Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ «TODOLIST» З ВИКОРИСТАННЯМ FLUTTER, FAST API ТА POSTGRESQL

DEVELOPMENT OF A MOBILE APPLICATION "TODOLIST" USING FLUTTER, FAST API AND POSTGRESQL

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗс-21
Волошин Владислав Віталійович

Керівник:
Паливода Данило Ігорович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Волошину Владиславу Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка мобільного додатку «TodoList» з використанням Flutter, Fast API та PostgreSQL»*
Керівник роботи: *асистент Паливода Д. І.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Flutter, FastAPI, PostgreSQL, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз предметної області, аналіз вимог до системи, опис обраних технологій, реалізація додатку, тестування та оптимізація, розробка бази даних , захист інформації, розробка документації.*

5. Перелік графічного матеріалу: *5 рисунків*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Паливода Д. І.*		
Специфікація вимог до розробленої системи	*Паливода Д. І.*		
Розробка об`єкта проектування	*Паливода Д. І.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	2,53 %		
Академічна доброчесність	*Паливода Д. І.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	вик.
2	Аналіз проблеми розробки та впровадження об`єкту проектування	до 01.03.2025 р.	вик.
3	Обгрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	вик.
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	вик.
5	Практична реалізація об`єкта проектування та розробка бази даних	до 26.04.2025 р.	вик.
6	Тестування та налагодження об`єкта проектування	до 03.05.2025 р.	вик.
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	вик.

Здобувач вищої освіти _____ Владислав ВОЛОШИН

Керівник кваліфікаційної роботи _____ Данило ПАЛИВОДА

АНОТАЦІЯ

Волошин В. В. Розробка мобільного додатку «TodoList» з використанням Flutter, Fast API та PostgreSQL. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі здійснено аналіз існуючих рішень у сфері управління завданнями та сформульовано завдання для розробки мобільного додатку. В другому розділі проведено опис технологій та інструментів, що використовуються при розробці програмного забезпечення, а також розглянуто методи проектування. У третьому розділі описано реалізацію системи та її функціональність. У висновках підсумовано основні досягнення, а також зазначено можливі напрями для подальшого розвитку системи.

Ключові слова: управління завданнями, мобільний додаток, технології, програмне забезпечення, методи проектування, функціональність, розробка.

ABSTRACT

Voloshyn V. Development of a Mobile Application "TodoList" Using Flutter, Fast API and PostgreSQL. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions and a list of references.

The first chapter presents an analysis of existing solutions in the field of task management and formulates the objectives for developing a mobile application. The second chapter describes the technologies and tools used in the software development process, as well as the design methods. The third chapter outlines the implementation of the system and its functionality. The conclusions summarize the key achievements and also highlight potential directions for further system development.

Keywords: task management, mobile application, technologies, software development, design methods, functionality, development.

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ РІШЕНЬ У СФЕРІ РОЗРОБКИ ДОДАТКІВ ДЛЯ УПРАВЛІННЯ ЗАВДАННЯМИ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ 9
1.1 Аналіз сучасного стану проблеми 9
1.2 Постановка завдання на кваліфікаційну роботу бакалавра 12
Висновки до розділу 1 13
РОЗДІЛ 2СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ 15
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення 15
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання 19
Висновки до розділу 2 23
РОЗДІЛ 3РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ «TODOLIST» 25
3.1 Практична реалізація об’єкта проектування 25
3.2 Тестування та налагодження інформаційно-комп’ютерної системи 28
3.3 Розробка бази даних 31
3.4 Захист інформаційно-комп’ютерної системи 34
3.5 Розробка документації для програмного продукту 35
Висновки до розділу 3 38
ВИСНОВКИ 40
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 44

ВСТУП

Актуальність теми. У сучасному світі мобільні додатки стали невід'ємною частиною повсякденного життя. Завдання організації особистих справ, управління часом та ресурсами набувають все більшої важливості, оскільки користувачі шукають ефективні інструменти для планування. Одним з найбільш поширених типів додатків є додатки для ведення списку завдань (To-Do List). Вони дозволяють організувати робочий процес, створюючи список завдань і даючи можливість їх редагувати, позначати виконаними та видаляти. Проблеми ефективного управління особистими справами стають актуальними для більшості користувачів, що підвищує попит на подібні програми.

Метою цієї кваліфікаційної роботи бакалавра є розробка мобільного додатку для управління списком завдань на базі платформ Flutter, FastAPI та PostgreSQL, що дозволить користувачам легко створювати, редагувати, видаляти та відмічати виконані завдання.

Об'єктом кваліфікаційної роботи бакалавра є технології розробки мобільних додатків, зокрема, використання фреймворку Flutter для клієнтської частини та FastAPI з PostgreSQL для серверної частини.

Предметом кваліфікаційної роботи бакалавра є процес розробки та інтеграція мобільного додатку для управління завданнями, а також дослідження бази даних, яка забезпечує зберігання інформації про завдання та їх статуси.

Для досягнення поставленої мети були поставлені наступні завдання:

1) аналіз існуючих рішень у сфері мобільних додатків для управління завданнями;

1) сформулювати функціональні та нефункціональні вимоги до майбутньої системи;

2) вибір технологій та інструментів розробки додатку;

3) реалізація серверної частини з використанням FastAPI для обробки запитів, інтеграція клієнтської частини на Flutter з сервером через REST API;

4) тестування та налагодження програмного продукту;

5) створення бази даних для зберігання даних про завдання;
6) реалізація базових механізмів безпеки;
7) розробка документації користувача.

Мобільний додаток «TodoList» може бути використаний в будь-якій сфері діяльності, де необхідно організовувати та відслідковувати виконання завдань. Це може бути як особисте використання для планування повсякденних справ, так і використання в корпоративному середовищі для управління проектами, виконання завдань співробітниками.

На сьогодні існує багато різних рішень для управління завданнями, від простих мобільних додатків до складних корпоративних систем. Однак, незважаючи на це, користувачі часто стикаються з проблемою обмежених функцій або незручного інтерфейсу. Сучасні тенденції вказують на потребу у багатофункціональних, зручних та адаптивних рішеннях, що забезпечують синхронізацію на різних пристроях.

Популярними є рішення, що підтримують інтеграцію з іншими інструментами та платформами, такими як Google Calendar, Slack, Trello тощо. Багато сучасних додатків для управління завданнями використовують cloud-based технології для синхронізації даних між різними пристроями, що підвищує зручність для користувачів.

Галузь застосування включає як особисте використання, так і можливість застосування у корпоративних середовищах для організації роботи співробітників.

Цей проєкт є частиною широкої теми розвитку мобільних додатків для особистої продуктивності. Він враховує досвід та існуючі рішення на ринку додатків для завдань, але надає новий підхід до архітектури та інтеграції з сучасними вебтехнологіями для покращення ефективності управління завданнями.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ РІШЕНЬ У СФЕРІ РОЗРОБКИ ДОДАТКІВ ДЛЯ УПРАВЛІННЯ ЗАВДАННЯМИ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасному світі управління завданнями стало важливим елементом організації особистого часу, продуктивності на роботі та у багатьох інших сферах діяльності. З ростом цифрових технологій та мобільних платформ з’явилося багато різних рішень для створення і ведення списків завдань, що дозволяють користувачам організовувати свої завдання, позначати виконані та отримувати нагадування. Цей процес став ще більш важливим в умовах швидко змінюваного ритму життя та необхідності оптимізації часу.

Дослідження у сфері розробки додатків для управління завданнями показують різноманіття підходів та технологій для вирішення цієї задачі. У роботі A. Jaiswal та співавторів [1] описується мобільний додаток TaskCO для Android, що дозволяє користувачам планувати завдання та інтегрувати їх з календарем, що робить додаток зручним для повсякденного використання. Стаття S. H. Shama та співавторів [2] дає огляд популярних додатків для управління завданнями, таких як Evernote і Google Keep, визначаючи важливі функціональні можливості, як синхронізація та інтеграція з іншими сервісами, що є важливим для створення нових рішень.

Дослідження Д. Костюченка [3], яке фокусується на розробці веб-додатку на основі C# та ASP.NET, показує ефективність використання цих технологій для інтеграції та створення зручного інтерфейсу. Рішення на основі Node.js, представлене в роботі А. В. Курнакова та співаторів [4], демонструє переваги асинхронних технологій для швидкої обробки запитів і масштабованості. У дослідженні А. С. Чернецького та Д. В. Тимофієнка [5] підкреслюється важливість використання REST API для забезпечення гнучкості в інтеграції системи. Всі ці дослідження надають цінні рекомендації для розробки

ефективних і масштабованих рішень у сфері управління завданнями, підтверджуючи актуальність використаних у цьому проєкті технологій Flutter, FastAPI та PostgreSQL.

