Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(навчально-науковий інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра

Здобувача вищої освіти Коваленка Ярослава Сергійовича
(ПІБ)
академічної групи 123м-23-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)
за освітньо-професійною програмою «Комп'ютерна інженерія»
(офіційна назва)

на тему «Обгрунтування структури комп'ютерної системи бюро перекладів компанії "InText" на основі хмарних сервісів та telegram боту»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Бешта Д.О.			
розділів:				
синтез системи	доц. Бешта Д.О.			
розроблення програмного забезпечення	ас. Панферова Я.В.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ _В.В. Гнатушенко
(підпис) (ініціали, прізвище)
«_____» _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра
(бакалавра, магістра)

здобувача вищої освіти _____Коваленко Я.С.____ **академічної групи** ____123м-23-1____
(прізвище та ініціали) (шифр)
спеціальності _____123 Комп'ютерна інженерія_____
за освітньою-професійною програмою _____«Комп'ютерна інженерія»_____
(офіційна назва)
на тему «Обгрунтування структури комп'ютерної системи бюро перекладів компанії "InText" на основі хмарних сервісів та telegram боту.»,
затверджену наказом ректора НТУ «Дніпровська політехніка» від 17 жовтня 2024 р. №1388-с

Розділ	**Зміст**	**Термін виконання**
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизується предмет та мета досліджень	11.10.2024
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	25.10.2024
Синтез системи	Розробка комп'ютерної системи	15.11.2024
Розроблення програмного забезпечення	Розробка програмного забезпечення	29.11.2024
Експериментальний розділ	Проведення і обробка результатів експериментів	06.12.2024

Завдання видано _____ доц. Бешта Д.О.
(підпис керівника) (ініціали, прізвище)
Дата видачі 06 вересня 2024 р.
Дата подання до екзаменаційної комісії 10.12.2024 р.
Прийнято до виконання _____ Коваленко Я.С
(підпис здобувача вищої освіти) (ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка: 72 с., 1 таблиці, 42 рис., 1 дод., 13 джерел.

ХМАРНІ ТЕХНОЛОГІЇ, БЮРО ПЕРЕКЛАДІВ, AWS, TELEGRAM-БОТ, MYSQL, DJANGO, EC2, S3.

Об’єкт розробки: комп’ютерна система бюро перекладів «InText».

Мета роботи: автоматизація роботи бюро перекладів через хмарні технології та Telegram-бот для ефективного зберігання, обробки замовлень і комунікації з клієнтами.

Методи дослідження: використання AWS для хмарного зберігання файлів (S3), бази даних (RDS, MySQL) та серверної обробки (EC2). Telegram-бот реалізовано мовою Python із використанням Django для веб-системи.

Пояснювальна записка обґрунтовує рішення щодо перенесення системи у хмарне середовище для підвищення надійності та доступності даних.

У теоретичному розділі описано інфраструктуру системи, вибір інструментів і технологій для автоматизації процесів.

У розділі розробки представлено створення Telegram-бота та веб-сервісу з інтеграцією хмарних технологій для обробки замовлень.

У експериментальному розділі перевірено роботу системи, її продуктивність і стабільність під навантаженням.

Практична значимість: розроблена система підвищує ефективність обробки замовлень і забезпечує надійність зберігання даних завдяки хмарним технологіям.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AWS – Amazon Web Services

S3 – Amazon Simple Storage Service

EC2 – Amazon Elastic Compute Cloud

RDS – Amazon Relational Database Service

MySQL – Система керування реляційними базами даних

Python – Мова програмування

Django – Веб-фреймворк

Telegram API – Інтерфейс програмування Telegram

Telegram-бот – Автоматизована програма для обробки запитів у Telegram

HTTP – HyperText Transfer Protocol

JSON – JavaScript Object Notation

SQL – Structured Query Language

API – Application Programming Interface

ВСТУП

Об’єкт дослідження: комп’ютерна система бюро перекладів «InText», що забезпечує автоматизацію обробки замовлень та взаємодію з клієнтами.

Предмет дослідження: розробка та впровадження функціонального комплексу, що включає хмарну інфраструктуру, базу даних та Telegram-бот для автоматизації робочих процесів бюро перекладів.

Мета і завдання дослідження: обґрунтування архітектури комп’ютерної системи та розробка технічних і програмних рішень для забезпечення автоматизованої обробки замовлень, передачі та зберігання файлів, а також оптимізації взаємодії між клієнтами та фахівцями компанії.

Методи дослідження: теорія збору й аналізу даних, методи проєктування хмарних інфраструктур, інженерія програмного забезпечення та методи ручного тестування веб- і бот-систем.

Ідея роботи: створення інтегрованої системи для забезпечення автоматизації бюро перекладів «InText», яка включає функціональність Telegram-бота для прийому замовлень, збереження даних у хмарному середовищі AWS, управління базою даних MySQL та надання зручного веб-інтерфейсу для працівників компанії.

На сьогоднішній день в бюро перекладів виникають виклики, пов’язані з відключенням електроенергії та ризиком втрати даних через відсутність централізованої інфраструктури. Для вирішення цих проблем було запропоновано перенести систему у хмарне середовище, що забезпечує надійне зберігання файлів, гнучке масштабування ресурсів і безперервну роботу сервісів у будь-яких умовах.

Наукові положення:

Встановлено, що інтеграція Telegram-бота з хмарними сервісами AWS та базою даних MySQL дозволяє забезпечити автоматизацію обробки замовлень, контроль статусу проєктів і надійне зберігання документів клієнтів. Оптимізація архітектури системи сприяє підвищенню ефективності комунікації та обробки запитів у реальному часі.

Наукові результати:

Запропоновано архітектуру системи, що поєднує Telegram-бот для обміну даними з клієнтами, хмарне сховище Amazon S3 для зберігання файлів та сервер EC2 для обробки замовлень. Розроблено програмний інтерфейс на основі Django, що взаємодіє з базою даних MySQL, забезпечуючи збереження та обробку метаданих замовлень.

Достовірність та обґрунтованість висновків:

Обґрунтовані висновки базуються на проведених проєктних і тестових дослідженнях розробленої системи, які підтверджують ефективність автоматизації процесів бюро перекладів «InText».

Практичне втілення результатів:

Створена система дозволяє автоматизувати процес прийому замовлень, управління файлами клієнтів та обробки замовлень завдяки інтеграції хмарної інфраструктури AWS, бази даних MySQL та Telegram-бота, що в результаті забезпечує підвищення продуктивності та якості обслуговування клієнтів бюро перекладів «InText».

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Стисла характеристика галузі

Галузь бюро перекладів — це сфера, що займається професійним наданням послуг перекладу та локалізації, забезпечуючи комунікацію між людьми та організаціями з різних мовних і культурних середовищ. Основною метою цієї сфери є точна та якісна передача змісту інформації, що дозволяє подолати мовний бар’єр та сприяє взаєморозумінню, культурному обміну й бізнес-співпраці на міжнародному рівні.

Сьогодні галузь перекладів швидко розвивається, адаптуючись до потреб глобалізації та новітніх технологічних рішень. Завдяки комп'ютерним технологіям, таким як системи автоматизованого перекладу, машинний переклад та спеціалізовані програми, процес перекладу став більш оптимізованим та доступним, що підвищує ефективність роботи перекладачів.

Послуги перекладу надаються як індивідуальними перекладачами, так і бюро перекладів, що дозволяє охопити різні потреби — від особистих документів до великих проєктів міжнародних компаній. Використання сучасних IT-рішень у перекладацькій сфері забезпечує високий рівень точності, захист даних і конфіденційності, а також дозволяє автоматизувати рутинні процеси, зберігати глосарії та бази термінів для швидкого доступу.

Завдяки новітнім технологіям, зокрема хмарним рішенням, автоматизованим системам перекладу та цифровим глосаріям, галузь бюро перекладів здатна ефективно обробляти великі обсяги текстів із високою швидкістю та точністю. Хмарні технології дозволяють зберігати й обмінюватися даними в режимі реального часу, що сприяє оперативному доступу до необхідної інформації, покращує співпрацю між командами перекладачів та забезпечує надійний захист даних і конфіденційності.

1.1 Характеристика і структура об’єкта впровадження

1.1.1 Характеристика об’єкта впровадження

InText – це бюро перекладів, розташоване у місті Дніпро, Україна, яке надає професійні послуги з перекладу текстів для бізнес-клієнтів та приватних замовників з усього світу. Основна спеціалізація компанії охоплює переклад текстів різних тематик та форматів, зокрема технічної, літературної, медичної, юридичної документації, веб-сайтів та інших типів контенту.

Компанія працює з широким спектром мов, серед яких англійська, німецька, французька, іспанська, італійська, польська, українська та багато інших. Висококваліфіковані перекладачі компанії володіють значним досвідом у своїх спеціалізаціях і використовують сучасні програмні інструменти для забезпечення максимальної точності та якості перекладів. Серед основних технологій, які застосовуються, – автоматизовані системи перекладу та інструменти для управління проектами.

Окрім базових послуг перекладу, компанія InText надає супутні послуги, що сприяють успішному просуванню клієнтів на міжнародному ринку. До них належать локалізація веб-ресурсів і програмного забезпечення, що дозволяє адаптувати продукти до потреб конкретних аудиторій та культур. Локалізація допомагає підвищити ефективність рекламних кампаній і збільшити обсяг продажів.

Компанія також пропонує послуги з редагування текстів, щоб забезпечити бездоганну якість перекладів та їхню відповідність культурним і мовним стандартам. Додатково, InText займається консультаціями з питань ведення міжнародного бізнесу, навчанням іноземних мов для співробітників компаній та іншими послугами, що підвищують конкурентоспроможність клієнтів на глобальному ринку.

1.1.2 Організаційна структура підприємства

Організаційна структура компанії побудована так, щоб забезпечити ефективне управління проектами. У керівництві компанії основну роль виконує

генеральний директор, який займається стратегічним плануванням і прийняттям важливих рішень. Йому підпорядковується ІТ-відділ, який координує роботу технічної підтримки та відповідає за впровадження сучасних технологій у роботу компанії.

Проектами керує команда менеджерів, яка займається взаємодією з клієнтами, аналізом вимог до проектів, розподілом завдань між перекладачами і контролем якості. Кожен проект передається перекладацькій команді, яка складається з досвідчених фахівців у своїх галузях. Переклади проходять обов’язковий контроль якості, який здійснюють редактори.

Крім того, в компанії функціонує маркетинговий відділ, завданням якого є залучення нових клієнтів і розвиток бізнесу. Така структура дозволяє бюро перекладів InText забезпечувати високу якість послуг, оперативно виконувати замовлення та задовольняти потреби клієнтів у багатомовних рішеннях.

1.2 Аналіз наявної проблеми бюро перекладів "InText"

Бюро перекладів "InText" стикається з низкою критичних проблем, пов’язаних із нестабільною інфраструктурою, що може призвести до значних перешкод у роботі. Основні проблеми включають:

1. Відключення електроенергії

У сучасних умовах періодичні відключення світла унеможливлюють стабільну роботу серверів та обладнання, які забезпечують зберігання та обробку замовлень клієнтів. Це призводить до:

- затримок у виконанні проектів;
- неможливості клієнтів отримати документи вчасно;
- падіння довіри до бюро з боку клієнтів.

2. Проблеми фізичного розташування серверів

Фізична інфраструктура, зокрема сервери та обладнання, що розміщені в певному регіоні, піддається загрозам у разі надзвичайних ситуацій, таких як:

- окупація територій, що може заблокувати доступ до обладнання;

- неможливість перевезення серверів через технічні або безпекові обмеження;
- відсутність стійкої альтернативи.

3. Спроби підтримувати роботу в умовах фізичних обмежень створюють додаткові ризики для бізнесу:

- витрати на обслуговування та підтримку локального обладнання;
- нестача резервних систем для забезпечення безперервного доступу до даних.

1.3 Завдання та мета роботи

Мета роботи: Розробка та обґрунтування структури інтегрованої комп'ютерної системи бюро перекладів "InText", яка базується на хмарних сервісах та Telegram-боті. Система має забезпечити автоматизацію прийому замовлень, обробку файлів і комунікацію з клієнтами, що дозволить підвищити стійкість до зовнішніх загроз, покращити якість обслуговування та оптимізувати внутрішні робочі процеси компанії.

Завдання для досягнення мети:

1. Аналіз поточного стану інформаційної системи бюро перекладів "InText":
   - дослідити існуючу інфраструктуру обробки замовлень та зберігання даних;
2. Формування вимог до інтегрованої хмарної системи:
   - визначити функціональні можливості системи, необхідні для стабільної роботи бюро перекладів;
   - сформувати технічні та програмні вимоги до хмарного сховища даних та Telegram-бота.
3. Розробка концепції інтеграції хмарних сервісів та Telegram-бота:
   - спроєктувати структуру та логіку взаємодії між хмарним сховищем, внутрішньою базою даних та користувачами;

- описати процеси автоматизації отримання замовлень, збереження файлів у хмарі та передачі статусу клієнтам.

4. Розробка проекту Telegram-бота для бюро перекладів:
   - створити архітектуру бота для обробки запитів клієнтів;
   - реалізувати функції прийому замовлень, прикріплення файлів, інформування клієнтів про статус їхніх замовлень та автоматизацію відповіді на типові запити.
5. Розробка хмарної системи зберігання та обробки даних:
   - описати використання хмарних платформ (наприклад, AWS або Google Cloud) для надійного зберігання документів;
   - реалізувати механізм завантаження, організації та захисту файлів.
6. Інтеграція компонентів системи:
   - здійснити технічне об'єднання Telegram-бота, хмарного сховища та інформаційної бази компанії;
   - забезпечити синхронізовану роботу компонентів для автоматичного виконання завдань.
7. Тестування і оцінка ефективності системи:
   - провести комплексне тестування системи для виявлення помилок та перевірки стабільної роботи;
   - оцінити покращення швидкості прийому та обробки замовлень, доступності даних та зручності для клієнтів.
8. Оцінка економічної доцільності впровадження системи:
   - проаналізувати витрати на розробку та впровадження хмарної системи;
   - порівняти потенційні вигоди, зокрема зменшення витрат на підтримку фізичної інфраструктури та покращення ефективності роботи.
9. Розробка рекомендацій щодо подальшого розвитку системи:
   - запропонувати додаткові функції для Telegram-бота та хмарної системи, які можуть покращити роботу компанії;
   - визначити можливі шляхи інтеграції інших технологій для оптимізації процесів бюро перекладів.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Значущість хмарних технологій та їх застосування

Хмарні технології відіграють ключову роль у сучасній організації зберігання, обробки та доступу до інформації. Вони дозволяють компаніям працювати більш ефективно та забезпечують гнучкість у масштабуванні ресурсів залежно від навантаження. У контексті бюро перекладів «InText», хмарні сервіси стали фундаментальною основою для забезпечення безперервної та надійної роботи.

Переваги хмарних технологій:

1. Надійність та доступність даних

Хмарна інфраструктура забезпечує постійний доступ до даних у реальному часі з будь-якого місця світу. Це особливо важливо для клієнтів, які знаходяться в різних часових зонах та потребують швидкого отримання готових документів. Використання Amazon Web Services (AWS) гарантує високий рівень доступності за допомогою глобальних дата-центрів.

2. Гнучке масштабування ресурсів

Завдяки хмарним обчисленням ресурси системи можна гнучко налаштовувати під змінні навантаження. У пікові періоди, коли кількість замовлень зростає, потужності можуть бути збільшені для забезпечення стабільної роботи. Після зниження навантаження ресурси можна оптимізувати, що зменшує фінансові витрати.

3. Автоматичне резервне копіювання та захист даних

AWS пропонує механізми автоматичного резервного копіювання, що забезпечує збереження інформації навіть у випадку надзвичайних ситуацій. Крім того, застосування швидкого відновлення даних дозволяє мінімізувати час простою системи.

Хмарні сервіси, які використовуються у бюро перекладів «InText»:

1. Amazon S3 (Simple Storage Service)[5]

Використовується для зберігання файлів клієнтів (вхідні тексти для перекладу, готові документи).

Забезпечує безпечний доступ до даних та їх резервне копіювання у хмарі.

Дозволяє інтегрувати завантаження файлів через Telegram-бот та веб-інтерфейс.

2. Amazon EC2 (Elastic Compute Cloud)[4]

Використовується для обробки замовлень та виконання серверних операцій.

Забезпечує високу продуктивність обчислень для генерації звітів, обробки текстів і взаємодії з базою даних.

3. MySQL у хмарному середовищі AWS RDS[6]

Використовується як основна база даних для зберігання інформації про замовлення, клієнтів та статуси проєктів.

Гарантує надійність та швидкий доступ до структурованої інформації.

4. AWS IAM (Identity and Access Management)

Забезпечує керування доступом до сервісів AWS для різних користувачів (адміністратори, співробітники тощо).

Підвищує рівень безпеки даних шляхом розмежування прав доступу.

Інтеграція з Telegram-ботом дозволяє клієнтам зручно надсилати замовлення, завантажувати файли та отримувати результати.

Автоматизація обробки замовлень через EC2 знижує навантаження на співробітників та підвищує швидкість виконання проєктів.

Синхронізація даних через S3 забезпечує безперебійний доступ до файлів як для працівників, так і для клієнтів.

2.2 Вибір оптимального середовища для комунікації

Ефективна комунікація є ключовим фактором для успішної роботи бюро перекладів «InText», оскільки вона забезпечує оперативний зв'язок з клієнтами, прийом замовлень та своєчасну доставку готових документів. Було

проаналізовано кілька основних середовищ для взаємодії з клієнтами з урахуванням їх переваг та недоліків.

Аналіз основних середовищ комунікації:

1. Email-сервіси

Переваги: Широко використовуються для офіційного листування, підтримують вкладення великих файлів та надають можливість архівації переписки.

Недоліки: Недостатня інтерактивність, затримки у відповіді, складність автоматизації процесів, що унеможливлює миттєвий обмін інформацією.

