FH University of Applied Sciences TECHNIKUM WIEN

Dokumentation & Projekttagebuch

Innovation Lab 2
Jahr 2025

Projekt: **IDERHA**

Team: Nr. **23**

Inhalt

1. Allgemeine Informationen

Projektname: IDERHA

Supervisor: Lukas Rohatsch

Innovation Lab 2, SS 2025

Projektteam:

Zehinovic Aldin, if22b130@technikum-wien.at - Teamleiter
Dervisefendic Armin, if23b040@technikum-wien.at

*Bitte beachten Sie: Um mögliche Urheberrechtsprobleme zu vermeiden, haben wir uns entschieden, unsere Lösung **eHealth Insights** anstelle von **IDERHA** zu nennen.*

Management-Summary des Projektes

Ziel ist die Entwicklung einer Website für einen eHealth Data Space, der als sichere und benutzerfreundliche Plattform dient. Die Website soll die Verarbeitung, Analyse und den Zugang zu Gesundheitsdaten revolutionieren, indem er einen transparenten und effizienten Datenfluss zwischen verschiedenen Akteur:innen im Gesundheitswesen ermöglicht, ohne die Sicherheit und den Datenschutz zu beeinträchtigen

Rahmenbedingungen und Projektumfeld

Rahmenbedingungen:

Für das **eHealth Insights**-Projekt werden folgende Rahmenbedingungen festgelegt:

Es werden Java mit Spring Boot für das Backend und React.js mit Vite für das Frontend verwendet. Die Daten werden in einer PostgreSQL-Datenbank gespeichert. GitHub dient als Versionskontrollsystem, Figma für den Webdesign-Prototyp. Zur Sicherstellung der Sicherheit und des Datenschutzes werden Spring Security für Authentifizierung und Autorisierung sowie CORS für die sichere Kommunikation zwischen Frontend und Backend implementiert. Zudem wird Docker für die Containerisierung der Anwendung verwendet.

Qualitätseigenschaften:

- **Benutzerfreundlichkeit:** Intuitive Bedienung, einfache Navigation und schneller Zugriff auf Datensätze und Analysewerkzeuge.
- **einfache Wartbarkeit/Flexibilität:** unkomplizierte Wartung und Weiterentwicklung möglich
- **Sicherheit und Datenschutz:** Höchste Sicherheitsstandards zum Schutz sensibler Gesundheitsdaten durch Verschlüsselung und Zugriffskontrollen.
- **Performance und Effizienz:** Schnelle und effiziente Datenverarbeitung für zeitnahe Ergebnisse in der Gesundheitsdatenanalyse.
- **Skalierbarkeit:** Effiziente Handhabung wachsender Datenmengen und einfacher Anschluss neuer Datenquellen.

Semester-Roadmap:

Bemerkung: *Zu Beginn des Semesters hat sich das Team von 5 Personen auf 2 reduziert: Aldin Zehinovic und Aldin Dervisefendic. Nach Rücksprache mit dem Betreuer haben wir beschlossen, den Umfang des Projekts in diesem Semester zu reduzieren, damit wir zwei die geplanten Funktionen umsetzen können.*

Sprint 1 (Dienstag, 25 März) - Docker & Pipeline

- Docker setup
- Pipeline - GitHub Actions

Sprint 2 (Dienstag, 8 April) - Rollenbasierter Zugriff & Frontend Test

- Login RBAC
- Frontend-Tests mit vitest und jsdom

Sprint 3 (Dienstag, 29 April) - API-Sicherheit & Datenübertragung

- Separate Datenbanken für die Login-Verwaltung
- Beheben von Problemen mit der DB-Backend-Verbindung
- Analytics erweitern

Sprint 4 (Dienstag, 13 Mai) - Aufgabe nicht hochgeladen

-

Sprint 5 (Dienstag, 27 Mai) - RBAC erweitern & Benutzerverwaltungsseite

- Behebung von Problemen mit der DB-Backend-Verbindung
- Backend mit mehreren Datenbanken
- RBAC mit den Rollen admin/hospital/researcher
- Benutzerverwaltungsseite für Administratoren und Krankenhäuser
- Analyseseite mit Einbeziehung aller Datenbanken

Sprint 6 (Dienstag, 10 Juni) - Upload Data

- Option: die Datenbank einem Krankenhausbenutzer zuzuweisen
- Voll funktionsfähige Seite zum Hochladen von Daten

Collaboration & Tooling:

- **Github:** https://github.com/mide553/IDERHA
- **PostgreSQL**
- **Figma**
- **Visual Studio Code**
- **Docker**

Anmerkungen:

Am Ende des 4. Semesters hat die Webseite ihre erste funktionsfähige Version.

2. Projekt-Kurzbeschreibung

Das Hauptziel von unserem Projekt besteht darin, einen innovativen eHealth Data Space zu entwickeln, der eine sichere, benutzerfreundliche Plattform zur Verarbeitung und Analyse von Gesundheitsdaten bietet. Diese Plattform soll den transparenten und effizienten Austausch von Daten zwischen verschiedenen Akteur:innen im Gesundheitswesen ermöglichen und gleichzeitig höchste Standards für Sicherheit und Datenschutz gewährleisten.

Erste funktionsfähige Version (SS25):
Der Zugriff auf die Plattform erfolgt über eine Web-Oberfläche mit Login-Funktion. Nach erfolgreicher Authentifizierung über das Login-Formular erhalten Nutzer*innen entsprechend ihrer Rolle Zugriff auf die Dienste von **eHealth Insights**.
Für jedes Krankenhaus wird ein eigener Docker-Container bereitgestellt, der eine separate Datenbank enthält. Diese Containerarchitektur stellt sicher, dass Patient*innendaten pro Krankenhaus strikt getrennt bleiben und nicht vermischt werden. Jeder Container verfügt über eine eigene API, über die intern kommuniziert wird, z. B. zur Verarbeitung der hochgeladenen Daten.

Die Dateneingabe erfolgt über eine dedizierte „Upload Data“–Seite, auf der Krankenhausnutzer*innen eine Datei auswählen und in das System hochladen können. Die hochgeladenen Daten werden anschließend der jeweiligen Krankenhausdatenbank zugeordnet.

