TeamWork

Gruppe 7

10. Juli 2022

Sascha Heinze
561055

Mark Unger
554906

Sinan Cem Ertogrul
578035

Mauriz Weymann
528279

Aufgabe 1

a) Schnittstelle

API Doku

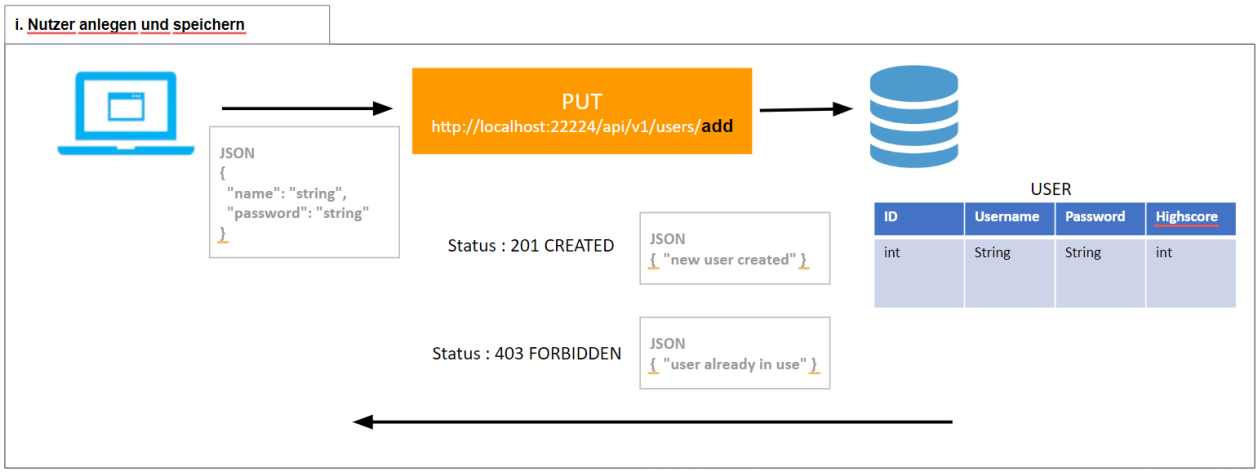
1. (16P) Erweitern Sie Ihr Quizprogramm um einen Server – Teil.

Nutzen Sie für die Kommunikation zwischen dem Client und dem Server die REST – Architektur.

a. (4P) Designen Sie Ihre Schnittstelle und dokumentieren Sie diese. Die Dokumentation wird den anderen Gruppen bereitgestellt, damit diese ihren Client an den Server anbinden können. (bis 16.06. 2022 vor der Übung). Die Schnittstelle soll folgende Funktionen bieten:

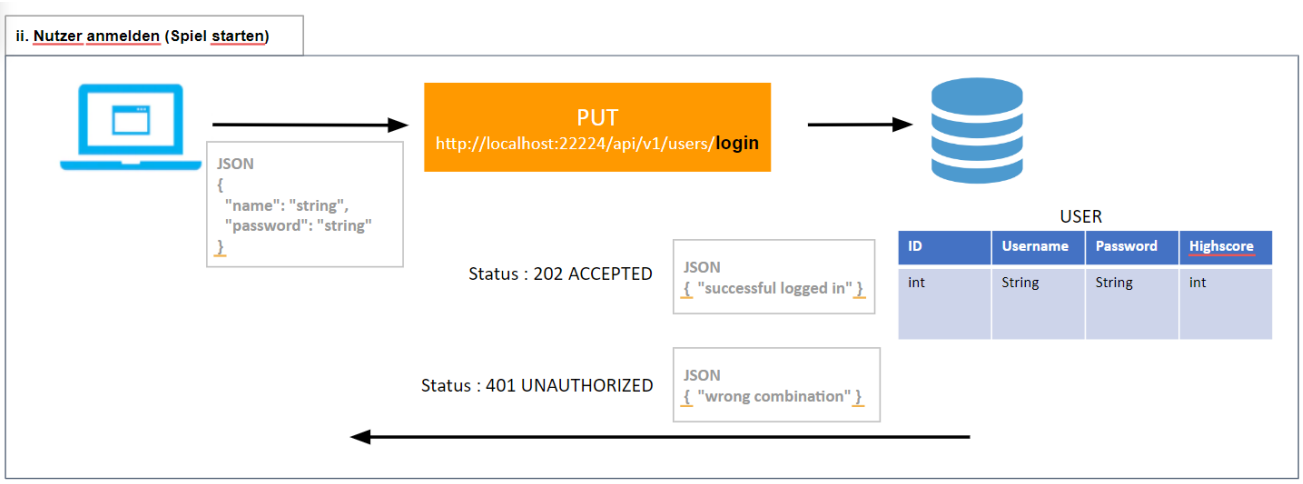
I. Nutzer anlegen und speichern
II. Nutzer anmelden (Spiel starten)
III. Frage(n) abrufen
IV. Frage(n) hochladen und speichern
V. Antwort(en) hochladen
VI. Punkteliste und Highscore aller Spieler abfragen

