**ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра автоматики та управління в технічних системах**

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Комп'ютеризовані системи управління»**
**спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»**
**на тему: «Телеграм-бот для інформаційної підтримки навчального процесу»**

Виконав:
студент IV курсу, групи ІА-62
Комарський Олександр Сергійович _____

Керівник:
старший викладач кафедри АУТС
Яланецький Валерій Анатолійович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра автоматики та управління в технічних системах**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

# ЗАВДАННЯ
## на дипломний проєкт студенту
## Комарському Олександру Сергійовичу

1. Тема проєкту «Телеграм-бот для інформаційної підтримки навчального процесу», керівник проєкту Яланецький Валерій Анатолійович старший викладач, затверджені наказом по університету від «07» травня 2020р. №1081-с

2. Термін подання студентом проєкту 9 червня 2020 року

3. Вихідні дані до проєкту: Мова програмування Python , середовище програмування PyCharm, обрана для розробки технологія – Django, СУБД – PostgreSQL, платформа Telegram.

4. Зміст пояснювальної записки 1. Вступ 2. Аналіз вимог до системи 3. Огляд існуючих рішень 4. Вибір технологій для розробки 5. Розробка чат-бота 6. Інструкція користувача 7. Висновки.

5. Перелік графічного матеріалу діаграма структури бази даних, діаграма станів користувача, діаграма розгортання, діаграма варіантів використання

6. Дата видачі завдання 6 лютого 2020 року.

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1 | Аналіз теоретичних матеріалів та вивчення предметної області | 15.04.2020 р. | |
| 2 | Вибір технологій та засобів для реалізації задачі | 19.04.2020 р. | |
| 3 | Огляд існуючих рішень з тематики роботи | 27.04.2020 р. | |
| 4 | Реалізація проекту | 05.05.2020 р. | |
| 5 | Оформлення текстової документації | 08.06.2020 р. | |

Студент Олександр КОМАРСЬКИЙ

Керівник Валерій ЯЛАНЕЦЬКИЙ

# АНОТАЦІЯ

Комарський О.С. Телеграм-бот для інформаційної підтримки навчального процесу. КПІ ім. Ігоря Сікорського, Київ, 2020.

Пояснювальна записка містить 60 с. тексту, 12 рисунків, 2 додатки та 17 літературних джерел.

Ключові слова: Телеграм-бот, бот, Python, Django, чат-бот, Telegram.

Об’єктом розробки є телеграм-бот для інформаційної підтримки навчального процесу.

Мета розробки – підвищення зручності отримання інформації щодо навчального процесу та спрощення організації зв’язку між студентами та викладачами.

У дипломному проекті був розроблений Телеграм-бот для інформаційної підтримки навчального процесу у месенджері Telegram з використанням Telegram Bot API. Було проведено ретельний аналіз подібних технологій розробки, та аргументований вибір найбільш оптимальних для реалізації бота. Користування ботом було зроблено зручним та інтуїтивно зрозумілим, як для студентів так й для викладачів.

# SUMMARY

Komarskyi O.S. Telegram bot for informational support of educational process. Igor Sikorsky KPI, Kyiv, 2020.

The explanatory note contains 60 pages. text, 12 images, 2 additions and 17 literature sources.

Keywords: Telegram bot, bot, Python, Django, chat bot, Telegram.

The object of development is a telegram bot for information support of the educational process.

The purpose of the development is to increase the convenience of obtaining information about the educational process and to simplify the organization of communication between students and teachers.

The diploma project developed a Telegram bot for information support of the learning process in the Telegram messenger using the Telegram Bot API. A thorough analysis of such development technologies was carried out, and the choice of the most optimal for the implementation of the bot was reasoned. Using the bot was made convenient and intuitive for both students and teachers.

# Пояснювальна записка
## до дипломного проєкту
## на тему: «Телеграм-бот для інформаційної підтримки навчального процесу»

Київ – 2020 рік

**ЗМІСТ**

# ВСТУП

Підвищення ефективності навчального процесу завжди було актуальним завданням для вищих навчальних закладів. Одним із складових показників якості навчання, є організація взаємодії між викладачем та студентом. Взаємодія повинна відбуватись швидко та не створювати зайвих труднощів. Не менш важливим показником є зручність доступу студентів до методичного забезпечень та важливої інформації щодо навчання. Особливо актуальними ці аспекти навчального процесу стають в період проведення дистанційного навчання, коли взаємодія між викладачем та студентом переходить в режим онлайн.

В сучасному світі не знайти найбільш швидкого та зручного засобу комунікації людей в мережі інтернет, ніж месенджери. Немає жодного смартфона, на якому б не було встановлено хоча б одного з них. Однією з вагомих переваг месенджерів є чат-боти. Чат-бот – це помічник, який може спілкуватись з користувачем за допомогою повідомлень, давати відповіді на запитання та виконувати різні функції за командами які в нього вклав розробник.

Є досить багато засобів зв’язку між викладачем та студентом, але не всі вони ефективні та зручні, саме тому виникає необхідність у створенні системи, яка об’єднає у собі сучасні засоби комунікації людей в мережі інтернет у вигляді чат-бота та велику базу даних вищого навчального закладу, зокрема дані про викладачів, електронний розклад для очних та заочних форм навчання та загальну інформацію про освітній процес.

Система повинна давати змогу кожному викладачу швидко та легко взаємодіяти з будь-якою групою студентів надсилаючи їм повідомлення або нове методичне забезпечення. В свою чергу, кожен студент незалежно від форми навчання повинен мати змогу в будь-який момент часу отримати доступ до матеріалів з предмету, методичних посібників та подивитись свій розклад онлайн. Впровадження такої системи автоматизує навчальний процес та суттєво покращить його ефективність.

Об’єктом розробки є чат-бот в месенджері Telegram для інформаційної підтримки навчального процесу.

Метою дипломного проекту є підвищення ефективності навчального процесу, шляхом спрощення організації взаємодії студентів з викладачами та забезпечення більш зручного доступу до методичних матеріалів.

Для досягнення поставленої мети були вирішені наступні завдання:

– огляд та аналіз існуючих систем, які могли б вирішувати це завдання;

– аналіз та вибір технологій для реалізації дипломного проекту;

– розробка чат-бота.

# 1 АНАЛІЗ ВИМОГ ДО СИСТЕМИ

Перед створенням та проектуванням системи потрібно проаналізувати та визначити основні вимоги до розроблюваної системи. Зазвичай ці вимоги класифікують як функціональні та нефункціональні.

## 1.1 Функціональні вимоги

Функціональні вимоги визначають та описують те, що система повинна робити та які функції надавати користувачам.

Проаналізувавши предметну область обраної теми, були визначені наступні функціональні вимоги, описані у наступних підрозділах.

### 1.1.1 Реєстрація

Реєстрація у будь-якій системі це важлива та необхідна вимога. Вона повинна відбуватись зручно та швидко для всіх користувачів. В боті реєстрація повинна проходити за двома різними сценаріями, залежно від обраної ролі користувача. До реєстрації висунуті наступні вимоги:

– можливість вибору ролі при реєстрації (студент або викладач);

– реєстрація студента повинна відбуватись за допомогою натискання користувачем по спеціальному посиланню, яке можливо отримати у головного адміністратора системи. Після цього студент буде записаний у відповідну до посилання групу;

– реєстрація викладача потребує верифікації його запиту головним адміністратором бота;

– автоматична генерація паролю до веб-панелі викладача, після підтвердження його запиту на верифікацію.

### 1.1.2 Предмети

Оскільки бот пов’язаний з навчальним процесом, методичним забезпеченням до предметів та розкладом викладачів та студентів, він повинен мати у собі інформацію щодо предметів та можливість керування ними. На основі цього висунуті такі функціональні вимоги:

– автоматичне додавання предметів викладача в базу, за його ПІБ;

– можливість перегляду для викладача його власних предметів в боті, в зручному вигляді;

– можливість перегляду предметів викладача в особистому кабінеті;

– можливість редагувати, додавати або видаляти предмети для викладачем як в боті, так й в його особистому кабінеті.

### 1.1.3 Методичне забезпечення

Для інформаційної підтримки навчального процесу бот повинен реалізовувати логіку для взаємодії з методичним забезпеченням, оскільки це важливо та має бути доступним в будь-який момент часу з будь-якого місця.

Виходячи з цього повинен мати наступні функції:

– об'єкт методичного забезпечення має містити в собі інформацію про предмет до якого відноситься, опис файлу та файл з методичними вказівками;

– можливість перегляду викладачем доданих ним методичних файлів в боті, в зручному вигляді;

– можливість перегляду методичного забезпечення викладача в його особистому кабінеті;

– можливість редагувати, додавати або видаляти методичне забезпечення як в боті, так й в особистому кабінеті викладача;

– можливість перегляду студентом методичного забезпечення за потрібним для нього предметом.

### 1.1.4 Розсилка повідомлень

Система повинна спрощувати взаємодію та спілкування між студентами та викладачами. Зазвичай це здійснюється шляхом використання електронної пошти, або передачі інформації за допомогою старости. Обидва ці способи мають великий недолік у часі, який пройде між повідомленням викладача та моментом коли воно дійде до студентів, оскільки не завжди старости мають змогу своєчасно передавати інформацію, або перевіряти електронну пошту. Для покращення комунікації сторін, бот повинен надавати змогу розсилки повідомлення викладачем по обраній ним групі студентів, які одночасно отримають повідомлення в месенджері. До реалізації розсилки є наступні функціональні вимоги:

– при створені повідомлення для розсилки, викладач повинен обрати необхідну йому групу для надсилання інформації. Після цього ввести текст для повідомлення, за бажанням додати файл та перевіривши те, як воно буде виглядати – надіслати його;

– студенти повинні мати змогу отримувати всі повідомлення від викладачів своєчасно та без затримок;

– до кожного повідомлення повинна додаватись інформація про викладача, який його надіслав.

### 1.1.5 Електронні черги

Під час навчального процесу, особливо перед кінцем семестру, існує проблема з чергами студентів до викладачів на захист лабораторних робіт або здачі інших боргів з предметів. Для спрощення цього процесу та зменшенню черг з студентів потрібно створити електронні черги за допомогою бота. Це покращить організованість, оскільки кожен зможе бачити скільки людей зараз здає та підійде в свій час. До реалізації електронних черг є наступні вимоги:

– можливість створювати, запускати та зупиняти чергу має можливість тільки викладач з власного предмета;

– можливість для викладача переглядати власні створені електронні черги, в зручному вигляді в боті;

– можливість для викладача бачити весь список студентів в черзі, пропускати студента, та зупиняти чергу;

– можливість пошуку студентом черги с необхідного йому предмету та запису до неї;

– можливість для студента переглядати черги в які він став, бачити весь список студентів та можливість вийти з черги.

### 1.1.6 Панель головного адміністратора

Для керування системою та її адміністрування необхідно реалізувати функціонал для головного адміністратора, який повинен регулювати процеси реєстрації користувачів та управляти даними в боті. Для цього висунуті деякі функціональні вимоги:

– можливість переглядати запити викладачів на верифікацію;

– можливість логіну в панелі адміністратора;

– можливість прийняти або відхилити запит на реєстрацію користувача;

– можливість перегляду груп студентів, які були автоматично додані в бот через сервіс API KPI rozklad;

– можливість редагувати, створювати та видаляти групи студентів за допомогою зручного інтерфейсу;

– можливість перегляду та надання студентам за запитом старости, посилання для реєстрації студента в боті, у відповідній групі студентів;

## 1.2 Нефункціональні вимоги

Для повного опису системи недостатньо лише функціональних вимог, тому потрібно брати до уваги нефункціональні вимоги, які глобально поділяються на наступні категорії:

– практичність. Відповідає за те, щоб програмне забезпечення було простим та легким у використанні для будь-якого користувача, не завдавало зайвих труднощів та було інтуїтивно зрозумілим;

– надійність. Відповідає за роботу програмного забезпечення, та описує як система повинна себе поводити у разі помилок чи збоїв в роботі та зберегти важливі дані навіть у цьому випадку;

– продуктивність. Визначає характеристики системи під час взаємодії з користувачами, тобто наступні характеристики - відповідає за оптимальний час на відповідь запита користувача, кількість користувачів, яку одночасно може прийняти система та визначає які системні ресурси їй на це знадобляться;

– можливість обслуговування. Означає можливість легко модифікувати, змінювати програмне забезпечення для введення нового функціоналу або з метою виправлення помилок.

Виходячи з цього можна визначити необхідні нефункціональні вимоги до системи:

– система повинна бути надійною та доступною для використання користувачами в будь-який момент часу;

– в разі збоїв в роботі програмне забезпечення повинно адекватно реагувати на це та намагатись самостійно відновити роботу;

– в разі помилок перш за все важливе збереження всієї інформації в базі даних;

– система має бути побудована таким чином, щоб в майбутньому для неї було легко створювати нові модулі та функції або виправляти код з метою покращення;

– система повинна бути кросплатформена, тобто працювати на різних операційних системах.

Висновок до розділу 1

В цьому розділі були визначені, поставлені та описані вимоги до проекту. Також, окрім функціональних вимог, були висунуті нефункціональні вимоги, які відповідають за стабільність та надійність роботи програми, визначають як повинна працювати система та якими характеристиками вона має обладати.

# 2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

## 2.1 Чат-бот «KPI schedule bot»

Цей бот створений на платформі телеграм та надає актуальну інформацію про розклад занять та сесії за обраною групою. Має досить багато зручних налаштувань та команд для перегляду розкладу, а саме – перегляд за сьогодні, за поточний тиждень, за наступний тиждень або повний розклад. Також є можливість налаштування нагадувань про розклад на наступний день, в заданий користувачем час. Бот несе корисну інформацію зв’язану з навчальним процесом, але тільки частково виконує поставлену мету.

Переваги:

– можливість перегляду власного розкладу або розкладу інших;

– гнучка система налаштувань для оповіщення про розклад;

Недоліки:

– весь функціонал зав’язаний лише на розкладі, що не повністю покриває інформаційну підтримку навчального процесу;

– не зберігає дані про розклад в власну базу даних, тому при проблемах в роботі сервісу КПІ для отримання розкладу, бот не може надати необхідну інформацію.

## 2.2 Веб-сервіс «Розклад КПІ»

Даний веб-сервіс надає зручну інформацію про розклад занять за групою, або за викладачем. Інтерфейс сайту хоч і зручний, але трохи застарілий. Іноді стаються помилки в роботі, через що сайт деякий час недоступний, через що в цей час неможливо отримати інформацію про розклад занять чи сесії.

Загалом, незважаючи на деякі нюанси, він чудово виконує свою роботу та користується популярністю серед студентів та викладачів. Але сервіс тільки

частково виконує задачі поставлені перед дипломним проектом, тому не вирішує основне завдання.

Переваги:

– зручний перегляд розкладу занять;

– існує можливість перегляду розкладу сесії;

Недоліки:

– застарілий інтерфейс;

– при великій кількості користувачів одночасно трапляються перебої в роботі сервісу.

## 2.3 Веб-сервіс «Електронний кампус»

Цей сервіс надає інформаційну підтримку навчальному процесу, та використовується як засіб організації взаємодії студентів з викладачами. Система надає доступ до методичного забезпечення та корисної інформації щодо навчального процесу. Використовується для організації зворотнього зв’язку від викладачів, за допомогою інтерфейсів виставлення оцінок або атестацій, які студенти можуть переглянути у відповідному розділі сайта.

Електронний кампус за функціоналом є наближеним до мети розроблюваної системи, але деякі функції реалізовані не дуже зручно.

Переваги:

– можливість перегляду власних предметів;

– можливість перегляду методичного забезпечення з предметів;

– можливість отримання інформації щодо успішності з навчальних предметів;

– реалізована система оцінки викладачів серед студентів;

– можливість авторизації через телеграм

Недоліки:

– застарілий інтерфейс;

– трапляються збої в роботі сервісу;

– не зовсім зручна реалізація деяких функцій;

– викладач немає змоги надсилати повідомлення всім студентам групи.

Висновок до розділу 2

В цьому розділі було розглянуто декілька існуючих рішень, які в певній мірі виконують деякі з вимог, поставлених як завдання дипломного проекту.

Чат-бот «KPI schedule bot» та Веб-сервіс «Розклад КПІ» надають лише частину з корисної інформації з бази даних університету, а саме надання інформації про розклад занять. Найбільш близьким виявився веб-сервіс «Електронний кампус», але він не забезпечує швидкого та зручного засобу взаємодії студента з викладачем.

Отже, серед існуючих рішень немає такого, що повністю виконує цілі та завдання, які були поставлені, тому актуальність роботи є цілком виправданою.

# 3 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ

На початку розробки будь-якої системи або програмного забезпечення кожен розробник аналізуючи висунуті вимоги до проекту має обрати стек технологій та засобів для реалізації продукту.

На цей вибір можуть впливати багато факторів, але з них можна виділити деякі основні, а саме:

– насамперед це цілі та вимоги до самої системи, наприклад – операційні системи, які повинні підтримувати програму та бути з нею сумісними, швидкість роботи, надійність, масштабованість, тощо. Розробник аналізує їх та виділяє технології, за допомогою яких можливо досягти потрібного результату;

– не менш важливий фактор це вміння, навички та досвід роботи розробника з технологіями виділеними з попереднього пункту. Зрозуміло, що при рівних умовах завжди надають перевагу технологіями в яких є більший досвід, тому що це впливає на швидкість та якість створення програмного забезпечення;

– також до уваги потрібно брати актуальність використання технології в момент розробки продукту та враховувати її перспективи в майбутньому, тому що використання застарілих технології може привести до нестабільної роботи системи через декілька років, або до важкої підтримки та розробки нових функцій продукту.

