SGDb: Semantic Video Game Database
Technical Report

Anastasia Chernysheva *a.chernysheva@oth-aw.de*
Jakob Götz *j.goetz@oth-aw.de*
Ardian Imeraj *a.imeraj@oth-aw.de*
Patrice Korinth *p.korinth@oth-aw.de*
Philipp Stangl *p.stangl1@oth-aw.de*

***Zusammenfassung*—Dieser Technical Report beschreibt die Architektur von SGDb – eine webbasierte Anwendung mit einer Graphen-basierten Suche von Videospielen.**

I. Einführung und Ziele

In den weiteren Abschnitten des Technical Reports wird zuerst auf die Lösungsstrategie in Abschnitt II eingegangen. Im nächsten Abschnitt III wird das Gesamtsystem aus Bausteinsicht beschrieben. Anschließend wird in Abschnitt IV die Verteilungssicht der Anwendung beschrieben. In Abschnitt V werden die angewandten Werkzeuge zur Entwicklung der Anwendung vorgestellt. Abschließend wird ein Fazit und Ausblick in Abschnitt VI gegeben.

II. Lösungsstrategie

Dieser Abschnitt enthält einen stark verdichteten Architekturüberblick. Eine Gegenüberstellung der wichtigsten Ziele und Lösungsansätze.

III. Bausteinsicht

Diese Sicht zeigt die statische Zerlegung des Systems in Bausteine sowie deren Beziehungen.

A. Gesamtsystem

B. Backend

Das Backend dient als Schnittstelle für die Interaktion zwischen Datenbank und dem Frontend. Zusätzlich ist es für die Verarbeitung von Ressourcen zuständig. Als Framework wird hierfür FastAPI [1] genutzt, welches als Sprache Python verwendet. Durch das verwendete Framework wurde eine REST-API verwirklicht, welche durch das Frontend angesprochen werden kann. Die API stellt für das Frontend folgende Daten bereit:

- Daten für einen Graphen mit einem Root-Knoten, welcher bei der Startpage angezeigt wird
- Daten, welche Spielinformationen über ein angefordertes Spiel enthalten
- Daten zur Darstellung einer Detailseite eines bestimmten Spiels
- Daten, welche einen Graphen bilden, welcher von Filtereinstellungen betroffen ist

Des Weiteren wird als Test-Framework im Backend Pytest [2] genutzt.

Durch das Python basierte Backend wird zur Integration mit der Graph-Datenbank die Python-Bibliothek SPARQLWrapper [3] genutzt. Diese dient als einfacher Python-Wrapper für SPARQL und ermöglicht so die Ausführung von Queries in der SPARQL-Syntax. Dazu bietet es die Möglichkeit, das Ergebnis in das gewünschte Format zu formatieren, sodass diese Daten leichter in Python weiterverarbeitet werden können. SPARQL selbst ist eine graphenbasierte Query-Sprache, die mit Daten im RDF-Format arbeitet. Zuerst wird ein Graphenobjekt definiert, das alle Daten aus der Graphdatenbank beinhaltet. Dieses dient als Grundlage für die weiteren Abfragen. Die Abfragen sind in die Wortsuche sowie die unterschiedlichen Filter unterteilt, die nach Datum, Genre, Bewertung, Entwickler und Plattform filtern können. Sie lassen sich beliebig kombinieren. Die Abfragen bekommen die im Frontend festgelegten Argumente und filtern zunächst nach den gewünschten Kriterien und liefern die Videospiel-IRI‘s in Form von Objekten zurück. Entsprechen die Objekte den Anforderungen, werden diese als Subjekte in der nächsten Abfrage genutzt, um alle zugehörigen Prädikate sowie Objekte der Videospiele im JSON-Format zurückzugeben. Diese werden im Frontend für die Darstellung der Knoten sowie für die Inhalte in der Detailseite genutzt.