i.) Nutzer anlegen und speichern:



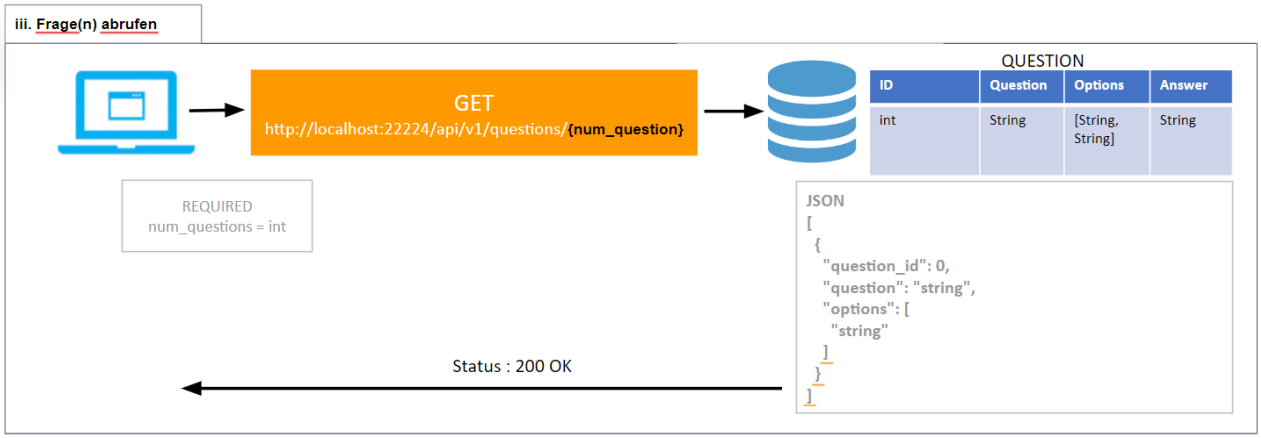
#ID wird automatisch vom Server gesetzt

ii.) Nutzer anmelden (Spiel starten)



