Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
_____Інститут електроенергетики _____
(інститут)
_____ факультет інформаційних технологій _____
(факультет)
Кафедра _____інформаційних технологій та комп'ютерної інженерії
_____
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ____бакалавра___
(бакалавра, спеціаліста, магістра)
студента _____Міняйленко Євгенія Олеговича_____
(ПІБ)
академічної групи _____126-19-1_____
(шифр)
спеціальності _____126 «Інформаційні системи та технології _____
(код і назва спеціальності)
за освітньо-професійною програмою _____
(за наявності)
_____«Інформаційні системи та технології»_____
(офіційна назва)
на тему Розробка веб-орієнтованої статистичної платформи з геоінформаційною підтримкою _____
(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	к.т.н., доц. Соколова Н.О.			
розділів:				

Рецензент	к.т.н., доц.каф ПЗКС Ширін А.Л.			

Нормоконтролер	д.т.н., проф. Коротенко Г.М.			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри
_____інформаційних технологій _____
_____та комп'ютерної інженерії_____
(повна назва)
_____ _Гнатушенко В.В._
(підпис) (прізвище, ініціали)

«_____»_____2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____**бакалавра**_____
(бакалавра, спеціаліста, магістра)

студенту_____Міняйленко Є.О._____ **академічної групи** _____126-19-1____
(прізвище та ініціали) (шифр)
спеціальності _____126 «Інформаційні системи та технології»_____
за освітньою-професійною програмою _____
(за наявності)
_____«Інформаційні системи та технології»_____
на тему ___Розробка веб-орієнтованої статистичної платформи з геоінформаційною підтримкою _____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 16.05.2023 р. .№ 350-с

Розділ	**Зміст**	**Термін виконання**
Розділ 1	Аналіз теми та постановка задачі	01.05.2023 – 08.05.2023
Розділ 2	Огляд методів автоматизованого управління транспортними потоками	17.05.2023 – 29.05.2023
Розділ 3	Експериментальна частина. Підготовка матеріалів для захисту роботи	30.05.2023 – 10.06.2023

Завдання видано _____ _____Н.О.Соколова_____
(підпис керівника) (прізвище, ініціали)

Дата видачі __01.05.2023 р._____

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ _____Міняйленко Є.О._____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 101 стор., 38 рис., 4 табл., 3 додатки, 16 джерел.

Об'єкт розробки: веб-орієнтована статистична платформа з геоінформаційною підтримкою.

Мета кваліфікаційної роботи: розробка веб-орієнтованої статистичної системи завантаженості пунктів пропуску кордонів країни на платформі чат-боту.

У вступі наведено стан проблеми та обґрунтована її актуальність.

Перший розділ включає в себе постановку завдання і характеристику предметної області.

У другому розділі був проведений огляд та вибір технологій для веб-орієнтованої статистичної платформи.

У третьому розділі була розроблена та протестована веб-орієнтована статистична платформа з геоінформаційною.

Практичне значення роботи полягає у створенні веб-орієнтованої статистичної системи завантаженості пунктів пропуску кордонів країни на платформі чат-боту, яка оперативно надають оновлену інформацію про пункти пропуску: завантаженість черг, час очікування на кордоні, пропуск вантажних та легкових авто тощо.

Список ключових слів: ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, МЕСЕНДЖЕРИ, ПЛАТФОРМА ДЛЯ СТВОРЕННЯ TELEGRAM-БОТІВ, СИСТЕМИ ШВИДКОГО ОБМІНУ ДАНИМИ, ЧАТ-БОТ

ABSTRACT

Explanatory note: 101 pages, 38 figures, 3 tables, 4 appendices, 16 sources.

The object of development: is a web-based statistical platform with geoinformation support.

The purpose of the thesis: development of a web-oriented statistical system for the workload of the country's border checkpoints on the chatbot platform.

The introduction describes the state of the problem and substantiates its relevance.

The first section includes the problem statement and the characterization of the subject area.

In the second section, a review and selection of technologies for web-based statistical platforms.

In the third section, a web-based statistical platform with geoinformation was developed and tested.

The practical significance of the work is to create a web-based statistical system for the congestion of the country's border checkpoints on the chatbot platform, which promptly provides updated information about the checkpoints: queue congestion, waiting time at the border, passage of trucks and cars, etc.

Keywords: INFORMATION TECHNOLOGY, MESSENGERS, PLATFORM FOR CREATING TELEGRAM BOTS, RAPID DATA EXCHANGE SYSTEMS, CHATBOT

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML - (HyperText Markup Language – мова розмітки гіпертексту) – стандартна мова розмітки документів у Всесвітній павутині;

PHP – Hypertext Preprocessor - «препроцесор гіпертексту», інструменти для створення персональних веб-сторінок;

MYSQL – вільна система управління базами даних;

CSS - Cascading Style Sheet – це каскадна таблиця стилів;

JavaScript - об'єктно-орієнтована скриптова мова програмування;

API – Application Programming Interface;

БД – база даних;

СУБД – система управління базами даних;

IDEF – Information Modeling – методологія моделювання інформаційних потоків.

.

ВСТУП

Актуальність роботи. У зв'язку зі складною ситуацією на кордонах пов'язаною з початком повномасштабної війни, створення веб-орієнтованої статистичної системи на платформі чат-боту для надання оновленої інформації про пункти пропуску на українських кордонах є дуже актуальним та корисним в наш час, особливо з урахуванням потреб українців, які активно виїжджають з країни через складну ситуацію.

Використання чат-боту в якості платформи для статистичної системи має свої переваги:

1. Зручність та доступність: Чат-боти є популярними інструментами комунікації, оскільки вони доступні на різних платформах, таких як мобільні пристрої та комп'ютери. Громадяни зможуть легко отримувати оновлену інформацію про пункти пропуску, використовуючи звичні для них засоби зв'язку.
2. Оперативність та актуальність: Чат-боти можуть надавати оперативну інформацію в режимі реального часу. Громадяни можуть швидко отримати оновлені дані про завантаженість черг, час очікування на кордоні, пропуск вантажних та легкових авто тощо безпосередньо у чаті.
3. Взаємодія та персоналізація: Чат-боти можуть забезпечити інтерактивну взаємодію з користувачами, запитуючи та враховуючи їхні персональні потреби. Це дозволяє забезпечити більш точну та індивідуалізовану інформацію про пункти пропуску.
4. Масштабованість: Чат-боти можуть обслуговувати одночасно багато користувачів, що робить їх ефективними для широкої аудиторії. Вони можуть виконувати багато завдань одночасно, надаючи актуальну інформацію в реальному часі.

Об'єктом дослідження є веб-орієнтована статистична системи на платформі чат-боту для надання оновленої інформації про пункти пропуску на українських кордонах.

Метою роботи є розробка веб-орієнтованої статистичної системи на платформі чат-боту для надання оновленої інформації про пункти пропуску на українських кордонах, для полегшення прийняття рішення потенційним клієнтам.

1 АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1 Основні особливості та проблеми сучасних веб-орієнтованих програмних проектів

Сьогодні існує широкий спектр програмних та технічних засобів для розробки веб-систем. Сучасна практика господарювання накладає вимоги до побудови інформаційних систем, які виявляються досить жорсткими. Для задоволення цих вимог інформаційна система повинна володіти наступними характеристиками:

- Розподілена модульна архітектура: система повинна бути розділена на модулі, що взаємодіють між собою, що сприяє підвищенню масштабованості та гнучкості системи.
- Централізований доступ та управління: система повинна забезпечувати централізований доступ до інформації та централізоване управління, що спрощує керування та забезпечує єдиною точкою контролю.
- Ефективне використання ресурсів: система повинна мінімізувати витрати на утримання, такі як обслуговування серверів, налаштування конфігурації робочих станцій, оновлення версій тощо.
- Інтеграція з іншими системами: система повинна підтримувати інтерфейси взаємодії з іншими системами та бути мультиплатформенною, що сприяє обміну даними та інтеграції з різними додатками та сервісами.
- Зручність для користувача: система повинна мати мінімальні вимоги до конфігурації робочих місць користувача і надавати зручний інтерфейс для взаємодії.
- Високі показники якості: система повинна відповідати високим стандартам якості, надійності, стабільності, безпеки та швидкодії.

На сьогоднішній день методика Web 3.0 набула значного поширення та впровадження. Вона передбачає проектування систем з урахуванням

мережевих взаємодій, і чим більше людей користується цими системами, тим краще вони працюють. Особливістю Web 3.0 є залучення користувачів до створення та перевірки контенту.

Web 3.0 слід розглядати як сукупність технологій, а не як окрему технологію. Його ключова ідея полягає в співпраці, колективному творчості, миттєвому поширенні та зворотному зв'язку зі споживачами інформації. Інтернет перетворюється на платформу та місце для спільної роботи, [1].

Характерною особливістю веб-орієнтованих інформаційних систем, зокрема систем управління контентом веб-сайтів, є наявність великої кількості документальної інформації. Ці документи часто генеруються динамічно на основі певних процедур.

Сьогодні стає очевидним, що сучасні веб-орієнтовані системи, зокрема системи управління контентом веб-сайтів, повинні все більше базуватися на семантиці предметної області та знаннях про неї. Це означає, що системи повинні розуміти контекст інформації, яку вони обробляють, та мати здатність взаємодіяти з користувачами та іншими системами на більш глибокому рівні.

Застосування семантичних технологій у веб-системах дозволяє досягти кращої організації та категоризації контенту, поліпшити пошукову функціональність, рекомендації та персоналізацію. Використання семантичних метаданих дозволяє системам розуміти сенс і зв'язки між різними елементами інформації, що сприяє покращенню якості обробки даних та забезпеченню більш точної та змістовної інформації для користувачів.

Додатково, сучасні веб-системи все частіше використовують технології штучного інтелекту, такі як машинне навчання та аналітику даних, для забезпечення автоматичного аналізу та обробки інформації, виявлення патернів та трендів, а також покращення взаємодії з користувачами.

Враховуючи швидкий розвиток технологій та зростання очікувань користувачів, веб-системи продовжують еволюціонувати, набуваючи більш

складних та інноваційних можливостей. Метою цих змін є поліпшення якості, зручності та значущості веб-досвіду користувача, щоб відповідати сучасним вимогам та задовольняти різноманітним потребам користувачів.

Одним з ключових напрямків розвитку веб-систем є забезпечення мобільності та адаптивності. В еру використання смартфонів і планшетних пристроїв, користувачі частіше звертаються до веб-систем через мобільні пристрої. Тому важливо, щоб веб-системи були розроблені з урахуванням адаптивного дизайну, який забезпечує оптимальне відображення та взаємодію з контентом незалежно від розміру екрану пристрою.

Окрім того, зростає значення безпеки веб-систем. У зв'язку зі збільшенням кількості кіберзагроз та злочинів в інтернеті, необхідно приділяти особливу увагу захисту даних користувачів, захисту від хакерських атак та забезпеченню конфіденційності. Веб-системи повинні використовувати шифрування даних, механізми аутентифікації та авторизації, а також проводити систематичні оновлення та моніторинг для виявлення потенційних вразливостей.

Також важливим аспектом є інтеграція з іншими системами та сервісами. Веб-системи повинні забезпечувати можливість обміну даними та взаємодії з іншими платформами, такими як соціальні мережі, платіжні системи, сервіси доставки тощо. Це дозволяє розширити функціональність веб-системи та забезпечити більш зручний досвід для користувачів.

Нарешті, зростає значення аналітики та використання даних. Веб-системи збирають велику кількість даних про користувачів, їх поведінку та взаємодію з системою.

Дані, що накопичуються веб-системами, можуть бути використані для отримання цінної інформації про користувачів, їхні вподобання та потреби. Аналітика даних дозволяє здійснювати сегментацію аудиторії, проводити персоналізовану рекламу та рекомендації, а також виявляти нові можливості та тенденції на основі аналізу зібраних даних.

Застосування машинного навчання та штучного інтелекту в аналітиці даних дозволяє автоматизувати процеси обробки та аналізу великого обсягу інформації. Алгоритми машинного навчання можуть виявляти складні зв'язки та патерни в даних, що допомагає виявити нові інсайти та зробити кращі рішення.

Загалом, розвиток веб-систем спрямований на покращення користувацького досвіду, ефективності та цінності інформації, забезпечення безпеки та інтеграції з іншими системами. Семантика, мобільність, безпека, аналітика даних та штучний інтелект є ключовими факторами, що визначають сучасні веб-системи і допомагають їм відповідати зростаючим вимогам та потребам користувачів.

Обрання систем, орієнтованих на веб-доступ, має просте пояснення, оскільки вони пропонують кілька переваг. По-перше, ці системи є кросплатформеними, тобто вони працездатні на різних платформах, таких як комп'ютери, смартфони, планшети і т. д. Для їх використання потрібен лише веб-браузер, логін та пароль.

По-друге, доступ до цих систем можна отримати з корпоративної локальної мережі (Intranet). Це означає, що вони доступні з будь-якого підключеного до мережі комп'ютера всередині організації. Крім того, якщо необхідний доступ до системи з Інтернету, то достатньо мати підключення до Інтернету, і можна користуватись системою.

Тому завдяки їхнім властивостям структура веб-сайту стає повністю керованою і підпорядкованою адміністратору. Це означає, що адміністратор має повний контроль над структурою веб-сайту і може легко додавати нові елементи. Для додавання нового елемента адміністратору достатньо лише ввести його інформацію та визначити необхідні параметри. Такий підхід спрощує адміністрування та розширення веб-сайту.

1.2 Історія виникнення чат-ботів

Чат-бот - це програма, яка використовується для взаємодії з користувачем через текстовий інтерфейс. Вона може мати елементи штучного інтелекту, такі як машинне навчання і обробка природної мови, або працювати на основі заздалегідь підготовлених відповідей, зокрема на основі правил.

Чат-боти можуть вести розмову з користувачем, відповідати на запитання, надавати інформацію і підтримку. Вони можуть бути застосовані в різних сферах, включаючи клієнтську підтримку, онлайн-консультування, автоматизацію завдань і багато іншого.

Завдяки постійному вдосконаленню технологій штучного інтелекту, чат-боти стають все більші розумними і здатними адаптуватись до потреб користувачів. Вони можуть вивчати попередні взаємодії, аналізувати тексти і враховувати контекст для надання більш точних і релевантних відповідей.

Загалом, чат-боти є потужним інструментом комунікації, який спрощує взаємодію з користувачами і може забезпечити швидку та ефективну підтримку у різних ситуаціях, [2] .

Класифікація чат-ботів може варіюватися в залежності від компанії або контексту, але можна виділити дві загальні класифікації з точки зору реалізації:

1. Бізнес-класифікація чат-ботів: ця класифікація відображає використання чат-ботів у різних сферах бізнесу і може включати такі типи як:
    - Чат-боти для обслуговування клієнтів: вони використовуються для надання підтримки клієнтам, відповіді на запитання, вирішення проблем і т.д.
    - Чат-боти для електронної комерції: вони можуть допомагати клієнтам з вибором товарів, здійсненням покупок, відстеженням замовлень і т.д.

- Чат-боти для маркетингу: вони використовуються для проведення маркетингових акцій, розсилки інформації, залучення нових клієнтів і т.д.

Бізнес-класифікація чат-ботів представлена на рисунку 1.1.

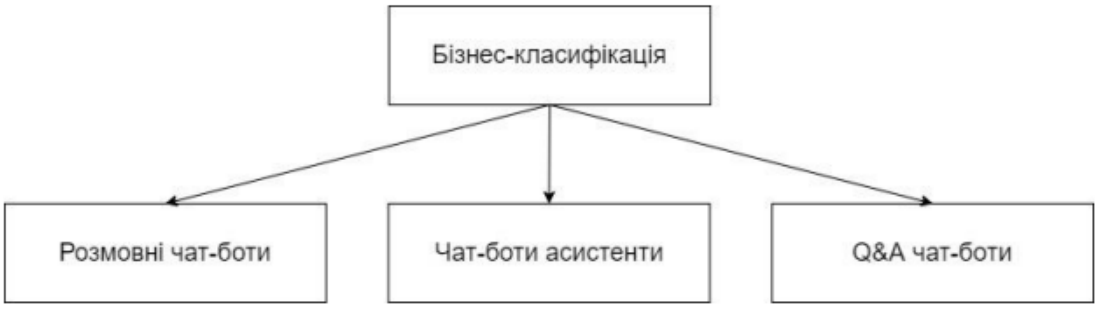


Рисунок 1.1 – Бізнес-класифікація чат-ботів

Бізнес-класифікація чат-ботів включає розмовні чат-боти, чат-ботів-асистентів і Q&A чат-ботів. Ось деякі деталі про кожен з цих типів:

1) Розмовні чат-боти: Ці чат-боти розроблені для відтворення розмови з людиною. Вони можуть вести неформальні діалоги, задавати запитання та реагувати на відповіді користувача. Розмовні чат-боти не мають конкретної мети і призначені для приємного та неформального спілкування.
2) Чат-боти-асистенти: Ці чат-боти мають конкретно визначену мету і призначені для надання допомоги або заміни в деяких бізнес-процесах. Вони можуть допомагати користувачам заповнювати веб-форми, надавати інформацію, вирішувати проблеми та виконувати завдання, пов'язані зі здійсненням певних операцій.
3) Q&A чат-боти: Ці чат-боти спеціалізовані на наданні простих відповідей на питання користувачів. Вони використовуються для швидкого та ефективного вирішення запитань, часто замінюючи розділи FAQ на веб-сайтах. Користувачі можуть задавати питання, а чат-боти надають відповіді на основі доступної інформації.

Враховуючи ці класифікації, підприємства можуть вибрати тип чат-бота, який найкраще відповідає їхнім потребам і цілям. Технічна класифікація чат-ботів представлена на рисунку 1.2.

Рисунок 1.2 – Технічна класифікація чат-ботів

2. Технічна класифікація чат-ботів включає наступні типи:

1) Основані на бізнес-правилах.

Цей тип чат-ботів, які основані на бізнес-правилах, працюють за заздалегідь визначеним сценарієм і мають структуру, схожу на дерево. Вони керують розмовою з користувачем в межах визначених варіантів і не дозволяють відхилитись від заданого шляху. В такому діалозі користувач може приймати рішення, але лише з обмеженого набору варіантів, які надаються чат-ботом у вигляді кнопок або меню. Ці чат-боти не можуть підтримувати вільний формат розмови, де користувач може відповідати у будь-якій формі або задавати довільні питання. Вони забезпечують структуровану та упорядковану взаємодію з користувачем у межах визначених правил і сценаріїв.

2) Основані на штучному інтелекті.

Чат-боти, які засновані на штучному інтелекті, використовують різні технології штучного інтелекту, такі як обробка природної мови (NLP), розуміння природної мови (NLU), нейронні мережі (NN) і т.д. Вони не мають заздалегідь визначеного сценарію розмови, оскільки їх поведінка базується на навчанні з використанням тренувальних даних.

Замість того, щоб мати жорстко визначені правила, ці чат-боти навчаються на основі попередніх діалогів, що використовуються в процесі тренування моделі машинного навчання. Чат-бот вирішує, яке питання задати

і яку відповідь надати, враховуючи знання, отримані з цих тренувальних даних. Це дозволяє їм мати більш гнучку поведінку і відповідати на різноманітні запитання.

Однак, основним недоліком чат-ботів, заснованих на штучному інтелекті, є потреба у великому обсязі тренувальних даних, щоб забезпечити логічну послідовність діалогу. Вони вимагають значних обсягів якісних даних для ефективної роботи. На сьогоднішній день такі чат-боти знаходяться в стадії прототипів, і продовжується дослідження для поліпшення їхньої ефективності та здатності розуміти та відтворювати природну мову.

3) Гібридні.

Гібридні чат-боти поєднують переваги обох типів чат-ботів. Вони ведуть розмову з користувачем за певним сценарієм, подібно до чат-ботів, заснованих на бізнес-правилах. Одночасно вони використовують штучний інтелект для розпізнавання намірів користувача та витягування цінних даних з повідомлень користувача, таких як повне ім'я, дата, період і т.д.

Гібридні чат-боти надають можливість більш гнучкої і розуміючої взаємодії з користувачами, оскільки вони можуть розпізнавати та використовувати цінну інформацію, що надається користувачем, для відповіді на запитання та надання персоналізованих послуг.

Гібридні чат-боти широко використовуються в комерційних додатках, оскільки їх гнучкість та здатність до адаптації робить їх ефективними у багатьох сферах, таких як електронна комерція, фінансові послуги, медична сфера та інші. Вони дозволяють покращити взаємодію з користувачами, забезпечити персоналізований сервіс та забезпечити ефективне виконання завдань у додатках.

Схема взаємодії користувача і чат-боту може бути наступною:

1) Користувач адресує свій запит через доступний канал комунікації, такий як розумний пристрій, асистент, мобільний телефон, звичайний дзвінок, месенджер або веб-чат.

2) Запит користувача містить намір, тобто конкретне бажання отримати відповідь на питання, послугу, товар або інший контент.
3) Відповідно до каналу комунікації, може знадобитися додаткова обробка або конвертація формату повідомлення. Наприклад, якщо користувач спілкується голосом, то текст повідомлення потрібно отримати за допомогою системи розпізнавання мови (ASR), а при синтезі мови (TTS) текстову відповідь потрібно перетворити на голосовий формат.

Канали комунікації можуть підтримувати візуальні елементи, такі як кнопки або картки товарів, з якими користувач може взаємодіяти. Це можливо, наприклад, у месенджерах або через спеціальні інтерфейси асистентів. Інтеграція з відповідними API дозволяє реалізувати цю можливість.

4) Отримавши запит користувача, чат-бот проводить обробку тексту, розпізнавання наміру та інші операції, необхідні для розуміння інтенції користувача та витягування необхідних даних.
5) На основі отриманих даних та логіки роботи чат-бота генерується відповідь, яка може бути текстовою, голосовою або комбінацією обох.
6) Користувач отримує відповідь чат-бота через вибраний канал комунікації. Це може бути текстове повідомлення, голосове повідомлення, візуальний елемент або їх комбінація, залежно від можливостей каналу.

Користувач може взаємодіяти з чат-ботом, надсилаючи нові запити та відповіді. Чат-бот обробляє ці повідомлення та продовжує діалог на основі отриманої інформації.

Процес взаємодії між користувачем і чат-ботом може продовжуватися, доки не буде досягнуто мети користувача або не буде завершено взаємодію.

В процесі взаємодії можуть бути використані додаткові сервіси або системи, такі як системи інтеграції зі сторонніми додатками або базами даних, щоб надати користувачу більш широкий функціонал чи інформацію.

Чат-бот може вести журнал взаємодії з користувачем, що дозволяє в подальшому аналізувати та вдосконалювати процес комунікації, а також збирати дані для статистики та аналітики.

Ця схема взаємодії користувача і чат-боту відображає загальний процес, але може варіюватися залежно від конкретної реалізації чат-бота та його функціональних можливостей.

Діалогова платформа виконує ключову роль у розумінні запитів користувача і ефективній обробці їх для формування відповідей. Вона використовує різноманітні технології, щоб досягти цієї мети. Ось кілька основних кроків, які здійснюються діалоговою платформою:

1) Нормалізація тексту: Запит користувача піддається обробці, де проводиться стандартизація та нормалізація тексту, щоб забезпечити єдність формату та усунути можливі неточності.
2) Морфологічний аналіз: Використовуються алгоритми та моделі для аналізу морфологічних особливостей тексту, таких як лематизація (приведення слів до базової форми) і визначення частин мови. Це допомагає уточнити смислове значення слів у контексті запиту.
3) Аналіз семантичної близькості: Застосовуються методи для визначення семантичної близькості між запитом користувача та іншими текстовими даними, такими як попередні запити або бази знань. Це допомагає зрозуміти сенс запиту та визначити його намір.
4) Ранжування гіпотез: Чат-бот може створити кілька гіпотез щодо наміру користувача на основі аналізу тексту. Ці гіпотези оцінюються і ранжуються за ймовірністю, щоб вибрати найбільш вірогідний намір.

5) Виділення іменованих сутностей: Виявлення іменованих сутностей, таких як імена людей, місця, дати або продукти, допомагає розуміти конкретні деталі запиту користувача.

