# Protokoll – Media Ratings Platform (Zwischenabgabe)

Name: Sabic Kenan (if24b080)

Matrikelnummer: 12027473

Abgabedatum: 19.10.2025

Git-Repository: https://github.com/tempestkny/Media-Ratings-Platform

## Projektbeschreibung

Die Media Ratings Platform (MRP) ist ein serverbasiertes System, das eine REST-Architektur verwendet und in der Programmiersprache C# implementiert ist.

Das System ermöglicht die Registrierung und Authentifizierung von Benutzern sowie die Verwaltung von Medieneinträgen, darunter Filme, Serien und Spiele.

Für die Netzwerkkommunikation wird ausschließlich die Standardklasse HttpListener eingesetzt. Die Nutzung von Web-Frameworks wie ASP.NET oder Spring ist explizit ausgeschlossen.

Im aktuellen Entwicklungsstand erfolgt die Datenspeicherung ausschließlich im Arbeitsspeicher (In-Memory).

Eine persistente Datenbankanbindung mittels PostgreSQL ist für die Endabgabe vorgesehen.

Zukünftige Versionen werden um Funktionen wie Bewertungen, Favoriten und Empfehlungen erweitert.

## Mindestanforderungen (Must Haves)

- Das Projekt ist in C# (.NET 10) umgesetzt.

- Der Server lauscht auf eingehende HTTP-Anfragen über HttpListener.

- Das Projekt kompiliert fehlerfrei und lässt sich mit 'dotnet run' starten.

- Es wird keine Web-Framework-Bibliothek (z. B. ASP.NET, Spring) verwendet – nur reine HTTP-Funktionalität.

Alle Mindestanforderungen sind erfüllt.

## Funktionale Features – REST Server

Der Server ist unter http://localhost:8080 erreichbar und verarbeitet Anfragen manuell in der Methode HandleRequest der Klasse SimpleHttpServer.

Die Datenübertragung erfolgt im JSON-Format. Hilfsfunktionen übernehmen das Einlesen und Schreiben der Daten.

Folgende Endpunkte sind implementiert:

1. POST /api/users/register – Registrierung eines neuen Benutzers

2. POST /api/users/login – Anmeldung eines Benutzers und Erhalt eines Tokens

3. POST /api/media – Erstellen eines neuen Medieneintrags (mit Authentifizierung)

4. GET /api/media – Auflisten aller Medieneinträge

5. GET /api/media/{id} – Abfrage eines bestimmten Medieneintrags

6. PUT /api/media/{id} – Bearbeiten eines Medieneintrags (nur durch Ersteller)

7. DELETE /api/media/{id} – Löschen eines Medieneintrags (nur durch Ersteller)

Die Authentifizierung basiert auf einem einfachen Token-basierten System.

Das verwendete Token-Format ist: benutzername-mrpToken.

Tokens werden nur im Arbeitsspeicher gespeichert und beim Neustart gelöscht.

## Funktionale Anforderungen

Implementierte Modellklassen:

- User: Benutzername, Passwort-Hash, optionale E-Mail und Lieblingsgenre

- MediaEntry: Titel, Beschreibung, Medientyp (Film, Serie, Spiel), Erscheinungsjahr, Genres, Altersfreigabe

- Rating: einfache Platzhalterklasse für zukünftige Bewertungsfunktionen

Die Registrierung und Authentifizierung der Benutzer erfolgt über die Komponente InMemoryUserStore.

Vor der Speicherung werden Passwörter mit dem SHA256-Algorithmus gehasht.

Sämtliche Daten werden temporär in einfachen In-Memory-Listen gespeichert.

Die Funktionalität der Anwendung wurde mit dem Testwerkzeug Postman überprüft.

Eine entsprechende Postman Collection (postman\_collection.json) ist beigefügt.

## Projektstruktur

Media\_Ratings\_Platform

├─ Program.cs

├─ Server

│├─ SimpleHttpServer.cs

│└─ TokenManager.cs

├─ Helper

│└─ HelperFunctions.cs

├─ InMemory

│├─ InMemoryUserStore.cs

│├─ InMemoryMediaStore.cs

│└─ InMemoryRatingStore.cs (Prototyp für spätere Version)

└─ Models

├─ User.cs

├─ MediaEntry.cs

├─ MediaType.cs

└─ Rating.cs

## Startanleitung

1. Projekt in einer beliebigen IDE öffnen.

2. Im Terminal ausführen: dotnet run

3. Die Konsole zeigt: Server is running on http://localhost:8080/

4. In Postman Anfragen an den Server schicken (siehe Endpunkte oben).

## Beispieltest mit Postman

1. Registrierung

POST http://localhost:8080/api/users/register

Body:

{ "username": "alice", "password": "12345" }

2. Anmeldung

POST http://localhost:8080/api/users/login

Body:

{ "username": "alice", "password": "12345" }

Antwort:

{ "token": "alice-mrpToken" }

## Designentscheidungen

- Verwendung von HttpListener, um die REST-Kommunikation ohne Framework zu realisieren.

- Einfache Projektstruktur mit Unterordnern für Modelle, Server, Hilfsklassen und In-Memory-Speicher.

- Die Datenhaltung erfolgt im aktuellen Stand ausschließlich im Arbeitsspeicher.

- Authentifizierung über selbst generierte Tokens.

- Der Quellcode ist bewusst einfach und lesbar gehalten. Komplexe Entwurfsmuster wurden vermieden.

## Zusammenfassung

Der aktuelle Entwicklungsstand erfüllt sämtliche in der Checkliste definierten Anforderungen.

Der Server arbeitet stabil, verarbeitet HTTP-Anfragen korrekt, unterstützt die Authentifizierung und liefert konsistente JSON-Antworten.

Die mit Postman durchgeführten Tests verliefen erfolgreich. Alle funktionalen Kernanforderungen des aktuellen Entwicklungsstandes wurden umgesetzt.