FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Semesterarbeit des Studiengangs CAS Fullstack-Development

von

Kay Roman Hertenstein

Roggwil, Januar – Juni 2025

Inhaltsverzeichnis

1 Applikationsidee

Die Idee für die Applikation, kam aus meinem Freundeskreis. Einige meiner Bekannten kommen aus dem Handwerklichen Tätigkeitsgebiet – dabei ging es darum dass häufig für kleinere handwerkliche/technische Arbeiten ein Unternehmen oder ein Spezialist beauftragt werden muss – was Verhältnismässig schnell hohe Kosten verursachen kann. Daraus entstand die Idee eine Applikation zu entwickeln welche eine einfache Möglichkeit bietet Dienstleistungen / Tätigkeiten anzubieten. Das Zielpublikum sind Fachkräfte welche sich im Nebenerwerb ein zusätzliches Einkommen verdienen wollen.

1.1 Anwendungsumfang

Die Anwendung ist eine Single-Page-Applikation (SPA) mit einem Dienstleistungs-Board als zentrale Funktionalität. Das Dienstleistungs-Board ermöglicht eine effiziente Suche und Navigation innerhalb der verfügbaren Angebote durch eine leistungsfähige Volltextsuche und Filterfunktionen.
Benutzer können sich registrieren und nach der Anmeldung eigene Dienstleistungs-Angebote erstellen, bearbeiten und löschen. Zudem sollen registrierte Nutzer ein öffentliches Profil mit ihren Kontaktinformationen erstellen und verwalten, welches für Nutzer als Referenz der angebotenen Dienstleistungen sichtbar ist.

1.1.1 Nicht - Funktionale Anforderungen Frontend

Das Frontend ist eine Webbasierte Benutzeroberfläche, die eine effiziente Interaktion mit der Applikation ermöglicht. In den nicht funktionalen Anforderungen werden die Rahmenbedingungen der Applikation sowie die Merkmale der Benutzerfreundlichkeit definiert. Die jeweilige nicht funktionale Anforderung enthält einen Verweis [**NFAFE-ID**] auf die genaue Spezifizierung, diese sind im Dokument **Spezifizierung_technische_Anforderungen_kh** Abschnitt 1.1 Spezifizierung Nicht Funktionale Anforderungen Frontend NFAFE definiert.

Allgemeines

- Responsives Webdesign **[NFAFE ID-001]**:
    - Optimierung der Benutzeroberfläche für verschiedene Endgeräte (Desktop, Tablet und Smartphone).

Perfomance [NFAFE ID-002]:

- Die Applikation soll ein genügendes Resultat im Prüfen mit Perfomance-Analyse Tools (z.B. Google Lighthouse) erzielen.

Verfügbarkeit [NFAFE ID-003]:

- Die Anwendung gewährleistet eine konsistente Nutzerfreundlichkeit durch benutzerfreundliche Fehlerseiten und angemessene Fehlermeldungen.

Sicherheit

- Nutzung von JWT (JSON Web Token) mit sicheren Ablaufmechanismen. **[NFAFE ID-004]**
- Formulare **[NFAFE ID-005]**:
    - Validierung der Eingaben zur Sicherstellung der Datenintegrität und Verhinderung von Injection-Angriffen.

- Implementierung eines «Honeypot»-Feldes zur Erkennung und Abwehr von Bot-Eingaben.
- Begrenzung der maximalen Anzahl an Anmeldeversuchen zur Prävention von Brute-Force-Angriffen. (optional)

1.1.2 Funktionale Anforderungen Frontend

In den funktionalen Anforderungen werden die konkreten Anwendungsfälle die das Frontend mit der dazu passenden Logik umsetzen soll definiert. Die jeweilige funktionale Anforderung enthält einen Verweis [**FAFE ID**] auf die genaue Spezifizierung, diese sind im Dokument **Spezifizierung_technische_Anforderungen_kh** Abschnitt 1.2 Spezifizierung Funktionale Anforderungen Frontend FAFE definiert.

Allgemeine Funktionen

- Navigation innerhalb der Webanwendung **[FAFE ID-001]**:
  - Bereitstellen einer Navigation für eine optimale Benutzerführung.

- Dienstleistungs-Übersicht und Suchfunktion:
  - Auflistung von Dienstleistungen mit relevanten Informationen (Titel, Beschreibung, Kategorie). **[FAFE ID-002]**
  - Volltextsuche & Filterung **[FAFE ID-003]**:
    - Implementierung einer leistungsfähigen Suchfunktion für gezielte Dienstleistungs-Suche.
    - Filteroptionen nach Kriterien: Kategorie, Ort
  - Anzeige Kontaktinformationen/öffentliches Profil **[FAFE ID-004]**:
    - Durch die Auswahl einer Dienstleistung erfolgt eine Weiterleitung zur Profilvorschau-Seite mit Kontaktinformationen des Dienstleisters.

- Registrierung und Authentifizierung **[FAFE ID-005]**:
  - Bereitstellung von nutzerfreundlichen Registrierungs- und Anmeldeformularen mit Validierung der Eingaben.
  - Ein Benutzer muss sich registrieren und anmelden können, um Zugriff auf sein persönliches Dashboard und die verfügbaren Funktionen der Plattform zu erhalten.

Funktionen für angemeldete Benutzer

- Dashboard & Verwaltung:
  - Bereitstellung einer zentralen Übersicht. **[FAFE ID-006]**
  - Verwaltung von Kontoeinstellungen und Profildaten. **[FAFE ID-007]**
- Dienstleistungs-Verwaltung **[FAFE ID-008]**:
  - Möglichkeit, eigene Dienstleistungen zu erstellen, bearbeiten und löschen (CRUD-Operationen).
- Öffentliches Profil **[FAFE ID-009]**:
  - Erstellung und Verwaltung eines öffentlichen Profils, das für andere Nutzer oder Interessenten sichtbar ist.

1.1.3 Nicht Funktionale Anforderungen Backend

In den nicht-funktionalen Anforderungen werden die Rahmenbedingungen des Backends sowie spezifische Merkmale zur Benutzerfreundlichkeit und Leistung definiert. Jede Anforderung enthält einen Verweis **[NFABE-ID]** auf die genaue Spezifizierung, diese sind im Dokument **Spezifizierung_technische_Anforderungen_kh** Abschnitt 1.3 Spezifizierung Nicht Funktionale Anforderungen Backend NFABE definiert.

Leistung [NFABE ID-001]

- Das System soll eine akzeptable Performance bereitstellen (Geschwindigkeit API – Anfragen)
- Datenbankabfragen sollen optimiert sein und mit minimaler Latenz ausgeführt werden.

Sicherheit [NFABE ID-002]

- Datenverschlüsselung: alle sensiblen Daten müssen bei der Speicherung verschlüsselt werden.
- Schutz vor Angriffen: Das System implementiert Schutzmechanismen vor gängigen Angriffsmethoden.
- Authentifizierung und Autorisierung: Das Backend muss eine sichere Authentifizierung implementieren.

1.1.4 Funktionale Anforderungen Backend

In den Funktionalen Anforderungen werden die konkreten Anwendungsfälle spezifiziert und deren Funktionsablauf beschrieben. Jede Anforderung enthält einen Verweis **[FABE-ID]** auf die genaue Spezifizierung, diese sind im Dokument **Spezifizierung_technische_Anforderungen_kh** Abschnitt 1.4 Spezifizierung Funktionale Anforderungen Backend FABE definiert.

Request-Verarbeitung [FABE ID-001]

- Entgegennahme und Bearbeitung eingehender Anfragen gemäss definierten API-Spezifikationen.

Datenverarbeitung und Konsistenz [FABE ID-002]

- Datenentgegennahme:
  - Validierung und Parsing eingehender Datenstrukturen.
- Datenspeicherung:
  - Persistente Speicherung in einer relationalen Datenbank.

API – Endpunkte

Bereitstellung von RESTful – API-Endpunkten für folgenden Entitäten:

- User – Management:
  - Authentifizierung **[FABE ID-003]**:
    - Sicherer Login-Prozess mit Passwort-Hashing und Token-basierter Authentifizierung JWT.
  - Registrierung **[FABE ID-004]**:
    - Erstellung neuer Benutzerkonten.
  - Kontoverwaltung (CRUD) **[FABE ID-005]**:
    - Bearbeitung, Aktualisierung und Löschung des Benutzerkontos.

- Dienstleistungen:
    - Abrufen aller verfügbaren Dienstleistungen. **[FABE ID-006]**
    - Verwaltung von Dienstleistungs – Daten mittels CRUD-Operationen (Create, Read, Update, Delete). **[FABE ID-007]**

- Öffentliches Profil **[FABE ID-008]**:
    - Verwaltung des öffentlichen Profils mittels CRUD-Operationen (Create, Read, Update, Delete).

1.1.5 API-Spezifikationen Backend

Die API-Spezifikation wurde basierend auf den funktionalen Anforderungen im Dokument **"Spezifizierung technische Anforderungen"** definiert. Die erste API-Spezifikation dient als Vorlage für die initiale Implementierung und stellt eine prototypische Spezifikation dar. Sie orientiert sich an den vorgegebenen Designmustern und wird entsprechend des Modulunterrichts optimiert.