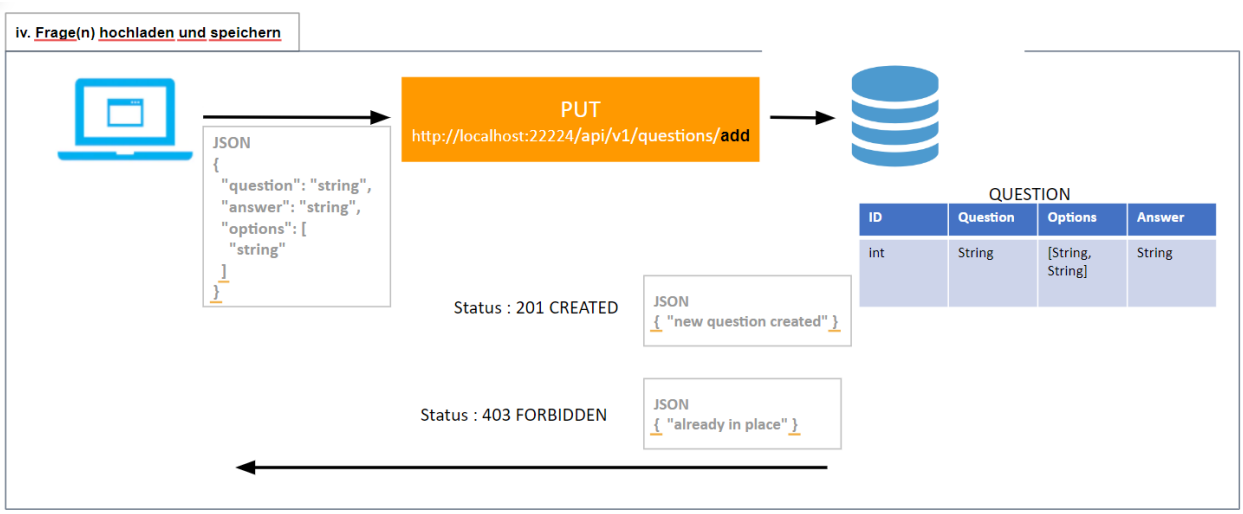
#Username und password mit Serverspeicher matchen und Status, ob Spielerdaten übereinstimmen

iii.) Frage(n) abrufen



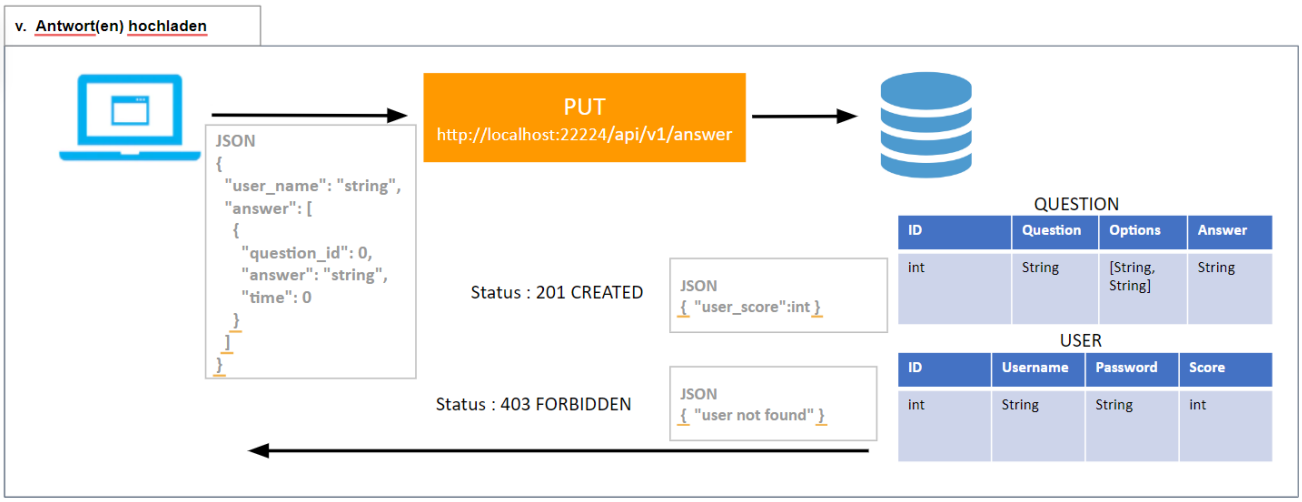
#Der Spieler muss am Anfang des Spiels die gewünschte Anzahl der Fragen angeben. Somit bekommt er als Response die Anzahl der Fragen, die er beantworten muss. Der Server liefert die geforderte Anzahl an Fragen zurück.

