 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-81

Збаровський Д. Д.

**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.

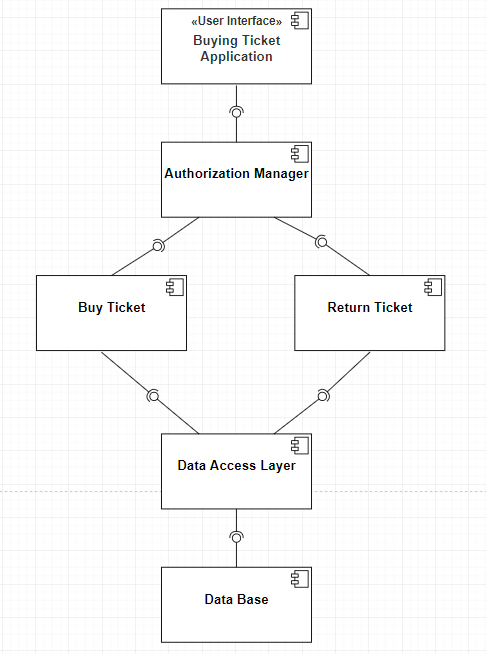
**Тематика веб-додатку:** продаж квитків на концерти

**Базові вимоги до додатку:  
Функціональні вимоги:** 1. Можливість реєстрації та аунтефікації користувача  
 2. Профіль адміністратора  
 3. Відображення всього списку доступних квитків на концерти  
 4. Можливість пошуку квитків  
 5. Можливість покупки/повернення квитка  
 6. Можливість подивитись опис про концерт

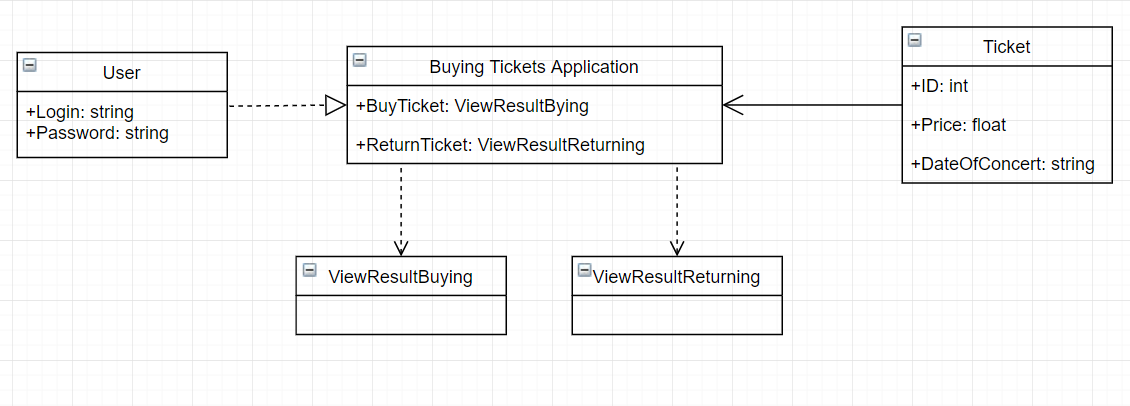
**Нефункціональні вимоги:** Кількість користувачів: 50  
 Downtime: під час обновлень  
 Інтеграція з іншими сервісами: через API  
 Доступність для людей з обмеженими можливостями: не передбачено

Безпека: надійне зберігання паролів  
 Вимоги до тестування: тестові користувачі

**Опис високорівневої архітектури:**Для даного додатку було обрано монолітну архітектуру, так як додаток не буде великого розміру, можна буде легше та швидше розроблювати і тестувати додаток. Використання мікросервісів лише додасть складнощі.

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

**Опис низькорівневої архітектури (LLD):**

****

**Відповіді на контрольні запитання:**

1. **Що таке архітектура додатку?**Архітектура додатку є набором структурних рішень щодо побудови додатку в залежності від заданих бізнес-правил. Проробка архітектури додатку є ключовим етапом при розробці, так як від правильних архітектурних рішень залежить якість процесу написання кодової бази, її підтримки, тестування, та експлуатації додатку.
2. **Різниця між монолітом та мікросервісами**

Монолітна архітектура містить всі модулі додатку в рамках одного суцільного сервісу. Мікросервісна архітектура є набором деяких відносно невеликих незалежних один від одного сервісних модулів.