2. Відеоконференції

Переваги: Відмінно підходять для обговорення складних проєктів та специфічних деталей в реальному часі.

Недоліки: Вимагають значних часових ресурсів як від компанії, так і від клієнта, а також стабільного інтернет-з'єднання. Не завжди необхідні для стандартних завдань бюро перекладів.

3. Месенджери

Переваги:

Швидкість обміну повідомленнями та файлами.

Зручність доступу з мобільних та настільних пристроїв.

Простота автоматизації процесів за допомогою ботів та інтеграцій.

Недоліки: Потреба у виборі конкретного месенджера, що відповідає потребам компанії та її клієнтів.

2.2.1 Вибір месенджера

На основі аналізу популярних месенджерів (Viber, WhatsApp, Facebook Messenger, Telegram) для впровадження у систему комунікації бюро перекладів «InText» було обрано Telegram як оптимальне рішення. Вибір обґрунтований наступними перевагами:

1. Швидкість і функціональність

Telegram забезпечує миттєвий обмін текстовими повідомленнями, аудіо та відеофайлами. Це дозволяє оперативно отримувати замовлення від клієнтів, а також надсилати готові результати.

2. Можливість інтеграції через Telegram API

Використання Telegram API дозволяє реалізувати автоматизовану взаємодію між клієнтами та системою бюро перекладів. Це включає:

– Прийом текстових заявок на переклад.

– Завантаження та зберігання документів у хмарних сховищах (наприклад, Amazon S3).

– Автоматичне повідомлення клієнтів про статус замовлення.

3. Автоматизація за допомогою Telegram-бота

Впроваджений Telegram-бот виступає основним інструментом для комунікації з клієнтами. Основні функції бота включають:

– Збір інформації про замовлення: клієнти можуть вводити текстову інформацію про потреби в перекладі.

– Передача файлів: можливість завантажувати файли будь-якого формату для подальшої обробки.

4. Гнучкість і безпека

Telegram забезпечує високу безпеку завдяки шифруванню даних, що є важливим фактором при роботі з конфіденційною інформацією клієнтів.

5. Зручність для користувачів

Telegram підтримується на всіх популярних платформах, що спрощує взаємодію для клієнтів, які можуть користуватися ботом зі смартфонів, планшетів чи ПК.

2.3 Інтеграція Telegram-бота із хмарними сервісами

Для автоматизації роботи бюро перекладів «InText» необхідно розробити Telegram-бот, який забезпечує ефективну взаємодію з клієнтами та інтегрується із хмарними сервісами. Основною функцією Telegram-бота є прийом замовлень

від клієнтів, зокрема текстових запитів та файлів, їх обробка та подальше зберігання в хмарній інфраструктурі AWS.

Telegram-бот побудований із використанням Telegram API та бібліотеки python-telegram-bot, яка надає зручні інструменти для розробки та обробки запитів користувачів у месенджері. Після отримання замовлення бот автоматично завантажує передані файли у хмарне сховище AWS S3, де вони зберігаються у відповідних папках, структурованих за ідентифікаторами замовлень. Усі мета-дані файлів, такі як ім'я, шлях до файлу та дата завантаження, зберігаються у базі даних MySQL, що дозволяє підтримувати зв'язок між файлами та конкретним замовленням.

Окрім функції завантаження файлів, бот також дозволяє відстежувати статус замовлень у режимі реального часу. Клієнти можуть отримати актуальну інформацію про хід виконання перекладу завдяки запитам до бази даних, які обробляє серверна частина системи. Статуси замовлень, як-от «Очікує підтвердження», «В обробці» або «Готово», оновлюються адміністраторами бюро перекладів, а зміни автоматично відображаються у відповідях Telegram-бота.

Для інтеграції із хмарними сервісами використовується бібліотека boto3, яка є офіційним SDK для роботи з Amazon Web Services у Python. Завдяки цьому забезпечується надійне збереження файлів у AWS S3 та зручний доступ до них із будь-якої точки світу.

Таким чином, Telegram-бот є ключовим інструментом для автоматизації комунікації з клієнтами та управління замовленнями. Його інтеграція із хмарними сервісами дозволяє спростити робочі процеси, забезпечити надійне збереження даних та підвищити рівень обслуговування клієнтів бюро перекладів «InText».

2.4 Обґрунтування вибору бази даних

Для зберігання інформації про замовлення та їх мета-даних у комп'ютерній системі бюро перекладів «InText» треба обрати реляційну базу даних MySQL. Це

рішення ґрунтується на перевагах, які забезпечують ефективну обробку структурованих даних, підтримку транзакцій та інтеграцію з хмарними середовищами.

MySQL є однією з найбільш популярних систем управління базами даних, що відповідає вимогам сучасних застосунків. Основним критерієм вибору є підтримка ACID-транзакцій (атомарність, узгодженість, ізольованість і довговічність), що забезпечує цілісність даних навіть у випадках збою системи чи мережі. Це є важливим фактором для стабільної та надійної роботи системи замовлень.

Ще однією значною перевагою MySQL є її масштабованість. У поєднанні з хмарною платформою AWS RDS (Relational Database Service), MySQL дозволяє обробляти великий обсяг даних, масштабувати ресурси в залежності від навантаження та забезпечувати високу продуктивність. Це особливо актуально для бюро перекладів, де кількість замовлень може значно зростати під час пікових періодів.

Для зберігання файлів клієнтів, таких як документи та додаткові матеріали, використовується хмарне сховище AWS S3. У цьому випадку MySQL виконує роль сховища мета-даних, де зберігаються відомості про завантажені файли: ім'я файлу, шлях до нього у хмарі, час завантаження та зв'язок із конкретним замовленням. Такий підхід дозволяє оптимізувати роботу системи, забезпечуючи швидкий доступ до необхідних даних та стабільну роботу навіть при високих навантаженнях.

Таким чином, використання MySQL у поєднанні з AWS RDS для зберігання даних та AWS S3 для роботи з файлами клієнтів дозволяє створити гнучку, надійну та ефективну структуру бази даних. Це рішення забезпечує продуктивність, безпеку та цілісність даних, що є критично важливим для роботи бюро перекладів «InText».

2.5 Вибір мови програмування

При розробці ПО комп'ютерної системи бюро перекладів «InText» необхідно обрати мову програмування Python[7]. Це рішення обґрунтоване рядом вагомих переваг, які роблять Python оптимальним вибором для реалізації проєкту.

Основною причиною вибору Python є її простота та читабельність синтаксису. Завдяки лаконічному коду розробка стала більш швидкою та зрозумілою, а кількість можливих помилок була мінімізована. Python дозволяє ефективно реалізовувати необхідний функціонал, що є важливим для оптимізації робочих процесів у компанії.

Крім того, Python має широкий набір бібліотек та інструментів, які значно полегшують інтеграцію з хмарними сервісами та сторонніми API. У межах даного проєкту було використано бібліотеку python-telegram-bot для розробки Telegram-бота та взаємодії з Telegram API. Для інтеграції з хмарними сервісами Amazon Web Services (AWS) було застосовано boto3 — офіційний SDK для роботи з AWS. Зокрема, ця бібліотека дозволяє легко взаємодіяти з хмарним сховищем S3 для зберігання файлів клієнтів та іншими сервісами, такими як RDS для роботи з базою даних MySQL.

Додатковою перевагою Python є його універсальність та широка підтримка спільноти розробників. Наявність великої кількості документації та навчальних ресурсів спростила процес розробки та вирішення технічних завдань. Мова Python забезпечує високу інтеграцію з сучасними технологіями та дозволяє створювати системи, що відповідають вимогам масштабованості та продуктивності.

Для написання коду було обрано середовище розробки Visual Studio Code. Воно забезпечує зручність роботи завдяки підтримці плагінів для Python, можливості налагодження коду, інтеграції з системами контролю версій, такими як Git, та оптимізованій роботі з різними модулями.

Використання Python дозволило досягти необхідної функціональності, забезпечити надійну інтеграцію між Telegram-ботом, хмарними сервісами та базою даних. Це, своєю чергою, сприяло створенню системи, яка відповідає

вимогам автоматизації процесів бюро перекладів «InText», забезпечуючи ефективність, гнучкість та надійність роботи.

2.6 Економічне обґрунтування використання хмарних сервісів

На таблиці 2.1 показано оцінку вартості[3] хмарних сервісів і супутніх послуг для підтримки системи, заснованої на Django.

Таблиця 2.1

Сервіс	Призначення	Вартість на місяць (USD)	Обґрунтування
Amazon EC2	Сервер для запуску Django-системи	20-50	Вартість залежить від типу інстансу (t2.micro для тестів або t3.medium для продакшн).
Amazon RDS	База даних MySQL	30-80	Залежить від розміру бази та обчислювальних ресурсів (db.t2.micro або db.t3.medium).
Amazon S3	Хмарне сховище для файлів	120	Вартість залежить від обсягу збережених файлів (0.023 USD за 1 ГБ на місяць). Візмемо ціну за 5тб
Telegram API	Інтеграція для Telegram-бота	Безкоштовно	Telegram API надається безкоштовно, тільки сервер несе витрати на обробку запитів.
Домен і SSL-сертифікат	Підтримка веб-сайту та безпечного з'єднання	10-15	Середня ціна домену та сертифіката на рік становить приблизно 120-150 USD.
Адміністрування системи	Підтримка хмарної інфраструктури	150-300	Витрати на зарплату системного адміністратора чи DevOps-інженера.
Загальна вартість		330-565	

2.7 Використання Amazon EC2 для обчислювальних ресурсів

Для забезпечення обробки замовлень та розгортання серверної частини системи необхідно обрати сервіс Amazon EC2 (Elastic Compute Cloud). EC2 є віртуальним сервером у хмарі, що забезпечує необхідну продуктивність, масштабованість та надійність роботи. Основними перевагами використання

Amazon EC2 для бюро перекладів «InText» є можливість гнучкого налаштування обчислювальних ресурсів залежно від навантаження, автоматизація розгортання додатків та централізоване управління. Сервер EC2 розміщує серверний додаток, який обробляє замовлення, завантаження файлів у хмару, а також забезпечує інтеграцію з Telegram-ботом та базою даних MySQL. Завдяки використанню EC2 гарантовано стабільну роботу системи та її доступність для клієнтів у режимі 24/7.

2.8 Використання Amazon S3 для зберігання файлів

Для надійного зберігання файлів клієнтів було обрано сервіс Amazon S3 (Simple Storage Service), що є одним із провідних рішень для хмарного зберігання даних. Amazon S3 забезпечує зручне завантаження, доступ та захист файлів завдяки гнучким політикам безпеки та масштабованості. Файли замовлень, завантажені через Telegram-бот, автоматично зберігаються у відповідних сховищах S3, що дозволяє надійно організувати їх зберігання та доступ. Окрім цього, S3 підтримує автоматичне резервне копіювання, що мінімізує ризик втрати даних. Інтеграція з іншими сервісами AWS, такими як EC2 та RDS, дозволяє створити централізовану та ефективну інфраструктуру для бюро перекладів «InText».

2.9 Висновки до теоретичного розділу

У цьому розділі було проведено детальний аналіз існуючих проблем, з якими стикається бюро перекладів «InText», та запропоновано ефективне рішення на основі сучасних технологічних інструментів. В якості основи для побудови системи були обрані хмарні сервіси AWS, які забезпечують надійне зберігання та обробку файлів у хмарному середовищі. Використання Telegram-бота дозволяє автоматизувати комунікацію з клієнтами, значно спрощуючи процес прийому замовлень, інформування про статуси та надання необхідної інформації в режимі реального часу. Для зберігання структурованих даних і

метаінформації було обрано базу даних MySQL, яка забезпечує цілісність, масштабованість і ефективну роботу з даними в поєднанні з хмарними сервісами.

Запропонована система поєднує надійність хмарних технологій, гнучкість Telegram як платформи для інтерактивного спілкування, а також продуктивність та стабільність бази даних MySQL. Це рішення дозволяє бюро перекладів «InText» забезпечити безперебійний доступ до інформації, автоматизувати ключові робочі процеси та мінімізувати ризики, пов’язані з відключенням електроенергії або фізичним переміщенням обладнання. У результаті реалізації запропонованої системи значно підвищується якість обслуговування клієнтів, оптимізується робота компанії та забезпечується її стабільна діяльність у будь-яких умовах.

3 СИНТЕЗ СИСТЕМИ

3.1 Постановка технічних вимог до комп’ютерної системи в цілому

3.1.1 Вимоги до реалізації системи

Комп'ютерна система бюро перекладів повинна забезпечувати надійну роботу, збереження та обробку даних, а також зручну взаємодію між клієнтами та працівниками. Основними вимогами є висока продуктивність, що дозволяє обробляти велику кількість запитів одночасно, а також надійність для забезпечення стабільного доступу до інформації у будь-яких умовах. Безпека системи має бути на найвищому рівні, з можливістю резервного копіювання даних і захистом від несанкціонованого доступу. Для ефективної роботи потрібна можливість гнучкого масштабування інфраструктури відповідно до зростаючих потреб компанії, а також швидкий доступ до ресурсів системи з будь-якої точки світу. Важливим є забезпечення автоматизації робочих процесів для скорочення часу обробки замовлень та підвищення якості обслуговування клієнтів.

3.1.2 Вимоги до функцій що впроваджується

Система повинна забезпечувати низку ключових функцій для ефективної роботи бюро перекладів. Основним функціоналом є прийом замовлень від клієнтів, обробка надісланих файлів та збереження їх у надійному середовищі. Важливою є можливість автоматичної зміни статусу замовлень залежно від етапу їх обробки, а також інформування клієнтів про хід виконання роботи. Система повинна підтримувати швидку комунікацію між користувачами та співробітниками бюро за допомогою інтерактивних засобів, що забезпечують миттєву передачу текстових повідомлень та файлів. Окрім цього, система має забезпечувати можливість доступу до замовлень у зручному інтерфейсі як для клієнтів, так і для адміністраторів, які мають право на управління замовленнями та перевірку їх стану. Інтеграція із зовнішніми платформами повинна бути безшовною для оптимізації процесів і підвищення продуктивності.

3.1.3 Показники призначення

Основними показниками є швидкість обробки замовлень, яка забезпечується автоматизацією прийому заявок та збереженням файлів у надійному середовищі. Важливим критерієм є точність і надійність виконання завдань, що гарантує збереження даних клієнтів та їх оперативну обробку без втрат. Система повинна демонструвати високу доступність і стабільність роботи, незалежно від зовнішніх умов, забезпечуючи безперервний доступ до інформації як для клієнтів, так і для співробітників бюро.

Ефективність комунікації оцінюється швидкістю передачі повідомлень та зворотного зв’язку, що дозволяє мінімізувати час взаємодії між клієнтами та виконавцями. Важливим показником є зручність користування системою завдяки інтуїтивному інтерфейсу та функціональності, що спрощує навігацію для користувачів різного рівня підготовки. Крім того, система повинна забезпечувати масштабованість, дозволяючи обробляти більшу кількість замовлень без зниження продуктивності.

Безпека даних також є ключовим показником: захист інформації клієнтів, автоматичне резервне копіювання та контроль доступу є невід'ємними складовими. У підсумку, впровадження системи має підвищити загальну ефективність роботи бюро перекладів, скоротити час обробки замовлень, знизити ризики втрати інформації та покращити рівень обслуговування клієнтів.

3.1.4 Вимоги до надійності

Система повинна мати механізми автоматичного резервного копіювання даних, що дозволяє оперативно відновити інформацію у випадку непередбачених обставин, таких як технічні збої або пошкодження даних. Крім того, важливим є контроль цілісності даних, що забезпечує їх захист від несанкціонованого доступу, втрат або помилок під час обробки.

Стійкість до зовнішніх факторів, таких як перебої в електропостачанні або нестабільний доступ до інтернету, є ключовим показником надійності. Для цього передбачається використання хмарних технологій, що дозволяють системі

працювати незалежно від фізичної інфраструктури та забезпечують безперервний доступ до даних з будь-якої точки світу.

Система повинна бути здатна до автоматичного моніторингу та виявлення помилок, що дозволяє своєчасно реагувати на потенційні проблеми та мінімізувати час простою. Крім того, важливою є можливість масштабування без втрати продуктивності, щоб забезпечити стабільну роботу при зростанні навантаження на систему.

3.1.5 Вимоги до програмного забезпечення

Вимоги до програмного забезпечення системи бюро перекладів передбачають ефективну роботу, надійність та можливість масштабування. Програмне забезпечення має мати модульну структуру, що дозволяє легко додавати нові функції та інтегрувати додаткові сервіси без значних змін у системі. Важливим є забезпечення кросплатформенності для доступу з різних пристроїв та операційних систем.

Інтерфейс має бути інтуїтивно зрозумілим і зручним для клієнтів та співробітників. Програмне забезпечення повинно забезпечувати роботу з базами даних для швидкого збереження, пошуку та обробки інформації, а також інтеграцію з хмарними сховищами для збереження файлів клієнтів.

Необхідна підтримка безпечної інтеграції з зовнішніми сервісами для автоматизації процесів і засобів аутентифікації користувачів для контролю доступу. Система має бути надійною, функціонувати без збоїв та мати інструменти для моніторингу продуктивності й логування подій. Це забезпечує стабільну роботу, безпеку даних та можливість подальшого розвитку системи.

3.1.6 Вимоги до захисту інформації

Вимоги до захисту інформації системи бюро перекладів передбачають забезпечення конфіденційності, цілісності та доступності даних. Захист інформації має охоплювати як збереження, так і передачу даних, забезпечуючи їх безпеку від несанкціонованого доступу чи втрати.

Для цього необхідно впровадити механізми аутентифікації користувачів, що гарантують доступ до системи лише авторизованим особам. Використання шифрування даних як під час збереження у базах даних, так і при передачі через мережу є критично важливим для запобігання перехопленню інформації.

