

# **Digitaler Campus**

---

Audit 3 im Entwicklungsprojekt WS2223

## **Inhaltsverzeichnis**

---

1. Expose
2. Stakeholderanalyse
3. Domänenmodell
4. Zielhierarchien
5. Alleinstellungsmerkmale
6. Projektrisiken
7. Erfordernisse
8. Anforderungen
9. Anwendungsmodellierungen
10. POCs
11. Rapid Prototype
12. Ausblick
13. Evaluation
14. Projektplan

## ***Expose***

- Problemstellung
- Zielsetzung, Version
- Leitthema
- Relevanz

## Problemstellung

Der Campus Gummersbach der Technischen Hochschule Köln verfügt als solche über mehrere Gebäude und Räume, welche nicht immer eindeutig gekennzeichnet sind. Gerade Studenten im ersten Semester haben oft keinen Überblick welche Funktion ein Raum hat, welches Material zu Verfügung steht oder ob ein Raum bereits belegt ist.

Es existieren zwar Angebote, um Studenten die Orientierung am Campus zu erleichtern, diese sind aber meistens unzureichend oder nicht ausreichend bekannt. Im Hochschul-Planungs-System (HoPS) [1] der TH Köln sind Informationen nur oberflächlich zu finden. In der 26-Seitiger Erstsemester Broschüre [2] ist nur eine Seite dem Campus gewidmet, wo dieser nur grob kategorisiert ist.

[1] HoPS | HochschulPlanungsSystem der TH Köln. (n.d.). HoPS | HochschulPlanungsSystem der TH Köln. Retrieved October 31, 2022, from <https://fhpwww.gm.fh-koeln.de/hops/modules/roomequipment/room.php?room=0215>

[2] HoPS | HochschulPlanungsSystem der TH Köln. (n.d.). HoPS | HochschulPlanungsSystem der TH Köln. Retrieved October 31, 2022, from <https://fhpwww.gm.fh-koeln.de/hops/modules/roomequipment/room.php?room=0215>

## Iteration

- Problemstellung spezifiziert
- Fokus auf Raumverfügbarkeit und Raummaterial

### **Zielsetzung, Version**

Das Ziel ist es, eine Abbildung des Campus in digitaler Form darzustellen. Hierbei soll es Benutzern möglich sein, nach Räumen und Equipment zu filtern. Zum Equipment gehören z. B. die benötigte Software, Hardware oder die verfügbaren Sitzplätze. Die Raumverfügbarkeit soll auf Basis aktueller Stundenpläne und Benutzereingaben ermittelt werden.

Zusätzlich zur Filterfunktion können sich Nutzer auch direkt Informationen und Metadaten zu individuellen Räumen anzeigen lassen.

### **Iteration**

- Technisch unabhängigspezifiziert
- Fokus auf Raumverfügbarkeit und Raummaterial

## Leitthema

### Device Shifting

Bislang befinden sich kleine Tafeln mit der Raumnummer, der Bezeichnung sowie gegebenenfalls dem Stundenplan des Raumes neben der Tür.

Die Informationstafeln neben jedem Raum werden durch E-Ink Paper Displays ersetzt. Über sie werden begrenzt Rauminformationen präsentiert.

Mittels eines NFC-Tags am Display sollen Anwender auf die Hauptanwendung geleitet werden. Dort erhalten sie näheren Überblick zu dem jeweiligen Raum und erhalten die Möglichkeit, sich für den entsprechenden Raum einzutragen.

### Synchronisation

Sollte sich ein Anwender für den Raum eintragen, so wird der Status in der Anwendung und dem E-Ink Paper Display aktualisiert und man erhält Einblick, ob ein Raum aktuell belegt ist oder nicht.

## Iteration

- Synchronisation auf Zielsetzung angepasst
- Device Shifting hinzugefügt

## **Relevanz**

### **Gesellschaftliche Relevanz**

- Verbesserte Orientierung am Campus
- Ansehen der Hochschule wird gesteigert
- Zufriedenheit der Studierenden hebt sich
- Wege und Zeit können eingespart werden, da die Raumbelegungen direkt angezeigt werden können

## ***Stakeholderanalyse***



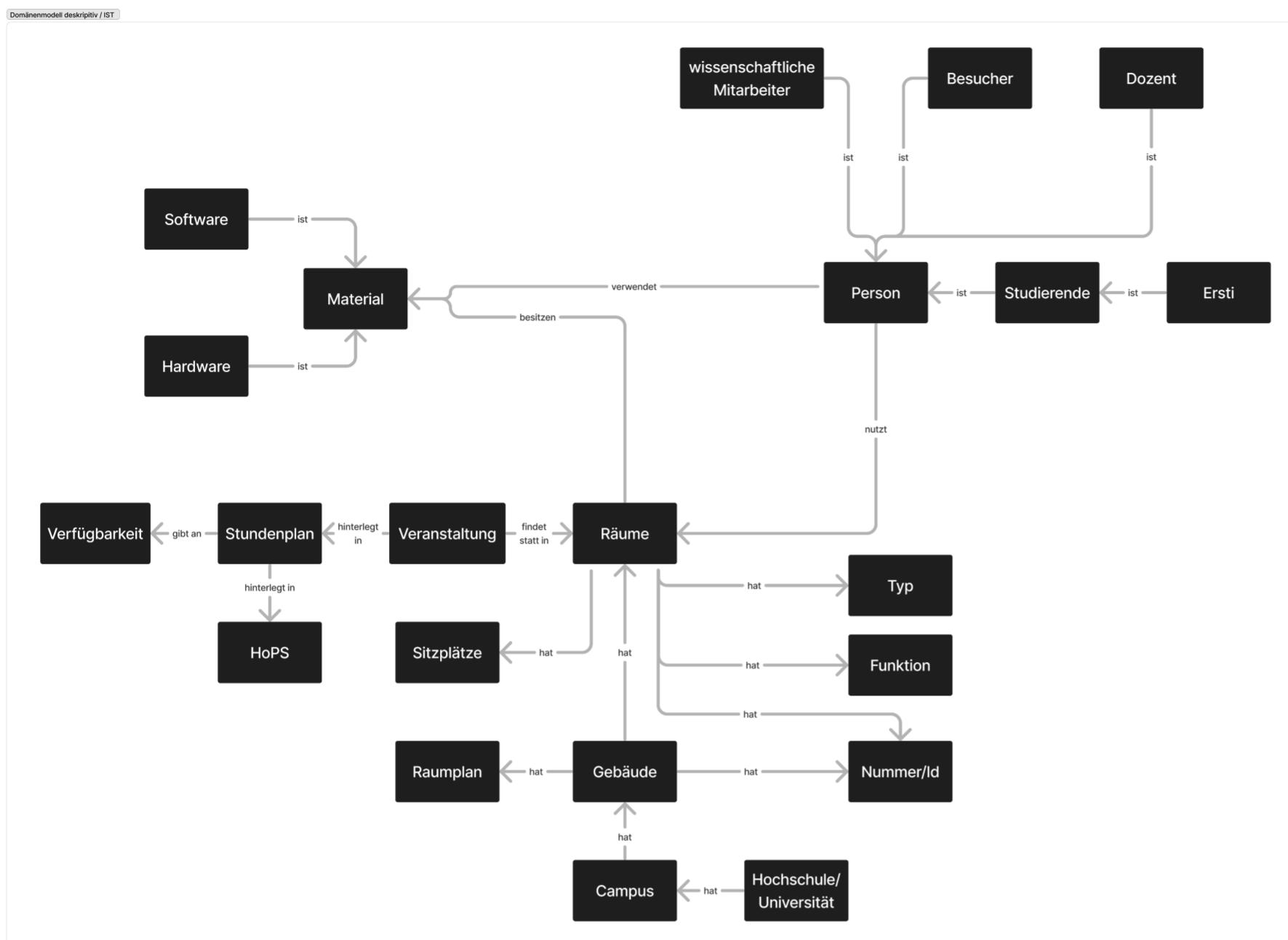
### Stakeholderanalyse

Bezeichnung		Bezug zum System	Objektbereich	Erfordernis / Erwartung	
Benutzer	<b>Studierende</b> <b>Wissenschaftliche Mitarbeiter</b> <b>Studentische Hilfskraft</b> <b>Pförtner</b>	Anteil	Nutzer des Systems	Als Benutzer muss man wissen das dieses System existiert, um es nutzen zu können.	
		Interesse	Nutzer des Systems	Als Benutzer muss man einen Überblick über die Räume der Hochschule haben, um effizient verfügbare Räume finden zu können.	
		Interesse	Nutzer des Systems	Als Benutzer muss man einen Überblick über die Räume der Hochschule haben, um effizient ein benötigtes Material finden zu können.	
		Anspruch	Nutzer des Systems	Als Benutzer müssen die Metadaten zuverlässig sein.	
		Anspruch	Nutzer des Systems	Als Benutzer müssen die Daten bezüglich der Raumverfügbarkeit zuverlässig sein.	
		Anspruch	Nutzer des Systems	Als Benutzer müssen die Daten bezüglich der Raumverfügbarkeit stets aktuell (synchronisiert) sein.	
		Anspruch	Nutzer des Systems	Als Benutzer muss die Interaktion zur Bestimmung der Raumverfügbarkeit angenehm, schnell und minimal sein, um die Anwendung ungestört nutzen zu können.	
		Dozent	Interesse	Als Dozent muss man einen Überblick über die Räume der Hochschule haben, um bei kurzfristigen Raumänderungen einen verfügbaren Raum finden zu können.	
		Interesse	Bereitstellung von Informationen	Als Hochschule muss man Informationen zu den Räumen haben, um diese bereitzustellen zu können.	
Hochschule		Anspruch	Förderer	Als Hochschule muss das System genutzt werden, um den Weiterbetrieb des Systems zu begründen	
Besucher		Interesse	Besucher des Campus	Als Besucher muss man einen Überblick über die Räume der Hochschule haben, um effizient einen gesuchten Raum finden zu können.	
primärer Stakeholder					
sekundärer Stakeholder					
tertiärer Stakeholder					