В залежності від складності та розмірів системи, вона може поєднувати в собі декілька технологій з однієї категорії. Наприклад серверна частина може бути створена у вигляді мікросервісної архітектури, тобто мати декілька серверів, які виконують різні функції та написані різними мовами програмування з використанням різних технології зв’язані в одну велику систему, але при цьому можуть бути навіть незалежними один від одного. Цей підхід до розробки стає доволі популярним, оскільки дозволяє різним розробникам одночасно працювати над окремими модулями не заважаючи один одному, що значно прискорює розробку.

## 3.1 Мова програмування

Мовою програмування для розробки back-end частини було обрано – Python версії 3.6. оскільки ця версія на даний момент стабільна та найбільш підтримувана різними модулями та бібліотеками.

Python - це інтерпретована, об'єктно-орієнтована мова програмування високого рівня, яка орієнтована на підвищення продуктивності розробки [1]. Високорівневі вбудовані структури даних у поєднанні з динамічною типізацією роблять її дуже привабливою для швидкого розвитку додатків. Простий, легкий у вивченні синтаксис Python підкреслює читабельність коду і, таким чином, зменшує витрати часу на підтримку програм написаних цією мовою. Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python та велика стандартна бібліотека доступні безкоштовно на всіх основних платформах, і їх можна вільно розповсюджувати.

Перевагами цього вибору є також те, що ця мова займає одне з найперших місць за популярністю використання, а отже має безліч зручних готових рішень, велике ком’юніті та підтримку від великих IT-компаній, які використовують цю мову у своїх проектах.

## 3.2 Веб-фреймворк

В мові програмування Python існує безліч зручних фреймворків для побудови веб-сервісів. На даний момент найбільшою популярністю користується Django 2 [2]. Створений він був для спрощення та прискорення розробки, зменшення повторення кода та ефективного використання вбудованих в нього модулів. Зокрема має власну адмін-панель, інтерфейси для взаємодії з різними базами даних, які дуже легко в нього інтегруються.

Основні переваги та можливості Django:

– вбудована адмін-панель;

–власна Django-ORM для спрощення написання запитів до бази даних, незалежно від того, яка з них використовується;

– вбудовані шаблони для представлень роутів;

– підтримка кешування;

– вбудована система авторизації користувачів;

– масштабованість, тобто дуже легко розширювати функціонал сервісу.

## 3.3 База даних PostgreSQL

PostgreSQL - це потужна об'єктно-реляційна база даних із відкритим кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають та масштабують найскладніші навантаження даних [3].

PostgreSQL має багато функцій, які спрямовані на допомогу розробникам у створенні програм, для захисту цілісності даних та побудови середовищ, стійких до відмов, та допомагають керувати вашими даними незалежно від того, наскільки великий чи малий набір даних ви маєте. Також PostgreSQL є безкоштовним та має відкритий код. Ви можете визначати власні типи даних, створювати власні функції, навіть писати код з різних мов програмування, не перекомпілюючи свою базу даних.

Дана база даних характеризується швидкістю, неймовірною гнучкістю та надійністю у зберіганні даних, та безвідмовної роботи. Працює за принципом клієнт-серверної моделі. Завдяки особливості реалізації зберігання даних в так названих «кластерах», будь-яку базу даних створену в postgreSQL дуже легко масштабувати.

Переваги:

– можливість створення складних структур даних;

– забезпечує цілісність та надійність зберігання даних;

– легко збільшувати та масштабувати;

–має велику кількість налаштувань та доповнень для проектування систем будь-якої складності.

Висновок до розділу 3

В розділі були розглянуті та описані технології, які були обрані для реалізації системи. Мовою програмування було обрано Python, як найбільш зручна та гнучка мова програмування. Для реалізації веб-панелі було обрано веб-фреймворк Django, оскільки він один з основних інструментів для цього на Python. Для бази даних було вирішено використовувати об’єктно-реляційну PostgreSQL через її гнучкість та надійність.

Вибір обраних технологій був аргументований перевагами кожної із них.

# 4 РОЗРОБКА ЧАТ-БОТА

## 4.1 Реєстрація телеграм бота

Перш ніж перейти до розробки системи та будь-яких дій у цей бік потрібно зареєструвати телеграм-бота та отримати ключ доступу до BOT API, за допомогою якого ми зможем керувати ботом [4].

Оскільки Telegram це месенджер, то й реєстрація бота відбувається в чаті, за допомогою іншого, офіційного телеграм-бота - BotFather [5]. Взагалі всі дії пов'язані з налаштуваннями власних телеграм-ботів виконуються лише через нього.

Для того, щоб знайти BotFather, потрібно в пошуку чатів ввести запит «BotFather» (рисунок 4.1) та обрати бота з відповідним ім'ям та відміткою офіційного акаунта, це означає що бот є підтвердженим та що ним можна безпечно користуватись.

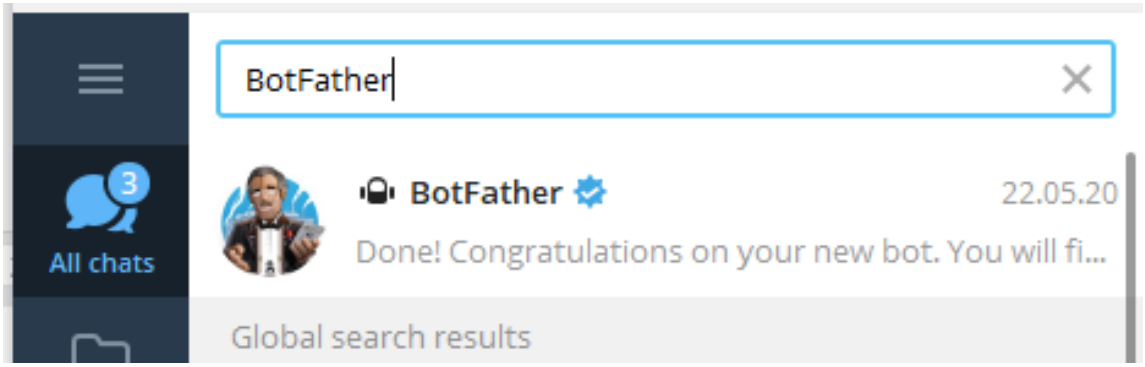


Рисунок 4.1 – Пошук BotFather бота за допомогою пошукової системи в месенджері Telegram

Далі потрібно розпочати діалог з ботом надіславши йому команду «/start», після цього з'явиться список можливих команд для налаштувань ботів (рисунок 4.2).

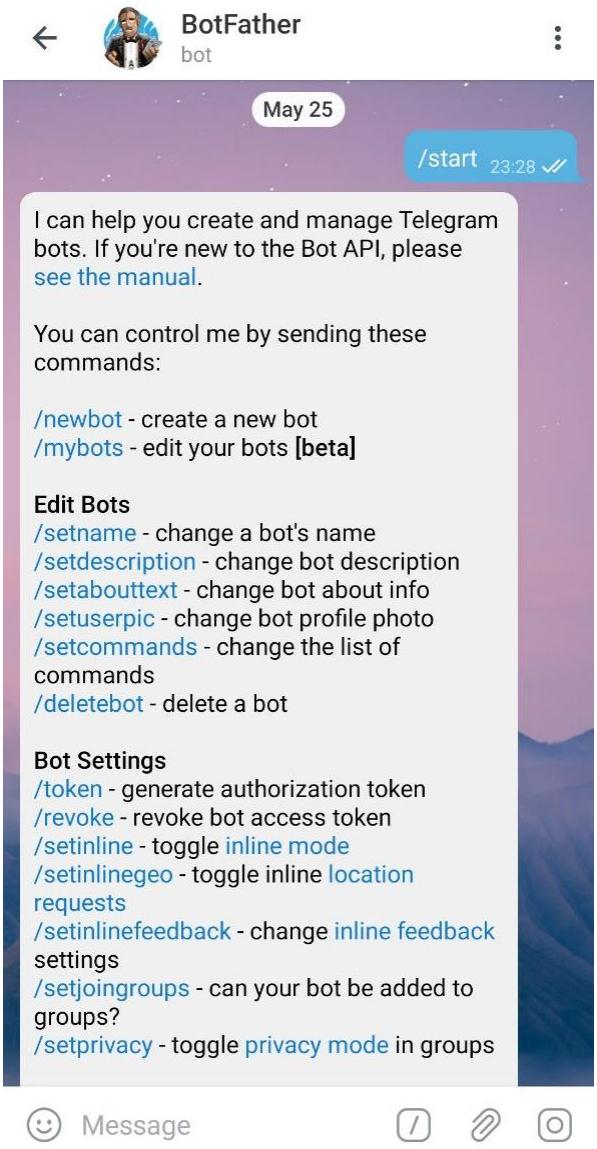


Рисунок 4.2 – Команди для налаштування бота

Серед цих команд для налаштувань бота знадобляться наступні:

– «newbot» - це перша команда, яка створює нового бота;

– «mybots» - повертає інформацію про вже створені користувачем боти та дозволяє їх налаштовувати та редагувати;

– «setname» - використовується для зміни ім’я бота;

– «setdescription» - дозволяє змінити опис про бота при першому відкритті чата з ним;

– «setabouttext» - дозволяє змінювати інформацію про бота на його сторінці;

– «setuserpic» - використовується для встановлення та зміни фотографії бота;

– «setcommands» - використовується для додавання нових команд в бота, за допомогою яких бот виконує запрограмовані дії;

– «deletebot» - ця команда повністю видаляє бота назавжди;

– «token» - використовується для отримання ключу доступу до керування ботом через Telegram BOT API;

– «set_inline» - вмикає інлайн мод для бота;

– «setjoingroups» - вмикає або вимикає можливість додавати бота до групових чатів;

– «setprivacy» - вмикає або вимикає можливість для бота читати групові чати;

– «revoke» - дозволяє видалити старий ключ доступу до бота, та згенерувати новий. Ця команда корисна в разі, якщо хтось інший випадково заволодів минулим токеном.

Для створення нового бота потрібно надіслати до BotFather команду «/newbot». Після цього необхідно обрати ім’я для бота, яке є насправді дуже важливим етапом, оскільки це перше, що бачить користувач знайшовши його. Ім’я повинно бути простим для того, щоб гарно запам’ятатись, але при цьому нести корисну інформацію. Наступним кроком є вибір унікальної назви, яка обов’язково повинна мати в собі слово «bot», щоб всі користувачі розуміли, що це не реальна людина. Коли попередні кроки були виконані, обов’язкова частина реєстрації закінчується й ботом вже можна користуватись, отримавши ключ доступу до Telegram Bot API (рисунок 4.3). Але краще додати більше корисної інформації про чат-бота, щоб користувач одразу розумів для чого він потрібен та ким був створений.

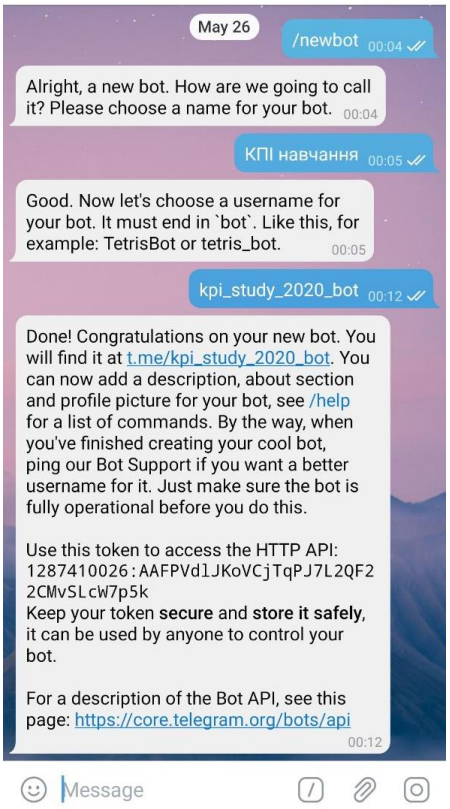


Рисунок 4.3 – Процес реєстрації нового бота та отримання ключа доступу до Telegram Bot API

Додавши інформацію про бота, опис та фото профілю, реєстрація завершена (рисунок 4.4) та можна переходити до розробки програмної частини.

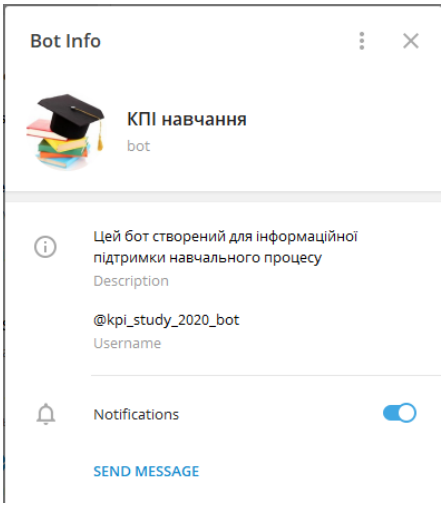


Рисунок 4.4 – Інформація про створеного бота

## 4.2 Реалізація серверної частини бота

### 4.2.1 Запити до Telegram Bot Api

Всі запити до Telegram Bot API описані у відкритій документації на офіційному сайті. Обмін запитами повинен відбуватись за допомогою безпечного протоколу HTTPS, який на відміну від протоколу HTTP передає дані в зашифрованому вигляді [6].

Запити доступні в API поділяються на GET та POST, а обмін інформацією відбувається у форматі даних JSON [7]. GET метод використовується тільки для отримання даних від API, а POST запит навпаки, коли потрібно надіслати інформацію у форматі JSON до API серверу.

Прийнято, що після кожного запиту, сервер повинен додавати до відповіді статус, тобто якщо запит був успішно виконаний або щось пішло не так – сервер повинен додати відповідний статус.

Кожен запит обов'язково має містити в собі ключ доступу бота та назву метода, який потрібно використати. Також необхідно передавати зазначені в документації поля, та використовувати відповідний тип запита.

В загальному вигляді запит виглядає наступним чином: https://api.telegram.org/bot<TOKEN>/METHOD_NAME, де замість TOKEN – повинен бути ключ доступу до API, який був отриманий при реєстрації бота в BotFather, а замість METHOD_NAME – назва методу, або функції до якої потрібно звернутись [8]. Наприклад, для отримання інформації про створеного бота потрібно зробити GET запит з назвою метода getME та підставити TOKEN - 1287410026:AAFPVdlJKoVCjTqPJ7L2QF22CMvSLcW7p5k, який закріплений за створеним ботом (рисунок 4.5). В результаті отримали інформацію про бота, яка знадобиться при розробці системи.

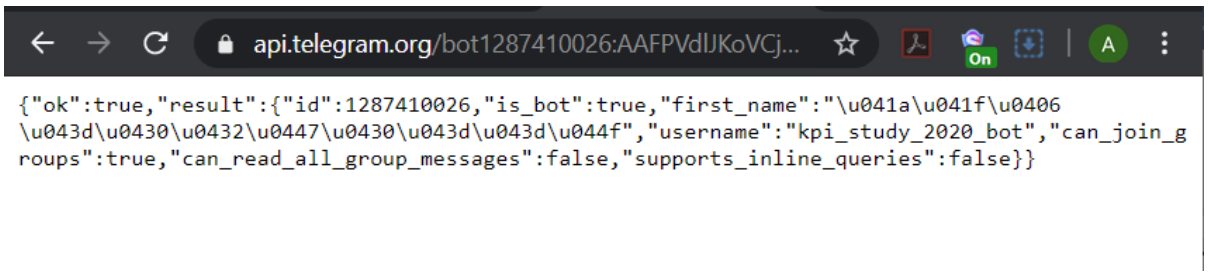


Рисунок 4.5 – Приклад запиту до Telegram Bot API для отримання інформації про бота

Запити, які були використані при розробці бота:

«sendMessage» - метод є основним, оскільки використовується для відправки текстового повідомлення. Також до нього можна прикріпити віртуальну клавіатуру (Keyboard Markup). Метод очікує наступні параметри:

– ідентифікатор куди потрібно надіслати повідомлення (chat_id);

– текст для повідомлення (text);

– повідомлення на яке потрібно відповісти (reply_to_message_id);

– тип розмітки для форматування повідомлення (parse_mode), може приймати значення «html» або «markdown»;

– параметр відповідальний за те, щоб надіслати повідомлення у беззвучному режимі (disable_notification);

– об’єкт віртуальної клавіатури, який описує кнопки, їх текст та порядок розміщення (reply_markup).

«sendPhoto» - метод використовується для відправки повідомлення у вигляді фотографії. Може приймати такі параметри:

– чат (chat_id);

– фото (photo) у вигляді файлу або ідентифікатору з чата telegram;

– текстовий опис фото (caption);

– об’єкт клавіатури (reply_markup).

«editMessageCaption» - даний метод використовується для редагування опису файлів, фото, відео або аудіо вкладень в повідомленні. Має наступні вхідні параметри:

– чат, де знаходиться повідомення (chat_id);

– повідомлення в цьому чаті (message_id);

– новий опис до вкладення (caption);

– нова клавіатура (reply_markup).

«editMessageText» - метод потрібен для зміни тексту повідомлення. На вхід очікує такі параметри:

– чат, де знаходиться повідомлення для редагування (chat_id);

– повідомлення в цьому чаті (message_id);

– новий текст для повідомлення (text);

– нова клавіатура до повідомлення (reply_markup).

«sendDocument» - для надсилання всіх файлів, окрім фото, відео та аудіо потрібно використовувати саме цей метод. Він приймає наступні параметри:

– чат (chat_id);

– файл (file) у вигляді файла або ідентифікатору з чата telegram;

– опис файла (caption);

– об’єкт клавіатури (reply_markup).