Цикл оброблення запиту клієнта включає наступні події і дії:

1) Система отримує запит клієнта і передає його в модуль управління діалогом - DialogManager.
2) DialogManager завантажує контекст діалогу з бази даних на основі отриманого запиту.
3) Запит клієнта разом з контекстом передається в NLU-модуль для обробки, де визначається намір клієнта та його параметри. У випадку обробки нетекстових подій, таких як кнопки, цей крок може бути пропущений.
4) З використанням сценарію діалогу і отриманих даних, DialogManager визначає наступний найбільш підходящий стан (блок, екран, сторінку діалогу), що найкраще відповідає вислову клієнта.
5) Виконується логіка або скрипти відповідно до заданого сценарію чат-бота.
6) Якщо це передбачено логікою, викликаються зовнішні інформаційні системи або сервіси.
7) Генерується текстова відповідь з використанням макропідстановок і функцій узгодження слів на природній мові, що допомагають зробити відповідь більш природною і зрозумілою для користувача.
8) Остаточна відповідь відправляється клієнту, може бути у формі тексту, голосового повідомлення (за допомогою TTS) або повідомлення про виконану дію.

Цей цикл дозволяє системі ефективно обробляти запити клієнтів і надавати їм відповіді згідно з встановленим сценарієм та логікою чат-бота.

Управління ходом діалогу (DialogManager) є важливою частиною функціонування системи. Цей компонент визначає загальний контекст сказаного і забезпечує зв'язок з попередніми і наступними висловлюваннями в рамках діалогу.

DialogManager виконує кілька важливих функцій. Він зберігає і керує контекстом діалогу, що дозволяє системі враховувати інформацію, що була вказана попередньо, і забезпечує послідовність обміну повідомленнями між системою і користувачем.

Окрім того, DialogManager може використовувати техніки заповнення слотів (slot filling), що дозволяють отримати додаткові дані, необхідні для обробки запиту. Ці дані можуть бути отримані з фрази клієнта, контексту попередніх фраз або явно запитані у користувача. Наприклад, якщо користувач хоче замовити продукт, система може запитати його адресу доставки, щоб виконати замовлення повністю.

Управління ходом діалогу дозволяє системі розуміти контекст і інтенцію користувача, використовувати попередню інформацію і забезпечувати більш персоналізовану та зручну взаємодію. Це допомагає зробити обробку запитів більш ефективною і точною, а також підвищує якість досвіду користувача з системою.

1.3 Месенджер Telegram

Після 2015 року Telegram став одним з най швидкозростаючих майданчиків для комунікації у мобільних соціальних мережах. Заснований у 2013 році, Telegram відразу привернув увагу користувачів своєю мобільною платформою месенджера, прив'язаною до телефонного номера.

На відміну від інших подібних платформ, Telegram забезпечував шифрування трафіку, що зробило його привабливим для користувачів, які цінували приватність та безпеку передачі даних. Також варто відзначити можливість створювати групи зі збереженням конфіденційності, що

привернуло увагу користувачів «невидимого Інтернету», які використовували технології прихованої комунікації.

Серед користувачів Telegram були як звичайні користувачі, так і кіберзлочинці. Для звичайних користувачів, Telegram пропонував простий інтерфейс та систему захисту даних, що дозволяє спілкуватися в чатах без необхідності заповнення профілю та збирання великої кількості інформації про користувача.

За своєю природою, Telegram був розроблений як месенджер і став популярним у широкому колі користувачів, незалежно від їх мотивацій або використання платформи.

Так, у Telegram для привернення аудиторії були створені канали-групи. Спочатку користувачі створювали канали для спілкування в груповому чаті зі своїми контактами або спільнотами з подібними інтересами. Згодом, мережеві ЗМІ також почали створювати свої канали у Telegram, з метою привернення платіжно-здатної мобільної аудиторії.

Відкриття каналів ЗМІ у Telegram стало одним зі способів привернення уваги до своїх змістових матеріалів та залучення аудиторії. За допомогою каналів-груп ЗМІ могли поширювати свої новини, статті, аналітику та інші контент-матеріали серед користувачів Telegram.

Зростання популярності каналів-груп серед ЗМІ було пов'язано з тим, що з часом стало складніше залучити платіжно-здатну мобільну аудиторію у традиційні медіа-канали. Telegram надавав можливість споживачам і контент-постачальникам взаємодіяти безпосередньо, без посередництва традиційних медіа-організацій.

Однією з проблем, з якими стикалися ЗМІ в Telegram, було агрегування і організація контенту. У порівнянні з індивідуальними користувачами, які могли публікувати новини і повідомлення в будь-який час і без певного принципу, ЗМІ мали потребу акцентувати увагу на своєму контенті та плановано розміщувати анонси своїх публікацій на веб-сайтах.

Для вирішення цієї проблеми ЗМІ знаходили різні інструменти автоматизації. Один із способів був використання ботів у Telegram, які дозволяли автоматично публікувати анонси новин та інший контент на каналах-групах ЗМІ. Це дозволяло ЗМІ планувати і організовувати свої повідомлення, створюючи певний графік розміщення контенту і підтримуючи постійну присутність в каналах користувачів.

Застосування автоматизації у формі ботів сприяло зручності і ефективності розміщення контенту ЗМІ в Telegram, дозволяючи їм краще організувати свою присутність і залучати увагу аудиторії до власного контенту, [3]. Borgatti, S. Analyzing social network

Боти стали цінним інструментом для ЗМІ в Telegram, дозволяючи автоматизувати процес публікації контенту і зберігати консистентність в передачі інформації. Це було особливо корисно для ЗМІ, які раніше працювали з телеграм-каналами і вже мали певну аудиторію в месенджері.

Завдяки програмним скриптам, ботам було можливо передати інструкції щодо виведення матеріалів, результатів пошуку або мультимедіа. Журналістам залишалося займатися своїми прямими обов'язками, а бот самостійно публікував контент у відповідності до встановлених правил.

Зростаючий інтерес до використання ботів призвів до створення окремого середовища керування ботами в Telegram, відомого як Botfather. Це середовище надавало можливість писати програми-відправники і створювати власні ЗМІ в соціальній мережі. Це зробило процес створення і управління ботами доступним для будь-якого користувача з базовими навичками роботи з програмними інтерфейсами додатків.

Telegram є однією з найпопулярніших мобільних соціальних мереж з великою кількістю користувачів, особливо в Західній Європі. Незважаючи на свою молодість, він зарекомендував себе як потужний месенджер з високим рівнем захисту даних.

Засновники Telegram акцентують увагу на безпеці свого додатку і стверджують, що він має один з найкращих захистів серед подібних

продуктів на ринку. Відкритий вихідний код дозволяє розробникам та спеціалістам з безпеки перевіряти програму на наявність потенційних вразливостей і виявляти їх.

Однією зі специфічних особливостей Telegram є відсутність реклами, що робить його привабливим для користувачів, які не бажають бути зайняті рекламними повідомленнями. Крім того, наявність відкритого вихідного коду дозволяє спільноті розробників перевіряти програму і сприяти її покращенню.

Ці фактори сприяють підвищенню довіри користувачів до Telegram, але варто зазначити, що довіра також будується на основі репутації, історії появи та використання додатку. Команда розробників має великий вплив на довіру користувачів, оскільки вони відповідають за безпеку та функціональність месенджера.

В Telegram існує функція "секретний чат", яка забезпечує наскрізне шифрування повідомлень. Ці чати мають додаткові заходи безпеки, такі як автоматичне видалення повідомлень після певного часу і неможливість їх відновлення. Це дає користувачам більшу приватність та конфіденційність у спілкуванні.

Однак, за замовчуванням наскрізне шифрування не включене, оскільки секретні чати пов'язані з конкретними пристроями і не можуть бути перенесені на інші пристрої. Це забезпечує зручність використання месенджера на різних пристроях, але водночас обмежує можливість продовжити розмову на іншому пристрої.

У Telegram також є різні методи авторизації і захисту облікового запису. Користувачі можуть створювати облікові записи та авторизуватися за допомогою текстового повідомлення, що містить автентифікаційний код. Додатково, Telegram надає можливість використовувати двоетапну верифікацію, яка дозволяє встановити пароль для кожного входу в обліковий запис, що підвищує рівень безпеки.

Ці функції дозволяють користувачам Telegram забезпечити приватність та безпеку свого облікового запису та комунікацій.

Переваги Telegram, які виникли протягом його існування:

1. Конфіденційність: Всі повідомлення в Telegram зашифровані, а функція "секретний чат" забезпечує додаткову приватність. Повідомлення також можуть бути автоматично видалені через певний час, що покращує конфіденційність спілкування.
2. Оптимізація: Сервери Telegram розташовані по всьому світу, що забезпечує надійну та стабільну роботу месенджера.
3. Швидкість: Telegram має високу швидкість передачі повідомлень завдяки оптимізації своєї інфраструктури.
4. Прозорість: Telegram має відкритий протокол, що означає, що його протокол може бути оглянутий та перевірений відкритою спільнотою розробників.
5. Доступність: Telegram безкоштовний для використання і не містить інтегрованої реклами. Він доступний для користувачів без додаткових витрат або обмежень.
6. Необмеженість: Telegram не накладає великих обмежень на розмір повідомлень та файлів, які можуть бути надіслані через месенджер. Користувачі можуть передавати великі файли без проблем.

Всі ці переваги сприяли популярності та зростанню користувацької бази Telegram.

1.4 Чат-боти в месенджері Telegram

У червні 2015 року Telegram відкрив платформу для створення ботів, що сприяло активному розвитку цієї функції у месенджері. Боти в Telegram є програмними скриптами, які взаємодіють з користувачами, відповідаючи на їх команди та виконуючи різноманітні завдання.

Цей функціонал був широко використаний, в тому числі й ЗМІ, які зацікавилися можливістю створення власних чат-ботів для активного

залучення мобільної аудиторії на свої веб-сайти. Використання ботів дозволяє економити час та зусилля, оскільки вони виконують роботу швидше і з більшою точністю порівняно з людьми.

Чат-боти в Telegram можуть бути налаштовані на надання адекватних відповідей за допомогою людської мови, якщо вони розпізнають та підтримують такі команди. Ці боти дозволяють імітувати людську активність в чатах та використовуються в різних сферах, таких як комп'ютерні ігри, інтернет-аукціони, реклама, електронна біржова торгівля тощо.

Отже, чат-боти в Telegram є одним з різновидів ботів, якими управляють програми замість людей, і вони знаходять широке застосування у месенджері.

боти в Telegram дійсно мають широкий спектр функцій і можуть виконувати різні дії. Вони можуть бути використані для перекладу тексту, коментування новин, навчання користувачів, проведення тестувань, пошуку та знаходження інформації, відповіді на запитання, гри та розваги, а також інтеграції з іншими сервісами.

Одна з переваг використання ботів в Telegram полягає в тому, що користувачам не потрібно виходити з месенджера для вирішення певної задачі. Замість цього вони можуть мати діалог з ботом в тому ж чаті, де спілкуються з іншими людьми. Це дозволяє користувачам одночасно взаємодіяти з ботами та людьми без зайвих витрат часу та зусиль.

Боти в Telegram можуть бути додані до існуючих групових чатів або поділитися з друзями та колегами. Платформа Telegram надає багато можливостей для створення прикладних рішень з використанням ботів.

Отже, боти в Telegram є розумними помічниками, які здатні виконувати різноманітні завдання. Вони працюють незалежно один від одного, але спільно утворюють команду асистентів, яким можна давати доручення, а вони відповідають на запити користувачів швидко, точно і чітко.

Боти в Telegram можуть діяти злагоджено і виконувати різні дії у визначений час. Вони можуть бути налаштовані для доставки свіжих новин,

прогнозу погоди, інформації про дорожні затори вранці, а ввечері надавати рекомендації щодо фільмів, анекдоти або рецепти.

Це робить ботів потужним інструментом для організації часу користувача та автоматизації повторюваних дій. Завдяки розкладу та налаштуванням, користувачі можуть отримувати потрібну інформацію та виконувати рутинні завдання без необхідності активно контактувати з ботом.

Початок роботи з ботом в Telegram є простим. Користувачам потрібно просто вибрати бота з каталогу, перейти за посиланням або знайти його за іменем через пошук. Зазвичай для запуску бота достатньо ввести команду /start або натиснути кнопку "Старт" на віртуальній клавіатурі. Боти також можуть використовувати програмовані кнопки на віртуальній клавіатурі, що робить їх більш зручними для взаємодії, надаючи інтуїтивно зрозумілий інтерфейс.

Загалом, боти в Telegram забезпечують легку і зручну взаємодію з користувачами, допомагають організувати час і автоматизувати рутинні завдання.

Боти в Telegram можуть надсилати користувачам інформацію про себе, інструкції та список доступних команд. Вони можуть виводити кнопки з командами на екрані для зручної навігації.

Вони мають потенціал бути корисними в різних сферах життя. Вони дозволяють пов'язувати об'єкти реального світу з користувачем. Особливо великі можливості відкриваються з розвитком Інтернету та інших галузей.

Люди використовують ботів Telegram для організації взаємодії, наприклад, дозволяючи замовникам бачити процес виконання робіт і контролювати робочий процес.

Застосування ботів обмежується лише уявою розробника. Простим прикладом може бути створення міні-представництва редакції або проекту. Ви можете надати опис свого видання або проєкту, контактні дані та налаштувати оновлення з випуском матеріалів.

За допомогою ботів можна проводити опитування, встановлювати нагадування або залучати бота до дискусій, передавши йому параметри для відповідей.

В цілому, боти Telegram є потужним інструментом, який може бути використаний у різних сферах життя для спрощення комунікації, автоматизації дій та вирішення різноманітних завдань.

Безпека використання чат-ботів є важливим питанням для багатьох користувачів. Однак, в межах мессенджера Telegram, боти не мають здатності виконувати небезпечні дії, оскільки вони функціонують на базі облікових записів користувачів, працюючи за алгоритмічною логікою. Ви можете видалити бота зі свого списку діалогів або заблокувати його, як і будь-кого іншого користувача, якщо це необхідно.

Важливо зазначити, що боти не можуть самостійно ініціювати надсилання повідомлень комусь. Користувач є ініціатором діалогу і самостійно взаємодіє з ботом. Якщо бот перебуває у групі, він не може бачити приватні розмови між учасниками групи, поки жоден з них не звернеться до бота. Це означає, що розробник бота не може перехоплювати повідомлення та стежити за конкретною людиною без її взаємодії з ботом.

Так само, як і з будь-яким іншим інтернет-сервісом, завжди варто бути обережним і перевіряти джерела, на які вас направляють боти. Якщо ви впевнені в надійності та джерелах інформації, які надає бот, використання чат-ботів у Telegram може бути безпечним та зручним способом отримання швидких відповідей та автоматизації дій.

Деякі відмінності між ботами і звичайними обліковими записами користувачів в Telegram, [4]:

1. Відсутність статусів онлайн і "був в мережі": Замість цього, боти відображають свою назву, що дозволяє користувачам розрізняти спілкування з людьми і програмами.

2. Обмежене місце на серверах: Ботам виділяється обмежений простір на серверах Telegram, і після певного терміну їх дані можуть бути видалені після обробки.
3. Боти не можуть самостійно починати спілкування: Користувач повинен або додати бота до групи, або першим почати діалог з ним.
4. Обов'язковий суфікс "bot" в імені бота: Ім'я бота повинно завжди закінчуватися на "bot" (наприклад, @musicbot).
5. Неотримання всіх повідомлень під час додавання в конференцію: за замовчуванням, бот не отримує всі повідомлення, які були надіслані до конференції до його додавання. Він починає обробку повідомлень, які надійшли після його додавання.

Ці відмінності визначають особливості функціонування ботів в Telegram і допомагають користувачам розрізняти їх від звичайних користувачів.

1.5 Аналіз аналогів

Відсутність авіаційного сполучення та інші сучасні обставини навантажують автомобільні пропускні пункти України з прикордонними державами. Для зручності було розроблено декілька ресурсів, які допоможуть знайти найшвидший спосіб перетину кордону.

Держприкордонслужба запустила оновлену інтерактивну карту на своєму веб-сайті. Це важливий інструмент для громадян, які планують виїзд з України, оскільки вони тепер можуть отримати актуальну інформацію про пункти пропуску та їх завантаженість.

Оновлення карти кожні три години гарантує, що інформація, яку користувачі бачать, є свіжою і точною. При натисканні на позначку пункту пропуску, громадяни можуть дізнатися про час його роботи, кількість легкових і вантажних автомобілів перед пропускними пунктами та швидкість оформлення легкових автомобілів за годину. Також наявні зображення з камер

відеоспостереження для окремих пунктів пропуску, що дозволяє користувачам перевірити поточну ситуацію на місці.

Це важлива інформація для подорожуючих та тих, хто має необхідність перетинати кордон. Завдяки такому інструменту громадяни можуть краще планувати свою поїздку та обирати оптимальний час та маршрут.

Оновлена інтерактивна карта Держприкордонслужби, [5] є важливим кроком у забезпеченні доступу до актуальної інформації та полегшенні процесу перетину кордону, рисунок 1.3.

Згідно з повідомленням, на інтерактивній карті Держприкордонслужби використовуються різні кольори, щоб позначити стан завантаженості пунктів пропуску та час очікування транспортних засобів. Ось як розподіляються кольори та їх значення:

- зелений колір: Використовується для позначення пунктів пропуску, де час очікування становить не більше однієї години. Це означає, що рух через ці пункти пропуску відбувається швидко, без значних затримок.
- синій колір: Використовується для пунктів пропуску, де час очікування транспортних засобів становить до 2-х годин. Це середній рівень завантаженості, де можуть бути невеликі затримки, але загалом рух відбувається досить швидко.
- червоний колір: Використовується для пунктів пропуску, де завантаженість висока, а час очікування автомобілів складає більше 2-х годин. Це означає, що рух через ці пункти пропуску може бути значно уповільнений, і чекати доведеться довше.

Таке розподілення кольорів дозволяє користувачам швидко отримати уявлення про загальний стан завантаженості та час очікування на різних пунктах пропуску. Вони можуть зробити інформований вибір щодо свого маршруту та планування часу перетину кордону.

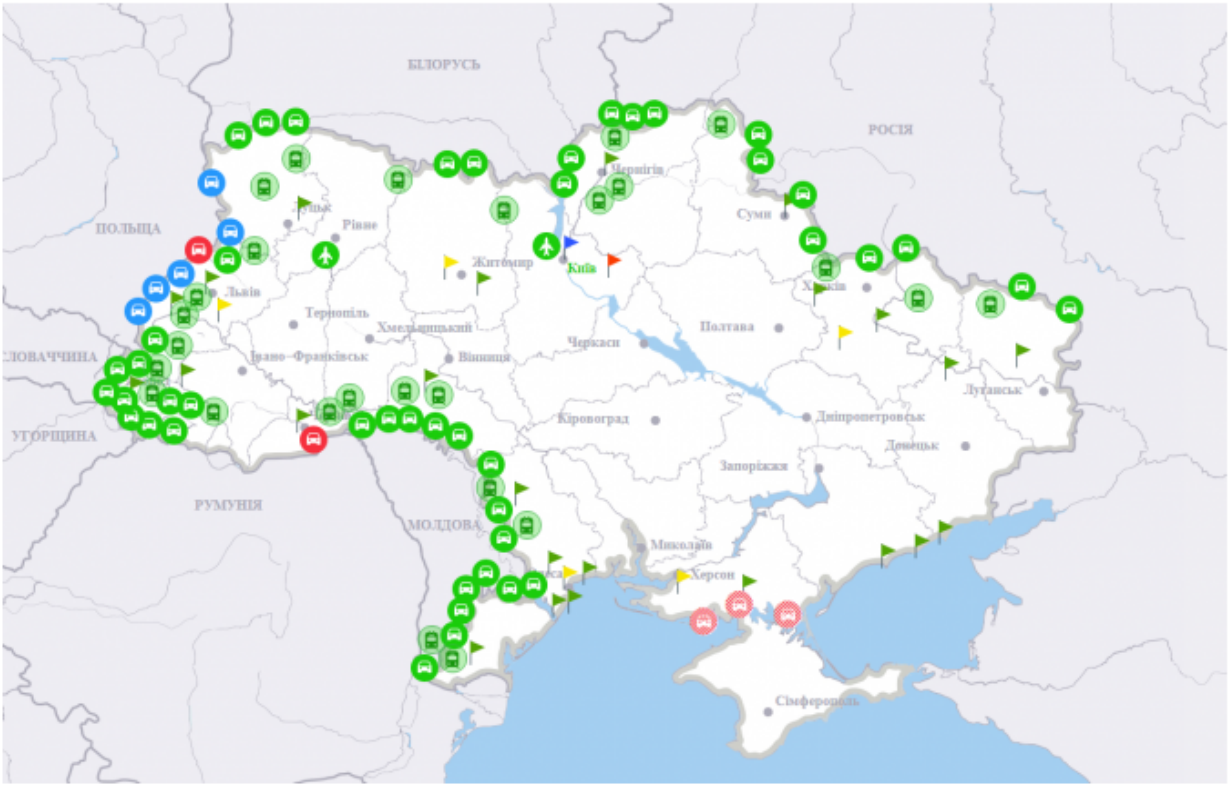


Рисунок 1.3 – Інтерактивна карта Держприкордонслужби

Для зручності користувачів також створений телеграм-ботом Кордони України Інфо, [6], рисунок 1.4.

Для початку потрібно натиснути “start” для активації бота та потім обрати опцію “пропускні пункти”. Після цього вибери країну, до якої потрібно виїхати. Бот містить корисні покликання для кожної країни, в яку можливий виїзд з України. Крім цього, в ньому можна знайти інформацію про трансфери, житло, необхідні документи для перетину кордонів з Україною і далі, про статус українця за кордоном та пільги.

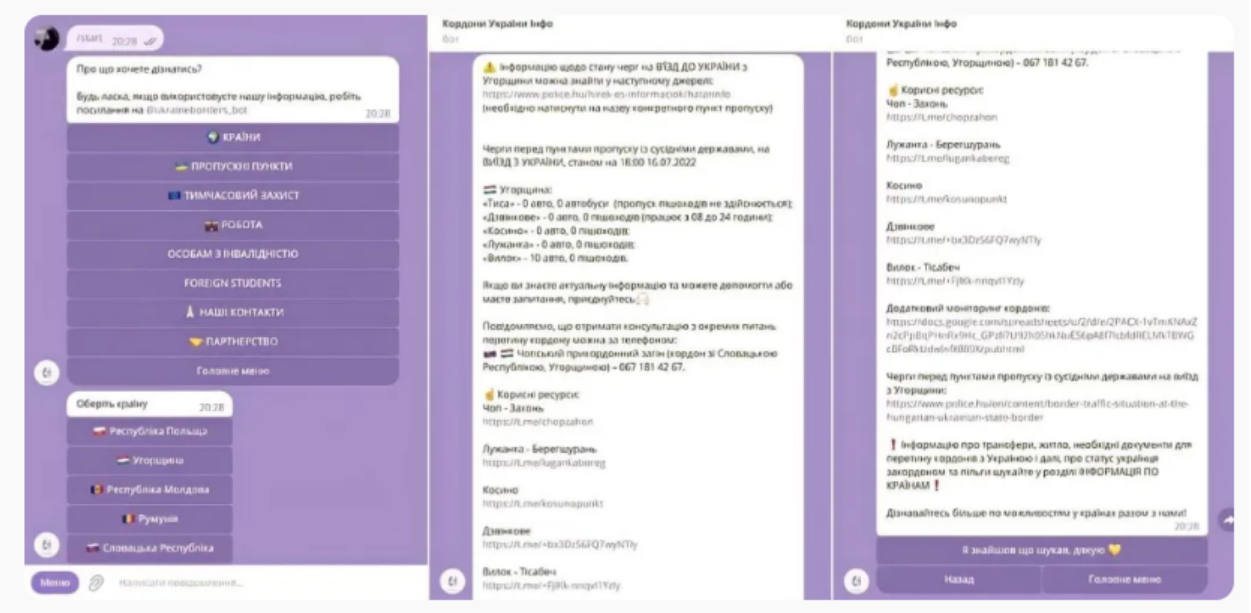


Рисунок 1.4 – Приклад роботи з чат-ботом

1.5 Висновки по першому розділу

В першому розділі проведений аналіз стану області рішення задачі. Прозорий та доступний обмін інформацією про ситуацію на пунктах пропуску є важливим для населення, бізнесу та державних органів як у мирний час, так і в надзвичайних періодах, зокрема під час карантинних обмежень. Отримання точних та орієнтовних даних про завантаженість пунктів пропуску, включаючи інформацію про час очікування, є важливим фактором при плануванні перетину кордону.

