МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

_____ червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,

освітньо- професійної програми «Інформатика»

на тему: «Інформаційна система розкладу навчального закладу»

здобувача групи ІН-92 Лифар Ігоря Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Ігор ЛИФАР
(підпис)

Керівник,
старший викладач, кандидат фізико-
математичних наук Галина ОЛЕКСІЄНКО _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-92 Лифар Ігоря Сергійовича

1. Тема роботи: «Інформаційна система розкладу навчального закладу» затверджую наказом по СумДУ від *«09» червня 2023 р. № 0646-VI0*
2. Термін здачі здобувачем кваліфікаційної роботи *до 16 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз літератури за темою практики. 2) Огляд існуючих інформаційних систем. 3) Вибір інструментів розробки веб-додатку. 4) Розробка інформаційної системи розкладу навчального закладу. 5) Аналіз результату.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 _23_ р.

Завдання прийняв до виконання _____ (підпис)
Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз літератури за темою практики*		
2	*Огляд існуючих інформаційних систем*		
3	*Вибір інструментів розробки веб-додатку*		
4	*Розробка інформаційної системи розкладу навчального закладу*		
5	*Аналіз результату*		
6	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 73 стр., 20 рис., 3 таблиць, 2 додатки, 19 використаних джерел.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки присвячена вирішенню проблем з відсутністю можливості доступу до функціоналу існуючої системи через Telegram, відсутності зручного функціоналу пошуку найближчої пари в межах поточного дня, можливості призначення посилання на пару для відповідного виду заняття певної дисципліни, а також швидкого додавання альтернативних пар. Розробка відповідної інформаційної системи розкладу дозволить усунути наведені вищі проблеми та дозволить оптимізувати роботу з розкладом й організацію навчального процесу.

Об'єкт дослідження – Інформаційна система розкладу навчального закладу.

Мета роботи – розробка веб-додатку розкладу для оптимізації організації навчального процесу.

Методи дослідження – аналіз існуючої інформаційної системи розкладу навчального закладу, варіантів доступу до неї, її функціональних можливостей та недоліків, пошук доступних і оптимальних інструментів для розробки веб-додатку.

Результати – розроблено ІС розкладу навчального закладу, яка розширила й інтегрувала у Telegram функціонал існуючої, реалізувавши можливості пошуку пари за часом; призначення посилань до занять та створення альтернативних пар студентами-старостами. Розроблена інформаційна система реалізована у вигляді веб-додатку, що використовує в

якості інтерфейсу можливості Telegram Bot API.

ІНФОРМАЦІЙНА СИСТЕМА, API JSON, TELEGRAM BOT API, C#, TELEGRAM.BOT FRAMEWORK, POSTGRES SQL, ERD, РІВНІ АРХІТЕКТУРИ.

ЗМІСТ

ВСТУП.....7
1. ІНФОРМАЦІЙНИЙ ОГЛЯД.....9
1.1. Огляд існуючої інформаційної системи.....9
1.2. Огляд аналогів інформаційних систем інших навчальних закладів.....11
1.3. Постановка задачі.....14
2. ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ ВЕБ-ДОДАТКУ.....17
2.1. Вибір відкритого API Telegram.....17
2.2. Вибір мови програмування.....20
2.3. Вибір фреймворків.....23
2.4. Вибір мови СУБД.....24
3. ПРОГРАМНА РЕАЛІЗАЦІЯ.....26
3.1. Побудова діаграми потоків даних.....26
3.2. Проектування бази даних.....28
3.3. Логічна реалізація бази даних.....30
3.4. Розробка веб-додатку під API існуючої системи.....33
3.5. Алгоритм роботи.....35
ВИСНОВКИ.....43
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....44
ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО КОДУ.....46
ДОДАТОК Б ФІЗИЧНА РЕАЛІЗАЦІЯ БД ВЕБ-ДОДАТКУ.....72

ВСТУП

Актуальність. Актуальність роботи обумовлена потребою інтеграції існуючої ІС розкладу навчального закладу у Telegram як можливість швидкого, альтернативного доступу до розкладу та необхідністю нового функціоналу, який би дозволив проводити швидкий пошук за параметром часу в межах дня й вирішив би питання з постійним місцем зберігання актуальних посилань на пари й відповідного пошуку потрібної в залежності від виду заняття. Завдяки правильно підібраній архітектури, яка використовується в даній роботі, забезпечується гнучкість та простота можливості розширення, таким чином дана система може бути доповнена відповідним функціоналом орієнтованим як для викладачів так і для співробітників деканату, що є гарним аргументом перспективності розробленої ІС.

Об’єкт дослідження. Процес розробки інформаційної системи розкладу навчального закладу.

Предмет дослідження. Функціональні можливості існуючої та розроблюваної ІС навчального закладу, механізм взаємодії між ними та з ними через Telegram.

Гіпотеза. Розроблена ІС вирішить основні недоліки відсутності функціоналу для пошуку і роботи з розкладом існуючої ІС , вирішить проблему зберігання та пошуку посилань для відповідних занять, пришвидшить доступ до існуючої ІС та організацію навчального процесу шляхом оптимізації роботи з розкладом.

Новизна. Розроблена система дозволить збільшити ефективність при роботі з існуючою ІС та розкладом . При чому, дана робота дозволить розкрити Telegram як додаткову платформу для впровадження навчального процесу навчального закладу.

Структура. Дана робота складається зі вступу, інформаційного огляду, постановки задачі, вибору інструментів розробки веб-додатку, програмної реалізації, висновків, списку використаних джерел та додатків.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1. Огляд існуючої інформаційної системи

Наразі, існуюча ІС розкладу навчального закладу реалізовано як сервіс доступний в Особистому Кабінеті сайту навчального закладу[1]. Також доступ до неї реалізовано через мобільний додаток Schedule SSU[2] та на сторонньому сайті My-university[3] у відповідному розділі серед перелічених навчальних закладів.

Розглянемо недоліки цих реалізацій в цілому:

1) Відсутній пошук за параметром часу у межах робочого дня;
2) Не зручне відображення для великих проміжків часу;
3) Не можливість внесення раптових змін. Наприклад: зміни відбулися менше ніж за день до заняття, раптова нарада, тощо;
4) В більшості випадків, відсутня інформація про посилання на пару в залежності від виду онлайн заняття. Якщо є посилання, то частіше лише для лекцій, яке не оновлюється;
5) Відсутність підтримки(інтеграції) у Telegram. Як коммерційне виключення, є можливість придбати дану підтримку на сторонньому сайті My-university[3].

Також, в результаті пошуку в Інтернеті було знайдено відкритий API JSON існуючої ІС розкладу навчального закладу доступний за посиланням: https://schedule.sumdu.edu.ua/index/json. Наразі, він надає лише 5 методів доступних для взаємодії з ним :

1) “method=getHelp” – надає список доступних методів у вигляді;
2) “method =getTeacher” – надає список викладачів в форматі JSON у вигляді пари ідентифікатор викладача-його ПІБ;
3) “method=getAuditoriums” – надає список аудиторій в форматі JSON у вигляді ідентифікатор аудиторії - її номер

4) “method=getGroups” – надає список навчальних груп в форматі JSON у вигляді пар ідентифікатор групи – її назва
5) “method=getSchedules” – надає розклад у форматі масиву JSON за такими параметрами:
    - id_grp –ідентифікатор навчальної групи,
    - id_fio(не обов’язковий параметр) – ідентифікатор викладача,
    - id_aud(не обов’язковий параметр) - ідентифікатор аудиторії,
    - date_beg, date_end – початок та кінець дати запиту відповідно у форматі “dd.mm.YYYY”.

Особливо цінним для розроблюваної ІС, виявився останній метод, який і буде найчастіше викликатись через доступний API для отримання розкладу. Тому детальніше розглянемо його основні поля результату запиту у форматі JSON, які найбільш необхідні для роботи майбутньої системи(див табл 1.1).

Таблиця 1.1 JSON відповідь

Назва поля JSON	**Значення**
DATE_REG	Дата пари (dd.mm.YYYY)
NAME_PAIR	Номер пари(“1 пара”, “2 пара”, …)
TIME_PAIR	Час початку та час закінчення пари
NAME_FIO	ПІБ викладача
NAME_AUD	Аудиторі
NAME_GROUP	Назва групи
ABBR_DISC	Скорочена назва дисципліни

KOD_DISC	ІД дисципліни
NAME_STUD	Вид заняття(лекція, практика, іспит …)
REASON	Статус заняття (онлайн/офлайн)

1.2. Огляд аналогів інформаційних систем інших навчальних закладів

В якості першої буде розглянута ІС розкладу КПІ https://schedule.kpi.ua/. Дана система реалізована у вигляді сайту на головній сторінці якого відразу розміщений весь функціонал роботи з розкладом (рис. 1.1):

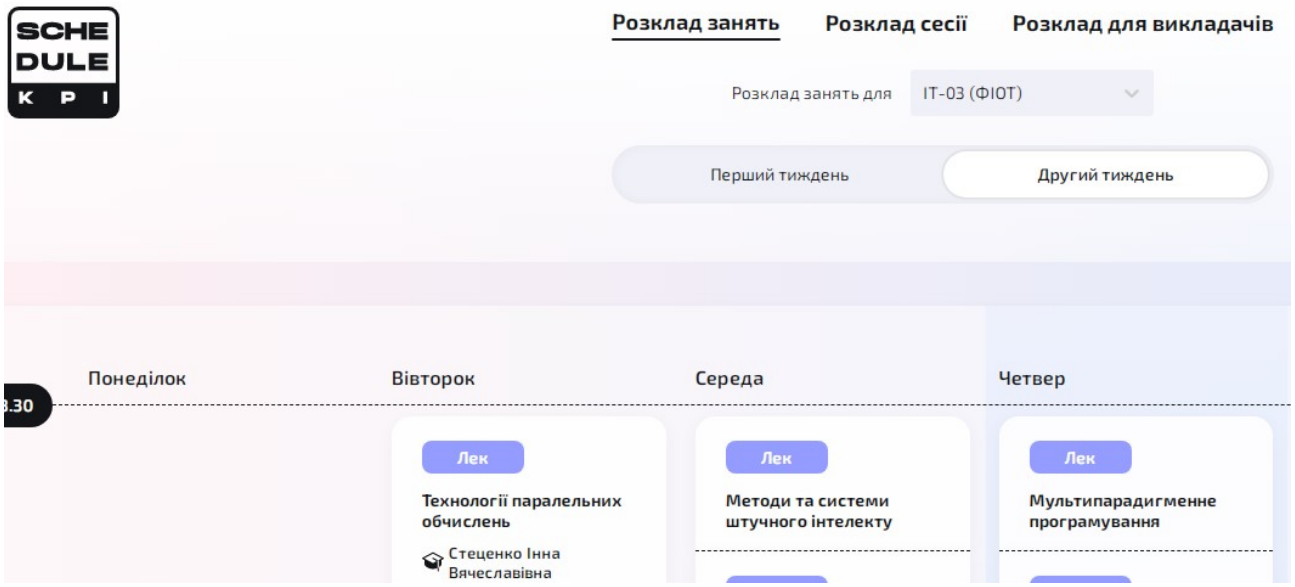

Рисунок 1.1 – Головна сторінка розкладу КПІ

Розглянувши функціональні можливості даної системи було виявлено такі недоліки:

1) Відсутній пошук занять для конкретної дати;
2) Відсутній пошук занять за конкретним часом поточного дня;
3) Відсутня інформація про посилання на заняття ;
4) Розклад надається лише на два тижні;

5) Єдиний доступний параметр пошуку для студентів є група;
6) Єдиний доступний параметр пошуку для викладачів є ПІБ викладача;
7) Не зручне відображення розкладу й відповідна навігація по ньому.

Другою буде розглянута ІС розкладу Львівської Політехніки https://student.lpnu.ua/students_schedule. Дана система реалізована у вигляді відповідної вкладці на сайті. Основною особливістю в порівнянні з попередньою ІС, є наявність отримання розкладу або на весь семестр або на певну половину семестру, при чому для відповідного типу заняття відображається потрібне посилання у розкладі(рис. 1.2):

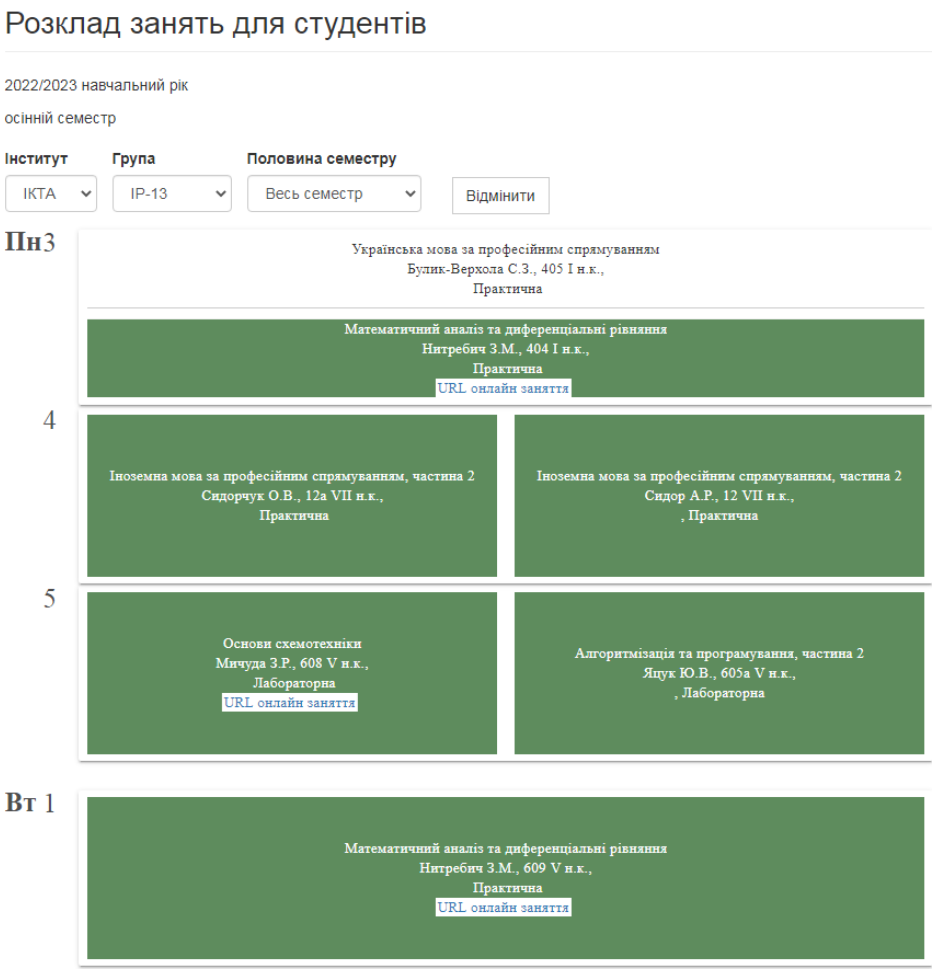

Рисунок 1.2 – Вкладка розкладу Львівської Політехніки

Але дана система також має свої недоліки:

1) Відсутній пошук занять для конкретної дати;
2) Відсутній пошук занять за конкретним часом поточного дня;
3) Розклад надається лише на половину або на весь семестр;
4) Не відображається час початку та кінця пари;
5) Не зручне відображення розкладу й відповідна навігація по ньому.

