Prototyping Projektdokumentation

Name: Anastasija Gojkovic

E-Mail: gojkoana@students.zhaw.ch

URL der deployten Anwendung: https://vaultofgames.netlify.app/

# Einleitung

GameVault ist eine webbasierte Anwendung zur Verwaltung und Darstellung einer umfangreichen Spielesammlung. Die Idee hinter der App ist es, eine übersichtliche und benutzerfreundliche Plattform zu bieten, auf der Informationen zu vielen Videospielen abgerufen und verwaltet werden können. Die Anwendung richtet sich an Nutzer, die sich einen Überblick über verfügbare Spiele verschaffen wollen.

Zu den Grundfunktionen gehören:

* Eine Übersicht aller Spiele mit Bild, Genre, Plattform und Preis
* Eine Detailansicht pro Spiel mit zusätzlichen Informationen
* CRUD-Funktionen für Spiele
* Eine eigene Entwickler-Übersicht, inklusive Detailansicht und Verknüpfung mit den zugehörigen Spielen
* Pagination zur strukturierten Navigation durch grosse Datenmengen

# DatenmodellEin Bild, das Entwurf, Diagramm, Zeichnung, Lineart enthält. KI-generierte Inhalte können fehlerhaft sein.

# Beschreibung der Anwendung

## Startseite und Navigation

Route: /

Ein Bild, das Text, Screenshot, Reklametafel, Himmel enthält.

KI-generierte Inhalte können fehlerhaft sein.

Die Startseite der Anwendung zeigt einen Willkommenstext und ein Hintergrundbild. Sie dient als visuelles Intro zur Anwendung. Sie enthält keine interaktiven Funktionen, sondern leitet Benutzer weiter zu den Bereichen Spiele oder Entwickler über die Navigation. Die Navigation ist von jeder Seite aus ersichtlich. Sie ist auch ausgelagert als Komponente Navbar.svelte und wird im +layout.svelte eingebunden. Die Navigations-Leiste enthält Links zur Startseite (GameVault oder ‘Jetzt Endecken’, «/»), Spieleübersicht (Spiele, «/games»), Entwicklerübersicht (Entwickler, «/developers»).

Dateien:

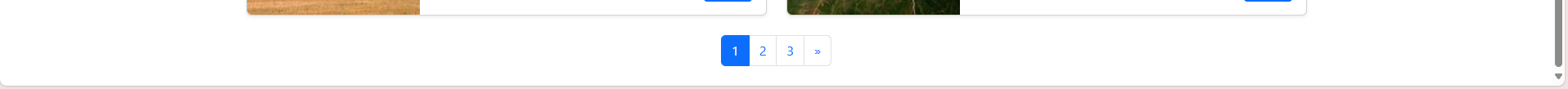
* lib/components/Navbar.svelte
* static/images/fallout\_wallpaper.jpg
* routes/+layout.svelte
* routes/+page.svelte

## Gameübersicht

Route: /games

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

KI-generierte Inhalte können fehlerhaft sein.



Auf dieser Seite werden alle Spiele in einer Kartenbasierten Übersicht dargestellt. Die Darstellung erfolgt über die Komponente GameCard.svelte. Es werden jeweils 20 Spiele pro Seite angezeigt. Mithilfe der eingebauten Pagination kann durch verschiedene Seiten geblättert werden. Die Pagination ist als Komponente Pagination.svelte ausgelagert. Jede Karte enthält ein Bild (Platzhalterbild von <https://picsum.photos>), den Namen, Genre, Plattform, Preis sowie die durchschnittliche Spielzeit. Über den Button «Details» gelangt man auf die Detailansicht eines Spiels unter /games/[appid].   
Oben rechts befindet sich ein Button «Spiel hinzufügen», der zum Erstellungsformular unter /games/create weiterleitet.

Dateien:

* lib/components/GameCard.svelte
* lib/components/Pagination.svelte
* lib/server/games.js
* routes/games/+page.svelte
* routes/games/+page.server.js

## Spiel hinzufügen

Route: /games/create

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

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite kann ein neues Spiel erstellt werden. Das Formular ist vollständig clientseitig validiert und umfasst alle relevanten Felder eines Spiels. Die Eingabefelder für Plattformen und Genres sind als kommagetrennte Texteingabe gelöst, welche serverseitig in Arrays umgewandelt werden. Pflichtfelder sind App ID, Name, Entwickler ID und Preis. Nach erfolgreicher Speicherung wird der Benutzer zurück auf die Spieleübersicht geleitet.