Allgemeine API-Prinzipien:

- **Format:** JSON
- **Authentifizierung:** Bearer Token mit JWT
- **Statuscodes:** Standard-HTTP-Statuscodes zur Fehler- und Erfolgsmeldung
- **Versionierung:** `/api/v1/`

1.1.5.1 API-Endpunkte

Dienstleistungen:
- **GET /api/v1/services/all/** – Alle Dienstleistungen abrufen
- **POST/api/v1/services** – Dienstleistung erstellen
- **PUT /api/v1/services/{id}/** – Dienstleistung aktualisieren
- **DELETE /api/v1/services/{id}/** – Dienstleistung löschen

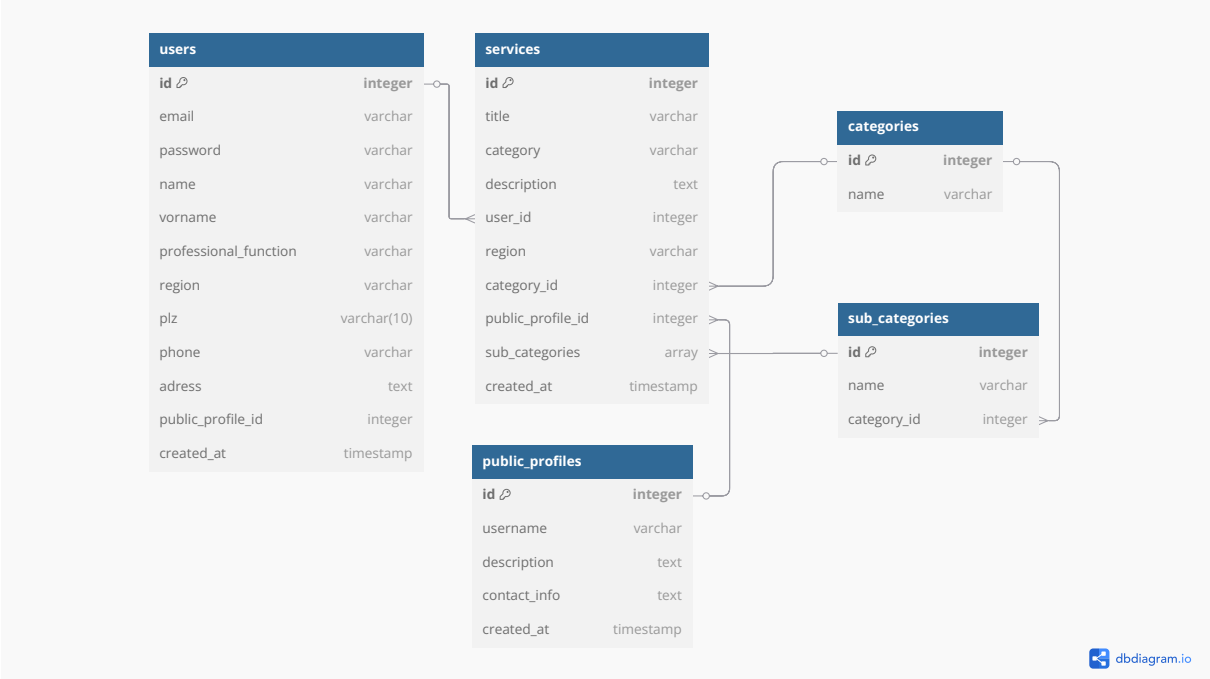
Usermanagement:
- **POST /api/v1/auth/register/** – Registriert einen neuen Benutzer
- **POST /api/v1/auth/login/** – Meldet einen Benutzer an und gibt ein JWT-Token zurück
- **POST /api/v1/auth/logout/** – Meldet den Benutzer ab und invalidiert das Token

Kontoverwaltung
- **POST /api/v1/users** – Konto erstellen
- **PUT /api/v1/users /{id}/** – Konto aktualisieren
- **DELETE /api/v1/users /{id}/** – Konto löschen
- **POST /api/v1/users /{id}/change-password/** – Passwort ändern
- **PUT /api/v1/users /{id}/credentials/** – Anmeldeinformationen aktualisieren

Profilverwaltung:
- **GET /api/v1/profiles/{id}/** – Profil abrufen
- **PUT /api/v1/profiles/{id}/** – Profil aktualisieren

1.1.6 Datenbank Spezifikationen Backend

Das folgende Diagramm beschreibt die Struktur und Spezifikation der relationalen Datenbank des Backend-Systems. Es dient der prototypischen Implementierung und soll gemäss dem agilen Entwicklungsprozess angepasst und optimiert werden.



2 Übersicht Technologien

Der Technologie Stack wurde auf Basis des Modulunterrichts definiert.

2.1 Architektur

Für die Umsetzung des Projekts habe ich mich bewusst für eine Trennung von Frontend und Backend entschieden. Diese Entscheidung basiert auf der Zielsetzung, eine konsistente und modulare Architektur der Applikation zu gewährleisten. Zudem wurde die Annahme berücksichtigt, dass die Anwendung in Zukunft weiterentwickelt wird, etwa durch die Entwicklung einer mobilen App mit React Native (dieser Aspekt ist jedoch nicht Teil der aktuellen Semesterarbeit).

Die Anwendung wird mit **Client-Side Rendering (CSR)** entwickelt. Diese Wahl habe ich getroffen, mit der Aussicht, dass die Komplexität der Anwendung in späteren Entwicklungsstadien deutlich zunimmt.

2.2 Frontend - Stack

- **Framework**: React.js
- **Programmiersprache**: JavaScript(ES6+)
- **Styling**: CSS / SCSS
- **State Management**: Redux oder Zustand (Präferenz gemäss Unterricht)
- **UI– Bibliothek**: Material-UI (MUI)
- **Build-Tool**: Vite
- **Testing**: Präferenz gemäss Unterricht (Unittests, Vitest, React Testing Libary)

2.3 Backend – Stack

- **Programmiersprache**: Python
- **Framework**: Django
- **Datenbank**: MariaDB / MySQL (Präferenz gemäss Unterricht)
- **API-Schnittstelle**: Restful API
- **Authentifizierung**: JWT (JSON Web Token)
- **Testing**: Präferenz gemäss Modulunterricht (Unittests, Pytest)

2.4 Infrastruktur & Deployment

- **Versionskontrolle**: Git / Gitlab
- **Containerisierung**: Docker mit Docker Compose (optional)

3 Zeitmanagement

Ich habe das Zeitmanagement unter Berücksichtigung des Modulunterrichts definiert.
Besonders habe ich bei der Planung Reserve Zeit eingeplant. Jede Phase wird im Dokument: **Semesterarbeit_Logbuch_kh** festgehalten.

3.1 Zeitmanagement Frontend

Phase	Aufgabe	Startdatum	Dauer	Logbuch/ID
Planung & Vorbereitung	Erstellen der Vorgaben / Spezifizierung	08.02.25	4 Wochen	1.1
Setup & Grundstruktur	Initialisierung des Projekts (Git, Basisstruktur)	22.02.25	2 Wochen	
	Einarbeiten UI Libary / Theme erarbeiten	22.02.25	2 Wochen	
UI/UX-Entwicklung	Umsetzung der Kernansichten gemäß Mockups	14.03.25	2 Wochen	
Interaktive Funktionen & Logik	Implementierung der UI-Logik mit Testdaten	28.03.25	4 Wochen	
Testing & Optimierung	UI-Tests (Vitest, React Testing Library) & Refactoring	02.05.25	TDD / 4 Wochen	
Finalisierung & Backend-Anbindung	Integration mit Backend & API-Anbindung	02.05.25	4 Wochen	
	Präsentation	07.06.25		
	Reserve – für finale Anpassungen & Abgabe	07.06.25	2 Wochen	

3.2 Zeitmanagement Backend

Phase	Aufgabe	Startdatum	Dauer	Logbuch/ID
Planung & Vorbereitung	Erstellen der Vorgaben / Spezifizierung	08.02.25	4 Wochen	1.1
Setup & Grundstruktur	Initialisierung des Projekts (Git, Basisstruktur)	08.03.25	1 Woche	
Datenbank & API-Design	Datenbankmodellierung & Migrationen (ORM, Relationsstrukturen)	15.03.25	4 Wochen	
API-Entwicklung	Implementierung der API-Endpunkte & Geschäftslogik	12.04.25	4 Wochen	
Sicherheit & Authentifizierung	Implementierung von JWT (Authentifizierung & Autorisierung)	10.05.25	2 Wochen	
Admin-Panel (optional)	Konfiguration des Admin-Panels			
Testing & Refactoring	Testen der API-Endpunkte & Refactoring	24.05.25	2 Wochen	
Finalisierung & Frontend-Integration	Frontend-Integration & Feinabstimmung	24.05.25	2 Wochen	
	Präsentation	07.06.25		
	Reserve – für finale Anpassungen & Abgabe	07.06.25	2 Wochen	

4 Wireframes & Mockup

Die Wireframes wurden mit Figma umgesetzt, dabei ist zu erwähnen, dass, das Mockup eher für den visuellen Aspekt und den Vorgaben des Moduls entsprechend umgesetzt wurde. Ich beziehe mich hier auf den Prozess der agilen Entwicklungsmethode besonders bei der Gestaltung und Design wähle ich einen möglichst flexiblen Ansatz um mich optimal der verwendeten UI Libary anzupassen.

https://www.figma.com/design/TEmmnTdPA9ZjcmP3sihkc1/Sidelink

5 Logbuch

Für eine konsistente und übersichtliche Logbuchführung wurde ein separates Dokument erstellt. **Semsesterarbeit_Logbuch_kh**

6 Reflexion

In diesem Projekt werden die meisten Themen abgedeckt, die in diesem Modul behandelt werden. Die Idee einer Dienstleistungsplattform bietet zahlreiche Möglichkeiten zur Verfeinerung und Erweiterung – beispielsweise durch ein zusätzliches Board zur Fachkraftsuche. Dennoch habe ich mich bewusst auf die essenziellen MUSS-Anforderungen konzentriert, um den Fokus der Spezifikation klar zu definieren und den Zeitplan einhalten zu können. Ich freue mich auf die praktische Umsetzung des Projekts und bin gespannt auf das Feedback der Lehrpersonen.

Quellenverzeichnis

- Umsetzung Wireframes & Mockup (Design Elemente als Vorlage): https://modernize-react-dark.netlify.app/dashboards/ecommerce
- Umsetzung Wireframes & Mockup generell: https://mui.com/?srsltid=AfmBOorVyUHTYQ89sF24XxX7p28YZZNYl2YpSHVpUFg7Z1wvQjnoHJYf

Anhang

- Dokument: Spezifizierung_technische_Anforderungen_kh
- Dokument: Semesterarbeit_Logbuch_kh

FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Spezifizierung technische Anforderungen

Der Semesterarbeit des Studiengangs CAS Fullstack-Development

von

Kay Roman Hertenstein

Roggwil, Januar – Juni 2025

Inhaltsverzeichnis

1 Technische Spezifikationen

Spezifizierung der Anforderungen der Semesterarbeit ‘SideLink’. Diese Anforderungen beziehen sich auf das Dokument Semesterarbeit_KayHertenstein, Abschnitt 1.1. Anwendungsumfang.
Die Spezifizierungen wurden gemäss Vorgabe des Moduls FSDev, Full-Stack Development, W4B-C-FS001, ZH-Sa-1, FS25 Abschnitt «Projektarbeit – Erwartung an Projektdokument» https://moodle.ffhs.ch/mod/page/view.php?id=4886905 erstellt.
In der technischen Spezifizierung werden folgende Kurzschreibweisen verwendet:
NFAFE: Nicht Funktionale Anforderungen Front-End
FAFE: Funktionale Anforderungen Front-End
NFABE: Nicht Funktionale Anforderungen Back-End
FABE: Funktionale Anforderungen Back-End

1.1 Spezifizierung Nicht Funktionale Anforderungen Frontend NFAFE

Nicht Funktionale Anforderungen Frontend.