Наступний аналог ІС розкладу належить Житомирській політехніки https://rozklad.ztu.edu.ua/. Дана система реалізована у вигляді сайту, де на головній сторінці відразу пропонується вибір групи із списків факультетів(рис. 1.3):

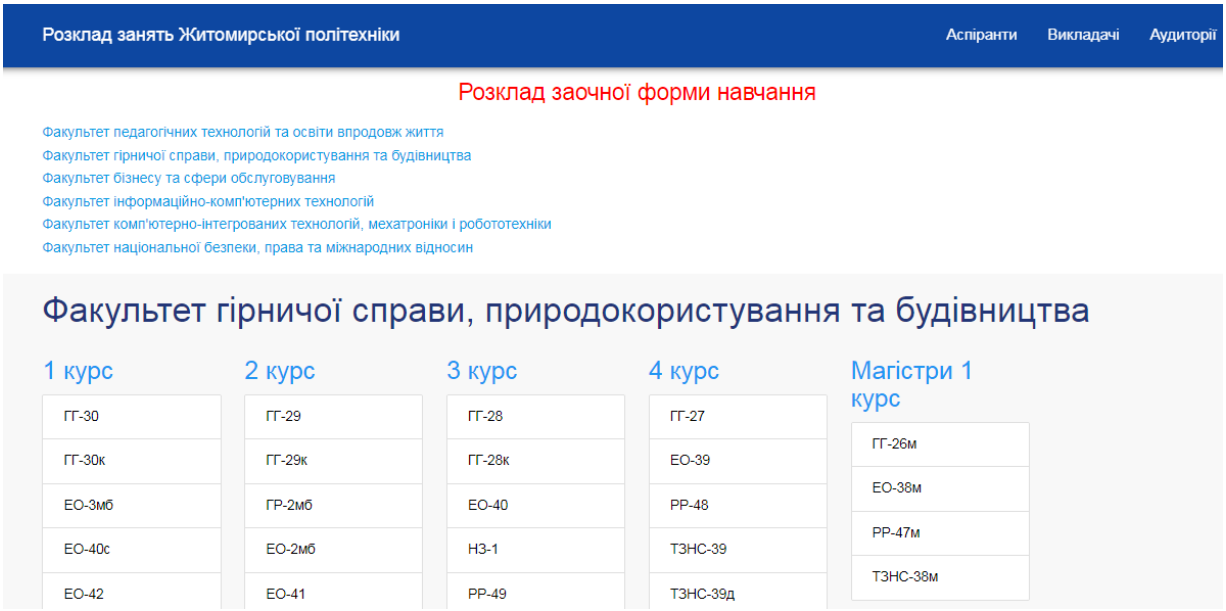

Рисунок 1.3 – Головна сторінка розкладу Житомирської політехніки

Дана система повторює функціонал розкладу КПІ з декількома відмінностями, а саме: вибір групи відбувається серед прямо наведених списків факультетів, а також наявний пошук розкладу за аудиторією(рис.1.4):

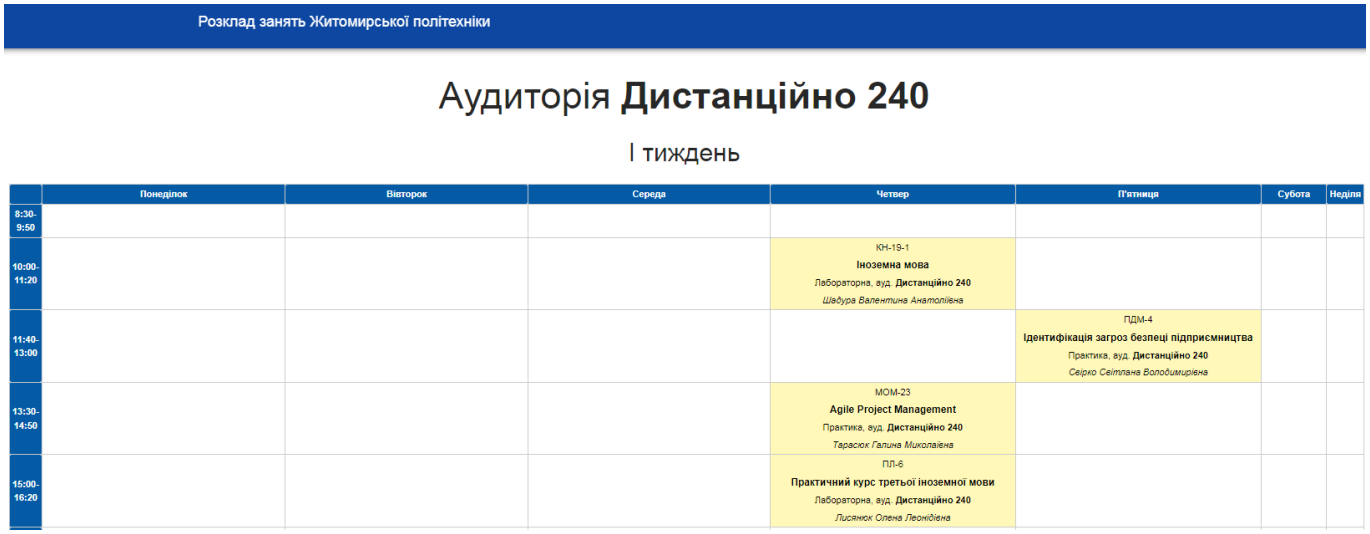

Рисунок 1.4 – Демонстрація розкладу Житомирської Політехніки для заданої аудиторії

В інших аспектах, дана система повторю недоліки розкладу КПІ :

1) Відсутній пошук занять для конкретної дати;
2) Відсутній пошук занять за конкретним часом поточного дня;
3) Відсутня інформація про посилання на заняття ;
4) Розклад надається лише на два тижні;
5) Не зручне відображення розкладу й відповідна навігація по ньому.

Додатково, до розглянутих у цьому пункті ІС, також справедливі недоліки існуючої ІС розкладу навчального закладу, що наведені у попередньому пункті й відповідно, дані ІС не можуть конкурувати з нею.

1.3. Постановка задачі

Метою даного проекту є розробка ІС розкладу навчального закладу на основі існуючої ІС розкладу навчального закладу для вирішення основних функціональних недоліків існуючої системи. Тому розроблювана ІС має відповідати наступним вимогам:

1) Наявність асоціації приватного Telegram чату користувача(студента) з навчальною групою існуючої ІС розкладу навчального закладу.

Отримана інформація має кожний раз використовуватися при надходженні запиту у Telegram в якості параметра пошуку;

2) Можливість пошуку студенту(асоційованому приватному Telegram чату користувача з навчальною групою) найближчої пари без зазначенням поточного часу та навчальної групи . В якості параметрів пошуку, веб-додаток має використовувати поточний час запиту у Telegram та навчальну групу з якою був асоційований даний вид Telegram чату з якого було отримано запит. При чому, якщо на час запиту буде відсутня пара або до її кінця залишилося менше 10 хвилин, то має повертається наступна найближча пара за часом;
3) Можливість пошуку студенту(асоційованому приватному Telegram чату користувача з навчальною групою) найближчої пари для явно вказаного у запиті часу пошуку . В якості параметрів пошуку, веб-додаток має використовувати зазначений користувачем час у запиті, навчальну групу з якою був асоційований даний вид Telegram чату з якого було отримано запит. При чому, якщо на час запиту буде відсутня пара або до її кінця залишилося менше 10 хвилин, то має повертається наступна найближча пара за вказаним часом;
4) Можливість проводиту наведні вище варіанти пошуків у груповому(Публічному) Telgram чаті. При чому, в якості навчальної групи використовуться група з якої був асоційований публічний Telegram чат користувача;
5) Можливість стати старостою асоційованому клієнту(студенту). При чому, має накладатися ліміт на існування не більше двох старост для однієї навчальної групи;
6) Можливість старості асоціювати Публічний Telegram чат з навчальною групою;

7) Можливість старостам назначати певному виду заняття для відповідної дисципліни необхідне посилання та/або відповідний ідентифікатор з паролем;
8) Можливість старостам назначати на певну пару введеного у запит дня у форматі ‘dd.mm.YYYY’ альтернативну пару з зазначенням коментаря про альтернативну пару та посилання на неї.

2. ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ ВЕБ-ДОДАТКУ

2.1. Вибір відкритого API Telegram

На даний момент, Telegram надає всього два види відкритих API для взаємодії з ним :

1) Telegram MTProto API – дає змогу створювати власні налаштовуванні клієнти Telegram;
2) Telegram Bot API - дає змогу легко створювати програми, що використовують повідомлення Telegram як інтерфейс.

Розглянемо кожний з названих API більш детальніше.

1. MTProto API .

   Даний API є на 100% відкритий для всіх розробників, які бажають створювати Telegram-додатки. Він побудований на основі протоколу шифрування MTProto за допомогою двійкових даних, серіалізованих певним чином та передаваймими за допомогою UDP, TCP або навіть HTTP в якості протоколу транспортного рівня. Відповідно, клієнти, які бажають використовувати даний API, мають реалізувати наведені вище деталі.

   Перевагами даного API є:

   1) Можливість авторизації як бот, чи користувач;
   2) Можливість завантажувати та вивантажувати будь-які файли, розміром до 2000 MiB кожен (~2 ГБ);
   3) Має менше накладних витрат завдяки прямому підключенню до Telegram;
   4) Можливий запуск кілька сесій одночасно (як для користувача, так і для бота);

5) Має набагато детальніші типи та більше методів ніж Bot API;
6) Отримувати додаткові оновлення, наприклад, про зміну імені користувача;
7) Має більше змістовних помилок на випадок, якщо щось пішло не так;
8) Швидше отримує оновлення версій API, а отже, і нові функції. Тоді як для оновлення Bot API потрібно, попередньо чекати оновлення MTproto API.

Відповідні недоліки:

1) Необхідність реалізації протоколу MTproto;
2) Складність реалізації протоколу MTproto;
3) Необхідність авторизації;
4) Необхідність реєстрації свого додатку в Telegram;
5) Необхідність відповідати Умовам використання API[4]

2. Bot API.

Даний API є спрощеною версією Telegram API. Він не потребує реалізації протоколу MTproto, за це відповідає сервер-посередник Telegram, який виконує все шифрування та зв'язок з Telegram API за нас. Для використання Bot API, необхідно лише організувати взаємодію з цим сервером-посередником через простий HTTPS-інтерфейс.

Переваги Bot API:

1) Не потрібно реалізувати MTproto протокол;
2) Легкість використання через HTTPS-інтерфейс.

3) Для авторизації потрібен лише ключ-токен бота, який надється під час створення акаунта бота у Telegram[5]
4) Наявність безліч бібліотек для різних мов програмування, що реалізують більшість методів цього API

Недоліки:

1) Типи Bot API часто пропускають деяку корисну інформацію про сутності Telegram, а деякі методи також обмежені;
2) Bot API повідомляє менш докладні повідомлення про помилки;
3) Bot API дозволяє використовувати тільки учетні записи ботів;
4) Проміжний сервер Bot API завершить будь-яку іншу сесію, якщо використовувати одного й того самого бота в паралельному з'єднанні;
5) Більші обмеження на розмір відправляймих та завантажуваних файлів.

Беручи до уваги те, що обираймий API необхідний лише для створення інтерфейсу взаємодії між ІС розкладу навчального закладу та студентами через Telegram, а не для створення окремого Telegram додатку, то критерієм вибору будуть швидкість реалізації та простота використання.

Отже, для розроблюваного веб-додатку буде обраний Telegram Bot API, оскільки його швидше застосувати та реалізувати через відсутності необхідності реалізації протоколу MTproto та наявності безліч бібліотек, які дозволять значно скоротити час на розробку цієї частини.

Наведена вище інформація про відкриті API Telegram цього розділу була створена на основі матеріалів [6, 7]

2.2. Вибір мови програмування

Широкий асортимент мов програмування пропонує багато гарних рішень для написання веб-додатків. Тому є можливість вибору за власним вподобанням, такими мовами стали : C++, C#, Java.

Розглянемо кожну з них:

1. C++

   Одна з найпотужніших мов програмування, C++ — це об'єктно-орієнтована мова програмування, яку можна використовувати для створення програм для операційної системи Windows. Також її можна використовувати для створення програм, які працюватимуть на будь-якій платформі, від настільного комп'ютера до сервера до хмари.

   Переваги:

   1) Найбільш поширена мова для веб-розробки;
   2) Його можна використовувати для веб-, мобільних, настільних і серверних розробок;
   3) Швидкість роботи програм на C++ практично не поступається програмам на C, хоча програмісти отримали в свої руки нові можливості і нові засоби.
   4) Наявність безкоштовних IDE

   Недоліки:

   1) Мова C++ є складною для вивчення і для компіляції;
   2) Жорстка типізація додатково ускладнює її вивчення;

3) Як правило, час розробки значно перевищує розробку на інших менш типізованих мовах.

Вищезазначена інформація про C++ була створена за допомогою матеріалів[8, 9].

2. C#

C# це об'єктно-орієнтована мова програмування з безпечною системою типізації. Синтаксис C# близький до C++ та Java. C# неймовірно гнучка, завдяки чому з її допомогою можна розробляти ігри, мобільні і веб-додатки, додатки для Windows, Android і iOS, працювати з файловими системами, забезпечувати безпеку та ін.

Переваги:

1) Високорівнева мова, яка спрощує синтаксис і робить його більш зрозумілим для людини;
2) C # має безпечну типізацію;
3) Не потрібно приділяти багато уваги таким проблемам, як втрата пам'яті, що є тривожною проблемою для C++;
4) Наявна гарно структурована та багата документація, численне ком'юніті та підтримка від IT-гіганта Microsoft.
5) Підтримка кросплатформеності;
6) Наявність безкоштовних IDE;

Недоліки:

1) Продуктивність нижча, ніж у складніших мов програмування(C++);

2) Неможливо модифікувати успадковану реалізацію під час виконання;
3) Велика розмірність програм за рахунок використання бібліотек загального призначення. Якщо для якоїсь задачі розробити вузькоспеціалізований програмний код, то цей код займатиме менше пам'яті, ніж код загального призначення;

Вищезазначена інформація про C# була створена за допомогою матеріалів[10–13].

3. Java

Найпопулярніша в світі мова програмування використовується для створення всього, від настільних додатків до веб-додатків. Java також неймовірно проста у використанні і може використовуватися для створення будь-чого, від простих веб-сторінок до складних програм.За допомогою Java можна створювати веб-сторінки, інтернет-магазини або навіть веб-програми, і ви зможете використовувати його на будь-якій платформі, включаючи мобільну.

Переваги:

1) Можливість використання для багатьох різних додатків
2) Підтримка кросплатформеності;
3) Є безпечною мовою програмування, оскільки не використовує явних вказівників.
4) Є надійною мовою програмування, оскільки використовує ефективне керування пам'яттю.

Недоліки:

1) Потребує інтерпретації під час виконання, що дозволяє їй працювати на кожній операційній системі, але це також робить її повільнішою за такі мови, як C та C++;
2) Мова програмування Java є дещо дороговартісною через високі вимоги до обробки та пам'яті. Для запуску програми на Java потрібне краще обладнання;
3) Відсутні безкоштовні версії IDE для можливості написання веб-додатків.

Вищезазначена інформація про Java була створена за допомогою матеріалів[8, 14].