Загалом, в літературі описано широкий спектр підходів до розробки додатків для управління завданнями, серед яких можна виділити два основні напрямки: мобільні додатки та програмне забезпечення для настільних ПК. Зокрема, існують рішення для операційних систем Android та iOS, а також веб-сервіси. Додатки можуть бути розроблені для конкретних задач (наприклад, для планування робочого часу в компаніях) або для загального використання, де кожен користувач може самостійно налаштовувати список завдань.

Мобільні додатки дозволяють користувачам мати доступ до своїх завдань у будь-який час і з будь-якого місця. Багато додатків забезпечують автоматичну синхронізацію завдань між різними пристроями (смартфони, планшети, ПК). Відомі додатки, такі як Google Keep, Trello, Todoist та інші, підтримують інтеграцію з різними платформами для зручного додавання завдань з електронної пошти, календаря або інших інструментів для комунікації.

Проте існують і певні недоліки. Багато популярних додатків обмежують доступ до певних функцій або дають лише обмежений доступ до архіву завдань без підписки. Деякі додатки зберігають дані користувачів на своїх серверах, що може викликати занепокоєння щодо конфіденційності. Додатки з великою кількістю функцій можуть бути занадто складними для користувачів, яким потрібні лише основні можливості для управління завданнями.

Патентний пошук у цій галузі показує, що існує багато рішень для управління завданнями, але більшість з них мають обмежений функціонал або фокусуються на конкретних типах завдань. Наприклад, патенти на мобільні додатки для управління проектами описують системи, які дозволяють групам користувачів спільно працювати над завданнями, але не враховують особисту продуктивність та потребу в простих і доступних інтерфейсах для індивідуальних користувачів.

Серед популярних аналогів проектованої системи можна виділити додатки Todoist, Trello, Microsoft To Do, Google Tasks та інші. Ці додатки мають ряд функцій, таких як створення завдань, їх категоризація, встановлення строків виконання, отримання нагадувань, а також можливість синхронізації між різними пристроями.

Переваги:

1) Todoist – проста та інтуїтивно зрозуміла система з розширеними функціями планування;

2) Trello – зручне управління проектами, можливість додавання користувачів і організації командної роботи;

3) Google Tasks – інтеграція з іншими сервісами Google, зокрема, з Gmail і Google Calendar.

Недоліки:

1) Todoist – обмежений доступ до певних функцій у безкоштовній версії;

2) Trello – більш складний інтерфейс для простих завдань, призначений в основному для командної роботи;

3) Google Tasks – обмежений функціонал порівняно з іншими додатками для управління завданнями.

Незважаючи на наявність численних рішень для управління завданнями, існує потреба в новому додатку, який би поєднував простоту використання, доступність на всіх платформах та можливість зберігати дані локально або в захищеному хмарному середовищі. Проектований додаток буде орієнтований на індивідуальних користувачів, які шукають простий і надійний інструмент для організації своїх завдань.

Отже, доцільність створення нового мобільного додатку для управління завданнями з використанням Flutter, FastAPI та PostgreSQL є очевидною, оскільки він поєднує в собі кращі риси існуючих рішень і надає можливість задовольнити потреби користувачів, що шукають простоту та ефективність у роботі з особистими завданнями.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є розробка мобільного додатку для управління списком завдань із використанням Flutter, FastAPI та PostgreSQL. Завдання, що ставляться в межах роботи, охоплюють кілька важливих етапів, зокрема:

2) проаналізувати існуючі мобільні додатки та веб-системи для управління завданнями, визначити їх переваги та недоліки, а також сформулювати вимоги до майбутнього додатку, враховуючи існуючі прогалини на ринку;

3) сформулювати функціональні та нефункціональні вимоги до майбутньої системи;

4) вибрати відповідні технології та інструменти для розробки кожної частини системи;

5) реалізувати серверну частину додатку на базі FastAPI, налаштувати API для обробки запитів від клієнтської частини, а також забезпечити взаємодію з базою даних, інтегрувати клієнтську частину додатку, розроблену на Flutter, з сервером через API для взаємодії з базою даних, додавання, редагування, перегляду та видалення завдань;

6) провести тестування мобільного додатку та серверної частини, перевірити коректність функціонування всіх компонентів, забезпечити їх стабільну роботу;

7) розробити структуру бази даних для зберігання інформації про завдання, зокрема визначити таблиці та зв’язки між ними, а також механізм взаємодії з базою даних через API;

8) оцінити можливі загрози безпеці програми, впровадити необхідні заходи для захисту даних користувачів і забезпечення стабільної та безпечної роботи додатку;

9) створити детальну документацію для користувачів і розробників, описати функціональні можливості програми, навести інструкції з використання та налаштування середовища для розробників.

Завдання на кваліфікаційну роботу бакалавра визначають комплексний підхід до створення мобільного додатку для управління завданнями, що включає як аналіз існуючих рішень, так і розробку архітектури системи, створення бази даних, реалізацію серверної частини та інтеграцію з клієнтською частиною. Оцінка безпеки та тестування є важливими етапами для забезпечення стабільності та надійності програмного продукту. У результаті виконання роботи буде розроблено сучасний додаток, що задовольняє потреби користувачів у простому та ефективному управлінні завданнями, з урахуванням інноваційних технологій та вимог безпеки.

Висновки до розділу 1

Аналіз сучасного стану проблеми показав, що на ринку існують різноманітні рішення для управління завданнями, проте ці рішення мають певні недоліки, зокрема обмежену функціональність у безкоштовних версіях, складність у використанні для простих завдань та проблеми з конфіденційністю даних. Це створює попит на нову систему, яка поєднувала б простоту використання, високий рівень функціональності та забезпечувала б надійний захист даних користувачів.

Сучасні тенденції показують, що користувачі шукають зручні та ефективні інструменти для управління своїм часом, які підвищують продуктивність як в особистій сфері, так і в командній роботі. Попит на мобільні додатки для управління завданнями залишається високим, оскільки такі інструменти дозволяють знижувати рівень стресу та організовувати робочі процеси.

Розробка мобільного додатку для управління завданнями є актуальним завданням, оскільки наявні рішення не відповідають усім вимогам користувачів.

Користувачі потребують інструментів, які поєднуватимуть простоту використання, надійність та високий рівень безпеки, чого не завжди забезпечують існуючі додатки. Це доводить необхідність створення нового програмного продукту, який би задовольняв сучасні вимоги.

Огляд існуючих рішень дозволив чітко визначити функціональні вимоги до майбутнього додатку, що базуватимуться на досвіді використання існуючих продуктів та враховуватимуть усі їхні недоліки. Визначені потреби й функціональні можливості додатку будуть враховані при розробці програмного продукту, що забезпечить його актуальність та відповідність вимогам користувачів.

Отже, цей розділ окреслює необхідність створення нового додатку для управління завданнями, з урахуванням існуючих проблем і потреб, та обґрунтовує доцільність проведення даної роботи.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

У цьому підрозділі розглядаються основні вимоги до розроблюваного мобільного додатку для управління завданнями, а також методи проектування, які будуть застосовуватися для створення програми. Зокрема, буде обґрунтовано вибір моделі програмної системи, визначено функціональні та нефункціональні вимоги до продукту, створено специфікацію вимог та модель елементів системи, а також проведено аналіз отриманих результатів.

Для моделювання програмного забезпечення для управління завданнями було обрано модель клієнт-сервер, де клієнтська частина (мобільний додаток на Flutter) взаємодіє з серверною частиною (FastAPI), що зберігає дані в базі даних PostgreSQL. Така модель дозволяє ефективно обробляти запити від користувачів, зберігати їх завдання на сервері, а також підтримувати синхронізацію між різними пристроями користувачів.

Моделювання програмної системи буде виконано з використанням UML-діаграм, які чітко відображатимуть структуру додатку, взаємодію компонентів та бізнес-логіку. Вибір цієї моделі обумовлений необхідністю забезпечити масштабованість додатку, легкість обробки запитів і взаємодії з базою даних, а також можливість подальшого розширення функціоналу.

У процесі розробки додатку важливо врахувати вимоги, які необхідно виконати для задоволення потреб користувачів. Вимоги можна поділити на функціональні та нефункціональні.

Функціональні вимоги:

1) користувач має можливість створювати нові завдання, редагувати їхній опис, назву, а також позначати їх як виконані;

2) користувач може переглядати список завдань з можливістю фільтрації за статусом (активні, виконані);

3) користувач може видаляти завдання з бази даних;

4) дані про завдання мають зберігатися в базі даних PostgreSQL для подальшого використання і синхронізації між пристроями;

5) використовуються механізми аутентифікації для доступу до даних користувача, а також шифрування з’єднань через HTTPS.

