

SGDb: Semantic Video Games Database

Anastasia Chernysheva
a.chernysheva@oth-aw.de

Jakob Götz
goetz@oth-aw.

Ardian Imeraj
a.imeraj@oth-aw.de

Patrice Korinth
p.korinth@oth-aw.de

Philipp Stangl
p.stangl1@oth-aw.de

Zusammenfassung—Wir beschäftigen uns mit der Speicherung und Suche vernetzter Informationen von Videospielen. Dazu stellen wir *SGDb* vor – eine webbasierte Anwendung mit einer Graphen-basierten Suche von Videospielen.

I. EINLEITUNG

Das erste Videospiel *Tennis for Two* aus dem Jahr 1958 ebnete den Weg für die Spieleindustrie. Über Jahrzehnte hinaus kamen immer mehr verschiedene Spiele auf den Markt, die sich in ihrer Art und Weise unterscheiden. Mit den heutigen Videospielen ist es möglich, sich mit Spielern aus der ganzen Welt zu messen oder gemeinsam zu spielen. Aus den unzählig veröffentlichten Videospielen ist eine breite Palette an Genres entstanden. Dabei wird in der Spieleindustrie grob in sechs Kategorien unterschieden, hierbei zählen die Strategie- und Action-Spiele zu den beliebtesten Genres [1]. Zu den anderen vier Kategorien zählen hier die Rollen-, Abenteuer-, Simulationen- und Sonstige Spiele. Jedes dieser Kategorien besitzt weitere Unterkategorien, die sich in ihrer Art und Weise unterscheiden.

Um einen Überblick zu erhalten, wie die Spiele zusammenhängen, präsentieren wir hier eine plattformunabhängige Anwendung, welche die Spiele in einem Graphen darstellt. In Abschnitt II des Konzeptpapiers werden bereits bestehende Anwendungen vorgestellt, die sich mit der Kategorisierung von Spielen und Darstellung von Graphen beschäftigen. In Abschnitt III werden die Anforderungen in Form von User Stories beschrieben. Zum Schluss wird auf die zu verwendenden Technologien zur Umsetzung des Projekts eingegangen, welche in Abschnitt IV näher beschrieben werden.

II. VERWANDTE ARBEITEN

Die *Internet Video Games Database* (IGDB) [2] gibt einen Eindruck, was ein Nutzer von einer Videospieldatenbank erwarten kann. IGDB umfasst derzeit 217.233 Spiele, eine Suchfunktion, die Möglichkeit Videospiele zu entdecken (z.B. kürzlich veröffentlichte Videospiele), und eine Entwickler-API. Über die API können Entwickler auf die Videospieldaten mittels Protokolle wie REST oder ProtoBuf zugreifen.

Wie die Visualisierung eines Graphen möglicherweise aussehen kann, demonstriert die Anwendung Obsidian. Diese erstellt aus von Nutzern kreierten Markdown Dateien eine Graphoberfläche. Über Verlinkungen zwischen den Markdown Dateien wächst der Graph an. Der Nutzer kann den Graph filtern oder gewisse Teile daraus hervorheben (vgl. Abbildung 1).

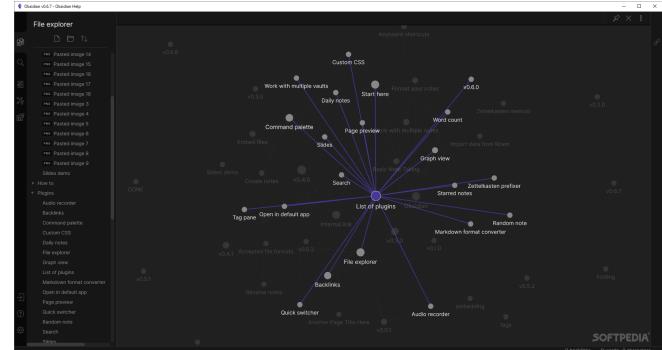


Abbildung 1. Obsidian Graphenoberfläche [3]

III. ANFORDERUNGEN

In der Anforderungsanalyse wurden zwei primäre Ziele identifiziert, die es umzusetzen gilt: (1) Die Überführung von relationalen Videospieldaten aus IGDB in eine semantische Graphdatenbank, und (2) die Möglichkeit zur Suche von Videospiele per Graph oder Texteingabe.

