МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
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 обліку оренди місць у колективному офісі»
здобувача групи ІНз-91с Савченко Дмитро Сергійович

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

_____ Дмитро САВЧЕНКО
(підпис)

Керівник, Наталія БАРЧЕНКО
к.т.н., доцент _____
(підпис)

СУМИ 2023

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
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 ІНз-91с Савченко Дмитро Сергійович

1. Тема роботи: «Інформаційна система обліку оренди місць у колективному офісі»
затверджую наказом по СумДУ від *«01» червня 2023 р. № 0475-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз відомих рішень. 2) Вибір основних компонентів боту та їх реалізація. 3) Комп'ютерна реалізація проекту та тестування.
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 ___ р.

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

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз відомих рішень*	березень 2023	
2	*Вибір основних компонентів боту та їх реалізація*	березень 2023	
3	*Комп'ютерна реалізація проекту та тестування*	квітень 2023	
4	*Оформлення пояснювальної записки до кваліфікаційної роботи*	травень 2023	

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

АНОТАЦІЯ

Записка: 35 стор., 5 рис., 1 додаток, 25 джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв’язанню важливої практичної задачі системи обліку оренди місць у колективному офісі, шляхом розробки відповідних методів, моделей та інформаційних технологій.

Об’єкт дослідження — чат-бот для месенджера Telegram

Мета роботи — інформаційне та програмне забезпечення для оренди місць у колективному офісі.

Методи дослідження — технології створення ІС для оренди місць, технології представлення таких ІС (веб-застосунок, чат-бот, мобільний додаток)

Результати — розроблено інформаційну систему з оренди місць у колективному офісі для месенджера Telegram. Створений продукт має зручний інтерфейс, процеси відбуваються у вигляді живого спілкування. Розроблено ІС за допомогою мови Python, під час тестування проблем не виявлено.

COWORKING, ІНФОРМАЦІЙНА СИСТЕМА, PYTHON, ASYNCIO, SQLITE

ЗМІСТ

ВСТУП

Розвиток коворкінгів став однією з найважливіших тенденцій в сучасному світі бізнесу. Вони надають можливість працювати в затишному та продуктивному середовищі, що сприяє збільшенню ефективності та креативності працівників [4].

Однак, з ростом популярності коворкінгів зростає й кількість клієнтів, що призводить до збільшення складності управління цими приміщеннями та їхніми ресурсами. Одним з основних аспектів управління коворкінгом є оренда місць праці, яка є основним джерелом прибутку для власників коворкінгів [2]. Тому, для забезпечення ефективного управління коворкінгом необхідно мати ефективну інформаційну систему оренди місць.

Така інформаційна система дозволить власникам коворкінгу з легкістю керувати розкладом оренди місць, забезпечити точну та своєчасну фінансову звітність, а також підвищити рівень безпеки та контролю доступу до приміщення. Також, вона зможе забезпечити користувачам легкий та швидкий процес бронювання місць, оплату та контроль їх використання.

Моя робота буде націлена на розробку та впровадження ефективної інформаційної системи оренди місць у коворкінгу. Для цього я проведу дослідження вимог користувачів та побудую моделі процесів, що відбуваються у коворкінгу [5]. Крім того, я використаю математичні методи та алгоритми, щоб розробити систему, яка буде максимально ефективною та з можливостями розширення та модифікації в майбутньому.

Метою моєї роботи є створення комплексної інформаційної системи, яка забезпечить ефективне та безпечне управління орендою місць у коворкінгу, зменшить витрати часу та ресурсів на ручну обробку даних, та поліпшить комунікацію між користувачами та власниками коворкінгу.

Результатом моєї роботи буде функціонуюча інформаційна система, яка буде включати в себе всі необхідні функції для ефективного управління орендою місць у коворкінгу. Крім того, я врахую потенційні потреби користувачів та власників коворкінгу щодо функцій та можливостей системи, що забезпечить високу цінність та задоволення від користування системою [1].

Ця робота є важливим кроком у розвитку коворкінгів як ефективного та зручного способу працювати. Моя інформаційна система дозволить власникам коворкінгу зосередитися на розвитку бізнесу та наданні якісних послуг користувачам, забезпечивши при цьому ефективне та безпечне управління орендою місць.

1 АНАЛІЗ ВІДОМИХ РІШЕНЬ

Для створення ефективної інформаційної системи обліку орендованих місць у колективному офісі, потрібно проаналізувати можливі аналоги, недоліки яких дозволять зрозуміти, на чому базуватиметься рішення.

1.1 Огляд існуючих рішень

На сьогоднішній день, в Україні існує кілька інформаційних систем для управління орендою місць у коворкінгах. Розглянемо найпопулярніші з них.

1.1.1 Сервіс Coworkify

Coworkify є веб-платформою для керування коворкінгами, яка надає власникам коворкінгів засоби для керування бронюваннями, платежами та ресурсами. Сервіс дозволяє власникам коворкінгу налаштувати різні типи членства та опції оренди для користувачів, а також створювати різні типи коворкінгів з різними послугами та ресурсами.

Однією з переваг Coworkify є його простий та зрозумілий інтерфейс. Користувачі можуть легко знайти та забронювати місце в коворкінгу, а власники коворкінгів можуть легко керувати своїми ресурсами та отримувати звіти про свій бізнес.

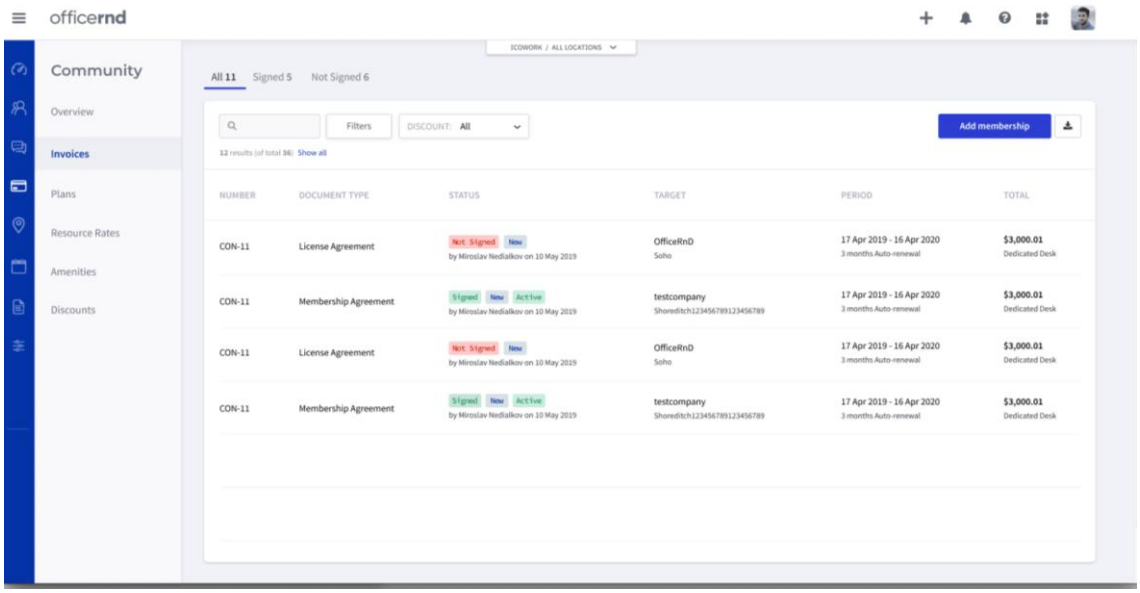

Рисунок 1.1 – зовнішній вигляд сервісу Coworkify

Крім того, Coworkify надає власникам коворкінгів можливість інтегрувати свій сайт з Coworkify API, що дозволяє отримувати замовлення та платежі безпосередньо на своєму сайті, а також налаштовувати сповіщення та повідомлення для користувачів.

Однак, Coworkify має свої обмеження. Наприклад, сервіс не надає можливості інтеграції з платформами електронної комерції, такими як PayPal або Stripe, що може становити проблему для користувачів, які хочуть платити за послуги через ці платформи.