Нефункціональні вимоги:

1) система повинна бути доступною 24/7;

2) система повинна бути здатною обробляти великий обсяг запитів, коли кількість користувачів збільшується;

3) інтерфейс має бути інтуїтивно зрозумілим для користувачів з різним рівнем досвіду;

4) додаток повинен працювати на всіх основних мобільних платформах (Android, iOS).

Для проектування системи будуть використовуватися UML-діаграми [6]. Use Case Diagram (рис. 2.1) для візуалізації функціональних можливостей додатку з точки зору користувача (створення завдання, редагування, видалення тощо). Користувач взаємодіє з додатком через різні варіанти використання: створення, перегляд, редагування та видалення завдань. Прецеденти взаємодіють із базою даних для збереження, оновлення та отримання інформації про завдання.

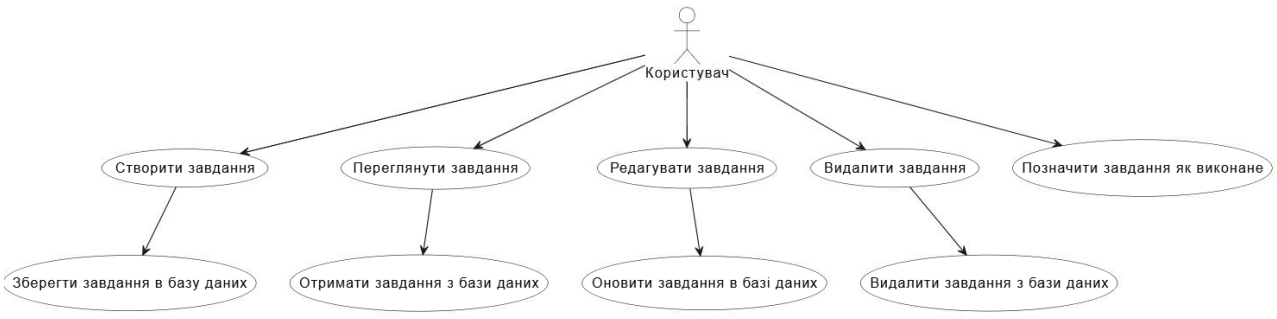

Рисунок 2.1 – Діаграма варіантів використання

Діаграма класів (рис. 2.2) для опису класів і їхніх взаємозв’язків у системі (Task, User, API-сервіс). Клас «Завдання містить» основні поля для завдання (id, назва, опис, виконано) та методи для створення, оновлення та видалення завдань. Клас «БазаДанихЗавдань» взаємодіє з базою даних для збереження, оновлення та видалення завдань.

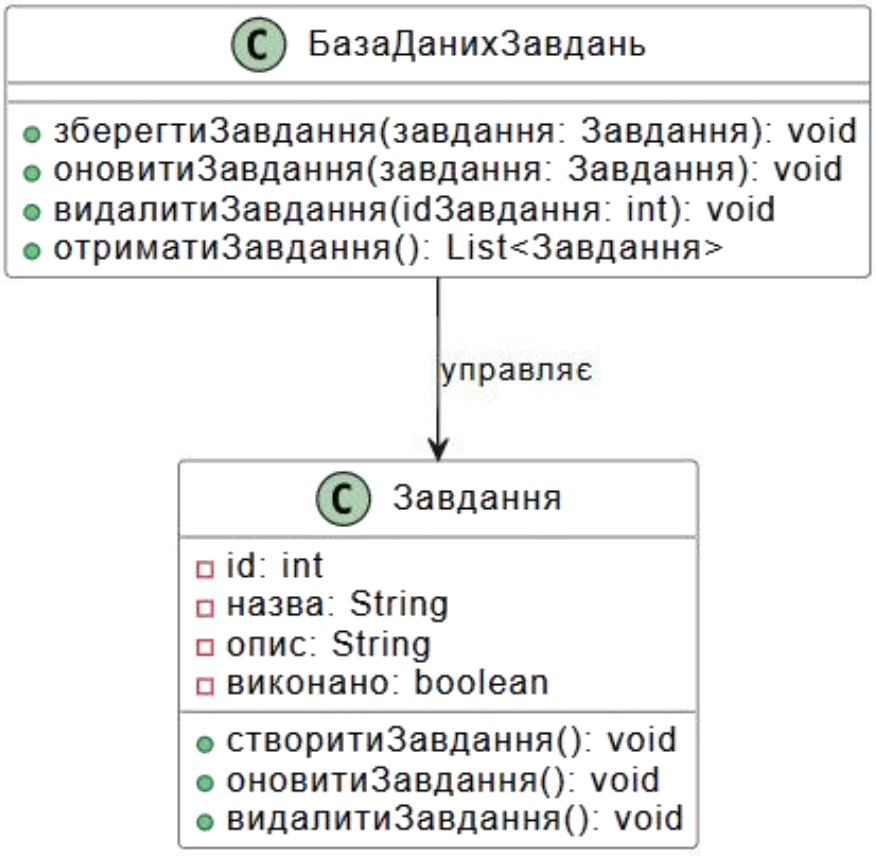

Рисунок 2.2 – Діаграма класів

Діаграма послідовності (рис. 2.3) для детального опису взаємодії між користувачем, клієнтським додатком і сервером при виконанні різних операцій (створення, редагування, видалення завдань).

Користувач вводить деталі завдання в мобільному додатку. Додаток передає ці дані на сервер FastAPI, який взаємодіє з базою даних для збереження нового завдання. Після підтвердження від бази даних сервер повертає інформацію про завдання в додаток.

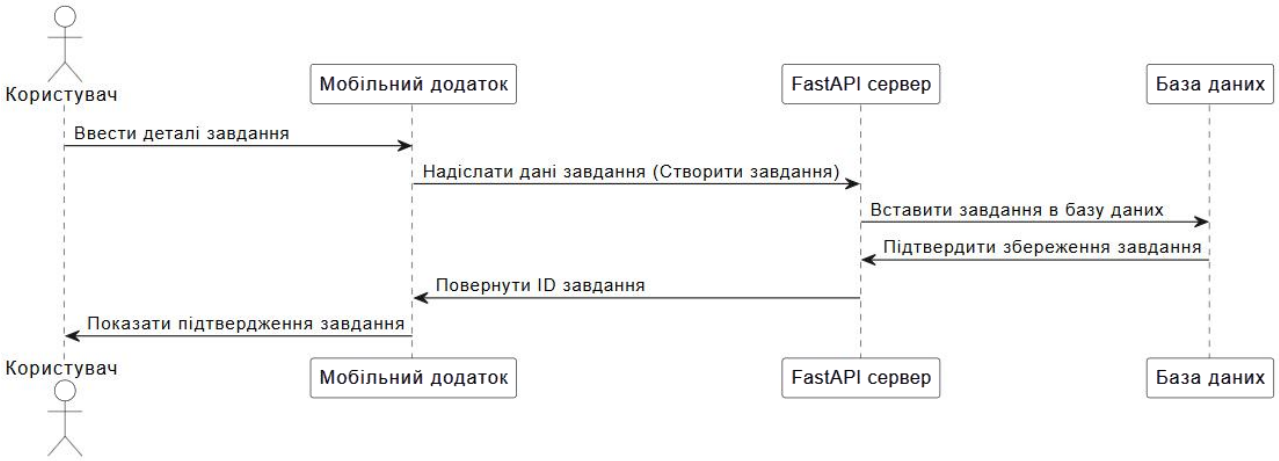

Рисунок 2.3 – Діаграма послідовності

Ці UML-діаграми допомагають чітко уявити структуру та поведінку мобільного додатку для управління завданнями. Діаграма варіантів використання описує функціональні можливості системи, діаграма класів показує структуру програмного забезпечення, а діаграма послідовності відображає взаємодію між компонентами під час виконання операцій. Ці діаграми служать основою для розробки, тестування та подальшого розвитку додатку.

Програмне забезпечення повинно виконувати наступні функції:

1) додавання нових завдань з описами та датами виконання;
2) редагування опису завдань та їхнього статусу;
3) можливість позначити завдання як виконане та перемістити його в архів;
4) можливість пошуку завдань за ключовими словами чи датами;
5) синхронізація даних між пристроями користувачів через API;
6) застосування HTTPS для забезпечення безпечних з’єднань та шифрування конфіденційної інформації.

UX/UI дизайн буде розроблений таким чином, щоб користувачі могли інтуїтивно взаємодіяти з додатком, виконуючи базові функції з управління

завданнями. Інтерфейс має бути простим, зрозумілим і зручним для користувачів будь-якого рівня.

