shortcode/ (visited on 04/05/2021) (cit. on p. 95).
- [Wor21g] WordPress.org. *Wordpress Shortcodes*. 2021. URL: https://codex.wordpress.org/Shortcode_API (visited on 04/05/2021) (cit. on p. 95).
- [Wor21h] WordPress.org. *wpdb*. 2021. URL: <https://developer.wordpress.org/reference/classes/wpdb/> (visited on 04/05/2021) (cit. on p. 68).
- [WSR20] Thomas Wildbolz, Hans P. Schnöll, and Christian Ramsauer. "Managing Access to Space, Tools, and Machines at the Schumpeter Laboratory for Innovation." In: *IJAMM* (Mar. 8, 2020). <https://ijamm.pubpub.org/pub/wm62oq82>. URL: <https://ijamm.pubpub.org/pub/wm62oq82> (cit. on pp. 22, 27, 29, 33, 37).