Отже, C# є найбільш підходящою мовою для написання даної роботи, оскільки вона є менш складною в порівнянні з C++, що значно зменшить час розробки. При цьому, різниця вибору між Java та C# є не значною через дуже подібний синтаксис та безпечність розробки, але відсутність безкоштовних IDE для написання веб-додатків з боку Java є критичною для даного проекту. Також, потрібно зазначити, що в якості вибору платформи для розробки на C# буде обрано .Core, яка забезпечує кросплатформеність на відмінну від .NET.

2.3. Вибір фреймворків

В даному розділі буде наведено обрані фреймворкі без порівняння, оскільки C# - мова орієнтована на використання фреймворків й в більшості випадків для рішення тих чи інших задач під різні версії платформи пропонується десятки варіантів вибору. Причому, в більшості випадків основними критеріями вибору є наявність документації, частота

оновлення, популярність та підтримка той чи іншої версії цільової платформи. Тому, не має сенсу порівнювати кожний з них:

1. Microsoft.Extensions.Logging.Console та Microsoft.Extensions.Configuration.Json – необхідні для реалізації процесу логування;
2. System.Configuration.ConfigurationManager – необхідний для роботи з фалом конфігурації;
3. Newtonsoft.Json – для роботи з даними у форматі JSON;
4. Npgsql - для роботи з СУБД Postgres SQL;
5. Telegram.Bot – реалізує більшість методів, що надає Telegram Bot API.

Також, необхідно зазначити, що обрані фреймворки підбиралися під 5.0 версію цільової платформи .Core .

2.4. Вибір мови СУБД

В результаті потреб бізнесу було створена безліч різних реляційних та не реляційних СУБД. Для більшості популярних з них існують фреймворки для роботи з ними у C# та відповідна документація. Відповідно, вибір СУБД не є критичним для даної розроблюваної ІС, тому відбуватиметься за персональними вподобаннями.

Таким чином, для розроблюваної ІС розкладу навчального закладу буде обрана СУБД Postgres SQL за наступні переваги над іншими в цілому:

1) Є не просто реляційною СУБД, а об’єктно-реляційною СУБД.
2) Дає можливість створювати, зберігати та витягувати складні структури даних.
3) Підтримує значну кількість різних типів даних, які в такому обсязі

не підтримуються більшістю інших СУБД.

4) Підтримує масиви для більшості типів даних, а також багатовимерні масиви.
5) Підтримує JSON, що дозволяє перейти к зберіганню schema-less даних в SQL базі даних.
6) Підтримує можливість створення нових типів даних, таких як составний, перерахований, діапазон та базовий. Дана функціональність не підтримується в СУБД MySQL, MariaDB та Firebird, оскількі вони не є об'єктно-реляційними.
7) Гарною перевагою є наявність псевдо типа serial, який об'єднує найкращі риси auto_increment із MySQL та sequence з Oracle.
8) Можливість писати функції на чистому SQL. Наприклад, функція, що складається з одного update та returning, повертає ідентифікатор щойно доданого значення. Дана можливість дозволяє позбутися від явного оголошення змінних, в які записуються дані і які потім повертаються в return.
9) Є можливість в CTE (WITH) використання в якості запроса не тільки SELECT, але й INSERT, UPDATE, DELETE RETURNING, що не доступно в Oracle.
10) Синтаксис LIMIT з OFFSET є більш зручнішим, аніж відповідний синтаксис з використанням ROWNUM, сортуванням й подзапросами в Oracle.
11) Відповідний синтаксис INSERT ALL є більш зручнішим, аніж в Oracle.
12) Встановлення СУБД PostgreSql є більш простішим та швидким в порівнянні з встановленням MySQL чи Oracle.

Вищезазначена інформація про переваги Postgres SQL була створена на основі матеріалів[15–18].

1. ПРОГРАМНА РЕАЛІЗАЦІЯ

1.1. Побудова діаграми потоків даних

Розробляється інформаційна система (ІС) рокладу навчального закладу призначена для функціонального розширення існуючої ІС розкладу навчального закладу та інтеграція її у Telegram, при цьому, використовуючи в якості інтерфейсу взаємодії з нею функціонал Telegram Bot API(Відповідний Приватний/Публічний Telegram чат). Основні користувачі системи – студенти(Приватний/Публічний Telegram чат асоційований з навчальною групою). За допомогою цієї системи вони можуть отримувати розклад для певної групи та часу з відповідними посиланнями для підключення отримуючи відповідне повідомлення у Приватний/Публічни Telegram чат звідки надійшов запит. Також користувачами ІС можуть бути старости, які уповноважені створювати альтернативні пари для певної дати, назначати посилання для певної дисципліни та виду пари. Ці процеси можна відобразити на DF-діаграмах (див. Рис 3.1, 3.2 і 3.3).

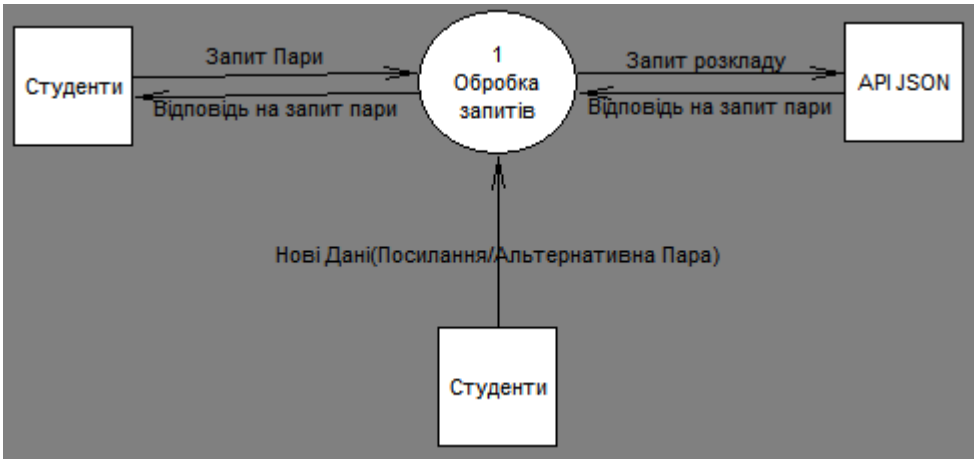

Рисунок 3.1.– DFD 0-го рівня для ІС

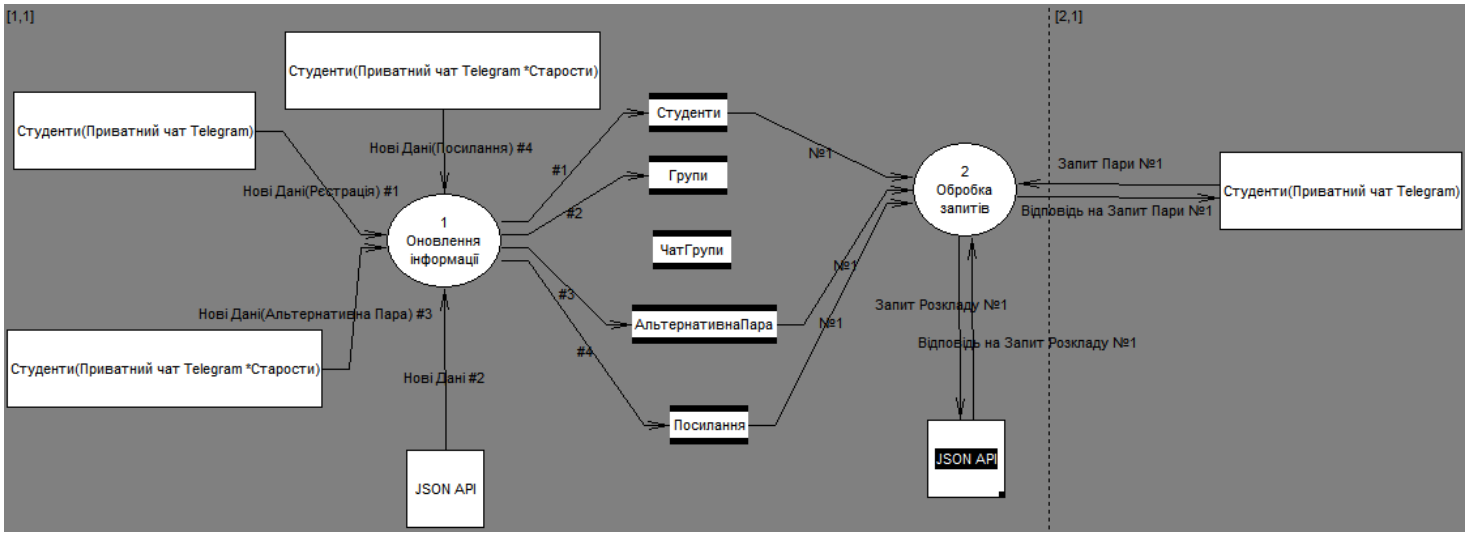

Рисунок 3.2 – DFD 1-го рівня процессу «Обробка запитів» у Приватних чатах Telegram

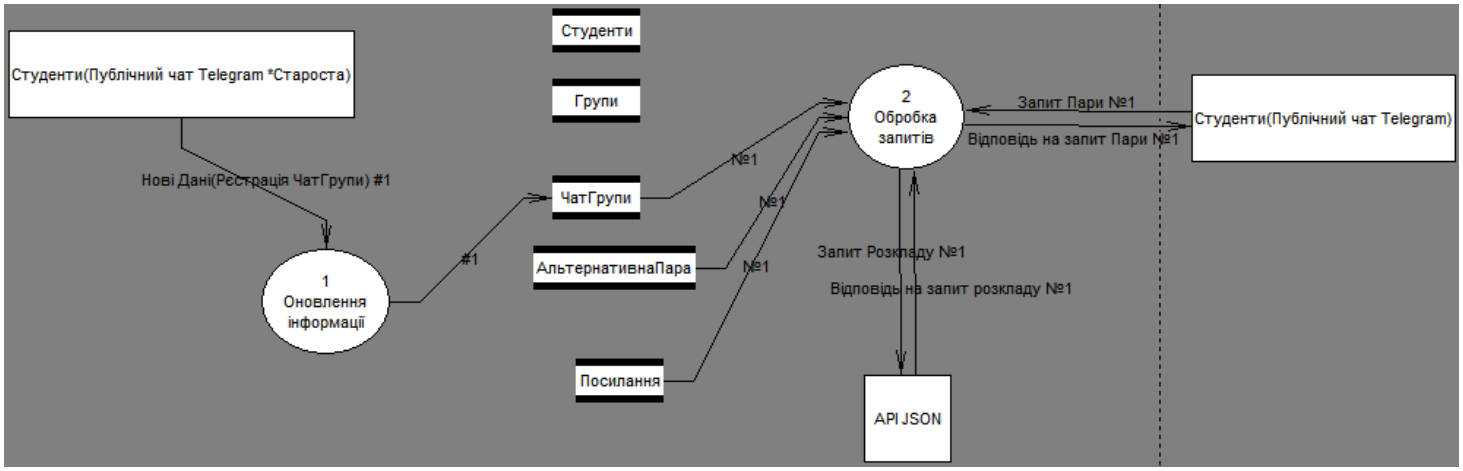

Рисунок 3.3 – DFD 1-го рівня процессу «Обробка запитів» у Публічних чатах Telegram

Для обробки процесів концептуальна модель на рис. 3..2 та 31.3 має наступні сховища даних: «Студенти», «Групи», «ЧатГрупи», «АльтернативнаПара» і «Посилання». Всі запити від користувачів діляться на процес «Оновлення інформації», який вносить зміни в сховища даних і процес «Обробки запиту», який обробляє запити від студентів(Приватний/Публічний Telegram чат), знаходить необхідну інформацію в сховищах даних і формує відповідь на запит.

1.2. Проектування бази даних

До розроблюваної ІС встановимо наступні бізнес-правила – Одна група може мати декілька старост(2), при цьому точна їх кількість має контролюватися на рівні самого веб-додатку; Один студент може бути одночасно старостою не більше ніж в одній групі.

Оскільки метою проекту є розробити ІС, яка б доповнювала існуючу ІС, то для уникнення дублювань, розроблювана ІС має зберігати мінімальну кількість полів для сутностей необхідних лише для можливості взаємодії з існуючою ІС. Відповідно будуть зберігатися такі дані :

1) Список студентів, старост та їх груп;
2) Посилання, паролі та ідентифікатори необхідні для підключення до відповідного виду пар(лекція, практичне зняття, тощо);
3) Список Telegram чатів/каналів кожної групи;
4) Заміни або додаткові пари у відповідний день та час.

Для усунення потенційної суперечливості і надмірності даних у відносинах, виявлених на етапі побудови концептуальної моделі («Студенти», «Групи», «ЧатГрупи», «АльтернативнаПара» і «Посилання») приведемо їх до третьої нормальної форми.

Зобразимо отримані відносини і їх зв'язки на ER-діаграмі (див. рис. 3.4).

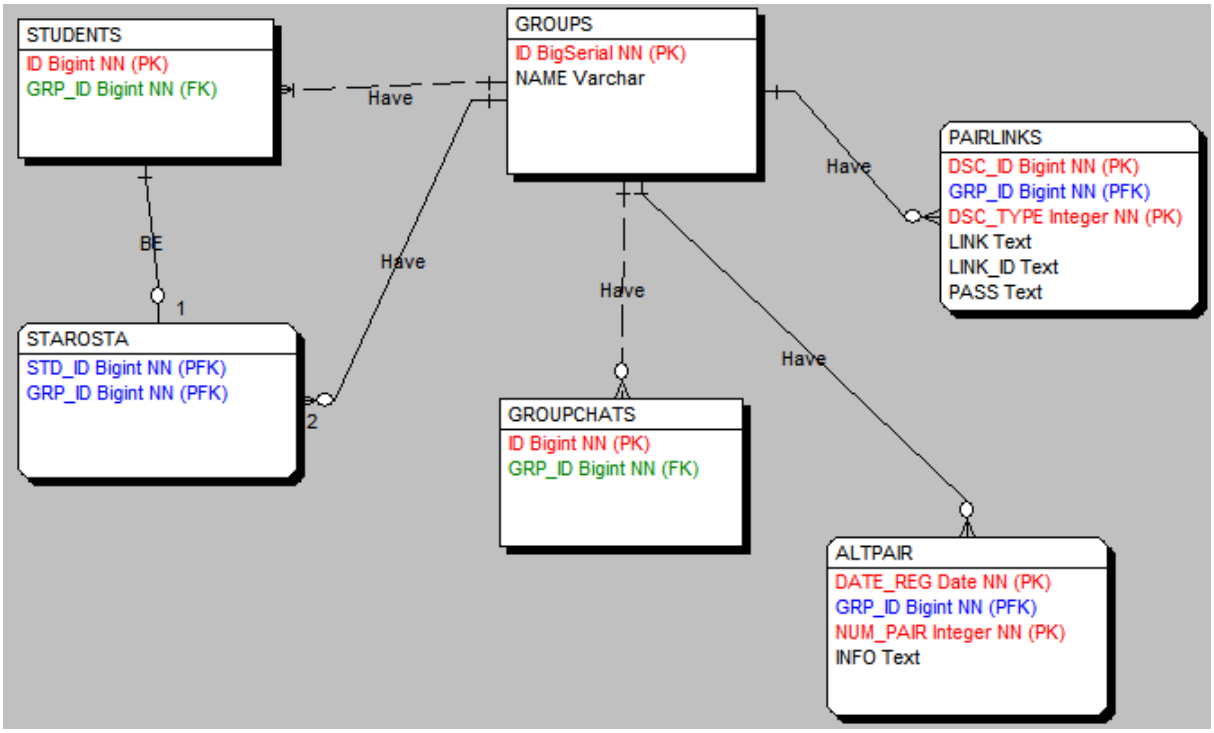

Рисунок 3.4. – ERD для IC розкладу навчального закладу

Для приведення відносин до 3НФ, а також для усунення неоднозначності зв'язку багато-до-багатьох розіб'ємо сутність «Студенти» на сутності «Студенти»(STUDENTS) і «Староста»(STAROSTA).