«sendVideo» - щоб надіслати через бота відео до користувача, потрібно звернутись до цього методу та надіслати наступні вхідні параметри:

– чат (chat_id);

– відео (video) у вигляді файла або ідентифікатору з чата telegram;

– опис до відео (caption);

– об’єкт клавіатури (reply_markup).

«sendAudio» - даний метод потрібно використати, якщо необхідно надіслати аудіо-файл. При цьому має такі вхідні параметри:

– чат (chat_id);

– аудіо-файл (audio) у вигляді файла або ідентифікатору з чата telegram;

– опис аудіо-файлу (caption);

– об’єкт віртуальної клавіатури (reply_markup).

«setWebhook» - метод використовується для встановлення веб-хуку для бота. Тобто про всі нові сповіщення телеграм сам буде надсилати повідомлення на сервер. Приймає наступні параметри:

– посилання, на яке будуть відправляться сповіщення (url);

– типи повідомлень про які потрібно сповіщати (allowed_updates)

– сертифікат з ключем доступу при зверненні до сервера (certificate).

«getMe» - метод повертає інформацію про бота, а саме:

– ідентифікатор бота (id);

– ім’я бота (fist_name);

– унікальне ім’я бота в телеграм (username);

– відмітка, що це бот (is_bot).

«getFile» - необхідно використовувати для того, щоб отримати інформацію про файл за його ідентифікатором (file_id). У відповідь повертає:

– ідентифікатор файла (file_unique_id);

– розмір файла (file_size);

– посилання для скачування файла (file_path).

«forwardMessage» - цей метод потрібно використовувати для пересилання повідомлення, наприклад з одного чата в інший. На вхід приймає наступні параметри:

– чат в який надіслати повідомлення (chat_id);

– чат з якого потрібно надіслати повідомлення (from_chat_id);

– повідомлення в чаті, яке необхідно переслати (message_id).

«deleteMessage» - метод використовується для видалення повідомлення з чату. На вхід очікує:

– чат (chat_id);

– повідомлення в цьому чаті (message_id).

### 4.2.2 Об’єкти даних у Telegram Bot Api

Для використання всіх наведених методів BotAPI потрібно розуміти як представлені дані з якими працює бот у вигляді об’єктів. Кожен об’єкт який передається у вигляді JSON даних, але всередині системи Телеграм, являють собою класи, опис яких наведено далі.

«User» - цей об’єкт визначає та представляю собою будь-якого користувача телеграм або телеграм-бота. Об’єкт створюється при реєстрації в месенджері. Має наступні поля:

– id – ідентифікатор користувача в системі;

– is_bot – визначає чи реальний це юзер, або бот;

– first_name – ім’я користувача;

– last_name – прізвище користувача;

– username – унікальне ім’я користувача;

– language_code – код мови, яка закріплена за користувачем.

«Message» - об’єкт представляю собою повідомлення. Повідомлення це основа будь-якого месенджеру, оскільки все спрямоване на обмін ними між користувачами. Телеграм визначив для цього об’єкту поля:

– message_id – ідентифікатор повідомлення в чаті;

– from – від кого було надіслане;

– date – дата відправки повідомлення;

– chat – чат в який вон обуло надіслано;

– text – текст;

– reply_markup – клавіатура до повідомлення;

– одне вкладення з можливих – аудіо (audio), документ (document), фото (photo), відео (video), голосове повідомлення (voice_note), відео-повідомлення (video_note), локація (location), контакт користувача (contact).

«Chat» - об’єкт  відображає чат. В телеграмі існують чати різних видів, групові, приватні, канали, приватні канали. Вся інформація про них зберігається в такому вигляді:

– id – ідентифікатор чата;

– type – тип чата;

– title – назва чата;

– username – унікальна назва чата;

– photo – зображення до чата;

– description – опис;

– permissions – масив з правами користувачів в чаті.

### 4.2.3 Обробка оновлень

Для отримання оновлень про нові повідомлення або інші дії користувачів відносно бота в Bot API існує два способи.

Перший спосіб має назву «getUpdates», його принцип дії в тому, що бот постійно робить запити до API з питанням чи не надійшли до нього нові сповіщення. Цей спосіб не є надійним, оскільки через помилки в програмному коді бот може перестати робити ці запити і тоді відбувається втрата повідомлень користувачів.

Другий спосіб є більш надійним, але потребує більших зусиль в налаштуваннях оскільки для функціонування потрібен веб-фреймворк, який буде приймати запити. Він полягає в тому, що до телеграм API надсилається метод «setWebhook», за допомогою якого повідомляється про те, що бот доступний за деякою веб-адресою й у разі оновлень телеграм сам буде надсилати за цим URL інформацію о нових повідомленнях, які веб-сервер бота повинен буде обробляти. При розробці системи був обраний саме цей спосіб, як більш правильний з точки зору архітектури.

В якості веб-сервера було обрано веб-фреймворк Django, який у разі запиту на відповідний URL про нове повідомлення обробляє його у окремому модулі логіки чат-бота та надсилає відповідь до телеграм API про успішно прийняте повідомлення.

### 4.2.4 Використання бібліотеки pyTelegramBotApi

Оскільки використовувати в коді HTTPS запити до API через відповідні веб-адреси є не дуже зручним, були створені спеціальні модулі та бібліотеки, які являють собою надбудови над запитами та використовуються для спрощення та зменшення повторного використання кода.

Для мови програмування Python найбільш зручною в використанні є бібліотека «pyTelegramBotApi», яка й була обрана для реалізації зв'язку системи з Bot API [9]. Вона містить в собі всі описані типи даних та методи API Telegram.

В реалізації серверної частини бота ця бібліотека грає важливу роль, оскільки вона відповідає спочатку за отримання та передачу даних до модулів логічної обробки, де відбувається логіка зв'язана з функціональними можливостями бота, запитами до бази даних, перевіркою та формуванням інформації для відповіді на прийняте повідомлення, а потім ці дані знову за допомогою відповідних надбудов бібліотеки над методами Bot API відправляються повідомленнями у відповідь до користувача.

Для прикладу на рисунку 4.6 наведено відправку повідомлення та клавіатури до викладача при потраплянні в головне меню. Спочатку створюється віртуальна клавіатура keyboard, яку буде потрібно надіслати. Для цього використовується метод бібліотеки ReplyKeyboardMarkup з параметром resize_keyboard, який означає зробити клавіатуру адаптивною на будь-якому пристрої. Після цього за допомогою метода keyboard.add(), додаються віртуальні кнопки, які приймають вхідним параметром назву кнопки, тобто текст який буде відображений на ній. Далі для надсилання повідомлення потрібно створити об'єкт

бота, за допомогою метода Telebot(), який приймає на вхід ключ доступу бота до Bot Api. Після цього можна надіслати повідомлення використовуючи метод send_message(), який приймає ідентифікатор чата, текст повідомлення, та створену вище клавіатуру.

Для того щоб надіслати віртуальну клавіатуру прикріплену саме до текстового повідомлення, в бібліотеці існує об'єкт InlineReplyKeyboardMarkup, для якого аналогічно до звичайної клавіатури необхідно додати кнопки, які можуть бути одним з можливих типів (кнопка-посилання, кнопка-дія, кнопка-запит). Кожна з них створена під конкретні цілі. Найчастіше використовується кнопка-дія, при натисканні на яку, бот отримує повідомлення про те, що потрібно виконати логіку закладену в цю кнопку.

```
keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
keyboard.add(get_translation_for(language, 'my_subjects_btn'))
keyboard.add(get_translation_for(language, 'my_study_resources_btn'))
keyboard.add(get_translation_for(language, 'broadcast_btn'))
keyboard.add(get_translation_for(language, 'online_queue_btn'))
keyboard.add(get_translation_for(language, 'my_schedule_btn'))

sender = Telebot(token)
sender.send_message(chat.id,
                    'Головне меню',
                    reply_markup=keyboard()
                    )
```

Рисунок 4.6 – Приклад запиту до Telegram Bot API з використанням модуля pyTelegramBotApi

### 4.2.5 Інтеграція з rozklad KPI API

Чат-бот інтегрується з відкритою базою даних КПІ за допомогою сервісу https://api.rozklad.org.ua/ [10]. Запити можна робити за двома протоколами HTTP або HTTPS. В боті були використані наступні можливості сервіса, описані далі.

Пошук викладача, здійснюється за допомогою метода teachers без параметрів або з параметром ПІБ. Наприклад пошук викладача за його ПІБ буде виглядати так: https://api.rozklad.org.ua/v2/teachers/Іванов+Іван+Іванович/. Щоб отримати список всіх викладачів, потрібно здійснити запит https://api.rozklad.org.ua/v2/teachers/?filter={'limit':100,'offset':0} (рисунок 4.7), де limit – це кількість викладачів у відповіді від одного до 100, offset – зміщення.

Для отримання одного викладача потрібно зробити запит https://api.rozklad.org.ua/v2/teachers/<TEACHER_ID>, де TEACHER_ID – це унікальний ідентифікатор викладача.

Кожен об’єкт викладача має наступні поля:

– унікальний ідентифікатор викладача в базі сервісу (teacher_id);

– ПІБ викладача (teacher_name);

– ПІБ викладача з його посадою (teacher_full_name);

– скорочене повне ім’я викладача (teacher_short_name);

– посилання на розклад викладача (teacher_url).

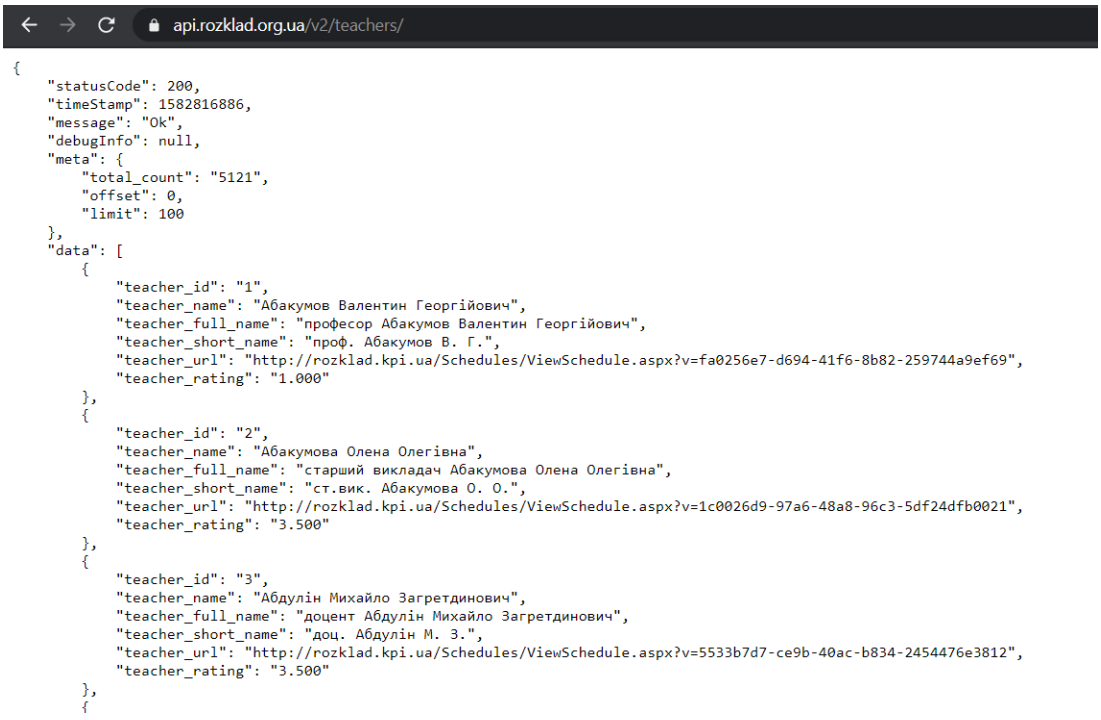


Рисунок 4.7 – Приклад запиту до KPI API rozklad для отримання всіх викладачів

Для отримання всіх груп студентів з бази даних сервіса потрібно зробити наступний запит: https://api.rozklad.org.ua/v2/groups/?filter={'limit':100,'offset':0}, де limit – це кількість груп у відповіді від одного до 100, offset – зміщення (рисунок 4.8).

Для отримання однієї групи потрібно зробити запит https://api.rozklad.org.ua/v2/groups/<GROUP_ID>, де GROUP_ID – це унікальний ідентифікатор групи.

Кожен об’єкт групи має наступні поля:

– унікальний ідентифкатор групи (group_id);

– повна назва групи (group_full_name);

– префікс назви групи (group_prefix);

– тип групи (group_type);

– посилання на інформацію про групу (group_url).


Рисунок 4.8 – Приклад запиту до KPI API rozklad для отримання всіх груп студентів

Для отримання розкладу групи студентів використовується запит https://api.rozklad.org.ua/v2/groups//lessons (рисунок 4.9), який має наступні параметри:

– номер дня за яким показати предмети (day_number);

– назва дня за яким показати предмети (day_name);

– предмети за номером заняття в розкладі (lesson_number);

– предмети за номером навчального тижня (lesson_week);

– показати предмети за типом заняття (lesson_type).

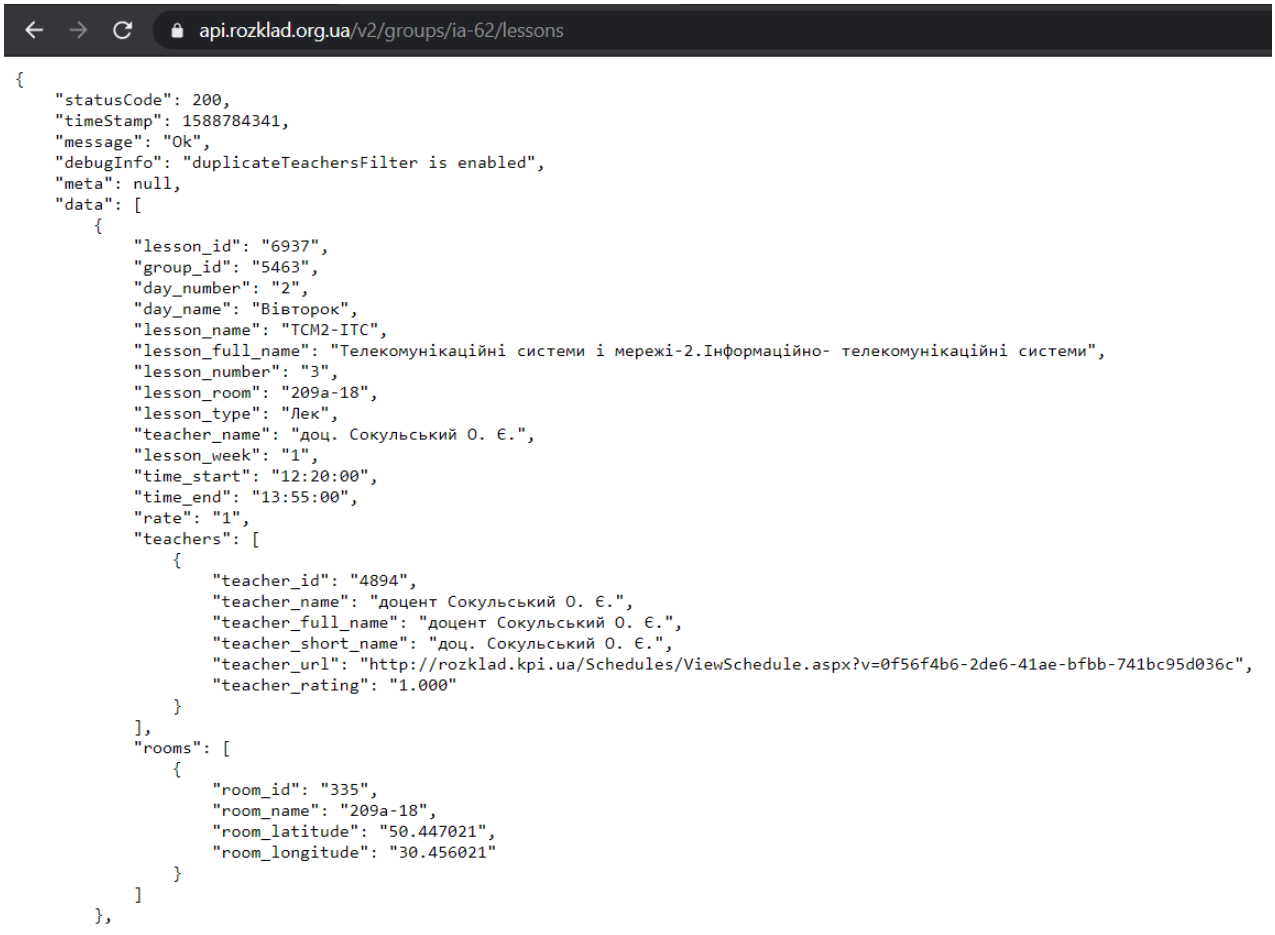


Рисунок 4.9 – Приклад запиту до KPI API rozklad для отримання розкладу групи

Кожен объект заняття має такі поля:

– ідентифікатор заняття в системі розкладу (lesson_id);

– ідентифікатор групи, до якої відноситься це заняття (group_id);

– номер дня, в який день проходить заняття (day_number);

– скорочена назва заняття (lesson_name);

– повна назва заняття (lesson_full_name);

– номер заняття за розкладом (lesson_number);

– аудиторія в якій пройде заняття (lesson_room);

– тип заняття (lesson_type);

– ім’я викладача (teacher_name);

– час початку заняття (time_start);

– час закінчення заняття (time_end).