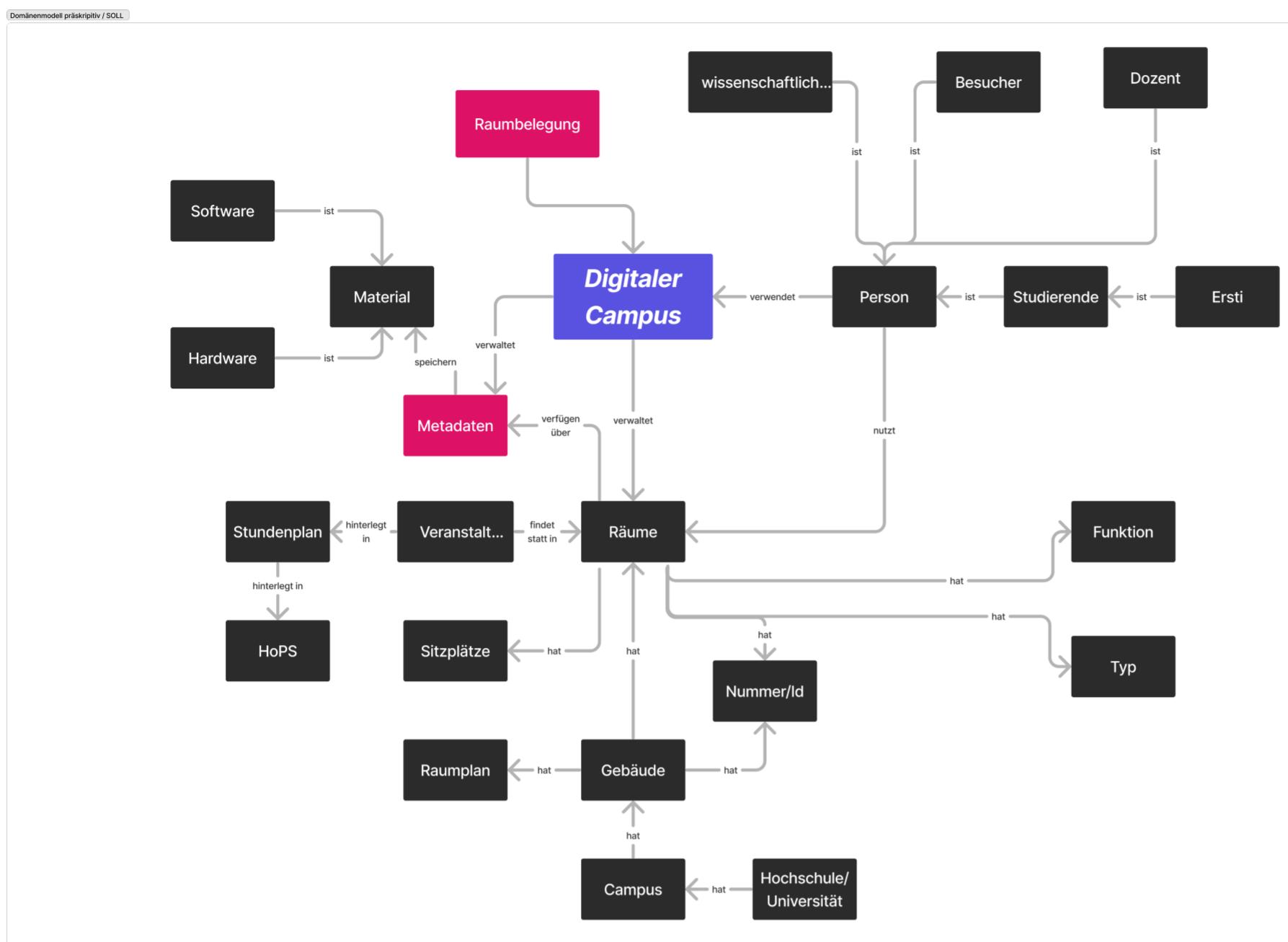
## ***Domänenmodell***

- Deskriptiv
- Präskriptiv

## Deskriptiv



## Präskriptiv



## **Zielhierarchien**

- Strategische Ziele
- Taktische Ziele
- Operative Ziele

## Zielhierarchien

---

### Strategische Ziele

Es soll eine interaktive Abbildung des Campus Gummersbach erstellt werden. Studenten und andere Stakeholder sollen eine bessere Einsicht über das vorhandene Material und die generelle Verfügbarkeit jeden Raumes bekommen.

### Taktische Ziele

Um einen besseren Durchblick in die Entwicklung der Modelle und des Systems zu erlangen, wird innerhalb des Teams ein Kanban-Board zur Planung und taktischen Verfolgung von Aufgaben erstellt.

### Operative Ziele

Kurzfristig gesehen müssen alle notwendigen Daten katalogisiert und etwaige Technologien bewertet und verglichen werden.

Die Digitale Abbildung des Campus Gummersbach und dessen Räume soll mit Dummy-Metadaten verknüpft werden. Außerdem müssen Informationen zu den verfügbaren Materialien innerhalb eines Raumes herausgesucht und gespeichert werden.

Die Raumdaten sollen in einer interaktiven Karte (2D oder 3D) für einen Nutzer einsehbar sein. Hierfür muss entschieden werden, wie die Karte innerhalb des Anwendung erfolgreich umgesetzt werden kann.

## **Alleinstellungsmerkmale**

- Physischer Raumplan
- HoPS Raumplan
- HoPS Ausstattung
- Alternative

## **Alleinstellungsmerkmale**

---

### **Physischer Raumplan**

- Größere Datentiefe (mehr Informationen können zu einem Raum dargestellt werden)
- Interaktivität nicht gegeben
- keine Informationen zur Raumverfügbarkeit vorhanden

### **HoPS Raumplan**

- Interaktivität nicht gegeben
- Übersichtlichkeit nicht gegeben
- keine aktuellen Informationen zur Raumverfügbarkeit vorhanden

### **HoPS Ausstattung**

- Interaktivität nicht gegeben
- Übersichtlichkeit nicht gegeben
- Ausstattung in HoPS nicht detailliert genug
- Keine intuitive Suche/Filterung nach Ausstattung in HoPS

**Alternative:** <https://www.mazemap.com/>

Vorteil gegenüber mazemap:

- Spezialisierung auf Campus-Umfeld
- Integration Raumverfügbarkeit
- Integration Rauminformationen (Equipment)

Ähnliches System muss für den Campus für die Einbindung von Gebäudeplänen erstellt werden: <https://davinci.stueber.de/floorplan.php>

## **Projektrisiken**

- Projektumfang
- Änderungsmanagement
- Kommunikation
- Ressourcen und Projektteam / Beschaffung
- System-Architektur
- Anforderungen
- Produkt-Design
- Technisches
- Integration
- Externe Risiken



# **Projektrisiken**

---

## **Projektumfang**

- Projektumfang ungenau definiert
- Schleichende Erweiterungen die den Projektumfang sprengen

## **Änderungsmanagement**

- Unnötige Refaktorisierung von Code
- Änderungen werden falsch priorisiert

## **Kommunikation**

- Terminabstimmung nicht immer möglich
- Zu viel Kommunikation
- Zu wenig Kommunikation

## **Ressourcen und Projektteam / Beschaffung**

- Erforderliche Daten sind nicht zugänglich
- Lernkurven zu hoch bzw. Erfahrung zu niedrig: neue Technologien erfordern zu viel Zeit zur Einarbeitung

## **System-Architektur**

- Gewünschte Architektur des Systems kann nicht wie geplant umgesetzt werden
- Architektur wird zu unflexibel entworfen

## **Anforderungen**

- Die Anforderungen wurden falsch/ungenau spezifiziert

## **Projektrisiken**

---

### **Produkt-Design**

- Das Design ist nicht intuitiv genug und wird nicht benutzt
- Die Darstellung ist nicht eindeutig. Anwender finden sich nicht zurecht
- Zu viele UI Komponenten
- Die Darstellung des Campus mit WebGL kann die Performance beeinträchtigen
- Anwendung ist entspricht nicht den W3C Accessibility Guidelines (WCAG)

### **Technisches**

- Messung der Raumverfügbarkeit ist zu ungenau (Untersuchung der jeweiligen Möglichkeiten zur Bestimmung der Verfügbarkeit ist erforderlich)
- Eingesetzte Technologien sind nicht für den Anwendungsfall geeignet
- Technologische Funktionen sind nicht skalierbar oder performant genug
- Informationssicherheit wird nicht genug beachtet
- Technologische Funktionen des Systems werden nicht dokumentiert
- Technologische Funktionen des Systems sind nicht oder nur schwer erweiterbar
- Zu viele Ressourcen/Skripte müssen geladen werden

### **Integration**

- Vorhandene Systeme (z.B. Transpondersystem) lassen sich nicht verwenden

### **Externe Risiken**

- Modulziele werden verfehlt
- Produkte zum Testen der PoCs sind nicht verfügbar
- Die Hochschule verbietet das Projekt

## ***Erfordernisse***



### Schablone

Als **spezifischer Benutzer** muss man **X wissen/verfügbar** haben, um **Y entscheiden/tun** zu können.