Доступ до файлів клієнтів, що зберігаються у хмарних сховищах, повинен контролюватися через політики прав доступу, а резервне копіювання даних забезпечить їх відновлення у разі непередбачуваних збоїв. Також важливо налаштувати моніторинг подій та логування дій користувачів для своєчасного виявлення та реагування на загрози безпеці.

Таким чином, система має забезпечувати захист інформації на всіх рівнях роботи, дотримуючись сучасних стандартів безпеки для надійного функціонування бюро перекладів.

3.2 Структура комп’ютерної системи

Взаємодія компонентів системи побудована на основі хмарної інфраструктури AWS. Telegram-бот використовується для прийому замовлень від клієнтів та передачі даних на EC2-сервер, де обробляється логіка системи. Файли клієнтів завантажуються у Amazon S3, а мета-дані замовлень зберігаються в базі даних Amazon RDS (MySQL). Веб-інтерфейс забезпечує адміністрування, моніторинг замовлень та їх створення синхронізуючись із сервером та базою даних. Усі компоненти працюють разом для забезпечення стабільної роботи системи та ефективної автоматизації бюро перекладів. Структуру можна побачити на рисунку 3.1.

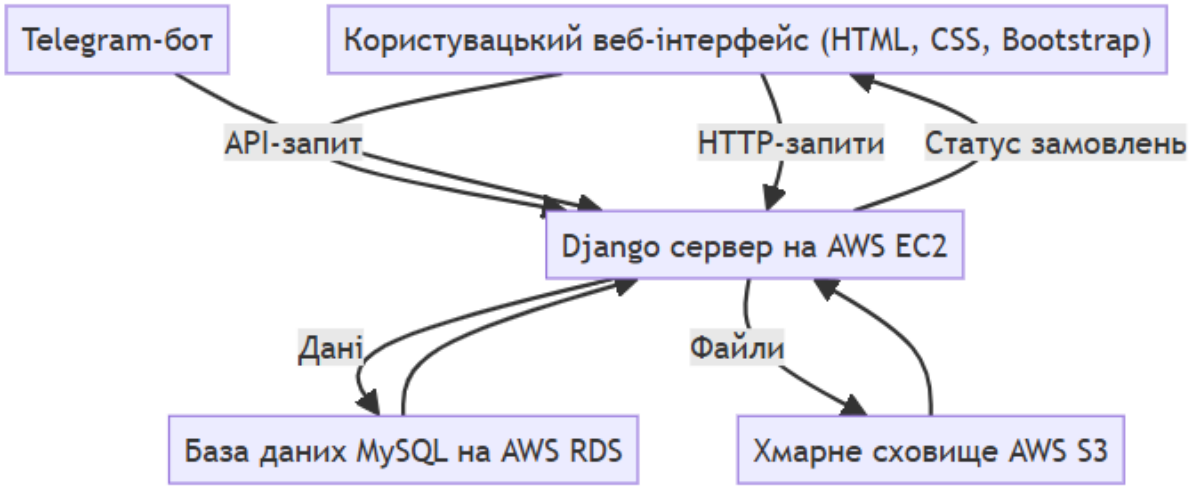


Рисунок 3.1 – Структура комп’ютерної системи

3.3 Структура бази даних

У проекті бюро перекладів «InText» реалізовано реляційну базу даних, що складається з декількох таблиць для збереження інформації про користувачів, замовлення та службові дані системи. База даних структурована таким чином, щоб забезпечити ефективну обробку даних, зберігання їх цілісності та зв'язність між елементами. В основі системи лежать таблиці auth_user, orders_order, а також допоміжні таблиці для реалізації автентифікації, управління сесіями та ролями користувачів.

Загальна структура бази даних представлена у вигляді пов'язаних таблиць, де основні зв'язки формуються через зовнішні ключі. Ця база даних предствлена на рисунку 3.2.

У проєкті передбачено такі таблиці:

auth_user – таблиця, що зберігає дані користувачів, включаючи ідентифікатор, ім'я, електронну пошту та хеш пароля.

orders_order – ключова таблиця для обробки замовлень, у якій зберігається інформація про текст замовлення, дату створення, статус і користувача, що його створив.

auth_group та auth_permission – таблиці, що реалізують механізм ролей та дозволів для користувачів.

Зв’язкові таблиці – auth_user_groups та auth_user_user_permissions для об'єднання користувачів з групами та дозволами.

django_migrations – службова таблиця для керування міграціями у структурі бази даних.

django_session – таблиця для зберігання сесій користувачів.

Загальна схема(рис. 3.2) наочно демонструє зв'язки між таблицями:

orders_order.user_id є зовнішнім ключем, що посилається на auth_user.id, забезпечуючи зв'язок між замовленнями та користувачами.

auth_user_groups реалізує багато-до-багатьох зв'язок між користувачами та групами через зовнішні ключі на auth_user.id та auth_group.id.

auth_user_user_permissions зв'язує користувачів із конкретними дозволами, спираючись на таблицю auth_permission.

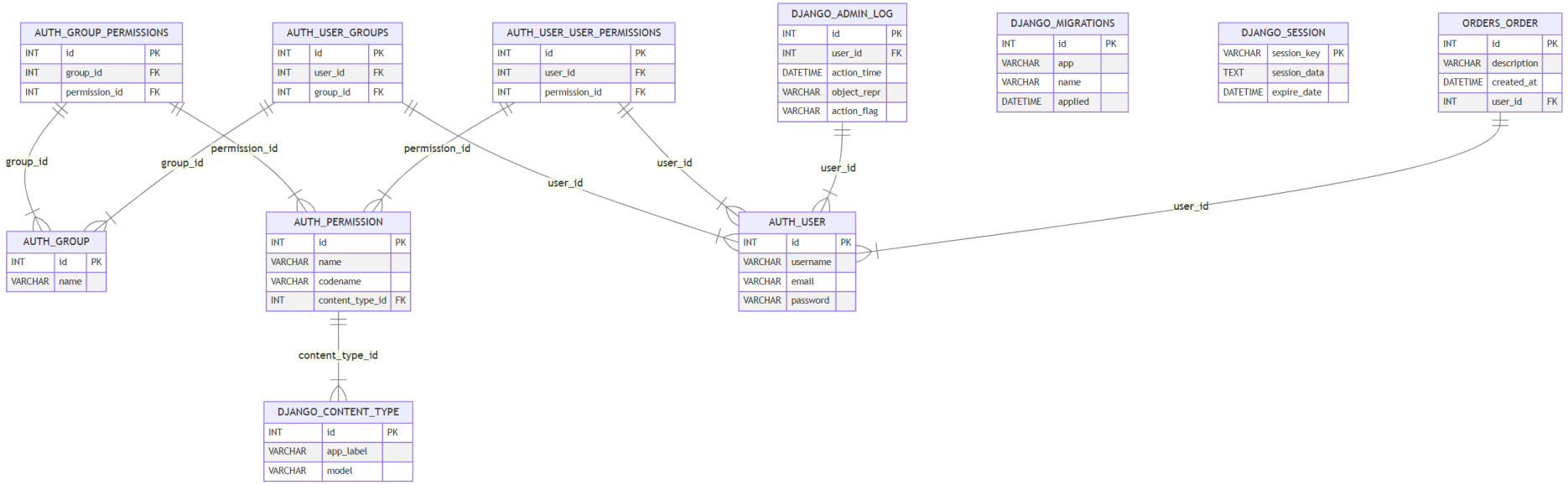


Рисунок 3.2 – Загальна структура бази даних

Детальний опис таблиць

Таблиця auth_user(рис. 3.3)

Призначення: Основна таблиця для зберігання даних користувачів.

Поля:

- id – унікальний ідентифікатор користувача.
- username – унікальне ім'я користувача.
- email – електронна пошта користувача.
- password – хеш пароля користувача.
- date_joined – дата реєстрації користувача.

Зв'язки:

Зовнішній ключ у таблиці orders_order через поле user_id.

AUTH_USER			
INT	id	PK	
VARCHAR	password		
DATETIME	last_login		
TINYINT	is_superuser		
VARCHAR	username		UNIQUE
VARCHAR	first_name		
VARCHAR	last_name		
VARCHAR	email		
TINYINT	is_staff		
TINYINT	is_active		
DATETIME	date_joined		

Рисунок 3.3 – таблиця auth_user

Таблиця orders_order(рис. 3.4)

Призначення: Зберігає інформацію про замовлення, створені користувачами.

Поля:

- id – унікальний ідентифікатор замовлення.
- user_id – зовнішній ключ на таблицю auth_user, що вказує на користувача, який створив замовлення.

– text – текстове поле для опису замовлення.

– created_at – дата та час створення замовлення.

– status – статус замовлення (наприклад, «в обробці», «виконано»).

Зв'язки:

Зв'язок з auth_user для ідентифікації замовника.

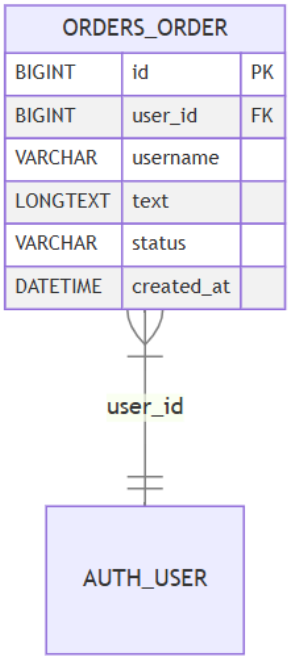


Рисунок 3.4 – таблиця orders_order

Таблиця auth_group(рис. 3.5)

Призначення: Реалізує систему груп для об'єднання користувачів за певними ролями.

Поля:

– id – унікальний ідентифікатор групи.

– name – ім'я групи.

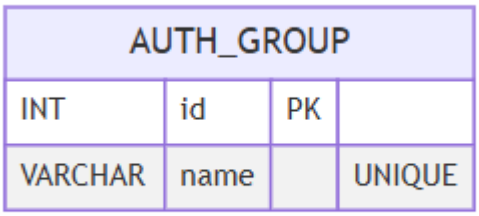


Рисунок 3.5 – таблиця auth_group

Таблиця auth_permission(рис. 3.6)

Призначення: Зберігає права доступу користувачів.

- Поля:
- id – унікальний ідентифікатор дозволу.
- name – ім'я дозволу.

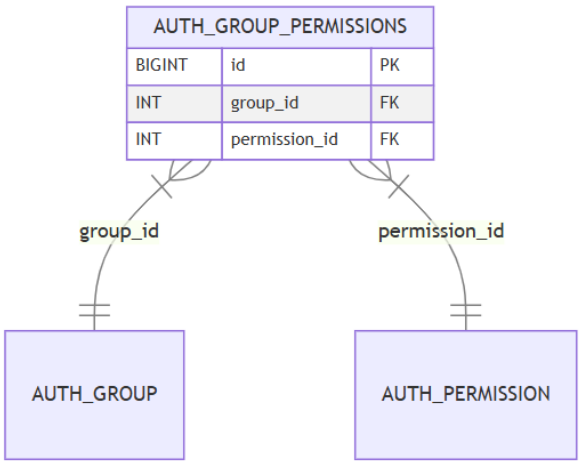


Рисунок 3.6 – таблиця auth_permission

Таблиця auth_user_groups (рис. 3.7)

Призначення: Зв'язкова таблиця для об'єднання користувачів у групи.

Поля:

- id – унікальний ідентифікатор запису.
- user_id – зовнішній ключ на auth_user.
- group_id – зовнішній ключ на auth_group.

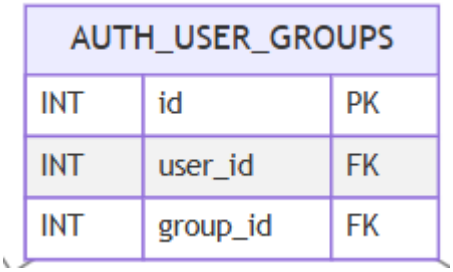


Рисунок 3.7 – таблиця auth_user_groups

Таблиця auth_user_user_permissions (рис. 3.8)

Призначення: Реалізує індивідуальні права доступу для користувачів.

Поля:

- id – унікальний ідентифікатор запису.
- user_id – зовнішній ключ на auth_user.
- permission_id – зовнішній ключ на auth_permission.

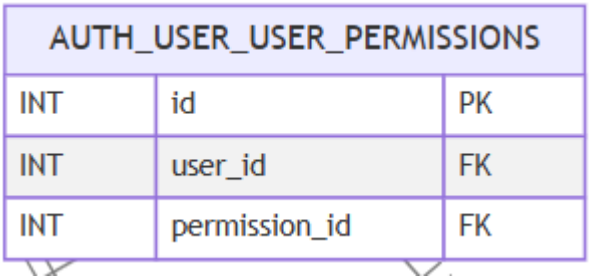


Рисунок 3.8 – таблиця auth_user_user_permissions

Таблиця django_migrations (рис. 3.9)

Призначення: Службова таблиця для зберігання міграцій.

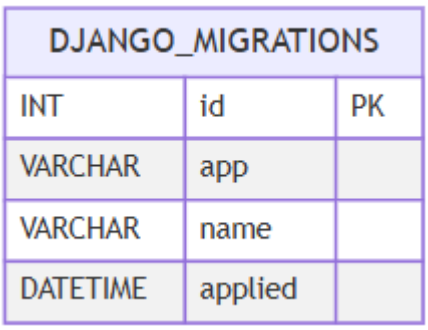


Рисунок 3.9 – таблиця django_migrations

Таблиця django_session (рис. 3.10)

Призначення: Використовується для збереження сесій користувачів.

DJANGO_SESSION		
VARCHAR	session_key	PK
TEXT	session_data	
DATETIME	expire_date	

Рисунок 3.10 – таблиця django_session

3.4 Вибір елементної бази комп'ютерної системи

Для побудови комп'ютерної системи бюро перекладів «InText» було обрано інфраструктуру на основі сервісів Amazon Web Services (AWS). Кожен елемент AWS був налаштований відповідно до вимог продуктивності, надійності та масштабованості, що дозволяє ефективно виконувати всі функції системи.

1. EC2 (Elastic Compute Cloud)

Для розміщення веб-додатку та логіки Telegram-бота було обрано віртуальний сервер на EC2. Основні параметри конфігурації включають:

- тип інстансу: t2.micro – оптимальний для малих навантажень з мінімальними витратами;
- операційна система: Ubuntu Server 20.04 LTS, що забезпечує стабільність і підтримку необхідного програмного середовища;
- сховище: 10 ГБ SSD (Elastic Block Store), що достатньо для операційної системи та базових файлів веб-додатку;
- доступність: Налаштовано автоматичний моніторинг стану інстансу через CloudWatch для швидкого виявлення та усунення помилок.

2. S3 (Simple Storage Service)

Хмарне сховище для зберігання файлів клієнтів було налаштовано з такими параметрами:

- storage Class: Standard – забезпечує високу швидкість доступу до даних;
- політики доступу: Налаштовано обмежений доступ до сховища через IAM (Identity and Access Management), що забезпечує безпеку завантаження файлів;
- версіонування: Включено для збереження попередніх версій файлів у разі випадкових змін або видалення;
- резервне копіювання: Автоматичне копіювання даних для забезпечення надійності.

3. RDS (Relational Database Service)

Для зберігання мета-інформації замовлень використовується база даних MySQL на платформі RDS з такими налаштуваннями:

- тип інстансу: db.t2.micro, що відповідає потребам невеликого навантаження;
- сховище: 20 ГБ SSD для швидкої обробки запитів.
- автоматичне резервне копіювання: Включено для забезпечення відновлення даних у разі помилки.
- захист даних: Доступ до бази обмежений через групи безпеки (Security Groups), що забезпечує безпеку з’єднань.

3.5 Висновки розділу синтезу системи

У цьому розділі було сформульовано основні вимоги до комп’ютерної системи бюро перекладів «InText» та обґрунтовано її архітектуру. Запропонована система базується на використанні хмарних технологій AWS для зберігання та обробки даних, інтеграції з Telegram-ботом для автоматизації комунікації та бази даних MySQL для збереження мета-інформації замовлень.

Розглянуто принципи побудови функціональної схеми, технічні вимоги до продуктивності, надійності та безпеки системи. Вибір Python та фреймворку Django забезпечив швидку розробку та інтеграцію з хмарними сервісами й API.

Розроблена система поєднує автоматизацію обробки замовлень, ефективне збереження даних та зручну комунікацію з клієнтами, що дозволяє підвищити якість обслуговування та оптимізувати роботу бюро перекладів.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ХМАРНИХ СЕРВІСІВ ТА TELEGRAM БОТУ

4.1 Розробка основних компонентів

Процес розробки системи бюро перекладів «InText» можна умовно поділити на дві основні складові: створення функціонального серверного середовища та налаштування інтеграції з користувацькими інтерфейсами та хмарними сервісами.

Цей процес включає два ключові етапи:

– розробка серверної частини системи на основі фреймворку Django;

– налаштування інтеграцій з AWS-сервісами та Telegram-ботом для взаємодії з користувачами.

4.1.1 Опис архітектури

Система складається з наступних компонентів:

1. Telegram-бот – забезпечує прийом замовлень від користувачів та передачу їх до системи;

2. Серверна частина – розроблена на основі Django та відповідає за обробку даних, взаємодію з базою даних та хмарними сервісами;

3. Хмарне сховище AWS S3 – використовується для зберігання файлів замовлень клієнтів;

4. База даних AWS RDS (MySQL) – зберігає інформацію про замовлення, статуси та дані користувачів;

5. EC2 сервер – хостить серверну частину та забезпечує її доступність.

4.1.2 Використані бібліотеки та технології

Для реалізації основних функцій системи були використані наступні бібліотеки та технології:

– Django – фреймворк для побудови серверної частини додатку;

– python-telegram-bot – бібліотека для розробки та інтеграції Telegram-бота;

– Boto3 – бібліотека для взаємодії з сервісами AWS, такими як S3 та RDS;

– MySQL – система керування базами даних для збереження структурованої інформації;

– Gunicorn та Nginx – для обробки запитів та розгортання серверної частини системи.