Відношенню «Студент» відповідає повна НФЗ: ID → GRP_ID.
Відношенню «ЧатГрупи» відповідає повна НФЗ: ID → GRP_ID.
Відношенню «АльтернативнаПара»(ALTPAIR) відповідає повна НФЗ: (DATE_REG, GRP_ID, NUM_PAIR) → INFO.
Відношенню «Посилання»(PAIRLINK) відповідає повна НФЗ: (DSC_ID, GRP_ID, DSC_TYPE) → LINK, LINK_ID, PASS.

Відношення «Староста» відповідає повній НФЗ, оскільки воно не містить не ключових атрибутів, які би повністю залежали лише від частини потенційного ключа, та не містить не ключових атрибутів, що

знаходилися б в транзитивній функціональній залежності від поточного потенційного ключа.

Під час аналізу вимог до ІС були встановленні наступні бізнес-правила. “Одна група може мати декілька старост”. Вказане гарантується відношенням один до багатьох між сутностями «Групи» та «Староста», що виражається наявністю альтернативного ключа GRP_ID у сутності «Староста». Також гарантується виконання правила “Один студент може бути одночасно старостою не більше ніж в одній групі”, що реалізується через відношення один до одного між сутностями «Студенти» та «Староста» і завдяки накладеному обмеженню унікальності на альтернативний ключ STD_ID в сутності «Староста».

1.3. Логічна реалізація бази даних

Проаналізувавши сутність, використовувані в моделі ІС, перейдемо до реалізації структури БД. Для цього представимо імена необхідних таблиць, атрибутів, типів, їх призначення та обмеження (див. табл. 3. 1).

Таблиця 3. 1 Структура БД

Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
GROUPCHATS	ID	Telegram Chat Id публічної групи/каналу/чату	BIGINT	PK	Не пустий
	GRP_ID	ID групи, що відповідає ID групи у існуючій навчальній ІС	BIGINT	FK	Не пустий
ALTPAIR	DATE_REG	Дата пари	DATE	PK	Не пустий

	GRP_ID	ID групи, що відповідає ID групи у існуючій навчальній ІС	BIGINT	PFK	Не пустий
	NUM_PAIR	Порядковий номер пари	INTEGER	PK	Не пустий
	INFO	Користувацька інформація стосовно пари	TEXT	-	-
GROUPS	ID	ID групи, що відповідає ID групи у існуючій навчальній ІС	BIGSERIAL	PK	Не пустий
	NAME	Назва навчальної групи	VARCHAR	-	-
STUDENTS	ID	Telegram User ID	BIGINT	PK	Не пустий
	GRP_ID	ID групи, що відповідає ID групи у існуючій навчальній ІС	BIGINT	FK	Не пустий
STAROSTA	STD_ID	ІД студента(Telegram User ID)	BIGINT	PFK	Не пустий, унікальний
	GRP_ID	ID групи, що відповідає ID групи у існуючій навчальній ІС	BIGINT	PFK	Не пустий
PAIRLINKS	DSC_ID	ID дисципліни, що відповідає коду дисципліни у	BIGINT	PK	Не пустий

		існуючій навчальній ІС			
	GRP_ID	ID групи, що відповідає ID групи у існуючій навчальній ІС	BIGINT	PFK	Не пустий
	DSC_TYPE	Тип пари(1 = 1пара, 2…)	INTEGER	PK	Не пустий
	LINK	Посилання на пару	TEXT	-	-
	LINK_ID	Ідентифікатор для підключення на пару	TEXT	-	-
	PASS	Пароль для підключення на пару	TEXT	-	-

1.4. Розробка веб-додатку під API існуючої системи

Архітектуру розроблюваного веб-додатку можна поділити на декілька рівнів:

1) Робота веб-додатку з БД (реалізація шаблону проектування Data Access Object);
2) Бізнес-логіка(обробка сутностей веб-додатку, що реалізують сутності БД та існуючої ІС навчального закладу та взаємодія з нею);
3) Інтерфейс взаємодії з веб-додатком (Реалізація Telegram Bot API, відповідні Controllers – оброблюють запити від користувачів).

Логіка взаємодії зазначених рівнів між собою є такою: рівень інтерфейсу може взаємодіяти з бізнес логікою, яка в свою чергу може взаємодіяти з рівнем роботи з БД. Така логіка декомпозиції архітектури на рівні, кожний з яких вирішує свою окрему задачу дозволяє максимально зрозуміло орієнтуватися в коді й відповідно легко проводити масштабування даного додатку. Наприклад, якщо повстає питання міграції на іншу СУБД, то достатньо лише розширити або переписати рівень, що відповідає за роботу з БД, в даному випадку клас DBWorker. Зобразимо дані зв’язки між рівнями архітектури (див рис. 3.5.):

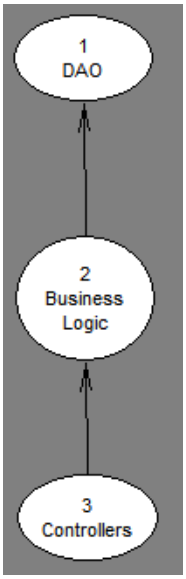

Рисунок 3.5. – Рівні архітектури веб-додатку

Розподіл класів веб-додатку за рівнями наведено у таблиці 3.2:

Таблиця 3.2

Рівень	**Класи веб-додатку**
Робота з БД	**DBWorker** – реалізує взаємодію з БД Postgres. Основні методи класу: **connect**() – підключення до БД Postgres; **readFromTable**() – зчитування даних з БД select-запросом; **insertToTable**() – реалізація вставки даних до БД insert командою; **updateTable**() – реалізація команди update до таблиць БД
Бізнес-логіка	Класи, що реалізують сутності БД: **User, Group, PairLink, AltPair**. Класи, для обробки класів сутностей БД: **PairDay** – для зберігання та обробки пар(**Pair**) за певний день у розкладі; **PairDays** – для зберігання та обробки днів(**PairDay**) розкладу; **Scheduler** - отримує та оброблює розклад, реалізує взаємодію та отримання інформації за допомогою API JSON методів наданих існуючою ІС навчального закладу; **JsonPair** – для десерелізації отриманих результатів розкладу через API JSON методів існуючої ІС розкладу навчального закладу; **Time, TimeInterval –** реалізація класів, що полегшуть роботу з даними, що являють собою час чи часовий інтервал, який зберігають сутності

	БД.
Controllers	**BotController** – реалізації інтерфейсу взаємодії з користувачем та обробка запитів через Telegram Bot API; **UserController** – обробка станів користувача на кожному етапі взаємодії з інтерфейсом веб-додатку через Telegram Bot API

Серед представлених вище класів буде наведено відповідний код лише основних класів системи в розділі Додаток А. Більш детально про допоміжні класи та програмну реалізацію в цілому можна побачити на моїй сторінці у GitHub.[19]

1.5. Алгоритм роботи

Гість

Користувач Telegram, що не асоціював себе з певною навчальною групою. Він не має доступу до функціоналу системи й може виконати лише одну команду ‘/reg’ - почати процес асоціації себе з певною навчальною групою.

Після введення даної команди система пропонує ввести назву навчальної групи або початкові літери її скороченої назви (рис. 3.6):

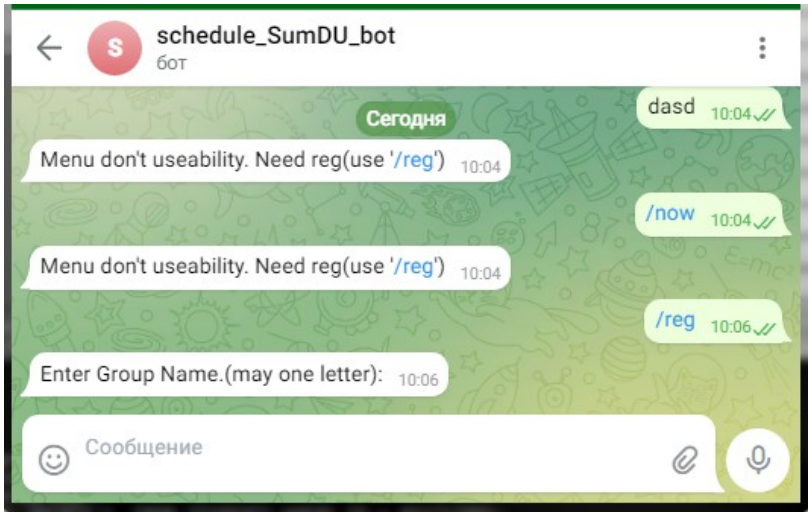

Рисунок 3.6 – Результат вводу команди ‘/reg’

Після вводу назви, користувачу виводиться список знайдених груп у системі(рис. 3.7):

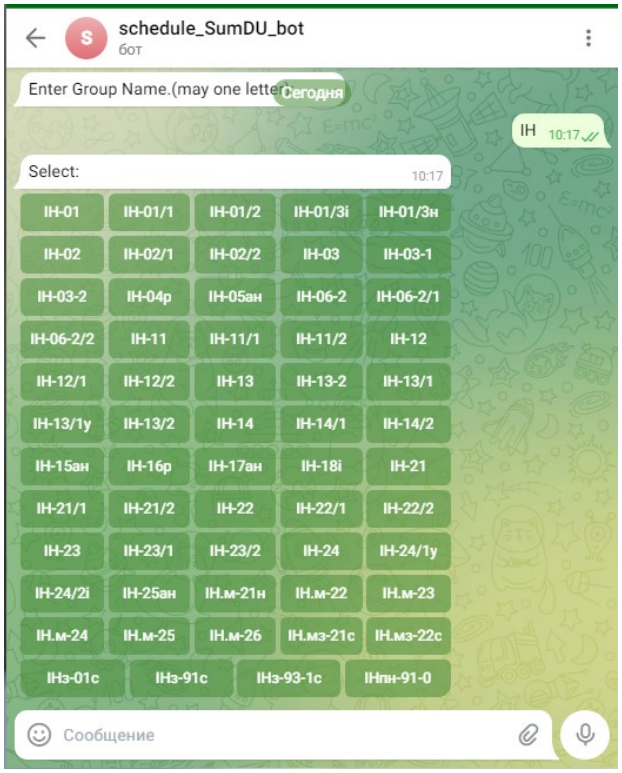

Рисунок 3.7 – Результат вводу назви групи

При натисненні відповідної кнопки, повідомлення зі списком знайдених груп видаляється та користувачу приходить нове повідомлення про обрану групу. На цьому процес умовної реєстрації завершився і тепер користувач є асоційованим до обраної групи і вважається студентом. Також, якщо в обраній групі менше 2-х старост, то користувачу пропонується стати старостою (рис. 3.8):

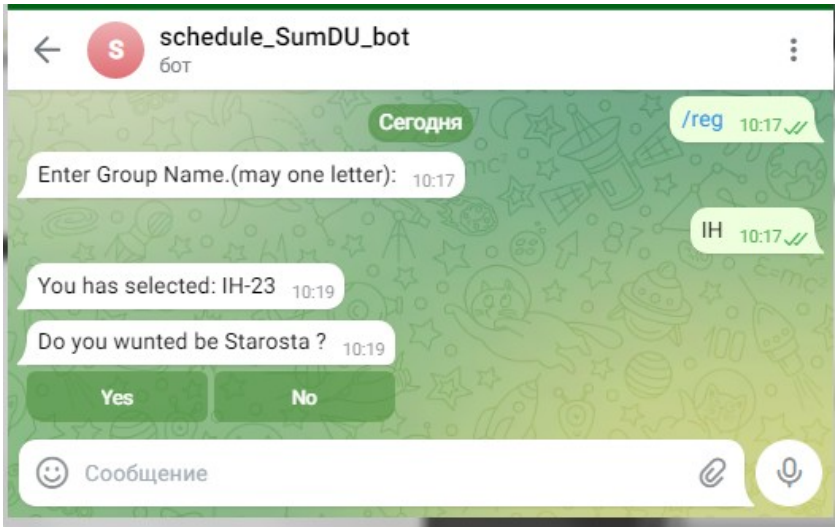

Рисунок 3.8– завершення умовної реєстрації користувача та вивід можливості стати старостою.

Далі студент може погодитись стати старостою нажавши кнопку ‘Yes’, і отримавши відповідне повідомлення про те, що йому доступний функціонал старости (рис. 3.9):

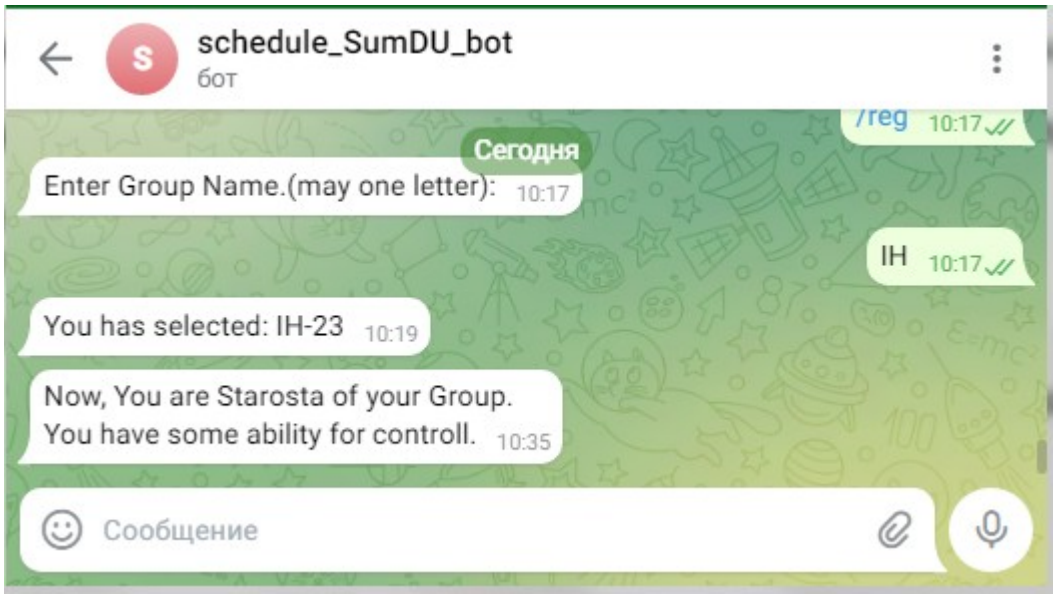

Рисунок 3.9 – результат натискання кнопки ‘Yes’

Якщо в момент натиснення кнопки в групі вже є 2 – старости , то студент отримає відповідне повідомлення про те, що він вже не може бути старостою(рис. 3.10):

