Projektdokumentation: TotalPETView

1. Einleitung

1.1 Projektziel

TotalPETView ist ein web-basiertes System zur Verwaltung und Visualisierung von medizinischen Bilddaten im DICOM-Format, spezialisiert auf PET-Scans. Es ermöglicht den Upload, die Suche und die Anzeige von DICOM-Dateien über eine benutzerfreundliche Oberfläche.

1.2 Motivation

Die Wahl fiel auf Django für das Backend aufgrund seiner robusten und sicheren Architektur sowie der schnellen Entwicklungszeit. Vue.js wurde für das Frontend gewählt, da es ein reaktives und modernes JavaScript-Framework ist, das sich gut für Single-Page-Applications eignet. Orthanc dient als leistungsstarker, Open-Source PACS-Server (Picture Archiving and Communication System), der die Speicherung und den Abruf von DICOM-Dateien übernimmt.

1.3 Use Case

Das System wird von medizinischem Fachpersonal, wie Radiologen oder Forschern, genutzt. Ein typischer Anwendungsfall ist der Upload von PET-Scans eines Patienten, die anschließende Suche nach diesen Scans in der Datenbank und die Visualisierung im integrierten DICOM-Viewer zur Befundung oder Analyse.

---

2. Anforderungen

2.1 Funktionale Anforderungen

- [x] Upload von DICOM-Dateien
- [x] Anzeige und Suche in PACS
- [x] Visualisierung im Viewer
- [x] Benutzerverwaltung

2.2 Nicht-funktionale Anforderungen

- [ ] Performance: Schnelle Ladezeiten für Bilder und Suchergebnisse.
- [x] Sicherheit: Sichere Authentifizierung und verschlüsselte Datenübertragung.
- [ ] Erweiterbarkeit: Modulare Architektur zur einfachen Integration neuer Funktionen.

---

3. Architektur

3.1 Architekturübersicht

Das System besteht aus drei Hauptkomponenten:

1. **Frontend (Vue.js)**: Die Benutzeroberfläche, die im Browser des Benutzers läuft.

2. **Backend (Django)**: Der Server, der die Geschäftslogik, Benutzerverwaltung und die Kommunikation mit dem Orthanc-Server übernimmt.
3. **Orthanc (PACS)**: Ein als Docker-Container laufender DICOM-Server, der die medizinischen Bilder speichert und über eine REST-API bereitstellt.

Diagramm: [Frontend] ↔ [Backend (Django)] ↔ [Orthanc (Docker)]

3.2 Technologiestack

- **Backend**: Django, Django REST Framework
- **PACS**: Orthanc (läuft als Docker-Container)
- **Frontend**: Vue.js, Axios
- **Datenbank**: SQLite (für Django-Benutzerverwaltung)
- **Weitere Tools**: Docker

3.3 Kommunikationsfluss

1. **Upload**: Der Benutzer wählt eine DICOM-Datei im Frontend aus. Die Datei wird über einen REST-API-Aufruf an das Django-Backend gesendet.
2. **Verarbeitung**: Das Django-Backend empfängt die Datei und leitet sie an den Orthanc-Server weiter, der sie speichert.
3. **Suche**: Der Benutzer gibt Suchkriterien im Frontend ein. Das Frontend sendet eine Anfrage an das Django-Backend.
4. **Abfrage**: Das Django-Backend fragt den Orthanc-Server nach passenden DICOM-Studien ab und gibt die Ergebnisse an das Frontend zurück.
5. **Anzeige**: Der Benutzer wählt eine Studie im Frontend aus. Das Frontend ruft die Bild-URL vom Orthanc-Server (über das Backend) ab und zeigt sie im Viewer an.

---

4. Installation & Setup

4.1 Voraussetzungen

- Python >= 3.8
- Node.js >= 18.x
- Docker und Docker Compose

4.2 Installationsschritte

```
# 1. Orthanc-Server starten
# (Eine orthanc.json Konfigurationsdatei und ein docker-compose.yml werden
vorausgesetzt)
docker-compose up -d orthanc

# 2. Backend installieren
cd backend
pip install -r requirements.txt
python manage.py migrate
```

```
# 3. Frontend installieren
cd ../frontend
npm install
```

4.3 Starten der Anwendung

```
# Backend-Server starten
cd backend
python manage.py runserver

# Frontend-Entwicklungsserver starten
cd ../frontend
npm run dev

# Orthanc läuft bereits im Hintergrund via Docker
```

---

5. Backend (Django)

5.1 Projektstruktur

Die Backend-Struktur ist modular aufgebaut:

- `api/`: Enthält die Logik für die API-Endpunkte (Upload, Suche).
- `authenticator/`: Verantwortlich für die Benutzerauthentifizierung und -verwaltung.
- `backend/`: Das Haupt-Django-Projekt mit den globalen Einstellungen.