Для отримання розкладу викладача, потрібно зробити запит https://api.rozklad.org.ua/v2/teachers/<TEACHER_ID>/lessons, де TEACHER_ID – ідентифікатор викладача в системі розкладу. Відповідь має такі поля (рисунок 4.10):

– ідентифікатор заняття  (lesson_id);

– група, до якої відноситься це заняття (group_id);

– номер дня, в який день проходить заняття (day_number);

– скорочена назва заняття (lesson_name);

– назва заняття (lesson_full_name);

– номер заняття за розкладом (lesson_number);

– аудиторія  (lesson_room);

– тип заняття (lesson_type);

– ім’я викладача (teacher_name);

– номер навчально тижня коли проходить заняття (lesson_week);

– час початку заняття (time_start);

– час закінчення заняття (time_end).

– об’єкти аудиторій для занятті (rooms)

– об’єкти груп для заняття (groups)

– статус відповіді (status_code)

– додаткові фільтри при запиті (info)

Також Запит має гнучкі параметри для фільтрації даних за якими потрібно отримати розклад:

– номер дня  (day_number);

– назва дня (day_name);

– номер заняття (lesson_number);

– номером навчального тижня (lesson_week);

– тип заняття (lesson_type).

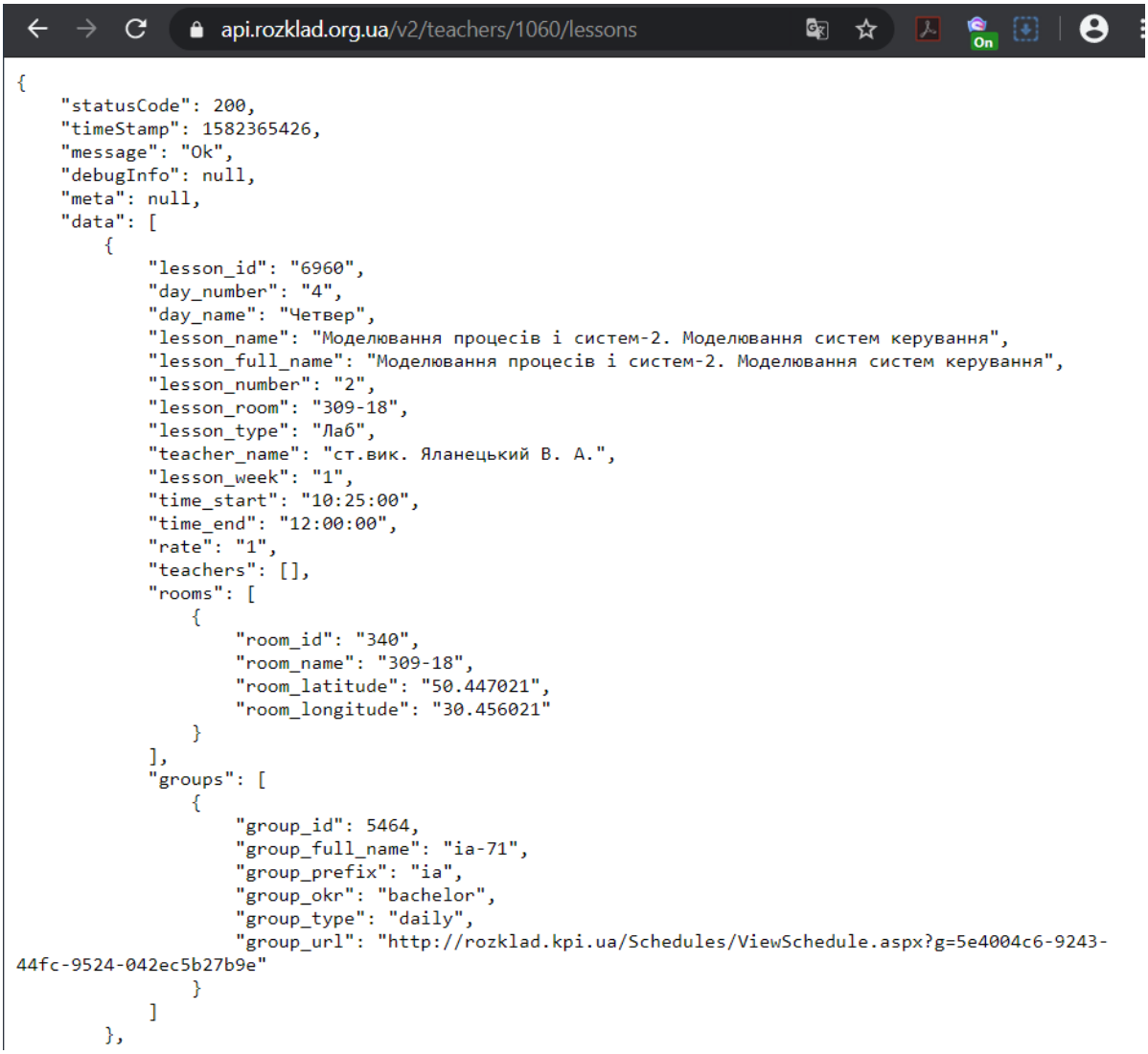


Рисунок 4.10 – Приклад запиту до KPI API rozklad для отримання розкладу викладача

### 4.2.6 Реалізація шаблону проектування State

Для реалізації пунктів та підпунктів меню бота було використано шаблон проектування – Стан.

Цей шаблон являє собою реалізацію скінченного автомату в програмуванні. Основна його ідея в тому, що об’єкт може знаходитись в одному з декількох станів, котрі весь час змінюється в залежності від дій об’єкта [11]. Набір цих станів та переходів між ними є визначеним та кінцевим. При цьому в залежності від стану в якому знаходиться об’єкт в даний момент часу, також залежить й поведінка та реакція програми на його дії, тобто для кожного стану вона буде різна.

Необхідність використання цього шаблону проектування в чат-боті зумовлена тим, що система отримує лише повідомлення, але при цьому неможливо визначити контекст та очікувану реакцію від користувача, який це надіслав, оскільки в різних пунктах меню бота однаковий текст повідомлення може означати різну поведінку. Тому було створено кінцевий набір станів кожен з яких відповідає деякому пункту меню бота, в яких може знаходитися користувач та функцій переходів між ними. Після кожної дії користувача, в базу даних зберігається його останній стан та при новому повідомленні в залежності від цього останнього стану визначається поведінка та реакція бота на запит.

Перелік та опис розроблених станів в яких може перебувати користувач та їх опис наведено далі. Також у додатку А наведено їх код, а на кресленику ІА62.110БАК.005.Д2 – діаграма переходів між станами.

StartState - перехід в цей стан відбувається при першому повідомленні від користувача до бота. Тобто визначає нового юзера. Далі користувач переходить до стану вибору ролі.

ChoiceRoleState - в цьому стані користувач повинен обрати роль, за якою хоче зареєструватись в боті, студент чи викладач. В залежності від його вибору переходить до станів TeacherRegistrationState або StudentRegistrationState.

TeacherRegistrationState - стан призначений для реєстрації користувача у ролі викладача. Реалізована логіка необхідних питань та створення заявки верифікації до головного адміна. Після підтвердження верифікації користувач перейде у стан головного меню MenuState.

StudentRegistrationState - цей стан відповідає за реєстрацію користувача у ролі студента. Для того, щоб верифікувати студента з якої він групи та взагалі студент, було створено логіку індивідуальних посилань, які повинен отримати староста у головного адміністратора та надати студентам. Після переходу по цьому посиланню студент зараховується до групи, яка була прив’язана до посилання. Після цього переходить до стану головного меню MenuState.

MenuState - стан визначає головне меню бота, з якого можна потрапити в інші стани, доступні користувачу в залежності від його ролі.

MySubjectsMenuState - стан є підменю для керування предметами викладача. З нього можна перейти до стану додавання нового предмету (AddSubjectState) або до стану перегляду власних предметів (MySubjectsState).

AddSubjectState - цей стан створений для додавання нових предметів викладача. Перейти до нього можна з стану MenuState тільки викладачу.

MySubjectsState - стан відповідає за перегляд та видалення предмету викладача. Перехід здійснюється лише з стану MenuState та доступний лише користувачу у ролі викладача.

TeacherStudyResourcesMenuState - стан є підменю для керування методичним забезпеченням доданим викладачем. З нього можна перейти до станів додавання нового матеріалу або до стану перегляду вже доданих методичних забезпечень (MyStudyResourceState).

AddStudyResourceSubjectState - стан є частиною форми для створення нового методичного забезпечення, а саме використовується для вибору предмета з якого потрібно додати файл. Далі здійснюється перехід до стану AddStudyResourceSubjectState.

AddStudyResourceFileState - цей стан використовується для збереження файла при створені методичного забезпечення. Після цього користувач переходить до стану AddStudyResourceNameState.

AddStudyResourceNameState - стан потрібен для додавання опису чи назви файла в процесі створення методичного забезпечення. Після цього методичне забезпечення успішно створено та користувач повертається до стану TeacherStudyResourcesMenuState.

MyStudyResourceState - цей стан використовується для перегляду та видалення власного методичного забезпечення викладача. Перехід можливий лише з стану MenuState та лише користувачу у ролі викладача.

TeacherQueueMenuState - стан являє собою підменю для керування електронними чергами викладача. З нього можна перейти до стану створення нової електронної черги (StartOnlineQueueState) або до перегляду вже активних онлайн черг (MyOnlineQueuesState).

MyOnlineQueuesState - цей стан використовується для перегляду та керування вже створеними електронними чергами викладача. Тобто подивитись інформацію про чергу, видалити її або викликати наступного студента.

StartOnlineQueueState - викладач потрапляє у цей стану для створення електронних черг на здачу лабораторних робіт з предмета.

AddBroadcastGroupState - в цьому стані викладач має обрати групу для якої буде здійснювати розсилку повідомлення. Після цього переходить до стану введення цього повідомлення (AddBroadcastMessageState).

AddBroadcastMessageState - цей стан використовується для введення повідомлення, яке буде надіслано студентам з обраної групи. Після цього користувач переходить до стану AddBroadcastFileState.

AddBroadcastFileState - при створенні розсилки, користувач який знаходиться в цьому стані має можливість додати до свого повідомлення або пропустити цей пункт перейшовши до підтвердження повідомлення для розсилки (AddBroadcastMessageConfirmState).

AddBroadcastMessageConfirmState - в цьому стані користувач бачить повідомлення, як воно буде виглядати, коли надійде до студентів. Після цього він може підтвердити або відхилити розсилку та повернутись в головне меню бота (MenuState).

ShowStudyResourceState - для перегляду методичного забезпечення в боті, студент переходить в цей стан з головного меню бота (MenuState). Виводиться список методичних матеріалів за кожним предметом.

StudentQueueMenuState - цей стан є підменю для вибору дії зв'язаної з електронними чергами. Звідси можна потрапити до стану пошуку електронної черги або до стану перегляду активних черг, в яких перебуває студент (StudentMyQueuesState).

StudentMyQueuesState - в цьому стані виводяться черги, в яких зареєструвався студент та інформація про них – повна черга, хто відповідає, хто наступний. Також є можливість покинути чергу.

StudentChoiceSubjectOnlineQueueState - цей стан потрібен для того, щоб студент спочатку обрав предмет з якого шукати електронну чергу, а вже потім перейшов до перегляду активних черг (StudentChoiceQueuesState).

StudentChoiceQueuesState - в цьому стані студент має можливість переглянути доступні електронні черги за обраним предметом, та зареєструватись в необхідній йому черзі.

## 4.3 Реалізація веб-панелі

Backend та Frontend веб-панелі створені за допомогою веб-фреймворку Django. Принцип його роботи побудован на основі патерну MVT – Model-View-Template (модель – представленя – шаблон) [12]. Архітектура цього патерну дозволяє розробнику окремо працювати з візуальним представленням та бізнес-логікою. Компоненти MVT можна використовувати окремо.

### 4.3.3 Модель в Django

Джерелом інформації про дані є моделі (model), вони містять в собі ключові поля та поведінку об’єктів системи. Зазвичай одна модель дзеркально відображена однією таблицею в базі даних.

Дані в моделі представлені атрибутами та полями. Оскільки модель являє собою звичайний клас, то вона не знає про інші рівні та компоненти. Модель відповідає за логіку, методи, властивості та різні маніпуляції з даними. Також саме моделі відповідають за створювання, зчитування, видалення та збереження об’єктів в базі даних.

### 4.3.4 Представлення в Django

Подання (view) вирішує декілька різних завдання: приймає HTTP-запити, реалізує логіку обробки цих запитів за допомогою певних методів та відправляє HTTP-відповідь у відповідь на запити.

Спочатку в залежності від URL отриманого запиту та інформації, яка міститься у ньому, Django знаходить відповідне представлення, яке пов’язане з отриманим запитом у окремому файлі urls.py. Далі в представленні відбувається обробка запиту, тобто реалізується бізнес-логіка, зчитуються або записується дані в базу даних та формується відповідь, яка буде згенерована у відповідний шаблон пов’язаний з цим представленням. Після генерації шаблона з даними, він надсилається по HTTP у відповідь на запит.

Тобто уявлення отримує дані від моделі і надає шаблонами (templates) доступ до цих даних або попередньо обробляє дані і потім надає до них доступ шаблонами. Цей алгоритм схематично зображено на рисунку 4.11.

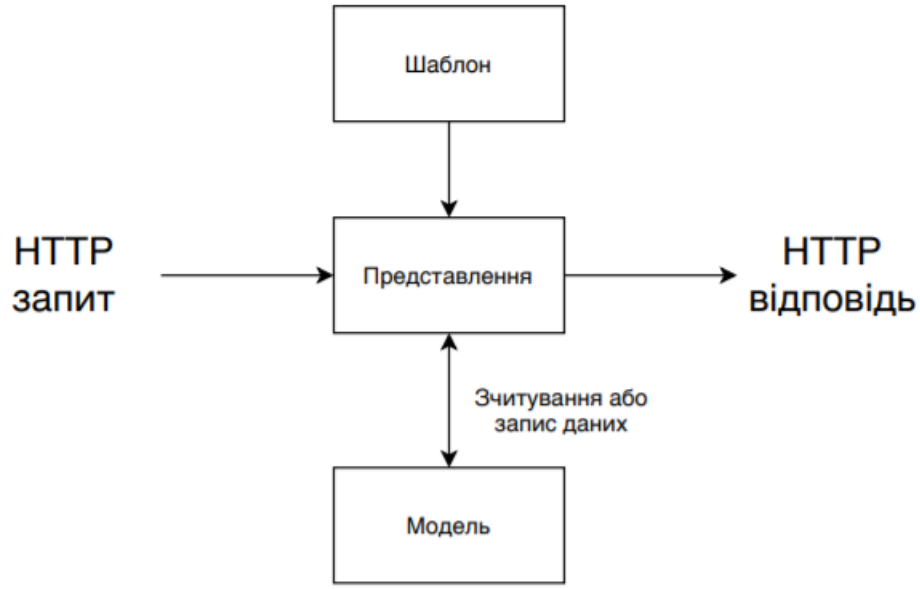


Рисунок 4.11 – Принцип роботи алгоритма «модель – представлення – шаблон»

### 4.3.5 Шаблони для веб-панелі

В Django реалізований потужний механізм роботи шаблонів та власна мова для їх розмітки [13]. Шаблони являють собою файли з HTML-кодом, за допомогою якого відображаються дані. Вміст файлів може бути статичним або динамічним. Шаблони не містять бізнес-логіки. Тому вони тільки відображають дані, які надає їм представлення.

В даному проекті для реалізації шаблонів було використано HTML. HTML це мова розмітки, яка дозволяє створювати та структурувати розділи, параграфи, заголовки, посилання та блоки для веб-сторінок. HTML не мова програмування, тому що він не має можливості створювати динамічні функції, він лише дозволяє організовувати та форматувати документ. При роботі з HTML використовуються прості структури – теги та атрибути, за допомогою яких відбувається розмітка веб-сторінки.

Для оформлення відображення HTML використовують CSS. CSS – це формальна мова для опису оформлення зовнішнього вигляду документа, створеного з використанням мови розмітки, тобто HTML. Призначення цієї мови в тому, щоб розділяти те, що задає зовнішній вигляд сторінки від її змісту. Можна визначати спосіб відображення кожного елементу (колір, шрифт, положення на сторінці. Зазвичай HTML описує лише послідовність розміщення елементів на сторінці, а за всі їх властивості та оформлення відповідає CSS.

Переваги CSS:

–забезпечує просту та швидку розробку, оскільки створене один раз оформлення елементу можна застосувати до інших;

–підвищує гнучкість та зручність редагування, тобто досить змінити щось в одному CSS класі, й оформлення зміниться усюди, де він використовується;

– робить код більш простим, зменшуючи кількість повторень;

–надає можливість легко застосовувати до одного й того ж документу різні стилі, в залежності від пристрою з якого його переглядають.

Тобто, CSS використовується не тільки для втілення рішень дизайну, а й значно спрощує роботу розробників та забезпечує гнучкість реалізації.

## 4.4 Розробка бази даних

В Django більшість взаємодій з базою даних здійснюються за допомогою механізму об’єктно-орієнтованого відображення (Object-Relational Mapper або ORM). Системи ORM автоматизують безліч типових взаємодій з базою даних та використовують знайомий об’єктно-орієнтований підхід замість SQL інструкцій [14]. Це означає, що коду стає значно менше ніж при використанні SQL запитів та він є більш читабельним. Також використовуючи ORM, розробник витрачає значно менше часу на написання та тестування коду, оскільки написання SQL складніше та легше зробити помилку.