Основні функціональні можливості інтерфейсу:

1) створення нового завдання через просту форму введення;
2) огляд активних та виконаних завдань у вигляді списку;
3) кнопки для редагування і видалення завдань;
4) перемикання між світлою і темною темою для зручності використання в різних умовах освітлення.

Технології інформаційного обслуговування та інформаційні потреби користувачів

Програмне забезпечення забезпечить високий рівень доступності даних через безпечний віддалений доступ. Інформаційні потреби користувачів полягають у можливості швидко створювати, переглядати та редагувати завдання, а також зберігати та синхронізувати їх між кількома пристроями.

На основі отриманих моделей можна стверджувати, що обрана архітектура програмного забезпечення з використанням клієнт-серверної моделі є оптимальною для цього типу додатку. Використання Flutter для клієнтської частини дозволить забезпечити кросплатформенність, а FastAPI для серверної частини – швидкість обробки запитів. У свою чергу, PostgreSQL забезпечить надійне зберігання даних та масштабованість системи.

Це рішення забезпечує ефективну роботу додатку при мінімальних витратах ресурсів, високій швидкості роботи і можливості розширення функціоналу в майбутньому.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

У цьому підрозділі здійснюється огляд інструментів, методів і алгоритмів, які будуть використовуватися для розробки мобільного додатку для управління завданнями. Зважаючи на специфіку завдання, важливо обрати інструменти, які

забезпечать ефективність, швидкість роботи та надійність системи. Оскільки додаток включає в себе мобільну частину, серверну частину та базу даних, кожен компонент вимагає використання оптимальних засобів і методів для вирішення поставлених задач.

Для розробки мобільного додатку обрано Flutter як основний фреймворк. Це сучасний інструмент для кросплатформної розробки мобільних додатків, що дозволяє створювати додатки для Android та iOS з одним кодом [7].

Переваги:

1) однією з головних переваг Flutter є можливість створювати додатки для різних платформ (iOS, Android) з використанням єдиного коду;

2) завдяки Hot Reload можна швидко змінювати код і негайно бачити результати;

3) Flutter має великий набір готових компонентів інтерфейсу, що дозволяє швидко створювати стильні та функціональні UI;

4) додатки на Flutter мають майже нативну продуктивність завдяки компіляції в машинний код.

Недоліки:

1) хоча Flutter активно розвивається, він може мати обмежену підтримку деяких специфічних функцій або сторонніх бібліотек;

2) додатки, створені на Flutter, можуть бути більшими за розміром порівняно з нативними додатками [8].

Для серверної частини обрано FastAPI. Це фреймворк для розробки веб-додатків, який працює на Python та підтримує створення високопродуктивних API [9].

Переваги:

1) FastAPI є одним із найшвидших фреймворків для створення API завдяки використанню стандарту ASGI та асинхронної обробки запитів;

2) FastAPI автоматично генерує документацію API за допомогою Swagger UI та ReDoc;

3) завдяки використанню Python 3.6+ типів, FastAPI надає зручну перевірку типів та автодоповнення;

4) FastAPI має простий і зрозумілий синтаксис, що дозволяє швидко розпочати розробку та підтримувати систему.

Хоча FastAPI стає дедалі популярнішим, він ще не настільки широко використовується, як більш відомі фреймворки, такі як Flask або Django, що може обмежити кількість доступних матеріалів та бібліотек [10].

Для зберігання даних вибрана PostgreSQL – реляційна система управління базами даних (СУБД), що має високу продуктивність і підтримує складні запити та транзакції [11].

Переваги:

1) PostgreSQL є однією з найбільш надійних та стабільних СУБД для великих та складних додатків;

2) PostgreSQL підтримує складні SQL-запити, транзакції, та індекси, що дозволяє ефективно працювати з великими обсягами даних;

3) база даних легко масштабується для обробки великих обсягів даних.

Недоліки:

1) для початківців налаштування PostgreSQL може бути складним, хоча для досвідчених розробників це не є значною проблемою;

2) PostgreSQL може вимагати більше ресурсів на великих навантаженнях порівняно з деякими іншими СУБД, наприклад, MySQL [12].

Для синтезу моделей програмного забезпечення будуть використовуватися стандартні методи об’єктно-орієнтованого моделювання (ООМ), серед яких можна виділити:

1) моделювання за допомогою різних UML-діаграм для моделювання структури класів, відносин між компонентами, а також взаємодії між об’єктами системи;

2) алгоритм CRUD (Create, Read, Update, Delete) для реалізації основних операцій над завданнями в додатку. Цей алгоритм забезпечує базову взаємодію

з базою даних для додавання, отримання, редагування та видалення завдань [13].

На рисунку 2.4 зображена діаграма активності, а саме: блок-схема додавання завдання для опису процесу управління завданнями користувача. Користувач вводить деталі завдання. Якщо введені дані є коректними, завдання зберігається в базі даних, і користувач отримує повідомлення про успішне збереження. Якщо дані некоректні, відображається повідомлення про помилку.

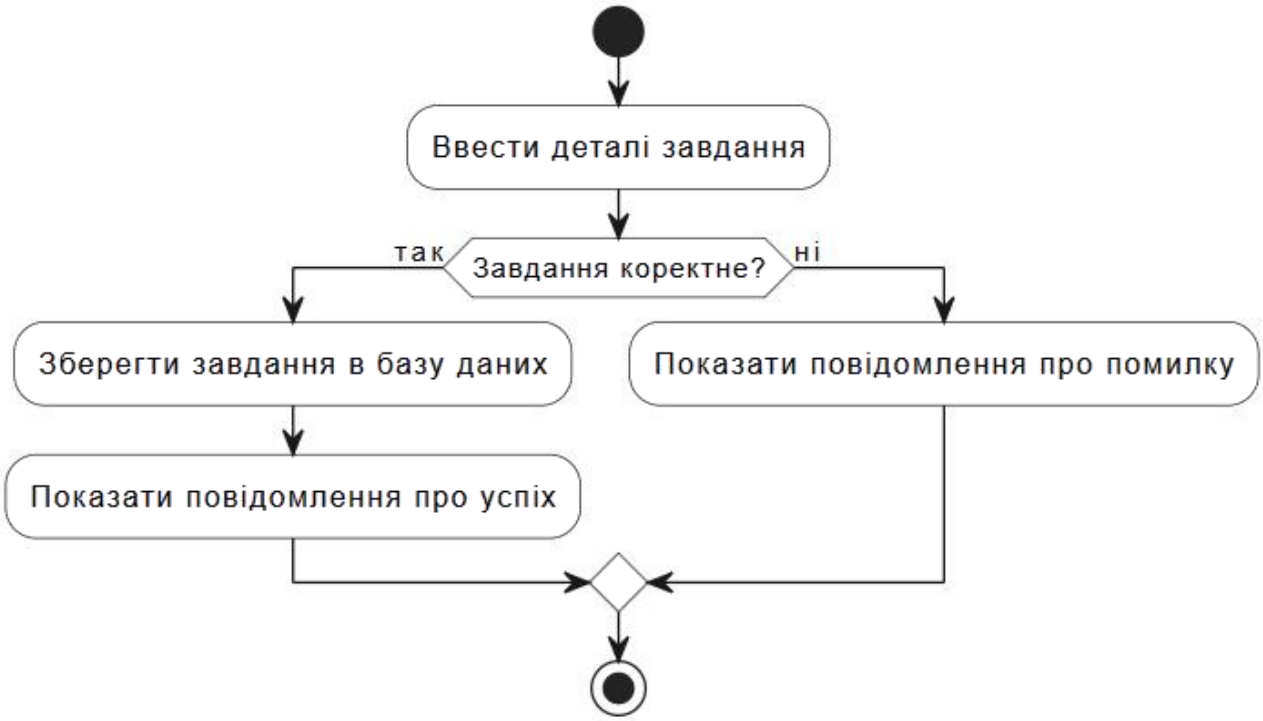

Рисунок 2.4 – Блок-схема додавання завдання

Опис алгоритму:

1) start – користувач вводить дані для нового завдання;

2) введення даних користувача – користувач заповнює форму для створення завдання;

3) перевірка коректності даних – якщо дані правильні, завдання додається в базу даних;

4) додати завдання в базу даних – якщо дані коректні, завдання зберігається в базі даних, і користувач отримує повідомлення про успіх;

5) показати повідомлення про помилку – якщо введені дані некоректні, відображається повідомлення про помилку;

6) stop – завершення процесу.