Крім того, Coworkify не має можливості інтеграції з Slack або іншими платформами для спілкування з користувачами, що може бути проблемою для власників коворкінгів, які хочуть забезпечити зручний та ефективний спосіб спілкування зі своїми користувачами [6].

У будь-якому випадку, Coworkify є гарним вибором для власників коворкінгів, оскільки сервіс надає потужний набір інструментів для керування бізнесом. Незважаючи на обмеження, Coworkify дозволяє власникам коворкінгів зосередитися на управлінні бізнесом та покращенні досвіду користувачів.

Крім того, Coworkify пропонує різні пакети та плани, які дозволяють власникам коворкінгів обирати найбільш підходящий план для свого бізнесу. Пакети Coworkify відрізняються за кількістю користувачів, послуг та функцій, які доступні для використання.

Також важливо зазначити, що Coworkify має потужний API, що дозволяє розширювати функціональність та інтегрувати з іншими додатками. Це може бути корисно для власників коворкінгів, які хочуть додатково налаштувати свій бізнес та інтегрувати його з іншими інструментами.

Отже, Coworkify є одним з кращих сервісів для керування коворкінгом на ринку, завдяки своїй потужній функціональності та зручному інтерфейсу. Будь-який власник коворкінгу зможе знайти та налаштувати плани та пакети,

які найкраще підходять для його бізнесу, та налаштувати всі необхідні інструменти для ефективного керування бізнесом.

1.1.2 Веб-застосунок Cobot

Cobot є ще одним популярним сервісом для керування коворкінгом. Цей сервіс надає багато функціональних можливостей для ефективного керування коворкінгом, включаючи управління членами, бронювання столів, фактурування та інше.

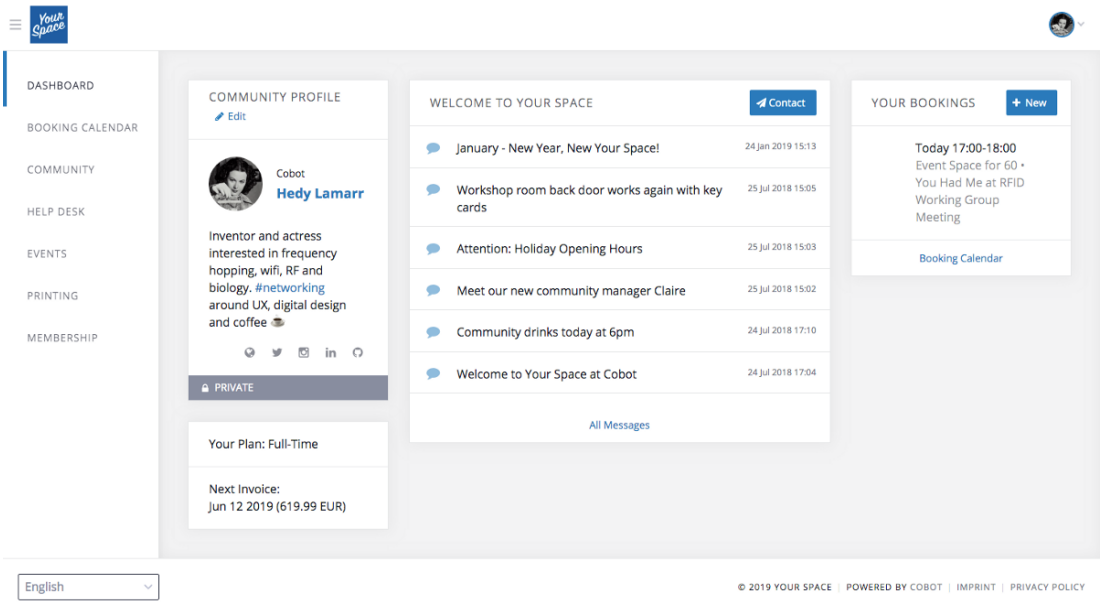

Рисунок 1.2 – зовнішній вигляд веб-застосунку Cobot

Інтерфейс Cobot є зручним та легким у використанні, що робить його доступним для використання навіть для тих, хто не має досвіду в управлінні коворкінгом. Сервіс дозволяє власникам коворкінгів налаштувати різні плани та пакети, які найкраще відповідають їхньому бізнесу.

Однією з найбільш корисних функцій Cobot є можливість взаємодії з іншими додатками та сервісами, такими як Slack, Zapier та інші. Це дозволяє власникам коворкінгів налаштувати більш інтегровану систему управління та забезпечити більш ефективне функціонування їхнього бізнесу [7].

Крім того, Cobot пропонує різні можливості налаштування системи, такі як налаштування додаткових полів для користувачів, встановлення прав

доступу до системи, налаштування сповіщень та інше. Ці можливості роблять Cobot більш гнучким та зручним у використанні для різних типів коворкінгів та бізнесів [8].

Нарешті, Cobot пропонує своїм користувачам потужний API, який дозволяє розширювати функціональність системи та інтегрувати з іншими додатками та сервісами. Це може бути особливо корисно для власників коворкінгів, які хочуть налаштувати свій бізнес та інтегрувати його з іншими інструментами, щоб отримати максимальну продуктивність та ефективність.

Одним з недоліків Cobot є його ціна. Плани починаються зі значної ціни за місяць, що може стати проблемою для менших коворкінгів, які не можуть собі цього дозволити. Крім того, Cobot не пропонує багато можливостей налаштування дизайну інтерфейсу, що може стати проблемою для власників коворкінгів, які хочуть налаштувати дизайн системи під свої потреби.

Однак, Cobot все ж є одним з найбільш розширених та потужних сервісів для керування коворкінгом на ринку. Він має багато корисних функцій та можливостей для власників коворкінгів, що дозволяє їм ефективно управляти своїм бізнесом та підтримувати високий рівень сервісу для своїх клієнтів.

1.1.3 Інтернет-платформа Nexudus

Останнім аналогом, який ми розглянемо, є інтернет-платформа Nexudus. Nexudus - це інтернет-платформа для коворкінгів, що дозволяє власникам коворкінгів керувати своїм бізнесом та надавати своїм клієнтам зручний інтерфейс для замовлення та оплати послуг [9]. Nexudus має багато корисних функцій, які допомагають власникам коворкінгів управляти своїм бізнесом та підтримувати високий рівень сервісу для своїх клієнтів [10].

Однією з головних переваг Nexudus є його легкість використання. Платформа має інтуїтивно зрозумілий інтерфейс, що дозволяє власникам коворкінгів легко використовувати різні функції системи. Крім того, Nexudus

дозволяє власникам коворкінгів залучати нових клієнтів через різні маркетингові кампанії та інструменти.

Іншою важливою функцією Nexudus є його інтегрованість. Платформа підтримує інтеграцію з багатьма іншими інструментами, такими як Stripe, PayPal, Slack, Google Calendar та інші. Це дозволяє власникам коворкінгів легко здійснювати операції з оплатою та комунікацією з клієнтами з одного місця.

Однак, Nexudus також має свої недоліки. Перш за все, платформа може бути досить дорогою для менших коворкінгів, які не можуть собі дозволити витрати на такий інструмент. Крім того, Nexudus може бути складним для встановлення та налаштування, що може вимагати додаткових зусиль від власників коворкінгів [12].

У загальному, Nexudus - це потужний інструмент для управління коворкінгами, який надає багато корисних функцій та може допомогти власникам коворкінгів ефективно керувати своїм бізнесом та надавати високоякісний сервіс для своїх клієнтів. Однак, перед вибором Nexudus, власникам коворкінгів слід враховувати вартість платформи та її складність налаштування [11].

Одна з ключових функцій Nexudus - це можливість керування членством та планами підписки. Платформа дозволяє власникам коворкінгів налаштовувати різні плани підписки, що містять різні можливості для клієнтів, такі як кількість годин, доступ до приватних офісів та інші. Крім того, Nexudus надає можливість керувати членством клієнтів та їх рахунками через вбудовану систему керування користувачами [15].