Для розроблюваної системи була спроектована та створена база даних, структура якої наведена у додатку ІА62.110БАК.005.Д1, а опис та призначення таблиць описані далі.

telegram_user - в таблиці зберігаються дані про користувача незалежно від ролі (студент, адміністратор або викладач). Інформація міститься у наступних полях:

– ідентифікатор (id);

– пароль (password);

– логін (username);

– ім’я (first_name);

– прізвище (last_name);

– електронну пошту (email);

– телеграм ідентифікатор (user_id_int);

– скорочене телеграм ім’я (telegram_username);

– мова (language);

– роль (role);

– ідентифікатор групи (group_id).

telegram_group - таблиця зберігає інформацію про групи студентів. Має такі атрибути:

– ідентифікатор (id);

– назву (name);

– посилання для вступу студента (group_url);

– тип групи (group_type);

– ідентифікатор групи в базі даних університету (kpi_id);

– тип групи за часом навчання (group_okr).

telegram_studyresource - таблиця зберігає інформацію про методичні забезпечення. Інформація зберігається у наступних полях:

– ідентифікатор (id);

– назва (name);

– файл (file);

– ідентифікатор файла в телеграм чаті (file_tg_id);

– ідентифікатор власника методичного забезпечення (owner_id);

– ідентифікатор предмету до якого відноситься (subject_id).

telegram_subject - таблиця зберігає дані про предмети в системі, має наступні поля:

– ідентифікатор (id);

– назва (name);

– ідентифікатор власника предмету (owner_id).

telegram_verifyteacherrequest - таблиця містить інформацію про запити викладачів на верифікацію. Містить наступні поля:

– ідентифікатор (id);

– статус перегляду (is_view);

– статус підтвердження (is_accept);

– ідентифікатор користувача, який створив запит (user_id).

telegram_state - таблиця використовується для реалізації шаблону Стан, зберігає останній стан юзера. Містить наступні поля:

– ідентифікатор (id);

– назва стану (name);

– ідентифікатор користувача (user_id).

telegram_onlinequeue - таблиця містить в собі інформацію про електронні черги на здачу лабораторних робіт. Має такі поля:

– власний ідентифікатор (id);

– статус активності (active);

– ідентифікатор предмету з якого черга (subject_id);

– ідентифікатор власника черги (owner_id);

– дата початку черги (start_date).

telegram_groupschedule - Таблиця відповідає за розклад занять групи студентів. Мість такі поля:

– ідентифікатор (id);

– розклад в форматі JSON (schedule_json).

telegram_studentinqueue - єднальна таблиця, яка відповідає за зв’язок таблиць електронних черг та студентів. Містить три поля:

– власний ідентифікатор (id);

– ідентифікатор електронної черги (online_queue_id);

– ідентифікатор студента (student_id).

telegram_teacherbroadcast - таблиця містить в собі інформацію про розсилку створену викладачем до студентів групи. Має наступні поля:

– власний ідентифікатор (id);

– тип вкладеного файлу (content_type);

– ідентифікатор файлу в телеграм чаті (file_tg_id);

– чи заповнені всі поля розсилки (is_full);

– текст (text);

– ідентифікатор групи для розсилки (group_id).

### 4.4.3 Налаштування бази даних

В даній системі використовується база даних PostgreSQL інструкції щодо встановлення якої знаходяться на офіційному сайті документацій, а для підключення потрібно ввести необхідні дані до файлу налаштувань проекту settings.py [15]. Для цього ввести назву бази даних (name), пароль (password), користувача для підключення (user) та хост (host) бази даних – місце де запущений сам процес PostgreSQL (рисунок 4.12). Після виконання цих інструкцій програма має можливість для зчитування або запису даних у відповідну базу.

В одному проекті можливо використання декількох баз даних, під різні цілі, тобто у разі розширення функціонала системи у майбутньому, для зменшення навантаження на одну базу даних, можливо використання іншої бази даних під ці цілі.

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.environ.get('DB_NAME'),
        'USER': os.environ.get('DB_USER'),
        'PASSWORD': os.environ.get('DB_PASS'),
        'HOST': os.environ.get('DB_HOST'),
        'OPTIONS': {
            'sslmode': 'disable',
        },
    },
}
```

Рисунок 4.12 – Налаштування підключення до бази даних PostgreSQL

Для безпеки дані для підключення не вводяться на пряму, а зберігаються у файлі під назвою .env і тільки під час запуску проекту, вони дістаються звідти за назвами змінних [16]. Це потрібно для того, щоб при передачі коду до інших осіб, вони випадково не побачили цю інформацію. Файл .env повинен створюватись на кожному сервері чи комп'ютері окремо, оскільки ці налаштувань зазвичай унікальні для кожного. Приклад .env файлу для проекту зображено на рисунку 4.13.

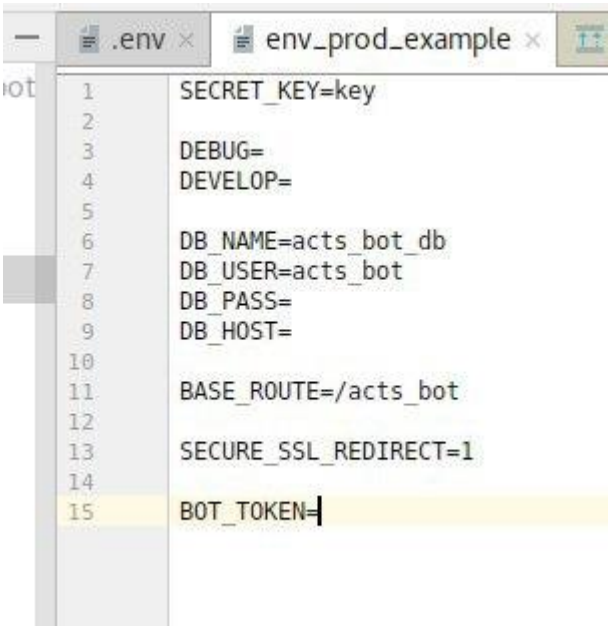


Рисунок 4.13 – Приклад .env файлу налаштувань бота

Для синхронізації моделей об’єктів та бази даних використовуються дві команди: python manage.py makemigrations та python manage.py migrate. Перша знаходить відмінності між моделями Django та об’єктами бази даних, та створює SQL міграцію, яка внесе зміни в базу даних, для синхронізації. Друга команда запускає міграції та виконує інструкції із файлі створених першою. Після цього дані стають синхронізовані.

Висновок до розділу 4

В даному розділі було описано власну реалізацію розроблюваної системи. Було описано взаємодію з Telegram Bot API, програмну реалізацію серверної частини телеграм-бота.

Був розглянутий принцип роботи backend та frontend частин системи, які написані на фреймворкі Django, реалізуючий шаблон проектування (модель – представлення – шаблон).

Також детально був описаний пункт проектування бази даних, з оглядом усіх таблиць які були створені під час реалізації системи.

# 5 ІНСТРУКЦІЯ КОРИСТУВАЧА

## 5.2 Чат-бот для викладача

### 5.2.3 Процес реєстрації

Процес реєстрації в системі розпочинається з того, що потрібно ініціювати діалог з ботом за допомогою кнопки «start». Якщо користувач раніше не писав боту, першим етапом буде потрібно обрати свою роль (рисунок Б.1), за допомогою відповідних кнопок на віртуальній клавіатурі.

Наступним кроком після того, як викладач обрав відповідну роль, йому потрібно ввести своє прізвище, ім’я та по батькові (рисунок Б.2).

Після введення викладачем свого ПІБ, він отримає повідомлення про те, що його заявка на верифікацію прийнята і йому потрібно буде зачекати, доки її розгляне адміністратор системи.

На цьому етапі заявка на верифікацію з’являється в адмін-панелі адміністратора. Коли заявка буде прийнята, система автоматично генерує для викладача логін та пароль для входу в особистий кабінет та надсилає цю інформацію користувачу (рисунок Б.3). Генерація пароля відбувається з використанням модулю Python Secrets [17].

Також в цей час відбувається інтеграція з rozklad KPI API, а саме пошук інформації про предмети та розклад викладача, які будуть відображені у відповідних пунктах меню в боті. На цьому реєстрація викладача завершена й він попадає в головне меню бота (рисунок Б.4).

### 5.2.4 Мої предмети

Цей пункт створений та використовується для перегляду, видалення або додавання предметів, які викладає викладач. Для цього користувачу потрібно перейти в відповідні підпункти за допомогою віртуальної клавіатури (рисунок Б.5).

Для повернення в головне меню потрібно натиснути на кнопку «Назад».Для того щоб додати предмет, потрібно ввести його назву (рисунок Б.6).

Список усіх предметів, з пагінацією по п’ять, можно подивитись натиснувши кнопку “Подивитись всі”. Всі предмети викладача виводяться окремими повідомленнями та під кожним є кнопка для видалення дисципліни (рисунок Б.7). Ці предмети автоматично підтягуються з бази даних rozklad KPI API.

### 5.2.5 Моє методичне забезпечення

Цей пункт використовується для додавання викладачем методичного забезпечення з предметів, які він викладає, для подальшого перегляду цих матеріалів студентами. Він поділяється на підпункти перегляду створених матеріалів, та додавання нових (рисунок Б.8).

Для того, щоб додати нове методичне забезпечення з предмету, потрібно натиснути кнопку «Додати файл» та після цього заповнити обов’язкову інформацію. Потрібно обрати предмет для якого створити файл, з списку який надішле бот за допомогою кнопки “Обрати” під необхідним предметом.

Після вибору предмета, необхідно надіслати боту файл будь-якого формату (документ, аудіо, відео або фото) та додати опис або назву до матеріалу (рисунок Б.9).

### 5.2.6 Розсилка

В боті викладач має можливість створити розсилку, тобто надіслати повідомлення всім студентам з обраної групи. Для цього в головному меню потрібно натиснути кнопку «Розсилка».

Для створення розсилки спочатку необхідно обрати групу, якій надіслати повідомлення. Оскільки груп у викладача може бути багато, було реалізовано

зручний пошук за цими групами в боті. Викладач може ввести повну або часткову назву групи, й в разі введення часткової назви бот запропонує можливі варіанти у вигляді кнопок на віртуальній клавіатурі (рисунок Б.10).

Коли групу було обрано, далі необхідно ввести текст повідомлення, з можливістю додати до розсилки файл будь-якого формату або пропустити цей етап (рисунок Б.11).

Останній етап, це перевірка повідомлення та підтвердження його відправки студентам. Для підтвердження потрібно натиснути кнопку “Так”. Після цього кожен студент з обраної групи отримає це повідомлення з підписом, від кого воно прийшло (рисунок Б.12).

### 5.2.7 Електронні черги

Цей розділ бота призначений для створення електронної черги студентів для здачі лабораторних робіт з предмета.

Щоб створити електронну чергу необхідно натиснути кнопку «Створити електронну чергу». Після цього потрібно обрати предмет з якого створити чергу, та підтвердити запуск черги (рисунок Б.13). Після підтвердження онлайн черга стає активною й студенти можуть записатись на здачу робіт у відповідному розділі бота.

Для керування та перегляду статусу електронної черги, потрібно перейти в підпункт “Мої активні черги” за допомогою відповідної кнопки. В цьому розділі виводяться активні черги з короткою інформацією про них, а саме - назва предмета, дата створення, кількість студентів в черзі та ім’я студента чия зараз черга (рисунок Б.14).

Для управління чергою, потрібно натиснути на кнопку “Керувати” під необхідним повідомленням. Після цього користувач отримує інформацію про всіх студентів в електронній черзі та може керувати чергою за допомогою віртуальних кнопок (рисунок Б.15).

Коли студент завершить свою відповідь, викладач має натиснути кнопку «Наступний студент», щоб черга перейшла до наступного в електронній черзі студента. Він отримає про це повідомлення, а також наступний за ним студент, отримає повідомлення, про те що скоро настане його час для відповіді, тому потрібно готуватись.

При натисканні викладачем на кнопку «Завершити», електронна черга завершується, а всі хто залишився в черзі - отримають про це повідомлення.

### 5.2.8 Розклад

Щоб переглянути свій розклад викладач має натиснути на кнопку “Мій розклад” в головному меню бота. Після цього він отримає інформацію про свій розклад за тижнем та днем неділі, назву предмета та інформацію про аудиторію, де буде проходити заняття (рисунок Б.16).

## 5.3 Чат-бот для студента

### 5.3.3 Процес реєстрації

Для реєстрації студента в боті, йому потрібно отримати через старосту посилання на реєстрацію. Староста групи повинен зв’язатись з відповідальним за це адміністратором, та отримати від нього посилання для реєстрації студентів групи та надіслати його студентам. Після переходу по цьому посиланню користувач реєструється в боті, як студент групи закріпленої за цим посиланням та попадає в головне меню (рисунок Б.17).

### 5.3.4 Методичне забезпечення

Цей пункт використовується для перегляду студентом методичного забезпечення з предметів, яке додали викладачі. Для цього потрібно в головному

меню натиснути кнопку “Методичне забезпечення”, далі бот спитає з якого предмету шукати файли. Студент може ввести повну або часткову назву та в разі введення часткової назви, бот запропонує можливі варіанти предметів з яких потрібно вибрати один (рисунок Б.18).

Після вибору предмету студент побачить список з методичного забезпечення, яке додали за цим предметом в бота з можливістю скачати ці файли (рисунок Б.19).

### 5.3.5 Електронні черги

Студент має можливість записатись в електронну чергу, яку створив викладач, для здачі лабораторних робіт. Для цього йому потрібно перейти в відповідний пункт меню за допомогою кнопки “Онлайн черги”. Цей пункт поділяється на два підпункти (рисунок Б.20), а саме - пошук черги та перегляд тих, в яких студент вже зареєструвався.

Для пошуку черги потрібно натиснути кнопку “Знайти чергу” та ввести часткову або повну назву предмета чергу з якого шукає користувач. Після вибору предмета бот надішле активні черги з детальною інформацією, такою як: назва предмета, дата створення черги, кількість студентів в черзі, ім’я студента який зараз відповідає та загальний список студентів в черзі. Під повідомленням інформації про чергу є кнопка яка використовується для того щоб встати в кінець черги (рисунок Б.21).

Після того як користувач встав в чергу бот додає його в загальний список студентів в черзі та попередить коли він буде наступним й коли настане його час йти відповідати до викладача.

Студент має можливість слідкувати за тим як рухається черга за допомогою пункту “Мої активні онлайн черги”. Перейшовши туди він побачить список електронних черг в яких він зареєструвався з детальною інформацією про кожну

та за бажанням зможе вийти з черги за допомогою кнопки “Вийти з черги” (рисунок Б.22).

### 5.3.6 Розклад студента

Студент завжди має можливість швидко та зручно переглянути свій розклад занять в боті. Для того щоб це зробити потрібно натиснути кнопку “Мій розклад” в головному меню. Після цього бот надішле розклад групи студента, за тижнями та днями неділі з інформацією про назву предмета та номер аудиторії, де пройде заняття (рисунок Б.23).

## 5.4 Веб-панель головного адміністратора

В системі створена веб-панель для адміністратора, який є відповідальним за валідацію викладачів та надання старостам посилань на групу для реєстрації студентів в боті.

### 5.4.3 Верифікація викладачів

Для перегляду запитів на верифікацію викладачів потрібно перейти на сторінку «Запити викладачів», посилання на яку знаходиться на панелі навігації зверху сайта (рисунок Б.24).

Адміністратор може підтвердити, або відхилити заявку за допомогою відповідної кнопки напроти користувача. При будь-якому виборі адміністратора, викладач отримає повідомлення, що його заявку прийняли або відхилили. Після цього заявка зникає з списка очікуючих на валідацію.

### 5.4.4 Управління групами

Цей розділ сайту використовується для перегляду, редагування, видалення або створення груп студентів в боті. Щоб потрапити на цю сторінку потрібно натиснути на посилання «Групи» на панелі навігації сайта. Далі одразу бачимо групи, які виводяться в вигляді таблиці з детальною інформацією про кожну (рисунок Б.25).

Групи виводяться з пагінацією по десять на сторінці, та для переходу між сторінками створено окремий інтерфейс знизу. Оскільки груп в боті багато, було додано пошук за назвою групи, який знаходиться зверху сторінки.

Адміністратор має можливість видалити групу натиснувши на кнопку «Видалити» напроти відповідної групи або редагувати її натиснувши на кнопку «Редагувати». Також він може додавати групи вручну за допомогою відповідної сторінки (рисунок Б.26) для переходу на яку потрібно натиснути кнопку «Додати».

Для створення групи потрібно заповнити поле “назва” та натиснути кнопку зберегти, після цього на групу автоматично згенерується посилання за яким студенти зможуть зареєструватись в цій групі. Аналогічно виглядає сторінка редагування групи.

## 5.5 Веб-панель викладача

Крім функціонала в боті, викладач також має можливість доступу до деяких функцій бота з веб-сайту в особистому кабінеті. В особистому кабінеті викладача доступні схожі функції, з тими що доступні в боті. Але використання бота не завжди зручне, коли користувач працює з комп’ютера без встановленого додатку месенджера Телеграм.

### 5.5.3 Логін

Після успішної реєстрації та проходження верифікації від адміністратора викладач отримує повідомлення з інформацією про логін, пароль та посилання на сторінку логіна (рисунок Б.27) до особистого кабінету.

### 5.5.4 Мої предмети

Ця сторінка дублює функціонал бота в перегляді предметів викладача, а також можливість їх створювати, редагувати та видаляти. Предмети виводяться в таблиці з пагінацією (рисунок Б.28).

Для редагування назви предмета потрібно натиснути кнопку «Редагувати» напроти необхідного в таблиці.

Для повного видалення предмета з бота, потрібно натиснути кнопку «Видалити» та після цього він більше не буде доступний в боті.

Для створення нового предмету потрібно натиснути кнопку “Додати” після чого користувача перенаправить на іншу сторінку з необхідним інтерфейсом (рисунок Б.29).

### 5.5.5 Методичне забезпечення

Для того щоб потрапити на цю сторінку потрібно натиснути на кнопку “Методичне забезпечення” в панелі навігації сайта. В цьому пункті знаходиться таблиця з доданим методичним забезпеченням, яке виводиться з пагінацією та інформацію про файл, предмет та опис. (рисунок Б.30).

Щоб видалити файл, потрібно натиснути кнопку “Видалити” напроти потрібного файла.

Для редагування потрібно натиснути кнопку “Редагувати”, та заповнити форму редагування.

Для створення нового файлу потрібно натиснути кнопку “Додати” та заповнити необхідні поля в окремому інтерфейсі (рисунок Б.31), який є аналогічним до інтерфейсу редагування файлу. Після створення нового методичного забезпечення воно одразу стає доступним для перегляду в боті.

Висновок до розділу 5

В цьому розділі були описані та надані детальні інструкції щодо використання системи для різних видів користувачів, а саме – для студента, викладача та головного адміністратора. Також були наведені різні приклади та демонстрації можливостей та функцій чат-бота та веб-панелі.

# ВИСНОВКИ

В дипломному проекті було реалізовано систему для інформаційної підтримки навчального процесу за допомогою телеграм-бота та адмін-панелі для керування ним. Було проведено ретельний аналіз існуючих рішень для визначення їх переваг та недоліків, які були враховані при власній реалізації завдання.

В ході розробки системи був проведений огляд та аналіз різних технічних засобів, серед яких через свої переваги були аргументовано обрані: мова програмування Python, веб-фреймворк Django для реалізації адмін-панелі та об’єктно-реляційна база даних PostgreSQL.

Створена система відповідає всім поставленим вимогам дипломної роботи, має широкий та зручний функціонал для підтримки навчального процесу, спрощує зв’язок студентів та викладачів, надає доступ до інформації доступної в боті цілодобово та незалежно від місця знаходження користувача, що є особливо важливим під час дистанційного навчання.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Python Developers Guide [Електронний ресурс]: Режим доступу: https://devguide.python.org/
2. Документація веб-фреймоворка Django [Електронний ресурс]: Режим доступу: https://docs.djangoproject.com/en/3.0/
3. Документація бази даних PostgreSQL [Електронний ресурс]: Режим доступу: https://www.postgresql.org/docs/
4. Стаття «Створення Telegram Bot» [Електронний ресурс]: Режим доступу: https://codeguida.com/post/410
5. Стаття «Telegram Bots» [Електронний ресурс]: Режим доступу: https://core.telegram.org/bots
6. Стаття «Простым языком об HTTP» [Електронний ресурс]: Режим доступу: https://habr.com/ru/post/215117/
7. Документація JSON [Електронний ресурс]: Режим доступу: https://www.json.org/json-en.html
8. Документація Telegram Bot API [Електронний ресурс]: Режим доступу: https://core.telegram.org/bots/api#available-methods
9. Документація бібліотеки pyTelegramBotAPI [Електронний ресурс]: Режим доступу: https://github.com/eternnoir/pyTelegramBotAPI
10. Документація бази даних rozklad KPI API [Електронний ресурс]: Режим доступу: https://api.rozklad.org.ua/
11. Стаття «State Design Pattern» [Електронний ресурс]: Режим доступу: https://sourcemaking.com/design_patterns/state
12. Стаття «Starting a Django Project» [Електронний ресурс]: Режим доступу: https://realpython.com/django-setup/
13. Стаття «Django’s Templates» [Електронний ресурс]: Режим доступу: https://djangobook.com/mdj2-django-templates/

14. Стаття «SQL vs ORM» [Електронний ресурс]: Режим доступу: https://habr.com/ru/company/pgdayrussia/blog/328690/
15. Стаття «PostgreSQL - CREATE Database» [Електронний ресурс]: Режим доступу: https://www.tutorialspoint.com/postgresql/postgresql_create_database.htm
16. Стаття «Virtual Environments and Packages» [Електронний ресурс]: Режим доступу: https://docs.python.org/3/tutorial/venv.html
17. Python Secrets Module [Електронний ресурс]: Режим доступу: https://pynative.com/python-secrets-module/

# ДОДАТОК А

## Лістинг реалізації станів користувача в боті