В цілому, обрані інструменти (Flutter для мобільної частини, FastAPI для серверної частини та PostgreSQL для бази даних) є оптимальними для розробки додатку для управління завданнями. Загалом, ці технології забезпечують ефективність, масштабованість і безпеку системи. Використання об’єктно-орієнтованого моделювання та алгоритмів CRUD дозволить створити стабільний і зручний додаток для користувачів, що задовольняє всі необхідні функціональні вимоги.

Висновки до розділу 2

В розділі 2 було проведено детальний аналіз та вибір інструментів і методів, що будуть використовуватися для розробки мобільного додатку для управління завданнями. Вибір інструментів, таких як Flutter для клієнтської частини, FastAPI для серверної частини та PostgreSQL для зберігання даних, є обґрунтованим та оптимальним для досягнення поставленої мети.

Flutter забезпечує швидку розробку кросплатформених мобільних додатків, що дозволяє створювати продуктивні та зручні інтерфейси. FastAPI дозволяє створити високопродуктивний сервер для обробки запитів, що важливо для масштабованих систем. PostgreSQL, у свою чергу, є надійною реляційною базою даних, яка забезпечує ефективне зберігання та обробку даних завдань.

Аналіз вимог до програмного забезпечення дозволив сформулювати функціональні та нефункціональні вимоги, що включають створення, редагування, перегляд і видалення завдань, а також захист даних користувачів і забезпечення їх синхронізації. Для реалізації цих вимог було вибрано відповідні технології та інструменти, що гарантують високу швидкість роботи та безпеку даних.

Моделювання системи за допомогою UML-діаграм дозволило наочно відобразити структуру додатку, взаємодію компонентів та алгоритми роботи з даними, що забезпечує зрозумілість і чіткість у процесі розробки. Вибір методів і підходів для реалізації цих завдань дозволяє ефективно досягти поставленої мети створення мобільного додатку, що задовольняє потреби користувачів у простому та безпечному управлінні завданнями.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ «TODOLIST»

3.1 Практична реалізація об’єкта проектування

Система складається з мобільного додатку на Flutter та серверної частини, що використовує FastAPI та PostgreSQL. Мобільний додаток взаємодіє з сервером через API, щоб виконувати CRUD операції (створення, читання, оновлення, видалення) завдань. Всі дані про завдання зберігаються у базі даних PostgreSQL.

Мобільний додаток реалізований за допомогою Flutter, що дозволяє створювати кросплатформні додатки. У додатку є можливість додавати нові завдання, переглядати їх, редагувати та видаляти. В лістингу 3.1 наведено фрагмент коду, який відповідає за додавання завдання.

Лістинг 3.1 – Додавання завдання на клієнті

```
Future<void> addTask() async {
  final task = taskController.text;
  if (task.isNotEmpty) {
    final response = await http.post(Uri.parse('$apiUrl/tasks/'),
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({'title': task, 'description': ''}));
    if (response.statusCode == 200) {
      taskController.clear();
      fetchTasks();
    }
  }
}
```

кінець лістингу 3.1

Користувач вводить завдання у текстове поле. Коли користувач натискає кнопку для додавання, відправляється HTTP POST запит на сервер з даними завдання (заголовок та опис). Після успішного додавання, викликається метод fetchTasks() для оновлення списку завдань.

Сервер на FastAPI обробляє запити від мобільного додатку та взаємодіє з базою даних PostgreSQL через SQLAlchemy. В лістингу 3.2 наведено фрагмент коду для створення нового завдання на сервері.

Лістинг 3.2 – Створення нового завдання на сервері

```
@app.post("/tasks/", response_model=TaskRead)
def create_task(task: TaskCreate, db: Session = Depends(get_db)):
    db_task = Task(title=task.title, description=task.description)
    db.add(db_task)
    db.commit()
    db.refresh(db_task)
    return db_task
```

кінець лістингу 3.2

При надходженні POST запиту на створення завдання, сервер отримує дані завдання через параметри TaskCreate. Дані завдання зберігаються в базі даних через об’єкт Task, після чого транзакція комітиться (зберігається). Після збереження, сервер повертає створене завдання у відповідь.

Для зберігання завдань використовується PostgreSQL. Клас Task описує структуру таблиці для зберігання завдань у базі даних. В лістингу 3.3 наведено фрагмент коду для моделі завдання.

Лістинг 3.3 – Модель завдання

```
class Task(Base):
    __tablename__ = "tasks"


    id = Column(Integer, primary_key=True, index=True)
    title = Column(String, index=True)
    description = Column(String, default="")
    is_done = Column(Boolean, default=False)
```

кінець лістингу 3.3

Модель Task відображає таблицю tasks в базі даних, з полями id, title, description та is_done. Id є унікальним ідентифікатором для кожного завдання. Поле is_done використовується для позначення, чи виконано завдання.

Алгоритм роботи програми можна описати таким чином:

1) користувач вводить завдання через мобільний додаток;

2) мобільний додаток надсилає дані завдання на сервер через API (HTTP POST запит);

3) сервер зберігає завдання в базі даних, а потім повертає підтвердження створення завдання;

4) додаток оновлює список завдань, щоб відобразити нове завдання.

На рисунку 3.1 зображено інтерфейс мобільного додатку для управління завданнями. Він містить поле для введення нового завдання, а також два списки: «Активні» та «Неактивні» завдання. Кожне завдання можна видалити за допомогою кнопки на правому боці.

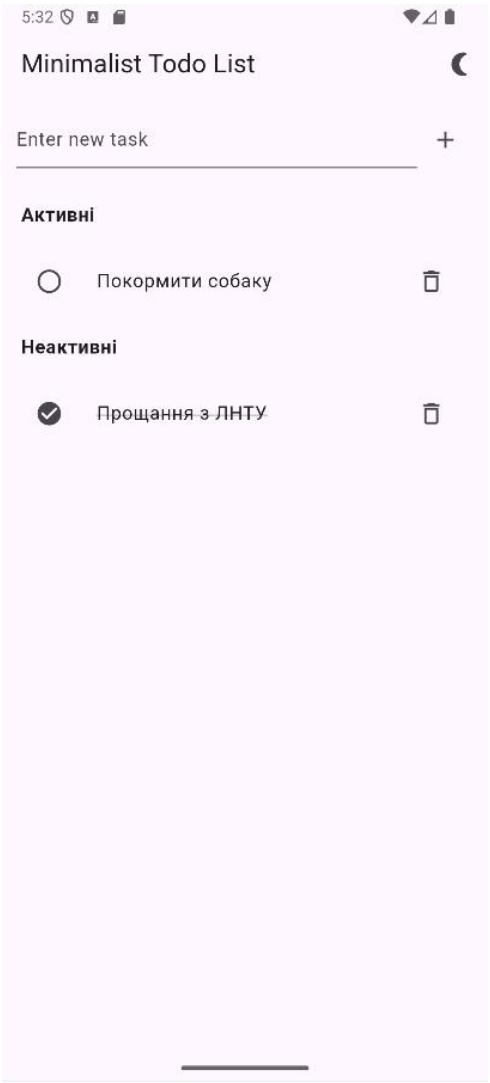

Рисунок 3.1 – Інтерфейс додатку

Предметною областю є управління завданнями, і основні задачі, які необхідно реалізувати, включають створення, редагування, видалення та перегляд завдань. Для вирішення цих задач застосовуються CRUD операції.

Створення завдання відбувається через мобільний додаток, який надсилає запит на сервер для створення нового завдання. Користувач може редагувати завдання, змінюючи його статус, наприклад, відзначити завдання як виконане або невиконане через відповідні кнопки в додатку. Видалення завдання здійснюється шляхом натискання кнопки видалення, що знаходиться поруч з кожним завданням. Список завдань відображається в мобільному додатку, де користувач може переглядати як активні, так і виконані завдання, що дозволяє йому ефективно управляти своїми справами.

В процесі розробки було використано реальні фрагменти коду, що демонструють практичну реалізацію основних функцій мобільного додатку та серверної частини. Користуючись Flutter, FastAPI та PostgreSQL, вдалося створити систему для ефективного управління завданнями, яка задовольняє вимоги користувачів щодо простоти використання, швидкості роботи та надійності.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування та налагодження є важливими етапами процесу розробки програмного забезпечення. Вони дозволяють виявити помилки на ранніх стадіях, забезпечити стабільність роботи програми та її відповідність вимогам. У межах цього розділу буде розглянуто методи тестування, налагодження програми, а також формування мінімальних системних вимог для роботи інформаційної системи.