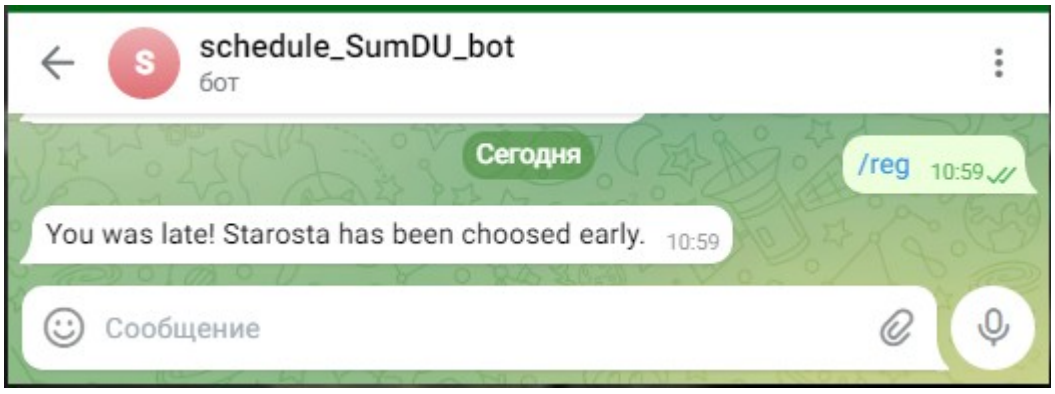

Рисунок 3.10 – результат натискання кнопки ‘Yes’ при наявності уже 2-х старост

Також студент може відмовитися натиснувши кнопку ‘No’, тоді він отримає відповідне повідомлення(рис. 3.11):

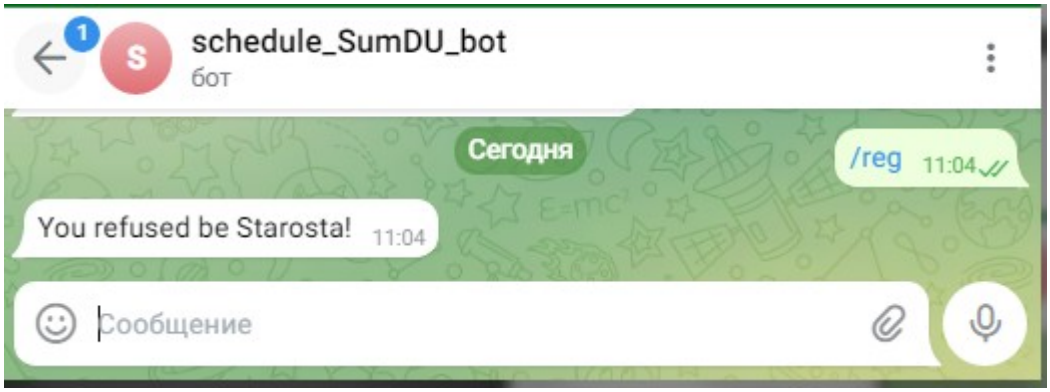

Рисунок 3.11 - результат натиснення кнопки ‘No’

Студент

Користувач, що пройшов умовну реєстрацію. Йому доступна команда **‘/reg’** для можливості стати старостою, якщо їх в групі менше 2-х. Логіка цієї команди повторює попередній функціонал можливості стати старостою в кінці умовної реєстрації не асоційованого користувача до навчальної групи.

Також йому доступні дві варіації команди ‘/now’, яка в такому вигляді поверне найближчу пару за поточним часом поточного дня. Також є можливість виконати дану команд для пошуку найближчої пари поточного дня за заданим часом ‘/now hh:mm’. Скріншот демонстрації цих команд (рис.3.12) :

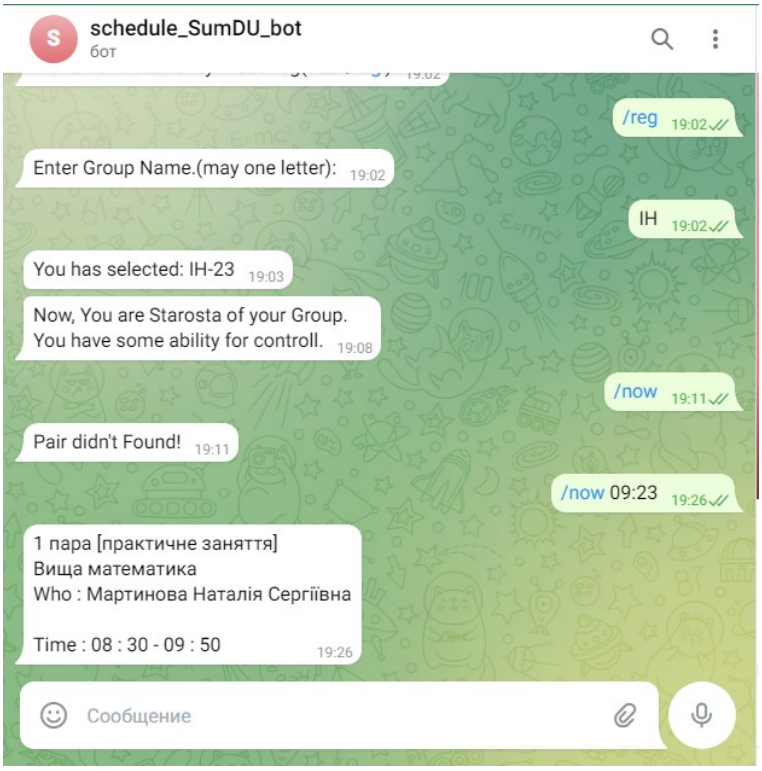

Рисунок 3.12 – результат виконання команди запиту пари

Як бачимо на рис. 3.12 - отриманий результат співпадає з результатом при таких же параметрах у існуючій ІС розкладу навчального закладу (рис. 3.13):

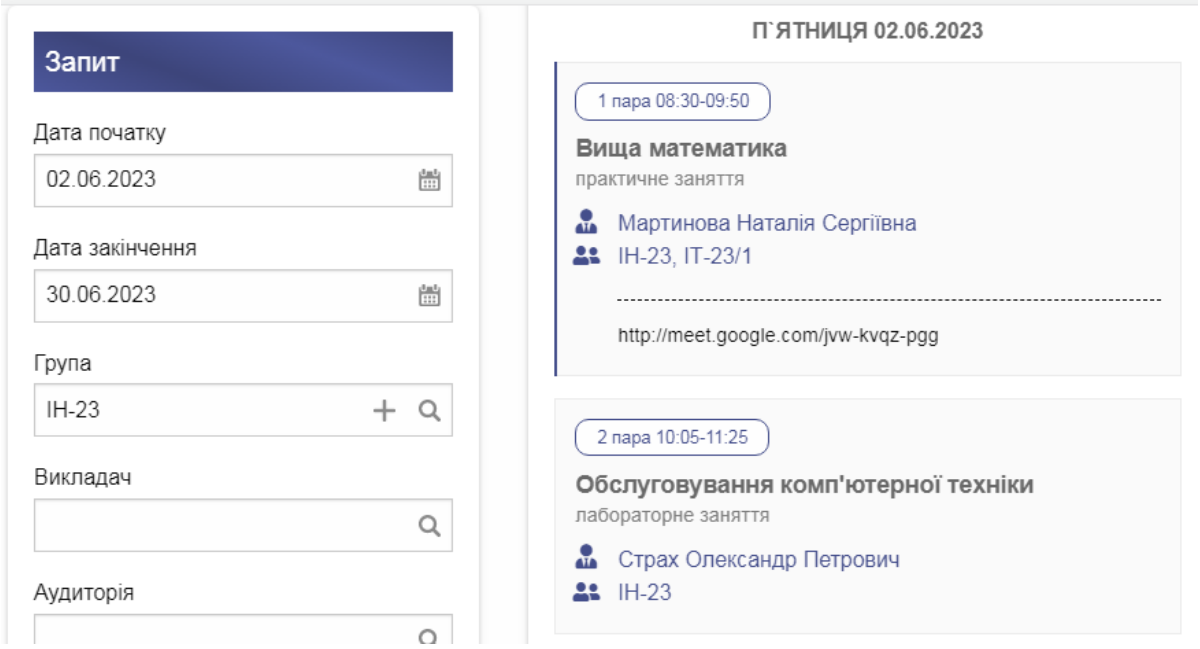

Рисунок 3.13 – результат запиту розкладу у існуючій ІС розкладу навчального закладу

Староста

Даному користувачу доступні усі команди користувача типу студент. Також йому додатково доступна команда ‘/link’ для призначення посилання для печерного виду заняття відповідної дисципліни. Скріншот демонстрації цієї команди наведено нижче(рис. 3.14):

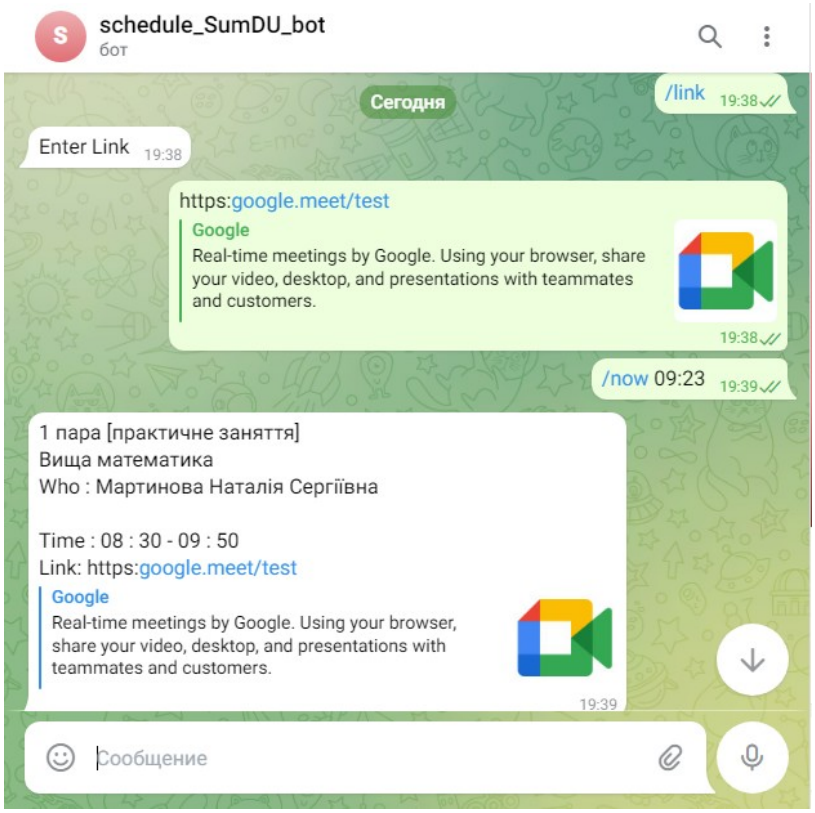

Рисунок 3.14 – Результат призначення дисципліні посилання та відповідна перевірка запитом

Також даному типу користувачу доступна команда створення альтернативної пари для заданого дня ‘/alt’ . Скріншот демонстрації даної команди наведено на рис. 3.15:

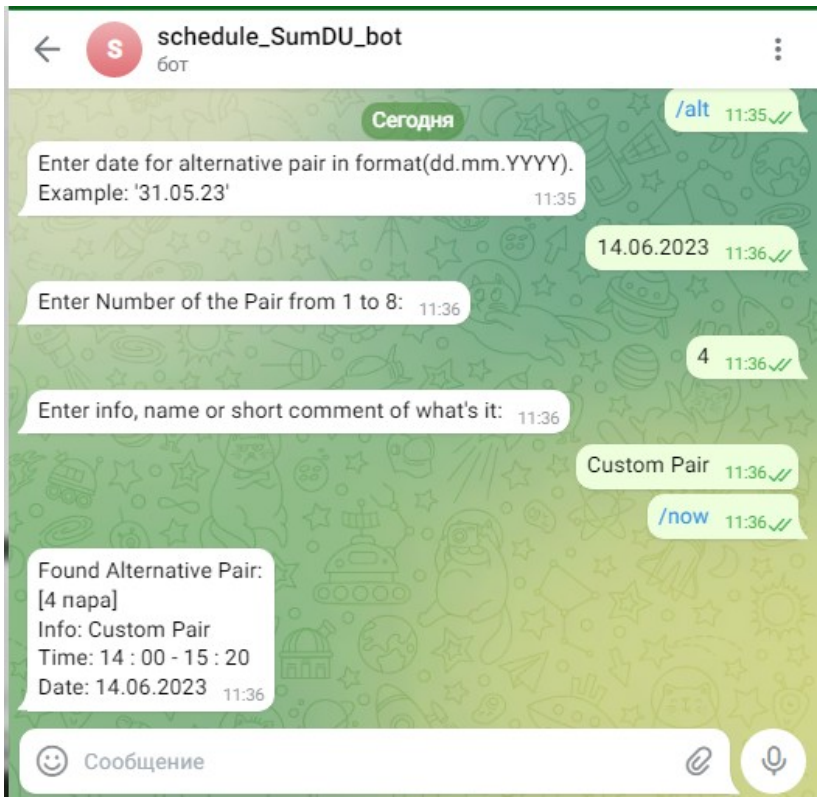

Рисунок 3.15 - скріншот створення та перевірка створеної альтернативної пари

Також старості на доступна умовна рєстрація публічного чату за допомогою команди ‘/reg’.

Усі наведені вище команди були показані при реєстрації й відповідній роботі у приватному чаті Telegram з ботом, якого реалізує створена ІС навчального розкладу . Для роботи цих команд у груповому(публічному) чаті Telegram, необхідно аби її зареєстрував староста за допомогою команди ‘/reg’, попередньо добавивши відповідного бота. Дана команда відразує реєструє на навчальну групу старости без необхідності вказувати її (рис. 3.16):

Рисунок 3.16 – Скріншот реєстрації публічного чату Telegram старостою

ВИСНОВКИ

В процесі виконання даної роботи, було проведено огляд існуючої ІС розкладу навчальоного закладу, недоліки її реалізацій в цілому та знайдено API JSON для взаємодії.

Також було обрано іструменти для розробки веб-додатку. Підібрано набір необхідних фремворків під платформу .Core 5-ї версії, які значно пришвидшили процес розробки. Спроєктована БД була впроваджена та активно використовується бізнес-логікою через клас, що реалізував шаблон DAO. В сукупності приведена БД ІС до 3НФ та продумана трьох рівнева архітектура веб-додатку, забезпечила гнучку основу для швидкого розширення як самого функціоналу ІС так і структуру реалізованої БД.

Отже було реалізовано ІС розкладу навчального закладу, яка функціонально розширила існуючу, реалізувавши можливості пошуку пари за часом; призначення посилань до занять та створення альтирнативної пари старостою. При чому, в якості ітерфейса було використано можливості Telgram бота через раелізований функціонал Telegram Bot API. Таким чином, розроблена система не просто доповнила функціонал існуючої, а й інтегрувала її у Telegram, що дало можливість альтернативного доступу до розкладу та значно оптимізує організацію навчального процесу та роботу з розкладом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Особистий кабінет СумДУ: URL: https://cabinet.sumdu.edu.ua/.
2. Schedule SSU - Apps on Google Play: URL: https://play.google.com/store/apps/details?id=com.SumDUschedule.SumDU&hl=en_US&pli=1.
3. Розклад СумДУ - червень 2023: URL: https://my-university.com.ua/universities/sumdu.
4. Telegram API Terms of Service: URL: https://core.telegram.org/api/terms.
5. Bots: An introduction for developers: URL: https://core.telegram.org/bots#how-do-i-create-a-bot.
6. Telegram APIs: URL: https://core.telegram.org/api#telegram-api.
7. MTProto vs. Bot API — Pyrogram Documentation: URL: https://docs.pyrogram.org/topics/mtproto-vs-botapi.
8. 5 найкращих мов програмування, які використовуються у веб-додатках - HashDork: URL: https://hashdork.com/uk/найкращі-мови-веб-додатків/.
9. Вивчення C++: Переваги та недоліки C++: URL: http://informatikpm11.blogspot.com/p/c.html.
10. 9 причин вивчити мову C# - FoxmindEd: URL: https://foxminded.ua/9-prichin-vivchiti-movu-c/.
11. C Sharp - Features, Advantages and Disadvantages: URL: https://urbannaturale.com/c-sharp-features-advantages-and-disadvantages/.
12. C#. Inheritance. Basic concepts. Advantages and disadvantages. General form. The simplest examples. Access modifier protected | BestProg: URL: https://www.bestprog.net/en/2020/02/24/c-inheritance-basic-concepts-advantages-and-disadvantages-general-form-the-simplest-examples-access-modifier-protected/.
13. Як вивчити мову програмування C# та стати .NET розробником | Блог: URL: https://edu.cbsystematics.com/ua/blog/learn-csharp-become-dnet.

