**ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Формат | Позначення | Найменування | Кількість листів | Примітка |
| 1 | А4 | КПІ.ІП-5203. ХХХХХХ.02.81 | WEB-застосування проведення тематичних дискусій та on-line опитувань. Технічне завдання |  |  |
| 2 | А4 | КПІ.ІП-5203. ХХХХХХ.02.81 | WEB-застосування проведення тематичних дискусій та on-line опитувань.Пояснювальна записка |  |  |
| 3 | А4 | КПІ.ІП-5203. ХХХХХХ.03.81 | Програма та методика тестування. |  |  |
| 4 | А4 | КПІ.ІП-5203. ХХХХХХ.04.51 | Керівництво користувача. |  |  |
| 5 | А4 | КПІ.ІП-5203. ХХХХХХ.04.51 | Керівництво адміністратора. |  |  |
| 6 | А3 | КПІ.ІП-5203. ХХХХХХ.04.51 | Діаграма сценаріїв використання. |  |  |
| 7 | А3 | КПІ.ІП-5203. ХХХХХХ.04.51 | Діаграма структури бази даних |  |  |
| 8 | А3 | КПІ.ІП-5203. ХХХХХХ.04.51 | Діаграма бізнес-процесу керування даними…. |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з трьох розділів, містить 3 таблиці, Х додатків та Х джерел – загалом ХХХ сторінок.

**Об`єкт дослідження:** веб системи моделі клієнт-сервер з реляційними базами даних, що використовуються для роботи сайту для комунікації користувачів.

**Мета дипломного проекту:**  створити веб сервіс, що може бути використаний для проведення дискусійних обговорень та опитувань онлайн.

У першому розділі було проаналізовано предметну модель та розроблено архітектуру для застосування. Побудовано структурну схему класів та діаграму послідовності.

У другому розділі проведено тестування веб застосунку за розробленим планом тестування. Описано процес тестування.

У третьому розділі описано розгортання та впровадження веб застосунку, а також наведено схему структурну розгортання.

У додатках наведено: опис програми, схема структурна класів програмного забезпечення, схема структурна послідовності виконання.

КЛЮЧОВІ СЛОВА: ОНЛАЙН ДИСКУСІЇ, ОНЛАЙН ОПИТУВАННЯ, КОМУНІКАЦІЙНИЙ СЕРВІС

ABSTRACT

Explanatory note of the diploma project consists of 3 sections, X annexes, 3 tables and 5 sources – total XX pages…..

**The object of study**: server application systems, which can be used for continuous integration and delivery with Github and Kubernetes integration.

**The aim of the diploma project**: create easy scalable web application, capable of fast, rapid execution of integration and delivery process to Kubernetes cluster.

In the first section, the architecture of microservice web applications was described and developed. A structural diagram of classes and a sequence diagram are constructed.

In the second secion, resulting web application was tested according to the developed test plan. The process of testing is described.

The third section describes the deployment and implementation of microservice web applications. The diagram of structural deployment is provided.

Annexes contain the description of the program, the diagram of the structural classes of the software, the scheme of the structural sequence of execution.

KEYWORDS: CONTINUOUS INTEGRATION, CONTINUOUS DELIVERY, MICROSERVICES

ВМЕСТО эТОГО ЛИСТА ВСТАВИТЬ ЛИСТ ТИТУЛА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

ЗМІСТ

[Перелік умовних позначень, символів, одиниць, скорочень і термінів 11](#_Toc9762368)

[Вступ 12](#_Toc9762369)

[1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 13](#_Toc9762370)

[1.1 Загальні положення 13](#_Toc9762371)

[1.2 Змістовний опис і аналіз предметної області 13](#_Toc9762372)

[1.3 Аналіз успішних IT-проектів 14](#_Toc9762373)

[1.4 Аналіз вимог до програмного забезпечення 15](#_Toc9762374)

[Побудуємо результуючу матрицю трасування на Рисунку 28](#_Toc9762375)

[1.5 Висновки по розділу 28](#_Toc9762376)

[2 Моделювання та конструювання програмного забезпечення 29](#_Toc9762377)

[2.1 Моделювання та аналіз програмного забезпечення 29](#_Toc9762378)

[2.2 Архітектура програмного забезпечення 29](#_Toc9762379)

[2.3 Конструювання програмного забезпечення 29](#_Toc9762380)

[2.4 Аналіз безпеки даних 56](#_Toc9762381)

[2.5 Висновки по розділу 57](#_Toc9762382)

[3 АНАЛІЗ ЯКОСТІ ТА ТеСТУВАННЯ програмного забезпечення 58](#_Toc9762383)

[3.1 Аналіз якості ПЗ 58](#_Toc9762384)

[3.2 Підходи до тестування 59](#_Toc9762385)

[3.2.1 Компонентне тестування 59](#_Toc9762386)

[3.2.2 Інтеграційне тестування 60](#_Toc9762387)

[3.2.3 Тестування продуктивності 60](#_Toc9762388)

[3.3 Критерії проходження тестування 60](#_Toc9762389)

[3.3.1 Компонентне тестування 60](#_Toc9762390)

[3.3.2 Інтеграційне тестування 61](#_Toc9762391)

[3.3.3 Тестування швидкодії 61](#_Toc9762392)

[3.4 Процес тестування 61](#_Toc9762393)

[3.4.1 Дані до тестів 61](#_Toc9762394)

[3.4.2 Задачі тесту 62](#_Toc9762395)

[3.4.3 План виконання 62](#_Toc9762396)

[3.5 Вимоги до середовища 62](#_Toc9762397)

[3.5.1 Апаратна частина 62](#_Toc9762398)

[3.5.2 Програмна частина 62](#_Toc9762399)

[3.5.3 Вимоги до безпеки 62](#_Toc9762400)

[3.5.4 Інструменти 62](#_Toc9762401)

[3.6 Опис контрольного прикладу 63](#_Toc9762402)

[4 впровадження та супровід програмного забезпечення 65](#_Toc9762403)

[4.1 Розгортання програмного забезпечення 65](#_Toc9762404)

[4.1.1 Створення даних авторизації для кластеру 65](#_Toc9762405)

[4.1.2 Створення даних авторизації для реестру Google Container Registry 65](#_Toc9762406)

[4.1.3 Встановлення мікросервісу неперервної доставки 67](#_Toc9762407)

[4.1.4 Встановлення мікросервісу неперерної інтеграції 67](#_Toc9762408)

[4.1.5 Встановлення мікросервісу API 67](#_Toc9762409)

[4.2 Інструкція користувача 68](#_Toc9762410)

[4.2.1 Регістрація 68](#_Toc9762411)

[4.2.2 Логін 69](#_Toc9762412)

[4.2.3 Додання токену доступа до GitHub 69](#_Toc9762413)

[4.2.4 Підключення репозиторію 70](#_Toc9762414)

[4.3 Інструкція адміністратора 73](#_Toc9762415)

[4.3.1 Регістрація 73](#_Toc9762416)

[4.3.2 Логін 73](#_Toc9762417)

[4.3.3 Додання нових адміністраторів 73](#_Toc9762418)

[4.3.4 Створення деплою 74](#_Toc9762419)

[4.3.5 Налаштування та деталі деплою 74](#_Toc9762420)

[Висновки 76](#_Toc9762421)

[ПЕРЕЛІК ПОСИЛАНЬ 78](#_Toc9762422)

[Додаток А Технічне завдання 79](#_Toc9762423)

[Додаток б Опис програми 80](#_Toc9762424)

[додаток В ПРОГРАМА ТА МЕТОДИКА ТЕСТУВАННЯ 81](#_Toc9762425)

[Додаток г керівництво системного програміста Додаток д керівництво програміста Додаток є керівництво користувача 82](#_Toc9762426)

[Додаток е Графічний матеріал 85](#_Toc9762427)

[Лист 1. Схема структурна варіантів використань 86](#_Toc9762428)

[Лист 2. Схема структурна станів системи 87](#_Toc9762429)

[Лист 3. Схема бази даних 88](#_Toc9762430)

[Лист 4. Схема структурна класів програмного забезпечення 89](#_Toc9762431)

[Лист 5. Креслення вигляду екранних форм 90](#_Toc9762432)

Перелік умовних позначень, символів, одиниць, скорочень і термінів

Дискусія –

Пост –

Фреймворк -

Драйвер –

СУБД-

Роути -

….

Вступ

Інформаційні технології є важливою складовою у сучасному суспільстві. Вони зайняли багато ніш, де є корисними для людини, зокрема дозволяють поширювати знання, обмінюватись думками та емоціями в реальному часі. Це все можливо завдяки мережі інтернет, що має велике поширення та доступна майже всім.