```
class StartState(base_state.BaseState):
    has_entry = False

    def process_message(self, message, user, sender: SenderClient):
        ref_id = message.text.replace('/start', '').split()
        if ref_id:
            link_id = ref_id[0]
            group = Group.objects.filter(link_id=link_id).first()
            if group:
                if user.role == ROLES[1][0] and user.full_register:
                    sender.send_message(message.chat.id,
                                        get_translation_for(user.language, 'you_already_register_as_teacher_msg'),
                                        parse_mode='html')
                    return base_state.RET_GO_TO_STATE, 'MenuState', message, user
                elif user.role == ROLES[2][0] and user.full_register:
                    sender.send_message(message.chat.id,
                                        get_translation_for(user.language, 'you_already_register_as_student_msg'),
                                        parse_mode='html')
                    return base_state.RET_GO_TO_STATE, 'MenuState', message, user
                else:
                    user.role = ROLES[2][0]
                    user.save(update_fields=['role'])
                    user.set_group(group)
                    message_template = get_translation_for(user.language,
                                                           'you_choice_group_msg').format(
                        group.name
                    )
                    sender.send_message(message.chat.id,
                                        message_template,
                                        parse_mode='html')
                    return base_state.RET_GO_TO_STATE, 'MenuState', message, user

        sender.send_message(message.chat.id,
                            get_translation_for(user.language, 'welcome_msg')
                            )
```

```
class MenuState(base_state.BaseState):

    def __init__(self):
        super().__init__()
        self._buttons = {...}
        self._base_keyboard = get_main_menu_keyboard

    def entry(self, message, user, sender: SenderClient, token):

        keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
        keyboard.add(get_translation_for(language, 'my_subjects_btn'))
        keyboard.add(get_translation_for(language, 'my_study_resources_btn'))
        keyboard.add(get_translation_for(language, 'broadcast_btn'))
        keyboard.add(get_translation_for(language, 'online_queue_btn'))
        keyboard.add(get_translation_for(language, 'my_schedule_btn'))

        sender = TeleBot(token)
        sender.send_message(chat.id,
                            'Головне меню',
                            reply_markup=keyboard()
                            )

        return base_state.RET_OK, None, None, None

    @only_teacher
    def my_subjects_btn(self, message, user, sender: SenderClient):
        return base_state.RET_GO_TO_STATE, 'MySubjectsMenuState', message, user

    @only_teacher
    def my_study_resources_btn(self, message, user, sender: SenderClient):
        return base_state.RET_GO_TO_STATE, 'TeacherStudyResourcesMenuState', message, user
```

```
    @only_teacher
    def online_queue_btn(self, message, user, sender: SenderClient):
        return base_state.RET_GO_TO_STATE, 'TeacherQueueMenuState', message, user

    @only_student
    def student_queue_btn(self, message, user, sender: SenderClient):
        return base_state.RET_GO_TO_STATE, 'StudentQueueMenuState', message, user

    @only_student
    def study_resources_btn(self, message, user, sender: SenderClient):
        return base_state.RET_GO_TO_STATE, 'StudentChoiceSubjectForStudyResourcesState', message, user


class ChoiceRoleState(base_state.BaseState):

    def __init__(self):
        super().__init__()
        self._buttons['student_btn'] = self.student_btn
        self._buttons['teacher_btn'] = self.teacher_btn
        self._base_keyboard = choice_role_keyboard

    def entry(self, message, user, sender: SenderClient):
        sender.send_message(message.chat.id, get_translation_for(user.language, 'choice_role_msg'),
                            reply_markup=self._base_keyboard(user.language),
                            parse_mode='html')

        return base_state.RET_OK, None, None, None

    def student_btn(self, message, user, sender: SenderClient):
        return base_state.RET_GO_TO_STATE, 'StudentRegistrationState', message, user

    def teacher_btn(self, message, user, sender: SenderClient):
        user.role = ROLES[1][0]
        user.save(update_fields=['role'])
        return base_state.RET_GO_TO_STATE, 'TeacherFullNameState', message, user


class TeacherRegistrationState(base_state.BaseState):

    def __init__(self):...

    def keyboard(self, language='ua'):
        keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
        keyboard.add(get_translation_for(language, 'back_btn'))
        return keyboard

    def entry(self, message, user, sender: SenderClient):
        sender.send_message(message.chat.id, get_translation_for(user.language, 'input_teacher_name_msg'),
                            reply_markup=self._base_keyboard(user.language),
                            parse_mode='html')

        return base_state.RET_OK, None, None, None

    def process_message(self, message, user, sender: SenderClient):
        user.create_verify_request(message.text)
        return base_state.RET_GO_TO_STATE, 'TeacherWaitValidationState', message, user
```

```
class StudentRegistrationState(base_state.BaseState):

    def __init__(self):
        super().__init__()
        self._base_keyboard = self.keyboard

    def keyboard(self, language='ua'):...

    def entry(self, message, user, sender: SenderClient):
        sender.send_message(message.chat.id, get_translation_for(user.language, 'need_link_to_student_register_msg'),
                            reply_markup=self._base_keyboard(user.language),
                            parse_mode='html')

        return base_state.RET_OK, None, None, None

    def process_message(self, message, user, sender: SenderClient):
        group = Group.objects.filter(link=message.text).first()
        if group:
            user.role = ROLES[2][0]
            user.save(update_fields=['role'])
            user.set_group(group)
            message_template = get_translation_for(user.language,
                                                   'you_choice_group_msg').format(
                group.name
            )
            sender.send_message(message.chat.id,
                                message_template,
                                parse_mode='html')
            return base_state.RET_GO_TO_STATE, 'MenuState', message, user
        else:
            sender.send_message(message.chat.id,
                                get_translation_for(user.language, 'need_link_to_student_register_msg'),
                                reply_markup=self._base_keyboard(user.language),
                                parse_mode='html')
            return base_state.RET_OK, None, None, None
```

```
class AddSubjectState(base_state.BaseState):

    def __init__(self):
        super().__init__()
        self._base_keyboard = self.keyboard

    def keyboard(self, language='ua'):...

    @only_teacher
    def entry(self, message, user, sender: SenderClient):...

    @only_teacher
    def process_message(self, message, user, sender: SenderClient):...

    def back_button(self, message, user, sender: SenderClient):
        return base_state.RET_GO_TO_STATE, 'MySubjectsMenuState', message, user
```

```python
class MySubjectsState(base_state.BaseState):

    def __init__(self):
        super().__init__()
        self._base_keyboard = get_pagination_keyboard
        self._buttons['next_btn'] = self.next_btn
        self._buttons['prev_btn'] = self.prev_btn

    def inline_keyboard(self, subject_pk, language='ua'):...

    @only_teacher
    def entry(self, message, user, sender: SenderClient):...

    @only_teacher
    def process_message(self, message, user, sender: SenderClient):...

    def next_btn(self, message, user, sender: SenderClient):
        user.pagination_page += 1
        user.save(update_fields=['pagination_page'])
        self.send_subjects(message, user, sender)

        return base_state.RET_OK, None, None, None

    def prev_btn(self, message, user, sender: SenderClient):...


class StudentFindQueuesState(base_state.BaseState):

    def __init__(self):...

    def inline_keyboard(self, queue_pk, language='ua'):...

    def entry(self, message, user, sender: SenderClient):...

    @only_student
    def process_message(self, message, user, sender: SenderClient):...

    def next_btn(self, message, user, sender: SenderClient):...

    def prev_btn(self, message, user, sender: SenderClient):...

