Projekt GameDB

Persönliche Ziele

- Meine Python-Kenntnisse auffrischen und erweitern
- Daten mit Hilfe von code generieren
- Bereits gesammeltes Wissen über MongoDB und Python verknüpfen

Lernjournal

Datum	*Journal*	*Dauer*
25.9.	Heute stellte ich die Struktur zum Arbeitsjournal zusammen, plante das Projekt und begann damit, ein Skript zu schreiben, welches mir eine Games Datenbank generiert. **Planung des Projekts** Begonnen habe ich dieses Projekt, wie die meisten anderen auch. Ich legte einen neuen Ordner an, mit den Dateien: «00_Project Overview», welche den Namen des Projektes «GameDB», die nach meinem Schema generierte Project ID «23-E-PY-01», sowie eine Checklist mit allen Schritten, die ich befolgen werde, um dieses Projekt zu meistern und nichts zu vergessen. Die andere Datei «00_GameDB Description» enthält die Anweisungen zum Projekt, also die von Herr Lurati ausgehändigte PDF Datei, sowie die Meilensteine dieses Projektes. **Das Aufsetzen der Datenbank** Die Datenbank und die Collection habe ich ganz einfach in der MongoDB Compass App erstellt. Das Skript, welches die Datenbank mit zufällig generierten Filmen füllt, habe ich in Python programmiert. im aktuellen Zustand ist es in der Lage, einen zufälligen Titel, eine zufällige Bewertung und zufällige Downloadzahlen zu generieren. So wurden die einzelnen Daten generiert: • Die **Namen** sollten jeweils aus einem Adjektiv und einem Nomen bestehen. Für die Nomen und Adjektive nutzte ich eine von ChatGPT generierte Liste. • Das **Ausgabejahr** ist ganz simple eine zufällige Zahl zwischen 1980 und 2023 • Die **Downloadzahlen** sind ebenfalls einfach eine zufällige Zahl zwischen 10 und 100'000 Um die Daten in die Datenbank einzufügen, verwendete ich PyMongo. Ich verband mich mit der Datenbank und generierte und setzte 100 Spiele ein. **Reflexion** Für das Befüllen der Datenbank wollte ich zuerst eine API verwenden, doch auch nach einer längeren Recherche konnte ich keine finden, die meinen Anforderungen entsprach, weshalb ich mich schliesslich dazu entschieden habe, alles vollkommen random selbst generieren zu lassen. Dieses Ereignis zog mich erst natürlich runter, da ich zum einen Zeit verschwendet habe, und ich zum anderen meine Vorstellung nicht umsetzen konnte. ChatGPT so zu verwenden war etwas neues für mich und ich kann	3 Lkt.

	mir vorstellen auch in Zukunft «nervigen» Code so generieren zu lassen. In retrospektive fällt mir auf, dass ich theoretisch auch ganze, echte, Daten mit ChatGPT hätte generieren lassen.	
19.10.	Heute beendete ich das Programm zur Generierung des Datensatzes. Zudem wendete ich es an und befüllte somit meine Datenbank. **Fertigstellung des Programms** Ich fügte den Code zur Generierung der restlichen Daten hinzu. Die Daten werden wie folgt generiert: **Bewertung**: Eine zufällige Zahl zwischen 1 und 10 inklusive einer Dezimalstelle. **Altersgrenze**: Entweder 0, 6, 12, 16 oder 18 **Genres**: Zwischen 1 und 4 Genres und entweder Multi- oder Singleplayer. Es ist in einem Set gespeichert, um sicherzustellen, dass kein Genre doppelt vorkommt. **Hello PyMongo!** Heute habe ich zudem meine ersten Zeilen PyMongo geschrieben. Es war nicht sonderlich herausfordernd, da der Syntax sehr nah am MongoDB Terminal ist. Ich verband mich mit der Datenbank und fügte die Daten ein **Reflexion** Ich lernte die Grundlagen von PyMongo. Ich kann mich mit einer Datenbank verbinden und neue Datensätze einfügen. Ich musste auch die Erfahrung machen, dass MongoDB keine Sets als Input nehmen kann. Es muss zuerst in eine Liste konvertiert werden. Mir machte es Spass, neues zu lernen und herauszutüfteln, wie ich die Genres generieren soll.	40’
23.10.	Heute begann ich mit der Programmierung des eigentlichen Programms. Ich Programmierte den "Startbildschirm" und die Auswahllogik. **Hi CLI** In der Schule programmierte ich das CLI. Ich informierte mich also im Internet zum Thema CLI-Libraries für Python. Ich strebte erst ein Selektionsmenü mit den Pfeiltasten an, doch diesen Plan verwarf ich und endete schliesslich bei rich, einer relativ simple gehaltene Library mit all den Funktionen, die ich benötigte. Ich begann mit dem Main Loop und dem ersten Auswahlmenü. Von dort aus habe und werde ich immer mehr Funktionen hinzufügen. Dieser Teil hat verhältnismässig die meiste Zeit in dieser Session beansprucht, da ich meine erste Idee mit den Selektionen verwarf. **Daten anzeigen** Unter «Anzeigen > Alle» gibt es nun die Auswahl zwischen «JSON» und «Tabelle». Wird «JSON» gewählt wird das Ergebnis der Search query einfach so (mit rich formatiert) geprintet. Wird jedoch «Tabelle» gewählt, werden die Daten in eine rich Tabelle eingefügt und diese ausgegeben. **Reflexion** Ich lernte das Arbeiten mit der Python Library rich. Ich lernte, wie ich mit Hilfe von rich die Farbe von Text ändern, Markdown verwenden, JSON formatieren und Tabellen generieren kann. Natürlich habe ich dabei meine Pythonkentnisse festigen und erweitern können. Das Arbeiten macht mir weiterhin Spass. Zum einen, weil ich mein Wissen auf die Probe stellen und neues dazulernen kann, und zum andern, weil ich erwartet habe,	3,5 Lkt.