Die Stakeholder Studierende, wissenschaftliche Mitarbeiter, studentische Hilfskraft, Pförtner und Dozent werden gemäß der Stakeholderanalyse als spezifischer Benutzer “Benutzer des Systems” zusammengefasst:

## Erfordernisse

---

Als **Benutzer des Systems** muss man **das Wissen über die Existenz verfügbar** haben, um **das System nutzen** zu können.

Als **Benutzer des Systems** muss man **eine Plattform geboten** bekommen, um **effizient verfügbare Räume finden** zu können.

Als **Benutzer des Systems** muss man **eine Plattform geboten** bekommen, um **effizient benötigtes Material finden** zu können.

Als **Benutzer des Systems** muss man **die Möglichkeit zu filtern verfügbar** haben, um **das benötigte Material innerhalb eines Raumes finden** zu können.

Als **Benutzer des Systems** muss man **die Möglichkeit zur virtuellen Belegung eines Raumes verfügbar** haben, um **die Verfügbarkeit eines Raumes im System beeinflussen** zu können.

Als **Dozent** muss man **einen Überblick über die Räume der Hochschule** verfügbar haben, um **bei kurzfristigen Raumänderungen einen verfügbaren Raum finden** zu können.

## **Anforderungen**

- Funktionale Anforderungen
- Non-Funktionale Anforderungen

### Funktionale Anforderungen

- [F10] Das System **muss** einem Benutzer die Möglichkeit bieten, die Verfügbarkeit eines Raumes einsehen zu können.
- [F20] Das System **muss** einem Benutzer die Möglichkeit bieten, das verfügbare Material/das Equipment innerhalb eines Raumes einsehen zu können.
- [F30] Das System **muss** einem Benutzer die Möglichkeit bieten, einen Raum anhand von Filterkriterien (Material/Equipment, Verfügbarkeit) zu suchen.
- [F40] Das System **muss** fähig sein, einen Link zu einem bestimmten Raum über NFC zur Verfügung zu stellen.
- [F50] Das System **muss** fähig sein, die Raumverfügbarkeit anhand des aktuellen Stundenplans festzulegen.
- [F60] Das System **muss** einem Benutzer die Möglichkeit bieten, virtuell einen Raum zu belegen.
- [F70] Das System **muss** fähig sein, eine angemeldete Raumverfügbarkeit für einen Raum automatisch nach einem Zeitraum wieder freizugeben.
- [F80] Das System **muss** fähig sein, eine Darstellung für jedes Display vor einem Raum einzeln zu generieren.
- [F90] Das System **muss** fähig sein, verschiedene Räume in einem Gebäudeplan zu markieren.
- [F100] Das System **muss** fähig sein, Tokens für die Authentifizierung von Nutzerbelegungen zu generieren.
- [F110] Das System **muss** fähig sein, Tokens für die Authentifizierung von Nutzerbelegungen zu validieren.
- [F120] Das System **muss** fähig sein, eine API zur Bereitstellung von Rauminformationsdaten zur Verfügung zu stellen.
- [F130] Das System **muss** fähig sein, asynchrone Nachrichten via PubSub Architektur zu versenden.

## Anforderungen

---

- [F140] Das System **sollte** einem Benutzer die Möglichkeit bieten, sich per Button vor einem Raum anzumelden.
  - [F150] Das System **sollte** fähig sein, eine Fuzzy Search anhand von Benutzereingaben durchführen zu können.
- 
- [F160] Das System **kann** fähig sein, einen Benutzer per Push-Benachrichtigung zu einer Eingabe aufzufordern.
  - [F170] Das System **kann** fähig sein, einen Benutzer per E-Mail zu einer Eingabe aufzufordern.
  - [F180] Das System **kann** fähig sein, einen Benutzer über die Verfügbarkeit eines Raumes per E-Mail zu informieren.
  - [F190] Das System **kann** einem Benutzer die Möglichkeit bieten, sich für Updates zur Verfügbarkeit eines Raumes per Eingabe der E-Mail anzumelden.
  - [F200] Das System **kann** fähig sein, den aktuellen Winkel der Sonneneinstrahlung in die Berechnung der Relevanz eines Raumes bei einer Suche mit einzubeziehen.
  - [F210] Das System **kann** fähig sein, die dynamische Verfügbarkeit eines Raumes anhand von historischen Daten auszuwerten und somit Prognosen über die Verfügbarkeit in der Zukunft für verschiedene Wochentage zu treffen.

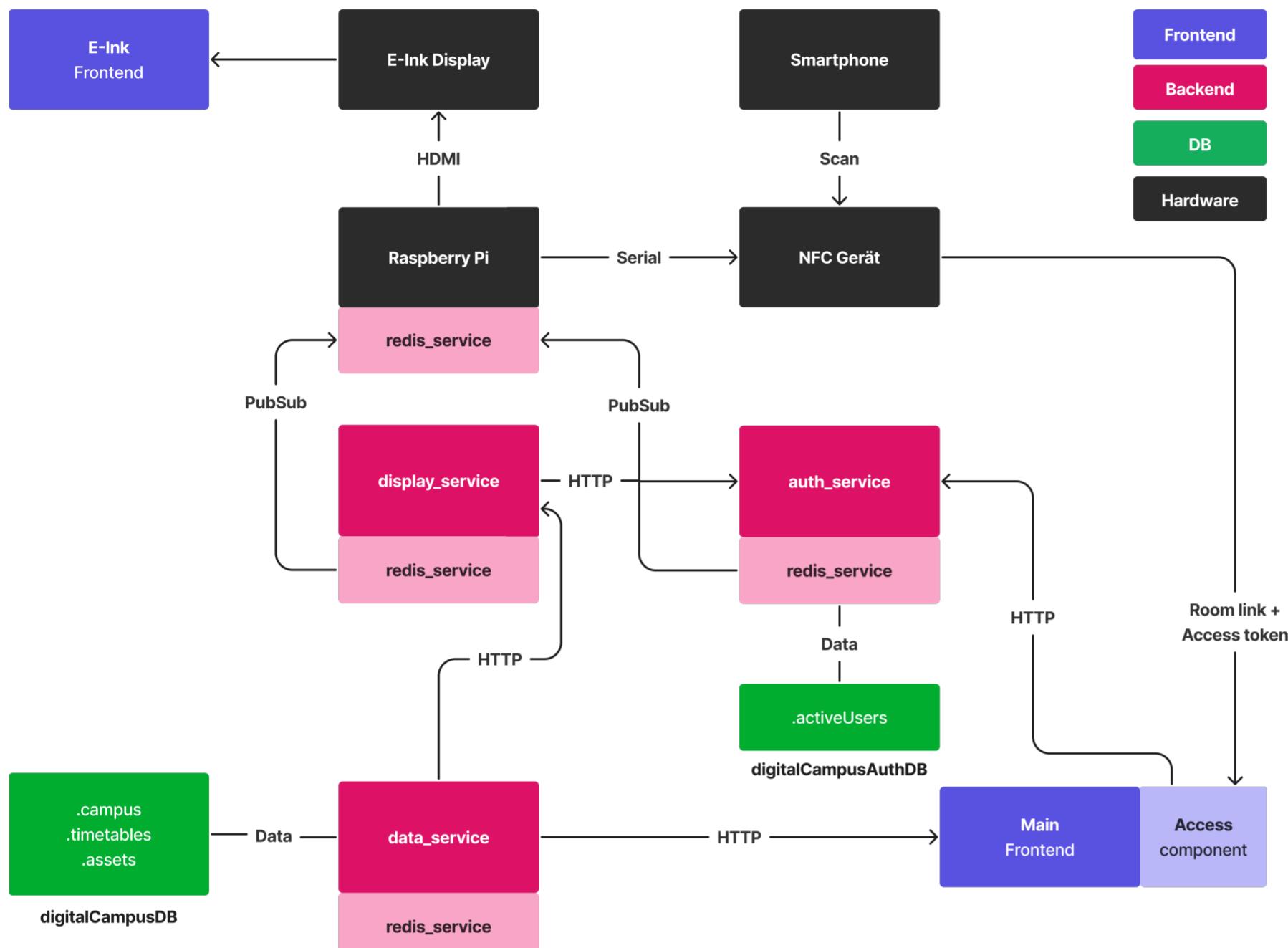
### Non-Funktionale Anforderungen

- [O10] Die Datenschutzverordnung **muss** eingehalten werden.
- [O20] Die Korrektheit der Daten **muss** immer gewährleistet werden.
- [Q10] Die Suche nach Rauminformationen und Raumverfügbarkeiten **muss** so gestaltet sein, dass eine einfache und intuitive Bedienung möglich ist.
- [Q20] Die Oberfläche des Systems **sollte** so gestaltet sein, dass eine einfache und intuitive Bedienung von jedem möglich ist.

