BD-Project

Project for the DataBase subject.

Dependencies

To run the project correclty, some techonologies is required. So that follow simple documentation to help you.

Technologies Used

Programming Languages

- Python
- SQL and PL/pgSQL

Database Management System

- PostgreSQL

Python Libraries

- Flask
- Psycopg2
- werkzeug.security
- flask_jwt_extended

Other Technologies

- Onda
- Postman

Tips to Dependencies

Before you do something, check this shell commands and verify what you need to install.

```
# check if already installed:
flask --version

sudo pip3 install flask

# check if it's correctly installed:
python3
>>> import flask
>>>                  # it's all right
>>> from werkzeug.security import generate_password_hash, check_password_hash
>>>                  # it's all right
```

Tools Installation

If you don't have some of the listed dependencies installed, here you can see how to do it.

Python and libraries

```
sudo apt install python3 python3-pip      # Install python and pip (alow to install other libraries)

sudo pip install flask
sudo pip install werkzeug
sudo apt install libpq-dev                # In case you haven't this library, you need to install (assume gcc is installed in OS by d
sudo pip install psycopg2
pip install flask-jwt-extended
```

pSQL

```
# Create the file repository configuration:
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

# Import the repository signing key:
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

# Update the package lists:
sudo apt-get update

# Install the latest version of PostgreSQL.
# If you want a specific version, use 'postgresql-12' or similar instead of 'postgresql':
sudo apt-get -y install postgresql
```

Postman

```
sudo apt install postman
```

`Note`: If you want to know more, go to the respective documentation at:

- Python (https://www.python.org/downloads/)
- Pip (https://pip.pypa.io/en/stable/installing/)
- PostgreSQL (https://www.postgresql.org/download/)
- Postman (https://postman.com/)

DataBase Setup

To setup all configs of the database, you need to access your postgreSQL DMBS by `psql` or `pgadmin4`. We used the `psql client` with the follow command:

```
# Use the default credentials:
# Username: postgres
# Password: postgres

psql -h localhost -p 5432 -U postgres
```

After acess, let's create a `database` where is stored all tables and work about `database design` and connect to him:

```
CREATE DATABASE dbshop;


# if you run:
\l
# it should list all available databases
# a similar result:
#                                 List of databases
#   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
#-----------+----------+----------+-------------+-------------+-----------------------
# dbshop    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
# postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
# template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
#           |          |          |             |             | postgres=CTc/postgres
# template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
#           |          |          |             |             | postgres=CTc/postgres
#(4 rows)
```

Now the `database` created, everything is ready to add tables and data

```
\c dbshop           # connect to dbase
\i schema.sql       # create the tables schemas
\i insert.sql       # add data
\i trigger.sql      # create all triggers

\i drop_tables.sql  # just in case if you want to drop all tables
```

`Note`: In case you use `pgadmin4` insted of the terminal, bellow follow the links that will help you to build everything.

- Create database (https://www.pgadmin.org/docs/pgadmin4/development/database_dialog.html)
- Run a script (https://linuxhint.com/run-sql-file-postgresql/)

User Manual

before starting -> run the script test.py to encript the first users passwords

to run the project -> python3 main.py

Users Loggin

Description: User authentication with username and password. It was created a superadmin to register other admins and sellers.

URL : `/api/login` **Method** : `PUT`

```
# Input value
# In this case represent a super admin
{
   "username": "SuperAdmin",
   "password": "SuperAdmin"
}
# Return value
{
   "status": 200,
   "token": "TOKEN YOU WILL USE AS INPUT"
}
```

Users Registration

Description: Only admin can registe other admins and sellers. Everyone can register it self as buyer.

URL : `/api/register`

Method : `POST`

Register admin/seller

```
# Input value
{
   "username": "Put a name",
   "nif": 0,
   "email": "Put an email",
   "adress": "Put an address",
   "password": "Put a pass",
   "token": "Token received in login",              # this token was passed in the loggin
   "user_type": "administrator/seller"
}
# Return value
{
    "message": "Regist completed",
    "result": user_id,
    "status": 200
}
```

Register buyer

```
# Input value
{
   "username": "Put a name",
   "nif": 0,
   "email": "Put an email",
   "adress": "Put an address",
   "password": "Put a pass",
}
# Return value
{
    "message": "Regist completed",
    "result": user_id,
    "status": 200
}
```

Product Creation

Description: Only sellers can create products.

