МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

ФІРКОВСЬКИЙ Дмитро Олегович

База знань чат-бота кафедри комп'ютерної інженерії / Knowledge base of the chatbot of the Department of Computer Engineering

спеціальність: 123 – комп'ютерної інженерії
освітньо-професійна програма – Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи Кім-21
Д.О. Фірковиський
_____

Науковий керівник:
О.Й. Піцун
_____

Кваліфікаційну роботу
допущено до захисту
«__»_____ 20__ р.

Завідувач кафедри
_____ Л.О. Дубчак

ТЕРНОПІЛЬ - 2022

ЗМІСТ

ВСТУП

Об'єкти, процеси та інші сутності матеріального світу в значній мірі відрізняються за своїми властивостями, проте їх об’єднує характеристика – необхідність впливу на пристрої реєстрації з метою прояву властивих їм параметрів. Кожна сутність повинна мати енергію, достатню для прояву властивостей в ході впливу на пристрій реєстрації. Описати математично перетворення енергії через процеси впливу дозволяє поняття сигналу. Сигнал – це формалізоване представлення сутності матеріального фізичного світу, яке містить інформацію про стан і властивості деякого об'єкта або процесу та з математичної точки зору є функцією однієї або декількох змінних.

Опрацювання сигналу дозволяє перевести цю інформацію у форму, зручну для подальшої комп'ютерної обробки. Трактування сигналу, як відображення у вигляді функції дозволяє розглядати сигнал як математичну модель. Сигнал, як модель, можна представити у математичній формі як вираз (формула), в тимчасовій (або просторовій) області – графік від змінної часу (координат простору), в частотній області – графік від значень частоти.

Комп'ютерне відтворення – результат реєстрації сигналів і їх представлення у цифровій формі, що дозволяє змінювати і відтворювати сигнали із параметрами, що задовільняють поставленим вимогам. Об'єкт комп'ютерної обробки може мати форму звуку, графіки або відео.

Отже, під сигналом розуміють фізичний процес, що несе повідомлення про будь-яку подію або стан об'єкта і протікає в просторі і в часі та охоплює певний спектральний діапазон.

Саме звукові сигнали дозволяють виявити різноманітні характеристики, які притаманні тому чи іншому об’єкту, тому їх часто називають інформаційними. Інформативні ознаки сигналу виявляють в процесі опрацювання сигналів із застосуванням різних програмних засобів й по них приймають рішення щодо стану об’єкта.

В останні роки можна спостерігати тенденцію того, наскільки глибоко сучасні технології вкоренилися в суспільстві. У цей час дуже важко знайти людину, яка б не користувалася гаджетами та Інтернетом у повсякденному житті. Наприклад, кожен день майже кожен володіє смартфоном, комп’ютером або багатьма іншими пристроями, які мають можливість підключення до Інтернету.

Найчастіше за допомогою цих пристроїв люди обмінюються інформацією, використовуючи для цього служби зв'язку. Також невід'ємною частиною нашого життя залишаються додатки та комп'ютерні програми, але з недавнього часу стрімко набирають популярність чат-боти. Вся справа в простоті використання, швидкому розвитку і можливості інтеграції бота в різні сервіси, наприклад, месенджери.

Чат-бот — це віртуальний помічник, який, залежно від призначення, створений для спілкування з користувачем за допомогою повідомлень.

Інструменти на основі штучного інтелекту у формі чат-ботів (їх ще називають віртуальними помічниками) здаються рішенням, яке заслуговує на інтерес. У цій статті представлено процес розробки віртуального асистента для кафедри комп’ютерної інженерії. Мета чат-бота полягає в тому, щоб зібрати в одному місці, а потім надати спеціалізовані ресурси знань окремому користувачеві вчасно.

Для впровадження цього чат-бота буде реалізовано три основні підпроекти:

- Рівень даних;
- Сервісний рівень;
- Інтерфейс користувача.

На рівні «Рівень даних» буде розміщена база даних, яка містить таблиці, які містять всю інформацію про користувачів та про університет. І розроблені основні функції для роботи з базою даних (додати, оновити, видалити, отримати).

На рівні «Services Layer» будуть розміщені основні сервіси (авторизація, реєстрація, статистика використання та наповнення бази даних) для роботи з базою даних і буде шифрування даних для захисту інформації.

Усі наші інтерфейси будуть розміщені на рівні "Бот", наприклад:

- Сайти;
- Мобільні додатки;
- Чат-боти для різних соціальних мереж.

Платформа Azure розроблена, щоб допомогти компаніям долати труднощі та досягати бізнес-цілей. Він пропонує інструменти, які підтримують усі галузі, включаючи електронну комерцію,фінансів і різноманітних компаній зі списку Fortune 500 і сумісний із технологіями з відкритим кодом. Це дає користувачам гнучкість у використанні своїх бажаних інструментів і технологій. Крім того, Azure пропонує чотири різні форми хмарних обчислень: інфраструктура як послуга (IaaS), платформа як послуга (PaaS), програмне забезпечення як послуга (SaaS) і безсерверне [24, 25, 38]..

Для зберігання нашої інформації ми будемо використовувати базу даних Azure, якщо порівнювати її з аналогічними сервісами (Amazon Web Service (AWS), Oracle Database), вона набагато дешевша і забезпечує кращу швидкість харчування з базою даних і витримує набагато більше навантаження на арі. ми також зберігатимемо наші сервіси, які використовуються для підключення інтерфейсів користувача до бази даних у їхньому хмарному сховищі.

Мета та завдання дослідження. Зважаючи на вищезазначене, метою кваліфікаційної роботи є розробка чат боту.

Для досягнення мети роботи слід виконати такі завдання:

- провести класифікацію чат-ботів;
- провести порівняльний аналіз серверних платформ та месенджерів;
- дослідити засоби хмарних сховищ;
- визначити основні вимоги до чат-боту;
- охарактеризувати алгоритми парсингу даних з файлу з розкладом у базу даних;
- проаналізувати і побудувати архітектуру проекту та схематичну базу даних;
- розробити багаторівневий проект чат-боту.

Об’єкт дослідження. Процеси розробки чат-боту.

Предмет дослідження. Багаторівневий проект чат-боту.

Методи дослідження. Методи аналізу і синтезу. Методи класифікації для розробки моделей для побудови алгоритму та схеми бази даних для реалізації чат-боту.

Наукова новизна одержаних результатів.

1. Розроблено алгоритм для реалізації чат-боту.

2. Удосконалено алгоритм для побудови багаторівневого проекту.

Практичне значення отриманих результатів. На основі розроблених алгоритмів із використанням мови C# розроблено багаторівневий проект чат-боту.

У першому розділі проаналізовано основні характеристики чат-ботів та їх класифікацію. Здійснено порівняння різних соціальних мереж у яких є можливість створювати чат бот. Проаналізовано різні хмарні сховища для зберегіння інформації.

У другому розділ розроблено основі функціональні вимоги до чат-боту. На основі функціональних вимог було розроблено алгоритми парсингу даних з файлу з розкладом у базу даних. Крім цього було проведено аналіз баз даних, та схематично побудовано базу даних, для чат-боту. Проаналізовано і побудовано архітектуру проекту. Досліджено основі бібілотеки і засоби для розробки чат-боту.

У третьому розділі описано програмне середовище для розробки чат-боту. Розроблено й описано діаграму варіантів використання програмної системи. Розроблено багаторівневий проект чат боту.

У додатках представлено копії публікацій результатів кваліфікаційної роботи.

1 АНАЛІЗ ЧАТ-БОТІВ ТА ХМАРНИХ СХОВИЩ

1.1 Класифікація чат-ботів

Система чат-ботів використовує технологію розмовного штучного інтелекту щоб імітувати обговорення (або чат) з користувачем природною мовою за допомогою програм обміну повідомленнями, веб-сайтів, мобільних програм або телефону. Він використовує мовні програми на основі правил для виконання функцій живого чату у відповідь на взаємодії користувача в реальному часі.

Для початку перед класифікацією чат-ботів проведемо аналіз чат-ботів, визначимо їх переваги і чому з часом вони набувають все більшої популярності.

Основні переваги:

- чат-боти надають потрібну інформацію там, де користувач спілкується з друзями, при цьому йому не потрібно перемикатися між різними додатками і чекати, поки з'явиться можливість сісти за комп'ютер або ноутбук;

- на відміну від сайтів, які більше орієнтовані на спілкування натисканням клавіш на клавіатурі, чат-боти орієнтовані на набір тексту пальцями на екранах смартфонів;

- повідомлення від чат-бота легко читаються, на відміну від екрану смартфона з інформацією на сайтах, які часто просто не оптимізовані для мобільних пристроїв;

- якщо користувач запустив чат-бота хоча б один раз, чат-бот буде збережений в контактах месенджера, до якого ви можете повернутися в будь-який час;

- чат-бот дозволяє уникнути витрат на утримання сервісів електронної пошти та SMS-повідомлень, так як в нього вже вбудована функція розсилки;

- доступність розсилок в ботах становить 80-95%, там, де доступні сервіси електронної пошти, розсилки, як правило, можуть "похвалитися" не більше ніж 20% електронних листів відкриті;

- створення чат-бота відбувається швидше і дешевше, ніж створення чи окремого додатку;

- чат-ботом можна користуватись на всіх платформах;

- чат-бот не вимагає віртуальної пам'яті(на відміну від додатка, його не потрібно завантажувати, оновлювати і встановлювати на пристрій);

- чат-бот, на відміну від веб-сайтів і мобільних додатків, може розподіляти своїх користувачів по групах в залежності від їх дій в боті і налаштовувати відправку адрес для кожної групи індивідуально;

- чат-боти дозволяють вам знизити вартість змісту онлайн-розмов або прискорити обслуговування клієнтів, що підвищує вашу лояльність.

- чат-боти проводять опитування для визначення потреб клієнта, безінтеграції з додатковими сервісами. Власник чат-бота може вносити пропозиції кожному клієнту індивідуально, враховуючи його переваги, або створити нову пропозицію, яка буде цікава більшості користувачів, або поліпшити роботу за допомогою зворотного зв'язку;

- чат-бот легко поширюється, так як користувач чат-бота може поділитися ним з будь-якою людиною в своїй записній книжці одним рухом пальця.

Чати мають два найпопулярніших варіанти комерційний і персональний.

Персональні чат-боти використовуються як персональні помічники користувачів, надсилати SMS, керувати календарем, для пошуку відео та аудіо, приймати дзвінки,

Комерційні чат-боти розроблені для компаній і призначені для залучення клієнтів. Підтримка бізнес-процесів навколо діалогу, маркетингу, продажів і маркетингу [14, 15, 38, 49]..

Є три основних інтерфейси для користувача, щоб використовувати чат бот це: кнопковий, розмовний і текстовий.

Кнопковий інтерфейс складається лише з кнопок. Даним методом користувач спілкується з чат ботом лише за допомогою команд. Користувач вибирає команди, які його цікавлять таким чином отримуючи інформацію, яка його цікавить.

Текстовий інтерфейс маю багато спільного з реальним спілкуванням. Користувач спілкується з чат-ботом за допомогою слів. Даний метод дає набагато більші можливості для користувача, але не в кожному проекті він потрібний, також цей інтерфейс є важчим для реалізації.

Розмовний інтерфейс це той самий текстовий, але текст вводиться не з клавіатури, а з мікрофону. Чат-бот отримує голосове повідомлення перетворює його в текстове, тоді чат-бот повертає для користувача інформацію в залежності від його запиту.

Ще чат-боти поділяються за принципом роботи: шаблон з штучним інтелектом і без. Чат-бот без штучного інтелекту формується за принципом дерева, де кожне рішення приводе нас до наперід відомого результату. Діалоги зазвичай структуровані і послідовні. Чат-боти на основі штучного інтелекту, вони навчаються під час діалогу. Дані чат-боти навчаються комунікативним навичкам під час діалогу.

Є три основні форми доступу: групи, підписки та вбудовані. Груповий доступ обслуговує кількох користувачів одночасно. Чат-бот за підпискою призначений лише для користувачів з підпсикою на даний чат-бот. Вбудованих ботів можна викликати в будь-якому діалоговому вікні спеціальна команда. Такі чат-боти виконують конкретні запити та отримують миттєві результати передається співрозмовнику.

У таблиці 1.1 показано основні атрибути чат-боту.

Таблиця 1.1 – основні атрибути чат-боту

Атрибут	Властивості
Принцип роботи	З штучним інтелектом Без штучного інтелекту
Інтерфейс використання	Розмовний, Текстовий, Кнопочний
Власник	Комерційний Персональний
Форми доступу	Доданий адміністратором За підпискою Вбудований

Чат-бот працює з сервером за принципом клієнт-сервер(рисунок 1.1). Цей принцип побудований на тому, що є один сервер і до нього може бути підключено багато різних клієнтів, у даному випадку чат-бот, але також його можна використовувати для сайті, мобільних додатків, тощо [1, 44, 48]..

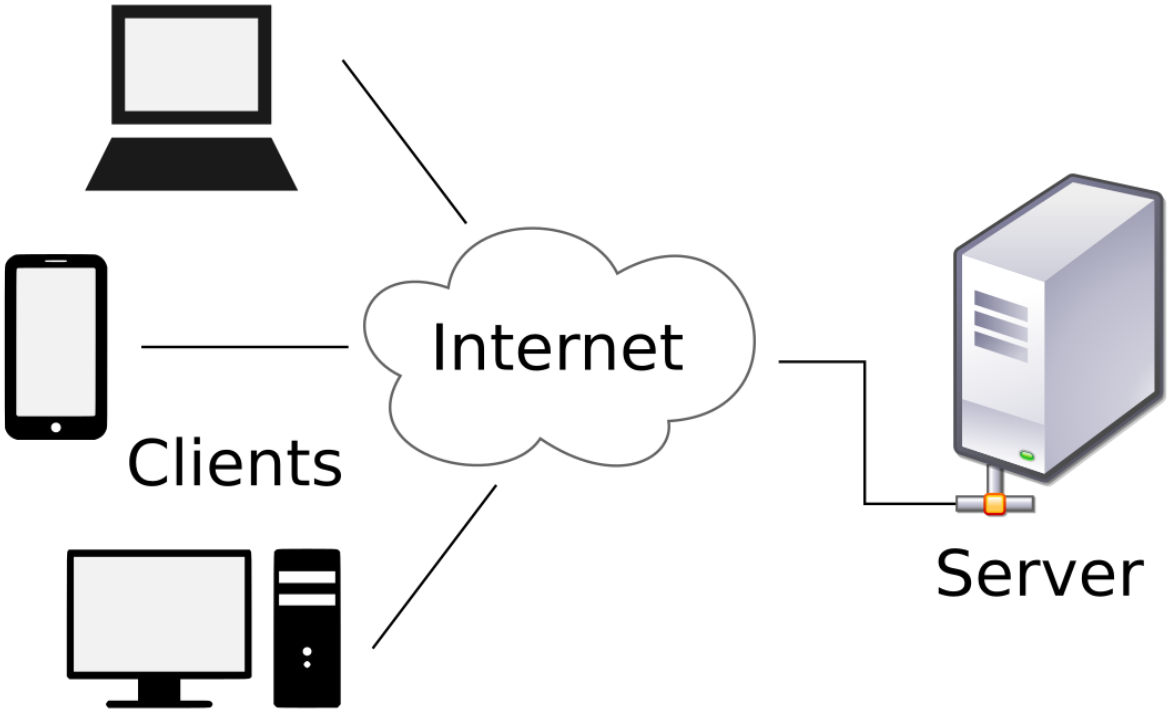

Рисунок 1.1 – Клієнт-сервер

Більш детально як працює сервер з клієнтом зображено на рисунку 1.2. Для створення чат-боту можуть використовуватись різні мови програмування, такі як: C#, Python, Php, Node.JS. Це мови які отримали славу у серверній розробці.

Рисунок 1.2 – Принцип роботи чат-боту

Back end – це серверна сторона додатку, там відбувається вся логіка програми. Він отримує запит від front end і повертає результат містить зв’язок із базою даних.

Back end означає будь-яку частину веб- сайту або програми , яку користувачі не бачать. Він контрастує з інтерфейсом , який стосується інтерфейсу користувача програми або веб-сайту . У термінології програмування серверна частина — це «рівень доступу до даних», а зовнішня частина — це «рівень представлення».

Більшість сучасних веб-сайтів є динамічними, тобто вміст веб-сторінки створюється на льоту. Динамічна сторінка містить один або кілька сценаріїв , які запускаються на веб-сервері під час кожного доступу до сторінки. Ці сценарії генерують вміст сторінки, який надсилається до веб-переглядача користувача. Усе, що відбувається до того, як сторінка відобразиться у веб-браузері, є частиною серверної частини.

Приклади внутрішніх процесів:

- обробка вхідного запиту веб-сторінки;
- запуск сценарію (PHP , ASP , JSP тощо) для створення HTML;
- доступ до даних, таких як статті, з бази даних за допомогою запитів SQL;
- зберігання або оновлення записів у базі даних;
- шифрування та дешифрування даних;
- обробка завантажень і завантажень файлів;
- обробка введених користувачем даних через JavaScript.

Фронтальна веб-розробка, також відома як розробка на стороні клієнта, — це практика створення HTML, CSS і JavaScript для веб-сайту або веб-програми, щоб користувач міг бачити їх і взаємодіяти з ними безпосередньо. Проблема, пов’язана з розробкою інтерфейсу, полягає в тому, що інструменти та методи, які використовуються для створення інтерфейсу веб-сайту, постійно змінюються, тому розробник повинен постійно бути в курсі того, як розвивається сфера.

Мета розробки сайту полягає в тому, щоб переконатися, що коли користувачі відкривають сайт, вони бачать інформацію в зручному для читання форматі та актуальному. Це ще більше ускладнюється тим фактом, що зараз користувачі використовують велику різноманітність пристроїв із різними розмірами екрану та роздільною здатністю, що змушує дизайнера враховувати ці аспекти під час розробки сайту. Їм потрібно переконатися, що їхній сайт правильно відображається в різних браузерах (кроссбраузерність), різних операційних системах

(кроссплатформенність) і на різних пристроях (кросспристройність), що вимагає ретельного планування з боку розробника.

Є два типу зв’язку між front end і back end:

- Webhook;

- Polling.

За визначенням, Webhook (також званий зворотним веб-дзвінком або HTTP push API) — це спосіб, за допомогою якого програма надає іншим програмам інформацію в реальному часі. Webhook доставляє дані в інші додатки, коли це відбувається, тобто ви отримуєте дані миттєво, на відміну від типових API, де вам потрібно дуже часто опитувати дані, щоб отримати їх у реальному часі. Це робить Webhook набагато ефективнішими для постачальника та споживача. Єдиним недоліком Webhook є складність їх початкового налаштування.

Webhook іноді називають зворотними API через можливість надати вам те, що становить специфікацію API, і ви повинні розробити API для використання веб-хуком. Webhook надсилає HTTP-запит до вашої програми (зазвичай POST), а потім вам доведеться його інтерпретувати.

Webhook – коли front end відправляє Http Post запит на back end про зміни у ньому. Тобто, front end сповіщає сервер про настання подій у ньому. Дані відправляються у форматі JSON або XML. Даний метод повинен містити певну автентифікацію облікого запису її можна здійснити один із наступних способів:

- Надати для серверу перелік ip-адрес користувачів, які маю доступ до неї, а ip-адреси які не входять у той список заборонити доступ;

- Встановлювати з’єднання з використання протолоколу TLS, задопомогою нього можна передавати дані по шифрованому каналі зв’язку.

- З допомогою SSL-сертифікатів можна отримати взаємну автентифікацію клієнта і серверу.

На рисунку 1.3 зображено схематичне зображення принципу Webhook.

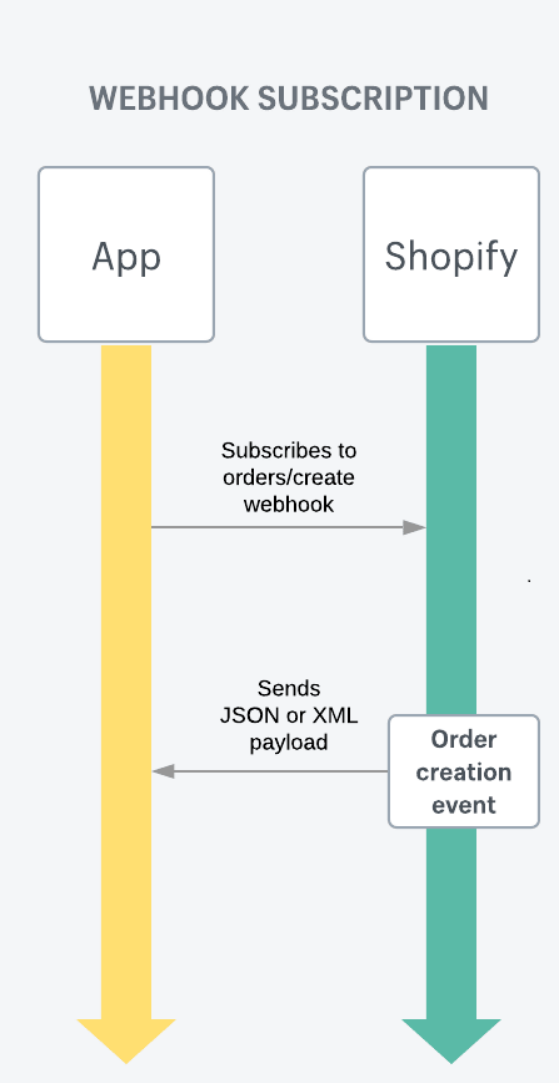

Рисунок 1.3 – Схематичне зображення Webhook

Polling – базується на тому, що сервер відправляє запит на сторону клієнта і отримує відповідь про зміни. Автентифіканція відбувається з використанням унікального токену для чат-боту. Якщо змін у чат-боті не було то на сервер приходить ні, якщо зміни були то відправляються зміни. Цей спосіб є дуже енергозатратний і потребує значних ресурсів серверу для постійної перевірки чи були, якісь обновлення у клієнта.

На рисунку 1.4 показано схематичне використання Polling.

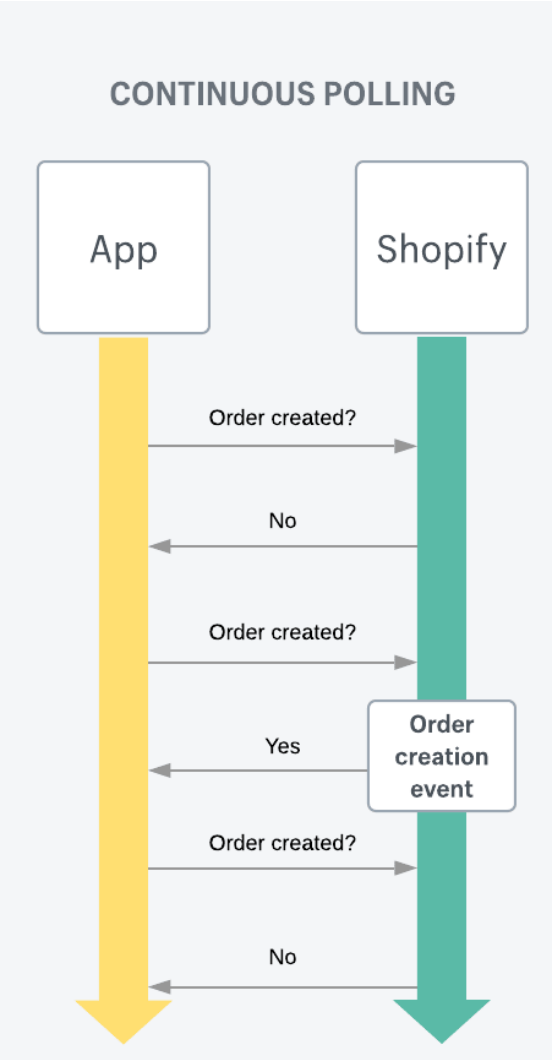

Рисунок 1.4 – Схематичне зображення polling

У даному розділі було проаналізовано основні атрибути чат-ботів, типи зв’язку у різних чат ботах такі, як webhook і polling. Було проаналізовано принцип роботи чат-боту.