## **Anwendungsmodellierungen**

- Architekturmodell
- PubSub
- API Json Schema
- Ablaufdiagramm
- Use Case Diagramm
- Kommunikationsdiagramm

# Architekturmodell



28

## Kommentar

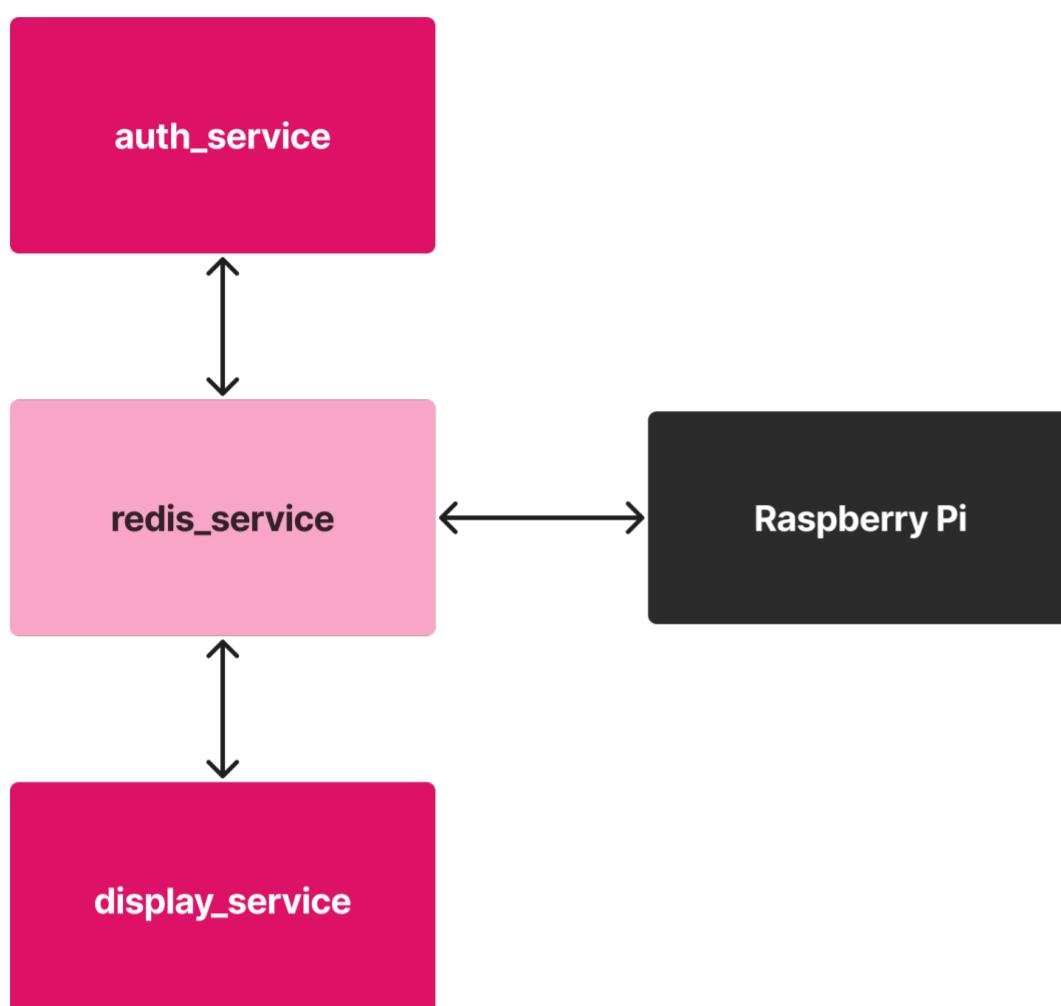
Das System wird in **vier verschiedene Bereiche** unterteilt.

1. Das **Main Frontend** agiert als zentrale Schnittstelle für Nutzerinteraktion und bietet eine Darstellung des Campus mit Raumbelegungen und weiteren Rauminformationen an.
2. Die in **Microservices** aufgeteilten Backend-Systeme stehen als Schnittstellen zur Datenbereitstellung, zur Generierung von Displayseiten der E-Ink Displays und zur Authentifizierung von Tokens zur Verfügung.
3. Die in der **MongoDB Atlas Cloud** aufgesetzten Datenbanken speichern zum einen die Raumdaten und zum anderen separat die zur Authentifizierung verwendeten Tokens.
4. Zur Hardware können die **Raspberry Pis** und die damit verbundenen **E-Ink Displays** und die **NFC-Geräte** gezählt werden.

Die Kommunikation innerhalb des System erfolgt dabei entweder über gängige **HTTP REST-Schnittstellen** oder intern über einen **asynchronen PubSub-Broker**.

Der Ablauf des System wurde im folgenden durch Sequenzdiagramme detaillierter dargestellt.

### Publish-Subscribe Architektur



### Kommentar

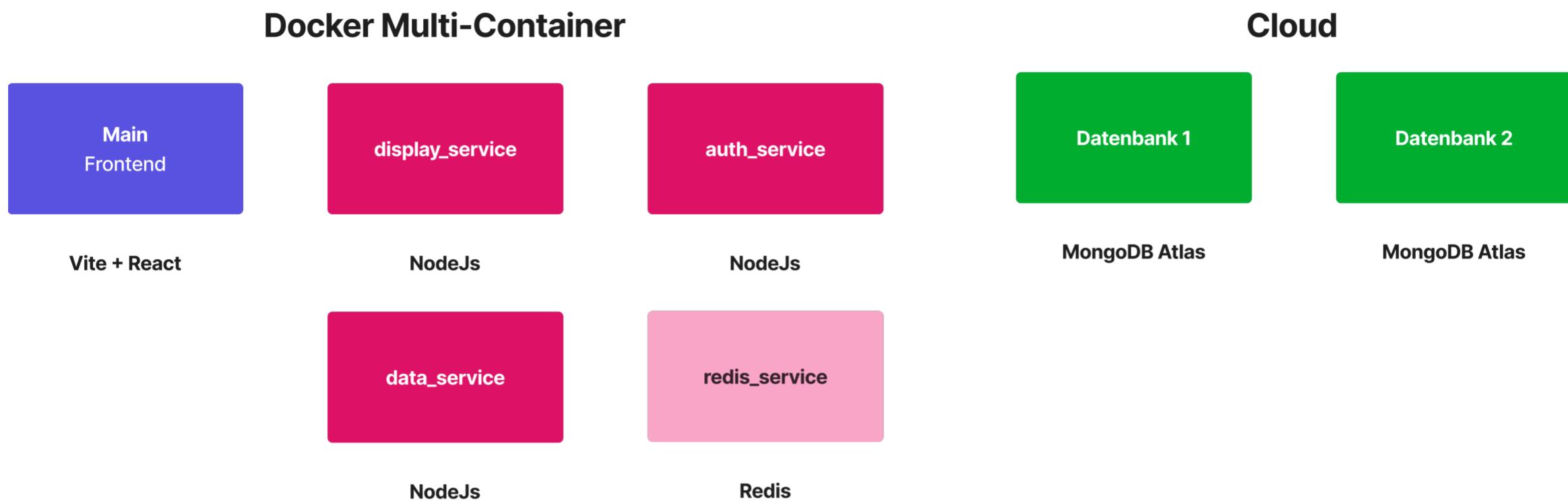
Innerhalb der Architektur wird ein **Redis Publish-Subscribe Server** zur asynchronen Kommunikation zwischen den Microservices und dem Raspberry Pi verwendet.

Über diesen werden Nachrichten innerhalb eines gemeinsamen Kanals übertragen und empfangen. Hervorzuheben ist hierbei, dass jedes System, welches dem Nachrichtenkanal beigetreten ist, die Nachrichten empfangen und für sich entscheiden kann, ob Funktionen ausgeführt werden sollen.

Der **display\_service** kann dabei "update" Events an den Raspberry Pi versenden, damit dieser die auf dem E-Ink Display angezeigte Seite neu lädt.

Der **auth\_service** sendet einen neuen Token an den Raspberry Pi, wenn das NFC-Gerät gescannt und ein Token von einem Nutzer verwendet wurde.

### Infrastruktur



30

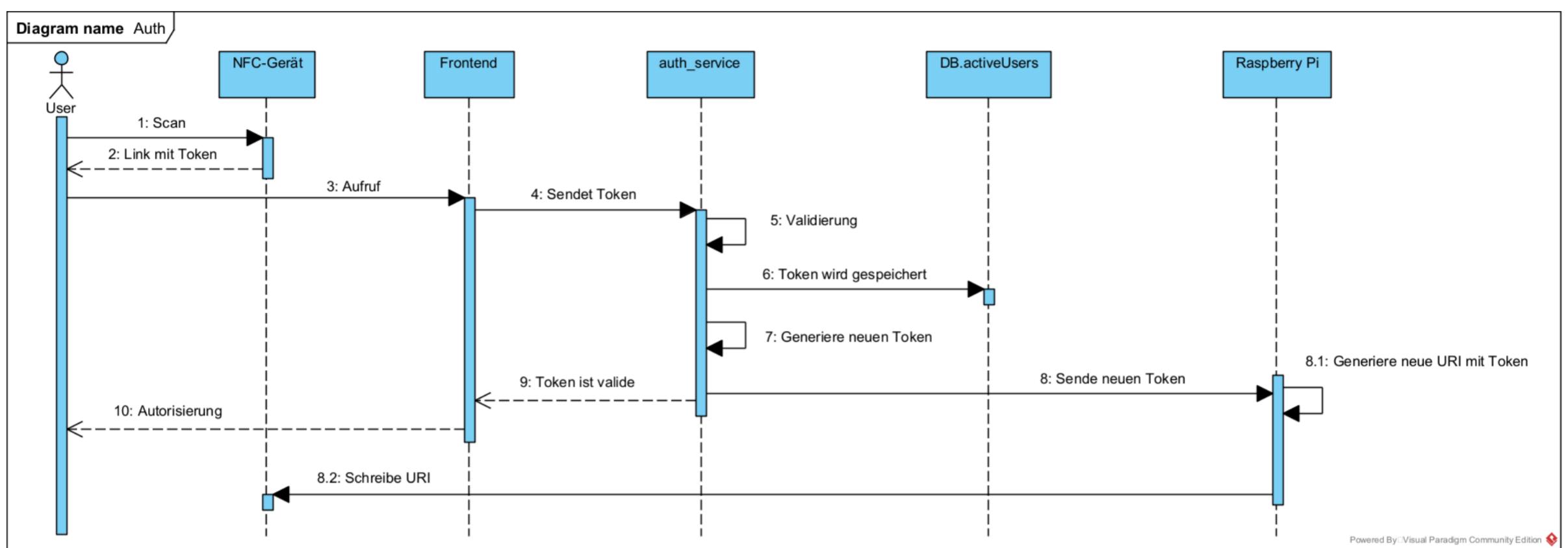
### Kommentar

Die Infrastruktur des Systems wird in zwei Bereiche unterteilt:

Auf der einen Seite werden die Microservices und das Frontend über Docker Multi-Container und **docker compose**, vor allem zur vereinfachten Cross-Plattform Entwicklung, aufgesetzt.