Dateien:

* routes/games/create/+page.svelte
* routes/games/create/+page.server.js
* lib/server/games.js

## Game Detailansicht

Route: /games/[appid]

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

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Säugetier, Screenshot, Wolf enthält.

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite werden alle wichtigen Informationen zu einem einzelnen Spiel angezeigt. Die Detailseite wird dynamisch anhand der appid des Spiels generiert. Falls kein Spiel zur übergebenen appid gefunden wird, erscheint eine Fehlermeldung. Angezeigt werden Name, Genres, Plattformen, Preis, Durchschnittliche Spielzeit, Entwicklernamen (wird per developer\_id aus der Developer-Collection geladen), Veröffentlichungsdatum, Mindestalter und das Bild (Platzhalter). Der Entwicklername ist ein Link der zur Entwickler Detailansicht auf /games/developer/[developer\_id] führt. Dazu sind auch 3 Buttons auf der Detailansicht. Bearbeiten führt zur Edit Seite (/games/[appid]/edit), Löschen löscht das Spiel und leitet zur Übersicht um und ‘Zurück zur Übersicht’ leitet wieder zur Spieleübersicht. Wenn ein Spiel nicht gefunden wird, wird der Fehlerfall abgefangen und es zeigt die 404 Meldung an «Spiel nicht gefunden». Wird der «Spiel Löschen» Button gedrückt, erscheint eine Warnmeldung (JavaScript Confirm). Nach Bestätigung wird das Spiel über ein POST-Request gelöscht und der Benutzer wird zurück auf die Übersicht /games geleitet.

Dateien:

* Routes/games/[appid]/+page.svelte
* Routes/games/[appid]/+page.server.js
* Lib/server/games.js

## Game bearbeiten

Route: /games/[appid]/edit

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

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite kann ein vorhandenes Spiel anhand seiner appid bearbeitet werden. Die Eingabefelder sind vorausgefüllt mit den aktuellen Daten des Spiels. Nach dem Speichern wird das Spiel aktualisiert und die Anwendung leitet zur Detailansicht des Spiels zurück. Alle Felder ausser appid sind bearbeitbar und Name, Entwickler ID und Preis müssen einen Wert haben.

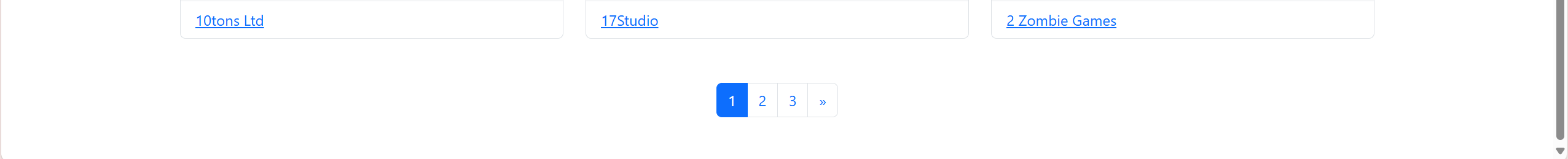
Dateien:

* Routes/games/[appid]/edit/+page.svelte
* Routes/games/[appid]/edit/+page.server.js
* Lib/server/games.js

## Entwickler Übersicht

Route: /developer

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

KI-generierte Inhalte können fehlerhaft sein. 

Auf dieser Seite werden alle Entwickler alphabetisch sortiert und gleichmässig in drei Spalten dargestellt. Es werden 60 Entwickler pro Seite angezeigt. Die Darstellung erfolgt als einfache Link-Liste, die einzelnen List Items werden mit dem ausgelagerten DeveloperListItem.svelte dargestellt. Die Pagination ist durch die ausgelagerte Komponente Pagination.svelte realisiert. Jeder Entwickler ist als Link dargestellt, der zur Detailseite unter /developers/[developer\_id] führt.

Dateien:

* Lib/server/developer.js
* Lib/components/Pagination.svelte
* Routes/developers/+page.svelte
* Routes/developers/+page.server.js

## Entwickler Details

Route: /developers/[developer\_id]

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

KI-generierte Inhalte können fehlerhaft sein.

