Lumen Filmhaus

Projektdokumentation SOL2

Leon Egli

leon.egli@student.ipso.ch



Inhalt

[1 Einleitung 4](#_Toc199006292)

[2 Informieren 5](#_Toc199006293)

[2.1 Ist Zustand 5](#_Toc199006294)

[2.2 Soll-Zustand 5](#_Toc199006295)

[2.3 Projektziel definieren 6](#_Toc199006296)

[2.3.1 Ziel 6](#_Toc199006297)

[2.3.2 Was soll erreicht werden? 7](#_Toc199006298)

[2.3.3 Erfolgskriterien 7](#_Toc199006299)

[2.4 Zielgruppe & Anforderungen 8](#_Toc199006300)

[2.4.1 Zielgruppe 8](#_Toc199006301)

[2.4.2 Kino-Mitarbeitende / Verwaltung 8](#_Toc199006302)

[2.4.3 Anforderungen 9](#_Toc199006303)

[2.5 Vorkenntnisse 10](#_Toc199006304)

[2.6 User Flow 11](#_Toc199006305)

[3 Planen 12](#_Toc199006306)

[3.1 Arbeitspakete 12](#_Toc199006307)

[3.2 Zeitplanung / Gantt 15](#_Toc199006308)

[3.3 Ressourcenplanung 16](#_Toc199006309)

[3.3.1 Technische Hilfsmittel (Hardware) 16](#_Toc199006310)

[3.3.2 Software & Tools (Entwicklung) 16](#_Toc199006311)

[3.3.3 Software & Tools (Organisation & Planung) 17](#_Toc199006312)

[3.4 Projektstruktur 18](#_Toc199006313)

[3.5 Risikoanalyse 19](#_Toc199006314)

[3.6 Wireframes 21](#_Toc199006315)

[3.6.1 Home / Home + Menü 22](#_Toc199006316)

[3.6.2 Zeit / Programm 23](#_Toc199006317)

[3.6.3 Filmdetails 24](#_Toc199006318)

[3.6.4 Kontakt 24](#_Toc199006319)

[3.6.5 Reservation 25](#_Toc199006320)

[3.7 Datenmodell 25](#_Toc199006321)

[3.7.1 Grundidee 25](#_Toc199006322)

[3.7.2 Zusammenfassung 26](#_Toc199006323)

[3.8 Meilenstein Planung 26](#_Toc199006324)

[4 Entscheiden 27](#_Toc199006325)

[4.1 Technologie-Entscheidungen 27](#_Toc199006326)

[4.2 Methodische Entscheidungen 27](#_Toc199006327)

[4.3 Organisatorische Entscheidungen 28](#_Toc199006328)

[4.4 Risiko-Management 28](#_Toc199006329)

[5 Realisieren 29](#_Toc199006330)

[5.1 Backend 29](#_Toc199006331)

[5.1.1 Entwicklungsumgebung und Tools 29](#_Toc199006332)

[5.1.2 Projektstruktur 29](#_Toc199006333)

[5.1.3 Datenbank und Migrationen 29](#_Toc199006334)

[5.1.4 Geschäftslogik und Validierung 30](#_Toc199006335)

[5.1.5 API-Controller im Überblick 30](#_Toc199006336)

[5.1.6 Swagger-Integration 31](#_Toc199006337)

[5.1.7 CORS und Middleware 31](#_Toc199006338)

[5.1.8 Test und Qualitätssicherung 31](#_Toc199006339)

[5.2 Frontend 32](#_Toc199006340)

[5.2.1 Entwicklungsumgebung und Aufbau 32](#_Toc199006341)

[5.2.2 Seitenstruktur und Funktion 33](#_Toc199006342)

[5.2.3 Abgrenzung 33](#_Toc199006343)

[5.2.4 Responsive Design 34](#_Toc199006344)

[5.2.5 JavaScript-Funktionen im Detail 34](#_Toc199006345)

[5.2.6 Benutzerfreundlichkeit und Barrierefreiheit 34](#_Toc199006346)

[5.2.7 Einbettung der API 35](#_Toc199006347)

[5.2.8 Farbkonzept und Designphilosophie 35](#_Toc199006348)

# Einleitung

In der heutigen digitalen Welt erwarten Kunden nicht nur ein gutes Angebot, sondern auch eine schnelle und unkomplizierte Art, Informationen abzurufen und Dienstleistungen in Anspruch zu nehmen – insbesondere online. Dies betrifft auch die Kinobranche. Viele moderne Kinos verfügen mittlerweile über benutzerfreundliche Webseiten, auf denen sich Besucher über Filme informieren, Vorstellungen wählen und Sitzplätze direkt online reservieren können.

Das Kino Lumen Filmhaus möchte diese digitale Entwicklung nutzen, um seinen Kunden einen verbesserten Zugang zum Filmangebot sowie die Möglichkeit zur Online-Reservation zu bieten. Derzeit besteht keine eigene Webplattform, über die dies möglich ist. Die Informationen zum aktuellen Programm werden grösstenteils analog oder über externe Quellen kommuniziert, was insbesondere für eine Jüngere, online-affine Zielgruppe nicht mehr zeitgemäss ist.

Im Rahmen dieses Projekts wird ein vollständiger Webauftritt für das Lumen Filmhaus konzipiert und realisiert. Die Website soll den Besuchern nicht nur ermöglichen, sich über aktuelle Filme, Vorstellungen und das Kino selbst zu informieren, sondern auch die direkte Online-Reservation von Sitzplätzen ermöglichen. Die Umsetzung erfolgt in einer modularen Architektur mit klarer Trennung von Frontend und Backend. Das Frontend wird mit HTML, CSS und JavaScript umgesetzt, während das Backend mit C# und ASP.NET Core Web API entwickelt wird. Die beiden Komponenten kommunizieren über standardisierte HTTP-Requests (REST API).

Neben dem funktionalen Ziel, eine moderne und benutzerfreundliche Website zu entwickeln, steht auch der technische Lernaspekt im Vordergrund. Das Projekt bietet die Gelegenheit, praxisnah Erfahrungen im Bereich Webentwicklung, API-Design, Datenmodellierung und Benutzerführung zu sammeln. Darüber hinaus wird das Projekt nach der IPERKA-Methode strukturiert geplant und durchgeführt, um ein methodisches Vorgehen zu gewährleisten und die Qualität der Umsetzung sicherzustellen.

Das Projekt wird innerhalb eines festgelegten Zeitrahmens bis Mitte Juni umgesetzt. Am Ende soll eine fertige Website mit funktionierender Reservationslogik, sauberer Struktur und dokumentiertem Entwicklungsprozess vorliegen. Diese Dokumentation beschreibt im Folgenden detailliert die Schritte von der Informationssammlung über die technische Umsetzung bis hin zur Auswertung.

# Informieren

Im Schritt Informieren werden alle relevanten Informationen zum Projekt gesammelt und dokumentiert. Dies umfasst die Analyse der Ausgangslage, die Zielgruppenbestimmung, die detaillierte Erfassung der funktionalen und nicht-funktionalen Anforderungen sowie die Auswahl der verwendeten Technologien. Ziel ist es, eine fundierte Basis zu schaffen, die die weiteren Projektphasen optimal vorbereitet.

## Ist Zustand

Das Kino Lumen Filmhaus betreibt aktuell keine eigene moderne Website zur Information über das Kinoprogramm oder zur Online-Reservation von Sitzplätzen. Der Ticketverkauf und die Kommunikation über das aktuelle Filmangebot erfolgen grösstenteils analog, über Aushänge im Kino oder über Drittplattformen, die nicht direkt vom Kino verwaltet werden.

Kundinnen und Kunden haben dadurch keine zentrale Anlaufstelle, um sich selbstständig und aktuell über Filme, Vorstellungen oder freie Plätze zu informieren. Für die heutige Zeit, in der Online-Dienste selbstverständlich sind, ist dies nicht mehr zeitgemäss. Besonders jüngere Zielgruppen, die an einfache Online-Reservationen und mobile Informationen gewöhnt sind, könnten dadurch verloren gehen.

Ausserdem fehlt es an einer optischen Repräsentation des Kinos im Internet, die das Angebot attraktiv präsentiert und Vertrauen aufbaut. Ebenso besteht keine Möglichkeit, Prozesse wie Reservationen digital zu erfassen, zu speichern oder auszuwerten.

## Soll-Zustand

Das Ziel dieses Projekts ist es, eine moderne, benutzerfreundliche Website für das Lumen Filmhaus zu realisieren, die sowohl informativ als auch funktional ist. Besucher/innen der Seite sollen sich über aktuelle Filme informieren, Vorstellungen einsehen und direkt online Sitzplätze reservieren können.

Die Website bietet dabei:

* Eine übersichtliche Startseite mit Filmvorschlägen
* Eine Angebotsseite mit Filmplakaten, Beschreibungen, Spieldauer und Spielzeiten
* Eine interaktive Reservationsseite mit Sitzplatzwahl und Formular
* Eine Kontakt- bzw. „Über uns“-Seite mit Informationen zum Kino

Technisch wird die Website modular aufgebaut: Das Frontend basiert auf HTML, CSS und JavaScript und wird unabhängig vom Backend, welches mit ASP.NET Core Web API in C# umgesetzt wird, betrieben. Die Kommunikation zwischen Frontend und Backend erfolgt über HTTP (REST API). Daten wie Filme, Vorstellungen und Reservationen werden zentral über die API verwaltet.

Ziel ist es, dem Kino eine zeitgemässe Online-Präsenz zu geben, die benutzerorientiert, funktional, erweiterbar und optisch ansprechend ist – mit Fokus auf Usability und klarer Struktur.

