 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №4**

з дисципліни «ВЕБ-технології та ВЕБ-дизайн»

# на тему: «Робота з даними»

**Виконав:**

студент гр. БС-83

Погребенко В.О.

**Перевірили:**

ас. Давидько О.Б.

ас. Матвійчук О.В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2020

**Мета роботи:**

Ознайомитися із створенням API та роботою з даними на основі обраних раніше фреймворків.

**Завдання:**

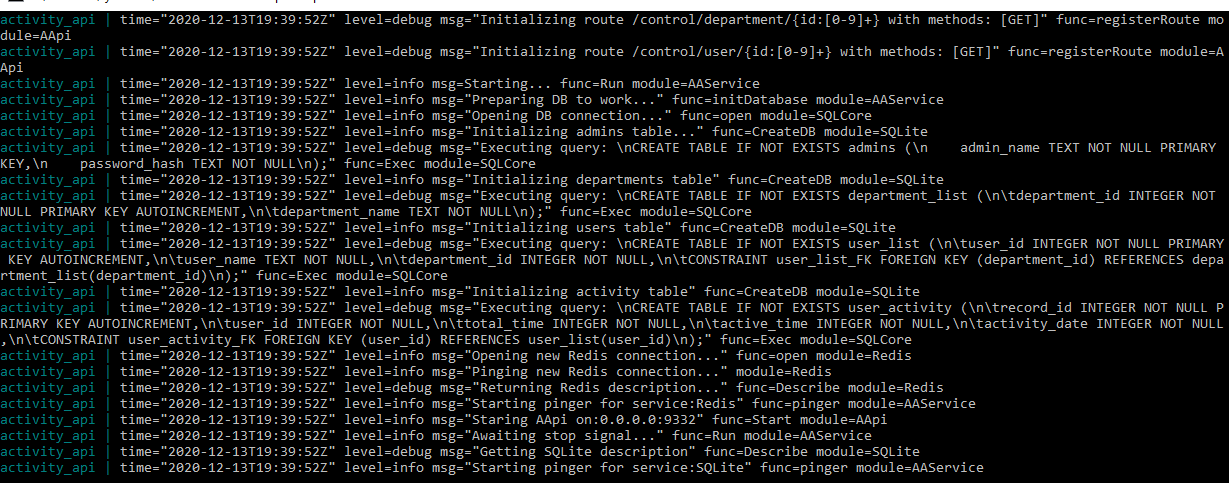
Створити простий додаток, що реалізує частину функціоналу, згідно архітектури, що була розроблена у попередній роботі. Вимоги до реалізації:

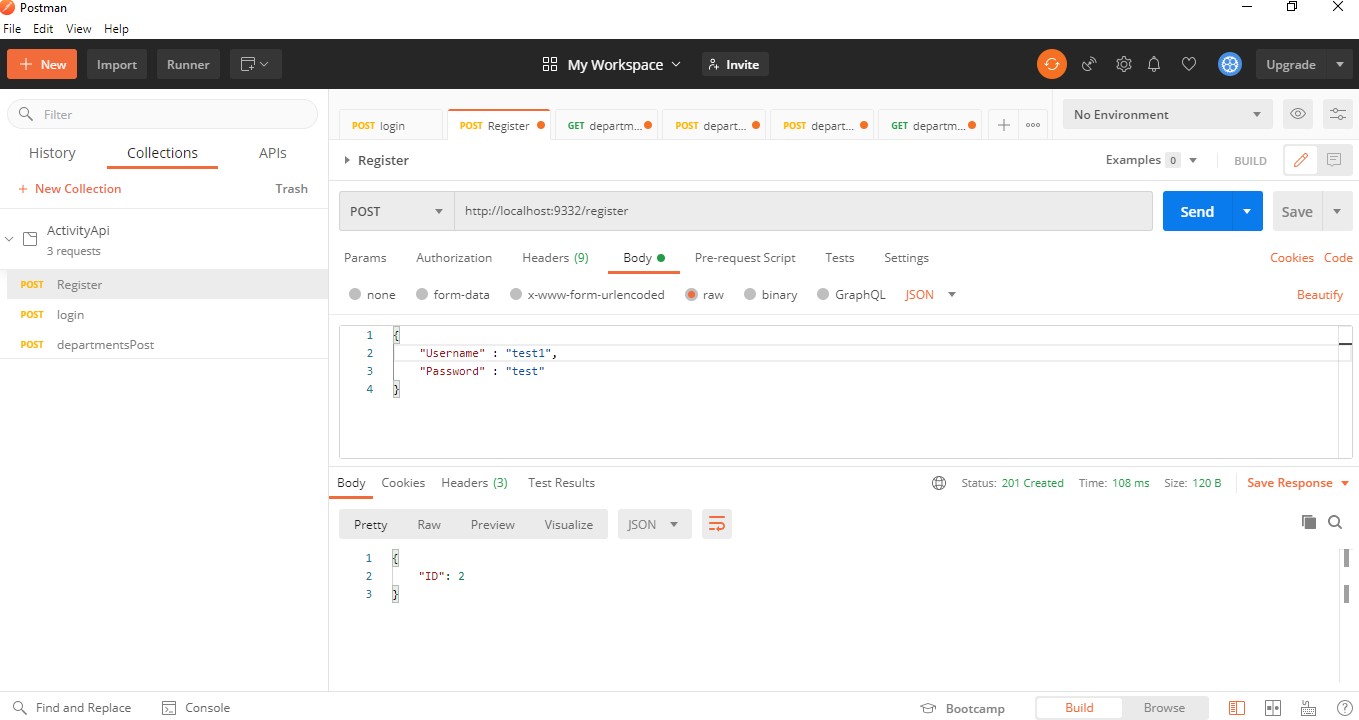
1. Реалізувати аутентифікацію та авторизацію
2. Створити точки доступу для базових CRUD операцій.
3. У випадку моноліту мають бути реалізовані операції для двох модулів та підключений Redis як кеш-сервер
4. Для мікросервісів мають бути реалізовані операції для одного з них + авторизація, як окремий сервіс, та API Gateway.
5. Дозволяється використання будь-яких фреймворків на обраній мові.