C. Datenbeschaffung und Datenbank

Für die Datenbeschaffung wird eine Client-Anfrage an die Endpunkte der IGDB-API [4] gesendet. Als Antwort wird im Anschluss der Bearbeitung ein Datensatz zurückgegeben, der wesentliche Informationen zu einem Spiel beinhaltet. Zu den Informationen zählen: Spiele-ID, Spieletitel, Beschreibung, Veröffentlichungsdatum, Genre, Spieleplattform, Bewertung, Entwicklername- und Land sowie eine URL des Coverbildes. Der Datensatz wird im JSON-Format gespeichert und auf 500 Spiele reduziert, die eine hohe Bewertung erzielt haben (>85). Begründet wird diese Bedingung durch das Bestreben, einen Datensatz zu erzeugen, der über viele Informationen zu den einzelnen Spielen verfügt. Der Datensatz weist an mancher Stelle numerische Werte auf, die jedoch nicht informativ wären für einen Durschnittsnutzer. So ist das Veröffentlichungsdatum im Unix-Zeitstempel angegeben. Dieser ist auch als "The Epoch"bekannt und zählt die Anzahl der vergangenen Sekunden seit 1.Januar 1970 [5]. Der Standort des Entwicklerunternehmens wird als dreistelliger Country-Code definiert, der von der International Organization for Standardization (ISO) entwickelt und im ISO-3166 publiziert wurde [6]. Es wurde eine Funktion implementiert, die die Unix-Zeit in ein reguläres Zeitformat *'DD-MM-YYYY'* und den ISO-Country-Code in Ländernamen konvertiert.

Der Datensatz wird anschließend mit Refine [7] strukturiert und in RDF-Triples (Graphs) umgewandelt. Für die Datenver-

waltung und Vernetzung der Informationen wird die Ontotext GraphDB [8] verwendet.

D. Frontend

Das Frontend ist für die Benutzerschnittstelle verantwortlich. Hauptbestandteile des Frontends sind die Suchmaske, der Graph und die Detailseite zu einem Videospiel. Für die Darstellung des Graphen und die Interaktion mit diesem wird Sigma.js [9] verwendet. Sigma.js rendert Graphen mit WebGL. Damit lassen sich größere Graphen schneller zeichnen als mit Canvas- oder SVG-basierten Lösungen. Das Graphenmodell wird in einer separaten Bibliothek namens Graphology [10] verwaltet. Dies ist eine Standardbibliothek mit Algorithmen aus der Graphentheorie und allgemeinen Hilfsprogrammen wie z.B. Graphengeneratoren.

Generierung Für das Graphen Layout wird der ForceAtlas2 Algorithmus [11] verwendet. Vor dem Start von ForceAtlas 2 Layout muss die Startposition jedes Knotens festgelegt werden. Daher müssen zwei Attribute namens x und y für alle Knoten des Diagramms definiert werden.

Dazu wird ein Graph Objekt mit dem Random layout erzeugt. Positionierung jedes Knotens, indem die Koordinaten gleichmäßig nach dem Zufallsprinzip auf dem Intervall [0, 1) auswählt.

Die Interaktion mit dem Backend erfolgt über die REST-Schnittstelle.

IV. Verteilungssicht

V. Entwicklungswerkzeuge

VI. Fazit und Ausblick

Im Rahmen der Arbeit wurde mit einem begrenzten Datensatz von 500 Spielen eine erste Version der Anwendung entwickelt.

Literatur

[1] S. Ramírez. “FastAPI Documentation.” [Online]. (2022), Adresse: https://fastapi.tiangolo.com/.

[2] K. et al. “Pytest Documentation.” [Online]. (2022), Adresse: https://pytest.org/.

[3] I. H. E. al. “SPARQLWrapper Documentation.” [Online]. (2022), Adresse: https://sparqlwrapper.readthedocs.io/en/latest/.

[4] IGDB. “IGDB-API.” [Online]. (2022), Adresse: https://api-docs.igdb.com.

[5] Datumsformat. “UNIX-Zeitstempel konvertieren.” [Online]. (2022), Adresse: https://www.datumsformat.de/unix/.

[6] I. Standarts. “ISO 3166 Country Codes.” [Online]. (o.J.), Adresse: https://www.iso.org/iso-3166-country-codes.html.

[7] Ontotext. “Ontotext Refine.” [Online]. (2022), Adresse: https://www.ontotext.com/products/ontotext-refine/.

[8] Ontotext. “Ontotext GraphDB.” [Online]. (2022), Adresse: https://www.ontotext.com/products/graphdb/.

[9] Sigma. “Sigma.js.” [Online]. (2022), Adresse: https://www.sigmajs.org/.

[10] Graphology. “Graphology.” [Online]. (2022), Adresse: https://graphology.github.io/.

[11] M. Jacomy, T. Venturini, S. Heymann und M. Bastian, “ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software,” *PloS one*, Jg. 9, Nr. 6, e98679, 2014.