1.2 Порівняльний аналіз платформ серверних платформ

У даному розділі буде розглянуто і проаналізовано два основних сервіси для роботи з серверними платформами такі, як:

- ASP.NET;
- PHP.

Для аналізу було обрано саме цих по причині їх високої популярності у розробці серверних додатків. З однієї сторони PHP це мова програмування, на яку покладається велика кількість програмістів, з іншого, ASP.NET має такий бренд, як Microsoft.

PHP це мова програмування з відкритим кодом, яка використовується для веб-розробки та може бути вбудована в HTML. З іншого боку, PHP також підходить для професійних програмістів, оскільки він має всі розширені функції.

Розглянемо переваги PHP, одна з основних переваг це його швидкий розвиток. Ринок вимагає швидшого процесу розробки додатків, і PHP пропонує компаніям, які займаються розробкою мобільних додатків, багатофункціональну перевагу. Цей фреймворк не тільки швидкий, але й забезпечує безпеку додатків. PHP простий в обслуговуванні та має величезну спільноту розробників, які можуть допомогти за потреби.

Основна проблема PHP це низька швидкість додатка, та великий час для опанування цього фреймворку.

ASP.NET це серверний інструмент веб-розробки також із вихідним кодом. Із використанням цього фреймворку можна створювати:

- веб-сторінки;
- веб-додатки;
- веб-сервери.

Переваги ASP.NET це є швидкісні додатки, кросплатформлені додатки, простота добавляння і видалення функцій, відкритий вихідний код.

Одним із головним недоліків є ASP.NET це прогалини в документації, це є відносно не велика спільнота.

Порівняємо цих два фреймворки ASP.NET і PHP. Один із головних критеріїв для порівняння це швидкодія. Продуктивність будь-якого фреймворку залежить від того, як написаний код. Продуктивність ASP.NET набагато вища ніж у PHP по причині того, що ASP.NET підтримує паралельне програмування, у той час коли у PHP його нема.

Наступним дуже важливим фактор для порівняння є безпека. Обидва фреймворки містять у собі інструменти для безпеки, але ASP.NET має вбудовані

функції для неї, у той час як PHP лише інструменти для цього. Тому в плані защити ASP.NET набагато кращий, бо більшість користувачів нехтує інструментами защити PHP, по цій причині є багато вразливих програм. На рисунку 1.5 зображено порівняння защити двох фреймворків.

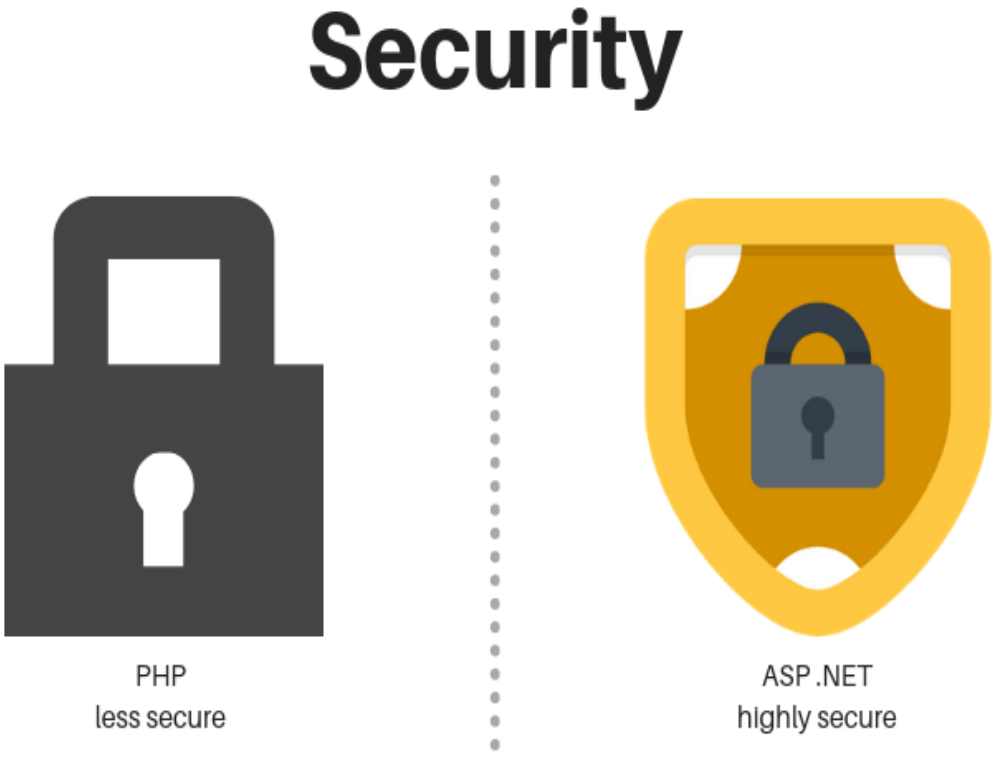

Рисунок 1.5 – Защита PHP і ASP.NET

Наступним фактором для порівняння буде ціна. Тут очевидним переможцем є PHP він є повністю безкоштовним, тоді як ASP.NET виставляє не велику ціну за розміщення, комісію що стягує Microsoft є дуже мала, але якщо порівняти її з безкоштовним є дуже не приємна.

Ще ASP.NET має такі переваги над PHP, як:

- свобода вибору мови (c#, Visual Basic.Net, C++ тощо);
- потокове виконання коду;
- зарплати набагато вищі, по причині того що ASP.NET розробників набагато менше.

У даному розділі було порівняно різні фреймворки для розробки, по таких критеріях як: ціна, швидкодія і безпека [1, 25, 26, 27, 28, 35].

1.3 Порівняльний аналіз месенджерів

Програма, мобільний додаток або веб-сервіс, за допомогою якого можна передавати між великою кількістю користувачів миттєві повідомлення та інший контент називають месенджер. Месенджер може також мати додаткові сервіси, такі як: голосові та відео виклики, чат-боти, ігри і тд.

Для аналізу нам потрібно обрати декілька сучасних месенджерів з підтримкою платформи чат-ботів. Візьмемо наступні месенджери: Facebook, Telegram, Discord та Viber.

Параметрами аналізу є: основний функціонал (листування, опитування, передача файлів, магазин, покупки, ігри), форма використання, доступ, інтерфейси, механізми зв'язку, протоколи передачі даних, формати передачі даних.

У таблиці 1.2 можна побачити порівняльну інформацію популярних месенджерів з підтримкою платформи чат-ботів:

Таблиця 1.2 – порівняння найбільш популярних платформ

Назва	Telegram	Facebook	Discord	Viber
Ціна	Безкоштновна	Майже безкоштовна	Майже безкоштовна	Майже безкоштовна
Інтерфейси користувача	Текстовий, розмовний, кнопочний	Текстовий, розмовний, кнопочний	Текстовий розмовний	Текстовий, розмовний, кнопочний
Тип зв’язку	Webhook, polling	Webhook	Webhook, polling	Webhook
Протокол передачі даних	https	https	https	https
Тип даних запитів	JSON, XML,URl-	JSON, form-data	JSON, XML,URl-	JSON, form-data

	query		query	

Після проведеного аналізу ми дійшли висновку, що ми досягнемо цілей дипломної роботи, використавши саме месенджер Telegram, адже, в порівнянні з іншими месенджерами, він має такі переваги:

- Telegram має безкоштовну форму використання;

- має повний перелік функціоналу, що дає змогу р еалізувати будь-які системні вимоги і, при необхідності, покращити і розширити їх;

- забезпечує механізм зв’язку опитування (Polling, Webook), що значно спрощує розробку системи та її налагодження ще на ранніх етапах.

Чат-бот у контексті Telegram – це окремий додаток, що працює в екосистемі Telegram. Користувачі взаємодіють з ботами надсилаючи їм повідомлення, команди та inline-запити. Для управління Telegram-ботом використовується API, що вимагає використання HTTPS-протоколу.

За допомогою бота Telegram ви можете:

- отримувати персоналізовані повідомлення та новини;

- прийом платежів від користувачів;

- інтеграція зі іншими сервісами, такими як: Gmail, Wiki, YouTube, Github, тощо;

- створення утиліт: перекладачів різних мов, редакторів тексту, нагадування про події і так далі;

- розробка ігор на основі технології HTML5.

З точки зору месенджера, Telegram-бот - це спеціальний акаунт, з яким користувачі можуть зв'язатися двома способами:

- відкрийте чат з ботом або додайте його в групу, потім відправте текст або команди через поле введення повідомлення;

- відкрийте будь-який чат і введіть «@bot name» у полі введення повідомлення. Так вони працюють з inline-ботами.

Але варто пам'ятати, що тільки користувач може ініціювати спілкування з ботом, але не навпаки. Це перешкоджає спаму від інших ботів.

Крім цього, у ботів обмежене хмарне сховище для повідомлень. Створіть локальне сховище для попереднього архівування старих повідомлень, якщо вам потрібно.

Проаналізуємо Telegram Bot API. Кожен бот потребує авторизації з боку серверів Telegram. При створенні бота генерується персональний id-токен (далі – <token>) авторизації (приклад – “98765:DIMA-FIR1505kovskiy-afasfdsfWR13d324f4”), який передається частиною URL при кожному запиті. Усі запити до Telegram Bot API повинні відбуватися за допомогою HTTPS протоколу на адресу: “https://api.telegram.org/bot<token>/function_name”. Підтримуються GET та POST HTTP запити. Для передавання даних використовуються чотири основних принципи:

- Multipart/form-data.
- URL;
- Appliacation/json;
- Application/x-www-form-urlencoded;

1.4 Аналіз засобів хмарних сховищ

У теперішні часи спектр хмарних сховищ для зберігання систем керуванням базою даних (СУБД) не складається лише з кількох хмарних продуктів, ринок бази даних вибухнув десятками хмарних продуктів, які варіюються від спеціально створених платформ СУБД, розроблених відповідно до унікального набору вимог, до технологій загального призначення, які мають набагато ширшу сферу застосування.

Він також включає широкий спектр керованих хмарних служб баз даних, що надаються постачальниками СУБД, що складається з баз даних як послуг (DBaaS) і сховищ даних як послуг (DWaaS). Щоб вибрати відповідного постачальника для своєї організації, групи керування даними повинні виконати добре продумане

порівняння доступних варіантів хмарних баз даних. Порівняємо три найпопулярніші постачальники DBaaS і DWaaS це:

- AWS (Amazon Web Servise);
- Microsoft;
- Oracle.

Будем порівнювати за такими основними критеріями: ціна, асортимент сервісів для даного проекту, механізм міграцій баз даних;

Важливим пунктом у ціні є повне розуміння ціноутворення, за оплату розподілу і використанням ресурсів. Фактично відбувається орендація місця та сплачуєм плату за використання, яка значно розрізняється в залежності від постачальника.

AWS є набагато важчий для розуміння ціноутворення за решту своїх конкурентів. Але ця складність викликається через дуже велику кількість можливих сервісів, це дає можливість краще адаптувати сервер під їх вимоги. Ціна змінюється у залежності від таких факторів:

- Регіон;
- Конфігурації сервера чи його відсутності;
- Характеристик (ЦП, пам’яті);
- Швидкість введення/виведення інформації;
- Резеревне сховище;

За допомогою калькулятора “https://cloud.netapp.com/aws-calculator”, було вирахувано приблизну вартість зберігання даного проекту на серверах AWS з усіма сервісами вона становитиме 177.95 долара США, скріншот розрахунку вартості на рисунку 1.6.

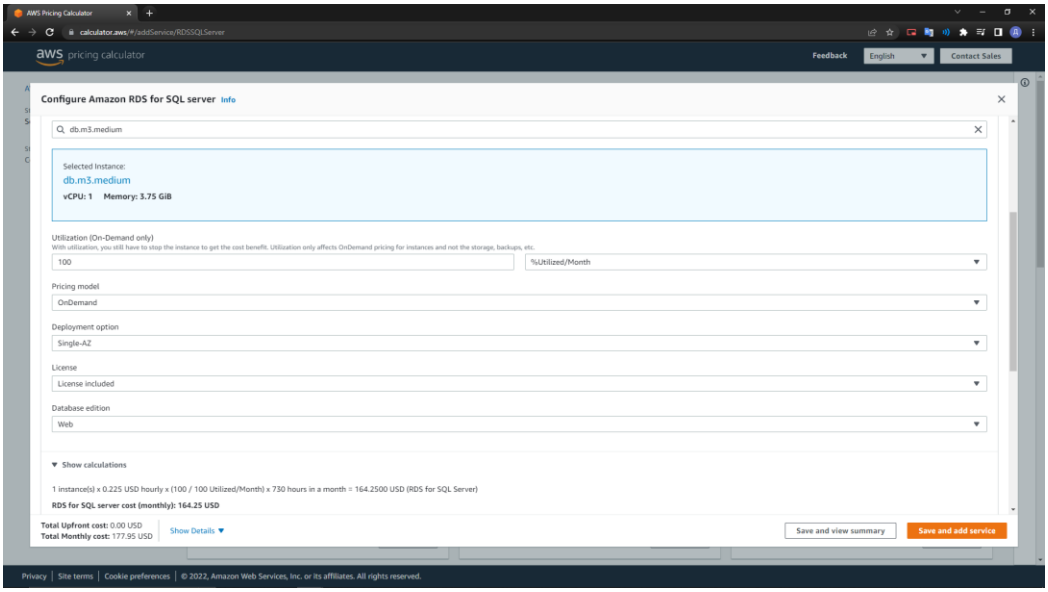

Рисунок 1.6 – скріншот вартості AWS

Microsoft не надає такий великий спектр можливостей для, як AWS. Microsoft пропонує два варіанти ціноутворення:

- Database Transaction Unit (DTU) ;
- Virtual Core (vCore).

Database Transaction Unit – це модель, що об’єднує цифрові ресурси, ресурси зберігання та вводу/виводу у комірки для об’єднаних навантажень.

Virtual Core – це модель, яка дає змогу вибрати між різними поколіннями апаратного забезпечення та окремо вибрати обчислення, пам’ять сховище та введення/виведення інформації, також дає змогу вибирати між ініціалізованим або безсерверним обчислювальним рівнем.

За допомогою калькулятора “https://azure.microsoft.com/ru-ru/pricing/calculator/”, було вирахувано приблизну вартість зберігання даного проекту на серверах Microsoft з усіма сервісами вона становитиме 9.67 доларів США, скріншот розрахунку вартості на рисунку 1.7.

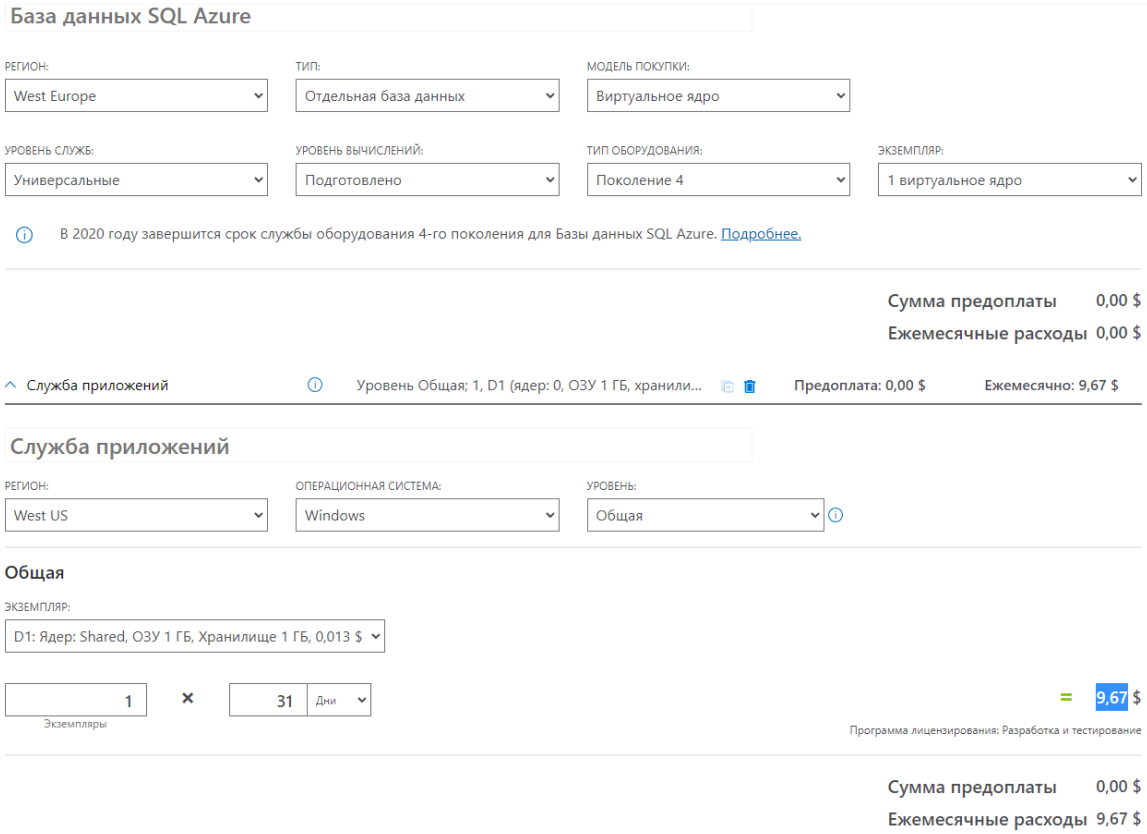

Рисунок 1.7 – скріншот вартості у Microsoft Azure

Oracle ставить по годину оплату на своїх користувачів базуючих на кількості виділених ЦП. Один процесор Oracle рівний двом віртуальним процес x86. Ціни на Oracle залежать від:

- Кількості читань і записів;
- Обсягу пам’яті.

За допомогою калькулятора “https://www.oracle.com/cloud /costestimator.html”, було вирахувано приблизну вартість зберігання даного проекту на серверах Oracle з усіма сервісами вона становитиме 46.37 США, скріншот розрахунку вартості на рисунку 1.8.

Другим пунктом для порівняння буде міграція локальної бази даних у глобальну.

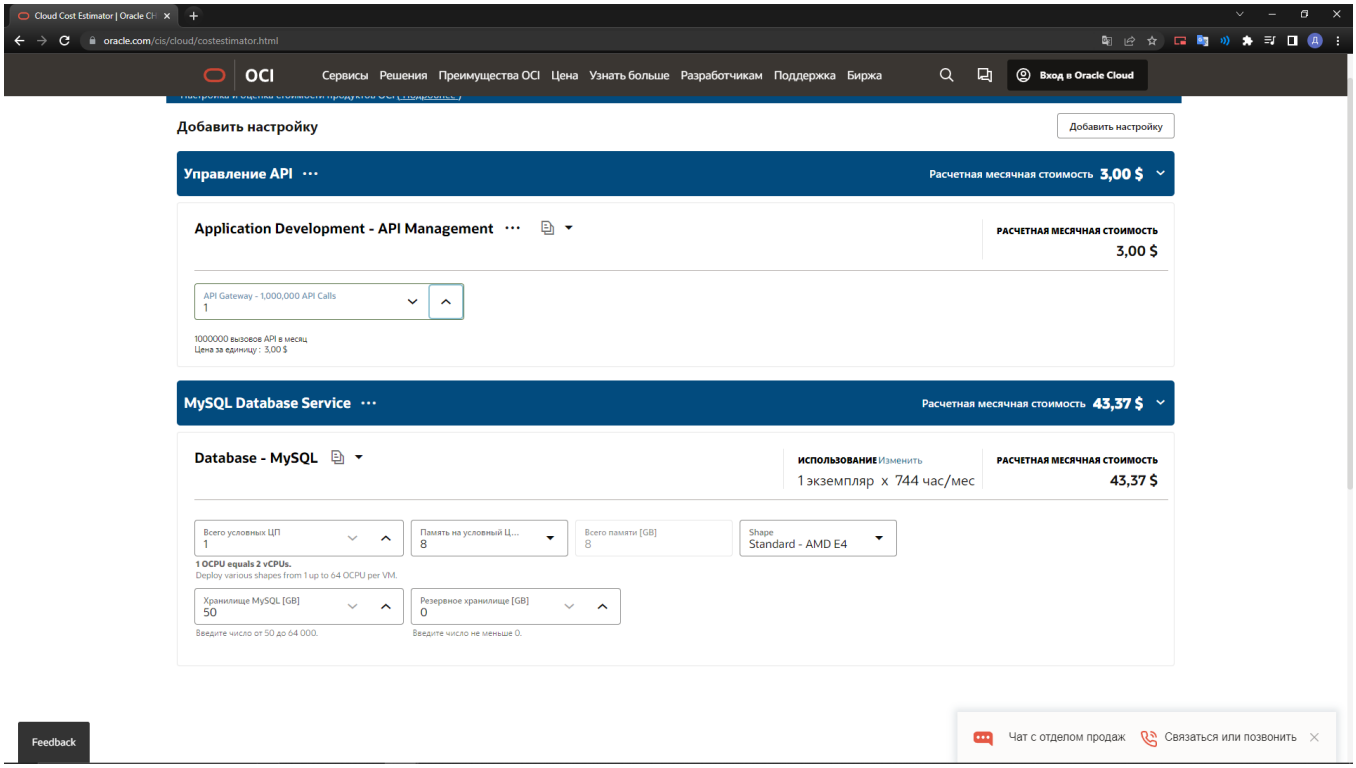

Рисунок 1.8 – Скріншот вартості у Oracle

AWS пропозиції міграції та конверсії від AWS домінують над конкурентами. AWS Database Migration Service — це комплексна утиліта реплікації, яка може використовуватися для початкового заповнення вашої платформи DBaaS даними, а потім підтримувати її в синхронізації з вихідною системою [32, 41, 45].

На додаток до реплікації даних між однорідними джерелами та цілями, служба дає змогу адміністраторам баз даних налаштовувати реплікацію між різними різнорідними платформами. Це дозволяє клієнтам перенести свої бази даних у хмару AWS і переключитися на іншу СУБД. Кількість вихідних і цільових систем, які підтримує служба, і її надійний набір функцій значно перевершують інші пропозиції міграції. Даний сервіс потребує додаткової оплати.

Microsoft служба міграції бази даних Microsoft Azure можна використовувати для міграції даних із SQL Server до бази даних SQL Azure, керованого екземпляра Azure SQL і SQL Server на віртуальних машинах Azure, а також із MongoDB до Azure Cosmos DB, MySQL до бази даних Azure для MySQL і PostgreSQL до Azure База даних для PostgreSQL.

Крім того, це дає змогу клієнтам переносити бази даних SQL Server, MySQL і PostgreSQL, що працюють на Amazon RDS, до своїх аналогів Azure SQL Database і Azure Database. Даний метод не сервіс додаткової оплати [10, 21, 39].