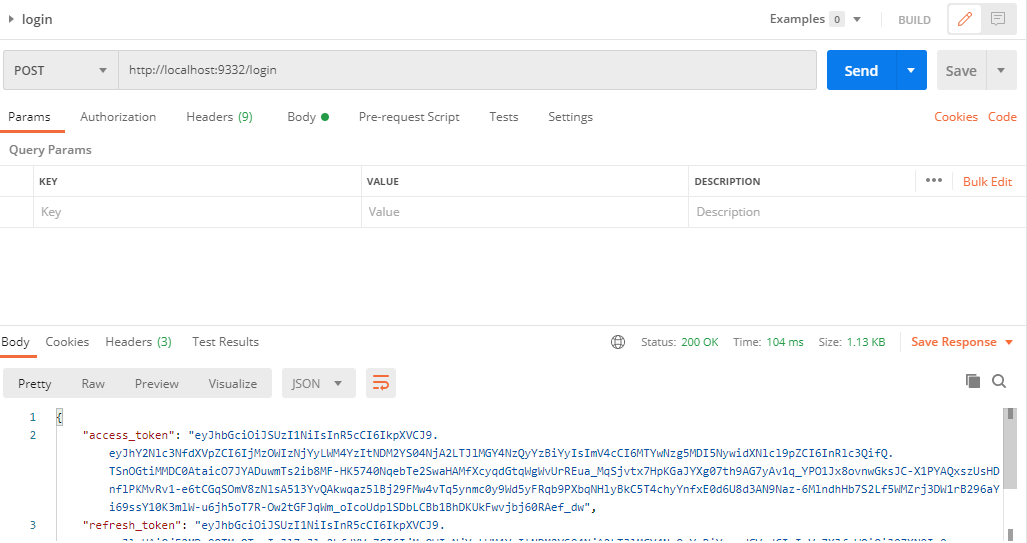
**Код за посиланням:**

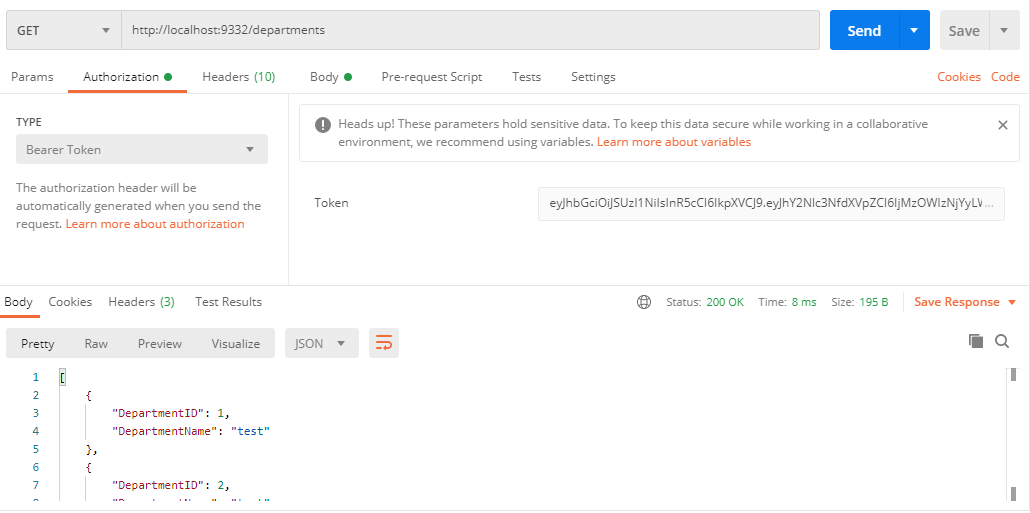
https://github.com/PogrebenkoBS81/Web\_course3/tree/master/lab4-5/activity\_api

Приклади роботи (запуск + реестрація, логін, додавання та отримання департаментів за допомогою Postman):

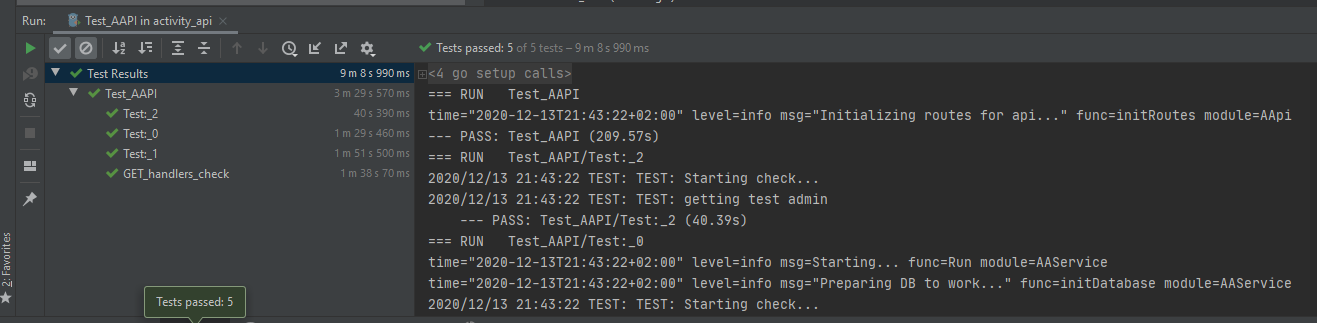








Пройдені базові функціональні тести всіх хендлерів:



**Контрольні питання:**

1. ***Різниця між аутентифікацією та авторизацією***

**аутентифікацією** *- це процес перевірки, хто є користувачем*, використовується для встановлення особи користувача, який намагається використовувати систему. Виконується завдяки перевірки унікальної інформації, наприклад, паролю та імені користувача.

