ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка
до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ КАЛЕНДАРНОГО ПЛАНУВАННЯ ОСОБИСТИХ ЗАДАЧ МОВОЮ C# НА БАЗІ ПЛАТФОРМИ ASP.NET**»

Виконав: студент 4-го курсу, групи ПД-41
спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності/спеціалізації)

Скидан П. В.

(прізвище та ініціали)

Керівник Негоденко О.В.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Київ - 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
Негоденко О.В.
“ ” 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

СКИДАНА ПАВЛА ВАЛЕРІЙОВИЧА
(прізвище, ім’я, по батькові)

1. Тема роботи: “Розробка програмних засобів для календарного планування особистих задач мовою C# на базі платформи ASP.NET”

Керівник роботи: Негоденко О.В. к.т.н., доц.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від “24” лютого 2023 року № 26.

2. Строк подання студентом роботи “01” червня 2023 року
3. Вхідні дані до роботи:

Методи обробки та зберігання даних:

Інструменти з розробки програмного забезпечення: ASP.NET Core 7.0, .NET Core 7.0, Entity Framework Core, IDE JetBrains Rider 2023.1, DMS JetBrains DataGrip 2023.1, Docker, Google Calendar API Library, PostgreSQL.

Науково-технічні документації та література по розробці мікро сервісної

архітектури:

Технічна документація Telegram API, Google Calendar API.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
   - 4.1. Аналіз предметної області
   - 4.2. Технічне завдання
   - 4.3. Функціональні вимоги
   - 4.4. Нефункціональні вимоги
   - 4.5. Асоціативна мапа
   - 4.6. Діаграма класі використання
   - 4.7. Інструменти та засоби розробки програмного забезпечення
   - 4.8. Архітектура системи
   - 4.9. Опис програмного забезпечення
5. Перелік демонстраційного матеріалу
   - 5.1. Титульний слайд.
   - 5.2. Мета, об’єкт, предмет, наукова новизна дослідження.
   - 5.3. Актуальність.
   - 5.4. Аналіз аналогів.
   - 5.5. Технічні завдання.
   - 5.6. Програмні засоби та інструменти реалізації.
   - 5.7. Розробка архітектури.
   - 5.8. Реалізація програми.
   - 5.9. Висновки
   - 5.10. Апробація результатів дослідження.
   - 5.11. Кінцевий слайд.
6. Дата видачі завдання «25» лютого 2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.04-14.04	Виконано
2	Аналіз та дослідження існуючих аналогів	15.04-17.04	Виконано
3	Проектування системи	18.04-21.04	Виконано
4	Створення та тестування програмного рішення	21.04-05.05	Виконано
5	Підготовка розділу 1	05.05-07.05	Виконано
6	Підготовка розділу 2	07.05-09.05	Виконано
7	Підготовка розділу 3	08.05-11.05	Виконано
9	Вступ, висновки, реферат	11.05-12.05	Виконано
10	Розробка обов‘язкових демонстраційних матеріалів	12.05-15.05	
11	Попередній захист роботи та перевірка на плагіат	19.05-25.05	
12	Здача роботи	01.06	

Студент _____ _____

(підпис) (прізвище та ініціали)

Керівник роботи _____ _____

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 43 с., 22 рис., 4 табл., 2 дод., 14 джерел.

Ключові слова: TELEGRAM, API, BOTFATHER, POSTGRESQL, ENTITY FRAMEWORK, C#, ASP.NET, .NET CORE, IDE.

Об'єкт дослідження – процес календарного планування особистих задач.

Предмет дослідження – програмне забезпечення календарного планування особистих задач.

Мета роботи – спрощення процесу календарного планування особистих задач за рахунок його реалізації всередині месенджеру Telegram мовою C#.

Для реалізації мети потрібно вирішити наступні питання:

1. Проаналізувати існуючі додатки та програмні засоби, які мають в собі схожий функціонал, та виявити їхні переваги та недоліки.
2. Сформувати функціональні та нефункціональні вимоги до застосунку на основі порівняльної характеристики аналогів.
3. Спроектувати архітектуру та розробити Telegram-бота відповідно до визначених потреб користувачів.
4. Провести функціональне тестування застосунку.
5. Пройти апробацію тез на науково-технічній конференції.

Практичне значення результату: розроблений бот буде використовуватися для спрощення процесу календарного планування задач із синхронізацією в Google Calendar.

Для розробки програмного забезпечення було використано мову програмування C#, фреймворк ASP.NET Core та Entity Framework.

Цей Telegram-бот націлений на використання звичайними користувачами для управління своїми задачами.

Галузь використання — керування задачами (англ. — task management)

ЗМІСТ

ВСТУП.....10
1. ТЕОРЕТИЧНА ЧАСТИНА.....11
1.1. Аналіз предметної галузі.....11
1.2. Google Tasks.....12
1.3. Todoist.....14
1.4. Trello.....16
1.5. Any.do.....18
1.6. Telegram-бот Ok, Bob!.....19
1.7. Таблиця порівняння аналогів.....20
2. ВИМОГИ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....21
2.1. Технічне завдання.....21
2.2. Функціональні вимоги.....21
2.2.1. Створення задачі.....21
2.2.2. Валідація введеного користувачем тексту.....22
2.2.3. Синхронізація створених задач із Google Calendar.....22
2.2.4. Відображення поточних задач.....22
2.2.5. Видалення задачі.....22
2.2.6. Відправлення нагадувань через сповіщення.....22
2.3. Нефункціональні вимоги.....23
2.3.1. Зрозумілість.....23
2.3.2. Надійність та стійкість.....23
2.3.3. Безпека.....23
2.3.4. Масштабованість.....23
2.4. Асоціативна мапа.....24
2.5. Діаграми використання.....25
3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....27
3.1. Засоби та інструменти розробки.....27

3.1.1. Мова C#.....27
3.1.3. ASP.NET Core.....28
3.1.4. Docker.....29
3.1.5. PostgreSQL.....31
3.1.6. Git.....32
3.1.7. JetBrains Rider.....32
3.1.8. JetBrains DataGrip.....33
3.2. Системна архітектура.....33
3.2.1. Взаємодія сервісу із інфраструктурою.....33
3.2.2. Взаємодія компонентів проекту.....36
3.2.3. Структура бази даних.....37
3.2.4. Команди та їх обробка.....38
3.3. Тестування програмного забезпечення.....40
4. ВИСНОВКИ.....42
5. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....43
ДОДАТОК А.....45
ДОДАТОК Б.....50

ВСТУП

Обґрунтування вибору теми та її актуальність. Сучасне становище речей вказує на те, які засоби комунікації між людьми в мережі та які інструменти по управлінню задачами, та організації часу в повсякденному житті є найпопулярнішими та найдієвішими. В рамках даної роботи розглянуто такі програмні засоби, як месенджер Telegram, та продукт від компанії Google - Google Calendar. Було обрано саме таку комбінацію, тому що наразі не існує інтеграції Google Calendar та Telegram-боту, яка б надавала функціонал створення та планування задач всередині популярного месенджеру та синхронізації із календарем системи Google, що використовується багатьма звичайними користувачами, не кажучи вже про компанії, які планують свої дедлайни, робочі зустрічі та інше за допомогою Google Calendar.

Об’єкт дослідження – процес календарного планування особистих задач.

Предмет дослідження – програмне забезпечення календарного планування особистих задач.

Мета роботи – спрощення процесу календарного планування особистих задач за рахунок його реалізації всередині месенджеру Telegram мовою C#.

Для реалізації мети потрібно вирішити наступні питання:

1. Проаналізувати існуючі додатки та програмні засоби, які мають в собі схожий функціонал, та виявити їхні переваги та недоліки.
2. Сформувати функціональні та нефункціональні вимоги до застосунку на основі порівняльної характеристики аналогів.
3. Спроектувати архітектуру та розробити Telegram-бота відповідно до визначених потреб користувачів.
4. Провести функціональне тестування застосунку.
5. Пройти апробацію тез на науково-технічній конференції.

Практичне значення результату: розроблений Telegram-бот спрощує процес календарного планування задач із синхронізацією в Google Calendar.

Галузь використання — керування задачами.

1. ТЕОРЕТИЧНА ЧАСТИНА

1.1. Аналіз предметної галузі

В житті людини час є одним із найцінніших ресурсів. Реалізація певних цілей, виконання задач та повсякденне життя вимагають правильного розподілу цього ресурсу. Варто зазначити, що цифрове наповнення світу на сьогоднішній день дозволяє використовувати технічні засоби для допомоги в організації та плануванні часу й задач.

Особливості використання вже існуючих засобів полягають в тому, що стає можливим використання переваг кожного з них. Тобто ті інструменти та технології, на основі яких працює вже створене програмне забезпечення, стають в нагоді та допомагають у вирішенні проблеми менеджменту часу й контролю за виконанням своїх задач.

Одним із найпопулярніших цифрових засобів комунікації в нашому суспільстві є месенджер Telegram. Він налічує понад 700 млн активних користувачів у всьому світі[1]. В якості основного комунікаційного засобу буде використано саме це програмне забезпечення, адже дуже велика кількість людей проводить частина їхнього часу в месенджері Telegram кожного дня.

Наступний програмний засіб, який буде використано для організації часу та контролю завдань, це Google Calendar. Цей сервіс є також надзвичайно популярним як серед окремих, тобто звичайних користувачів, так і серед корпоративних рішень в організації роботи, плануванні та менеджменту.

1.2. Цільова аудиторія та вимоги

Основна цільова аудиторія цього програмного забезпечення, а саме Telegram-боту, є звичайний користувач месенджеру Telegram, який також може мати обліковий запис в Google та використовувати Google Calendar для організації своєї активності. Такі засоби дозволяють сформувати необхідні вимоги щодо того

програмного забезпечення, яке буде розроблено:

1. Створення та планування задач, дедлайнів та зустрічей.

2. Відстеження та контроль виконання завдань та цілей.

3. Організація роботи в Google Calendar.

4. Отримування нагадувань та сповіщень про найближчі події, задачі, які очікують виконання.

5. Синхронізація даних між різними пристроями та платформами.

6. Інтеграція з Telegram.

Найбільшим обмеженням в цій сфері може бути вид зайнятості людини, що може полягати в щоденній роботі, часі, який треба приділити власним потребам тощо.

Оскільки менеджмент часу дозволяє встановити певний режим дня, впорядкувати свої задачі для найкращої продуктивності, а також сконцентруватися на чомусь конкретному, що є надзвичайно важливим та необхідним в сучасному світі[2], то такий інструмент, як Telegram-бот дозволить набагато продуктивніше організовувати свою роботу за рахунок того, що однією з найвагоміших причин втрати людиною дорогоцінного часу є якраз перебування онлайн в месенджерах, де буде знаходитись цей засіб.

1.3. Google Tasks

Цей додаток є безкоштовним інструментом для управління задачами, що дозволяє користувачеві створювати, організовувати та відслідковувати свої завдання та цілі, які синхронізуються між пристроями.

Основі функції, які має додаток:

1. Створення задач і списків задач;

2. Гнучке налаштування термінів виконання завдання та встановлення нагадувань;

3. Збереження історії тих завдань, які були виконані або видалені/скасовані;

4. Зручна синхронізація даних серед пристроїв користувача та інтеграція із власними продуктами.

Дуже зручно те, що надано можливість створювати багато дощок, де створюються задачі під різні проекти, сфери життя, часові проміжки, типи активності встановлених цілей чи досягнень. Функціонал розподілу задач, тобто задач, на ще менші переліки, дозволяє виокремлювати ще більш деталізовані області для ефективного виконання запланованих дій. Користувачу також запропоновано перегляд виконаних задач для аналізу та відстеження свого прогресу й успіху.

