Київський національний університет імені Т. Шевченка
Факультет комп'ютерних наук та кібернетики

Об'єктно-орієнтоване програмування

Київ-2019

Загальні вимоги

БД має близько 5 таблиць. Можна більше, але не варто, адже занадто об'ємна робота. Для БД використовувати патерн DAO. Для розвертання БД використовувати Liquibase.

Web server можна використовувати glassFish чи Tomcat (будь-який веб-сервер, що підтримує Java).

Gradle використовується для збірки проекту. Для deployment-у використовувати plugin cargo (він закидує готовий зібраний проект на Tomcat).

Фреймворк для unit-тестування: Junit Для перевірки коректності роботи патерну DAO: Mockito

Архітектурний патерн проекту MVC. View реалізувати як single page, але ніхто не забороняє використовувати багатосторінкові сайти.

Всі мови на яких написані проект мусять підпадати під code convention відповідної мови.

Додати захист від XSS-атаки.

Додати валідацію форми реєстрації як на front-end та на back-end (захист від postman).

Перша лаба

Для роботи з базою даних використовувати JDBC.

Веб-сервер написано на Java Servlets.

Друга лаба

Проект-моноліт

Заміняємо Java Servlets на Spring.

JDBC заміняєтья на Hibernate/Jooq.

Для авторизації використовувати Spring.Security

Перевірка endpoint-ів за допомогою інтеграційних тестів (Spring.Boot.Test)

Третя лаба

Створити 2 мікросервіси:

1. Сервіс авторизації (випилити з Spring.Security і використати це в Keycloack. Застосування jwt.token)
2. Бізнес сервіс (де реалізовані основна бізнес-логіка).

Налаштувати balance loader на всіх мікросервісах. Вже реалізований balance loader можна поглянути в Spring Neflix (Spring Cloud).

Застосувати протокол OAuth2. Всі запити йдуть по токенах, перевірку яких здійснює spring security.

Спілкування між двома мікросервісами реалізовано на Kafka.

Обидва мікросервіси написані на Java Spring Boot.

Далі ці три сервіси треба закатати в Docker (Kafka, сервіс авторизації, бізнес сервіс).

Використовувати Kubernetes як оркестратор (orchestrator).