В всесвітній мережі існує безліч сайтів та сервісів, що мають різні функції та цілі, однією з таких цілей є комунікація між людьми. Для того, щоб кожен міг в зручному форматі обмінятись думками з іншими створена велика кількість форумів, чатів, систем ведення блогу. Кожна з цих систем має свою модель яка зручна для якогось із аспектів комунікації. Також сайти можуть мати свою тематику, та додаткові можливості. Були створені різні моделі обміну інформацією, деякі сервіси комбінують їх, це, наприклад, соціальні мережі, інші ж залишаються в рамках однієї. Практика створення рішень для комунікації між людьми досить поширена тому існує багато різноманітних прикладів впровадження цих комунікаційних моделей.

Сучасні рішення для проведення дискусій в веб просторі мають багато цікавих рішень, що робить їх корисними для широкого кола людей та є необхідними в рамках будування спільнот навколо професійних та соціальних тематик.

Мета розробки – систематизування теоретичних знань отриманих впродовж учбового процесу для створення сервісу з можливістю комунікації on-line. Призначення розробки - використання проекту у сфері онлайн сервісів для проведення дискусій.

Завданням даної роботи є розробка ВЕБ-застосунку для проведення онлайн дискусій між користувачами, а також проведенням онлайн опитувань. Компонентами розроблюваної системи є серверний застосунок та база даних. Результатом роботи є система яку можна використовувати в якості сервісу проведення дискусій та опитувань різної тематики.

# АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Загальні положення

Дискусійний формат обговорень посідає важливе місце в області систем комунікації, він зазвичай має за собою деяку мету – вирішення питання, поширення та якісний аналіз

## Змістовний опис і аналіз предметної області

Дискусійний формат обговорень – це формат що має за мету отримати структуровану та компетентну відповідь на поставлене питання об дати якісний аналіз щодо певного явища. Він має складатись щонайменше з постановки питання або інформаційного повідомлення від автора та обговорення представлених даних від користувачів, зацікавлених в темі. Для цього цей формат повідомлення зазвичай включає такі елементи як пост (інформаційне повідомлення автора), коментарі - обговорення інших користувачів даної теми, до того ж кожен коментар має вказувати або на пост, або на коментар іншого автора, до якого був доданий коментар. Також модель обговорення має включати в себе оцінювання матеріалів лайком або дизлайком.

Така система дає можливість отримати спільноту, що може саморегулюватись шляхом оцінки кожного повідомлення, оскільки це змушує автора робити більш корисний інформаційний внесок в повідомлення. Також це дає можливість виділяти найбільш якісні повідомлення на передній план, тому даний спосіб регуляції є найкращим вибором, коли є необхідність вивчати цей матеріал пізніше. Система побудови коментарів в ієрархічному вигляді дає можливість орієнтуватись в коментарях та вибирати цікавий напрямок обговорення читачу.

Опитування є корисним доповненням до дискусійного формату, оскільки одним з видів обговорення та отримання якісної оцінки будь якому явищу є вибір з декількох варіантів. В даному проекті буде реалізовано опитування саме у цьому вигляді, тому складовими опитування будемо:

* Поставлене питання
* Набір варіантів вибору
* Його результуюча оцінка спільнотою.

Також одним із складових моделі дискусій є поділення на спільноти – групи людей зі спільними інтересами або об’єднаними будь якою темою, що може мати необхідність в обговоренні. Цей поділ є необхідним, тому що зосереджує увагу та змістовний склад обговорень в потрібне русло та не заважає людям не зацікавлений в дискусійній темі.

## Аналіз успішних IT-проектів

Серед сервісів зі схожими властивостями можна виділити такі сервіси як Reddit, Habrahabr, Dou.ua – їх досить багато, оскільки такий формат є зручним для комунікації, коли необхідно дати можливість спілкуватись та мати корисний контент, що зручний читачам. Проте між ними є досить великі відмінності, вони можуть мати додаткові специфічні можливості, користі для загальної тематики сайту, різний користувацький інтерфейс.

Reddit – відомий сайт, що містить загальну тематику, в ньому є розділення на сабреддіти (тематики), в яких можливо створювати пости, коментувати їх, оцінювати як пости так і коментарі. Відображує найкращі відповіді в пріоритеті. Має досить старий традиційний дизайн. В порівнянні з даним проектом Reddit має старий дизайн, не має функціонал швидкого опитування.

Habrahabr – сайт, що має загальну професійну тематику – IT. Містить багато цікавих та професійних статей, головною особливістю є те, що має строгу систему, завдяки якій на ньому коментує та створює пости відносно невелика кількість користувачів, але при цьому дані що публікуються загалом мають більшу інформаційну цінність. Має також рейтингову систему коментарів та постів, хаби (тематики), блоги компаній. На відміну від даного проекту Habrahabr має обмежену тематику та рейтинг, що розповсюджується на весь сайт, замість одної спільноти.

Dou.ua – сайт-форум для розробників з України, має менш строгу систему, серед тематики як ІТ-технології, так і працевлаштування в ІТ, має додаткові розділи пов’язані з цим а також містить рейтинг компаній, виділяє найбільш оцінені коментарі. В порівнянні з Dou.ua, даний проект не має спеціалізації на темі та не має специфічного функціоналу – весь контент створюється в рамках моделі спільнот.

## Аналіз вимог до програмного забезпечення

Варіанти використання в системі представлені нижче:

Таблиця 1.1 – Варіант використання UC001

|  |  |
| --- | --- |
| Назва | Авторизація користувача |
| Опис | Користувач має можливість авторизуватись в системі. |
| Учасники | Користувач |
| Передумови |  |
| Постумови | Користувач проходить авторизацію |
| Основний сценарій | 1. Система показує вікно входу, що містить логін та пароль. Користувач заповнює поля та натискає кнопку «Увійти». Система авторизує користувача |
| Розширення сценаріїв | 4.1 Система виявляє, що введені дані некоректні  4.1.а Система демонструє повідомлення про помилку. |

Таблиця 1.2 – Варіант використання UC002

|  |  |
| --- | --- |
| Назва | Реєстрація користувача |
| Опис | Користувач має можливість реєструватись в системі. |
| Учасники | Користувач |
| Передумови | Користувач не має облікового запис в системі |
| Постумови | Користувач має обліковий запис в системі |
| Основний сценарій | 1. Система показує вікно реєстрації з полями «Ім’я», «Стать», «Логін», «Пароль» та кнопку «Реєстрація». Користувач заповнює всі поля, натискає на кнопку та переходить на сторінку користувача. |
| Розширення сценаріїв | 4.1 Система виявляє, що введені логін існує  4.1.а Система демонструє повідомлення про існуючий логін. |

Таблиця 1.3 – Варіант використання UC003

|  |  |
| --- | --- |
| Назва | Створення нової спільноти користувачем |
| Опис | Користувач може створювати спільноти. |
| Учасники | Користувач |
| Передумови | Користувач зареєстрований в системі |
| Постумови | Користувачем створена спільнота |
| Основний сценарій | 1. Користувач вводить значення в поля «URL спільноти», «Назва спільноти», «Опис спільноти» дані. Та натискає кнопку «Створити спільноту» |

Продовження таблиці 1.3

|  |  |
| --- | --- |
| Розширення сценаріїв | 4.1 Система виявляє, що введена назва спільноти та URL існує  4.1.а Система демонструє повідомлення про існуючу спільноту. |

Таблиця 1.4 – Варіант використання UC004

|  |  |
| --- | --- |
| Назва | Створення посту |
| Опис | Користувач може створювати пости |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та належить хоча б одній спільноті |
| Постумови | Створений новий пост користувачем |
| Основний сценарій | 1. Користувач в вікні створення посту вводить дані в поля «Заголовок посту», «Пост», «Спільнота посту», при необхідності додає опитування, ввівши його назву та вибравши пункти опитування. Натискає кнопку «Створити пост». В системі створюється пост. |
| Розширення сценаріїв | 4.1 Система виявляє, що пост занадто короткий  4.1.а Система демонструє повідомлення про мінімальну довжину посту.  4.2 Система виявляє, що пост занадто довгий  4.2.а Система демонструє повідомлення про максимальну довжину посту. |

Таблиця 1.5 – Варіант використання UC005

|  |  |
| --- | --- |
| Назва | Коментування посту, коментарю |
| Опис | Користувач може коментувати пости та коментарі |
| Учасники | Користувач, автор посту або коментарю |
| Передумови | Користувач має обліковий запис в системі та належить спільноті, до якої доданий пост |
| Постумови | До посту доданий коментар |
| Основний сценарій | 1. Користувач натискає кнопку «Відповісти» до посту або коментарю. Вводить повідомлення коментаря в поле «Коментар». Та натискає «Надіслати» |
| Розширення сценаріїв | 4.1 Система виявляє, що коментар занадто довгий  4.1.а Система демонструє повідомлення про максимальну довжину коментаря. |