4.1.2.1 Django

Django є основним фреймворком для серверної частини системи. Його функції включають:

– обробка HTTP-запитів: забезпечує обробку GET та POST-запитів від клієнтів і Telegram-бота;

– робота з базою даних: завдяки ORM (Object-Relational Mapping), Django дозволяє легко взаємодіяти з базою даних MySQL для створення, читання, оновлення та видалення даних;

– реалізація бізнес-логіки: керування замовленнями, статусами та іншими даними системи;

– робота з формами: автоматичне створення та обробка форм для введення даних користувачами;

– аутентифікація та авторизація: надає засоби для контролю доступу користувачів до функцій системи.

Налаштування цього фреймворку подано в Додатку А.

4.1.2.2 python-telegram-bot

Бібліотека python-telegram-bot використовується для розробки та інтеграції Telegram-бота. Основні функції:

– прийом текстових повідомлень та файлів: бот обробляє введені клієнтом дані та завантажує їх у систему;

– відповіді клієнтам: автоматичне надсилання підтверджень та статусів замовлень через API Telegram;

– взаємодія з серверною частиною: передача отриманих даних до серверної логіки на основі Django;

– обробка команд: реалізація команд для зручності використання бота.

Налаштування цієї бібліотеки подано в Додатку А

4.1.2.3 Boto3

Boto3 є офіційною бібліотекою AWS SDK для Python. Вона використовується для інтеграції хмарних сервісів AWS, зокрема:

– AWS S3: завантаження файлів замовлень у хмарне сховище та отримання доступу до них через URL;

– AWS RDS: підключення до бази даних MySQL, виконання запитів для збереження та обробки даних;

– моніторинг ресурсів: забезпечення взаємодії із хмарною інфраструктурою для стабільної роботи сервісу.

4.1.2.4 Gunicorn та Nginx

Gunicorn: відповідає за обробку запитів до серверної частини на основі Django, забезпечуючи швидке виконання програмного коду.

Nginx: працює як веб-сервер і зворотний проксі, обробляючи зовнішні запити від клієнтів і передаючи їх до Gunicorn.

4.1.2.5 MySQL

MySQL використовується як основна база даних для зберігання інформації про замовлення та користувачів. Основні функції включають:

– збереження даних: зберігання метаданих замовлень, статусів та користувацької інформації;

– швидкий пошук та вибірка: ефективне виконання SQL-запитів для отримання потрібних даних;

– забезпечення цілісності: дотримання транзакційності та ACID-властивостей для надійності даних.

4.2 Опис розробленої програми

4.2.1 Загальні відомості

4.2.1.1 Найменування програми

Програма InText Translation System розроблена для автоматизації процесів бюро перекладів «InText». Вона поєднує зручний веб-інтерфейс для взаємодії з клієнтами, серверну частину для обробки замовлень і хмарну інфраструктуру для зберігання та обробки даних.

4.2.1.2 Функціональне призначення

Основна функція програми полягає в забезпеченні надійного та автоматизованого середовища для прийому замовлень, зберігання файлів і надання клієнтам актуальної інформації про статус перекладів. Програма також інтегрована з Telegram-ботом, що дозволяє клієнтам швидко створювати замовлення та отримувати сповіщення. Основні функції включають:

- створення замовлень через веб-інтерфейс або Telegram-бот;
- збереження файлів у хмарному сховищі;
- відстеження статусу замовлень;
- автоматичне резервування даних.

4.2.1.3 Структура веб-інтерфейсу комп’ютерної системи з інтеграцією Telegram-бота

Веб-інтерфейс та Telegram-бот є ключовими компонентами комп’ютерної системи бюро перекладів, які взаємодіють між собою та з іншими елементами

інфраструктури для забезпечення автоматизації та ефективності роботи. Це можна побачити на рисунку 3.1.

1. Клієнтська частина (Frontend)

Реалізована за допомогою HTML, CSS та JavaScript, клієнтська частина веб-інтерфейсу забезпечує користувачам доступ до функцій системи через зручний інтерфейс. Основні елементи включають:

- форма створення замовлення – дозволяє користувачам вводити текст заявки та завантажувати файли для обробки;
- сторінка статусу замовлень – показує користувачу поточний статус обробки його заявки, синхронізуючись із базою даних;
- панель адміністратора – надає адміністраторам доступ до перегляду заявок, зміни їх статусів та завантаження файлів.

2. Серверна частина (Backend)

Серверна логіка розроблена на Django з використанням Python. Основні завдання включають:

- обробка даних із веб-форми – збереження тексту та файлів замовлень у базі даних і хмарному сховищі;
- взаємодія з хмарними сервісами – завантаження файлів на Amazon S3 та запис мета-даних у Amazon RDS (MySQL);
- API для Telegram-бота – забезпечує комунікацію між сервером та ботом для автоматичної обробки запитів користувачів.

3. Telegram-бот

Telegram-бот є додатковим інтерфейсом для комунікації користувачів із системою. Основні функції бота включають:

- прийом замовлень – користувач може надіслати текстове замовлення та завантажити файл безпосередньо через Telegram;
- перевірка статусу замовлень – бот запитує дані із бази даних та повертає актуальну інформацію про поточний статус заявки;

– реалізація здійснюється за допомогою python-telegram-bot для розробки логіки та інтеграції через Telegram API.

4. База даних

Основна база даних системи – Amazon RDS (MySQL), яка зберігає:

– дані про користувачів і їх замовлення;
– статуси замовлень та мета-дані файлів;
– історію взаємодії з Telegram-ботом.

5. Інтеграція з хмарними сервісами

Amazon S3 – зберігання файлів, які завантажують користувачі як через веб-інтерфейс, так і через Telegram-бота.

Amazon EC2 – хостинг серверної частини системи, що забезпечує стабільну роботу як веб-інтерфейсу, так і Telegram-бота.

4.2.1.4 Виклик та завантаження

Взаємодія програми відбувається через запити від клієнта до сервера за допомогою HTTP/HTTPS. Клієнтська частина надсилає запити через веб-інтерфейс або Telegram-бот, сервер обробляє ці запити та взаємодіє з хмарною інфраструктурою для збереження файлів і оновлення статусів. Відповіді повертаються у вигляді веб-сторінок або повідомлень у Telegram, забезпечуючи зворотний зв'язок із користувачем.

4.2.1.5 Логіка взаємодії з користувачем

Логіка взаємодії з користувачем у розробленій системі бюро перекладів «InText» побудована на принципах простоти, доступності та ефективності. Система надає можливість клієнтам та співробітникам здійснювати основні операції як через веб-інтерфейс, так і через Telegram-бот.

Клієнти можуть взаємодіяти із системою наступним чином:

1. Створення замовлення: Користувач заповнює форму на веб-сайті або відправляє текстове повідомлення через Telegram-бот. У разі потреби додає файли, які завантажуються до хмарного сховища AWS S3.
2. Отримання статусу замовлення: Після обробки замовлення система надсилає оновлення через веб-інтерфейс або повідомлення у Telegram.
3. Отримання результату: Завершені файли перекладу доступні для завантаження за посиланням, що надсилається клієнту у відповідь.

Співробітники бюро перекладів взаємодіють із системою таким чином:

1. Перевірка та управління новими замовленнями через адміністративну панель веб-додатка.
2. Оновлення статусу замовлень після кожного етапу виконання роботи.
3. Завантаження готових файлів через веб-інтерфейс та їх відправлення клієнтам.

Адміністратор системи має розширені права доступу для моніторингу продуктивності програми, управління користувачами, налаштування серверів та бази даних, а також для інтеграції додаткових функціональних можливостей.

Завдяки поєднанню веб-інтерфейсу та Telegram-бота користувач отримує гнучкість у виборі середовища взаємодії з програмою, а співробітники бюро – ефективний інструмент для оптимізації робочих процесів.

4.2.2 Опис команд Telegram-бота та їх функціональності

В боті використову.ться наступні помнади та кнопки:

- /start;
- /help;
- /status;
- Створити замовлення
- Перевірити статус

Команда /start(рис. 4.1)

Призначення: Ініціалізує роботу користувача з ботом.

Логіка роботи:

Після запуску команди користувачу відображається меню з кнопками:

– Створити замовлення

– Перевірити статус

Результат: Зручний інтерфейс для навігації з вибором дій через кнопки.

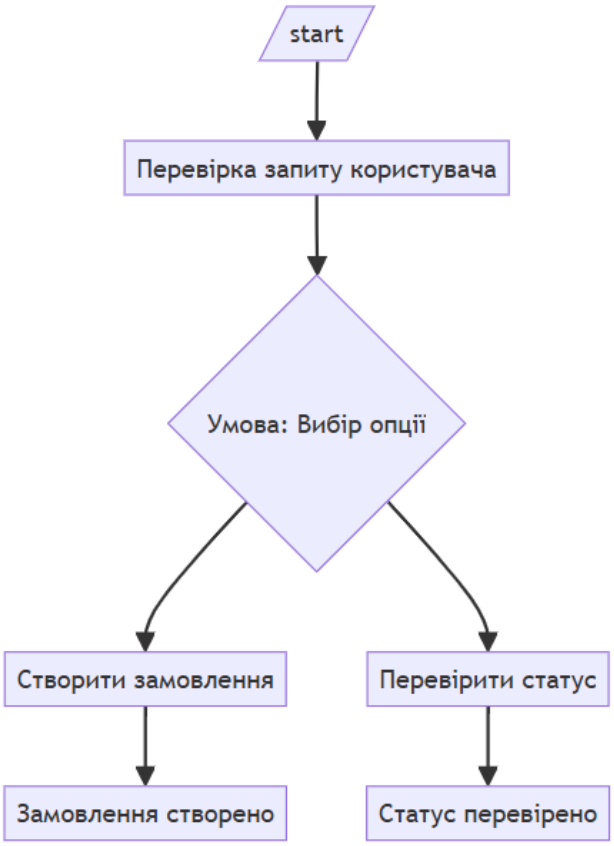


Рисунок 4.1 – Схема команди /start

Команда /help(рис. 4.2)

Призначення: Надання користувачу списку доступних команд і пояснення, як працювати з ботом.

Логіка роботи:

Бот відповідає текстовим повідомленням зі списком доступних команд та інструкціями для взаємодії.

Результат: Користувач отримує вичерпну інформацію про можливості бота.

Рисунок 4.2 – Схема команди /help

Функція Створити замовлення(рис. 4.3)

Призначення: Дозволяє користувачу створити нове замовлення, надіславши текст або файл.

Логіка роботи:

Користувач вибирає опцію "Створити замовлення".

Якщо користувач надсилає текст:

– Бот зберігає текст у базу даних як нове замовлення зі статусом "Очікує підтвердження".

Якщо користувач надсилає файл:

– Файл завантажується на AWS S3.
– Бот зберігає посилання на файл у базу даних.
– Повідомлення користувачу про успішне створення замовлення та його номер.

Результат: Створене замовлення у базі даних із текстом або файлом, доступним через хмарне сховище.

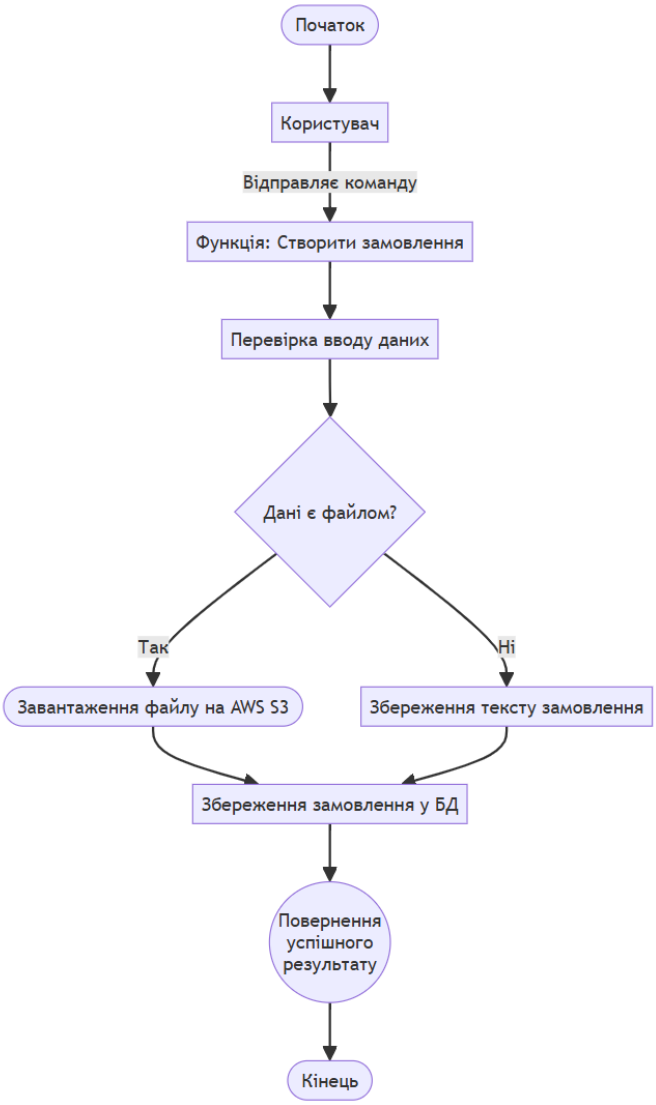


Рисунок 4.3 – Схема функції створення замовлення

Функція Перевірити статус(рис. 4.4).

Призначення: Надання користувачу можливості дізнатись статус його замовлення.

Логіка роботи:

Користувач обирає опцію "Перевірити статус"

Бот просить ввести номер замовлення.

Після введення номера бот перевіряє статус замовлення у базі даних і надсилає відповідь.

Результат: Користувач отримує актуальну інформацію про статус свого замовлення.

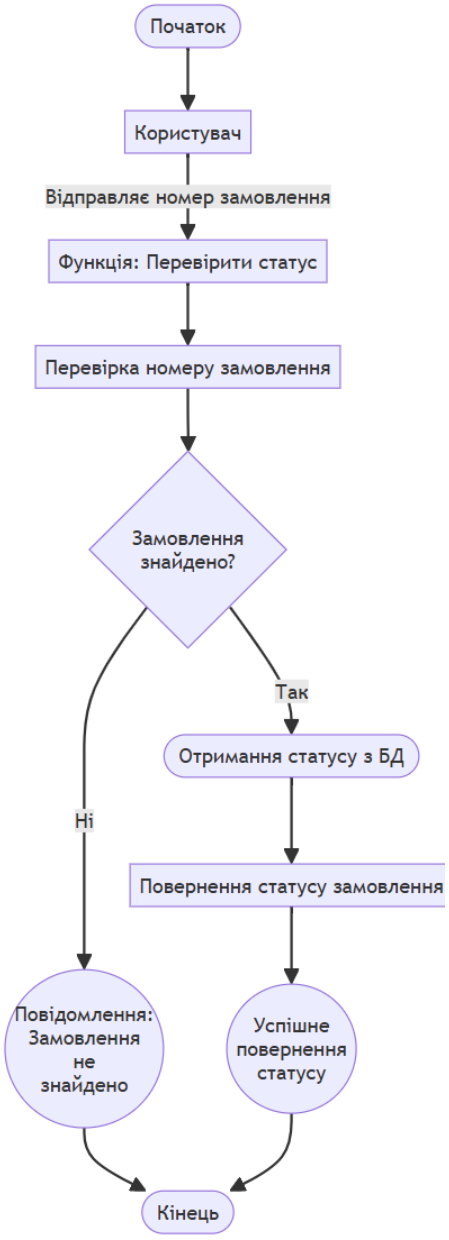


Рисунок 4.4 – Схема функції перевірки статусу

Команда /status(рис. 4.5)

Призначення: Окремий спосіб перевірки статусу замовлення за його номером.

Логіка роботи:

Користувач вводить /status разом із номером замовлення (наприклад: /status 123).

Бот перевіряє у базі даних статус замовлення та повертає відповідь.

Результат: Швидка перевірка статусу замовлення за номером.

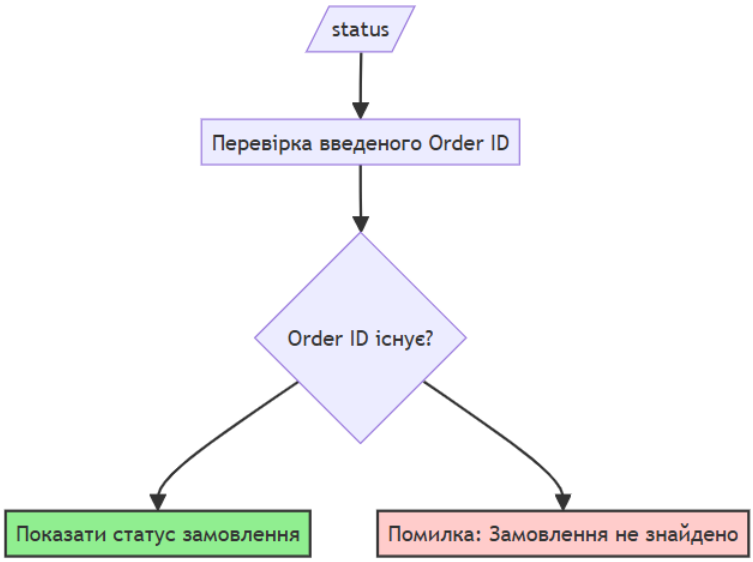


Рисунок 4.5 – Схема команди /status

Функція обробки текстових повідомлень

Призначення: Обробка текстових команд або надсилання файлів поза основними командами.

Логіка роботи:

Якщо користувач відправляє текст або файл, бот обробляє повідомлення залежно від поточного стану (створення замовлення, перевірка статусу тощо).

Результат: Гнучка взаємодія з користувачем для обробки непередбачених ситуацій.

