Kos Umut, Özcan Emirhan

4bWI  HTL Dornbirn

Filmfolio

Inhalt

[1. Einleitung 3](#_Toc191629032)

[1.1 Projektteam 3](#_Toc191629033)

[1.2 Hintergrund 3](#_Toc191629034)

[1.3 Ziel der Arbeit 3](#_Toc191629035)

[1.4 Mockups 4](#_Toc191629036)

[1.5 Projekthandbuch 5](#_Toc191629037)

[1.5.1 Projektauftrag 5](#_Toc191629038)

[1.5.2 Projektumwelten 6](#_Toc191629039)

[2. Problemstellung 7](#_Toc191629040)

[2.1 Identifikation des Problems 7](#_Toc191629041)

[2.2 Zielgruppe 7](#_Toc191629042)

[3. Projektbeschreibung 8](#_Toc191629043)

[3.1 Funktionen von FimFolio 8](#_Toc191629044)

[4. Technische Umsetzung 9](#_Toc191629045)

[4.1 Technologien 9](#_Toc191629046)

[4.1.1 loadTopMovies und loadTopSeries 9](#_Toc191629047)

[4.1.2 searchMovies und performSearch 9](#_Toc191629048)

[4.1.3 showDetailCard 10](#_Toc191629049)

[4.1.4 Firebase-Integration: loadFavorites, loadWatched, loadPlanned 11](#_Toc191629050)

[4.1.5 Benutzer-Authentifizierung: onAuthStateChanged und signOut 12](#_Toc191629051)

[4.1.6 Firestore Regeln 13](#_Toc191629052)

[4.1.7 Beispiel eines Firestore Schreibvorgang 13](#_Toc191629053)

[5. Herausforderungen 14](#_Toc191629054)

[5.1 Technische Herausforderungen 14](#_Toc191629055)

[5.1.1 Unendliche Anmeldeschleife 14](#_Toc191629056)

[5.1.2 Wechsel von SQL zu Firebase 14](#_Toc191629057)

[5.1.3 Designprobleme in der Detailansicht 14](#_Toc191629058)

[6. Ergebnisse 15](#_Toc191629059)

[6.1 Funktionalität 15](#_Toc191629060)

[6.1.1 Benutzerverwaltung 15](#_Toc191629061)

[6.1.2 Filme- und Serien-Suche 16](#_Toc191629062)

[6.1.3 Anzeige von Top-Inhalten 17](#_Toc191629063)

[7. Fazit und Ausblick 18](#_Toc191629064)

[7.1 Zusammenfassung 18](#_Toc191629065)

[7.2 Zukünftige Entwicklungen 19](#_Toc191629066)

[8. Anhang 20](#_Toc191629067)

[Literaturverzeichnis 20](#_Toc191629068)

[Abbildungsverzeichnis 20](#_Toc191629069)

# Einleitung

## Projektteam

Das Projektteam besteht aus:

* Micheal Leeb – Projektauftraggeber
  + Zuständigkeiten: Überwachung des Projektstandes
* Umut Kos – Projektleiter
* Zuständigkeiten: Backend, Unterstützung bei Frontend
* Emirhan Özcan – Projektmitglied
  + Zuständigkeiten: Frontend

## Hintergrund

Projektauftrag von Herr Leeb 05.11.2024, jeder konnte sich etwas selbst überlegen und musste dies dann umsetzen.

## Ziel der Arbeit

Das Ziel des Projekts ist die Erstellung einer benutzerfreundlichen Website mit einem ansprechenden User Interface, die Filme und Serien übersichtlich darstellt. Eine API wird integriert, die automatisch Filme und Serien erfasst und deren Informationen abruft. Die Website zeigt realistische Bewertungen (Ratings) für Filme und Serien sowie Poster und Beschreibungen zu jedem Titel an. Zudem soll eine intuitive und einfache Nutzung sichergestellt werden, sodass Benutzer schnell und einfach nach Informationen zu Filmen und Serien suchen können.[[1]](#footnote-1)

## Mockups

Ein Bild, das Text, Screenshot, Rechteck, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 1: Mockup Startseite

Ein Bild, das Text, Screenshot, Website, Webseite enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 2: Mockup Detailansicht