Oracle. Oracle підтримує численні методи міграції власних локальних баз даних до OCI, і тепер пропонує Oracle Cloud Infrastructure Database Migration, керовану службу для міграції баз даних Oracle, включно з тими, що працюють в Amazon RDS. Щоб допомогти клієнтам перенести бази даних, що не належать до Oracle, до OCI, він покладається насамперед на Oracle SQL Developer, інструмент, який включає майстер міграції для переміщення локальних баз даних, таких як SQL Server, IBM Db2, Sybase Adaptive Server і Teradata Database, до Oracle Autonomous Database . SQL Developer також надає окрему утиліту міграції для клієнтів, які хочуть перейти з Amazon Redshift на автономну базу даних. Даний метод не сервіс додаткової оплати [41, 43, 47].

Oracle, Microsft і AWS усі мають потрібні нам сервіси для роботи з базою даних і апі. Для Microsft це: Azure Sql Database (для бази даних), App Services (для апі). Для Oracle це: MySqL DataBase Service (для бази даних), Application Development – Api Maneger (для апі). Для AWS це: Api Restful (для апі),Amazon RDS for SQL Server (для бази даних).

Отримані дані запишем у таблиці 1.4.

Таблиця 1.4 – порівняння хмарних сховищ

	Microsft	Azure	Oracle
Ціна	177.95	9.67	46.37
Маппінг	Так,але за додаткову плату	Так	Так
Типи бд для мапінгу	Більшість можливих локальні міграції	Sql server, MySQL PostgreSQL	Oracle SQL Developer, IBM Db2
Сервіси	Наявні	Наявні	Наявні

Перед вибором хмарного сховища потрібно розглянути їх недоліки.

AWS завдяки широкому та глибокому набору хмарних служб СУБД середовища AWS можуть бути складними для адміністрування, а правильне надання нових систем є складним завданням. Деякі рецензенти також кажуть, що конкуренти, зокрема Google, Microsoft і Oracle, починають знижувати ціни на AWS. Крім того, AWS надає обмежені локальні пропозиції через AWS Outposts, що може бути проблемою для клієнтів, які мають значні інвестиції у власні центри обробки даних.

Oracle. Ліцензування та переговори щодо контрактів Oracle завжди були складними та надто напруженими для клієнтів, і ця проблема залишається актуальною для хмарних пропозицій, незважаючи на зручний інструмент оцінки вартості. Oracle також має найнижчу частку ринку хмарних систем порівняно з AWS, Microsoft і Google. Крім того, його гетерогенні параметри СУБД загалом відсутні, хоча Oracle зараз використовує більше багатохмарний підхід із власними службами баз даних.

Висновком даного підпукту є те, що для наших операцій підійде хмарне сховище від Azure.

1.5 Постановка задачі

Сьогодні можна спостерігати тенденцію того, наскільки глибоко сучасні технології вкоренилися в суспільстві. У цей час дуже важко знайти людину, яка б не користувалася гаджетами та Інтернетом у повсякденному житті. Наприклад, кожен день майже кожен володіє смартфоном, комп’ютером або багатьма іншими пристроями, які мають можливість підключення до Інтернету.

Найчастіше за допомогою цих пристроїв люди обмінюються інформацією, використовуючи для цього служби зв'язку. Також невід'ємною частиною нашого життя залишаються додатки та комп'ютерні програми, але з недавнього часу стрімко набирають популярність чат-боти. Вся справа в простоті використання, швидкому розвитку і можливості інтеграції бота в різні сервіси, наприклад, месенджери.

Чат-бот — це віртуальний помічник, який, залежно від призначення, створений для спілкування з користувачем за допомогою повідомлень.

Зважаючи на це, у кваліфікаційній роботі слід виконати такі завдання:

- провести класифікацію чат-ботів;
- охарактеризувати порівняльний аналіз платформ серверних платформ;
- проаналізувати види месенджерів;
- проаналізувати засоби хмарних сховищ;
- розробити основі функціональні вимоги до чат-боту;
- провести аналіз алгоритмів парсингу даних з файлу з розкладом у базу даних;
- побудувати схему БД
- розробити і реалізувати архітектуру програмного засобу.

1.6 Висновки до розділу

Проаналізовано основні характеристики чат-ботів та їх класифікацію. Було проаналізовано описаний механізм роботи клієнт сервер та front end, back end. Здійснено порівняння різних соціальних мереж у яких є можливість створювати чат бот. Проаналізовано різні хмарні сховища для зберегіння інформації.

У результаті виявленно, що існують такі типи чат-ботів, як: зі штучним інтелектом і без. Було проаналізовано, порівняно і виявлення усіх переваг і недоліків різних типів з’єднання, а саме polling і webhook.

Під час порівняння різних хмарних середовищ було встановлено, що для наших цілей буде використано Microsoft Azure, його ціна є набагато нижчою ніж у аналогів, а ще є можливість мапінгу з використанням SQL server.

Також під час аналізування різних соціальних мереж було обрано меседжер Telegram для наших цілей. Даний месенджер є безкоштовним, у наявності є документоване API, а розроблені чат-боти можна використовувати з polling.

2 АНАЛІЗ ТА ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ КЕРУВАННЯ ЧАТ-БОТОМ

2.1 Основі функціональні вимоги до чат-боту

Інформаційна система — це організаційно-технічна система, в якій реалізована технологія обробки інформації за допомогою технічних засобів і програмного забезпечення.

Підприємства, корпорації та інші організації користуються інформаційними системи для здійснення та управління операціями, спілкування з клієнтами та постачальниками, а також для конкуренції на ринку. Інформаційні системи існують для керування між організаційними ланцюгами постачання і електронними ринками. Для прикладу: корпорації використовують інформаційні системи для опрацювання фінансових рахунків, управління персоналом та заохочення потенційних клієнтів через онлайн-рекламу.

Більшість великих корпорацій цілком побудовані на основі інформаційних систем. Наприклад, eBay, який є в основному аукціонним ринком. Amazon - зростаючий центр електронної комерції та постачальник послуг хмарних обчислень. Google – пошукова компанія, яка переважно отримує свій дохід від рекламних ключових слів для пошуку в Інтернеті. Alibaba – це електронний ринок для компаній.

Крім того, уряди впроваджують інформаційні системи для надання економічно ефективних послуг громадянам країни. Інформаційні системи супроводжують цифрові товари, наприклад, електронні книги, відеопродукція і програмне забезпечення, а також ігри та соціальні мережі. Ми користуємося інформаційними системами на основі Інтернету протягом більшої частини нашого часу, включаючи спілкування в соцмережах, навчання, покупки, банківські справи та розваги.

У всіх інформаційних системах вирішуються такі задачі:

- передача даних;
- аналіз і обробка даних, прийняття рішень;

- зберігання даних.

Залежно від поставленої задачі кожна інформаційна система вимагає певного способу реалізації методів і їх вирішення. Інакше кажучи, основа будь-якої інформаційної системи складається з:

- засобів фіксування і збору інформації;

- носіїв інформації;

- засобів передавання даних і повідомлень серед складників інформаційної системи.

- засоби аналізу, обробки та представлення інформації.

Крім того, із точки зору розробки програмного забезпечення слід:

1. Створити специфікацію програмного забезпечення.

2. Зробити проектування ПЗ;

3. Провести програмування та тестування;

4. Провести розгортання ПЗ.

Процес створення специфікації програмного забезпечення є процесом, який забезпечує аналіз вимог і документований опис поведінки системи, враховуючи усі вимоги, технічні можливості та обмеження. У контексті дипломної роботи потрібно створити список функціональних вимог до системи керування чат-ботом і серверу:

1. Можливість переглядати користувачів за допомогою сервера;

2. Можливість додавати нові пари(в ручну або парсингом файлу);

3. Можливість перегляди пари по днях, по групах, по вчителях;

4. Можливість надсилання текстових та медіа повідомлень не одній людині, а цілим групам користувачів за допомогою сервісу розсилки, а також планування розсилки повідомлень (сповіщення про уроки, можливість розсилки даних факультету для всіх користувачів);

5. Пошук і відображення корпусів на карті.

У даному розділи було визначено основі функції, які повинні бути реалізовані у дипломній роботі [2, 8, 9, 13].

2.2 Алгоритми парсингу даних з файлу з розкладом у базу даних

Розклад для користувачів подається у форматі xml перед додавання його потрібно спарсити у потрібний нам формат.

Алгоритм синтаксичного аналізу документа розбиває документ на його найбільші складові, як правило, речення та пропозиції. Початковим кроком зазвичай є перетворення речень вихідного тексту в основний формат, який називається Sentence Graph. Розбір документа також включає токенізацію. Де вихідні речення розбиваються на основи слів і розділові знаки. У цьому блозі ми пояснюємо 7 найкращих алгоритмів аналізу документів.

Алгоритм CYK — це алгоритм для розбору контекстно-вільних граматик (CFG). Використовується для визначення того, чи належить даний рядок CFG до даної мови. CFG описує мову, а алгоритм перевіряє, чи задовольняє рядок S умовам, визначеним у CFG.

Він знаходить N найбільш вірогідних контекстно-вільних граматик для набору речень S. Покладається на принцип, що малоймовірно, що більше однієї відносно короткої граматики. Воно буде відповідати заданому реченню. Зокрема, алгоритм CYK використовує тест динамічного програмування, щоб передбачити, чи є рядок мовою граматики чи ні.

Алгоритм Ерлі - це низхідний парсер, який працює з контекстно-вільними граматиками. Він розроблений для вирішення практичної проблеми аналізу, відомої як проблема зсуву-зменшення. Для порівняння, він зберігає простоту та ефективність парсерів LL.

Це низхідний алгоритм для генерації ліворуч ациклічного дерева синтаксичного аналізу складових із формального опису речення. Це один із складних і ефективних алгоритмів аналізу. Виробляється на сьогоднішній день і успішно застосовується для таких завдань, як. Ідентифікація речень, класифікація текстів, машинний переклад і статистичне тегування частин мови.

Загалом він використовував діаграму для синтаксичного аналізу та реалізував як динамічну програму, що покладається на вирішення простіших підзадач.

Подібним чином мета алгоритму полягає в тому, щоб вирішити, чи дана граматика генерує даний текст.

Алгоритм синтаксичного аналізу LL (LL розшифровується як Left-to-right, Leftmost derivation) є найпростішим з усіх парсерів. Його зрозуміло реалізувати порівняно з іншими алгоритмами аналізу. Цей метод призначений для створення аналізатора для певної мови. Синтаксичний аналізатор використовує набір рукописних правил для розпізнавання різних токенів у даній мові програмування.

Алгоритм аналізу LR — це висхідний алгоритм аналізу, який став практичним, простим і ефективним. З моменту появи комп'ютерних мов. Розбір LR залишається стандартним алгоритмом у сучасних мовах програмування. Особливо для синтаксичних аналізаторів, які реалізовані як компоненти компілятора або використовуються з мовами програмування загального призначення.

Це метод для створення аналізаторів лінійного часу для граматик мови синтаксичного аналізу зверху вниз. Він зберігає весь набір виробничих правил і використовує їх для генерації крайнього лівого похідного. Замість стандартного алгоритму LR(k), який створює таблицю аналізу LALR. У результаті він знаходить найкращий розбір даного речення.

Це реалізується на практиці як частина ширшого робочого столу або інструментарію, а не ізольовано. Основна перевага полягає в тому, що він може обробляти ліво-рекурсивні граматики без повернення назад. У ньому пояснюється, що він не вироблятиме лівий факторинг під час фази синтаксичного аналізу, вимагаючи додаткового правого факторінгу під час виконання для належної оцінки.

Комбінатор аналізатора приймає функції аналізатора та виводить нову функцію аналізатора. Він відіграє видатну роль у розробці синтаксичного аналізатора завдяки інтеграції простого попереднього аналізу.

Він добре підходить для створення складних і швидких аналізаторів тексту. Крім того, він може обробляти контекстно-вільну та контекстно-залежну граматику. Основне правило використання синтаксичного аналізатора-комбінатора полягає у створенні коду шляхом реалізації граматичних правил замість написання синтаксичного аналізатора безпосередньо будь-якою іншою мовою програмування.

Загалом, алгоритм Пратта дозволяє створити «аналізатор загального призначення». До якого ви можете підключити всі правила парсера, необхідні для вашої мови. Це низхідний аналізатор загального призначення.

Його можна програмувати будь-якою мовою програмування. Можливість чисельних обчислень, але щоразу реалізована в Prolog або Lisp. Алгоритм працює за допомогою рекурсії, поки не досягне кінцевих маркерів (токенів). У цей момент він сортує свій список нетерміналів. І створює нові дерева похідних від кожного нетермінала, приєднуючи правила до терміналів як дітей.

Підхід аналізу документів швидкий і реалізований без складних навичок програмування. Це також відкритий код. Перш за все, алгоритми аналізу документів можуть ефективно розділяти текст і робити тексти придатними для подальшої трансформації. Таким чином, якщо ви хочете створити купу текстових даних, вичистіть деяку інформацію. Алгоритми розбору документів можуть ефективно допомогти вам вирішити проблему.

Visionify має явну перевагу в наданні послуг аналізу документів . Їх тривалий досвід дає їм неперевершену здатність аналізувати та структурувати існуючі документи. Зв’яжіться з нами, щоб отримати демо-версію наших рішень для обробки документів .

Щоб автоматично заповнювати базу даних серверу, вам потрібно отримати дані афтефікованого користувача. Ці дані вимагають від користувача вказати кафедру та курс навчання. На наступному кроці програма «витягує» файл із бази даних. Цей файл містить розклади для вибраних курсів і факультетів. З отриманого файлу програма спочатку шукає всі групи у файлі, а користувач вибирає потрібну групу. Група. Програма аналізує дані відповідно до відповідних даних. Назви груп містять дані про дні тижня, дні тижня містять дані про предмети, а предмети містять дані про назви предметів, пару годин і класи. Номер, ПІБ викладача (рис. 2.1).

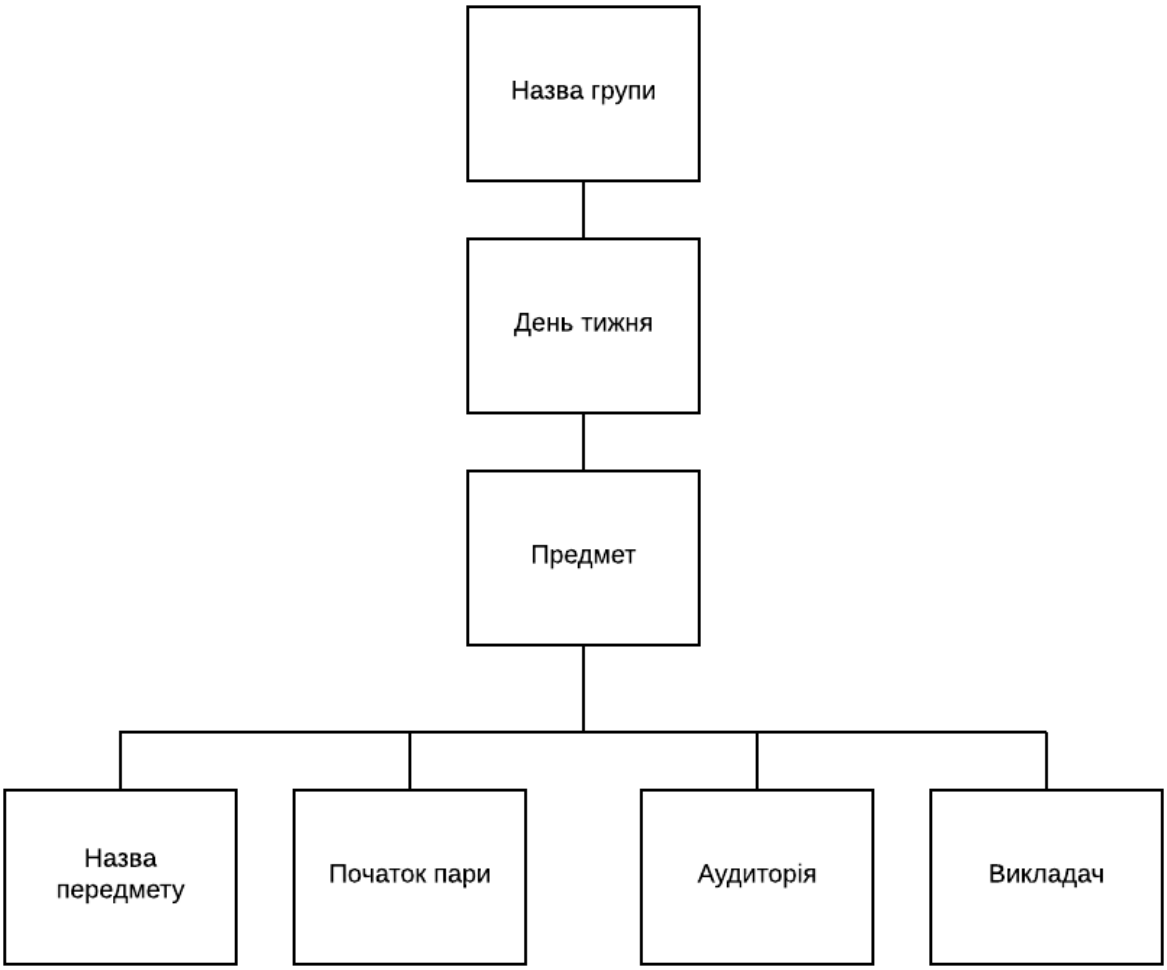

Рисунок 2.1 – Схема збереження даних

Файли з розкладом завдань додаються адміністратором у ручну. На 2.2 показано алгоритм парсингу розкладів у файлі.

Елементи алгоритму:

- group - назва потрібної групи.

- column – кількість стовпців у файлі Excel.

- row - кількість рядків у файлі Excel.

- current_line – поточний рядок (його вміст).

Цей алгоритм заснований на повторенні всіх рядків і стовпців у файлі. Вибрана група шукається у файлі. Читаючи файл, перевірте, чи це файл уроку чи файл розкладу. Для файлів перевірки дані фіксуються в додатку за кожен день.

Якщо файл є розкладом, дані записуються по днях і тижнях і циклічно прокручуються. Кожен тег містить відповідні пари. Пара містить такі дані:

- ім’я;
- лектор;
- аудиторія;
- тип пари (консультації, іспит, пари);
- час початку заняття.

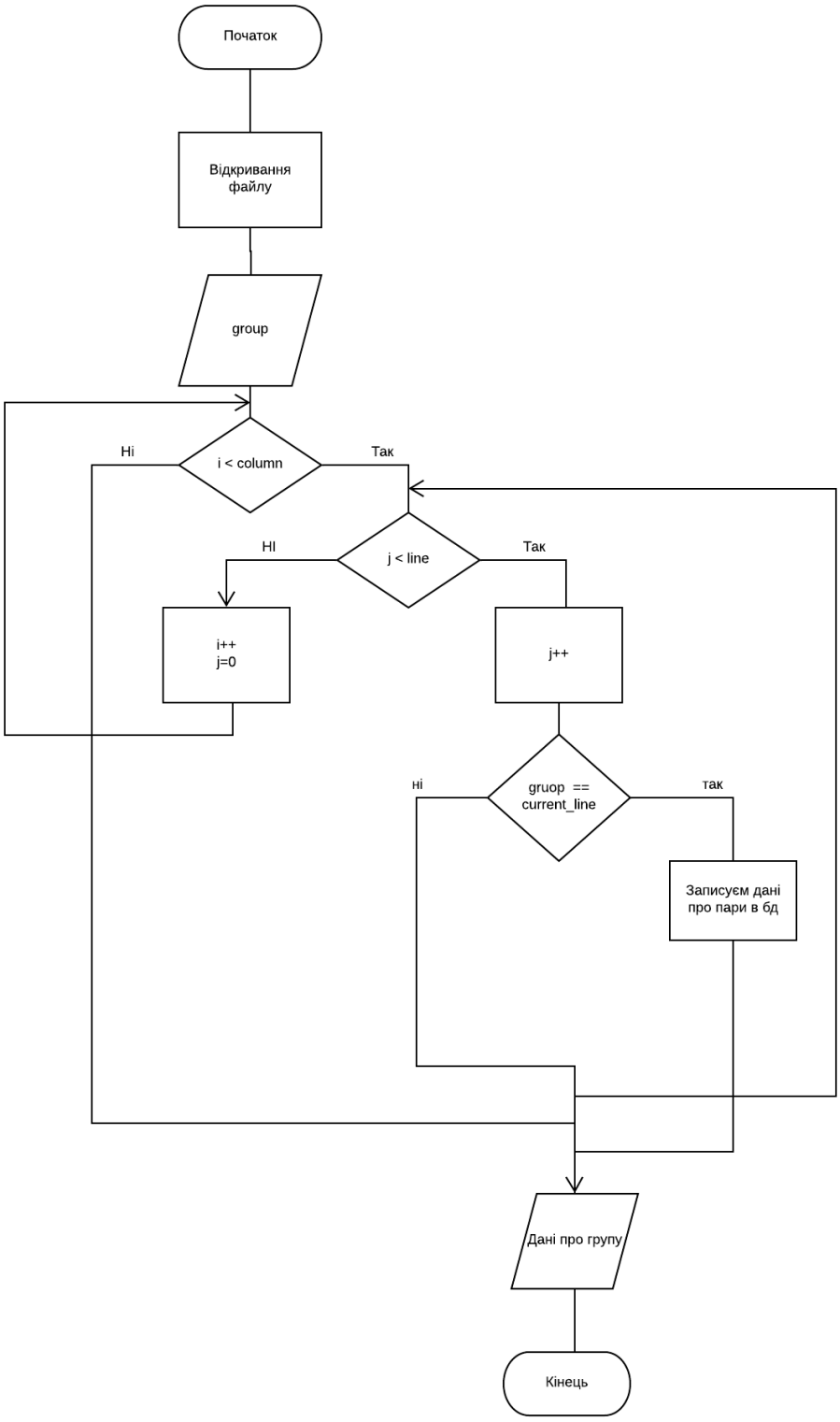

Рисунок 2.2 – Алгоритм пошуку у Excel файлі

Перед тим як добавляти інформацію до бази даних відбувається провірка чи існує група, вчитель з відповідної назвою, якщо не існує то добавляється новий і почанається заповнюватись відповідної інормацією, якщо існує то інформацію добавляється до поточного вчителя або групи.

У даному розділі було розроблено алгоритм парсингу XML файлу з розкладом у інформацію для бази даних

2.3 Аналіз і схематична побудова бази даних

База даних відноситься до колекції логічно пов’язаної інформації, організованої таким чином, щоб її можна було легко отримати, керувати та оновлювати. Доступ до баз даних зазвичай здійснюється в електронному вигляді з комп’ютерної системи, і вони зазвичай контролюються системою керування базами даних (СУБД).

Адміністратор бази даних, або DBA, відповідає за підтримку, захист і роботу баз даних, а також гарантує, що дані правильно зберігаються та витягуються.

Крім того, адміністратори баз даних часто співпрацюють з розробниками для розробки та впровадження нових функцій і вирішення будь-яких проблем. Адміністратор бази даних повинен добре розуміти як технічні, так і бізнес-потреби.