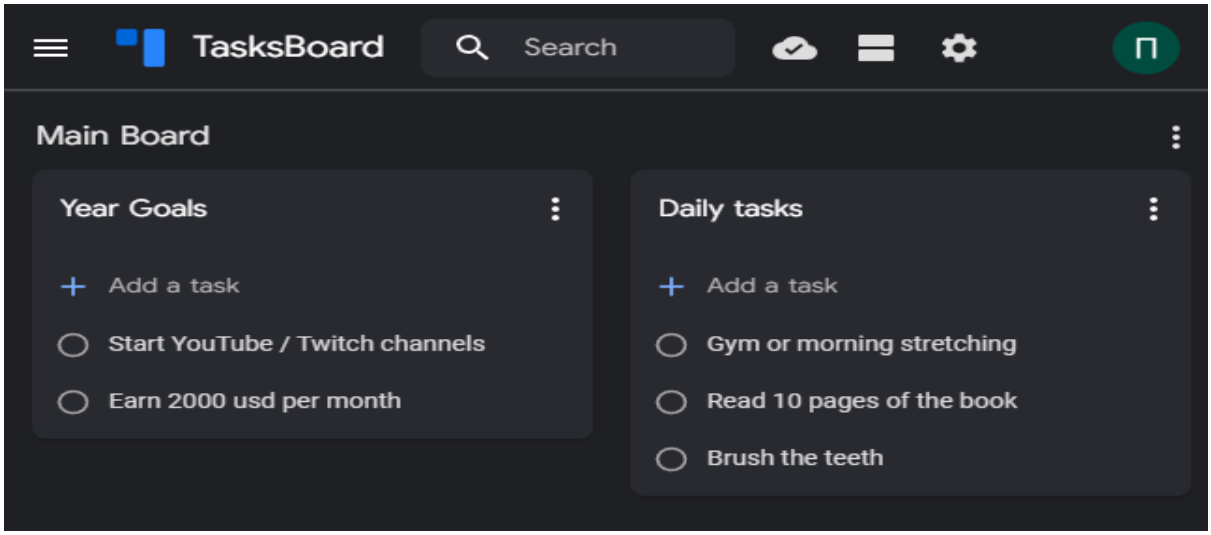


Рисунок 1. Зовнішній вигляд програми Google Tasks

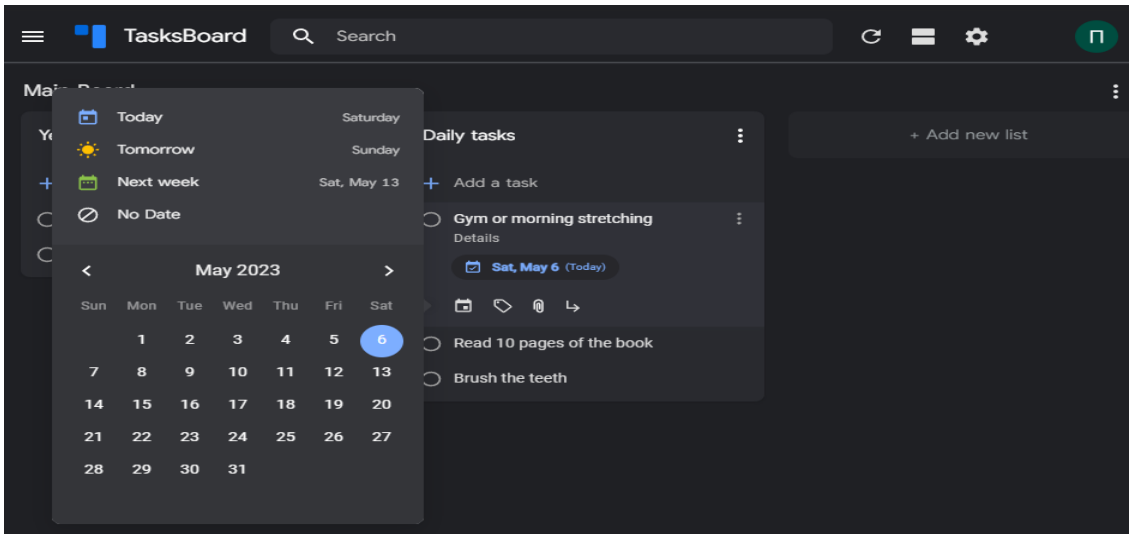


Рисунок 2. Розширені налаштування (дата) для задач в програмі Google Tasks

1.4. Todoist

Todoist – це відомий онлайн сервіс, за допомогою функціоналу якого можна здійснювати управління як і невеликими окремими задачами, додаючи до них опис, дату, пріоритет, так і створювати додаткові підзадачі, також відомі як підзадачі. Цей програмний засіб також має можливість синхронізувати всі дані акаунту користувача між всіма його пристроями та девайсами.

Основний функціонал Todoist:

1. Створення задач, підзадач.
2. Налаштування дати, опису, сповіщень, прикріплення файлів до задач.

Головні переваги цієї платформи:

1. Синхронізація з Google Calendar, Dropbox.
2. Підтримка різних платформ (кросплатформеність).
3. Дуже зручний інтерфейс, інтуїтивно зрозумілий, приємна кольорова гама.

Функціонал Todoist дуже схожий на Google Tasks, особливо своїми базовими елементами, такими як механізм задача-підзадача, сповіщення, налаштування дати, опису та інших параметрів для задачі. Проте деякі функції заблоковані в безкоштовній версії додатку: сповіщення, створення нових проектів.

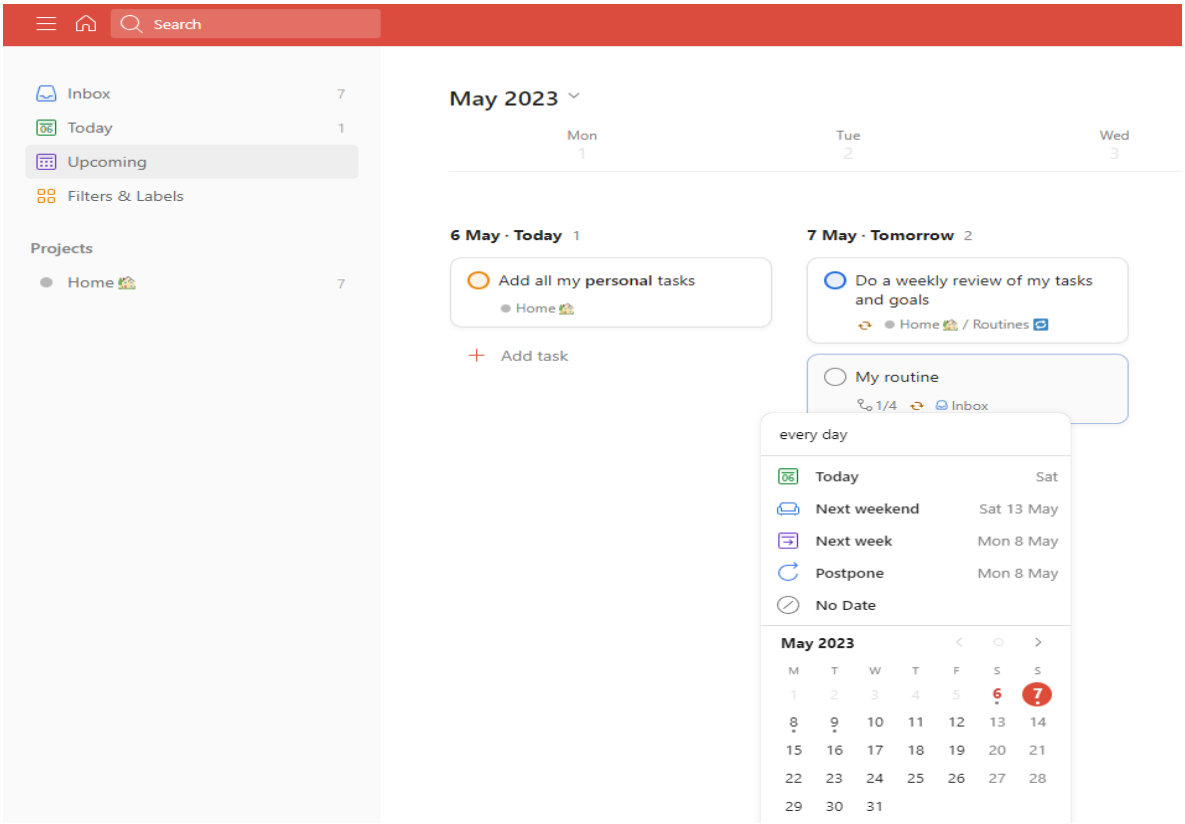


Рисунок 3. Інтерфейс додатку Todoist

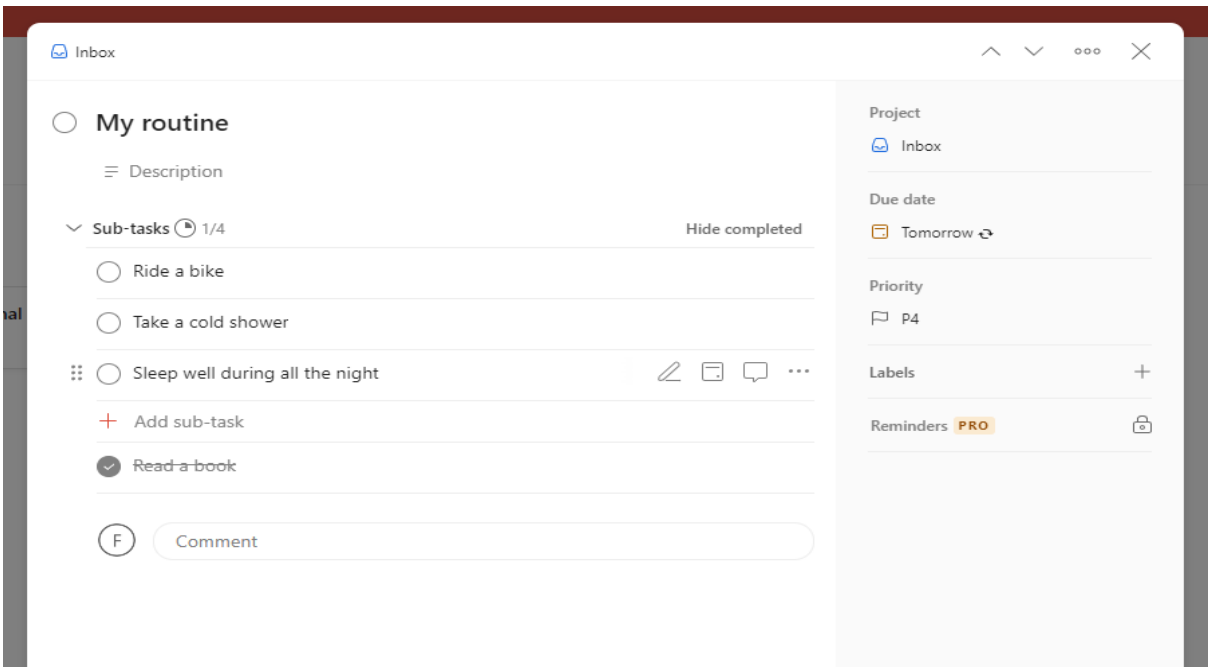


Рисунок 4. Меню для створення підзадач і сповіщень, та налаштування задачі

1.5. Trello

Trello - це платформа для управління проектами та завданнями, що заснована на дошках, де розміщені картки. Кожна картка містить в собі всі дані про конкретне завдання, а дошка відображає всі завдання, тобто всі картки, що потребують уваги. Простий та зрозумілий вигляд: Trello має простий та інтуїтивно зрозумілий інтерфейс, який дозволяє швидко зорієнтуватися в системі та розпочати роботу.

Розглянемо переваги сервісу Trello:

1. Можливість використовувати додаток для планування будь-яких проектів, від невеликих особистих задач до великих повноцінних командних проектів. Багато компаній навіть використовує Trello для менеджменту.

2. На картках завдань можна відслідковувати статус, а також додавати опис, коментарі, терміни виконання, прикріпляти необхідні файли, посилання та додаткову інформацію.

3. Наявний функціонал створення окремих робочих просторів для різних проектів, що полегшує та оптимізує управління роботою та зберігання інформації.

На основі цих даних з опису платформи було зроблено висновок, що Trello все ж відрізняється від попередніх сервісів з управління задачами, тому що орієнтований більше на промислове використання, тобто компанії, підприємства, тощо.