Im Zuge unserer Konzeption arbeiteten wir schrittweise auf die Erstellung erster Mockups hin, um ein klares Bild vom späteren Seitenaufbau zu erhalten. Diese wurden auf der Seite Figma erstellt. Abbildung 1 zeigt einen Entwurf für die Startseite, während Abbildung 2 eine Detailansicht demonstriert. Diese Mockups dienten als Grundlage, um Layout und Inhalte frühzeitig zu visualisieren und nutzerfreundlich zu gestalten. [[2]](#footnote-2)

## 1.5 Projekthandbuch

### 1.5.1 Projektauftrag

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 3: Projekthandbuch

In unserem Projekthandbuch „FilmFolio“ beschrieben wir die Planung, Durchführung und den Abschluss eines Vorhabens zur Erstellung einer Website für Film- und Serienbewertungen.

Wir begannen mit einer Darstellung der Projektgrundlagen. Dabei hielten wir im Projektauftrag die wichtigsten Eckdaten fest – vom Projektstart über das Projektende bis hin zu Zielen und Nicht-Zielen. Unser Ziel war es, eine benutzerfreundliche Website zu erstellen, die Informationen über Filme und Serien mithilfe einer API bezog und sowohl Poster als auch Bewertungen und Beschreibungen übersichtlich darstellte. Gleichzeitig grenzten wir aus, was wir nicht leisten würden, zum Beispiel die Entwicklung einer mobilen App.[[3]](#footnote-3)

### 1.5.2 Projektumwelten

Ein Bild, das Text, Screenshot, Diagramm, Kreis enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 4: Projektumwelten

Die Grafik stellt das Projekt FilmFolio in der Mitte dar und zeigt, welche Umwelten es umgeben. Dazu zählen zum Beispiel interne Abteilungen des Unternehmens, externe Lieferanten (etwa ein Technologie- oder API-Anbieter), Behörden, der Wettbewerb und der Projektauftraggeber. Die einzelnen Kreise bzw. Felder verdeutlichen, mit welchen Akteuren FilmFolio interagiert und welche potenziellen Einflüsse oder Abhängigkeiten daraus entstehen.[[4]](#footnote-4)

# Problemstellung

## Identifikation des Problems

In diesem Projekt, einer Film- und Serienplattform namens "FilmFolio", gibt es mehrere Herausforderungen, die identifiziert werden müssen:

1. Benutzerregistrierung und -anmeldung: Die Implementierung eines sicheren und benutzerfreundlichen Registrierungs- und Anmeldeprozesses ist entscheidend. Es müssen Fehlerbehandlungen für ungültige E-Mail-Adressen, bereits verwendete E-Mail-Adressen und schwache Passwörter implementiert werden.

2. Datenverwaltung: Die Anwendung muss in der Lage sein, Benutzerdaten, einschließlich Favoriten, sicher zu speichern und abzurufen. Hierbei ist die Integration mit Firebase Firestore wichtig, um eine effiziente Datenverwaltung zu gewährleisten.

3. API-Integration: Die Anwendung nutzt die API von The Movie Database (TMDB) für die Suche und Anzeige von Filmen und Serien. Es ist wichtig, sicherzustellen, dass die API-Anfragen korrekt verarbeitet werden und dass die Benutzeroberfläche die Ergebnisse ansprechend darstellt.[[5]](#footnote-5)

4. Themenwechsel: Die Möglichkeit, zwischen einem hellen und einem dunklen Modus zu wechseln, sollte nahtlos funktionieren und die Benutzereinstellungen speichern.

5. Fehlerbehandlung: Eine umfassende Fehlerbehandlung ist erforderlich, um den Benutzern klare und hilfreiche Fehlermeldungen anzuzeigen, insbesondere bei Netzwerkproblemen oder ungültigen Eingaben.