Роль DBA стає все більш важливою в сучасному інформаційному бізнес-середовищі. У всьому світі все більше і більше організацій покладаються на дані, щоб отримати аналітичну інформацію про ринкові умови, нові бізнес-моделі та заходи щодо скорочення витрат. Очікується, що глобальний ринок хмарних обчислень також розшириться, оскільки компанії перенесуть свої бізнес-операції в хмару.

Існує декілька типів адміністраторів баз даних, кожен із яких має певні обов’язки та відповідальність. Найпоширеніші типи администраторів баз даних включають системних адміністраторів, архітекторів баз даних, аналітиків баз даних, розробників моделей даних, менеджерів баз даних додатків, орієнтованих на завдання адміністраторів баз даних, аналітиків продуктивності, адміністраторів сховищ даних і хмарних администраторів баз даних. Є три основних типи зв’язків у базі даних це:

- Один до одного – коли один об’єкт може містити лише один екземпляр об’єкту. Наприклад є база даних у якій є таблиці студент і користувач, користувач містить дані з сайту а студент містить дані головні дані про студента, то в такому випадку використовується зв’язок один до одного.

- Один до багатьох – коли один об’єкт може містити в собі багато екзампляр об’єкту. Наприклад, є об’єкт екземпляру вчитель він може містити багато обєктів екземпляру урок.

- Багато до багатьох – коли кілька записів у таблиці пов’язанні з великою кількістю запитів у іншій таблиці. Наприклад, є таблиця клас і студент, то один студент може містити декілька класів, так само один клас може містити декілька студентів.

База даних даної дипломної роботи повинна виконувати завдання поставленні у розділі 2.1. Для цього вона повинна містити такі таблиці, як:

- User з атрибутами: Id(унікальний персональний ідентифікатор користувача), FirstName, SecondName, ChatId(автоматично створюється телеграм, персональний ідентифікатор чату). Ця таблиця буде зберігати інформацію про користувача і дані для відправки повідомлень чат-ботом;

- Student з атрибутами: Id, User_id, Group_id. Ця таблиця буде використовуатись для присвоєння певному користувачу групи, і змінні її без зміни даних про самого користувача. Ця таблиця буде містити в собі об’єкт User, зв’язок один до одного.

- Group з атрибутами: Id, Name, Student_Id, SheduleDay_Ids. Ця таблиця буде містити користувачів, які є учачниками групи і розклад заннять по денно. Ця таблиця буде містити в собі об’єкти Srudents, зв’язок один до багатьох, а ще будуть містити об’єкти SheduleDays, тип зв’язку такий самий.

- SheduleDay з атрибутами Id, DayWeek_Id, Lesson_Ids. Таблиця для розбиття пар по днях. Таблиця буде містити об’єкт DayWeek, зв’язок один до одного, і буде містити Lesson зв’язок один до багатьох.

- DayWeek з атрибутами Id, Name. Таблиця для зручного сортування пар по дням. Ця таблиця буде містити лише дні і унікальні індефікатори.

- Lesson з атрибутами Id, Name, Time, RoomNumber, LessonType. Містить дані про урок, де і котра година буде відбуватись.

- Teacher з атрибутами Id, FullName, Email, Lesson_Ids . Таблиця яка містить для вчителя і буде використовуватись для зв’язку з викладачем, і пошуку викладача де саме у нього пари, а коли вихідний день. Буде містити об

- Corpus з атрибутами Id, CorpusName, CorpusAbreviatura, CorpusNumber, Adress, Latitude (широта), Longitude (довгота). Містить дані про корпуса і де вони знаходяться.

Для побудови схематичного зображення бази даних використаєм додаток “https://app.diagrams.net/ ”. Діаграму бази зображено на рисунку 2.3.

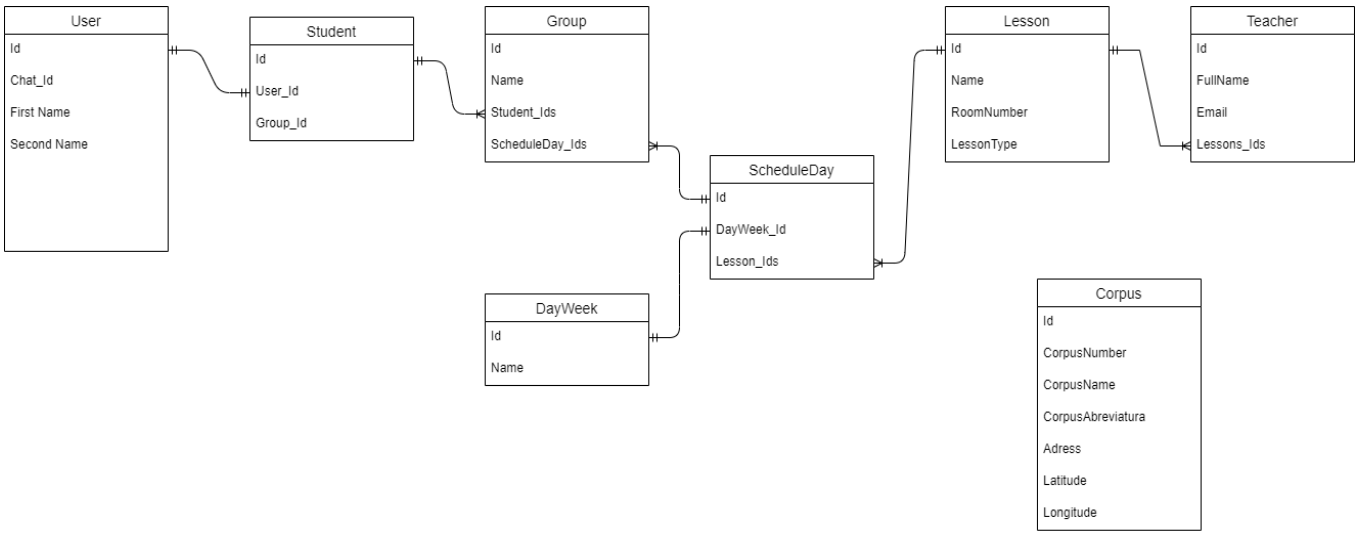

Рисунок 2.3 – Діаграма бази даних

У даному розділі було розроблено діаграму бази даних для дипломної роботи її таблиці і сутності, що найкраще підходить для наших функції і є зручною у використанні. Ознайомлено з типами зв’язків у базах даних [44, 45, 46].

2.4 Аналіз і побудова архітектури проекту

Архітектура ПЗ – це процес структурування ПЗ на слабкозв’язані та незалежнічастини, формування зв’язків та опис процесів передачі даних між ними. Архітектура ПЗ будується з метою найкращої відповідності вимогам проекту.

Архітектура програмного забезпечення підтримує аналіз якості системи, коли команди приймають рішення щодо системи, а не після впровадження, інтеграції чи розгортання.

Незалежно від того, чи йдеться про розробку нової системи, розробку успішної системи чи модернізацію застарілої системи, цей своєчасний аналіз дозволяє командам визначити, чи дадуть обрані ними підходи прийнятне рішення. Ефективна архітектура служить концептуальним клеєм, який утримує кожну фазу проекту разом для всіх його зацікавлених сторін, забезпечуючи гнучкість, економію часу та коштів і раннє визначення ризиків проектування.

Побудова ефективної архітектури, яка забезпечує швидке постачання продуктів для сьогоднішніх потреб, а також досягнення довгострокових цілей, може виявитися складним завданням. Нездатність визначити, визначити пріоритети та керувати компромісами між архітектурно значущими якостями часто призводить до затримок проекту, дорогої переробки або ще гірше.

Ефективна архітектура програмного забезпечення, що підтримується методами гнучкої архітектури, забезпечує ефективну постійну еволюцію системи. Такі практики включають документування архітектурних елементів і взаємозв’язків, призначених для досягнення ключових якостей; неодноразова оцінка архітектури на відповідність бізнес-цілям і місії організації; і аналіз розгорнутої системи на відповідність архітектурі.

При правильному виконанні ці методи забезпечують передбачувану якість продукту, менше проблем, пов’язаних із подальшою обробкою, економію часу та коштів на інтеграцію та тестування, а також економічно ефективну еволюцію системи.

Згідно з сформованими функціональними вимогами вирішено розглянути наступні архітектурні шаблони та стилі:

1. Клієнт-серверна архітектура.
2. Front end та back end.
3. N-рівневу архітектуру

Багаторівнева архітектура — це концепція клієнт-серверної архітектури в розробці програмного забезпечення, де функції представлення, обробки та керування даними логічно та фізично розділені. Кожна з цих функцій працює на окремому комп'ютері або в окремих кластерах, тому кожен може надавати послуги з максимальною потужністю, оскільки немає спільного використання ресурсів. Це розділення полегшує керування кожним окремо, оскільки виконання роботи над одним не впливає на інші, ізолюючи будь-які проблеми, які можуть виникнути.

N-рівневу архітектуру також називають багаторівневою архітектурою, оскільки програмне забезпечення розроблено таким чином, що функції обробки, керування даними та презентації фізично та логічно розділені. Це означає, що ці різні функції розміщуються на кількох машинах або кластерах, забезпечуючи надання послуг без спільного використання ресурсів і, таким чином, ці служби надаються з максимальною потужністю. «N» у назві n-tier architecture означає будь-яке число від 1.

Ваше програмне забезпечення не лише виграє від можливості отримувати послуги за найкращими цінами, але й ним легше керувати. Це пояснюється тим, що коли ви працюєте над одним розділом, внесені вами зміни не вплинуть на інші функції. І якщо є проблема, ви можете легко визначити її походження.

N-рівнева архітектура передбачає поділ програми на три різні рівні (рисунок 2.3) :

1. логічний рівень;
2. рівень презентації ;
3. рівень даних.

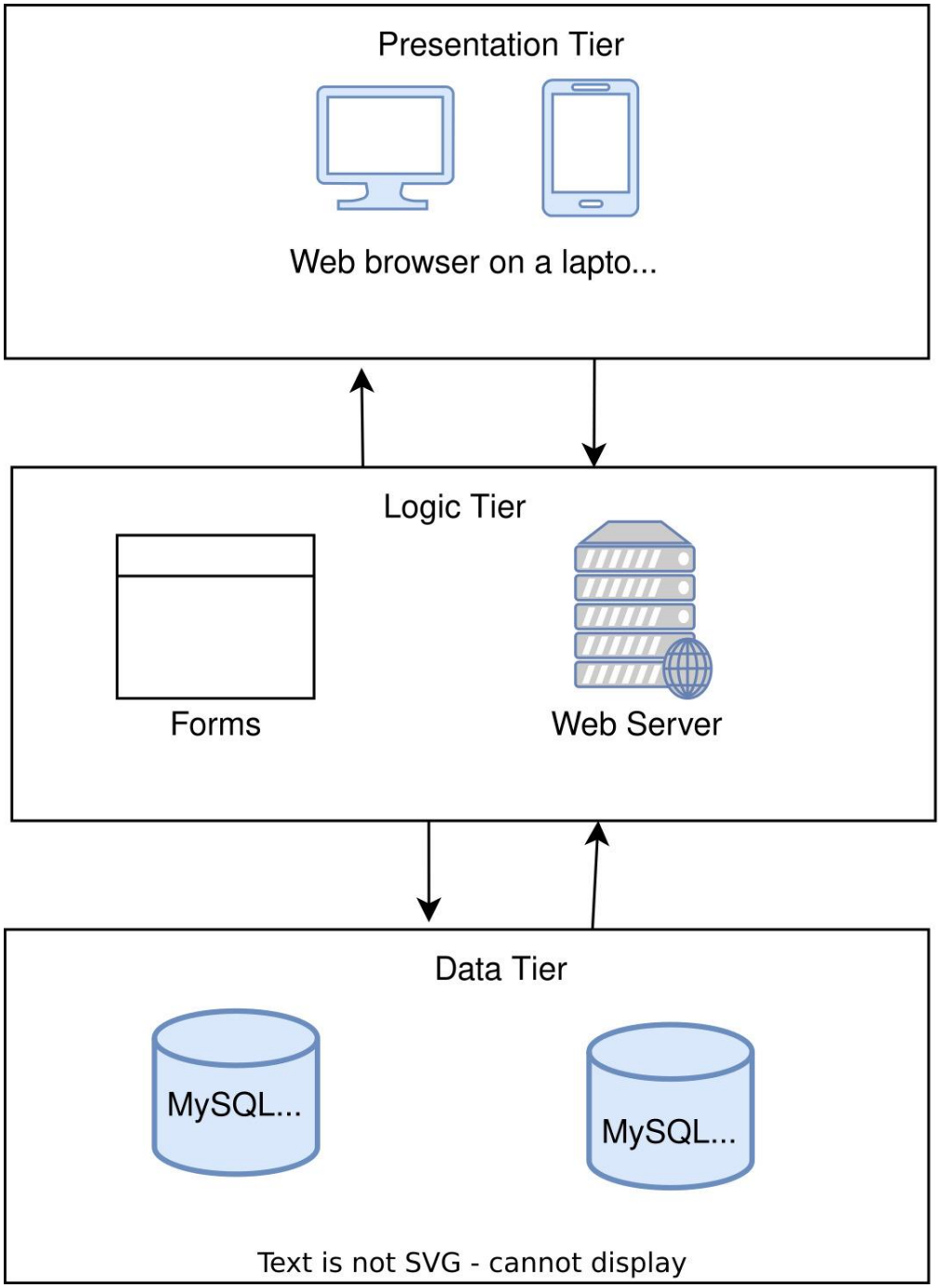

Рисунок 2.4 – Зображення n-рівневої архітектури

Окреме фізичне розташування цих рівнів – це те, що відрізняє n-рівневу архітектуру від структури модель-подання-контролер, яка розділяє лише рівні представлення, логіку та дані в концепції. N-рівнева архітектура також відрізняється від структури MVC тим, що перша має середній рівень або логічний рівень, який полегшує всі зв'язки між різними рівнями. Коли ви використовуєте структуру MVC, взаємодія, яка відбувається, є трикутною; замість того, щоб проходити через рівень логіки, рівень керування отримує доступ до шарів моделі та перегляду, тоді як

рівень моделі отримує доступ до рівня перегляду. Крім того, рівень керування створює модель, використовуючи вимоги, а потім надсилає цю модель на рівень перегляду.

Це не означає, що ви можете використовувати лише структуру MVC або n-рівневу архітектуру. Існує багато програмного забезпечення, яке об'єднує ці дві системи. Наприклад, ви можете використовувати n-рівневу архітектуру як загальну архітектуру або використовувати структуру MVC на рівні презентації.

Використання n-рівневої архітектури для програмного забезпечення має кілька переваг. Це масштабованість, простота управління, гнучкість і безпека.

- Безпека: Ви можете захистити кожен із трьох рівнів окремо, використовуючи різні методи.

- Легко керувати: ви можете керувати кожним рівнем окремо, додаючи або змінюючи кожен рівень, не впливаючи на інші рівні.

- Масштабований: якщо вам потрібно додати більше ресурсів, ви можете зробити це для кожного рівня, не впливаючи на інші рівні.

- Гнучкість: окрім ізольованої масштабованості, ви також можете розширювати кожен рівень будь-яким способом, який диктують ваші вимоги.

Отже, за допомогою n-рівневої архітектури ви можете запроваджувати нові технології та додавати більше компонентів без необхідності переписувати всю програму чи перепроектувати все програмне забезпечення, що полегшує масштабування та обслуговування. Тим часом, з точки зору безпеки, ви можете зберігати конфіденційну або конфіденційну інформацію на логічному рівні, тримаючи її подалі від рівня презентації, таким чином роблячи її більш безпечною.

Серед інших переваг:

- Більш ефективний розвиток. N-рівнева архітектура дуже зручна для розробки, оскільки на кожному рівні можуть працювати різні команди. Таким чином, ви можете бути впевнені, що спеціалісти з дизайну та презентації працюють над рівнем презентації, а експерти з баз даних – над рівнем даних.

- Легко додавати нові функції. Якщо ви хочете представити нову функцію, ви можете додати її до відповідного рівня, не впливаючи на інші рівні.

- Легко використовувати повторно. Оскільки програма розділена на незалежні рівні, ви можете легко використовувати кожен рівень для інших проектів програмного забезпечення. Наприклад, якщо ви хочете використовувати ту саму програму, але для іншого набору даних, ви можете просто скопіювати рівні логіки та представлення, а потім створити новий рівень даних [50].

2.5 Аналіз основних бібілотек і засобів для роботи

Опишемо основні бібліотеки і засоби для роботи ngrok, nuget package та entity framework.

ngrok — це кросплатформна програма, яка дає змогу розробникам видавати локальний сервер розробки в Інтернет з мінімальними зусиллями. Програмне забезпечення створює враження, що ваш локальний веб-сервер розміщено на піддоміні ngrok.com, що означає, що публічна IP-адреса чи доменне ім’я на локальній машині не потрібні. Подібних функцій можна досягти за допомогою зворотного тунелювання SSH, але для цього потрібні додаткові налаштування, а також розміщення вашого власного віддаленого сервера.

ngrok може обходити обмеження NAT Mapping і брандмауера, створюючи довготривалий TCP-тунель із випадково згенерованого субдомену на ngrok.com (наприклад, 3gf892ks.ngrok.com) до локальної машини. Після вказівки порту, який прослуховує ваш веб-сервер, клієнтська програма ngrok ініціює безпечне з’єднання з сервером ngrok, а потім будь-хто може надсилати запити до вашого локального сервера з унікальною адресою тунелю ngrok. Посібник розробника ngrok містить більш детальну інформацію про те, як це працює.

Різноманітні тунельні сервери доступні в усьому світі та розташовані в США (Огайо), Європі (Франкфурт), Азії (Сінгапур) та Австралії (Сідней). Крім того, серверне програмне забезпечення ngrok може бути самостійно розміщено на VPS або виділеному сервері.

За замовчуванням ngrok створює кінцеві точки HTTP і HTTPS, що робить його корисним для тестування інтеграції зі сторонніми службами або API, які потребують дійсних доменів SSL/TLS. Інші варіанти використання включають: швидку демонстрацію локальних демонстрацій клієнтам, тестування серверних програм для мобільних додатків і запуск персональних хмарних служб із домашнього ПК.

Однією з високо оцінених особливостей ngrok є можливість відстежувати та відтворювати HTTP-запити через веб-консоль ngrok. Функція відтворення дуже корисна під час тестування викликів API або веб-хуків, оскільки можна легко перевірити весь вміст заголовків і дані запитів/відповідей в одному місці через інтерфейс консолі.

Преміум-версія ngrok, ngrok link, доступна для розробників для використання у виробництві та пропонує такі функції, як автоматизація API та керування обліковими даними, що робить її придатною для віддаленого керування Інтернетом речей у професійному середовищі.

Важливим складником кожної сучасної платформи розробки є механізм, за допомогою якого розробники можуть створювати, ділитися та використовувати корисний код. Такий код дуже часто групується в «пакети», які містять скомпільований код (наприклад, бібліотеки DLL) разом з іншим вмістом, необхідним для проектів, які використовують ці пакети.

Для .NET (включно з .NET Core), підтримуваним Microsoft механізмом спільного використання коду є NuGet , який визначає, як пакети для .NET створюються, розміщуються та споживаються, і надає інструменти для кожної з цих функцій.

Коротше кажучи, пакет NuGet — це один ZIP-файл із розширенням .nupkg, який містить скомпільований код (DLL), інші файли, пов’язані з цим кодом, і описовий маніфест, який має в собі таку інформацію, як номер версії пакета. Розробники спільного коду розробляють пакети та публікують на публічному або приватному хості.

А вже споживачі, які користуються цими пакетами, отримують пакети від відповідних хостів, додають їх до власних проектів, а потім викликають

функціональність пакета в коді власного проекту. Пізніше NuGet самостійно опрацьовує всі проміжні деталі.

NuGet підтримує приватний хостинг із публічним хостингом nuget.org, тому споживачі можуть користуватися пакетами NuGet, щоб ділитися приватним кодом зі своєю організацією чи колегами. Пакети NuGet також можна використовувати для факторизації свого коду лише для використання у вашому проекті.

Інакше кажучи, пакет NuGet може спільно використовуватися, але не вимагає жодних конкретних можливостей для спільного використання. Пакет NuGet – це одиниця коду.

Потік пакетів між творцями, хостами та споживачам, як публічний хост, NuGet має центральне сховище з понад 100 тисячами унікальних пакетів на nuget.org. Цими пакетами кожного дня користуються мільйони розробників .NET/.NET Core. NuGet також дозволяє розташовувати ваші пакети приватно в хмарі (Azure DevOps), у вашій приватній мережі або навіть у локальній файловій системі. Тобто до цих пакетів мають доступ лише ті розробники, яким доступний цей хост. Це дуже зручно, адже ваші приватні пакети є доступними лише для певних груп споживачів.

Ви також можете використовувати параметри конфігурації, щоб на 100% контролювати, до яких хостів може отримати доступ певний комп’ютер, і забезпечити отримання пакетів із потрібних джерел, а не з публічних сховищ, як-от nuget.org.

Незалежно від свого типу, хост діє як точка контакту між розробниками пакету і його споживачами. Автор створює певний пакет NuGet і публікує його на хості. Пізніше користувачі шукають на доступних хостах корисні та сумісні пакети та завантажують ці пакети, щоб включити їх у свої проекти. Після встановлення API пакет стає доступним для решти коду вашого проекту.(рисунок 2.5).

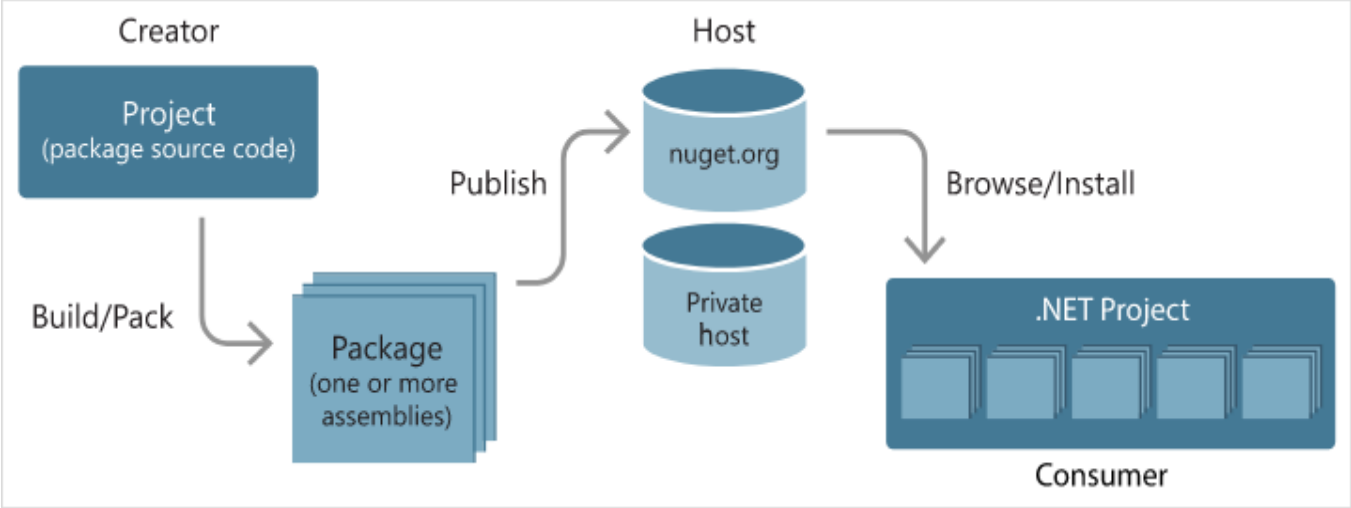

Рисунок 2.5 – Відносини між творцями пакетів, хостами пакетів і споживачами пакетів

«Сумісний» пакет означає, що там містяться збірки, скомпільовані хоча б для однієї цільової платформи .NET, що сполучна з цільовою структурою проекту-споживача. Розробники мають змогу формувати спеціальні пакети для фреймворків. Ви можете використовувати елементи керування UWP або користуватися більшим діапазоном можливостей. Аби якнайбільше підвищити сумісність пакетів, розробники звертаються до .NET Standard, який доступний для всіх проектів .NET і .NET Core. Це найбільш ефективний інструмент як для авторів, так і для користувачів пакету, адже один пакет (зазвичай має в собі одну збірку) працює для всіх проектів-споживачів.