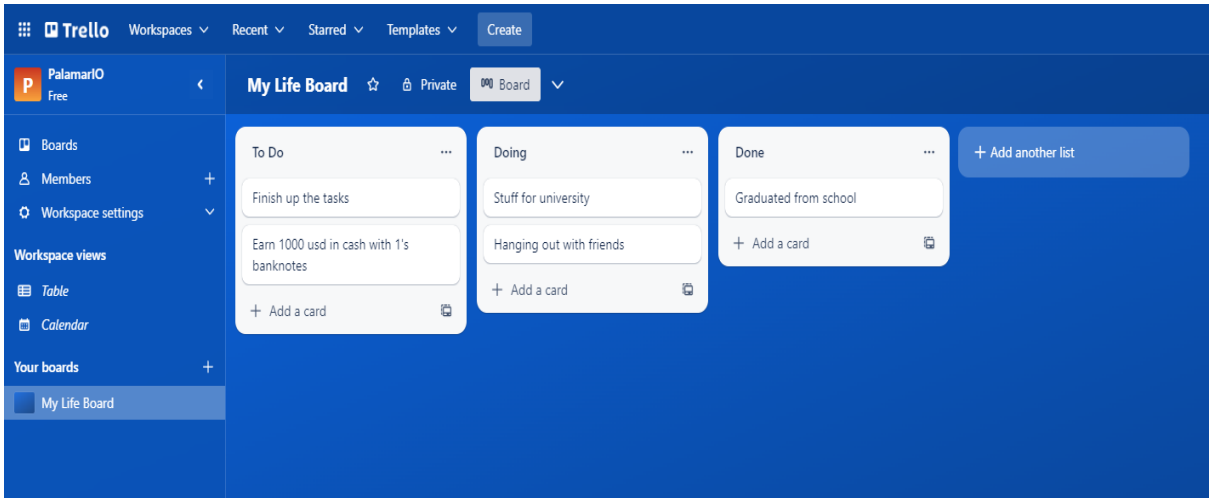


Рисунок 5. UI Trello із основними елементами управління та менеджменту

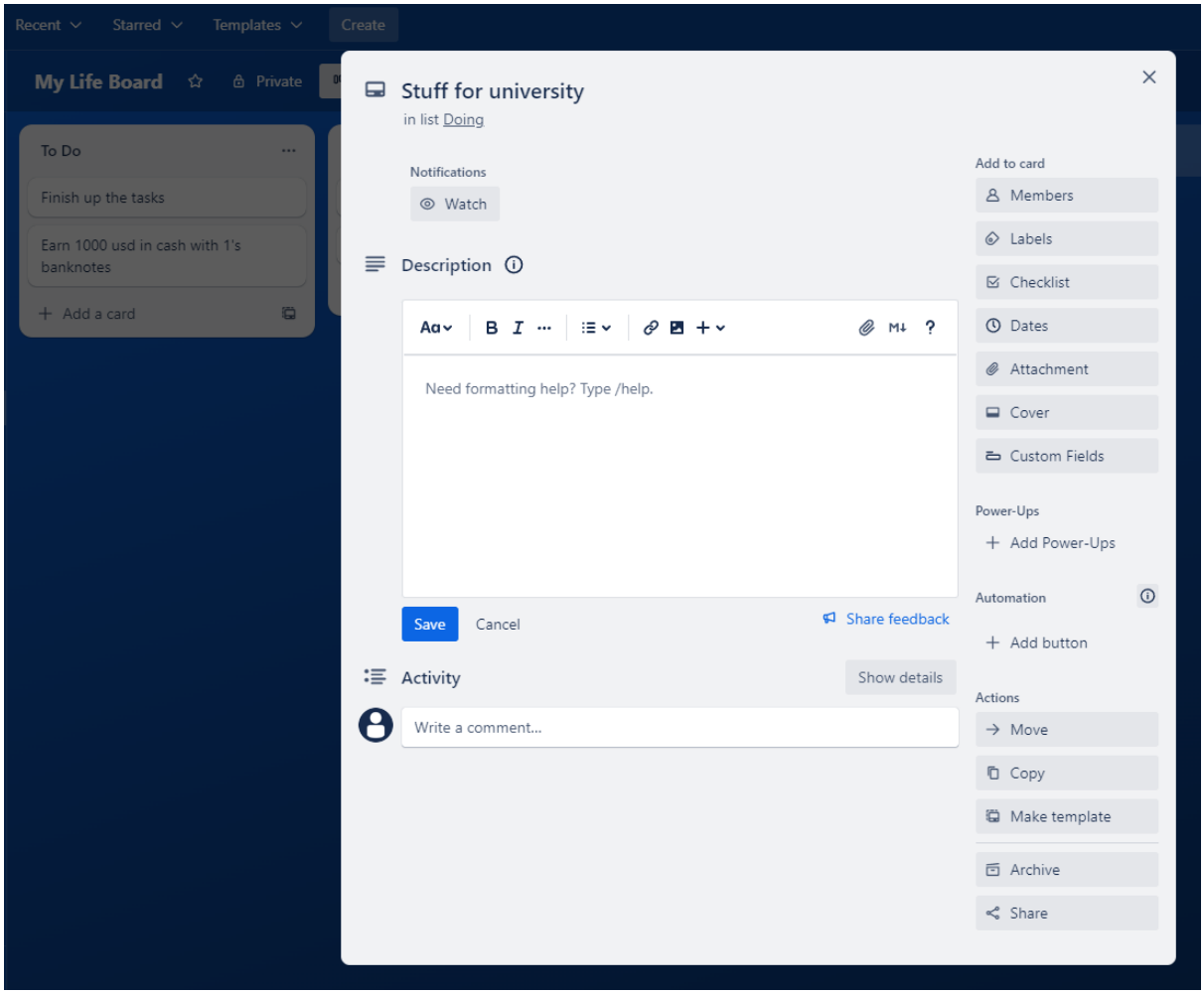


Рисунок 6. UI Trello із функціоналом взаємодії з карткою

1.6. Any.do

Any.do - це інтуїтивно зрозумілий та простий додаток, який найкращим чином адаптований під прості особисті задачі, управління задачами невеликої команди, проекту. Сервіс має простий та приємний інтерфейс. У порівнянні із платформами, що вказані вище, цей додаток призначений для простого менеджменту, і в ньому бракує функціоналу, який є необхідним для великих проектів та компаній.

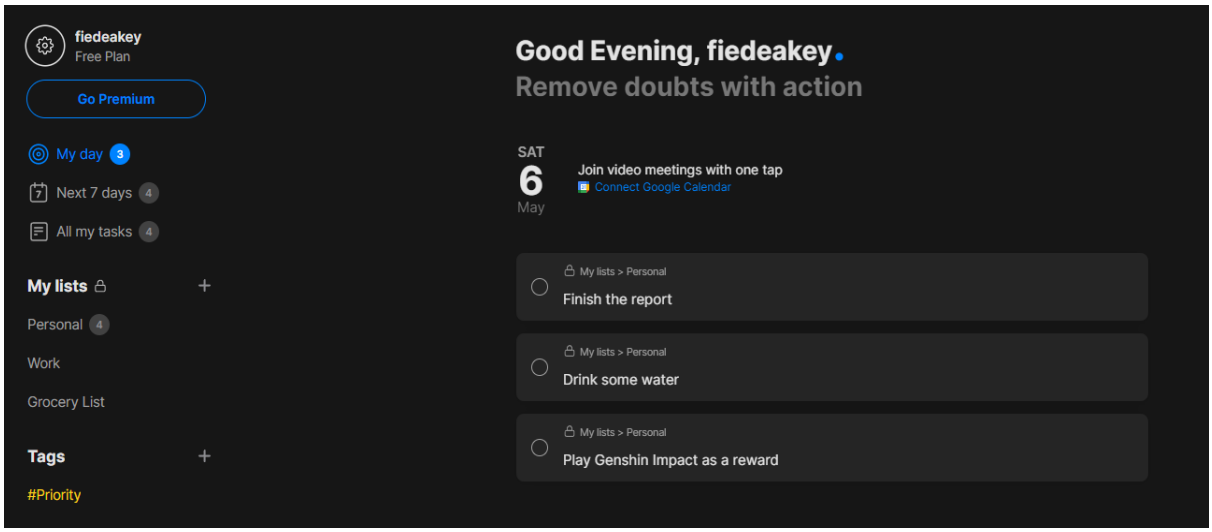


Рисунок 7. UI Any.do для створення задач та їхнього відстеження

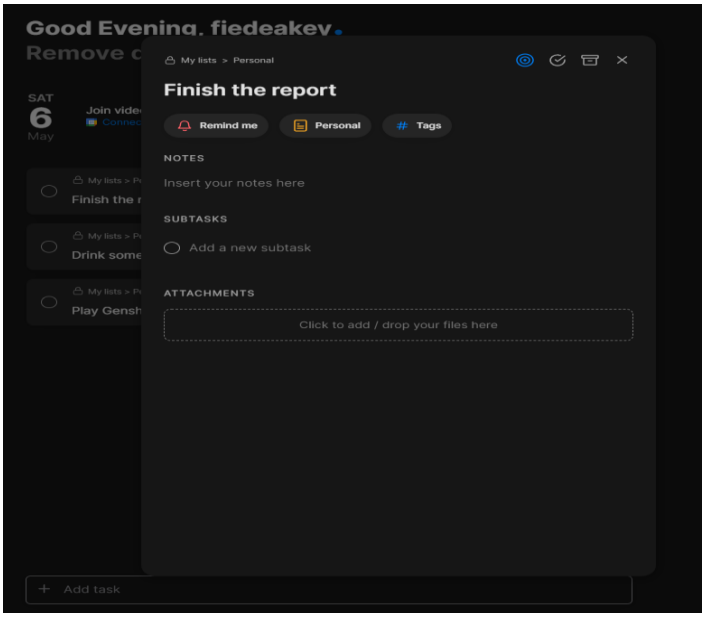


Рисунок 8. UI Any.do для редагування та заповнення елементів задачі

1.7. Telegram-бот Ok, Bob!

Цей Telegram-бот створений для менеджменту задач всередині мессенджеру. Він дозволяє створювати задачі, задавати їм терміни виконання. Функціонал боту дуже обмежений, оскільки тут неможлива синхронізація із Google Calendar, не можна ставити пріоритети задач, додавати опис та інше (див. мал. 9).

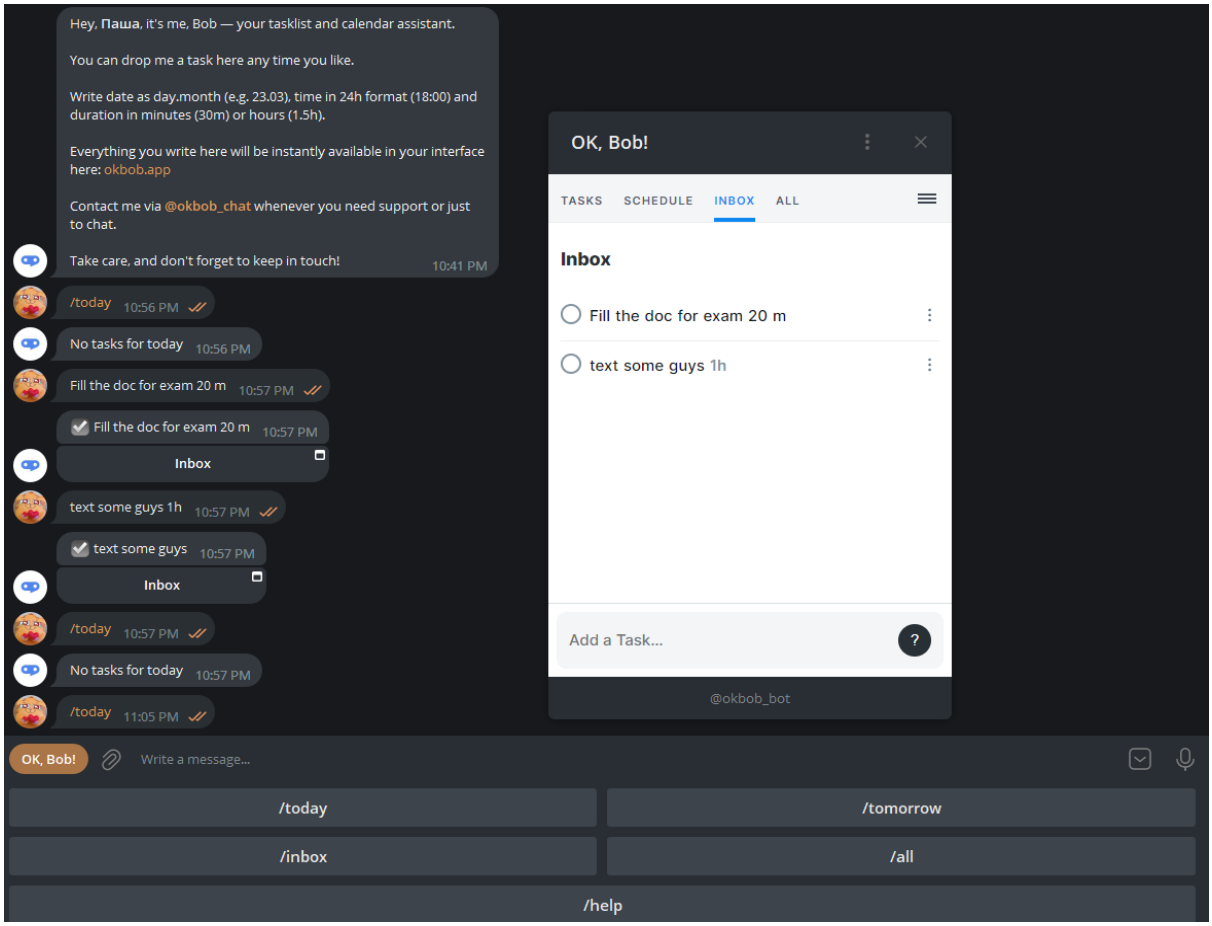


Рисунок 9. UI Ok, Bob! для редагування та заповнення елементів задачі

Отже, цей бот – найпопулярніший серед менеджерів задач в месенджері Telegram, але має досить урізаний функціонал для такої предметної галузі.

1.8. Таблиця порівняння аналогів

Таблиця 1. Зведені результати аналізу характеристик додатків для календарного планування особистих задач

Характеристика	**Google Tasks**	**Todoist**	**Trello**	**AnyDo**	**Ok, Bob!**	**TasokiBot**
Створення, редагування, видалення задач	+	+	+	+	+	+
Додавання тегів, опису, файлів до задачі	+	+	+	+	+	+
Встановлення дати	+	+	+	+	+	+
Наявність інтеграції із Telegram	-	-	-	-	+	+
Інтеграція із Google Calendar	+	-	-	-	-	+
Синхронізація між пристроями	+	+	+	+	+	+

2. ВИМОГИ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Технічне завдання

Технічне завдання передбачає розробку Telegram-бота з використанням технології ASP.NET, фреймворку .NET Core 7.0 та мови програмування C#. Бот буде у вигляді єдиного сервісу, що забезпечуватиме його функціональність.

Основними вимогами до бота є здатність створювати задачі на основі інструкцій, які користувач передає йому. Це означає, що бот повинен мати можливість розпізнавати команди та параметри, передані йому через Telegram-інтерфейс, і створювати відповідні задачі з вказаними даними. Для збереження та організації цих задач бот може використовувати базу даних й ті можливості, які надає ASP.NET.

Крім того, бот повинен синхронізовувати створені задачі з сервісом Google Calendar. Це означає, що бот повинен мати доступ до Google Calendar API та виконувати необхідні операції для створення подій у календарі користувача. Це забезпечить синхронізацію між Telegram-ботом та календарним сервісом, що дозволяє користувачеві бачити свої завдання на обох платформах.

Додатковим функціоналом, який необхідно реалізувати, є механізм відправки повідомлень користувачеві для нагадування про своєчасне виконання завдань. Бот повинен мати можливість створювати та надсилати повідомлення з вказаною інформацією та нагадуваннями на основі налаштувань користувача.

2.2. Функціональні вимоги

2.2.1. Створення задачі

Бот повинен вміти створювати задачі згідно інформації, яку користувач вводить за допомогою тексту. Для цього необхідна послідовна логіка, яка буде вести юзера покроково по стадіям створенням задачі.