1.1.1 NFAFE ID-001

Ziel

Die Webanwendung soll auf verschiedenen Endgeräten (Desktop, Tablet, Smartphone) optimal dargestellt werden. Die Benutzeroberfläche passt sich dynamisch an verschiedene Bildschirmgrößen an, sodass die Nutzung auf allen Geräten einwandfrei funktioniert.

Ereignis

Ein Benutzer öffnet die Webanwendung auf einem beliebigen Endgerät oder ändert die Größe des Browserfensters.

Vorbedingungen

- Die Webanwendung ist geladen.
- Der Benutzer nutzt ein internetfähiges Endgerät (z. B. PC, Tablet oder Smartphone).
- Die Benutzeroberfläche passt sich dynamisch an die Bildschirmgröße an.

Standardablauf

1. Der Benutzer öffnet die Webanwendung auf einem Endgerät.
2. Das Layout und die Inhalte passen sich automatisch an die Bildschirmgröße an.
3. Die Navigation wird je nach Gerätetyp angepasst (z. B. als Hamburger-Menü auf mobilen Geräten).
4. Bilder und Texte werden in einer skalierbaren Größe dargestellt.
5. Interaktive Elemente (z. B. Buttons, Formulare) bleiben nutzbar und gut erreichbar.

Alternativablauf

- Falls der Benutzer auf einem sehr kleinen oder ungewöhnlichen Displayformat unterwegs ist, kann es zu Darstellungsalternativen kommen.

- Falls Medien (z. B. Bilder oder Videos) nicht korrekt skaliert werden, kann die b.P. die Seite neu laden.

Nachbedingungen Erfolg

- Die Webanwendung wird auf allen unterstützten Geräten korrekt angezeigt.
- Die Navigation und alle Funktionen bleiben voll nutzbar.

Nachbedingungen Fehler

- Falls das Layout nicht korrekt dargestellt wird, kann der Benutzer ggf. eine eingeschränkte Benutzererfahrung haben.

Klassifizierung

Nicht-Funktional, MUSS

Aufwand

MITTEL

1.1.2 NFAFE ID-002

Ziel

Die Webanwendung soll eine hohe Performance gewährleisten, sodass Ladezeiten minimiert und Nutzerinteraktionen ohne Verzögerungen erfolgen. Die Anwendung muss in Performance-Analyse-Tools (z. B. Google Lighthouse) ein akzeptables Ergebnis erzielen, um eine optimale Nutzererfahrung sicherzustellen.

Ereignis

Ein Benutzer öffnet die Webanwendung oder interagiert mit dieser (z. B. durch Navigieren zwischen Seiten, Laden von Inhalten oder Absenden eines Formulars).

Vorbedingungen

- Die Webanwendung ist geladen.
- Der Benutzer nutzt ein internetfähiges Endgerät (z. B. PC, Tablet oder Smartphone).
- Der Server ist erreichbar und reagiert auf Anfragen.
- Die Anwendung verwendet optimierte Ressourcen (minimierte CSS/JS-Dateien).

Standardablauf

1. Der Benutzer öffnet die Webanwendung.
2. Die Anwendung lädt die Inhalte innerhalb einer akzeptablen Zeit.
3. Unnötige Ressourcen werden nicht geladen (durch Lazy Loading).
4. Clientseitiges Caching optimieren die Ladegeschwindigkeit.
5. Hintergrundprozesse und API-Anfragen werden asynchron verarbeitet, um die Reaktionszeit der Anwendung zu verbessern.

6. Die Anwendung erreicht in Google Lighthouse eine Mindestpunktzahl in der Kategorie "Performance".

Alternativablauf

- Falls die Ladezeit über dem definierten Schwellenwert liegt, wird ein Ladeindikator angezeigt.
- Falls eine externe Ressource nicht verfügbar ist, wird eine alternative Darstellung verwendet oder ein Fallback-Mechanismus aktiviert.

Nachbedingungen Erfolg

- Die Anwendung lädt innerhalb der definierten Zeitvorgaben und bleibt reaktionsschnell.
- Performance-Analyse-Tools (z. B. Google Lighthouse) zeigen ein akzeptables Ergebnis in der Performance-Bewertung.

Nachbedingungen Fehler

- Falls die Anwendung nicht performant geladen wird, kann es zu längeren Wartezeiten oder eingeschränkter Benutzererfahrung kommen.
- Falls die Performance-Analyse ein unzureichendes Ergebnis liefert, müssen Optimierungsmaßnahmen ergriffen werden.

Klassifizierung

Nicht-Funktional, MUSS

Aufwand

MITTEL

1.1.3 NFAFE ID-003

Ziel

Die Webanwendung soll eine hohe Verfügbarkeit und eine konsistente Nutzererfahrung gewährleisten, auch im Falle von Fehlern. Benutzerfreundliche Fehlerseiten und angemessene Fehlermeldungen für Fehler wie 404 (Seite nicht gefunden) und 500 (Serverfehler).

Ereignis

Ein Benutzer ruft eine Seite auf, die nicht existiert (z. B. 404-Fehler), oder es tritt ein Serverfehler auf (z. B. 500-Fehler).

Vorbedingung

- Die Webanwendung ist verfügbar und geladen.
- Der Benutzer nutzt ein internetfähiges Endgerät (z. B. PC, Tablet oder Smartphone).

Standardablauf

1. Der Benutzer navigiert zu einer Seite der Webanwendung.
2. Falls der Benutzer eine ungültige URL eingibt oder eine nicht existierende Seite aufruft, wird eine benutzerfreundliche **404-Fehlerseite** angezeigt.
3. Die Fehlermeldungen sind leicht verständlich und enthalten Empfehlungen zur Weiterbehandlung (z. B. "Zurück zur Startseite").
4. Alle Fehlerseiten sollten weiterhin die grundlegende Navigation und Layout-Elemente enthalten, um eine konsistente Nutzererfahrung zu gewährleisten.

Alternativablauf

- Bei Serverüberlastung oder bei einem langsamen Netzwerk wird dem Benutzer eine Ladeanzeige oder ein Hinweis zur Verfügbarkeit der Anwendung.

Nachbedingungen Erfolg

- Die Anwendung zeigt immer eine konsistente und benutzerfreundliche Fehlerseite an, auch im Falle von 404 oder 500 Fehlern.
- Fehlermeldungen sind klar formuliert und ermöglichen es dem Benutzer, die Anwendung weiter zu nutzen oder geeignete Maßnahmen zu ergreifen.

Nachbedingung Fehler

- Spezifikation dient der Fehlerbehandlung.

Klassifizierung
Nicht-Funktional, MUSS

Aufwand
GERING bis MITTEL

1.1.4 NFAFE ID-004

Ziel
Die Webanwendung soll JWT (JSON Web Token) für die Authentifizierung von Benutzern nutzen und sicherstellen, dass Ablaufmechanismen für die Tokens implementiert sind, um eine sichere Verwaltung der Benutzersitzungen zu gewährleisten.

Ereignis
Ein Benutzer loggt sich erfolgreich in die Webanwendung ein, und ein JWT wird erstellt und für die Benutzerauthentifizierung verwendet.

Vorbedingung

- Der Benutzer hat ein gültiges Benutzerkonto in der Anwendung.
- Der Benutzer hat seine Anmeldedaten erfolgreich eingegeben.

Standardablauf

1. Der Benutzer gibt seine Anmeldedaten ein und sendet diese an den Server.
2. Der Server überprüft die Anmeldedaten und erstellt ein JWT mit den erforderlichen Benutzerinformationen
3. Das JWT wird an den Benutzer zurückgegeben und im SessionStorage des Browsers gespeichert. Falls erfolgreich, wird das Formular verarbeitet und die Aktion abgeschlossen.
4. Bei jeder weiteren Anfrage des Benutzers an die Webanwendung wird das JWT im Authorization-Header (Bearer Token) der Anfrage übermittelt.
5. Falls das JWT abgelaufen ist, wird der Benutzer aufgefordert, sich erneut anzumelden.

Alternativablauf

- Wenn das JWT abgelaufen ist wird der Benutzer zur Login-Seite weitergeleitet und aufgefordert, sich erneut anzumelden.
- Wenn der Benutzer die Sitzung verlässt oder sich ausloggt, wird das JWT aus dem SessionStorage gelöscht, um den Zugriff zu verhindern.

Nachbedingungen Erfolg

- Das JWT wird korrekt generiert und genutzt, um sicherzustellen, dass nur authentifizierte Benutzer auf die Webanwendung zugreifen können.
- Ablaufmechanismen garantieren, dass abgelaufene Tokens nicht verwendet werden können und eine erneute Authentifizierung erforderlich ist.

Nachbedingung Fehler

- Falls das JWT nicht korrekt generiert oder überprüft wird, könnte der Benutzer keinen Zugriff auf die Anwendung erhalten

Klassifizierung

Nicht-Funktional, MUSS

Aufwand

MITTEL - HOCH

1.1.5 NFAFE ID-005

Ziel
Sicherstellung der Integrität und Sicherheit von Formulareingaben durch Validierung, Schutzmechanismen gegen automatisierte Angriffe.

Ereignis
Ein Benutzer füllt ein Formular aus und sendet es ab (Login, Registrierung, Kontoinformationen).

Vorbedingung

- Die benutzende Person befindet sich auf einer Seite mit einem Formular.
- Die benutzende Person gibt Daten in das Formular ein und bestätigt die Eingabe.