Zusätzlich existiert eine zentrale, private Datenbank, die ausschließlich für Benutzer-Login und rollenbasierte Zugriffskontrolle (RBAC) verwendet wird. Es sind drei Rollen definiert:

- **Admin**: Zuständig für Benutzer- und Datenbankmanagement.
- **Hospital**: Kann Patientinnendaten hochladen und Forscherinnen zur Plattform hinzufügen.
- **Researcher**: Hat Zugriff auf eine Analyse-Seite, auf der Daten aus den jeweiligen Krankenhausdatenbanken mithilfe benutzerdefinierter Abfragen ausgewertet werden können.

Alle Patient*innendaten werden in sicheren Datenbanken gespeichert und durch verschlüsselte Übertragungsprotokolle sowie rollenbasierte Zugriffskontrollen geschützt.

Zukünftige Implementierungen (WS25):

Im weiteren Projektverlauf sind verschiedene Erweiterungen geplant, um die Plattform funktional und qualitativ weiterzuentwickeln. Dazu zählen die Integration von Visualisierungen wie Diagrammen oder Graphen zur besseren Darstellung medizinischer Zusammenhänge sowie die Möglichkeit, Daten in gängigen Formaten wie PDF oder DOC zu exportieren. Zudem werden systematische Backend-Tests durchgeführt, um die Stabilität und Zuverlässigkeit des Systems sicherzustellen. Die Performance der Anwendung wird optimiert, Sicherheitsmaßnahmen weiter ausgebaut und umfassende Maßnahmen zur Qualitätssicherung umgesetzt. Durch das Einholen und Auswerten von Nutzerfeedback sollen weitere Verbesserungen identifiziert werden. Abschließend werden Deployment und Projektdokumentation durchgeführt, um die Lösung produktionsreif bereitzustellen.

Größte Herausforderungen:

Die zentralen Herausforderungen des Projekts bestehen darin, eine hohe Systemleistung und Sicherheit zu gewährleisten, die Interoperabilität zwischen verschiedenen Gesundheitsdienstleistern zu fördern und strenge Datenschutzvorgaben einzuhalten. Darüber hinaus ist eine intuitive Benutzeroberfläche für das Nutzer:innenportal entscheidend, um eine einfache Verwaltung und Interaktion mit Gesundheitsdaten zu ermöglichen.

Mehrwert für die Anwender:innen:

eHealth Insights plant, die Produktivität zu steigern, indem die Analyse und Interpretation von Gesundheitsdaten erleichtert wird. Dadurch sollen auch Nutzer*innen ohne tiefgehende Kenntnisse in Datenvisualisierung schnell umsetzbare Erkenntnisse gewinnen können. Ziel ist es außerdem, den globalen Zugriff auf große Datenmengen für verschiedene Nutzergruppen zu ermöglichen und die Option zu bieten, Daten in gängigen Formaten wie PDF oder DOC zu exportieren.

Projektumfang:

Das Hauptziel von **eHealth Insights** ist die Entwicklung eines innovativen eHealth Data Space, der eine sichere und benutzerfreundliche Plattform zur Verarbeitung, Analyse und Visualisierung von Gesundheitsdaten bereitstellt.

Nicht-Ziele:

Es findet keine direkte Interaktion zwischen Nutzer*innen und der Datenbank statt, d.h. alle Abfragen laufen über das Backend. Die Plattform erhebt keinen Anspruch auf medizinische Entscheidungshilfe oder klinische Diagnoseunterstützung. Die Integration von Real-Time-Streaming oder KI-Modulen ist aktuell nicht Teil des Projektumfangs.

3. Spezifikation der Lösung

3.1 Systemumfeld

Die Hauptkomponenten des Systems sind:

- **Frontend:** Eine webbasierte Benutzeroberfläche für unterschiedliche Nutzergruppen (“Admin”, “Hospital”, “Researcher”).
- **Backend:** Eine Spring Boot API zur Datenverarbeitung, Authentifizierung und Kommunikation mit der Datenbank.
- **Datenbank:** Eine PostgreSQL-Datenbank, die über Docker-Container verwaltet wird und Patientendaten sowie Berechtigungen speichert.
- **Schnittstellen:** APIs zur Anbindung mehrerer „Krankenhaus“-Datenbanken und zur Datenauswertung mit Visualisierungstools.

Systemgrenzen:

- Datenverarbeitung und Speicherung erfolgt ausschließlich innerhalb der Plattform.
- Keine direkte Interaktion zwischen Nutzer:innen und der Datenbank – alle Operationen laufen über das Backend.
- Rollenbasierte Zugriffskontrollen zur Einschränkung der Datenverfügbarkeit.
- Admins sind für das Benutzer- und Datenbankmanagement zuständig.

3.2 Features (Funktionale Anforderungen)

Die Kernfunktionalitäten von **eHealth Insights** umfassen:

- Benutzerverwaltung

- Login
- Rollenbasierte Zugriffskontrolle (“Admin”, “Hospital”, “Researcher”)
- Sitzungsverwaltung und sichere Authentifizierung

- Datenmanagement

- Speicherung von Patient*innendaten in sicheren Datenbanken
- Unterstützung für mehrere Krankenhäuser über separate Docker-Container

- Analyse & Visualisierung

- Darstellung von Patientendaten in einer übersichtlichen UI
- Integration von Graph-Datenbanken zur Analyse von Beziehungen in den Daten
- Export-Funktion für Berichte (PDF, DOC)

- API & Schnittstellen

- Das Projekt folgt einer typischen REST-API-Architektur
- Spring Boot läuft auf Port 8080
- HTTP-Anfragen/-Antworten mit JSON-Daten

- Benutzeroberfläche

- Moderne UI mit React.js + Vite
- Dynamische Datenvisualisierung für medizinische Analysen