Таблиця 1.6 – Варіант використання UC006

|  |  |
| --- | --- |
| Назва | Вхід до спільноти |
| Опис | Користувач може входити до існуючої спільноти |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та не належить даній спільноті |
| Постумови | Користувач належить даній спільноті |

Продовження таблиці 1.6

|  |  |
| --- | --- |
| Основний сценарій | 1. Користувач входить на сторінку спільноти та натискає «Долучитись», система додає користувача до спільноти. |
| Розширення сценаріїв |  |

Таблиця 1.7 – Варіант використання UC007

|  |  |
| --- | --- |
| Назва | Вихід користувача зі спільноти |
| Опис | Користувач може покинути спільноту за необхідністю |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та належить спільноті |
| Постумови | Користувач більше не належить спільноті |
| Основний сценарій | 1. Користувач заходить на сторінку спільноти та натискає кнопку «Покинути спільноту». Система видаляє запис про належність користувача спільноті та напрявляє користувача на сторінку списка спільнот даного користувача. |
| Розширення сценаріїв |  |

Таблиця 1.8 – Варіант використання UC008

|  |  |
| --- | --- |
| Назва | Голосування в опитуванні |
| Опис | Користувач може проголосувати в опитування в пості |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та належить спільноті, до якої належить пост з опитуванням |
| Постумови | Голос користувача врахувався в опитуванні |
| Основний сценарій | 1. Користувач заходить на сторінку посту, голосує . Система запам’ятовує його голос |
| Розширення сценаріїв |  |

Таблиця 1.9 – Варіант використання UC009

|  |  |
| --- | --- |
| Назва | Відміна голосу в опитуванні |
| Опис | Користувач може відмінити свій голос в опитуванні |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та належить спільноті, до якої належить пост з опитуванням, в якому користувач вже проголосував |
| Постумови | Голос користувача відмінився в опитуванні |

Продовження таблиці 1.9

|  |  |
| --- | --- |
| Основний сценарій | 1. Користувач заходить на сторінку посту, бачить результати голосування в пості. Натискає кнопку «Відмінити голос» |
| Розширення сценаріїв |  |

Таблиця 1.10 – Варіант використання UC010

|  |  |
| --- | --- |
| Назва | Зміна даних користувача |
| Опис | Користувач може змінити дані про себе на своїй сторінці |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі |
| Постумови | Дані користувача оновлені |
| Основний сценарій | Користувач заходить на власну сторінку профіля та натискає «Редагувати», вводить в відповідні поля своєго профілю дані та натискає «Зберегти» |
| Розширення сценаріїв |  |

Таблиця 1.11 – Варіант використання UC011

|  |  |
| --- | --- |
| Назва | Вихід з облікового запису |
| Опис | Користувач може вийти зі свого облікового запису |

Продовження таблиці 1.11

|  |  |
| --- | --- |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та ввійшов в систему |
| Постумови | Користувач вийшов з системи |
| Основний сценарій | Користувач натискає на будь якій сторінці зверху кнопку «Вийти» та покидає обліковий запис. |
| Розширення сценаріїв |  |

Таблиця 1.12 – Варіант використання UC012

|  |  |
| --- | --- |
| Назва | Користувач видаляє спільноту |
| Опис | Користувач може видаляти спільноти |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та є творцем спільноти |
| Постумови | Спільнота видалена |
| Основний сценарій | Користувач заходить даної спільноти та натискає кнопку «Видалити» |
| Розширення сценаріїв |  |

Таблиця 1.13 – Варіант використання UC013

|  |  |
| --- | --- |
| Назва | Користувач видаляє коментар |
| Опис | Користувач може видаляти коментарі |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та є автором коментаря |
| Постумови | Коментар видалений |
| Основний сценарій | Користувач заходить на сторінку посту, де мав свій коментар. Натискає на кнопку «Видалити коментар» |
| Розширення сценаріїв |  |

Таблиця 1.14 – Варіант використання UC014

|  |  |
| --- | --- |
| Назва | Голосування за пост |
| Опис | Користувач може голосувати за пост |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та входить в спільноту, до якої доданий пост |
| Постумови | Пост має голос користувача |
| Основний сценарій | Користувач заходить на сторінку з постом, натискає на кнопку «лайк» або «дизлайк». В системі оновлюються дані про голосування користувача за пост |

Продовження таблиці 1.12

|  |  |
| --- | --- |
| Розширення сценаріїв |  |

Таблиця 1.15 – Варіант використання UC015

|  |  |
| --- | --- |
| Назва | Користувач голосує за коментар |
| Опис | Користувач може голосувати за коментарі |
| Учасники | Користувач |
| Передумови | Користувач має обліковий запис в системі та належить спільноті, до якої доданий пост з даним коментарем |
| Постумови | Обновився голос за коментар |
| Основний сценарій | Користувач заходить на сторінку посту, та натискає кнопку «лайк» або «дизлайк» до відповідного коментаря. Система оновлює голос користувача за коментар |
| Розширення сценаріїв |  |

Також система має такі функціональні вимоги:

Таблиця 1.1 – Опис функціональної вимоги REQ001

|  |  |
| --- | --- |
| Номер | REQ001 |
| Назва | Реєстрація користувачів в системі |
| Опис | Система надає можливість користувачу реєструватись в системі |

Таблиця 1.2 – Опис функціональної вимоги REQ002

|  |  |
| --- | --- |
| Номер | REQ002 |
| Назва | Створення спільнот |
| Опис | Система надає можливість зареєстрованому користувачу створювати спільноти. |

Таблиця 1.3 – Опис функціональної вимоги REQ003

|  |  |
| --- | --- |
| Номер | REQ003 |
| Назва | Створення постів та опитувань |
| Опис | Система надає можливість зареєстрованому користувачу створювати пости та додавати в них опитування в разі необхідності. |

Таблиця 1.4 – Опис функціональної вимоги REQ004

|  |  |
| --- | --- |
| Номер | REQ004 |
| Назва | Коментування постів та коментарів |
| Опис | Система надає можливість користувачу коментувати пости та коментарі. |

Таблиця 1.5 – Опис функціональної вимоги REQ005

|  |  |
| --- | --- |
| Номер | REQ005 |
| Назва | Голосування в постах |
| Опис | Система надає можливість користувачу голосувати в постах та переглядати результат голосування. |

Таблиця 1.6 – Опис функціональної вимоги REQ006

|  |  |
| --- | --- |
| Номер | REQ006 |
| Назва | Оцінювання постів та коментарів |
| Опис | Система відображає користувачу оціюнювати пости та коментарі |

Таблиця 1.7 – Опис функціональної вимоги REQ007

|  |  |
| --- | --- |
| Номер | REQ007 |
| Назва | Оновлення даних сторінки |
| Опис | Система дозволяє користувачу оновлювати дані своєї сторінки |

Таблиця 1.8 – Опис функціональної вимоги REQ008

|  |  |
| --- | --- |
| Номер | REQ008 |
| Назва | Видалення постів та коментарі |
| Опис | Система дозволяє користувачу видаляти свої пости та коментарі |

Таблиця 1.9 – Опис функціональної вимоги REQ009

|  |  |
| --- | --- |
| Номер | REQ009 |
| Назва | Вихід та вхід зі спільноти |
| Опис | Система дозволяє користувачу вийти зі спільноти або ввійти до неї |

Таблиця 1.10 – Опис функціональної вимоги REQ010

|  |  |
| --- | --- |
| Номер | REQ010 |
| Назва | Видалення спільноти |
| Опис | Система дозволяє користувачу видаляти спільноти, які він створив |

Таблиця 1.11 – Опис функціональної вимоги REQ011

|  |  |
| --- | --- |
| Номер | REQ011 |
| Назва | Вхід в обліковий запис |
| Опис | Система дозволяє користувачу входити в власний обліковий запис |

Таблиця 1.12 – Опис функціональної вимоги REQ012

|  |  |
| --- | --- |
| Номер | REQ012 |
| Назва | Вихід з облікового запису |
| Опис | Система дозволяє користувачу вийти з облікового запису |