    def send_active_queues(self, message, user, sender: SenderClient):
        queues = self.get_queues_for_args(message, user, sender)
        queues_for_send = queues[(user.pagination_page * 5):(
                (user.pagination_page + 1) * 5)]
        if queues_for_send:
            sender.send_message(message.chat.id, get_translation_for(user.language,
                                                                     'search_queue_entry'),
                                parse_mode='html', reply_markup=self._base_keyboard(user.language,
                                                                                    user.pagination_page,
                                                                                    queues))
            for queue in queues_for_send:
                sender.send_message(message.chat.id, queue.get_queue_full_info_for_student(user.language, user),
                                    reply_markup=self.inline_keyboard(queue.pk, user.language),
                                    parse_mode='html')
        else:
            sender.send_message(message.chat.id, get_translation_for(user.language,
                                                                     'search_queue_not_found'),
                                parse_mode='html')
```

# ДОДАТОК Б

Ілюстрації до інструкції користувача

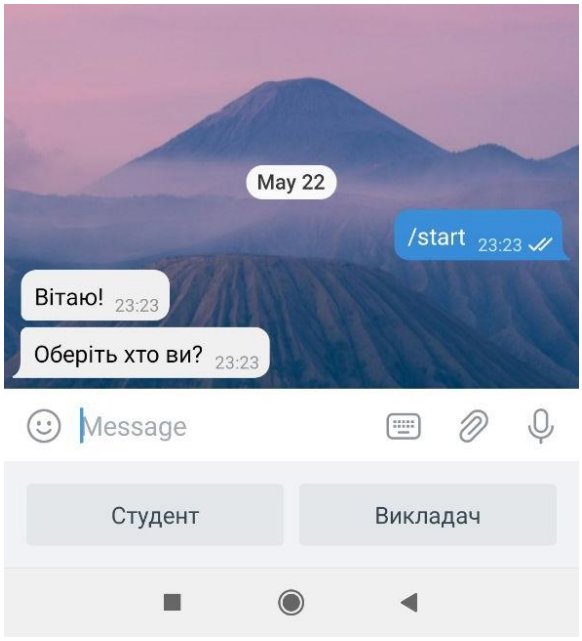


Рисунок Б.1 – Вибір ролі

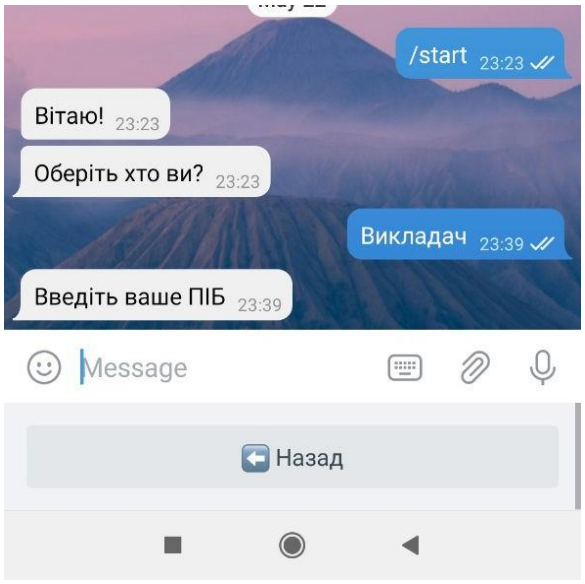


Рисунок Б.2 – Введення ПІБ

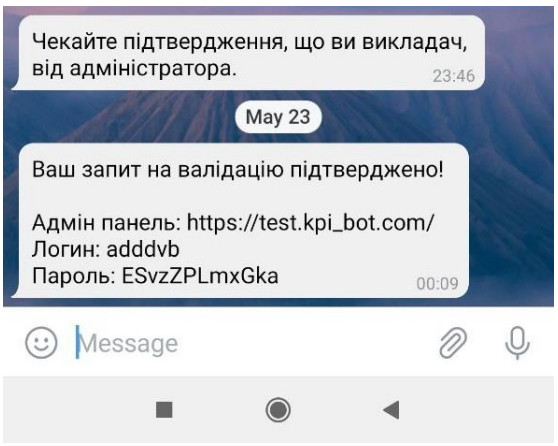


Рисунок Б.3 – Повідомлення про підтвердження запиту на валідацію

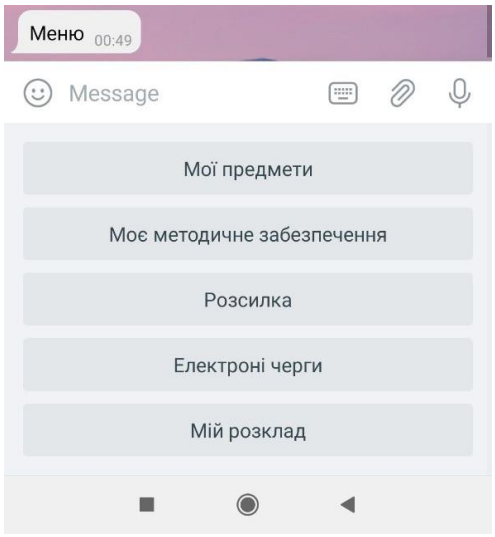


Рисунок Б.4 – Головне меню

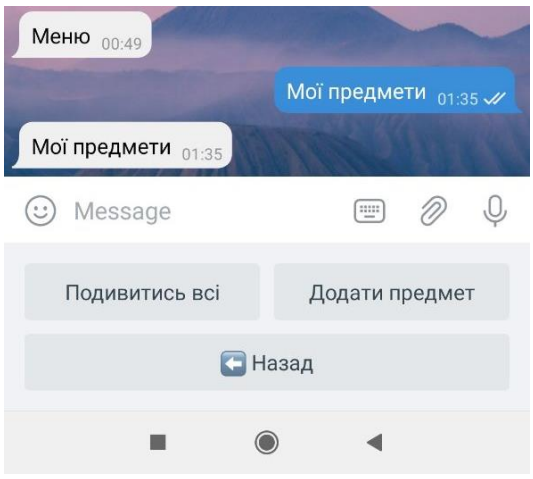


Рисунок Б.5 – Меню пункту мої предмети

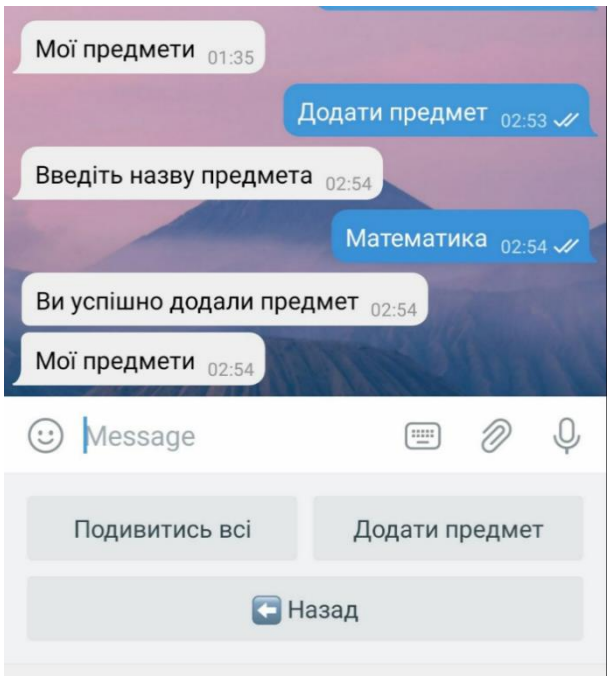


Рисунок Б.6 – Додавання нового предмету

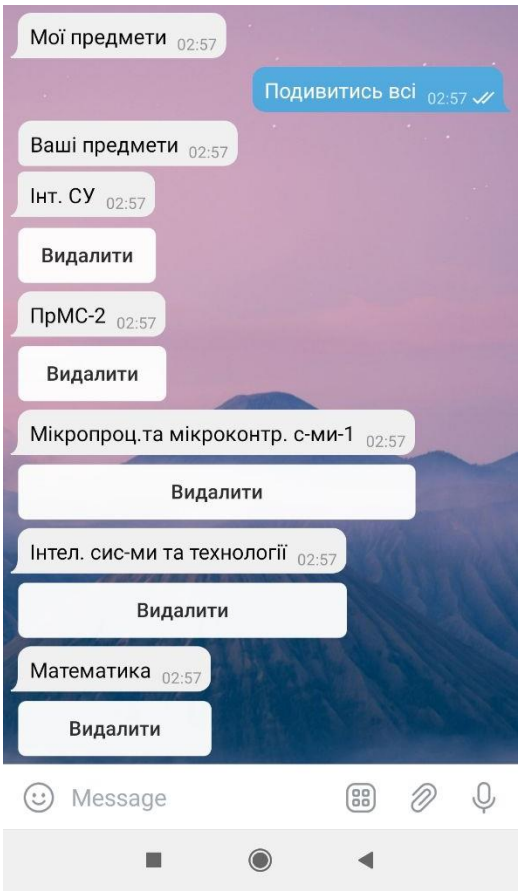


Рисунок Б.7 – Список предметів викладача

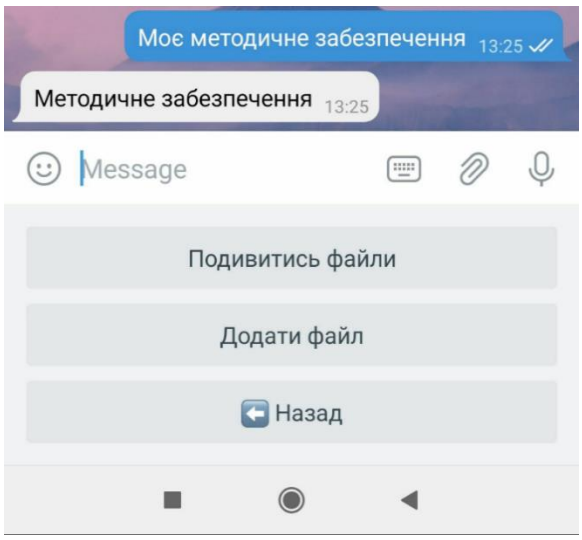


Рисунок Б.8 – Підпункти меню методичного забезпечення