2.2.2. Валідація введеного користувачем тексту

Потрібно правильно обробляти введені команди чи текстові дані, що надходять до бота ззовні, і видавати відповідні попереджувальні повідомлення користувачеві, які застерігатимуть його від некоректної взаємодії з ботом. Така перевірка повинна відбуватися шляхом зіставлення отриманого повідомлення із певними шаблонами, які запрограмовані всередині боту.

2.2.3. Синхронізація створених задач із Google Calendar

Бот повинен вміти синхронізувати задачі користувача із сервісом Google Calendar, де будуть відображатися задачі згідно із запланованою для них датою. Якщо завдання будуть видалятися із бази боту, при синхронізації вони повинні також видалятися із календаря.

2.2.4. Відображення поточних задач

Користувач повинен мати можливість через певну команду отримати повний список своїх задач, де буде виведено дату їх виконання та опис. Задачі матимуть певний формат відображення.

2.2.5. Видалення задачі

Видалення задачі повинно відбуватися через її пошук за назвою в системі. За допомогою регулярного виразу потрібно зіставити введену назву задачі з існуючими задачами та видалити правильну. Після такого видалення задача видаляється одразу із Google Calendar.

2.2.6. Відправлення нагадувань через сповіщення

За деякий час до завершення терміну задач потрібно відправляти в чат користувачу нагадування про її виконання. Таким чином користувач матиме pop-up повідомлення, які будуть спонукати його на виконання встановлених планів, цілей та термінів.

2.3. Нефункціональні вимоги

2.3.1. Зрозумілість

Користувач повинен мати змогу легко користуватися ботом. Інтерфейс та команди мають бути інтуїтивно зрозумілими, щоб юзер міг без перешкод почати використання.

2.3.2. Надійність та стійкість

При введенні некоректних даних або команд система повинна продовжувати свою роботу і правильно реагувати на запити користувача. При виникненні критичної помилки необхідно інформувати користувача про збій в системі. При раптовому відключенні системи необхідно, щоб її робота була поновлена протягом двох годин.

2.3.3. Безпека

Бот буде оптимізовано використовувати доступні ресурси шляхом зберігання тільки необхідної інформації в базі даних. Токен боту зберігається всередині коду і має обмежений доступ лише до деяких його частин.

2.3.4. Масштабованість

Середовище боту повинно підтримувати можливість додавання нових сервісів в проєкт без модифікації існуючого коду та конфігурації.

2.3.5. Системні вимоги

Повинні бути надані наступні вимоги:

Таблиця 2. Рекомендовані системні вимоги

CPU	Intel Xeon 16x 4.3 GHz
RAM	32 GB

SSD	2 TB
Network	Speed up to 1 Gbit/s

Якщо на ранньому етапі розробки не можуть бути надані рекомендовані системні вимоги, необхідно надати мінімальні (див. табл. 3.)

Таблиця 3. Мінімальні системні вимоги

CPU	Intel Xeon 4x 3.0 GHz
RAM	8 GB
SSD	512 GB
Network	Speed up to 100 Mbit/s

2.4. Асоціативна мапа

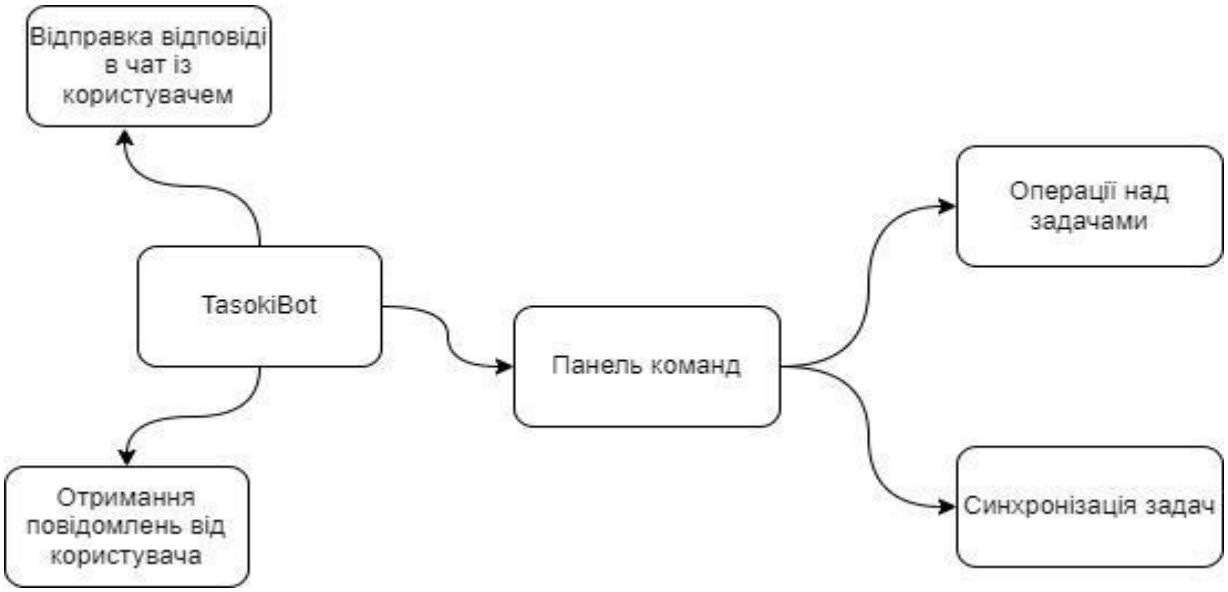


Рисунок 10. Асоціативна мапа TasokiBot

2.5. Діаграми використання

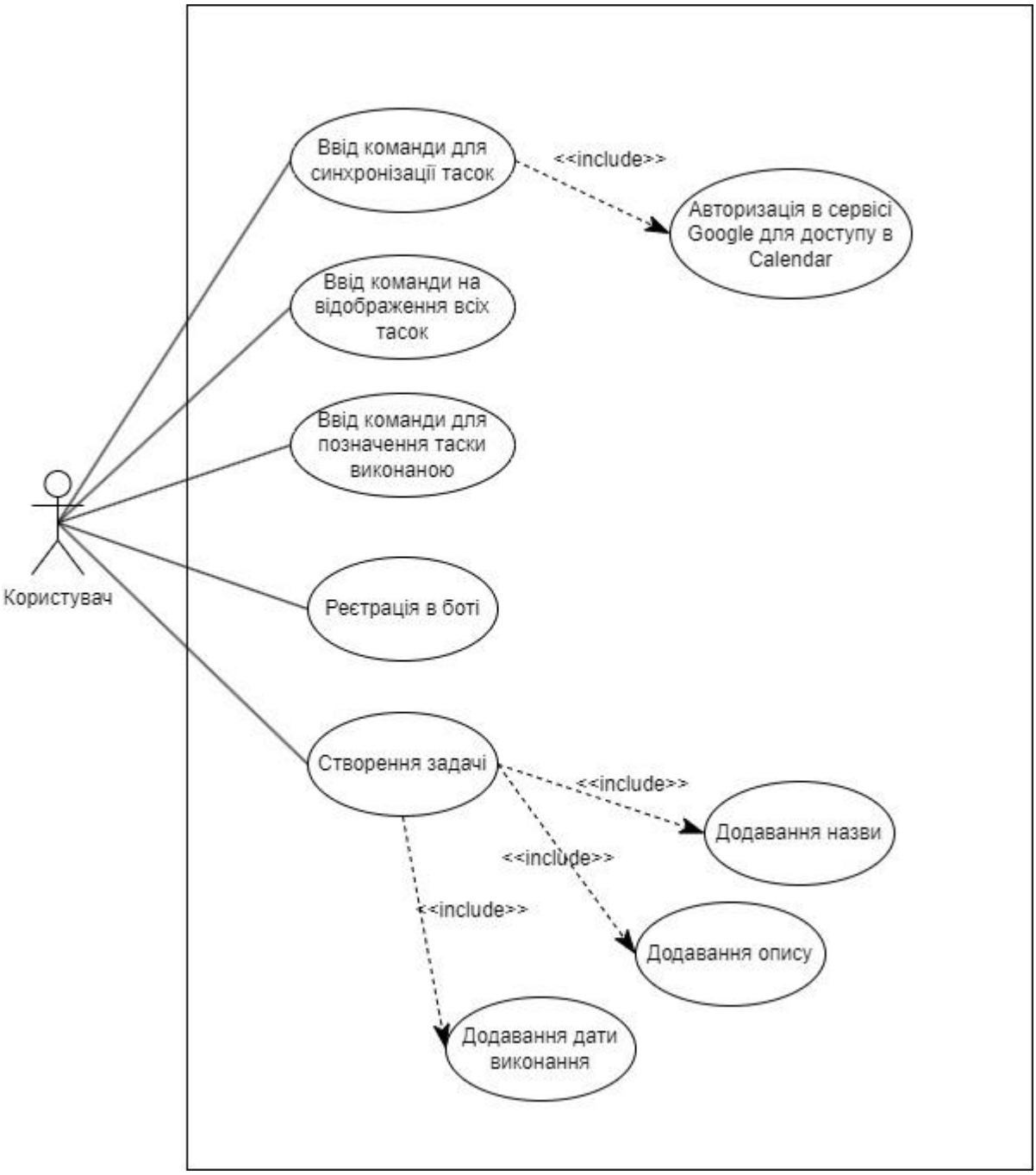


Рисунок 11. Діаграма використання

2.6. Діаграми класів

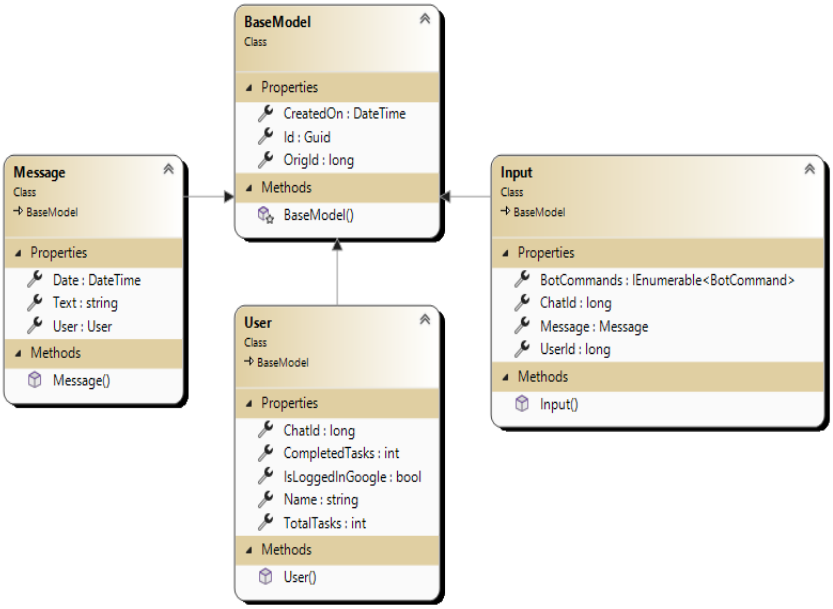


Рисунок 12. Діаграма класів - TasokiBot.Domain

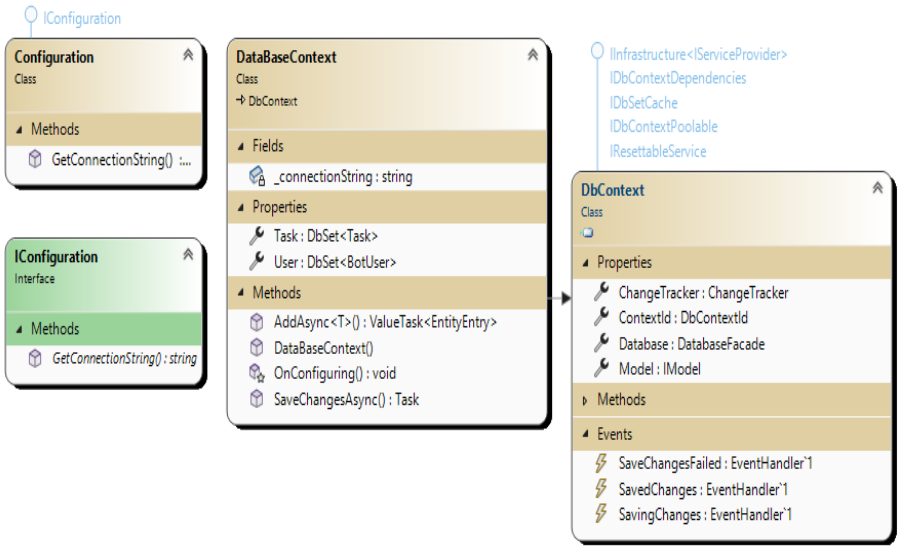


Рисунок 13. Діаграма класів - TasokiBot.Database

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Засоби та інструменти розробки

3.1.1. Мова C#

C# – це об'єктно-орієнтована мова програмування, яка була створена корпорацією Microsoft. Цей інструмент підтримує всі популярні платформи, тобто такі операційні системи як Windows, Mac OS, Android[8]. Однією з ключових особливостей мови є автоматичне відслідковування використання пам'яті та забезпечення безпеки програми. Таку можливість надають механізми, які вбудовані в мову, наприклад, так званий, garbage–collector. Він аналізує програму, яка виконується, та вивільнює пам’ять за рахунок чистки тих даних, які не використовуються. Також ця мова має дуже велику кількість допоміжних бібліотек, які можуть бути використані як додатковий інструмент при розробці програмного забезпечення для прискорення та полегшення самого процесу створення додатку.

Основні переваги C#:

Об'єктно-орієнтований підхід. Мова C# базується на основі об'єктно-орієнтованої парадигми, що дозволяє розробникам організовувати код у вигляді класів та об'єктів, що сприяє більшій модульності, повторному використанню і масштабованості програм.