3.3 Screenshots

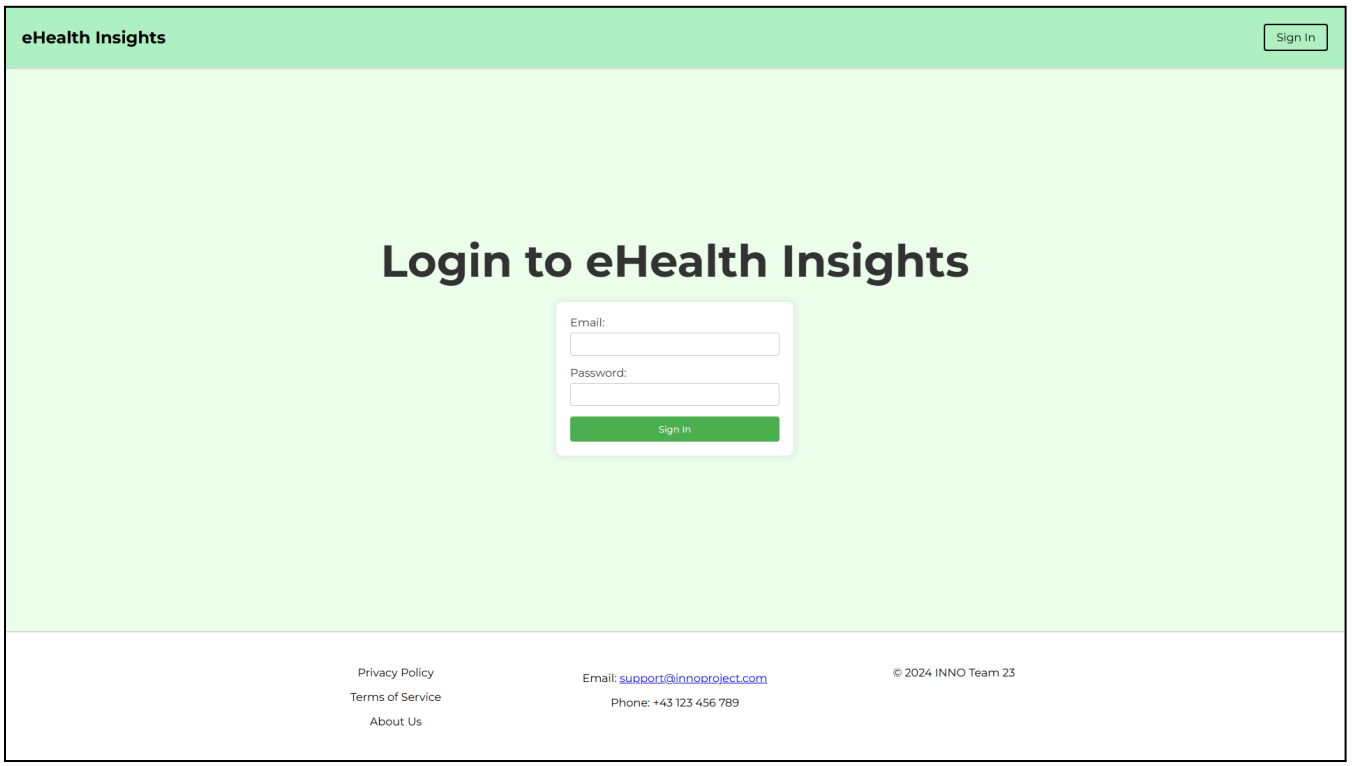

Abbildung 3.3.1: Login-Seite

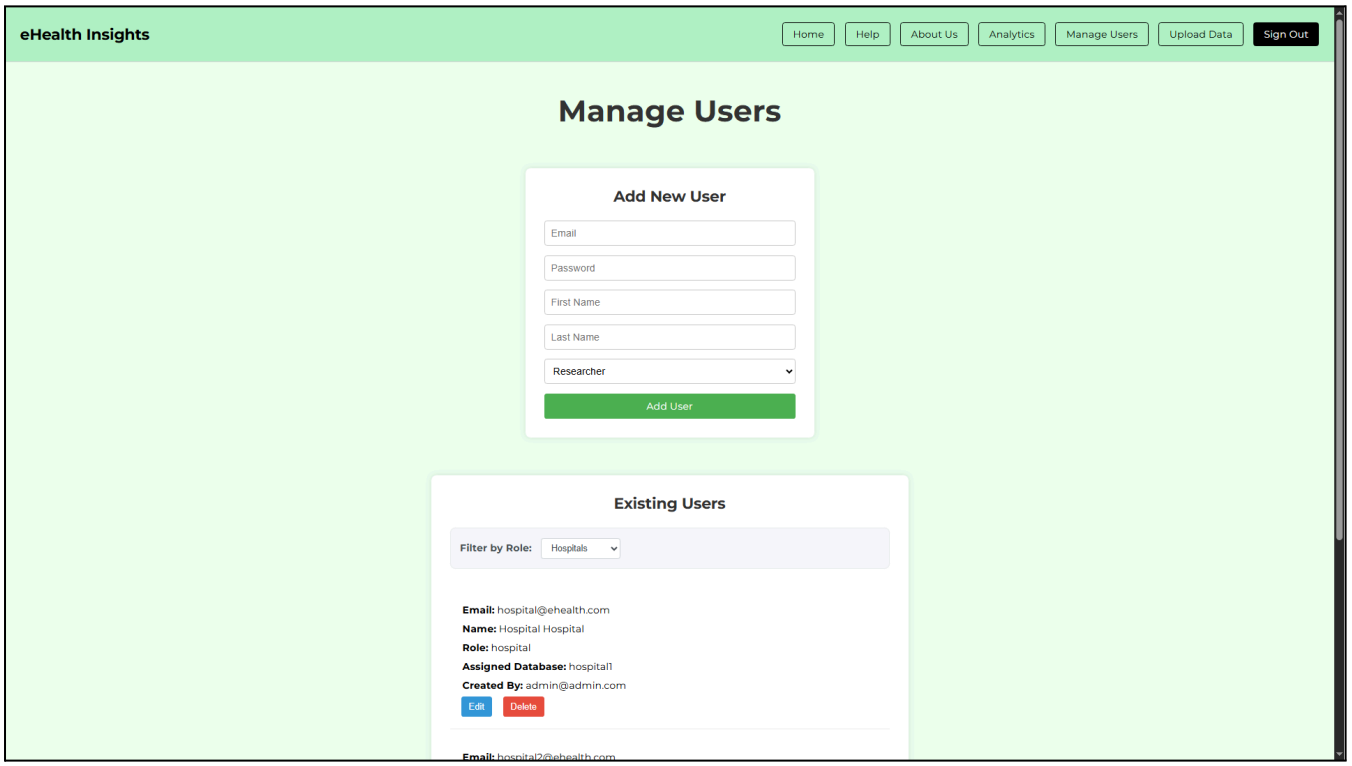

Abbildung 3.3.2: "Manage Users"-Seite (Admin)

