Національний технічний університет України

Київський політехнічний інститут ім. Ігоря Сікорського

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**ЗВІТ**

**ПРО ПРОХОДЖЕННЯ ПЕРЕДДИПЛОМНОЇ ПРАКТИКИ**

Студента(ки) 4 курсу ІТ-04 групи

Спеціальності 121 «Інженерія програмного забезпечення»

Чапчи Святослава Олександровича

(прізвище, ім’я, по батькові студента)

Термін практики з « 15 » квітня 2024 р. по « 19 » травня 2024 р.

База практики: ТОВ «ІНМЕДІА»

(назва підприємства)

Керівник від підприємства

старший інженер

(науковий ступінь, вчене звання, посада)

Закревський І. Г.

(прізвище, ім’я, по батькові, підпис)

Керівник практики від кафедри ІПІ

\_\_доцент кафедри

(науковий ступінь, вчене звання, посада,)

Родіонов П. Ю.

(прізвище, ім’я, по батькові, підпис)

**Київ-2024 р.**

**Пояснювальна записка  
до дипломного проєкту**

на тему: Веб-застосунок для впорядкування активностей і підтримки емоційного стану

КПІ.ІТ-0425.045440.02.81

Київ – 2024

Зміст

[Вступ 5](#_Toc166880883)

[1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 7](#_Toc166880884)

[1.1 Аналіз предметної області 7](#_Toc166880885)

[1.2 Аналіз існуючих рішень 9](#_Toc166880886)

[1.2.1 Аналіз відомих програмних продуктів 9](#_Toc166880887)

[1.2.2 Аналіз відомих алгоритмічних та технічних рішень 13](#_Toc166880888)

[1.3 Опис бізнес-процесів 17](#_Toc166880889)

[1.4 Постановка задачі 21](#_Toc166880890)

[Висновки до розділу 21](#_Toc166880891)

[2 РОЗРОБЛЕННЯ вимог до програмного забезпечення 23](#_Toc166880892)

[2.1 Варіанти використання програмного забезпечення 23](#_Toc166880893)

[2.2 Аналіз системних вимог 37](#_Toc166880894)

[2.3 Розроблення функціональних вимог 37](#_Toc166880895)

[2.4 Розроблення нефункціональних вимог 41](#_Toc166880896)

[Висновки до розділу 41](#_Toc166880897)

[3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 43](#_Toc166880898)

[3.1 Архітектура програмного забезпечення 43](#_Toc166880899)

[3.2 Обґрунтування вибору засобів розробки 46](#_Toc166880900)

[3.3 Конструювання програмного забезпечення 48](#_Toc166880901)

[3.4 Аналіз безпеки даних 60](#_Toc166880902)

[Висновки до розділу 61](#_Toc166880903)

[4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 63](#_Toc166880904)

[4.1 Аналіз якості ПЗ 63](#_Toc166880905)

[4.2 Опис процесів тестування 63](#_Toc166880906)

[4.3 Опис контрольного прикладу 64](#_Toc166880907)

[Висновки до розділу 64](#_Toc166880908)

[5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 65](#_Toc166880909)

[5.1 Розгортання програмного забезпечення 65](#_Toc166880910)

[5.2 Супровід програмного забезпечення 65](#_Toc166880911)

[Висновки до розділу 65](#_Toc166880912)

[ВИСНОВКИ 66](#_Toc166880913)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 67](#_Toc166880914)

[ДОДАТКИ 71](#_Toc166880915)

Перелік умовних позначень

|  |  |  |
| --- | --- | --- |
| IDE | – | Integrated Development Environment – інтегроване середовище розробки. |
| API | – | Application programming interface, прикладний програмний Інтерфейс |
| SDK | – | Software development kit |
| IT | – | Інформаційні технології |
| DAL | – | Data Access Layer |
| BLL | – | Business Logic Layer |
| PL | – | Presentation Layer |
| СУБД | – | Система управління базою даних |
| MVC | – | Model View Controller – патерн програмування |
| CSS | – | Cascading Style Sheets |
| JSON | – | JavaScript Object Notation – запис об’єктів JavaScript |
| SQL | – | Structured Query Language |
| HTML | – | HyperText Markup Language, стандартизована мова  розмітки документів для перегляду веб-сторінок у  браузері |

Вступ

У наш час з'являється все більше соціальних мереж та додатків, які створюють для користувача зручні умови у вигляді різного роду систем рекомендацій та практичного функціоналу. У той же час маркетологи з усіх сфер шукають більш вишукані способи, як зацікавити та задовольнити клієнта. Враховуючи це, а також те, що кількість завдань постійно зростає, люди часто стикаються з проблемою управління часом та збереженням балансу між роботою та особистим життям. Це призводить до стресу, вигоряння та зниження загального рівня життя. Саме тому і зростає потреба в інструментах, що допомагають підвищувати продуктивність та забезпечують емоційний добробут, і тому впровадження таких інструментів стає все більш актуальним.

Наукові дослідження підтверджують ефективність цифрових інструментів для підвищення продуктивності та підтримки емоційного стану. Наприклад, дослідження Гарвардської школи медицини показують, що регулярна рефлексія і ведення щоденника сприяють зниженню рівня стресу та покращенню емоційного благополуччя [1]. Інші дослідження демонструють, що трекінг звичок допомагає користувачам встановлювати та досягати особистих цілей, а застосунки з мотивуючими цитатами можуть значно підвищити рівень мотивації та натхнення [2].