З другого боку, творці пакетів, які потребують API, відмінних від .NET Standard, можуть створити окрему збірку для кожної цільової структури, яка їм потрібна, і включити всі ці збірки в один пакет, який називається «багатоцільовим». Після встановлення такого пакета NuGet буде видобувати лише збірки, необхідні для вашого проекту. Отже, це зведе до мінімуму пакети остаточного додатка та збірки, утворені цим проектом. Звичайно, багатоцільові пакети складніші для самих розробників.

Однією з головних особливостей системи керування пакетами є здатність просто будувати на основі роботи інших. Отже, більша частина того, що робить NuGet - це саме керування цим деревом залежностей або «графом» від імені проекту. Коротко кажучи, користувачу потрібно просто керувати пакетами, які

використовує його проект. Якщо один із цих пакетів використовує інші пакети, які теж можуть використовувати інші - NuGet забезпечить всі залежності нижчого рівня.

На рисунку 2.6 показано проект, який залежить від п’яти пакетів, які, у свою чергу, залежать від кількох інших.

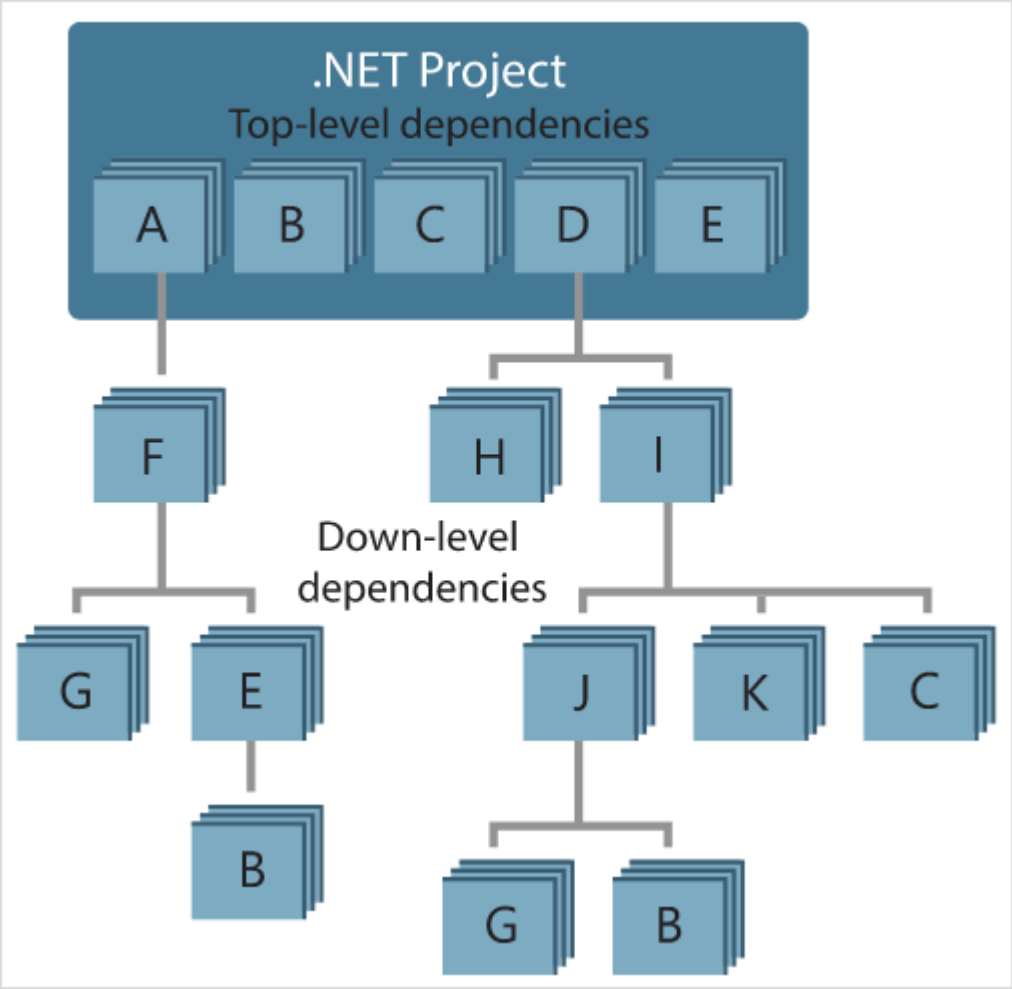

Рисунок 2.6 – Приклад графіка залежностей NuGet для проекту .NET

Зверніть увагу, що деякі пакунки з’являються кілька разів у графі залежностей. Наприклад, є три різні споживачі пакета B, і кожен споживач також може вказати іншу версію для цього пакета (не показано). Це звичайне явище, особливо для пакетів, які широко використовуються. На щастя, NuGet виконує всю важку роботу, щоб точно визначити, яка версія пакета B задовольняє всіх споживачів. Потім NuGet робить те саме для всіх інших пакетів, незалежно від того, наскільки глибокий графік залежностей.

Відстеження посилань і відновлення пакетів. Оскільки проекти можна легко переміщати між комп'ютерами розробників, репозиторіями керування вихідним кодом, серверами збірки тощо, дуже непрактично зберігати бінарні збірки пакетів NuGet безпосередньо прив'язаними до проекту. Це зробило б кожну копію проекту надмірно роздутою (і, отже, втрачало б місце в репозиторіях керування вихідним кодом). Також було б дуже складно оновити двійкові файли пакетів до новіших версій, оскільки оновлення потрібно було б застосовувати до всіх копій проекту.

Натомість NuGet підтримує простий довідковий список пакетів, від яких залежить проект, включаючи залежності верхнього та нижнього рівнів. Тобто щоразу, коли ви встановлюєте пакет із якогось хоста в проект, NuGet записує ідентифікатор пакета та номер версії в список посилань. (Видалення пакета, звичайно, видаляє його зі списку.) Потім NuGet надає засоби для відновлення всіх пакетів, на які посилаються, за запитом, як описано в розділі Відновлення пакета .

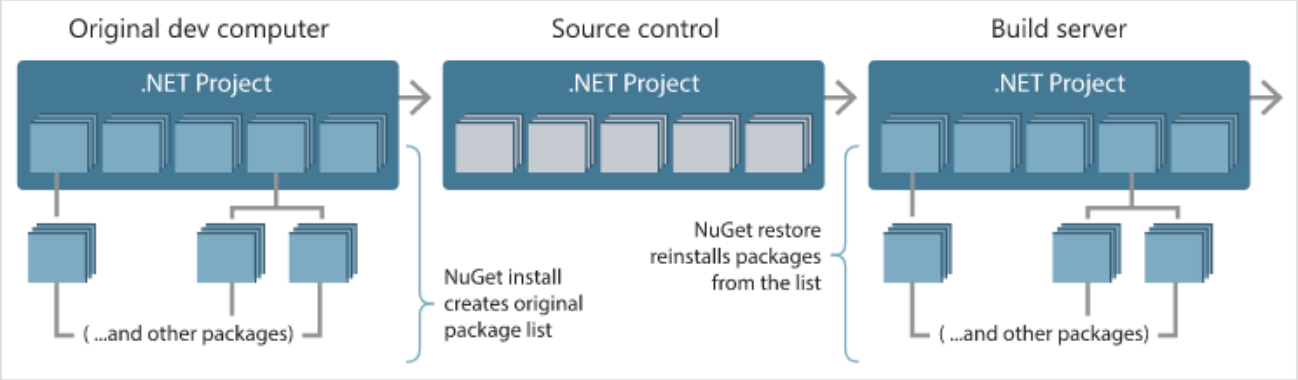

Рисунок 2.6 – Список посилань NuGet створюється під час інсталяції пакета, і його можна використовувати для відновлення пакетів деінде

Маючи лише список посилань, NuGet може перевстановити — тобто відновити — усі ці пакети з загальнодоступних і/або приватних хостів у будь-який час. Переносячи проект у систему керування джерелами або надаючи до нього доступ іншим способом, ви включаєте лише список посилань і виключаєте будь-які двійкові файли пакетів [18].

Комп'ютери, які отримують проекти, наприклад сервери збірки, що в свою чергу отримують копії проектів - частину автоматизованої системи розгортання,

просто надсилають запит NuGet відновити залежності, коли це необхідно. Системи збірки (такі як Azure DevOps) забезпечують кроки «відновлення NuGet» для цієї цілі. Так само, коли автор отримує копію проекту (наприклад, під час клонування сховища), він може викликати команду, як-от nuget restore (NuGet CLI), dotnet restore (dotnet CLI) або Install-Package (Package Manager Console) для того, щоб отримати всі потрібні пакети. Що стосується Visual Studio, вона автоматично відновлює пакет у процесі створення проекту (якщо увімкнено автоматичне відновлення).

Отже, зрозуміло, що головна роль NuGet для розробників полягає в підтримці цього списку посилань від імені вашого проекту та забезпеченні засобами для ефективного відновлення (і оновлення) цих посилань на пакети. Цей список підтримується в одному з двох форматів керування пакунками.

PackageReference (або «посилання на пакет у файлах проекту») | (NuGet 4.0+) Зберігає список залежностей проекту найвищого рівня безпосередньо у файлі проекту, тому окремий файл не потрібен. Пов’язаний файл obj/project.assets.jsondинамічно генерується для керування загальним графіком залежностей пакетів, які використовує проект разом із усіма залежностями нижнього рівня. PackageReference завжди використовується проектами .NET Core.

packages.config: (NuGet 1.0+) XML-файл, який підтримує плоский список усіх залежностей у проекті, включаючи залежності інших встановлених пакетів. Встановлені або відновлені пакети зберігаються в packagesпапці.

Який формат керування пакетами використовується в будь-якому конкретному проекті, залежить від типу проекту та доступної версії NuGet (та/або Visual Studio). Щоб перевірити, який формат використовується, просто знайдіть packages.configy корені проекту після встановлення вашого першого пакета. Якщо у вас немає цього файлу, пошукайте елемент <PackageReference> безпосередньо у файлі проекту.

Платформа Entity Framework із відкритим кодом для додатків .NET підтримується Microsoft і дає змогу розробникам маніпулювати даними за допомогою об’єктів доменного класу, не концентруючись на базових таблицях і стовпцях бази даних, у яких зберігаються дані. Entity Framework дозволяє

розробникам працювати на вищому рівні абстракції під час роботи з даними, створюючи та обслуговуючи орієнтовані на дані програми з меншим кодом, ніж традиційні програми.

Офіційне визначення терміну: «Entity Framework — це об’єктно-реляційний маппер (O/RM), який дозволяє розробникам .NET працювати з базою даних за допомогою об’єктів .NET. Тому зникає потреба більшості коду у доступі до даних, який зазвичай потрібно писати розробникам».

На наступному малюнку показано, де Entity Framework підходить для вашої програми.

На рисунку 2.7 показано, де Entity Framework підходить для вашої програми.

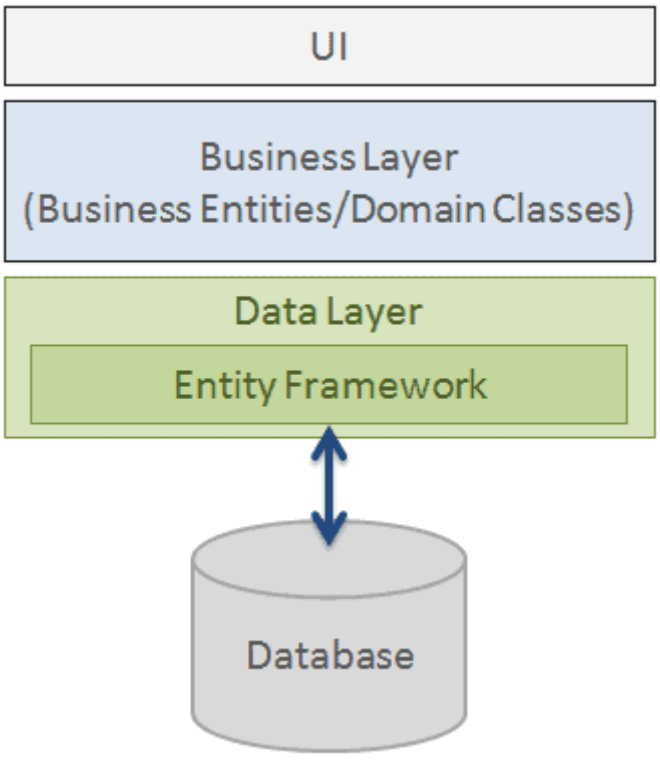

Рисунок 2.7 – Використання Entity Framework

Згідно наведеного вище малюнку, Entity Framework підходить для зв’язку між бізнес-сутностями (класами домену) і базою даних. Платформа зберігає дані, які містяться у властивостях бізнес-сутностей, а також отримує дані з бази даних і автоматично перетворює їх на об’єкти бізнес-сутностей.

Функції Entity Framework. Кросплатформенність: EF Core — це кросплатформна структура, яка може працювати на Windows, Linux і Mac.

Моделювання: EF (Entity Framework) створює EDM (Entity Data Model) на основі сутностей POCO (Plain Old CLR Object) із властивостями get/set різних типів даних. Він використовує цю модель, коли надсилає запити або зберігає дані сутності в базовій базі даних.

Запити: Entity Framework дозволяє використовувати запити LINQ (C#/VB.NET) з метою отримання даних із основної бази даних. Постачальник бази даних перекладе ці запити LINQ на мову запитів, специфічну для бази даних (наприклад, SQL для реляційної бази даних). Entity Framework також дозволяє нам виконувати необроблені запити SQL прямо до бази даних.

Відстеження змін: Entity Framework відстежує зміни, які відбулися в екземплярах ваших об’єктів (значення властивостей), які потрібно надіслати до бази даних.

Збереження: Entity Framework виконує команди INSERT, UPDATE і DELETE для бази даних на основі змін, які відбулися у ваших сутностях під час виклику SaveChanges()методу. Entity Framework також надає асинхронний SaveChangesAsync()метод.

Паралелізм: EF використовує Optimistic Concurrency за замовчуванням, аби захистити зміни від перезапису, внесені іншим користувачем після того, як дані були отримані з бази даних.

Транзакції: Entity Framework автоматично керує транзакціями під час запиту або збереження даних. Він також надає параметри для налаштування керування транзакціями.

Кешування: EF включає перший рівень кешування з коробки. Тобто повторні запити користувача повертатимуть дані з кешу замість потрапляння в базу даних.

Вбудовані угоди: Entity Framework дотримується угод щодо шаблону програмування конфігурації та включає в себе набір правил за замовчуванням, що автоматично налаштовують модель Entity Framework.

Конфігурації: Entity Framework дає змогу нам налаштовувати модель EF за допомогою атрибутів анотації даних або Fluent API для заміни умовних угод.

Міграції: Entity Framework надає набір команд міграції, які можна виконати на консолі диспетчера пакетів NuGet або в інтерфейсі командного рядка, щоб створити базову схему бази даних або для керування нею.

2.6 Висновки до розділу

Розроблено основі функціональні вимоги до чат-боту. На основі функціональних вимог було розроблено алгоритми парсингу даних з файлу з розкладом у базу даних. Крім цього було проведено аналіз баз даних, та схематично побудовано базу даних, для чат-боту. Проаналізовано і побудовано архітектуру проекту. Досліджено основі бібілотеки і засоби для розробки чат-боту.

3 РЕАЛІЗАЦІЯ ЧАТ-БОТУ

3.1 Структура програмного модуля чат-боту

Для реалізації чат-боту розроблено програмне забезпечення, діаграму використання якого зображено на рисунку 3.1.

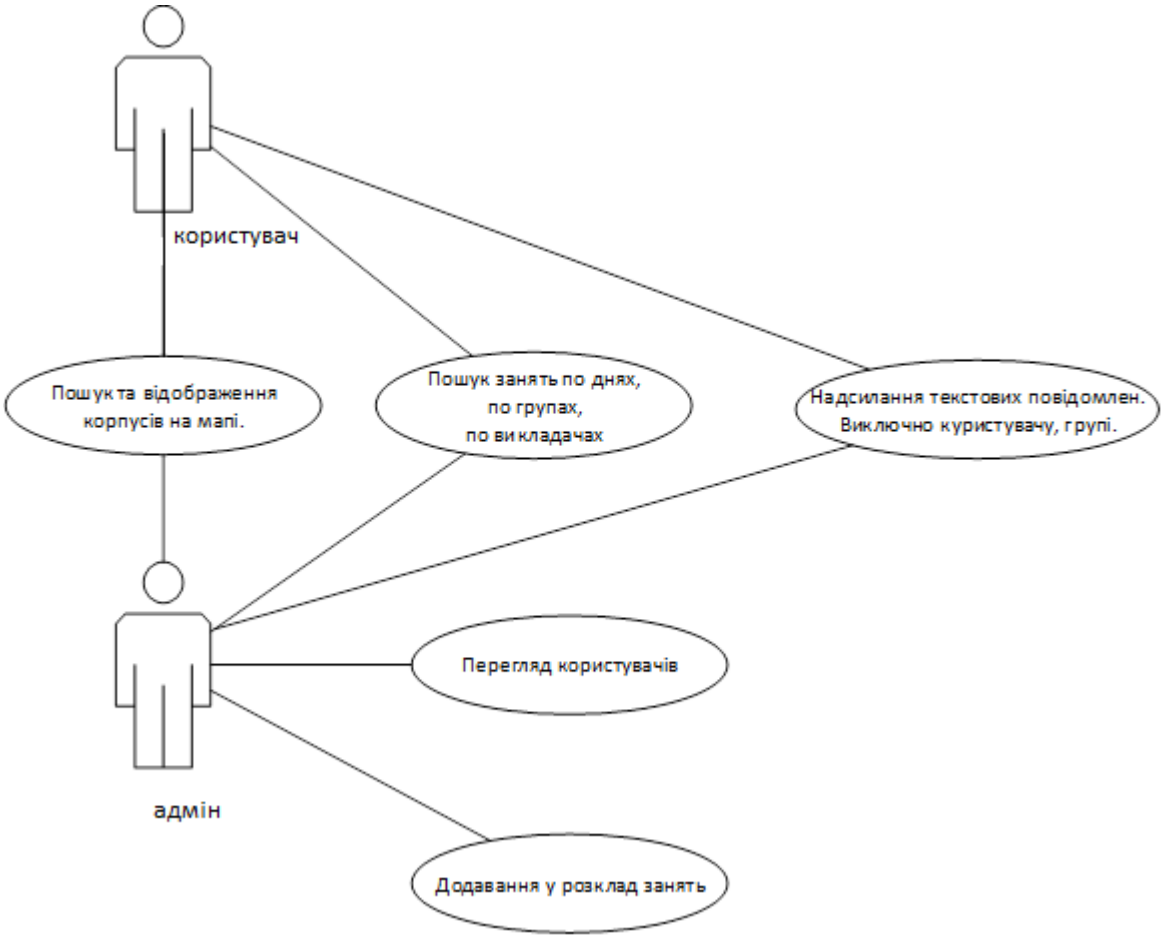

Рисунок 3.1 – Діаграма варіантів чат-боту

Адміністратор розробленої програмної системи, у ролі якого можуть бути викладач, працівники деканату, секретар, що матиме можливість вводити вхідні дані, такі як до прикладу зміна розкладу.

Як показано на рисунку 3.1, програмна система персоналу та студентам навчального закладу можливість комунікації, редагування занять, їх пошуку та контролю студентів.

Функція “ Пошук та відображення корпусів на мапі. ”, це процес пошуку потрібного корпусу та аудиторії де повинне відбутись заняття, або ж пошук аудиторії, де знаходиться відповідний викладач, кафедра, що значно спростить навігацію студентів та працівників навчального закладу.

Функція “ Надсилання текстових повідомлень”, відповідає за можливість комунікації, а саме надсилання текстових та медіа повідомлень не одній людині, а цілим групам користувачів за допомогою сервісу розсилки, а також планування розсилки повідомлень (сповіщення про заняття, можливість розсилки даних факультету для всіх користувачів).

Функція “ Пошук та відображення корпусів на мапі. ”, це процес пошуку потрібного корпусу та аудиторії де повинне відбутись заняття, або ж пошук аудиторії, де знаходиться відповідний викладач, кафедра.

Функція “ Перегляд користувачів. ”, це процес переглядату користувачів за допомогою сервера.

Функція “ Додавання у розклад занять. ”, дозволить додавати та корегувати заняття (в мануальному режимі або за допомогою технології парсингу файлу).

Зважаючи на вищезазначене, у наступному підрозділі охарактеризуємо програмне середовище для реалізації чат-боту.

3.2 Початок роботи у Visual Studio

Для розробки чат бота у Visual Studio 2019 під час її установки потрібно вибрати пункти “ASP.net and web development” і “Azure development” (рисунок 3.2). Перший пакет потрібний для створення web-проектів і їх запуску на локальному сервері із допомогою, другий для публікації проекту і його бази даних на сервери Azure і зміни його в реальному часі.

