ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
(повне найменування вищого навчального закладу)
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
(повне найменування інституту, назва факультету (відділення))
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до кваліфікаційної роботи

бакалавра
(освітньо-кваліфікаційний рівень)

на тему: «Розробка інтернет-сервісу для обміну повідомленнями в реальному часі з урахуванням сучасних вимог безпеки та масштабованості»

Виконав: здобувач *4* курсу, групи *4ПР2*
напряму підготовки (спеціальності)
121 «Інженерія програмного забезпечення»
(шифр і назва напряму підготовки, спеціальності)

Тищенко Микола Олександрович
(прізвище та ініціали)

Керівник к.т.н., доцент Доровська І.О
(прізвище та ініціали)

Рецензент *Вороненко М.О*
(прізвище та ініціали)

Хмельницький – 2025 року

ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Інститут, факультет, відділення *Інформаційних технологій та дизайну*
Кафедра, циклова комісія *Програмних засобів і технологій*
Освітньо-кваліфікаційний рівень *бакалавр*
ОПП *Програмна інженерія*
(шифр і назва)
Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії *програмних засобів і технологій*

_____ к.т.н., доцент О.Е. Огнєва
«___» _____ 2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Тищенко Микола Олександрович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) « Розробка інтернет-сервісу для обміну повідомленнями в реальному часі з урахуванням сучасних вимог безпеки та масштабованості»

керівник проекту (роботи) к.т.н, доцент Доровська Ірина Олександрівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «_20_» січня 2025 року №_95-с_

2. Строк подання студентом проекту(роботи)

3. Вихідні дані до проекту (роботи)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження та аналіз предметної області
2. Аналіз вимог та архітектурних рішень для чат-системи
3. Проектування програмного продукту
4. Розробка та застосування програмного продукту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Комп'ютерна презентація

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	09.02.2025	Виконано
2	Підбір літератури	10.02.2025 – 25.02.2025	Виконано
3	Аналіз предметної області	28.02.2025 – 10.03.2025	Виконано
4	Розробка концептуальної моделі	12.03.2025 – 22.03.2025	Виконано
5	Розробка та проектування системи	25.03.2025 – 30.03.2025	Виконано
6	Розробка та проектування бази даних	02.04.2025 – 10.04.2025	Виконано
7	Розробка інтерфейсу додатка	12.04.2025 – 20.04.2025	Виконано
8	Написання вихідного коду програми	21.04.2025 – 20.05.2025	Виконано
9	Оформлення пояснювальної записки	21.05.2025 – 01.06.2025	Виконано
10	Захист кваліфікаційної роботи	12.06.2025	Виконано

Здобувач _____ М.О.Тищенко
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ І.О.Доровська
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 128 сторінок, 74 рисунка, 2 додатки, 50 використаних джерел.

Мета роботи — створити сучасний веб-додаток схожий на Telegram Web для обміну повідомленнями в реальному часі, який буде відповідати актуальним вимогам до безпеки, масштабованості та зручності користування.

Об'єкт дослідження — веб-додатки для обміну повідомленнями.

Предмет дослідження — підхід до розробки архітектури, вибір технологій та створення масштабованої системи обміну повідомленнями з високим рівнем безпеки.

Методи дослідження — аналіз сучасних чат-платформ, дослідження можливостей мікросервісної архітектури, вивчення принципів роботи з сесіями, чергами повідомлень та контейнеризацією. Практична розробка з використанням NestJS, NextJS, RabbitMQ, Docker, Redis, Elasticsearch та інших сучасних інструментів.

Результати роботи:

- Розроблено чат-додаток подібний до Telegram Web.
- Реалізовано мікросервісну архітектуру з використанням NestJS, RabbitMQ та API Gateway.
- Створено базу даних за допомогою PostgreSQL та ORM Prisma.
- Запроваджено Redis для керування сесіями.
- Здійснено інтеграцію з Elasticsearch для швидкого пошуку користувачів.
- Розроблено інтерактивний інтерфейс за допомогою NextJS із застосуванням App Router, Zustand та Tailwind CSS.
- Увесь застосунок зібрано в Docker-контейнери для зручного розгортання через Docker Compose.

Новизна роботи — полягає у комплексному підході до створення сучасного чату з використанням мікросервісної архітектури, контейнеризації та інтеграції з інструментами для пошуку та обробки даних у реальному часі.

На відміну від типових реалізацій, тут зроблено акцент на безпечну автентифікацію через сесії, гнучку структуру сервісів та ефективну роботу з великим навантаженням.