Можливі сфери застосування веб-застосунку для впорядкування активностей та підтримки емоційного стану є надзвичайно широкими. Він може бути корисним для будь-якої категорії користувачів, від студентів, що прагнуть покращити свою успішність, до професіоналів, які шукають способи підвищити продуктивність та знизити рівень стресу на роботі. Застосунок може знайти своє місце у сфері охорони здоров'я, де його можуть використовувати як інструмент для підтримки психотерапевтичного процесу. Крім того, він може бути корисним для коучів та консультантів з особистісного розвитку, що допомагають своїм клієнтам досягати поставлених цілей та покращувати якість життя.

Метою цього проєкту є покращення продуктивності людей через розробку зручної платформи, яка дозволяє ефективно організовувати та керувати своїми активностями, а також спостерігати за емоційним станом. Ця платформа спрямована на надання користувачам інструментів для планування задач, розвитку корисних звичок і відмови від шкідливих звичок, а також надання можливості аналізувати свій прогрес і емоційне самопочуття.

Для досягнення поставленої мети необхідно виконати наступні завдання: провести детальний аналіз предметної області та існуючих рішень, сформулювати та проаналізувати функціональні та нефункціональні вимоги до програмного забезпечення, розробити високорівневу архітектуру застосунку, здійснити аналіз та вибір інструментів розробки, зконструювати та реалізувати серверну та клієнтську частини, провести аналіз якості реалізованого програмного забезпечення, а також розгорнути застосунок у хмарному середовищі.

# ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

## Аналіз предметної області

Веб-застосунки для впорядкування активностей і підтримки емоційного стану набувають все більшої популярності в сучасному світі. Такі продукти є комплексними, адже вони об`єднують у собі функціональність різних застосунків таких як: менеджери задач, трекери звичок, як поганих так і корисних, та інструменти для самоаналізу. Тим самим допомагаючи користувачам організувати свій час, слідкувати за прогресом та підтримувати ментальне здоров’я.

Однією з головних функцій таких застосунків є трекінг звичок, як позитивних, так і негативних. Звичка – це автоматизована поведінка, яка повторюється регулярно і часто без свідомого зусилля . Розробка застосунків для трекінгу звичок базується на дослідженнях у галузі психології, які показують, що ведення записів про свої звички допомагає покращити самоусвідомлення та дисципліну [2]. Відповідно до теорії Джеймса Кліра, автора книги "Атомні звички", малими, поступовими змінами можна досягти великих результатів у розвитку нових звичок і відмові від старих [3].

Застосунки для управління задачами – це програмні рішення, за допомогою яких зручно делегувати завдання та контролювати їх своєчасне виконаня. У таких програмах часто використовують матрицю Ейзенхауера для допомоги користувачам у пріоритизації задач. Матриця ділить задачі на чотири квадранти: термінові та важливі, не термінові, але важливі, термінові, але не важливі, і ні термінові, ні важливі. Такий підхід допомагає користувачам зосередитися на найважливіших завданнях, уникаючи прокрастинації та стресу [4].

Інструменти для самоаналізу дозволяють користувачам відстежувати свій прогрес, оцінювати досягнення та розуміти власні емоційні реакції на різні події. Такі засоби часто представлені у вигляді збору та вивіду статистики продуктивності, та опитувань щодо власного почуття. Такий підхід допомагає користувачам краще розуміти себе, виявляти патерни поведінки та емоцій, а також приймати обґрунтовані рішення щодо поліпшення свого життя та здоров'я.

Недоліки сучасних застосунків часто включають спеціалізацію на одній з сфер саморозвитку, занадто загальні рекомендації та відсутність гнучкості у налаштуваннях під індивідуальні потреби користувачів. Існують окремі застосунки для менеджменту задач, для відстеження звичок та опитування користувачів. Через що, користувачи змушені мати декілька застосунків одразу, які не пов’язані між собою. Вони можуть не враховувати специфіку різних типів завдань і звичок, що призводить до зниження мотивації користувачів. Крім того, багато застосунків мають обмежені можливості для аналізу даних і надання зворотного зв’язку, що ускладнює оцінку прогресу та прийняття рішень.

Покращення ситуації в цій галузі можливо через агрегацію декількох засобів, через вдосконалення користувацького інтерфейсу та функціональності для спрощення налаштувань і використання. Інтеграція з іншими сервісами, такими як календарі та додатки для нотаток, може зробити використання застосунку більш зручним і корисним для користувачів. Крім того, важливо забезпечити гнучкість у налаштуваннях, щоб користувачі могли адаптувати застосунок під свої індивідуальні потреби.

У рамках свого дипломного проєкту було обрано шлях агрегації таких засобів як: менеджмент звичок, менеджмент задач, а також інструменти для самоаналізу. Для покращення користувацького досвіду також буде розроблено зручний користувацький інтерфейс. Це рішення дозволить користувачам ефективніше організовувати свій час і покращувати свій емоційний стан без складних налаштувань і додаткових функцій, які можуть ускладнити використання застосунку.

## Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку для впорядкування активностей та підтримки емоційного стану. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

### Аналіз відомих програмних продуктів

Для аналізу існуючих програмних продуктів було створено перелік критеріїв, розроблений на основі детального вивчення предметної області.

