Технології серверного програмного забезпечення

Лабораторна робота 4

Аутентифікація

Мета роботи: Додати можливість аутентифікації в проект

Завдання:

- Навчитись додавати можливість аутентифікації за домопогою JWT
- Протестувати роботу застосунку з аутентифікацією

Практичне завдання:

Додати ендпоінти реєстрації та логіну в застосунок. Захистити всі інші ендпоінти таким чином щоб до них мали доступ тільки аутентифіковані користувачі(які надали коректний токен). Створити flow в postman який зможе відтворити реєстрацію, логін, та поддальше використання отриманого токену для доступу до інших ендпоінтів.

Методичні рекомендації:

1. Не забудьте створити тег для позначення закінчення другої лабораторної.
2. Встановити бібліотеку flask-jwt-extended за допомогою команди pip install flask-jwt-extended
3. Оновити файл requirements.txt за допомогою команди - pip freeze > requirements.txt
4. Згенерувати секретний ключ для JWT в консолі python за допомогою команд:

```
Python Console
>>> import secrets
>>> secrets.SystemRandom().getrandbits(128)
```

5. Після цього скопіювати виведене значення та присвоїти його змінній середовища `JWT_SECRET_KEY` за допомогою команди

   ➜export JWT_SECRET_KEY="your secret key from previous step"
6. Тепер в функції-фабриці нашого застосунку ми повинні додати такі рядки:

```
app.config["JWT_SECRET_KEY"] = os.getenv("JWT_SECRET_KEY")
db.init_app(app)

api = Api(app)

jwt = JWTManager(app)
```

7. Встановити бібліотеку passlib за допомогою команди pip install passlib
8. Оновити файл requirements.txt за допомогою команди - pip freeze > requirements.txt
9. Реалізуйте ендпоінти реєстрації(замініть ним ендпоінт створення користувача) та ендпоінт логіну
   для реєстрації:
   на реєстрації користувач має надати пароль, і зберегти пароль Ви повинні таким чином:
   ```
   user = UserModel(
                       username=user_data["username"],
            password=pbkdf2_sha256.hash(user_data["password"]),
           )
   ```
   на ендпоінті логіну потрібно перевірити пароль, якщо він коректний то видати токен користувачу в відповіді
   ```
   if user and pbkdf2_sha256.verify(user_data["password"],
   user.password):
                  access_token = create_access_token(identity=user.id)
   ```
   де `create_access_token` це функція бібліотеки `flask_jwt_extended`
10. Ендпоінти, які не мають бути доступні неавторизованим користувачам потрібно захистити декоратором `@jwt_required()`
11. В застосунку потрібно також додати JWTManager та обробники помилок, зв’язаних з jwt таким чином:

```
app.config["JWT_SECRET_KEY"] = "jose"
jwt = JWTManager(app)

@jwt.expired_token_loader
def expired_token_callback(jwt_header, jwt_payload):
    return (
        jsonify({"message": "The token has expired.", "error":
"token_expired"}),
        401,
    )

@jwt.invalid_token_loader
```

```
def invalid_token_callback(error):
    return (
        jsonify(
            {"message": "Signature verification failed.", "error":
"invalid_token"}
        ),
        401,
    )

@jwt.unauthorized_loader
def missing_token_callback(error):
    return (
        jsonify(
            {
                "description": "Request does not contain an access
token.",
                "error": "authorization_required",
            }
        ),
        401,
    )
```

Після виконання створіть тег для позначення закінчення лабораторної

Критерії оцінювання:

Розподіл балів за лабораторну 4:
5 балів - реалізація ендпоінтів для реєстрації та логіну користувача
9 балів - коректна робота нових ендпоінтів
5 балів - нові ендпоінти відображені в postman flow
1 бал - наявність правильного .gitignore