Die Datenbasis wurde über **MongoDB Atlas** in einem Cloud-Cluster integriert und kann dadurch über die Docker Container hinaus gewartet werden. Da keine sensiblen bzw. personenbezogenen Daten gespeichert werden, besteht keine Gefahr den **Datenschutz** zu verletzen. Bei einer Weiterentwicklung muss hier natürlich im Verlauf weiter genauer evaluiert werden.

### Sequenzdiagramm - Aufruf via NFC



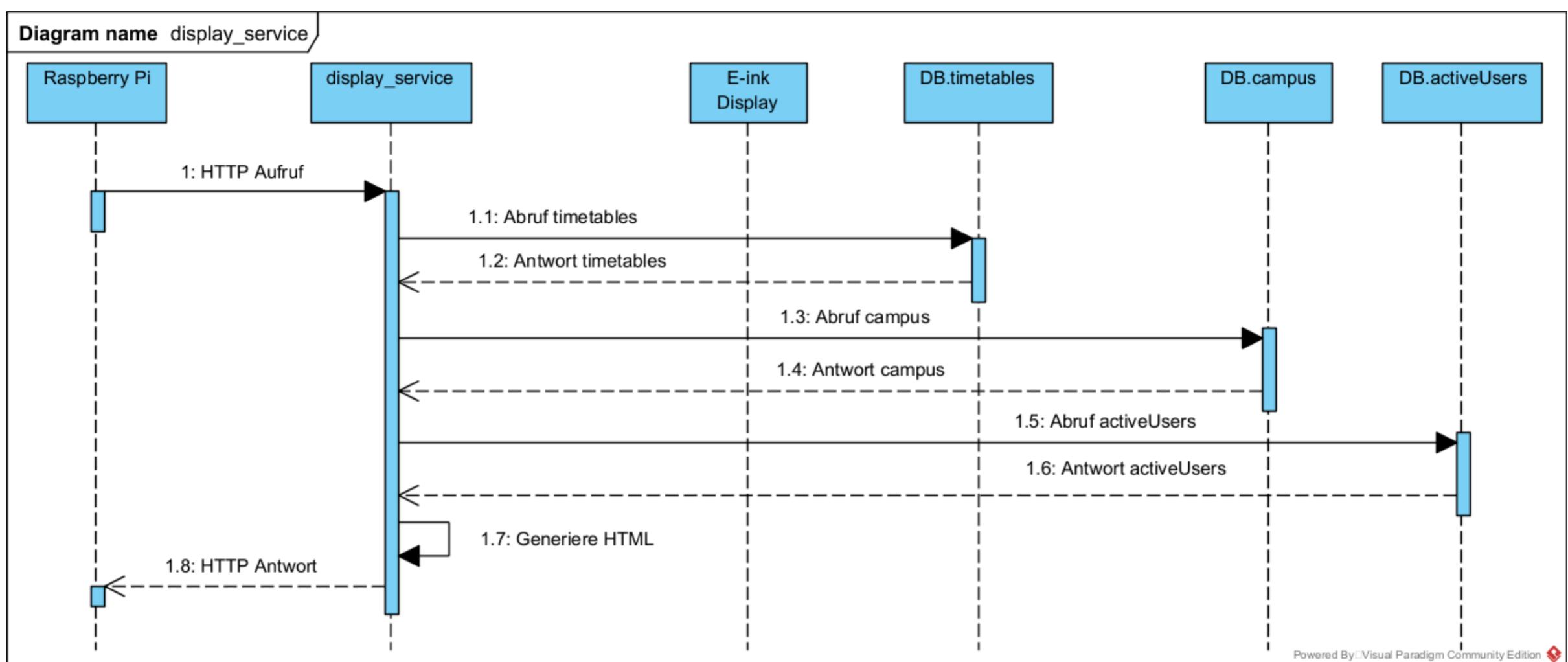
### Kommentar

1. Scan: Ein Benutzer scannt den NFC-tag.
2. Link mit Token: Ein Link zur Raumbelegungskomponente des Frontends inklusive Token wird an das Endgerät des Benutzer gesendet.
3. Aufruf: Der Benutzer ruft das Frontend auf dem Endgerät auf.
4. Sendet Token: Das Frontend sendet via HTTP-POST den Token an den auth\_service.
5. Validierung: Der auth\_service validiert das Token.

#### Bei validen Token

6. Token wird gespeichert: Das Token wird in der DB.activeUsers Datenbank gespeichert.
7. Generiere neuen Token: Generiert eine neues Token.
8. Sende neuen Token: Sendet neu generiertes Token an den Raspberry Pi
  - 8.1 Generiere neue Uri mit Token: Generiert neuen Link für den NFC-tag.
  - 8.2 Schreibe URI: Schreib den neu generierten Link auf den NFC-tag.
9. Token ist Valide: Das Frontend erhält eine HTTP Antwort mit code 200.
10. Autorisierung: Die Raumbelegungskomponente des Frontends wird auf dem Endgerät des Benutzers freigeschaltet.

### Sequenzdiagramm - Aufruf via Pi



### Kommentar

1. HTTP Aufruf: Der Raspberry Pi ruft die URL für das eigene E-Ink Display vom display\_service auf.
- 1.1 Abruf timetables: Der display\_service ruft die erforderlichen Stundenplandaten ab.
- 1.2 Antwort timetables: Die Daten aus der Datenbank können zwischengespeichert werden.
- 1.3 Abruf campus: Der display\_service ruft die erforderlichen Campus/Rauminformationen ab.
- 1.4 Antwort campus: Die Daten aus der Datenbank können zwischengespeichert werden.
- 1.5 Abruf activeUsers: Der display\_service ruft die erforderlichen aktiven Raumbelegungen ab.
- 1.6 Antwort activeUsers: Die Daten aus der Datenbank können zwischengespeichert werden.
- 1.7 Generiere HTML: Der display\_service generiert mithilfe der serverseitigen Templatingsprache EJS eine für den Raum angepasste HTML Seite.
- 1.8 HTTP Antwort: Der Raspberry Pi erhält die Webseite als HTTP Response und kann diese auf dem E-Ink Display anzeigen.

## **Durchgeführte POCs**

Frontend

- Digitale Darstellung Campus Gummersbach

Backend

- Seitengenerierung E-Ink Display

Hardware

- Automatisierter Seitenaufruf Raspberry Pi
- Integration ins Netzwerk

## **Durchzuführende POCs**

- Progammierbarer NFC Tag



# *Digitale Darstellung Campus Gummersbach*

Frontend

### **Beschreibung**

Es soll geprüft werden welche Darstellung sich am Besten für den Campus Gummersbach eignet.

- 2D Für die Darstellung soll eine Vektor-Grafik des Campus Gummersbach (.svg) verwendet werden.
- 3D Für die Darstellung soll ein 3D-Modell (.glTF) des Campus Gummersbach verwendet werden.
- 2.5D Für die Darstellung soll eine generische Lösung implementiert werden. Gebäude, Etagen und Räume des Campus Gummersbach werden als JSON oder XML Elementen in einem Schema definiert. Anhand von Koordinaten sollen digitale Abbildungen dieser generiert werden. Eine Skelettdarstellung ist ausreichend.

## **Exit-Kriterien**

### **2D**

1. Die Vektor-Grafik lässt sich auf einer Website darstellen.
2. Die Vektor-Grafik lässt sich interaktiv bedienen. (e.g.  
Räume werden visuell hervorgehoben)

### **3D**

1. Ein 3D-Modell wird von einem anderen Team zur Verfügung gestellt werden.
2. Das 3D-Modell entspricht unseren Anforderungen.
3. Das 3D-Modell lässt sich auf einer Website darstellen
4. Das 3D-Modell lässt sich interaktiv bedienen. (e.g. Räume werden visuell hervorgehoben)

### **2.5D**

1. Die Daten die für die Darstellung relevant sind, können in einem Schema beschrieben werden.
2. Überführung der Daten in XML oder JSON.
3. Eine Skelletdarstellung des Campus lässt sich aus den Daten generieren.
4. Die Skelletdarstellung lässt sich interaktiv bedienen. (e.g.  
Räume werden visuell hervorgehoben)

## **Fail-Kriterien**

### **2D**

- Die Vektor-Grafik entspricht nicht unseren Anforderungen.
- Die Vektor-Grafik lässt sich nicht einbinden.
- Die Vektor-Grafik lässt sich nicht interaktiv bedienen.