* застосунок має надавати змогу керувати задачами;
* застосунок має надавати змогу керувати корисними звичками;
* застосунок має надавати змогу керувати шкідливими звичками;
* застосунок має надавати змогу керувати списком бажань;
* застосунок має надавати змогу проходити опитування з самопочуття та керувати ними;
* застосунок має надавати змогу керувати цитатами та виводити їх на головну сторінку;
* застосунок має надавати рекомендації, якщо користувач не знає чим зайнятись;
* застосунок сумісний з більшістю сучасних платформ та має зручний інтерфейс;

В результаті аналізу були виявлені наступні сучасні веб-застосунки для організації активностей та підтримки емоційного стану:

#### Quitzilla

Quitzilla – це мобільний застосунок, призначений для відстеження та підтримки користувачі у процесі відмови від шкідливих звичок. Застосунок надає інструменти для відстеження прогресу, мотиваційні цитати та нагадування, що допомагають користувачам залишатися мотивованими і досягати свої цілей [5].

Переваги:

* дозволяє користувачам бачити свій прогрес у реальному часі, включаючи кількість днів без шкідливої звички;
* пропонує мотиваційні цитати та досягнення;
* має простий і зрозумілий інтерфейс;
* можливість підключення до спільноти користувачів, що підтримує та ділиться досвідом;

Недоліки:

* деякі функції можуть бути доступні тільки у преміум-версії;
* застосунок допомогає користувачам тільки зі шкідливими звичками;
* деяким користувачам може не вистачати запропонованих мотиваційних інструметів для успішної відмови від звичок;

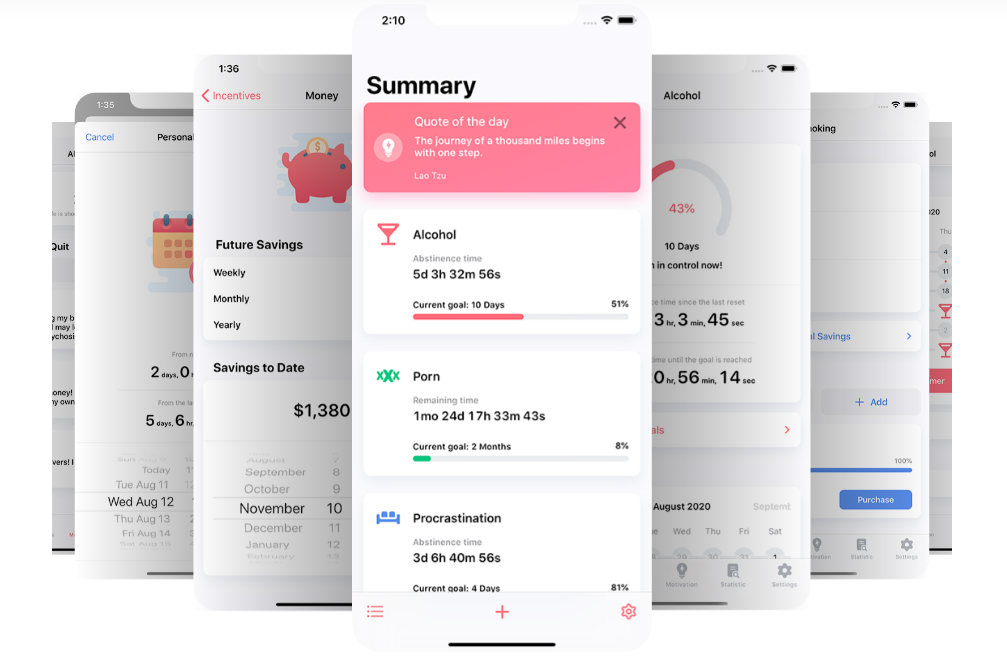


Рисунок 1.1 – Мобільний застосунок Quitzilla[5]

#### Todoist

Todoist – це популярний веб-застосунок для управління завданнями та проєктами, який допомагає користувачам організовувати свій час і підвищувати продуктивність [6].

Переваги:

* доступний на різних пристроях, що дозволяє користувачам синхронізувати свої завдання на всіх пристроях;
* має простий та зрозумілий інтерфейс;
* можливість співпраці над проєктами з іншими користувачами;
* підтримує інтеграцію з іншими інструментами, такими як Google Calendar, Dropbox, Slack;

Недоліки:

* деякі функції можуть бути доступні тільки у преміум-версії;
* застосунок допомогає користувачам тільки з менеджментом задач;
* деяким користувачам може знадобитися час, щоб повністю освоїти всі можливості та функції;
* вартість преміум-підписки може здатися завеликою, особливо якщо їм потрібні лише деякі з додаткових функцій;