На Західному кордоні України діючі сервіси митної та прикордонної служби, які надають інформацію про завантаженість пунктів пропуску, зараз мають спільний недолік - вони подають інформацію у вигляді кількості транспортних засобів або осіб, а не в орієнтованому часі очікування. Це різниться з інформацією, яка надається на ресурсах суміжних країн-партнерів.

Крім того, варто відзначити, що навіть у межах одного відомства дані щодо завантаженості конкретних пунктів пропуску можуть відрізнятися. Це може створювати певні труднощі та незручності для користувачів, які

потребують точної інформації для планування своїх маршрутів та перетинання кордону.

Розглянутий сучасний інструментарій, такий як інтерактивна карта Держмитслужби, мають свої обмеження та можуть бути недостатньо оновленими. Якщо дана карта не оновлюється в режимі реального часу та інформація на ній пізніше надходить порівняно з іншими джерелами, це може ускладнити планування подорожей та перетинання кордону.

Також, камери відеоспостереження на деяких пунктах пропуску наразі не функціонують з міркувань безпеки. Це може обмежити доступ до візуальної інформації про поточну ситуацію на пунктах пропуску.

Неповнота інформації щодо змін у правилах перетину кордону, зокрема, скасування платежів на ввезення автомобілів, також може викликати певні труднощі та незручності для громадян.

Враховуючи ці проблеми, важливо прагнути до постійного вдосконалення інформаційних ресурсів та забезпечення достовірної, своєчасної та повної інформації для громадян та бізнесу, що планують перетинати кордон.

2 ОГЛЯД ТА ВИБІР ТЕХНОЛОГІЙ ДЛЯ ВЕБ-ОРІЄНТОВАНОЇ СТАТИСТИЧНОЇ ПЛАТФОРМИ

2.1 Огляд та вибір серверних платформ

Сьогодні існує багато платформ серверних платформ для розробки веб орієнтованих систем. Прикладом сучасних систем є чат боти. Кожна з існуючих платформ має свої переваги та недоліки, які необхідно розглянути для вибору найбільш підходящої для конкретної розробки. У цьому розділі ми розглянемо декілька платформ серверних платформ для розробки чат ботів та порівняємо їхні переваги та недоліки.

Dialogflow [7] є однією з найбільш популярних платформ серверних платформ для розробки чат ботів. Вона має багатий набір інструментів для створення чат-ботів та інтеграції з різними платформами, такими як Facebook, Telegram, Slack тощо. Dialogflow також має можливість розпізнавання мовлення за допомогою машинного навчання, що робить його популярним серед розробників.

Переваги [7]:

- легкість використання та інтеграції з різними платформами;
- розпізнавання мовлення за допомогою машинного навчання;
- широкий набір інструментів для створення чат-ботів.

Недоліки [7]:

- відсутність можливості розгортання на власному сервері;
- відсутність повної контролю над даними користувачів;
- microsoft bot framework

Microsoft Bot Framework [8] є ще однією популярною платформою серверних платформ для розробки чат-ботів. Вона має можливість інтеграції з різними платформами, такими як Skype, Slack, Telegram, та інші. Microsoft Bot Framework також надає багатий набір інструментів для створення чат-ботів.

Переваги [8]:

- легкість використання та інтеграції з різним;
- відкритий код, що дозволяє змінювати та налаштовувати різні елементи фреймворку;
- наявність можливості розгортання на власному сервері.

Недоліки [8]:

- обмеження у можливостях розпізнавання мовлення;
- не такий широкий набір інструментів, як у dialogflow.

Amazon Lex [9] - це платформа серверних платформ для розробки чат-ботів, створена Amazon. Вона має можливість інтеграції з Amazon Web Services та іншими платформами. Amazon Lex також має можливість розпізнавання мовлення за допомогою машинного навчання.

Переваги:

- легкість використання та інтеграції з amazon web services та іншими платформами;
- розпізнавання мовлення за допомогою машинного навчання.

Недоліки:

- високі витрати на використання amazon web services;
- обмеження у можливостях налаштування та змінення функціональності.

Наступні поширені та часто вживані серверні платформи для розробки серверних додатків та чат-ботів: ASP.NET та PHP. З одного боку, PHP є мовою програмування з відкритим кодом, яка використовується для веб-розробки та може бути вбудована в HTML. З іншого боку, ASP.NET має бренд Microsoft та є серверним інструментом веб-розробки з вихідним кодом.

PHP [10] має кілька переваг:

- швидкий розвиток;
- багатофункціональність;
- забезпечення безпеки додатків.

Цей фреймворк також простий в обслуговуванні та має велику спільноту розробників, які можуть допомогти за потреби. Однак, основна проблема

PHP полягає в низькій швидкості додатка та великому часі для опанування фреймворку.

З іншого боку, ASP.NET має свої переваги, такі як швидкісні додатки, кросплатформність та простота добавлення та видалення функцій. Але одним із головних недоліків цієї платформи є прогалини в документації та невелика спільнота.

Один з головних критеріїв порівняння цих двох фреймворків - це швидкодія. При цьому, продуктивність будь-якого фреймворку залежить від якості написання коду. ASP.NET набагато продуктивніший за PHP, оскільки підтримує паралельне програмування, тоді як у PHP його немає.

При порівнянні двох фреймворків, безпека є важливим фактором. ASP.NET має вбудовані функції безпеки, такі як автентифікація та авторизація, обмеження доступу до ресурсів, підтримка шифрування даних та запобігання атакам типу SQL injection і Cross-site scripting. Ці функції вбудовані в фреймворк і не вимагають додаткової настройки або програмування. ASP.NET також має можливості для моніторингу безпеки та реагування на потенційні загрози.

PHP також містить інструменти безпеки, такі як функції шифрування, обмеження доступу та захист від атак типу SQL injection і Cross-site scripting. Однак, PHP не має вбудованих функцій безпеки, що означає, що розробник повинен вручну налаштувати та програмувати функції безпеки для свого додатку. Це може призвести до того, що деякі розробники можуть не налаштувати безпеку належним чином, що зробить їх додатки вразливими до атак. В табл.2.1 наведено порівняння рівня безпеки між двома фреймворками.

Отже, хоча обидва фреймворки мають інструменти для забезпечення безпеки, ASP.NET набагато кращий в забезпеченні безпеки, оскільки він має вбудовані функції безпеки, які не вимагають додаткової настройки або програмування. З іншого боку, PHP [10] вимагає від розробника належного програмування функцій безпеки для його додатку, що може бути вимогливим і витратним.

Таблиця 2.1 – Порівняння сучасних технологій розробки веб-сайтів

Технологія / Характеристика	PHP	JSP	ASP.NET
Багатоплатформеність	+	+	-
Продуктивність	+	+/-	+/-
Простота використання	+	+/-	+/-
Наявність доступних програмних бібліотек	+	+	+
Розподіл дизайну та логіки	+/-	+/-	+
Відкритий код	+	-	-

Наступним критерієм для порівняння є вартість. У цьому питанні PHP очевидний лідер, оскільки це безкоштовний фреймворк, в той час як ASP.NET вимагає певної плати за розміщення. Хоча комісія, яку стягує Microsoft, є досить невеликою, порівняно з безкоштовним варіантом вона є недоцільною. Крім того, ASP.NET має інші переваги перед PHP, такі як:

- свобода вибору мови програмування, включаючи c#, visual basic.net, C++ тощо;
- потокове виконання коду;
- високі заробітні плати, оскільки на ринку праці недостатньо кваліфікованих розробників asp.net.

2.2 Огляд та вибір хмарних сховищ

В останні роки хмарні технології стали дедалі більш популярними серед розробників, оскільки вони надають зручний та доступний спосіб зберігання та обробки даних. проведемо огляд та проаналізуємо засоби хмарного сховища для розробки чат-бота в месенджері Telegram.

Вибір хмарних сховищ для зберігання систем керування базами даних (СУБД) не обмежується декількома продуктами. На ринку баз даних доступно безліч хмарних продуктів, починаючи від спеціально розроблених платформ СУБД, які створені для виконання унікальних завдань, до

технологій загального призначення, які мають широкий спектр застосувань. Одним з популярних засобів для хмарного сховища є Amazon S3 (Simple Storage Service). Цей сервіс надає безліч можливостей для зберігання та керування даними. Наприклад, він підтримує різні типи даних, включаючи зображення, відео, аудіо та тексти. Крім того, Amazon S3 забезпечує високу доступність даних та захист від відмов [11].

Іншим популярним сервісом для хмарного сховища є Google Cloud Storage. Цей сервіс надає зручний та швидкий спосіб зберігання та керування даними. Google Cloud Storage підтримує різні типи даних, включаючи зображення, відео, аудіо та тексти. Крім того, він забезпечує високу доступність даних та можливості для забезпечення безпеки даних.

Ще одним засобом для хмарного сховища є Microsoft Azure Blob Storage. Цей сервіс надає можливість зберігання та керування даними будь-якого типу та розміру. Microsoft Azure Blob Storage забезпечує високу доступність даних та можливості для забезпечення безпеки даних.

Порівняємо три найпопулярніші постачальники DBaaS і DWaaS це: AWS (Amazon Web Service, рис.2.1), Microsoft та Oracle за ціновою політикою, доступні сервіси для даного проекту, механізм міграцій БД.

За ціновою політикою [11]:

1. AWS: Вартість послуг залежить від типу бази даних та кількості ресурсів, які ви використовуєте. Ціни можуть коливатися від безкоштовних до кількох тисяч доларів на місяць.

2. Microsoft: Ціни на послуги від Microsoft також залежать від типу бази даних та кількості ресурсів. Вартість послуги SQL Database від Microsoft починається від $5 на місяць.

3. Oracle: Ціна на послуги Oracle Cloud також залежить від типу бази даних та обсягу ресурсів. Вартість послуги Oracle Database починається від $175 на місяць.

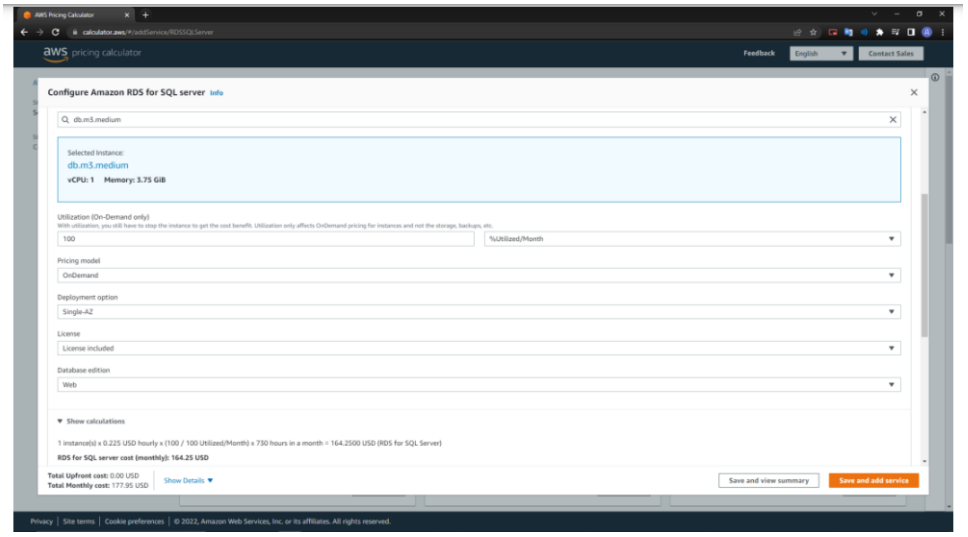


Рисунок 2.1 – Фрагмент Amazon Web Services

Зо доступними сервісамидля даного проекту:

1. AWS: AWS пропонує широкий спектр сервісів для різних типів баз даних, таких як SQL, NoSQL та графові бази даних.

2. Microsoft: Microsoft пропонує послуги SQL Database, Azure Cosmos DB, PostgreSQL, MySQL та MariaDB.

3. Oracle: Oracle пропонує послуги Oracle Database, MySQL, NoSQL Database Cloud та Autonomous Data Warehouse.

За механізмом міграцій БД [12]:

1. AWS: AWS пропонує сервіс Database Migration Service, який дозволяє легко мігрувати бази даних з інших постачальників до AWS.

2. Microsoft: Microsoft пропонує різні інструменти для міграції баз даних, такі як Azure Database Migration Service та Azure Site Recovery.

3. Oracle: Oracle також пропонує інструменти для міграції баз даних, такі як Data Pump та GoldenGate.

Oracle, Microsoft та AWS надають всі необхідні сервіси для роботи з базами даних та API. Для Microsoft це Azure SQL Database для баз даних та App Services для API. Для Oracle це MySQL Database Service для баз даних та Application Development - API Manager для API. Для AWS це RESTful API для API та Amazon RDS для SQL Server для баз даних. Інформацію, отриману з цих джерел, ми можемо представити у вигляді таблиці 2.2.

Таблиця 2.2 – Порівняльний аналіз сервісів СУБД

Критерій	Сервіри		
	Oracle	Azure	Microsoft
Цінова політика	46$	19$	178$
Відповідність	+	+	+ (але за додаткову оплату)
Тип БД для мапінгу	Oracle SQL Developer, IBM Db2	Sql server, MySQL PostgreSQL	Більшість можливих локальні міграції
Сервіси	+	+	+

Перед вибором хмарного сховища важливо враховувати їх недоліки. У випадку AWS можуть виникати складнощі з адмініструванням через широкий та глибокий набір хмарних служб СУБД. Крім того, AWS має обмежені локальні пропозиції через AWS Outposts, що може становити проблему для клієнтів зі значними інвестиціями у власні центри обробки даних. Конкуренти, зокрема Google, Microsoft та Oracle, починають зменшувати ціни на AWS, що також може впливати на вибір хмарного сховища.

Щодо Oracle, ліцензування та переговори щодо контрактів завжди були складними та надто напруженими для клієнтів, і ця проблема залишається актуальною і для хмарних пропозицій. Крім того, частка ринку хмарних систем Oracle є найнижчою порівняно з AWS, Microsoft і Google. У Oracle відсутні гетерогенні параметри СУБД, хоча зараз він використовує більше багатохмарний підхід із власними службами баз даних.

Таким чином, можна зробити висновок,що AWS, Microsoft та Oracle - це три найпопулярніших постачальників DBaaS та DWaaS. Кожен з них пропонує різні послуги та ціни, тому вибір залежить від потреб вашого

проекту. Механізм міграції баз даних також є важливим критерієм при виборі постачальника.

При розробці чат-бота в Telegram, важливо мати зручний та безпечний спосіб зберігання даних. Вибір сервісу для хмарного сховища залежить від потреб розробника та можливостей проекту.

2.3 Порівняльний аналіз веб-орієнтованих систем на платформі чат-ботів

Веб-орієнтовані системи на платформі чат-ботів стають все популярнішими з кожним роком. Чат-боти стали незамінними в різних галузях бізнесу, таких як клієнтська підтримка, продажі, маркетинг та інші.

У цьому пункті ми порівняємо дві веб-орієнтовані системи на платформі чат-ботів: Dialogflow від Google та Bot Framework від Microsoft; та популярні месенджери. Розглянемо їхні переваги та недоліки з точки зору функціональності, інтеграції, розширюваності та вартості.

Розглянемо пеший критерій – функціональність. Dialogflow має великий набір інструментів для розробки інтелектуальних чат-ботів, включаючи можливість розпізнавання мови, розуміння намірів користувача та генерацію відповідей. Завдяки цьому бот може здійснювати складніші операції, такі як бронювання квитків, покупки товарів тощо.

Bot Framework також має великий набір інструментів для створення чат-ботів, але він не такий розгалужений як у Dialogflow. Однак, Bot Framework має більше можливостей для інтеграції з іншими платформами та сервісами.

Другий критерій – інтеграція. Dialogflow може бути легко інтегрований з іншими сервісами Google, такими як Google Assistant та Google Home. Він також підтримує інтеграцію з іншими платформами через API.

Bot Framework може бути інтегрований з різними сервісами Microsoft, такими як Cortana, Skype, Microsoft

Наступний етап проведемо аналіз веб-орієнтованих систем на платформі чат-ботів таких месенджерів як Facebook, Telegram та Skype. Вибір цих месенджерів обумовлений їх популярністю та широкою підтримкою платформи чат-ботів, що дає можливість порівняти та проаналізувати їх можливості та обрати найбільш оптимальний варіант для реалізації поставлених завдань у дипломній роботі.

Facebook, Telegram та Skype є відомими месенджерами, які мають велику кількість користувачів по всьому світу та підтримують різноманітні функції, включаючи чат-ботів. Facebook Messenger має більше 1,3 мільярдів активних користувачів, Telegram має більше 500 мільйонів користувачів, Skype має більше 300 мільйонів користувачів. Крім того, всі ці месенджери підтримують чат-ботів та надають можливість розробникам створювати та впроваджувати свої власні чат-боти в цих месенджерах.

Проведення порівняльного аналізу дозволить визначити найбільш підходящий месенджер для використання у кваліфікаційній роботі, враховуючи поставлені завдання та потреби користувачів. В таблиці 2.3 наведем порівняльний аналіз використання чат-ботів в месенджерах.

Таблиця 2.3 – Порівняльний аналіз месендерів для чат-боту

Критерій	Чат-боти в мессенджерах		
	Telegram	Facebook	Skype
Спосіб застосування	Безкоштовна	Умовно безкоштовна	Умовно безкоштовна
Можливості доступу	Входження до груп, підписка на контент, вбудований у діалог	Входження до груп, підписка на контент, вбудований у діалог	Входження до груп на основі підписки

Користувацькі інтерфейси	Текстові, кнопкові та голосові		
Механізми зв’язку	webhook, polling	webhook	webhook
Протоколи передачі даних	HTTP, HTTPS		
Способи передачі даних	JSON, URL query, multipart/formdata	JSON, multipart/form -data	JSON, multipart/form -data
Можливості функціоналу	Листування, опитування, передача файлів, магазин покупок, ігри	Листування, опитування, передача файлів, магазин покупок, ігри	Листування, передача файлів, магазин покупок, ігри

Таким чином, згідно з проведеним аналізом табл.2.3, Telegram є найбільш підходящим месенджером для досягнення цілей дипломного проекту. Його переваги порівняно з іншими месенджерами включають:

- повністю безкоштовну форму використання;
- підтримку механізму зв'язку не тільки webhook , а й polling, що значно спрощує розробку та налагодження системи на початкових етапах;
- підтримку повного списку функціональних можливостей, що дозволяє реалізувати всі вимоги системи та дозволяє удосконалювати та розширювати їх за потреби.

2.4 Вибір технології для розробки веб-орієнтованої системи

Інформаційна система - це складна система, яка включає в себе технічні і програмні засоби для обробки інформації. Веб-орієнтована система є одним із прикладів інформаційної системи. За допомогою таких систем здійснюється передача, аналіз, обробка, зберігання та представлення даних, що дозволяє розв'язувати конкретні завдання. Для цього веб-орієнтована система повинна мати засоби передачі даних та повідомлень, засоби аналізу

інформації, засоби фіксації та збору даних, а також засоби збереження інформації. Зокрема, веб-орієнтована система може використовуватися для передачі даних, обробки даних, аналізу даних та прийняття рішень на їх основі. Застосування інформаційних систем може бути корисним для підприємств, установ, організацій, а також для окремих користувачів.

З точки зору проектування програмного забезпечення (ПЗ), необхідно виконати такі етапи як:

1. Сформулювати специфікацію ПЗ, що охоплює аналіз вимог та документацію опису поведінки системи з урахуванням технічних можливостей та обмежень.
2. Провести проектування ПЗ.
3. Здійснити програмування та тестування.
4. Провести розгортання ПЗ.

З точки зору проектування програмного забезпечення (ПЗ), завдання полягає у створенні системи керування чат-ботом з наступними функціональними вимогами:

1. Забезпечення можливості перегляду користувачів чат-ботом, включаючи загальну інформацію про користувача, дату останнього візиту та його стан.
2. Розробка чату з можливістю листування між ботом та користувачами у форматі текстових та медіа повідомлень, а також зберігання історії листувань.
3. Розробка файлового сервісу для зберігання будь-яких файлів, які були задіяні у роботі бота.
4. Розробка сервісу розсилок з можливістю надсилання текстових та медіа повідомлень групам користувачів: з можливістю планування розсилок на певну дату, перегляду поточного стану розсилок та архіву розсилок.
5. Забезпечення можливості перегляду статистичних даних, таких як кількість нових користувачів за певний період часу, кількість повідомлень за

період часу, кількість запланованих та активних розсилок та об'єм використаного сховища даних.

Наступним важливим кроком потрібно сформулювати конкретні вимоги для інтерфейсу користувача, програмного інтерфейсу, комунікаційних протоколів та бази даних на основі списку функціональних вимог.

Для системи керування чат-ботом, ми будемо розробляти інтерфейс користувача у вигляді крос-браузерного веб-інтерфейсу. Це спрощує розробку та підтримку, оскільки браузерні додатки підтримуються на будь-яких платформах та операційних системах (ОС) і залежать тільки від використовуваного браузеру. Розробка додатків для персональних комп’ютерів (ПК) та мобільних платформ вимагає більшої роботи з точки зору кросплатформеності кодової бази. У нашій роботі було обрано технологічний стек, який складається з PHP 8.0, Laravel Framework 9 та бази даних MySQL 8.0 для бекенду. Для фронтенду обрано bootstrap 5, laravel-mix 6, postcss 8.1, sass 1.3, sass-loader 12 та jquery 3.6.

PHP 8.0 [10]- це остання версія мови PHP, яка містить багато нових функцій та покращень продуктивності. Laravel Framework 9 - це високорівневий веб-фреймворк, який підтримує MVC-архітектуру, має зручний синтаксис та багато функціональних можливостей. Використання Laravel Framework 9 дозволяє швидко створювати веб-додатки та API.

MySQL 8.0 [11] - це безкоштовна реляційна база даних з відкритим кодом. Вона має високу продуктивність, підтримує ACID транзакції та має багато функціональних можливостей, таких як JSON-дані та аналітика.

Bootstrap 5 [13] - це один з найпопулярніших фреймворків для створення веб-сайтів та додатків. Він надає широкий набір готових елементів, які можна використовувати для швидкого розроблення інтерфейсу користувача.

Laravel Mix 6 [14]- це пакет з відкритим кодом для збирання та оптимізації ресурсів фронтенду. Він підтримує збирання JavaScript, SASS, LESS та багато інших форматів файлів.

PostCSS 8.1 та Sass 1.3 [15] - це препроцесори CSS, які дозволяють розширити функціональність CSS. Sass-loader 12 - це webpack loader для збірки Sass файлів.

jQuery 3.6 [16] – це бібл іотека JavaScript, яка забезпечує багато корисних функцій для роботи зі сторінками веб-сайту. Вона є однією з найпопулярніших бібліотек JavaScript і широко використовується для розробки фронтенд-частин веб-додатків. jQuery дозволяє просто та ефективно взаємодіяти з HTML-документом, здійснювати анімації, збільшувати ефективність роботи зі сторінкою, працювати з AJAX-запитами та багато іншого.

Крім того, в якості сервера веб-додатка було обрано Apache, який є одним з найбільш популярних сервісів веб-додатків в світі. Він підтримує багато різних протоколів, включаючи HTTP і HTTPS, і може бути налаштований для роботи з різними базами даних, такими як MySQL.

Для розробки фронтенду використовується Bootstrap 5, який є одним з найпопулярніших фреймворків для розробки веб-інтерфейсів. Bootstrap надає набір готових стилів і компонентів, які дозволяють швидко розробити естетичний інтерфейс користувача, що працює на різних пристроях і платформах. Bootstrap дозволяє створювати респонсивні сторінки, які працюють на будь-яких пристроях, включаючи мобільні пристрої. Також фреймворк має багато корисних додаткових функцій, таких як підтримка Sass, JavaScript-плагіни та інші.