	dass dieses Projekt mich an meine Grenzen bringen würde, was es glücklicherweise nicht tut. Ich glaube das Projekt ist auf einem guten Weg. Ich glaube zwar, dass ich etwas effizienter arbeiten könnte, aber bisher habe ich nicht das Gefühl, dass ich nicht rechtzeitig fertig werden werde. Im nächsten Schritt will ich die weiteren Funktionen von «Anzeigen» programmieren. Und später die anderen CRUD-Operatoren hinzufügen	
30.10.	Heute beende ich das «Anzeigen» Menü und arbeitete am Code, der es ermöglicht, Daten in der Datenbank zu verändern. **Das erste Untermenü ist fertig!!** Heute habe ich in der Schule das Untermenü von «Anzeigen» fertiggestellt, d. h. die Funktion hinzugefügt, einen zufälligen Eintrag anzusehen. Ich nutzte «db.collection.aggregate([{"$sample": {"size": 1}}]», um einen zufälligen Eintrag zu erhalten. Dieser wird dann (nach Wahl des Users) als JSON oder Tabelle ausgegeben. Ich hatte ursprünglich vor, auch andere Kriterien zur Auswahl der Anzeige zur Verfügung zu stellen, wie u. A. genaue Filter oder die Ausgabe in „Blöcken“, was ich jedoch wegliess, um in der Zeit zu bleiben. Habe ich am Ende jedoch noch genug Zeit übrig, werde ich diese Möglichkeiten hinzufügen. Ausserdem habe ich den Code, der den Output einer Query in eine Tabelle umformatiert, in eine externe Funktion kopiert, um den Code sauberer zu halten. **Daten Ändern** Ausserdem arbeitete ich in der Schule (und auch ca. eine Lektion Zuhause) am «Verändern» Untermenü. In diesem gibt es die Möglichkeit, Daten nach dem Titel, dem/n Genre/s und dem Entstehungsjahr zu bearbeiten. Wählt man einer dieser Optionen aus, wird man aufgefordert, Titel/Genre(s)/Entstehungsjahr einzugeben. Anschliessend wird der bzw. werden die Datensätz/e in einer Tabelle ausgegeben und man erhält die Möglichkeit, Name, Genres, Entstehungsjahr, Downloads, Bewertung und Mindestalter zu überschreiben. Bestätigt man dies, werden die Daten in die Datenbank übernommen. Ich hatte ursprünglich vor, auch andere Kriterien zur Auswahl des zu bearbeitenden Eintrags zur Verfügung zu stellen, wie u. A. das Mindestalter, was ich jedoch wegliess, um in der Zeit zu bleiben. Habe ich am Ende noch genug Zeit übrig, werde ich auch diese Möglichkeit hinzufügen. **Reflexion** Heute lernte ich nicht gross neues aktiv. Ich habe jedoch das in der Vergangenheit gelernte gut festigen können. Es machte mir grossen Spass herumzutüfteln, wie ich den Code zum Auswahlverfahren der Felder im «Verändern» Menü möglichst effizient und wiederverwendbar gestalte. Dies hat unter anderem dazu beigetragen, dass ich heute vermutlich das Meiste an Python dazulernte, da ich noch nicht viel mit Dictionaries gearbeitet habe. Etwas, dass ich heute feststellen musste ist, dass ich alleine mit dem Arbeiten in der Schule das Projekt nicht mit all den Funktionen umsetzen kann, die ich mir vorgestellt habe. Ich habe also die unwichtigsten bzw. die zeitaufwendigsten ausgelassen und werde sie evtl. zu Hause noch dazufügen, wenn ich die Zeit finde.	4,5 Lkt.