iv.) Frage(n) hochladen und speichern



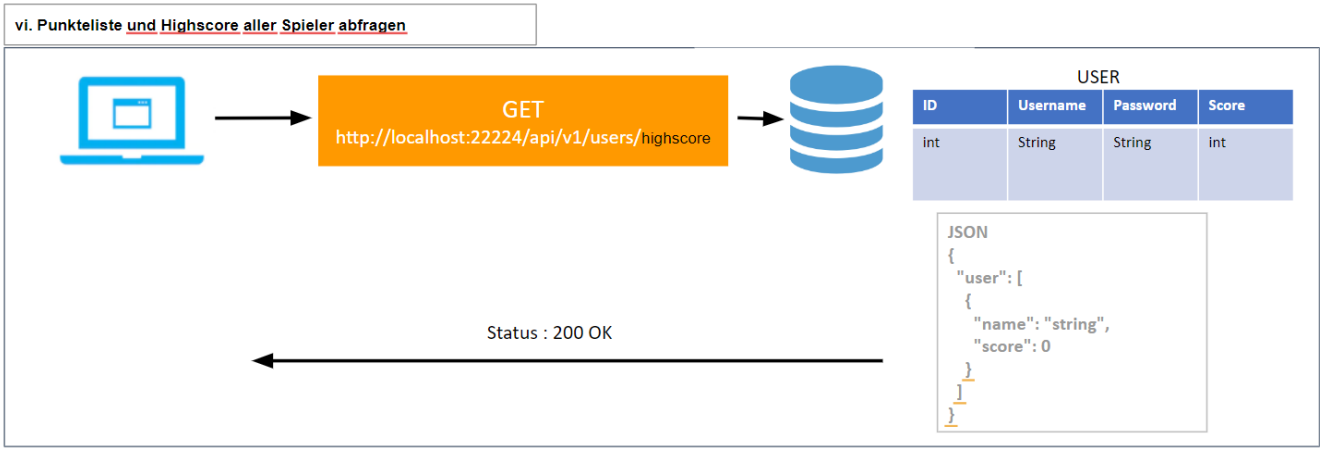
#Eine neue Frage wird zusammen mit zwei Antwortoptionen, sowie der richtigen Antwort an den Server geschickt. Dieser speichert dies alles und vergibt eine neue ID.

v.) Antwort(en) hochladen



#Antworten werden mit jeweiliger ID an den Server übergeben. Die Auswertung erfolgt auf dem Server und das Ergebnis wird dem User zugeordnet.

vi.) Punkteliste und Highscore aller Spieler abfragen



#Die Highscore-Liste wird mit den Namen der Spieler und dem jeweiligen Punktestand zurückgegeben.

Aufgabe 2

REST-Operationen

—

Welche Operationen sind bei REST erlaubt? Nennen Sie vier.

GET	Daten von einer bestimmten Ressource werden angefragt/geladen.
POST	Operationen werden an der Ressource durchgeführt.
PUT	Die Ressource wird angelegt oder ersetzt.
DELETE	Die Ressource wird gelöscht.

Aufgabe 3

WAL-Verfahren

Was passiert beim Write-Ahead-Logging-Verfahren (WAL)?

Von usw.at:
Write-Ahead Logging (WAL) ist eine Standardtechnik für das Aufzeichnen (Loggen) von Transaktionen. Zusammengefasst ist das zentrale Konzept von WAL, dass Änderungen in Datendateien (wo Tabellen und Indexe abgelegt sind) erst geschrieben werden dürfen, wenn diese Änderungen geloggt wurden, das heißt, nachdem Sie im Log aufgezeichnet wurden und die Logeinträge auf ein permanentes Speichermedium geschrieben wurden.
Wenn so vorgegangen wird, dann müssen die Datenseiten nicht nach jeder Transaktion auf die Festplatte zurückschreiben, weil bekannt ist, dass im Falle eines Absturzes die Datenbank mit dem Log wiederhergestellt werden kann.
Änderungen, die noch nicht auf die Datenseiten übertragen wurden, werden zuerst aus den Logaufzeichnungen wiederhergestellt (das ist Vorwärts-Wiederherstellung, auch REDO genannt) und dann werden Änderungen aus nicht abgeschlossenen Transaktionen aus den Datenseiten entfernt (Rückwärts-Wiederherstellung, UNDO).

Von Wikipedia:
Das sogenannte write ahead logging (WAL) ist ein Verfahren der Datenbanktechnologie, das zur Gewährleistung der Atomarität und Dauerhaftigkeit von Transaktionen beiträgt. Es besagt, dass Modifikationen vor dem eigentlichen Schreiben (dem Einbringen in die Datenbank) protokolliert werden müssen.