Ще однією важливою функцією Nexudus є його функціонал для бронювання робочих місць. Клієнти можуть забронювати робоче місце через онлайн-систему, що дозволяє уникнути конфліктів щодо доступності місць та збільшити ефективність використання простору коворкінгу.

Крім того, Nexudus дозволяє власникам коворкінгів стежити за використанням простору та послуг, наданих клієнтам. Платформа збирає статистику щодо використання місць та зручностей коворкінгу, що може допомогти власникам коворкінгу приймати рішення щодо удосконалення сервісу та оптимізації використання простору [10].

Недоліком Nexudus є його висока вартість, особливо для менших коворкінгів.

Враховуючи зазначені особливості та переваги, можна зробити висновок, що інформаційна система оренди місць у коворкінгу є важливим елементом для забезпечення ефективного управління коворкінгом та задоволення потреб клієнтів.

Оцінюючи різні сервіси для оренди місць у коворкінгу, можна зазначити, що кожен з них має свої переваги та недоліки, але загалом вони допомагають коворкінгам ефективно керувати своєю бізнес-діяльністю та полегшують процес резервування та управління місцями в коворкінгу [18].

Таким чином, створення власної інформаційної системи оренди місць у коворкінгу може бути вигідним рішенням, яке дозволить коворкінгу збільшити ефективність своєї роботи, задовольнити потреби своїх клієнтів та збільшити свій бізнес-потенціал.

Таблиця 1.1 Порівняльна характеристика існуючих аналогів

Сервіс	Ціна за місяць (в грн)	Наявність україномовного інтерфейсу	Інтерфейс
Coworkify	765 - 3080	Є	Інтуїтивний, адаптивний
Cobot	2878 - 4604	Немає	Простий
Nexudus	5479 - 10 958	Немає	Зручний, відповідає останнім трендам UI/UX

З таблиці видно, що Coworkify має найбільш доступну ціну для користувачів, а також має україномовний інтерфейс. У свою чергу, Cobot та Nexudus мають більш високі ціни, але зручний інтерфейс. Також варто зауважити, що наявність україномовного інтерфейсу є важливою для користувачів з України, які можуть не знати англійської мови на достатньому рівні для розуміння інтерфейсу [23].

Отже, спираючись на проаналізовані аналоги, ми можемо перейти до постановки задачі щодо розробки інформаційної системи з обліку оренди місць у колективному офісі.

1.2 Постановка задачі

Завдання даного проекту полягає у розробці інформаційної системи обліку оренди місць у коворкінгу, яка допоможе забезпечити ефективне керування ресурсами коворкінгу та підвищити задоволеність клієнтів.

Необхідно розробити наступні функціональні можливості інформаційної системи:

1. Реєстрація клієнтів. Система має надавати можливість зареєструватись новим клієнтам та зберігати їхні особисті дані.
2. Керування місцями. Система має надавати можливість керувати доступними місцями в коворкінгу, додавати нові місця, змінювати їх статус (вільне, зайняте, заброньоване тощо).
3. Бронювання місць. Система має надавати можливість клієнтам бронювати вільні місця в коворкінгу.
4. Керування користувачами системи. Система має надавати можливість керувати правами доступу користувачів, зокрема, обмежувати доступ до певних функціональних можливостей системи.

Для досягнення цієї мети потрібно виконати наступні завдання:

1. Спроектувати функціональні вимоги інформаційної системи.

2. Обрати, в якому вигляді буде представлена інформаційна система (веб-застосунок, telegram-bot, мобільний додаток тощо).
3. Обрати технології для реалізації інформаційної системи, спроектувати для неї базу даних.
4. Провести тестування ІС.

2 ВИБІР ОСНОВНИХ КОМПОНЕНТІВ БОТУ ТА ЇХ РЕАЛІЗАЦІЯ

2.1 Проектування функціоналу інформаційної системи

Згідно постановки задачі, спочатку ми повинні розглянути та спроектувати функціональні вимоги системи. Почнемо з найпростішого – реєстрація користувача.

Процес реєстрації користувача в системі обліку оренди місць у коворкінгу може складатись з наступних етапів:

1. Введення особистих даних. Користувач повинен ввести своє ім'я, прізвище, контактні дані (телефон, електронна пошта), дату народження та інші особисті дані. Для забезпечення безпеки системи, може бути запропоновано введення пароля або використання інших методів автентифікації (наприклад, двофакторна аутентифікація).
2. Вибір пакету послуг. Після введення особистих даних користувачеві може бути запропоновано вибір пакету послуг. Він може включати різні опції оренди місць, доступ до додаткових послуг (наприклад, конференц-зали, роздруківки, кухня), терміни оренди та інші параметри.
3. Підтвердження реєстрації. Після введення всіх необхідних даних та вибору пакету послуг, користувач повинен підтвердити свою реєстрацію. Це може бути здійснене через електронну пошту або за допомогою СМС-повідомлення [9].
4. Активація акаунту. Після підтвердження реєстрації акаунт користувача може бути активований. Користувач повинен отримати доступ до свого акаунту та може розпочати бронювання місць в коворкінгу.
5. Додаткові кроки. В залежності від особливостей системи та правил коворкінгу, користувач може бути запрошений здійснити додаткові кроки (наприклад, підписання угоди про оренду місця, сплачення передплати та ін.).

Спроектувавши узагальнений процес реєстрації користувача, перейдемо до фунціональної вимоги «Керування місцями».

Процес керування місцями у системі обліку оренди місць у коворкінгу включає наступні етапи:

1. Додавання нових місць: адміністратор системи має можливість додавати нові місця до бази даних. Для цього він вводить потрібні характеристики місця, такі як розташування, розмір, наявність електроживлення, доступ до Інтернету та інші важливі параметри.
2. Зміна статусу місць: система повинна мати можливість змінювати статус місць у коворкінгу, такі як "вільне", "зайняте", "забронюване" та інші. Це може бути зроблено як автоматично, на основі бронювання користувачами, так і вручну, адміністратором системи [21].
3. Відстеження наявності місць: система повинна мати можливість відстежувати наявність місць у режимі реального часу, щоб користувачі могли бронювати вільні місця.
4. Керування цінами: система повинна мати можливість налаштувати різні ціни на різні місця у коворкінгу, залежно від їхніх параметрів та розташування.
5. Звіти: система повинна зберігати інформацію про кількість вільних, зайнятих та забронюваних місць, а також про дохід, отриманий від оренди місць. Це допоможе адміністраторам коворкінгу більш ефективно керувати його ресурсами та робити ділові рішення на підставі даних.

Тепер перейдемо до процесу «Бронювання місць». Процес бронювання місць у системі обліку оренди місць в коворкінгу складається з наступних кроків:

1. Вибір вільного місця. Користувач переходить до розділу "Керування місцями" у системі та переглядає доступні вільні місця. Він вибирає зручне для себе місце та переходить до бронювання.

2. Заповнення форми бронювання. Користувач заповнює форму бронювання, в якій вказує дату та час, на які він бажає забронювати місце. Також в формі можуть бути додаткові поля для вказання додаткових побажань користувача.
3. Підтвердження бронювання. Після того, як користувач заповнив форму бронювання, система перевіряє наявність вільного місця на вказаний час та відправляє запит на підтвердження бронювання відповідному адміністратору коворкінгу. Якщо місце вільне, адміністратор підтверджує бронювання та повідомляє користувача про це.
4. Оплата бронювання. Після підтвердження бронювання користувач має здійснити оплату за заброньоване місце. Він може використати онлайн-платіжну систему, що підключена до даної інформаційної системи.
5. Використання забронюваного місця. Після оплати користувач може використовувати заброньоване місце на вказаний час та дату. Інформація про заброньоване місце відображається у його особистому кабінеті в системі.

І наостанок розглянемо процес «Керування користувачами в системі». Процес керування користувачами системи полягає в наданні адміністраторам можливості керувати правами доступу користувачів до функціональних можливостей системи.

