AST-Assignment #3

Prof: Nico Hochgeschwender
T.A.: Samuel Parra: samuel.parra@smail.inf.h-brs.de

June 2020

So far we have designed and programmed a Facebook copycat (of sorts) using the concepts of object oriented programming. We have used data structures like dictionaries and lists to store the data of the users, posts, and groups. The reality is that these systems have to manage a lot more data, and to store it they use databases, which can be classified into relational and NoSQL databases.

In this assignment we are changing our system from the object-oriented paradigm to the graph paradigm! We are going to use a type of NoSQL database called a graph database to store the data of our users, their friends, their posts, and their groups.

1 Graph databases

Take your time to read this article: `https://www.compose.com/articles/introduction-to-graph-databases/`, it will gives you an introduction to graph databases as well as the differences with relational databases.

2 Set up your PC

In order to work with graph databases you'll need to install a graph database platform, and a driver that allows you to connect to it in python.

- The graph platform: We will be using Neo4j. you can download it from `https://neo4j.com/download/`. The file is an `.AppImage` file, to run it you need to make it executable: `$ chmod a+x name_of_file.AppImage`, to run it do: `$ $./name_of_file.AppImage`
- The driver: You will find in this link a list of available drivers with a short example, we recommend you use `py2neo` `https://neo4j.com/developer/python/`.

You can try to create a hello world program to check if everything is in order. You may play around with this example (if you are using `py2neo`) `https://github.com/neo4j-examples/movies-python-py2neo`

You will need to create a local graph database from neo4j, call it `Facebook` with the password set to `facebook`.

3 Set up your Facebook system

In order to use the graph database your system might undergo some refactoring depending on your components and their relationships. The idea is to store the instances (such as users) as nodes. Since this is the objective you must look at your model and think whether or not your current design will allow you to follow this idea. If necessary you will need to adapt your model and refactor your code.

You will also need to create a `database.py` that contains all methods related to the graph database. For example, it can consist of:

`setup()` which connects to the database

`populate()` which populates the database with some elements (preferably, it only runs when a flag is true; i.e. `--populate=True`).

`perform_query(query)` which perform some query and returns the result.

But it can also be more specific:

`search_user(name)`

`see_friends(user)`

In order to get information from the database you will need to create queries, in neo4j queries are built using the Cypher language. You can learn the basics here: `https://neo4j.com/developer/cypher-basics-i/`. You will need to integrate these queries into your system, meaning that you should be able to look up:

- A user by name.
- The friends of a user.
- The post of a user.
- The people that are members of a certain group.
- The post of a certain group.

etc...

If everything goes right, your system should persistently store your Facebook session!

4 assignment submission

- This assignment can be submitted by individuals or pairs only.
- The deadline for this assignment is **22.06.2020**. Please *do not* make more commits on your GitHub repository after the deadline.
- All models should be placed on the `./documentation` folder.
- All the code should be place on the `./code` folder.
- Submission is made through GitHub. Please, try to commit your progress constantly as commits after the deadline won't be taken into consideration.

5 Notes

- We expect a command line interface (CLI) to use your program, not a GUI.
- use the **default address, and username** for the database with the password "facebook". Please confirm that the default is user:neo4j.
- You should be able to explain your code and your design decisions.
- Update `requirements.txt` before submitting.
- You can contact the T.A. for any questions you may have with the email address provided at the beginning of this document. Please, write [AST-Assignment #] in the subject and your name(s) in the body of the email.

6 Resources

- Code Style: `https://docs.python-guide.org/writing/style/`
- py2neo documentation: `https://py2neo.org/2.0/intro.html`