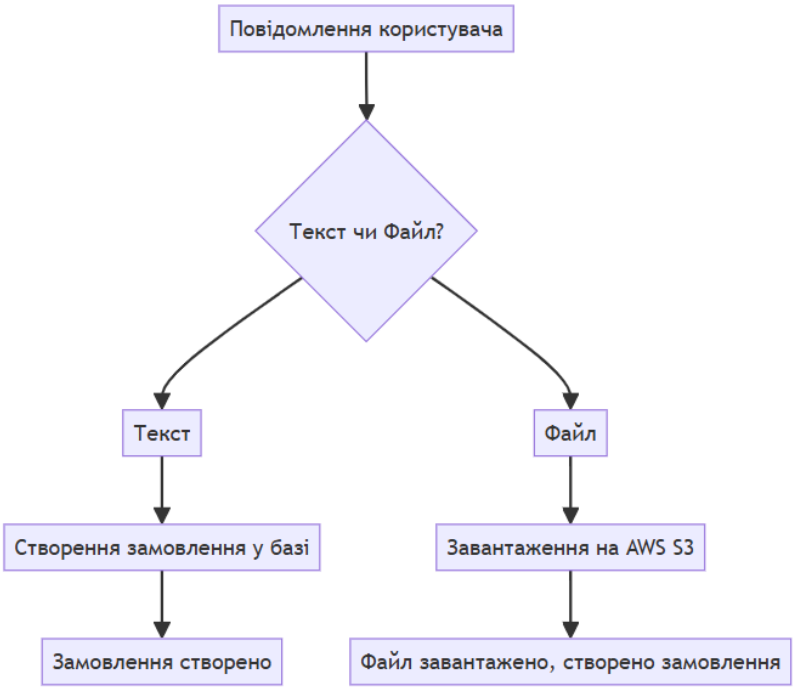


Рисунок 4.6 – схема функція обробки текстових повідомлень

4.2.3 Опис функціональних можливостей веб-інтерфейсу

1. Функція "Обробка створення замовлення":

Відповідає за обробку POST-запиту від користувача, валідацію форми та збереження даних у базу даних. Якщо дані містять файл, він зберігається у хмарному сховищі AWS S3.

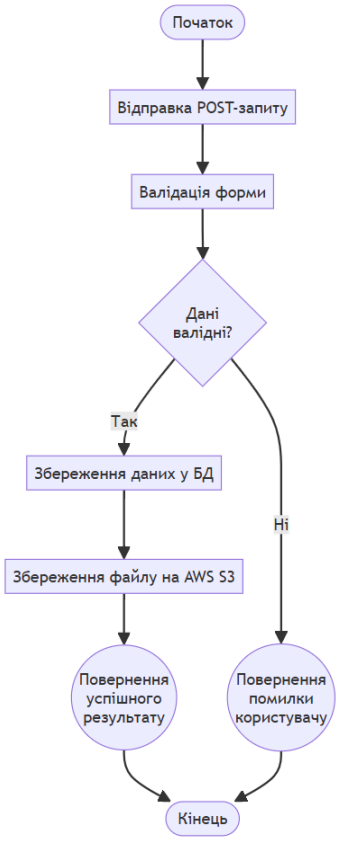


Рисунок 4.7 – Функція "Обробка створення замовлення":

2. Функція "Відображення списку замовлень":

Запит на сервер обробляється у Django View, отримуються дані з бази даних і рендериться шаблон для відображення списку замовлень.

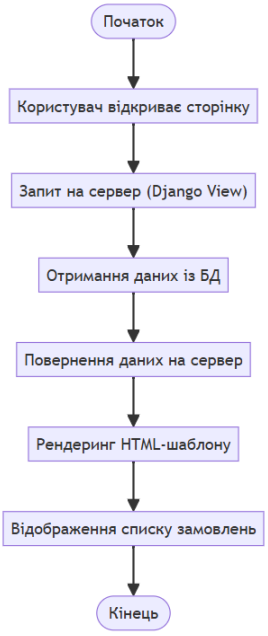


Рисунок 4.8 – Функція "Відображення списку замовлень":

3. Функція "Редагування замовлення":

Виконує перевірку наявності замовлення у БД, валідацію даних та оновлення інформації у базі даних.

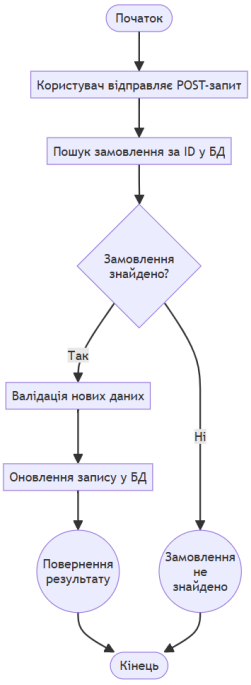


Рисунок 4.9 – Функція "Редагування замовлення":

4. Функція "Завантаження файлу":

Перевіряє формат файлу, зберігає його у AWS S3 та оновлює відповідний запис у базі даних.

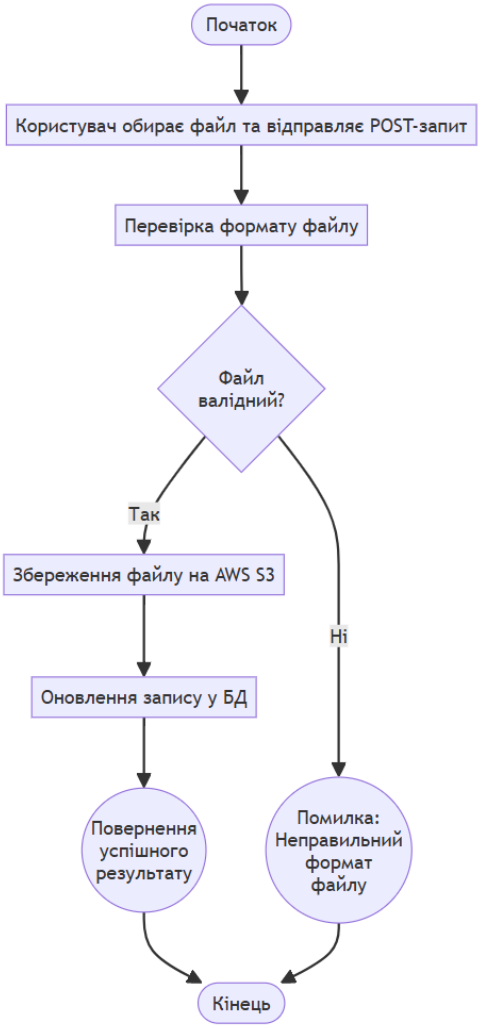


Рисунок 4.10 – Функція "Завантаження файлу"

На рисунку 4.11 показана схема навігації яка складається з головної сторінки, сторінки замовлень на якій можна сформувати замовлення, сторінки статусу замовлень. Конфігурацію цих сторінок подано у Додатку А.

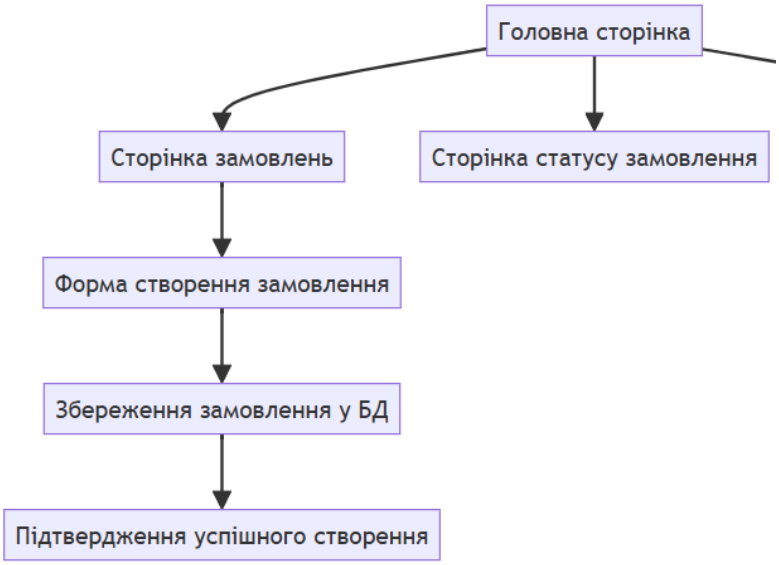


Рисунок 4.11 – Схема навігації веб-інтерфейсу

4.3 Висновки

У результаті розробки системи бюро перекладів «InText» було створено комплексне програмне забезпечення, що поєднує сучасні веб-технології, хмарні сервіси та інтерфейс Telegram-бота. Реалізована система дозволяє автоматизувати процеси прийому та обробки замовлень, забезпечує надійне збереження даних у хмарному сховищі AWS, а також інтегрує ефективний засіб комунікації з клієнтами.

Основні компоненти, такі як серверна частина на основі Django, база даних MySQL та інтеграція з хмарними сервісами, забезпечують стабільну роботу системи, високу продуктивність та безпеку даних. Гнучкість та масштабованість системи дозволяють легко впроваджувати нові функціональні можливості для подальшого розвитку компанії.

Запропоноване рішення дозволяє бюро перекладів «InText» ефективно долати технічні виклики, пов'язані з фізичною інфраструктурою, і підвищити якість надання послуг для клієнтів.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Завдання та мета експерименту

Метою експерименту є перевірка працездатності, надійності та ефективності функціонування розробленої комп'ютерної системи бюро перекладів «InText», що поєднує хмарні сервіси AWS, веб-систему та Telegram-бот для автоматизації робочих процесів.

Основні завдання, які ставляться перед експериментом, включають:

– Перевірку коректності роботи веб-системи при створенні замовлень, завантаженні файлів і обробці даних;

– Тестування роботи Telegram-бота на предмет коректного прийому команд, обробки запитів користувачів та передачі даних до хмарного сховища S3 і бази даних MySQL;

– Аналіз продуктивності та навантажувальних характеристик системи під різними умовами використання;

– Оцінку надійності системи, включаючи захист даних, збереження інформації та стабільність взаємодії між компонентами;

– Перевірку ефективності інтеграції веб-додатку та Telegram-бота з хмарною інфраструктурою AWS (S3, RDS, EC2).

В результаті проведення експерименту планується підтвердити, що система відповідає технічним вимогам, стабільно функціонує при навантаженні та забезпечує автоматизацію процесів бюро перекладів, мінімізуючи вплив зовнішніх факторів.

5.2 Методика проведення експерименту

5.2.1 Умови проведення експерименту

Експериментальна перевірка системи проводиться в середовищі, що імітує реальні робочі умови бюро перекладів «InText». Для цього використовуються наступні компоненти:

- Хмарні сервіси AWS для зберігання файлів і запуску обчислювальних ресурсів;
- База даних MySQL для збереження текстових замовлень і мета-даних;
- Telegram-бот як інтерфейс взаємодії з клієнтами;
- Веб-додаток для управління замовленнями.

Середовище тестування включає реальні сценарії використання системи, такі як створення замовлень, завантаження файлів, обробка даних і надання звітів про статуси.

5.2.2 Інструменти та методи тестування

Для веб-ресурсу:

– Ручне тестування інтерфейсу з метою виявлення помилок у верстці, функціоналі та логіці роботи. Було перевірено коректність відображення елементів, навігацію між сторінками та збереження даних на сервері.

Для Telegram-бота:

– Ручне надсилання текстових запитів для перевірки роботи команд, обробки сценаріїв та відповіді бота. Тестування охоплювало взаємодію з користувачем, обробку введених даних та перевірку на різні можливі помилки.

– Аналіз взаємодії з базою даних MySQL для перевірки коректності збереження, оновлення та отримання інформації про замовлення.

5.2.3 Методи оцінки результатів

Результати експерименту оцінюються на основі таких показників:

1. Функціональність:
   - Веб-додаток успішно приймає замовлення, зберігає їх у базі даних і завантажує файли у хмарне сховище AWS S3.
   - Telegram-бот коректно взаємодіє з клієнтами для отримання замовлень і надає статуси.

2. Продуктивність:
   - Час відповіді системи на одночасні запити в умовах навантаження.
   - Швидкість завантаження та обробки файлів у хмарному середовищі.
3. Надійність:
   - Перевірка стійкості системи до помилок, збоїв і навантажень.
   - Тестування резервного копіювання та відновлення даних.
4. Безпека:
   - Захист переданих даних за допомогою шифрування.
   - Обмеження доступу через механізми аутентифікації та авторизації.

5.3 Хід експерименту

5.3.1 Експеримент з веб-ресурсом

На початку експерименту перейдемо на ip адресу EC2, в нашому випадку http://54.198.120.52/. Успішний вхід на стартову сторінку показано на рисунку 5.1.

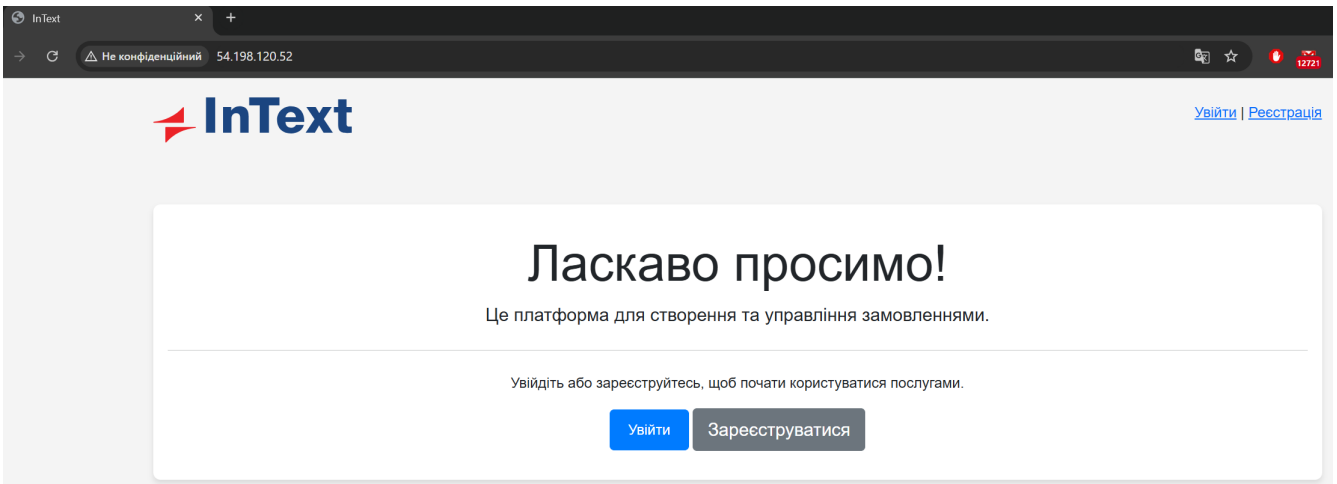


Рисунок 5.1 – Головна сторінка

Наступним кроком зареєструємо нового користувача з username test_mag_kvalif, поштою та паролем. Вікно реєстрації показано на рисунку 5.2.

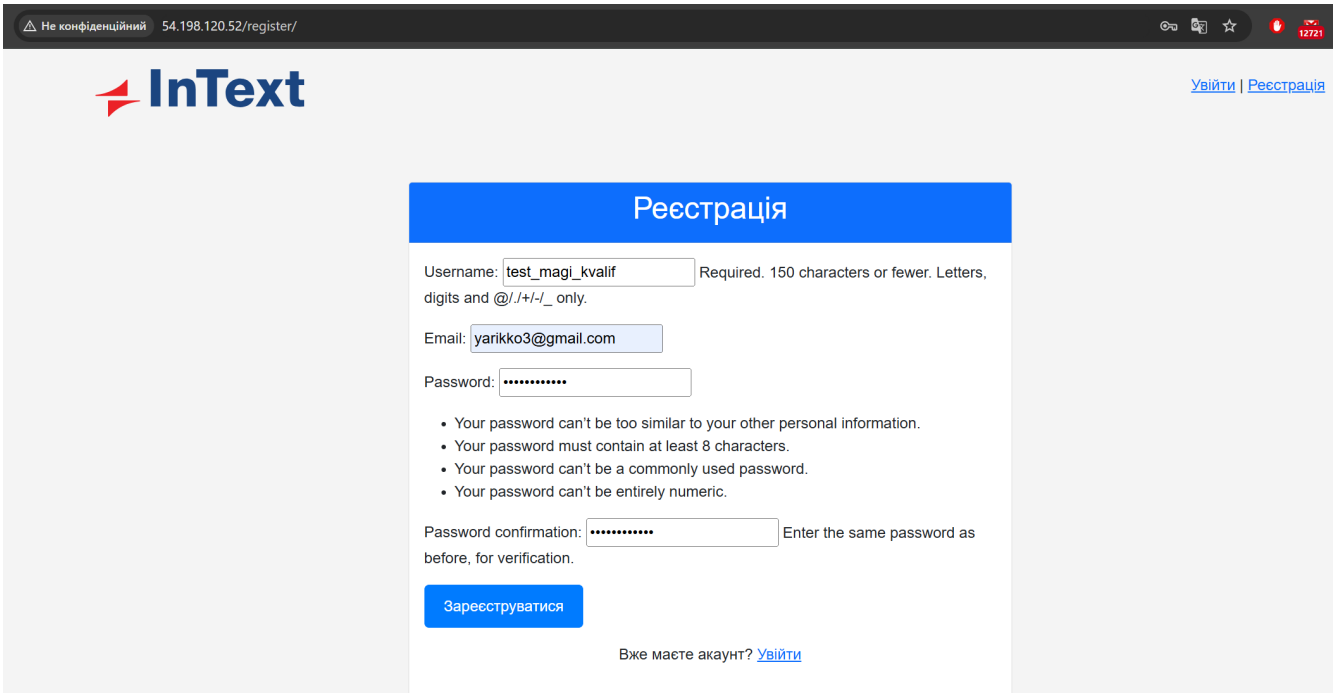


Рисунок 5.2 – Вікно реєстрації

Після реєстрації потрапляємо на вікно створення замовлення. Створимо замовлення з описом та прикріпимо файл для перекладу. Створення замовлення показано на рисунку 5.3.