Durch das WAL-Prinzip wird ein sogenanntes "update-in-place" ermöglicht, d. h. die alte Version eines Datensatzes wird durch die neue Version an gleicher Stelle überschrieben. Das hat vor allem den Vorteil, dass Indexstrukturen bei Änderungsoperationen nicht mit aktualisiert werden müssen, weil die geänderten Datensätze immer noch an der gleichen Stelle zu finden sind. Die vorherige Protokollierung einer Änderung ist erforderlich, um im Fehlerfall die Wiederholbarkeit der Änderung sicherstellen zu können.

Aufgabe 4

Waisenbehandlung mittels Epochen

—

Beschreiben Sie ein Verfahren zur Waisenbehandlung mittels Epochen. Was ist hierbei eine Epoche?

Waise/Orphan:
Eine Waise ist ein Klient, der den Aufruf (z.B. wegen eines ggf. zu kurzen Timeout) abgebrochen hat und nicht mehr am Ergebnis interessiert ist oder der mittlerweile überhaupt nicht mehr zur Verfügung steht.

Epoche:
Epochen entstehen dadurch, dass die Zeit in sequentiell nummerierte Abschnitte aufgeteilt wird.
Eine Epoche ist somit ein sequentiell nummerierter Zeitabschnitt.
Waisenbehandlung per reincarnation (Wiederverkörperung/Wiedergeburt)

- bei Wiederanlauf eines Clients wird eine neue Epoche eröffnet
    - der Beginn einer Epoche wird allen Maschinen mitgeteilt (broadcast)
    - auf den Maschinen werden daraufhin alle Anbieter (Prozesse) terminiert
- jede Anforderungs- und Antwortnachricht enthält eine Epochenkennung
    - damit können unerwartete Antworten von Waisen herausgefiltert werden

Aufgabe 5

Hashwertermittlung

Dateien:

5word.py

dictionary.txt

Hashing.py

Schreiben Sie ein möglichst schnelles Programm, dass folgende Hashwerte errät.

Die Hashwerte sind als Hexadezimalwerte (Funktion: hexdigest()) angegeben. Die Werte sind mit dem Pythonmodul hashlib (https://docs.python.org/3/library/hashlib.html) generiert.
Der Zeichenvorrat ist gegeben durch alle Buchstaben des Alphabetes sowohl kleine Buchstaben a-z und auch die großen Buchstaben A-z. Es können in einem Wort Zeichen auch mehrfach vorkommen! Die maximale Länge pro Wort beträgt 5 Zeichen. Das Ergebnis ist ein Satz in englischer Sprache. Es sind vollständige Worte, keine kryptischen Zeichenkombinationen.
Das von Ihnen geschriebene Programm soll den Satz möglichst schnell erraten. Der Weg, ob Brute Force oder mit einem Wörterbuch ist dabei Ihnen überlassen. Am Ende soll die gesamte Laufzeit und das Ergebnis angezeigt werden.
Für die Bewertung wird das Programm auf einem 6 Kern Rechner mit 32 GB Ram unter Linux getestet. Die 3 schnellsten Programme bekommen noch 5 Zusatzpunkte (Zeitmessung erfolgt extern). Für die 3 kleinsten Programme (gemeint ist die Anzahl der Zeilen und Größe des Quellcodes) gibt es ebenfalls 5 Zusatzpunkte. Die 3 Programme mit der informativsten Ausgabe, während der Berechnung, erhalten ebenfalls 5 Zusatzpunkte. Es werden maximal für eine Kategorie die Zusatzpunkte vergeben.