Standardablauf

6. Der Benutzer füllt das Formular mit den erforderlichen Informationen aus.
7. Das System führt eine clientseitige Validierung der Eingaben durch (z. B. Pflichtfelder, Zeichenbeschränkungen, Formatprüfung).
8. Das System sendet die eingegebenen Daten an das Backend.
9. Falls erfolgreich, wird das Formular verarbeitet und die Aktion abgeschlossen.

Alternativablauf

- Ungültige Eingabe: Der Benutzer erhält eine Fehlermeldung zur Korrektur der Eingaben.
- Erkennung eines Bot-Angriffs durch das Honeypot-Feld: Die Anfrage wird blockiert, ohne eine sichtbare Reaktion für den Benutzer.

Nachbedingungen Erfolg

- Das Formular wird erfolgreich verarbeitet und die beabsichtigte Aktion durchgeführt.

Nachbedingung Fehler

- Der Benutzer erhält eine Fehlermeldung und kann das Formular korrigieren.

Klassifizierung

Nicht-Funktional, MUSS

Aufwand

GERING bis MITTEL

1.2 Spezifizierung Funktionale Anforderungen Frontend FAFE

Funktionale Anforderungen Frontend.

1.2.1 FAFE ID-001

Ziel

Die Webanwendung soll eine benutzerfreundliche und intuitive Navigation bereitstellen, die eine optimale Benutzerführung ermöglicht. Die Menüführung soll auf verschiedenen Bildschirmgrößen und Endgeräten (Desktop, Tablet, Smartphone) reibungslos funktionieren.

Ereignis

Der Benutzer nutzt die Navigation, um zwischen verschiedenen Bereichen der Webanwendung zu wechseln.

Vorbedingungen

- Die Webanwendung ist geladen.
- Die Navigationselemente sind sichtbar und zugänglich.
- Der Benutzer kann durch Klicken oder Tippen mit der Navigation interagieren.

Standardablauf

1. Der Benutzer öffnet die Webanwendung.
2. Der Benutzer sieht die Hauptnavigation, die abhängig vom Endgerät entweder als horizontales Menü oder als Hamburger-Menü angezeigt wird.
3. Der Benutzer klickt auf ein Navigations-Element (z. B. „Board“, „Login“, „Dashboard“).
4. Der Benutzer wird zur entsprechenden Seite weitergeleitet.
5. Der Benutzer kann die Navigation jederzeit nutzen, um zwischen Seiten zu wechseln.

Alternativablauf

- Falls der Benutzer eine nicht existente Seite aufruft, wird eine „404 – Seite nicht gefunden“-Meldung angezeigt.
- Falls die Navigation auf einem kleineren Bildschirm nicht korrekt angezeigt wird, kann der Benutzer versuchen, die Seite neu zu laden oder auf die Desktop-Version zu wechseln.

Nachbedingungen Erfolg

- Der Benutzer kann sich problemlos innerhalb der Webanwendung bewegen.
- Der Benutzer erreicht die gewünschte Seite durch Nutzung der Navigation.

Nachbedingungen Fehler

- Falls die Navigation fehlschlägt, bleibt der Benutzer auf der aktuellen Seite.

Klassifizierung

Funktional, MUSS

Aufwand

GERING bis MITTEL

1.2.2 FAFE ID-002

Ziel
Ein Benutzer soll eine übersichtliche Liste von Dienstleistungen mit relevanten Informationen (Titel, Beschreibung, Kategorie) einsehen können.

Ereignis
Der Benutzer öffnet die Webseite oder navigiert zur Dienstleistungs-Übersicht (Board).

Vorbedingungen

- Die Webanwendung ist geladen.
- Der Benutzer befindet sich auf der Dienstleistungs-Übersichtsseite.

Standardablauf

1. Der Benutzer navigiert zur Dienstleistungs-Übersicht.
2. Das System ruft verfügbare Dienstleistungs-Angebote über die API ab.
3. Die Angebote werden in einer Liste angezeigt, inklusive:
    - Titel
    - Beschreibung
    - Datum
4. Der Benutzer kann durch die Angebote paginieren und einzelne Angebote aufrufen.

Alternativablauf

- Es sind keine Job-Angebote vorhanden: Eine Meldung wird angezeigt ("Derzeit keine Dienstleistungen verfügbar").
- Ein technischer Fehler verhindert das Laden der Angebote: Eine Fehlermeldung erscheint und ein erneuter Ladeversuch wird angeboten.

Nachbedingungen Erfolg

- Die Angebote werden korrekt geladen und dem Benutzer angezeigt.

Nachbedingungen Fehler

- Der Benutzer erhält eine Fehlermeldung, falls kein Inhalt geladen werden kann.

Klassifizierung

Funktional, MUSS

Aufwand

GERING

1.2.3 FAFE ID-003

Ziel
Ein Benutzer soll eine leistungsfähige Such- und Filterfunktion nutzen können, um gezielt nach Dienstleistungen zu suchen.

Ereignis
Der Benutzer gibt einen Suchbegriff in das Suchfeld ein oder wählt eine Filteroption aus.

Vorbedingungen

- Der Benutzer befindet sich auf der Dienstleistungs-Übersichtsseite.
- Es existieren bereits Dienstleistungsangebote in der Datenbank und das System hat die Angebote via API geladen.

Standardablauf

1. Der Benutzer gibt einen Suchbegriff in das Suchfeld ein oder wählt eine Filteroption (Kategorie, Ort) aus.
2. Die Geschäftslogik (Frontend) verarbeitet die Anfrage und liefert relevante Ergebnisse zurück.
3. Die Ergebnisse werden dem Benutzer dynamisch in der Übersicht angezeigt.

Alternativablauf

- **Keine passenden Suchergebnisse:** Eine Meldung wird angezeigt ("Keine Ergebnisse gefunden").

Nachbedingungen Erfolg

- Der Benutzer erhält eine gefilterte Liste relevanter Dienstleistungsangebote.

Nachbedingungen Fehler

- Der Benutzer erhält eine Fehlermeldung oder eine Meldung, dass keine passenden Ergebnisse gefunden wurden.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL bis HOCH

1.2.4 FAFE ID-004

Ziel
Durch die Auswahl einer Dienstleistung soll der Benutzer zur Profilvorschau-Seite des jeweiligen Dienstleisters weitergeleitet werden, auf der die relevanten Kontaktinformationen und öffentlich sichtbaren Profildetails angezeigt werden.

Ereignis
Die benutzende Person wählt eine Dienstleistung aus der Übersicht aus.

Vorbedingung

- Die benutzende Person befindet sich auf der Dienstleistungsseite.
- Die Dienstleistung ist einem Dienstleister mit öffentlichem Profil zugeordnet.

Standardablauf

1. Der Benutzer wählt eine Dienstleistung aus der Übersicht aus.
2. Das System sendet eine Anfrage zur Abrufung der Profildaten des Dienstleisters.
3. Das System ermittelt die zugehörigen Profildaten und sendet sie an das Frontend.
4. Der Benutzer wird auf die Vorschau-Seite weitergeleitet, wo das öffentliche Profil des Dienstleisters mit der ausgewählten Dienstleistung angezeigt wird.

Alternativablauf

- Kein öffentliches Profil vorhanden: Der Benutzer erhält eine Meldung, dass kein öffentliches Profil für diesen Dienstleister verfügbar ist.

Nachbedingungen Erfolg

- Der Benutzer wird zur Vorschau-Seite weitergeleitet und kann die relevanten Informationen einsehen.

Nachbedingung Fehler

- Der Benutzer bleibt auf der aktuellen Seite und erhält eine Fehlermeldung.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

1.2.5 FAFE ID-005

Ziel
Ein Benutzer soll sich registrieren und anmelden können, um personalisierte Funktionen der Plattform zu nutzen. Dabei werden alle Eingaben validiert.

Ereignis
Der Benutzer öffnet das Registrierungs- oder Anmeldeformular und gibt die erforderlichen Daten ein.

Vorbedingungen

- Der Benutzer befindet sich auf der Registrierungs- oder Login-Seite.
- Die erforderlichen Eingabefelder sind vorhanden.
- Das Backend ist verfügbar.

Standardablauf

1. Der Benutzer öffnet das Registrierungs- oder Login-Formular.
2. Der Benutzer gibt seine Daten (E-Mail, Passwort oder Registrationsdaten) in das Formular ein.
3. Das System überprüft die Eingaben auf Korrektheit (gültige E-Mail-Adresse, Passwortstärke).
4. Nach erfolgreicher Validierung werden die Daten an das Backend gesendet.
5. Das Backend verarbeitet die Anfrage und gibt eine Bestätigung zurück.
6. Der Benutzer wird angemeldet und weitergeleitet auf das Dashboard oder erhält einen Link zum Login (neue Registration).

Alternativablauf

- **Ungültige Eingaben:** Das System zeigt eine adäquate Fehlermeldung (z. B. "Ungültige E-Mail-Adresse", "Passwort zu schwach").
- **E-Mail bereits registriert:** Das System gibt eine Meldung aus ("E-Mail-Adresse ist bereits vergeben").
- **Netzwerk-/Serverfehler:** Eine Fehlermeldung wird angezeigt.

Nachbedingungen Erfolg

- Der Benutzer wird erfolgreich registriert oder angemeldet.

Nachbedingungen Fehler

- Der Benutzer erhält eine entsprechende Fehlermeldung und kann seine Eingaben korrigieren.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

1.2.6 FAFE ID-006

Ziel
Ein angemeldeter Benutzer soll eine zentrale Übersicht über seine gespeicherten Daten und relevanten Funktionen der Plattform erhalten.

Ereignis
Der Benutzer meldet sich an und navigiert zu seinem Dashboard.

Vorbedingungen

- Der Benutzer ist erfolgreich eingeloggt.
- Eine Verbindung zum Backend ist verfügbar.
- Die relevanten Daten (Kontodaten, Profildaten, Dienstleistungsangebote) sind im System gespeichert.

Standardablauf

1. Der Benutzer meldet sich an und wird auf sein Dashboard weitergeleitet.
2. Das Dashboard lädt die persönlichen Daten des Benutzers.
3. Der Benutzer sieht eine Übersicht über seine erstellten Dienstleistungsangebote, Kontodaten und weitere relevante Informationen.
4. Der Benutzer kann über das Dashboard verschiedene Funktionen aufrufen (Kontoeinstellungen, Dienstleistungsangebote verwalten, Profil bearbeiten).