Керування пам'яттю. В мові вивикористовується автоматичне управління пам'яттю (Garbage Collection), що дозволяє розробникам запобігти проблеми з ручним виділенням та звільненням пам'яті.

Безпека типів. C# має строгу типізацію, яка перевіряє правильність використання типів даних на етапі компіляції, що допомагає уникнути помилок при виконанні програми.

Підтримка багатопоточності. C# має вбудовану підтримку багатопоточного програмування, що дозволяє створювати програми, які можуть

працювати з кількома потоками одночасно, що сприяє поліпшенню ефективності роботи додатків.

Великий обсяг стандартної бібліотеки та Nuget. C# має велику стандартну бібліотеку класів, яка надає широкий набір функцій і інструментів для розробки додатків. Це включає роботу з мережевими протоколами, базами даних, графічними елементами і багато іншого. Також об’ємна кількість функціоналу знаходиться в пакетах Nuget, що можуть легко додаватися в проєкт.

Інтеграція з платформою .NET. C# є основною мовою програмування платформи .NET, що дозволяє розробляти додатки, що працюють на різних операційних системах, таких як Windows, MacOS і Linux.

3.1.2. .NET Core

.NET Core — це безкоштовна платформа з відкритим вихідним кодом для розробки кросплатформенних програм[9]. Цей фреймворк дуже часто використовується для розробки десктопних додатків, веб-додатків, мікросервісів, проектів SaaS тощо. .NET Core підтримує розробку додатків на мовах програмування C#, Visual Basic і F#. Для розробки додатків на платформі фреймворку .NET Core використовуються такі інтегровані середовища, як Visual Studio, або JetBrains Rider. Ця версія є покращеною версією .NET Framework, що наразі є застарілою та виходить із використання.

3.1.3. ASP.NET Core

Це технологія від Microsoft, яка слугує для створення різноманітних веб-додатків, веб-серверів тощо. ASP.NET Core має свій легкий контейнер для впровадження залежностей і може працювати незалежно від кросплатформеного середовища .NET Core на Windows, Mac OS та Linux. Крім того, він є opensource-фреймворком і повністю доступний на GitHub[10]. Цей інструмент допомагає при розробці веб-сервісів, які повинні постійно приймати, обробляти та відправляти дані, а також дозволяє розроблені сервіси зручно розгортати на серверах,

налаштовуючи підключення до інших компонентів, таких як база даних, брокер повідомлень, кеш-сховище тощо.

3.1.3.1. Основні особливості ASP.NET Core

Мультиплатформність. Технологія надає можливість для розробки веб-додатків для такого набору платформ, що включає Windows, macOS і Linux. Це також робить можливим створення розробниками додатків, які можуть працювати на різних серверних платформах.

Ефективність роботи. ASP.NET Core забезпечує швидку обробку запитів і відповідей завдяки оптимізації та використанню технології асинхронної обробки.

Підтримка API. Фреймворк має вбудовану підтримку створення веб-сервісів і API. Розробники можуть легко створювати REST API, використовуючи вбудовані інструменти та атрибути маршрутизації, що спрощує розробку спілкування між мікро сервісами при розробці додатків, що використовують саме такий тип архітектури.

Безпека. ASP.NET Core має механізми для забезпечення безпеки додатків, включаючи механізми аутентифікації, авторизації та захисту від атак.

Інструменти розробки. ASP.NET Core інтегрується з різними середовищами розробки, такими як Visual Studio, JetBrains Rider. Це надає програмістам потужні інструменти для написання, відладки та тестування коду.

3.1.4. Docker

Docker – це технологія із відкритим кодом, яка дозволяє ізолювати частини програми, мікросервіси в окремих контейнерах, і забезпечити їхню стабільну та незалежну роботу[11]. Ці контейнери всередині себе містять повний набір необхідних компонентів для запуску та роботи програми. Величезна перевага Docker полягає в тому, що його контейнери можуть бути запущені будь-де, на будь-якому сервері, який має встановлений Docker. Завдяки цьому відпадає необхідність встановлювати додаткові бібліотеки, залежності, та ін. Docker дозволяє легко й

швидко розгортати та масштабувати додатки, забезпечує їхню стабільну роботу та безпеку.

3.1.4.1. Особливості технології Docker:

Масштабованість. Docker дозволяє легко масштабувати додатки шляхом запуску декількох контейнерів одночасно. Це надає гнучкість та можливість розподілу навантаження між сервісами для забезпечення високої ефективності роботи.

Контейнеризація. Можливість пакування додатків та їхніх залежностей у контейнери, які забезпечують повноцінне ізольоване середовище для виконання програм, дозволяє кожному окремому контейнеру працювати незалежно від інших контейнерів.

Переносимість. Контейнери Docker можуть бути виконані на будь-якій системі, яка підтримує Docker. Це означає, що додатки можуть розроблятися і тестуватися на одній системі, в одному середовищі, а потім розгортатися на іншій платформі без необхідності у встановленні додаткових залежностей.

Ізоляція. Кожен контейнер Docker ізольований від інших, що забезпечує безпеку та незалежність роботи програм. Це означає, що проблеми, що виникають в одному контейнері, не впливають на інші, забезпечуючи стабільність системи в цілому.

Технологія "Інфраструктура як код". Ця технологія дозволяє розробникам описувати інфраструктуру та конфігурацію додатків у вигляді коду, використовуючи Dockerfile та docker-compose.yml. Це значно полегшує процес розгортання, керування та підтримки інфраструктури, надаючи багато переваг, одна з яких, це кодове представлення інфраструктури, що забезпечує повну відтворюваність середовища. Завдяки Dockerfile та docker-compose.yml всі необхідні налаштування та залежності можуть бути описані як код і збережені у системі контролю версій. Це зменшує можливість виникнення проблем, пов'язаних з різними конфігураціями середовищ.

docker-compose.yaml

F: > TasokiBotEnvironment > docker-compose.yaml

```yaml
version: '3.9'

services:
  redis:
    image: redis:latest
    restart: always
    container_name: Tasoki_redis
    environment:
      - REDIS_PASSWORD=password
    ports:
      - 6380:6379
    volumes:
      - ./docker_data/redis:/data

  db:
    image: postgres:latest
    restart: always
    container_name: Tasoki_DB
    environment:
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=password
    ports:
      - '5432:5432'
    volumes:
      - ./docker_data/postgre:/data
```

Рисунок 14. Приклад вигляду файлу docker-compose.yaml

3.1.5. PostgreSQL

PostgreSQL — це об'єктно-реляційна система керування базами даних з відкритим кодом, яка забезпечує стабільність, надійність та розширюваність. Завдяки відкритості коду розробники мають змогу налаштовувати різноманітний функціонал цієї технології під себе[12]. PostgreSQL підтримує широкий спектр функціональних можливостей, що робить його популярним в різних сферах, від невеликих проектів до великих систем.

Основною концепцією PostgreSQL є реляційна модель даних. Вона базується на табличній структурі, де дані організовані у таблицях з колонками, які прописані під різними типами даних, та рядками, що містять конкретні значення. PostgreSQL підтримує SQL, що дозволяє зручно взаємодіяти із базою даних.

Однією з ключових особливостей PostgreSQL є його розширюваність. Вона надає розробникам можливість збільшувати функціонал БД за рахунок використання плагінів, власних функцій, типів даних та мов програмування. Завдяки цьому, PostgreSQL може бути настроєний під конкретні потреби і доповнений для підтримки додаткових опцій. Він підтримує реплікацію, що дозволяє створювати декілька копій бази даних для розподілення навантаження. Також PostgreSQL може бути інтегрований з системами кешування та зберігання даних, такими як Redis та Apache Kafka, для підвищення продуктивності та оптимізації роботи з даними. Підтримується транзакційна система, що дозволяє забезпечити цілісність даних та стабільність бази даних, тому що при такій системі всі операції транзакції виконуються повністю або не виконуються взагалі.

3.1.6. Git

Git - це система керування версіями, яка надає широкий спектр функціоналу для ефективного керування змінами в файловій системі проекту. Основна мета Git полягає в забезпеченні точного відслідковування змін у файлах, управлінні версіями проекту та спільній роботі над кодом. Його розподілена архітектура дозволяє кожному розробнику мати повну копію репозиторію на своєму локальному пристрої, що дає можливість працювати незалежно від мережі. Кожна копія Git містить повну історію проекту; зміни можуть бути легко об'єднані та синхронізовані з іншими копіями. Завдяки Git розробники можуть ефективно співпрацювати, використовуючи гілки розробки, які дозволяють паралельно працювати над різним функціоналом та злити зміни в основну гілку по завершенні роботи.

3.1.7. JetBrains Rider

JetBrains Rider - це інтегроване середовище розробки (ICP), призначене для проектів, розроблених на платформі .NET. Воно надає потужні інструменти для розробки програмного забезпечення, забезпечуючи широку функціональність та підтримку різних мов програмування, таких як C#, VB.NET, ASP.NET, JavaScript,

TypeScript та ін[13]. Його функціонал також доповнюється інтеграцією з іншими інструментами, що допомагають при розробці, а саме Git, GitHub і Azure DevOps. Це дозволяє розробникам легко керувати версіями коду, спільно працювати та забезпечувати ефективну роботу з проектами. Також варто зазначити, що Rider має просунуті технології рефакторингу коду та кастомізації стандартів написання програм під конкретні потреби. Автогенерація коду, яка також стає в нагоді розробникам, допомагає скоротити час на написання одноманітних шматків необхідного коду. Rider містить в собі потужні інструменти для відлагодження програм, що допомагає максимально швидко знаходити баги або місця в коді, які можна оптимізувати.

3.1.8. JetBrains DataGrip

JetBrains DataGrip - це інтегроване середовище для роботи з базами даних від JetBrains, що призначене для зручного управління реляційними та нереляційними базами даних, включаючи PostgreSQL, MySQL, MongoDB, ClickHouse та інші[14]. Воно має зручний текстовий редактор SQL, що дозволяє редагувати та форматувати SQL-запити. DataGrip надає зручні інструменти для аналізу, які допомагають виявити місця, що є точками найдовшого виконання при запитів до бази даних. Також можливо створювати скрипти в автоматичному режимі, за допомогою яких можно розгортати повністю базу даних з нуля.

3.2. Системна архітектура

3.2.1. Взаємодія сервісу із інфраструктурою

В даному проекті розробляється Telegram-бот під назвою TasokiBot, в корені програми якого присутній один сервіс, що відповідає за передачу запитів або дій користувача в рамках боту до самого програмного забезпечення для обробки. Він оточений кількома допоміжними бібліотеками, які надають основний функціонал та мають в собі прописану логіку роботи.

В таблиці 4 приведено структуру проекту та короткий опис кожного елементу.

Таблиця 4. Компоненти проєкту TasokiBot та їхній опис

Назва елементу	Тип	Опис
TasokiBot.Core	Мікросервіс	Відповідає за отримання будь якої інформації від бота
TasokiBot.Dal	Бібліотека	Містить в собі класи-моделі, що представляють об’єкти для взаємодії з базою даних
TasokiBot.Database	Бібліотека	Має файли з конфігурацією бази даних та міграціями
TasokiBot.DependencyInjection	Бібліотека	Налаштовує залежності для роботи системи
TasokiBot.Domain	Бібліотека	Містить в собі класи доменних моделей для використання всередині системи
TasokiBot.GoogleCalendar	Бібліотека	Необхідні методи та дані для інтеграції та

		роботи із Google Calendar API
TasokiBot.Infrastructure	Бібліотека	Основні класи із функціоналом системи: обробка даних, взаємодія з БД, відправка відповіді користувачеві
TasokiBot.Extension	Бібліотека	Утиліти та допоміжний функціонал
TasokiBot.Tests	Бібліотека	Містить в собі юніт тести для перевірки безпомилкової роботи коду

Таблиця 4. Компоненти проєкту TasokiBot та їхній опис

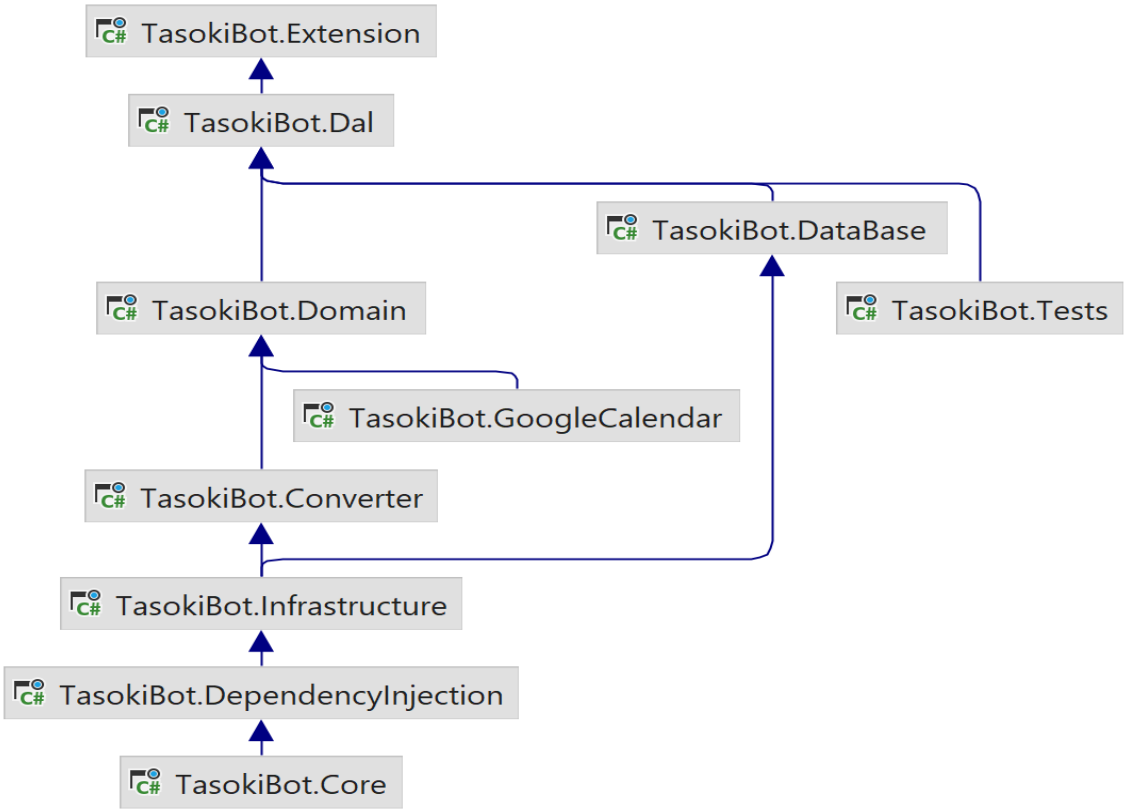


Рисунок 15. Структура залежності бібліотек в проєкті

3.2.2. Взаємодія компонентів проекту

В даному програмному забезпеченні було задіяно невелику кількість компонентів. В корені програми знаходиться сервіс TasokiBot.Core, який відповідає за отримання будь-якої інформації від користувача боту в Telegram. Далі цей сервіс використовує функціонал бібліотек, що також були реалізовані в даному проєкті, для забезпечення коректної роботи програми. Для встановлення підключення до боту використовується бібліотека TBot, яка знаходиться у вільному доступі та було встановлена до проєкту із менеджеру пакетів NuGet. Ця бібліотека використовує Redis в своїй роботі.