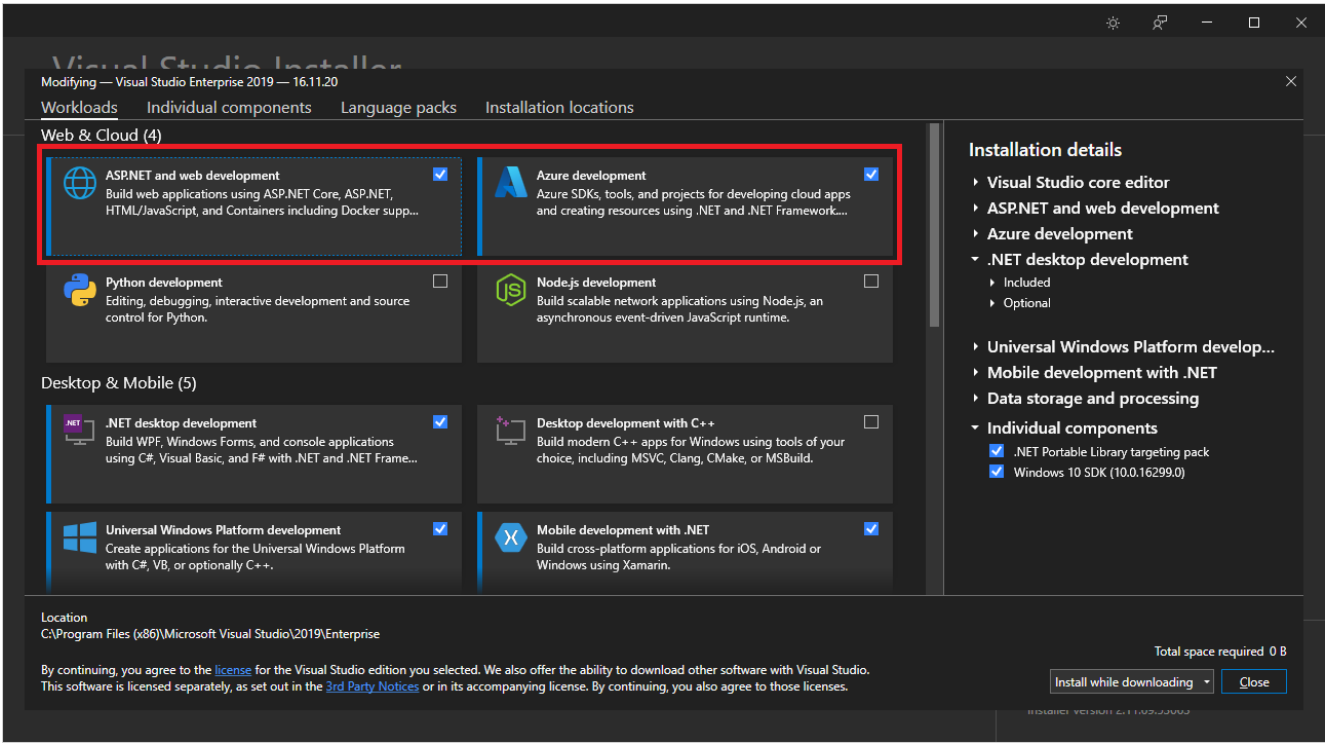

Рисунок 3.2 – Пакети для чат боту

Коли встановим проект, перейдем до створення самого проекту. Шаблон для створення нашого проекту це ASP.Net core web api, цей проект буде сервером на нашому проекті, він буде відповідати за головні запити і контакт ngrok з нашим проектом. Наступим кроком буде вибір іменні нашого проекту, дальше оберем мову програмування, у даному випадку .Net 5, у основні характеритистки. Поставим галочку для “Configure for HTTPS” і знімемо за не потрібності з “Enable Docker”(Рисунок 3.3).

Docker це програмне забезпечення для автоматизації розгортання додатків і управління ними в контейнерних середовищах. З 2015 року ми використовуємо libcontainer, власну бібліотеку, яка абстрагує можливості віртуалізації ядра Linux. З появою ініціативи Open Container почався перехід від монолітної до модульної архітектури.

“Configure for Https” – автоматичне генерування папки properties, файл launchSetting.com, і добавляє у файл startup.cs команду UseHttpsRedirection, детальніне розеберм цей файл у наступних розділах.

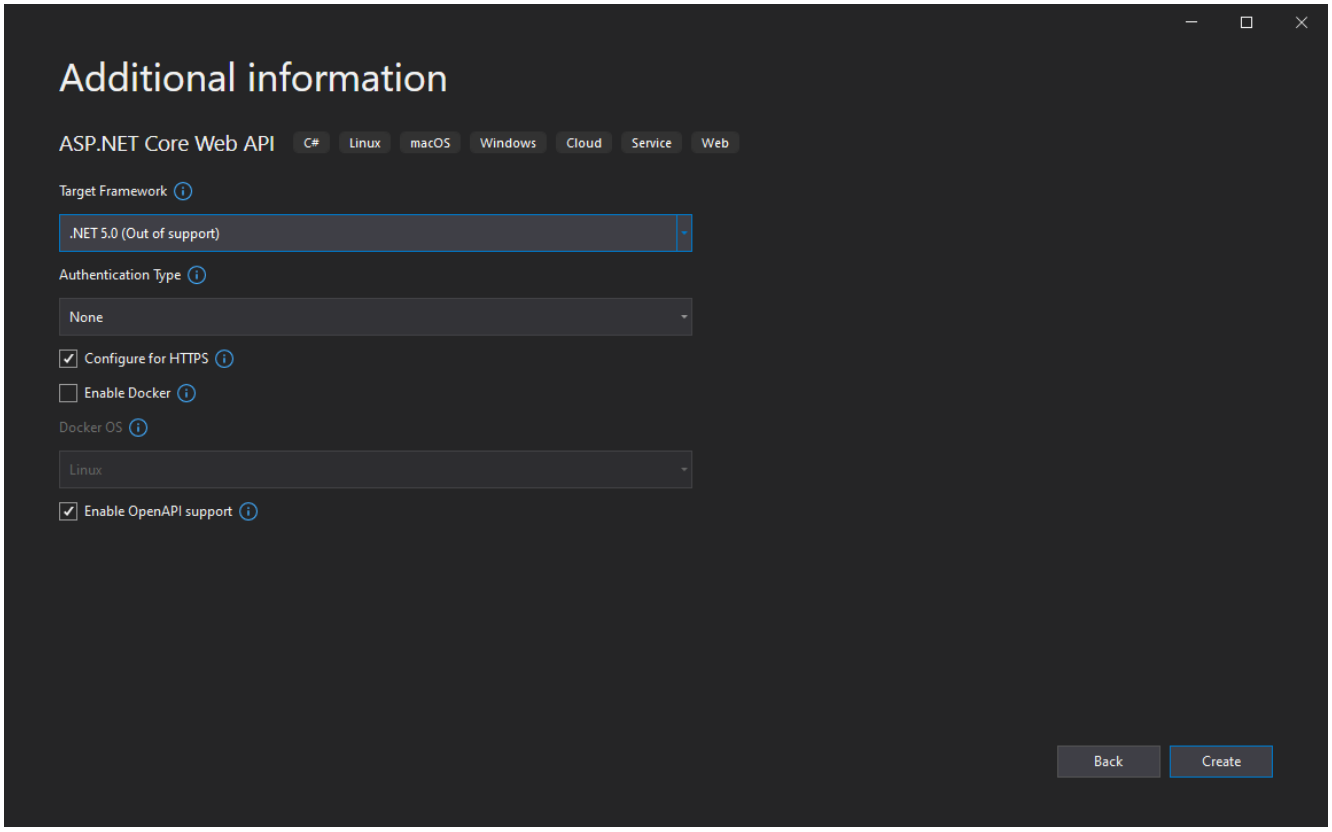

Рисунок 3.3 – Основні характеристики проекту

Наступним етапом ми добавим три під проекти до нашого проекту. Це буде три однакових під проекти з типом “Class Library”, назвемо їх DomainLayer, ServiceLayer, UITelegram.

DomainLayer – буде містити підключення до бази даних, і основні функції для роботи з базою даних.

ServiceLayer – буде містити основні сервіси, які потрібні для нашого проекту.

TelegramUI – буде містити користувацький інтерфейс, який бачить користувач у телеграм, і основні команди для роботи із ним.

ZUNU – це головний проект, який буде містити всі запити, рядок на підключення до бази даних, і підключення до Telegram.

Наступним етапом буде додавання до кожного проекту власних бібліотек, кожний проект буде містити власні бібліотеки, які ми добавим за допомогою Nuget Packeges.

Для DomainLayer добавим такі бібліотеки, як:

- Microsoft.EntityFrameworkCore;

- Microsoft.EntityFrameworkCore.Design;
- Microsoft.EntityFrameworkCore.SqlServer;
- Microsoft.EntityFrameworkCore.Tools.

Для ServiceLayer добавим таку бібліетеку як: AutoMapper, бібліотека для передавання змінних між рівнями, на нижчий або вищий рівень, вона працює по назві змін, або ми можем на пряму вказати, які дані вона повинна куди записувати.

Для прикладу у нас на рівні є DomainLayer, є дві сутності: Student і User, а на рівні ServiceLayer то StudentModel, StudentModel містить поля зі Student і поля зі User, то для них буде такий мапер:

```
CreateMap<Student, StudentModel>()
        .ForMember(sm => sm.FirstName, s => s.MapFrom(s => s.User.FirstName))
        .ForMember(sm => sm.SecondName, s => s.MapFrom(s =>
s.User.SecondName))
        .ForMember(sm => sm.ChatId, s => s.MapFrom(s => s.User.ChatId))
            .ReverseMap();
```

Розеглянемо детальніше цей кусок коду:

- .ReverseMap – вказує що маппер буде працювати у дві сторони;
- CreateMap вказує, які саме об’єкти ми будем мапити.
- ForMember містить у собі два елемента, які розділенні комою. Перший елемент, за допомогою лямбда виразів вказуєм, який саме елемент, другий елемент вказуєм куда саме вставити.

Для використання AutoMapper у потрібний клас, який буде містити мапінги потрібно наслідувати від класу “Profile”.

Для ZUNU бібліотеки добавим такі, як:

- Microsoft.EntityFrameworkCore.Design;
- Microsoft.AspNetCore.Mvc.NewtonsoftJson – для можливості отримання автоматичної конвертації JSON файлу в модель;
- Telegram.Bot – бібліотека для створення webhook зв’язку між сервером і телеграмом. Ця бібліотека дозволяє відправляє дані для клієнта, елементи для представлення кнопок і самі кнопки для користувача.
- Swashbuckle.AspNetCore – бібліотека для підключення Swagger елемнта. Swagger – це інтерфейс для зручного показування запитів, і робити і з ним. На

ньому POST,GET,UPDATE,DELETE запити відображаються різними кольорами. Його можна використовувати для режиму адміністратора,щоб заповнювати базу даних і провіряти її. Під час її використання ще показуються шляхи для використання запитів .Приклад зображення інтерфейсу Swagger на рисунку 3.4.

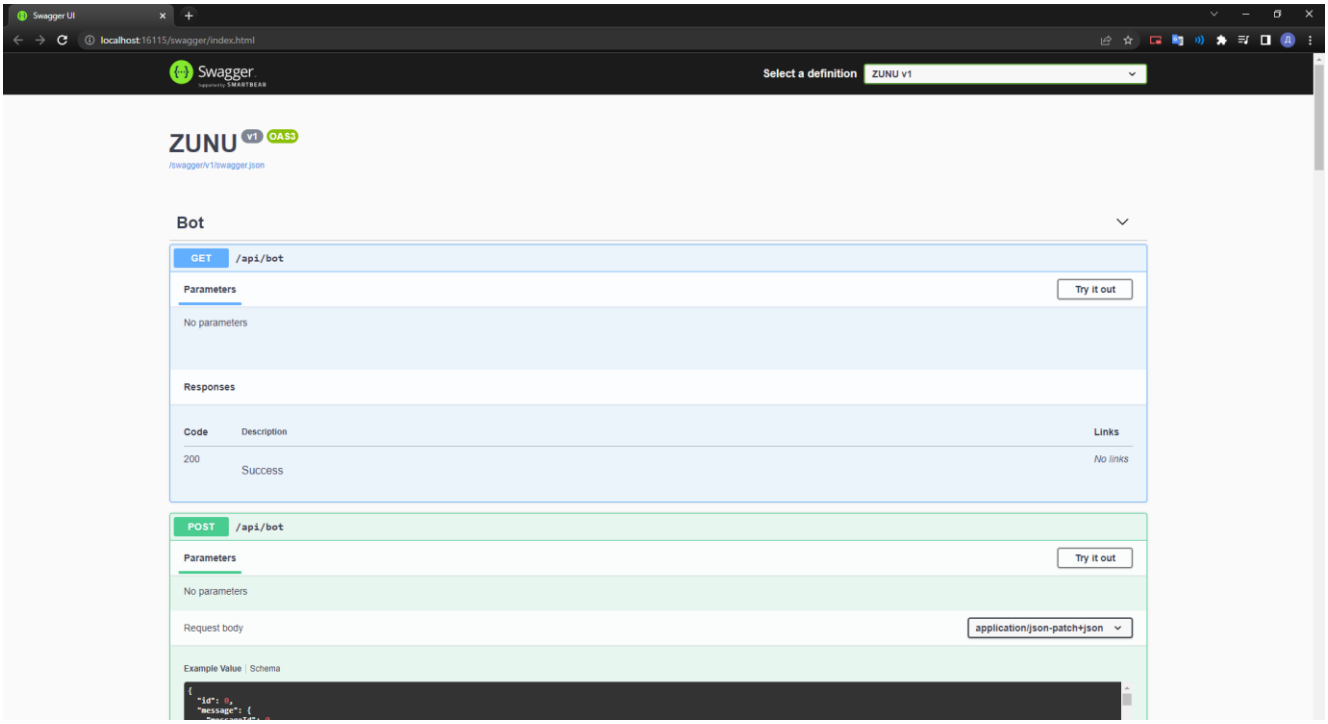

Рисунок 3.4 – Приклад використання Swagger

Для UITelegram лише одна бібліотека Telegram.Bot.

У даному розділі було створено проект і під проекту для дипломної роботи, встановлено основні пакети для кожного підпроекту, і розглянуто деякі бібліотеки для роботи нашої дипломної роботи.

3.3 Розробка багаторівневого проекту

У даному розділі ми розберем кожний підпроект дипломної роботи, які були розроблені у попередньому розділі. Почнем із найнижчого рівня це DomainLayer.

DomainLayer цей рівень відповідає за підключення до бази даних, містить сутності бази даних, і основні функції для роботи з базою даних. На рисунку 3.5 зображено архітектуру проекту даного шару.

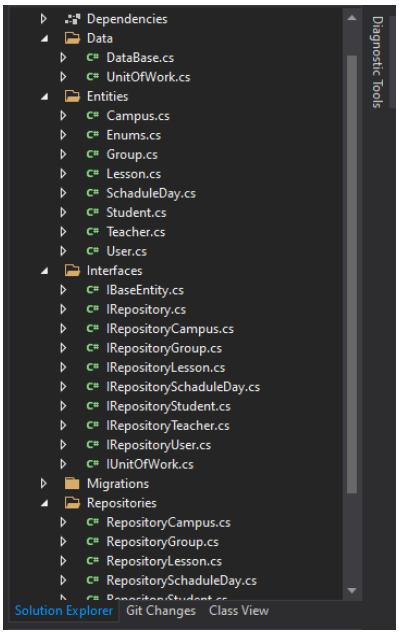

Рисунок 3.5 – Файлова система DomainLayer

У даному розділі відбувається створення бази даних, почнем розбір із цього. База даних буде створена з допомогою методу Code-First. Цей метод автоматично створює міграції з сутностей і зв’язків. Зв’язки прописуються безпосередньо у самих сутностях і у файлі DataBase. Для створення міграцій нам протрібно прописати у консольному вікні команду “Add-Migration name”, вона створює папку Migrations і добвляє у неї файл з іменем нашої міграції, яку ми вписали в комадному рядку. Цю міграцію ми можем використовувати для створення таблиць, у різних базах даних. Для створення тествої бази даних, було використанно SQL Server Object Explorer, і рядок підключення “Server=(localdb)\\mssqllocaldb;Database=ZUNU;Trusted_Connection=True;”. Щоб вести зміни, або створити нові таблиці у базі даних потрібно лише ввести у консольному вікні “Update-Database”.

Самі таблиці прописуються у файлі DataBase, із допомогою елементів DBSet. Він використовується для створення таблиць.Сам DataBase повинен наслідуватись від класу Context, який і відповідає за підключення до бази даних.

Зв’язки прописуються у функції OnModelCreatind. Прописуються із допомогою команди Entity у ній ми вказуєм, яку саме сутність використовуєм, потім прописуєм тип з’єднання. Також е конструкторі даного файлу передаєтья стрічка підключення із файлу StartUp проекту ZUNU, розберем його детальніше у наступних пукнтах. На рисунку 3.6 показано скріншот коду файлу DataBase.

```
namespace DomainLayer.Data
{
    public class DataBase : DbContext
    {
        public DbSet<Campus> Campuses { get; set; }
        public DbSet<Group> Groups { get; set; }
        public DbSet<Lesson> Lessons { get; set; }
        public DbSet<SchaduleDay> SchaduleDays { get; set; }
        public DbSet<Student> Students { get; set; }
        public DbSet<Teacher> Teachers { get; set; }
        public DbSet<User> Users { get; set; }
        public DataBase(DbContextOptions<DataBase> options) : base(options)
        {

        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<User>()
                 .HasOne(u => u.Student)
                 .WithOne(s => s.User)
                 .HasForeignKey<Student>(u => u.UserId);
            modelBuilder.Entity<Student>()
                .HasOne(s => s.Group)
                .WithMany(g => g.Students)
                .HasForeignKey(g => g.GroupId);
            modelBuilder.Entity<SchaduleDay>()
                .HasOne(sd => sd.Group)
                .WithMany(g => g.SchaduleDays)
                .HasForeignKey(sd => sd.GroupId);
            modelBuilder.Entity<Lesson>()
                .HasOne(l => l.SchaduleDay)
                .WithMany(sd => sd.Lessons)
                .HasForeignKey(l => l.SchaduleDayId);
            modelBuilder.Entity<Lesson>()
                .HasOne(l => l.Teacher)
                .WithMany(t => t.Lessons)
                .HasForeignKey(l => l.TeacherId);
        }
    }
}
```

Рисунок 3.6 – Код файлу DataBase

Наступним етап розробки є UnitOfWork файл. UnitOfWork це патерн програмування для групування однієї або кількох операцій (зазвичай це операції CRUD бази даних) в одну транзакцію або «одиницю роботи», щоб усі операції проходили або не виконувалися як одна одиниця. Простими словами, ми можемо сказати, що для конкретної дії користувача, скажімо, бронювання на веб-сайті, усі транзакції, такі як вставка/оновлення/видалення тощо, виконуються в одній транзакції, а не в кількох транзакціях бази даних. Це означає, що одна одиниця роботи тут включає операції вставки/оновлення/видалення, все в одній транзакції,

так що всі операції або проходять, або не виконуються як одна одиниця. У конструктор передаєм DataBase, який відповідає за базу даних. UnitOfWork містити дані про всі методи таблиць, які ми маєм і метод для асихроного зберігання. Він наслідується від інтерфейсу IUnitOfWork. Методи які містять у IUnitOfWork показано на рисунку 3.7.

```
namespace DomainLayer.Interfaces
{
    public interface IUnitOfWork
    {
        IRepositoryCampus RepositoryCampus { get; }
        IRepositoryGroup RepositoryGroup { get; }
        IRepositorySchaduleDay RepositorySchaduleDay { get; }
        IRepositoryLesson RepositoryLesson { get; }
        IRepositoryStudent RepositoryStudent { get; }
        IRepositoryTeacher RepositoryTeacher { get; }
        IRepositoryUser RepositoryUser { get; }
        Task SaveAsync();
    }
}
```

Рисунок 3.7 – Код файлу IUnitOfWork

У папці Entites містяться усі наші таблиці, сама структура бази даних. Детальний опис бази даних у розділі 2.3.

У папці Repositories містять основні методи для роботи з базою даних такі, як:

- отримати усі елементи таблиці (GetAllAsync);
- отримати елемент таблиці з відповідним Id (GetByIdAsync);
- створити новий елемент у таблиці (AddAsync);
- видалити елемент у таблиці (Delete);
- видалити елемент по Id з таблиці (DeleteByIdAsync);
- оновлення елементу в таблиці (Update);

Ці методи містяться у всіх таблицях, ще є додатковий метод для деяких таблиць отримання елемнт або елменти з таблиці, але з інформації про елемнти, які містять у таблицях, що містять зв’язки з ними.

Назви, що містять приставку Async є асинхрониими. Запити виконуються за допомогою LINQ. LINQ це потужний набір технологій, заснований на інтеграції можливостей запитів безпосередньо в мову C#. Запити LINQ — це першокласна мовна конструкція в C# .NET, як і класи, методи, події. LINQ забезпечує послідовне виконання запитів до об’єктів (LINQ to Objects), реляційних баз даних (LINQ to SQL) і XML (LINQ to XML).(1)

Тепер розглянем ServiceLayer. Даний шар виконує основні сервіси при роботі з базою, також у ньому прописаний персональний помилковий менеджер і маппер. На рисунку 3.8 зображенно файлову систему ServiceLayer.

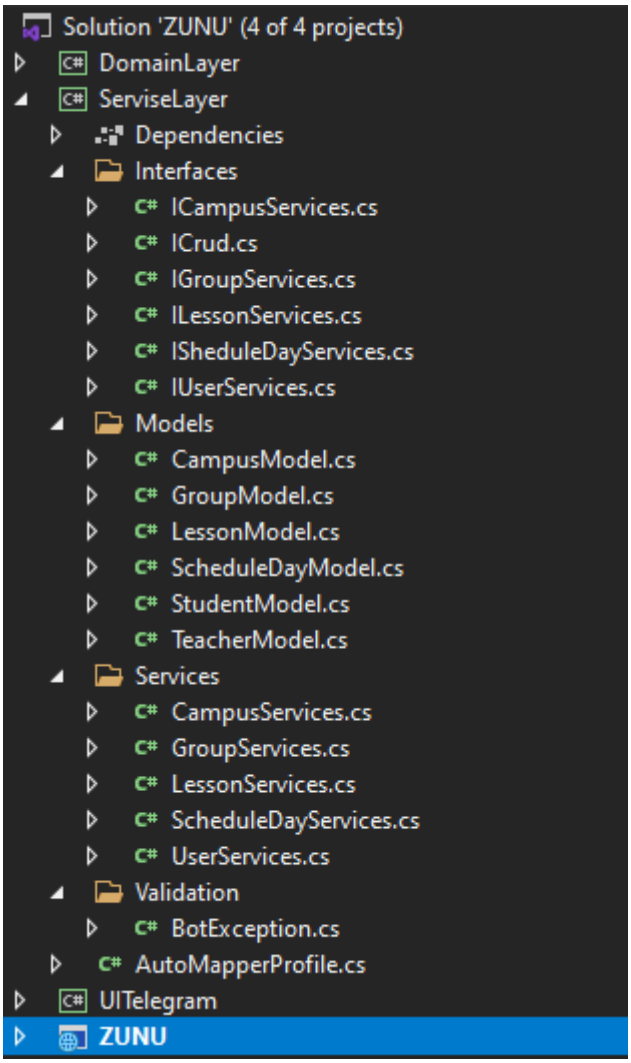

Рисунок 3.8 – Файлова система ServiceLayer

У папці interfaces містять інтетрфейси для методів, що потрібні у нашому проекті. Розглянемо детальніше кожен із них:

- ICrud – базовий інтерфейс для кожного інтерфейсу, містить такі методи, як: отримати всі елементи, отримати елемент по унікальному індефікатору, оновити елемент, додати елемент, видалити елемент.

- ILessonServices – інтерфейс містить такі як: методи від інтерфейсу ICrud, пошук парів за викладачем, пошук парів за днем, пошук парів за групою.

- IUserServices – містить методи від ICrud, ще містятить методи для зміни створення нових елементів користувача.

- ICampusServices – містить методи від ICrud, ще містить методи: пошуку по абревіатурі корпусу, по імені, по номеру.

У папці Services ми реалізуєм вище описані інтерфейси.

Наступним етапом буде опис папки Validation, ця папка містить лише один клас BotException. Цей клас використовується для провірки чи правильні дані вносяться у базу даних, якщо ні то видається відповідні помилки. Наприклад, пошта викладача не відповідає формату text@text.text, довжина викладача не відповідає дійсності, не існує такий день.

AutoMapperProfile файл для створення мапінгу між DomainLayer і ServiceLayer.

