Rishi Lalbahadur

IT 490

May 11, 2020

OpenFECData.com - Backend Documentation

Summary and Flow

The backend machine serves as a storage of user information and for communication of user information, serving the user, to the front end Apache server. The backend consists of a MySQL server with a database to hold all user information. Three Python scripts are used for communication between the backend and the frontend via RabbitMQ message broker.

The scripts will consume data from the frontend server via RabbitMQ, and the backend server will execute queries in the database. Each piece of data will pass through a certain queue in the RabbitMQ server. The scripts control the communication of data.

About MySQL database

The MySQL database stores rows each containing columns for a user ID, email address, password, a first name, a last name, and API history. The row of a user ID, email address, password, first name, and last name are created upon user registration where the ID number is automatically incremented by the database server upon registration. The password column will store user passwords as hashed passwords sent by the frontend server based on the front end's hashing algorithm. The backend engineer should be able to see a hashed password and will not know how to decode it. The frontend server will handle the decoding of passwords upon logging in where a user inputs their password as normal and sends a hashed input to match the hashed password stored in the database table. Lastly, an already inputted row in the database table will be referred to for user login.

Login Script

The Python login script contains code of an instance of a RabbitMQ connection containing the RabbitMQ server's IP address and credentials. The script will connect the backend server to the login queue in the RabbitMQ server. The frontend will send a JSON containing the user email address and a hashed password where the frontend will hash the user's input password. The backend will consume the JSON. The script will open the MySQL server via Python MySQL Connector where the code contains MySQL server credentials. The script

will then parse the JSON into a list, so the input can be placed as variables into a MySQL query. The script will execute a query needed for logging in. The query will be to select the ID number that matches both the email address and password (in hashed version) inputted. The script will send the word “true” to the front end if there is a match in the database, and the word “true” will allow the front end to log the user in. The script will send the word “false” if there is no match.

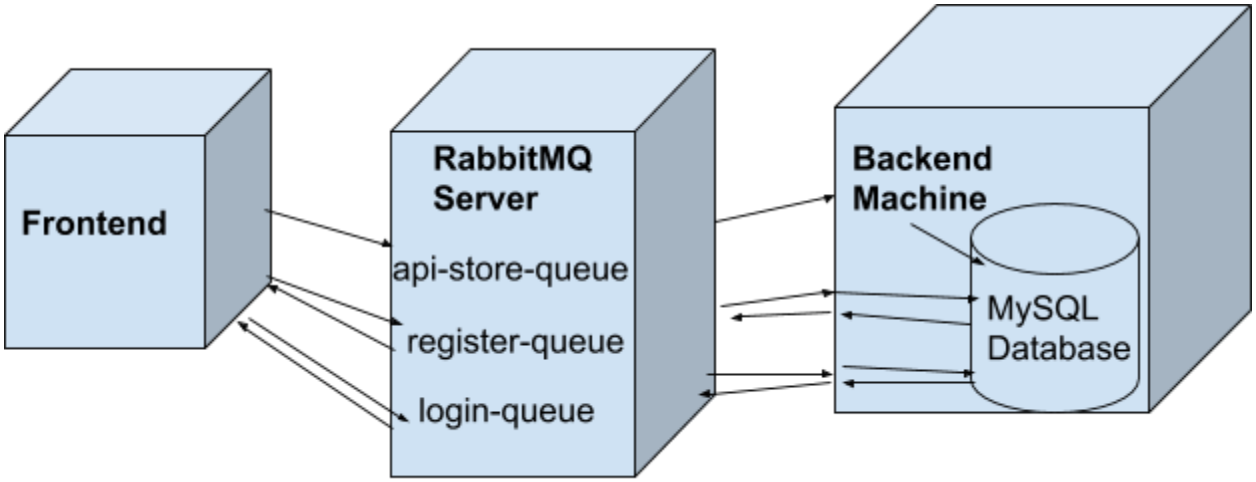
Register Script

The Python registration script contains code of an instance of a RabbitMQ connection containing the RabbitMQ server’s IP address and credentials. The script will connect the backend server to the registration queue in the RabbitMQ server. The frontend will send a JSON containing the user email address, a hashed password where the front end will hash the user’s input password, a first name, and a last name. The backend will consume the JSON. The script will open the MySQL server via Python MySQL Connector where the code contains MySQL server credentials. The script will then parse the JSON into a list, so the input can be placed as variables into a MySQL query. The script will execute a query needed for registration. The query will be to insert a new row in the designated table containing an auto incremented ID number, the email address, password (in hashed version), first name, last name, and a blank history column. The script will send the word “true” to the front end if the insertion is true. The script will send the word “false” if the insertion does not work.

API History Script

The Python API history script contains code of an instance of a RabbitMQ connection containing the RabbitMQ server’s IP address and credentials. The script will connect the backend server to the api history queue in the RabbitMQ server. The frontend will send a JSON containing an email address of the logged in user and a JSON string of the OpenFEC API query data. The backend will consume the JSON. The script will open the MySQL server via Python MySQL Connector where the code contains MySQL server credentials. The script will then parse the JSON into a list, so the input can be placed as variables into a MySQL query. The script will execute a query needed for storing the API history. The query will be to insert the API JSON string in the history column of the logged in email address. The backend is not required to send a response to the storage of the API JSON string, so the backend will not send a response to the front end.

Flow Chart


Frontend
RabbitMQ
Server
api-store-queue
register-queue
login-queue
Backend
Machine
MySQL
Database