## Projektziel definieren

### Ziel

Das Ziel dieses Projekts ist die Konzeption und Entwicklung eines modernen, benutzerfreundlichen Webauftritts für das Kino Lumen Filmhaus. Die Website soll sowohl als Informationsplattform über aktuelle Filme und Angebote dienen, als auch ein funktionales Online-Reservationssystem für Sitzplätze bieten.

Die Seite wird mit HTML, CSS und JavaScript im Frontend realisiert und über eine C# ASP.NET Core Web API im Backend mit Daten versorgt. Das Backend stellt Schnittstellen bereit, über die Filme, Vorstellungen und Sitzplatzreservationen verwaltet werden können.

### Was soll erreicht werden?

1. **Informationszugang verbessern**
   * Benutzer sollen sich jederzeit über das Filmprogramm, Spielzeiten und Details zu Filmen informieren können.
2. **Online-Reservation ermöglichen**
   * Nutzer können direkt über die Website einen Film und eine Vorstellung wählen, Sitzplätze reservieren und ihre Kontaktdaten eingeben.
3. **Zugänglichkeit erhöhen**
   * Die Website ist mobilfähig und für verschiedene Bildschirmgrössen optimiert (Responsive Design).
4. **Modulare Architektur**
   * Frontend und Backend sind technisch getrennt und kommunizieren ausschliesslich über HTTP-Requests (REST-AP
5. **Technisches Lernziel**
   * Ziel ist auch die praktische Anwendung und Vertiefung von Webtechnologien wie HTML, CSS, JS und ASP.NET Core sowie der Aufbau einer sauberen Projektstruktur.

### Erfolgskriterien

* Die Website ist funktional erreichbar (lokal oder deployed)
* Eine Reservation kann vollständig durchgeführt werden (inkl. Sitzplatzwahl)
* Daten werden vom Backend geladen und verarbeitet
* Das Design ist übersichtlich, modern und mobilfreundlich
* Die wichtigsten Fehlerfälle werden abgefangen (z. B. doppelte Sitz Wahl, leere Felder)

## Zielgruppe & Anforderungen

### Zielgruppe

Die Website des Lumen Filmhauses richtet sich an folgende primäre Nutzergruppen:

* Kinobesucher/innen
* Personen, die das Kino besuchen möchten und sich vorab online über Filme, Spielzeiten und freie Plätze informieren wollen.
* Altersspanne: hauptsächlich 14–60 Jahre
* Geräte: Smartphone, Tablet, Laptop
* Technisches Niveau: unterschiedlich (von digital-affin bis technik-unerfahren)

#### Bedürfnisse:

* Schneller Überblick über aktuelle Filme
* Einfache Navigation
* Möglichkeit zur Online-Reservation
* Mobile Nutzung

### Kino-Mitarbeitende / Verwaltung

* Interne Nutzer, die Reservationen und das Filmangebot pflegen möchten (in späteren Versionen)

#### Bedürfnisse:

* Überblick über aktuelle Reservationen (optional)
* Technisch saubere Umsetzung für einfache Erweiterbarkeit

### Anforderungen

#### Funktionale Anforderungen (Was muss das System können?)

| **Nr.** | **Beschreibung** |
| --- | --- |
| F1 | Die Website zeigt eine Übersicht aller aktuellen Filme an |
| F2 | Zu jedem Film gibt es eine Detailansicht mit Beschreibung, Dauer, Bild und Vorstellungen |
| F3 | Nutzer/innen können eine Vorstellung auswählen |
| F4 | Nutzer/innen können Sitzplätze interaktiv wählen |
| F5 | Nutzer/innen können ihre Reservation mit Name und E-Mail absenden |
| F6 | Die API verarbeitet die Reservation und speichert sie |
| F7 | Erfolgreiche und fehlerhafte Eingaben werden durch das System rückgemeldet |
| F8 | Die Website ist auf Desktop und Mobilgeräten benutzbar (Responsive Design) |
| F9 | Daten (Filme, Sitze, Reservationen) werden über eine API geladen/gespeichert |

Tabelle 1: Funktionale Anforderungen

#### Nicht-funktionale Anforderungen (Wie soll das System sein?)

| **Nr.** | **Beschreibung** |
| --- | --- |
| N1 | Die Website lädt schnell (auch bei langsamer Verbindung) |
| N2 | Die Bedienung ist intuitiv und selbsterklärend |
| N3 | Der Webauftritt ist optisch ansprechend und dem Thema „Kino“ angepasst |
| N4 | Die Architektur ist sauber getrennt in Frontend (HTML/CSS/JS) und Backend (ASP.NET Core API) |
| N5 | Der Code ist dokumentiert und verständlich strukturiert |
| N6 | Das System ist lokal lauffähig und bei Bedarf online hostbar |

Tabelle 2: Nicht-Funktionale Anforderungen

#### Technische Anforderungen

| **Bereich** | **Technologie** |
| --- | --- |
| Frontend | HTML, CSS, JavaScript |
| Backend | C# mit ASP.NET Core Web API |
| Kommunikation | RESTful API über HTTP |
| Datenhaltung | JSON-Dateien (lokal) oder In-Memory-Storage |
| Entwicklungsumgebung | Visual Studio Code, Postman, Browser |

Tabelle 3: Technische Anforderungen

## Vorkenntnisse

Vor Beginn des Projekts verfügte ich über fundierte Kenntnisse in Webentwicklung mit HTML, CSS, JavaScript sowie C# und ASP.NET Core. Erste Erfahrungen mit Entity Framework, relationalen Datenbanken und responsivem Design konnte ich bereits in früheren Schulprojekten sammeln. Diese Fähigkeiten bildeten die Grundlage für die selbstständige Umsetzung dieses Projekts.

## User Flow

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

KI-generierte Inhalte können fehlerhaft sein.Der User Flow beschreibt den typischen Ablauf, den eine Benutzerin oder ein Benutzer auf der Website des Lumen Filmhauses durchläuft, um eine Sitzplatzreservation vorzunehmen. Der Ablauf ist so gestaltet, dass er einfach, intuitiv und in wenigen Schritten durchführbar ist:

1. **Startseite:**  
   Die Benutzerin oder der Benutzer öffnet die Website und erhält eine Übersicht über aktuelle Filme.
2. **Film auswählen:**  
   Durch Klick auf einen Film gelangt man zu einer Detailseite mit weiteren Informationen und verfügbaren Vorstellungen.
3. **Vorstellung (Datum & Uhrzeit) wählen:**  
   Eine Vorstellung wird aus einer Liste ausgewählt, z. B. „Mittwoch, 20:15 Uhr“.
4. **Sitzplatz auswählen:**  
   Eine interaktive Saalansicht zeigt freie und belegte Plätze. Der Benutzer wählt verfügbare Sitzplätze aus.

Abbildung 1: User Flow

1. **Formular ausfüllen:**  
   Der Benutzer gibt persönliche Daten wie Name und E-Mail-Adresse an.
2. **Reservation absenden:**  
   Nach Klick auf „Reservieren“ werden die Daten an das Backend gesendet und gespeichert.
3. **Bestätigungsseite wird angezeigt:**  
   Eine Nachricht informiert über die erfolgreiche Reservation.

Dieser Ablauf stellt sicher, dass die Benutzer schnell und ohne Umwege eine Reservation vornehmen können – von der ersten Information bis zur fixen Platzbuchung.

# Planen

Im Schritt Planen erfolgt die detaillierte Aufteilung des Projekts in überschaubare Arbeitspakete mit klaren Beschreibungen, Prioritäten und Zeitrahmen. Zudem werden die benötigten Ressourcen definiert, die Projektstruktur festgelegt sowie Risiken analysiert und Massnahmen zur Qualitätssicherung geplant. Diese Phase bildet das organisatorische und methodische Fundament für die erfolgreiche Umsetzung.

## Arbeitspakete

Im Rahmen der Planungsphase wurden die notwendigen Arbeitspakete definiert. Jedes Paket umfasst eine klar abgegrenzte Aufgabe mit zugehöriger Priorität, Zeitrahmen und spezifischen Hinweisen zur Umsetzung. Die Aufgaben orientieren sich an der funktionalen Struktur des Projekts (Frontend, Backend, Design, Testing, Dokumentation) und sind in der Projektmanagement-Software Clickup.com eingetragen.

