Prototyping Projektdokumentation

Name: Kai Bleuel

E-Mail: bleuekai@students.zhaw.ch

URL der deployten Anwendung: https://game-library-3us7.vercel.app/

# Einleitung

Die Game Library ist eine Web-Applikation zur Verwaltung von Videospielen und Spieleplattformen.  
Benutzer können Spiele mit Attributen wie Name, Genre, Plattform und "gespielt"-Status hinzufügen, bearbeiten und löschen. Zusätzlich können Plattformen verwaltet und mit Spielen verknüpft werden. Die Daten werden in einer MongoDB-Datenbank gespeichert und dynamisch auf der Webseite dargestellt. Das Design wurde mit Bootstrap angepasst und ist responsive. Die Anwendung wurde mit SvelteKit und Vercel deployed.

# Datenmodell

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

KI-generierte Inhalte können fehlerhaft sein.

# Beschreibung der Anwendung

## Startseite

Route: /

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

KI-generierte Inhalte können fehlerhaft sein.

Die Startseite zeigt die Navigationsleiste der Anwendung und dient als Einstiegspunkt.  
Über die Navigationsbar kann der Benutzer auf die Seiten "Games" und "Platforms" zugreifen.  
Der Titel der Anwendung ist "GameStack". Es werden auf der Startseite selbst keine Daten angezeigtLediglich ein Logo und ein Informationstext, der erklärt, wofür diese Webseite gedacht ist. Weiter hat es buttons names View Games und View Platforms um wie beim Navigationsbar weiter zu gelangen.

Layout und globale Styles:

Dateien:

* src/routes/+layout.svelte (für Navbar und Layout)
* src/styles.css (globale Styles)

## Games-Liste

Route: /games

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

KI-generierte Inhalte können fehlerhaft sein.**

Auf der /games-Seite werden alle gespeicherten Spiele in einer Tabelle angezeigt. Zu jedem Spiel werden Name, Genre, Plattform und der "Played"-Status angezeigt. Über Dropdown-Menüs können die angezeigten Spiele nach Genre, Plattform und "Played"-Status gefiltert werden. Die Titel der Tabelle können angeklickt werden, um zum Beispiel nach Rating zu sortieren. Jedes Spiel kann angeklickt werden, um mehr Informationen darüber zu erhalten (siehe unten /games/[id]) Zusätzlich gibt es einen Button, um ein neues Spiel hinzuzufügen.  
  
Dateien:

* src/routes/games/+page.svelte
* src/routes/games/+page.server.js
* src/lib/db.js (für die Datenbank-Abfragen)

## Neues Game anlegen

Route: /games/new

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

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite kann ein neues Spiel erstellt werden. Es wird ein Formular angezeigt, in dem Name, Genre, Plattform und "Played"-Status ausgewählt werden können. Zu erwähnen ist, dass Genre und Platforms Dropdown Menüs sind. Platforms wird aus der Datenbank gezogen und enthält immer die aktuellen Plattformen (siehe unten /platform/new). Weiter können Kommentare hinterlegt und ein Image URL hinterlegt werden. Beim Speichern wird das Spiel in der MongoDB-Datenbank abgelegt und erscheint anschliessend in der Games-Liste.

Dateien:

* src/routes/games/new/+page.svelte
* src/routes/games/new/+page.server.js
* src/lib/db.js

## Game Detailansicht

Route: /games/[id]

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

KI-generierte Inhalte können fehlerhaft sein.

Diese Seite zeigt die Detailansicht eines einzelnen Spiels an. Alle Informationen zum Spiel (Name, Genre, Plattform, Played-Status) werden übersichtlich dargestellt. Von hier aus kann das Spiel bearbeitet oder gelöscht werden. Oder man kann Back to Games anklicken.

Dateien:

* src/routes/games/[id]/+page.svelte
* src/routes/games/[id]/+page.server.js
* src/lib/db.js

## Game bearbeiten

Route: /games/[id]/edit

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

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite kann ein bestehendes Spiel bearbeitet werden. Die Felder sind vorbefüllt mit den vorhandenen Informationen und können geändert werden. Nach dem Speichern wird das Spiel in der Datenbank aktualisiert.