Для цього адміністратор повинен мати право доступу до панелі управління користувачами системи, де можливо виконувати наступні дії:

1. Створення користувачів. Адміністратор може створювати нових користувачів системи шляхом введення їхніх особистих даних (ім'я, прізвище, контактна інформація тощо). Після цього користувач отримує доступ до особистого кабінету в системі, де можливо здійснювати бронювання місць в коворкінгу.
2. Редагування користувачів. Адміністратор може змінювати особисті дані користувачів, а також їх права доступу до функціональних можливостей

системи. Наприклад, адміністратор може обмежити доступ користувача до функції редагування місць в коворкінгу.

3. Видалення користувачів. Адміністратор може видаляти користувачів з системи, якщо вони більше не є клієнтами коворкінгу. При видаленні користувача з системи, його особисті дані також будуть видалені.
4. Керування правами доступу. Адміністратор може керувати правами доступу користувачів до функціональних можливостей системи. Наприклад, адміністратор може обмежити доступ користувача до функції редагування місць в коворкінгу, або надати йому права доступу до функції створення нових місць.

В цілому, процес керування користувачами системи полягає в тому, щоб забезпечити адміністраторам повний контроль над правами доступу користувачів.

2.2 Огляд можливих представлень інформаційної системи

Тепер ми перейдемо до огляду можливих представлень майбутньої інформаційної системи.

Опишемо можливість представлення інформаційної системи обліку оренди місць у коворкінгу у вигляді web-застосунку.

Web-застосунок - це програмне забезпечення, яке забезпечує доступ до інформаційної системи через браузер. Користувач може використовувати цей застосунок на будь-якому пристрої, який має підключення до Інтернету, без необхідності установки додаткового програмного забезпечення [14].

Потенційні переваги представлення системи у вигляді web-застосунку можуть включати:

1. Доступність. Web-застосунки доступні з будь-якого пристрою, який має доступ до Інтернету, що дозволяє користувачам звертатися до системи з будь-якого місця, незалежно від місця розташування сервера.

2. Зручність. Користувач може використовувати застосунок звичайним браузером, що робить його користування більш зручним та інтуїтивно зрозумілим.
3. Безпека. Застосунок може забезпечити зручний та безпечний доступ до системи з використанням протоколу HTTPS, який забезпечує шифрування даних під час передачі через мережу Інтернет [13].
4. Оновлення. Застосунок може бути оновлюваний на сервері, що дозволяє користувачам отримувати доступ до нових функцій та покращень без необхідності встановлення оновлень на своєму пристрої.

Недоліками такого представлення є:

1. Залежність від Інтернету: web-додатки вимагають наявності Інтернет-підключення, що може бути недоступним в деяких місцях або в певний час.
2. Незабезпеченість безпеки: web-додатки можуть стати мішенню для хакерських атак, тому необхідно добре захистити систему від таких небезпек.
3. Проблеми з продуктивністю: при роботі з великою кількістю даних, web-додатки можуть сповільнюватись, що може вплинути на продуктивність користувачів [19].

Тепер розглянемо інформаційну систему через призму telegram-боту. Застосування Telegram-боту для інформаційної системи обліку оренди місць у коворкінгу може мати наступні переваги та недоліки:

Перевагами такого підходу є:

1. Зручність користування. Бот дозволяє користувачам зручно та швидко взаємодіяти з системою, не виходячи з Telegram-месенджера.
2. Широкий охоп користувачів. Telegram-месенджер є популярним серед користувачів у різних країнах світу, тому застосування бота може дозволити залучити більше клієнтів до коворкінгу.

3. Більш висока відкритість. Telegram-боти зазвичай знаходяться у відкритому доступі, що дозволяє легше залучати нових користувачів та взаємодіяти з ними [25].

Недоліками такого представлення є:

1. Обмежені можливості. Боти зазвичай мають обмежені функціональні можливості порівняно з веб-застосунками, тому деякі функції можуть бути важкі або неможливі для реалізації в такому середовищі.
2. Обмежені можливості візуалізації. У порівнянні з веб-застосунками, Telegram-боти мають обмежені можливості візуалізації даних та інтерфейсу.
3. Залежність від Telegram. Використання Telegram-бота залежить від доступності месенджера та його функціонування.

У випадку з інформаційною системою обліку оренди місць у коворкінгу, Telegram-бот може мати вигоди, оскільки дозволить користувачам швидко та зручно бронювати місця та отримувати інформацію про доступність місць у режимі реального часу. Однак, для повного функціонального покриття, можливо знадобиться розробити додатковий сервіс за межами Telegram.

Останнім ми розглянемо представлення інформаційної системи через призму розробки мобільного додатку.

Розглянувши інформаційну систему обліку оренди місць у коворкінгу, можна сказати, що мобільний додаток має такі переваги порівняно з веб-застосунком або telegram-ботом:

1. Зручність використання: мобільні додатки розроблені спеціально для користувачів мобільних пристроїв, що забезпечує зручний інтерфейс та швидкий доступ до функціоналу системи [20].
2. Можливість роботи в офлайні: мобільний додаток може зберігати дані на пристрої користувача, що дозволяє працювати з системою навіть без доступу до Інтернету.

3. Більші можливості: мобільний додаток може використовувати додаткові можливості мобільного пристрою, такі як GPS-локація, камера та інші.
4. Більш висока безпека: мобільні додатки можуть бути забезпечені більш високим рівнем захисту, наприклад, за допомогою використання відбитків пальців або розпізнавання обличчя для авторизації.

Водночас, є також і недоліки такого підходу.

1. Обмежені можливості в порівнянні з веб-застосунком. Мобільні додатки, як правило, не мають такого ж широкого функціоналу, як веб-застосунки. Це може обмежувати можливості користувачів, які потребують виконання певних задач, що вимагають більшого доступу до даних.
2. Відсутність підтримки деяких платформ. Мобільний додаток може бути розроблений для певної платформи, наприклад, iOS або Android, і не підтримувати інші платформи. Це може становити проблему для користувачів, які використовують інші платформи.
3. Проблеми з безпекою. Мобільні додатки можуть мати проблеми з безпекою, такі як можливість злому інформації, що зберігається на пристрої користувача, або підвищення ризику викрадення особистої інформації.
4. Обмежена зручність введення даних. На мобільних пристроях може бути обмежена зручність введення даних, зокрема, через обмежений розмір екрану і обмежені можливості введення тексту.
5. Обмежена масштабованість. Мобільний додаток може мати обмежену масштабованість, особливо якщо в ньому використовується база даних або інші ресурси, що потребують більшої потужності обчислювального обладнання.

Отже, розглянувши різні варіанти представлення інформаційної системи, ми дійшли до висновку, що краще за все буде розробити telegram-бот. Telegram є одним з найпопулярніших месенджерів не лише в Україні, а і

також у світі. Робота над візуальною складовою є мінімальною та пов’язана з розташуванням кнопок або відповідями на певні команди. Для взаємодії з такою інформаційною системою потрібен лише встановлений месенджер та працюючий сервер для бота.

2.2 Реєстрація бота в системі

Перед реалізацією програмного бота у месенджері Telegram, необхідно зареєструвати його. Цей процес досить простий та автоматизований. У Telegram є окремий акаунт з назвою @BotFather, який допоможе вам у створенні власного бота. На зображенні 2.6 показано початок роботи з цим ботом [16].

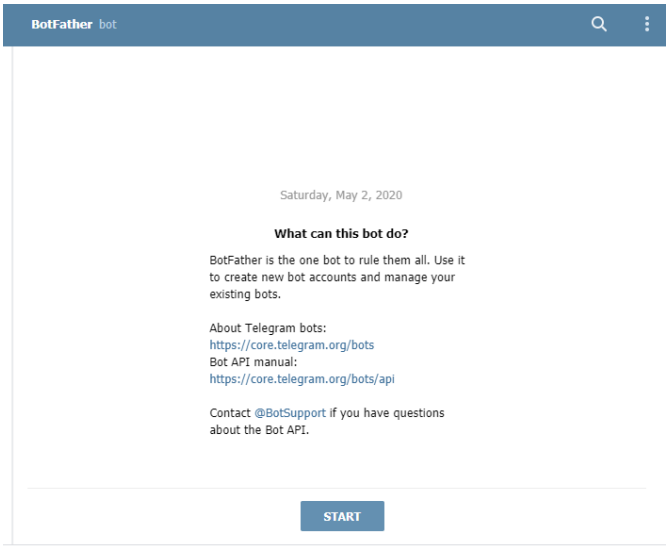

Рисунок 2.1 - Початок роботи з BotFather

Після того, як ви ознайомилися зі списком базових команд для роботи з @BotFather, необхідно створити нового бота, натиснувши команду /newbot (рисунок 2.2). Після цього, ви можете вибрати ім'я для вашого бота та його юзернейм, який повинен закінчуватися на "bot".