Для зберігання даних користувача ботом було застосовано технологію PostgreSQL. Взаємодія головних компонентів системи наочно проілюстрована (див. рис. 15), де спілкування між базою даних та основним сервісом відбувається через протокол TCP/IP, а звернення до Redis — за RESP.

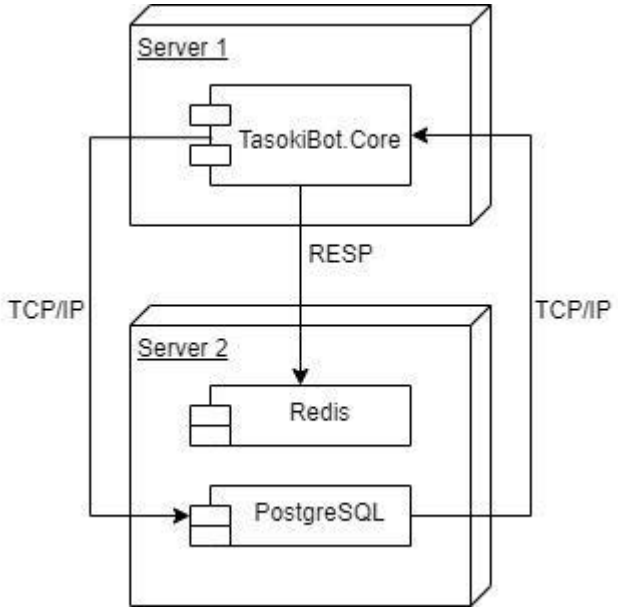


Рисунок 16. Діаграма розгортання

3.2.3. Структура бази даних

За допомогою системи керування базами даних DataGrip було створено базу даних на основі технології PostgreSQL із наступними таблицями:

1. **Таблиця User.** Її було створено для зберігання детальної інформації про користувача ботом, для того щоб можна було ідентифікувати юзера згодом і надати йому доступ до функціоналу боту.

2. **Таблиця Task.** Ця таблиця слугує сховищем даних про ті задачі, що користувач створив. Завдяки foreign key є можливість посилання на таблицю User.

3. **Таблиця __EFMigrationsHistory.** В цій системній таблиці, яка створюється автоматично, зберігаються дані про виконані міграції над конкретною базою даних. Це робиться для того, що PostgreSQL міг визначити, які зміни в структурі бази даних відбулися, а яких ще немає.

Структуру цих таблиць, а також існуючі зв’язки наведено на рисунках 16-17.

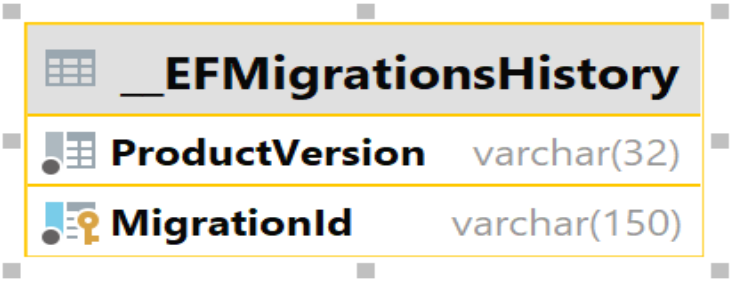


Рисунок 17. Структура таблиці __EFMigrationsHistory

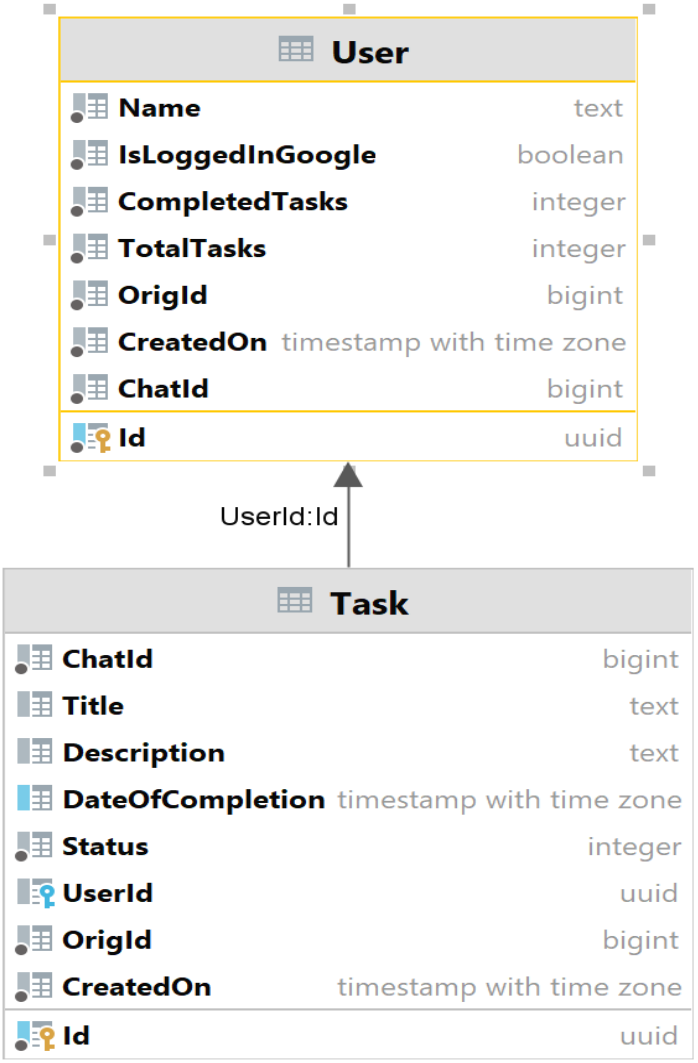


Рисунок 18. Структура таблиці **User** і **Task,** а також відображення зв'язку між ними

3.2.4. Команди та їх обробка

Для взаємодії користувача з ботом потрібно, щоб був певний набір команд, які будуть виконувати певний функціонал. Згідно правил проєктування, для реалізації подібної поведінку необхідно розробити рішення спочатку на рівні абстракції. В проєкті TasokiBot було застосовано патерн State Machine, який доволі зручним методом вирішує проблему імплементації багатьох станів роботи системи. Протягом взаємодії користувача із ботом в програмі змінюються стани створюваної користувачем задачі. Наприклад, спочатку задача порожня, оскільки тільки-но

створена. Згодом дані про задачу заповнюються назвою, описом, датою тощо (див. рис. 18).

Команди та їх призначення:

1. /newtask – розпочати процес створення задачі.
2. /sync – синхронізувати задачі із Google Calendar.
3. /delete [номер] – видалити задачу.