| **Nr.** | **Arbeitspaket** | **Priorität** | **Zeitraum** | **Beschreibung** |
| --- | --- | --- | --- | --- |
| 1 | Ist-/Soll-Zustand definieren | Hoch | 15.–16. Mai | Analyse der aktuellen Situation und Zielvorstellung des Webauftritts |
| 2 | Projektziel definieren | Hoch | 15.–16. Mai | Klare Zielformulierung mit Nutzen für Benutzer und Betreiber |
| 3 | Zielgruppe & Anforderungen | Hoch | 15.–17. Mai | Sammlung funktionaler und nicht-funktionaler Anforderungen sowie Zielgruppenanalyse |
| 4 | User Flow erstellen | Hoch | 15.–17. Mai | Darstellung des typischen Nutzerwegs: Film Wahl → Sitz Wahl → Reservation |
| 5 | Datenmodell entwerfen | Hoch | 16.–18. Mai | Definition der Datenstrukturen für Filme, Vorstellungen, Sitze und Reservationen |
| 6 | User-Flow skizzieren | Mittel | 17.–18. Mai | Visualisierung oder tabellarische Darstellung des Ablaufs aus Benutzersicht |
| 7 | Wireframes: Startseite / Programm | Hoch | 18.–19. Mai | Grobe Layout-Skizze für Startseite mit Navigation und Filmübersicht |
| 8 | Wireframes: Reservation / Benutzer | Hoch | 18.–19. Mai | Layoutentwurf für Reservation mit Formular und Sitzplatzansicht |
| 9 | Meilenstein Planung | Niedrig | 19. Mai | Markierung des Planungsabschlusses im Projektzeitplan |
| 10 | Frontend-Setup erstellen | Hoch | 19.–20. Mai | Anlegen von HTML-Grundstruktur und Ordnern für CSS, JS, Bilder etc. |
| 11 | Backend-Projekt (ASP.NET Core) | Hoch | 19.–20. Mai | Erstellung des C# Web API-Projekts mit Routing, Middleware und CORS |
| 12 | API: GET /api/filme | Hoch | 20.–21. Mai | Bereitstellung von Dummy-Daten zu Filmen über einen GET-Endpoint |
| 13 | API: GET /api/sitze | Hoch | 21.–22. Mai | Rückgabe belegter Sitze pro Film und Vorstellung über Parameter |
| 14 | API: POST /api/reservieren | Hoch | 22.–23. Mai | Empfangen, validieren und Speichern von Reservationen |
| 15 | Frontend: Filmübersicht einbauen | Hoch | 23.–24. Mai | Anzeige der Filme auf der Startseite anhand von API-Daten |
| 16 | Frontend: Reservationsformular | Hoch | 25.–26. Mai | HTML-Formular zur Erfassung von Reservation (Name, E-Mail etc.) |
| 17 | Sitzplatzwahl (Frontend) | Mittel | 26.–27. Mai | Interaktive Anzeige von verfügbaren und belegten Sitzplätzen (z. B. als Grid) |
| 18 | Erfolgs-/Fehlerseite anzeigen | Mittel | 27.–28. Mai | Rückmeldung an Benutzer über Erfolgs- oder Fehlermeldung nach Absenden |
| 19 | Funktionstest API + UI | Hoch | 28.–29. Mai | Manuelle Tests von Frontend-Funktionalität und API-Anbindung mit Testdaten |
| 20 | Responsive Design umsetzen | Mittel | 29.–30. Mai | Anpassung der Darstellung für mobile und Tablet fähige Endgeräte (Media Queries) |
| 21 | Fehlerbehandlung erweitern | Hoch | 31. Mai – 1. Juni | Validierung im Frontend und Backend (z. B. Pflichtfelder, Doppelsitzprüfung) |
| 22 | Meilenstein Entwicklung | Niedrig | 1. Juni | Zwischenkontrolle und Projektfortschritt-Check |
| 23 | Projekt-Dokumentation schreiben | Hoch | 1.–15. Juni | Erstellung der IPERKA-Dokumentation in Word (mit Screenshots, Codebeispielen etc.) |
| 24 | Präsentation vorbereiten | Hoch | 2.–5. Juni | Entwicklung der Präsentation mit Folien, Bildern und Zusammenfassung |
| 25 | Abschluss & letzte Tests | Hoch | 2.–15. Juni | Finales Durchtesten auf Desktop & Mobil, Korrekturen vor Abgabe |
| 26 | Meilenstein Abschluss | Niedrig | 15. Juni | Projektabschluss markieren, Abgabe der finalen Version + Präsentation |

Tabelle 4: Arbeitspakete

## Ein Bild, das Text, Screenshot, Zahl, Schrift enthält. KI-generierte Inhalte können fehlerhaft sein.Zeitplanung / Gantt

Abbildung 2: Gantt

## Ressourcenplanung

Im Rahmen der Planung wurde festgelegt, welche technischen, materiellen und digitalen Ressourcen für die erfolgreiche Umsetzung des Projekts benötigt werden. Die Ressourcen lassen sich in drei Hauptkategorien unterteilen:

### Technische Hilfsmittel (Hardware)

| **Ressource** | **Verwendung** |
| --- | --- |
| Laptop/PC | Hauptarbeitsgerät für Entwicklung & Dokumentation |
| Maus & Tastatur | Komfort beim Entwickeln und Navigieren |
| Bildschirm | Zusätzlicher Bildschirm für effizientes Arbeiten |
| Smartphone | Tests für Responsiveness auf Mobilgeräten |
| Kosten | Es gab keine Kosten da sämtliche Tools bereits in Besitz waren |

Tabelle 5: Technische Hilfsmittel

### Software & Tools (Entwicklung)

| **Tool** | **Verwendung** |
| --- | --- |
| Visual Studio Code | Hauptentwicklungsumgebung für Frontend und Backend |
| .NET SDK (.NET 8) | Framework zur Erstellung der ASP.NET Core Web API |
| Postman | Testen und Verwalten von API-Endpunkten |
| Browser (Opera/Edge) | Darstellung und Testen der Website im Frontend |
| Live Server (VS Code Extension) | Lokale Vorschau des Frontends im Browser |
| Word | Verfassen der IPERKA-Dokumentation |
| PowerPoint / Canva | Erstellung der Projektpräsentation |
| MariaDB | Datenbank |

Tabelle 6: Software & Tools(Entwicklung)

### Software & Tools (Organisation & Planung)

| **Tool** | **Verwendung** |
| --- | --- |
| Clickup.com | Projektplanung, Zeitplanung, Aufgabenübersicht |
| Draw.io / Balsamiq | Erstellen von User-Flows, Wireframes, Datenmodellen |
| GitHub | Versionskontrolle |

Tabelle 7: Software & Tools (Organisation & Planung)

## Projektstruktur

Das Projekt ist in zwei Hauptbereiche gegliedert: das Frontend und das Backend. Die Trennung dieser Komponenten stellt sicher, dass Benutzeroberfläche und Datenlogik unabhängig voneinander entwickelt und gewartet werden können.

Das Frontend wird mit HTML, CSS und JavaScript umgesetzt und stellt die grafische Benutzeroberfläche dar, über die Nutzer/innen mit dem System interagieren. Es enthält unter anderem die Startseite, die Filmliste, das Reservationsformular sowie eine mögliche Detailansicht einzelner Filme. Alle Dateien werden in einem eigenen Projektordner mit dem Namen lumen-frontend verwaltet. Die Ordnerstruktur umfasst separate Verzeichnisse für CSS-Dateien (Design), JavaScript-Dateien (Funktionalität) sowie Bilder (z. B. Filmplakate, Icons, Logo).

Das Backend basiert auf C# mit ASP.NET Core Web API. Es stellt verschiedene Schnittstellen (REST-APIs) bereit, über die Daten wie Filme, Vorstellungen, Sitzplatzbelegungen und Reservationen verwaltet werden. Das Backend ist in einem separaten Projektordner namens lumen-backend untergebracht. Es enthält Controller für die einzelnen API-Endpunkte, Modellklassen zur Abbildung der Datenstrukturen sowie lokale JSON-Dateien zur Datenspeicherung. Das Backend verarbeitet Anfragen vom Frontend, führt Validierungen durch und gibt passende Antworten im JSON-Format zurück.

Zur besseren Übersicht existieren zusätzlich ein Ordner für die Projektdokumentation (inkl. Screenshots, Dokumentation und Präsentation) sowie alle für das Projekt benötigten Hilfsdateien.

Diese Struktur ermöglicht ein sauberes, wartbares und skalierbares Projekt, bei dem sowohl Frontend- als auch Backend-Komponenten unabhängig getestet und erweitert werden können.

## Risikoanalyse

Bei der Umsetzung eines Webprojekts wie diesem können verschiedene Risiken auftreten, die den Zeitplan oder die Qualität der Lösung beeinträchtigen könnten. Die folgenden Risiken wurden identifiziert und entsprechende Massnahmen zur Vorbeugung oder Lösung definiert:

| **Risiko** | **Beschreibung** | **Auswirkung** | **Gegenmassnahme / Lösung** |
| --- | --- | --- | --- |
| Unklare Anforderungen | Es fehlen Details zur genauen Funktionsweise (z. B. Platzlogik, Datenmodell) | Verzögerungen, Nacharbeiten | Anforderungen frühzeitig schriftlich definieren und regelmässig mit Betreuungsperson abgleichen |
| Zeitmangel durch andere Verpflichtungen | Parallellaufende Schulaufgaben oder private Termine | Verpasste Deadlines, Stress | Frühzeitige Zeitplanung mit Pufferphasen, regelmässige Fortschrittskontrolle |
| Technische Probleme im Backend (ASP.NET Core) | Fehlerhafte Routen, Validierungen oder JSON-Verarbeitung | Funktionsausfälle | Schrittweises Testen mit Postman, Logs verwenden, API in kleinen Einheiten entwickeln |
| Frontend kommuniziert nicht korrekt mit Backend | Fehler bei fetch(), falsche Endpunkte, CORS-Probleme | Keine Datenanzeige oder Fehlerseiten | Backend lokal testen, CORS aktivieren, API vor Integration vollständig prüfen |
| Unerwartete Formatfehler in JSON-Daten | Struktur in JSON-Dateien nicht wie erwartet (z. B. Tippfehler) | API-Absturz | JSON-Dateien mit Tools validieren (z. B. [https://jsonlint.com](https://jsonlint.com/)), strukturierte Modelle nutzen |
| Responsiveness funktioniert nicht auf allen Geräten | Website ist auf Smartphones schwer bedienbar | Schlechte Benutzererfahrung | Mobile-First-Ansatz, gezieltes Testing auf verschiedenen Displaygrössen |
| Reservierungsdaten gehen verloren | Keine echte Datenbank, nur temporäre Speicherung | Datenverlust bei Neustart | Daten als JSON speichern, Backups anlegen, klare Struktur bei Speicherung |
| Fehlende Validierung im Formular | Ungültige oder unvollständige Benutzereingaben | Fehlerhafte oder unbrauchbare Daten | Validierungen im Frontend (JS) und Backend (Modelprüfung) einbauen |
| Projektumfang zu gross | Zu viele Ideen oder Erweiterungen verzögern die Basisfunktionalität | Funktionalität nicht abgeschlossen | Fokus auf Minimalversion (MVP), Erweiterungen erst nach Kernfunktionen umsetzen |

Tabelle 8: Risikoanalyse

## Wireframes

Zur besseren Vorstellung der geplanten Benutzeroberfläche wurden sogenannte Wireframes erstellt. Dabei handelt es sich um einfache, grafische Skizzen, die den grundsätzlichen Aufbau und die Struktur der einzelnen Seiten des Webauftritts visualisieren – ohne Design oder Farben, aber mit Fokus auf Inhalte, Anordnung und Benutzerführung.