Ключові слова: чат-додаток, обмін повідомленнями, мікросервіси, NestJS, NextJS, Redis, RabbitMQ, Elasticsearch, Docker, масштабування, безпека.

Анотація

Кваліфікаційна робота бакалавра має наступні структурні частини: вступ, чотири розділи, висновки, список використаних джерел та додатки.

Перший розділ «Дослідження та аналіз предметної області» складається з наступних підрозділів: «Огляд сучасних чат-додатків та їх архітектури», «Особливості мікросервісної архітектури для чат-додатків», «Сучасні технології для розробки високонавантажених бекенд-систем», «Сучасні технології для розробки фронтенду чат-додатків» та «Огляд технічних реалізацій популярних чат-додатків». У межах цього розділу здійснено глибокий аналіз функціональних та технічних характеристик популярних чат-систем, порівнюються підходи до архітектури, зокрема переваги та недоліки мікросервісного підходу, а також вивчаються сучасні інструменти для побудови як серверної, так і клієнтської частин застосунків.

Другий розділ «Технічні вимоги та архітектурні рішення для чат-системи» містить такі підрозділи: «Функціональні вимоги до системи», «Візуальне моделювання системи: діаграми та схеми взаємодії» та «Проектування зовнішнього вигляду користувацького інтерфейсу чату». На основі проведеного аналізу сформовано перелік основних функціональних потреб майбутнього продукту, побудовано візуальні моделі взаємодії користувача з системою, а також розроблено первинні макети інтерфейсу, що забезпечують зручність та інтуїтивну зрозумілість у користуванні.

Третій розділ «Проектування програмного продукту» включає: «Архітектурні рішення та компонентна структура бекенду», «Проектування бази даних та інтеграція хмарного сховища», «Застосування контейнеризації за допомогою Docker та Docker Compose» та «Архітектура та проектування фронтенду». У цьому розділі акцент зроблено на технічній реалізації ключових частин системи: детально описано логіку взаємодії між мікросервісами, структуру бази даних і принципи зберігання медіа, а також подано опис розгортання за допомогою Docker. Окремий підрозділ присвячений побудові клієнтської частини з урахуванням сучасних вимог до UI/UX.

Четвертий розділ «Реалізація та практичне використання продукту» складається з підрозділів: «Демонстрація реалізованого продукту», «Технічна інструкція для розробників» та «Напрями удосконалення та майбутні перспективи». У завершальній частині роботи представлено повнофункціональний чат-додаток, що відповідає визначеним вимогам. Детально описано процес розгортання системи, наведено інструкції для розробників щодо локального запуску та супроводу проекту. Також запропоновано можливі напрями подальшого розвитку, зокрема розширення функціональності, масштабування та покращення продуктивності.

ABSTRACT

The bachelor's qualification work has the following structural parts: introduction, four chapters, conclusions, list of sources used and appendices.

The first section "Research and analysis of the subject area" consists of the following subsections: "Overview of modern chat applications and their architecture", "Features of microservice architecture for chat applications", "Modern technologies for developing high-load backend systems", "Modern technologies for developing the frontend of chat applications" and "Overview of technical implementations of popular chat applications". Within the framework of this section, an in-depth analysis of the functional and technical characteristics of popular chat systems is carried out, approaches to architecture are compared, in particular the advantages and disadvantages of the microservice approach, and modern tools for building both the server and client parts of applications are studied.

The second section, “Technical Requirements and Architectural Solutions for a Chat System,” includes the following subsections: “Functional Requirements for the System,” “Visual Modeling of the System: Interaction Diagrams and Schemes,” and “Designing the Appearance of the Chat User Interface.” Based on the analysis, a list of the main functional needs of the future product was formed, visual models of user interaction with the system were built, and initial interface mockups were developed to ensure convenience and intuitive usability.

The third section, “Software Product Design,” includes: “Architectural Solutions and Component Structure of the Backend,” “Database Design and Cloud Storage Integration,” “Application of Containerization Using Docker and Docker Compose,” and “Frontend Architecture and Design.” This section focuses on the technical implementation of key parts of the system: the logic of interaction between microservices, the database structure, and the principles of media storage are described in detail, and a description of deployment using Docker is provided. A separate section is devoted to building the client part taking into account modern requirements for UI/UX.