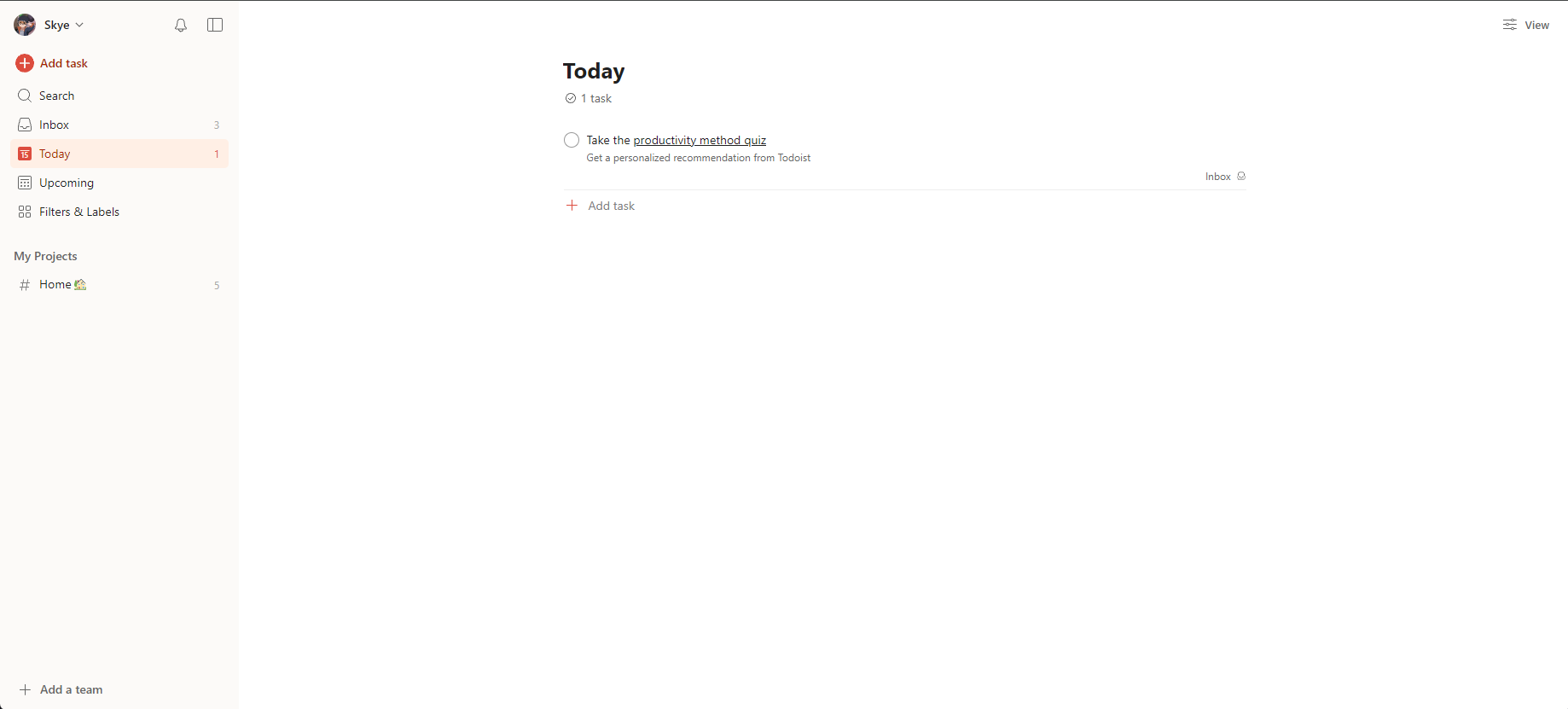


Рисунок 1.2 – Головна сторінка веб-застосунку Todoist[6]

#### DailyHabits

DailyHabits – це веб-застосунок, призначений для відстеження та розвитку корисних звичок. Він допомагає користувачам формувати нові звички та підтримувати мотивацію через регулярне відстеження та аналітику [7].

Переваги:

* доступний на різних пристроях, що дозволяє користувачам синхронізувати свої завдання на всіх пристроях;
* має простий та зрозумілий інтерфейс;
* надає детальную статистику та графіки, що дозволяють бачити прогрес у досягненні встановлених цілей;
* дозволяє встановлювати нагадування для кожної звички, що допомагає не забуати про виконання запланованих завдань;

Недоліки:

* деякі функції можуть бути доступні тільки у преміум-версії;
* застосунок допомогає користувачам тільки з менеджментом задач;
* деяким користувачі можуть вважати, що варіанти налаштування застосунку є недостатньо гнучкими для їх потреб;