Alternativablauf

- **Fehlende Daten:** Falls keine Daten vorhanden sind, zeigt das System eine leere Übersicht mit einer Aufforderung zum Erstellen neuer Inhalte.
- **Serverfehler:** Falls das Dashboard nicht geladen werden kann, erscheint eine Fehlermeldung mit der Möglichkeit, die Seite neu zu laden.

Nachbedingungen Erfolg

- Das Dashboard wird erfolgreich geladen und zeigt aktuelle Benutzerinformationen an.
- Der Benutzer kann alle relevanten Funktionen direkt vom Dashboard aus erreichen.

Nachbedingungen Fehler

- Der Benutzer erhält eine Fehlermeldung und kann das Dashboard nicht nutzen.
- Falls das Problem beim Laden der Daten liegt, kann der Benutzer die Seite neu laden.

Klassifizierung

Funktional, MUSS

Aufwand

HOCH

1.2.7 FAFE ID-007

Ziel
Ein angemeldeter Benutzer soll seine Kontoeinstellungen und Profildaten verwalten, bearbeiten und speichern können.

Ereignis
Der Benutzer navigiert in seinem Dashboard zu den Kontoeinstellungen oder Profildaten und nimmt Änderungen vor.

Vorbedingungen

- Der Benutzer ist erfolgreich eingeloggt.
- Die Verbindung zum Backend ist vorhanden.
- Die aktuellen Kontoeinstellungen und Profildaten sind im System gespeichert.

Standardablauf

1. Der Benutzer meldet sich an und öffnet die Seite für Kontoeinstellungen
2. Das System lädt die aktuellen Einstellungen und zeigt sie in einem Formular an.
3. Der Benutzer nimmt Änderungen an den Daten vor (z. B. Name, E-Mail-Adresse, Passwort).
4. Der Benutzer speichert die Änderungen durch Klicken auf "Speichern".
5. Das System validiert die Eingaben und sendet die aktualisierten Daten an das Backend.
6. Das Backend verarbeitet die Daten, speichert sie und gibt eine Bestätigung zurück.
7. Der Benutzer erhält eine Erfolgsmeldung und sieht die aktualisierten Informationen.

Alternativablauf

- **Ungültige Eingaben:** Falls fehlerhafte Daten eingegeben werden (z. B. falsches E-Mail-Format, zu kurzes Passwort), wird eine entsprechende Fehlermeldung angezeigt.
- **Serverfehler:** Falls die Änderungen nicht gespeichert werden können, erhält der Benutzer eine Fehlermeldung mit der Möglichkeit, es erneut zu versuchen.
- **Abbruch:** Falls der Benutzer die Seite verlässt, bevor er speichert, bleiben die ursprünglichen Daten unverändert.

Nachbedingungen Erfolg

- Die aktualisierten Kontoeinstellungen werden erfolgreich gespeichert und angezeigt.
- Der Benutzer sieht eine Bestätigung der erfolgreichen Aktualisierung.

Nachbedingungen Fehler

- Der Benutzer erhält eine Fehlermeldung und kann die Eingaben korrigieren oder es später erneut versuchen.
- Falls der Fehler serverseitig auftritt, bleibt die ursprüngliche Version der Daten erhalten.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

1.2.8 FAFE ID-008

Ziel

Ein Benutzer kann eigene Dienstleistungen erstellen, bearbeiten und löschen (CRUD-Operationen) im Dashboard.

Ereignis

Die benutzende Person hat eine Dienstleistung erstellt, bearbeitet oder gelöscht und bestätigt die Aktion im Dashboard.

Vorbedingung

- Die benutzende Person befindet sich im Dashboard-Bereich „Dienstleistungen“.
- Im Fall von Bearbeiten oder Löschen: Die Dienstleistung existiert bereits.

Standardablauf

1. Die benutzende Person navigiert zum Bereich „Dienstleistungen“ im Dashboard.
2. Die benutzende Person klickt auf den Button „Dienstleistung erstellen“ und gibt die erforderlichen Details wie Beschreibung, Kategorie, Ort und Titel an.
3. Die benutzende Person klickt auf den Button „Speichern“, um die neue Dienstleistung zu erstellen.
4. Die benutzende Person klickt auf eine bestehende Dienstleistung und bearbeitet die Details (z.B. Titel, Beschreibung).
5. Die benutzende Person klickt auf den Button „Speichern“, um die bearbeitete Dienstleistung zu speichern.
6. Die benutzende Person klickt auf den Button „Löschen“ einer Dienstleistung, und eine Bestätigungsabfrage erscheint.

7. Nach Bestätigung der Löschung wird die Dienstleistung gelöscht und nicht mehr angezeigt.
8. Eine Erfolgsmeldung wird angezeigt, die die Durchführung der Aktion bestätigt (Erstellung, Bearbeitung oder Löschung).

Alternativablauf

- **Fehlende Eingaben:** Die benutzende Person erhält eine Fehlermeldung, wenn nicht alle erforderlichen Felder ausgefüllt wurden (z.B. Name der Dienstleistung).
- **Fehler bei der Löschung:** Wenn die Dienstleistung nicht gelöscht werden kann, erhält die benutzende Person eine Fehlermeldung.

Nachbedingungen Erfolg

- Die benutzende Person erhält eine Erfolgsmeldung.
- Die Dienstleistung wird im Dashboard korrekt angezeigt, wenn sie erstellt wurde.
- Die bearbeitete Dienstleistung zeigt die neuen Details.
- Die gelöschte Dienstleistung ist nicht mehr im Dashboard sichtbar.

Nachbedingung Fehler

- Die benutzende Person erhält eine Fehlermeldung und kann die Aktion ggf. korrigieren.
- Die Dienstleistung bleibt unverändert (bei Bearbeitungs- oder Löschfehlern).

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

1.2.9 FAFE ID-009

Ziel

Ein Benutzer kann ein öffentliches Profil erstellen und verwalten, das für andere Nutzer oder Interessenten sichtbar ist.

Ereignis

Die benutzende Person hat ein öffentliches Profil erstellt oder bearbeitet und speichert die Änderungen im System.

Vorbedingung

- Die benutzende Person befindet sich im Bereich „Profil“ des Dashboards.
- Die benutzende Person hat ein Benutzerkonto und ist angemeldet.

Standardablauf

Beschreibung des Ablaufs bei Erfüllung der Vorbedingungen:

1. Die benutzende Person navigiert zum Bereich „Öffentliches Profil“ im Dashboard.
2. Die benutzende Person klickt auf den Button „Profil erstellen“ oder „Profil bearbeiten“.
3. Die benutzende Person gibt die erforderlichen Informationen ein, z.B.:
    - Angezeigter Name
    - Berufliche Informationen oder Kurzbeschreibung
    - Soziale Medien oder Kontaktmöglichkeiten
4. Die benutzende Person klickt auf den Button „Speichern“, um das öffentliche Profil zu erstellen oder die Änderungen zu speichern.
5. Das Profil wird für andere Nutzer oder Interessenten sichtbar und kann über eine dem Benutzer zugewiesene Dienstleistung aufgerufen werden.
6. Eine Erfolgsmeldung wird angezeigt, die die erfolgreiche Erstellung oder Bearbeitung des Profils bestätigt.

Alternativablauf

Beschreibung, wenn die Aktion nicht ausgeführt werden kann:

- **Fehlende Eingaben:** Die benutzende Person erhält eine Fehlermeldung, wenn nicht alle erforderlichen Felder ausgefüllt wurden (z.B. Name oder Kontaktmöglichkeiten).
- **Fehler beim Speichern:** Falls es ein Problem beim Speichern des Profils gibt, erhält die benutzende Person eine Fehlermeldung.

Nachbedingungen Erfolg

- Die benutzende Person erhält eine Erfolgsmeldung.
- Das öffentliche Profil ist nun für andere Nutzer sichtbar und kann über eine dem Benutzer zugewiesene Dienstleistung aufgerufen werden.

Nachbedingung Fehler

- Die benutzende Person erhält eine Fehlermeldung und kann die Eingaben korrigieren.
- Das Profil bleibt unverändert, bis die Fehler behoben sind.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

1.3 Spezifizierung Nicht Funktionale Anforderungen Backend NFABE

Nicht Funktionale Anforderungen Backend.

1.3.1 NFABE ID-001

Ziel

Das System soll eine akzeptable Performance bereitstellen, indem es schnelle Antwortzeiten für API-Anfragen sicherstellt und die Datenbankabfragen so optimiert, dass sie mit minimaler Latenz ausgeführt werden.

Ereignis

Ein Benutzer führt eine API-Anfrage aus oder das System führt eine Datenbankabfrage durch.

Vorbedingung

- Das System ist ordnungsgemäß konfiguriert und betriebsbereit.
- Die Anfrage enthält alle erforderlichen Parameter gemäß der API-Spezifikation.
- Die API-Endpunkte sind im Backend definiert und ordnungsgemäß konfiguriert.

Standardablauf

1. Der Client sendet eine API-Anfrage an das Backend.
2. Das Backend überprüft die Anfrage auf Validität.
3. Die API-Anfrage wird innerhalb einer akzeptablen Zeit beantwortet (abhängig von der Art der Anfrage).
4. Die Datenbankabfrage wird durchgeführt, wenn erforderlich, und liefert die benötigten Daten zurück.
5. Alle Datenbankabfragen werden optimiert und möglichst schnell ausgeführt, ohne die Systemressourcen unnötig zu belasten.
6. Das Backend liefert die Antwort auf die API-Anfrage an den Client innerhalb einer akzeptablen Antwortzeit.

Alternativablauf

- Falls die Datenbankabfragen aufgrund von unoptimierten Abfragen oder hoher Serverlast langsamer sind, kann die Antwortzeit bis zu 500 ms betragen.
- Der Client erhält eine adäquate Statusmeldung.

Nachbedingungen Erfolg

- Die API-Anfragen werden innerhalb einer akzeptablen Zeit beantwortet.
- Alle Datenbankabfragen werden effizient und mit minimaler Latenz ausgeführt.