Die Wireframes dienen als Planungsgrundlage für die spätere Umsetzung im Frontend. Sie helfen dabei, die Benutzerführung frühzeitig zu überdenken, Funktionen sinnvoll zu platzieren und eine logische Navigationsstruktur zu entwickeln.

Erstellt wurden Wireframes für die folgenden Seiten:

* **Home / Home + Menü**: Übersicht über aktuelle Filme und Navigation
* **Zeit / Programm**: Datum auswählen, Filmprogramm
* **Film**: Einzelheiten, Informationen, Beschreibung, Trailer
* **Reservationsseite**: Formular zur Eingabe von Name, E-Mail und Sitzplatzwahl
* **Kontaktseite**: Um das Kino zu erreichen, über Textnachricht

Durch die visuelle Planung mit Wireframes können mögliche Schwachstellen im Aufbau bereits früh erkannt und angepasst werden, bevor die technische Umsetzung beginnt.

### Home / Home + Menü

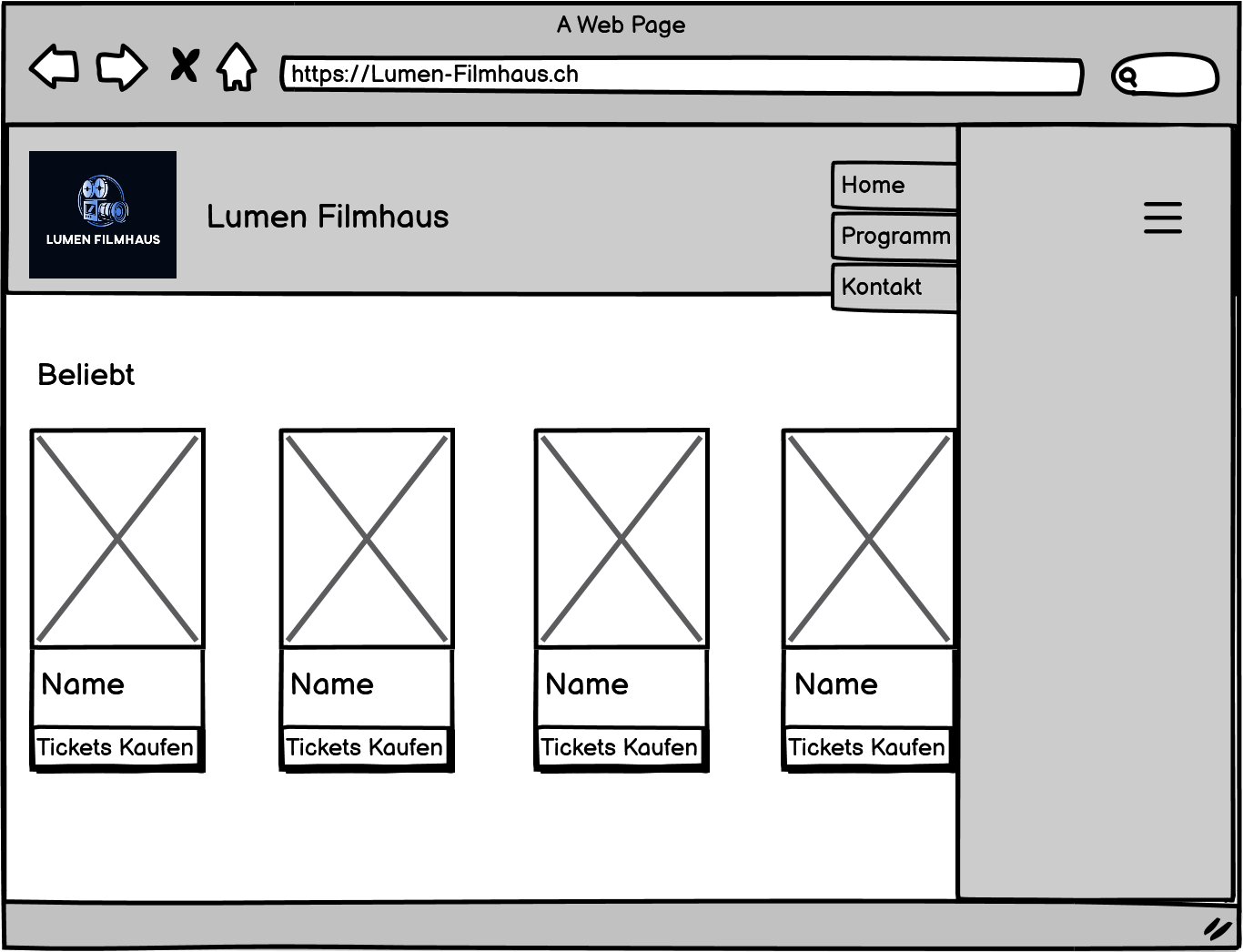


Abbildung 3: Wireframe Home + Menü

Abbildung 4: Wireframe Home

### Zeit / Programm

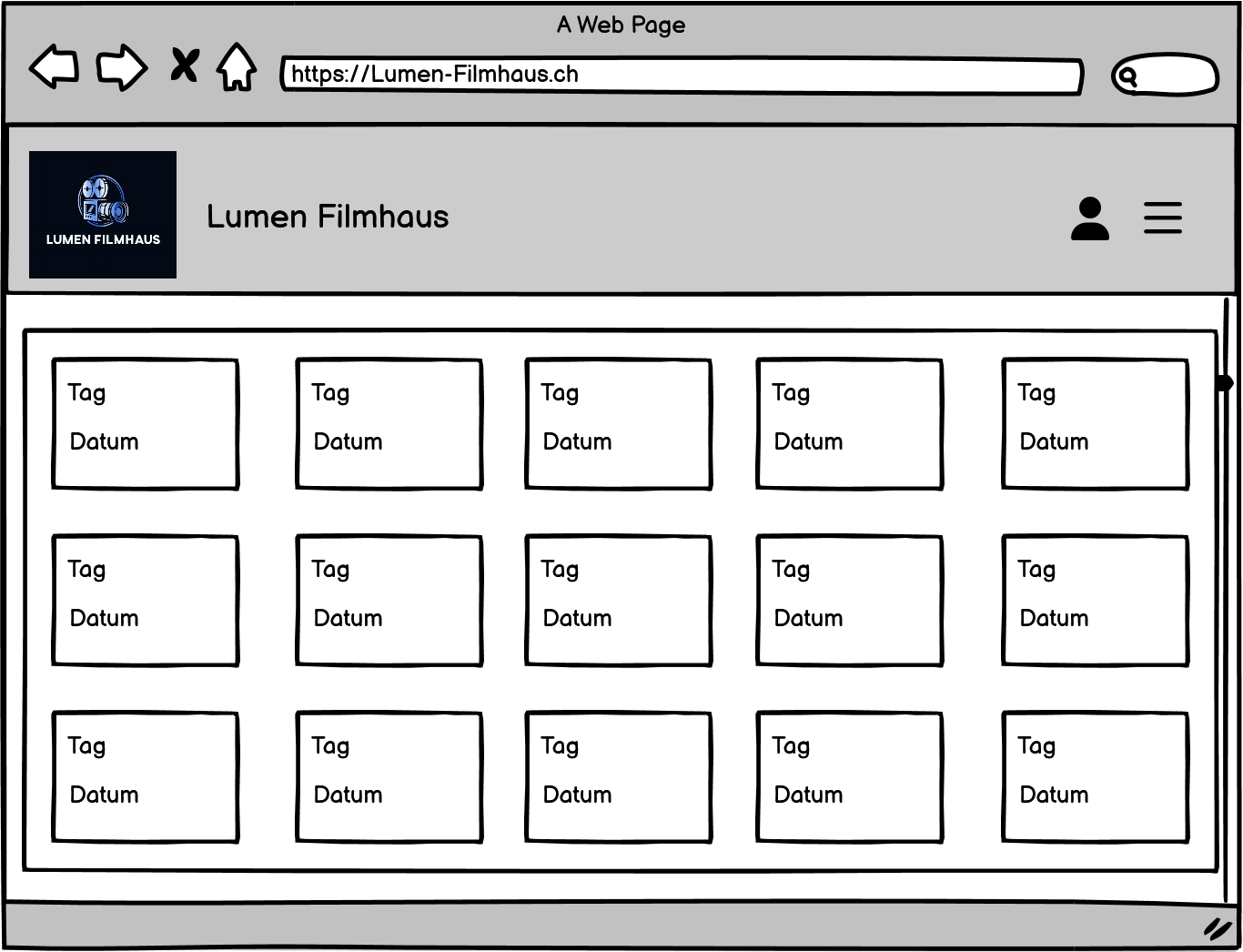


Abbildung 5: Wireframe Zeit

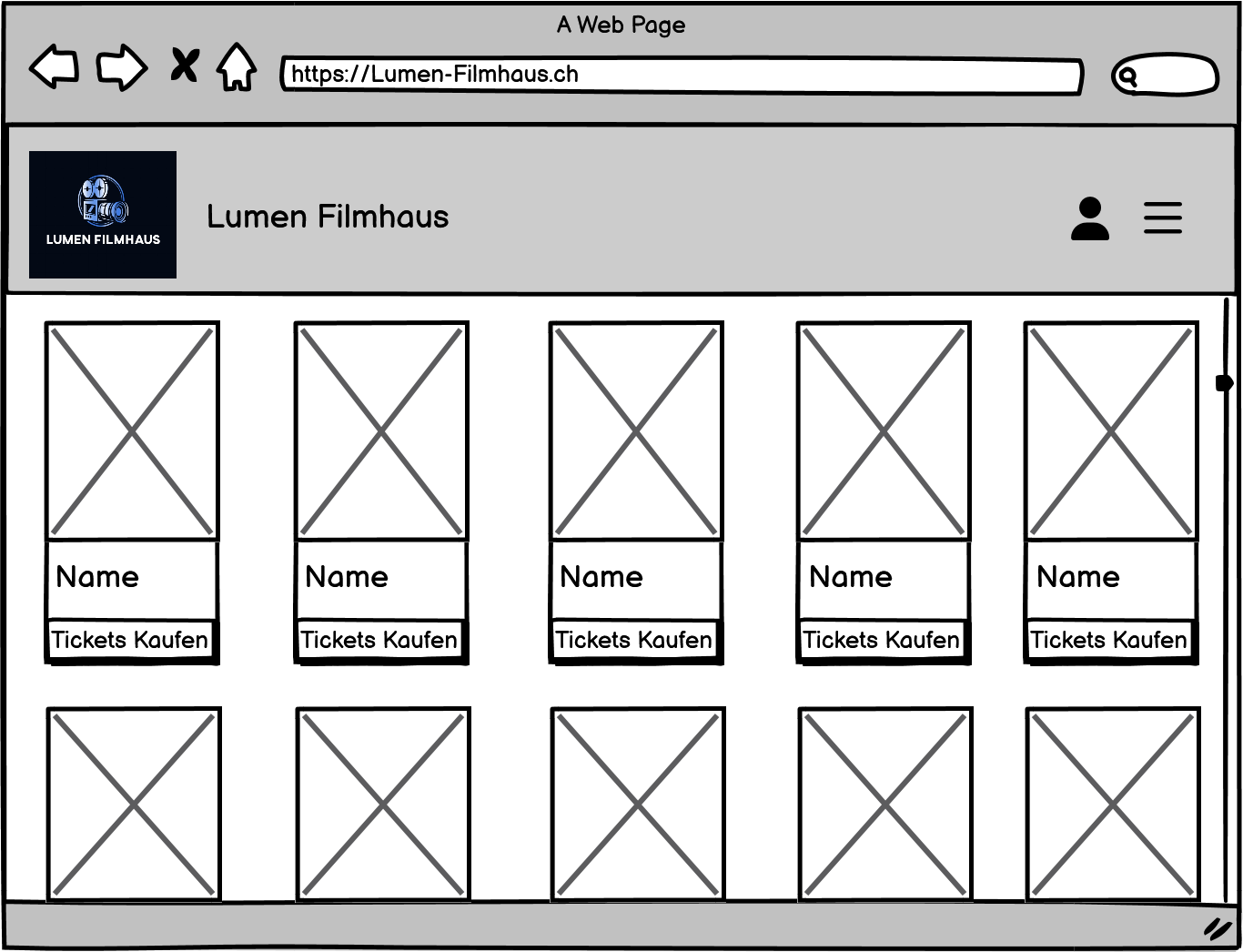


Abbildung 6:Wireframe Programm

### Filmdetails

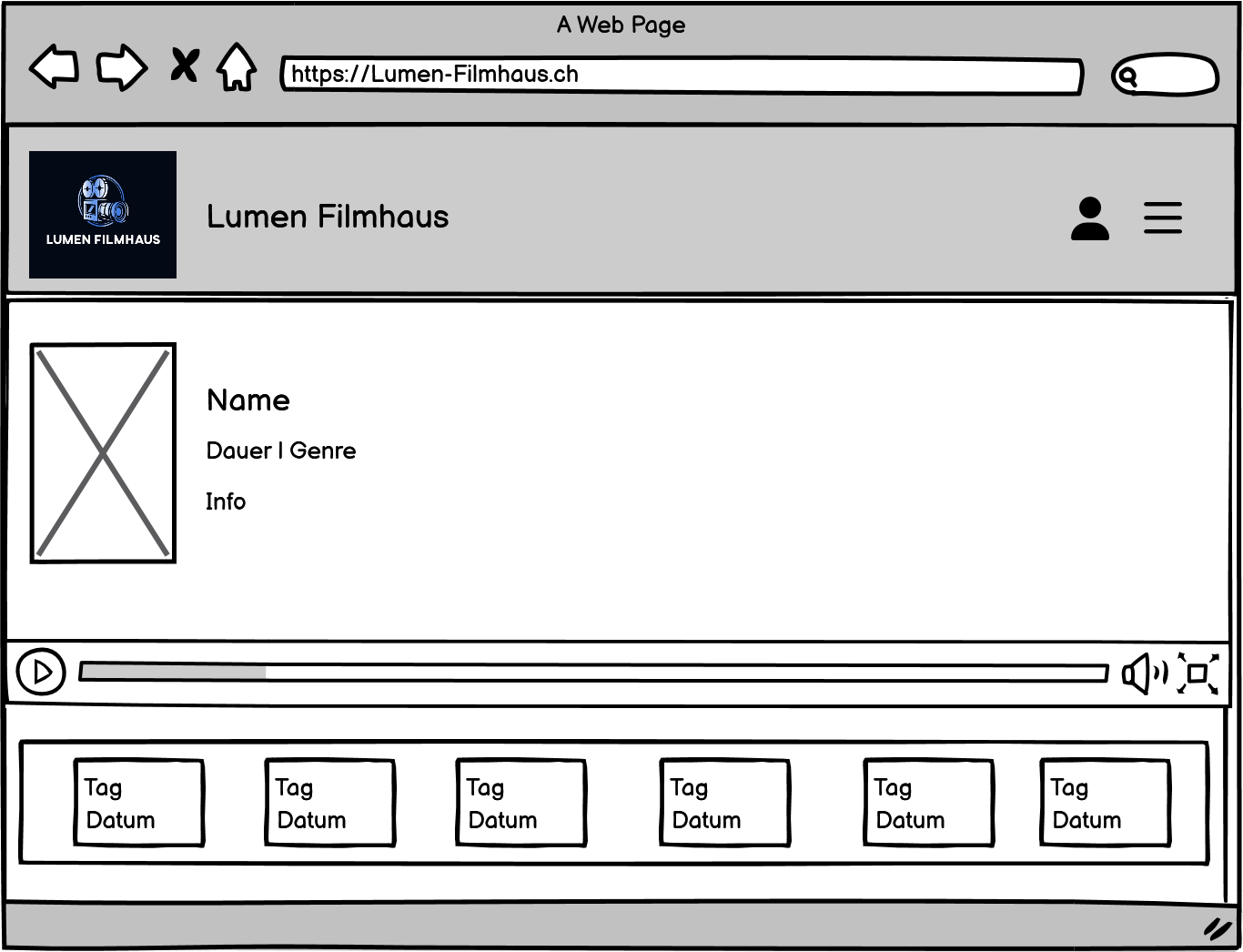


Abbildung 7: Wireframe Filmdetails

### Kontakt

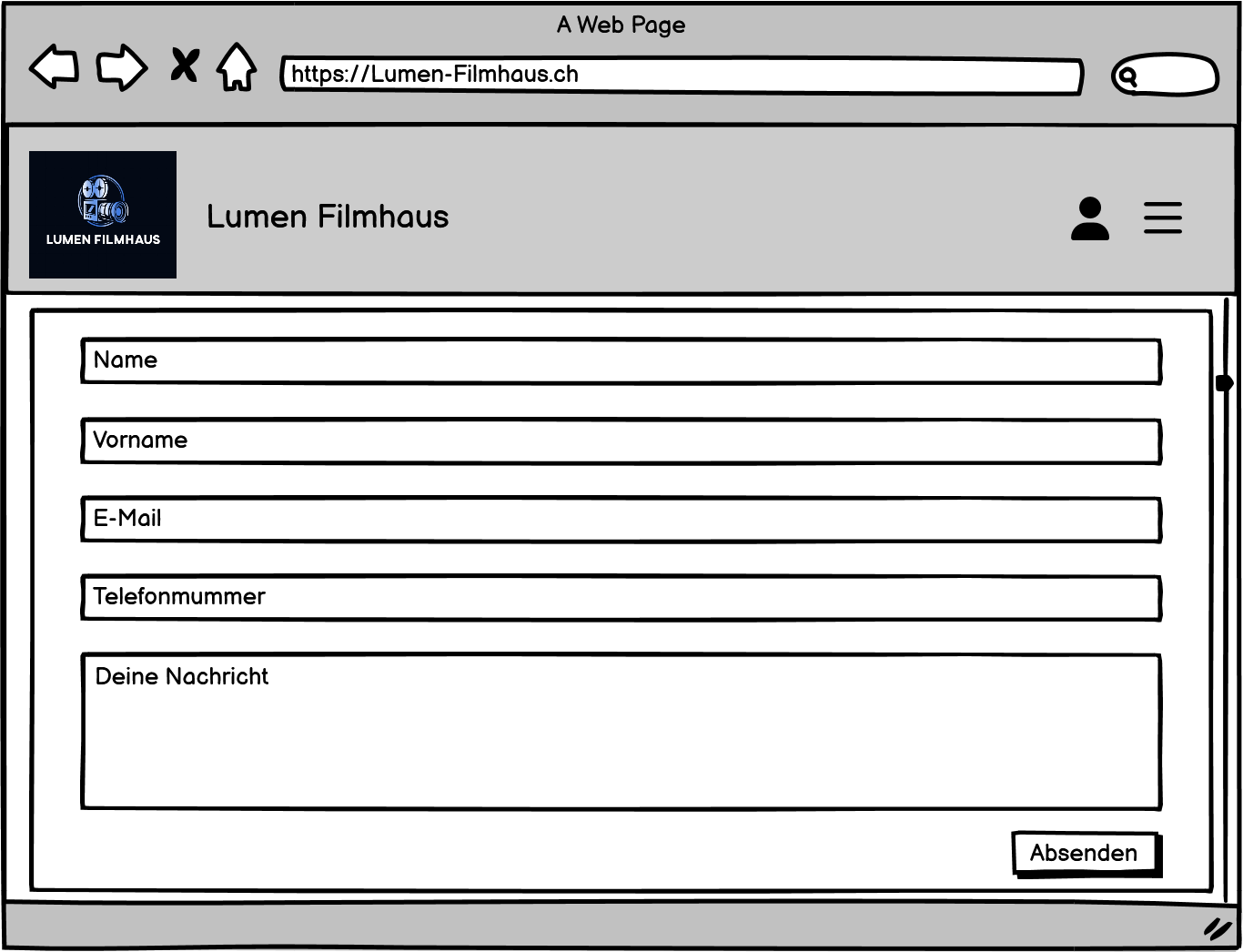


Abbildung 8: Wireframe Kontakt

### Reservation

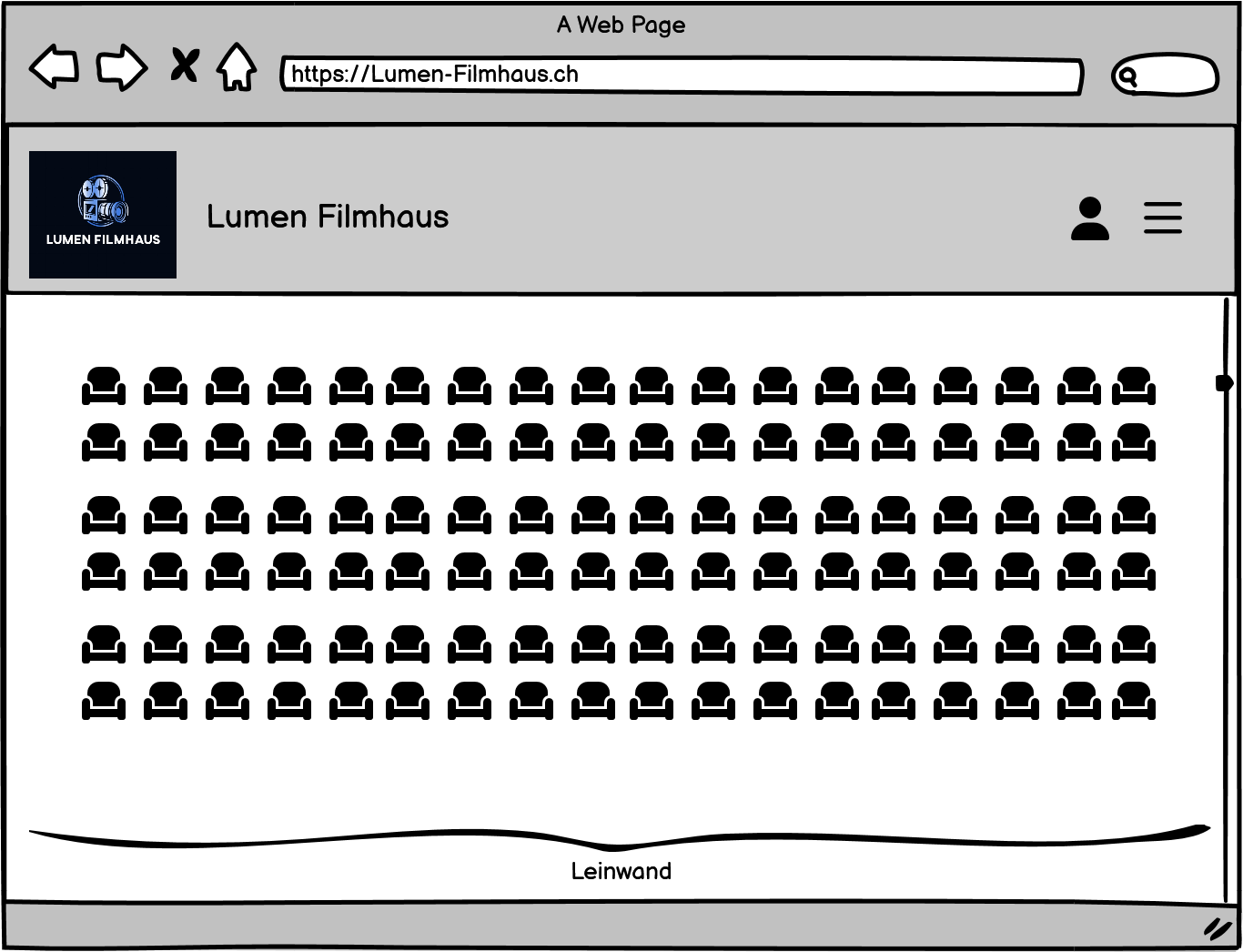


Abbildung 9: Wireframe Reservation

## Datenmodell

Zur Planung der technischen Umsetzung wurde ein einfaches, aber funktionales Datenmodell entworfen, das die Kernelemente des Projekts abbildet: Filme, Vorstellungen und Reservationen. Ziel war es, eine saubere Datenstruktur zu definieren, die alle nötigen Informationen für eine Sitzplatzreservation bereitstellt, dabei jedoch schlank und verständlich bleibt.