14. Advantages and disadvantages of Java - Javatpoint: URL: https://www.javatpoint.com/advantages-and-disadvantages-of-java.
15. Що таке PostgreSQL? Вступ, переваги & Недоліки: URL: https://uk.css-code.org/8225113-what-is-postgresql-introduction-advantages-and-disadvantages#menu-5.
16. Порівняння систем управління базами даних (СУБД): MySQL, PostgreSQL та інші | АлтексСофт: URL: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/.
17. Sqlite vs mysql vs postgresql порівняння систем управління базами даних - вибір бази даних -: URL: https://jak.koshachek.com/articles/sqlite-vs-mysql-vs-postgresql-porivnjannja-sistem.html.
18. Firebird vs. MariaDB vs. PostgreSQL Comparison: URL: https://db-engines.com/en/system/Firebird%3BMariaDB%3BPostgreSQL.
19. Igorstar23/TelegramBotTest23: Information system of the schedule of the educational institution: URL: https://github.com/Igorstar23/TelegramBotTest23.

ДОДАТОК А
ЛІСТИНГ ПРОГРАМНОГО КОДУ

А1 User.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TelegramBotTest
{
    class User
    {
        public long Id { get; set; }
        public long GrpID { get; set; }

        public UserStatus Status { get; set; }

        public User()
        {
            GrpID = Group.nullGrpId;
        }
        public User(long Id, long grpId = Group.nullGrpId, UserStatus status
= UserStatus.NeedRegistration)
        {
            this.Id = Id;
            this.GrpID = grpId;
            this.Status = status;
        }
        public User(UserStatus status)
        {
            Status = status;
        }
        public User(long id, UserStatus status)
        {
            Id = id;
            Status = status;
        }

        public enum UserStatus
        {
            NeedRegistration = 0,
            StartRegistration = 1,
            InRegistrationProcess = 2,
            IsRegisteredGroup = 3,
            IsStartRegisterStarosta = 4,
```

```
            IsRegisteredStarosta = 5
        }
    }
}
```

A2 Group.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TelegramBotTest
{
    class Group
    {
        public long Id { get; set; }
        public string GrpName  { get; set; }

        public const int nullGrpId = -1;
        public const string nullGrpName = "NONE";

        public Group()
        {
            Id = nullGrpId;
            GrpName = nullGrpName;
        }
        public Group(long id, string grpName)
        {
            Id = id;
            GrpName = grpName;
        }
        public Group(string strValues)
        {
            string []res = strValues.Split(':');
            res[0] = res[0].Trim(' ');
            Id = long.Parse(res[0]);
            GrpName = string.IsNullOrEmpty(res[1]) ||
string.IsNullOrWhiteSpace(res[1]) ? "NONE" : (res[1].Trim(' '));
        }

        public override string ToString()
        {
            return $"GRP[{Id}:{GrpName}]";
        }
    }
}
```

A3 PairLink.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TelegramBotTest
{
    class PairLink
    {
        public long DSC_ID { get; set; }
        public long GRP_ID { get; set; }
        public DSC_TYPES DSC_TYPE { get; set; }

        public string Link { get; set; }
        public string LinkId { get; set; } // Zoom Id
        public string Pass { get; set; } // Zoom Pass

        public PairLink()
        {
            Link = "NONE";
            LinkId = "NONE";
            Pass = "NONE";
        }

        public PairLink(long dSC_ID, long gRP_ID, DSC_TYPES dSC_TYPE) :
this()
        {
            DSC_ID = dSC_ID;
            GRP_ID = gRP_ID;
            DSC_TYPE = dSC_TYPE;
        }
        public PairLink(long dSC_ID, long gRP_ID, DSC_TYPES dSC_TYPE, string
link)
            : this(dSC_ID, gRP_ID, dSC_TYPE)
        {
            Link = link;
        }
        public PairLink(long dSC_ID, long gRP_ID, DSC_TYPES dSC_TYPE,
            string link, string linkId, string pass)
            : this(dSC_ID, gRP_ID, dSC_TYPE, link)
        {
            LinkId = linkId;
            Pass = pass;
        }



        public enum DSC_TYPES { NONE = -1, LECTION = 0, LAB = 1, ATEST = 2,
EXAM = 3 }
```

```
        public static DSC_TYPES getDscType(string type)
        {
            if (string.IsNullOrEmpty(type) ||
string.IsNullOrWhiteSpace(type)) return DSC_TYPES.NONE;

            switch(type.ToLower())
            {
                case var text when text.Contains("лекція")||
text.Contains("lec"):
                    return DSC_TYPES.LECTION;
                break;

                case var text when text.Contains("лаб") ||
text.Contains("прак")||text.Contains("lab"):
                    return DSC_TYPES.LAB;
                break;

                case var text when text.Contains("атестація")||
text.Contains("atest"):
                    return DSC_TYPES.ATEST;
                break;

                case var text when text.Contains("іспи")||
text.Contains("exam"):
                    return DSC_TYPES.EXAM;
                break;

                default: return DSC_TYPES.NONE;
            }
        }

        public override string ToString()
        {
            StringBuilder stringBuilder = new StringBuilder();

            if (Link != "NONE") stringBuilder.Append($"Link: {Link}\n");
            if (LinkId != "NONE") stringBuilder.Append($"Connect ID:
{LinkId}\n");
            if (Pass != "NONE") stringBuilder.Append($"Password: {Pass}");

            return stringBuilder.ToString();
        }
    }

}
```

A4 AltPair.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TelegramBotTest
{
    class AltPair
    {
        public string DATE_REG { get; set; }
        public NUMBER_PAIR NUM_PAIR { get; set; }
        public long GRP_ID { get; set; }
        public string INFO { get; set; }


        public AltPair(string date)
        {
            if (!Pair.isNormalDate(date)) date = Pair.getNormalDate(date);
            DATE_REG = date;
            NUM_PAIR = NUMBER_PAIR.NONE;
        }
        public AltPair(string date, NUMBER_PAIR numPair):this(date)
        {
            NUM_PAIR = numPair;
        }
        public AltPair(string date, NUMBER_PAIR numPair, long
grpId):this(date, numPair)
        {
            GRP_ID = grpId;
        }
        public AltPair(string date, NUMBER_PAIR numPair, long grpId,
            string info) : this(date, numPair, grpId)
        {
            INFO = (string.IsNullOrEmpty(info) ||
string.IsNullOrWhiteSpace(info))? "NONE" : info;
        }
        /**
         * <summary>Zero-based number of pair</summary>
         */
        public enum NUMBER_PAIR
        {
            NONE = -1,
            FIRST,
            SECOND,
            THIRD,
            FOURTH,
            FIFTH,
            SIXTH,
            SEVENTH,
```

```
            EIGHTH
        }
        public static int NumberPairToIntBasedOne(NUMBER_PAIR num) { return
((int)num) + 1; }
        public static NUMBER_PAIR ToNumberPair(string num)
        {
            if (string.IsNullOrEmpty(num) || string.IsNullOrWhiteSpace(num))
return NUMBER_PAIR.NONE;

            switch(num.ToLower())
            {
                case "1 пара": return NUMBER_PAIR.FIRST;
                case "2 пара": return NUMBER_PAIR.SECOND;
                case "3 пара": return NUMBER_PAIR.THIRD;
                case "4 пара": return NUMBER_PAIR.FOURTH;
                case "5 пара": return NUMBER_PAIR.FIFTH;
                case "6 пара": return NUMBER_PAIR.SIXTH;
                case "7 пара": return NUMBER_PAIR.SEVENTH;
                case "8 пара": return NUMBER_PAIR.EIGHTH;
                default: return NUMBER_PAIR.NONE;
            }
        }
        /**
         * <summary>
         * get Zero-base enum
         * </summary>
         * <param name="num">must be from 0 to 8</param>
         * <returns>NUMBER_PAIR if <paramref name="num"/> is from 1 to 8;
otherwise NUMBER_PAIR.NONE</returns>
         */
        public static NUMBER_PAIR ToNumberPair(int num)
        {
            if (num < 1 || num > 8) return NUMBER_PAIR.NONE;
            return (NUMBER_PAIR)(num - 1);
        }
        public static string NumberPairToString(NUMBER_PAIR num)
        {
            if (num == NUMBER_PAIR.NONE) return NUMBER_PAIR.NONE.ToString();

            switch (num)
            {
                case NUMBER_PAIR.FIRST: return "1 пара";
                case NUMBER_PAIR.SECOND: return "2 пара";
                case NUMBER_PAIR.THIRD: return "3 пара";
                case NUMBER_PAIR.FOURTH: return "4 пара";
                case NUMBER_PAIR.FIFTH: return "5 пара";
                case NUMBER_PAIR.SIXTH: return "6 пара";
                case NUMBER_PAIR.SEVENTH: return "7 пара";
                case NUMBER_PAIR.EIGHTH: return "8 пара";
```

```
                default: return NUMBER_PAIR.NONE.ToString();
            }
        }
        public static TimeInterval NumberPairToTimeInterval(NUMBER_PAIR num)
        {
            if (num == NUMBER_PAIR.NONE) return null;
            switch(num)
            {
                case NUMBER_PAIR.FIRST: return new TimeInterval("08:30-
09:50");
                case NUMBER_PAIR.SECOND: return new TimeInterval("10:05-
11:25");
                case NUMBER_PAIR.THIRD: return new TimeInterval("11:40-
13:00");
                case NUMBER_PAIR.FOURTH: return new TimeInterval("14:00-
15:20");
                case NUMBER_PAIR.FIFTH: return new TimeInterval("15:35-
16:55");
                case NUMBER_PAIR.SIXTH: return new TimeInterval("17:10-
18:30");
                case NUMBER_PAIR.SEVENTH: return new TimeInterval("18:45-
20:00");
                case NUMBER_PAIR.EIGHTH: return new TimeInterval("20:15-
21:35");
                default: return null;
            }
        }
        public static List<TimeInterval>
NumberPairLToTimeIntervalL(List<NUMBER_PAIR> nums)
        {
            if (nums == null || nums.Count <= 0) return null;

            List<TimeInterval> intervals = new List<TimeInterval>();
            foreach(var num in nums)
            {
                intervals.Add(NumberPairToTimeInterval(num));
            }
            return intervals;
        }
        public static List<NUMBER_PAIR> getNumberPairAllList()
        {
            List<NUMBER_PAIR> list = new List<NUMBER_PAIR>();
            foreach (int i in Enum.GetValues(typeof(NUMBER_PAIR)))
            {
                NUMBER_PAIR num = AltPair.ToNumberPair(i);
                if (num != NUMBER_PAIR.NONE) list.Add(num);

            }
            return list;
```

```
        }
        public static List<TimeInterval> getAllTimeIntervalsPairs()
        {
            return NumberPairLToTimeIntervalL(getNumberPairAllList());
        }
    }
}
```

A5 Scheduler.cs

```
using System;
using System.IO;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using Telegram.Bot;
using Telegram.Bot.Args;
using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types.ReplyMarkups;
using Newtonsoft.Json;
using Microsoft.Extensions.Logging;

namespace TelegramBotTest
{
    // For getting data from site and convert ...
    class Scheduler
    {
        private DBWorker _DB;
        private static Scheduler s_instance;
        private static ILogger<Scheduler> _logger;

        private Scheduler(DBWorker db, ILogger<Scheduler> logger)
        {
            _DB = db;
            _logger = logger;
        }
        public static Scheduler getInstance(DBWorker db, ILogger<Scheduler>
logger) { return s_instance ??= new Scheduler(db, logger); }

        public static List <JsonPair> ToListJsonPairs(string jsonString)
        {
            List<JsonPair> jsonPairs =
JsonConvert.DeserializeObject<List<JsonPair>>(jsonString);
            return jsonPairs;
        }
```

```
        public static string GET(string url, string data = null, int recLvl
= 0)
        {
            string strReq = url;
            strReq = (data != null) ? strReq + "?" + data : strReq;
            WebRequest req = WebRequest.Create(strReq);
            try
            {
                WebResponse resp = req.GetResponse();
                Stream stream = resp.GetResponseStream();
                StreamReader sr = new StreamReader(stream);
                string Out = sr.ReadToEnd();
                sr.Close();
                return Out;

            } catch(HttpRequestException e)
            {
                if (recLvl > 5)
                {
                    _logger.LogError("SSL Ban.Count Try = {recLvl}: {e}",
recLvl, e);
                    return null;
                }

                System.Threading.Thread.Sleep(200);
                return GET(url, data, ++recLvl);
            }
        } // IT to do get respone on url with data
        public static string getStrFromJson(string json)
        {
            string res = json.Trim('{').Trim('}').Replace('"', ' ');
            return Regex.Unescape(res);
        }

        public PairDay getPairDayNow(User user, string url, string data =
null)
        {
            string date =
Pair.getUnNormalDate(Pair.getNormalDate(DateTime.Now));
            data = $"{data}{user.GrpID}&date_beg={date}&date_end={date}";
            List<JsonPair> jsonPairs = ToListJsonPairs(GET(url, data));
            List<Pair> pairs = ToPairList(jsonPairs);
            return new PairDay(pairs);
        }
        public PairDay getPairDayNow(long grpId, string url, string data =
null)
        {
            string date =
Pair.getUnNormalDate(Pair.getNormalDate(DateTime.Now));
```

```
            data = $"{data}{grpId}&date_beg={date}&date_end={date}";
            List<JsonPair> jsonPairs = ToListJsonPairs(GET(url, data));
            List<Pair> pairs = ToPairList(jsonPairs);
            return new PairDay(pairs);
        }
        public PairDays getPairDays(User user, string url, string data =
null, string[] date = null)
        {
            date ??= new string[] {
Pair.getUnNormalDate(Pair.getNormalDate(DateTime.Now)),
Pair.getUnNormalDate(Pair.getNormalDate(DateTime.Now)) };
            date[1] ??= new string(date[0]);
            data = $"{data}
{user.GrpID}&date_beg={date[0]}&date_end={date[1]}";
            List<JsonPair> jsonPairs = ToListJsonPairs(GET(url, data));
            List<Pair> pairs = ToPairList(jsonPairs);
            return new PairDays(pairs);
        }
        public PairDays getPairDays(long grpId, string url, string data =
null, string[] date = null)
        {
            date ??= new string[] {
Pair.getUnNormalDate(Pair.getNormalDate(DateTime.Now)),
Pair.getUnNormalDate(Pair.getNormalDate(DateTime.Now)) };
            date[1] ??= new string(date[0]);
            data = $"{data}{grpId}&date_beg={date[0]}&date_end={date[1]}";
            List<JsonPair> jsonPairs = ToListJsonPairs(GET(url, data));
            List<Pair> pairs = ToPairList(jsonPairs);
            return new PairDays(pairs);
        }

        public List<Pair> ToPairList(List<JsonPair> jsonPairs)
        {
            List<Pair> pairs = new List<Pair>();
            foreach(JsonPair json in jsonPairs) { pairs.Add(new
Pair(json));}
            return pairs;
        }
        public void UpdateGroup(string connection)
        {
            Group[] oldGrps = _DB.readAllGroups();
            List<Group> oldGrpsList = new List<Group>();
            List<Group> needUpdateList = new List<Group>();
            List<Group> needInsertList = new List<Group>();
            foreach(Group grp in oldGrpsList) { oldGrpsList.Add(grp); }

            string res = Scheduler.GET(connection, "method=getGroups");
            res = Scheduler.getStrFromJson(res);
            string[] rows = res.Split(',');
```

```
            Group[] grps = new Group[rows.Length];
            for (int i = 0; i < grps.Length; i++)
            {
                grps[i] = new Group(rows[i]);

                if (oldGrpsList.Find(x => x.Id == grps[i].Id) == null)
                {
                    needInsertList.Add(grps[i]);
                }
                else if (oldGrpsList.Find(x => x.Id == grps[i].Id &&
x.GrpName != grps[i].GrpName) != null)
                {
                    needUpdateList.Add(grps[i]);
                }
            }

            if (needInsertList.Count != 0)
_DB.insertPoolGroups(needInsertList);

            foreach(Group grp in needUpdateList)
            {
                _DB.updateGroup(grp);
            }
        }

    }
}
```