5.2 API-Endpunkte

Endpoint	Methode	Beschreibung	Auth erforderlich
`/api/register/`	POST	Registriert einen neuen Benutzer	Nein
`/api/login/`	POST	Meldet einen Benutzer an und gibt ein Token zurück	Nein
`/api/upload/`	POST	Lädt eine DICOM-Datei hoch	Ja
`/api/search/`	GET	Sucht nach DICOM-Studien	Ja
`/api/studies/<study_id>/`	GET	Ruft Details zu einer Studie ab	Ja

5.3 Integration mit Orthanc

Die Kommunikation zwischen Django und Orthanc erfolgt über REST-API-Aufrufe. Das Django-Backend sendet HTTP-Anfragen an die Orthanc-API, um DICOM-Dateien hochzuladen (`/instances`), zu suchen (`/tools/find`) und abzurufen. Die URL und die Zugangsdaten für den Orthanc-Server sind in der `settings.py` von Django konfiguriert.

---

6. PACS (Orthanc)

6.1 Konfiguration

Orthanc wird über eine `orthanc.json`-Datei konfiguriert, die im Docker-Setup eingebunden wird. Wichtige Einstellungen sind:

- `StorageDirectory`: Der Pfad, an dem die DICOM-Dateien gespeichert werden.
- `RemoteAccessAllowed`: Muss auf `true` gesetzt sein, um den Zugriff vom Django-Backend zu erlauben.
- `RegisteredUsers`: Definiert die Benutzer und Passwörter für den API-Zugriff.

6.2 Verbindung zu Django

Django authentifiziert sich bei Orthanc über HTTP Basic Authentication. Die Anmeldeinformationen werden bei jedem API-Aufruf im Header mitgesendet.

6.3 Beispiel-API-Aufruf (von Django an Orthanc)

```
# Beispiel in Python mit der requests-Bibliothek
import requests

files = {'file': open('image.dcm', 'rb')}
response = requests.post(
    'http://localhost:8042/instances',
    files=files,
    auth=('orthanc_user', 'orthanc_password')
)
```

---

7. Frontend (Vue.js)

7.1 Projektstruktur

- `src/components/`: Wiederverwendbare UI-Komponenten (z.B. Layout, Buttons).
- `src/views/`: Seiten-Komponenten, die über den Router aufgerufen werden (z.B. Login, Upload, Viewer).
- `src/router.js`: Definiert die URL-Pfade und die zugehörigen Vue-Komponenten.
- `src/store/`: Vuex-Store für das State-Management (z.B. Authentifizierungsstatus).

7.2 Routing

- `/login`: Anmeldeseite
- `/register`: Registrierungsseite
- `/upload`: Seite zum Hochladen von DICOM-Dateien.
- `/search`: Suchseite für DICOM-Studien.
- `/viewer/<study_id>`: Viewer zur Anzeige einer bestimmten Studie.

7.3 Kommunikation mit Backend

Die Kommunikation mit dem Django-Backend erfolgt über `axios`. API-Anfragen werden an die Endpunkte des Django-Servers gesendet. Für geschützte Endpunkte wird ein JWT (JSON Web Token) im `Authorization`-Header mitgesendet.

```
// Beispiel für einen Upload mit axios
import axios from 'axios';

const formData = new FormData();
formData.append('file', this.selectedFile);

axios.post('/api/upload/', formData, {
  headers: {
    'Authorization': `Bearer ${localStorage.getItem('token')}`,
    'Content-Type': 'multipart/form-data'
  }
});
```

---

8. Sicherheit

- **Authentifizierung**: Die Benutzerauthentifizierung erfolgt über JWTs. Das Backend stellt nach erfolgreicher Anmeldung ein Token aus, das vom Frontend für alle weiteren Anfragen verwendet wird.
- **Autorisierung**: API-Endpunkte sind geschützt und erfordern ein gültiges JWT.
- **CORS**: Cross-Origin Resource Sharing ist auf dem Backend konfiguriert, um Anfragen vom Frontend zu erlauben.
- **Orthanc-Sicherheit**: Der Orthanc-Server ist durch ein Passwort geschützt und sollte nicht direkt aus dem Internet erreichbar sein, sondern nur über das Backend.