6. Performance-Optimierung: Die Anwendung sollte optimiert werden, um eine schnelle Ladezeit und eine reibungslose Benutzererfahrung zu gewährleisten, insbesondere bei der Anzeige von Inhalten und der Verarbeitung von Benutzereingaben.[[6]](#footnote-6)

## 2.2 Zielgruppe

Filmliebhaber: Personen, die regelmäßig Filme und Serien konsumieren und nach einer Plattform suchen, die ihnen hilft, neue Inhalte zu entdecken und ihre Favoriten zu speichern.[[7]](#footnote-7)

# Projektbeschreibung

## Funktionen von FimFolio

FilmFolio bietet eine Vielzahl von Funktionen, die in den bereitgestellten Code-Snippets implementiert sind. Hier sind einige der Hauptfunktionen:

1. Benutzerauthentifizierung:

* Benutzer können sich anmelden und registrieren.
* Es gibt eine Fehlerbehandlung für ungültige Anmeldedaten.

2. Themenwechsel:

* Benutzer können zwischen einem hellen und einem dunklen Thema wechseln.
* Die gewählte Thematik wird in der lokalen Speicherung oder in der Firebase-Datenbank gespeichert.

3. Favoritenverwaltung:

* Benutzer können Filme und Serien zu ihren Favoriten hinzufügen.
* Die Favoriten werden in der Firebase-Datenbank gespeichert und können abgerufen werden.[[8]](#footnote-8)

4. Filme und Serien suchen:

* Benutzer können nach Filmen und Serien suchen.
* Die Suchergebnisse werden von der TMDB-API abgerufen und angezeigt.

5. Top-Filme und -Serien anzeigen:

* Die Anwendung zeigt die besten Filme und Serien an, die von der TMDB-API abgerufen werden.[[9]](#footnote-9)

# Technische Umsetzung

## Technologien

### Ein Bild, das Screenshot, Text, Schrift enthält. KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Screenshot, Text enthält. KI-generierte Inhalte können fehlerhaft sein.loadTopMovies und loadTopSeries

Abbildung 5: loadTopSeries Code

Abbildung 6: loadTopMovies Code

Die Funktionen loadTopMovies() und loadTopSeries() rufen über die TMDb-API die bestbewerteten Filme bzw. Serien auf Deutsch ab, verarbeiten die Daten und zeigen sie mit displayTopContent() an, wobei loadTopMovies() zudem die Seitenzahl erhöht.

### searchMovies und performSearch

Die Funktionen searchMovies() und performSearch() ermöglichen die Suche nach Filmen und Serien über die TMDb-API.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 7: searchMovies Code

searchMovies(query): Sucht sowohl Filme als auch Serien basierend auf der Benutzereingabe und zeigt die Ergebnisse auf der Seite an. Die Funktion setzt dabei currentType auf 'movie', um gezielt nur Filme zu verarbeiten.[[10]](#footnote-10)

Ein Bild, das Text, Screenshot, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 8: performSearch Code

performSearch(): Führt parallelisierte Anfragen für Filme und Serien durch, berechnet einen Score zur Sortierung der Ergebnisse basierend auf Bewertung, Popularität und Anzahl der Bewertungen. Anschließend werden die Ergebnisse im Grid-Layout angezeigt.

### showDetailCard

Ein Bild, das Text, Elektronik, Screenshot, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 9: showDetailCard Code

Die Funktion showDetailCard(item) zeigt detaillierte Informationen zu einem ausgewählten Film oder einer Serie an. Sie lädt zusätzlich Streaming-Informationen von TMDb und zeigt den Anbieter an. Der Nutzer kann Inhalte als "Watched", "Planned to Watch" oder "Favorit" markieren. Firebase wird genutzt, um diese Informationen für den angemeldeten Benutzer zu speichern.[[11]](#footnote-11)

### Firebase-Integration: loadFavorites, loadWatched, loadPlanned

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 10: loadFavorites Code

loadFavorites(): Lädt alle Favoriten des angemeldeten Nutzers aus Firebase und zeigt sie im Grid-Layout an.

Ein Bild, das Text, Screenshot, Display, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 11: loadWatched Code

Abbildung 7 loadWatched

loadWatched(): Zeigt bereits gesehene Titel aus der Firebase-Datenbank an.[[12]](#footnote-12)

Ein Bild, das Text, Screenshot, Display, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 12: loadPlanned Code

* loadPlanned(): Listet alle geplanten Filme und Serien auf, die der Nutzer später ansehen möchte.