Nachbedingung Fehler

- Falls die API-Anfrage länger als 500 ms dauert oder Datenbankabfragen länger als 1000 ms benötigen, wird eine Fehlerbehebung erforderlich sein, um die Leistung zu optimieren.

Klassifizierung

Nicht-Funktional, MUSS

Aufwand

GERING

1.3.2 NFABE ID-002

Ziel

Das System stellt sicher, dass alle sensiblen Daten bei der Speicherung verschlüsselt werden. Zudem werden Schutzmechanismen gegen gängige Angriffe implementiert, und eine sichere Authentifizierung sowie Autorisierung für den Zugriff auf das Backend bereitgestellt.

Ereignis

Ein Benutzer interagiert mit der Anwendung, gibt sensible Daten ein oder fordert geschützte Ressourcen an.

Vorbedingung

- Das System ist betriebsbereit und alle relevanten Sicherheitsmechanismen sind aktiviert.
- Der Benutzer hat ein gültiges Konto und die erforderlichen Berechtigungen für den Zugriff auf das Backend.

Standardablauf

1. Ein Benutzer überträgt sensible Daten.
2. Datenvalidierung und Verarbeitung (Schutzmechanismen):
    - Das Backend validiert die eingehenden Daten (z. B. Format des Passworts) und prüft, ob die Daten korrekt sind:
        - SQL-Injection: Werden durch prepared Statements verhindert.
        - Cross-Site-Scripting: Wird durch Output Encoding verhindert.
        - Cross-Site-Request-Forgery: Tokens werden für alle Formulare zur Verfügung gestellt. (optional)
        - Brute-Force-Attacken: Anzahl Anmeldeversuche wird ausgewertet. (optional)
3. Datenverschlüsselung:
    - Die Daten werden vor der Speicherung verschlüsselt.
4. Daten werden gespeichert oder ein JSON Web Token wird generiert und ausgestellt.
5. Das Backend liefert nach erfolgreicher Verarbeitung eine Statusmeldung zurück.

Alternativablauf

- **Datenvalidierung schlägt fehl:** Das Backend gibt eine Fehlermeldung mit Statuscode zurück.
- **Datenverschlüsselung kann nicht erfolgen:** Wenn Fehler beim Verschlüsseln der Daten auftritt wird die Verarbeitung abgebrochen und eine Meldung mit Statuscode 500 (internal Server Error) wird zurückgegeben.
- **Daten werden nicht gespeichert oder das Token wird nicht generiert:** Die Anfrage wird Abgebrochen und eine Meldung mit Statuscode (503 Service Unavailable) wird an den Client gesendet.

Nachbedingungen Erfolg

- Der Benutzer wird nach erfolgreicher Authentifizierung mit einem gültigen JWT ausgestattet, dass ihm Zugriff auf die geschützten Ressourcen gewährt.
- Alle Daten werden sicher verschlüsselt gespeichert und über eine sichere Verbindung übertragen.
- Der Benutzer kann nach der Authentifizierung auf die entsprechenden Funktionen zugreifen, basierend auf seinen Berechtigungen.
- Angriffe werden erfolgreich erkannt und abgewehrt.

Nachbedingung Fehler

- Wenn die Authentifizierung fehlschlägt (falsche Zugangsdaten), wird der Benutzer aufgefordert, die richtigen Anmeldeinformationen einzugeben.
- Bei ungültigem JWT-Token oder abgelaufenem Token wird der Zugriff verweigert und eine 401 Unauthorized-Fehlermeldung ausgegeben.
- Wenn sensible Daten nicht korrekt verschlüsselt oder entschlüsselt werden können, wird der Zugriff auf diese Daten verweigert und eine Fehlermeldung angezeigt.
- Bei einem erfolgreichen Angriff greifen weitere Schutzmechanismen im System ein (z.B. Serververwaltung).
- Meldung durch einen Benutzer im Falle von auffälligem Verhalten oder fehlenden Daten.

Klassifizierung

Nicht-Funktional, MUSS

Aufwand

MITTEL

1.4 Spezifizierung Funktionale Anforderungen Backend FABE

Funktionale Anforderungen Backend

1.4.1 FABE ID-001

Ziel

Das Backend nimmt eingehende Anfragen entgegen, verarbeitet sie gemäß definierten API-Spezifikationen und liefert eine entsprechende Antwort zurück.

Ereignis

Eine eingehende Anfrage wird vom System empfangen und durch den entsprechenden API-Endpunkt verarbeitet.

Vorbedingung

- Eine API-Anfrage wurde vom Frontend oder einer externen Quelle an das Backend gesendet.
- Die Anfrage enthält alle erforderlichen Parameter gemäß der API-Spezifikation.
- Die API-Endpunkte sind im Backend definiert und ordnungsgemäß konfiguriert.

Standardablauf

1. Eine eingehende Anfrage wird über einen definierten API-Endpunkt an das Backend gesendet (z.B. POST, GET, PUT, DELETE).
2. Das Backend überprüft die Anfrage auf Validität:
    - Sind alle erforderlichen Parameter vorhanden?
    - Sind die Parameter korrekt formatiert (z.B. JSON, URL-Parameter)?
3. Das Backend verarbeitet die Anfrage unter Verwendung der definierten Geschäftslogik:
    - Bei einer POST-Anfrage werden neue Datensätze erstellt.
    - Bei einer GET-Anfrage wird eine Datenabfrage durchgeführt.
    - Bei einer PUT-Anfrage werden bestehende Daten aktualisiert.
    - Bei einer DELETE-Anfrage werden Daten gelöscht.
4. Nach der Bearbeitung wird eine Antwort gemäß der API-Spezifikation zurückgegeben:
    - Erfolgreiche Anfragen erhalten eine 2xx-Statusmeldung und entsprechende Daten (falls zutreffend).
    - Fehlerhafte Anfragen erhalten eine passende Fehlermeldung mit einem entsprechenden Fehlercode (z.B. 400 für ungültige Eingaben, 500 für Serverfehler).
5. Die Antwort wird an den Client (Frontend oder externe Quelle) gesendet.

Alternativablauf

- **Fehlende oder ungültige Parameter:** Das Backend gibt eine Fehlermeldung (z.B. 400 Bad Request) zurück, wenn erforderliche Parameter fehlen oder falsch formatiert sind.

- **Interner Serverfehler:** Wenn ein unerwarteter Fehler im Backend auftritt (z.B. Datenbankprobleme, Serverfehler), gibt das Backend eine Fehlerantwort mit dem Statuscode 500 zurück.
- **Nicht autorisierte Anfrage:** Bei einer nicht autorisierten Anfrage (z.B. fehlende Authentifizierung oder unzureichende Berechtigungen) wird eine Fehlermeldung mit dem Statuscode 401 oder 403 zurückgegeben.

Nachbedingungen Erfolg

- Die Anfrage wird erfolgreich bearbeitet und eine korrekte Antwort wird an den Client zurückgegeben.
- Die Daten im Backend wurden wie vorgesehen verarbeitet (z.B. gespeichert, abgerufen, aktualisiert oder gelöscht).

Nachbedingung Fehler

- Der Client erhält eine Fehlermeldung und kann die Anfrage ggf. korrigieren.
- Es erfolgt keine Änderung oder Aktualisierung von Daten im Backend, wenn ein Fehler aufgetreten ist.

Klassifizierung

Funktional, MUSS

Aufwand

HOCH

1.4.2 FABE ID-002

Ziel

Sicherstellung einer konsistenten, validierten und performanten Verarbeitung von Daten durch strukturierte Entgegennahme, Speicherung und effizienten Abruf aus einer relationalen Datenbank.

Ereignis

Ein Benutzer oder ein System sendet eine Anfrage zur Datenverarbeitung, sei es durch das Erstellen, Abrufen, Aktualisieren oder Löschen von Datensätzen.

Vorbedingung

- Eine gültige Anfrage wurde an das Backend gesendet.
- Die Anfrage enthält alle erforderlichen und korrekt formatierten Daten gemäß den API-Spezifikationen.
- Der Benutzer oder das System besitzt die notwendigen Berechtigungen zur Ausführung der jeweiligen Aktion.

Standardablauf

1. Die Anfrage wird an das Backend gesendet.
2. Datenentgegennahme: Das System validiert die eingehenden Daten ([siehe NFABE ID-02]) gemäß den vordefinierten Schemaanforderungen und parst die Datenstruktur.
3. Datenspeicherung:
    - Falls es sich um eine Erstellungs- oder Aktualisierungsanfrage handelt, speichert das System die validierten Daten in einer relationalen Datenbank.
    - Die Speicherung erfolgt unter Einhaltung von Konsistenz- und Integritätsregeln ([siehe NFABE ID-02]).
4. Die Antwort wird an den Benutzer oder das System zurückgesendet.

Alternativablauf

- **Ungültige oder unvollständige Daten:**
    - 400 Bad Request, wenn die übermittelten Daten nicht den Spezifikationen entsprechen.
- **Fehlende Berechtigungen:**
    - 403 Forbidden, wenn der Benutzer keine ausreichenden Rechte besitzt.
- **Daten nicht gefunden:**
    - 404 Not Found, wenn die angeforderten Daten nicht existieren.
- **Datenbankfehler oder Serverprobleme:**
    - 500 Internal Server Error, wenn ein unerwarteter Fehler auftritt.

Nachbedingungen Erfolg

- Die Daten wurden erfolgreich gespeichert, aktualisiert oder abgerufen.
- Die Datenbank bleibt konsistent und erfüllt die definierten Integritätsbedingungen.

Nachbedingung Fehler

- Die Anfrage wird abgelehnt und eine entsprechende Fehlermeldung wird zurückgegeben.
- Ungültige oder unvollständige Daten werden nicht gespeichert.

Klassifizierung

Funktional, MUSS

Aufwand

HOCH

1.4.3 FABE ID-003

Ziel
Bereitstellung eines sicheren Login-Prozesses unter Verwendung von Passwort-Verschlüsselung und JSON Web Token (JWT) für die Authentifizierung von Benutzern.

Ereignis
Ein Benutzer sendet eine Login-Anfrage mit seinen Zugangsdaten (Benutzername und Passwort), und das Backend validiert die Eingaben und stellt ein JWT für die autorisierte Sitzung aus.