**авторизація** *- це процес перевірки того, до чого користувач має доступ*. Тобто перевірка чи може користувач виконувати певні дії.

1. ***Що таке Redis?***

Redis (Remote Dictionary Server ) – БД з відкритим source кодом, що використовується як база даних, кеш-пам’ятки та message broker. Redis надає такі структури даних, як рядки, хеші, списки, набори, відсортовані набори із запитами діапазонів, растровими зображеннями, гіперлогічними журналами, геопросторовими індексами та потоками.

Зазвичай використовується для зменшення навантаження на БД та програмний додаток завдяки кешуванню часто запитуємих данних.

1. ***Алгоритми витіснення даних із кеша.***

* Алгоритм Беладі - відкидати з кешу ту інформацію, яка не знадобиться в майбутньому найдовше. Так як в загальному випадку неможливо передбачити, коли саме в наступний раз буде потрібно саме ця інформація, то на практиці (знову ж таки, в загальному випадку) подібна реалізація неможлива.
* LRU (Last Recently Used) - витісняється та інформація, що невикористовується найдовше. Цей алгоритм вимагає відстеження того, що і коли використовувалося, що може виявитися досить накладно, особливо якщо потрібно проводити додаткову перевірку, щоб в цьому переконатися
* MRU (Most Recently Used) - в першу чергу витісняється останній використаний елемент. «Коли файл періодично сканується по циклічної схемою, MRU - найкращий алгоритм витіснення».

Крім того: Псевдо-LRU, сегментований LRU, etc

1. ***HTTP методи для CRUD операцій.***

**CRUD** – *Create, Reaed, Update, Delete*.

**Методи**:

* GET – запит
* POST – створення
* PUT - оновлення
* DELETE - втдалення

1. ***Що таке ORM?***

**ORM (Object-relational mapping) –** технологія, що дозволяє перетворювати несумісні типи моделей в ООП. Використовується для розповсюдження процесу збереження об'єктів у реляційній базі даних та їх виведення.

Говорячи про ORM, більшість людей мають на увазі бібліотеку, що реалізує **Object-relational mapping**.

Бібліотека ORM - написана на певній мові бібліотека, яка інкапсулює код, необхідний для обробки даних, наприклад, код без буде ORM виглядати як:

book\_list = new List();

sql = "SELECT book FROM library WHERE author = 'Linus'";

data = query(sql); // I over simplify ...

while (row = data.next())

{

book = new Book();

book.setAuthor(row.get('author');

book\_list.add(book);

}

А з використанням ORM:

book\_list = BookTable.query(author="Linus");

1. ***Anemic та Rich моделі опису сутностей в ООП. Переваги та недоліки кожної з моделей.***

І **Anemic Domain Model** і **Rich Domain Model -** концепції побудови логічної архітектури.

**Anemic Domain Model**,- концепція, в якій об'єкти предметної області містять мало або взагалі не містять бізнес-логіки. Її переваги в тому, що *вона проста до побудови*, і *більш проста для прогнозування та керування*. Проте її недоліки в тому, що тоді логіку *неможливо реалізувати у вигляді дійсного ООП*, *вона порушує виразність моделі та потрібен окремий бізнес – рівень для зберігання логіки.*

**Rich Domain Model –** концепція, в якій бізнес-об'єкти містять бізнес-логіку (або бізнес-логіка містить стан бізнес-об'єктів). Основні переваги в *зручному використанні об’ектів та їх методів обробки* та в *інкапсуляції* (взаємодія через інтерфейс) . Недоліки у більш складній реалізації та біль складному керуванні БД.