Serverdefinition

Rotkäppchen Spiel Jump’n’Run

Verfasserinnen:	Sabrina Gasser E-Mail: sabrina.gasser@stud.fhgr.ch Tabea Kölliker E-Mail: tabea.koelliker@stud.fhgr.ch Tanya Zoller E-Mail: tanya.zoller@stud.fhgr.ch
Referent:	Marc-Alexander Iten
Modul:	DGUI
Bearbeitungszeitraum:	17. Februar 2022 bis 3. Juni 2022

Zürich, Juni 2022

1 Technologie-Stack

1.1 Frontend

JavaScript

- JS wurde benutzt, weil es eine flexible, leistungsstarke Sprache ist, die von den Browsern unterstützt wird
- Für die Interaktivität von Webseiten
- Ist schnell und weit verbreitet (findet viel Hilfe im Internet)
- Gelernt in der Schule, wir wollten sie einsetzen & dazulernen

Benutzte Library

- Kaboom js (https://kaboomjs.com/lib/0.5.0/kaboom.js)
- Man braucht keine fortgeschrittene Programmierkenntnisse, um ein Spiel programmieren zu können
- Die Programmierer:innen bleiben motiviert, da die Erstellung einfacher ist als ohne die Kaboom Library

HTML & CSS

- Für die Struktur & das Aussehen

1.2 Backend

PHP

- Im Spiel soll ein Kontaktformular integriert werden. Um mit der Datenbank zu kommunizieren, wird die serverseitige Sprache PHP eingesetzt

1.3 Datenspeicher

Relationale Datenbank

- Es soll eine relationale Datenbank eingesetzt werden
- Für diese Datenbank wird MySQL verwendet
- Da man sich mit einem Benutzernamen einloggen soll, das Konto anschliessend an den Highscore und die Messages hängt und die Daten immer wieder abgerufen werden (weil die Benutzenden mehrmals spielen können), ist eine relationale Datenbank geeignet.
- Die relationalen Datenbanken sind gut strukturiert und reduzieren die Fehlerwahrscheinlichkeit

2 Frontend-API

Neue API (da Ergänzung mit Passwort & Level)

POST /jump-and-run/register
{id: int, username: string, password: string}
GET /jump-and-run/register/
{id: int, username: string, password: string}

POST /jump-and-run/login
{username: string, password: string, user-token: string}
GET /jump-and-run/login/{user-token}
{username: string, password: string, user-token: string}

POST /jump-and-run/logout
{user-token: string}

POST /jump-and-run/message
{id: int, username: string, time: string, message: string}
GET /jump-and-run/message
{id: int, username: string, time: string, message: string}

POST /jump-and-run/score
{ id: int, username: string, score: int, level: int}
GET /jump-and-run/ score
{ id: int, username: string, score: int, level: int}

3 Datenbank

user

user_id (PK)	Int
username	String
password	String
user-Token	String

Es ist geplant, ein Login & Passwort zu erstellen, wobei sich die Benutzer:innen jederzeit von allen Geräten einloggen können. Bewusst wurde entschieden kein E-Mail zu nutzen, da es nicht benötigt wird und es wichtig ist, den Datenschutz der Personen zu beachten.

message

message_id (PK)	Int
user_id (FK)	Int
time	Timestamp
message	String

score

score_id (PK)	Int
user_id (FK)	Int
score	Int
level	Int

contact

contact_id (PK)	Int
e-mail	String
message	String
vorname	String
nachname	String

4 Backend

4.1 Register

API /jump-and-run/register → ein Benutzerkonto kann eröffnet werden

ID

- Primary Key
- gespeichert als Zahl (int)
- aufsteigend, beginnend bei 1
- unique (darf in der Tabelle nicht 2x vorkommen)
- wird automatisch generiert, muss nicht vom User eingegeben werden

Username

- gespeichert als Zeichenkette (string)
- min. 5 Zeichen, max. 20 Zeichen, Sonderzeichen & Zahlen sind erlaubt
- unique (darf in der Tabelle nicht 2x vorkommen)

Password

- wird auf dem Frontend als HD5-Hash umgewandelt

- wird als HD5-Hash gespeichert
- min. 8 Zeichen, enthält min. 1 Sonderzeichen & 1 Zahl

4.2 Login

API /jump-and-run/login/{username} → ein Benutzer kann sich einloggen

Username

- wird in der Datenbank gesucht
- da er eindeutig ist, kann er dem entsprechenden User zugeordnet werden

Password

- wird auf dem Frontend als HD5-Hash umgewandelt
- wird abgeglichen mit dem Passwort von der Registration
- wurde beim jump-and-run/login definiert

User-token

- identifiziert die Benutzersitzung
- wird automatisch generiert, muss nicht vom User eingegeben werden

4.3 Logout

API /jump-and-run/logout → Ein Benutzer kann sich ausloggen, Vorgang beendet die Benutzersitzung

User-token

- wurde beim Login generiert
- der Wert identifiziert den Benutzer, der abgemeldet werden soll.

4.4 Nachricht

API /jump-and-run/message → Nachricht kann verfasst werden, um im Chat miteinander zu kommunizieren (die ganze Community, nicht nur zwei Personen untereinander)

ID

- Primary Key
- gespeichert als Zahl (int)
- aufsteigend, beginnend bei 1
- unique (darf in der Tabelle nicht 2x vorkommen)
- wird automatisch generiert, muss nicht vom User eingegeben werden

Username

- Fremdkey aus der Tabelle «user»
- siehe Spezifikationen im Kapitel 4.1 User

Time

- gespeichert als Zeichenkette (string) mit der aktuellen Zeit & dem Datum

Message

- Fliesstext
- max. 200 Wörter
- gespeichert als Zeichenkette (string)

4.5 Score

API /jump-and-run/score → Score kann gespeichert werden, um den eigenen Highscore zu sehen oder den von allen (sofern der User das möchte)

ID

- Primary Key
- gespeichert als Zahl (int)
- aufsteigend beginnend bei 1
- unique (darf in der Tabelle nicht 2x vorkommen)
- wird automatisch generiert, muss nicht vom User eingegeben werden

Username

- Fremdkey aus der Tabelle «user»
- siehe Spezifikationen im Kapitel 4.1 Username

Score

- Zahl
- wird vom Local Storage aus dem Spiel.js übernommen (da wird der Score definiert)

Level

- Zahl
- wird vom Local Storage aus dem Spiel.js übernommen (da wird der Score definiert)

4.6 Contact

Speichern in Datenbank contact → Der User kann in einem Kontaktformular den Programmiererinnen eine Nachricht schicken.

contact_id

- Primary Key
- gespeichert als Zahl (int)
- aufsteigend beginnend bei 1
- unique (darf in der Tabelle nicht 2x vorkommen)
- wird automatisch generiert, muss nicht vom User eingegeben werden

e-mail

- enthält ein @ & min. einen Punkt
- gespeichert als Zeichenkette (string)

message

- Fliesstext
- max. 300 Wörter
- gespeichert als Zeichenkette (string)

Vorname

- gespeichert als Zeichenkette (string)

Nachname

- gespeichert als Zeichenkette (string)