1. **Що таке шина даних?**Шина даних - це комунікаційна система, роль якої полягає у створенні зв'язків між існуючими компонентами інформаційної системи для уникнення прямого зв'язку між ними.
2. **Паттерн черга повідомлень?**Черга повідомлень – патерн, що полягає у передачі деяких повідомлень (даних) через чергу клієнтам, що відразу видаляються після їх доставки.
3. **Чим відрізняються сервіси у SOA та мікросервісах?  
   Функціональна сумісність** - SOA сприяє використанню кількох гетерогенних протоколів через компонент проміжного програмного забезпечення для обміну повідомленнями. Мікросервіси намагаються спростити шаблон архітектури за рахунок скорочення числа варіантів інтеграції. Отже, якщо ви хочете інтегрувати кілька систем, що використовують різні протоколи, в гетерогенную середу, вам необхідно розглянути SOA. Якщо всі ваші сервіси можуть бути доступні через один і той же протокол віддаленого доступу, то мікросервіси - кращий варіант для вас.

**Обмежений контекст** - SOA заохочує спільне використання компонентів, тоді як мікросервіси намагаються мінімізувати спільне використання через «обмежений контекст». Обмежений контекст відноситься до з'єднання компонента і його даних як єдиного блоку з мінімальними залежностями. Оскільки SOA використовує кілька сервісів для виконання бізнес-запитів, системи, побудовані на SOA, ймовірно, будуть працювати повільніше, ніж мікросервіси.

**Комунікаці**я - в SOA ESB може стати єдиною точкою відмови, яка впливає на всю систему. Оскільки кожен сервіс зв'язується через ESB, якщо один з сервісів сповільнюється, він може засмітити ESB запитами на цей сервіс. З іншого боку, мікросервіси набагато краще переносять помилки. Наприклад, якщо один мікросервіс має помилку пам'яті, то буде порушено тільки цей мікросервіс. Всі інші мікросервіси будуть продовжувати обробляти запити регулярно.

**Розмір** - останнє, але не менш важливе, основна відмінність між SOA і мікросервісамі полягає в розмірі і обсязі. Приставка «мікро» в мікросервісах відноситься до зернистості внутрішніх компонентів, тобто вони повинні бути значно менше, ніж SOA. Сервісні компоненти в мікросервісах, як правило, мають одну мету, і вони дійсно добре виконують це завдання. З іншого боку, сервіси SOA зазвичай включають в себе набагато більше бізнес-функцій, і вони часто реалізуються як повні підсистеми.

1. **Паттерни проектування – загальне визначення.**Паттерни проектування – це повторювана архітектурна конструкція, що представляє собою рішення проблеми проектування в рамках деякого часто виникає контексту.

Зазвичай шаблон не є закінченим зразком, який може бути прямо перетворений в код; це лише приклади розв'язання задач, який можна використовувати в різних ситуаціях. Об'єктно-орієнтовані шаблони показують відносини і взаємодії між класами або об'єктами, без визначення того, які кінцеві класи або об'єкти додатки будуть використовуватися.

1. **Що таке DDD?**DDD – це це набір принципів і схем, спрямованих на створення оптимальних систем об'єктів. Зводиться до створення програмних абстракцій, які називаються моделями предметних областей. У ці моделі входить бізнес-логіка, що встановлює зв'язок між реальними умовами області застосування продукту і кодом.

Предметно-орієнтоване проектування не є який-небудь конкретної технологією або методологією. DDD - це набір правил, які дозволяють приймати правильні проектні рішення. Даний підхід дозволяє значно прискорити процес проектування програмного забезпечення в незнайомій предметної області.

1. **Що таке CQRS?**CQRS - підхід проектування програмного забезпечення, при якому код, що змінює стан, відділяється від коду, просто читає цей стан. Подібне розділення може бути логічним і грунтуватися на різних рівнях. Крім того, воно може бути фізичним і включати різні ланки (tiers), або рівні.

В основі цього підходу лежить принцип Command-query separation (CQS).

Основна ідея CQS в тому, що в об'єкті методи можуть бути двох типів:

Queries: Методи повертають результат, не змінюючи стан об'єкта. Іншими словами, у Query не існує будь яких побічних ефектів.

Commands: Методи змінюють стан об'єкта, що не повертаючи значення.