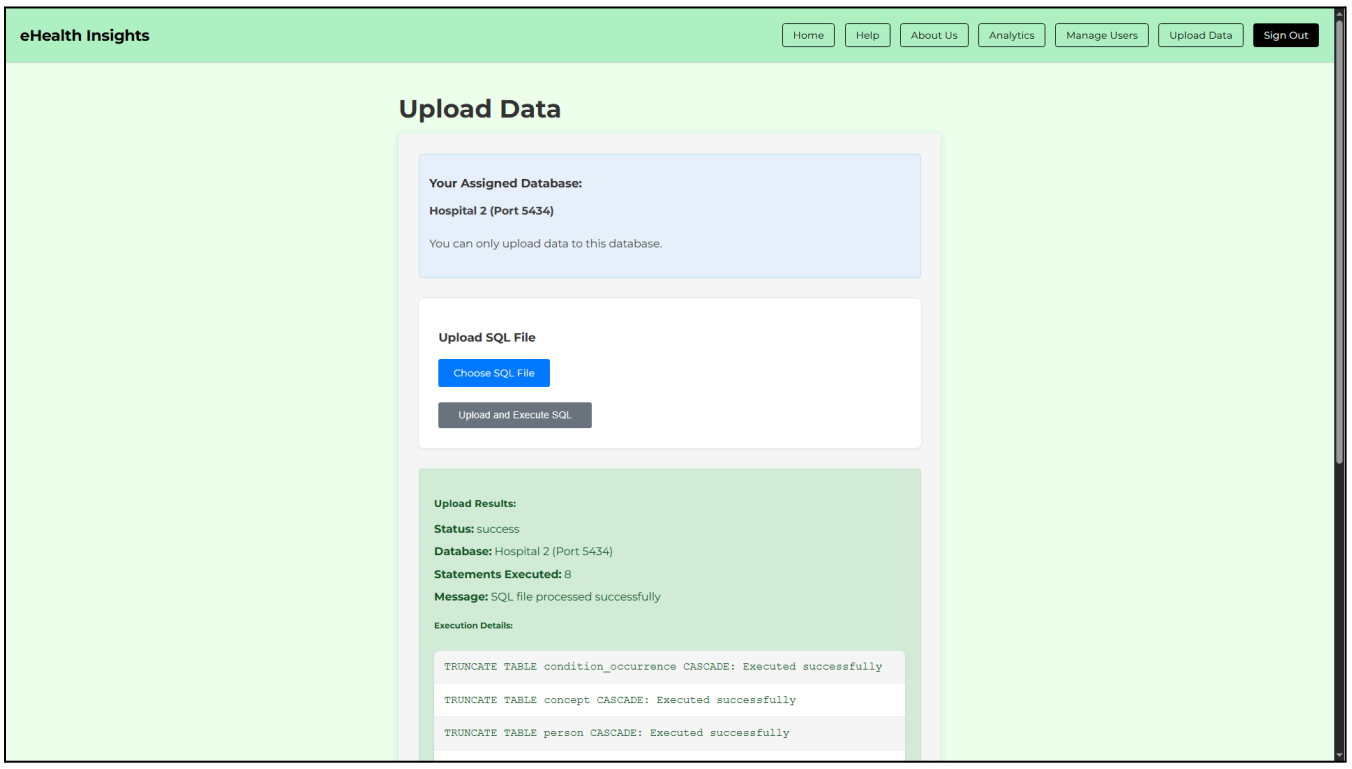

Abbildung 3.3.3: "Upload Data"-Seite (Hospital)

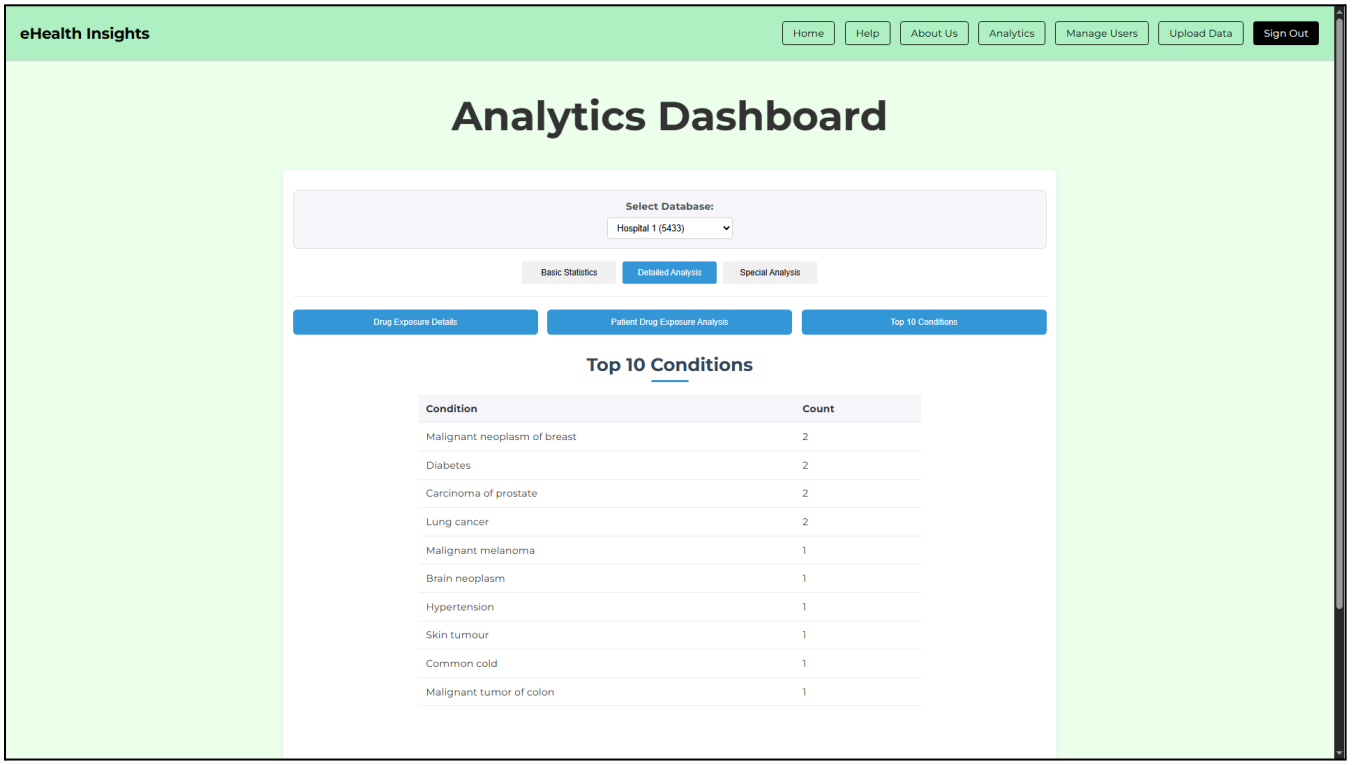

Abbildung 3.3.4: “Analytics Dashboard”-Seite (Researcher)