### Grundidee

Die Informationen zu den Filmen – wie Titel, Beschreibung, Bild und Trailer – werden fest im Quellcode definiert und nicht in der Datenbank gespeichert. In der Datenbank werden nur die Filmreferenz (z. B. Film-ID) sowie die zugehörigen Vorstellungen und Reservationen gespeichert.

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

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 10: Datenmodell

### Zusammenfassung

Das Datenmodell unterstützt alle wesentlichen Abläufe des Projekts: Es erlaubt die Zuordnung von Vorstellungen zu Filmen und die Erfassung von Sitzplatzreservationen. Die gewählte Struktur ist leicht verständlich, einfach zu implementieren und bietet zugleich eine solide Grundlage für eine spätere Erweiterung – beispielsweise durch eine detaillierte Sitzplatzverwaltung oder Benutzerkonten.

## Meilenstein Planung

Durch die detaillierte Planung mit klaren Arbeitspaketen, Ressourcenübersicht, Strukturierung des Projekts und einem sauberen Datenmodell ist die Grundlage für die technische Umsetzung geschaffen.

Meilenstein: Abschluss der Planungsphase am [Geplant 19. Mai 2025, Ist 17.Mai 2025]

Nächster Schritt: Beginn der Realisierungsphase (Realisieren)

# Entscheiden

Im Schritt Entscheiden werden basierend auf den gesammelten Informationen und der Planung konkrete technische, methodische und organisatorische Entscheidungen getroffen. Diese legen die Rahmenbedingungen für die Umsetzung fest und sichern einen zielgerichteten und effizienten Projektverlauf.

## Technologie-Entscheidungen

* Das Frontend wird mit HTML, CSS und JavaScript realisiert, um eine flexible und leicht erweiterbare Benutzeroberfläche zu schaffen.
* Das Backend wird als ASP.NET Core Web API in C# entwickelt, um stabile und performante Schnittstellen für die Datenbereitstellung und -verarbeitung zu bieten.
* Die Architektur sieht eine Trennung von Frontend und Backend vor, wobei das Frontend über REST-API mit dem Backend kommuniziert.