Для забезпечення стабільності та надійності системи використовуються кілька основних методів тестування. Тестування юнітів – це тестування окремих одиниць коду – функцій та методів [14]. Для цього використовуються інструменти, такі як Flutter Test для тестування мобільного додатку та pytest для

серверної частини на FastAPI. Юніт-тести перевіряють правильність роботи окремих функцій і методів, наприклад, додавання нового завдання або зміни його статусу. Фрагмент коду для тестування (Flutter) наведено в лістингу 3.4.

Лістинг 3.4 – Фрагмент коду тестування на Flutter Test

```
test('Test Task creation', () {
  final task = Task(id: 1, title: 'Test Task', description: 'Test
Description');
  expect(task.title, 'Test Task');
  expect(task.description, 'Test Description');
});
```

кінець лістингу 3.4

Інтеграційне тестування – це тестування взаємодії між компонентами додатку, наприклад, перевірка правильної взаємодії між мобільним додатком, сервером і базою даних [15]. Для серверної частини тестування може включати перевірку API запитів на створення, отримання, оновлення та видалення завдань. Фрагмент коду для тестування API (FastAPI) наведено в лістингу 3.5.

Лістинг 3.5 – Фрагмент коду тестування API

```
def test_create_task(client):
    response = client.post('/tasks/', json={'title': 'Test Task',
'description': 'Test Description'})
    assert response.status_code == 200
    assert response.json()['title'] == 'Test Task'
```

кінець лістингу 3.5

Функціональне тестування – це тестування функціональності додатку з точки зору користувача, коли перевіряється, чи працюють всі функції програми згідно з вимогами, наприклад, чи правильно додаються та видаляються завдання, чи зберігається правильний статус завдання.

Тестування навантаження – це тестування, яке дозволяє оцінити, як система поводиться при великому навантаженні, тобто під час одночасної роботи великої кількості користувачів. Для цього можна використовувати інструменти для навантажувального тестування API, наприклад, Apache JMeter.

В свою чергу, налагодження – це процес пошуку і усунення помилок, які були виявлені в результаті тестування. Для ефективного налагодження використовується кілька підходів. Важливу роль у налагодженні відіграє логування. У Flutter можна використовувати вбудовані можливості для виведення логів через print() або більш розширене логування за допомогою бібліотек, як-от logger. У FastAPI можна використовувати стандартне логування Python для запису помилок і важливих подій, що допомагає швидко виявити причину несправностей.

Для мобільного додатку використовується стандартний дебагер Flutter DevTools, який дозволяє здійснювати покрокове виконання коду, перевіряти значення змінних і усувати помилки на етапі виконання. Для серверної частини можна використовувати дебагер pdb в Python для пошуку помилок у коді сервера.

Крім тестування в емуляторах та симуляторах, важливо тестувати додаток на реальних пристроях, щоб переконатися в коректній роботі з різними версіями ОС, типами екранів і характеристиками пристроїв.

Для коректної роботи інформаційної системи потрібно враховувати мінімальні вимоги до апаратного забезпечення та програмного середовища. Мобільний додаток (Flutter):

1) Android версія 5.0 (Lollipop) і вище, мінімум 2 ГБ оперативної пам’яті, 50 МБ вільного простору на диску;
2) iOS 11.0 і вище, мінімум 2 ГБ оперативної пам’яті.

Серверна частина (FastAPI):

1) операційна система: Linux, macOS або Windows;
2) Python 3.7 або вище;
3) база даних PostgreSQL 12 або вище;
4) процесор мінімум 2 ядра;
5) оперативна пам’ять мінімум 2 ГБ;
6) мережеві вимоги – підключення до Інтернету для доступу до сервера і бази даних.

В процесі тестування були виявлені деякі проблеми, наприклад, затримка при великих навантаженнях на сервер. Для вирішення цієї проблеми було оптимізовано запити до бази даних та додано кешування на сервері. Також була виявлена невідповідність UI на різних пристроях. Були внесені корективи в макети додатку, щоб забезпечити правильне відображення на різних екранах. Ці недоліки були усунені в результаті тестування та налагодження, що дозволило забезпечити стабільну та ефективну роботу програми.

Тестування та налагодження є невід'ємною частиною процесу розробки програмного забезпечення, і ці етапи дозволили забезпечити високий рівень якості додатку для управління завданнями. Використання різних методів тестування, включаючи юніт-тести, інтеграційні тести та тестування навантаження, дозволило виявити й усунути проблеми на ранніх етапах, що забезпечило стабільну роботу системи.

3.3 Розробка бази даних

Розробка бази даних є одним із основних етапів створення системи управління завданнями, оскільки вона дозволяє ефективно зберігати та обробляти дані, забезпечуючи їхню цілісність, доступність і безпеку. У рамках цього розділу описується методика створення схеми бази даних для програми управління завданнями, використовуючи PostgreSQL (як обраний СУБД), а також реалізація SQL-запитів для роботи з даними.

Для створення бази даних для системи управління завданнями була використана PostgreSQL, оскільки вона забезпечує високу продуктивність, надійність і можливість масштабування. Нижче описано, як була побудована схема бази даних для зберігання завдань у системі:

База даних включає таблицю tasks – це таблиця для зберігання завдань користувачів. Вона містить основні поля, необхідні для опису завдання:

1) id – унікальний ідентифікатор завдання (тип даних Integer, первинний ключ);

2) title – назва завдання (тип даних String);

3) description – опис завдання (тип даних String);

4) is_done – статус виконання завдання (тип даних Boolean), який визначає, чи виконано завдання.

SQL код для створення таблиці наведено в лістингу 3.6.

Лістинг 3.6 – SQL код для створення таблиці завдань

```
CREATE TABLE tasks (
  id SERIAL PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  description TEXT DEFAULT '',
  is_done BOOLEAN DEFAULT FALSE
);
```

кінець лістингу 3.6

Id визначається як SERIAL, що дозволяє автоматично генерувати унікальні значення для кожного завдання. Title є обов'язковим полем, тому для нього встановлено обмеження NOT NULL. Description має значення за замовчуванням пустий рядок, якщо опис не вказано. Is_done має значення за замовчуванням FALSE, що означає, що завдання не виконано, поки користувач не відзначить його як виконане.

Для взаємодії з базою даних використовуються стандартні SQL-запити, що дозволяють виконувати основні операції над завданнями: додавання, оновлення, видалення та отримання даних.

Для додавання нового завдання до бази даних використовується SQL-запит INSERT INTO. Цей запит дозволяє додавати нові записи у таблицю tasks (ліст. 3.7).

Лістинг 3.7 – Створення нового завдання

```
INSERT INTO tasks
(title, description, is_done)
VALUES ('Нове завдання', 'Опис нового завдання', FALSE);
```

кінець лістингу 3.7

Цей запит вставляє нове завдання з заданими title, description та is_done (встановлюється на FALSE, що означає, що завдання ще не виконано).

Для зміни статусу завдання (наприклад, для позначення його як виконаного) використовується SQL-запит UPDATE (ліст. 3.8).

Лістинг 3.8 – Оновлення статусу завдання

```
UPDATE tasks
SET is_done = TRUE
WHERE id = 1;
```

кінець лістингу 3.8

Цей запит оновлює поле is_done в таблиці tasks, встановлюючи його значення на TRUE для завдання з id = 1.

Для видалення завдання з бази даних використовується SQL-запит DELETE (ліст. 3.9).

Лістинг 3.9 – Видалення завдання

```
DELETE FROM tasks
WHERE id = 1;
```

кінець лістингу 3.9

Цей запит видаляє завдання з id = 1.

Для вибірки лише виконаних завдань (тобто з is_done = TRUE) використовується запит SELECT з фільтром WHERE (ліст. 3.10).

Лістинг 3.10 – Отримання виконаних завдань

```
SELECT * FROM tasks
WHERE is_done = TRUE;
```

кінець лістингу 3.10

Загалом, цей запит повертає всі завдання, де значення атрибуту is_done дорівнює TRUE.

Після створення схеми бази даних важливо провести тестування та перевірку її роботи. Перш за все, необхідно виконати запити для додавання, оновлення, видалення та отримання завдань, щоб переконатися, що всі операції працюють коректно. Це дозволить виявити будь-які помилки або неточності в запитах. Наступним етапом є тестування на реальних даних, що включає перевірку ефективності запитів при великих обсягах даних, що допомагає виявити потенційні вузькі місця у роботі бази даних. Також важливо провести оптимізацію запитів, використовуючи індекси для полів, які часто використовуються в запитах, таких як id, title або is_done, щоб підвищити швидкість обробки запитів.

Загалом, розробка бази даних для системи управління завданнями за допомогою PostgreSQL та SQLAlchemy забезпечує надійне зберігання даних та ефективну взаємодію з базою даних. Створення таблиці, написання SQL-запитів для основних операцій CRUD та налаштування оптимізації через індекси є основою для подальшої роботи додатку. Тестування та налагодження бази даних гарантують стабільну та швидку роботу системи.