3.4. Sequenzdiagramme

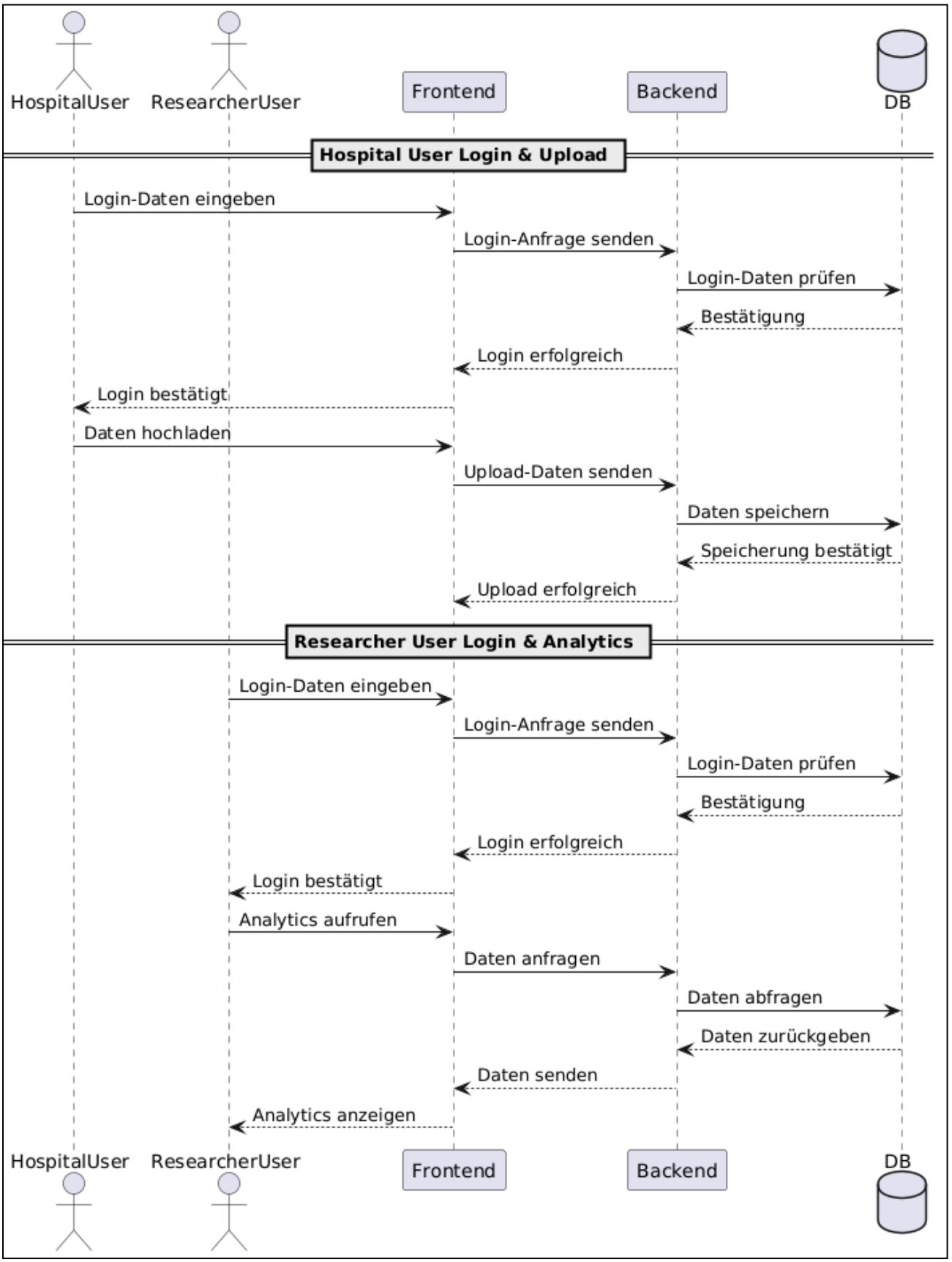

Abbildung 3.4.1: Hospital User Login & Upload und Researcher-User Login & Analytics

3.5 Schnittstellen

Die Hauptschnittstellen sind:

- **Frontend → Backend:** REST API für Datenabfragen und Benutzerinteraktionen.
- **Backend → Datenbank:** Kommunikation mit der jeweiligen PostgreSQL-Datenbank über Spring Boot. Für jedes Krankenhaus gibt es eine eigene Datenbank, die in einem separaten Docker-Container betrieben wird und über eine eigene API verfügt.
- **Krankenhäuser → Frontend:** Daten werden über die „Daten hochladen“–Seite im Frontend hochgeladen und gelangen so in den jeweiligen Krankenhaus-Container.

3.6 Qualitätseigenschaften

Die Lösung zeichnet sich durch eine moderne, containerisierte Microservice-Architektur aus, die auf Java (Spring Boot) und React.js basiert und eine hohe Skalierbarkeit, Erweiterbarkeit und Wartbarkeit gewährleistet. Die Plattform ist über Docker-Container modular aufgebaut und erlaubt durch die klare Rollenverteilung (Admin, Hospital, Researcher) eine sichere, DSGVO-konforme Nutzung. Performance und Effizienz werden durch schnelle REST-APIs und eine entkoppelte Datenverarbeitung erreicht, während GitHub Actions, automatisierte Tests und strukturierte Datenhaltung (OMOP-CDM) die Qualitätssicherung unterstützen. Die benutzerfreundliche Oberfläche und die einfache Integration neuer Datenquellen tragen zur hohen Usability und Zukunftsfähigkeit bei.

3.7 Sonstige wesentliche Lösungsmerkmale

- Automatisierte Frontend- und Backend-Tests.
- Pipeline mit GitHub Actions.
- Einhaltung von Datenschutzstandards (DSGVO).

4. Aufwandschätzung

4.1 Aufwandschätzung mittels Delphi-Methode

Zu Beginn des Semesters hat unser Team aus fünf Personen die Aufwandschätzung für die geplanten Arbeitspakete anhand der Delphi-Methode durchgeführt. Insgesamt wurden 8 Themenbereiche (Themes), darunter 12 Epics, sowie 21 User Stories erfasst und bewertet.

Für jede User Story wurden optimistische, wahrscheinliche und pessimistische Schätzwerte erfasst und im Team diskutiert, um eine fundierte und realistische Einschätzung zu gewährleisten.