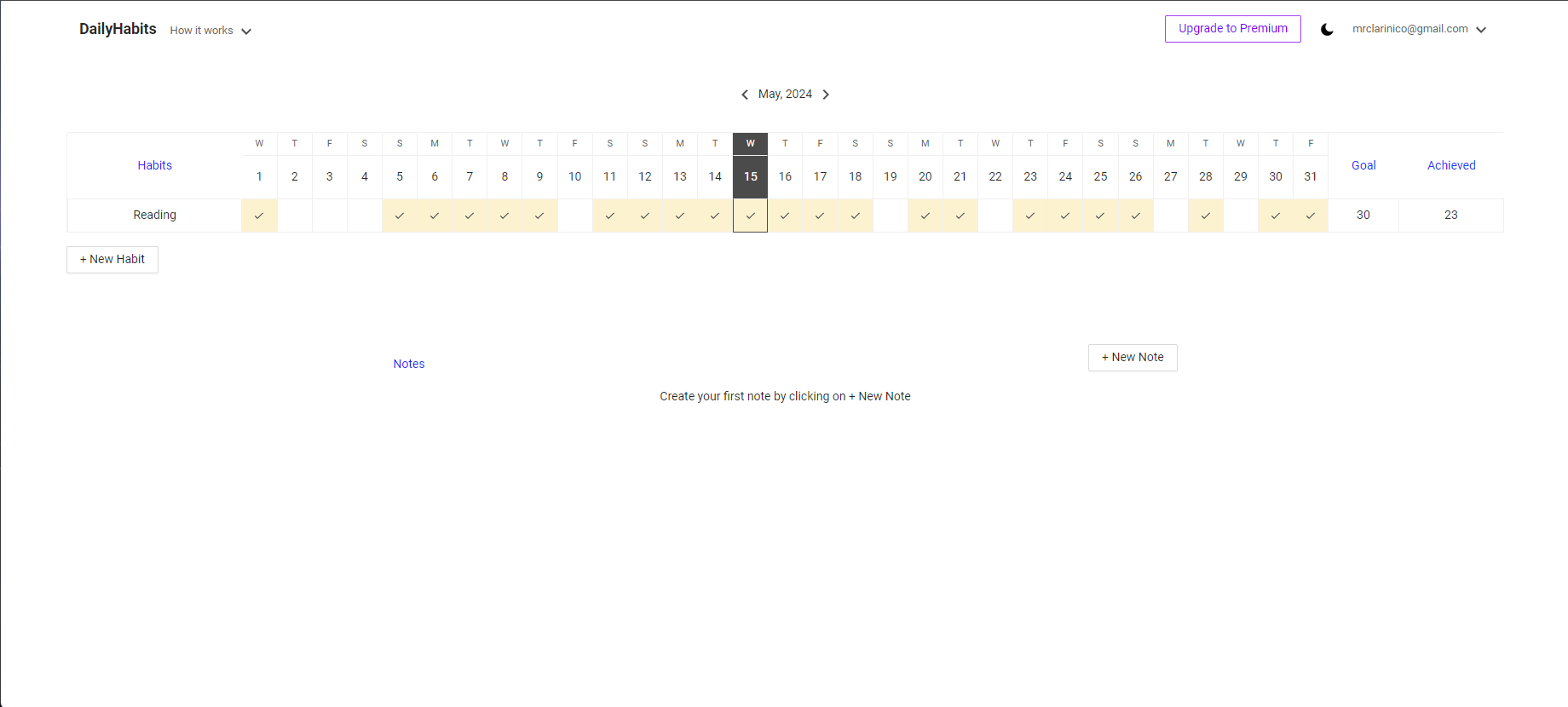


Рисунок 1.3 – Головна сторінка веб-застосунку DailyHabits[7]

У таблиці 1.1 представлено порівняння існуючих застосунків. З цього можна зробити висновок, що застосунок bbettr, який планується розробити, пропонує більш широкий спектр можливостей, включаючи різноманітні види активностей, рекомендації для користувачів, а також інструменти для саморефлексії.

Таблиця 1.1 – Порівняння з аналогом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функціонал | Дипломний проєкт (bbettr) | Quitzilla | Todoist | DailyHabits |
| Керування задачами | Присутнє | Відсутнє | Присутнє | Відсутнє |
| Керування корисними звичками | Присутнє | Відсутнє | Відсутнє | Присутнє |
| Керування шкідливими звичками | Присутнє | Присутнє | Відсутнє | Відсутнє |
| Керування списком бажань | Присутнє | Відсутнє | Присутнє | Присутнє |
| Опитування з самопочуття | Присутнє | Відсутнє | Відсутнє | Відсутнє |
| Керування цитатами | Присутнє | Присутнє | Відсутнє | Відсутнє |
| Рекомендації, що зробити | Присутнє | Відсутнє | Присутнє | Відсутнє |
| Сумісний з більшістю платформ | Присутнє | Відсутнє | Присутнє | Присутнє |

### Аналіз відомих алгоритмічних та технічних рішень

Далі будуть розглянуті готові алгоритмічні рішення та архітектура.

#### Аналіз алгоритмічних рішень

Система авторизації є важливою для кожного сучасного веб-застосунку, оскільки вона дозволяє обмежити доступ до конфіденційної інформації лише авторизованим користувачам. Шляхом автентифікації система перевіряє, що дані маніпулюються лишне користувачами з відповідними дозволами, що дозволяє уникнути неправомірних змін в інформації.

За основу системи автентифікації було обрано поєднання парольної та авентифікації на основі токенів. Це поєднання дає простий, звичний метод авторизації та можливість користувачу не вводити обліковані дані кожного разу завдяки обміну токенів. Двофакторна автентифікація та за допомогою пошти є найбезпечнішими методами, які не потребують біометричних даних, але між тим, вони потребують наявність засобів для отримання коду підтвердження, що може ускладнити доступ користувача до застосунку.

Головною проблемою автентифікації на основі токенів є те, що токени можуть бути викрадені, тому необхідно побудувати безпечну систему та зберігати їх належним чином. Найпопулярнішими варіантами авторизації на основі токенів є JWT (JSON Web Token) та OAuth 2.0.

JWT автентифікація представляє собою компактний, URL-безпечний токен, який містить закодовану інформацію у форматі JSON. Токен складається з трьох частин: заголовка, корисного навантаження і підпису, які з’єднані крапками і закодовані в Base64. Він простий у використанні, масштабнований, але якщо ключи підпису буде скомпрометований, зловмисники зможуть створювати власні токени [8].

OAuth 2.0 – протокол авторизації, який дозволяє додаткам отримувати обмежений доступ до ресурсів користувача без передачі пароля. Використовує токени доступу (access token), які можуть бути короткочасними та оновлюватися за допомогою токенів оновлення (refresh token). Проблема такого підходу у залежності від провайдерів авторизації [9].