A. Suche per Graph

Als Benutzer möchte ich in eine Graphenoberfläche, welche visuell Verbindungen zu anderen Spielen aufzeigt. Somit kann ich direkt ähnliche Spiele, welche mir ebenfalls gefallen könnten, sehen. Akzeptanzkriterien sind:

- Auf der Startseite ist ein Graph vorhanden, der alle in der Datenbank vorhandenen Videospiele und deren Beziehungen zueinander anzeigt.
 - Schwebt man mit der Maus über einen Knoten öffnet sich ein Informationsfeld, indem angezeigt wird:
 - Titel des Videospiels
 - Genre
 - Bewertung
 - Schwebt man mit der Maus über einen Knoten werden alle verbundenen Knotenpunkte und Kanten im Graphen hervorgehoben.

B. Suche per Texteingabe

Als Benutzer möchte ich ein Videospiel per Texteingabe suchen können, da sich ein spezielles Spiel bei einem Graphen mit vielen Knoten und Kanten nur schlecht finden lässt. Akzeptanzkriterien sind:

- Ein Eingabefeld für die textuelle Suche ist vorhanden.
 - Ist die Suche erfolgreich, werden relevante Knotenpunkte und Kanten im Graphen hervorgehoben.

- Ist die Suche fehlerhaft oder liefert kein Ergebnis, wird der Benutzer darüber benachrichtigt.

C. Detailseite für Videospiel

Als Benutzer möchte ich eine Detailseite zu einem jeweili-gen Videospiel, damit ich ohne die Anwendung zu verlassen die wichtigsten Informationen zu einem Spiel einsehen kann. Akzeptanzkriterien sind:

- Die Detailseite öffnet sich per Klick auf einen Knoten im Graphen.
- Details die angezeigt werden:
 - Titel
 - Erscheinungsdatum
 - Beschreibung des Videospiele
 - Bewertung
 - Genre
 - (Optional) Internetpräsenz (z.B. offizielle Website)
- (Optional) Anzeige des Videospiel-Covers.

D. Filter

Als Benutzer möchte ich mit einer Filteroption Knoten und somit Spiele im Graphen visuell hervor heben. Dadurch kann auch in einem großen Graphen der Nutzer eine Gruppe von Spielen problemlos finden. Akzeptanzkriterien sind:

- Schaltfläche für Filter ist vorhanden.
- Filteroptionen sind:
 - Genre
 - Plattform
 - Hersteller
 - Erscheinungsjahr
- Nur das Filterergebnis soll im Graphen sichtbar sein.
- Wird der Filter aufgehoben sollen alle entfernten Konten wieder angezeigt werden.

E. Testabdeckung

Als Entwickler möchte ich eine ausreichende Testabdeckung, damit Fehler frühzeitig erkannt werden. Akzeptanzkri-terien sind:

- Die Abdeckungsrate liegt bei mindestens 50%.
- Sowohl Frontend- als auch Backend-Code wird mit ge-eignetem Testwerkzeug getestet.

F. (Optional) Cloud-Kompatibilität

Als Entwickler möchte ich eine Cloud-kompatible Anwen-dung für eine Bereitstellung der Anwendung. Akzeptanzkrite-rien sind:

- Für jede Komponente der Anwendung ist ein Dockerfile erstellt und lauffähig.
- Mit Docker Compose ist eine YAML-Datei vorhanden, um mit einem einzigen Befehl alles Container zu starten und zu beenden.
- (Optional) Die Anwendung ist bei einem Cloud-Anbieter (z.B. Amazon Web Services) bereitgestellt und erreichbar.

IV. METHODEN

Ziel bei der Entwicklung und bei der späteren Verwen-dung ist eine plattformunabhängige Anwendung, die eine Schnittstelle für die *Linked Open Data Cloud* als auch der Bedienoberfläche der Anwendung bereitstellt. Daraus ergeben sich folgende technische Schlüsselbausteine:

Für die Repräsentation der semantischen Daten im Frontend wird das Framework Svelte verwendet. Die Daten werden voraussichtlich in einer Ontotext GraphDB Instanz persistiert. Im Backend wird FastAPI für eine REST-Schnittstelle und RDFLib zum arbeiten mit RDF verwendet. Die Kommunikati-on zwischen Frontend und Backend wird über eine RESTful-API abgewickelt. Die einzelnen Komponenten der Anwendung (Frontend, Backend und Datenbank) werden mittels Docker containerisiert.

LITERATUR

- [1] Statistica. (2022). “Statistica.” [Online], Adresse: <https://de.statista.com/prognosen/999755/deutschland-beliebteste-genres-bei-videospielen>.
- [2] IGDB. (2022). “IGDB.” [Online], Adresse: <https://www.igdb.com/>.
- [3] Softpedia. (2022). “Obsidian.” [Online], Adresse: https://windows-cdn.softpedia.com/screenshots/Obsidian-app_1.png.