Тепер приступим до третього розділу UITelegram, даний розід відповідає за відправку повідомлень користувачу, обробку інформацію, що відправляє користувач, додавання нового користувача у базу даних, обробку запитів, що відправляє користувач. На рисунку 3.9 зображенно файлову систему підпроекту UITelegram.

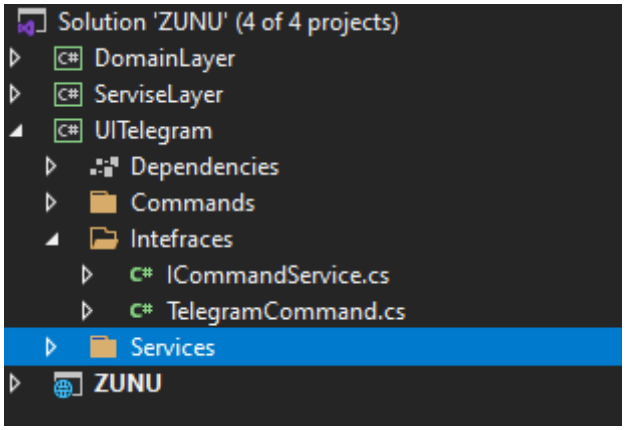

Рисунок 3.9 – Файлова система UITelegram

Даний підпроект містить три папки:

- Commands;
- Interfaces;
- Services.

У папці інтерфейс міститься два файла: TelegramCommand і ICommandService. TelegramCommand це абстрактний клас, що містить два абстракних методи і абстрактне поле. Абстаркний клас це клас, що може містити абстрактні методи і обичні. Від абстракного класу не можуть бути створенні об’єкти, він може використовуватись лише для наслідування. Абстрактний метод це метод, що не містить ніякої реалізації і він повинен бути реалізовний у дочірньому елементі. На рисунку 3.10 показано абстрактний клас TelegramCommand.

```
namespace UITelegram.Intefraces
{
    public abstract class TelegramCommand
    {
        public abstract string Name { get; }

        public abstract Task Execute(Message message, ITelegramBotClient client);

        public abstract bool Contains(Message message);

    }
}
```

Рисунок 3.10 – Абстрактний клас TelegramCommand

Розберем детальніше ці методи і за що вони відповідають.

- Name – тут зберігається назва команди цієї функції у телеграм;
- Contains – метод провіряє чи текст що відправив користувач сходиться з назвої команди відповідного метода;
- Execute - метод містить логіку для відповідної команди з телеграм.

У папці Services лише один файл, CommandService він наслідується від інтерфейсу ICommandService, у ньому міститься перелік всі можливих команд і по

черзі він нам їх надає для провірки чи відповідає те, що відправив користувач, якійсь команді.

У папці Commands зберігаються файли для опрацювання команд, розглянемо одну із них, візьмемо StartCommand. Рисунок 3.11

```
public class StartCommand : TelegramCommand
{
    public override string Name => @"/start";

    public override bool Contains(Message message)
    {
        if (message.Type != MessageType.Text)
            return false;

        return message.Text.Contains(Name);
    }
    public override async Task Execute(Message message, ITelegramBotClient client)
    {
        var chatId = message.Chat.Id;
        var keyBoard = new InlineKeyboardMarkup
        (
            new[]
            {
                new[]
                {
                    InlineKeyboardButton.WithCallbackData("Корпуси","/corpuses")
                },
                new[]
                {
                    InlineKeyboardButton.WithCallbackData("Виберіть свою групу", "/setgroup")
                },
                new []
                {
                    InlineKeyboardButton.WithCallbackData("Пошук по викладачам", "/teachers")
                },
                new []
                {
                    InlineKeyboardButton.WithCallbackData("Мої пари", "/lessons_my")
                }
            }
        ) ;
        await client.SendTextMessageAsync(chatId, "Чат-бот фкіт",
            parseMode: ParseMode.Markdown, replyMarkup: keyBoard);
    }
}
```

Рисунок 3.11 – Клас StartCommand

Name – назва команди. У методі Contains відбувається провірка на те, які дані відправив користувач. У методі Execute відправляєм для користувача, головне меню проекту. Для відправки повідомлень використовується метод SendTextMessageAsync, перша зміна у ньому це унікальний індефікатор чату, другий це повідомелння яке отримає користувач, третій це який стиль кнопки і

останнє що ми передаєм, це тип клавіатури що буде користувач для обирання наступної дії.

Останній підпроект це ZUNU. На рисунку 3.12 зображенно файлову систему для даного підпроеку.

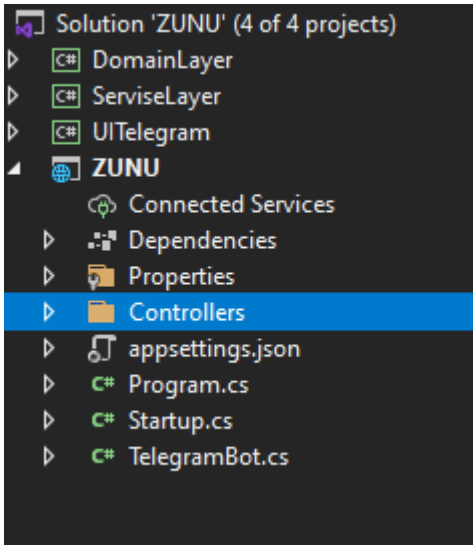

Рисунок 3.12 – Файлова система ZUNU

У папці Controllers зберігаються наші запити, і шяхи до них. Всі контроллери насдіюються від класу BaseController. Файл appsetings.json містити наші тимчасові дані такі, як:

- Силка на підключення до програми телеграм;
- Стрічка на підключення до бази даних;
- Персональни токен для чат бота у телеграм.

Program.cs є стартовим у нашому проекті він запускає StartUp.cs. У StartUp.cs прописані наші виклики основних сервісів. У ньому є контсруткор два методи і одне поле. Поле потрібне для отримання конфігурацій з фалу appsettings.json, у конструкторі ми передаєм саму інфомацію для заповнення файлу. Метод ConfigureServices Цей метод викликається середою виконання. Використовуйться цей метод, щоб додавати служби до контейнера. Пешим що добавляєм це кнтролери, які містять запити. За допомогою команки AddConrollers, а якщо бути точним воно додає всі класи що наслудіються від

класу BaseController, або Contoller. Також є три оснвних способа додавання сервісів, це:

- Singelton - контейнер створить і надасть спільний доступ до єдиного екземпляра служби протягом усього життя програми;

- Transient - контейнер створюватиме новий екземпляр указаного типу служби кожного разу, коли ви запитаєте про це;

- Scoped - контейнер створюватиме екземпляр указаного типу служби один раз на запит і буде надано спільний доступ в одному запиті.

У нашому випадку для реалізації функцій ідеально підходить Scoped, його і використаєм для впровадження сервісів із ServiceLayer і UITelegram. За допомогою метода AddDbContext впроваджуєм нашу бадинх і передаєм їй у конструктор нашу стрічку для підключення до бази даних. Для впровадження Swagger використаєм команду AddSwaggerGen. Для впровадження телеграм бот сервера і створення зв'язку між телергам ботом і нами використаєм AddTelegramBotClient більш детально розгленемо, як це відбувається дальше.

У даному файлі є ще один метод Configure. Цей метод викликається середою виконання. Використовується цей метод для налаштування конвеєра запиту HTTP.

Файл TelegramBot.cs містить сам код для створення зв'язку між ngrok і проектом. Код TelegramBot.cs зображенно на рисунку 3.13.

```
namespace ZUNU
{
    public static class TelegramBot
    {
        public static IServiceCollection AddTelegramBotClient(this IServiceCollection serviceCollection,
            IConfiguration configuration)
        {
            var client = new TelegramBotClient(configuration["Token"]);
            var webHook = $"{configuration["Url"]}api/bot";
            client.SetWebhookAsync(webHook).Wait();

            return serviceCollection
                .AddTransient<ITelegramBotClient>(x => client);
        }
    }
}
```

Рисунок 3.13 – Код TelegramBot

У цьому файлі ми створює телеграм клієнт з відповідним токен, наступним етапом створимо WebHook відповідно до силки у ngrok. І створюєм метод для файлу StartUp.

У даному розділі дипломної роботи було створено, усі підпроекти дипломної роботи, реалізована архітектура і детально розібраний кожен із рівнів.

3.4 Розгортання серверу і бази даних на сервісах Azure

У даному розділі ми розмістим базу даних і наш сервер на сервах Azure. Для початку створим групу ресурсів, яка буде містити у собі ті сервіси, які потрібні для проекту. Наступним етапом створим для неї пусту базу даних яку ми заповним за допомогою міграцій. Під час створення бази даних ми повинні обрати у якій групі ресурсів вона буде зберігатись, локацію де вона буде працювати, кількість ядер і кількість пам’яті і чи потрібне резевне сховище, назву бази даних. На рисунку 3.14 зображено параметри для створення пустої бази даних на серверах Azure.

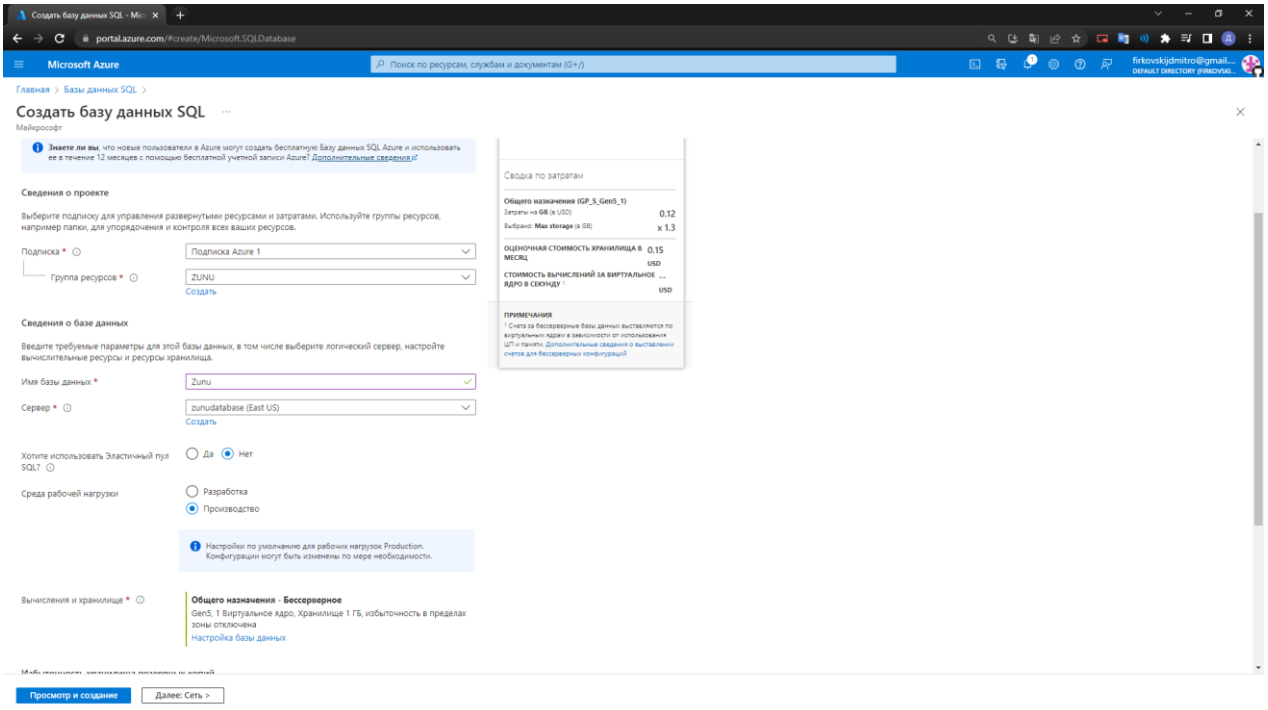

Рисунок 3.14 – Створення бази даних на Azure

У базі даних на Azure знаходим вкладку настройки у ній стрічки підключення. Зараз це “Server=tcp:zunudatabase.database.windows.net,1433;Initial Catalog=ZUNUDataBase;Persist Security Info=False;User ID=CloudSA2d7fc7ad; Password={your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;”. Використаєм її для підключення нашого проекту до бази даних Azure. Добавим її у json файл що є в підпроекті ZUNU. За допомогою команди Update-Database внесем у базу даних Azure наші таблиці.

Протестуєм чи працює база даних, відправивши на неї дані про корпус за допомогою Swagger. Дані для провірки: numberCamus:6, name “Facultet Computer Information Technolegy”, latitude: 49.56082036746616, longitude: 25.596077303403288, address : “вулиця Олени Теліги, 8, Тернопіль, Тернопільська область, 46000”, abbreviature : “FCIT”. На рисунку 3.15 тестування базового запиту.

Настпуним етапом розгорнем на серверах Azure сервер дипломної роботи, для цього натиснем на ZUNU підпроект правою кнопкою миші і виберем вкладку “Publish”.

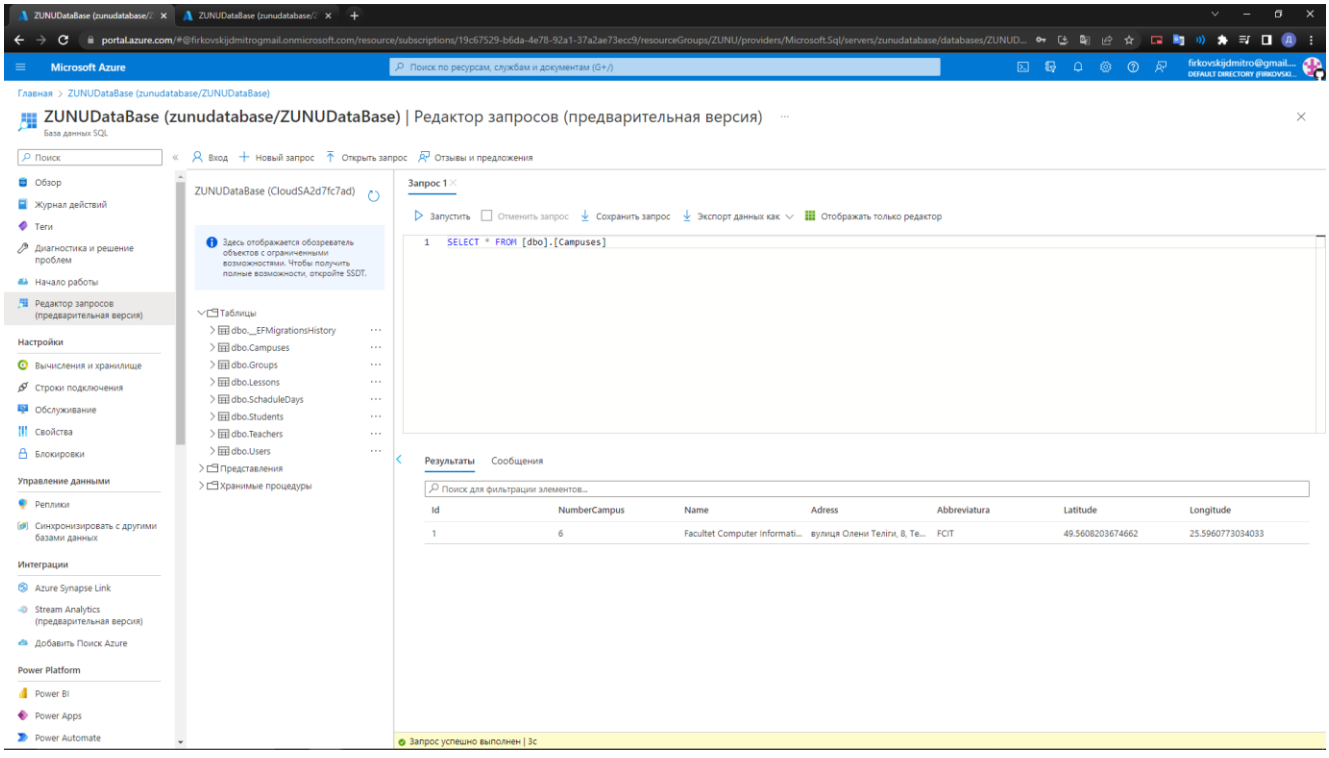

Рисунок 3.15 – Тестування бази даних

У вікні “Publish” оберем на, якому саме сервіси буде зберігатись проект, обираєм Azure. Після цього обираєм, який саме сервіс потрібен Azure App Service Windows. Після цього ми побачим вікно “Publish”. Після натискання плюса, ми можем обрати:

- Локацію серверу;
- Кількість виділеної пам’яті;
- Кількість ядер;
- Хостинг план;
- Ресурс група.

Останнім етапом є “API menagement” ми можем підключити наш сервер до певного API, цей етап можна пропустити. Після цього бачим сторінку, про те де буде зберігатись наш проект, ця сторінка зображена на рисунку 3.16.

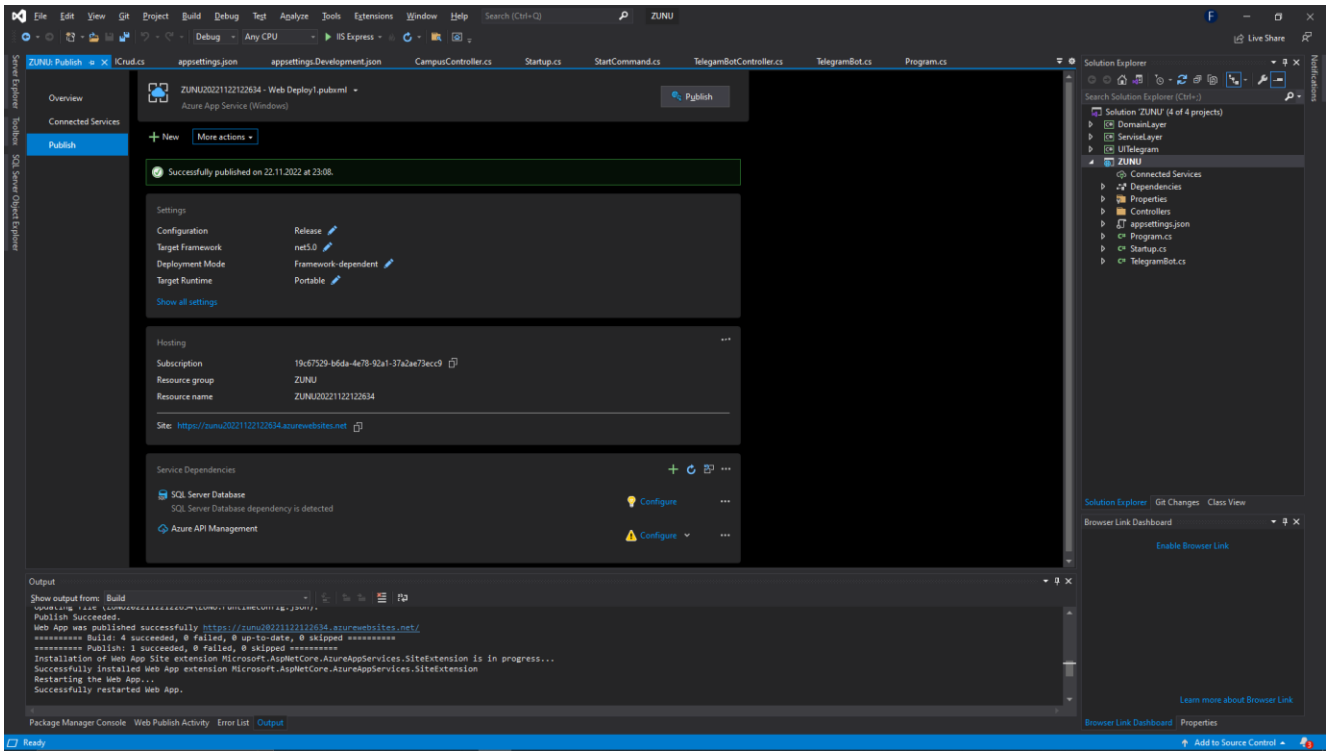

Рисунок 3.16 – Дані про публікацію проекту

Сервер нашого проекту тепер знаходиться по силці https://zunu20221122122634.azurewebsites.net, тепер для того щоб внести зміни на

сервер нам потрібно нажати лише кнопку “Publish” або вибрати функцію автоматичого оновлення при внесенні змін на github проект.

Останнім етапом розгортання нашого проекту буде створення WebHook зв’язку серверу з Telegram. Для цього використаєм консоль вікно ngrok, виконаєм команду для створення зв’язку “ngrok http https://zunu20221122122634 .azurewebsites.net”. На рисунку 3.17 поканно результат роботи ngrok.

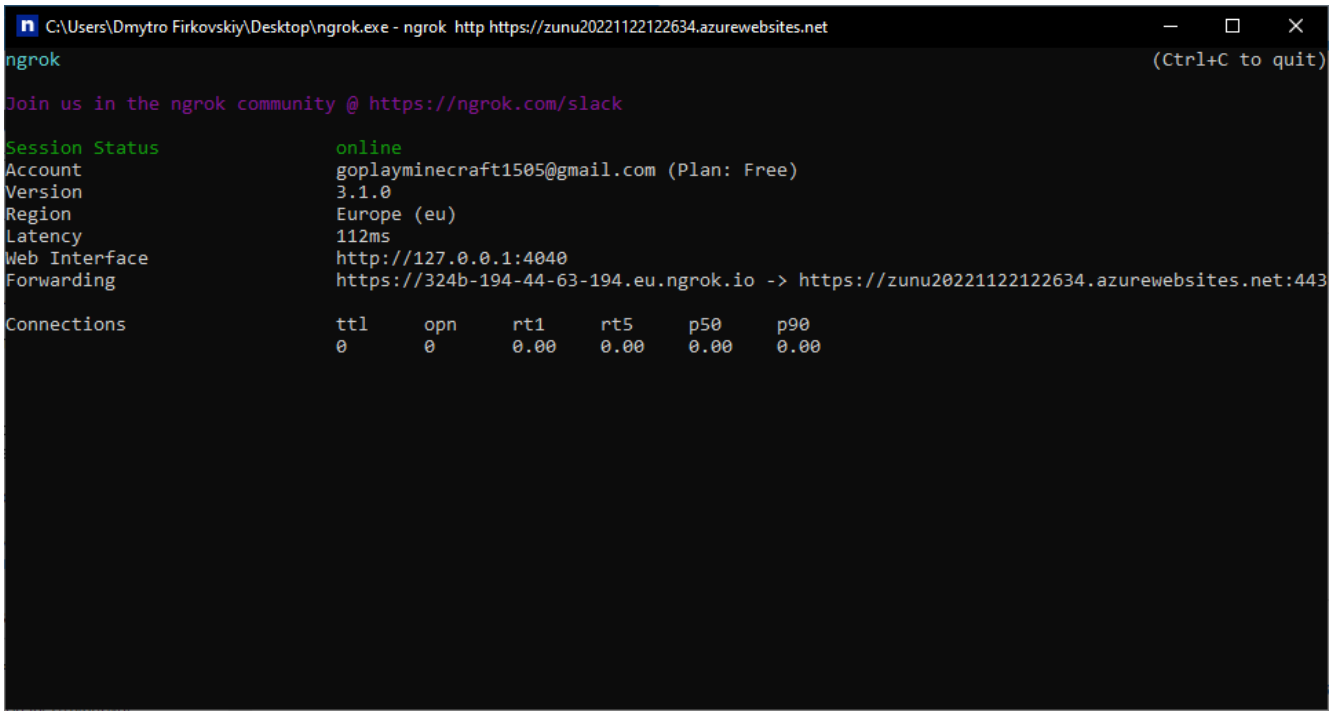

Рисунок 3.16 – Результат роботи Ngrok

Тепер добавим силку що видав нам Ngrok, а саме “https://324b-194-44-63-194.eu.ngrok.io”, її потрібно добавити у підпроект ZUNU файл appseting.json у поле URL, і опублікуєм проект повторно для того, щоб оновити дані в проекті. Виконаєм команду у пошуковим вікні будь якого бразера “https://api.telegram.org/bot5576744910:AAFgmEKHNoJxPsyUu0CiL9ACTg_l2_TMEWo/setwebhook?url= https://324b-194-44-63-194.eu.ngrok.io /api/bot” , де ми вказуєм токен нашого телеграм бота, дальше вказуєм тип з’єданання у даному випадку setwebhook, наступним рядком є сама стрічка яку нам повертає ngrok для стоврення зв’язку.