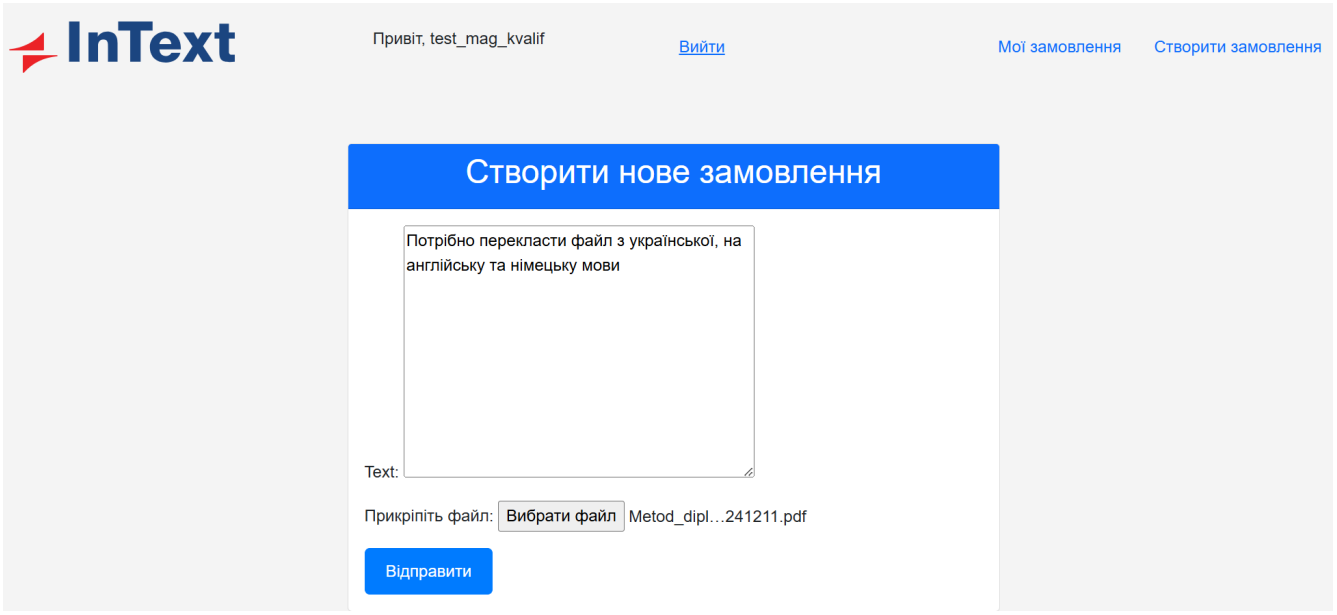


Рисунок 5.3 – Створення замовлення

Після натискання кнопки Відправити, замовлення створюється. На новій сторінці показується номер замовлення та успішність його створення. Це показано на рисунку 5.4.

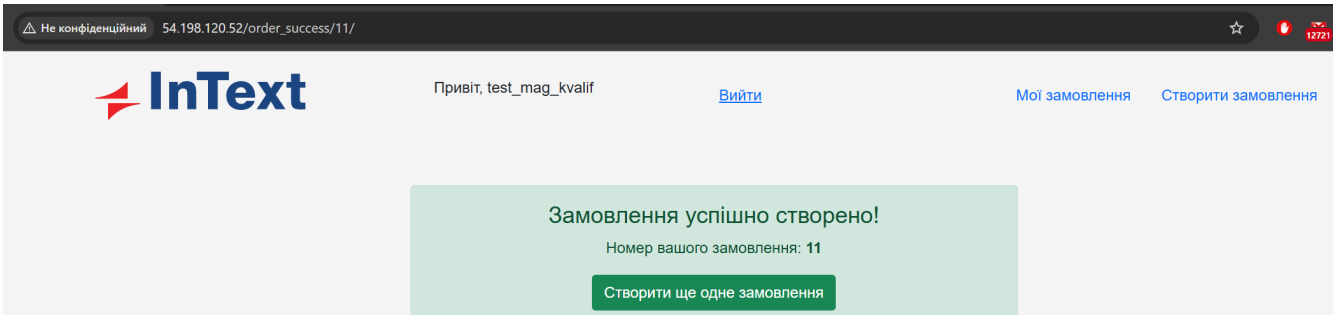


Рисунок 5.4 – Успішне створене замовлення

Перевіримо, чи завантажився файл в S3. Для цього зайдемо в AWS консоль, та знайдемо папку з номером замовлення. У папці orders було створено окрему папку з номером замовлення, в нашому випадку 11, і якій лежить наш тестовий файл. Це показано на рисунку 5.5.

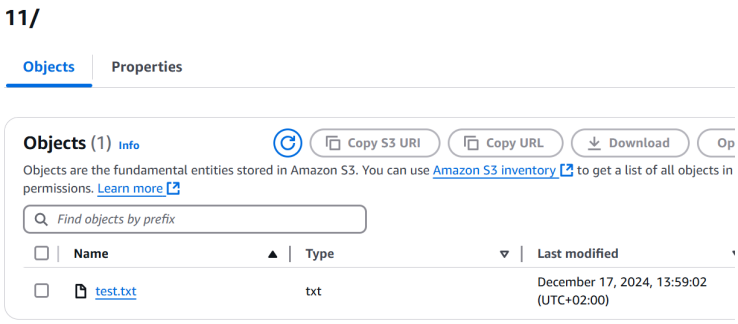
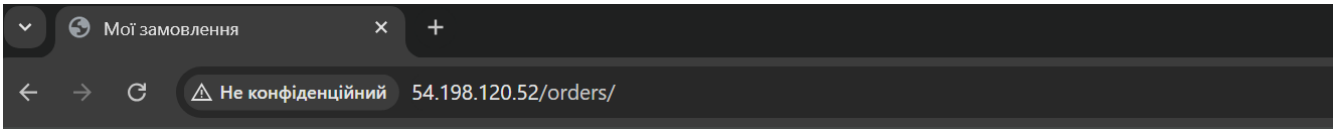


Рисунок 5.5 – Файл у s3 бакеті

Повернемось до нашого сайту, та створимо ще одне замовлення. Після цього перейдемо до списку наших замовлень. Список замовлень показано на рисунку 5.6.



Мої замовлення

- **ID:** 12 | **Статус:** Очікує підтвердження | **Текст:** test test test
- **ID:** 11 | **Статус:** Очікує підтвердження | **Текст:** Потрібно перекласти текст з української мови, на англійську та німецьку

Створити нове замовлення

Рисунок 5.6 – Список створених замовлень

Відкриємо базу даних в MySQL Workbench та подивимось як були створені замовлення. Це можна побачити на рисунку 5.7.

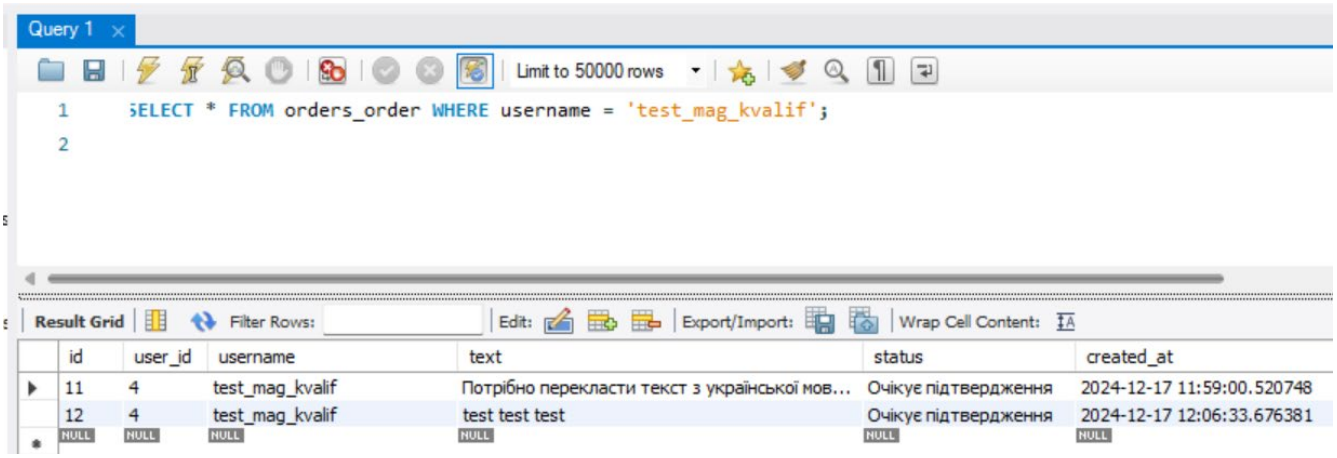


Рисунок 5.7 – База даних с замовленнями

Далі, натиснемо кнопку вийти, яку можна побачити на рисунку 5.4. Та вийдемо з акаунту і повернемось на головну сторінку. Це можна побачити на рисунку 5.8.

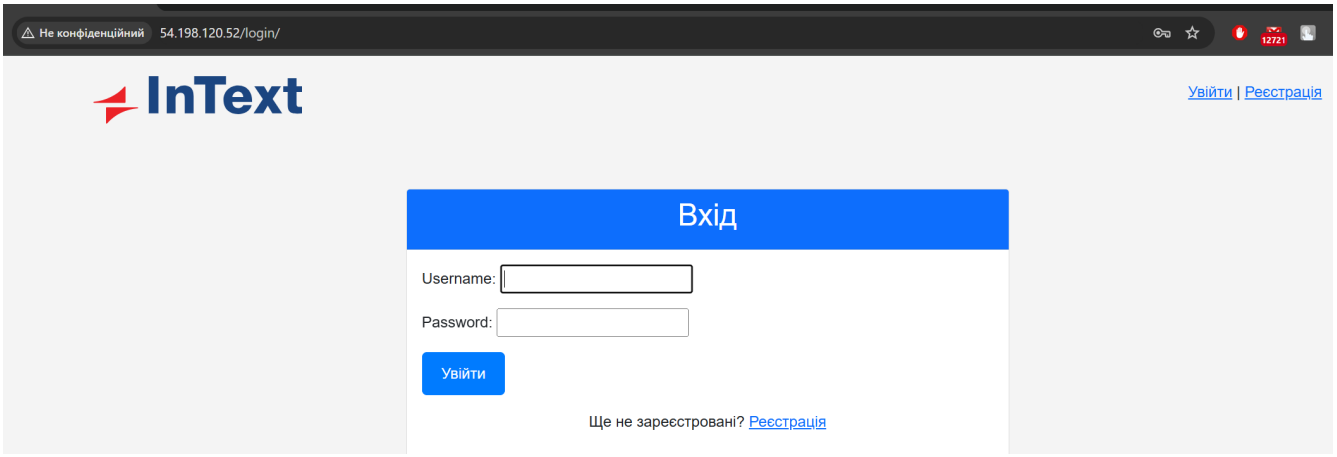


Рисунок 5.8 – Вихід з аккаунту

Якщо після виходу з аккаунту, спробувати повернутись на сторінку назад, то вийде помилка, що свідчить про те, що користувач успішно вийшов і система не дає йому повернутись. Це показано на рисунку 5.9.

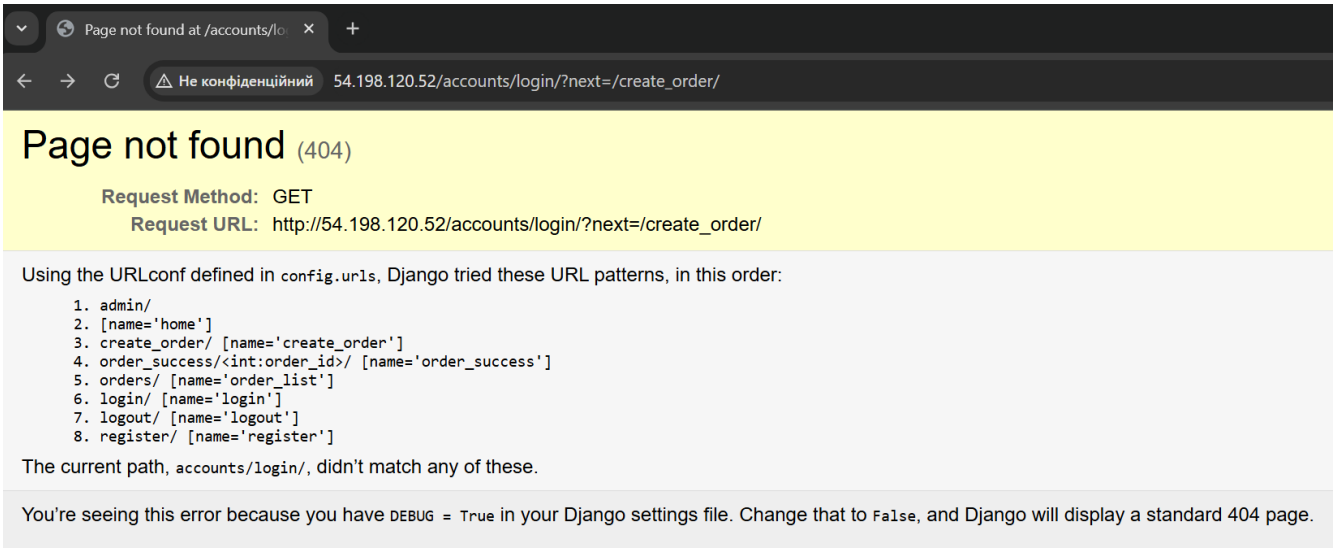


Рисунок 5.9 – Перевірка успішного виходу

Також було налаштовано адмінський акаунт і сторінку, для керування цими замволеннями. Для цього треба зайти на http://54.198.120.52/admin/, та ввести данні для логіну, які були створені при налаштуванні сервісу. Вигляд адмінської сторінки можна побачити на рисунку 5.10.

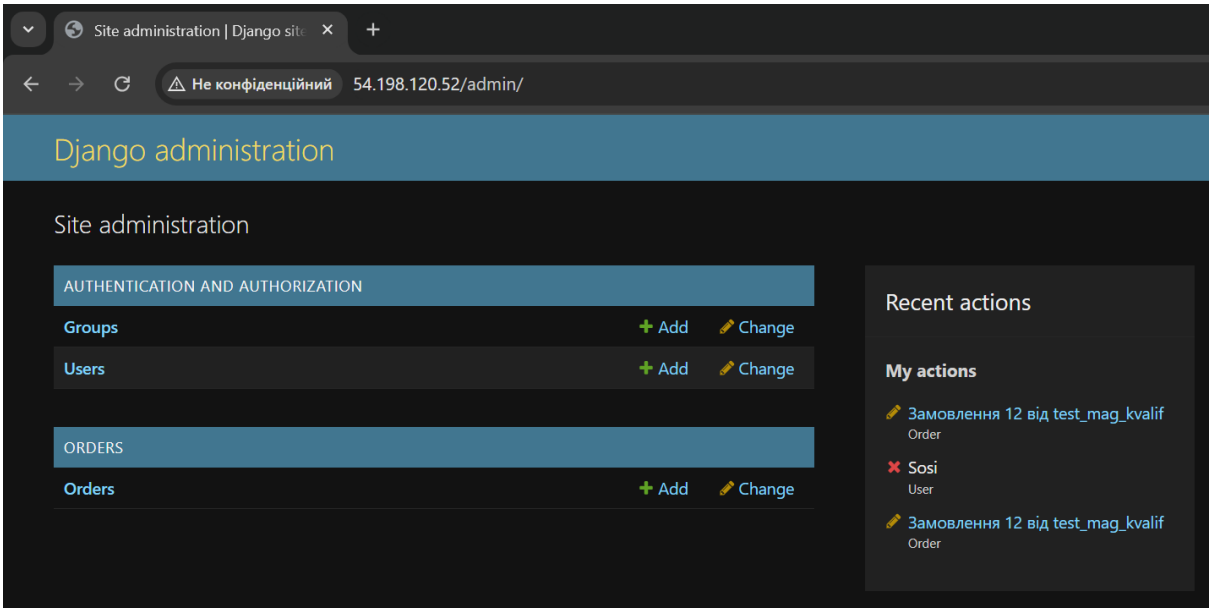


Рисунок 5.10 – Головна сторінка аккаунту адміна

Можна побачити 2 розділи: Users та Orders. В розділі Users можна керувати користувачами, додавати їх та видаляти. Цей розділ показано на рисунку 5.11.

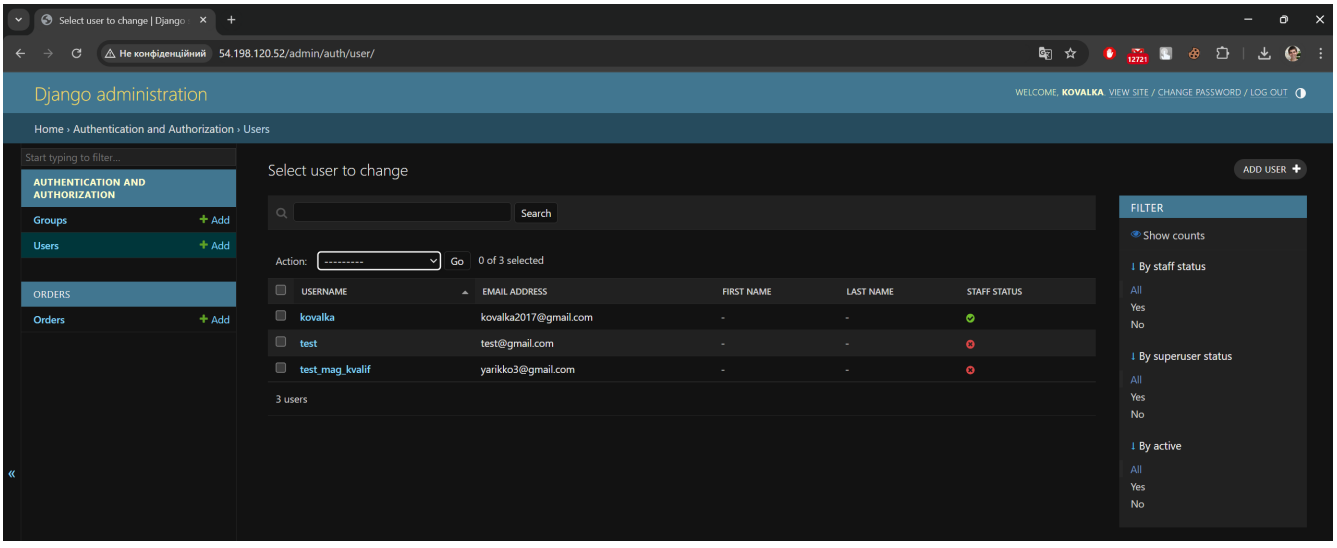


Рисунок 5.11 – Розділ з користувачами

В розділі Orders можна побачити всі замовлення, хто їх створив, який в них статус та коли воно було створено. На рисунку 5.12 можна побачити цю сторінку.