The fourth section “Implementation and practical use of the product” consists of the following subsections: “Demonstration of the implemented product”, “Technical instructions for developers” and “Directions for improvement and future prospects”. The final part of the work presents a fully functional chat application that meets the specified requirements. The system deployment process is described in detail, instructions for developers on local launch and project support are provided. Possible directions for further development are also proposed, in particular, expanding functionality, scaling and improving performance.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 9
ВСТУП 11
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 14
1.1 Огляд сучасних чат-додатків та їх архітектури 14
1.2 Особливості мікросервісної архітектури для чат-додатків 16
1.3 Сучасні технології для розробки високонавантажених бекенд-систем 20
1.4 Сучасні технології для розробки фронтенду чат-додатків 24
1.5 Огляд технічних реалізацій популярних чат-додатків 28
Висновок до розділу 1 38
РОЗДІЛ 2. ТЕХНІЧНІ ВИМОГИ ТА АРХІТЕКТУРНІ РІШЕННЯ ДЛЯ ЧАТ-СИСТЕМИ 39
2.1. Функціональні вимоги до системи 39
2.2. Візуальне моделювання системи: діаграми та схеми взаємодії 40
2.3. Проектування зовнішнього вигляду користувацького інтерфейсу чату 44
Висновок до розділу 2 50
РОЗДІЛ 3. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ 51
3.1. Архітектурні рішення та компонентна структура бекенду 51
3.2 Проектування бази даних та інтеграція хмарного сховища 73
3.3 Застосування контейнеризації за допомогою Docker та Docker Compose 79
3.4 Архітектура та проектування фронтенду 82
Висновок до розділу 3 97
РОЗДІЛ 4. РЕАЛІЗАЦІЯ ТА ПРАКТИЧНЕ ВИКОРИСТАННЯ ПРОДУКТУ 98
4.1 Демонстрація реалізованого продукту 98
4.2 Технічна інструкція для розробників 111
4.3 Напрями удосконалення та майбутні перспективи 114
Висновок до розділу 4 115
ВИСНОВОК 116
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 118
ДОДАТКИ 121

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API (Application Programming Interface)	інтерфейс програмування застосунків, що забезпечує взаємодію між різними програмними компонентами.
JWT (JSON Web Token)	стандарт передачі інформації між клієнтом і сервером (у проекті використовується сесійна аутентифікація).
NestJS	фреймворк для розробки масштабованих серверних застосунків на основі Node.js.
NextJS	фреймворк для React, що підтримує SSR (Server Side Rendering), SSG (Static Site Generation), ISR (Incremental Static Regeneration).
Nginx	веб-сервер і API Gateway для балансування навантаження та проксування запитів до мікросервісів.
PostgreSQL	реляційна база даних, що використовується у проєкті разом із Prisma ORM.
Prisma	ORM (Object-Relational Mapping) для зручної роботи з базами даних у TypeScript.
RabbitMQ	брокер повідомлень для асинхронної комунікації між мікросервісами.
Redis	база даних ключ-значення, використовується для кешування, зберігання сесій, та управління чергами.
Zustand	легка бібліотека для керування глобальним станом у React-застосунку.
SQL (Structured Query Language)	мова запитів до реляційних баз даних.
WebSocket	протокол реального часу для двосторонньої комунікації між клієнтом і сервером.
Docker	платформа для контейнеризації застосунків і сервісів.
Docker Compose	інструмент для конфігурації та запуску декількох контейнерів Docker одночасно.
TypeScript	мова програмування на основі JavaScript з підтримкою статичної типізації.
Elasticsearch	пошуковий рушій, що використовується для реалізації пошуку користувачів у застосунку.
React Query	бібліотека для управління серверним станом у React-застосунках, з підтримкою кешування, синхронізації та повторних запитів.
SSR (Server Side Rendering)	рендеринг сторінок на сервері для покращення SEO та швидкості завантаження.

HTTP-only cookie	cookies, які недоступні з JavaScript, використовуються для безпечного зберігання сесій.
App Router	нова архітектура маршрутизації в NextJS 13+ із підтримкою layout'ів, server components і server actions.
Microservices	архітектурний підхід, за якого застосунок складається з незалежних сервісів, які можна масштабувати окремо.
API Gateway	точка входу до мікросервісної системи, яка виконує маршрутизацію, аутентифікацію, та інші функції.
Session-based Authentication	спосіб аутентифікації, за якого сесія зберігається на сервері, а клієнт отримує лише ідентифікатор сесії

ВСТУП

Розвиток цифрових технологій та глобальна діджиталізація суттєво вплинули на засоби комунікації, зробивши онлайн-спілкування невід'ємною частиною сучасного суспільства. Сьогодні веб-додатки для обміну повідомленнями стали широко розповсюдженим засобом миттєвої та ефективної комунікації в реальному часі. Проте створення таких систем є складним технічним викликом, що потребує врахування продуктивності, безпеки, масштабованості та зручності використання.