## Побудуємо результуючу матрицю трасування на Рисунку

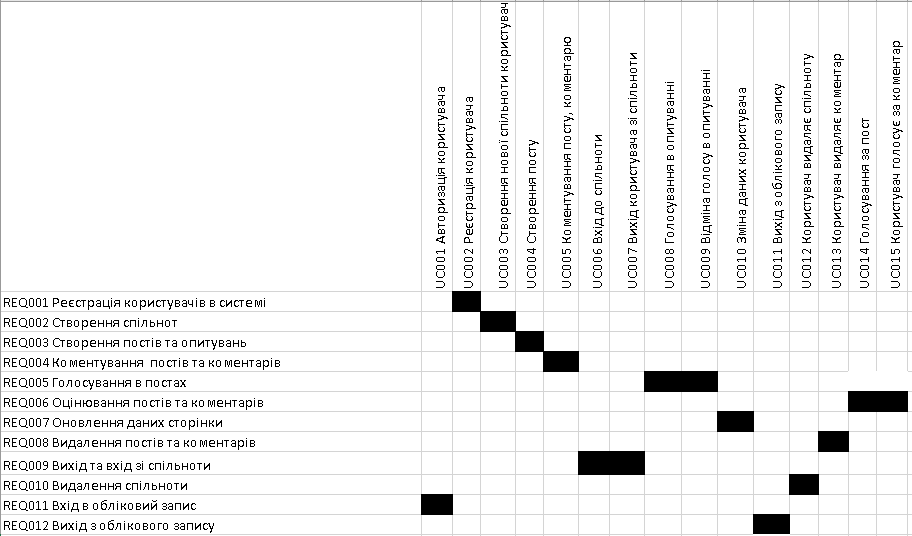


Рисунок -

## Висновки по розділу

У цьому розділі було описано та проаналізовано предметну область розробки. Було виділено успішні IT- проекти у даній області та виконано порівняння даного проекту с готовими продуктами.

# Моделювання та конструювання програмного забезпечення

## Моделювання та аналіз програмного забезпечення

Для створення програмного забезпечення необхідно провести детальний аналіз та моделювання, для чого буде використано методологію створення діаграм BPMN та буде відображено основні процеси використання проекту користувачем.

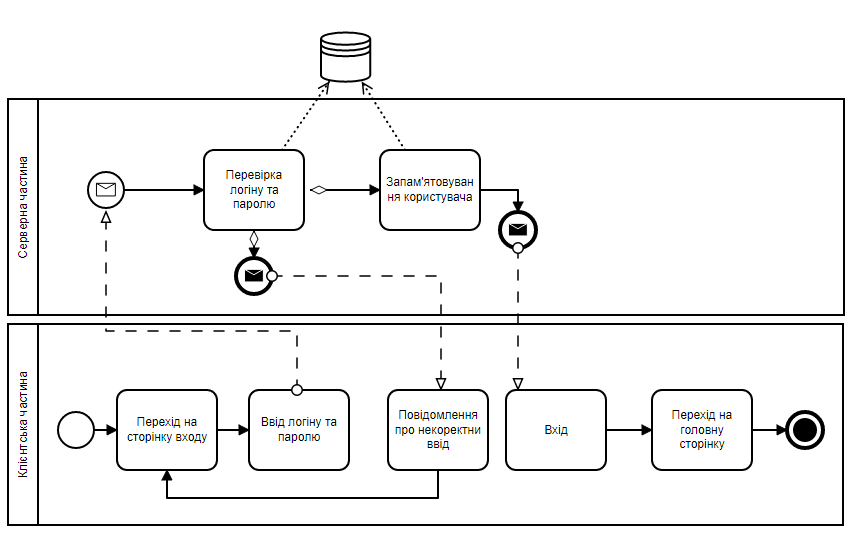


Рисунок – Вхід користувача на сайт

Послідовний опис процесу входу користувача на сайт:

* Користувач заходить на сторінку входу
* Вводить дані для входу
* Відбувається відправлення даних на сервер
* Якщо вони коректні – то сесія користувача запам’ятовується та повідомляється про успішний вхід
* Якщо некоректний, то повідомляється про помилку

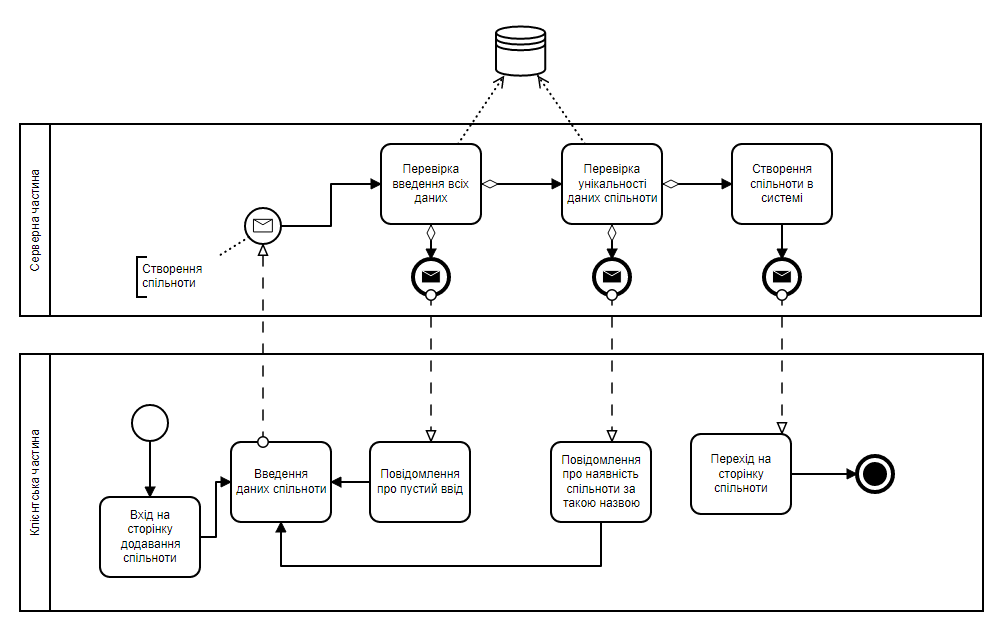


Рисунок – Створення спільноти

Опис процесу створення спільноти:

* Користувач входить на сторінку додавання спільноти
* Вводить дані нової спільноти
* Відправляє на сервер запит на створення
* Якщо все вірно, то сервер повідомляє користувача про успішне створення
* Інакше повідомляє користувача, що вже є наявна спільнота або назва є некоректною

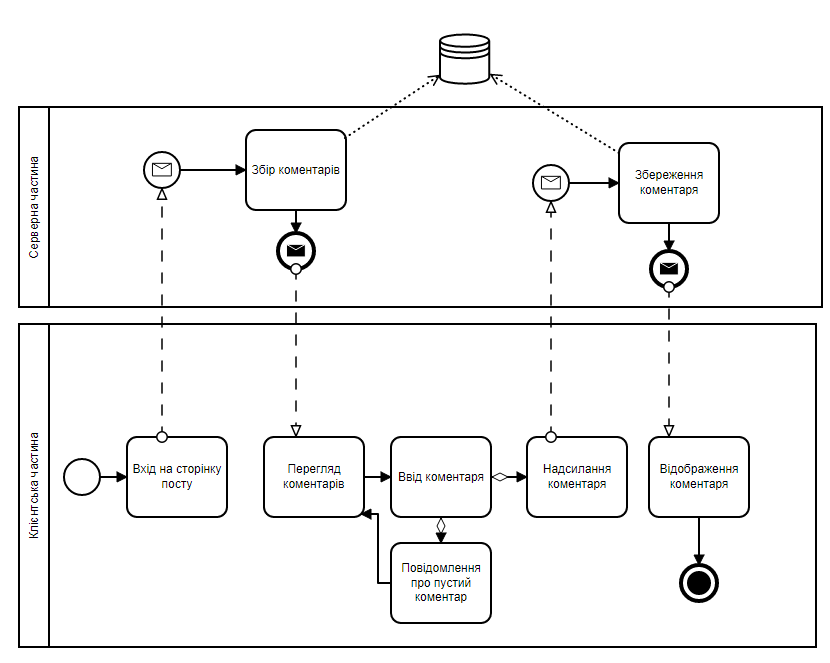


Рисунок – Додання коментаря

Послідовний опис процесу додавання коментаря:

* Вхід на сторінку посту
* Завантаження коментарів до посту
* Користувач шукає коментар до якого відповісти або відповідає до посту
* Вводить текст коментарю
* Відбувається відправлення запиту на сервер про створення коментаря
* Якщо дані коментаря коректні, то він зберігається в базу даних та відправляється користувачу як оновлений
* Інакше користувача повідомляють про помилку