### **3D**

- Ein 3D-Modell kann nicht zur Verfügung gestellt werden.
- Das erhaltene 3D-Modell entspricht nicht unseren Anforderungen.
- Die Erstellung eines 3D-Modell ist zu Aufwendig.
- Die eigene Erstellung des Modells ist zu Aufwendig.

### **2.5D**

- Relevante Daten, um den Campus Gummersbach darzustellen, lassen sich nicht durch ein Schema beschreiben.
- Der Campus Gummersbach lässt sich durch ein Schema beschreiben. Die Pflege der Daten ist allerdings zu aufwendig.

## **Fallbacks**

Die Darstellungsmöglichkeiten 2.5D, 2D und 3D werden in dieser Reihenfolge durchgeführt und dienen jeweils als Fallback für den Vorherigen.

## Durchführung

### 2.5D

1. Der Maßstab wird durch den Gebäudeplan ermittelt.
2. Eine vereinfachte Darstellung des Campus Gummersbach wird als Vektor-Grafik erstellt.
3. **[FAIL]** Eine Skelletdarstellung lässt sich nicht durch XML realisieren.  
Daten können nicht der Vektor-Grafik entnommen werden.
4. Aus der Vektor-Grafik wird ein JSON-Objekt erzeugt. JSON Schema wird erstellt.
5. **[EXIT]** Eine Skelletdarstellung lässt sich durch das JSON realisieren.

### Verwendete Packages

three  
react-three/fiber  
react-three/drei

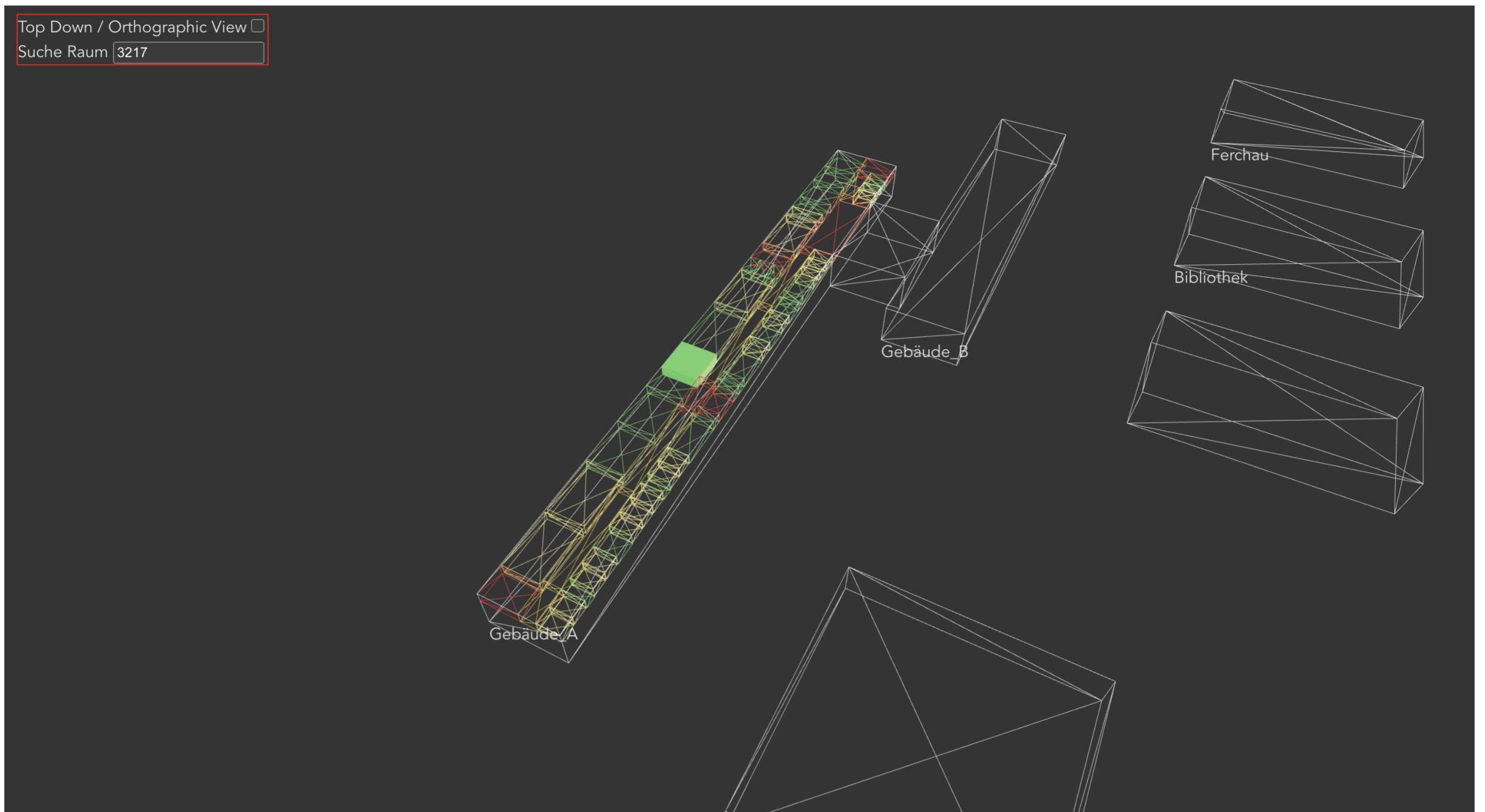
### 2D

1. **[EXIT]** Eine 2D Darstellung lässt sich durch die OrthographicCamera in react-three/drei simulieren.

### 3D

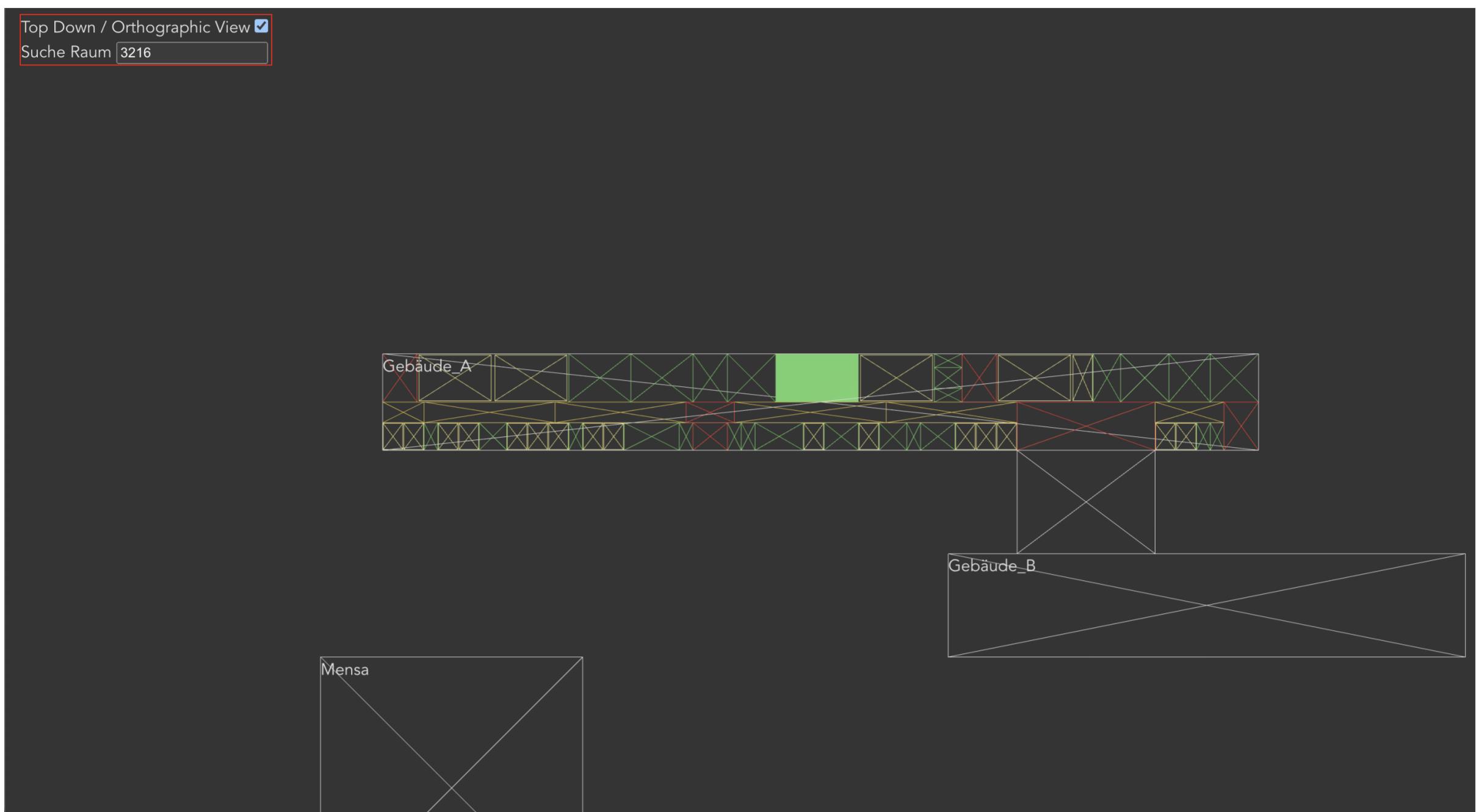
1. Modellierung einer reduzierten Darstellung eines Raumes. (Export als .gltf)
2. **[EXIT]** Modell erfolgreich in Website importiert.