Auf Basis dieser Werte wurde mithilfe der PERT-Formel (Gewichtung 1:4:1) ein kalibrierter Erwartungswert für den Aufwand berechnet, der die Unsicherheit in der Schätzung abbildet. Die Schätzung umfasst zudem Standardabweichungen, Varianzen sowie die Zuordnung zu T-Shirt-Sizes für Aufwand und Businesswert.

Zur Veranschaulichung sind im Folgenden zwei zentrale Tabellen aus dem Excel-Dokument eingefügt:

- Die Haupttabelle mit den User Stories, Schätzwerten und T-Shirt-Sizing
  [Abbildung 4.1.1 , Abbildung 4.1.2]
- Die Tabelle „Aufwandschätzung mit Zusagesicherheit“, welche den erwarteten Aufwand in Personenstunden bei unterschiedlichen Konfidenzniveaus (2 % bis 98 %) zeigt
  [Abbildung 4.1.3]
- Diese Daten werden zusätzlich durch ein Streudiagramm visualisiert:
  - X-Achse: Erwarteter Aufwand in idealen Personenmonaten
  - Y-Achse: Zusagesicherheit (Wahrscheinlichkeit der Einhaltung)
    [Abbildung 4.1.4]

Aus der Analyse ergibt sich bei einer 50%igen Zusagesicherheit ein Aufwand von ca. 68 Personenstunden (etwa 8,5 Personentage).

Das vollständige Excel-Dokument mit allen Berechnungen, Annahmen und Detaildaten ist im zugehörigen GitHub-Repository enthalten und kann dort eingesehen werden.

Delphi Verfahren mit PERT - Unsicherheit ausdrückbar durch Schätzbereich: Bre							
ID	Themes / Areas / Arbeitspakete Toplevel	Epics / User Stories / Arbeitspakete TopLevel	User Stories / Detail Level / Beschreibung	Optimistisch (Sp)	Wahrscheinlich (Sp)	Pessimistisch (Sp)	Dokumentation von Annahmen, Überlegungen, Risiken, Bedingungen und Diskussionsergebnissen aus dem Schätzmeeting
				39,0	68,0	98,0	
	Benutzerverwaltung & RBAC	RBAC-System entwickeln					
1			Als Admin möchte ich Benutzer mit spezifischen Rollen (Admin/Hospital/Researcher) anlegen können	2	3	5	Rollen sind festgelegt; Risiko falscher Rollenzuweisung; RBAC muss konsistent umgesetzt werden.
2			Als Entwickler möchte ich Zugriffe per Role-Based Access Control (RBAC) einschränken	3	5	7	
		Benutzerregistrierung und -verwaltung					
3			Als Hospital-User möchte ich neue Forscher*innen zur Plattform einladen können	2	3	4	
4			Als Benutzer möchte ich mich sicher einloggen können (Spring Security)	2	3	4	
5			Als Admin möchte ich Benutzer deaktivieren/löschen können	1	2	3	
	Daten-Upload & Datenmanagement	Daten von Krankenhäusern hochladen					
6			Als Hospital-User möchte ich eine Datei hochladen, die in die richtige Krankenhaus-Datenbank gespeichert wird	2	4	6	Einheitliches Dateiformat angenommen; Risiko ungültiger Dateien; serverseitige Validierung notwendig.
7			Als Entwickler möchte ich sicherstellen, dass hochgeladene Daten validiert werden	2	3	4	

Abbildung 4.1.1

D	I	J	K	L	M	N	O	P	Q
							T-Shirt-Sizing		
User Stories / Detail Level / Beschreibung	Erwartet nach PERT 1:4:1 (Sp)	Kalibrierter erwarteter Aufwand (Ph)	p(range) (5-100)	Divisor (0,25-6)	Standard-Abweichung (Ph)	Varianz	Aufwand (T-Shirt: XS-XXL)	Businesswert (T-Shirt: XS-XXL)	Nettobusiness-wert (T-Shirt: XS-XXL)
	68,17	68,17			5,777	33,372			
			100,00	6,00					
Als Admin möchte ich Benutzer mit spezifischen Rollen (Admin/Hospital/Researcher) anlegen können	3,17	3,17	80,00	2,60	1,154	1,331	M	XXL	XXL
Als Entwickler möchte ich Zugriffe per Role-Based Access Control (RBAC) einschränken	5,00	5,00	75,00	2,10	1,905	3,628	M	XXL	XXL
			100,00	6,00					
Als Hospital-User möchte ich neue Forscher*innen zur Plattform einladen können	3,00	3,00	85,00	2,60	0,769	0,592	M	L	L
Als Benutzer möchte ich mich sicher einloggen können (Spring Security)	3,00	3,00	80,00	2,60	0,769	0,592	M	XXL	XXL
Als Admin möchte ich Benutzer deaktivieren/löschen können	2,00	2,00	90,00	3,30	0,606	0,367	S	L	L
			100,00	6,00					
Als Hospital-User möchte ich eine Datei hochladen, die in die richtige Krankenhaus-Datenbank gespeichert wird	4,00	4,00	75,00	2,10	1,905	3,628	M	XXL	XXL
Als Entwickler möchte ich sicherstellen, dass hochgeladene Daten validiert werden	3,00	3,00	80,00	2,60	0,769	0,592	M	L	L

Abbildung 4.1.2

Aufwandschätzung mit Zusagesicherheit (Konfidenzintervall)				
Zusagesicherheit (% Konfidenz)	Aufwand (Personenstunden)	Aufwand (Personentage brutto)	Aufwand (Personenmonate brutto)	Aufwand (Personenjahre brutto)
2%	56,61	7,08	0,35	0,03
10%	60,77	7,60	0,38	0,03
16%	62,39	7,80	0,39	0,03
20%	63,31	7,91	0,40	0,03
25%	**64,30**	**8,04**	**0,40**	**0,03**
30%	65,16	8,15	0,41	0,03
40%	66,72	8,34	0,42	0,03
50%	**68,17**	**8,52**	**0,43**	**0,04**
60%	69,61	8,70	0,44	0,04
70%	71,17	8,90	0,44	0,04
75%	**72,04**	**9,00**	**0,45**	**0,04**
80%	73,02	9,13	0,46	0,04
84%	73,94	9,24	0,46	0,04
90%	75,56	9,45	0,47	0,04
98%	**79,72**	**9,97**	**0,50**	**0,04**