3.4 Захист інформаційно-комп’ютерної системи

У процесі розробки мобільного додатку для управління завданнями було реалізовано кілька основних механізмів захисту, які забезпечують безпеку системи, захист даних користувачів та стабільну роботу додатку.

Для авторизації та автентифікації користувачів використовується механізм JWT (JSON Web Tokens). Після того як користувач вводить свої облікові дані, сервер перевіряє їх, і в разі успішної автентифікації генерується токен, який передається користувачеві. Цей токен використовується для подальших запитів до захищених маршрутів API. Це гарантує, що тільки авторизовані користувачі можуть отримати доступ до своїх даних.

Також для базового захисту безпечної передачі даних між мобільним додатком і сервером використовується SSL/TLS шифрування через HTTPS. Усі

запити, що надсилаються від мобільного додатку до серверу, проходять через захищене з’єднання, що унеможливлює перехоплення або зміну даних під час передачі.

Також у системі використовуються стандартні методи для захисту паролів користувачів. Паролі зберігаються в базі даних у вигляді хешів з використанням алгоритму bcrypt. Це гарантує, що навіть при доступі до бази даних паролі залишаються непотрібними для розкриття, оскільки вони не зберігаються у відкритому вигляді.

Додатково, захист від атак, таких як DDoS, реалізовано через Rate Limiting. Це обмежує кількість запитів від одного користувача за певний період часу, що допомагає запобігти перевантаженню серверу та зменшити ймовірність успішних атак. Ці механізми забезпечують захист даних користувачів і стабільну роботу системи, захищаючи її від основних загроз і атак.

3.5 Розробка документації для програмного продукту

У цьому розділі наведені мінімальні системні вимоги, необхідні для роботи розробленої системи, а також кроки, які потрібно виконати для встановлення та налаштування програмного продукту, зокрема для веб-додатків.

Для ефективної роботи розробленої системи управління завданнями необхідно дотримуватись певних мінімальних системних вимог. Мобільний додаток, створений за допомогою Flutter, має вимогу до операційної системи Android версії 5.0 (Lollipop) або вище, або iOS версії 11.0 і вище. Для коректної роботи додатку необхідно мати мінімум 2 ГБ оперативної пам’яті та не менше 50 МБ вільного простору на диску для встановлення додатку.

Що стосується серверної частини, то вона може працювати на операційних системах Linux, macOS або Windows. Для запуску серверу потрібна версія Python 3.7 або вище, а також платформа для серверного

хостингу, що підтримує ASGI (наприклад, Uvicorn). База даних для зберігання інформації повинна бути PostgreSQL версії 12 або вище. Для стабільної роботи серверу необхідний процесор мінімум з 2 ядрами, 2 ГБ оперативної пам’яті та від 100 МБ вільного місця на диску для зберігання серверних файлів і бази даних. Для користувачів мобільного додатку та веб-сервісу надані довідкові матеріали.

Інструкція з використання мобільного додатку. Мобільний додаток для управління завданнями дозволяє користувачам створювати нові завдання, змінювати їх статус і видаляти їх. Для створення нового завдання необхідно ввести назву в текстове поле та натиснути кнопку додавання. Завдання автоматично з’являється в списку «Активні». Користувач може змінювати статус завдання, позначаючи його як виконане чи невиконане, натискаючи відповідні кнопки поруч із завданням. Для видалення завдання потрібно натискати на іконку кошика, що знаходиться біля кожного завдання.

Крім того, додаток підтримує зміну темної та світлої теми інтерфейсу. Для цього достатньо натискати іконку в верхньому правому куті додатку, щоб перемикатися між темною та світлою темою, що дає користувачам можливість налаштувати вигляд додатку відповідно до своїх уподобань.

Для синхронізації даних між різними пристроями необхідно увійти в систему за допомогою облікового запису (якщо така функція передбачена). Дані автоматично синхронізуються між усіма пристроями, на яких користувач входить у систему, забезпечуючи доступ до актуальної інформації в будь-який час.

Інструкція для адміністратора серверної частини. Процес налаштування серверної частини на платформі FastAPI передбачає кілька основних етапів. Спочатку необхідно створити середовище для Python, встановивши відповідні пакети через pip. Далі налаштовується сервер для запуску API з використанням Uvicorn. Після налаштування середовища та встановлення залежностей, сервер можна запустити командою uvicorn main:app --reload, що дозволить розгорнути FastAPI на вашому сервері.

Для підключення до бази даних PostgreSQL необхідно налаштувати рядок підключення в конфігураційному файлі. Використовуваний драйвер для цього – SQLAlchemy, що дозволяє працювати з базою даних через ORM. Важливо, щоб база даних була налаштована і готова до підключення, а також необхідно забезпечити правильну роботу запитів через перевірку підключення за допомогою тестових запитів після налаштування.

Для налаштування SSL/TLS для захищених з'єднань на сервері необхідно отримати сертифікат від сертифікаційного центру (або використовувати самопідписаний сертифікат для тестування). Після отримання сертифіката потрібно налаштувати сервер для обробки HTTPS-запитів замість HTTP, змінивши конфігурації сервера. Це забезпечить шифрування всіх переданих даних між клієнтом та сервером, гарантуючи конфіденційність і цілісність інформації.

Рекомендації щодо вирішення поширених проблем. Однією з найбільш поширених проблем є відсутність доступу до бази даних. Це може бути спричинено неправильними налаштуваннями підключення або проблемами з мережею. Для вирішення цієї проблеми слід перевірити рядок підключення до бази даних у конфігураційному файлі та переконатися, що база даних запущена і доступна. Також варто перевірити налаштування фаєрволів чи мережевих політик, що можуть блокувати доступ до сервера бази даних.

Ще однією поширеною проблемою є збої в аутентифікації користувачів, наприклад, коли токен не є дійсним. Якщо користувач отримує помилку через неправильний токен, слід перевірити механізм генерації та перевірки токенів. Можливо, токен не був згенерований належним чином або вийшов з ладу через закінчення терміну дії. Для виправлення цієї проблеми необхідно переконатися, що сервер правильно перевіряє токен і що термін його дії належним чином обмежений. Для полегшення цього процесу варто впровадити автоматичне оновлення токенів при їх завершенні.

Отже, розробка супровідної документації для програмного продукту включає надання мінімальних системних вимог, створення довідкових

матеріалів для користувачів, а також опис процесу встановлення та налаштування як для мобільного додатку, так і для серверної частини. Кроки, що стосуються налаштування серверу, бази даних та інтеграції функцій виклику довідки, забезпечують зручність у роботі з системою та її подальшу супровідну підтримку.

Висновки до розділу 3

У розділі 3 було описано процес практичної реалізації, тестування, налагодження та захисту інформаційно-комп'ютерної системи, що включає розробку мобільного додатку для управління завданнями, серверної частини та бази даних.

Під час реалізації об'єкта проектування були використані ефективні методи розробки, зокрема Flutter для створення кросплатформеного мобільного додатку, FastAPI для серверної частини, а також PostgreSQL для зберігання даних. Всі ці інструменти дозволяють створити систему, яка працює швидко, надійно та зручна для користувачів.

Проведене тестування та налагодження забезпечили правильність роботи всіх функцій програми, зокрема створення, редагування, видалення та перегляд завдань. Для перевірки системи були використані юніт-тести, інтеграційне тестування та навантажувальні тести, що дозволило виявити та усунути потенційні проблеми до випуску системи.

У розділі також описано механізми захисту системи, включаючи автентифікацію користувачів за допомогою JWT, шифрування даних за допомогою SSL/TLS, захист від декомпіляції коду та хешування паролів за допомогою алгоритму bcrypt. Всі ці заходи сприяють забезпеченню високого рівня безпеки та захисту персональних даних користувачів.

Документація для користувачів була створена з урахуванням основних функціональних можливостей системи, а також надано детальне керівництво з установлення та налаштування системи на сервері та мобільних пристроях.

Отже, можна зробити висновок, що розділ 3 продемонстрував успішне застосування сучасних технологій для створення безпечної, надійної та зручної системи, призначеної для управління завданнями, що відповідає усім поставленим вимогам.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено мобільний додаток для управління завданнями, що поєднує сучасні технології кросплатформної розробки та серверного програмування. Вибір Flutter для створення додатку та FastAPI для серверної частини дозволив забезпечити високу продуктивність та ефективну взаємодію компонентів системи. Для зберігання даних використано PostgreSQL, що дозволяє масштабувати систему та гарантує надійність зберігання інформації.