Для збирання і пакування фронтенду використовується Laravel Mix 6, який є пакетом для Laravel, що дозволяє збирати, оптимізовувати і пакувати ресурси для веб-додатків, такі як CSS і JavaScript. Laravel Mix використовує Webpack, що дозволяє автоматично компілювати, обрізати і об'єднувати ресурси для оптимальної продуктивності. Laravel Mix дозволяє швидко налаштовувати роботу зі статичними файлами та зменшує зусилля, необхідні для ручної компіляції та мініфікації цих файлів.

Для розробки стилів використовується Sass 1.3, який є препроцесором CSS. Sass дозволяє використовувати змінні, міксіни, наслідування і інші корисні функції, що спрощують розробку і підтримку CSS-коду.

Для збирання Sass в CSS використовується sass-loader 12, який є плагіном для Webpack. Sass-loader дозволяє автоматично компілювати Sass в CSS при збиранні фронтенду.

2.5 Висновки до другого розділу

У даому розділі було розглянуто проведено огляд та вибір технологій для розробки веб-орієнтованої статистичної платформи з геоінформаційною підтримкою такі як Node.js, Ruby on Rails та PHP. На основі аналізу вимог та обмежень було зроблено вибір серверної платформи, що надає високу продуктивність, масштабованість та широкий вибір зовнішніх бібліотек і модулів. Були розглянуті різні хмарні сховища для зберігання даних платформи, такі як Amazon S3, Google Cloud Storage та Microsoft Azure Storage. З урахуванням характеристик, надійності та вартості було зроблено вибір хмарного сховища Amazon S3, яке забезпечує масштабованість, безпеку та доступність даних. Було проведено аналіз різних веб-орієнтованих систем на платформі чат-ботів, таких як Facebook Messenger, Telegram, Slack тощо. Критерії включали функціональні можливості, популярність та зручність використання. На основі аналізу було вибрано платформу Telegram, яка має широкі можливості для розробки чат-ботів та має значну користувацьку базу.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СТАТИСТИЧНОЇ ПЛАТФОРМИ З ГЕОІНФОРМАЦІЙНОЮ ПІДТРИМКОЮ

3.1 Структура веб-орієнтованої системи

Структура веб-орієнтованої статистичної платформи з геоінформаційною підтримкою складається з наступних функціональних елементів (рис. 2.1), які допомагають забезпечити зручний та швидкий доступ до інформації про перетин кордону України на основі відгуків користувачів.

Рисунок 3.1 – Структура веб-орієнтованої статистичної платформи

Основним елементом платформи є сайт, з картою України з іконками, що показують перетини кордону для пішоходів, автомобілів та потягів. На головній сторінці сайту розміщена кнопка "Статистика", натискання на яку відкриває відповідну сторінку, на якій можна переглянути відгуки користувачів про перетин кордону (рис.3.2).

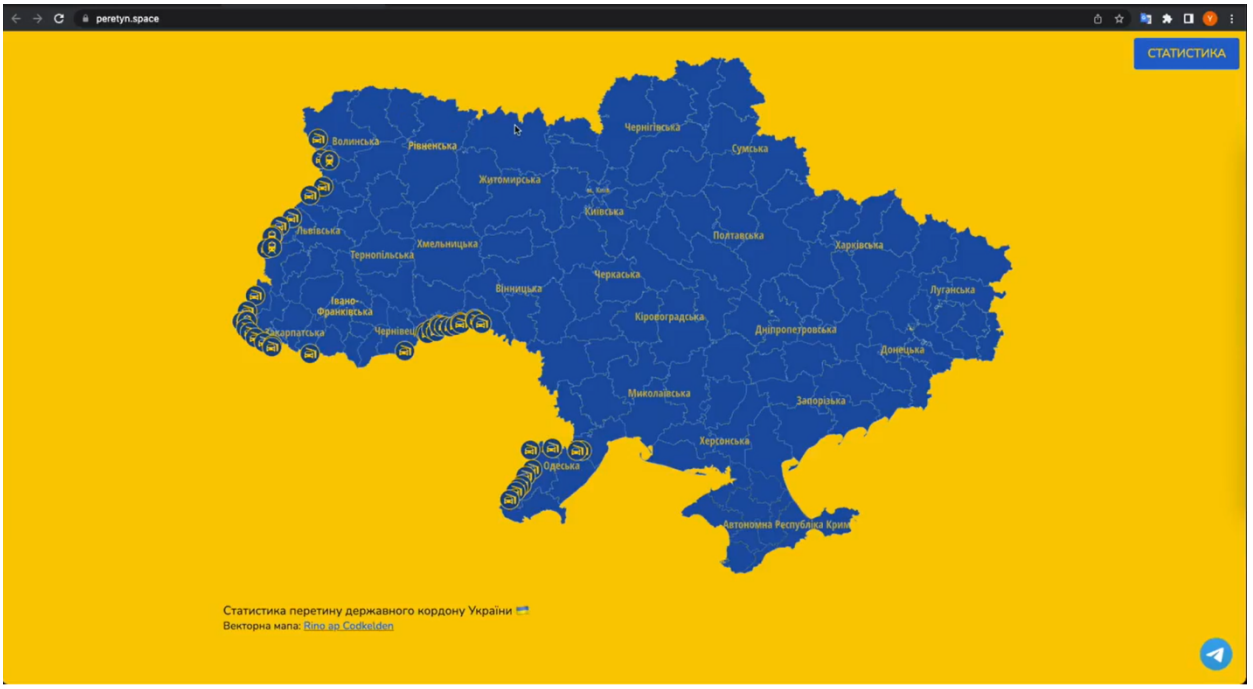


Рисунок 3.2 – Головне меню веб-орієнтованої системи

Структура сторінки статистики включає в себе можливості сортування відгуків за рейтингом (найбільше позитивних або негативних відгуків), алфавітом або пошуком за назвою пункту митного контролю. Відгук містить інформацію про користувача, який залишив відгук (ім'я, але з урахуванням конфіденційної інформації воно частково записано), дату та час створення відгуку, оцінку та сам відгук. На рисунку 3.3 наведено структурну схему взаємодії користувачів (це люди, які здійснили перетин митного кордону) та адміністратори.

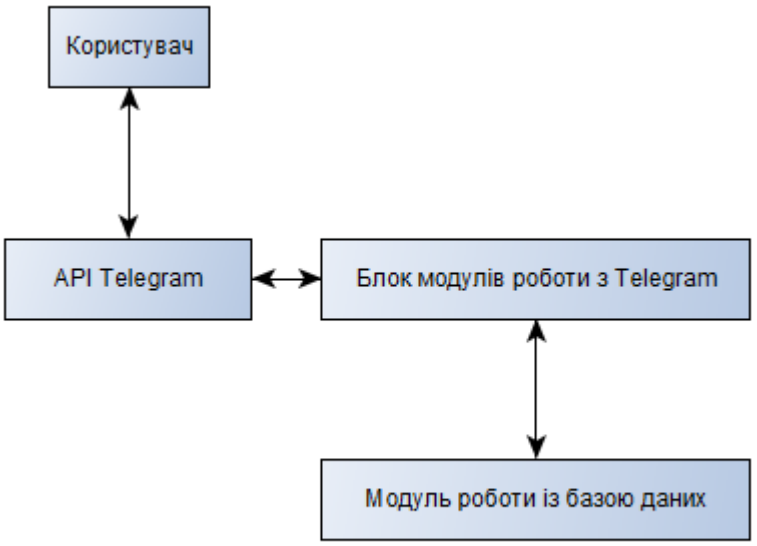


Рисунок 3.3 – Структурна схема взаємодії користувачів з чат-ботом

Статистичні дані формуються на основі чат-бота в Телеграмі (“Diploma Status”), за допомогою якого користувачі можуть залишити свої відгуки про перетин кордону. Це дозволяє отримати найбільш точну та актуальну інформацію, яка потім використовується для аналізу та статистики.

Загалом, структура веб-орієнтованої статистичної платформи з геоінформаційною підтримкою забезпечує зручний та швидкий доступ до інформації про перетин кордону, а також дозволяє користувачам залишати відгуки та ділитися досвідом з іншими людьми. Більше того, система статистичної звітності допомагає збирати, обробляти та аналізувати дані, що забезпечує можливість моніторингу ситуації на кордоні та розробки подальших стратегій розвитку.

Крім того, додаткові можливості, такі як сортування відгуків та пошук за пунктом пропуску, дозволяють користувачам швидко та легко знайти потрібну інформацію. Також, наявність карти з іконками перетину кордонів робить систему зручною та зрозумілою для користувачів.

Усі дані в системі зберігаються в базі даних (MySQL 8.0) [17], що дозволяє забезпечувати надійне зберігання інформації та швидкий доступ до

неї. Крім того, система захищена від несанкціонованого доступу за допомогою механізмів аутентифікації та авторизації.

Взаємодію між користувачами та веб-орієнтованою системою відображає діаграма використання (Use Case Diagram). Вона допомагає зрозуміти, як система повинна працювати, які дії можуть виконувати користувачі та які можливі наслідки цих дій (рис.3.4).

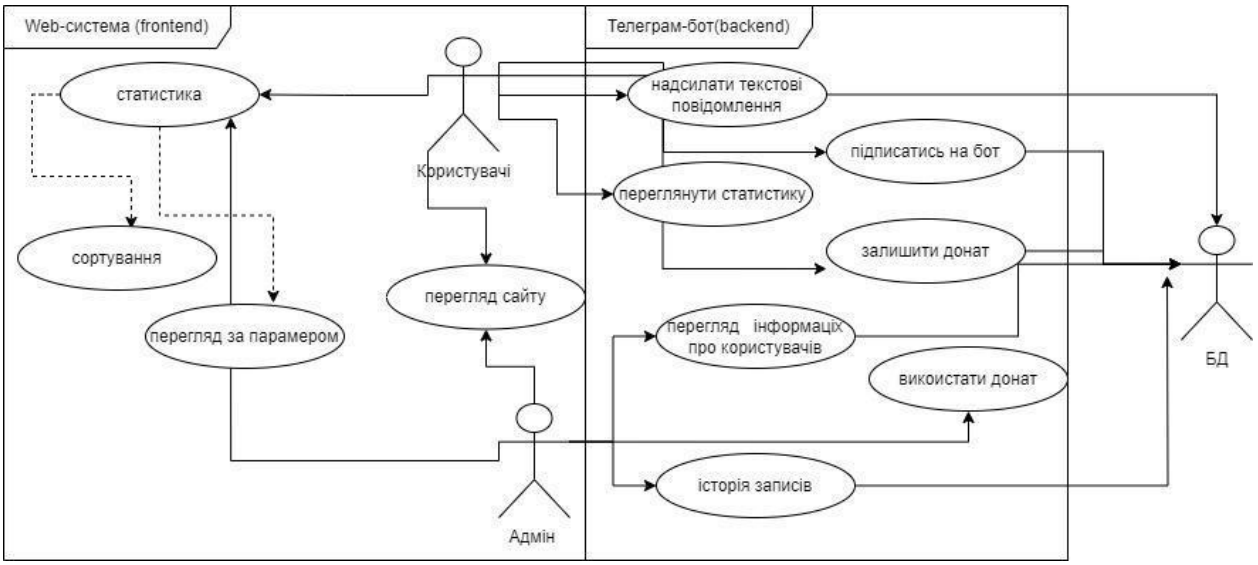


Рисунок 3.4 – Діаграма використання веб-орієнтованої системи

На діаграмі зображені актори (користувачі) та їх можливі взаємодії з системою та Телеграм-ботом. Основні актори чат-боту - це звичайні користувачі та адміністратор. Користувачі можуть запитувати у бота інформацію про статистику перетину кордону, переглядати ТОП-5 статистику, редагувати свої дані, підписуватися на бот та залишати донат.

Звичайний користувач може використовувати такі функції, як:

- почати спілкування з ботом та надіслати текстове повідомлення;
- переглянути статистику по митним пунктам пропуску;
- редагувати дані;
- підписатись на бот;
- залишити донат.

Адміністратор може використовувати такі функції:

- надавати доступ до сайту;
- перегляд інформації про користувачів;
- використати донат.

Ці функції будуть доступні відповідним користувачам за допомогою команд, які можуть бути введені в бота. Наприклад, щоб переглянути поточну статистику по митним пунктам пропуску, користувач може ввести команду "/statistics". Таким чином, користувачі та адміністратор мають доступ до функцій, які необхідні для використання нашої статистичної платформи з геоінформаційною підтримкою.

Діаграма використання допомагає розуміти, які можливості має користувач та адміністратор, як вони взаємодіють з чат-ботом та які дії можуть виконувати. Вона допомагає краще зрозуміти функціонал нашої системи та визначити можливість додавання нових функцій.

Для моделювання функціональних процесів використана методологія IDEF0, яка базується на графічній мові опису систем. Спочатку проводиться опис системи в цілому та її взаємодії з навколишнім світом за допомогою контекстної діаграми, а потім система функціонально декомпозується, тобто розбивається на підсистеми, кожна з яких описується окремо на діаграмах декомпозиції. Процеси зображуються за допомогою блоків на цих діаграмах, а зв'язки між процесами - стрілками. Результат моделювання функціональних процесів наведено на рис. 3.5.

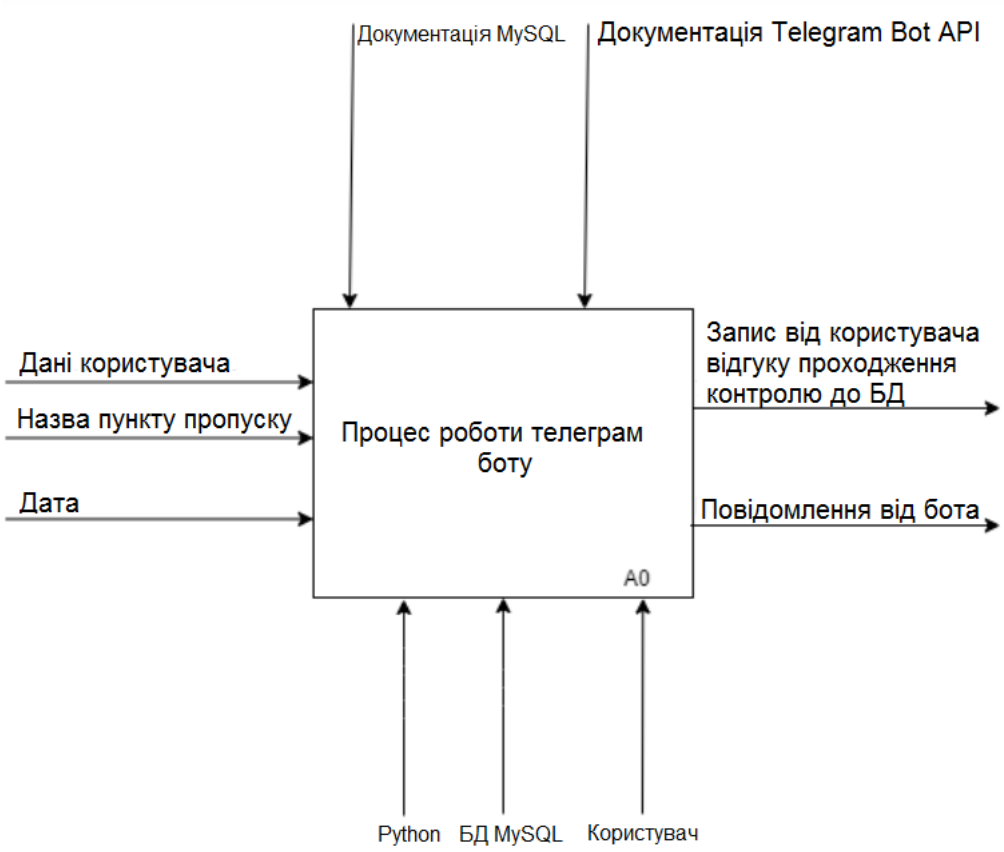


Рисунок 3.5 – IDEF0 діаграма

IDEF0 діаграма першого рівня для чат-боту може бути використана для опису його контексту та загального огляду його функцій та представлена на рис.3.6.

Ця діаграма містить блок, що представляє бота, який знаходиться в центрі діаграми. Окремі блоки навколо нього представляють взаємодію бота з іншими сутностями, наприклад, з користувачем та з Telegram API.

Стрілки, які з'єднують блоки, показують взаємодію між ними. Наприклад, з'єднувальна стрілка може з'єднувати блок нашого бота з блоком, який представляє процес обробки введеної користувачем команди. Інші стрілки можуть зображувати потік даних між різними блоками, які включають в себе передачу даних між нашим ботом та Telegram API, а також між нашим ботом та базою даних.

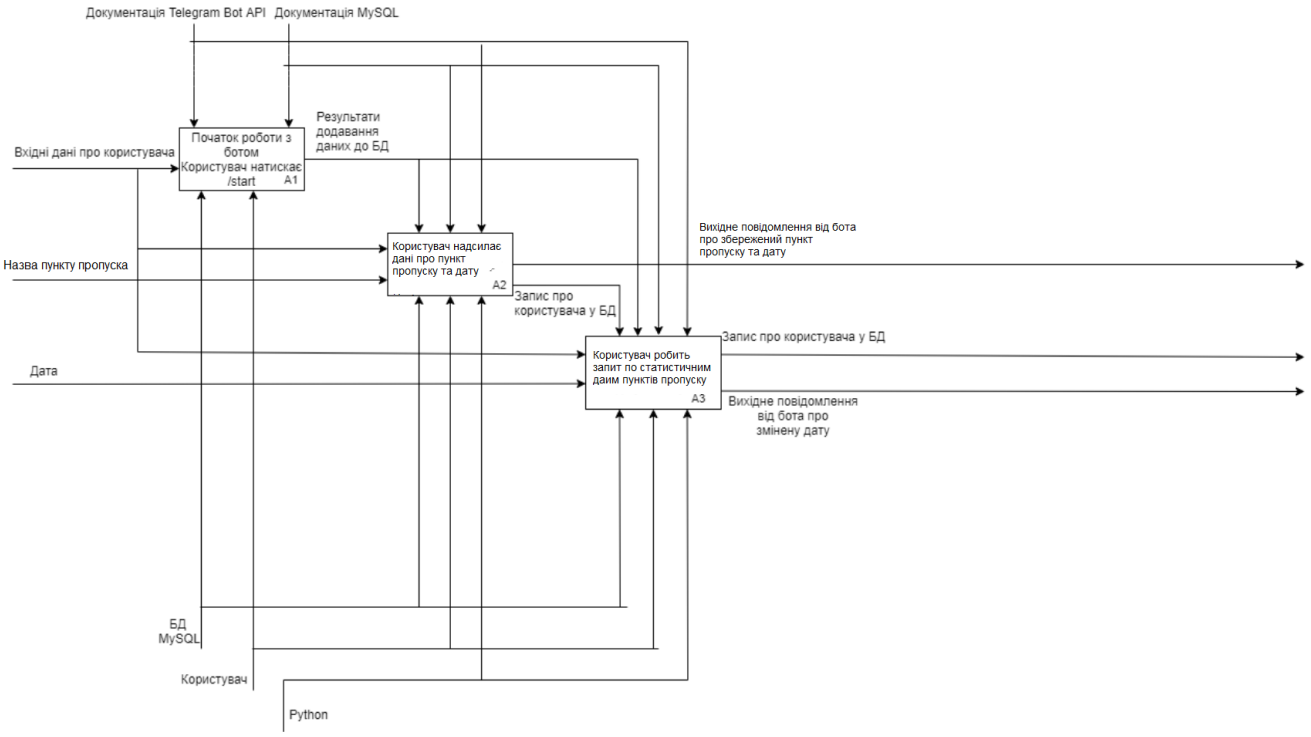


Рисунок 3.6 – IDEF0 діаграма першого рівня

IDEF0 діаграма першого рівня є корисним інструментом для визначення загального контексту та функцій нашого бота, що допомагає нам зрозуміти взаємодію між його складовими частинами та елементами.

3.2 Алгоритм роботи системи

Система працює за простим алгоритмом. Користувач спочатку відправляє команду або запит чат-ботові, який потім передає його до програмного забезпечення на сервері розробника. Сервер обробляє запит та надає відповідь чат-боту, який відображає відповідь на екрані користувача.

Робота користувача починається з команди "/start" або автоматично за допомогою Telegram API, який надає унікальний ідентифікаційний номер користувача на платформі та його ім'я, що користувач вводив у месенджері. Для взаємодії з користувачем, чат-бот використовує кнопки з різними варіантами дій, які розглядає як окремі команди (рис.3.7):

"/status" – показати статистику перетину по всім КПП;

"/status5" - показати статистику перетину ТОП5 КПП. Тут є можливість обрати термін за 7 днів, 31 дні або 365 днів;

"/subscribe" – підписатись на бот.

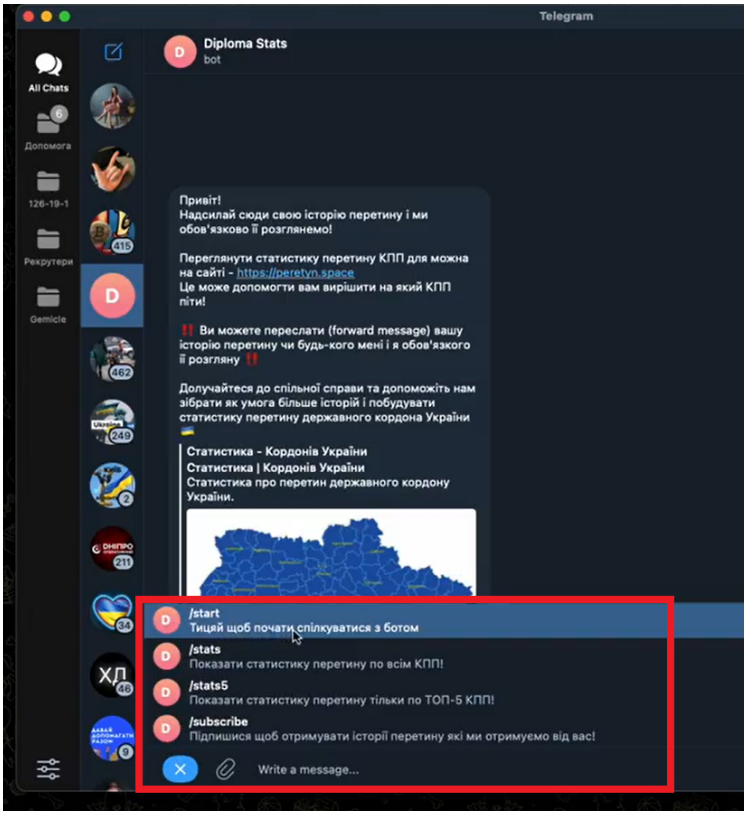


Рисунок 3.7 – Команди дляя роботи чат-бота

Графічно алгоритм роботи веб-орієнтованої системи з використанням Телеграм-боту наведено на рис.3.8.

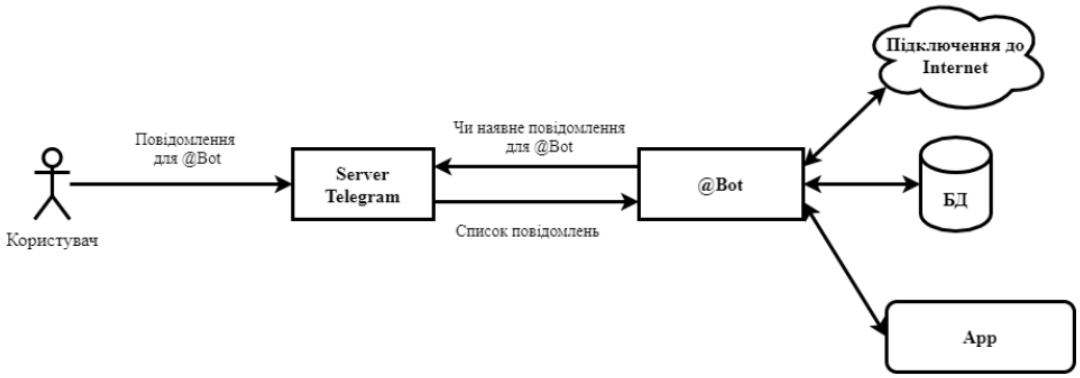


Рисунок 3.8 – Схема роботи Телеграм-бота

У меню налаштувань користувач може змінити свої персональні дані в будь-який момент. Користувачеві надається можливість пройти тестування з

обраних дисциплін, після перегляду їх програм. Результат тестування можна переглянути в повідомленні про статистику користувача.