## Digitale Darstellung Campus Gummersbach

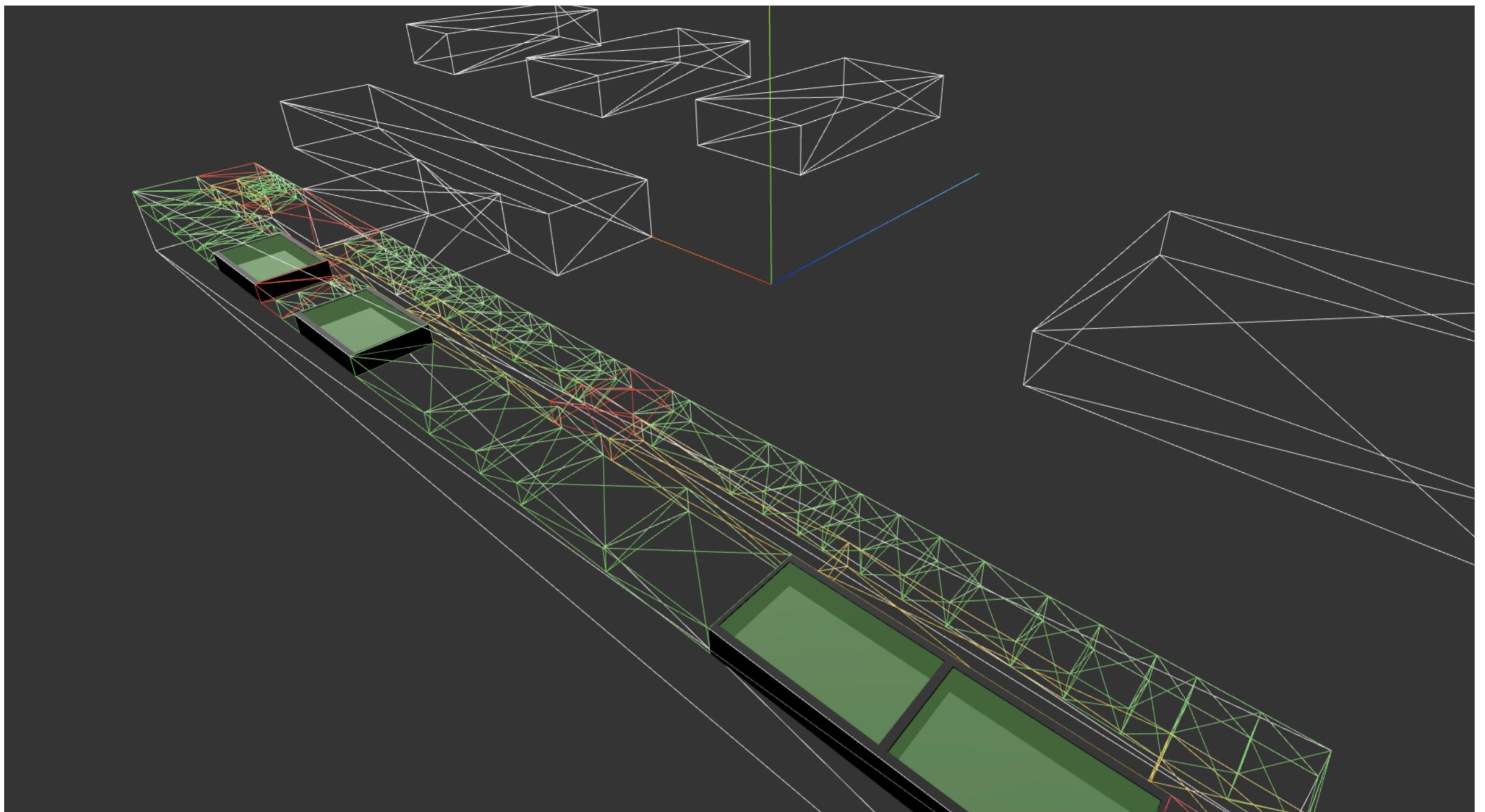


2.5D - Eine Skelletdarstellung des Campus lässt sich in die Szene integrieren.

# Digitale Darstellung Campus Gummersbach



2D - Es lässt sich die Orthografische Ansicht wechseln.



3D - Es lassen sich 3D Modelle nachträglich in die Szene einfügen.

## ***Seitengenerierung E-Ink Display***

Backend



### **Beschreibung**

Ein E-Ink Display soll vor einem Raum platziert werden, Daten über einen PubSub-Channel aus dem Backend erhalten und diese verarbeiten.

Der Fokus liegt hierbei auf dem Backend welches dynamisch eine einfache, an ein E-Ink Display angepasste, Webseite generiert und diese bereitstellt.

### Exit-Kriterien

1. Das Backend kann aktuelle Daten über den Raumzustand empfangen.
2. Eine Seite kann dynamisch für einen Raum generiert werden.
3. Bei Aktualisierungen zur z.B. Raumbelegung kann das Backend dies verarbeiten und eine neue Seite bereitstellen. (Relevanz für den funktionalen Prototypen)

### Fail-Kriterien

- Die Datenbasis gibt nicht die erforderlichen Informationen zu einem Raum her.
- Die Seite ist nicht auf die Darstellung auf einem E-Ink Display angepasst.

### **Fallbacks**

Ein Display sollte selber Informationen darstellen, wenn z.B. die generierte Seite nicht erreicht werden konnte.

## Durchführung

Im ersten Schritt der Durchführung wurde die Darstellung auf einem E-Ink / e-Paper Display recherchiert. Hervorzuheben sind hierbei die **Limitierungen der Technologie** und die damit verbundenen Limitierungen in der Erstellung des Designs:

- **Kontrast** ist wichtig (Begrenzte Farbwiedergabe)
- **Keine Animationen** (Ghosting und Smearing sollen vermieden werden)
- **Fokus auf HTML** und kein Nachladen von Javascript (Lightweight)

### Entwicklung Wireframe



[1] <https://gomakethings.com/building-websites-that-work-on-an-e-ink-kindle/>

[2] [https://www.reddit.com/r/eink/comments/lkc0ea/tip\\_to\\_make\\_web\\_browsing\\_much\\_better\\_on\\_eink/](https://www.reddit.com/r/eink/comments/lkc0ea/tip_to_make_web_browsing_much_better_on_eink/)

## Durchführung

Für die Generierung der Seite wird im Backend, genauer auf dem display\_service, auf die **Templating Sprache EJS** gesetzt. Mit dieser kann die Seite **serverseitig gerendert** werden und der Client erhält die passende HTML-Datei.

- Die Seite kann **erfolgreich** auf dem Backend **generiert** werden. Die Exit Kriterien wurden erfüllt.
- Die **Darstellung** wurde auf ein E-Ink Display **angepasst**, konnte jedoch aufgrund der Verfügbarkeit noch nicht auf einem Display getestet werden.
- Die **dynamische Aktualisierung** von Seiten wird für den funktionalen Prototypen implementiert, wenn auch die erforderlichen Services für Audit 4 verfügbar sind.

### Stundenplan Templating

```
● ● ●
1 <table>
2   <thead>
3     <tr>
4       <th class="time-col">Uhrzeit</th>
5       <th>Montag</th>
6       <th>Dienstag</th>
7       <th>Mittwoch</th>
8       <th>Donnerstag</th>
9       <th>Freitag</th>
10      </tr>
11    </thead>
12    <tbody>
13    <% for (let i = 8; i < 21; i++) { %>
14      <% let hour = i.toString().padStart(2, "0") %>
15      <tr>
16        <td class="time-col"><%= hour %>:00</td>
17        <td><%= monday.find(t => t.time === `${hour}:00`)? .name || " " %></td>
18        <td><%= tuesday.find(t => t.time === `${hour}:00`)? .name || " " %></td>
19        <td><%= wednesday.find(t => t.time === `${hour}:00`)? .name || " " %></td>
20        <td><%= thursday.find(t => t.time === `${hour}:00`)? .name || " " %></td>
21        <td><%= friday.find(t => t.time === `${hour}:00`)? .name || " " %></td>
22      </tr>
23    <% } %>
24  </tbody>
25 </table>
```

## ***Automatisierter Seitenaufruf Raspberry Pi***

Hardware



### **Beschreibung**

Der Raspberry Pi 3 (Rpi3) soll eine Website auf einem E-ink Display aufrufen. Der Aufruf soll nach Erhalten eines Events in nodejs geschehen, d. h. der Aufruf muss innerhalb von nodejs stattfinden.

### Exit-Kriterien

- Die Seite wird aufgerufen und auf dem Display angezeigt.