Zum Testen: hashlib.sha256(b"Test").hexdigest() ergibt:
532eaabd9574880dbf76b9b8cc00832c20a6ec113d682299550d7a6e0f345e25

hashlib.sha3_512:
656ac31bcca3e16bb8d7f864659edceabc5b7de905899052dd9fab980ff33c61a32336ec54f0213515c794fa7352fa3bb0527e32cd659d2dd564847546055501

hashlib.sha512:n
7d7764888dc13da6436666cd97043a5ead014596cd9aef6d12f95a5602d15993beebd44b92014732c340239e7cd1f07c5ed4035d1a02151916e94d43ffa55cfa

hashlib.sha1:
bbccdf2efb33b52e6c9d0a14dd70b2d415fbea6e

hashlib.sha3_384:
d6904cf515fa8e8921c2aab7c9aaafe2232dd065387e17f48da712ec0a25e69ca329453bb545ac4a8ed06d2bce141b96

hashlib.sha256:
0ebb429fa86d481c2630fac53db1c91cffed5d4d41d1021c179444eb67e7ee0b

Fortsetzung Aufgabe 5

—

Bei der Bearbeitung der Aufgabe haben wir uns dafür entschieden, ein dictionary zu verwenden.
Unsere Idee war es nämlich, dass wir alle Hashcodes in einer Schleife durch unser dictionary durchlaufen lassen und damit hoffentlich schneller als bei einer Brute-Force-Methode sind.

Da wir keinen direkten Downloadlink zu englischen Wörtern mit bis zu fünf Buchstaben Länge gefunden haben, haben wir unser dictionary mittels Scraping selbst erstellt.

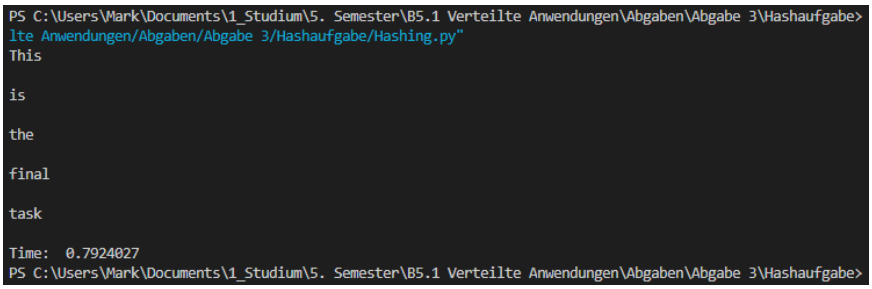
Zur Erstellung des dictionary's wurde die Datei "5word.py" verwendet.

Wenn das Skript komplett durchgelaufen ist, werden die Wörter im dictionary aber bspw. noch wie folgt dargestellt:
['aa']
['Aa']
['ab']
['Ab']
usw.

Daher sind wir zu guter Letzt noch in den Text Editor gegangen, haben nach folgenden Zeichen:
"[", "]" , " ' " (einzelnes Anführungszeichen)
gesucht und diese durch eine leere Zeichenkette ersetzt.

Somit haben wir unser fertiges dictionary erstellt und konnten dies für unsere Abfrage mittels der Datei "Hashing.py" verwenden.
Die Datei ist so aufgebaut, dass im Endeffekt aus jedem Wort im dictionary ein Hashwert gebildet wird und sofern dieser Wert mit dem vorgegeben Wert übereinstimmt, wird das aktuell überprüfte Wort ausgegeben und der nächste Hashwert überprüft.

Nach eigener Messung hat die Überprüfung ~ 0,79s benötigt:



Quellen

Aufgabe 2:
aus dem Skript

Aufgabe 3:
usw.at: http://dev.usw.at/manual/postgres/Orig_DE/wal.html
Wikipedia: https://de.wikipedia.org/wiki/WAL-Prinzip

Aufgabe 4:
https://www4.cs.fau.de/Lehre/SS13/V_VS/Vorlesung/VS.pdf

Aufgabe 5:
Tutorial zum Scrapen:
https://www.youtube.com/watch?v=Zp417bPfvN0

Websiten für das dictionary:
https://wordfind.com/length/2-letter-words/
https://wordfind.com/length/3-letter-words/
https://wordfind.com/length/4-letter-words/
https://wordfind.com/length/5-letter-words/

Tutorial zur “Dictionary-Attack”:
https://www.youtube.com/watch?v=OLHwDiZYF8E