3.3 Розробка фронтенду

Для розробки сервера чат-бота було використано Telegram Bot API [18], яке дозволяє швидко та просто створювати інтегровані чат-боти в месенджері Telegram, використовуючи HTTP запити до сервера Telegram. Для створення чат-бота використовується Westacks, що є батьківським класом для інших ботів в системі Telegram. [19]

Запити можна використовувати для отримання інформації про оновлення чат-боту, такі як нові повідомлення, зміни повідомлень, відправлення мультимедійних даних та інше. Під час розробки чат-боту було використано Telegram Bot API та асинхронний фреймворк aiogram, який базується на Python стандартній бібліотеці asyncio та асинхронному веб-фреймворку aiohttp. Aiogram було обрано через його простоту використання та те, що він є повністю асинхронним.

Згідно з обраною архітектурою, необхідно розбити систему на мікросервіси. Існують різні підходи до процесу декомпозиції, такі як розбиття на основі бізнес-можливостей або на основі субдоменних моделей. Незалежно від обраного підходу, важливо дотримуватись принципу єдиної відповідальності та принципу відкритості/закритості. Для розбиття системи керування чат-ботом на мікросервіси, доцільним буде визначити окремі субдоменні моделі. Субдоменні моделі включатимуть наступне (рис.3.9):

1. Принцип відкритості/закритості - компоненти системи мають бути відкритими для розширення, але закритими для змін.
2. Принцип єдиної відповідальності - кожен мікросервіс повинен мати лише одну відповідальність.
3. Для декомпозиції системи керування чат-ботом доцільно визначити окремі субдоменні моделі, які включатимуть:

4. Модуль логіки чат-бота – строг визначена логікою дій бота, а також дані, пов'язані з чат-ботом.
5. Модуль взаємодії з Telegram Bot API.
6. Модуль авторизації користувачів.
7. Модуль керування файлами.
8. Модуль роботи з повідомленнями в чаті.
9. Модуль здійснення розсилок.

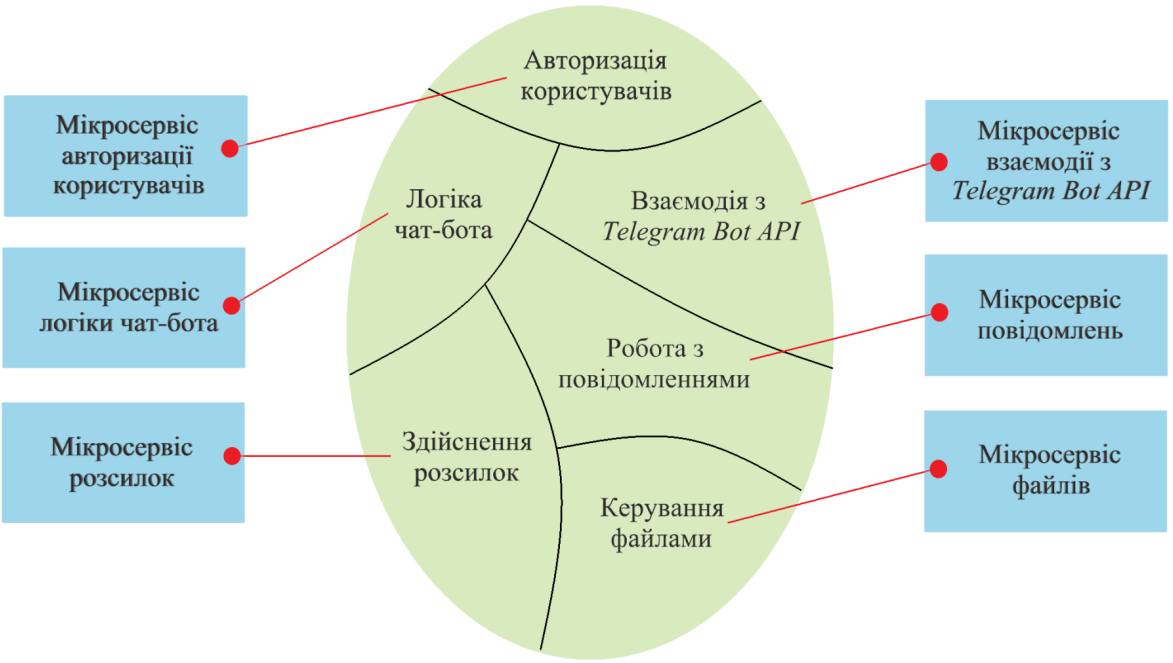


Рисунок 3.9 – Сервіси для керування ботом

Слід визначити API кожного мікросервісу, що включає операції (команди) та події. Команди сервісу можуть мати дві мети: виконувати операції, що ініціюються зовнішніми клієнтами або іншими сервісами, або ж використовуватись для забезпечення взаємодії між сервісами. Події, зазвичай, використовуються для забезпечення взаємодії між сервісами, синхронізації та узгодженості даних, а також для повідомлення зовнішніх клієнтів, наприклад WebSockets можуть надсилати події до браузера клієнта.

Починаємо створення API сервісів з того, що визначаємо операції, які будуть доступні для кожного сервісу. Крім того, встановлюємо зв'язок між сервісами та їх операціями (див. рис.3.10).

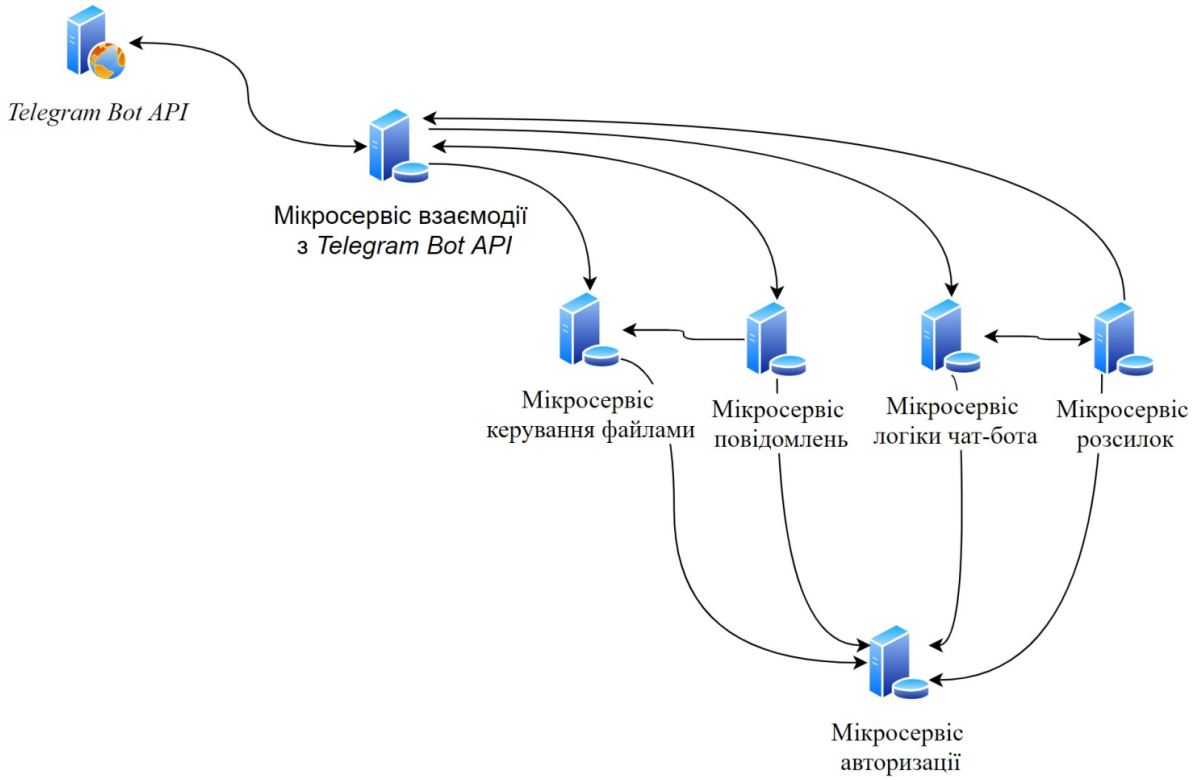


Рисунок 3.10 –Графічне представлення сервісів

Операції API мікросервісу взаємодії з Telegram Bot API передбачають створення різних запитів до API Telegram, які дозволяють отримувати різні дані про користувачів, чати, повідомлення та інші параметри. Основні операції API мікросервісу взаємодії з Telegram Bot API можуть включати наступні (табл.3.1):

– отримання доступу до Telegram Bot API, яке може здійснюватися шляхом авторизації токеном бота;

– створення запитів до API Telegram [18] для отримання інформації про користувачів, чати, повідомлення та інші параметри;

– обробка вхідних запитів та відповідей, що отримуються від Telegram Bot API, для забезпечення зворотного зв'язку з іншими сервісами;

− розробка функціоналу бота, який здійснює взаємодію з користувачами, на основі отриманих даних з Telegram Bot API;

− забезпечення безпеки даних та приватності користувачів, шляхом використання методів шифрування та збереження даних на захищеному сервері;

− розробка механізмів моніторингу та логування, які дозволяють відслідковувати роботу бота та виявляти можливі проблеми з безпекою даних;

− підтримка та оновлення API мікросервісу взаємодії з Telegram Bot API з метою забезпечення його безперебійної роботи та відповідності змінам в Telegram Bot API.

Таблиця 3.1 – Взаємодія з Telegram Bot API [18]

№	Операція	Опис	Взаємодія
	connectToTelegramAPI(params) createFileServiceConnection(params) createBotGateServiceListener(params) createBotLogicServiceConnection	встановлює з'єднання з Telegram Bot API за допомогою переданих параметрів (наприклад, токен бота)	Ініціалізація сервісу
	sendMessage(chatId, text)	надсилає повідомлення з вказаним текстом в чат з вказаним ідентифікатором	Сервіс повідомлення
	getUpdates(offset, limit)	отримує список оновлень (нових	Сервіс логіки

		повідомлень, запитів на зміну стану тощо) з Telegram Bot API з вказаним зміщенням та лімітом.	
	setWebhook(url)	встановлює вебхук для отримання повідомлень від Telegram Bot API за вказаною адресою URL	Сервіс логіки
	deleteWebhook()	видаляє вебхук, щоб зупинити отримання повідомлень за допомогою вебхука	Сервіс видалння

Технологічний стек фронтенду використовує bootstrap 5, laravel-mix 6, postcss 8.1, sass 1.3, sass-loader 12 та jquery 3.6, буде наступним:

3.3.1 Налаштування Bootstrap 5

Bootstrap 5 - це фреймворк для розробки веб-інтерфейсів, який забезпечує широкий набір компонентів та інструментів для швидкого та ефективного розробки. Bootstrap дозволяє розробникам створювати красиві та адаптивні веб-сторінки, які працюють на різних пристроях.

Лістинг 3.1

```
import _from 'lodash';
```

```
window._ = _;

import $ from 'jquery'
window.jQuery = window.$ = $

import 'bootstrap';

/**
 * We'll load the axios HTTP library which allows us to easily issue requests
 * to our Laravel back-end. This library automatically handles sending the
 * CSRF token as a header based on the value of the "XSRF" token cookie.
 */

import axios from 'axios';
window.axios = axios;

window.axios.defaults.headers.common['X-Requested-With'] =
'XMLHttpRequest';

/**
 * Echo exposes an expressive API for subscribing to channels and listening
 * for events that are broadcast by Laravel. Echo and event broadcasting
 * allows your team to easily build robust real-time web applications.
 */

// import Echo from 'laravel-echo';

// import Pusher from 'pusher-js';
// window.Pusher = Pusher;
```

```
// window.Echo = new Echo({
//     broadcaster: 'pusher',
//     key: import.meta.env.VITE_PUSHER_APP_KEY,
//                     wsHost:     import.meta.env.VITE_PUSHER_HOST     ??
`ws-${import.meta.env.VITE_PUSHER_APP_CLUSTER}.pusher.com`,
//     wsPort: import.meta.env.VITE_PUSHER_PORT ?? 80,
//     wssPort: import.meta.env.VITE_PUSHER_PORT ?? 443,
//      forceTLS: (import.meta.env.VITE_PUSHER_SCHEME ?? 'https') ===
'https',
//     enabledTransports: ['ws', 'wss'],
// });
```

У лістингу 3.1 наведено фрагмент імпорту бібліотек та інструментів, які зазвичай використовуються у веб-розробці. Деталі коду:

− import _ from 'lodash'; window._ = _; імпортує бібліотеку Lodash і додає її до глобального об'єкта вікна як _, роблячи її доступною в усьому додатку. Lodash - це утиліта, яка надає багато корисних функцій для роботи з масивами, об'єктами та іншими структурами даних у JavaScript;

− import $ from 'jquery' window.jQuery = window.$ = $ імпортує бібліотеку jQuery і додає її до глобального об'єкта вікна як $ і jQuery, роблячи її доступною в усьому додатку. jQuery - популярна бібліотека JavaScript, яка полегшує маніпулювання HTML DOM і обробку подій;

− import 'bootstrap'; імпортує бібліотеку Bootstrap, яка надає набір стилів CSS і компонентів JavaScript для створення адаптивних, мобільних веб-сторінок;

− import axios from 'axios'; window.axios = axios; імпортує бібліотеку Axios, яка надає зручний спосіб робити HTTP-запити від клієнта до сервера. Цей код додає Axios до глобального об'єкта вікна як axios, роблячи його доступним у всьому додатку;

− window.axios.defaults.headers.common['X-Requested-With'] = 'XMLHttpRequest'; встановлює стандартний HTTP-заголовок для всіх запитів

Axios. Цей заголовок використовується, щоб вказати, що запит було зроблено за допомогою XMLHttpRequest, що необхідно для належної роботи CSRF-захисту Laravel.

3.3.2 Налаштування Laravel-mix 6

Laravel-mix 6 - це інструмент для збирання та оптимізації ресурсів (зображень, стилів, скриптів тощо) для веб-сайтів. Laravel-mix 6 дозволяє зручно налаштувати збірку ресурсів та генерувати оптимізований код.

Лістінг 3.2

```
<?php

/*
|--------------------------------------------------------------------------
| Create The Application
|--------------------------------------------------------------------------
|
| The first thing we will do is create a new Laravel application instance
| which serves as the "glue" for all the components of Laravel, and is
| the IoC container for the system binding all of the various parts.
|
*/

$app = new Illuminate\Foundation\Application(
    $_ENV['APP_BASE_PATH'] ?? dirname(__DIR__)
);

/*
|--------------------------------------------------------------------------
| Bind Important Interfaces
|--------------------------------------------------------------------------
```

```
|
| Next, we need to bind some important interfaces into the container so
| we will be able to resolve them when needed. The kernels serve the
| incoming requests to this application from both the web and CLI.
|
*/

$app->singleton(
    Illuminate\Contracts\Http\Kernel::class,
    App\Http\Kernel::class
);

$app->singleton(
    Illuminate\Contracts\Console\Kernel::class,
    App\Console\Kernel::class
);

$app->singleton(
    Illuminate\Contracts\Debug\ExceptionHandler::class,
    App\Exceptions\Handler::class
);
/*
|--------------------------------------------------------------------------
| Return The Application
|--------------------------------------------------------------------------
|
| This script returns the application instance. The instance is given to
| the calling script so we can separate the building of the instances
| from the actual running of the application and sending responses.
|
```

```
*/
return $app;
```

Код лістінга 3.2 створює новий інстанс додатку Laravel, який є основою для всіх компонентів Laravel. Він також реєструє необхідні інтерфейси в контейнері залежностей (IoC), які дозволяють вирішувати залежності при запиті. Крім того, цей код повертає створений інстанс додатку для використання в інших скриптах. В цьому коді ми також можемо побачити використання змінної оточення $_ENV['APP_BASE_PATH'], щоб визначити шлях до кореневої директорії додатку.

Лістинг 3.3

```
<?php return array (
 'laravel/sail' =>
 array (
  'providers' =>
  array (
   0 => 'Laravel\\Sail\\SailServiceProvider',
  ),
 ),
 'laravel/sanctum' =>
 array (
  'providers' =>
  array (
   0 => 'Laravel\\Sanctum\\SanctumServiceProvider',
  ),
 ),
 'laravel/tinker' =>
 array (
  'providers' =>
  array (
   0 => 'Laravel\\Tinker\\TinkerServiceProvider',
```

```
    ),
  ),
  ....
  'spatie/laravel-signal-aware-command' =>
   array (
    'providers' =>
    array (
                                                    0 =>
'Spatie\\SignalAwareCommand\\SignalAwareCommandServiceProvider',
    ),
    'aliases' =>
    array (
      'Signal' => 'Spatie\\SignalAwareCommand\\Facades\\Signal',
    ),
  ),
   'westacks/telebot' =>
   array (
    'providers' =>
    array (
      0 => 'WeStacks\\TeleBot\\Laravel\\Providers\\TeleBotServiceProvider',
    ),
    'aliases' =>
    array (
      'TeleBot' => 'WeStacks\\TeleBot\\Laravel\\TeleBot',
```

Код лістингу 3.3 містить конфігурацію постачальників послуг для Laravel. Кожен елемент масиву представляє пакет, який можна встановити через Composer і його послуги, які повинні бути зареєстровані у Laravel. Кожен пакет має список постачальників послуг та, можливо, список псевдонімів. Постачальники послуг - це класи, які реєструють функціональність пакета у Laravel, наприклад, реєстрація маршрутів,

контролерів, сервіс-провайдерів, вказання шляху до файлів міграцій тощо. Псевдоніми дозволяють коротко іменувати фасади класів. В даному коді зазначені пакети: Laravel Sail, Laravel Sanctum, Laravel Tinker, Laravel UI, Carbon, Collision, Termwind, Laravel Backup, Laravel Ignition, Laravel Signal Aware Command, WeStacks TeleBot.

3.3.3 Інструменти PostCSS 8.1 та Sass 1.3

PostCSS 8.1 - це інструмент для збільшення функціональності CSS. Він дозволяє розробникам використовувати різні плагіни для покращення роботи з CSS, такі як autoprefixer, cssnano тощо.

Лістинг 3.4

```
const postcss = require('postcss');
const autoprefixer = require('autoprefixer');

const css = `
  body {
    display: flex;
    justify-content: center;
    align-items: center;
  }
  a {
    transition: color 0.2s;
  }
`;

postcss([autoprefixer])
  .process(css, { from: undefined })
  .then(result => {
    console.log(result.css);
  });
```

Лістинг 3.4 використовує PostCSS та плагін Autoprefixer для того, щоб додати префікси вендорів до CSS властивостей. В результаті виконання коду в консолі буде виведений CSS з доданими префіксами вендорів.

Sass 1.3 - це мова, що розширює CSS. Sass дозволяє розробникам використовувати змінні, міксіни, функції та інші функції, які полегшують розробку CSS.

Sass-loader 12 - це модуль Webpack, який дозволяє Webpack читати та компілювати Sass-файли в CSS.

Лістинг 3.5

{flex-direction:column!important}.flex-xl-row-reverse{flex-direction:row-reverse!important}.flex-xl-column-reverse{flex-direction:column-reverse!important}.flex-xl-grow-0{flex-grow:0!important}.flex-xl-grow-1{flex-grow:1!important}.flex-xl-shrink-0{flex-shrink:0!important}.flex-xl-shrink-1{flex-shrink:1!important}.flex-xl-wrap{flex-wrap:wrap!important}.flex-xl-nowrap{flex-wrap:nowrap!important}.flex-xl-wrap-reverse{flex-wrap:wrap-reverse!important}.justify-content-xl-start{justify-content:flex-start!important}.justify-content-xl-end{justify-content:flex-end!important}.justify-content-xl-center{justify-content:center!important}.justify-content-xl-between{justify-content:space-between!important}

3.3.3 Бібліотека jQuery 3.6

jQuery 3.6 - це JavaScript-бібліотека, яка дозволяє розробникам використовувати різні функції та методи для зручної та ефективної розробки скриптів на стороні клієнта. jQuery дозволяє зробити роботу з JavaScript більш простою та ефективною.

Лістинг 3.6

```
<php?
  // Підключення до бази даних
  $db_host = 'localhost';
  $db_user = 'username';
  $db_pass = 'password';
  $db_name = 'database';
```

```
$db = mysqli_connect($db_host, $db_user, $db_pass, $db_name);

// Перевірка з'єднання з базою даних
if (!$db) {
  die('Could not connect to database');
}

// Вибірка даних з таблиці
$query = "SELECT * FROM users";
$result = mysqli_query($db, $query);

// Перетворення результату запиту в JSON-формат
$data = array();
while ($row = mysqli_fetch_assoc($result)) {
  $data[] = $row;
}
$json_data = json_encode($data);
?>

<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>jQuery and PHP Example</title>
 <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
 <script>
  $(document).ready(function() {
    // Отримання даних з сервера в форматі JSON
    var data = <?php echo $json_data; ?>;
```

```
    // Пошук елементу списку
    var searchInput = $('#search-input');
    var resultsList = $('#results-list');

    // Обробник події введення тексту
    searchInput.on('input', function() {
      var searchText = searchInput.val().toLowerCase();
      resultsList.empty();
      $.each(data, function(index, value) {
        if (value.name.toLowerCase().indexOf(searchText) !== -1) {
          resultsList.append('<li>' + value.name + '</li>');
        }
      });
    });
  });
  </script>
</head>
<body>
  <input type="text" id="search-input" placeholder="Search...">
  <ul id="results-list"></ul>
</body>
</html>
```

У лістингу 3.6 ми використовуємо jQuery для створення простого пошукового поля, яке використовує дані з бази даних для пошуку за іменем користувача. Код на PHP виконує запит до бази даних і перетворює результат у формат JSON, який потім використовується у нашому скрипті jQuery.

3.4 Розробка бекенду

Технологічний стек Бекенд розроблений з використанням PHP 8.0, Laravel Framework 9 та Database MySQL 8.0. Головне ядро програми

міститься в файлі composer.json. Фрагмент коду представлено на рис.3.11, лістинг в Додатку А.

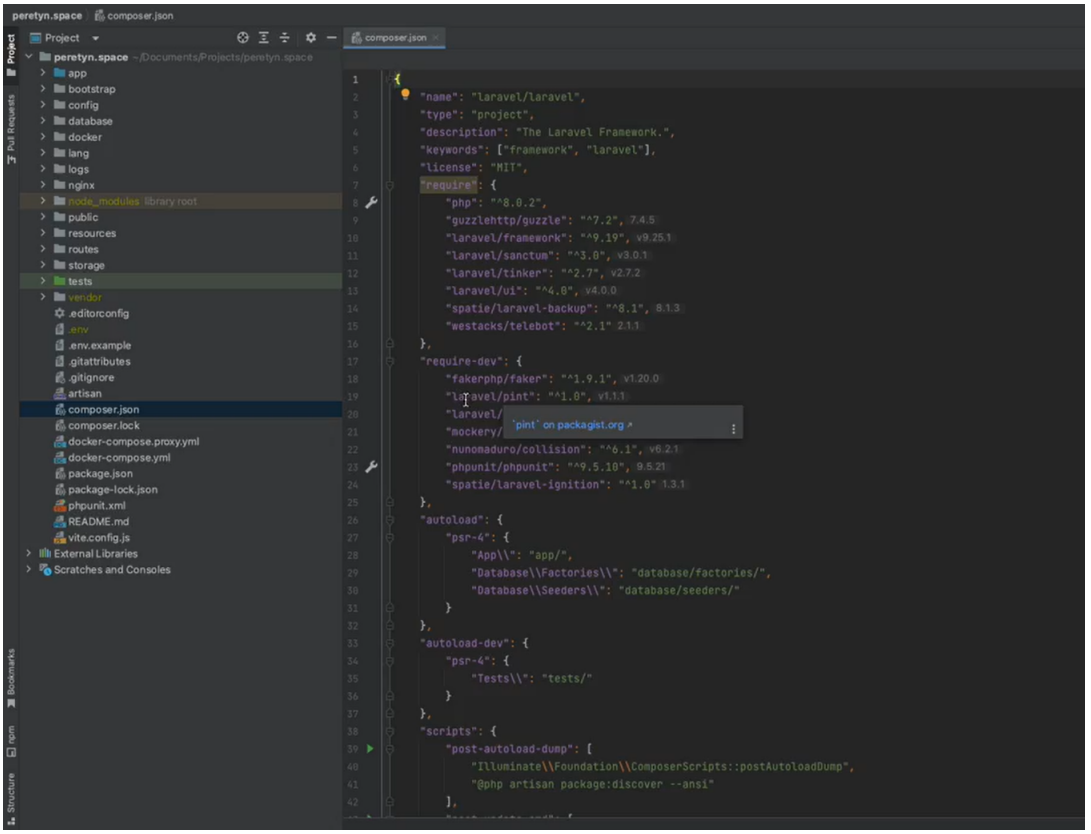


Рисунок 3.11 – Фрагмент лістингу ядра програми

Фрагмент лістинга на рис.3.11 є файлом конфігурації композера для Laravel бекенду системи.