Diese Seite zeigt Detailinformationen zu einem Entwickler basierend auf dessen developer\_id an. Zusätzlich werden alle zugehörigen Spiele angezeigt, also alle Games, deren developer\_id mit dem aktuellen Entwickler übereinstimmt.

# Erweiterungen

## Pagination

Auf der Seite zur Spiel- und Entwicklerübersicht wurde eine eigene Pagination-Funktionalität implementiert. Pro Seite werden nur bestimmte Anzahl an Datensätzen angezeigt. Die Aktuelle Seitenzahl wird über die URL übergeben. Die Navigation erfolgt über eine wiederverwendbare Komponente Pagination.svelte, welche auf beiden Übersichtsseiten eingebunden wird. Es werden jeweils die angrenzenden Seiten sowie 10 Vor/Zurück Buttons angezeigt.

Ein Bild, das Screenshot, Text enthält.

KI-generierte Inhalte können fehlerhaft sein.

Dateien:

* Lib/components/Pagination.svelte
* Routes/games/+page.svelte
* Routes/games/+page.server.js
* Routes/developers/+page.svelte
* Routes/developers/+page.server.js
* Lib/server/games.js
* Lib/server/developers.js

## Bidirektionale Verknüpfung

Um die Verbindung zwischen Spiel und Entwickler für Nutzer intuitiv nachvollziehbar zu machen, wurde eine bidirektionale Verlinkung implementiert. In der Spiel-Detailansicht wird der Name des Entwicklers angezeigt und ist verlinkt auf die entsprechende Entwickler-Detailseite. In der Entwickler-Detailansicht werden alle von diesem Entwickler erstellten Spiele angezeigt. Jedes Spiel ist dabei als GameCard (Komponente) mit Link zur jeweiligen Spiel-Detailansicht dargestellt.

Ein Bild, das Text, Screenshot, Wasser, weiß enthält.

KI-generierte Inhalte können fehlerhaft sein. Ein Bild, das Text, Screenshot, Visitenkarte enthält.

KI-generierte Inhalte können fehlerhaft sein.

Dateien:

* Routes/games/[appid]/+page.svelte
* Routes/games/[appid]/+page.server.js
* Routes/developers/[developer\_id]/+page.svelte
* Routes/developers/[developer\_id]/+page.server.js
* Lib/server/games.js
* Lib/server/developers.js

## Komponenten

Um den Code wartbarer und modularer zu gestalten, wurden wiederverwendbare Svelte-Komponenten erstellt. Diese Kapselung verbessert die Struktur der Anwendung, vermeidet Duplikate und vereinfach zukünftige Erweiterungen oder Änderungen.

Verwendete Komponenten:

* GameCard.svelte
* Pagination.svelte
* Navbar.svelte
* DeveloperListItem.svelte

Dateien:

* lib/components/GameCard.svelte
* lib/components/Pagination.svelte
* lib/components/Navbar.svelte
* routes/+page.layout
* routes/developers/+page.svelte
* routes/games/+page.svelte

## Fehlerbehandlung und Redirects

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Verwendet bei:

* Detailansicht von Spielen und Entwicklern
* Ungültige Formulareinträge
* Löschen von Datensätzen
* Nach erfolgreichen Formularaktionen

Die Anwendung enthält systematische Fehlerbehandlung sowie gezielte Redirects zur Verbesserung der Nutzerführung und Robustheit.

**Fehlerbehandlung**

Nicht gefundene Ressourcen: Wenn ein Spiel oder Entwickler nicht in der Datenbank gefunden wird, wird ein 404-Fehler geworfen. Dies wird in der load-Funktion durch throw error(404, '...') ausgelöst und auf der Seite benutzerfreundlich angezeigt.

Serverseitige Fehler: Falls eine Datenbankaktion fehlschlägt (z. B. Spiel nicht gefunden beim Löschen), wird der Nutzer nicht im Unklaren gelassen, Fehlermeldungen werden geloggt und die Weiterleitung findet dennoch statt.

**Redirects**

Nach erfolgreichem Erstellen/Bearbeiten/Löschen: Nach POST-Aktionen wird der Benutzer mit throw redirect(303, '/...') zurück zur Übersicht geleitet.

Dateien:

* /routes/games/[appid]/+page.server.js
* /routes/developers/[developer\_id]/+page.server.js
* /routes/games/create/+page.server.js
* /routes/games/[appid]/edit/+page.server.js
* /routes/games/[appid]/delete/+server.js