A6 UserController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TelegramBotTest
{
    class UserController
    {
        private List<User> _userRegistrationList;
        private List<User> _waitUserRegInputList;
        private List<User> _waitUserStarostRegClickList;

        private List<User> _waitStarostaSelectDscList;
        private List<User> _waitStarostaSelectLinkTypeClickList;
        private List<User> _waitStarostaLinkYesNoClick;
        private List<User> _waitStarostaLinkEnter;
        private List<User> _waitStarostaEnterZoomIdLink;
```

```
        private List<User> _waitStarostaEnterZoomPass;

        private List<long> _InCreateAltPair;
        private List<long> _waitEnterDateAltPair;

        /**
         * <summary>
         *  Contain string DATE_REG of AltPair
         *  for user with userId
         * </summary>
         */
        private SortedList<long, string> _waitEnterNumberAltPair;
        /**
         * <summary>
         * For contain AltPair for User
         * with long userId
         * </summary>
         */
        private SortedList<long, AltPair> _waitEnterInfoAltPair;

        private SortedList<long,PairLink> _waitStarostaDonePairLinkList;
        private SortedList<long, PairLink> _waitStarostaYesNoZoomIdLink;
        private SortedList<long, PairLink> _waitStarostaYesNoZoomPass;

        private static UserController s_instance;

        private UserController()
        {
            _userRegistrationList = new List<User>();
            _waitUserRegInputList = new List<User>();
            _waitUserStarostRegClickList = new List<User>();

            _waitStarostaSelectDscList = new List<User>();
            _waitStarostaSelectLinkTypeClickList = new List<User>();
            _waitStarostaLinkYesNoClick = new List<User>();
            _waitStarostaLinkEnter = new List<User>();
            _waitStarostaEnterZoomIdLink = new List<User>();
            _waitStarostaEnterZoomPass = new List<User>();

            _InCreateAltPair = new List<long>();
            _waitEnterDateAltPair = new List<long>();
            _waitEnterNumberAltPair = new SortedList<long, string>();
            _waitEnterInfoAltPair = new SortedList<long, AltPair>();

            _waitStarostaDonePairLinkList = new SortedList<long,
PairLink>();
            _waitStarostaYesNoZoomIdLink = new SortedList<long, PairLink>();
            _waitStarostaYesNoZoomPass = new SortedList<long, PairLink>();
        }
```

```
        public static UserController getInstance() { return s_instance ??=
new UserController(); }

        #region Get Index
        public int getUserRegistrationIndex(long userId) { return
_userRegistrationList.FindIndex(x => x.Id == userId); }
        public int getUserRegistrationIndex(User user)
        {
            if (user == null) return -1;
            return getUserRegistrationIndex(user.Id);
        }

        public int getWaitUserInputIndex(long userId) { return
_waitUserRegInputList.FindIndex(x => x.Id == userId); }
        public int getWaitUserInputIndex(User user)
        {
            if (user == null) return -1;
            return getWaitUserInputIndex(user.Id);
        }

        public int getWaitUserStarostaRegClickIndex(long userId) { return
_waitUserStarostRegClickList.FindIndex(x => x.Id == userId); }
        public int getWaitUserStarostaRegClickIndex(User user)
        {
            if (user == null) return -1;
            return getWaitUserStarostaRegClickIndex(user.Id);
        }

        public int getWaitStarostaSelectDscIndex(long userId) { return
_waitStarostaSelectDscList.FindIndex(x => x.Id == userId); }
        public int getWaitStarostaSelectDscIndex(User user)
        {
            if (user == null) return -1;
            return getWaitStarostaSelectDscIndex(user.Id);
        }

        public int getWaitStarostaSelectLinkTypeClickIndex(long userId) {
return _waitStarostaSelectLinkTypeClickList.FindIndex(x => x.Id ==
userId); }
        public int getWaitStarostaSelectLinkTypeClickIndex(User user)
        {
            if (user == null) return -1;
            return getWaitStarostaSelectLinkTypeClickIndex(user.Id);
        }

        public int getWaitStarostaLinkYesNoClickIndex(long userId) { return
_waitStarostaLinkYesNoClick.FindIndex(x => x.Id == userId); }
        public int getWaitStarostaLinkYesNoClickIndex(User user)
        {
```

```
            if (user == null) return -1;
            return getWaitStarostaLinkYesNoClickIndex(user.Id);
        }

        public int getWaitStarostaLinkEnterIndex(long userId) { return
_waitStarostaLinkEnter.FindIndex(x => x.Id == userId); }
        public int getWaitStarostaLinkEnterIndex(User user)
        {
            if (user == null) return -1;
            return getWaitStarostaLinkEnterIndex(user.Id);
        }

        public int getWaitStarostaDonePairLinkIndex(long userId)
        {
            return _waitStarostaDonePairLinkList.IndexOfKey(userId);
        }
        public int getWaitStarostaDonePairLinkIndex(User user)
        {
            if (user == null) return -1;
            return getWaitStarostaDonePairLinkIndex(user.Id);
        }

        public int getWaitStarostaYesNoZoomIdLinkIndex(long userId) { return
_waitStarostaYesNoZoomIdLink.IndexOfKey(userId); }
        public int getWaitStarostaYesNoZoomIdLinkIndex(User user)
        {
            if (user == null) return -1;
            return getWaitStarostaYesNoZoomIdLinkIndex(user.Id);
        }

        public int getWaitStarostaEnterZoomIdLinkIndex(long userId) { return
_waitStarostaEnterZoomIdLink.FindIndex(x => x.Id == userId); }
        public int getWaitStarostaEnterZoomIdLinkIndex(User user)
        {
            if (user == null) return -1;
            return getWaitStarostaEnterZoomIdLinkIndex(user.Id);
        }

        public int getWaitStarostaYesNoZoomPassIndex(long userId) { return
_waitStarostaYesNoZoomPass.IndexOfKey(userId); }
        public int getWaitStarostaYesNoZoomPassIndex(User user)
        {
            if (user == null) return -1;
            return getWaitStarostaYesNoZoomPassIndex(user.Id);
        }

        public int getWaitStarostaEnterZoomPassIndex(long userId) { return
_waitStarostaEnterZoomPass.FindIndex(x => x.Id == userId); }
        public int getWaitStarostaEnterZoomPassIndex(User user)
```

```
        {
            if (user == null) return -1;
            return getWaitStarostaEnterZoomPassIndex(user.Id);
        }

        public int getInCreateAltPairIndex(long userId) { return
_InCreateAltPair.FindIndex(x => x == userId); }
        public int getWaitEnterDateAltPairIndex(long userId) { return
_waitEnterDateAltPair.FindIndex(x => x == userId); }
        public int getWaitEnterNumberAltPairIndex(long userId) { return
_waitEnterNumberAltPair.IndexOfKey(userId); }
        public int getWaitEnterInfoAltPairIndex(long userId) { return
_waitEnterInfoAltPair.IndexOfKey(userId); }
        #endregion

        #region Is In
        public bool IsInRegistration(User user) { return
getUserRegistrationIndex(user) != -1; }
        public bool IsInWaitUserInput(User user) { return
getWaitUserInputIndex(user) != -1; }
        public bool IsInWaitUserStarostaClick(User user) { return
getWaitUserStarostaRegClickIndex(user) != -1; }
        public bool IsInWaitStarostaSelectDsc(User user) { return
getWaitStarostaSelectDscIndex(user) != -1; }
        public bool IsInWaitStarostaSelectLinkTypeClick(User user) { return
getWaitStarostaSelectLinkTypeClickIndex(user) != -1; }
        public bool IsInWaitStarostaLinkYesNoClick(User user) { return
getWaitStarostaLinkYesNoClickIndex(user) != -1; }
        public bool IsInWaitStarostaLinkEnter(User user) { return
getWaitStarostaLinkEnterIndex(user) != -1; }

        public bool IsInWaitStarostaDonePairLink(long userId) { return
getWaitStarostaDonePairLinkIndex(userId) != -1; }
        public bool IsInWaitStarostaDonePairLink(User user) { return
getWaitStarostaDonePairLinkIndex(user.Id) != -1; }
        public bool IsInWaitStarostaYesNoZoomIdLink(long userId) { return
getWaitStarostaYesNoZoomIdLinkIndex(userId) != -1; }
        public bool IsInWaitStarostaYesNoZoomIdLink(User user) { return
getWaitStarostaYesNoZoomIdLinkIndex(user) != -1; }
        public bool IsInWaitStarostaEnterZoomIdLink(long userId) { return
getWaitStarostaEnterZoomIdLinkIndex(userId) != -1; }
        public bool IsInWaitStarostaEnterZoomIdLink(User user) { return
getWaitStarostaEnterZoomIdLinkIndex(user) != -1; }
        public bool IsInWaitStarostaYesNoZoomPass(long userId) { return
getWaitStarostaYesNoZoomPassIndex(userId) != -1; }
        public bool IsInWaitStarostaYesNoZoomPass(User user) { return
getWaitStarostaYesNoZoomPassIndex(user) != -1; }
        public bool IsInWaitStarostaEnterZoomPass(long userId) { return
getWaitStarostaEnterZoomPassIndex(userId) != -1; }
```

```
        public bool IsInWaitStarostaEnterZoomPass(User user) { return
getWaitStarostaEnterZoomPassIndex(user) != -1; }

        public bool IsInCreateAltPair(long userId) { return
getInCreateAltPairIndex(userId) != -1; }
        public bool IsInWaitEnterDateAltPair(long userId) { return
getWaitEnterDateAltPairIndex(userId) != -1; }
        public bool IsInWaitEnterNumberAltPair(long userId) { return
getWaitEnterNumberAltPairIndex(userId) != -1; }
        public bool IsInWaitEnterInfoAltPair(long userId) { return
getWaitEnterInfoAltPairIndex(userId) != -1; }
        #endregion

        #region Add
        public bool AddToRegistrationUser(User user)
        {
            if (user == null || IsInRegistration(user)) return false;

            _userRegistrationList.Add(user);
            return true;
        }
        public bool AddToWaitUserInput(User user)
        {
            if (user == null || IsInWaitUserInput(user)) return false;

            _waitUserRegInputList.Add(user);
            return true;
        }
        public bool AddToWaitUserStarostaClick(User user)
        {
            if (user == null || IsInWaitUserStarostaClick(user)) return
false;

            _waitUserStarostRegClickList.Add(user);
            return true;
        }
        public bool AddToWaitStarostaSelectDsc(User user)
        {
            if (user == null || IsInWaitStarostaSelectDsc(user)) return
false;

            _waitStarostaSelectDscList.Add(user);
            return true;
        }
        public bool AddToWaitStarostaSelectLinkClick(User user)
        {
            if (user == null || IsInWaitStarostaSelectLinkTypeClick(user))
return false;
```

```
            _waitStarostaSelectLinkTypeClickList.Add(user);
            return true;
        }
        public bool AddToWaitStarostaLinkYesNoClick(User user)
        {
            if (user == null || IsInWaitStarostaLinkYesNoClick(user)) return
false;

            _waitStarostaLinkYesNoClick.Add(user);
            return true;
        }
        public bool AddToWaitStarostaLinkEnter(User user)
        {
            if (user == null || IsInWaitStarostaLinkEnter(user)) return
false;

            _waitStarostaLinkEnter.Add(user);
            return true;
        }
        public bool AddToWaitStarostaEnterZoomIdLink(User user)
        {
            if (user == null || IsInWaitStarostaEnterZoomIdLink(user))
return false;

            _waitStarostaEnterZoomIdLink.Add(user);
            return true;
        }
        public bool AddToWaitStarostaEnterZoomPass(User user)
        {
            if (user == null || IsInWaitStarostaEnterZoomPass(user)) return
false;

            _waitStarostaEnterZoomPass.Add(user);
            return true;
        }

        public bool AddToWaitStarostaDonePairLink(long userId,  PairLink
pairLink)
        {
            if (pairLink == null || IsInWaitStarostaDonePairLink(userId))
return false;

            _waitStarostaDonePairLinkList.Add(userId, pairLink);
            return true;
        }
        public bool AddToWaitStarostaYesNoZoomIdLink(long userId, PairLink
pairLink)
        {
```

```
            if (pairLink == null || IsInWaitStarostaYesNoZoomIdLink(userId))
return false;

            _waitStarostaYesNoZoomIdLink.Add(userId, pairLink);
            return true;
        }
        public bool AddToWaitStarostaYesNoZoomPass(long userId, PairLink
pairLink)
        {
            if (pairLink == null || IsInWaitStarostaYesNoZoomPass(userId))
return false;

            _waitStarostaYesNoZoomPass.Add(userId, pairLink);
            return true;
        }

        public bool AddToInCreateAltPair(long userId)
        {
            if (IsInCreateAltPair(userId)) return false;
            _InCreateAltPair.Add(userId);
            return true;
        }
        public bool AddToWaitEnterDateAltPair(long userId)
        {
            if (IsInWaitEnterDateAltPair(userId)) return false;
            _waitEnterDateAltPair.Add(userId);
            return true;
        }

        public bool AddToWaitEnterNumberAltPair(long userId, string date)
        {
            if (string.IsNullOrEmpty(date) ||
string.IsNullOrWhiteSpace(date)
                || IsInWaitEnterNumberAltPair(userId))
            {
                return false;
            }
            _waitEnterNumberAltPair.Add(userId, date);
            return true;
        }
        public bool AddToWaitEnterInfoAltPair(long userId, AltPair altPair)
        {
            if (altPair == null || IsInWaitEnterInfoAltPair(userId)) return
false;
            _waitEnterInfoAltPair.Add(userId, altPair);
            return true;
        }
        #endregion
```