Vorbedingung

- Der Benutzer hat ein Konto im System registriert.
- Die Benutzeranfrage enthält den Benutzernamen und das Passwort im Anfrage-Body.

Standardablauf

1. **Login-Anfrage**
    - Beschreibung: Der Benutzer sendet eine POST-Anfrage mit seinen Zugangsdaten (Benutzername und Passwort).
    - Verhalten:
        1. Das Backend empfängt die Login-Daten im Body der Anfrage (z. B. JSON mit username und password).
        2. Das Backend überprüft, ob der Benutzer existiert und ob das angegebene Passwort korrekt ist.
        3. Das Passwort wird im Backend entschlüsselt.
        4. Bei Erfolg wird ein JWT (JSON Web Token) mit einer bestimmten Gültigkeitsdauer erstellt und zurückgegeben.
        5. Das JWT enthält die Benutzerinformationen (z. B. Benutzer-ID) und wird im Authorization-Header der Antwort als Bearer-Token zurückgegeben.
        6. Fehlende oder ungültige Anmeldedaten: Wenn der Benutzername oder das Passwort falsch sind, gibt das Backend eine Fehlermeldung mit dem Statuscode 401 (Unauthorized) zurück.
2. **Verwendung des JWT für zukünftige Anfragen**
    - Beschreibung: Der Benutzer kann nun auf geschützte Endpunkte zugreifen, indem er das erhaltene JWT im Authorization-Header jeder zukünftigen Anfrage sendet.
    - Verhalten:
        1. Der Benutzer sendet eine Anfrage an einen geschützten Endpunkt und fügt den Authorization-Header mit dem JWT hinzu.
        2. Das Backend validiert das JWT, indem es die Signatur überprüft und sicherstellt, dass es noch gültig ist.
        3. Wenn das JWT gültig ist, wird die Anfrage verarbeitet.
        4. Wenn das JWT ungültig oder abgelaufen ist, gibt das Backend eine Fehlermeldung mit dem Statuscode 401 (Unauthorized) zurück.

Alternativablauf

- **Ungültige Zugangsdaten:**
  - 401 Unauthorized, wenn der Benutzername oder das Passwort falsch ist.
  - Der Benutzer muss die Eingaben erneut versuchen.
- **Abgelaufenes oder ungültiges JWT:**
  - 401 Unauthorized, wenn das JWT ungültig oder abgelaufen ist.
  - Der Benutzer muss sich erneut anmelden, um ein neues Token zu erhalten.

Nachbedingungen Erfolg

- Ein JWT wird erfolgreich generiert und an den Benutzer zurückgegeben, der es für zukünftige authentifizierte Anfragen verwenden kann.
- Der Benutzer kann nun auf geschützte Ressourcen zugreifen.

Nachbedingung Fehler

- Der Benutzer erhält eine Fehlermeldung (401 Unauthorized) und kann sich mit den richtigen Anmeldedaten erneut anmelden.

Klassifizierung

Funktional, MUSS

Aufwand

HOCH

1.4.4 FABE ID-004

Ziel

Bereitstellung eines sicheren Endpunkts zur Registrierung neuer Benutzerkonten.

Ereignis

Ein Benutzer sendet eine Anfrage zur Registrierung eines neuen Kontos, das Backend validiert die Eingabedaten und erstellt das Benutzerkonto, sofern alle Voraussetzungen erfüllt sind.

Vorbedingung

- Der Benutzer hat keine bestehenden Anmeldeinformationen mit demselben Benutzernamen oder der gleichen E-Mail-Adresse im System.
- Die Eingabedaten enthalten alle relevanten Daten für eine Registration.

Standardablauf

Registrierungsanfrage:

- **Beschreibung**: Der Benutzer sendet eine POST-Anfrage zur Registrierung eines neuen Kontos mit den erforderlichen Feldern (z.B. Benutzername, E-Mail, Passwort).
- **Verhalten**:
    1. Der Server empfängt die Eingabedaten im Body der Anfrage.
    2. Das Backend überprüft, ob der Benutzername und die E-Mail-Adresse bereits in der Datenbank existieren:
        - Falls der Benutzername oder die E-Mail bereits existiert, gibt das Backend eine Fehlermeldung mit dem Statuscode 409 (Conflict) zurück.
    3. Wenn keine Konflikte festgestellt werden, wird das Passwort im Backend sicher verschlüsselt.
    4. Das System erstellt das Benutzerkonto in der Datenbank mit den validierten Daten (Benutzername, E-Mail und verschlüsseltes Passwort).
    5. Erfolgreiche Antwort: 201 Created mit einer Bestätigung der Kontoerstellung (optional: E-Mail-Bestätigung).

Alternativablauf

- **Doppelte E-Mail oder Benutzername**:
    - 409 Conflict, wenn die E-Mail-Adresse oder der Benutzername bereits existiert.
    - Der Benutzer muss eine andere E-Mail-Adresse oder einen anderen Benutzernamen wählen.
- **Ungültige Eingabedaten**:
    - 400 Bad Request, wenn die eingegebenen Daten fehlerhaft sind.

Nachbedingungen Erfolg

- Das Benutzerkonto wird erfolgreich erstellt und in der Datenbank gespeichert.
- Der Benutzer erhält eine Bestätigung der erfolgreichen Registrierung, und einem Link zum Login.

Nachbedingung Fehler

- Der Benutzer erhält eine Fehlermeldung und wird aufgefordert, seine Eingaben zu korrigieren.
- Keine Daten werden im System gespeichert, wenn der Registrierungsprozess nicht erfolgreich ist.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

1.4.5 FABE ID-005

Ziel
Bereitstellung von RESTful-API-Endpunkten zur Bearbeitung, Aktualisierung und Löschung von Benutzerkonten, um Nutzern die Verwaltung ihrer Kontodaten zu ermöglichen.

Ereignis
Ein authentifizierter Benutzer sendet eine Anfrage zur Bearbeitung, Aktualisierung oder Löschung seines Benutzerkontos.

Vorbedingung

- Der Benutzer ist authentifiziert und verfügt über ein gültiges JWT.
- Die Anfrage enthält gültige und überprüfbare Daten.

Standardablauf

1. **Bearbeiten/Aktualisieren der Benutzerinformationen**
    - Beschreibung: Der Benutzer kann seine Profildaten aktualisieren, z.B. E-Mail, Benutzername, Ort/Region oder Passwort.
    - Verhalten:
        1. Der Server empfängt eine PUT-Anfrage mit den neuen Benutzerdaten im Body.
        2. Das JWT wird validiert.
        3. Das Backend überprüft, ob der Benutzername oder die E-Mail-Adresse bereits vergeben sind.
        4. Falls das Passwort geändert wird, wird das neue Passwort sicher verschlüsselt.
        5. Die neuen Daten werden gespeichert.
        6. Erfolgreiche Antwort: 200 OK mit aktualisierten Benutzerdaten.
        7. Falls fehlerhafte Eingaben oder Konflikte bestehen: 400 Bad Request oder 409 Conflict.
2. **Löschen des Benutzerkontos**
    - Beschreibung: Der Benutzer kann sein Konto dauerhaft löschen.
    - Verhalten:
        1. Der Server empfängt eine DELETE-Anfrage.
        2. Das JWT wird validiert.
        3. Falls das eingegebene Passwort korrekt ist, wird das Benutzerkonto gelöscht.
        4. Optional: Die Daten werden anonymisiert oder in einer separaten Tabelle archiviert.
        5. Erfolgreiche Antwort: 204 No Content.
        6. Falls das Passwort falsch ist: 401 Unauthorized.

Alternativablauf

- **Ungültige Eingabedaten bei Aktualisierung**:
  - 400 Bad Request, wenn die eingegebenen Daten fehlerhaft sind.
- **Doppelte E-Mail oder Benutzername bei Aktualisierung**:
  - 409 Conflict, wenn die neue E-Mail oder der neue Benutzername bereits vergeben sind.
- **Falsches Passwort bei Kontolöschung**:
  - 401 Unauthorized, wenn das eingegebene Passwort nicht mit dem gespeicherten übereinstimmt.

Nachbedingungen Erfolg

- Die Benutzerdaten werden erfolgreich aktualisiert und gespeichert.
- Bei Kontolöschung wird das Konto entfernt oder anonymisiert.

Nachbedingung Fehler

- Der Benutzer erhält eine Fehlermeldung und wird aufgefordert, die Eingaben zu korrigieren.
- Keine Änderungen werden gespeichert, wenn eine Aktualisierung oder Löschung fehlschlägt.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

1.4.6 FABE ID-006

Ziel

Beim Öffnen der Seite ‘Board’ soll das System automatisch alle verfügbaren Dienstleistungen aus der Datenbank abrufen und anzeigen.

Ereignis

Ein Benutzer öffnet das Board in der Anwendung.

Vorbedingung

- Der Benutzer greift auf das Board über die Benutzeroberfläche zu.
- Der Server ist erreichbar.

Standardablauf

1. **Öffnen des Boards**
    - Der Benutzer navigiert zur Board-Ansicht.
2. **Automatisches Laden der Dienstleistungen**
    - Beschreibung: Die Anwendung sendet automatisch eine GET-Anfrage an das Backend, um alle verfügbaren Dienstleistungen abzurufen.
    - Verhalten:
        1. Die Board-Seite wird geladen.
        2. Die Anwendung sendet eine Anfrage an den API-Endpunkt.
        3. Das Backend ruft alle verfügbaren Dienstleistungen aus der Datenbank ab.
        4. Die Liste der Dienstleistungen wird als JSON zurückgegeben.
        5. Die Anwendung zeigt die Dienstleistungen im Board an.
        6. Erfolgreiche Antwort: 200 OK mit einer Liste von Dienstleistungen.

Alternativablauf

- **Keine Dienstleistungen vorhanden**:
    - 200 OK mit einer leeren Liste, falls keine Dienstleistungen im System gespeichert sind.
- **Server nicht erreichbar**:
    - 500 Internal Server Error oder 503 Service Unavailable, falls das Backend nicht erreichbar ist.

Nachbedingungen Erfolg

- Die Dienstleistungen werden erfolgreich geladen und im Board angezeigt.

Nachbedingung Fehler

- Der Benutzer erhält eine entsprechende Fehlermeldung oder eine leere Liste.