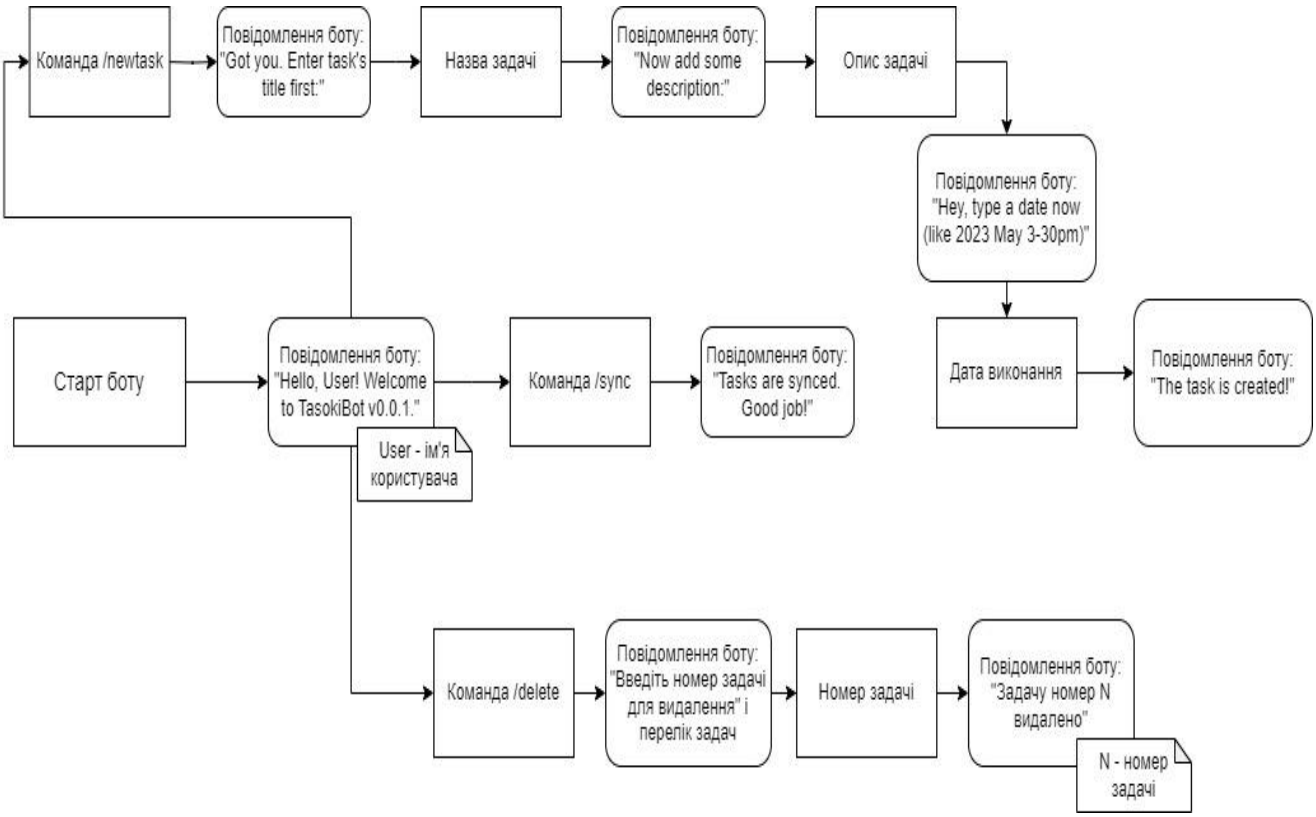


Рисунок 19. Структура боту

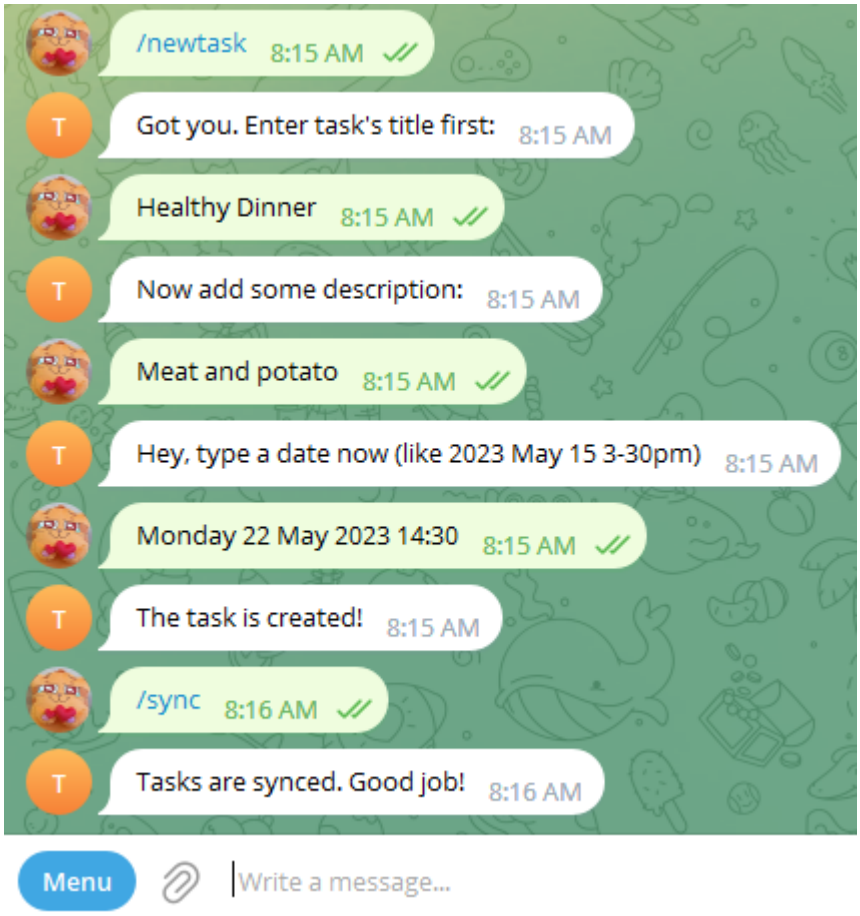


Рисунок 20. Процесс заповнення по мірі наповнення задачі

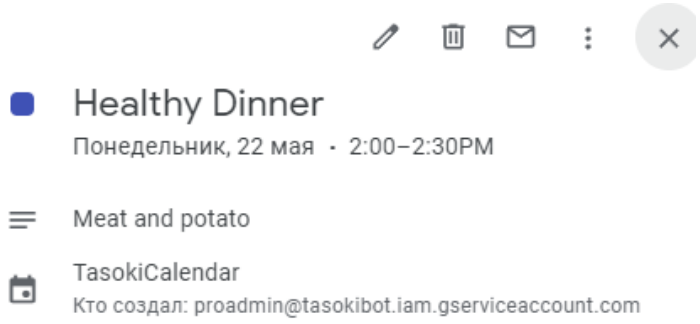


Рисунок 21. Елемент Google Calendar після синхронізації

3.3. Тестування програмного забезпечення

Безперебійну роботу TasokiBot, як і будь-якого іншого програмного забезпечення, гарантує етап тестування. В даному проєкті для автоматичного тестування коду було обрано засоби бібліотеки NUnit, яка знаходиться у вільному доступі в NuGet. На прикладі нижче наведено один із юніт-тестів, які є в проєкті. В

даному тесті перевіряється робота методу по конвертації дати в правильний формат, його поведінка при різних вхідних даних (див. рис. 19).

```
[Test]
public void ConvertToDateTimeAsync_ValidDateString_ReturnsDateTime()
{
    // Arrange
    string dateString = "2022-11-25 09:30:00";

    // Act
    DateTime? result = Utilities.ConvertToDateTimeAsync(dateString);

    // Assert
    Assert.IsNotNull(result);
    Assert.AreEqual(expected: new DateTime(year: 2022, month: 11, day: 25, hour: 9, minute: 30, second: 00), actual: result);
}

[Test]
public void ConvertToDateTimeAsync_InvalidDateString_ReturnsNull()
{
    // Arrange
    string dateString = "May 45 -3409 67:23";

    // Act
    DateTime? result = Utilities.ConvertToDateTimeAsync(dateString);

    // Assert
    Assert.IsNull(result);
}

[Test]
public void ConvertToDateTimeAsync_DateInThePast_ReturnsNull()
{
    // Arrange
    string dateString = "2020-01-01";

    // Act
    DateTime? result = Utilities.ConvertToDateTimeAsync(dateString);

    // Assert
    Assert.IsNull(result);
}
```

Рисунок 22. Приклад одного з юніт тестів проєкту TasokiBot

4. ВИСНОВКИ

За результатами виконання дипломної роботи було розроблено Telegram-бота для календарного планування особистих задач. Він є актуальним, оскільки наразі не існує зручного вбудованого інструменту в месенджер Telegram для планування задач і їхньої синхронізації з сервісом Google Calendar.

1. Проведено аналіз календарного планування задач. Ключові проблеми, пов’язані з тим, що не дуже зручно використовувати сторонні додатки для такого планування. Це призводить до того, що планування задач потребує більше часу і встановлення додаткових програм.

2. Проведено аналіз засобів розробки програмного забезпечення. Обрано мову програмування C# та фреймворк .NET Core, що забезпечує швидкість, зручність у використанні, оскільки є багато бібліотек, що створені під конкретні задачі інтеграції із Telegram та Google Calendar.

3. Розглянуто програмне забезпечення, яке може бути використано для календарного планування задач: Google Tasks, Todoist, Trello, Any.do, Ok, Bob!. Ключовим недоліком існуючих систем є те, що вони не мають інтеграції із популярними соціальними мережами й сервісами водночас.

4. Розроблено телеграм-бота TasokiBot для календарного планування задач.

5. Робота пройшла апробацію на науково-технічній конференції “Застосування програмного забезпечення в інфокомунікаційних технологіях”.

5. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лужна С. Відомо, чому Telegram став таким популярним останнім часом [Електронний ресурс] / Софія Лужна // iTechUA. – 2023. – Режим доступу до ресурсу: https://itechua.com/articles/207735.

2. Причепа І. В. Тайм-менеджмент як дієвий інструмент ефективного використання часу успішного менеджера за сучасних умов [Електронний ресурс] / І. В. Причепа, І. Л. Соломонюк, Т. В. Лесько // Електронне наукове фахове видання "Ефективна економіка"". – 2018. – Режим доступу до ресурсу: http://ir.lib.vntu.edu.ua/bitstream/handle/123456789/23511/51339.pdf?sequence=2&isAllowed=y.

3. Google Tasks [Електронний ресурс] – Режим доступу до ресурсу: https://tasksboard.com/.

4. Todoist [Електронний ресурс] – Режим доступу до ресурсу: https://todoist.com/app/.

5. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/home.

6. AnyDo [Електронний ресурс] – Режим доступу до ресурсу: https://www.any.do/.

7. Telegram-бот OkBob [Електронний ресурс] – Режим доступу до ресурсу: https://t.me/okbob_bot.

8. C#: що це за мова та де її використовують [Електронний ресурс] – Режим доступу до ресурсу: https://robotdreams.cc/uk/blog/284-s-chto-eto-za-yazyk-i-gde-ego-ispolzuyut.

9. What is .NET? Introduction and overview [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/dotnet/core/introduction.

10. Зашко Б. Переваги використання технології ASP.NET Core для створення веб-серверу [Електронний ресурс] / Б. Зашко – Режим доступу до ресурсу: https://elartu.tntu.edu.ua/bitstream/lib/34293/2/VIII_NTK_2020_Zashko_B-Advantages_of_using_ASP_NET_141.pdf.

11. Мельник В. М. Використання Docker для створення відокремленого дослідницького середовища / В. М. Мельник, А. І. Нагорнюк. // Науковий журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництва". – 2019. – №35. – С. 2.

12. Umur C. Citus: Distributed PostgreSQL for Data-Intensive Applications / C. Umur, E. Ozgun, P. Sumedh., 2021. – 13 с. – (SIGMOD ’21).

13. Rider. Fast & powerful cross-platform .NET IDE [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/rider/.

14. DataGrip. Many databases, one tool [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/datagrip/.

ДОДАТОК А

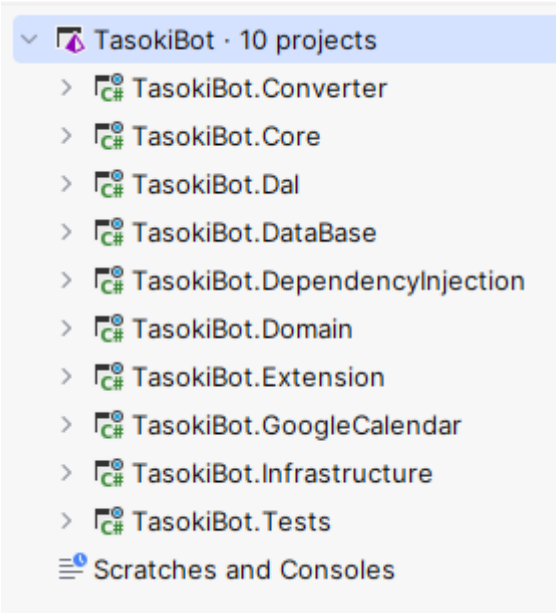


Рисунок 1. Структура проєкту TasokiBot

```
using ...

namespace TasokiBot.DataBase.Context;

12 usages  Pavlo Skydan +1 *
public class DataBaseContext : DbContext
{
    private readonly string _connectionString;

    1 usage  Pavlo Skydan *
    public DataBaseContext(string connectionString)
    {
        _connectionString = connectionString;
    }

    Pavlo Skydan +1
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseNpgsql(_connectionString);
    }

    3 usages  Pavlo Skydan
    public Task SaveChangesAsync()
    {
        return base.SaveChangesAsync();
    }

    2 usages  Palamar Afk
    public ValueTask<EntityEntry> AddAsync<T>(T entity)
    {
        return entity != null ? base.AddAsync(entity) : new ValueTask<EntityEntry>();
    }

    public DbSet<BotUser> User { get; set; }
    public DbSet<TasokiBot.Dal.Models.Task> Task { get; set; }
}
```

Рисунок 2. Клас контексту бази даних DataBaseContext

```
using ...

namespace TasokiBot.DependencyInjection;

Palamar Afk +1
public static class DependencyInjection
{
    1 usage  Palamar Afk +1
    public static void AddCoreDependencies(this IServiceCollection services, ConfigurationManager configurationManager)
    {
        IConfiguration configuration = new Configuration();
        services.AddTransient<IMainService, MainService>();
        services.AddTransient<IUserService, UserService>();
        services.AddTransient<ITaskService, TaskService>();
        services.AddTransient<IUserRepository, UserRepository>();
        services.AddTransient<ITaskRepository, TaskRepository>();
        services.AddTransient<IConfiguration, Configuration>();
        services.AddScoped<DataBaseContext>(_ => new DataBaseContext(configuration.GetConnectionString("PostgreSqlConnection", configurationManager)));
        services.AddTBotRedisLimiter(configuration.GetConnectionString("RedisConnection", configurationManager));
    }
}
```

Рисунок 3. Додавання залежностей через Dependency Injection

```
public class TaskRepository : ITaskRepository
{
    private readonly DataBaseContext _context;

    Palamar Afk
    public TaskRepository(DataBaseContext context)
    {
        _context = context;
    }

    0+1 usages  Palamar Afk
    public async Task SaveTaskAsync(TaskItem taskItem)
    {
        await _context.AddAsync( entity: taskItem.ToDal());
        await _context.SaveChangesAsync();
    }

    0+1 usages  Palamar Afk +1
    public async Task UpdateTaskAsync(TaskItem task)
    {
        _context.Update( entity: task.ToDal());
        await _context.SaveChangesAsync();
    }

    0+1 usages  Palamar Afk
    public Task<Dal.Models.Task?> GetLastTaskAsync()
    {
        return _context.Set<Dal.Models.Task>() // DbSet<Task>
            .AsNoTracking() // IQueryable<Task>
            .OrderByDescending(x :Task => x.CreatedOn) // IOrderedQueryable<Task>
            .FirstOrDefaultAsync(); // Task<Task?>
    }

    [SuppressMessage( category: "ReSharper.DPA",  checkId: "DPA0009: High execution time of DB command", MessageId = "time: 2205ms")]
    0+1 usages  Pavlo Skydan *
    public Task<List<Dal.Models.Task>> GetAllTasksNoSyncedAsync()
    {
        return _context.Set<Dal.Models.Task>() // DbSet<Task>
            .AsNoTracking().Where(x :Task => x.Status != TaskState.Synced).ToListAsync(); // Task<List<...>>
    }
}
```

Рисунок 4. Код репозиторію для роботи із об’єктами задач

```
using TasokiBot.Domain.Models;
using TasokiBot.Extension;

namespace TasokiBot.Infrastructure;

Palamar Afk +1
public static class TaskStateProcessor
{
    1 usage  Palamar Afk
    public static TaskItem AddTitle(this TaskItem taskItem, string? title)
    {
        taskItem.Title = title;
        taskItem.Status = TaskState.TitleAdded;
        return taskItem;
    }

    1 usage  Palamar Afk
    public static TaskItem AddDescription(this TaskItem taskItem, string? description)
    {
        taskItem.Description = description;
        taskItem.Status = TaskState.DescriptionAdded;
        return taskItem;
    }

    1 usage  Palamar Afk +1
    public static TaskItem AddDate(this TaskItem taskItem, DateTime? dateTime)
    {
        taskItem.DateOfCompletion = dateTime;
        taskItem.Status = TaskState.Configured;
        return taskItem;
    }
}
```

Рисунок 5. Статичний клас із методами розширення для об’єкту задачі

```
using ...

namespace TasokiBot.Dal.Models;

[Index(nameof(DateOfCompletion))]
12 usages  Palamar Afk +1  1 extension method  1 exposing API
public class Task : BaseEntity
{
    2 usages
    public long ChatId { get; set; }
    2 usages
    public string? Title { get; set; }
    2 usages
    public string? Description { get; set; }
    3 usages
    public DateTime? DateOfCompletion { get; set; }
    3 usages
    public TaskState Status { get; set; }
    2 usages
    public Guid? UserId { get; set; }

    1 usage  Palamar Afk +1
    public Task(
        Guid id,
        long chatId,
        DateTime createdOn,
        string? title,
        string? description,
        DateTime? dateOfCompletion,
        Guid? userId,
        TaskState status) : base(id, chatId, createdOn)
    {
        ChatId = chatId;
        Title = title;
        Description = description;
        DateOfCompletion = dateOfCompletion;
        UserId = userId;
        Status = status;
    }
}
```

Рисунок 6. DAL модель об’єкту задачі

ДОДАТОК Б

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ КАЛЕНДАРНОГО ПЛАНУВАННЯ ОСОБИСТИХ ЗАДАЧ МОВОЮ C# НА БАЗІ ПЛАТФОРМИ ASP.NET

Виконав студент 4 курсу
групи ПД-41
Скидан Павло Валерійович
Керівник роботи
К.т.н, доц Негоденко Олена Василівна

Київ - 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 2

Мета – спрощення процесу календарного планування особистих задач за рахунок його реалізації всередині месенджеру Telegram мовою C#.