### Fail-Kriterien

- Der Aufruf der Seite funktioniert nicht
- Die Seite wird aufgerufen, aber nicht auf dem Display gezeigt
- Das Programm funktioniert, nur wenn es lokal aufgerufen wird, nicht via ssh

### Fallbacks

- Falls keine Möglichkeit besteht, die Seite via nodejs zu öffnen, muss ein shell Skript geschrieben werden, welches diese Aufgabe übernimmt.
- Das Programm muss mittels angeschlossener Tastatur auf dem pi gestartet werden

### Durchführung

Eine Library wurde gefunden, welche die benötigte Funktionalität anbietet: [open-npm](#)

1. Der Rpi3 wurde via HDMI mit einem Monitor verbunden (Wir warten auf die Lieferung eines E-ink Displays)
2. Es wurde eine Verbindung zu dem Rpi3 via ssh aufgebaut.
3. Eine simple Integration der library wurde vorgenommen und getestet.
4. **[FAIL]** Kriterium 1 tritt ein, die Seite wird nicht geöffnet. Die console.log Statements werden allerdings ausgeführt.
5. Derselbe Code wurde auf lokal auf der Maschine ausgeführt und funktioniert vollständig (inklusive Aufrufen der Website)
6. **[EXIT]** Eine Tastatur wurde an den Rpi3 angeschlossen, um die Datei lokal ohne ssh auszuführen. Dies ist erfolgreich und das Kriterium tritt ein.

```
const open = require('open');
const domain = 'https://www.th-koeln.de/';
(async function () {
  console.log("pi code in progress");
  await open(domain);
  console.log("pi code done");
})();
```

Es wird angenommen, dass die Ausführung via ssh nicht funktioniert wegen fehlender X11 forwarding Konfiguration bei der ssh Verbindung. Der Fallback, das Programm mittel Tastatur auszuführen, ist nach aktuellem Stand ausreichend. Das Einrichten der Funktionalität zur Ausführung via ssh wurde in den Backlog zu Audit 4 verschoben.

## *Integration ins Netzwerk*

Hardware

### **Beschreibung**

Ein Raspberry Pi 3 (Rpir3) soll in das System integriert werden und über ein lokales Netzwerk von den anderen Services ansprechbar sein.

### **Exit-Kriterien**

- Der Rpi3 kann in einer lokal laufenden nodejs Umgebung Nachrichten empfangen, welche über den redis-PubSub Messenger versendet werden und selber Nachrichten verschicken.

### **Fail-Kriterien**

- Der Rpi3 lässt sich nicht integrieren

### **Fallbacks**

Teile des Systems müssen deployed werden, damit der Rpi3 über das Internet Zugriff hat.

### **Durchführung**

1. Das aktuelle Standard Raspberry Pi OS wurde auf dem Rpi3 installiert.
2. Nodejs wurde auf dem Rpi3 installiert
3. "Hello World" Programm um node zu testen
4. Einfügen der "redisPubSub.js" + Konfiguration um Verbindung zu Docker Netzwerk herzustellen.
5. **[EXIT]** Ausführen der redisPubSub.js um festzustellen ob die Verbindung klappt.

## *Programmierbarer NFC Tag*

Hardware



### **Beschreibung**

Der Raspberry Pi soll ein einen Nachricht (Link + Token) an ein NFC board senden, damit Geräte mit NFC-lese Funktion diese Nachricht via Scannen des NFC-Boards erhalten können.

### **Exit-Kriterien**

- Die Nachricht wird auf das NFC board übertragen.
- Ein NFC-lese Gerät scannt das Board und liest die übertragene Nachricht.

### **Fail-Kriterien**

- Die Übertragung der Nachricht an das NFC Board funktioniert nicht.
- Ein NFC-lese Gerät scannt das Board und bekommt eine andere Nachricht.

## **Fallbacks**

- Ein QR-code mit der selben Nachricht wird angeboten.
- Eine URI wird auf dem E-Ink Display angezeigt.

### Durchführung

Für die Durchführung fehlt eine Lieferung aus dem moxdLab.

Es fand bereits eine Auseinandersetzung mit der Dokumentation des georderten Raspberry Pi NFC-HAT statt, sodass die Durchführung nicht viel Zeit in Anspruch nehmen sollte.

#### Geplante Durchführung:

1. Installieren der NFC-HAT Hardware auf dem Pi.
2. Installieren der notwendigen Software auf dem Pi.
3. Option 1: Schreiben eines shell-Skripts zur Übertragung der Nachricht und Ausführung via NodeJs.
4. Option 2: Integration einer npm Library, welche dieselbe Funktion übernimmt.

## **Ausblick - Was steht noch an?**

- Hardware
- Backend
- Frontend
- Datenbank

## **Hardware**

- Integration NFC-Gerät
- Integration E-Ink Display

## **Backend**

- Auth\_Service als Microservice zur Generierung und Authentifizierung von Tokens.
- Verwaltung aktiver Belegungen von Räumen und Weitergabe durch API.
- Implementierung der PubSub Events.

## **Frontend**

### **Design**

- Wireframe
- Screendesign

### **Umsetzung**

- <Layout />
- <Scene />
- <Raumbelegung />

## **Datenbank**

- Datenbank für aktive Nutzer muss eingepflegt werden.
- Räume müssen ergänzt werden.

## **Evaluationen - Was könnte man nochmal Iterieren?**

- Hardware NFC/ E-Ink
- Auth Service
- Wireframe / Screendesign
- Frontend, Render, Layout

## **Evaluationen - Was könnte man nochmal Iterieren?**

---

### Hardware

- Integration NFC-Gerät
- Integration E-Ink Display

### Frontend

- Darstellung des Campus Gummersbach, Design

## ***Projektplan***

- Audit 3
- Audit 4



# Projektplan

---

## Audit 3

▼ Audit 3 12 Dec 12, 2022 – Jan 12

#	Description	Assignee	Status	Start Date	End Date	Duration	Progress	Comments
19	Kernelfragen #42	antonztsv, calvinhnzr, and foseph	Audit 3			1.5	1.5	Ready
20	Fachperspektiven-Spezifische Leitfragen #43	antonztsv, calvinhnzr, and foseph	Audit 3			1.5	1.2	Ready
21	Digitale Darstellung Campus Gummersbach #27	calvinhnzr	Audit 3			20	16	Done
22	Erfordernisse #64	antonztsv	Audit 3			1	1	Done
23	Anforderungen #65	antonztsv	Audit 3			1	1	Done
24	Überarbeitung Modellierung Architektur #86	antonztsv	Audit 3			1	4	Done
25	Raumverfügbarkeit - E-Ink Display Seitengenerierung #30	antonztsv	Audit 3			3.5	4.5	Done
26	Modellierungen iterieren #58	antonztsv, calvinhnzr, and foseph	Audit 3			2	3	Done
27	Deliverables für den 4. Audit (Projektplan) #60	antonztsv, calvinhnzr, and foseph	Audit 3			0.5	0.5	Done
28	Modellierung der Anwendungslogik #59	antonztsv and foseph	Audit 3			1	1.5	Done
29	Integration eines Raspberry Pi in das Netzwerk #106	foseph	Audit 3			2	0.5	Done
30	Automatisierter Seitenaufruf durch Raspberry Pi #109	foseph	Audit 3			2	2	Done

72

## Kommentar

Wie im Projektplan zu Audit 3 zu sehen, wurden die zuvor spezifizierten Proof of Concepts erfolgreich durchgeführt und bereits in einem abgekapselten Umfang implementiert. Zuvor fehlende und bereits existierende Modellierungen wurden erstellt bzw. iteriert.

Die Architektur sowie die generelle Anwendungslogik konnten konkretisiert und ebenfalls in Modellierungen festgehalten werden.

## Audit 4

Audit 4 11 Jan 16 - Feb 23								
31	⌚ Raumverfügbarkeit - Programmierbarer NFC Tag #62	 floseph	<input type="button" value="Audit 4"/>	1	1	 Backlog	<input type="button" value="proof of concept"/>	
32	⌚ Integration des NFC board an den Raspberry Pi #108	 floseph	<input type="button" value="Audit 4"/>	1		 Backlog	<input type="button" value="proof of concept"/>	
33	⌚ Integration E-Ink Display #112		<input type="button" value="Audit 4"/>	1		 Backlog	<input type="button" value="audit 4"/>	
34	⌚ Implementierung auth_service #113		<input type="button" value="Audit 4"/>	3		 Backlog	<input type="button" value="backend"/>	<input type="button" value="feature"/>
35	⌚ Verwaltung aktiver Belegungen von Räumen und Weitergabe durch API #114		<input type="button" value="Audit 4"/>	3		 Backlog	<input type="button" value="backend"/>	<input type="button" value="feature"/>
36	⌚ Implementierung der PubSub Events #115		<input type="button" value="Audit 4"/>	2		 Backlog	<input type="button" value="backend"/>	<input type="button" value="feature"/>
37	⌚ Wireframe und Screendesign #116		<input type="button" value="Audit 4"/>	12		 Backlog	<input type="button" value="frontend"/>	<input type="button" value="task"/>
38	⌚ Implementierung Komponente Raumbelegung #117		<input type="button" value="Audit 4"/>	1		 Backlog	<input type="button" value="feature"/>	<input type="button" value="frontend"/>
39	⌚ DB Collection activeUsers #118		<input type="button" value="Audit 4"/>	0.5		 Backlog	<input type="button" value="database"/>	<input type="button" value="task"/>
40	⌚ Erstellung Poster #119	 antonztsv, calvinhnzr, and floseph	<input type="button" value="Audit 4"/>	3		 Backlog	<input type="button" value="audit 4"/>	
41	⌚ Fazit und kritisch reflektiertes Prozessassessment #120	 antonztsv, calvinhnzr, and floseph	<input type="button" value="Audit 4"/>	1		 Backlog	<input type="button" value="audit 4"/>	

## Kommentar

Für Audit 4 wird festgehalten, dass die fehlenden Proof of Concepts durchgeführt werden sobald die reservierte und bestellte Hardware verfügbar sind.  
Außerdem werden weitere, für den funktionalen Prototypen erforderliche Funktionen, implementiert und erweitert.

Am Ende wird ein Poster mit den Ergebnissen des Entwicklungsprojektes erstellt und ein kritisch reflektiertes Prozessassessment durchgeführt.