Klassifizierung

Funktional, MUSS

Aufwand

GERING

1.4.7 FABE ID-007

Ziel
Bereitstellung von API-Endpunkten zur Erstellung, Abruf, Aktualisierung und Löschung von Dienstleistungen, um eine vollständige Verwaltung der Dienstleistungsdaten zu ermöglichen.

Ereignis
Ein authentifizierter Benutzer mit entsprechenden Rechten sendet eine Anfrage zur Erstellung, Aktualisierung, Anzeige oder Löschung einer Dienstleistung.

Vorbedingung

- Der Benutzer ist authentifiziert und besitzt die erforderlichen Berechtigungen für die jeweilige Aktion.
- Die Anfrage enthält gültige und vollständige Daten gemäß API-Spezifikationen.

Standardablauf

1. **Erstellen einer neuen Dienstleistung**
    - Beschreibung: Ein Benutzer kann eine neue Dienstleistung hinzufügen.
    - Verhalten:
        1. Die Anwendung sendet eine POST-Anfrage mit den notwendigen Daten (Titel, Beschreibung, Kategorie, Ort/Region).
        2. Das Backend validiert die Eingaben.
        3. Falls die Validierung erfolgreich ist, wird die Dienstleistung in der Datenbank gespeichert.
        4. Erfolgreiche Antwort: 201 Created mit der neu erstellten Dienstleistung.
2. **Abrufen aller Dienstleistungen**
    - Beschreibung: Alle Dienstleistungen des Benutzers können abgerufen werden.
    - Verhalten:
        1. Das Backend ruft die angeforderten Daten aus der Datenbank ab.
        2. Erfolgreiche Antwort: 200 OK mit der Dienstleistung oder einer Liste von Dienstleistungen.
3. **Aktualisieren einer bestehenden Dienstleistung**
    - Beschreibung: Ein Benutzer kann eine bestehende Dienstleistung bearbeiten.
    - Verhalten:
        1. Die Anwendung sendet eine PUT-Anfrage mit den aktualisierten Daten.
        2. Das Backend prüft, ob die Dienstleistung existiert und ob der Benutzer berechtigt ist, sie zu ändern.
        3. Falls erfolgreich, werden die Daten aktualisiert und gespeichert.
        4. Erfolgreiche Antwort: 200 OK mit den aktualisierten Daten.
4. **Löschen einer Dienstleistung**
    - Beschreibung: Ein Benutzer kann eine Dienstleistung dauerhaft entfernen.
    - Verhalten:
        1. Die Anwendung sendet eine DELETE-Anfrage.
        2. Das Backend prüft, ob die Dienstleistung existiert
        3. Falls erfolgreich, wird die Dienstleistung aus der Datenbank entfernt.
        4. Erfolgreiche Antwort: 204 No Content.

Alternativablauf

- **Dienstleistung nicht gefunden**:
  - 404 Not Found, wenn die angeforderte Dienstleistung nicht existiert.
- **Ungültige Eingabedaten**:
  - 400 Bad Request, wenn die übermittelten Daten fehlerhaft oder unvollständig sind.

Nachbedingungen Erfolg

- Die Dienstleistung wird erfolgreich erstellt, aktualisiert, abgerufen oder gelöscht.

Nachbedingung Fehler

- Der Benutzer erhält eine Fehlermeldung und muss die Eingaben korrigieren oder sich authentifizieren.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

1.4.8 FABE ID-008

Ziel

Bereitstellung von API-Endpunkten zur Erstellung, Anzeige, Bearbeitung und Löschung des öffentlichen Profils eines Benutzers, um eine individuelle Präsentation für andere Nutzer oder Interessenten zu ermöglichen.

Ereignis

Ein authentifizierter Benutzer sendet eine Anfrage zur Erstellung, Aktualisierung, Anzeige oder Löschung seines öffentlichen Profils.

Vorbedingung

- Der Benutzer ist authentifiziert und besitzt die erforderlichen Berechtigungen für die jeweilige Aktion.
- Die Anfrage enthält gültige und vollständige Daten gemäß API-Spezifikationen.

Standardablauf

1. **Erstellen eines öffentlichen Profils**
    - Beschreibung: Ein Benutzer kann sein öffentliches Profil anlegen.
    - Verhalten:
        1. Die Anwendung sendet eine POST-Anfrage mit den notwendigen Profildaten (Name, Beschreibung, Ort/Region)
        2. Falls die Validierung erfolgreich ist, wird das Profil in der Datenbank gespeichert.
        3. Erfolgreiche Antwort: 201 Created mit den gespeicherten Profildaten.
2. **Abrufen eines öffentlichen Profils**
    - Beschreibung: Andere Nutzer können das öffentliche Profil eines bestimmten Benutzers durch die Auswahl einer Dienstleistung welche vom Benutzer erstellt wurde sehen.
    - Verhalten:
        1. Die Anwendung sendet eine GET-Anfrage mit der Benutzer-ID.
        2. Das Backend ruft die entsprechenden Profildaten aus der Datenbank ab.
        3. Falls erfolgreich, wird das Profil zurückgegeben.
        4. Erfolgreiche Antwort: 200 OK mit Profildaten.
3. **Aktualisieren des öffentlichen Profils**
    - Beschreibung: Ein Benutzer kann sein eigenes öffentliches Profil bearbeiten.
    - Verhalten:
        1. Die Anwendung sendet eine PUT-Anfrage mit den aktualisierten Daten.
        2. Das Backend überprüft, ob der Benutzer berechtigt ist, das Profil zu ändern.
        3. Falls erfolgreich, werden die Daten aktualisiert und gespeichert.
        4. Erfolgreiche Antwort: 200 OK mit den neuen Profildaten.
4. **Löschen eines öffentlichen Profils**
    - Beschreibung: Ein Benutzer kann sein öffentliches Profil entfernen.
    - Verhalten:
        1. Die Anwendung sendet eine DELETE-Anfrage.
        2. Das Backend überprüft, ob der Benutzer berechtigt ist, das Profil zu ändern.
        3. Falls Validierung erfolgreich, wird das Profil aus der Datenbank entfernt oder anonymisiert.
        4. Erfolgreiche Antwort: 204 No Content.

Alternativablauf

- **Profil nicht gefunden**:
    - 404 Not Found, wenn das angeforderte Profil nicht existiert.
- **Ungültige Eingabedaten**:
    - 400 Bad Request, wenn die übermittelten Daten fehlerhaft oder unvollständig sind.

Nachbedingungen Erfolg

- Das öffentliche Profil wird erfolgreich erstellt, aktualisiert, abgerufen oder gelöscht.

Nachbedingung Fehler

- Der Benutzer erhält eine Fehlermeldung und muss die Eingaben korrigieren oder sich authentifizieren.

Klassifizierung

Funktional, MUSS

Aufwand

MITTEL

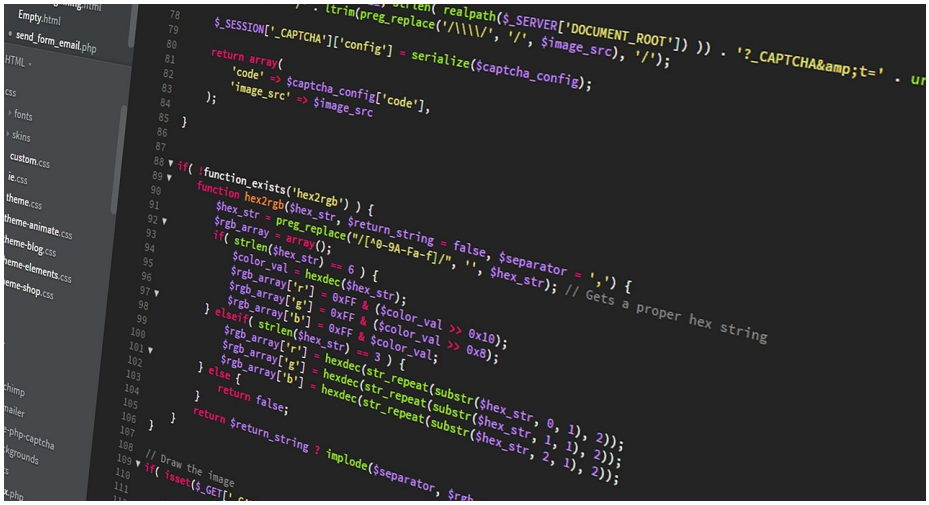
FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Logbuch

Der Semesterarbeit des Studiengangs CAS Fullstack-Development

von

Kay Roman Hertenstein



Roggwil, Januar – Juni 2025

Inhaltsverzeichnis

1 Logbuch

1.1 Eintrag: Erstellen der Projektvorgabe

Name: Kay Hertenstein
Projekt/Tätigkeit: Erstellen der technischen Anforderungen
Zeitraum: 08.02.25 – 01.03.25

Tätigkeiten / Aufgaben:
Gemäss der Modulvorgabe, erstelle ich das Dokument für die Semesterarbeit/Projektarbeit. Durch das genaue erstellen der Dokumentation werde ich mir den konkreten Aufgaben bewusst, welche ich umsetzten werde. Ich merke wie eine sauber erstellte Dokumentation ein wesentlicher Faktor ist für eine effiziente Softwareentwicklung.

Erreichte Ziele / Ergebnisse:
Ich konnte die technischen Anforderungen definieren. Konkrete Eckdaten der Applikation werden festgelegt. Erfolgreich ein eigenes Mockup der Applikation erstellt.

Schwierigkeiten / Probleme:
Definieren der Anforderungen und Zeitmanagement bezüglich Themen welche später im Modul unterrichtet werden. Einbringen des agilen Entwicklungsansatzes in Spezifikationen.

Lösungen / Maßnahmen:
In der Dokumentation dokumentieren, dass sich die Spezifikationen/Präferenzen gemäss Unterricht gewählt werden mit Verweis auf prototypische Implementierungen.

Zusammenfassung:
In der Softwareentwicklung spricht man oft von agilen Prozessen, diese in funktionalen Anforderungen miteinzubeziehen hat mich verunsichert. Durch das Erstellen der Spezifikation hat meine Idee eine konkrete Form erhalten. Viele weitere Möglichkeiten (Features) werden mir bewusst.