У даному розділі було проведенно публікацію бази даних і серверу на сервер Azure. Проведенно зв’язок між ngrok і нашим Azure сервером. Проведенно

зв’язок між отриманим Ngrok webhook з’єданням і телеграм ботом. Було добавленно Azure базу даних до даного проекту.

3.5 Тестування телеграм бота і серверу.

У даному розділі буде проведенно тестування основних команд телеграм бота і тестування запитів серверу. Тестування буде проходити з мобільного пристрою.

Перш за все проведем команду /start воно повинна викликати головне меню. На рисунку 3.17 зображенно скріншоти роботи команди.

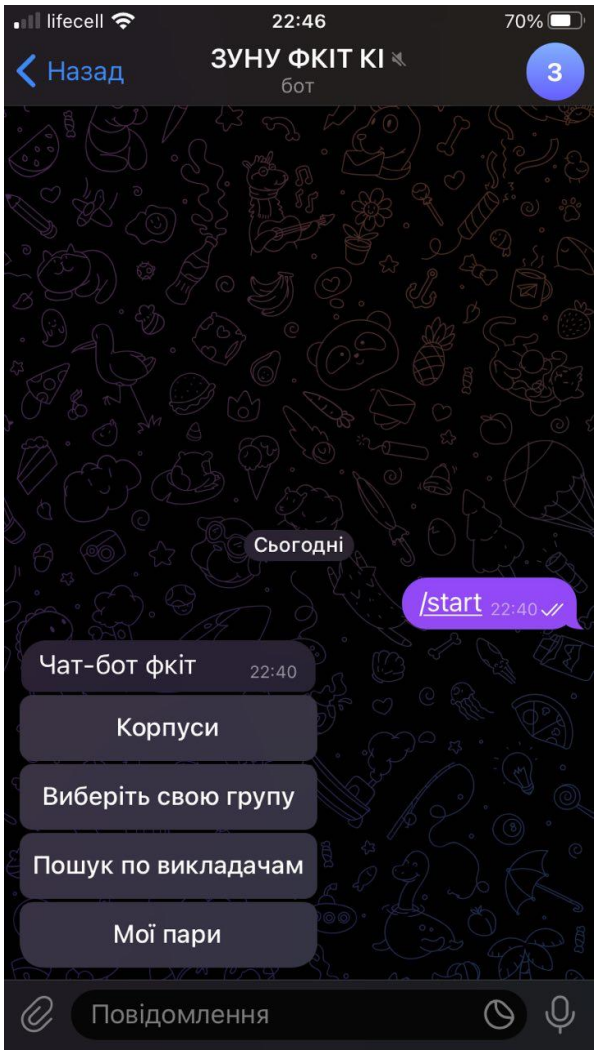

Рисунок 3.17 – Скріншот роботи команди /start

Наступним етапом протестуєм вікно корпуси воно відповідає за відображення всіх корпусів для користувача.також у ній можна вибрати корпус і він покажеться на карті. Корпуси вкладку можна викликати за допомогою команди /corpuses або натиснувши на відповідну кнопку в головному меню. Результат тестування зображення на рисунку 3.18

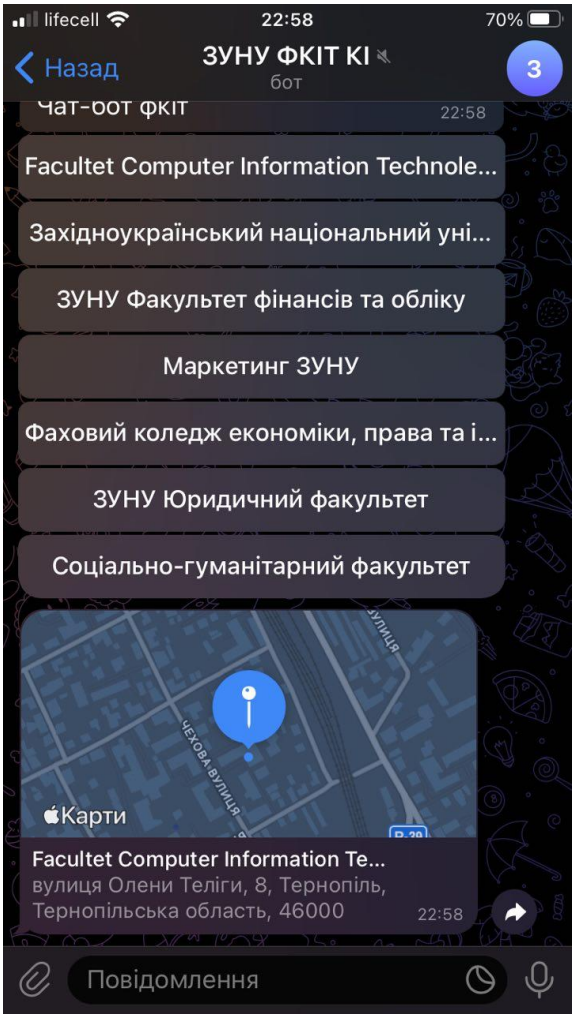

Рисунок 3.18 – Скріншот роботи команди /corpuses

Одним із дуже важливим етапом є оповіщення студента про початок пари. Протестуєм систему роботи сповіщень, для цього відправим студентам групи про початок пари. На рисунку 3.19 скріншот тестування системи оповіщень.

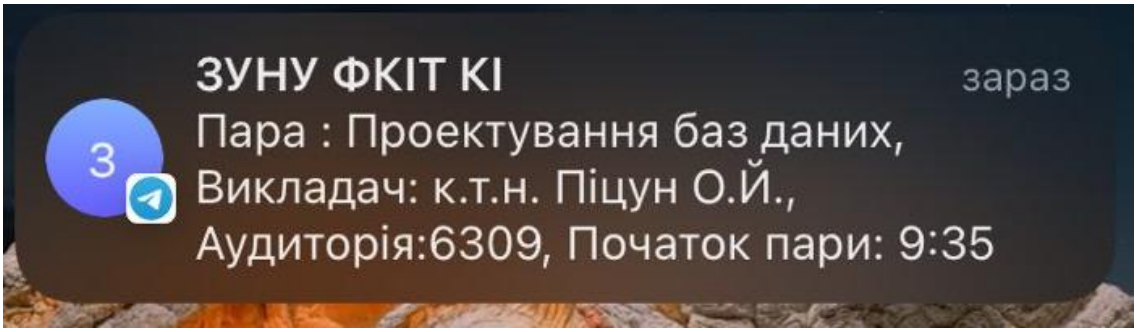

Рисунок 3.19 – Тестування системи оповіщень

Проведем тестування швидкості реагування додатку на запити для цього скористаємся ngrok. Він містить дані про швидкість відповіді на запит. Рисунок 3.20 тестування швидкості реагування запитів. Середня швидкість реагування серверу на запит 0.35с це є дуже хорошим результатом.

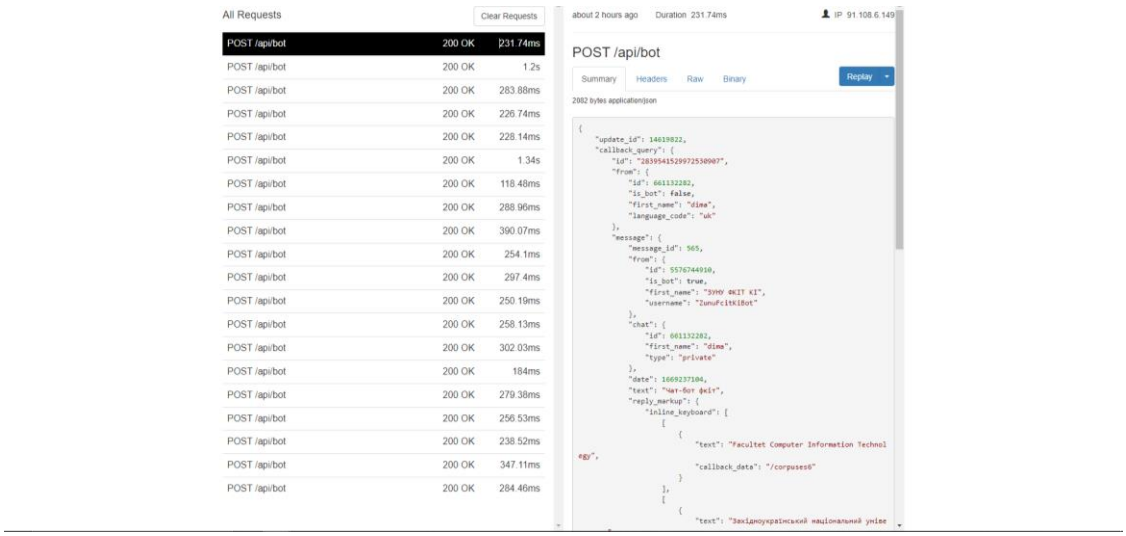

Рисунок 3.20 – Швидкість реагування серверу на запити

Наступним етапом протестуєм базу даних на наявність оновлень і відправлених нами даних на серверах Azure (рисунок 3.21).

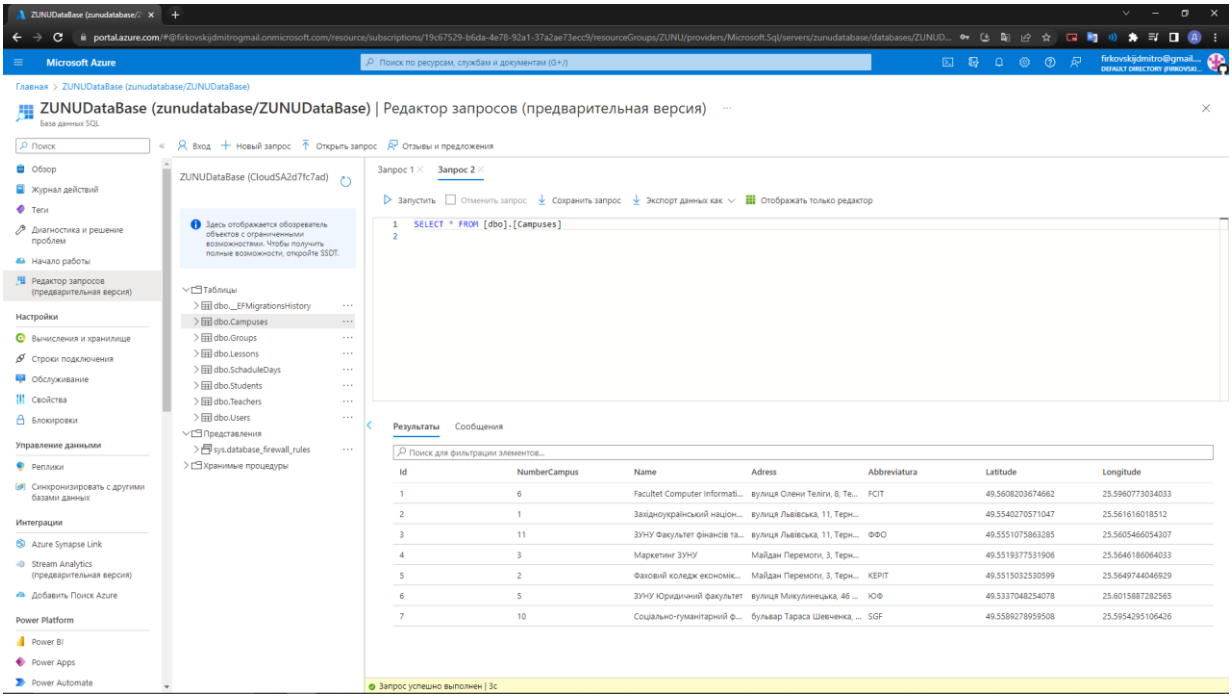

Рисунок 3.21 – Провірка бази даних на наявність оновлень

У даному розділі було провденно тестування швидкості запити і швидкості реагування серверу, протестованно користувацький інтерфейс користувача, протестованно базу даних на наявність оновлень.

3.6 Висновок до розділу

Описано програмне середовище для розробки чат-боту. Розроблено й описано діаграму варіантів використання програмної системи. Розроблено багаторівневий проект чат боту. Розгорнуто сервер і бази даних на сервісах azure. Створено тунель між розгорнутим сервером і телеграм. Проведено тестування користувацького інтерфейсу, бази даних, оповіщень чат боту. Реалізовано сам чат бот.

ВИСНОВКИ

1. Проведено аналіз та класифікацію чат-ботів на основі даного аналізу досліджено серверні платформи та проведено порівняльний аналіз месенджерів;
2. Досліджено засоби хмарних сховищ та встановлено постановку задачі.
3. Було визначено основні вимоги до чат-боту.
4. Охарактеризовано алгоритми парсингу даних з файлу з розкладом у базу даних.
5. Проаналізовано і побудувано архітектуру проекту та схематичну базу даних.
6. Розроблено структуру програмного модуля для чат-боту, досліджено середовище розробки, на основі вище наведених даних розроблено багаторівневий проект чат-боту.
7. Проведено тестування користувацького інтерфейсу, бази даних, серверу, тунелю, що з’єднує сервех і телеграм бот.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абстрактні методи: веб-сайт. URL: http://uk.swewe.net/word_show.htm/?51000_1&%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%96_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B8 (дата звернення: 25.10.2022).

2. Архітектура та проектування програмного забезпечення: веб-сайт. URL: https://learn.ztu.edu.ua/mod/book/tool/print/index.php?id=278 (дата звернення: 25.10.2022).

3. Диференціальні рівняння як математична модель реальних процесів: веб-сайт. URL: https://naurok.com.ua/diferencialni-rivnyannya-yak-matematichna-model-realnih-procesiv-109415.html (дата звернення: 25.10.2022).

4. Діаграма : веб-сайт. URL: https://stud.com.ua/35679/informatika/df-diagrama (дата звернення: 25.10.2022).

5. Дослідження процесів міграції баз даних в корпоративних системах: веб-сайт URL:

https://openarchive.nure.ua/bitstream/document/10888/1/2019_M_ST_Koshova_A_A.pdf (дата звернення: 25.10.2022).

6. IT у освіті: веб-сайт. URL: https://mon.gov.ua/storage/app/media/zagalna (дата звернення: 25.10.2022).

7. Керування базами даних: веб-сайт. URL: https://pidru4niki.com/81326/tehnika/sistemi_keruvannya_bazami_danih (дата звернення: 25.10.2022).

8. Кешування даних: веб-сайт. URL: http://yoip.com.ua/keshuvannya-danih/ (дата звернення: 25.10.2022).

9. Методичні вказівки до оформлення курсових, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп'ютерна інженерія» / І.В. Гураль, Л.О.Дубчак / під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 34 с.

10. Методичні вказівки ЗНУ: веб-сайт. URL: https://moodle.znu.edu.ua/pluginfile.php?file=/632726/mod_resource/content/1/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BA%D0%B0.docx (дата звернення: 25.10.2022).

11. Методичні рекомендації до виконання дипломної роботи з освітньокваліфікаційного рівня «Магістр». Спеціальність «Комп'ютерна інженерія» / О.М. Березький, Л.О. Дубчак. Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2020. 47 с.

12. Моделі і методи проектування інформаційних систем: веб-сайт. URL: http://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/183252/index.html (дата звернення: 25.10.2022).

13. Основи програмування Java: веб-сайт. URL: http://iwanoff.inf.ua/java_ua/LabTraining03.html (дата звернення: 25.10.2022).

14. Послуги центрів обробки даних: веб-сайт. URL: https://new.siemens.com/ua/uk/markets/tsentry-obrobky-danykh/servis.html (дата звернення: 25.10.2022).

15. Побудова архітектури та структури ПЗ: веб-сайт. URL: https://studfile.net/preview/5200675/page:21/ (дата звернення: 25.10.2022).

16. Платформи корпоративних інформаційних систем: веб-сайт. URL: https://evnuir.vnu.edu.ua/bitstream/123456789/17724/3/net_metod.pdf (дата звернення: 25.10.2022).

17. Реляційні бази даних, їхні об’єкти. Системи управління базами даних: веб-сайт. URL: https://tokmakgroup.blogspot.com/p/blog-page_13.html?m=0 (дата звернення: 25.10.2022).

18. Розмежування доступу та аудит в операційній системі LINUX: веб-сайт. URL: https://ppt-online.org/819104 (дата звернення: 25.10.2022).

19. 11 типів сучасних баз даних: короткий опис, схеми і приклади БД: веб-сайт. https://senior.ua/articles/11-tipv-suchasnih-baz-danih-korotkiy-opis-shemi--prikladi-bd (дата звернення: 25.10.2022).

20. Форми подання інформації: веб-сайт. URL: https://stud.com.ua/50115/informatika/formi_podannya_informatsiyi (дата звернення: 25.10.2022).

21. Чат-бот, що це: веб-сайт. URL: https://creativesmm.com.ua/shho-take-chatbot-ta-komu-vonu-potribni/ (дата звернення: 25.10.2022).

22. Що таке бази даних, їх призначення та види?: веб-сайт. URL: https://futurenow.com.ua/shho-take-bazy-danyh-yih-pryznachennya-ta-vydy/ (дата звернення: 25.10.2022).

23. Що таке Microsoft Azure?: веб-сайт. URL: https://test.devzone.org.ua/post/korotkii-poglyad-u-svit-blakiti (дата звернення: 25.10.2022).

24. Як переглянути зміни, які ви додали на Карти Google: веб-сайт. URL: https://support.google.com/maps/answer/7055486?hl=uk&co=GENIE.Platform=Android (дата звернення: 25.10.2022).

25. Як перемикатися між додатками: веб-сайт. URL: https://www.telusuri.info/articles/windows-phone/kak-pereklyuchatsya-mezhdu-prilozheniyami-v-windows-phone.html (дата звернення: 25.10.2022).

26. Computing Machinery and Intelligence. / A. M. Turing. In: Mind 49, 1950, 433–460 с.

27. Chatbots.org. Consumers Say No to Chatbot Silos in US and UK Survey. 2017: веб-сайт. URL: https://www.chatbots.org/images/news/chatbot_survey_2018.pdf (дата звернення: 25.10.2022).

28. C. Olah. Understanding LSTM Networks. Aug. 2015: веб-сайт. URL: http://colah. github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення: 25.10.2022).

29. Data Base: веб-сайт. URL: http://bukvar.su/informatika_programmirovanie/page,3,175014-Relyacionnye-Bazy-Dannyh-SQL-standartnyiy-yazyk-relyacionnyh-baz-dannyh.html (дата звернення: 25.10.2022).

30. Drift. The 2018 State of Chatbots Report. Jan. 2018: веб-сайт. URL: https://blog. drift.com/chatbots-report (дата звернення: 25.10.2022).

31. ELIZA—a computer program for the study of natural language communication between man and machine. / J. Weizenbaum. In: Communications of the ACM 9, Jan. 1966, 36–45 с.

32. Entity Framework NuGet: веб-сайт. URL: https://www.educba.com/entity-framework-nuget/ (дата звернення: 25.10.2022).

33. Entity Framework Core: веб-сайт. URL: https://learn.microsoft.com/ru-ru/ef/core/get-started/overview/install (дата звернення: 25.10.2022).

34. Grid і бази даних: веб-сайт. URL: https://shpora.me/mike/db2 (дата звернення: 25.10.2022).

35. Hosting: веб-сайт. URL: https://coggle.it/diagram/XdJiT4umpH1LlRLz/t/%D1%85%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3-1-veb-hosting (дата звернення: 25.10.2022).

36. iGen: веб-сайт. URL: https://www.radiosvoboda.org/a/28705680.html (дата звернення: 25.10.2022).

37. Information technologies as a factor of social transformation of society: веб-сайт. URL: http://www.dy.nayka.com.ua/?op=1&z=629 (дата звернення: 25.10.2022).

38. I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to Sequence Learning with Neural Networks”. In: CoRR abs/1409.3215 (2014). arXiv: 1409.3215: веб-сайт. URL: http://arxiv.org/abs/1409.3215 (дата звернення: 25.10.2022).

39. J. Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: CoRR abs/1412.3555 (2014). arXiv: 1412.3555: веб-сайт. URL: http://arxiv.org/abs/1412.355 (дата звернення: 25.10.2022).

40. J. Pennington, R. Socher, and C. D. Manning. “GloVe: Global Vectors for Word Representation”. In: Empirical Methods in Natural Language Processing (EMNLP). 2014, pp. 1532–1543: веб-сайт. URL: http://www.aclweb.org/anthology/D14-1162 (дата звернення: 25.10.2022).

41. K. Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: CoRR abs/1406.1078 (2014). arXiv: 1406.1078: веб-сайт. URL: http://arxiv.org/abs/1406.1078 (дата звернення: 25.10.2022).

42. Manage accounts, properties, and users : веб-сайт. URL: https://support.google.com/analytics/answer/1009702?hl=uk (дата звернення: 25.10.2022).

43. O. Vinyals et al. “Show and Tell: A Neural Image Caption Generator”. In: CoRR abs/1411.4555 (2014). arXiv: 1411.4555: веб-сайт. URL: http://arxiv.org/abs/1411.4555 (дата звернення: 25.10.2022).

44. R. Pascanu, T. Mikolov, and Y. Bengio. “Understanding the exploding gradient problem”. In: CoRR abs/1211.5063 (2012). arXiv: 1211 . 5063: веб-сайт. URL: http : //arxiv.org/abs/1211.5063 (дата звернення: 25.10.2022).

45. S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: Neural Comput. 9.8 (Nov. 1997), pp. 1735–1780. issn: 0899-7667. doi: 10.1162/neco.1997.9.8.1735: веб-сайт. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735 (дата звернення: 25.10.2022).

46. SQL: веб-сайт. URL: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/8868/1/sql.pdf (дата звернення: 25.10.2022).

47. SQL Developer: веб-сайт. URL: http:// https://www.oracle.com/database/sqldeveloper/ (дата звернення: 25.10.2022).

48. T. Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: CoRR abs/1301.3781 (2013). arXiv: 1301.3781: веб-сайт. URL: http:// arxiv.org/abs/1301.3781 (дата звернення: 25.10.2022).

49. Trends : веб-сайт. URL: https://www.prostir.ua/?kb=zroblena-domashka-znannya-svojih-sylnyh-storin-i-novyh-trendiv-podaruyut-nuo-uspih-v-novomu-rotsi (дата звернення: 25.10.2022).

50. Why default Azure database connection string has Pooling=False : веб-сайт. URL: https://stackoverflow.com/questions/37714753/why-default-azure-database-connection-string-has-pooling-false (дата звернення: 25.10.2022).