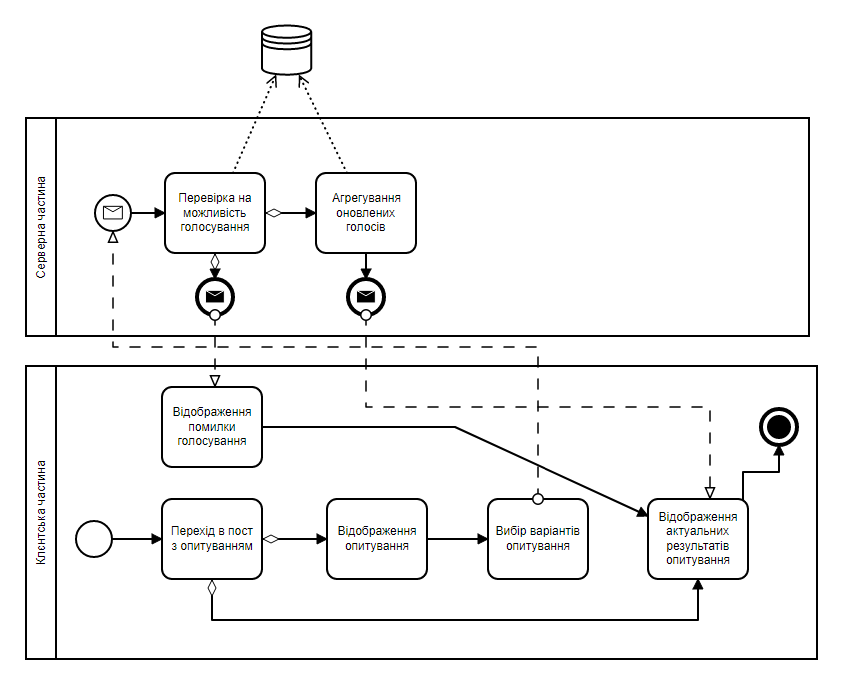


Рисунок – Бізнес процес голосування

Опис бізнес процесу голосування:

* Користувач заходить на пост з опитуванням
* Якщо він вже голосував, то отримає агреговані дані про голосування в опитуванні
* Якщо не голосував, то має список варіантів, за які він повинен голосувати
* Користувач голосує в опитуванні, дані відправляються на сервер
* Сервер перевіряє коректність даних, якщо вони коректні, то зберігає дані про голосування, агрегує нові та повертає користувачу
* Інакше повідомляє користувача про помилку

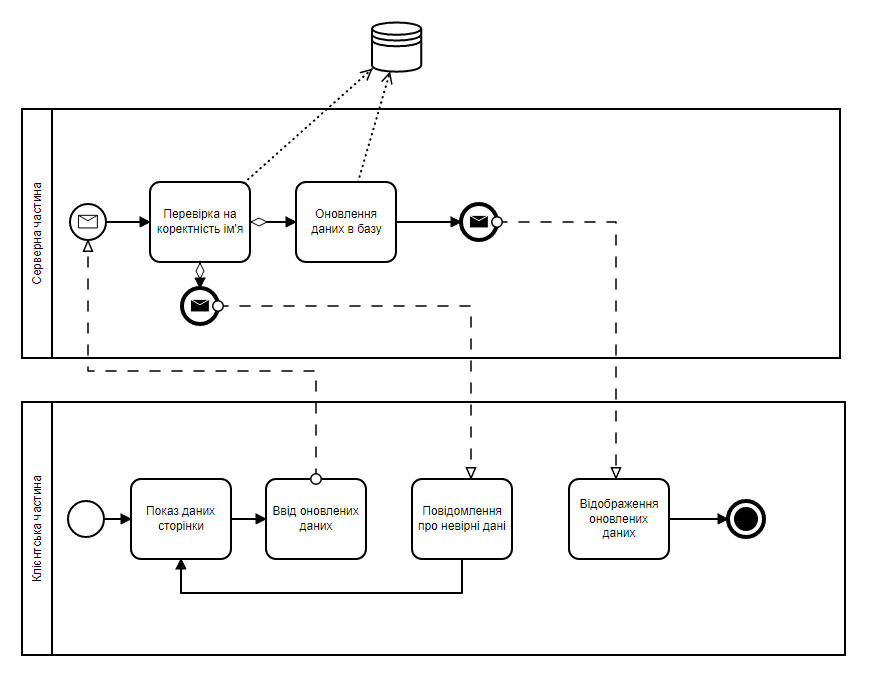


Рисунок – Оновлення даних користувача

Послідовний опис процесу оновлення даних:

* Користувач переходить на сторінку свого профілю
* Редагує дані та натиску кнопку оновлення
* Дані відправляються на сервер
* Якщо дані коректні, то вони оновлюються і повертається користувачу
* Інакше повідомляється про помилку оновлення

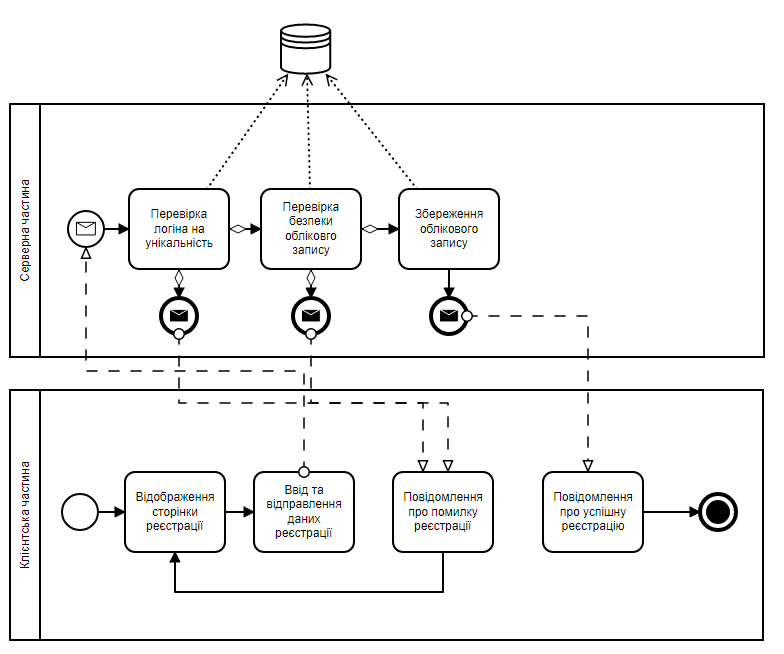


Рисунок – Реєстрація користувача

Наведено послідовний опис процесу реєстрації користувача:

* Користувачу відображується сторінка реєстрації
* Він вводить дані реєстрації
* Відправляється запит на реєстрацію на сервер
* Якщо дані не унікальні, то користувач повідомляється про некоректність даних
* Якщо дані не відповідають параметрам безпеки, то користувач повідомляється про невідповідність параметрам безпеки
* Якщо все вірно, то обліковий запис створюється, та користувач повідомляється про це

## Архітектура програмного забезпечення

Зобразимо загальну архітектуру клієнт-серверного застосування на Рисунку

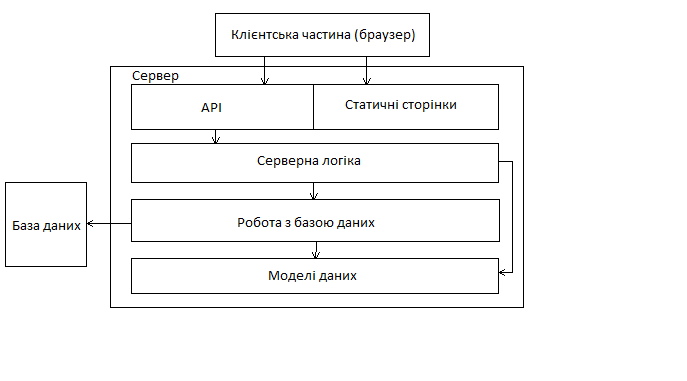


Рисунок – Принципова архітектура дипломного проекту

Розроблена система буде виконуватись під операційними системами на базі ядра Linux, оскільки ця платформа є найвигіднішим та найзручнішим варіантом для проектів такого виду.

Використовується мова програмування F# на платформі .NET Core та фреймворку Giraffe в якості серверної частини. Typescript разом з бібліотекою React для браузерної частини. PostgreSQL в якості технології реляційної БД. ВЕБ-сервер працює за принципами SPA: сервер отримує надходження запиту, якщо це запит на сторінку повертає компільований Frontend-додаток, якщо запит на API – виконує необхідну обробку даних та повертає результат у вигляді JSON. Серверна частина складається з моделі, модулю роботи з базою даних, модулем логіки за стосунку та модулем будування pipeline-ів обробки запиту, засоби програмування на F# дає можливість лаконічно робити Dependency injection, побудову rout-ів, робити обробку даних.