### Benutzer-Authentifizierung: onAuthStateChanged und signOut

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 13: onAuthStateChanged Code

onAuthStateChanged(auth, (user)): Überwacht den Anmeldezustand des Nutzers und passt die Navigation dynamisch an. Bei angemeldeten Nutzern werden Favoriten, Watched- und Planned-Inhalte geladen.[[13]](#footnote-13)

### 4.1.6 Firestore Regeln

Ein Bild, das Text, Screenshot, Software, Display enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 14: Firestore Regeln

Diese Firestore-Sicherheitsregeln sorgen dafür, dass nur der jeweils angemeldete Benutzer (request.auth.uid) Zugriff auf seine eigenen Dokumente in den Pfaden watched, planned und favorites hat. Konkret dürfen dort nur Anfragen zugelassen werden, wenn der Pfadparameter {userId} der UID des angemeldeten Users entspricht – so bleibt jeder Benutzer auf seine eigenen Daten beschränkt.[[14]](#footnote-14)

### 4.1.7 Beispiel eines Firestore Schreibvorgang

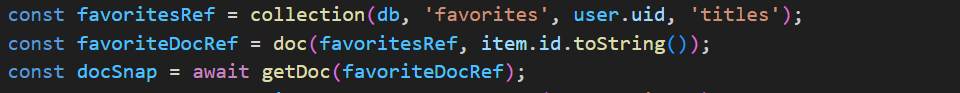


Abbildung 15: Schreibvorgang

Wenn es die Collection favorites oder die Unter-Collection titles für user.uid noch nicht gibt, legt Firestore sie beim Schreiben mit setDoc(...) an. Dasselbe gilt für deine anderen Collections wie watched oder planned.[[15]](#footnote-15)

# Herausforderungen

## Technische Herausforderungen

### Unendliche Anmeldeschleife

Problem:

Nach dem Anmelden konnte sich der Nutzer weiterhin erneut anmelden, was zu einer unendlichen Schleife führte.

Ursache:

Fehlerhafte Implementierung der Authentifizierungslogik in Firebase, wodurch der Anmeldestatus nicht korrekt überwacht wurde.

Lösung:

Die Funktion onAuthStateChanged(auth, (user) => {...}) sorgt nun dafür, dass die Navigation dem tatsächlichen Anmeldestatus entspricht.[[16]](#footnote-16)

### Wechsel von SQL zu Firebase

Problem:

Die Nutzung einer SQL-Datenbank führte zu Problemen bei der Frontend-Backend-Verbindung und der Benutzerverwaltung.

Ursache:

SQL erforderte eine komplexe Backend-Infrastruktur, die unser Projekt überforderte.

Lösung:

Firebase bot eine einfache Authentifizierung und Datenbankverwaltung (Firestore), was den Workflow erheblich erleichterte.[[17]](#footnote-17)

### 5.1.3 Designprobleme in der Detailansicht

Problem:

Das Layout der Detailansicht verschob sich häufig, was das Design unprofessionell wirken ließ.

Ursache:

Fehlende CSS-Regeln und dynamische Inhalte, die nicht korrekt gerendert wurden.[[18]](#footnote-18)

Lösung:

Ein Grid-Layout und flexible CSS-Einheiten (flex, grid) stabilisierten das Design.

# Ergebnisse

## Funktionalität

### Benutzerverwaltung

Ein Bild, das Screenshot, Text, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 16 Login Screenshot

Registrierung und Anmeldung: Nutzer können sich sicher über Firebase authentifizieren. Eingaben werden validiert, und der Anmeldestatus wird persistent gespeichert.

Ein Bild, das Text, Screenshot, Multimedia-Software, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 17 Favoriten Screenshot

Benutzerprofile: Jeder Nutzer kann seine eigenen Favoriten, „Watched“- und „Planned to Watch“-Listen verwalten.[[19]](#footnote-19)

### Filme- und Serien-Suche

Ein Bild, das Text, Screenshot, Multimedia-Software, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 18 Film/Serie Suche Screenshot

Dynamische Suchfunktion: Nutzer können Filme und Serien in Echtzeit suchen. Suchergebnisse werden basierend auf Bewertung, Popularität und Anzahl der Bewertungen sortiert