Наприклад, для даного проекту було обрано назву "Booking the Place".

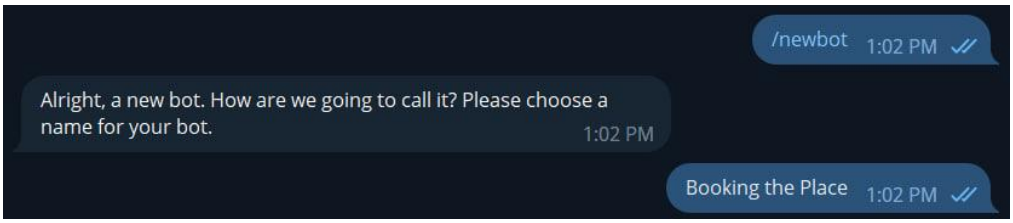

Рисунок 2.2 - Створення нового бота

Отримуєте унікальний токен (рисунок 2.3), який є цифровим ключем для доступу до HTTP API. Токен відрізняється від звичайного пароля тим, що постійно оновлюється та є відомим лише користувачу в момент ідентифікації. HTTP є гіпертекстовим протоколом передачі даних прикладного рівня моделі OSI, а API - це набір інтерфейсів, що надаються додатком або сервісом для використання в зовнішніх програмних продуктах [18].

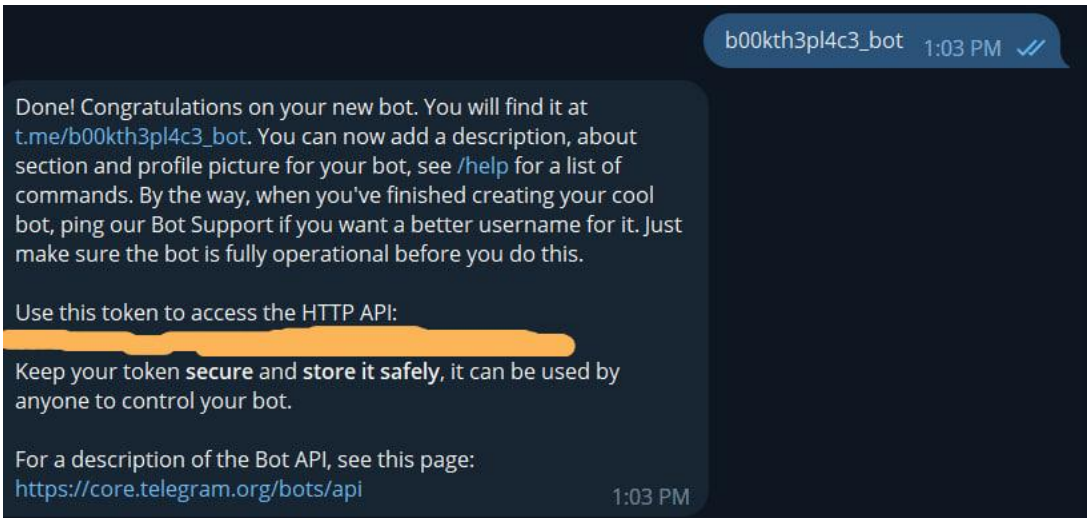

Рисунок 2.3 – отримання токену для взаємодії з Telegram API

За бажанням також можна встановити фотографію telegram-бота, його опис, встановити базові команди, проте в даному проекті це не є доцільним, адже нам потрібно розробити працюючий MVP (minimum viable product, мінімально життєздатний продукт) [11].

3 КОМП'ЮТЕРНА РЕАЛІЗАЦІЯ ПРОЕКТУ ТА ТЕСТУВАННЯ

3.1 Вибір середовища розробки

Для реалізації telegram-боту обліку оренди місць у коворкінгу потрібно обрати основні інструменти. Можна його розробити за допомогою готових рішень на кшталт Node-RED або Chatfuel, можна використати мову програмування та наявні в ній бібліотеки. Так як готові рішення є платними, не надають достатньої гнучкості для масштабування та мають обмежений функціонал для розробки кінцевого продукту, відразу перейдемо до розгляду мов програмування, які допоможуть реалізувати проект.

Існує декілька мов програмування, які можуть бути використані для розробки Telegram-ботів. Ось декілька з них:

1. Python: Python є дуже популярною мовою програмування для розробки Telegram-ботів. У Telegram існує офіційна бібліотека python-telegram-bot, яка містить всі необхідні інструменти для створення бота на Python.
2. Node.js: Node.js є популярною платформою для розробки веб-додатків і додатків на основі JavaScript. Для розробки Telegram-ботів на Node.js можна використовувати бібліотеку Telegraf.
3. Java: Java є однією з найпопулярніших мов програмування в світі. Для розробки Telegram-ботів на Java можна використовувати бібліотеку TelegramBots.

Ці мови програмування не є єдиними варіантами для розробки Telegram-ботів, але вони є досить популярними та мають багато готових бібліотек і інструментів для роботи з Telegram API.

Розглянемо кожну з них окремо та почнемо з Python. Python є однією з найпопулярніших мов програмування для розробки telegram-ботів, і є досить ефективним інструментом для цієї задачі. Ось деякі переваги та недоліки використання Python для розробки telegram-ботів:

Переваги:

- Має простий і зрозумілий синтаксис, що дозволяє розробникам створювати програми швидко і без особливих зусиль.
- Є багато бібліотек та фреймворків для розробки ботів на Python, таких як python-telegram-bot, Telebot, Pyrogram тощо [13].
- Має велику та активну спільноту, яка надає допомогу, якщо виникають якісь проблеми або питання.
- Python є мультиплатформеним, тобто може працювати на різних операційних системах, включаючи Windows, Mac та Linux.

Недоліками даної мови є:

- Може бути повільнішим порівняно з іншими мовами програмування, такими як Java або C++. Однак, це не є серйозною проблемою для розробки телеграм-ботів [7].
- Має деякі обмеження в обробці багатопоточних додатків, що може бути недоліком для деяких великих ботів з високим навантаженням.
- Деякі функції, які можуть бути доступні в інших мовах програмування, можуть бути не так просто реалізовані в Python [6].

У загальному, Python є хорошим вибором для розробки telegram-ботів, завдяки своїй простоті, швидкості розробки та підтримці спільнотою.

Також розглянемо Node.js - серверна платформа для JavaScript, яка дозволяє виконувати код JavaScript на стороні сервера. Node.js зазвичай використовується для створення швидких та масштабованих додатків з великою кількістю одночасних підключень, що робить його гарним вибором для розробки Telegram-ботів. Ось деякі переваги та недоліки використання Node.js для розробки Telegram-ботів:

Переваги:

- Швидкість виконання: Node.js виконує код JavaScript на стороні сервера, що дозволяє використовувати один мову програмування для якнайшвидшого виконання завдань;

- Бібліотеки: Node.js має багато корисних бібліотек, які допомагають в розробці Telegram-ботів, наприклад, "node-telegram-bot-api" або "telegraf" [22];
- Асинхронний код: Node.js підтримує асинхронний код, що дозволяє виконувати декілька завдань одночасно, що зменшує час очікування на відповідь.