Клієнтська частина складається з точки входу – яка в залежності від шляху запиту відображає потрібний компонент, компонентів - основних інтерфейсних блоків програми та міні-компонентів, що дозволяють повторне використання та використовуються в різних блоках, вони з себе представляють окремі елементи інтерфейсу.

## Конструювання програмного забезпечення

Опишемо конструктивні складові серверної частини застосування

Таблиця 2.2 – Опис класів (структур) системи

|  |  |
| --- | --- |
| **Клас/модуль** | **Опис** |
| Program | Вхідний клас програми, задає основну конфігурацію |
| AuthHandlers | Клас, що містить обробку авторизаційних запитів |
| EntityHandlers | Клас, що містить логіку обробки шаблоних запитів на об'єкти системи |
| Communities.Users | Модуль, що містить логіку обробку користувачів в спільнотах |
| BuildsListComponent | Клас, який містить бізнес-логіку компоненту списку побудов. |
| Communities.Posts | Модуль, що містить логіку обробки постів спільноти |
| Communities.Posts.Poll | Модуль, що містить логіку обробки опитувань |
| Users | Клас, що містить логіку обробки користувачів, збереження, валідацію даних |
| Auth | Класс, що містить логіку обробки авторизаційних даних |

Продовження таблиці 2.2

|  |  |
| --- | --- |
| **Клас/модуль** | **Опис** |
| ApiHandlers | Модуль, що містить допоміжну логіку обробки запитів на API частину |
| ApiHandlers | Модуль, що містить логіку обробки запитів на API |
| Models | Модуль, що містить основні структури даних проекту |
| DataAccess | Модуль, що містить всі запити до бази даних |
| DataAccess.EntityQueries | Клас, що містить логіку обробки шаблонних запитів на базові об'єкти системи |
| DataAccess.SqlQueries | Клас, що містить SQL-запити до бази даних |
| AuthUtil | Клас, що містить системну логіку роботи з авторизацією |
| Logger | Клас, що містить логіку для логіювання |
| Common.Builders | Модуль, що містить основні елементи побудови виразів білдерів на F# |
| Settings | Клас, що містить обробку налаштувань додатку |
| Helpers | Клас, що містить допоміжні нерозподілені функції |

Розглянемо детальніше кожну функцію по всім класам та функціям в таблиці, що буде наведена нижче

Таблиця 2.3 – Опис методів класів та інтерфейсів системи

|  |  |  |
| --- | --- | --- |
| **Клас/Модуль** | **Метод** | **Опис** |
| Program | main() | Вхідна точка програми, налаштовує програму та сервіси згідно налаштувань |
| Program | configureLogging() | Налаштування логування даних |
| Program | configureServices() | Налаштування сервісів додатку |
| Program | configureApp() | Налаштування основної частини додатку |
| Program | cookieAuth() | Налаштування cookies |
| Program | errorHandler() | Обробка помилкових ситуацій |
| Logger | log(s: string) | Логування інформації |
| Logger | logError(s: string) | Логування помилки |
| Logger | logSql(sql: string) | Логування запиту на сервер бази даних |
| Logger | logWarn(s: string) | Логування за попередженням |
| Settings | getAppSettings() | Отримання налаштувань запуску |
| AuthHandlers | makeLogin() | Обробка запиту логіну в системі |
| AuthHandlers | makeRegister() | Обробка запиту реєстрації в системі |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Клас/Модуль** | **Метод** | **Опис** |
| AuthHandlers | makeLogout() | Обробка запиту вихода користувача з облікового запису |
| EntityHandlers | getEntity() | Обробка запиту на базовий об’єкт системи |
| EntityHandlers | getEntities() | Змінює статус користувача на розробника або адміністратора. |
| Communities.Users | getIsUserInCommunity(userId: int, communityId: int) | Перевірити чи користувач є в спільноті |
| Communities.Users | addUserToCommunity(userId: int, communityId: int) | Додати користувача до спільноти |
| Communities.Users | findUserInCommunity(query: string, communityId: int) | Пошук користувача в спільноті |
| Communities.Users | removeUserFromCommunity(userId: int, communityId: int) | Видалення користувача зі спільноти |
| Communities.Posts | createPost(communityId: int, authodId: int, postModel: Post) | Створює пост |
| Communities.Posts | deletePost(postId: int) | Видаляє пост |
| Communities.Posts | getPostStatistics(postId: int) | Отримання статистики посту |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Клас/Модуль** | **Метод** | **Опис** |
| Communities.Posts.Poll | createPoll(postId: int, data: PostData) | Створити опитування |
| Communities.Posts.Poll | voice(postId: int, voiceNumber, int) | Голосувати за пост |
| Communities.Posts.Poll | getStats(postId: int) | Отримати дані голосування |
| Users | getUserPageData(userId: int) | Отримати дані користувача для відображення на сторінці |
| Users | updateUserData(userId: int, data: UserData) | Обновити дані користувача |
| Auth | getHasExistingLogin(s: string) | Перевіряє чи є вже наявний користувач |
| Auth | registerUser(data: UserRegisterData) | Реєстрація користувача |
| Auth | logoutUser(sessionId: int) | Вихід користувача за сесії |
| Auth | loginUser(userId: int) | Вхід користувача в сесію |
| ApiHandlers | getCurrentUserHandler() | Отримати дані для заголовка в поточного користувача |
| ApiHandlers | searchOnSite() | Пошук на сайті |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Клас/Модуль** | **Метод** | **Опис** |
| ApiHandlers | getUserFeed() | Отримати пости, які мають відображатись в стрічці новин користувача |
| ApiHandlers | getCommunityPageCardInfo() | Отримати деталі для сторінки спільноти |
| ApiHandlers | getCommunityPosts() | Отримати пости спільноти |
| ApiHandlers | getUserCardInfo() | Отримати деталі для карточки користувача |
| ApiHandlers | getUserPosts() | Отримати пости користувача |
| ApiHandlers | getPostData() | Отримати дані поста |
| ApiHandlers | getPostComments() | Отримати коментарі поста |
| ApiHandlers | getListCommunities() | Отримати список спільноти |
| ApiHandlers | getListUsers() | Отримати список користувачів |
| ApiHandlers | getPollInfo | Отримати дані опитування |
| ApiHandlers | createPost() | Створити пост |
| ApiHandlers | deletePost() | Видалити пост |
| ApiHandlers | votePost() | Оцінка поста |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Клас/Модуль** | **Метод** | **Опис** |
| ApiHandlers | voteComment() | Проголосувати за коментар |
| ApiHandlers | votePoll() | Голосування в опитуванні |
| ApiHandlers | makeComment() | Створити коментар |
| ApiHandlers | enterCommunity() | Ввійти в спільноту |
| ApiHandlers | exitCommunity() | Вийти зі спільноти |
| ApiHandlers | createCommunity() | Створити спільноту |
| ApiHandlers | banUser() | Забанити користувача в спільноті |
| ApiHandlers | deleteCommunity() | Видалити спільноту |
| ApiHandlers | updateCommunity() | Оновити дані спільноти |
| Models | modelToJson(model: object) | Конвертувати модель в JSON |
| Models | jsonToModel(json: string) | Конвертувати JSON в модель |
| DataAccess | createTransaction() | Створити транзакцію до бази даних |
| DataAccess | createSqlQuery(sql: string, params: SqlParams) | Створити запит SQL |
| DataAccess | genereateModelSqlQuery() | Згенерувати модель SQL |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Клас/Модуль** | **Метод** | **Опис** |
| DataAccess.EntityQueries | getEntity(data: QueryData) | Отримати дані об’єкта |
| DataAccess.EntityQueries | getAllEntities(data: QueryData, paging: PagingData) | Отримати всі об’єкти за умовою |
| DataAccess.SqlQueries | getUserProfile() | Запит на профіль користувача |
| DataAccess.SqlQueries | searchInSite() | Пошук на сайті |
| DataAccess.SqlQueries | getUserFeed | Запит на стрічку новин користувача |
| DataAccess.SqlQueries | getCommunityPageCardInfo() | Отримати дані для сторінки спільноти |
| DataAccess.SqlQueries | getCommunityPosts() | Отримати пости спільноти |
| DataAccess.SqlQueries | getUserCardInfo() | Отритмати дані для сторінки користувача |
| DataAccess.SqlQueries | getUserPosts() | Отримати пости користувача |
| DataAccess.SqlQueries | getPostData() | Отримати дані посту |
| DataAccess.SqlQueries | getPostComments() | Отримати коментарі посту |
| DataAccess.SqlQueries | getListCommunities() | Отримати список спільнот |
| DataAccess.SqlQueries | getListUsers() | Отримати список користувачів |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Клас/Інтерфейс** | **Метод** | **Опис** |
| DataAccess.SqlQueries | createPost() | Створити пост в спільноті |
| DataAccess.SqlQueries | deletePost() | Видалити пост зі спільноти |
| DataAccess.SqlQueries | votePost() | Оцінити пост значенням |
| DataAccess.SqlQueries | voteComment() | Оцінити коментар у пості |
| DataAccess.SqlQueries | makeComment() | Створити коментар до посту |
| DataAccess.SqlQueries | register | Зареєструватись в системі |
| DataAccess.SqlQueries | getLoginData() | Отримати інформацію про обліковий запис користувача |
| DataAccess.SqlQueries | checkCredentials() | Перевірити дані входу на правильність |
| DataAccess.SqlQueries | setTokenToUser() | Створити сесію користувача |
| AuthUtil | encryptPasswrod() | Створити шифрування для паролю, щоб зберігати його в базі даних |
| AuthUtil | generateToken() | Сгенерувати токен для сесії |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Клас/Інтерфейс** | **Метод** | **Опис** |
| Common.Builders | maybeBuilder | Створити білдер для Maybe монади |
| Common.Builders | resultBuilder | Створити білдер для Result монади |
| Common.Builders | asyncBuilder | Створити білдер для Async монади |
| Helpers | trimAll(s: string) | Видалити всі зайві пробіли |