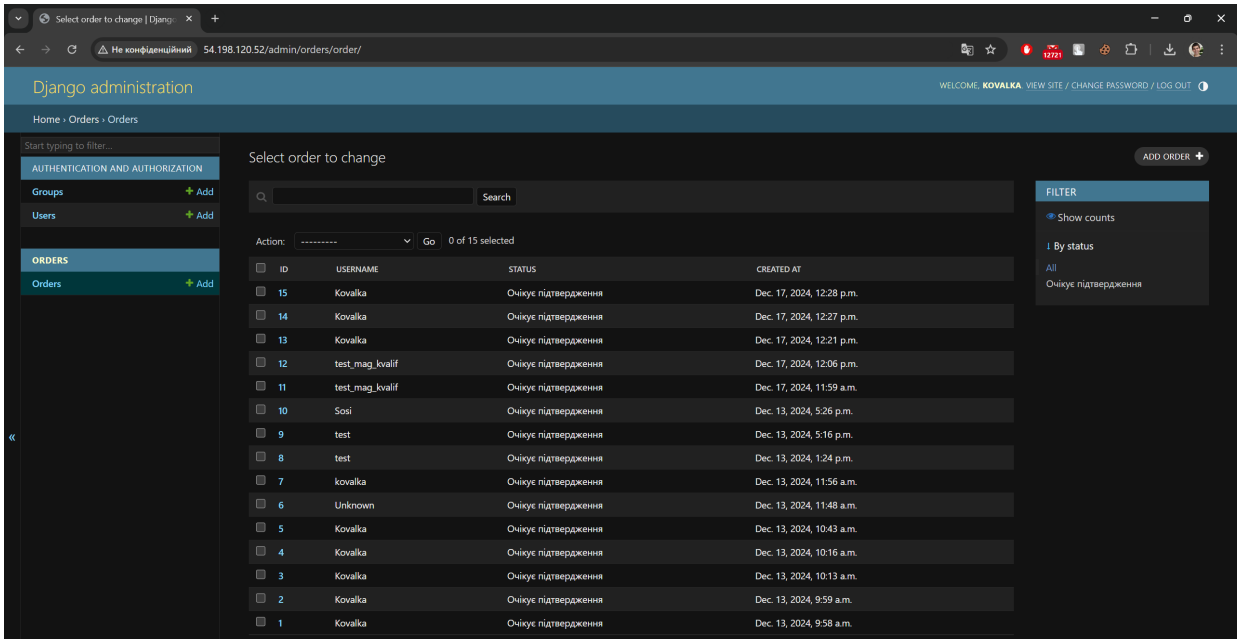


Рисунок 5.12 – Розділ з замовленнями

Зайдемо в наше створене замовлення та змінемо йому статус на «В роботі». Це показано на рисунку 5.13.

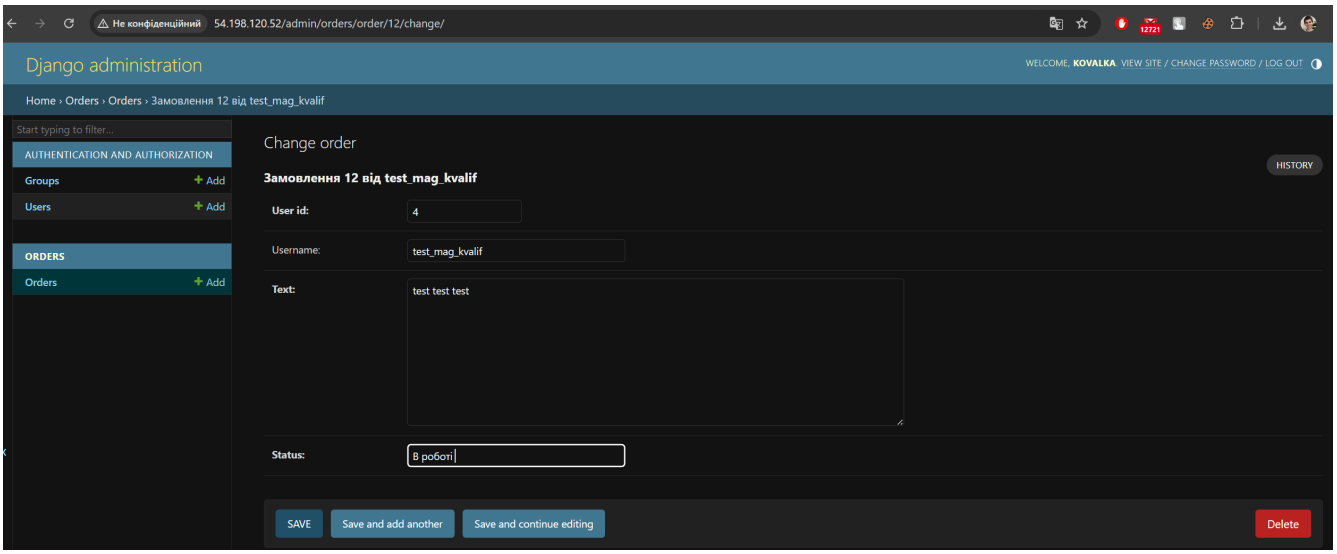


Рисунок 5.13 – Вікно змінення статусу замовлення

Перевіремо, чи змінився статус замовлення у загальному списку. Результат перевірки показано на рисунку 5.14.

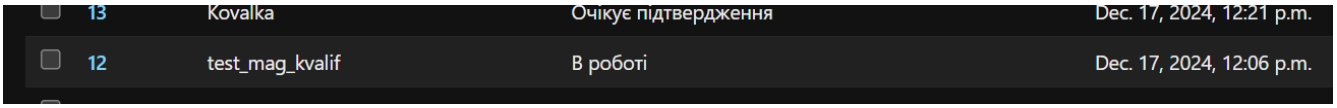


Рисунок 5.14 – Змінене замовлення

5.3.2 Експеримент з телеграм ботом

Для початку перейдемо в телеграм бота @Intext_client_service_bot та введемо команду /start. Нам одразу буде запропоновано 2 кнопки(рисунок 5.15): Створити замовлення, та перевірити статус. Спочатку перевіримо статус замовлення яке ми створювали через веб сайт, в нього було номер 12. На рисунку 5.16 показано його статус.

Рисунок 5.15 – Кнопки для керування

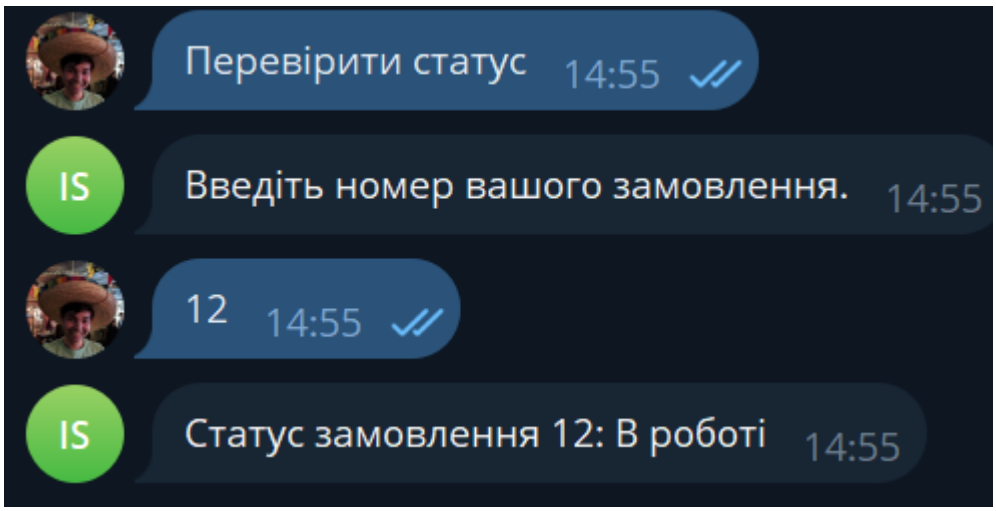


Рисунок 5.16 – Перевірка статусу

Наступним кроком створимо нове замовлення. Завантажимо файл та додамо опис. Процес створення показано на рисунку 5.17.

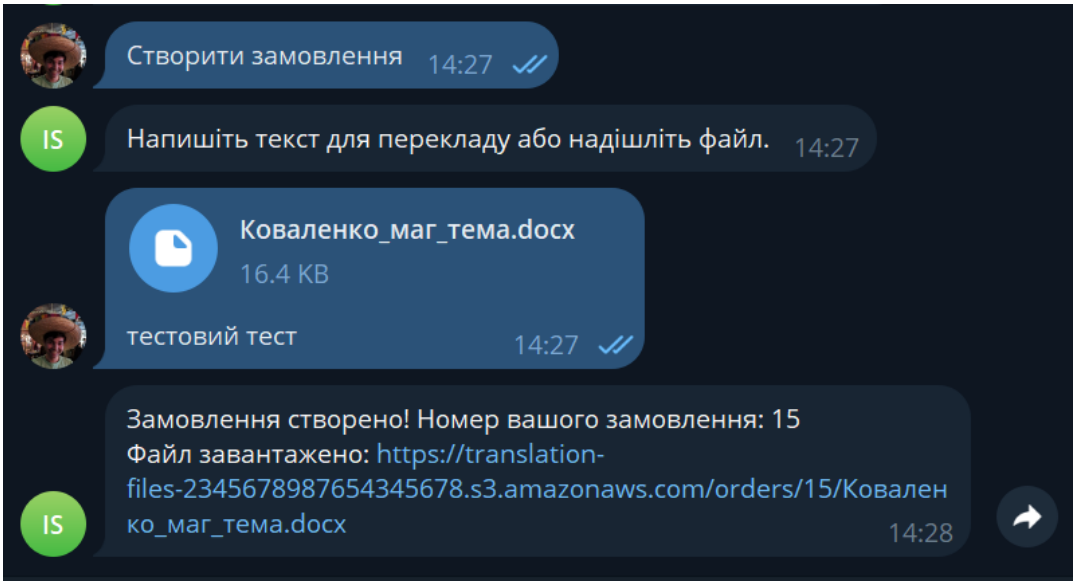


Рисунок 5.17 – Створення нового замовлення

Перейдемо в AWS консоль і перевіремо завантажений файл. Він з’явився. Це можна побачити на рисунку 5.18.

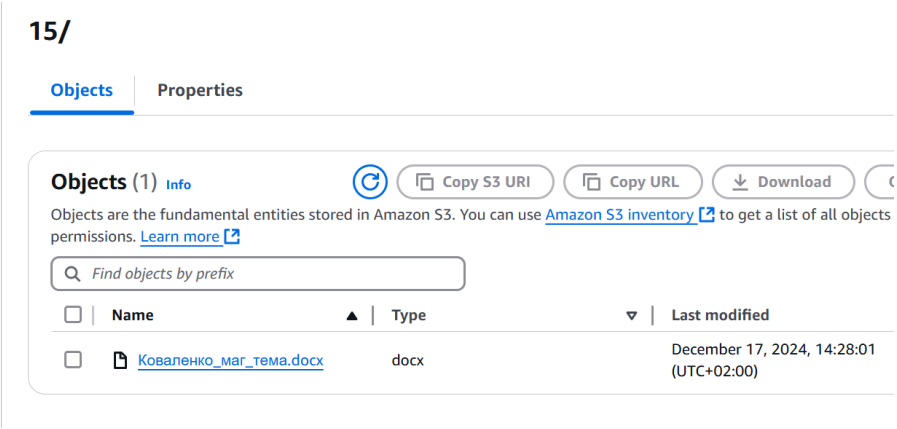


Рисунок 5.18 – Завантажений файл з боту

5.4 Висновки

У ході проведення експериментів було перевірено функціонування основних елементів системи бюро перекладів «InText», зокрема веб-ресурсу та Telegram-бота. Експеримент з веб-ресурсом підтвердив коректність виконання ключових функцій: реєстрації користувачів, створення замовлень, збереження файлів у хмарному сховищі AWS S3 та відображення замовлень у базі даних MySQL. Також було

успішно перевірено обмеження доступу після виходу з акаунту, що забезпечує необхідний рівень безпеки системи.

Експеримент з Telegram-ботом продемонстрував його здатність інтегруватися з базою даних та хмарним сховищем, виконуючи основні завдання: перевірку статусу існуючих замовлень та створення нових замовлень з автоматичним збереженням завантажених файлів у AWS S3. Обидві складові системи показали високу стабільність та надійність, підтверджуючи відповідність технічним вимогам і ефективну інтеграцію веб-ресурсу та Telegram-бота.

Таким чином, результати експерименту підтвердили працездатність і узгодженість роботи системи бюро перекладів «InText» на всіх етапах взаємодії користувачів з сервісом.

ВИСНОВОК

У кваліфікаційній роботі було розроблено та обґрунтовано комплексну комп’ютерну систему для автоматизації діяльності бюро перекладів «InText». Робота вирішує актуальні проблеми, пов’язані з надійністю, доступністю та стабільністю функціонування системи в умовах потенційних зовнішніх загроз, таких як відключення електроенергії або недоступність фізичних серверів.

У процесі дослідження було проведено аналіз існуючої проблематики та визначено ключові завдання, необхідні для досягнення мети. Для вирішення поставлених завдань обґрунтовано використання хмарних технологій, які забезпечують високу надійність та гнучкість інфраструктури. Зокрема, у системі застосовано сервіси AWS: S3 для зберігання файлів, EC2 для обчислювальних задач та RDS для організації бази даних MySQL.

Розроблена система включає Telegram-бот, що автоматизує комунікацію з клієнтами, обробку замовлень та відправлення сповіщень, що значно підвищує швидкість і ефективність взаємодії. Логіка функціонування системи реалізована за допомогою мови програмування Python з використанням фреймворку Django та бібліотеки python-telegram-bot.

У межах роботи було побудовано архітектуру системи, розроблено її основні компоненти та інтегровано програмне забезпечення з хмарними сервісами. Особливу увагу приділено розробці серверної частини та інтерфейсу для обробки замовлень, що забезпечує зручність як для клієнтів, так і для адміністраторів системи.

Проведене тестування продемонструвало стабільну та надійну роботу системи у реальних умовах експлуатації. Було перевірено коректність виконання основних функцій, включно зі збереженням даних, обробкою запитів, інтеграцією з хмарними ресурсами та комунікацією через Telegram-бот. Результати підтвердили відповідність системи технічним вимогам та поставленим завданням.

Таким чином, у ході виконання роботи було досягнуто основної мети – розроблено надійну, функціональну та гнучку комп’ютерну систему для автоматизації діяльності бюро перекладів «InText». Запропоноване рішення забезпечує ефективну обробку замовлень, надійне зберігання даних, оптимізує робочі процеси та підвищує рівень обслуговування клієнтів. Система готова до впровадження та подальшої масштабованості відповідно до потреб компанії.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ 3008-2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. – К.: Держстандарт, 2015. – 37 с.

2. Цвіркун Л.І. Атестація здобувачів вищої освіти. Методичні рекомендації до виконання кваліфікаційної роботи магістра здобувачами галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія / Л.І. Цвіркун, В.В. Гнатушенко, С.М. Ткаченко ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро: НТУ «ДП», 2024. – 54 с.

3. Amazon Web Services – [Електронний ресурс] – https://aws.amazon.com/ (дата звернення: 02.11.2024).

4. Amazon EC2: Налаштування серверів – [Електронний ресурс] – https://aws.amazon.com/ec2/ (дата звернення: 08.11.2024).

5. Amazon S3: Хмарне сховище – [Електронний ресурс] – https://aws.amazon.com/s3/ (дата звернення: 10.11.2024).

6. Amazon RDS: Бази даних у хмарі – [Електронний ресурс] – https://aws.amazon.com/rds/ (дата звернення: 09.11.2024).

7. Python Software Foundation – [Електронний ресурс] – https://www.python.org/ (дата звернення: 17.11.2024).

8. Django: Офіційна документація – [Електронний ресурс] – https://docs.djangoproject.com/ (дата звернення: 20.11.2024).

9. Python Telegram Bot – [Електронний ресурс] – https://python-telegram-bot.readthedocs.io/ (дата звернення: 15.11.2024).

10. Telegram Bot API – [Електронний ресурс] – https://core.telegram.org/bots/api (дата звернення: 15.11.2024).

11. Bootstrap: Фреймворк для адаптивного веб-дизайну – [Електронний ресурс] – https://getbootstrap.com/ (дата звернення: 07.12.2024).

12. Ручне тестування інтерфейсу – [Електронний ресурс] – https://www.guru99.com/manual-testing.html (дата звернення: 25.11.2024).