### Anzeige von Top-Inhalten

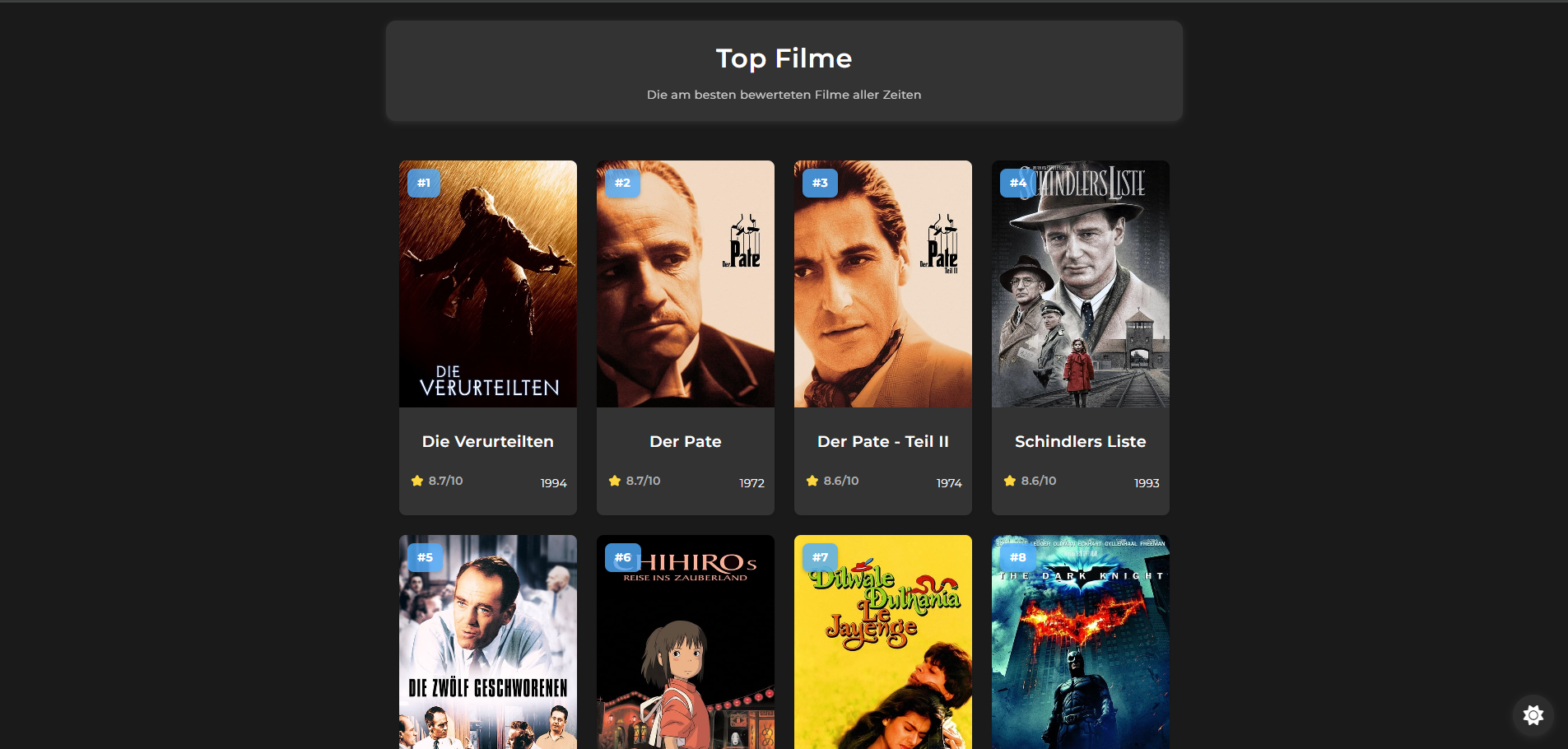


Abbildung 19: Top Filme Screenshot

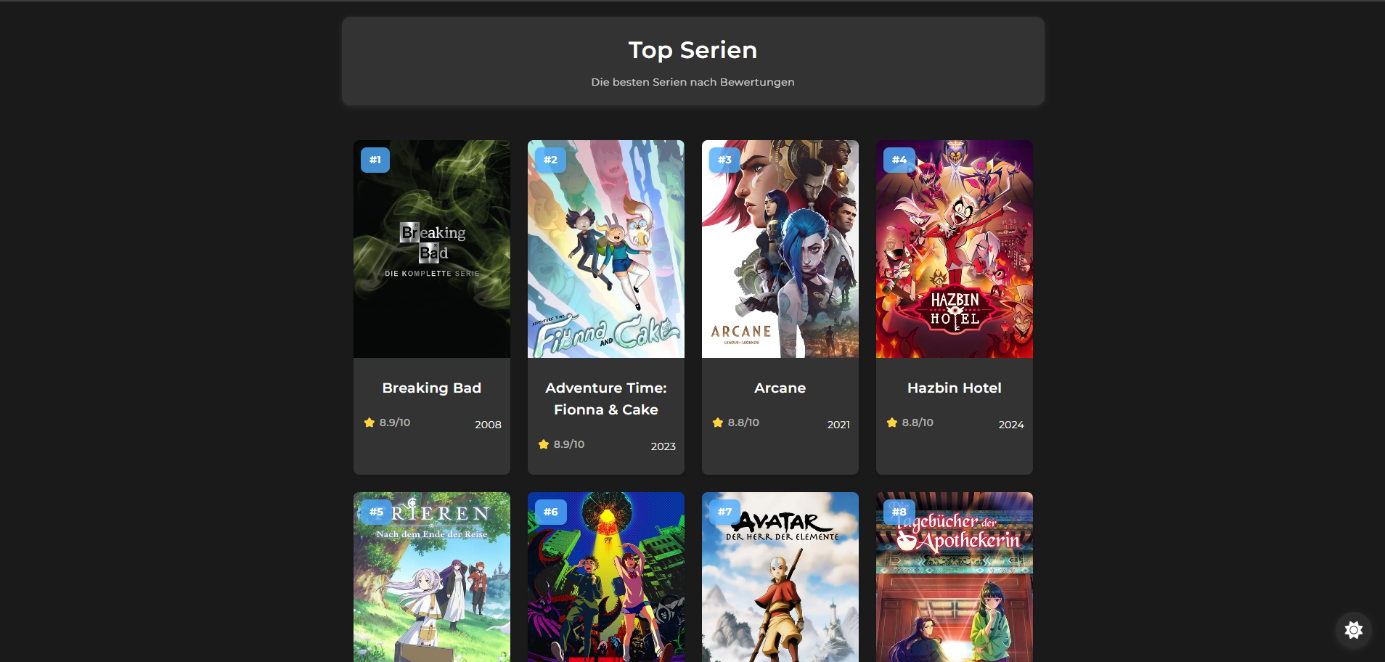


Abbildung 20: Top Serien Screenshot

Top 100 Filme und Serien: Die Anwendung zeigt die am besten bewertete Inhalte basierend auf der TMDb-API an.[[20]](#footnote-20)

# Fazit und Ausblick

## Zusammenfassung

Mit dem Projekt FilmFolio haben wir eine Website entwickelt, die ähnlich wie IMDb funktioniert. Dabei haben wir viel über Webentwicklung gelernt, vor allem durch die Nutzung von Firebase, der TMDb-API und moderner Webtechnologien wie JavaScript, CSS-Grid und flexible Layouts.

Am Anfang gab es einige Schwierigkeiten, zum Beispiel, dass man sich unendlich oft anmelden konnte oder dass das Design der Detailansicht manchmal verrutscht ist. Besonders herausfordernd war es, dass die SQL-Datenbank nicht richtig funktioniert hat. Der Wechsel zu Firebase war dann aber die richtige Entscheidung, weil es die Benutzerverwaltung und die Datenspeicherung viel einfacher gemacht hat.

Am Ende konnten wir eine Website erstellen, auf der Nutzer Filme und Serien suchen, Details anschauen und eigene Favoriten- und Watchlists erstellen können. Die Funktionen wie „Watched“, „Planned to Watch“ und „Favoriten“ kamen gut an. Unsere Testnutzer fanden die Seite insgesamt sehr benutzerfreundlich.

