 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №3**

з дисципліни «ВЕБ-технології та ВЕБ-дизайн»

# на тему: «Архітектурна постановка»

**Виконав:**

студент гр. БС-83

Погребенко В.О.

**Перевірили:**

ас. Давидько О.Б.

ас. Матвійчук О.В.

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

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

(підпис викладача)

Київ-2020

**Завдання:**

1. Розробити архітектурну постановку для веб-додатку. Тематику веб-додатку обрати самостійно.
2. Звіт з роботи має містити такі елементи:
   1. Тематика веб-додатку
   2. Базові вимоги до додатку: функціональні вимоги, запланована кількість користувачів, можливі downtime, інтеграції з іншими сервісами, доступність для людей з обмеженими можливостями, безпека, вимоги до тестування.
   3. Опис високорівневої архітектури. Має бути обґрунтовано вибір моноліту, SOA або мікросервісної архітектури. Має бути представлена діаграма компонентів та їх взаємодії.
   4. Постановка має містити опис низькорівневої архітектури для одного із сервісів (у випадку мікросервісів) або хоча б двох модулів у моноліті. Необхідно розробити UML діаграми класів. У випадку моноліту - діаграма взаємодії модулів.
3. У наступних роботах передбачається реалізація створеної постановки.

**Тематика веб-додатку**

Тематика – моніторинг активності працівників і відділів певного підприємства, за допомогою якого можна буде оцінити продуктивність та активність роботи.

Приблизний вигляд БД:

UserList(UserID, UserName, DepartmentID)

DepartmentList(DepartmentID, DepartmentName)

UserActivity(UserID, Day, TotalTime, ActiveTime)

**Базові вимоги до додатку:**

Функціональні вимоги:

* + 1. Можливість отримати список відділів
    2. Можливість отримати користувачів по відділам
    3. Можливість редактувати список відділв (додавати\видаляти\редактувати)
    4. Можливість редактувати список користувачів (додавати\видаляти\редактувати)
    5. Можливість отримати активність по користувачам за вказанний період
    6. Можливість отримати активність по відділам за вказанний період

Нефункціональні вимоги:

*Кількість користувачів*: обмежена «залізом», може бути визначена більш точно тільки після проведення тестів сервісу.

*Downtime*: зазвичай, під час апдейтів.

*Інтеграції з іншими сервісами*: Не передбачено. При необхідності, взаємодія з іншими сервісами буде проводитися через API.

*Доступність для людей з обмеженими можливостями*: не розроблюється. Можливо буде підтримано у фронтєнді.

*Безпека*: авторизація, зберігання паролів у вигляді хешу та використання JWT токенів.

*Вимоги до тестування*: Вимоги:

- перевірити доступ до хендлерів без авторизації

- перевірити максимальну кількість користувачів

- базова перевірка всіх хендлерів

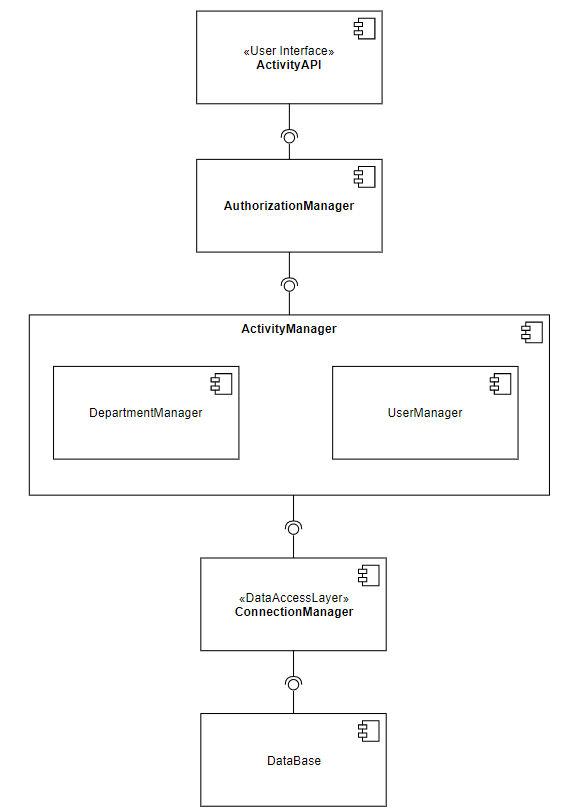
- etc…

**Опис високорівневої архітектури:**

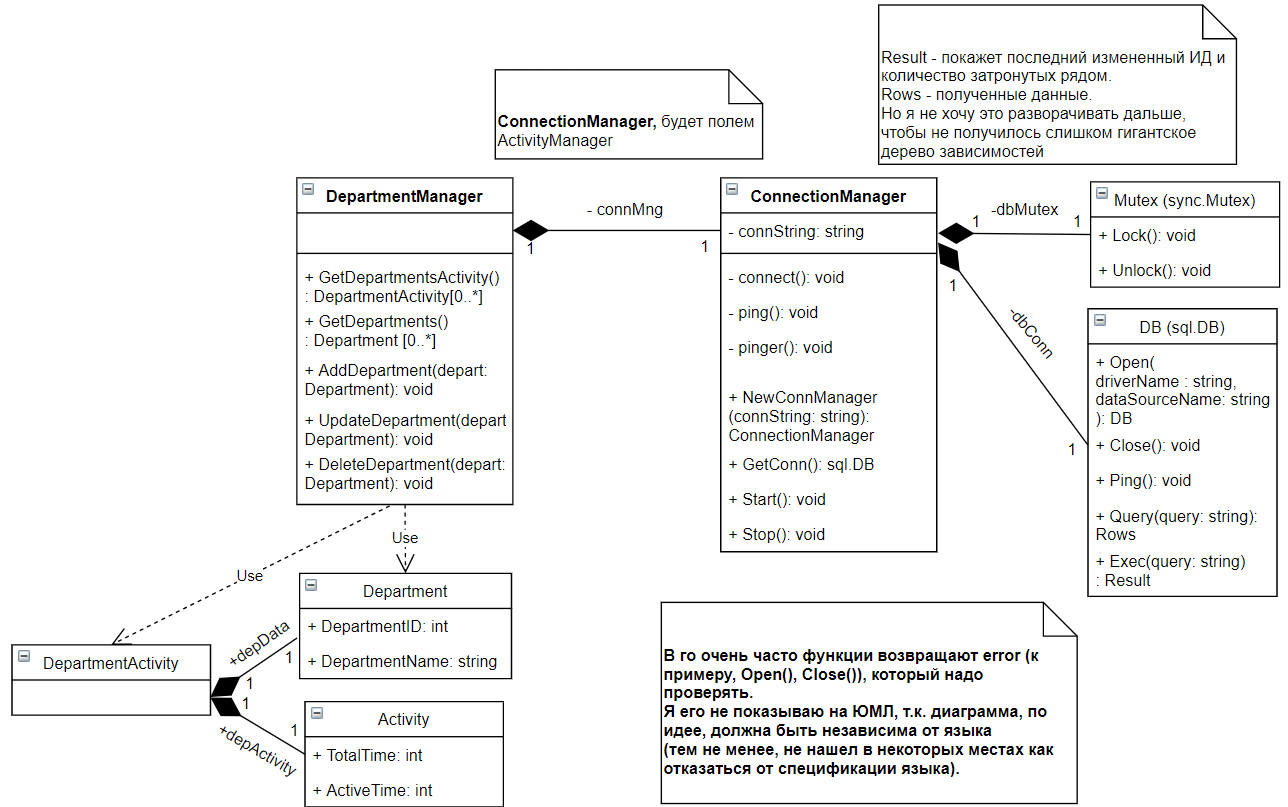
Було обрано монолітну архітектуру, бо:

* розроблюваний додаток не є достатньо великим для використання мікросервісів або SOA. І використання мікросервісів тільки додасть зайві складнощі, та може не дати ніякого позитивного ефекту.
* додаток можна буде простіше та швидше розроблювати, тестувати да розгортати.

**Діаграма компонентів та їх взаємодій (HLD)**



**Опис низькорівневої архітектури (LLD):**



**Контрольні питання:**

1. **Що таке архітектура додатку?**

Архітектура додатку – це сукупність рішень щодо організації програмного додатку, яка визначає всю організацію – всі елементи, їх інтерфейси, етс.

Проробка архітектури додатку - ключовий етапом розробки, бо від нього залежить легкість написання, деплою, підтримки, етс програмного забеспечення.

1. **Різниця між монолітом та мікросервісами**

|  |  |  |
| --- | --- | --- |
|  | Монолит | Микросервисы |
| Деплой (розгортання) | Ціла система за раз | Мікросервіси можуть розгортатися окремо |
| Надійність | Один збій може призвести до падіння всієї системи | Відмова одного сервісу не впливає на інші |
| Можливість розширення | Низька, тільки вертікальна | Висока |
| Гнучкість | Складно впроваджувати нові технології, використовувати різні мови, етс | Можливість просто впроваджувати і використовувати різні мови і технології |
| Оновлюваність | Довго збирати та оновлювати. | Можливо швидке і просте оновлення |
| Тестування | Можливість end-to-end тестування | Кожен сервіс повинен тестуватися окремо |
| Безпека | Забезпечити безпеку простіше, тому що вся обробка і передача даних відбувається на системному рівні. | Спілкування між сервісами по API gateway підвищує ризики проблем з безпекою |

1. **Що таке шина даних?**

Загальне поняття **-** комунікаційна система, що створює зв’язок між певними компонентами ІС для уникнення прямого зв’язку між ними.

*“Allows send of messages/events between components of an application without them needing to know about each other. They only need to know about the type of the message/event being sent.”*

1. **Паттерн “черга повідомлень”**

Суть паттерну полягає у передачі повідомлень через чергу клієнтам, де повідомлення видаляються одразу після їх доставки.

1. **Чим відрізняються сервіси у SOA та мікросервісах?**

Основні відмінності:

|  |  |
| --- | --- |
| **SOA** | **Микросервисы** |
| Використовує архітектурний підхід «share-as-much-as-possible». | Використовує архітектурний підхід «share-as-little-as-possible». |
| Використовує ESB для зв’язку | Використовує більш прості системи для зв’язку |
| Максимізує можливість повторного використання служб | Робить натиск на меншу залежність компонентів |

1. **Паттерни проектування - загальне визначення**

Паттерн проектування – набір певних рішень, концепція, що вирішує часто виникаючі проблеми під час проектування програмного додатку. За допомого них можно чистіше та краще спроектувати та розробити програму.

1. **Що таке DDD?**

DDD – підхід, набір принципів, до розробки програмного забезпечення, що спрямовані на створення оптимальної моделі реальної системи чи процесу. В основу DDD полягають «доменні моделі» - моделювання сутності із домену, де домен - сфера, на якій базується бізнес-логіка.

1. **Що таке CQRS?**

CQRS – паттерн, реалізацією принципу «Command Query Separation» під час розробки програмного забезпечення. Принцип полягає у тому, що певний метод повинен бути або командою, що робить певну дію, або певним запитом, що повертає дані. Проте метод не повинен змінювати та повертати дані одночасно.