Abbildung 4.1.3

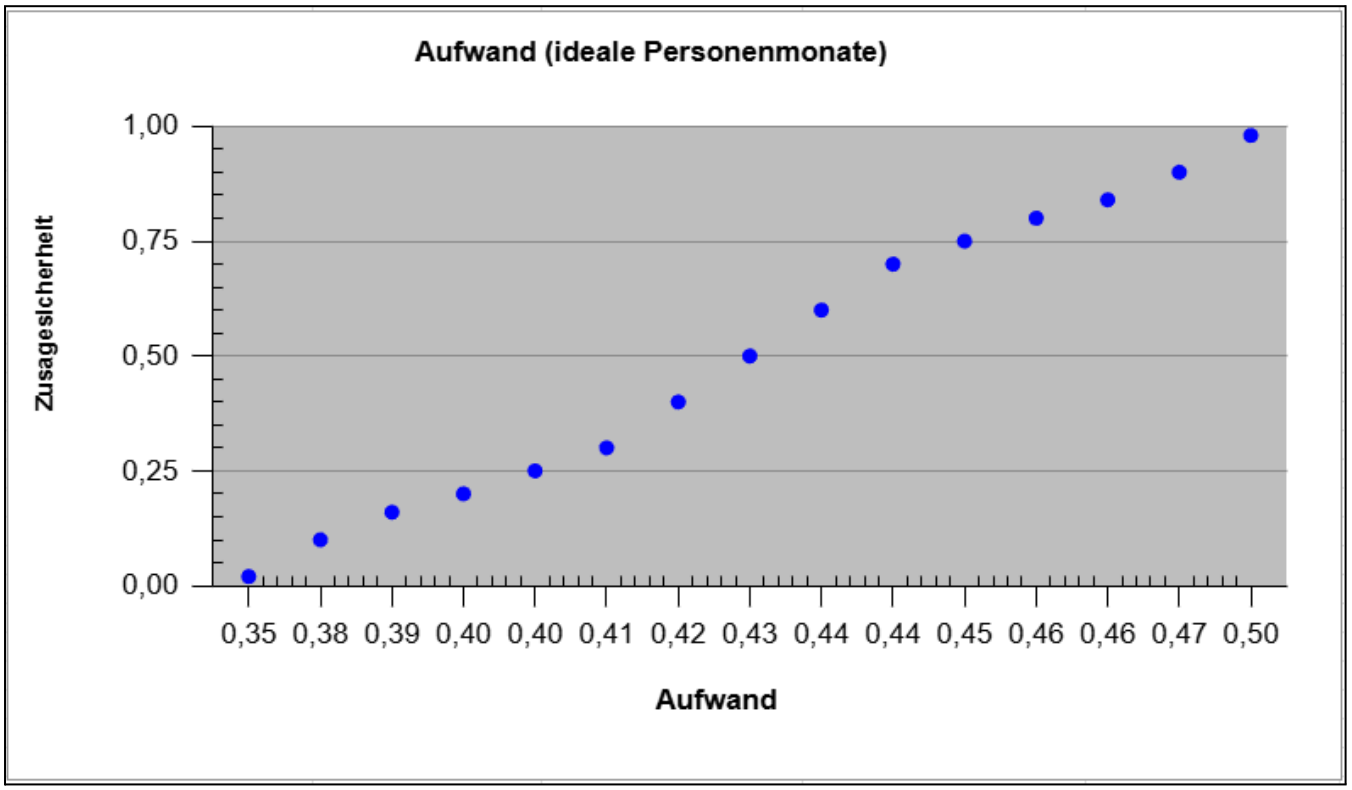

Abbildung 4.1.4

5. Auslieferung

In diesem Abschnitt wird der Lieferumfang unserer Lösung sowie alle notwendigen Informationen für die Übergabe an Kunden oder Betriebsteams beschrieben. Dies umfasst die fertige Lösung inklusive Source-Code, die Systemarchitektur, Datenhaltung, erforderliche Lizenzen sowie eine ausführliche Installations- und Betriebsvorbereitung.

5.1. Fertige Lösung inklusive Source-Code

Unsere Lösung besteht aus folgenden Hauptkomponenten:

- **Frontend:** Eine benutzerfreundliche Web-Oberfläche auf Basis von React, die den Zugriff auf Gesundheitsdaten sowie deren Visualisierung ermöglicht.
- **Backend:** Ein RESTful API-Service, implementiert mit Spring Boot, der Daten verarbeitet und mit der zentralen Datenbank sowie externen Services kommuniziert.
- **Datenbank:** Eine zentrale PostgreSQL-Datenbank, die alle gesammelten Gesundheitsdaten speichert und eine effiziente Verarbeitung ermöglicht. Die Datenbankstruktur orientiert sich am OMOP Common Data Model (CDM).
- **Docker-Container:** Die PostgreSQL-Datenbank wird in einem Docker-Container betrieben, um eine einfache Bereitstellung und Verwaltung sicherzustellen.

Der vollständige Source-Code wird im GitHub-Repository (https://github.com/mide553/IDERHA) hinterlegt und steht für Weiterentwicklung sowie Wartung bereit.

5.2. Systemarchitektur und Datenhaltung

Die Lösung basiert auf einer modularen Microservice-Architektur, die folgende Struktur aufweist:

- **Frontend:** React-basiertes Web-Frontend, kommuniziert über HTTP mit dem Backend.
- **Backend:** Spring Boot Service mit REST-API-Endpunkten, verwaltet die Logik und Datenverarbeitung.
- **Datenbank:** PostgreSQL-Datenbank, betrieben in einem Docker-Container, mit strukturierter Speicherung nach OMOP-CDM.
- **Containerisierung:** Nur die Datenbank wird mittels Docker betrieben, Frontend und Backend laufen als eigenständige Services.

5.3 Lizenzierung und Copyright

- Die Lösung nutzt Open-Source-Komponenten wie React, Spring Boot und PostgreSQL, jeweils unter ihren jeweiligen Lizenzen (MIT, Apache 2.0, PostgreSQL License).
- Für den Betrieb sind keine zusätzlichen kostenpflichtigen Lizenzen erforderlich.

5.4 Hardware-Anforderungen

- Die Lösung ist für den Betrieb auf Standard-Serverinfrastruktur ausgelegt.
- Empfohlen wird ein Server mit mindestens 4 CPU-Kernen, 8 GB RAM und ausreichend Speicherplatz für Datenhaltung (abhängig vom erwarteten Datenvolumen).
- Docker und Docker Compose müssen auf dem Zielsystem installiert sein, um die Datenbank im Container auszuführen.