Розглянемо конструктивну структуру клієнтської частини додатку, вона складається за модулів логіки, віджетів та компонентів. Модулі логіки містять код для обробки даних, взаємодії з серверною частиною додатку. Віджети – це найголовніші інтерфейсні елементи додатку. Компоненти – допоміжні блоки для створення інтерфейсу. Перелік конструктивних елементів представлений в таблиці.

Таблиця –

|  |  |  |
| --- | --- | --- |
| **Назва** | **Тип** | **Опис** |
| CommunityList | Віджет | Відображає список спільнот з фільтрацією і сортуванням |
| CommunityWidget | Віджет | Відображає сторінку спільноти, містить інформацію про спільноту та статистичні дані |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Назва** | **Тип** | **Опис** |
| IndexWidget | Віджет | Представляє основний віджет застосунку, відображає найрелевантіші пости та основні операції |
| LoginWidget | Віджет | Відображення сторінки входу, взаємодія з сервером для входу |
| NewPostWidget | Віджет | Віджет для створення нового посту користувача |
| PostWidget | Віджет | Відображення посту та коментарів в ньому |
| RegisterWidget | Віджет | Сторінка реєстрації |
| UsersListWidget | Віджет | Віджет списку користувачів з сортуванням та фільтрами |
| UserWidget | Віджет | Відображення сторінки користувача |
| CommentTree | Компонент | Відображення дерева коментарів |
| CommunitySmallCard | Компонент | Картка спільноти в списках |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| **Назва** | **Тип** | **Опис** |
| Header | Компонент | Шапка сайту з основними даними |
| OneComment | Компонент | Відображення коментаря |
| PostContent | Компонент | Відображення контенту посту |
| PostLong | Компонент | Відображення посту зі всіма даними |
| PostShort | Компонент | Preview посту в списках |
| TextEditor | Компонент | Редактор для вводу текстових даних |
| UserSmallCard | Компонент | Preview картки користувача в списках |
| CommonLogic | Модуль логіки | Логіка яка використувується в усьому клієнтському застосунку |
| CommonApi | Модуль логіки | Взаємодія з сервером |
| PollLogic | Модуль логіки | Логіка автооновлення даних опитування в пості |
| CommentLogic | Модуль логіки | Логіка автооновлення коментарів в пості |

## Аналіз безпеки даних

Безпека даних є багатогранною для такого застосунку, для забезпечення безпеки даних необхідно врахувати такі пункти:

* Безпека зберігання даних в БД
* Безпека обробки запитів до API
* Безпека роботи з сесіями користувача

Для забезпечення безпеки зберігання даних в БД необхідно зберігати пароль користувача, защифрувавши його за допомогою Argon2, цей спосіб наразі вважається найбезпечнішим, оскільки це хешування має досить великий час роботи, що не дає його використовувати при підстановках паролю та надійний алгоритм хешування, який на даний момент ще не скомпроментований.

Безпека обробки запитів включає в себе екранування всіх строк для запитів до API, коректний набір операцій, що не дозволяє зробити запит з більш великим рівнем доступу.

Безпека роботи з сесіями реалізована через механізм від ASP.NET Core, що надає спосіб обробки запитів з обліковими даними.

## Висновки по розділу

У даному розділі було створено архітектуру веб сервісу, обрано технології його розробки, зокрема: серверну мову програмування, що є зручною для такого класу задач, клієнтську мову програмування та фреймворк для створення інтерфейсу, також обрано реляційну СУБД, що є зручною, та має драйвер в серверній мові. Також було представлено структуру класів та функцій.

# АНАЛІЗ ЯКОСТІ ТА ТеСТУВАННЯ програмного забезпечення

## Аналіз якості ПЗ

Якість програмного забезпечення – важлива характеристика продукту, оскільки вона є складовою актуальності для кінцевого користувача, зокрема є важливою для клієнт-серверних багатокористувацьких сервісів.

Для забезпечення якості одним із важливих процесів в розробці програмного забезпечення є його тестування. Тому воно є одним із головних етапів в при створенні програмних продуктів. Тестування дозволяє не допустити неякісний продукт до наступних етапів розгортання програмного продукту, завчасно знайти помилки в життєвому циклі розробки.

Для якісного тестування слід дотримуватись правил та стандартів, що були складені на основі досвіду в великій кількості програмних продуктів. Тому їх дотримання є необхідною умовою в процесах розробки. Зокрема існує такий стандарт тестування як IEEE 829—1998 Standard for Software Test Documentation.

До плану тестування ввійде обсяг, підхід, ресурси та план усіх методів тестування. План описує програмні об’єкти що будуть протестовані, тип тестів, ресурси та план необхідний для виконання тестування.

У рамках цього плану буде виконано тестування частини продукту, що відповідає за основну логіку сайту.

У даному плані будуть протестовані наступні функції:

* реєстрація користувачів;
* авторизація користувачів;
* оновлення даних користувачів;
* створення спільнот;
* оновлення даних спільнот;
* вхід до спільнот;
* створення посту;
* коментування посту
* оцінка посту та коментарю
* створення опитування
* голосування та перегляд результатів опитування.

Налаштовані наступні тестові модулі:

* реєстрація користувачів;
* авторизація користувачів;
* оновлення даних користувачів;
* створення спільнот;
* оновлення даних спільнот;
* вхід до спільнот;
* створення посту;
* коментування посту
* оцінка посту та коментарю
* створення опитування
* голосування та перегляд результатів опитування.

## Підходи до тестування

В рамках даного плану будуть використані наступні методи тестування:

* компонентне;
* інтеграційне;
* продуктивності.

### Компонентне тестування

Методом компонентного тестування будуть перевірені логічно окремі частини веб сервісу, такі як:

* роути сервісу;
* методи доступу до бази даних;
* окремі допоміжні функції;
* методи авторизації

### Інтеграційне тестування

Методом інтеграційного тестування будуть перевірені взаємодії між модулями системи, такі як:

* взаємодія веб сервера з базою даних;
* взаємодія веб сервера з клієнтом;

### Тестування продуктивності

Методом тестування продуктивності буде перевірена швидкодія наступних елементів системи:

* запити до бази даних;
* обробка сервером запитів без бази даних;
* обробка сервером запитів з базою даних.

## Критерії проходження тестування

### Компонентне тестування

Для компонентного тестування критерієм проходження є успішне виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим.

### Інтеграційне тестування

Для інтеграційного тестування критерієм проходження є успішне виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим.

### Тестування швидкодії

Для тестування швидкодії критерієм проходження є успішне виконання тесту з кожним доступним набором параметрів (кількість даних у одному запиті, кількість запитів, конкурентність) не довше ніж максимально допустимий час. У разі якщо хоча б один варіант тесту не був успішним або виконувався довше максимально допустимого часу – тестування вважається не пройденим.

## Процес тестування

### Дані до тестів

Вхідними даними для компонентного тестування є набори параметрів на яких очікується певний результат, що є вихідними даними даного тесту.

Вхідними даними для інтеграційного тестування є набори повідомлень що будуть передані від одного компоненту системи до іншого відповідно до конкретного тесту. Вихідними даними для даного виду тестування є результат роботи останнього компоненту у ланцюзі (наприклад, запис у базі даних у випадку тестування взаємодії API серверу та бази даних).

