Architekturdokument

Inhalt

[Technologieübersicht 3](#_Toc192665637)

[Architekturübersicht 3](#_Toc192665638)

[Struktur (Hauptkompunenten / Klassen) 3](#_Toc192665639)

[UML-Klassendiagramm 4](#_Toc192665640)

[Verhalten (zentraler Systemablauf) 4](#_Toc192665641)

[UML-Aktivitätsdiagramm 4](#_Toc192665642)

# Technologieübersicht

Django (Python Web-Framework):

* Ermöglicht schnelle Entwicklung mit dem MVC-Pattern (Model-View-Controller)
* Eingebaute Datenbankverwaltung (ORM für SQLite, PostgreSQL etc.)
* Gute Erweiterbarkeit für spätere API-Integration

SQLite (Datenbank):

* Einfach zu integrieren und leichtgewichtig für kleine bis mittelgroße Projekte
* Keine externe Installation erforderlich

HTML und CSS (Frontend):

* Ermöglicht eine Benutzeroberfläche zu erstellen

# Architekturübersicht

Das System basiert auf einer 3 **Schichtenarchitektur** mit Django als Kerntechnologie. Dies ermöglicht eine **klare Trennung** zwischen Datenhaltung, Geschäftslogik und Benutzeroberfläche.

Präsentationsschicht:

* HTML-Templates + CSS für das UI
* Django Views zur Steuerung der Seitenlogik

Anwendungsschicht:

* Django Views & Models für Geschäftslogik
* Verwendung von Django ORM zur Datenverarbeitung

Datenhaltungsschicht:

* SQLite zur Speicherung der Filme und Benutzerinformationen

# Struktur (Hauptkompunenten / Klassen)

* **Movie (Model):** Repräsentiert einen Film mit Titel, Genre, Bewertung und Poster-URL
* **RecommendationEngine (Service-Klasse):** Implementiert die Filterlogik für Filmempfehlungen
* **Views (Controller-Schicht):** Verarbeitet Nutzeranfragen und stellt Daten für Templates bereit
* **Templates (Präsentationsschicht):** HTML-Dateien zur Darstellung der Benutzeroberfläche

# Verhalten (zentraler Systemablauf)

1. **Filmempfehlung generieren:**
   * Nutzer gibt Genre in das Suchfeld ein
   * Das System ruft passende Filme aus der Datenbank ab
   * Die gefilterten Ergebnisse werden dem Nutzer präsentiert

# UML-Sequenzdiagramm

test