Об'єкт – процес календарного планування особистих задач.

Предмет – програмне забезпечення календарного планування особистих задач.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати існуючі додатки та програмні засоби, які мають в собі схожий функціонал, та виявити їхні переваги та недоліки.
2. Сформувати функціональні та нефункціональні вимоги до застосунку на основі порівняльної характеристики аналогів.
3. Спроектувати архітектуру та розробити Telegram-бота відповідно до визначених потреб користувачів.
4. Провести функціональне тестування застосунку.

АНАЛІЗ АНАЛОГІВ

5

АНАЛІЗ АНАЛОГІВ

Характеристика	**Google Tasks**	**Todoist**	**Trello**	**AnyDo**	**Ok, Bob!**	**TasokiBot**
Створення, редагування, видалення задач	+	+	+	+	+	+
Додавання тегів, опису, файлів до задачі	+	+	+	+	-	+
Встановлення дати	+	+	+	+	+	+
Наявність інтеграції із Telegram	-	-	-	-	+	+
Інтеграція із Google Calendar	+	-	-	-	-	+
Синхронізація між пристроями	+	+	+	+	+	+

6

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

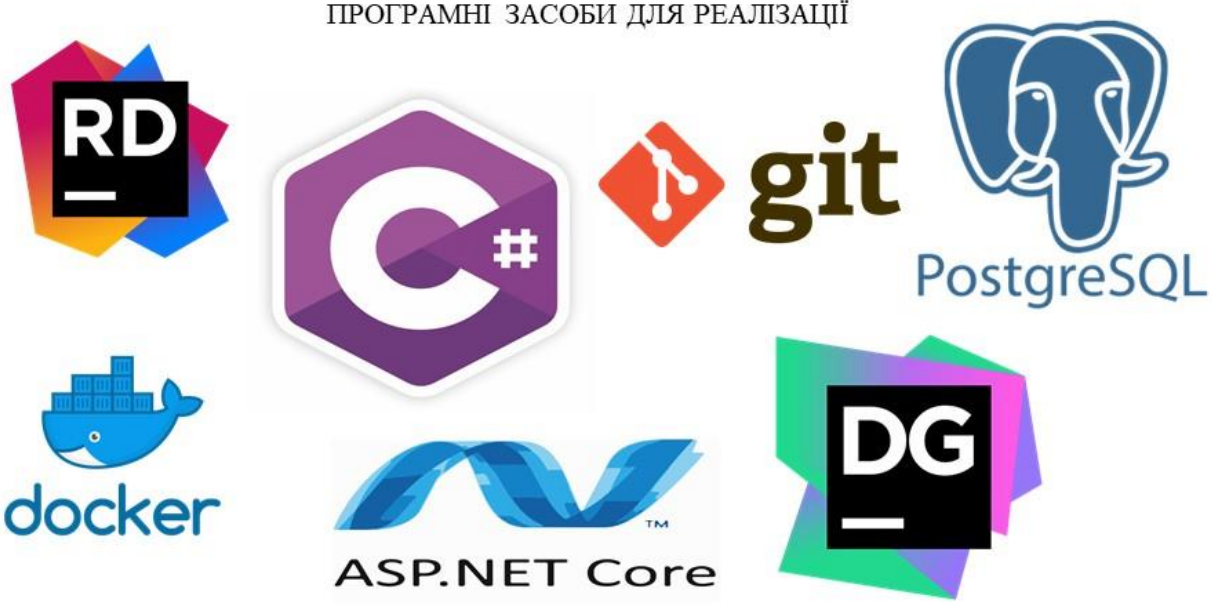
1. Створення задачі.
2. Валідація введеного користувачем тексту.
3. Синхронізація створених задач із Google Calendar.
4. Відображення поточних задач.
5. Видалення задачі.
6. Відправлення нагадувань через сповіщення.

Нефункціональні вимоги

1. Зрозумілість.
2. Надійність та стійкість.
3. Безпека.
4. Масштабованість.

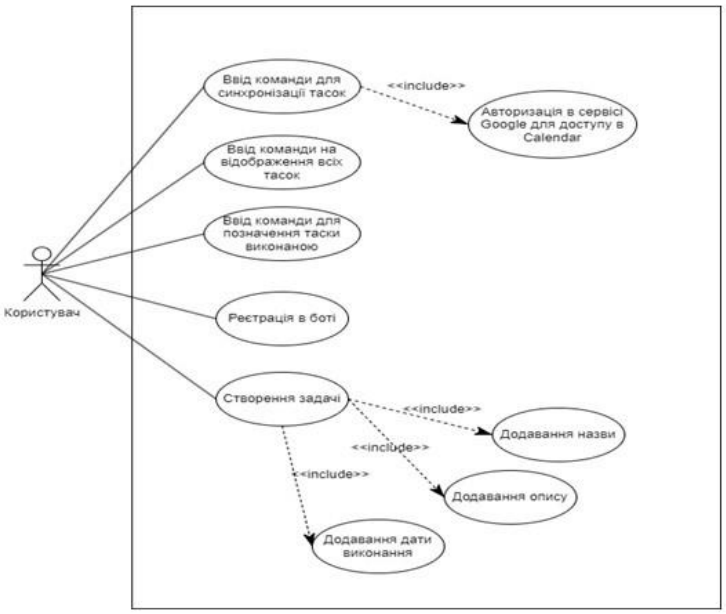
7

ПРОГРАМНІ ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ



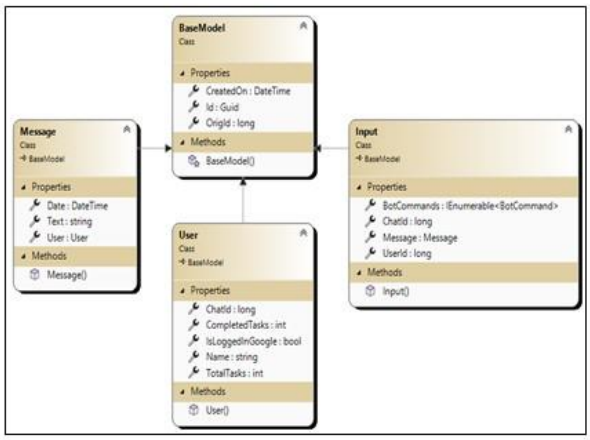
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

8

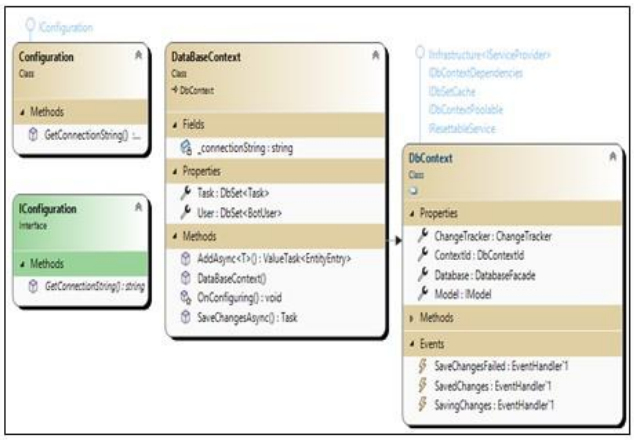


ДІАГРАМА КЛАСІВ

9

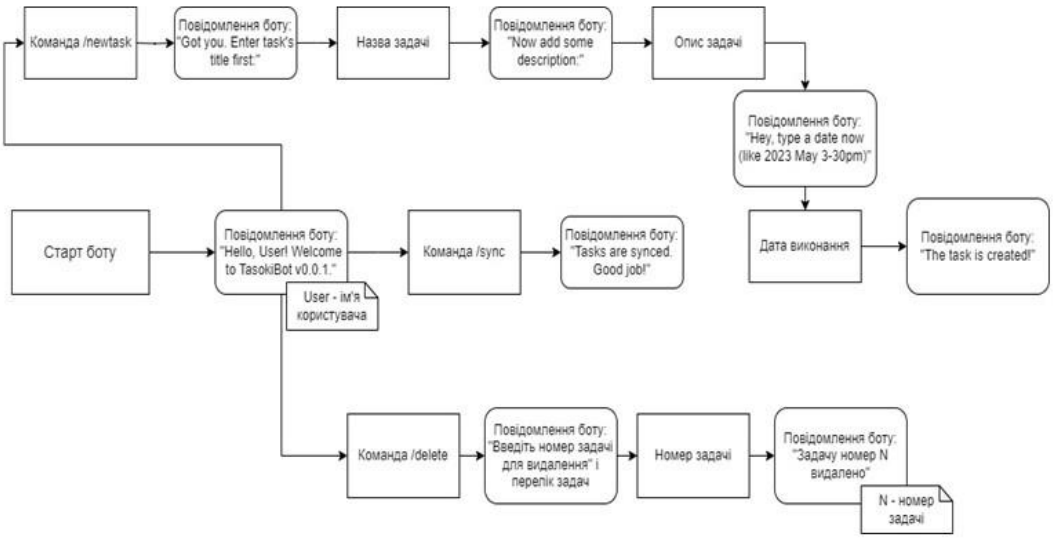


Діаграма класів - TasokiBot.Domain



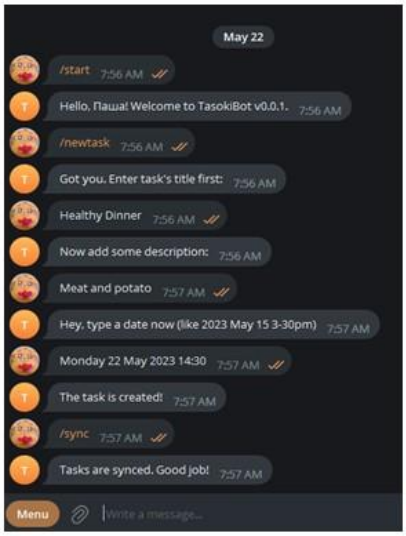
Діаграма класів - TasokiBot.Database

Структура чат-боту TasokiBot



10

Процесс заповнення інформації про задачу



Заповнення задачі та синхронизація

Елемент Google Calendar після синхронізації

Тестування

```
[Test]
public void ConvertToDateTimeAsync_ValidDateString_ReturnsDateTime()
{
    // Arrange
    string dateString = "2022-11-25 09:30:00";

    // Act
    DateTime? result = Utilities.ConvertToDateTimeAsync(dateString);

    // Assert
    Assert.IsNotNull(result);
    Assert.AreEqual( expected: new DateTime( year: 2022,  month: 11,  day: 25,  hour: 9,  minute: 30,  second: 00),  actual: result);
}

[Test]
public void ConvertToDateTimeAsync_InvalidDateString_ReturnsNull()
{
    // Arrange
    string dateString = "May 45 -3409 67:23";

    // Act
    DateTime? result = Utilities.ConvertToDateTimeAsync(dateString);

    // Assert
    Assert.IsNull(result);
}

[Test]
public void ConvertToDateTimeAsync_DateInThePast_ReturnsNull()
{
    // Arrange
    string dateString = "2020-01-01";

    // Act
    DateTime? result = Utilities.ConvertToDateTimeAsync(dateString);

    // Assert
    Assert.IsNull(result);
}
```

Юніт-тест роботи методу валідації дати в проєкті TasokiBot

13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Скидан П. Розробка програмних засобів для календарного планування особистих задач мовою c# на базі платформи ASP.NET / П. Скидан, О. Негоденко. – Київ: Всеукраїнська науково-технічна конференція Застосування програмного забезпечення в інфокомунікаціних технологіях. Збірник тез, 2023. – 157 с.

2. Скидан П. Мікросервісна архітектура при розробці телеграм ботів / П. Скидан, О. Негоденко. – Київ: Всеукраїнська науково-технічна конференція Застосування програмного забезпечення в інфокомунікаціних технологіях. Збірник тез, 2023. – 157 с.

14

ВИСНОВКИ

1. Проведено аналіз календарного планування задач. Визначено ключові проблеми, з якими зтикаються користувачі під час використання сторонніх додатків для такого виду планування. На основі цього встановлено функціональні та нефункціональні вимоги до додатку.
2. Розглянуто програмне забезпечення, яке може бути використано для календарного планування задач в якості аналогів: Google Tasks, Todoist, Trello, Any.do, Ok, Bob!. Порівняно їх функціонал, основні можливості та виявлено недоліки та переваги існуючих систем, а також інші показники, які відрізняють аналоги від розробленого додатку.
3. Проведено аналіз засобів розробки програмного забезпечення. Обрано мову програмування C# та фреймворк .NET Core, що забезпечують швидкість та ефективність у вирішенні задач інтеграції із Telegram та Google Calendar.
4. Розроблено архітектуру телеграм-бота TasokiBot для календарного планування задач. Реалізовано функціонал створення задачі, заповнення інформації про неї, синхронізації існуючих задач із Google Calendar.
5. Проведено функціональне тестування застосунку.

Дякую за увагу!