

# README

---

## Quickstart

Wir verwenden Docker, um die Umgebung für unsere API bereitzustellen. Voraussetzung dafür ist das Installieren von Docker, welches je nach Betriebssystem unterschiedlich gemacht werden muss. Die Dokumentation findet sich hier: <https://docs.docker.com/install/>.

Ist Docker installiert und gestartet, können in einen Commandline Fenster im Projektverzeichnis folgende Befehle ausgeführt werden:

```
shell> docker build -t wdb_mc1_web -f ./dockerfile .
shell> docker-compose up -d
```

## Webserver

Für das Aufsetzen des Webserver muss eine Datenbank installiert werden. Wir verwenden MariaDB Server, welches von hier bezogen werden kann: <https://mariadb.org/download/>. Nach abgeschlossener Installation kann die Datenbank folgendermassen aufgesetzt werden:

```
shell> mysql

MariaDB [(none)]> CREATE DATABASE wdb;
MariaDB [(none)]> CREATE USER 'api'@'%' IDENTIFIED BY 'some_password';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON wdb.* TO 'api'@'%';
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> (CTRL + C)

shell> mysql wdb < ./src/sql/structure.sql
```

Die Datenbank "wdb" ist nun aufgesetzt und kann mit Daten befüllt werden.

Als Nächstes müssen die Zugangsdaten der Datenbank dem Webserver mitgeteilt werden. Dazu die Datei 'credentials.json.sample' nach 'credentials.json' kopieren und die Felder "host, port, user, password, db" auf die Zugangsdaten des Datenbankservers anpassen.

Als Nächstes muss in den Docker Container gewechselt werden. Im Container muss folgender Befehl ausgeführt werden, um die Python Abhängigkeiten zu installieren und die API zu starten:

```
shell> pipenv run python ./src/main.py
```

Die API ist nun unter <http://localhost:5000> verfügbar.

## Test

Um die API zu testen muss man in den Docker Container wechseln und dort in der Shell folgenden Befehl ausführen:

```
shell> pipenv run python src/test.py
```

Waren alle Tests erfolgreich, wird folgende Meldung ausgegeben:

```
-----  
Ran (n) tests in (n)s  
  
OK
```

## Branching System

In diesem Projekt arbeiten wir mit dem "Gitflow Workflow". Dabei wird pro Feature lokal ein Feature-Branch erstellt, nach Beedigung lokal mit dem "dev" Branch gemerged und nach "origin/dev" gepusht. Für die Auslieferung der Software müssen gemäss Kapitel "Test" alle Tests erfolgreich durchlaufen werden. Erst dann darf der Fortschritt in den "main" Branch gemerged werden. Als Tag wird der Versionscode im Branch "main" gekennzeichnet.

Weitere Informationen: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

## Übersicht Calls

### User

Method	Route	Description
GET	/user	Get all users
GET	/user/<user_id>	Get a specific user
GET	/user/<user_id>/ratings	Get all ratings of a specific user
GET	/user/<user_id>/ratings/movies	Get all movies which a specific user has rated
POST	/user	Create a user
DELETE	/user/<user_id>	Delete a user
PUT	/user/<user_id>	Update a user

### Movie

Method	Route	Description
--------	-------	-------------

Method	Route	Description
GET	/movie	Get all movies. Possible params: ?limit_n= only returns top n movies by vote average. ?title= only returns movies with titles containing the given string
GET	/movie/<movie_id>	Get a specific movie
POST	/movie	Create a movie
DELETE	/movie/<movie_id>	Delete a movie
PUT	/movie/<movie_id>	Update a movie

## Ratings

Method	Route	Description
POST	/rating	Create a rating
PUT	/rating/<rating_id>	Update a rating
DELETE	/rating/<rating_id>	Delete a rating

## Sample Requests



Wurde die API gemäss Quickstart, Webserver gestartet, können Abfragen gemäss eingestellter URL im Connection File gemacht werden. Die Software Postman bietet sich dazu an. Als Beispiel wurden folgende Abfragen getestet:

GET /users

wdb / /user

Save

...



GET

127.0.0.1:5000/user

Send

ParamsAuthHeaders (6)BodyPre-req. TestsSettingsCookies



Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

200 OK528 ms280 BSave Response

PrettyRawPreviewVisualizeJSON



```
1  []
2  [
3    1,
4    "Hanspeter Peterhans"
5  ],
6  [
7    2,
8    "Werner Würschtli"
9  ],
10 [
11  3,
12  "Chantal Schläppi"
13 ]
14 ]
```

GET /ratings

GET127.0.0.1:5000/user/1/ratings

Send

ParamsAuthHeaders (6)BodyPre-req. TestsSettings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

200 OK470 ms220 BSave Response

PrettyRawPreviewVisualizeJSON

1[

2[

31,

41,

593.0

6],

7[

81,

94,

1067.0

11]

12]

GET /does\_not\_exist

GET

127.0.0.1:5000/does\_not\_exist

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

404 NOT FOUND 1277 ms 285 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

"message": "Record not found: [http://127.0.0.1:5000/does\\_not\\_exist](http://127.0.0.1:5000/does_not_exist),

"status": 404

# Report

Sie finden unseren Bericht in [./docs/Report.pdf](#).