- "name": "laravel/laravel" - назва проекту.
- "type": "project" - тип проекту.
- "description": "The Laravel Framework." - опис проекту.
- "keywords": ["framework", "laravel"] - ключові слова для проекту.
- "license": "MIT" - ліцензія проекту.

Далі йдуть залежності для проекту:

- "php": "^8.0.2" - версія PHP, необхідна для проекту.
- "guzzlehttp/guzzle": "^7.2" - Guzzle - HTTP-клієнт для PHP.
- "laravel/framework": "^9.19" - Laravel - основний фреймворк для створення бекенду.
- "laravel/sanctum": "^3.0" - пакет для аутентифікації API.

- "laravel/tinker": "^2.7" - інтерактивна консоль для Laravel.
- "laravel/ui": "^4.0" - пакет для швидкого створення інтерфейсу користувача.
- "spatie/laravel-backup": "^8.1" - пакет для резервного копіювання бази даних Laravel.
- "**westacks/telebot**": "^2.1" - пакет для роботи з Telegram Bot API. Він дозволяє виконувати основні операції, такі як надсилання повідомлень, відповідей на повідомлення, налаштування клавіатури та багато іншого. Цей пакет дозволяє розробникам швидко і легко створювати чат-ботів на основі Telegram Bot API. Він містить кілька різноманітних функцій та класів, що дозволяють здійснювати зручну роботу з ботом, включаючи обробку різноманітних типів повідомлень та взаємодію з Telegram API через HTTP-запити. Для використання цього пакету потрібно встановити його за допомогою Composer, як показано в composer.json відповідної системи, та налаштувати його на основі документації. Після цього можна використовувати його функції та класи для розробки Telegram-ботів на PHP.

Залежності для розробки проекту:

- "fakerphp/faker": "^1.9.1" - генератор тестових даних.
- "laravel/pint": "^1.0" - пакет для генерації маршрутів.
- "laravel/sail": "^1.15" - легкий спосіб налаштування контейнера Docker для розробки Laravel-додатків.
- "mockery/mockery": "^1.4.4" - бібліотека для тестування.
- "nunomaduro/collision": "^6.1" - пакет для виведення дружелюбних повідомлень про помилки.
- "phpunit/phpunit": "^9.5.10" - бібліотека для тестування.
- "spatie/laravel-ignition": "^1.0" - пакет для відлагодження та аналізу помилок.

Також є налаштування автозавантаження класів для проекту, скрипти для виконання після різних подій та інші налаштування для оптимізації роботи проекту.

Файл docker-compose.yml використовується для конфігурації та запуску контейнерів з додатком (рис.3.12). У файлі можна вказати параметри для кожного контейнера, такі як образ, порти, змінні середовища, томи, мережі та інші налаштування. Конфігурація включає список контейнерів, що будуть запущені, та інші параметри, такі як назва проекту, мережі, об’єднання контейнерів в сервіси, змінні середовища і т.д.

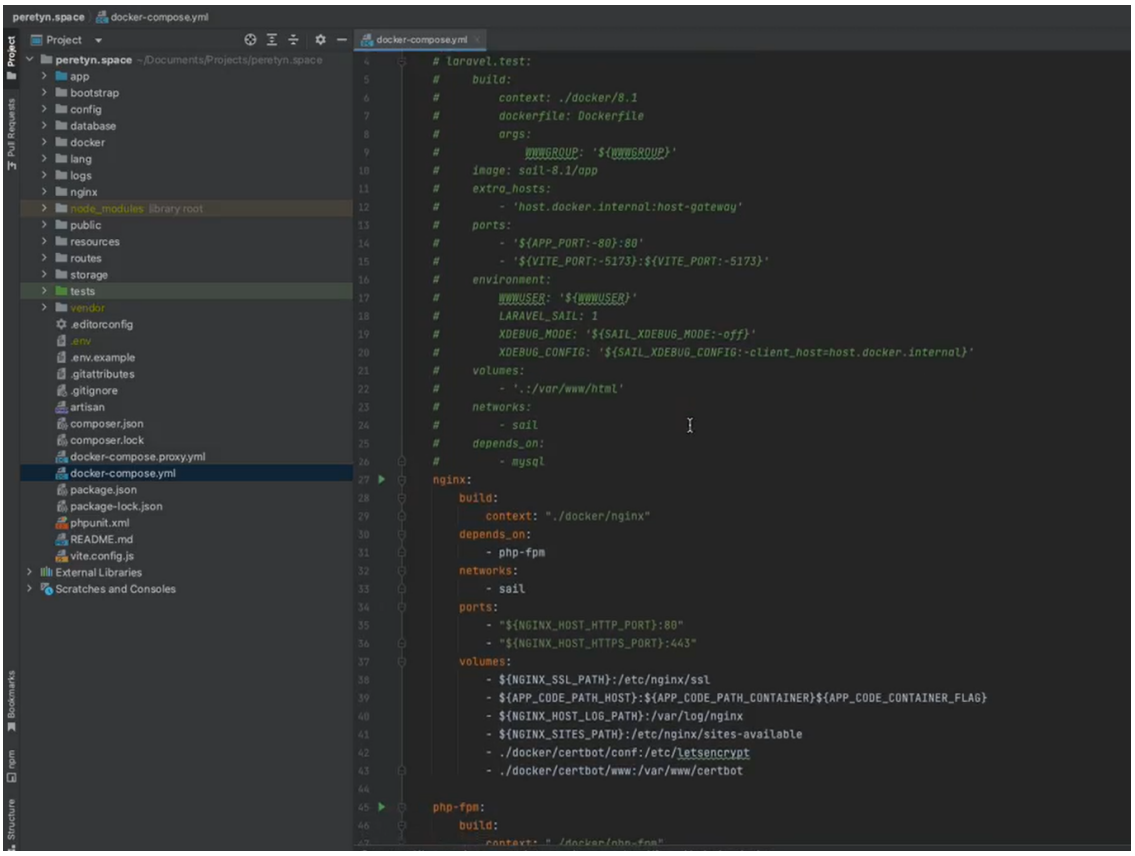


Рисунок 3.12 – Фрагмент лістингу для конфігурації та запуску контейнерів з додатком

Фрагмент лістингу з файлу "compose.yml" визначає конфігурацію та взаємодію сервісів в Docker Compose проекті. У цьому файлі описані різні сервіси, які будуть запущені разом, в тому числі nginx, php-fpm, mysql та certbot.

Кожен з цих сервісів має свої налаштування, такі як контекст будування Docker-образу, залежності від інших сервісів, мережі, порти, середовища, які сервіс використовує, і т.д.

Сервіс nginx, наприклад, будується з контексту "./docker/nginx", має залежність від сервісу php-fpm, використовує мережу sail, прослуховує порти 80 та 443, має кілька об'ємів, включаючи SSL-сертифікати, журнали, конфігураційні файли тощо.

Крім того, у файлі визначені мережі та томи (volumes), які використовуються для забезпечення взаємодії та зберігання даних між різними сервісами.

У цілому, файл "compose.yml" забезпечує зручний та організований спосіб налаштування та запуску додатків, складаються з кількох сервісів, в середовищі Docker Compose.

3.4.1 Розробка бази даних

Для зберігання інформації про переміщення користувачів митні кордони України, необхідні для роботи бота та ведення статистики, використовується база даних. Для створення цієї бази була розроблена діаграма, що відображає структуру бази даних на рис.3.13.

Для роботи з веб-орієнтованою системою створено БД, які містять інформацію про:

- митні пункти пропуску;
- дані користувачів;
- відгуки;
- позиція пунктів на карті веб-орієнтованої системи;
- історію користувачів в Телеграм;
- донтати;
- тощо.

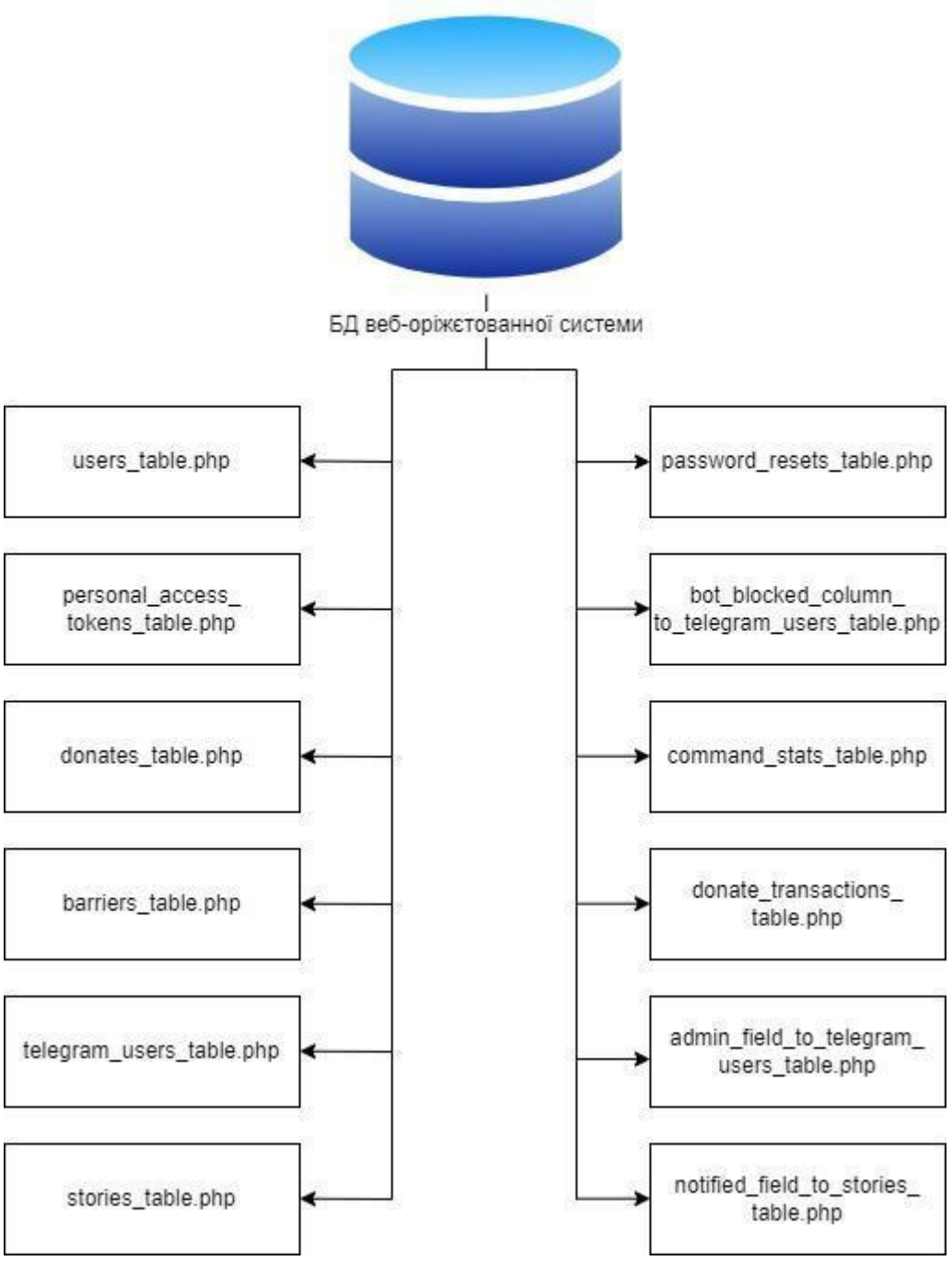


Рисунок 3.13 – Структура баз даних веб-орієнтовано системи

БД «Barrier» код являє собою PHP-скрипт, який визначає клас з назвою "BarrierSeeder". Цей клас розширює клас "Seeder", який надається фреймворком Laravel, і використовується для заповнення бази даних початковими даними про пункти пропуску державного кордону України. Метод "run" класу BarrierSeeder викликається і створює нові екземпляри моделі Barrier за допомогою методу "create". Модель Barrier імпортується за

допомогою інструкції use, і її властивості, такі як "name", "type", "has_pedestrians", "position_top" і "position_left", встановлюються зі значеннями для кожного нового екземпляра моделі (рис.3.14). Ці моделі представляють назви митниць, розташовані на різних пунктах пропуску вздовж українського кордону.

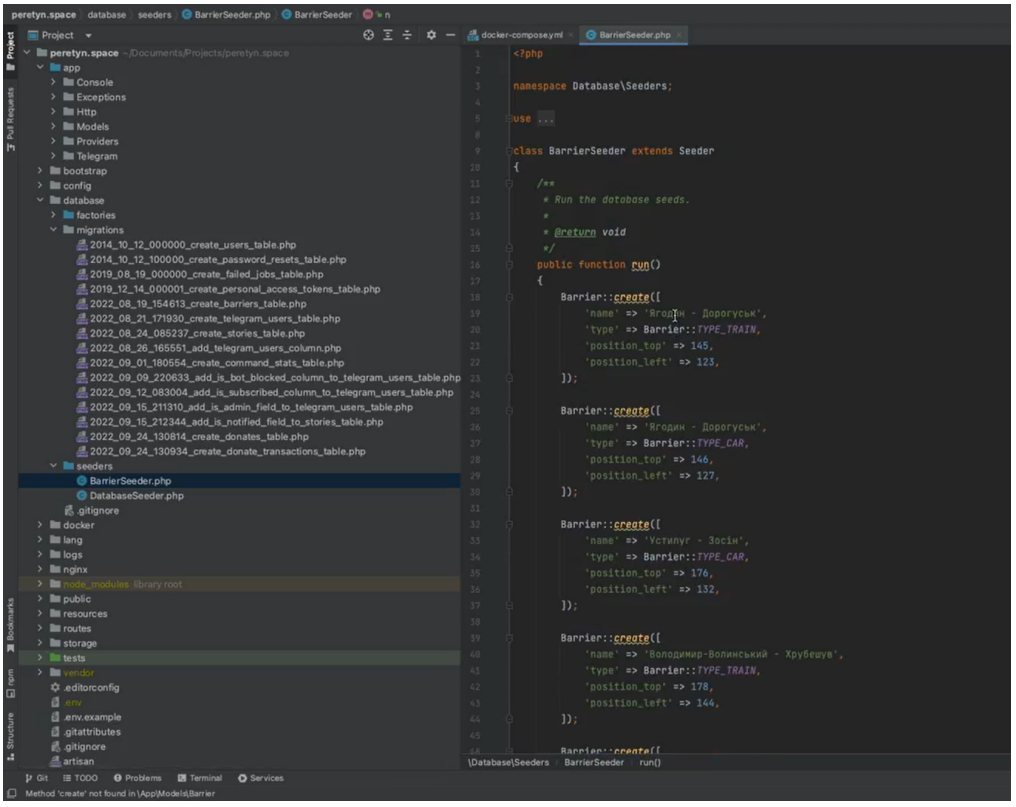


Рисунок 3.14 – Фрагмент лістингу БД "BarrierSeeder"

БД "users" (рис.3.15) створює таблицю у базі даних за допомогою фасаду Schema. Таблиця "users" містить поля для збереження даних про користувачів, такі як ім'я, email, хешований пароль та інші. Метод up() виконує створення таблиці "users", додавши до неї відповідні стовпці. id() - це метод, що створює автоінкрементний первинний ключ для таблиці "users". Метод down() виконує зворотню міграцію - видаляє таблицю "users". Використовується для збереження даних про користувачів у веб-додатку.

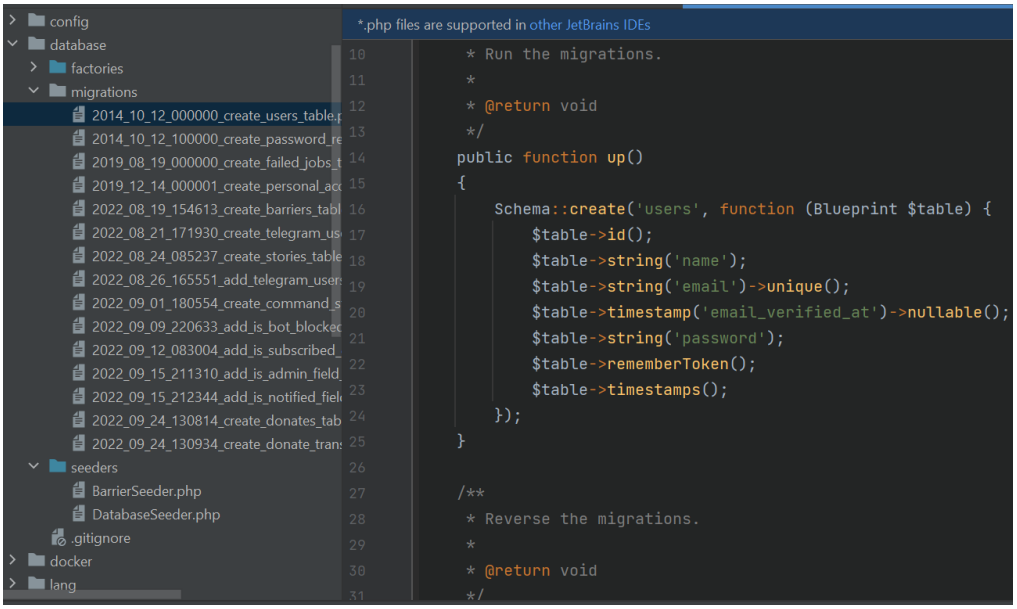


Рисунок 3.15 – Фрагмент лістингу БД "users"

Лістінг створених БД наведено в додатку БД. Для роботи з веб-орієнтованої статистичної платформи з геоінформаційною підтримкою ми створили таблицю “stories” (рис.3.16).

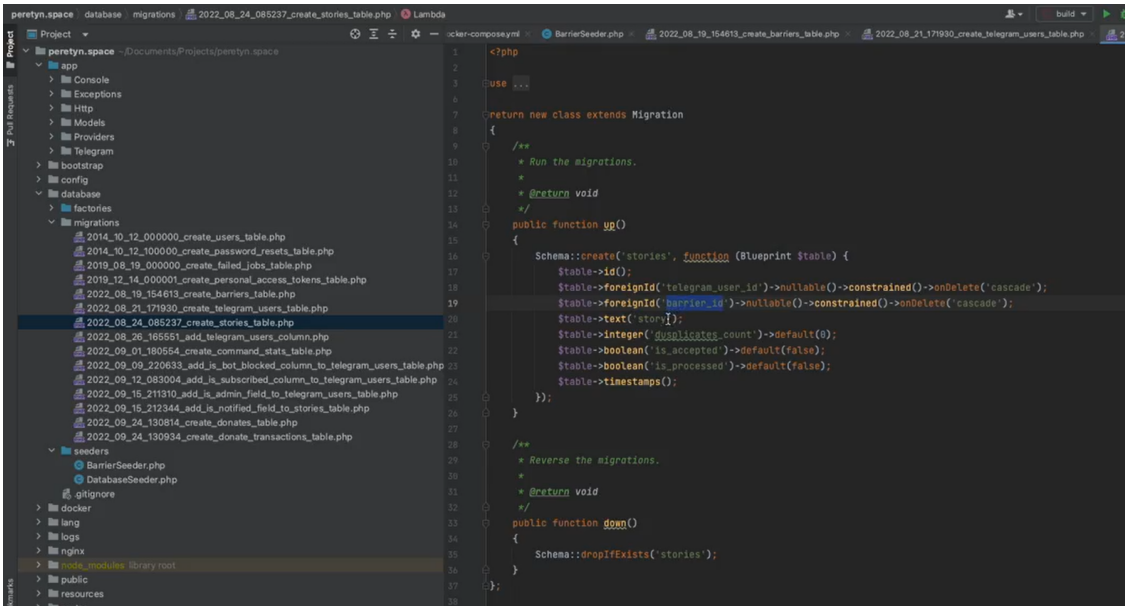


Рисунок 3.16 – Фрагмент лістингу БД “stories”

Вона була створена з метою ідентифікації введених даних про пункт пропуску, відгук, дату, час від користувача з Телеграму. При цьому, якщо

користувач повторно введе відгук, то дублікат не буде створюватись, а буде враховано кількість разів введеного відгуку. Таблиця містить наступні поля:

- id - унікальний ідентифікатор історії;
- telegram_user_id - зовнішній ключ, що посилається на ідентифікатор користувача Telegram, який подав історію. Це поле може бути порожнім, оскільки необов'язково всі історії повинні мати зв'язок з користувачами Telegram;
- barrier_id - зовнішній ключ, що посилається на ідентифікатор бар'єру, який став перешкодою у житті героя історії. Це поле також може бути порожнім;
- story - текст історії;
- duplicates_count - кількість повторів історії в базі даних. Початково це поле має значення 0;
- is_accepted - прапорець, який вказує, чи була історія підтверджена модератором. Початково це поле має значення false;
- is_processed - прапорець, який вказує, чи була історія оброблена ботом. Початково це поле має значення false;
- timestamps - поля, які зберігають дату та час створення та оновлення запису.

Ця таблиця також встановлює зв'язки з іншими таблицями за допомогою методу constrained(), що дозволяє створити зовнішні ключі та налаштувати їх для автоматичного видалення залежних записів при видаленні запису з батьківської таблиці.

Схема таблиць БД наведено на рис.3.17 та додатку Б.

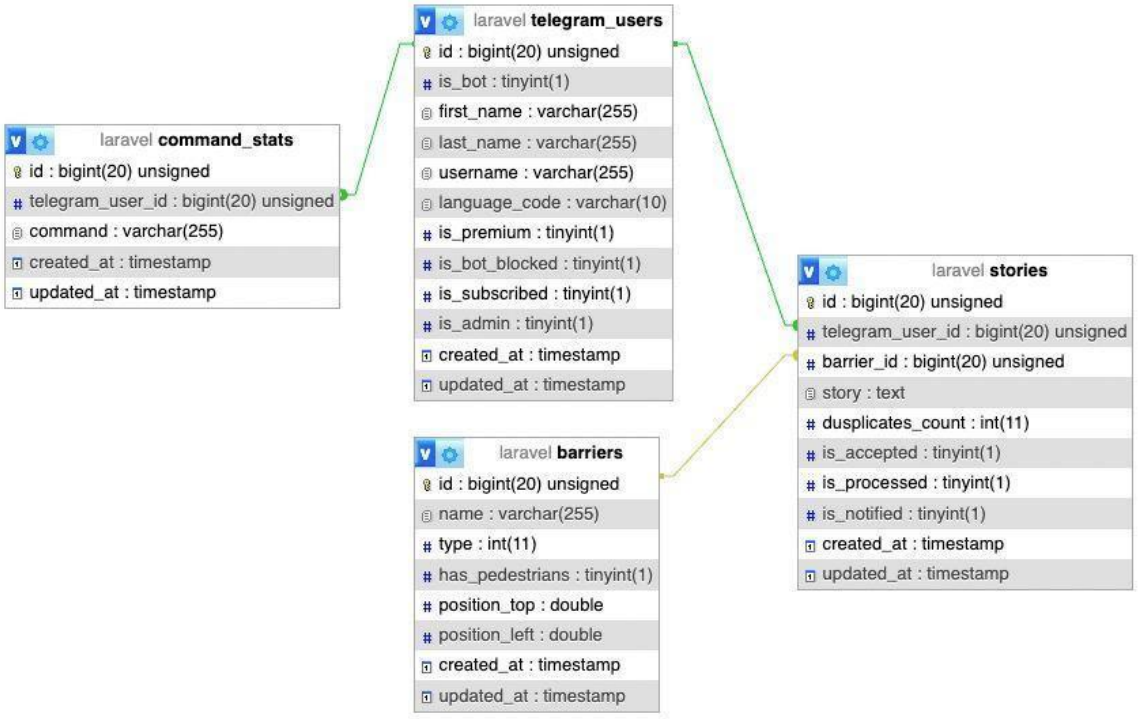


Рисунок 3.17 – Схема (кор логіка) БД для взаємодії статистика, пункти, користувачі телеграм

Сервіс буде забезпечувати роботу користувачів за допомогою створення сесій та токенів доступу (рис.3.18). Для подальшої ідентифікації користувача буде використовуватись механізм PHP та webhook, polling.