5.5 Voraussetzungen für Installation und Betrieb

Vor dem Download und der Installation der Lösung sollten folgende Voraussetzungen erfüllt sein:

- Docker (empfohlen Version 24.0 oder höher) und Docker Compose für die Containerisierung und Verwaltung der PostgreSQL-Datenbank.
- Git (optional, zum Klonen des Repositories).
- Java Development Kit (JDK), idealerweise OpenJDK 17, für den Betrieb des Spring Boot-Backends. Stellen Sie sicher, dass java in der Systemumgebung verfügbar ist.
- Maven (Version 3.6 oder höher), um das Backend zu bauen und auszuführen.
- Node.js (Version 16 oder höher) und npm für die Installation und den Betrieb des React-Frontends.
- Netzwerkzugang, damit alle benötigten Abhängigkeiten und Containerimages heruntergeladen werden können.
- Freie Ports 3000 (Frontend) und 8080 (Backend) auf dem Hostsystem.

5.6 Installation und Betrieb

Installationsschritte:

1. Repository klonen:
   git clone https://github.com/mide553/IDERHA

2. In das Projektverzeichnis wechseln:
   cd IDERHA

3. Docker-Container für die Datenbank starten:
   docker-compose up -d

4. Backend starten:
   cd backend
   mvn clean install
   mvn spring-boot:run

5. Frontend im neuen Terminal starten:
   cd react-app
   npm install
   npm install -g vite
   npm run dev

Die Lösung ist anschließend über http://localhost:3000 (Frontend) erreichbar. Das Backend läuft auf Port 8080.

6. Unser Projekt-Tagebuch

3. Semester

24.09.2024 14:30 erstes Meeting mit Prof. Lukas Rohatsch über Zoom: Besprechung der Projektanforderungen, Fragerunde zur Klärung von Unklarheiten.

02.10.2024 17:30 Teammeeting über Discord: Festlegung der Technologie (Java, Spring Boot, React, Postgresql). Einigung der Personen, welche das Projekt nach außen vertritt (Mehdi Erdem) und Stellvertreter (Aldin Zehinovic). Besprechung und Beginn der Überführung der Projektspezifikationen in die Dokumentation für Punkt 1 & 2.

10.10.2024 15:00 Teammeeting zur Fertigstellung von Punkt 1 & Punkt 2 der Dokumentation für die Abgabe. Des Weiteren wurde eine grobe Aufgabenverteilung für den ersten Sprint festgelegt, um den Arbeitsfortschritt zu organisieren. Eine E-Mail wurde an Herrn Rohatsch gesendet, um die Terminabstimmung für weitere Treffen zu ermöglichen.

18.10.2024 17:30 Das Meeting mit Herrn Prof. Rohatsch fand statt, um die ausgearbeiteten Punkte 1 & 2 des Project Diaries zu besprechen. (Projektkurzbeschreibung, Spezifikation der Lösung…). Weiters haben wir das Datenbankmodell OMOP besprochen und die hilfreichen Liinks bezüglich des Projekts bekommen.

25.11.2024 Besprechung des UI Oberfläche und des Architecture Design des Central Nodes (Aldin, Armin, Mehdi). Datenbank aufsetzen (pgadmin).

05.12.2024 11:00 Das Meeting mit Herrn Prof. Rohatsch fand statt, es wurden die nächsten Schritte besprochen.

12.12.2024 14:00 Teammeeting über Discord, hier haben wir die Aufgaben für Frontend und Backend geteilt.

10.01.2025 16:00 Uhr, war unser letztes Meeting mit Herrn Prof. Rohatsch, wir haben besprochen, mit was wir in der 4.Semester weiter machen werden: - Docker images für die Datenbanken und einen central node - Entwurf von Daten in Graphen

18.01.2025 17:00 Wir haben uns getroffen und an der der Dokumentation gearbeitet (Aldin, Armin, Mehdi)

26.01.2025 16:00 Wir haben uns getroffen und an der der Dokumentation gearbeitet und das Video gedreht (Aldin, Armin, Mehdi)

28.01.2025 21:00 letzte Kontrolle der Dokumentation und Präsentation bevor der Abgabe (Aldin, Armin, Mehdi)

4. Semester

09.03.2025 Projektbesprechung

10.03.2025 Dokumentation & Projekttagebuch und Aufwandschätzung

24.03.2025
Erste gemeinsame Planung des Semesters nach Reduktion des Teams auf zwei Personen. Besprechung der Ziele, Priorisierung der Funktionen, grobe Sprintplanung erstellt.

25.03.2025 Sprint Upload
Excel Datei ausfüllen.

28.03.2025 Meeting mit dem Supervisor
Wir besprachen, was wir im Sprint umgesetzt hatten. Außerdem sprachen wir über sinnvolle Aufgaben für den kommenden Sprint.

08.04.2025 Meeting mit dem Supervisor
Besprechung zum Thema Rollenbasierter Zugriff und Frontend-Tests. Wir präsentierten die Fortschritte aus Sprint 2. Danach planten wir gemeinsam die nächsten Aufgaben im Bereich API-Sicherheit.

15.05.2025 Team Meeting

29.04.2025 Meeting mit dem Supervisor
Review des aktuellen Stands bezüglich API-Sicherheit, getrennten Datenbanken und Analytics-Erweiterung. Wir besprachen daraufhin den Fokus für Sprint 4.

13.05.2025 Sprint 4
Obwohl der Sprint planmäßig abgeschlossen und fertiggestellt war, haben wir versehentlich die Abgabe vergessen. Im Meeting besprachen wir den fertigen Sprintinhalt ausführlich mit unserem Supervisor. Er gab uns wertvolles Feedback zur Umsetzung und half, Prioritäten für den nächsten Sprint klar zu definieren.

20.05.2025 Team Meeting

27.05.2025 Meeting mit dem Supervisor
Er gab uns wertvolles Feedback und wies darauf hin, dass wir separate private und öffentliche Datenbanken einrichten sollten. Außerdem half er uns, Prioritäten für den nächsten Sprint zu setzen.

02.06.2025 Team Meeting

10.06.2025 Letztes Sprint-Meeting des Semesters.
Wir präsentierten die Upload-Seite sowie die Zuweisungsfunktion für Krankenhausnutzer. Unser Mentor gab positives Feedback zur Umsetzung. Abschließend besprachen wir die Pläne für das nächste Semester.

19.06.2025 Team Meeting
Dokumentation und Video