Аналіз виконаних завдань показав, що розроблена система успішно вирішує проблему управління завданнями, надаючи користувачам інтуїтивно зрозумілий інтерфейс для створення, редагування, перегляду та видалення завдань. Завдяки реалізованим алгоритмам CRUD та можливості синхронізації даних між пристроями, додаток забезпечує зручність у використанні та доступ до актуальної інформації з будь-якого пристрою.

Було проведено детальний аналіз існуючих рішень у галузі управління завданнями, а також описано основні вимоги та завдання для розробки нового програмного продукту. Виявлено, що багато існуючих рішень задовольняють потреби користувачів, однак вони можуть бути складними у використанні та мають обмежену функціональність. Завдання цієї роботи полягало у створенні зручного та простого додатку для управління завданнями, зокрема для мобільних пристроїв, з можливістю синхронізації даних і підтримкою високої продуктивності.

Було здійснено всебічний аналіз вимог до розробленого програмного забезпечення, що включає як функціональні, так і нефункціональні вимоги. Після детального вивчення існуючих рішень було визначено основні функції, які повинні бути реалізовані в системі, зокрема: можливість створення, редагування, видалення та перегляду завдань. Окрему увагу було приділено зручності інтерфейсу, що відповідає сучасним стандартам UX/UI-дизайну, а

також інтеграції з базою даних для ефективного зберігання та обробки інформації.

Вибір засобів і методів для реалізації проекту було здійснено на основі аналізу сучасних технологій. Для клієнтської частини був обраний Flutter, оскільки він дозволяє створювати кросплатформні додатки, що знижує витрати часу на розробку та забезпечує високу продуктивність. Для серверної частини було вибрано FastAPI, оскільки цей фреймворк забезпечує швидке створення API та підтримує асинхронність, що дозволяє обробляти запити з високою швидкістю. Для зберігання даних обрана PostgreSQL, оскільки вона є потужною реляційною базою даних з можливістю масштабування та надійністю. Вибрані інструменти дозволяють створити ефективну систему управління завданнями, що відповідає всім вимогам.

Було описано процес практичної реалізації основних компонентів системи для управління завданнями. Мобільний додаток був реалізований з використанням Flutter, а серверна частина була розроблена за допомогою FastAPI. Використання SQLAlchemy для роботи з базою даних PostgreSQL дозволило ефективно організувати зберігання та обробку даних завдань. Реалізовані основні CRUD операції (створення, читання, оновлення, видалення) для завдань, що забезпечують зручний інтерфейс для користувачів.

Тестування системи підтвердило її ефективність та надійність. Проведено юніт-тести, інтеграційні тести та тестування на навантаження. Це дозволило виявити та усунути помилки на ранніх етапах розробки, що забезпечило стабільну роботу системи. Впровадження Rate Limiting для захисту від DDoS-атак значно підвищило безпеку системи, а проведене тестування на великих обсягах даних виявило можливості для подальшої оптимізації.

Розробка бази даних була успішно завершена з використанням PostgreSQL для зберігання даних про завдання. Моделювання таблиць через SQLAlchemy дозволило зручно працювати з реляційними даними, а також забезпечити швидкий доступ до інформації про завдання користувачів.

Оптимізація запитів через індекси дозволила підвищити швидкість виконання запитів і забезпечити ефективне використання ресурсів системи.

Вжиті заходи захисту системи, включаючи JWT для авторизації, SSL/TLS для шифрування даних, хешування паролів за допомогою bcrypt та захист від DDoS-атак через Rate Limiting, забезпечили високий рівень безпеки та захисту персональних даних користувачів. Всі заходи реалізовані відповідно до сучасних стандартів безпеки, що гарантує конфіденційність та цілісність даних.

Документація для користувачів була розроблена з урахуванням необхідності надання чітких інструкцій щодо використання додатку та налаштування серверної частини. Інструкції з встановлення, налаштування та використання системи дозволяють легко інтегрувати продукт у будь-якому середовищі. Довідка для користувачів інтегрована в систему, що дає змогу швидко отримати необхідну інформацію без необхідності звертатися до зовнішніх джерел.

Розробка системи управління завданнями продемонструвала успішне застосування сучасних інструментів та технологій для створення ефективного програмного продукту. Процес розробки, тестування, захисту та документування дозволив створити надійну та безпечну систему, що відповідає вимогам користувачів та сучасним стандартам у галузі управління завданнями. Система забезпечує зручний інтерфейс для користувачів, високу продуктивність і надійний захист даних, що робить її перспективною для застосування в різних сферах діяльності.

З точки зору світових тенденцій, реалізований проєкт відповідає поточним вимогам до ефективних інструментів для особистої продуктивності та організації роботи. Подібні системи вже широко використовуються в бізнесі, освіті та в особистих цілях для організації повсякденних задач. Розроблена система може бути використана в багатьох сферах, таких як управління проектами, навчання, особиста продуктивність, а також в корпоративному середовищі для координації завдань і планування.

Програмний продукт може бути використаний в різних галузях, де важливе управління завданнями та контроль за їх виконанням. Це включає сфери, як-от управління проектами, планування в компаніях, а також у сферах, де необхідна індивідуальна організація роботи, наприклад, в освітніх закладах для планування навчальних завдань.

Подальші шляхи вдосконалення можуть включати розширення можливостей для командного використання системи, розробку більш гнучких механізмів для налаштування інтерфейсу користувача та інтеграцію з хмарними сервісами для автоматичної синхронізації даних. Це дозволить значно підвищити функціональність додатку та зробити його ще більш корисним для користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. TaskCO. Android App for Task Management / A. Jaiswal et al. 2022 5th International Conference on Advances in Science and Technology (ICAST). Mumbai. India. 2-3 December 2022. URL: https://doi.org/10.1109/icast55766.2022. 10039511 (date of access: 01.03.2025).

2. Review on Existing Note-Taking and Task-Management Apps / S. H. Shama et al. International Journal of Advances in Computer Science and Technology. 2024. Vol. 13, no. 2. P. 59-64. URL: https://doi.org/10.30534/ijacst/2024/021322024 (date of access: 01.03.2025).

3. Костюченко Д. Розробка веб-додатку для управління завданнями (task management system) з використанням C# та ASP. NET. Інформаційно-комунікаційні технології в освіті. 2024. № 12. 3 с.

4. Курнаков А. В., Закаблук І. Ю., Здоренко Ю. М. Розробка веб-додатку для управління задачами користувачів на основі платформи Node. js (Doctoral dissertation. Національний університет «Полтавська політехніка імені Юрія Кондратюка»). 2025. С. 496-497.

5. Чернецький А. С., Тимофієнко Д. В. Розробка мобільного додатку для управління завданнями Rest-орієнтованого веб-сервісу. Молодь і індустрія 4.0 в XXI столітті: матеріали XX Міжнар. форуму молоді, 4-5 квіт. 2024 р. Харків: ДБТУ. 2024. С. 258.

6. UML Diagrams in Software Engineering Research: A Systematic Literature Review / H. Koç et al. Proceedings. 2021. Vol. 74, no. 1. P. 13. URL: https://doi.org/10.3390/proceedings2021074013 (date of access: 15.03.2025).

7. Flutter – Build apps for any screen. URL: https://flutter.dev/ (date of access: 17.03.2025).

8. Tashildar A., Shah N., Gala R., Giri T., Chavhan P. Application development using flutter. International Research Journal of Modernization in Engineering Technology and Science. 2020. Vol. 2, no. 8. P. 1262-1266.

9. FastAPI. URL: https://fastapi.tiangolo.com/ (date of access: 17.03.2025).

10. Lubanovic B. FastAPI. O'Reilly Media, Inc., 2023. 280 p.

11. PostgreSQL. URL: https://www.postgresql.org/ (date of access: 17.03.2025).

12. Makris A., Tserpes K., Spiliopoulos G., Zissis D., Anagnostopoulos D. MongoDB Vs PostgreSQL. A comparative study on performance aspects. GeoInformatica. 2021. no. 25. P. 243-268.

13. Що таке CRUD простими словами: функції, переваги та приклади. Highload.tech – медіа для розробників. URL: https://highload.tech/uk/shho-take- crud- prostimi- slovami- funktsiyi- perevagi- ta- prikladi/ (дата звернення: 18.03.2025).

14. Khorikov V. Unit testing principles, practices, and patterns. Simon and Schuster. Simon and Schuster. 2020. 304 p.

15. Що таке інтеграційне тестування? Типи, процес і впровадження. ZAPTEST. URL: https://www.zaptest.com/uk/що-таке-інтеграційне-тестування-глиб (дата звернення: 03.05.2025).