Dateien:

* src/routes/games/[id]/edit/+page.svelte
* src/routes/games/[id]/edit/+page.server.js
* src/lib/db.js

## Plattformen-Liste

Route: /platforms

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

KI-generierte Inhalte können fehlerhaft sein.

Auf der /platforms-Seite werden alle verfügbaren Spieleplattformen aufgelistet. Jede Plattform wird in einer Tabelle angezeigt. Plattformen können gelöscht werden, wenn sie nicht mehr benötigt werden. Es hat einen Button um neue Plattformen hinzuzufügen.

Dateien:

* src/routes/platforms/+page.svelte
* src/routes/platforms/+page.server.js
* src/lib/db.js

## Neue Plattform anlegen

Route: /platforms/new

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

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite kann eine neue Plattform hinzugefügt werden. Über ein einfaches Formular kann der Name der Plattform sowie wenn gewünscht ein Manufacturer eingetragen werden. Nach dem Speichern erscheint die neue Plattform in der Plattformen-Liste und steht zur Auswahl beim Erstellen neuer Spiele zur Verfügung.

Dateien:

* src/routes/platforms/new/+page.svelte
* src/routes/platforms/new/+page.server.js
* src/lib/db.js

# Erweiterungen

## Plattform-Verwaltung

Auf der Seite /platforms können Plattformen verwaltet werden (Erstellen, Anzeigen, Löschen). Dies wurde mit eigenen Routen und einer MongoDB-Anbindung realisiert.

Dateien:

* src/routes/platforms/+page.svelte
* src/routes/platforms/+page.server.js
* src/routes/platforms/new/+page.svelte
* src/routes/platforms/new/+page.server.js

## Bearbeiten von Games

Bestehende Games können auf der Seite /games/[id]/edit bearbeitet werden. Die Daten werden geladen und in einem Formular dargestellt. Nach dem Speichern wird das Game aktualisiert.

Dateien:

* src/routes/games/[id]/edit/+page.svelte
* src/routes/games/[id]/edit/+page.server.js

## Filterfunktion für Games

Auf der Seite /games können die Games nach Genre, Plattform und Played-Status gefiltert werden. Die Filter wurden mittels Formularfeldern und Bindings in Svelte umgesetzt.

Datei:

* src/routes/games/+page.svelte

## Suchfunktion

Auf der Seite /games können Games zusätzlich über ein Suchfeld nach Titel gefiltert werden. Dies erfolgt clientseitig per Bindings in Svelte.

Datei:

* src/routes/games/+page.svelte

## Sortieren

Auf der Seite /games können die Games alphabetisch oder numerisch sortiert werden. Dazu wurde eine Sortier-Logik clientseitig in der Svelte-Komponente implementiert.  
Datei:

* src/routes/games/+page.svelte

## Bestätigungsdialog beim Löschen

Beim Löschen von Games erscheint ein Bestätigungsdialog, der den Löschvorgang absichert (confirm()).  
Datei:

* src/routes/games/+page.svelte

## Individuelles Styling (Dark Theme)

Die Anwendung nutzt ein individuelles dunkles Farbschema (Dark Theme), das über eine eigene CSS-Datei definiert wird.

Datei:

* src/styles.css

## Kombination von Bootstrap und eigenem CSS

Es wurde Bootstrap eingebunden und gezielt mit eigenem CSS kombiniert, um ein individuelles Layout und Farbschema zu realisieren.

Datei(en):

* src/app.html
* src/styles.css

## Anzeigen von Bildern über URL (image URL)

Beim Erstellen oder Bearbeiten eines Games kann im Feld "Image URL" eine beliebige Bild-URL (z.B. aus dem Internet) angegeben werden. Dadurch können Spiele-Cover oder andere passende Bilder flexibel eingebunden werden, ohne die Bilder selbst hochzuladen.

Das Bild wird auf der Detailseite eines Games (/games/[id]) unterhalb der Spieldetails angezeigt.

Das Feld "Image URL" ist eine Erweiterung des Standarddatenmodells und wird in der MongoDB-Datenbank mitgespeichert.

Datei(en):

* src/routes/games/new/+page.svelte (Eingabefeld im Formular "Neues Game")
* src/routes/games/[id]/edit/+page.svelte (Bearbeitungsformular)
* src/routes/games/[id]/+page.svelte (Anzeige des Bildes in der Detailansicht)
* src/lib/db.js (MongoDB-Schema / Felddefinition)