* Für die erste Umsetzung wird die Datenhaltung mit lokalen JSON-Dateien realisiert; eine spätere Migration zu einer Datenbank ist möglich.

## Methodische Entscheidungen

* Das Projekt wird nach dem IPERKA-Modell strukturiert durchgeführt, um einen klaren Ablauf mit definierten Phasen zu gewährleisten.
* Die Planung erfolgt mit detaillierten Arbeitspaketen, die Prioritäten und Zeitrahmen enthalten.
* Die Qualitätssicherung erfolgt durch manuelle Tests, API-Tests mit Postman und regelmässige Review-Runden.
* Die Versionskontrolle erfolgt lokal; ein Einsatz von Git ist bei Bedarf vorgesehen.

## Organisatorische Entscheidungen

* Die Entwicklung erfolgt mit Visual Studio Code als Hauptarbeitsumgebung.
* Für die Projektplanung und Aufgabenverwaltung wird clickup.com eingesetzt.
* Der Projektzeitraum ist vom 15. Mai bis 15. Juni 2025 festgelegt, mit Pufferzeiten für Tests und Nachbesserungen.
* Die Kommunikation und Dokumentation erfolgen über Word (für die schriftliche Arbeit) und PowerPoint (für die Präsentation).

## Risiko-Management

* Risiken wie Zeitmangel, technische Schwierigkeiten und unklare Anforderungen werden durch frühzeitige Planung, regelmässige Kontrolle und Testphasen minimiert.
* Ein Fokus auf ein Minimal Viable Product (MVP) stellt sicher, dass die Kernfunktionen termingerecht umgesetzt werden können.

# Realisieren

## Backend

In dieser Phase haben wir das Backend für den Webauftritt des Lumen-Filmhauses vollständig umgesetzt. Folgendes wurde dabei realisiert:

### Entwicklungs­umgebung und Tools

Als Entwicklungsumgebung dient Visual Studio Code mit installiertem .NET7-SDK. Die Datenbank läuft lokal in MariaDB und wird über HeidiSQL verwaltet. Für die Datenmodellierung und Migrationen kommt das Tool dotnet-ef zum Einsatz, dass wir via Tool-Manifest in der Projekt­mappe eingebunden haben. API-Aufrufe testen wir in VS-Code direkt über eine .http-Datei.