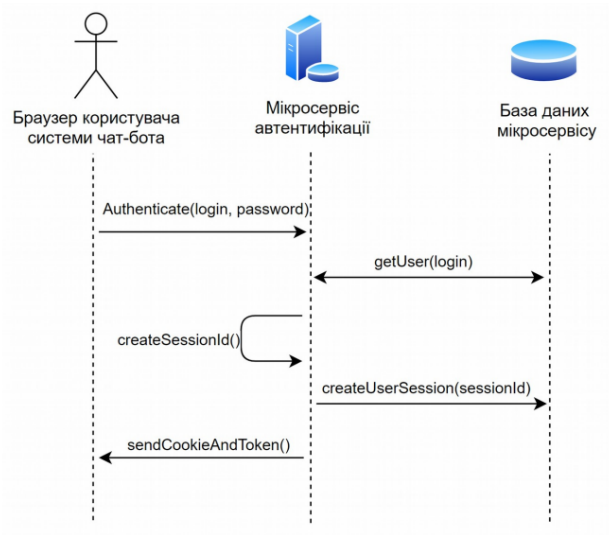


Рисунок 3.18 – Аутентифікація користувача у системі чат-боту

Для узагальнення процесу взаємодії користувача з базою даних, що дає відповіді на повідомлення, що надсилає користувач представити схематично на рис.3.19.

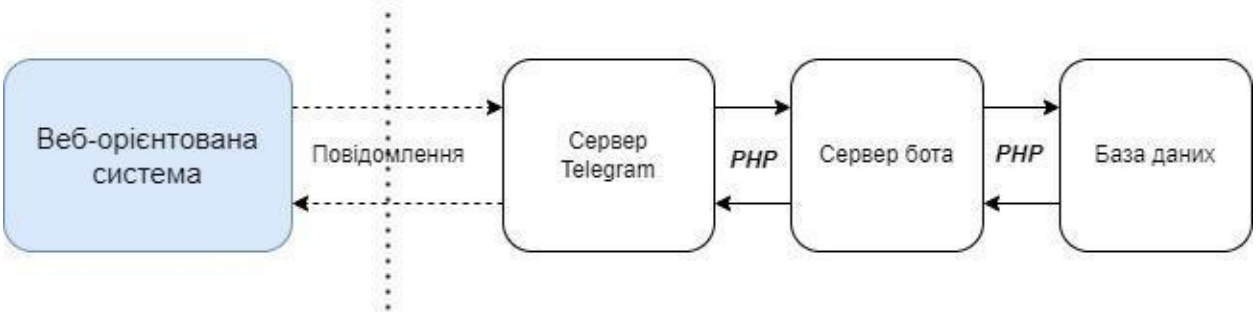


Рисунок 3.19 – Структурна схема взаємодії користувача веб-орієнтованої системи з базою даних

3.5 Тестування веб-орієнтованої системи

В цьому пункті буде здійснено тестування основних команд телеграм-бота, а також запитів серверу. Тестування буде виконуватись з комп'ютера.

Для взаємодії між внутрішніми компонентами, розробленої веб-орієнтованої системи ілюструє внутрішню структуру програмного продукту у вигляді структурної діаграми на рис.3.20.

Як описувалось раніше наша веб-орієнтована система має головну сторінку, на якій представлено карту України з іконками, що показують перетини кордону для пішоходів, автомобілів та потягів.

Кнопка «Статистика» містить інформацію від користувачів, про проходження митного контролю, про позитивні чи негативні враження (рис.3.21).

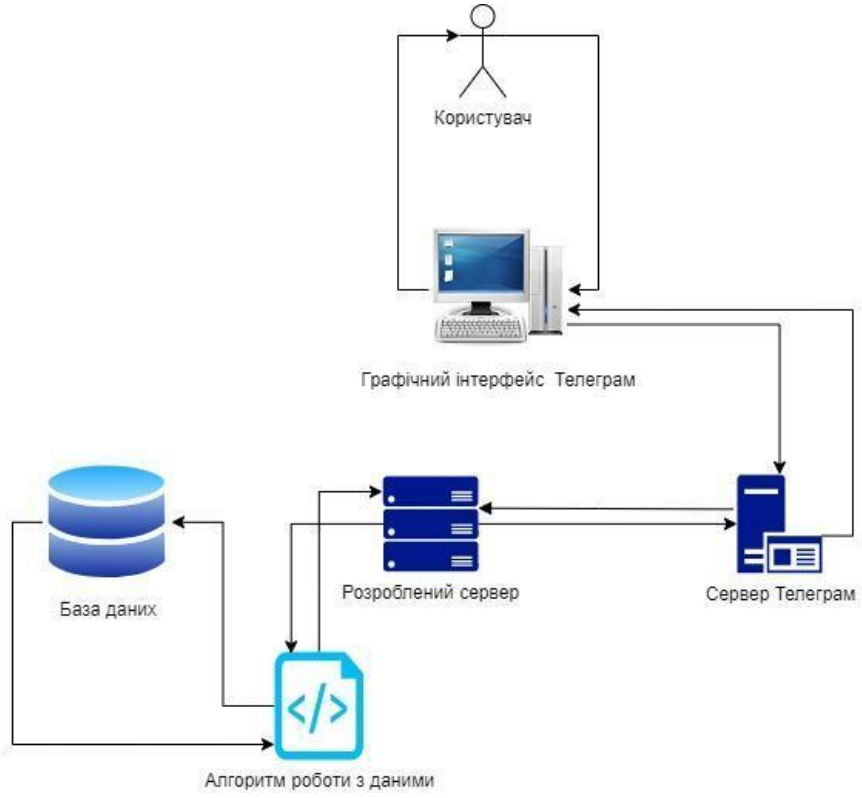


Рисунок 3.20 – Діаграма взаємодії користувачів з прогамним продуктом

На рис. 3.21 можна бачити статистичні дані по кожній митниці (ліворуч позитивні відгуки, а праворуч – негативні).

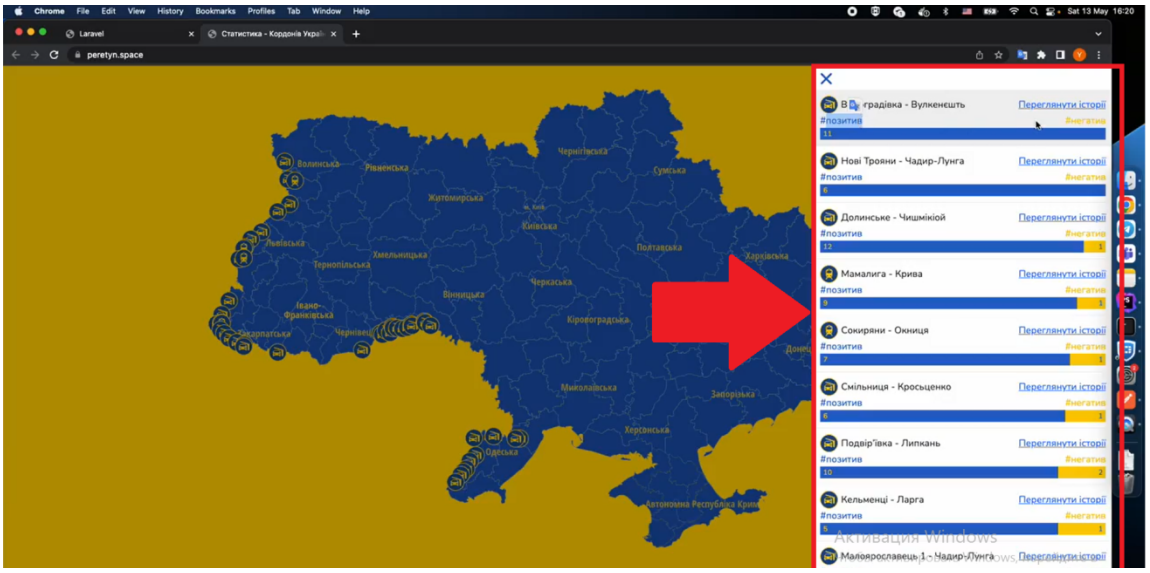


Рисунок 3.21 – Вкладка «Статистика»

Праворуч є посилання «Переглянути історію» (рис.3.22). Відкривається інформація про обраний пункт пропуску. Для даної кваліфікаційної роботи

ми наповнили випадковими значеннями. Виводиться ім’я користувача (але з урахуванням прав користувачів, ім’я не повністю відображається), дата та час сформованого відгуку. На одній сторінці виводиться 10 відгуків, для продовження перегляду треба натиснути на кнопку «Наступна».

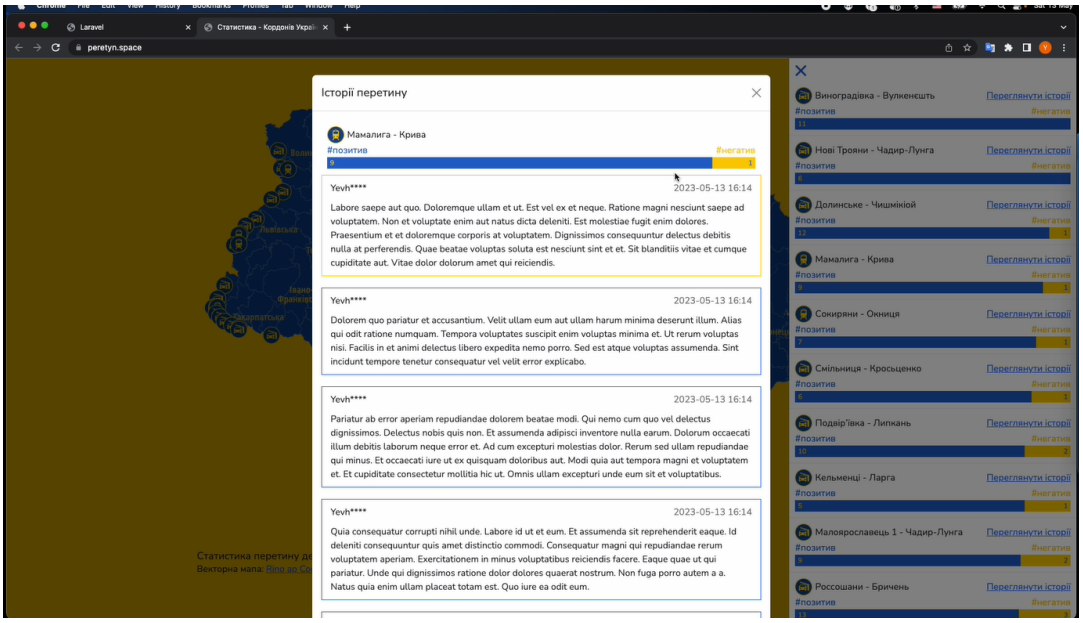


Рисунок 3.22 – Перегляд відгуків

Для того щоб додати відгук і як результат поповнити статистичні дані потрібно перейти до боту в Телеграм (рис.3.23). На головній сторінці є іконка Телеграм треба на неї натиснути.

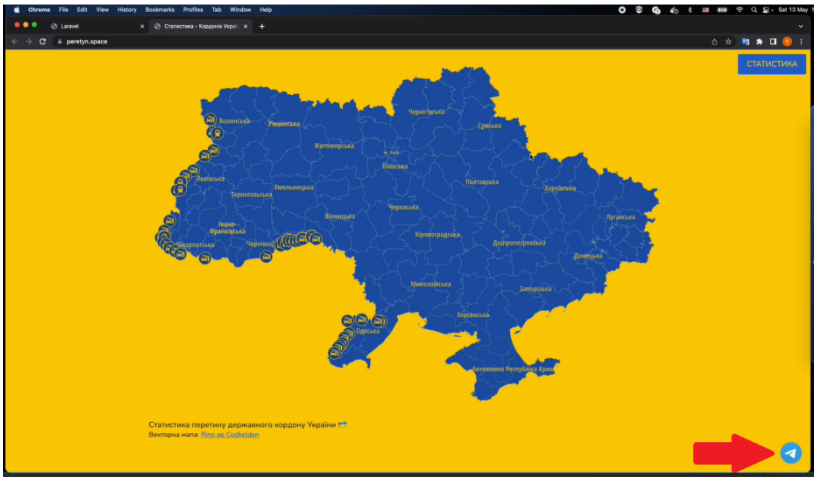


Рисунок 3.23 – Взаємодія з чат-ботом

Щоб розпочати використання бота, слід натиснути на кнопку «Start» на головному екрані (рис. 3.24). Після цього ви побачите меню користувача, яке зображено на рисунку 3.25.

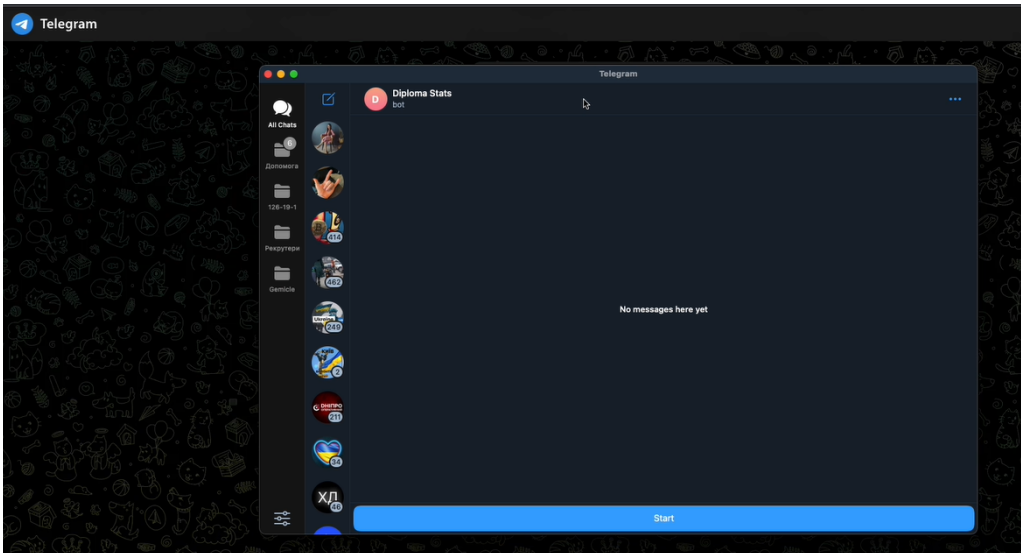


Рисунок 3.24 – Початок роботи з ботом

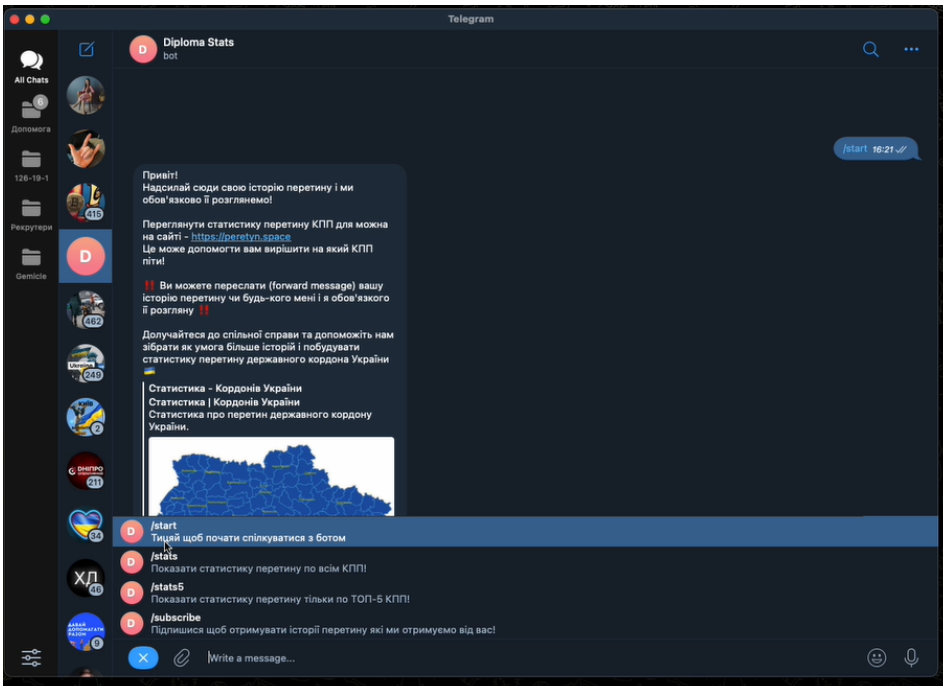


Рисунок 3.25 – Команди бота

Протестуємо команду «Топ 5» (рис.3.26). бот пропонуємо обрати кількість днів для виведення статистики:

– 7 днів;

– 31 день;
– 365 днів.

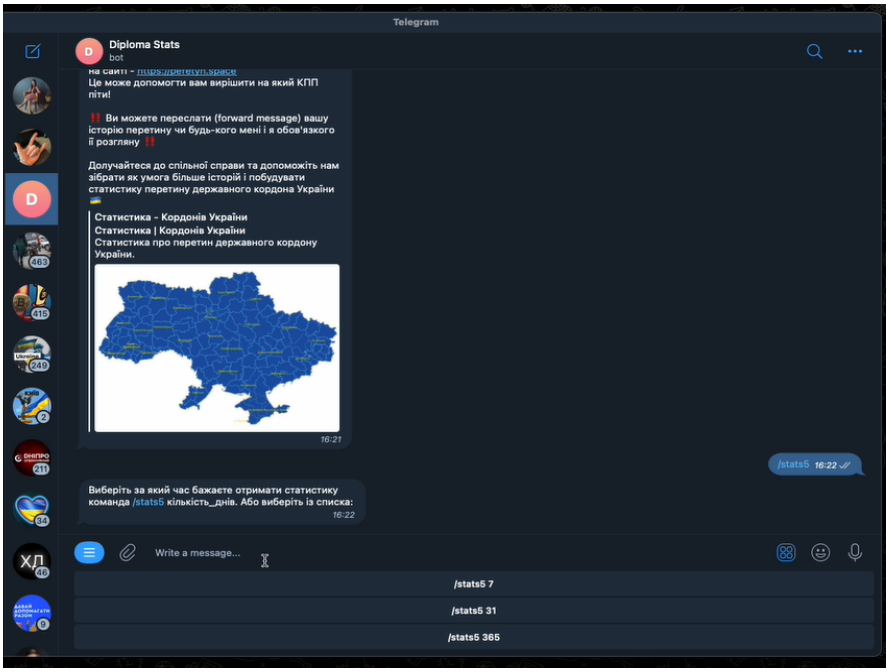
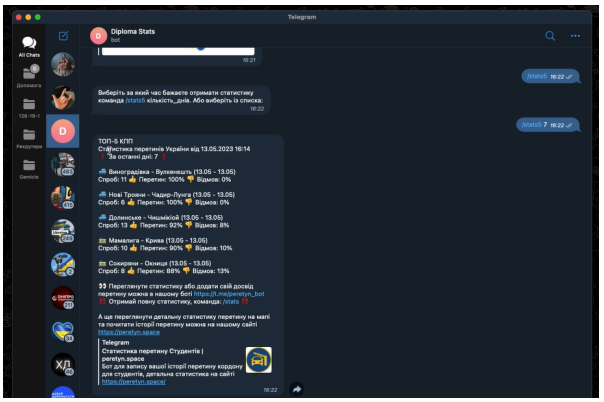
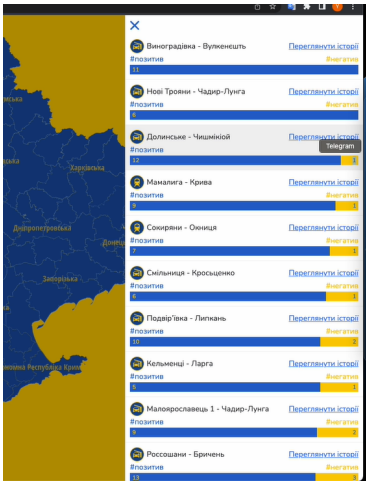


Рисунок 3.26 – Тестування команди «Топ5»

Оберемо за 7 днів (рис.3.27). На екран виводиться популярна інформація за 7 днів та тип транспорту, яким було перетнуто кордон. На рис.3.27 перевіряємо статистичні дані бота з даними сайту і переконуємось, що все працює правильно.





а) б)

Рисунок 3.27 – Статистика за 7 днів

Тепер спробуємо додати нову історію. Напишемо невеликий текст і бот пише повідомлення, що інформація за коротка. Це дозволяє уникнути використання «ботів» замість справжніх користувачів.

Рисунок 3.28 – Введення даних

Спробуємо написати більше тексту і після цього бачимо, що чат-бот пропонуємо обрати пункт пропуску для даного відгуку.

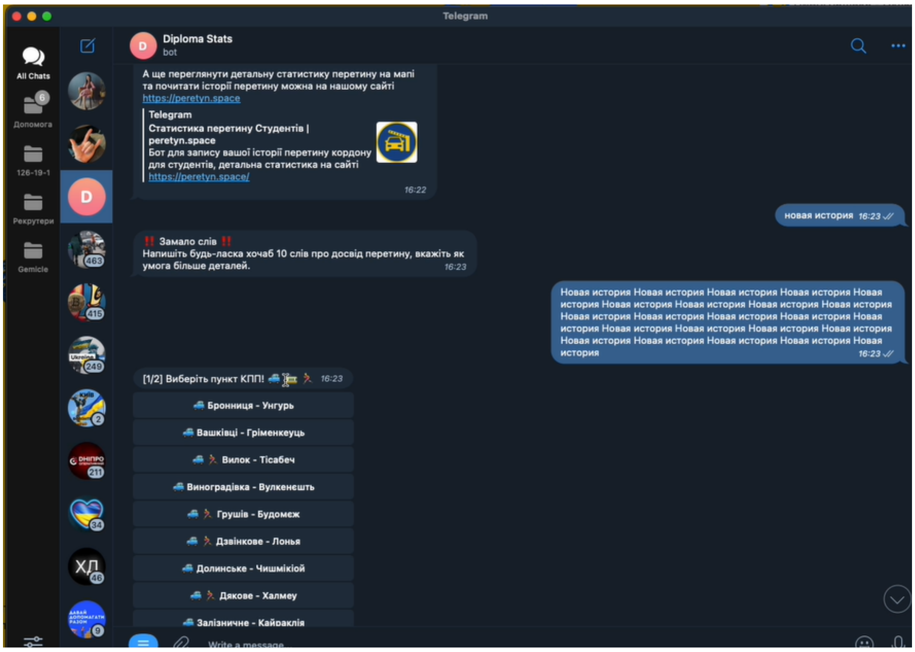
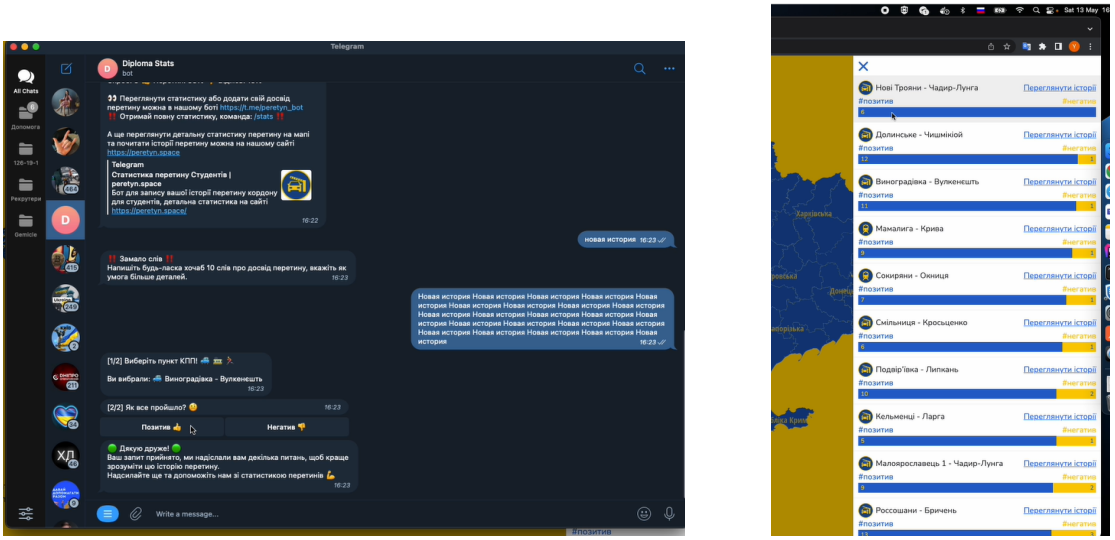


Рисунок 3.29 – Надання відгуку

Обираємо пункт пропуску, обираємо «Виноградівка» і ставимо оцінку проходження митного контролю (поставимо негативний). Перевіримо оновлену інформацію на веб-орієнтованій системи (рис. 3.30). На рис. 3.30 б можемо побачити, що додано ожин негативний відгук.