URL : `/api/product/add`

Method : `POST`

```
# Input value
{
   "type":"Put a type",
   "description":"Put a description",
   "height":0,
   "weight":0,
   "colour":"Put a color",
   "stock":0,
   "price":0,
   "token": "Token received in login",              # this token was passed in the loggin
}
# Return value
{
    "message": "Product added successfully",
    "results": id_prod,
    "status": 200
}
```

Update product details

Description: Only sellers can update products.

URL : `/api/product/{prod_id}` **Method** : `PUT`

```
{
# Input value
   "description": "new description",
   "height": "new height",
   "weight": "new weight",
   "colour": "new colour",
   "price": "new price",
   "token": "Token received in login",              # this token was passed in the loggin
}
# Return value
{
    "message": "Product values updated",
    "status": 200
}
```

Obtain the product details

Description: All users can see the product details. Obtain details of all versions of a product.

URL : `/api/product/{product_id}`

Method : `GET`

```
   # No input

   # Return value
   {
    "colour": [
        "colour1",
        ...
    ],
    "comments": [
        "comment1",
        ...
    ],
    "description": [
        "description1",
        ...
    ],
    "height": [
        0,
        ...
    ],
    "price": [
        0,
        ...
    ],
    "rating": "0",
    "status": 200,
    "stock": [
        0,
        ...
    ],
    "weight": [
        0,
        ...
    ]
}
```

Do a order

Description: Only buyers can make a order.

URL : `/api/order` **Method** : `PUT`

```
# Input value
{

   "cart": [[Prod1, Quantity1], [Prod2, Quantity2], ...],
   "token": "Token received in login",              # this token was passed in the loggin

}
# Return value
{
   "message": "ORDER COMPLETED",
   "result": order_id,                     #order id
   "status": 200
}
```

Rate and comment a product

Description: A buyer can rate and comment about a product that he/she bought.

URL : `/api/rating/{product_id}` **Method** : `POST`

```
# Input value
{
    "rating": 0,
    "comment":"Put a comment",
    "token": "Token received in login",              # this token was passed in the loggin
}

# Return value
{
    "status": 200
}
```

Comment/Question about a product in users forum

Description: All user can do a question or a comment about a product.

Leave your question

URL : `/api/questions/{prod_id}`

Method : `POST`

```
# Input values
{
   "question": "Put a question",
   "token": "Token received in login",              # this token was passed in the loggin
}
#return values
{
    "message": "Question done",
    "result": comment_id,
    "status": 200
}
```

Answer a question

URL : `/api/questions/{prod_id}/{comment_id}`

Method : `POST`

```
# Input values
{
   "question": "Put a question",
   "token": "Token received in login",              # this token was passed in the loggin
}
# Return values
{
    "message": "Answered successfully",
    "result": comment_id,
    "status": 200
}
```

Sales statistics of the last 12 months

Description: Obtain the sales statistics of the last 12 months per month, including the number of sales and total value.

URL : `/api/rating/{product_id}` **Method** : `POST`

```
# No input

# Return value
{
    "results": [
        {
            "month": "07-2021",
            "orders": 1,
            "total_value": "1622"
        },
        {
            "month": "09-2021",
            "orders": 2,
            "total_value": "2144"
        },
        {
            "month": "03-2022",
            "orders": 1,
            "total_value": "2784"
        },
        {
            "month": "01-2022",
            "orders": 1,
            "total_value": "4800"
        },
        {
            "month": "05-2022",
            "orders": 5,
            "total_value": "14717"
        }
    ],
    "status": 200
}
```

Co-workers

João Moreira - joaomoreira@student.dei.uc.pt https://github.com/JoaoESmoreira (https://github.com/JoaoESmoreira)

Tomás Pinto - tomaspinto@student.dei.uc.pt https://github.com/TC121121 (https://github.com/TC121121)