### Projektstruktur

Das Projekt gliedert sich in vier Hauptordner:

* **Models/** enthält die C#-Klassen Film, Datum und Reservation mit ihren Eigenschaften und Daten­annotationen.
* **Data/** enthält AppDbContext.cs, unseren Entity-Framework-Kontext.
* **Controllers/** enthält die drei API-Controller (FilmeController, VorstellungenController, ReservationController), die alle Endpunkte bereitstellen.
* **Program.cs** steuert die Service-Registrierung (Datenbank, CORS, Swagger) und konfiguriert die Middleware.

### Datenbank und Migrationen

Über appsettings.json ist die Verbindung zu MariaDB definiert. Der AppDbContext richtet drei Tabellen ein: Filme, Datum (Vorstellungen) und Reservation. Migrationen werden mit

```  
dotnet tool run dotnet-ef migrations add InitialCreate

dotnet tool run dotnet-ef database update

```

erzeugt und angewandt. In einem Seed-Block in Program.cs prüfen wir beim Start, ob die Tabellen leer sind, und fügen dann unsere zehn Beispiel-Filme sowie fünfzehn Vorstellungstermine automatisch ein.

### Geschäfts­logik und Validierung

In der ReservationController. PostReservation-Methode werden zuerst Name, E-Mail und Sitzplätze auf Vorhandensein geprüft. Anschliessend ermitteln wir alle bereits belegten Plätze derselben Vorstellung und vergleichen sie mit den angefragten Plätzen. Sollte sich dort eine Überschneidung ergeben, liefert die API einen HTTP 409 Conflict zurück. Nur wenn alle Plätze frei sind, wird die neue Reservation in der Datenbank gespeichert; die int-ID erhält die Datenbank automatisch per Auto-Increment.

### API-Controller im Überblick

* **FilmeController** bietet GET /api/filme und GET /api/filme/{id}, um Filmlisten und Einzeldaten zu liefern.
* **VorstellungenController** stellt mit GET /api/vorstellungen/{filmId} die Termine für einen Film bereit.
* **ReservationController** unterstützt:
  + GET /api/reservation/{id} (einzelne Buchung),
  + GET /api/reservation/belegte/{vorstellungId} (Liste aller belegten Plätze),
  + POST /api/reservation (neue Buchung mit Duplikat-Check).  
    Weitere Methoden für PUT und DELETE können bei Bedarf ergänzt werden.

### Swagger-Integration

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

KI-generierte Inhalte können fehlerhaft sein.Mit dem NuGet-Paket Swashbuckle.AspNetCore sind unsere Endpunkte automatisch dokumentiert. Beim Start im Entwicklungs­modus stehen die Swagger-UI und die swagger.json unter http://localhost:5029/swagger bzw. /swagger/v1/swagger.json zur Verfügung. XML-Kommentardateien aus den Controller- und Model-Klassen können in Swagger eingebunden werden, um jeden Endpunkt und jedes DTO ausführlich zu beschreiben.

Abbildung 11: Swagger Dokumentation

### CORS und Middleware

Damit unser Frontend (z. B. unter http://localhost:5500) API-Aufrufe durchführen darf, haben wir in Program.cs eine CORS-Policy aktiviert, die alle Methoden und Header von definierten Ursprüngen zulässt. Ein globaler Exception-Handler fängt unerwartete Fehler ab und verhindert unformatierte Stacktraces im Browser.

### Test und Qualitätssicherung

Alle Endpunkte lassen sich direkt in der .http-Datei in VS-Code oder in Swagger ausprobieren. Neben manuellen Tests empfehlen sich spätere Integrationstests mit WebApplicationFactory<T>, um CRUD-Szenarien und Fehlerschleifen automatisch zu prüfen.

Mit dieser Umsetzung ist das Backend vollständig einsatzbereit und liefert alle Daten für das Frontend zuverlässig und sicher.

## Frontend

Das Frontend des Webauftritts für das Lumen Filmhaus wurde vollständig mit HTML, CSS und JavaScript in Visual Studio Code umgesetzt. Es dient als Benutzeroberfläche zur Darstellung des Filmprogramms, zur Sitzplatzwahl und zur Kommunikation mit der Backend-API. Ziel war eine responsive, benutzerfreundliche Oberfläche, die sowohl auf Desktop- als auch auf Mobilgeräten intuitiv bedienbar ist.

### Entwicklungsumgebung und Aufbau

Die Frontendstruktur befindet sich im Ordner lumen-frontend und ist wie folgt gegliedert:

* **HTML/** – enthält alle statischen HTML-Dateien wie index.html, programm.html, reservation.html, kontakt.html sowie die Film-Detailseiten.
* **CSS/** – enthält die zentrale Datei style.css, in der sämtliche Layout-, Farb- und Responsivitätsregeln definiert sind.
* **JS/** – enthält die JavaScript-Logik, insbesondere:
  + api.js (API-Kommunikation),
  + programm.js (Daten vom Backend laden),
  + reservation.js (Sitzplatzlogik),
  + reservation-form.js (Formularvalidierung und POST-Logik).
* **IMG/** – enthält alle verwendeten Bilder (Logos, Filmcover, …).
* **VIDEO/** – beinhaltet die Filmtrailer im MP4-Format.

### Seitenstruktur und Funktion

Das Frontend umfasst folgende Kernseiten:

**index.html** – Startseite mit beliebten Filmen und CTA-Buttons zum Programm und Detailseiten der bestimmten Filme.

**programm.html** – Herzstück der Seite. Hier wird das aktuelle Kinoprogramm aus der Datenbank geladen und in übersichtlichen Filmkarten pro Tag gruppiert angezeigt. Die Daten stammen via GET /api/filme und /api/vorstellungen/{filmId} vom Backend.

**reservation.html** – Sitzplatzwahl mit gridbasiertem Saalplan. Bereits belegte Sitze werden beim Laden per API abgerufen und visuell gesperrt. Der Nutzer kann bis zu 8 Plätze auswählen. Eine automatische Sperre blockiert mehr Auswahl und zeigt einen Hinweistext auf Gruppenbuchungen.

**reservation-form.html** – Formularseite zur Eingabe von Name und E-Mail, nachdem die Plätze ausgewählt wurden. Die Daten werden per POST an das Backend übermittelt. Ein Notification-System (Toast-artig) gibt visuelles Feedback bei Erfolg oder Fehler (z. B. Konflikt).

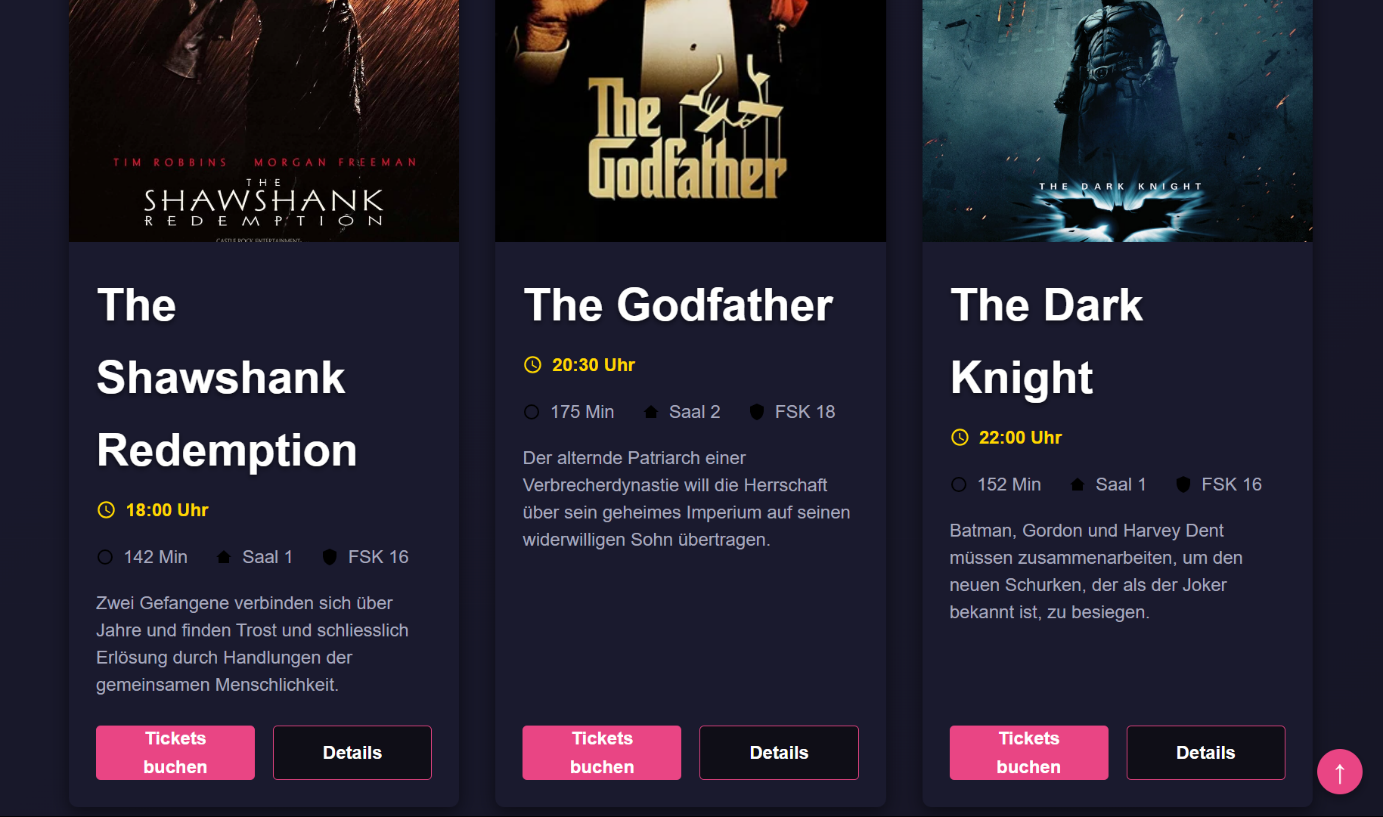
**Film-Detailseiten** (HTML/Film-Details/\*.html) – Für jeden Film gibt es eine eigene Unterseite mit:

* Titel, Genre, Altersfreigabe, Beschreibung
* Filmtrailer (MP4) mit Play-/Pause-Button
* Vorstellungen als Buttons mit direkter Verlinkung zur Sitzplatzwahl

#### Vorstellung

##### Home/Index

##### Programm



##### Filmdetails

##### Reservation / Reservationsformular



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

KI-generierte Inhalte können fehlerhaft sein.