Insgesamt hat uns FilmFolio nicht nur technisch weitergebracht, sondern wir haben auch viel über Teamarbeit und Projektmanagement gelernt. Diese Erfahrungen werden uns sicher bei zukünftigen Projekten helfen.[[21]](#footnote-21)

## Zukünftige Entwicklungen

Um FilmFolio weiter zu verbessern, haben wir folgende Ideen und Ziele:

* Erweiterung der Benutzerprofile:
  + Eigene Rezensionen schreiben
  + Bewertungen abgeben (Sterne, Kommentare)
* Optimierung der Suchfunktion:
  + Mehr Filtermöglichkeiten (z. B. Genre, Erscheinungsjahr, Bewertung)
  + Erweiterung der Sortierfunktionen (z. B. nach Popularität, Erscheinungsdatum)
* Internationalisierung:
  + Mehrsprachige Version der Website (z. B. Englisch, Französisch)
* Benachrichtigungssystem:
  + Push-Benachrichtigungen bei neuen Inhalten in den Favoritenlisten
* Mobile App:
  + Entwicklung einer App für iOS und Android
  + Optimierung der Nutzererfahrung auf mobilen Geräten
* Design-Verbesserungen:
  + Anpassung des Layouts für eine noch bessere Usability
  + Einsatz von Animationen und visuellen Effekten[[22]](#footnote-22)

# Anhang

# Literaturverzeichnis

Kos, U. (21. 02 2025).

Özcan, E. (21. 02 2025).

# Abbildungsverzeichnis

[Abbildung 1: Mockup Startseite 4](#_Toc191628849)

[Abbildung 2: Mockup Detailansicht 4](#_Toc191628850)

[Abbildung 3: Projekthandbuch 5](#_Toc191628851)

[Abbildung 4: Projektumwelten 6](#_Toc191628852)

[Abbildung 5: loadTopSeries Code 9](#_Toc191628853)

[Abbildung 6: loadTopMovies Code 9](#_Toc191628854)

[Abbildung 7: searchMovies Code 9](#_Toc191628855)

[Abbildung 8: performSearch Code 10](#_Toc191628856)

[Abbildung 9: showDetailCard Code 10](#_Toc191628857)

[Abbildung 10: loadFavorites Code 11](#_Toc191628858)

[Abbildung 11: loadWatched Code 11](#_Toc191628859)

[Abbildung 12: loadPlanned Code 12](#_Toc191628860)

[Abbildung 13: onAuthStateChanged Code 12](#_Toc191628861)

[Abbildung 14: Firestore Regeln 13](#_Toc191628862)

[Abbildung 15: Schreibvorgang 13](#_Toc191628863)

[Abbildung 16 Login Screenshot 15](#_Toc191628864)

[Abbildung 17 Favoriten Screenshot 16](#_Toc191628865)

[Abbildung 18 Film/Serie Suche Screenshot 16](#_Toc191628866)

[Abbildung 19: Top Filme Screenshot 17](#_Toc191628867)

[Abbildung 20: Top Serien Screenshot 17](#_Toc191628868)

1. (Kos, 2025) [↑](#footnote-ref-1)
2. (Özcan, 2025) [↑](#footnote-ref-2)
3. (Özcan, 2025) (Kos, 2025) [↑](#footnote-ref-3)
4. (Kos, 2025) [↑](#footnote-ref-4)
5. (Kos, 2025) [↑](#footnote-ref-5)
6. (Özcan, 2025) [↑](#footnote-ref-6)
7. (Özcan, 2025) [↑](#footnote-ref-7)
8. (Kos, 2025) [↑](#footnote-ref-8)
9. (Özcan, 2025) [↑](#footnote-ref-9)
10. (Kos, 2025) (Özcan, 2025) [↑](#footnote-ref-10)
11. (Özcan, 2025) [↑](#footnote-ref-11)
12. (Kos, 2025) [↑](#footnote-ref-12)
13. (Kos, 2025) (Özcan, 2025) [↑](#footnote-ref-13)
14. (Kos, 2025) [↑](#footnote-ref-14)
15. (Kos, 2025) [↑](#footnote-ref-15)
16. (Kos, 2025) [↑](#footnote-ref-16)
17. (Kos, 2025) [↑](#footnote-ref-17)
18. (Özcan, 2025) [↑](#footnote-ref-18)
19. (Kos, 2025) [↑](#footnote-ref-19)
20. (Özcan, 2025) [↑](#footnote-ref-20)
21. (Özcan, 2025) [↑](#footnote-ref-21)
22. (Emirhan, 2025) [↑](#footnote-ref-22)