	Im nächsten Schritt werde ich das Löschen oder einfügen Menü Programmieren. Dies wird vermutlich nicht sehr viel Zeit in Anspruch nehmen, da ich viel Code von heute wiederverwenden kann.	
5.11.	Heute habe ich die letzten beiden 'Grundfunktionen', also das Einfügen und Löschen von Daten hinzugefügt. **Leichter als gedacht** Heute habe ich Zuhause die letzten vorgeschriebenen Funktionen, also das «Einfügen» und «Löschen» von Documents gecodet. Es dauerte weniger lange, wie eingeplant, da ich das Menü zum Einfügen von Daten fast eins zu eins von der «Verändern» Funktion übernehmen konnte. Ich habe lediglich die Variablen umbenannt und «update_many()» mit «insert_one()» ersetzt. Auch die Implementierung von «Löschen» war nicht sonderlich aufwändig. Dort konnte ich den Teil vom «Verändern» Code übernehmen, der für die Selektion des zu verändernden Documents zuständig ist. **Reflexion** Mir ist wieder einmal klar geworden, dass man das Rad wirklich nicht neu erfinden muss. Und wenn man bereits Code hat, der das tut, was er soll, kann man diesen ruhig wiederverwenden/kopieren, um ans Ziel zu kommen. Dementsprechend ist es ziemlich gut verlaufen und es hat mich gefreut, dass ich vierzig Minuten weniger lange arbeiten musste als gedacht.	20’

Gesamtreflexion

Ich finde das Projekt ist mir alles in allem gut gelungen. Ich habe zwar ein paar Features ausgelassen, die ich ursprünglich eingeplant hatte, wie z. B. Filter beim «Anzeigen» Menü, das Anzeigen von Documents in «Blöcken» oder mehr Möglichkeiten, nach Einträgen zu suchen, die man bearbeiten will, aber dafür ist mir die Führung des Journals im vergleich zu früheren Projekten umso besser gelungen. Ich nahm mir fast jedes Mal im Unterricht Zeit, um das Journal zu führen oder machte es direkt im Anschluss zuhause. Auch das Zeitmanagement ist mir gut gelungen. Ich musste zwar wenige, kleinere Features auslassen, konnte dafür aber fast alles während der Zeit im Unterricht erledigen, was mein Ziel war, da ich nebenbei schon genug zu tun habe.

Das Programmieren in Python machte mir Spass, da es mich nicht überforderte. Es motivierte mich sehr, dass ich nur sehr wenig nachschlagen musste und einfach so meine Ideen in Code umsetzen konnte. Aber auch, wenn ich mal etwas z. B. zu PyMongo Recherchierte gefiel mir die Kniffelei, wie ich den code möglichst optimal schreiben könnte, ohne den Druck, dass er perfekt optimiert sein muss.

Ich glaube, ich habe meine Ziele erreicht. Ich konnte mein bisheriges Python Wissen abrufen und habe auch ein wenig dazugelernt. Ich habe mich mit Dictionaries auseinandergesetzt und habe neu List Comprehensions, also eine effiziente Art, mit Listen zu arbeiten gelernt und dies genutzt, um den User Input flexibler zu gestalten. Auch das Ziel, mit Code Daten für die Datenbank zu generieren ist mir gelungen, auch wenn ich es leider nicht so machen konnte, wie zum Anfang gedacht. Das Letzte Ziel war fast unmöglich nicht zu erreichen, da ich natürlich mit Python arbeitete und die PyMongo und MongoDB Queries fast identisch sind. Es bereitete mir Freude, endlich ein Projekt zu machen, bei welchem ich

wirklich mein bisheriges Wissen verknüpfen konnte. Beim Lernen von Python oder auch jetzt MongoDB konnte ich mir nicht wirklich vorstellen, wie man die Themen verknüpfen könnte, ohne dass es eine riesige Applikation wird.

Dieses Projekt hat mich dazu inspiriert, mit Lynn zusammen ein freiwilliges Projekt bei Frau Duc zu machen, indem wir eine Java-App schreiben, mit der eine MySQL Datenbank für ein Hotel verwaltet werden kann.

Zusammengefasst finde ich, dass mir das Projekt gut gelungen ist, für das, dass ich mir in meiner Freizeit nur wenig Zeit genommen habe. Ich hatte zu Beginn Angst, dass ich diesem Projekt nicht gewachsen bin, doch die Umsetzung fiel mir leicht. Ich musste zwar kleinere Features auslassen, doch trotz allem bin ich zufrieden mit dem Resultat und konnte sogar Inspiration für ein neues Projekt nehmen.