```
#region Un Add
public bool UnRegistrationUser(User user)
{
    int index = getUserRegistrationIndex(user);

    if (index == -1) return false;
    _userRegistrationList.RemoveAt(index);
    return true;
}
public bool UnWaitUserInput(User user)
{
    int index = getWaitUserInputIndex(user);

    if (index == -1) return false;
    _waitUserRegInputList.RemoveAt(index);
    return true;
}
public bool UnWaitUserStarostaClick(User user)
{
    int index = getWaitUserStarostaRegClickIndex(user);

    if (index == -1) return false;
    _waitUserStarostRegClickList.RemoveAt(index);
    return true;
}
public bool UnWaitStarostaSelectDsc(User user)
{
    int index = getWaitStarostaSelectDscIndex(user);

    if (index == -1) return false;
    _waitStarostaSelectDscList.RemoveAt(index);
    return true;
}
public bool UnWaitStarostaSelectLinkTypeClick(User user)
{
    int index = getWaitStarostaSelectLinkTypeClickIndex(user);

    if (index == -1) return false;
    _waitStarostaSelectLinkTypeClickList.RemoveAt(index);
    return true;
}
public bool UnWaitStarostaLinkYesNoClick(User user)
{
    int index = getWaitStarostaLinkYesNoClickIndex(user);

    if (index == -1) return false;
    _waitStarostaLinkYesNoClick.RemoveAt(index);
    return true;
}
```

```
public bool UnWaitStarostaLinkEnter(User user)
{
    int index = getWaitStarostaLinkEnterIndex(user);

    if (index == -1) return false;
    _waitStarostaLinkEnter.RemoveAt(index);
    return true;
}

public bool UnWaitStarostaDonePairLink(User user)
{
    int index = getWaitStarostaDonePairLinkIndex(user);

    if (index == -1) return false;
    _waitStarostaDonePairLinkList.RemoveAt(index);
    return true;
}
public bool UnWaitStarostaYesNoZoomIdLink(User user)
{
    int index = getWaitStarostaYesNoZoomIdLinkIndex(user);

    if (index == -1) return false;
    _waitStarostaYesNoZoomIdLink.RemoveAt(index);
    return true;
}
public bool UnWaitStarostaEnterZoomIdLink(User user)
{
    int index = getWaitStarostaEnterZoomIdLinkIndex(user);

    if (index == -1) return false;
    _waitStarostaEnterZoomIdLink.RemoveAt(index);
    return true;
}
public bool UnWaitStarostaYesNoZoomPass(User user)
{
    int index = getWaitStarostaYesNoZoomPassIndex(user);

    if (index == -1) return false;
    _waitStarostaYesNoZoomPass.RemoveAt(index);
    return true;
}
public bool UnWaitStarostaEnterZoomPass(User user)
{
    int index = getWaitStarostaEnterZoomPassIndex(user);

    if (index == -1) return false;
    _waitStarostaEnterZoomPass.RemoveAt(index);
    return true;
}
```

```
        public bool UnInCreateAltPair(long userId)
        {
            int index = getInCreateAltPairIndex(userId);

            if (index == -1) return false;
            _InCreateAltPair.RemoveAt(index);
            return true;
        }
        public bool UnWaitEnteDaterAltPair(long userId)
        {
            int index = getWaitEnterDateAltPairIndex(userId);

            if (index == -1) return false;
            _waitEnterDateAltPair.RemoveAt(index);
            return true;
        }

        public bool UnWaitEnterNumberAltPair(long userId)
        {
            int index = getWaitEnterNumberAltPairIndex(userId);

            if (index == -1) return false;
            _waitEnterNumberAltPair.RemoveAt(index);
            return true;
        }
        public bool UnWaitEnterInfoAltPair(long userId)
        {
            int index = getWaitEnterInfoAltPairIndex(userId);

            if (index == -1) return false;
            _waitEnterInfoAltPair.RemoveAt(index);
            return true;
        }
        #endregion

        #region GET USER
        public User getUserInWaitUserStarostaClick(long userId) { return
_waitUserStarostRegClickList.Find(x => x.Id == userId); }
        public User getUserInWaitStarostaLinkYesNoClick(long userId) {
return _waitStarostaLinkYesNoClick.Find(x => x.Id == userId); }
        public User getUserInWaitStarostaSelectDsc(long userId) { return
_waitStarostaSelectDscList.Find(x => x.Id == userId); }
        public User getUserInWaitStarostaSelectLinkTypeClick(long userId) {
return _waitStarostaSelectLinkTypeClickList.Find(x => x.Id == userId); }
        #endregion

        #region GET PAIRLINK
        public PairLink getPairLinkInWaitStarostaDonePairLink(long userId)
```

```
        {
            int index = getWaitStarostaDonePairLinkIndex(userId);

            if (index == -1) return null;
            return _waitStarostaDonePairLinkList.ElementAt(index).Value;
        }
        public PairLink getPairLinkInWaitStarostaYesNoZoomIdLink(long
userId)
        {
            int index = getWaitStarostaYesNoZoomIdLinkIndex(userId);

            if (index == -1) return null;
            return _waitStarostaYesNoZoomIdLink.ElementAt(index).Value;
        }
        public PairLink getPairLinkInWaitStarostaYesNoZoomPass(long userId)
        {
            int index = getWaitStarostaYesNoZoomPassIndex(userId);

            if (index == -1) return null;
            return _waitStarostaYesNoZoomPass.ElementAt(index).Value;
        }
        #endregion

        /**
         * <returns>
         * Return string DATE_REG of AltPair
         * if good; otherwise null
         * </returns>
         */
        public string getStrWaitEnterNumberAltPair(long userId)
        {
            int index = getWaitEnterNumberAltPairIndex(userId);

            if (index == -1) return null;
            return _waitEnterNumberAltPair.ElementAt(index).Value;
        }
        public AltPair getAltPairWaitEnterInfoAltPair(long userId)
        {
            int index = getWaitEnterInfoAltPairIndex(userId);

            if (index == -1) return null;
            return _waitEnterInfoAltPair.ElementAt(index).Value;

        }
        public bool UpdateRegistrationUser(User user)
        {
            if (!IsInRegistration(user)) return false;

            _userRegistrationList[getUserRegistrationIndex(user)] = user;
```

```
            return true;
        }

        public bool UpdateWaitStarostaDonePairLink(long userId, PairLink
pairLink)
        {
            int index = getWaitStarostaDonePairLinkIndex(userId);

            if (index == -1) return false;

            _waitStarostaDonePairLinkList[index] = pairLink;
            return true;
        }

    }
}
```

A7 BotController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Telegram.Bot;
using Telegram.Bot.Args;
using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types.ReplyMarkups;

namespace TelegramBotTest
{
    class BotController
    {
        private TelegramBotClient _Bot;
        private static BotController s_instance;

        private BotController(TelegramBotClient bot) { _Bot = bot; }
        public static BotController getInstance(TelegramBotClient bot) {
return s_instance ??= new BotController(bot); }

        public int getNumberButtonRows(int count)
        {
            if (count == 0) return 0;
            int numberRows = (count) / 5;
            numberRows += (count % 5 != 0) ? 1 : 0;
            return numberRows;
        }
        public void sendRegisterText(User user, long chatId)
        {
            string text = "Enter Group Name.(may one letter):";
```

```
            _Bot?.SendTextMessageAsync(chatId, text);
        }
        public void sendDscList(long chatId, SortedList<long, string>
disciplines, string message)
        {
            List<List<InlineKeyboardButton>> rowsList = new
List<List<InlineKeyboardButton>>();
            foreach (var dsc in disciplines)
            {
                InlineKeyboardButton keyboardButton = new
InlineKeyboardButton();
                keyboardButton.Text = dsc.Value;
                keyboardButton.CallbackData = dsc.Key.ToString();
                List<InlineKeyboardButton> row = new
List<InlineKeyboardButton>();
                row.Add(keyboardButton);
                rowsList.Add(row);
            }
            InlineKeyboardMarkup keyboardMarkup = new
InlineKeyboardMarkup(rowsList);
            sendReplyMarkup(chatId, message, keyboardMarkup);
        }
        public bool sendGroupTgListButtons(Group[] grpList, long chatId)
        {
            if (grpList == null || grpList.Length == 0)
            {
                _Bot.SendTextMessageAsync(chatId, "Not Found Any Group!
Repeat Input");
                return false;
            }

            int numberRows = getNumberButtonRows(grpList.Length);

            List<List<InlineKeyboardButton>> rowsList = new
List<List<InlineKeyboardButton>>();
            for (int i = 0; i < numberRows; i++)
            {
                List<InlineKeyboardButton> keyboardButtons = new
List<InlineKeyboardButton>();
                rowsList.Add(keyboardButtons);
            }

            for (int i = 0; i < grpList.Length; i++)
            {
                InlineKeyboardButton inlineKeyboardButton = new
InlineKeyboardButton();
                inlineKeyboardButton.Text = grpList[i].GrpName;
                inlineKeyboardButton.CallbackData = $"{grpList[i].Id}";
```

```
                int index = i + 1;
                index = index / 5;
                index = (i + 1) % 5 == 0 ? --index : index;

                rowsList[index].Add(inlineKeyboardButton);
            }
            InlineKeyboardMarkup inlineKeyboardMarkup = new
InlineKeyboardMarkup(rowsList);
            sendReplyMarkup(chatId, "Select:", inlineKeyboardMarkup);
            return true;
        }
        public void sendYesNo(long chatId, string question, string specYes =
null, string specNo = null)
        {
            InlineKeyboardButton keyYes = new InlineKeyboardButton();
            InlineKeyboardButton keyNo = new InlineKeyboardButton();
            keyYes.Text = "Yes";
            keyYes.CallbackData = specYes ??= "/yes";
            keyNo.Text = "No";
            keyNo.CallbackData = specNo ??= "/no";
            List<InlineKeyboardButton> row = new
List<InlineKeyboardButton>();
            row.Add(keyYes);
            row.Add(keyNo);
            InlineKeyboardMarkup inlineKeyboard = new
InlineKeyboardMarkup(row);
            sendReplyMarkup(chatId, question, inlineKeyboard);
        }
        public void sendPair(long chatId, Pair pair)
        {
            string text = $"{pair.NumPair} [{pair.TypePair}]\
n{pair.ShortNamePair}\nWho : {pair.TeacherName}\n";
            text = $"{text}\nTime : {pair.PairTime}\n";

            if (!string.IsNullOrEmpty(pair.Auditoria)||!
string.IsNullOrWhiteSpace(pair.Auditoria))
            {
                text = $"{text}\nAuditoria: {pair.Auditoria}\n";
            }
            else
            {
                //text = $"{text}\n";

                if (pair.WorkLink != null)
                {
                    text = $"{text}{pair.WorkLink}";
                }
            }
            _Bot.SendTextMessageAsync(chatId, text);
```

```
        }
        public void sendAltPair(long chatId, AltPair altPair)
        {
            string text = $"\nFound Alternative Pair:\
n[{AltPair.NumberPairToString(altPair.NUM_PAIR)}]\nInfo: {altPair.INFO}\n";
            text = $"{text}Time:
{AltPair.NumberPairToTimeInterval(altPair.NUM_PAIR)}\n";
            text = $"{text}Date: {Pair.getUnNormalDate(altPair.DATE_REG)}";
            _Bot.SendTextMessageAsync(chatId, text);
        }

        public void sendLinkDscTypeListButtons(long chatId, long dscId,
string message)
        {
            string[] types = { "лекція", "лабораторна\n/практична",
"атестація", "іспит" };
            List<InlineKeyboardButton> keyboardButtons = new
List<InlineKeyboardButton>();
            foreach(var type in types)
            {
                keyboardButtons.Add(getButtonDscType(dscId, type));
            }
            InlineKeyboardMarkup keyboardMarkup = new
InlineKeyboardMarkup(keyboardButtons);
            sendReplyMarkup(chatId, message, keyboardMarkup);
        }

        public void sendReplyMarkup(long chatId, string message,
IReplyMarkup replyMarkup)
        {
            _Bot.SendTextMessageAsync(chatId, message, ParseMode.Default,
                null, false, false, 0, false, replyMarkup
            );
        }

        public InlineKeyboardButton getButtonDscType(long dscId, string
type)
        {
            InlineKeyboardButton keyboardButton = new
InlineKeyboardButton();
            keyboardButton.Text = type;
            keyboardButton.CallbackData =
$"{dscId}={PairLink.getDscType(type)}";
            return keyboardButton;
        }
        public static Time getTimeInMessage(string message)
        {
            string strhour = message.Split("/now")[1].Split(":")[0];
            string strminute = message.Split("/now")[1].Split(":")[1];
```

```
            int hour;
            int.TryParse(string.Join("", strhour.Where(c =>
char.IsDigit(c))), out hour);

            int minute;
            int.TryParse(string.Join("", strminute.Where(c =>
char.IsDigit(c))), out minute);

            Time responseTime = new Time(hour, minute);
            return responseTime;
        }
    }
}
```

ДОДАТОК Б
ФІЗИЧНА РЕАЛІЗАЦІЯ БД ВЕБ-ДОДАТКУ

Сценарій створення структури реляційної БД в Postgres:

```
DROP TABLE ALTPAIR;
DROP TABLE GROUPCHATS;
DROP TABLE PAIRLINKS;
DROP TABLE STAROSTA;
DROP TABLE STUDENTS;
DROP TABLE GROUPS;

CREATE TABLE GROUPS(
   ID BIGSERIAL,
   NAME VARCHAR,
   PRIMARY KEY(ID)
);

CREATE TABLE STUDENTS(
   ID BIGINT, -- USER TELEGRAM ID
   GRP_ID BIGINT,
   PRIMARY KEY(ID)
);

CREATE TABLE STAROSTA(
   STD_ID BIGINT,
   GRP_ID BIGINT,
   PRIMARY KEY(STD_ID, GRP_ID)
);

CREATE TABLE PAIRLINKS(
   DSC_ID BIGINT, --KOD_DISC
   GRP_ID BIGINT,
   DSC_TYPE INT, --TYPE DISC: None, lec, lab ...
   LINK TEXT,
   LINK_ID TEXT, --ZOOM ID
   PASS TEXT, --ZOOM PASS
   PRIMARY KEY(DSC_ID, GRP_ID, DSC_TYPE)
);
CREATE TABLE GROUPCHATS(
```

```
    ID BIGINT, --TELEGRAM CHAT ID
    GRP_ID BIGINT,
    PRIMARY KEY(ID)
);
CREATE TABLE ALTPAIR(
    DATE_REG DATE,
    NUM_PAIR INT,
    GRP_ID BIGINT,
    INFO TEXT,
    PRIMARY KEY(DATE_REG, NUM_PAIR, GRP_ID)
);
ALTER TABLE STUDENTS ADD FOREIGN KEY(GRP_ID) REFERENCES GROUPS(ID);
ALTER TABLE STAROSTA ADD FOREIGN KEY(STD_ID) REFERENCES STUDENTS(ID);
ALTER TABLE STAROSTA ADD FOREIGN KEY(GRP_ID) REFERENCES GROUPS(ID);
ALTER TABLE PAIRLINKS ADD FOREIGN KEY(GRP_ID) REFERENCES GROUPS(ID);
ALTER TABLE GROUPCHATS ADD FOREIGN KEY(GRP_ID) REFERENCES GROUPS(ID);
ALTER TABLE ALTPAIR ADD FOREIGN KEY(GRP_ID) REFERENCES GROUPS(ID);
```