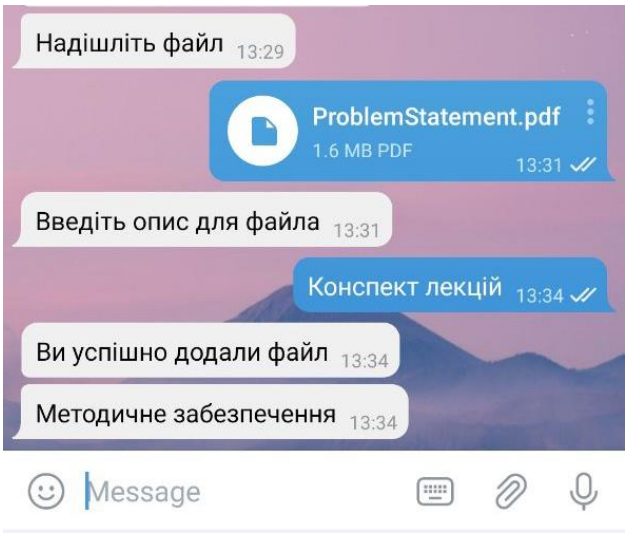


Рисунок Б.9 – Додавання файлу та його опису для методичного забезпечення

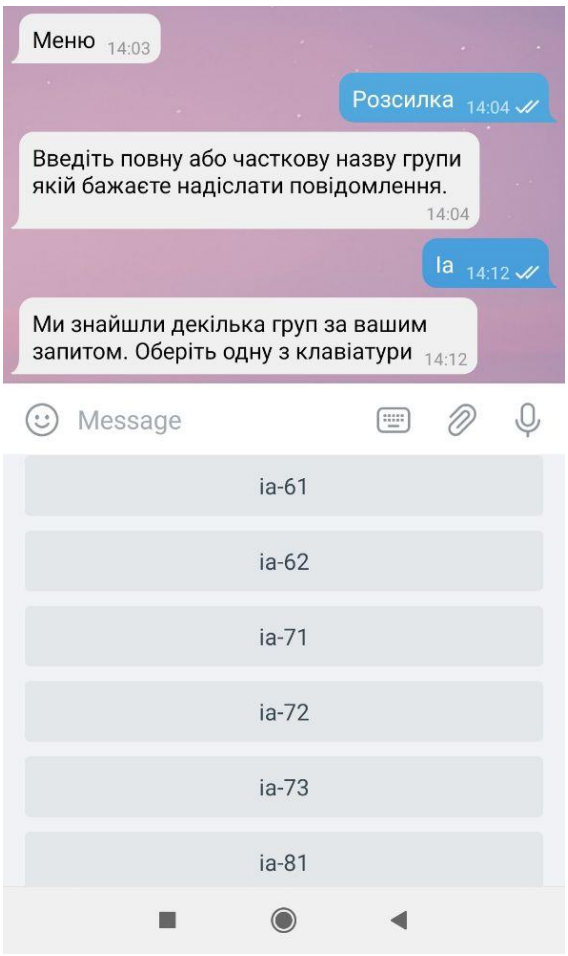


Рисунок Б.10 – Вибір групи для розсилки повідомлення


Рисунок Б.11 – Створення повідомлення для розсилки

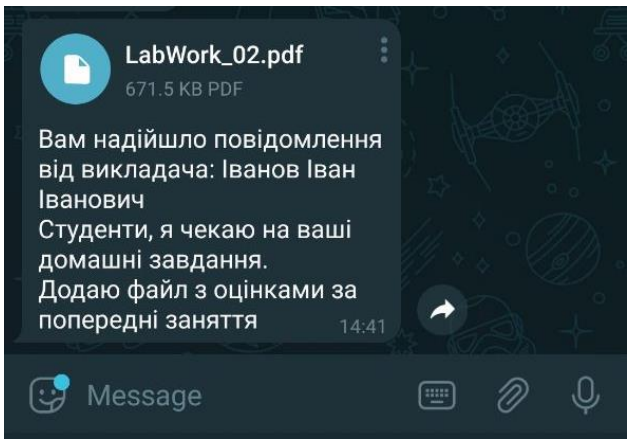


Рисунок Б.12 – Студент отримав повідомлення від викладача

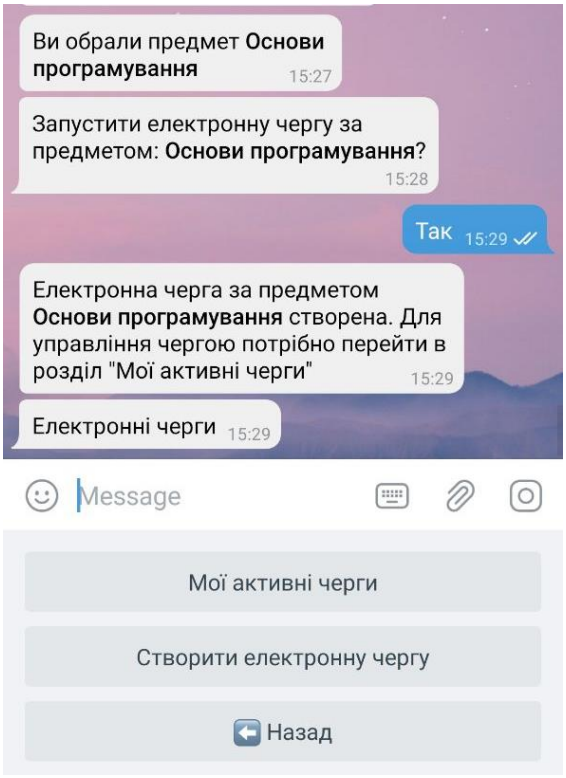


Рисунок Б.13 – Процес створення електронної черги

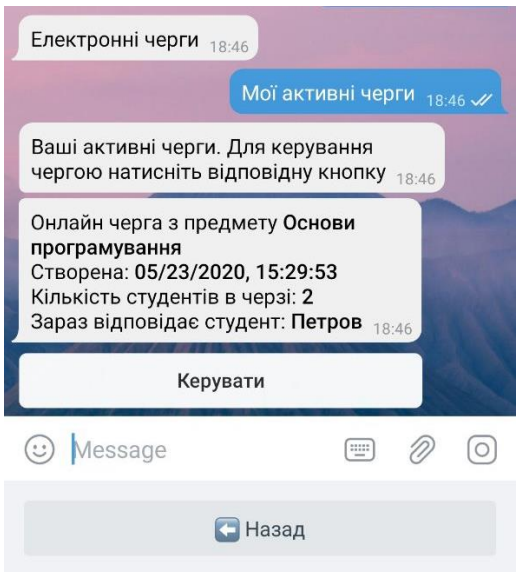


Рисунок Б.14 – Інформація про активні черги

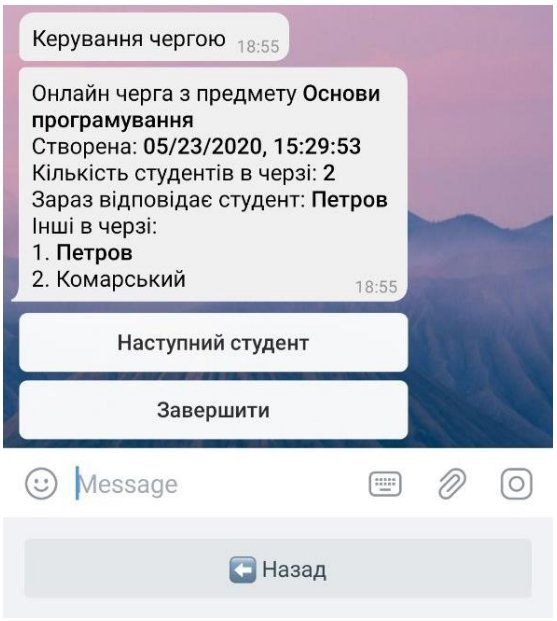


Рисунок Б.15 – Детальна інформація про електронну чергу

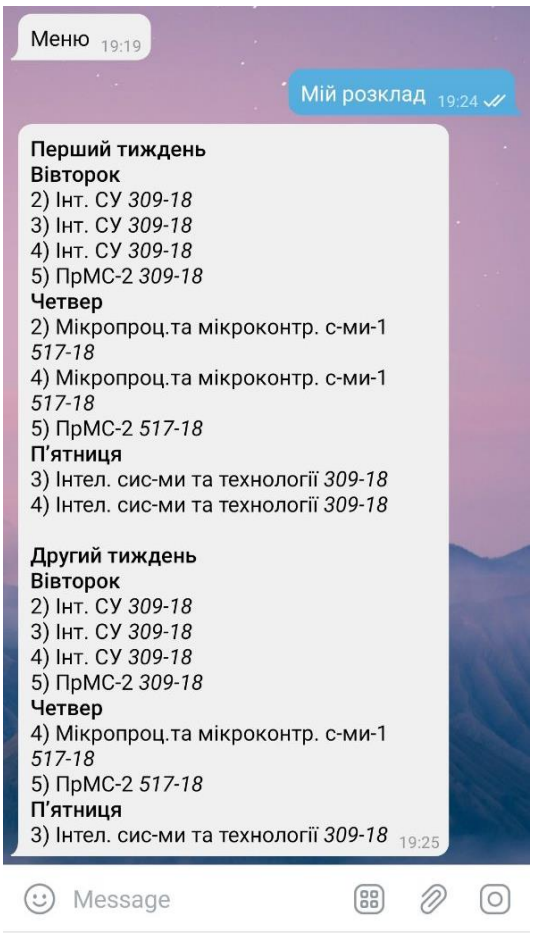


Рисунок Б.16 – Інформація про розклад викладача

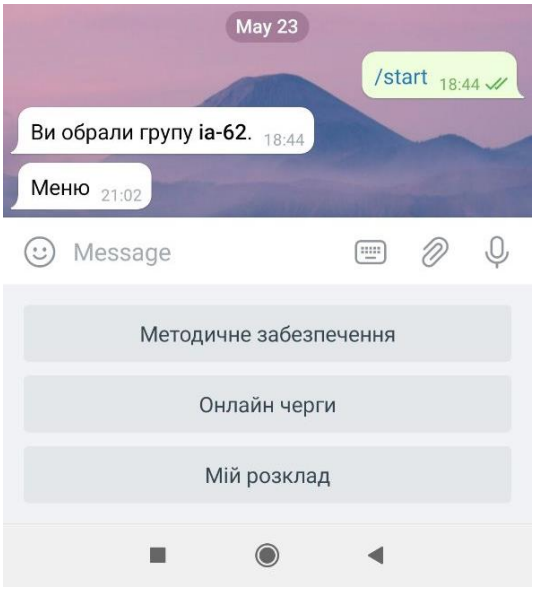


Рисунок Б.17 – Головне меню бота

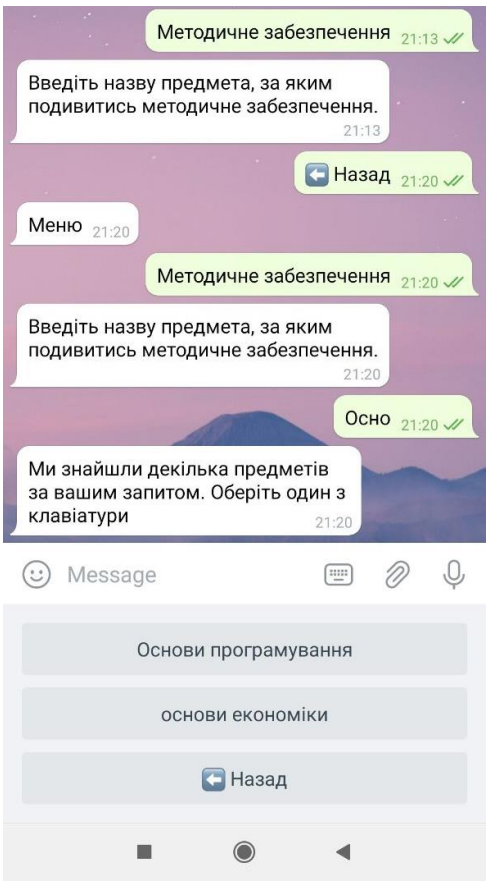


Рисунок Б.18 – Пошук предмета для перегляду методичного забезпечення

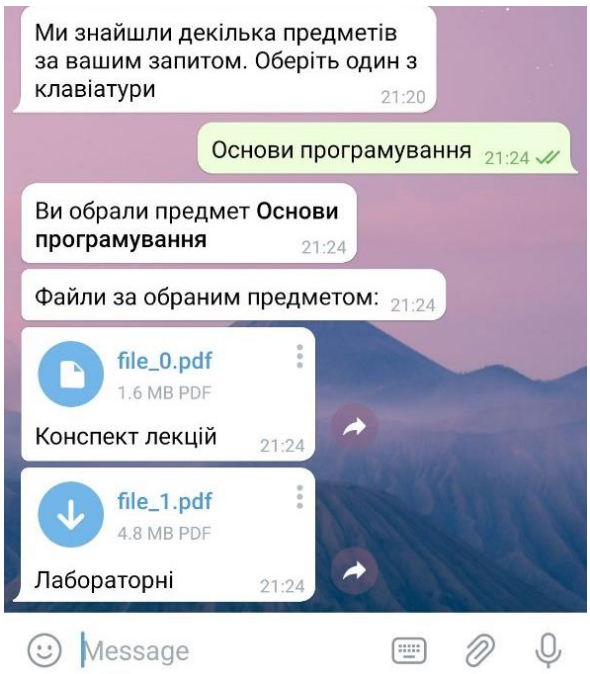


Рисунок Б.19 – Перегляд методичного забезпечення

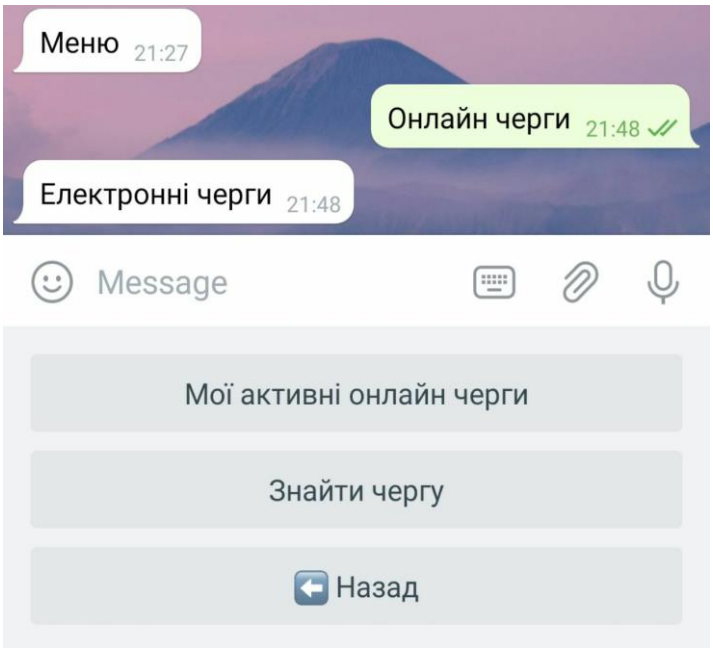


Рисунок Б.20 – Меню електронних черг студента

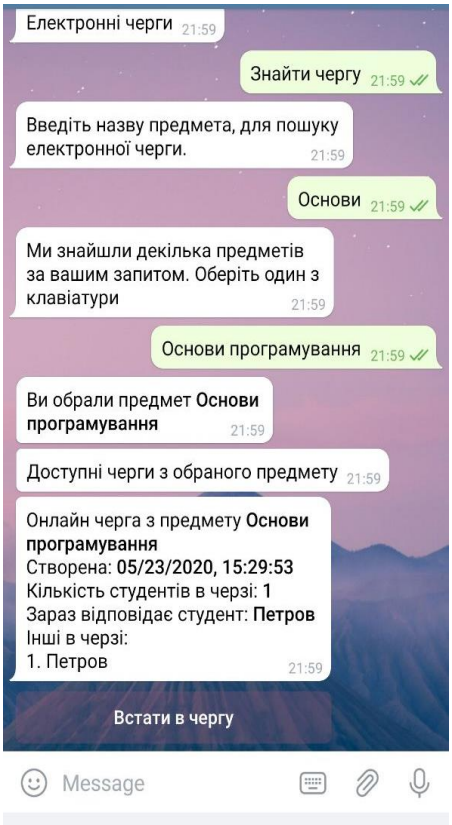


Рисунок Б.21 – Пошук електронної черги

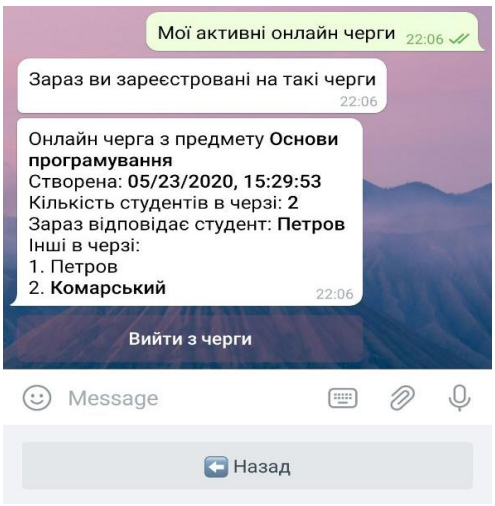


Рисунок Б.22 – Перегляд активних електронних черг студента

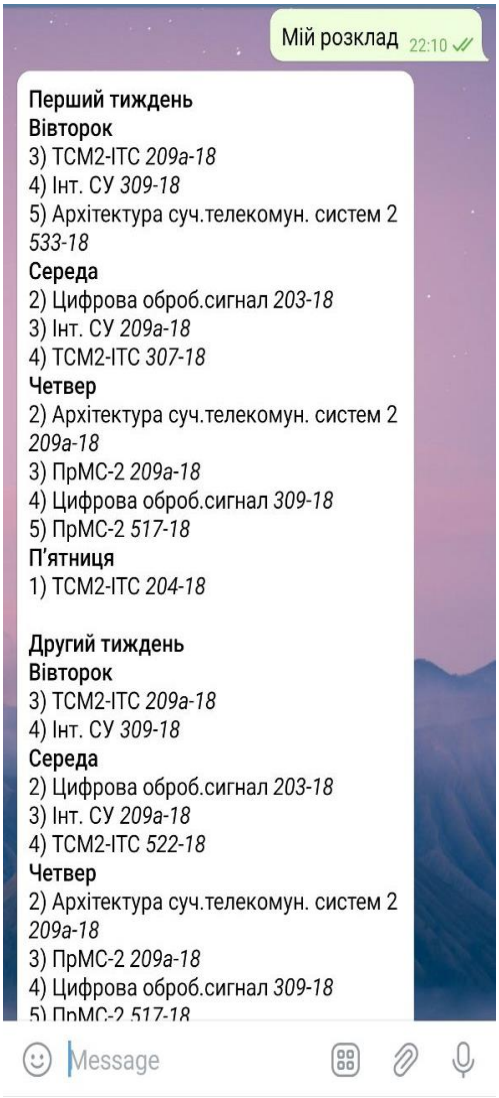


Рисунок Б.23 – Перегляд розкладу студента

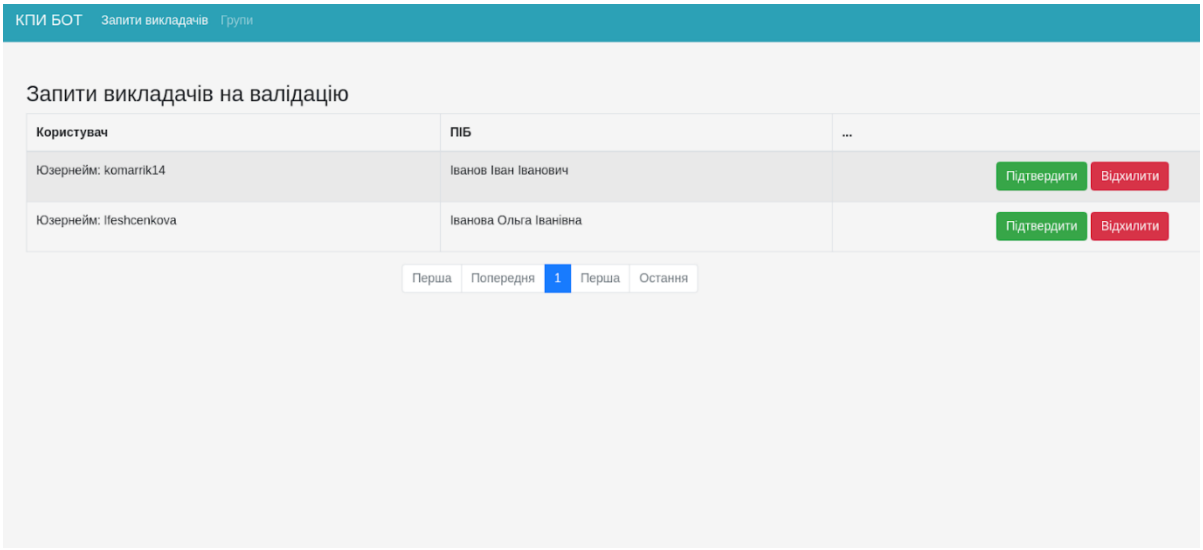


Рисунок Б.24 – Сторінка верифікації викладачів

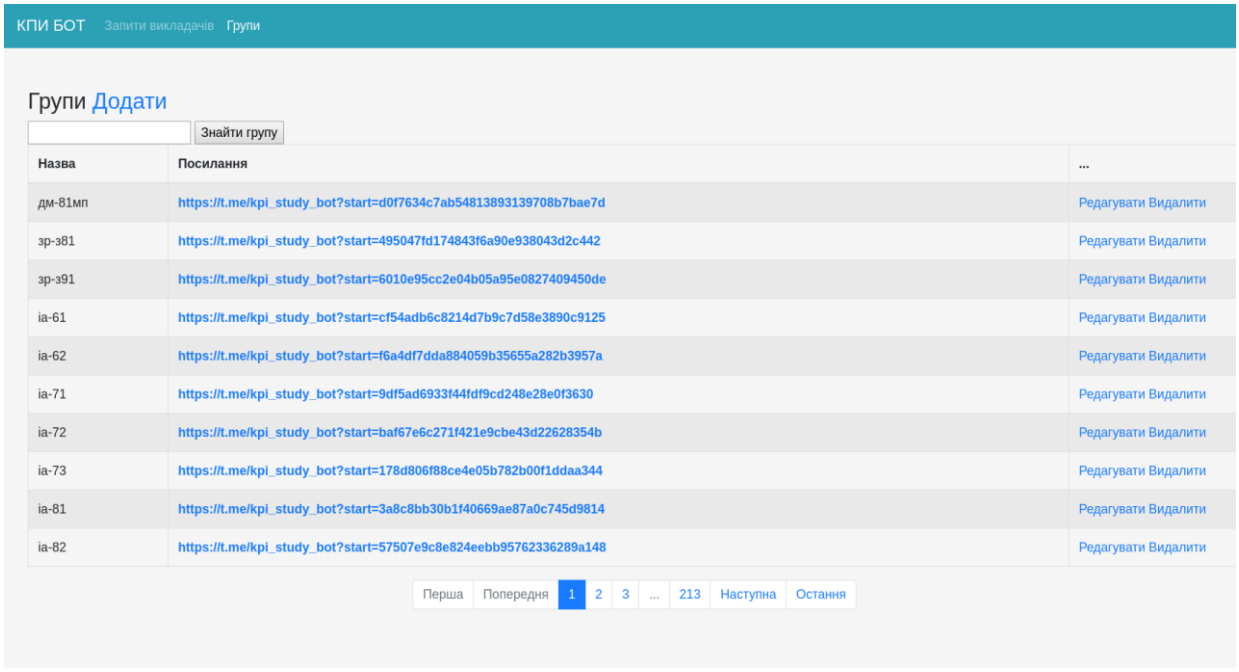


Рисунок Б.25 – Сторінка перегляду груп


Рисунок Б.26 – Сторінка створення групи

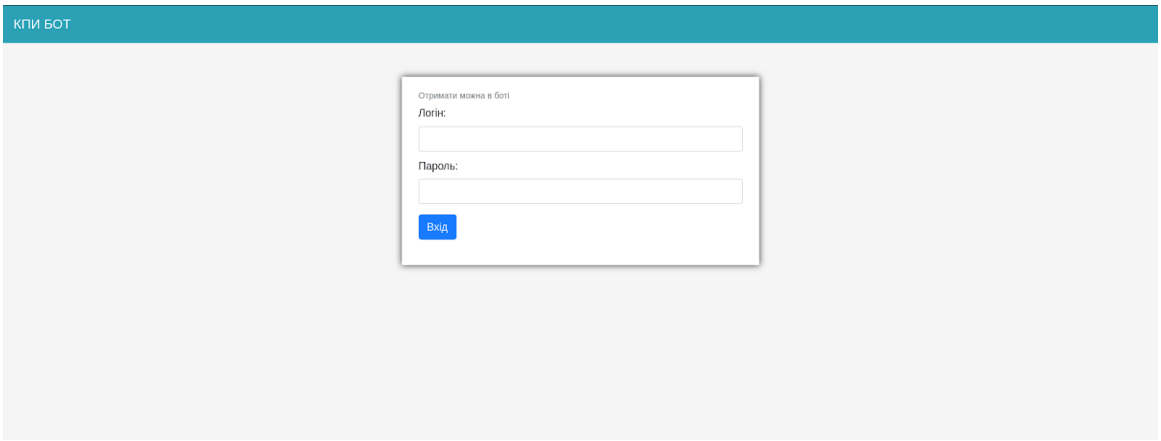


Рисунок Б.27 – Сторінка входу

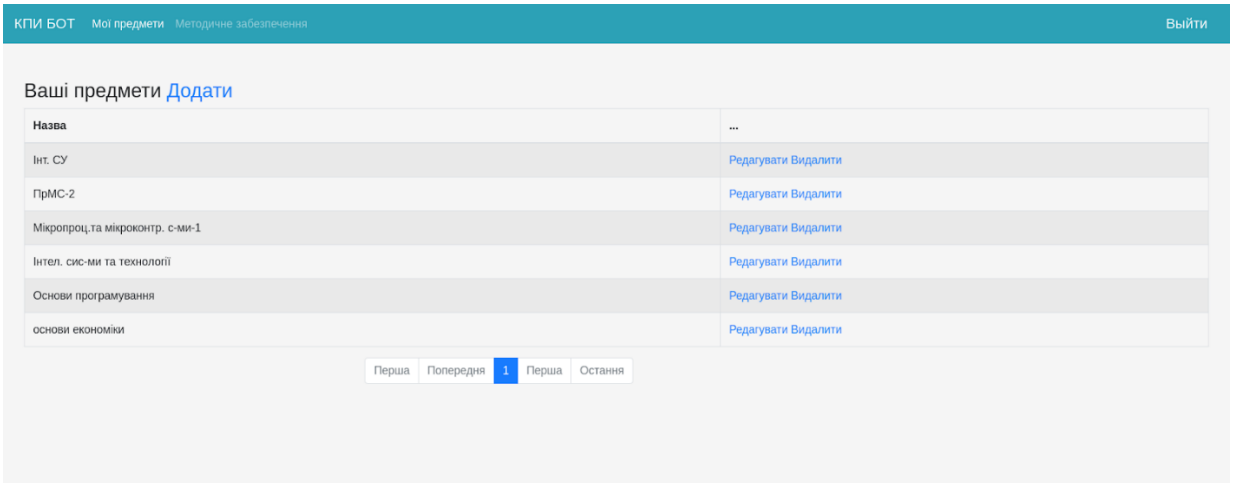


Рисунок Б.28 – Сторінка перегляду предметів викладача

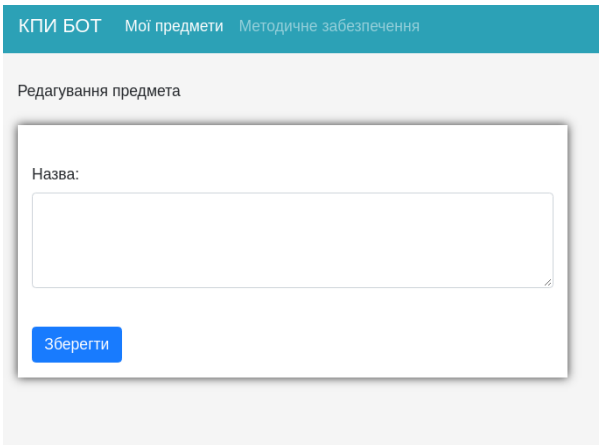


Рисунок Б.29 – Сторінка редагування предмета

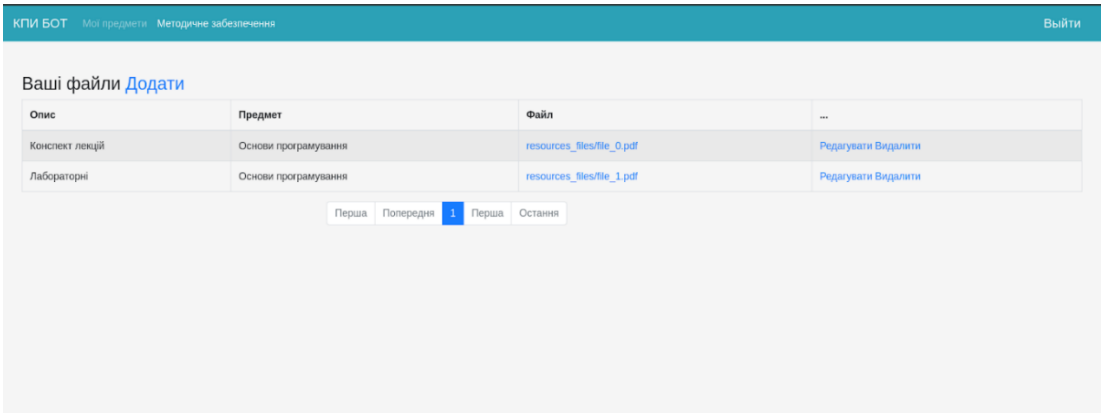


Рисунок Б.30 – Сторінка методичного забезпечення

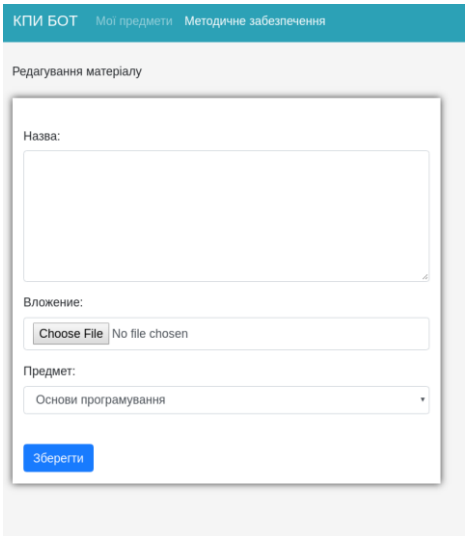


Рисунок Б.31 – Сторінка редагування файлу методичного забезпечення