Недоліки:

- Складність: Навіть досвідченим розробникам може бути важко зрозуміти асинхронний код, який використовується в Node.js;
- Система зборки: Відсутність стандартної системи зборки може бути складнішою для новачків;
- Потребує ресурсів: Node.js використовує пам'ять більше, ніж інші мови програмування, що може бути проблемою для менш потужних серверів.

В цілому, Node.js є гарним вибором для розробки Telegram-ботів, особливо якщо вам важливо максимально скоротити час відповіді та оптимізувати час виконання.

Останньою мовою програмування, яку ми розглянемо, є Java. Java є однією з популярних мов програмування для розробки Telegram-ботів. Розглянемо деякі її переваги та недоліки:

Переваги:

- Велика кількість бібліотек та фреймворків, що полегшують розробку та роботу з Telegram API.
- Java має сильну типізацію, що дозволяє розробникам виявляти помилки на ранніх етапах та запобігати їх поширенню.
- Мова програмування Java є кросплатформеною та може бути використана на будь-якій операційній системі, що дозволяє створювати ботів, які можуть працювати на різних платформах [19].
- Java має велику та активну спільноту розробників, що дає можливість швидко отримати відповіді на запитання та розв'язати проблеми.

Недоліки:

- При використанні Java для розробки Telegram-бота може виникнути додатковий навчальний поріг для розробників, оскільки мова має складну синтаксичну структуру.
- Через велику кількість функцій та високий рівень абстракції Java може бути важко відлагоджувати код.
- В порівнянні з іншими мовами програмування, такими як Python або JavaScript, використання Java може бути більш витратним за часом на написання коду [24].

В цілому, Java може бути хорошим вибором для розробки Telegram-бота з великою кількістю функціоналу та вимогами до безпеки.

Проаналізувавши дані мови програмування, було прийнято рішення щодо використання Python. Його синтаксична простота, велика підтримка серед спільноти, наявність великої кількості інструментів нівелюють слабкі сторони, які було виявлено під час аналізу.

Разом з Python було обрано такі інструменти для реалізації цілі:

1. Бібліотека Aiogram - це асинхронний фреймворк для розробки ботів в Telegram на мові Python. Вона надає простий і зрозумілий інтерфейс для взаємодії з Telegram Bot API і дозволяє швидко створювати складні боти з використанням різноманітних функцій, таких як inline-режим, робота з базою даних та інші. Aiogram також підтримує різні методи аутентифікації та може бути легко інтегрована з іншими бібліотеками для розробки ботів. Вона має велику кількість корисних функцій та зручну документацію, що робить її популярним вибором серед розробників Telegram-ботів на Python.
2. SQLAlchemy - це бібліотека для роботи з базами даних у Python. Вона дозволяє зручно взаємодіяти з різними СУБД, такими як PostgreSQL, MySQL, SQLite, та інші. SQLAlchemy надає об'єктно-орієнтований підхід до роботи з базами даних та можливість використовувати мову

SQL для роботи з даними. SQLAlchemy також має розширення для роботи з асинхронними запитами, що є важливим для розробки додатків, які взаємодіють зі зовнішніми сервісами. За допомогою SQLAlchemy можна виконувати різноманітні запити до баз даних, включаючи вставку, оновлення, видалення та вибірку даних, а також виконувати транзакції.

3. SQLite - це легка вбудовувана реляційна система управління базами даних, яка зберігає дані в одному файлі на диску. Вона працює на різних операційних системах, включаючи Windows, Linux та macOS, і є досить популярною в веб-розробці, особливо в проектах з невеликим обсягом даних. SQLite підтримує майже всі стандартні операції SQL, включаючи JOIN, WHERE, GROUP BY, ORDER BY та інші. Вона має простий синтаксис та добре документована, що робить її дуже простою в використанні. SQLite зазвичай використовують для зберігання даних на клієнтській стороні або як тимчасову базу даних на сервері. Вона є популярним вибором для мобільних додатків та різноманітних десктопних програм, що працюють зі збереженням даних.

3.2 Проектування бази даних

Після визначення зовнішньої та внутрішньої структури telegram-боту, потрібно було вирішити, як зберігати інформацію про користувачів, доступні місця у коворкінгу та їх бронювання. Для коректної роботи telegram-боту необхідно було спроектувати базу даних та підключити її до боту. База даних telegram-бота «Booking the place» містить таблиці: users, places, usersplaces.

На рисунку 3.1 зображена структура бази даних telegram-боту.

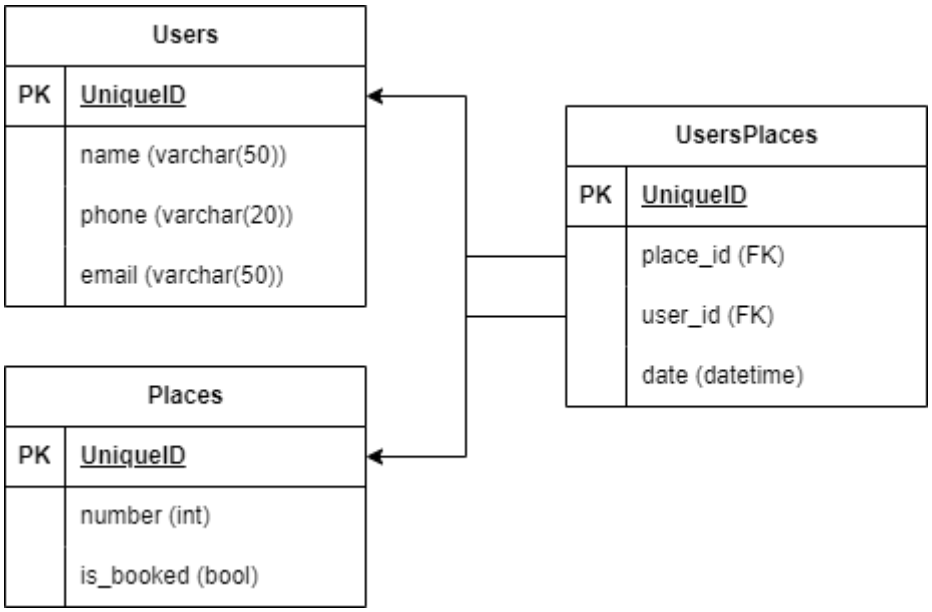

Рисунок 3.1 - ERD-діаграма бази даних telegram-бота

Розглянемо детальніше дану структуру:

1. Таблиця users. Вона містить у собі такі компоненти:
    1.1. Name – ім’я того, хто бронюватиме місце у коворкінгу.
    1.2. Phone – номер мобільного телефону, за яким можна буде зв’язатися з особою у разі потреби.
    1.3. Email – електронна пошта особи, за якою можна буде надсилати підтвердження щодо бронювання місця, його скасування, акційні пропозиції.
2. Таблиця places. Вона містить у собі такі компоненти:
    2.1. Number – унікальний номер місця, яке присутнє у коворкінгу.
    2.2. Is_booked – флаг того, чи заброньоване місце, чи ні.
3. Таблиця usersplaces. Створена для many-to-many зв’язку між users та places, призначена для того щоб вести облік орендованих місць. Містить у собі такі компоненти:
    3.1. Place_id – зовнішній ключ, за яким доступна інформація про заброньоване місце.

3.2. User_id – зовнішній ключ, за яким доступний користувач, що забронював місце.

3.3. Date – дата, на коли саме було забронювано місце у коворкінгу.

Для забезпечення ефективної взаємодії між telegram-ботом та базою даних буде використовуватись ORM SQLAlchemy, що дозволить використовувати знайомий синтаксис Python для роботи з базою даних [17].