а) б)

Рисунок 3.30 – Тестування доданої інформації

Перевіримо адміністрування нашої веб-орієнтованої системи. На рис. 3.31 наведено дані про адміністратора.

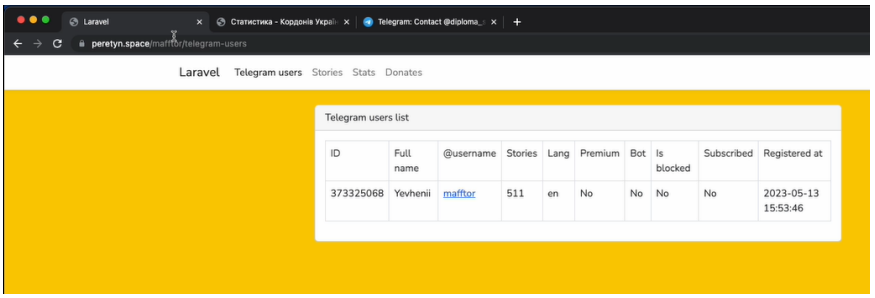


Рисунок 3.31 – Обліковий запис адміністратора

Вгорі наведено доступне меню для адміністратора, який може переглядати інформацію про користувача, мову інтерфейсу, кількість історій, чи це бот, тощо. Наприклад вкладка «Stories» містить відгуки користувачів (рис.3.32).

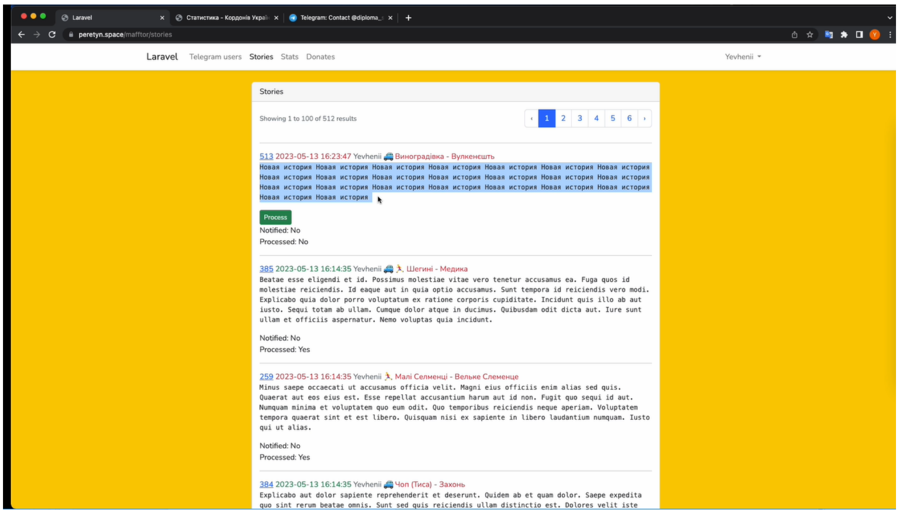


Рисунок 3.32 – Редагування відгуків

Отже, в цьому пункті було проведено тестування серверу на запити, перевірено відображення користувацького інтерфейсу та перевірено базу даних на наявність оновлень.

3.6 Висновки до третього розділу

У даному розділі була розглянута структура і функціональні можливості веб-орієнтованої платформи. Основним елементом платформи є сайт, який містить карту України з інтерактивними іконками, що показують перетини кордону для пішоходів, автомобілів та потягів. На головній сторінці розміщена кнопка "Статистика", яка відкриває сторінку з відгуками користувачів про перетин кордону. Структура сторінки статистики дозволяє сортувати відгуки за рейтингом, алфавітом або шукати за назвою пункту митного контролю. Кожен відгук містить інформацію про користувача, дату та час створення, оцінку та сам відгук.

Важливим аспектом розробки є зберігання та захист інформації. Усі дані в системі зберігаються в базі даних MySQL 8.0, що забезпечує надійне зберігання та швидкий доступ до них. Безпека системи забезпечена за допомогою механізмів аутентифікації та авторизації, що захищають її від несанкціонованого доступу.

Проведені тестування серверу на запити дозволяє зручно та ефективно надавати інформацію користувачам, сприяти обміну досвідом та взаємодії між користувачами та адміністраторами. Крім того, система забезпечує надійне зберігання даних та захист їх від несанкціонованого доступу.

ВИСНОВКИ

В рамках випускної кваліфікаційної роботи бакалавра було розроблено та впроваджено веб-орієнтовану статистичну платформу з геоінформаційною підтримкою. Ідея роботи полягала в перегляді інформації, щодо перетину кордонів України про враження користувачів та ведення статистичних даних з використанням Telegram боту, що дозволяє візуалізувати статистичні дані та геодані на карті в режимі реального часу

В ході розробки платформи використано модульний підхід та розроблено базу даних, що дозволило створити зручний інтерфейс користувача, який дозволяє взаємодіяти зі статистичними даними та геоданими на карті. Реалізовано функції додавання, редагування та видалення даних.

В роботі використано такі технології, як PHP, Laravel, JavaScript, Vue.js, Leaflet.js, та інші. Застосування сучасних технологій дозволило забезпечити швидке та зручне взаємодію користувача з платформою.

Проведено тестування функціональності та продуктивності розробленої платформи, що показало її ефективність та надійність. Розроблена платформа може бути використана в різних сферах, які потребують візуалізації статистичних даних та геоданих на карті.

Розроблена веб-орієнтована статистична платформа з геоінформаційною підтримкою може бути корисною для різних користувачів, які працюють зі статистичними даними та геоданими.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Еволюція до web 4.0. коротка історія розвитку інтернет технологій web 1.0, web 2.0, web 3.0. Переваги web 4.0. vc.ru. [Електрон. ресурс]. – Спосіб доступу: URL: https://vc.ru/u/739868-andrey-nikolaev/238951-evolyuciya-do-web-4-0-kratkaya-istoriya-razvitiya-internet-tehnologiy-web-1-0-web-2-0-web-3-0-preimushchestva-web-4-0

2. Що таке чат-бот? [Електронний ресурс] – Режим доступу до ресурсу: https://creativesmm.com.ua/shho-take-chatbot-ta-komu-vonu-potribni/

3. Borgatti, S. Analyzing social network

4. Telegram FAQ [Електронний ресурс]. – Режим доступу: https://telegram.org/faq Chandra, A. (2018). Building Chatbots with Google Dialogflow: Create Custom Chatbots for Your Business using Dialogflow. Packt Publishing.

5. Інтерактивна карта Держприкордонслужби [Електронний ресурс] – Режим доступу до ресурсу: https://dpsu.gov.ua/ua/map/

6. Телеграм-ботом «Кордони України Інфо» [Електронний ресурс] – Режим доступу до ресурсу: https://t.me/ukraineborders_bot

7. Schwichtenberg, C. Developing Bots with Microsoft Bots Framework: Create Intelligent Bots using MS Bot Framework and Azure Cognitive Services. Apress : Apress; 1st ed. Edition, 2019, 298 p.

8. Lavin, A. Hands-On Chatbots and Conversational UI Development: Build chatbots and voice user interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills: Packt Publishing, 2019, 392 p.

9. Welling, L., Thomson, L. PHP and MySQL Web Development. Addison-Wesley Professional, 2017, 960 p.

10. Freeman E., Freeman E. Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages (3rd Edition). O'Reilly Media, 2020, 485 p.

11. Redmond E., Wilson, J. R. SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL. Addison-Wesley Professional, 2018, 763 p.

12. Lacroix E., Akyildiz O. Bootstrap 5 From Scratch: Build Responsive Websites and Applications with Bootstrap 5: Independently published, 2021, 208 p.

13. Laravel Mix Cookbook: 100+ recipes for building and optimizing web applications with Laravel Mix [Електронний ресурс] - https://laravel.com/docs/10.x/mix

14. Sass Team. (2021). Sass Documentation. Retrieved [Електронний ресурс] - https://sass-lang.com/documentation

15. Freeman A., Freeman E. jQuery: Novice to Ninja (2nd Edition). SitePoint, 2015, 368 p.

16. Порівняння MariaDB та MySQL 2019 [Електронний ресурс]:– Режим доступу: https://netpoint-dc.com/blog/mariadb-mysql-2019/.

17. Telegram Bot API документація [Електронний ресурс]:– Режим доступу: https://core.telegram.org/bots/api

18. Telegram-bot-api VS aiogram [Електронний ресурс]:– Режим доступу: https://www.libhunt.com/compare-tdlib--telegram-bot-api-vs-aiogram

Додаток А.
Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Найменування	Кільк. аркушів	Примітки
1				
2		Документація		
3				
4	**ІТКІ.КР.23.06.ДА.ПЗ**	Пояснювальна записка		Формат А4
5				
6		Презентація		
7				
8		Диск CD-R з презентацією	1	Диск CD-R
9				

Зм.	Ар-куш	№ докум.	Підпис	Дата	**ІТКІ.КР.23.06.ДА.ПЗ.**			
Розроб.		Міняйленко Є.О.			**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Арку-шів
Керівник		Соколова Н.О.				Н	1	1
Рецензент		Ширін А.Л.				НТУ «ДП», 12; 126-19-1		
Н.контр.		Коротенко Г.М.						
Зав. каф.		Гнатушенко В.В.						

Додаток Б.

Фрагмент лістингу програми

Лістинг **docker-compose.yml**

```
# For more information: https://laravel.com/docs/sail
version: '3'
services:
    # laravel.test:
    #     build:
    #         context: ./docker/8.1
    #         dockerfile: Dockerfile
    #         args:
    #             WWWGROUP: '${WWWGROUP}'
    #     image: sail-8.1/app
    #     extra_hosts:
    #         - 'host.docker.internal:host-gateway'
    #     ports:
    #         - '${APP_PORT:-80}:80'
    #         - '${VITE_PORT:-5173}:${VITE_PORT:-5173}'
    #     environment:
    #         WWWUSER: '${WWWUSER}'
    #         LARAVEL_SAIL: 1
    #         XDEBUG_MODE: '${SAIL_XDEBUG_MODE:-off}'
    #         XDEBUG_CONFIG: '${SAIL_XDEBUG_CONFIG:-client_host=host.docker.internal}'
    #     volumes:
    #         - '.:/var/www/html'
    #     networks:
    #         - sail
    #     depends_on:
    #         - mysql
    nginx:
        build:
            context: "./docker/nginx"
        depends_on:
            - php-fpm
        networks:
            - sail
        ports:
            - "${NGINX_HOST_HTTP_PORT}:80"
            - "${NGINX_HOST_HTTPS_PORT}:443"
        volumes:
            - ${NGINX_SSL_PATH}:/etc/nginx/ssl
            -
${APP_CODE_PATH_HOST}:${APP_CODE_PATH_CONTAINER}${APP_CODE_CONTAINER_FL
AG}
            - ${NGINX_HOST_LOG_PATH}:/var/log/nginx
            - ${NGINX_SITES_PATH}:/etc/nginx/sites-available
            - ./docker/certbot/conf:/etc/letsencrypt
            - ./docker/certbot/www:/var/www/certbot

    php-fpm:
        build:
            context: "./docker/php-fpm"
        volumes:
```

```
        -
${APP_CODE_PATH_HOST}:${APP_CODE_PATH_CONTAINER}${APP_CODE_CONTAINER_FL
AG}
      networks:
        - sail
      expose:
        - "9000"
  mysql:
      image: 'mysql/mysql-server:8.0'
      ports:
        - '${FORWARD_DB_PORT:-3306}:3306'
      environment:
        MYSQL_ROOT_PASSWORD: '${DB_PASSWORD}'
        MYSQL_ROOT_HOST: "%"
        MYSQL_DATABASE: '${DB_DATABASE}'
        MYSQL_USER: '${DB_USERNAME}'
        MYSQL_PASSWORD: '${DB_PASSWORD}'
        MYSQL_ALLOW_EMPTY_PASSWORD: 1
      volumes:
        - 'sail-mysql:/var/lib/mysql'
        -
'./vendor/laravel/sail/database/mysql/create-testing-database.sh:/docker-entrypoint-initdb.d/10-create-testi
ng-database.sh'
      networks:
        - sail
      healthcheck:
        test: ["CMD", "mysqladmin", "ping", "-p${DB_PASSWORD}"]
        retries: 3
        timeout: 5s
  certbot:
      # build:
      #     context: "./docker/certbot"
      # volumes:
      #     - ./docker/certbot/data/certbot/certs/:/var/certs
      #     - ./docker/certbot/letsencrypt/:/var/www/letsencrypt
      # environment:
      #     - CN="${APP_DOMAIN}"
      #     - EMAIL="${APP_EMAIL}"
      # networks:
      #     - sail
      image: certbot/certbot
      volumes:
        - ./docker/certbot/conf:/etc/letsencrypt
        - ./docker/certbot/www:/var/www/certbot
networks:
  sail:
      driver: bridge
volumes:
  sail-mysql:
      driver: local
```

Лістинг **app.php**

```
<?php

/*
|--------------------------------------------------------------------------
| Create The Application
|--------------------------------------------------------------------------
|
| The first thing we will do is create a new Laravel application instance
| which serves as the "glue" for all the components of Laravel, and is
| the IoC container for the system binding all of the various parts.
|
*/

$app = new Illuminate\Foundation\Application(
    $_ENV['APP_BASE_PATH'] ?? dirname(__DIR__)
);

/*
|--------------------------------------------------------------------------
| Bind Important Interfaces
|--------------------------------------------------------------------------
|
| Next, we need to bind some important interfaces into the container so
| we will be able to resolve them when needed. The kernels serve the
| incoming requests to this application from both the web and CLI.
|
*/

$app->singleton(
    Illuminate\Contracts\Http\Kernel::class,
    App\Http\Kernel::class
);

$app->singleton(
    Illuminate\Contracts\Console\Kernel::class,
    App\Console\Kernel::class
);

$app->singleton(
    Illuminate\Contracts\Debug\ExceptionHandler::class,
    App\Exceptions\Handler::class
);

/*
|--------------------------------------------------------------------------
| Return The Application
|--------------------------------------------------------------------------
|
| This script returns the application instance. The instance is given to
| the calling script so we can separate the building of the instances
| from the actual running of the application and sending responses.
|
*/
```

```
return $app;
```

Лістинг **services.php**

```
<?php

return [

    /*
    |--------------------------------------------------------------------------
    | Third Party Services
    |--------------------------------------------------------------------------
    |
    | This file is for storing the credentials for third party services such
    | as Mailgun, Postmark, AWS and more. This file provides the de facto
    | location for this type of information, allowing packages to have
    | a conventional file to locate the various service credentials.
    |
    */

    'mailgun' => [
        'domain' => env('MAILGUN_DOMAIN'),
        'secret' => env('MAILGUN_SECRET'),
        'endpoint' => env('MAILGUN_ENDPOINT', 'api.mailgun.net'),
        'scheme' => 'https',
    ],

    'postmark' => [
        'token' => env('POSTMARK_TOKEN'),
    ],

    'ses' => [
        'key' => env('AWS_ACCESS_KEY_ID'),
        'secret' => env('AWS_SECRET_ACCESS_KEY'),
        'region' => env('AWS_DEFAULT_REGION', 'us-east-1'),
    ],
```

Лістинг **telebot.php**

```
<?php

return [
    /*--------------------------------------------------------------------------
    | Default Bot Name
    |--------------------------------------------------------------------------
    |
    | Here you may specify which of the bots you wish to use as
    | your default bot for regular use.
    |
    */

    'default' => 'bot',

    /*--------------------------------------------------------------------------
    | Your Telegram Bots
    |--------------------------------------------------------------------------
    | You may use multiple bots. Each bot that you own should be configured here.
    |
    | See the docs for parameters specification:
```

```
    | https://westacks.github.io/telebot/#/configuration
    |
    */

    'bots' => [
        'bot' => [
            'token' => env('TELEGRAM_BOT_TOKEN'),
            'name' => env('TELEGRAM_BOT_NAME', null),
            'api_url' => env('TELEGRAM_API_URL', 'https://api.telegram.org/bot{TOKEN}/{METHOD}'),
            'exceptions' => true,
            'async' => false,

            'webhook' => [
                // 'url'              => env('TELEGRAM_BOT_WEBHOOK_URL',
env('APP_URL').'/telebot/webhook/bot/'.env('TELEGRAM_BOT_TOKEN')),
                // 'certificate'      => env('TELEGRAM_BOT_CERT_PATH',
storage_path('app/ssl/public.pem')),
                // 'ip_address'       => '8.8.8.8',
                // 'max_connections'  => 40,
                // 'allowed_updates'  => ["message", "edited_channel_post", "callback_query"],
                // 'secret_token'     => env('TELEGRAM_KEY', null),
            ],

            'poll' => [
                // 'limit'            => 100,
                // 'timeout'          => 0,
                // 'allowed_updates'  => ["message", "edited_channel_post", "callback_query"]
            ],

            'handlers' => [
                \App\Telegram\Commands\StartCommand::class,
                \App\Telegram\Commands\StatsCommand::class,
                \App\Telegram\Commands\Stats5Command::class,
                \App\Telegram\Commands\SubscribeCommand::class,

                \App\Telegram\Handlers\StoryHandler::class,
                \App\Telegram\Handlers\CallbackQueryHandler::class,
            ],
        ],

        // 'second_bot' => [
        //     'token'        => env('TELEGRAM_BOT2_TOKEN',
'123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11'),
        // ],
    ],
];
```

Додаток В. Фрагмент лістингу створення таблиць БД

Фрагмент лістингу **backup.sql**

```
MysQL dump 10.13
Distrib 8.0.32, for Linux (×86 64)
Host: localhost
atabases aravel
Server version
8.0.32
/*!40101 SET @OLD CHARACTER SET CLIENT @@CHARACTER SET CLIENT */;
7*!40101 SET COLD_CHARACTER_SET_RESULTS-@@CHARACTER_SET_RESULTS */:
/*!40101 SET COLD COLLATION CONNECTION @@COLLATION CONNECTION
/*!50503 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ ZONE-@@TIME_ZONE */;
/*!40103 SET TIME_ ZONE-*+00:00' */;
7*!40014 SET @OLD_UNIQUE_CHECKS @@UNIQUE_CHECKS, UNIQUE_CHECKS0 */;
/*!40014 SET GOLD FOREIGN KEY CHECKS @@FOREIGN KEY CHECKS, FOREIGN KEY
CHECKS»0 */;
/*!40101 SET GOLD_SQL_MODE-Q0SQL _MODE, SQL _MODE- "NO AUTO_VALUE_ON
_ZERO* */;
/*!40111 SET GOLD_SQL_NOTES @@SQL_NOTES, SQL_NOTES 0 */;
Table structure for table 'barriers*
DROP TABLE IF EXISTS 'barriers';
7*!40101 SET Osaved cs client
- @@character_set_client
/*!50503 SET character set client = utf8mb4 */;
CREATE TABLE 'barriers' (
'id' bigint unsigned NOT NULL AUTO_ INCREMENT,
'name' varchar (255) COLLATE utf8m4_unicode_ci NOT NULL,
'type' int DEFAULT NULL,
"has _pedestrians' tinyint (1) NOT NULL DEFAULT '0',
'position top' double NOT NULL,
'position left' double NOT NULL,
'created at timestamp NULL DEFAULT NULL,
'updated at' timestamp NULL DEFAULT NULL, PRIMARY KEY id')
) ENGINE InnoDB AUTO INCREMENT 54 DEFAULT CHARSET utf8mb4 COLLATE utf8mb4
unicode ci;
/*!40101 SET character set client
• @saved es client */;
```

Фрагмент лістингу **tokens_table.php**

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
   /**
    * Run the migrations.
    *
```

```
     * @return void
     */
    public function up()
    {
        Schema::create('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamp('expires_at')->nullable();
            $table->timestamps();
        });
    }
```

Фрагмент лістингу **add_telegram_users_column.php**

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('telegram_users', function (Blueprint $table) {
            $table->boolean('is_premium')->default(false)->after('language_code');
        });
    }
```

Фрагмент лістингу **bot_blocked_column_to_telegram_users_table.php**

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('telegram_users', function (Blueprint $table) {
            $table->boolean('is_bot_blocked')->default(false)->after('is_premium');
        });
```

```
    }
```

Фрагмент лістингу **create_donate_transactions_table.php**

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('donate_transactions', function (Blueprint $table) {
            $table->id();
            $table->foreignId('donate_id')->constrained()->onDelete('cascade');
            $table->string('first_name');
            $table->string('last_name');
            $table->integer('amount');
            $table->string('comment')->nullable();
            $table->timestamps();
        });
    }
```

Фрагмент лістингу **create_donates_table.php**

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('donates', function (Blueprint $table) {
            $table->id();
            $table->string('link');
            $table->integer('target_price')->default(0);
            $table->text('description')->nullable();
            $table->timestamps();
        });
    }
```

Додаток Г. Діаграми баз даних

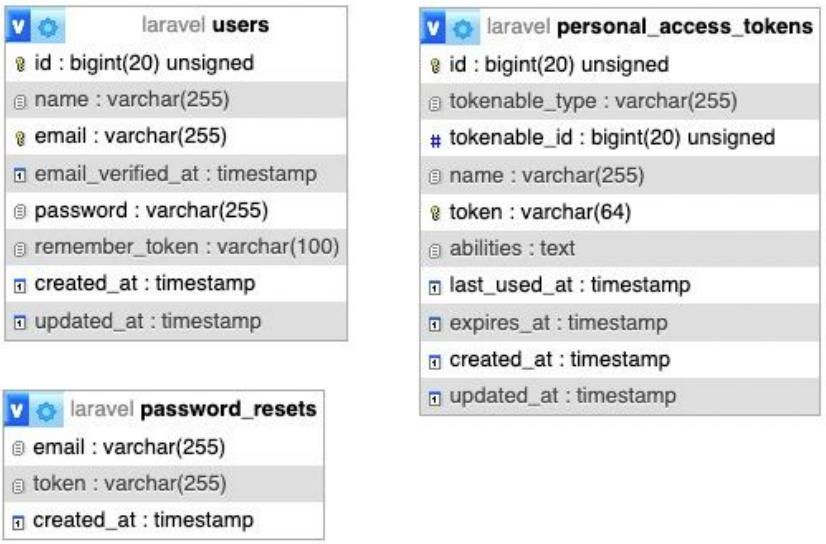


Рисунок Г.1 – БД для роботи фреймворка

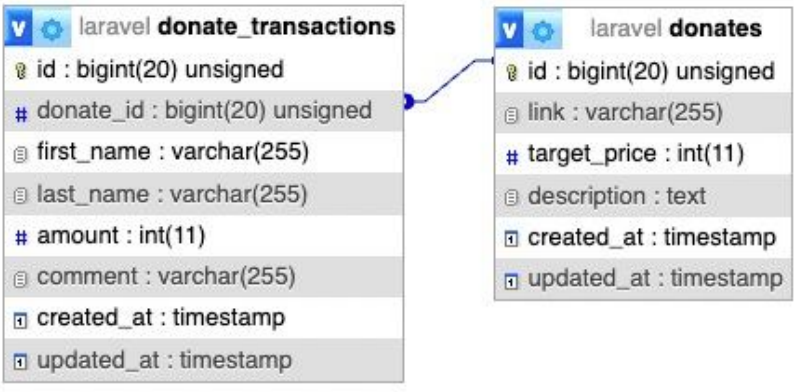


Рисунок Г.2 – БД для донатів