##### Kontakt

### Abgrenzung

Das Projekt konzentriert sich ausschliesslich auf die Kernfunktionalitäten eines Kinobuchungssystems für Kunden: Darstellung des Programms, Sitzplatzwahl und Reservierung. Aus Gründen der Fokussierung und Umsetzbarkeit wurden bestimmte Funktionen bewusst nicht realisiert, wie zum Beispiel:

- Benutzerkonten bzw. Login-System

- Adminoberfläche zur Verwaltung von Filmen und Vorstellungen

- Online-Zahlung oder Reservierungsverbindlichkeit

Diese Funktionen könnten später modular ergänzt werden, wurden aber im Rahmen des SOL2-Projekts ausgeklammert.

### Responsive Design

Das Design wurde vollständig responsiv gestaltet. Dazu wurden Media Queries für Viewports bis 768 px und 480 px erstellt. Besonders wichtig war:

* Filmkarten stapeln sich bei kleiner Auflösung vertikal.
* Navigation wird bei kleineren Viewports durch ein Burger-Menü ersetzt.
* Hero-Sektionen wie Filmtrailer passen sich der Breite des Bildschirms an und werden nicht beschnitten.
* Buttons und Texte skalieren responsiv mit.

### JavaScript-Funktionen im Detail

**programm.js**: Lädt Filme und Vorstellungstermine via API, gruppiert diese nach Datum, rendert Filmkarten dynamisch.

**reservation.js**: Lädt belegte Plätze, verwaltet ausgewählte Sitze, prüft Limits, erlaubt visuelle Interaktion mit Platz-Grid.

**reservation-form.js**:

* Validiert das Format von Name (Vorname Nachname) und E-Mail.
* Zeigt Benachrichtigungen (success, error, warning) mit Timeout.
* Führt die finale API-Reservierung durch (POST /api/reservation).

**script.js**: Enthält das Menüverhalten (z. B. Burger-Icon und Overlay) und globale UI-Interaktionen wie den „Zurück nach oben“-Button.

### Benutzerfreundlichkeit und Barrierefreiheit

Ein besonderes Augenmerk lag auf klarer Typografie, ausreichendem Kontrast, Tooltip-Hinweisen, Tastaturzugänglichkeit der Formulare und klarer Fehleranzeige. Tooltips an Sitzen und Buttons geben direkte Orientierung. Fehlermeldungen sind farblich und semantisch gekennzeichnet.

### Einbettung der API

Die Basis-URL für die API ist zentral als API\_HOST definiert und kann bei Deployment leicht angepasst werden (z. B. zu einem gehosteten Backend). Die Kommunikation erfolgt ausschliesslich über Fetch mit JSON.

### Farbkonzept und Designphilosophie

Das Farbkonzept des Lumen Filmhaus Frontends wurde nicht zufällig gewählt, sondern bewusst auf die besondere Atmosphäre eines Kinos abgestimmt. Ziel war es, ein digitales Erlebnis zu schaffen, das bereits auf der Website die Stimmung eines Kinobesuchs vermittelt – dunkel, fokussiert, mit akzentuierten Lichtpunkten.

#### Farbwahl im Detail

Das zentrale Farbschema wurde über CSS-Variablen im :root-Block definiert und in der gesamten Applikation verwendet:

|  |  |  |
| --- | --- | --- |
| Variable | Farbwert (HEX) | Verwendung |
| color-bg-dark | #0f0e17 | Tiefes Nachtblau – Kinosaal-Feeling |
| color-bg-light | #1a1a2e | Etwas aufgehellt für Sektionen |
| color-primary | #e94584 | Magenta – Lichtkegel, Logos, Buttons |
| color-primary-light | #ff6b9e | Hover-Farbe für Buttons |
| color-secondary | #ffd700 | Gold – Highlights, z.B. Vorstellungszeit |
| color-accent | #25aae1 | Blau – modern, technisch, für CTAs |
| color-text-light | #fffffe | Reinweiss – gute Lesbarkeit |
| color-text-muted | #a7a9be | Sekundärtexte, Legenden |
| color-border | #2e2e48 | Ränder |
| color-shadow | RGBA(0, 0, 0, 0.5) | Schatten  RGBA (0, 0, 0, 0.5) = schwarz mit 50% Deckkraft |

Tabelle 9: Farbtheorie

#### Designphilosophie

**Kinogefühl**: Dunkle Hintergrundfarben erzeugen ein immersives, kinoähnliches Ambiente.

**Markenidentität**: Das kräftige **Magenta** zieht sich konsistent durch Buttons, Highlights und Header – und sorgt für hohen Wiedererkennungswert.

**Exklusivität**: Die Farbe **Gold** wird gezielt für besondere Elemente verwendet, z. B. Vorstellungszeit, Auszeichnungen oder Hervorhebungen.

**Technischer Touch**: Das **Akzentblau** dient als Kontrast für Call-to-Actions wie „Jetzt buchen“ oder Menü-Icons.

Dieses Farbsystem ist visuell harmonisch, kontrastreich und modern – perfekt für eine Kino-Webseite.

### Funktions- und UI-Tests

Das Frontend wurde iterativ getestet:

**Responsives Verhalten**: Regelmässig in der Chrome-DevTools-Geräteansicht (z. B. iPhone 13, Samsung Note) und mit responsivetesttool.com kontrolliert. Besonderer Fokus lag auf dem Filmprogramm, Trailer-Videogrösse und Menüverhalten (Burger-Menü).

**Interaktive Komponenten**: Alle Buttons, Sitzplatzinteraktionen und Formulareingaben wurden manuell auf Desktop und mobilen Geräten geprüft.

**Fehlermeldungen**: Szenarien mit Konflikten (z. B. doppelt belegte Sitze) wurden gezielt getestet, um die Error-Meldungen in realistischen Abläufen zu prüfen.

**Notification-System (uivers.io)**

Für Benutzerfeedback wurde ein eigenes Notification-System inspiriert von uivers.io entwickelt. Dieses zeigt Hinweise in der unteren Ecke des Bildschirms an und basiert auf einem flexiblen showNotification()-JavaScript-Modul. Es unterstützt folgende Typen:

* success – z. B. bei erfolgreicher Buchung
* warning – z. B. bei 8+ Sitzplätzen (Gruppenreservierung)
* error – z. B. bei bereits belegten Plätzen
* info – z. B. Hinweise bei Formularfeldern

Die visuelle Gestaltung basiert auf:

* leichtem Schatten
* subtiler Hintergrundstruktur (Gitterraster)
* farbcodierten Icons
* animierter Progress-Bar (Timeout-Ablauf)

Dadurch erhalten Nutzer unmittelbares Feedback, ohne die Seite neu zu laden oder Popups nutzen zu müssen.

## Meilenstein Entwicklung

Der Meilenstein „Entwicklung“ markierte den zentralen Abschnitt der Projektarbeit. In dieser Phase wurden sowohl das Backend als auch das Frontend funktional umgesetzt. Ziel war es, die technische Basis für eine interaktive, responsive Kinowebseite mit Reservierungssystem bereitzustellen. Die Entwicklung erfolgte in mehreren Etappen, wobei jedes Teilsystem separat getestet und integriert wurde.

### Backend-Entwicklung

Im Backend wurde mit ASP.NET Core Web API gearbeitet. Die Datenhaltung erfolgte in einer lokal laufenden MariaDB-Instanz, mit Anbindung über Entity Framework Core. Zu den wesentlichen Komponenten zählten:

* Datenmodellierung (Filme, Vorstellungen, Reservationen)
* REST-Schnittstellen für die CRUD-Operationen
* Validierung und Geschäftslogik (z. B. Verhinderung von Doppelbuchungen)
* Automatisiertes Seeding von Testdaten
* Konfiguration von CORS, Swagger und Middleware

### Frontend-Entwicklung

Das Frontend wurde parallel in HTML, CSS und JavaScript entwickelt. Ziel war ein intuitives Nutzererlebnis mit modernen UI-Komponenten. Zu den Highlights zählen:

* Dynamisches Laden des Filmprogramms über die API
* Visuelle Sitzplatzwahl mit Interaktion und Limitprüfung
* Formularseite mit Validierung und direkter Rückmeldung (Notification-System)
* Film-Detailseiten mit Trailer, Beschreibung und Direktverlinkung
* Responsive Design mit Breakpoints für verschiedene Endgeräte

### Iteratives Vorgehen

Die Entwicklung erfolgte iterativ: Nach jedem funktionalen Abschnitt wurden Tests durchgeführt und Feedback eingearbeitet. So konnte sichergestellt werden, dass sowohl die technische Funktionalität als auch die Benutzerfreundlichkeit den Erwartungen entsprechen.