Kos Umut, Özcan Emirhan

4bWI  HTL Dornbirn

Filmfolio

Inhalt

[1. Einleitung 2](#_Toc189833996)

[1.1 Hintergrund 2](#_Toc189833997)

[1.2 Ziel der Arbeit 2](#_Toc189833998)

[2. Problemstellung 2](#_Toc189833999)

[2.1 Identifikation des Problems 2](#_Toc189834000)

[2.2 Zielgruppe 2](#_Toc189834001)

[3. Projektbeschreibung 2](#_Toc189834002)

[3.1 Funktionen von FimFolio 2](#_Toc189834003)

[4. Technische Umsetzung 2](#_Toc189834004)

[4.1 Technologien 2](#_Toc189834005)

[5. Herausforderungen 2](#_Toc189834006)

[5.1 Technische Herausforderungen 2](#_Toc189834007)

[6. Ergebnisse 2](#_Toc189834008)

[6.1 Funktionalität 2](#_Toc189834009)

[6.2 Feedback 2](#_Toc189834010)

[7. Fazit und Ausblick 2](#_Toc189834011)

[7.1 Zusammenfassung 2](#_Toc189834012)

[7.2 Zukünftige Entwicklungen 2](#_Toc189834013)

[8. Anhang 3](#_Toc189834014)

[8.1 Quellcode 3](#_Toc189834015)

[8.2 Screenshots 3](#_Toc189834016)

[9. Literaturverzeichnis 3](#_Toc189834017)

# Einleitung

## Projektteam

Das Projektteam besteht aus:

* Micheal Leeb – Projektauftraggeber
* Umut Kos – Projektleiter
* Emirhan Özcan - Projektmitglied

## Hintergrund

Projektauftrag von Herr Leeb 05.11.2024, jeder konnte sich etwas selber 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)

# 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.[[2]](#footnote-2)

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.[[3]](#footnote-3)

## 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.[[4]](#footnote-4)

# 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.[[5]](#footnote-5)

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.[[6]](#footnote-6)

# 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 1: loadTopSeries Code

Abbildung 2: 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 3: 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.[[7]](#footnote-7)

Ein Bild, das Text, Screenshot, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 4: 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 5: 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.[[8]](#footnote-8)

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

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

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 6: 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 7: loadWatched Code

Abbildung 7 loadWatched

**loadWatched()**: Zeigt bereits gesehene Titel aus der Firebase-Datenbank an.[[9]](#footnote-9)

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

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 8: 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 9: 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.[[10]](#footnote-10)

### 4.1.6 Firestore Regeln

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

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 10: 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.[[11]](#footnote-11)

### 4.1.7 Beispiel eines Firestore Schreibvorgang

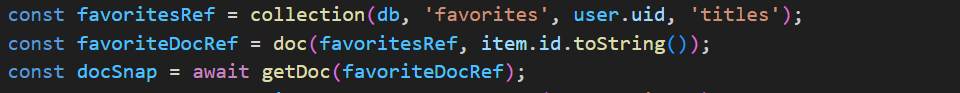


Abbildung 11: Schreibvorgang

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.[[12]](#footnote-12)

# 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.[[13]](#footnote-13)

### **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.[[14]](#footnote-14)

### 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.[[15]](#footnote-15)

**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 12 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 13 Favoriten Screenshot

**Benutzerprofile:** Jeder Nutzer kann seine eigenen Favoriten, „Watched“- und „Planned to Watch“-Listen verwalten.[[16]](#footnote-16)

### Filme- und Serien-Suche

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

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 14 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