3.2 Програмна реалізація бота

Для програмної реалізації інформаційної системи у вигляді telegram-бота, потрібно спроектувати її структуру. Виглядатиме вона наступним чином:

- Bot – фундамент інформаційної системи, що буде поділений на так звані «chains» (або ж «ланцюги») та «middlewares» (або ж «проміжні засоби». За допомогою «Ланцюгів» ми матимемо можливість розбити великий проект на декілька модулів, що відповідатимуть за різні функціональні вимоги. Завдяки такому кроку, кожен наступний розробник без проблем зможе ознайомитися з окремими функціями продукту та у випадку потреби зайнятися їх масштабуванням.
- В ІС «Booking the place» буде представлено такі chains, як registration_user (сервіс регістрації користувача), book_place (сервіс оренди місця), cancel_place (сервіс скасування оренди місця), first_touch (сервіс, який викликається під час першого контакту з telegram-ботом). Також для кожного з ланцюгів буде розроблена унікальна клавіатура, яка зробить зручною взаємодію користувача з telegram-ботом.
- Проміжний засіб unthrottling забезпечуватиме захист від спаму.
- Файл конфігурації bot_core.py, який викликається при запуску бота на сервері та виконує підключення до бази даних, встановлює зв’язок з Telegram API і підключає chains.

- Bot_database – директорія, яка буде містити вихідний код для таблиць БД у вигляді ORM-моделей та певні методи, за допомогою яких відбуватиметься комунікація між telegram-ботом та базою даних.

Для початку розробки, ми створюємо файл .env у директорії проекту, де будуть зберігатися наступні дані: токен бота, user id адміністратора та параметри бази даних (хост, порт, назва бази даних, логін, пароль). Файл .env має таку структуру:

1. Telegram bot token – токен для взаємодії з Telegram API.
2. Database host – адреса бази даних, з якою взаємодіятиме бот.
3. Database port – порт бази даних.
4. Database name – ім’я бази даних, з якою треба буде взаємодіяти ІС.
5. Database username – логін для аутентифікації до бази даних.
6. Database password – пароль для аутентифікації до бази даних.

Отже, спроектувавши основи ІС та обравши усі потрібні інструменти для розробки, ми можемо розробити telegram-бота. Вихідний код буде представлено у додатках до диплому.

3.4 Тестування Telegram боту

Після процесу реалізації ІС для оренди місць у коворкінгу у вигляді telegram-бота ми провели тестування продукту сторонніми користувачами. Під час тестування не було виявлено серйозних проблем.

На рисунку 3.5 представлено повідомлення після надсилання команди /start та процес реєстрації користувача.

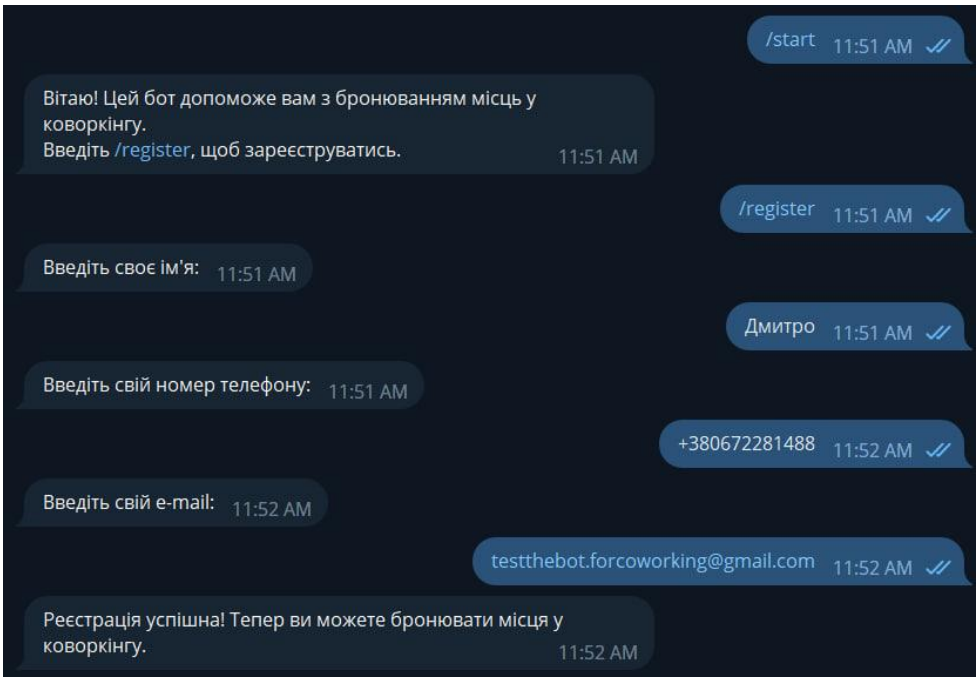

Рисунок 3.5 – Початок роботи із ботом та процес реєстрації користувача

Як ми бачимо, реєстрація відбувається у вигляді живого спілкування, тобто telegram-бот просить користувача надати ім’я, номер телефону та e-mail.

На рисунку 3.6 представлено процес бронювання місця, яке також відбувається у вигляді такого собі спілкування із адміністратором.

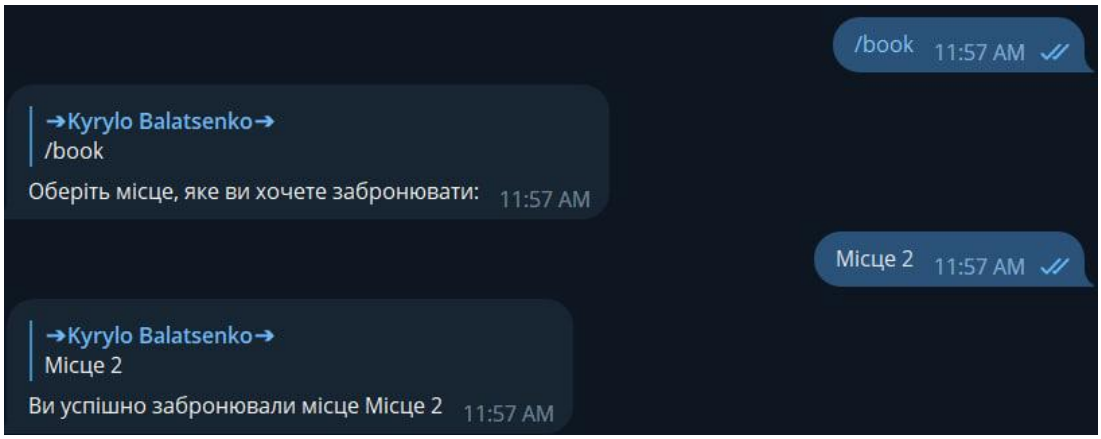

Рисунок 3.6 – процес бронювання місця.

Тобто, як ми бачимо, ці процеси відбуваються доволі швидко та лише при наявності месенджера Telegram. У перспективі є такі шляхи:

1. Розробити веб-застосунок для адміністрування таких запитів на бронювання.

2. У веб-застосунку створити функціонал для ведення статистики щодо бронювання місць (скільки було заброньовано, коли, в який час, хто найчастіше бронює місце тощо).
3. Використати асинхронні підходи для оптимізації роботи бота при високих навантаженнях за допомогою бібліотеки aiogram.
4. Дану ІС контейнеризувати за допомогою Docker задля швидкого розгортання іншими власниками коворкінгу.
5. Мігрувати з СУБД SQLite до СУБД PostgreSQL.

ВИСНОВКИ

У процесі виконання завдання були вирішені наступні завдання:

- Вивчено та проаналізовано сучасні мови програмування, які можна використовувати для створення Telegram-бота.
- Проведено аналіз тематичної літератури та огляд існуючих рішень, пов'язаних з розробкою Telegram-ботів.
- Створено та налаштовано роботу Telegram-бота з урахуванням вимог та функціональності.
- Проведено тестування, щоб переконатися у працездатності бота та відповідності його функціональності вимогам.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв’язанню важливої практичної задачі системи обліку оренди місць у колективному офісі, шляхом розробки відповідних методів, моделей та інформаційних технологій.

Результатом цієї роботи є розроблена інформаційна систему з оренди місць у колективному офісі для месенджера Telegram-бот під назвою "Booking the place", який має зручний інтерфейс, процеси відбуваються у вигляді живого спілкування та дозволяє швидко та зручно бронювати місця у коворкінгу, оптимізуючи процес бронювання та економлячи час як для користувачів, так і для власників коворкінгу.