Вхідними даними до тестування швидкодії є набори даних, що покривають усі варіанти роботи системи у конкретному випадку. Вихідними даними є швидкість обробки запитів, кількість оброблених запитів, кількість неправильних реакцій на набір даних, дані по навантаженню на апаратну платформу (завантаженість процесору, вільна оперативна пам’ять, завантаженість мережі тощо).

### Задачі тесту

Кожен тест повинен перевірити як правильність програми у відповідності до умов виконання тесту (test-driven development), так і виявити можливі помилки у роботі.

### План виконання

Компонентне тестування повинне виконуватися до інтеграційного, яке, у свою чергу, виконується до тестування швидкодії.

## Вимоги до середовища

### Апаратна частина

Вимоги до апаратної частини співпадають з вимогами з технічного завдання.

### Програмна частина

Для виконання тестування платформа повинна мати операційну систему на базі Linux.

### Вимоги до безпеки

Для тестування бажано створити відповідний тестовий акаунт.

### Інструменти

Для виконання тестування використовувати наступні програмні інструменти:

* Insomnia;
* Vegeta (https://github.com/tsenart/vegeta) – для тестування швидкодії.

## Опис контрольного прикладу

У якості прикладу можна навести тестування логіну користувача за допомогою утиліти Insomnia.

Вхідні дані:

* пара логін та пароль *testAccount2 IncorrectPassword* (не існуючий пароль);
* пара логін та пароль *testAccount2 Password* (існуючий користувач).

Вихідні дані:

* повідомлення про помилку у вигляді JSON стрічки  
  *{ "message": "Password is invalid","success": false}*;
* повідомлення {"success": true} та закодовний Cookies з токеном

Для виконання тесту потрібно в Insomnia створити POST запит за адресою <http://SITE_URL/login> де SITE\_URL - адреса веб сервісу.

У якості тіла запиту використати JSON об’єкт наступного формату:

*{“login”: “LOGIN”, “password”: “PASSWORD”}* де LOGIN та PASSWORD – логін та пароль з однієї з пар вхідних даних. Після введення даних натиснути кнопку “Send” та перевірити відповідність вихідних даних до наведених далі (Рисунок 3.1).

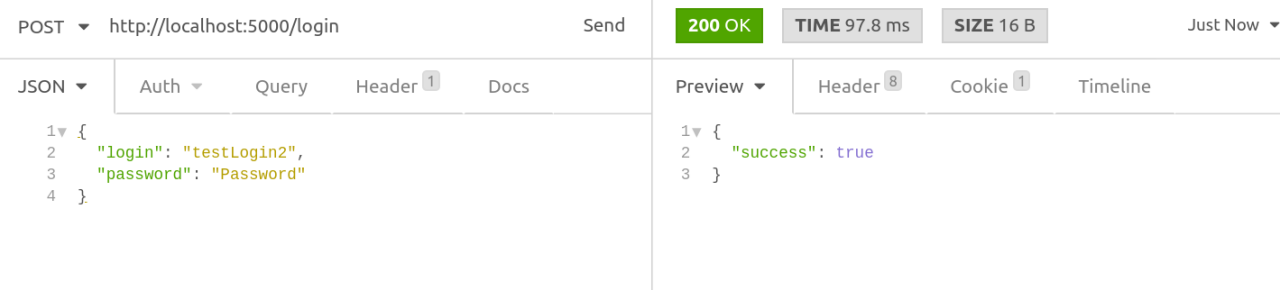


Рисунок 3.1 – Приклад виконання тесту

# впровадження та супровід програмного забезпечення

## Розгортання програмного забезпечення

Для повного розгортання веб серверу необхідно:

* Встановити основний сервіс;
* Встановити СУБД PostgreSQL
* Налаштувати файл WebApp.dll.config
* Здійснити міграцію структури БД в СУБД
* Запустити основний сервіс.

### Встановлення основного сервісу

Для встановлення основного сервісу необхідно взяти файли директорії /Bin/Server та помістити в необхідну робочу папку.

### Встановити СУБД PostgreSQL

Для встановлення СУБД необхідно перейти на сайт PostgreSQL[4] та завантажити для необхідної платформи. Далі, користуючись інструкцією з встановлення встановити та вибрати порт, стандартного користувача та пароль. Необхідно, щоб доступ до серверу СУБД був з серверу з веб сервісом.

### Налаштувати файл WebApp.dll.config

Необхідно налаштувати порт застосунку та дані доступу до бази даних, для цього слід заповнити поля файлу /Bin/WebApp.dll.config, а саме port, dbConnectionString

### Здійснити міграцію структури БД в СУБД

Для здійснення міграції слід скопіювати вміст файлу /Sql/Initial.sql та виконати його в IDE для СУБД до потрібної бази даних.

### Запустити основний сервіс

Для запуску серверу слід виконати команду dotnet run WebApp.dll до файлу, взятого з папки /Bin/WebApp.dll

## Інструкція користувача

Інструкція наведена в додатку Інструкція користувача

## Інструкція адміністратора

Інструкція наведена в додатку Інструкція системного програміста

Висновки

В процесі розробки дипломного проекту був проведений повний цикл робіт для аналізу та створення програмного продукту. Він був розділений на складові, що є невід’ємними частинами проекту, для кожної був виділений розділ у дипломній документації.

У розділі “Аналіз вимог до програмного забезпечення” був проведений цілісний аналіз предметної області, вимог до програмного продукту та їх ролі в проекті.

У розділі “Моделювання та конструювання програмного забезпечення” було проведене моделювання застосунку, зокрема зроблені всі необхідні описові матеріали логічної складової сервісу, на основі цього було проведення конструювання, результатом якого було отримано цілісну архітектуру веб сервісу для проведення онлайн дискусій.

У розділі “Аналіз якості та тестування програмного забезпечення” було проаналізовано показники якості застосування, обґрунтовано необхідність тестування застосунку та наведено всі необхідні тестові матеріали.

У розділі “Впровадження та супровід програмного забезпечення” було описано процес впровадження сервісу, зокрема його встановлення на апаратну частину та належне налаштування, описано можливості для супроводу. Також було розроблено інструкцію користувача та адміністратора.

В рамках дипломного проекту були застосовані навики, отримані під час навчання, проаналізовано їх можливість та доречність в застосуванні відносно проекту та виведено необхідний набір матеріалу до проекту, зокрема були використанні навики побудови архітектури, засобів проектування, паралельного розподілення задач, захисту даних та мультипарадигменного програмування.

Розроблений дипломний проект є логічно цілісним та коректним, надійно протестованим, тому призначений для повноцінного застосування в цільових сферах комп'ютерних технологій.

ПЕРЕЛІК ПОСИЛАНЬ

1. CircleCI https://circleci.com - офіційна сторінка у мережі Інтернет.
2. Jenkins <https://jenkins.io> - офіційна сторінка у мережі Інтернет.
3. Helm <https://helm.sh> - офіційна сторінка у мережі Інтернет.
4. deploy-node-app <https://github.com/kubesail/deploy-node-app> - офіційна сторінка у мережі Інтернет.
5. TEST PLAN OUTLINE IEEE829 <https://www.fit.vutbr.cz/study/courses/ITS/public/ieee829.html> -  офіційний опис стандарту
6. Google Cloud Platform https://cloud.google.com/kubernetes-engine/docs/how-to/cluster-access-for-kubectl – Configuring cluster access for kubectl (Налаштування доступу до кластеру за допомогою утиліти kubectl)
7. Google Cloud Platform <https://cloud.google.com/container-registry/docs/advanced-authentication#json_key_file> - Advanced authentication (Додаткові методи аутентіфікації)
8. Docker Documentation docs.docker.com/engine/reference/commandline/ - Docker CLI reference (Опис утиліти командного рядку Docker)
9. Kubernetes Documentation <https://kubernetes.io/docs/home/> - Документація оркестратору Kubernetes
10. Golang Docs <https://golang.org/doc/> - Документація мови Go
11. Typescript Documentation <https://www.typescriptlang.org/docs/home.html> - Документація мови Typescript

Додаток А Технічне завдання

Все эти листы не печатаются! Они нужны только для того, чтоб сделать правильное содержание

Додаток б Опис програми

додаток В ПРОГРАМА ТА МЕТОДИКА ТЕСТУВАННЯ

Все эти листы не печатаются! Они нужны только для того, чтоб сделать правильное содержание

Додаток г керівництво системного програміста  
Додаток д керівництво програміста   
Додаток є керівництво користувача

Додаток е Графічний матеріал

Лист 1. Схема структурна варіантів використань

Лист 2. Схема структурна станів системи

Лист 3. Схема бази даних

Лист 4. Схема структурна класів програмного забезпечення

Лист 5. Креслення вигляду екранних форм