Актуальність теми. Актуальність дослідження зумовлена зростаючим попитом на безпечні, масштабовані й високонавантажені системи для обміну повідомленнями. Зростаючі вимоги до онлайн-комунікації як з боку бізнесу, так і з боку кінцевих користувачів, стимулюють розвиток гнучких рішень, здатних забезпечити швидкий, надійний та захищений обмін інформацією.

Незважаючи на наявність численних готових рішень, жодне з них не стало універсальним стандартом, оскільки відрізняються архітектурою, підходами до безпеки та продуктивності. Це створює потребу в індивідуалізованих рішеннях, адаптованих під конкретні вимоги користувачів та організацій.

Особливої уваги потребує забезпечення безпеки. Використання Redis для управління сесіями, реалізація CORS і захист від XSS, CSRF та SQL-ін’єкцій. Крім того, впровадження мікросервісної архітектури, контейнеризації через Docker та використання систем черг повідомлень (RabbitMQ) забезпечують масштабованість і стабільну роботу при високих навантаженнях.

Мета і задачі дослідження. Метою роботи є розробка веб-додатку для обміну повідомленнями в реальному часі, що відповідатиме сучасним вимогам до безпеки, масштабованості та продуктивності, забезпечуючи при цьому зручний інтерфейс для користувачів.

Для досягнення мети передбачено вирішення наступних завдань:

- Аналіз існуючих рішень — дослідити існуючі платформи для обміну повідомленнями, визначити їхні переваги та недоліки.
- Розробка архітектури системи — спроектувати мікросервісну архітектуру на базі NestJS та RabbitMQ.
- Зберігання даних — впровадити PostgreSQL для збереження будь якої інформації у додатку.
- Розробка серверної частини — реалізувати WebSocket функціонал для обміну повідомленнями та безпечну аутентифікацію користувачів.
- Створення клієнтської частини — розробити інтерфейс користувача на NextJS із використанням Zustand та Tailwind CSS, що гарантуватиме зручний та адаптивний UX/UI-дизайн.
- Інтеграція безпеки — реалізувати систему безпеки з використанням Redis для сесій, а також додаткових захисних механізмів, що допоможе мінімізувати ризики атак.
- Налаштування автоматизованого розгортання — за допомогою Docker і Docker Compose.
- Тестування та оптимізація продуктивності — симулювати навантаження для оцінки стійкості системи.
- Розширення функціональності — передбачити можливість інтеграції аудіо та відеодзвінків та розширені можливості адміністрування чатів.

Об’єкт дослідження. Веб-додаток для обміну повідомленнями, що реалізує функціональність аналогічну Telegram Web з використанням мікросервісної архітектури та технологій NestJS і NextJS, з акцентом на забезпечення безпеки та масштабованості.

Предмет дослідження. Архітектурні рішення та технологічні аспекти реалізації веб-чат додатку, зокрема: мікросервісна організація на базі NestJS, механізми чергування повідомлень через RabbitMQ, система аутентифікації з Redis, методи захисту від веб-загроз, та клієнтська реалізація на NextJS з оптимізованим інтерфейсом.

Наукова новизна одержаних результатів. Наукова новизна полягає у створенні високопродуктивного чат-додатку на базі сучасної архітектури. Розроблено мікросервісну систему з використанням NestJS, RabbitMQ для взаємодії сервісів і Redis для зберігання сесій. Усі компоненти розгортаються через Docker, що спрощує налаштування, також застосовано Elasticsearch для швидкого пошуку користувачів, навіть за неточними запитами. Замість JWT реалізовано сесійну автентифікацію, що забезпечує кращий контроль безпеки.

На клієнті використано Next.js з SSR для стабільної роботи і React Query для кешування запитів та зниження навантаження на сервер.

Практичне значення одержаних результатів. Реалізований чат-додаток може бути використаний як основа для створення інших систем, які працюють у реальному часі, та повинні легко витримувати великі навантаження. Тут використано такі рішення, як мікросервісна архітектура, збереження сесій у Redis і пошук через Elasticsearch. На фронтенді також було застосовано NextJS і React Query. Реалізовані рішення забезпечують швидкий та приємний досвід для користувачів, та нові можливості для інших розробників.

Апробація результатів бакалаврської кваліфікаційної роботи. У процесі виконання роботи основні технічні рішення було перевірено на практиці — під час внутрішнього тестування. Крім того, було обговорено реалізацію з іншими студентами та викладачами на семінарах, що допомогло отримати погляд з боку, загалом обрана структура показала себе дуже добре при розробці цього проєкту.