13. Інструкція з тестування Telegram ботів – [Електронний ресурс] – https://telegram.org/faq#bots (дата звернення: 08.12.2024).

ДОДАТОК А

Текст програми чат-боту, django модулів та веб-застосунку

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ТЕКСТ ПРОГРАМИ ЧАТ-БОТУ, GJANGO МОДУЛІВ ТА ВЕБ-ЗАСТОСУНКУ БЮРО ПЕРЕКЛАДУ «INTEXT»

Текст програми
804.02070743.24008-01 12 01
Листів 27

2024

АНОТАЦІЯ

Дана програма містить програмний код чат-боту Telegram, модулів Django та веб-застосунку, що є складовими комп'ютерної системи бюро перекладів «InText».

Програма призначена для автоматизації прийому замовлень, обробки текстових запитів користувачів, завантаження файлів у хмарне сховище та інтеграції з базою даних. Вона забезпечує взаємодію з користувачами через чат-бот у Telegram, а також через веб-інтерфейс, дозволяючи відстежувати статус замовлень і керувати ними.

Програма написана мовою Python з використанням фреймворку Django для серверної частини та бібліотеки python-telegram-bot для реалізації функціоналу бота. Зберігання даних виконується у базі даних MySQL, а файли завантажуються на хмарний сервіс Amazon S3. Середовище розробки – Visual Studio Code.

ЗМІСТ

1 Telegram

1.1 telegram_bot.py

```
import os
import django
from telegram.ext import Application, CommandHandler, MessageHandler
from telegram import Update, ReplyKeyboardMarkup
from telegram.ext import CallbackContext, filters
from asgiref.sync import sync_to_async
import boto3
from botocore.exceptions import NoCredentialsError

# Діагностика: Перевірка змінної середовища
print("DJANGO_SETTINGS_MODULE:", os.environ.get("DJANGO_SETTINGS_MODULE"))

# Ініціалізація Django
os.environ.setdefault("DJANGO_SETTINGS_MODULE",    "config.settings")     # Замість config.settings вкажіть шлях до ваших налаштувань
print("Після встановлення DJANGO_SETTINGS_MODULE")
django.setup()
print("Django успішно ініціалізовано")

from orders.models import Order  # Припускаємо, що є модель Order

# Отримання токена з змінної середовища
TOKEN = os.getenv("TELEGRAM_BOT_TOKEN")
AWS_ACCESS_KEY_ID = os.getenv("AWS_ACCESS_KEY_ID")
AWS_SECRET_ACCESS_KEY = os.getenv("AWS_SECRET_ACCESS_KEY")
AWS_BUCKET_NAME = os.getenv("AWS_BUCKET_NAME")
```

```
# Ініціалізація клієнта S3
s3_client = boto3.client(
    's3',
    aws_access_key_id=AWS_ACCESS_KEY_ID,
    aws_secret_access_key=AWS_SECRET_ACCESS_KEY
)

async def upload_to_s3(file_path, file_name, order_id):
    s3_key = f"orders/{order_id}/{file_name}"
    try:
        s3_client.upload_file(file_path, AWS_BUCKET_NAME, s3_key)
        file_url = f"https://{AWS_BUCKET_NAME}.s3.amazonaws.com/{s3_key}"
        return file_url
    except FileNotFoundError:
        print("The file was not found.")
        return None
    except NoCredentialsError:
        print("Credentials not available.")
        return None

# Команда /start
async def start(update: Update, context: CallbackContext) -> None:
    keyboard = [
        ["Створити замовлення", "Перевірити статус"]
    ]
      reply_markup = ReplyKeyboardMarkup(keyboard, resize_keyboard=True,
one_time_keyboard=True)
    await update.message.reply_text(
```

```
        "Привіт! Виберіть одну з опцій:",
        reply_markup=reply_markup
    )

# Команда /help
async def help_command(update: Update, context: CallbackContext) -> None:
    await update.message.reply_text(
        "Доступні команди:\n"
        "/start - Почати роботу з ботом\n"
        "/help - Отримати список доступних команд\n"
        "Використовуйте кнопки під клавіатурою для взаємодії з ботом."
    )

# Обробка текстових повідомлень
async def handle_message(update: Update, context: CallbackContext) -> None:
    text = update.message.text.lower() if update.message.text else None

    if text == "створити замовлення":
        context.user_data['awaiting_order'] = True
         await update.message.reply_text("Напишіть текст для перекладу або надішліть файл.")
    elif text == "перевірити статус":
        context.user_data['awaiting_status'] = True
        await update.message.reply_text("Введіть номер вашого замовлення.")
    elif context.user_data.get('awaiting_order', False):
        if update.message.document:
            file_id = update.message.document.file_id
            file = await context.bot.get_file(file_id)
            file_path = f"/tmp/{update.message.document.file_name}"
```

```
            await file.download_to_drive(file_path)


            user = update.message.from_user
            order = await sync_to_async(Order.objects.create)(
                user_id=user.id,
                username=user.username,
                text=f"Файл: {update.message.document.file_name}",
                status="Очікує підтвердження"
            )


                            file_url    =    await    upload_to_s3(file_path,
update.message.document.file_name, order.id)
            if file_url:
                order.text = f"Файл: {file_url}"
                await sync_to_async(order.save)()
                context.user_data['awaiting_order'] = False
                await update.message.reply_text(
                          f"Замовлення створено! Номер вашого замовлення:
{order.id}\nФайл завантажено: {file_url}"
                )
            else:
               await update.message.reply_text("Помилка під час завантаження
файлу. Спробуйте ще раз.")
        elif text:
            user = update.message.from_user
            order = await sync_to_async(Order.objects.create)(
                user_id=user.id,
                username=user.username,
                text=text,
```

```
                status="Очікує підтвердження"
            )
            context.user_data['awaiting_order'] = False
            await update.message.reply_text(
                f"Замовлення створено! Номер вашого замовлення: {order.id}"
            )
        else:
            await update.message.reply_text("Будь ласка, надішліть текст або файл для замовлення.")
    elif context.user_data.get('awaiting_status', False):
        try:
            order_id = int(text)
            order = await sync_to_async(Order.objects.get)(id=order_id)
            context.user_data['awaiting_status'] = False
            await update.message.reply_text(
                f"Статус замовлення {order_id}: {order.status}"
            )
        except ValueError:
             await update.message.reply_text("Будь ласка, введіть коректний номер замовлення.")
        except Order.DoesNotExist:
            context.user_data['awaiting_status'] = False
             await update.message.reply_text("Замовлення з таким номером не знайдено.")
    else:
       await update.message.reply_text("Я вас не розумію. Використайте кнопку на клавіатурі.")


# Обробка статусу замовлення
async def check_status(update: Update, context: CallbackContext) -> None:
```

```
    if len(context.args) == 0:
            await update.message.reply_text("Будь ласка, вкажіть номер замовлення. Наприклад: /status 123")
        return

    order_id = context.args[0]
    try:
        order = await sync_to_async(Order.objects.get)(id=order_id)
        await update.message.reply_text(
            f"Статус замовлення {order_id}: {order.status}"
        )
    except Order.DoesNotExist:
          await update.message.reply_text("Замовлення з таким номером не знайдено.")

def main():
    # Ініціалізація бота
    application = Application.builder().token(TOKEN).build()

    # Обробники команд
    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("help", help_command))
    application.add_handler(CommandHandler("status", check_status))

    # Обробник текстових повідомлень
                application.add_handler(MessageHandler(filters.TEXT | filters.Document.ALL & ~filters.COMMAND, handle_message))

    # Запуск бота
```

```
    application.run_polling()

if __name__ == '__main__':
    main()
```

2 Django

2.1 Orders

2.1.1 admin.py

```
from django.contrib import admin
from .models import Order

@admin.register(Order)
class OrderAdmin(admin.ModelAdmin):
    list_display = ('id', 'username', 'status', 'created_at')
    list_filter = ('status',)
    search_fields = ('username', 'text')
```

2.1.2 forms.py

```
from django import forms
from orders.models import Order
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

# Форма для створення замовлення
class OrderForm(forms.ModelForm):
    class Meta:
        model = Order
        fields = ['text', 'file']
```

```
    file = forms.FileField(required=False, label="Прикріпіть файл")


class RegisterForm(UserCreationForm):
    email = forms.EmailField(required=True)


    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
```

2.1.3 models.py

```
from django.db import models


class Order(models.Model):
    user_id = models.BigIntegerField()  # ID користувача Telegram
     username = models.CharField(max_length=150, blank=True, null=True)  # Username у Telegram
    text = models.TextField()  # Текст замовлення
          status = models.CharField(max_length=50, default="Очікує підтвердження")  # Статус замовлення
     created_at = models.DateTimeField(auto_now_add=True)  # Час створення замовлення


    def __str__(self):
        return f"Замовлення {self.id} від {self.username or 'Користувач'}"
```

2.1.4 urls.py

```
from django.urls import path
from orders import views
```

```
from django.contrib.auth import views as auth_views

urlpatterns = [

    path('', views.home, name='home'),

    path('create_order/', views.create_order, name='create_order'),

              path('order_success/<int:order_id>/',      views.order_success,
name='order_success'),

    path('orders/', views.order_list, name='order_list'),

    path('login/', auth_views.LoginView.as_view(template_name='login.html'),
name='login'),  # Логін

         path('logout/',   auth_views.LogoutView.as_view(next_page='login'),
name='logout'),  # Вихід

    path('register/', views.register, name='register'),  # Реєстрація

]
```

2.1.5 views.py

```
from django.shortcuts import render, redirect

from django.contrib.auth.decorators import login_required

from orders.forms import OrderForm

from orders.models import Order

import boto3

from django.conf import settings

from django.contrib.auth import login

from django.contrib.auth.forms import UserCreationForm

from django.shortcuts import render, redirect

from .forms import RegisterForm


@login_required

def create_order(request):
```

```
    if request.method == 'POST':
        form = OrderForm(request.POST, request.FILES)
        if form.is_valid():
            order = form.save(commit=False)
            order.status = "Очікує підтвердження"

            # Перевірка автентичності користувача
            if request.user.is_authenticated:
                order.user_id = request.user.id  # Прив'язуємо замовлення до залогіненого користувача
                order.username = request.user.username  # Використовуємо username залогіненого користувача
            else:
                return redirect('login')  # Перенаправлення на сторінку логіну, якщо неавторизований

            order.save()

            # Завантаження файлу в S3, якщо файл додано
            if 'file' in request.FILES:
                file = request.FILES['file']
                s3 = boto3.client('s3', aws_access_key_id=settings.AWS_ACCESS_KEY_ID,
                                  aws_secret_access_key=settings.AWS_SECRET_ACCESS_KEY)
                bucket_name = settings.AWS_STORAGE_BUCKET_NAME
                s3_file_path = f"orders/{order.id}/{file.name}"
                s3.upload_fileobj(file, bucket_name, s3_file_path)

            return redirect('order_success', order_id=order.id)
```

```
    else:
        form = OrderForm()
    return render(request, 'create_order.html', {'form': form})


@login_required
def order_success(request, order_id):
    return render(request, 'order_success.html', {'order_id': order_id})
@login_required
def order_list(request):
       orders   =   Order.objects.filter(user_id=request.user.id).order_by('-
created_at')
    return render(request, 'order_list.html', {'orders': orders})
def register(request):
    if request.method == 'POST':
        form = RegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)  # Автоматично логінити після реєстрації
             return redirect('create_order')  # Перенаправлення на створення
замовлення
    else:
        form = RegisterForm()
    return render(request, 'register.html', {'form': form})
def home(request):
    return render(request, 'home.html')
```

2.2 config

2.2.1 settings.py

```
"""
Django settings for config project.

Generated by 'django-admin startproject' using Django 5.1.4.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.1/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-qx0asjvd(l%p0wrww5ol+g*xpzbce9593u6%c7ohcg#nvx79!g'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```

```
ALLOWED_HOSTS = ['54.198.120.52', 'localhost', '127.0.0.1']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'orders',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'config.urls'

TEMPLATES = [
```

```
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ["templates"],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'config.wsgi.application'

# Database
# https://docs.djangoproject.com/en/5.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mag_database',
        'USER': 'admin',
        'PASSWORD': 'Yarikko2024!',
        'HOST': 'database-1.cnmmkao6m3q7.us-east-1.rds.amazonaws.com',
    }
```

```
}


# Password validation
#         https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-
validators


AUTH_PASSWORD_VALIDATORS = [
    {
                                                                    'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
                                                                    'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
                                                                    'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
                                                                    'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/5.1/topics/i18n/


LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.1/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

STATIC_URL = '/static/'
STATIC_ROOT = BASE_DIR / 'staticfiles/'

APPEND_SLASH = True
AWS_ACCESS_KEY_ID = 'AKIAVYV52D4MCRUCXZO5'  # Доступ до AWS
AWS_SECRET_ACCESS_KEY = 'aGk02dFM960D8OzWqlzxDiIX1+h4xshkXXCzfz1k'
AWS_STORAGE_BUCKET_NAME = 'translation-files-2345678987654345678'
LOGIN_REDIRECT_URL = 'create_order'
LOGOUT_REDIRECT_URL = 'login'
CSRF_TRUSTED_ORIGINS = ['http://3.91.8.176']
```

2.2.2 urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('orders.urls')),
          ]
```

3 HTML

3.1 base.html

```
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}InText{% endblock %}</title>
                                    <link                  rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.c
ss">
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
        }
        .navbar {
            background-color: #ffffff;
            border-bottom: 2px solid #eaeaea;
```

```
        }
        .navbar-brand img {
            height: 50px;
        }
        .btn-primary {
            background-color: #007bff;
            border-color: #007bff;
            font-size: 1rem;
            padding: 10px 20px;
            border-radius: 5px;
        }
        .btn-primary:hover {
            background-color: #0056b3;
            border-color: #0056b3;
        }
        .jumbotron {
            background-color: #ffffff;
            padding: 2rem 1rem;
            border-radius: 0.5rem;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>
    <div class="container">
        <header class="d-flex justify-content-between align-items-center py-
3">
            <a href="/" class="navbar-brand">
```

```
                        <img  src="https://intext.com/dist/img/svg/logo.svg"
alt="Логотип">
            </a>
            {% if user.is_authenticated %}
                    <p  class="text-end">Привіт,  {{  user.username  }}  <form
method="post"   action="/logout/"   style="display:inline;">{%   csrf_token
%}<button class="btn btn-link" type="submit">Вийти</button></form></p>
                <nav class="nav">
                    <a class="nav-link" href="/orders/">Мої замовлення</a>
                        <a  class="nav-link"  href="/create_order/">Створити
замовлення</a>
                </nav>
            {% else %}
                    <p  class="text-end"><a  href="/login/">Увійти</a>  |  <a
href="/register/">Реєстрація</a></p>
            {% endif %}
        </header>
        {% block content %}{% endblock %}
    </div>
</body>
```

3.2 create_order.html

```
{% extends 'base.html' %}

{% block content %}
<div class="row justify-content-center mt-5">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header text-center bg-primary text-white">
                <h2>Створити нове замовлення</h2>
```

```
            </div>
            <div class="card-body">
                <form method="post" enctype="multipart/form-data">
                    {% csrf_token %}
                    {{ form.as_p }}
                        <button type="submit" class="btn btn-primary btn-block">Відправити</button>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

3.3 home.html

```
{% extends 'base.html' %}

{% block content %}
<div class="jumbotron text-center mt-5">
    <h1 class="display-4">Ласкаво просимо!</h1>
        <p class="lead">Це платформа для створення та управління замовленнями.</p>
    <hr class="my-4">
    <p>Увійдіть або зареєструйтесь, щоб почати користуватися послугами.</p>
   <a class="btn btn-primary btn-lg" href="/login/" role="button">Увійти</a>
          <a class="btn btn-secondary btn-lg" href="/register/" role="button">Зареєструватися</a>
</div>
{% endblock %}
```

3.4 login.html

```
{% extends 'base.html' %}


{% block content %}
<div class="row justify-content-center mt-5">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header text-center bg-primary text-white">
                <h2>Вхід</h2>
            </div>
            <div class="card-body">
                <form method="post">
                    {% csrf_token %}
                    {{ form.as_p }}
                        <button type="submit" class="btn btn-primary btn-
block">Увійти</button>
                </form>
                     <p class="text-center mt-3">Ще не зареєстровані? <a
href="/register/">Реєстрація</a></p>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

3.5 order_list.html

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Мої замовлення</title>
</head>
<body>
    <h1>Мої замовлення</h1>
    <ul>
        {% for order in orders %}
            <li>
                <strong>ID:</strong> {{ order.id }} |
                <strong>Статус:</strong> {{ order.status }} |
                <strong>Текст:</strong> {{ order.text }}
            </li>
        {% empty %}
            <li>Немає замовлень.</li>
        {% endfor %}
    </ul>
    <a href="/create_order/">Створити нове замовлення</a>
</body>
</html>
```

order_success.html

```
{% extends 'base.html' %}

{% block content %}
<div class="row justify-content-center mt-5">
    <div class="col-md-6">
        <div class="alert alert-success text-center">
            <h4>Замовлення успішно створено!</h4>
```

```
            <p>Номер вашого замовлення: <strong>{{ order_id }}</strong></p>
            <a href="/create_order/" class="btn btn-success">Створити ще одне замовлення</a>
        </div>
    </div>
</div>
{% endblock %}
```

3.6 register.html

```
{% extends 'base.html' %}

{% block content %}
<div class="row justify-content-center mt-5">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header text-center bg-primary text-white">
                <h2>Реєстрація</h2>
            </div>
            <div class="card-body">
                <form method="post">
                    {% csrf_token %}
                    {{ form.as_p }}
                        <button type="submit" class="btn btn-primary btn-block">Зареєструватися</button>
                </form>
                    <p class="text-center mt-3">Вже маєте акаунт? <a href="/login/">Увійти</a></p>
            </div>
        </div>
```

```
    </div>
</div>
{% endblock %}
```