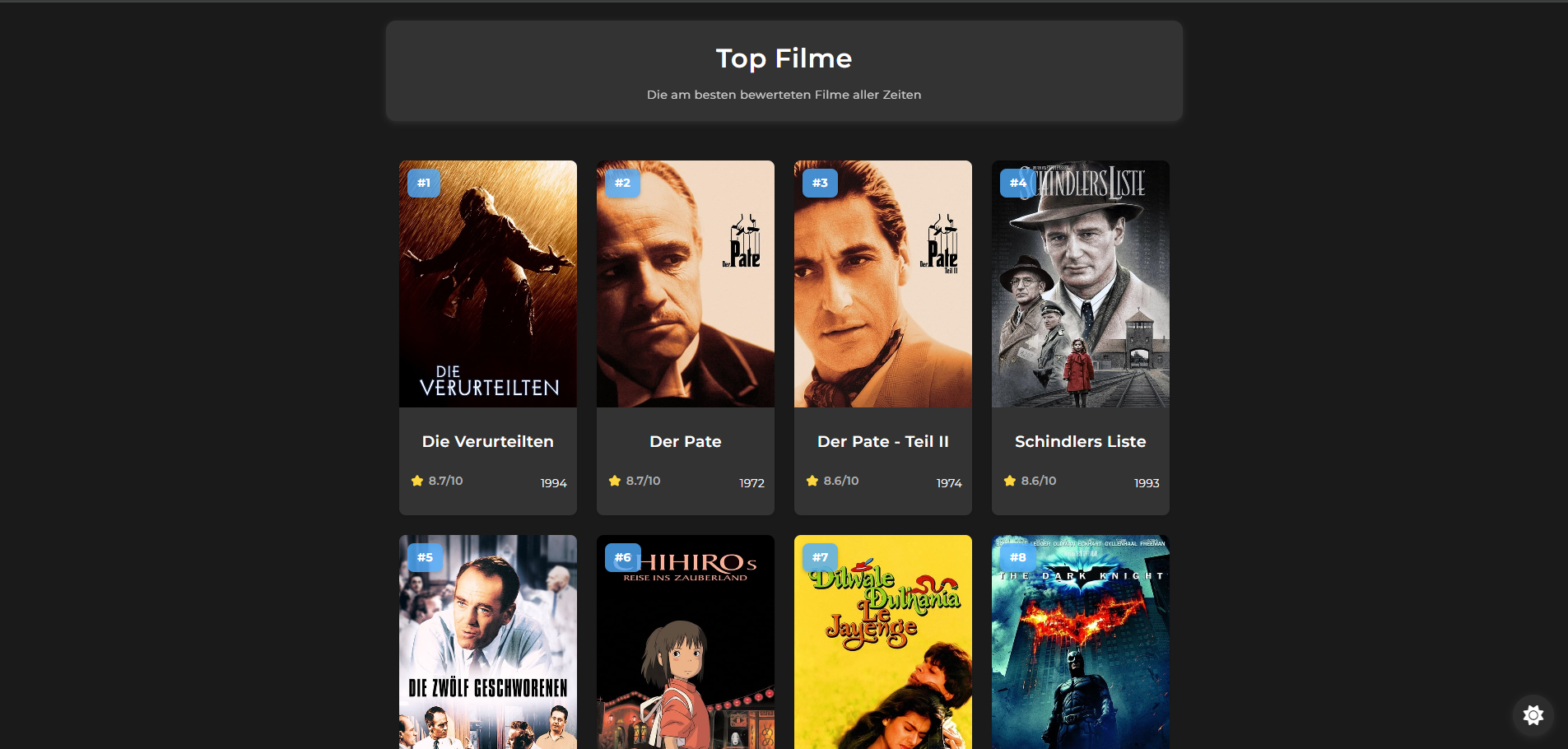


Abbildung 15: Top Filme Screenshot

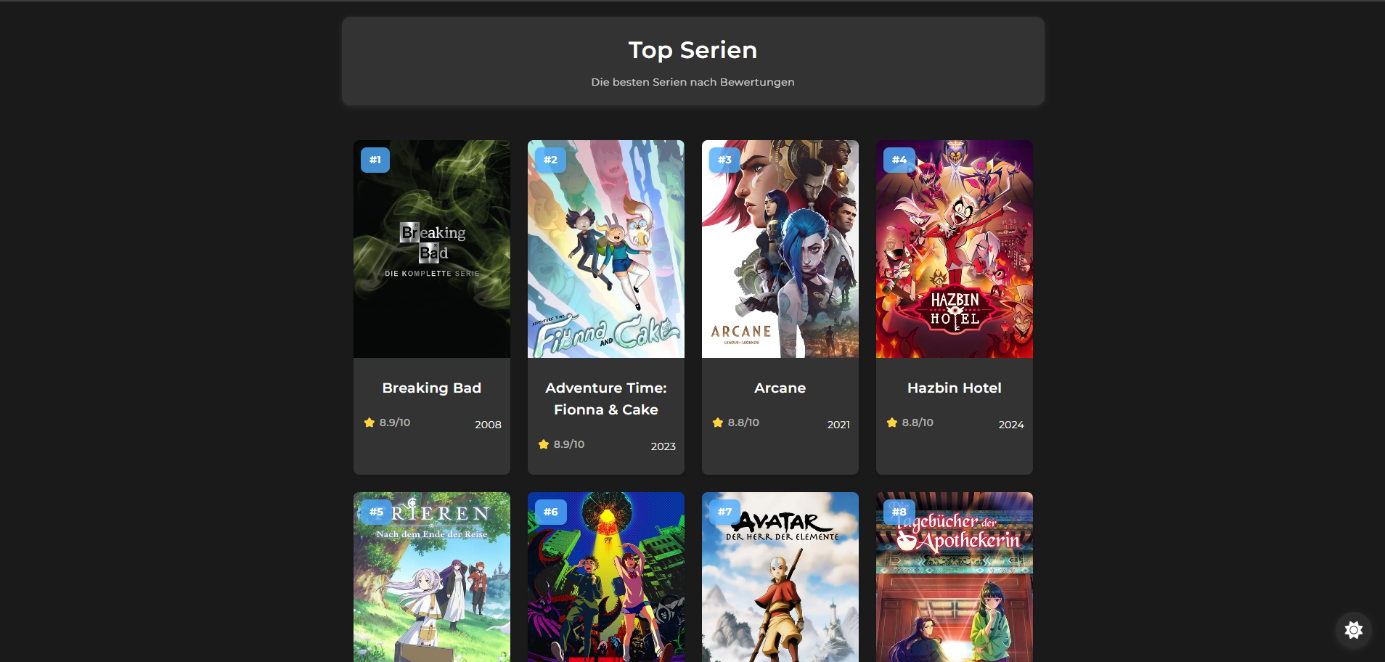


Abbildung 16: Top Serien Screenshot

**Top 100 Filme und Serien:** Die Anwendung zeigt die am besten bewertete Inhalte basierend auf der TMDb-API an.[[17]](#footnote-17)

# 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.[[18]](#footnote-18)

## 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[[19]](#footnote-19)

# Anhang

## Quellcode

## Screenshot

# Literaturverzeichnis

Kos, U. (21. 02 2025).

Özcan, E. (21. 02 2025).

[Abbildung 1: loadTopSeries Code 5](#_Toc191485359)

[Abbildung 2: loadTopMovies Code 5](#_Toc191485360)

[Abbildung 3: searchMovies code 5](#_Toc191485361)

[Abbildung 4: performSearch Code 6](#_Toc191485362)

[Abbildung 5: showDetailCard Code 6](#_Toc191485363)

[Abbildung 6: loadFavorites Code 7](#_Toc191485364)

[Abbildung 7: loadWatched Code 7](#_Toc191485365)

[Abbildung 8: loadPlanned Code 8](#_Toc191485366)

[Abbildung 9: onAuthStateChanged Code 8](#_Toc191485367)

[Abbildung 10: Firestore Regeln 9](#_Toc191485368)

[Abbildung 11: Schreibvorgang 9](#_Toc191485369)

[Abbildung 12 Login Screenshot 11](#_Toc191485370)

[Abbildung 13 Favoriten Screenshot 12](#_Toc191485371)

[Abbildung 14 Film/Serie Suche Screenshot 12](#_Toc191485372)

[Abbildung 15: Top Filme Screenshot 13](#_Toc191485373)

[Abbildung 16: Top Serien Screenshot 13](#_Toc191485374)

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