Universität Regensburg

Philosophische Fakultät III
Sprach- , Literatur- und Kulturwissenschaften
Institut für Information und Medien, Sprache und Kultur (I:IMSK)
Lehrstuhl für Informationswissenschaft

Analysieren und Visualisieren mit Python
SS 2016
Leitung: Müller, Manuel Tonio

Magic Collection Manager

Simon Geyer
1755523
Medieninformatik / Informationswissenschaft
4. Semester B.A.
Email: simon.geyer@stud.uni-regensburg.de

Oliver Irlbacher
1652900
Medieninformatik / Informationswissenschaft
6. Semester B.A.
Email: oliver.irlbacher@stud.uni-regensburg.de

Felix Riedl
1711046
Medieninformatik / Informationswissenschaft
5. Semester B.A.
Email: felix1.riedl@stud.uni-regensburg.de

Martin Steinhauer
1712960
Medieninformatik / Informationswissenschaft
5. Semester B.A.
Email: martin.steinhauer@stud.uni-regensburg.de

Simon Geyer
1755523
Medieninformatik / Informationswissenschaft
4. Semester B.A.
Email: simon.geyer@stud.uni-regensburg.de

Abgegeben am 25.07.2016

Inhalt

1 Kurzbeschreibung

Bei dem Magic Collection Manager handelt es sich um einen REST-Service der, mit Hilfe einer Magic Karten API (https://deckbrew.com/api/), es dem Nutzer ermöglicht seine „Magic: The Gathering“ (http://magic.wizards.com) Kartensammlung zu virtualisieren und auf einem Server mit Hilfe einer SQL-Datenbank (https://docs.python.org/2/library/sqlite3.html) zu speichern. Der Service bietet die Möglichkeit nach Magickarten zu suchen, diese zu filtern, einen Benutzer zu erstellen und seine Kollektion zu verwalten.

2 Technische Umsetzung

2.1 Architektur

Die Architektur des REST-Service lässt sich in folgende Punkte unterteilen:

- Data-Provider (DataProvider.py): Stellt die Verbindung zur sqlite3 Datenbank (https://docs.python.org/2/library/sqlite3.html) sicher und verarbeitet sämtliche Datenbankoperationen.
- Rest-Provider (RestProvider.py): Stellt die Verbindung zur deckbrew-API (https://deckbrew.com/api/) her, verarbeitet die Anfragen an die API und gibt das Ergebnis zurück.
- Controller (App.py): Stellt mit Hilfe von flask (http://flask.pocoo.org/) eine REST-Schnittstelle zur Verfügung. Verarbeitet die Anfragen und leitet diese an die entsprechenden Provider weiter.

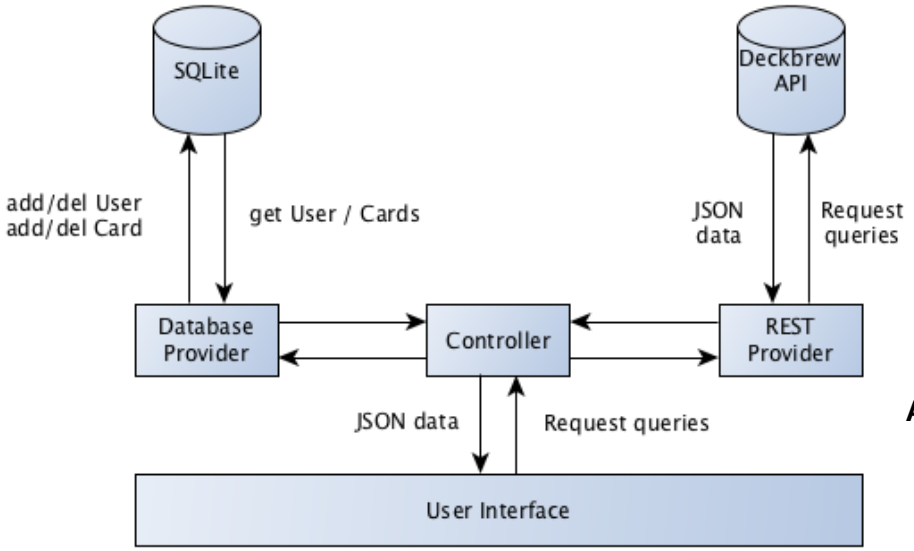

Abbildung 1: Software Architektur

2.2 REST-Routen

Der Magic Collection Manager bietet folgende REST-Routen an:

- GET /find - Liefert alle Karten, entsprechend der angegebenen Filter-Parameter zurück. Es sind folgende Parameter möglich: type, subtype, supertype, name, oracle, set, rarity, color, multicolor, multiverseid, format, status. Da die Parameter an die Deckbrew-API (https://deckbrew.com/api/) weitergeleitet werden, können dort detailiertere Informationen zu den einzelnen Parametertypen entnommen werden.
- GET /users – Liefert alle, in der Datenbank hinterlegten, Benutzer zurück. Wurde hauptsächlich zu Testzwecken verwendet.
- GET /user/{username}/cards – Liefert alle Karten aus der Kollektion eines Benutzers zurück. Optional können auch hier dieselben Filter-Parameter wie in der „GET /find"-Route benutzt werden.
- POST /user – Fügt einen Benutzer zur Datenbank hinzu. Ein „name"-Parameter, der den Benutzernamen repräsentiert muss mitgesendet werden.
- POST /user/{username}/card – Fügt eine Karte zur Kollektion eines Benutzers hinzu. Ein „name"-Parameter, der den Kartennamen repräsentiert muss mitgesendet werden.
- DELETE /user – Löscht einen Benutzer und dessen gespeicherte Karten aus der Datenbank. Ein „name"-Parameter, der den Benutzernamen repräsentiert muss mitgesendet werden.
- DELETE /user/{username}/card/{cardname} – Löscht eine Karte aus der Kollektion des Benutzers.

2.3 Testen der REST-Schnittstelle

Die REST-Schnittstelle wurde mit Postman (https://www.getpostman.com/) und folgenden Anfragen getestet:

- GET /find Request: localhost:5000/find?name=voice&type=creature – Außerdem wurde mit diversen Parametern wie „colors", „rarity" getestet. Liefert eine JSON Liste der entsprechenden Karten zurück.
- GET /user/[Benutzername]/cards Request: localhost:5000/user/[Benutzername]/cards?name=voice&type=creature – Auch hier

wurde diverse Parameter getestet. Liefert eine JSON Liste der entsprechenden Karten zurück.

- GET /users Request: localhost:5000/users – Liefert eine Liste aller Benutzer zurück.
- POST /user Request: localhost:5000/user – Body Parameter name : [Benutername] – Liefert bei einem Erfolg eine 200 Response, bei einem Fehler einen 404 Response zurück.
- POST /user/[Benutzername]/card Request: localhost:5000/user/[Benutername]/card – Parameter name : [Kartenname] z.B. „path to exile“ – Liefert bei einem Erfolg eine 200 Response, bei einem Fehler einen 404 Response zurück.
- DELETE /user/[Benutzername]/card/[Kartenname] Request: localhost:5000/user/[Benutzername]/card/[Kartenname] z.B. „path to exile“ - Liefert bei einem Erfolg eine 200 Response, bei einem Fehler einen 404 Response zurück.
- DELETE /user Request: localhost:5000/user – Body Parameter name : [Benutzername] - Liefert bei einem Erfolg eine 200 Response, bei einem Fehler einen 404 Response zurück.

3 Problemstellungen

Im Zuge der Implementierung hat sich herausgestellt, dass eine Sortierfunktion der zurück gelieferten Karten keine Kernfunktionalität des Magic Collection Managers ist. Da sich das Projekt auf einen REST Service beschränkt und somit die Sortierung der Ergebnisse eher eine Angelegenheit des Benutzerinterfaces darstellt, weshalb auf dessen Umsetzung verzichtet worden ist. Im Zuge der damit frei gewordenen Ressourcen wurde das Benutzermanagment erweitert und umfasst somit, das anlegen und löschen einzelner Benutzer. Auch wurde die Codebasis überarbeitet um diese Effizienter und Schlanker zu gestaltet.

Darüber hinaus, wäre der nächste Schritt eine Benutzerauthentifizierung zu implementieren um die Sicherheit der am Server gespeicherten Daten zu gewährleisten.