Розроблений чат-бот має декілька особливостей, які роблять його зручним та привабливим для користувачів:

- Інтуїтивно зрозумілий та зручний інтерфейс, що дозволяє легко орієнтуватися та взаємодіяти з ботом.
- Наявність меню з кнопками, що спрощує вибір опцій та команд, замість необхідності вводити текстові команди.
- Бот доступний для використання цілодобово, що дає можливість користувачам бронювати місця у коворкінгу в зручний для них час.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. BotFather Documentation. URL: https://core.telegram.org/bots#botfather (дата звернення: 15.05.2023).
2. Aiogram Documentation. URL: https://docs.aiogram.dev/ (дата звернення: 05.05.2023).
3. SQL Alchemy Documentation. URL: https://docs.sqlalchemy.org/ (дата звернення: 02.06.2023).
4. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 12.05.2023).
5. Python Programming Language. URL: https://www.python.org/ (дата звернення: 28.05.2023).
6. Node.js Documentation. URL: https://nodejs.org/en/docs/ (дата звернення: 20.05.2023).
7. JavaScript Programming Language. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 08.05.2023).
8. Java Programming Language. URL: https://docs.oracle.com/en/java/ (дата звернення: 23.05.2023).
9. Telegram Bot API Documentation. URL: https://core.telegram.org/bots/api (дата звернення: 07.05.2023).
10. Java Telegram Bot Library. URL: https://github.com/pengrad/java-telegram-bot-api (дата звернення: 04.05.2023).
11. Node.js Telegram Bot Library. URL: https://github.com/yagop/node-telegram-bot-api (дата звернення: 17.05.2023).
12. Official SQLite Website. URL: https://www.sqlite.org/ (дата звернення: 10.06.2023).
13. SQLAlchemy ORM Documentation. URL: https://docs.sqlalchemy.org/ (дата звернення: 01.06.2023).

14. SQLite Database Programming with Python and SQLite. URL: https://www.sqlitetutorial.net/sqlite-python/ (дата звернення: 14.05.2023).
15. Telegram Bot Development Cookbook. URL: https://leanpub.com/telegram-bot-development-cookbook (дата звернення: 06.05.2023).
16. «Як створити чат-бот для Telegram-каналу – інструкція для адміністраторів»: стаття від Netpeak. URL: https://bit.ly/43AxbPs.
17. pyTelegramBotAPI Documentation. URL: https://pyrogram.org/ (дата звернення: 25.05.2023).
18. python-telegram-bot Documentation. URL: https://python-telegram-bot.readthedocs.io/ (дата звернення: 02.06.2023).
19. Telegram Bot API Wrapper for Python Documentation. URL: https://github.com/python-telegram-bot/python-telegram-bot (дата звернення: 08.05.2023).
20. Telegram Bot API Python SDK. URL: https://github.com/eternnoir/pyTelegramBotAPI (дата звернення: 15.05.2023).
21. How to Build a Telegram Bot Using Python and Django. URL: https://studygyaan.com/django/how-to-build-telegram-bot-using-python-django (дата звернення: 19.05.2023).
22. Telebot: Python Telegram Bot API Wrapper. URL: https://github.com/eternnoir/pyTelegramBotAPI (дата звернення: 28.05.2023).
23. Creating a Telegram Bot Using Python (YouTube tutorial by Corey Schafer). URL: https://www.youtube.com/watch?v=ny2vD9JBOHY (дата звернення: 06.05.2023).
24. Telegram Bot Development with Python and Django. URL: https://simpleisbetterthancomplex.com/tutorial/2017/03/19/how-to-build-a-grantly-bot-with-telegram-using-django.html (дата звернення: 12.05.2023).

25. How to Create a Telegram Bot Using Python and the Telegram Bot API. URL: https://www.section.io/engineering-education/how-to-create-a-telegram-bot-using-python/ (дата звернення: 23.05.2023).

ДОДАТОК

```
import logging
from aiogram import Bot, Dispatcher, types

# Ініціалізація бота
bot = Bot(token="YOUR_TOKEN")
dp = Dispatcher(bot)

# Налаштування рівня журналювання
logging.basicConfig(level=logging.INFO)

# Словник для зберігання зареєстрованих користувачів
registered_users = {}

# Команда /start для реєстрації користувача
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    chat_id = message.chat.id
    if chat_id not in registered_users:
        # Отримання даних від користувача
        await message.reply("Для реєстрації, будь ласка, надайте наступну
інформацію:\n"
                            "Ім'я:")
        # Встановлення стану реєстрації
        dp.register_next_step_handler(message, process_name_step)
    else:
```

```
        await message.reply("Ви вже зареєстровані!")

# Обробка кроку реєстрації імені
async def process_name_step(message: types.Message):
    chat_id = message.chat.id
    name = message.text
    # Збереження імені до словника
    registered_users[chat_id] = {"name": name}
    await message.reply("Номер телефону:")

    # Встановлення наступного кроку реєстрації
    dp.register_next_step_handler(message, process_phone_step)

# Обробка кроку реєстрації номера телефону
async def process_phone_step(message: types.Message):
    chat_id = message.chat.id
    phone = message.text
    # Збереження номера телефону до словника
    registered_users[chat_id]["phone"] = phone
    await message.reply("E-mail:")

    # Встановлення наступного кроку реєстрації
    dp.register_next_step_handler(message, process_email_step)

# Обробка кроку реєстрації e-mail
async def process_email_step(message: types.Message):
    chat_id = message.chat.id
```

```
    email = message.text
    # Збереження e-mail до словника
    registered_users[chat_id]["email"] = email

    await message.reply("Ви успішно зареєстровані!")

# Команда для бронювання місця
@dp.message_handler(commands=['book'])
async def book_place(message: types.Message):
    chat_id = message.chat.id
    if chat_id in registered_users:
        # Отримання дати від користувача
        await message.reply("Введіть дату бронювання (у форматі ДД.ММ.РРРР):")
        # Встановлення стану бронювання
        dp.register_next_step_handler(message, process_booking_date_step)
    else:
        await message.reply("Ви не зареєстровані. Будь ласка, скористайтеся командою /start для реєстрації.")

# Обробка кроку бронювання дати
async def process_booking_date_step(message: types.Message):
    chat_id = message.chat.id
    booking_date = message.text
    # Тут ви можете додати логіку для перевірки та збереження бронювання
```

```
    await message.reply(f"Місце було успішно заброньовано на
{booking_date}!")

# Запуск бота
if __name__ == '__main__':
    from aiogram import executor
    executor.start_polling(dp, skip_updates=True)
```

Моделі бази даних в SQLAlchemy

```
from sqlalchemy import create_engine, Column, Integer, String, Boolean,
ForeignKey, Date
from sqlalchemy.orm import relationship, sessionmaker
from sqlalchemy.ext.declarative import declarative_base

# підключення до бази даних SQLite
engine = create_engine('sqlite:///coworking.db')
Session = sessionmaker(bind=engine)
session = Session()

# оголошення базового класу моделі
Base = declarative_base()

# клас моделі для таблиці "users"
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    name = Column(String(50))
```

```
    phone = Column(String(20))
    email = Column(String(50))

    # зв'язок один-багато з таблицею "users_places"
    places = relationship('UserPlace', back_populates='user')

# клас моделі для таблиці "places"
class Place(Base):
    __tablename__ = 'places'
    id = Column(Integer, primary_key=True)
    number = Column(Integer)
    is_booked = Column(Boolean)

    # зв'язок один-багато з таблицею "users_places"
    bookings = relationship('UserPlace', back_populates='place')

# клас моделі для таблиці "users_places"
class UserPlace(Base):
    __tablename__ = 'users_places'
    id = Column(Integer, primary_key=True)
    place_id = Column(Integer, ForeignKey('places.id'))
    user_id = Column(Integer, ForeignKey('users.id'))
    date = Column(Date)

    # зв'язок багато-один з таблицею "users"
    user = relationship('User', back_populates='places')
```

```
    # зв'язок багато-один з таблицею "places"
    place = relationship('Place', back_populates='bookings')

Session = sessionmaker(bind=engine)
session = Session()
```