Для розробки програмного продукту було обрано JWT автентифікацію з використанням токенів оновлення. Рішення таке було прийнято через те, що ми будемо зберігати JWT токени у локальному сховищі, щоб система могла відправляти їх з клієнту до серверу разом із запитами. Цей токен буде мати коротку тривалість життя, але також постійно оновлюватись токеном оновлення, який буде передаватись через кукіси (HttpOnly Cookie) [10]. Завдяки цьому токен оновлення не зможе бути викраденим.

Система рекомендації – є однією з основних задач веб-застосунку впорядкування активностей та підтримки емоційного стану. Цю задачу можна вирішити трьома способами: сортування та фільтрування, сортування та фільтрування з наперед прописаними правилами, використання штучного інтелекту. Сортування та фільтрування – є найпростішим з запропонованих методів, але й найменш персоналізованим. Сортування та фільтрування з наперед прописаними правилами є більш оптимальним способом вирішення задачі, оскільки враховує, які активності є важливішими. Використання штучного інтелекту є найперсоналізованішим варіантом для системи рекомендації, але потребує значну частку ресурсів та може займати довше часу. Тому було вирішено зупинитися на сортуванні та фільтруванні з наперед прописаними правилами.

#### Аналіз архітектури

На сьогоднішній день існує безліч відомих архітектур програмного забезпечення, таких як модель-вид-контролер, клієнт-сервер, шарова архітектура та інші. Кожна з цих архітектур має свої унікальні переваги і може бути ефективно застосована в різних умовах. Одним з найкращих підходів є поєднання кількох архітектурних стилів, що дозволяє максимально використати їх сильні сторони і створити більш гнучке і ефективне рішення. Наприклад, можна інтегрувати шарову архітектуру для структурованої організації логіки додатка та MVC для забезпечення гнучкості і чіткого розділення обов'язків між різними компонентами системи. Це поєднання дозволяє досягти високої продуктивності, зручності підтримки та масштабованості програмного забезпечення.

Модель-вид-контролер (Model-View-Controller, MVC) розділяє програму на три основні компоненти: модель, вид і контролер. Модель управляє даними та логікою, вид відповідає за відображення даних, а контролер обробляє введення користувача і координує роботу між моделлю та видом [11]. Серед переваг можна виділити чітке розділення між даними, логікою і представлення та гнучкість у оновленні одного компоненту без впливу на інші, але серед недоліків: підвищене навантаження через більшу кількість компонентів, складність впровадження та хоча залежність між компонентами мінімальна, все одно існує необхідність узгоджувати роботу між ними.

Клієнт-сервер (Client-Server) розділяє систему на клієнтську частину, яка взаємодіє безпосередньо з користувачем, та серверну частину, яка обробляє запити від клієнтів, виконує бізнес логіку і взаємодіє з базою даних [12]. Цей вид архітектури дає можливість клієнтським пристроям бути легкими та доступними, оскільки основне навантаження несе сервер, а також сервер забезпечує централізоване управління даними та безпекою. Недоліками є те, що сервер є центральною точкою відмови, і якщо він падає, то вся система може перестати працювати. Також наявна залежність від мережі, через що можуть з’являтися проблеми з продуктивністю та надійністю.

Шарова архітектура (Layered Architecture) розділяє програму на кілька рівнів, кожен з яких виконує певну функцію. Зазвичай виділяють рівень представлення (PL), рівень бізнес-логіки (BLL) та рівень доступу до даних (DAL). Це дає архітектурі модульність, тобто легкість розробки та тестуванню завдяки чіткому розділенню на шари та перевикористання коду, завдяки чому ми зможемо повторно використовувати компоненти в інших проектах, або частинах програми. Але з цим шарова архітектура також може бути складною для змін, адже внесення вимагитиме редагування кількох рівнів, а також мати проблеми з продуктивністю через додаткові рівні абстракції [13].

Для розробки програмного продукту було обрано поєднання архітектури клієнт-сервер та шарової архітектури, що складається із: рівня доступу до даних (DAL), рівня бізнес-логіки (BLL), рівня представлення (PL).

## Опис бізнес-процесів

Для опису основних бізнес процесів програмного забезпечення використовується BPMN модель (рисунки 1.4 – 1.8)

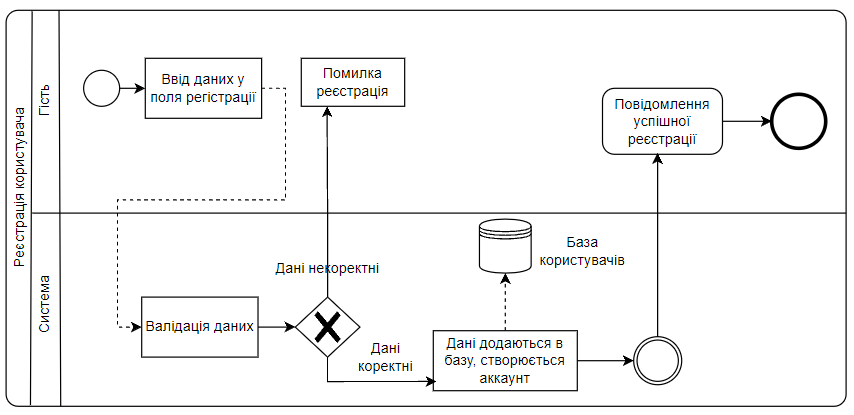


Рисунок 1.4 – BPMN модель створення облікового запису користувача

Опис послідовності створення облікового запису користувача:

* користувач заходить на сторінку для авторизації;
* користувач заповнює необхідні поля для авторизації;
* якщо заповнені поля не відповідають встановленим вимогам, вони підсвічуються як помилкові;
* якщо всі поля заповнені правильно, створюється обліковий запис користувача та додається до бази даних;
* користувач отримує повідомлення про успішну реєстрацію;

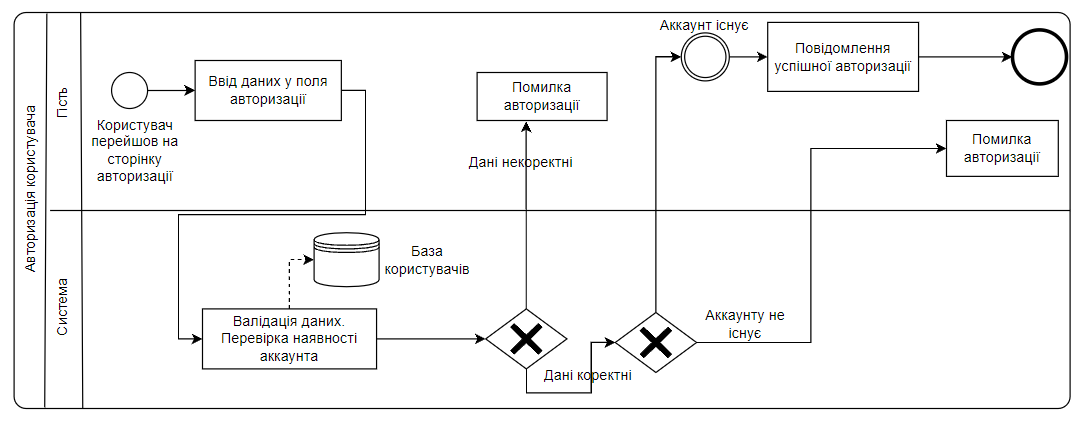


Рисунок 1.5 – BPMN модель входу в аккаунт

Опис послідовності входу в систему користувача:

* користувач заходить на сторінку для авторизації;
* користувач вводить дані у відповідні поля для авторизації;
* якщо введені дані не відповідають встановленим вимогам, відповідні поля підсвічуються помилкою;
* якщо такий обліковий запис існує в базі, користувач успішно авторизується;
* якщо облікового запису не існує, користувач отримує повідомлення про це;

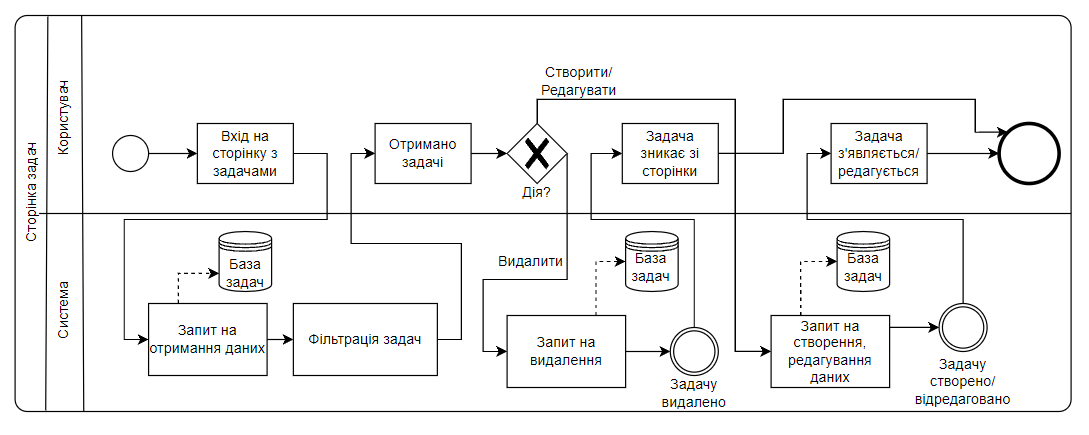


Рисунок 1.6 – BPMN модель входу на сторінку з задачами

Опис послідовності входу на сторінку з задачами:

* користувач переходить на сторінку з задачами;
* задачі надходять з бази даних, фільтруються та надходять користувачу;
* користувач робить запит на видалення задачі;
* запит оброблюється на сервері та задача зникає зі сторінки користувача;
* користувач робить запит на створення/редагування задачі;
* запит оброблюється на сервері та задача з’являється/редагується на сторінці користувача;

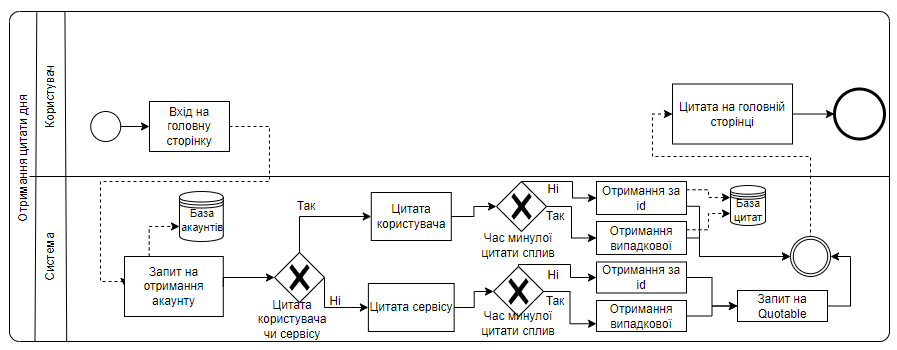


Рисунок 1.7 – BPMN модель отримання цитати дня

Опис послідовності отримання цитати дня:

* користувач заходить на головну сторінку;
* запит надходить на сервер, де дані користувача підтягуються з бази даних;
* проходить перевірка, користувач хоче бачити свої цитати чи цитати з сервісу;
* проходить перевірка чи сплив час минулої цитати;
* якщо цитата користувача та час сплив, надходить запит на випадкову цитату до бази даних;
* якщо цитата користувача та час не сплив, надходить запит на конкретну цитату до бази даних;
* якщо цитата з сервісу та час сплив, надходить запит на випадкову цитату до сервісу Quotable;
* якщо цитата з сервісу та час не сплив, надходить запит на конкретну цитату до сервісу Quotable;
* цитата надходить на головну сторінку та відображається користувачу;

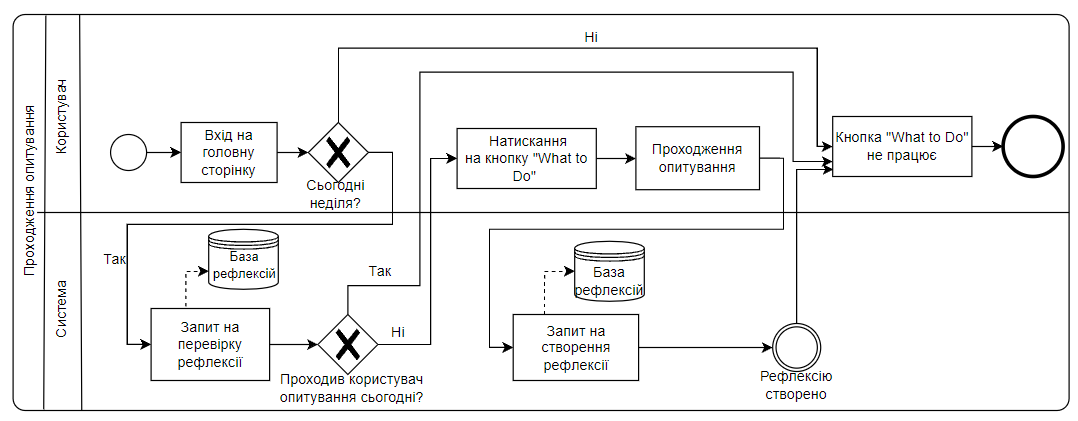


Рисунок 1.8 – BPMN модель проходження опитування

Опис послідовності проходження опитування:

* користувач заходить на головну сторінку;
* йде перевірка сьогодні неділя чи ні, якщо ні, то кнопка з опитуванням не працює;
* якщо сьогодні неділя, робиться запит до бази даних, щоб перевірити, чи проходив користувач опитування сьогодні, якщо так, то кнопка з опитуванням не працює;
* якщо користувач не проходив, то кнопка з опитуваннями працює і користувач може пройти опитування;
* після проходження опитування, робиться запит на створення рефлексії і після створення, кнопка з опитуванням не працює, а рефлексію створено;

## Постановка задачі

Мета цієї розробки полягає в підвищенні рівня продуктивності людей шляхом створення зручної платформи, яка дозволяє організовувати та керувати своїми діяльностями, а також відстежувати свій емоційний стан. Ця платформа спрямована на надання користувачам інструментів для планування завдань, формування корисних звичок і відмови від шкідливих, а також можливості аналізувати свій прогрес і емоційний стан.

Для досягнення поставленої мети необхідно виконати наступні завдання:

* сформулювати та проаналізувати функціональні та нефункціональні вимоги до програмного забезпечення;
* розробити високорівневу архітектуру застосунку;
* здійснити аналіз та вибір інструментів розробки;
* розробити та описати схему БД, а також підходи до роботи з нею;
* зконструювати та реалізувати серверну та клієнтську частини;
* провести аналіз якості реалізованого програмного забезпечення;
* розгорнути застосунок у хмарному середовищі;

## Висновки до розділу

У цьому розділі було детально розглянуто та описано обрану предметну область. було проведено всебічний аналіз предметної області та сфери дослідження. У ньому були визначені основні положення і терміни, що мають відношення до проекту. Також розглянуті ключові концепції та методології, які застосовуються для організації активностей та підтримки емоційного стану користувачів.

Було розглянуто відомі програмні рішення, які було порівняно з дипломним проєктом у таблиці 1.1. Також було проаналізовано алгоритмічні рішення, архітектуру та допоміжні програмні засоби та засоби розробки. Цей аналіз допомагає глибше розібратися в предметній області та визначити шляхи реалізації програмного продукту та вплив на структуру розробки.

Після цього був сформований список бізнес процесів завдяки використанню BPMN моделей, якими були описані основні процеси: авторизації, реєстрації, керуванню задачами, проведеню опитувань, отриманню випадкової цитати. Завдяки цьому було докладно вивчено та уточнено поточні бізнес-процеси, що дало змогу виявити можливість оптимізації та потенційні проблеми та ризики.

В кінці розділу було сформовано мету розробки та необхідні для досягнення мети задачі.

# РОЗРОБЛЕННЯ вимог до програмного забезпечення

## Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є керування активностями та проходження опитування, більше функцій можна побачити на рисунку 2.1.

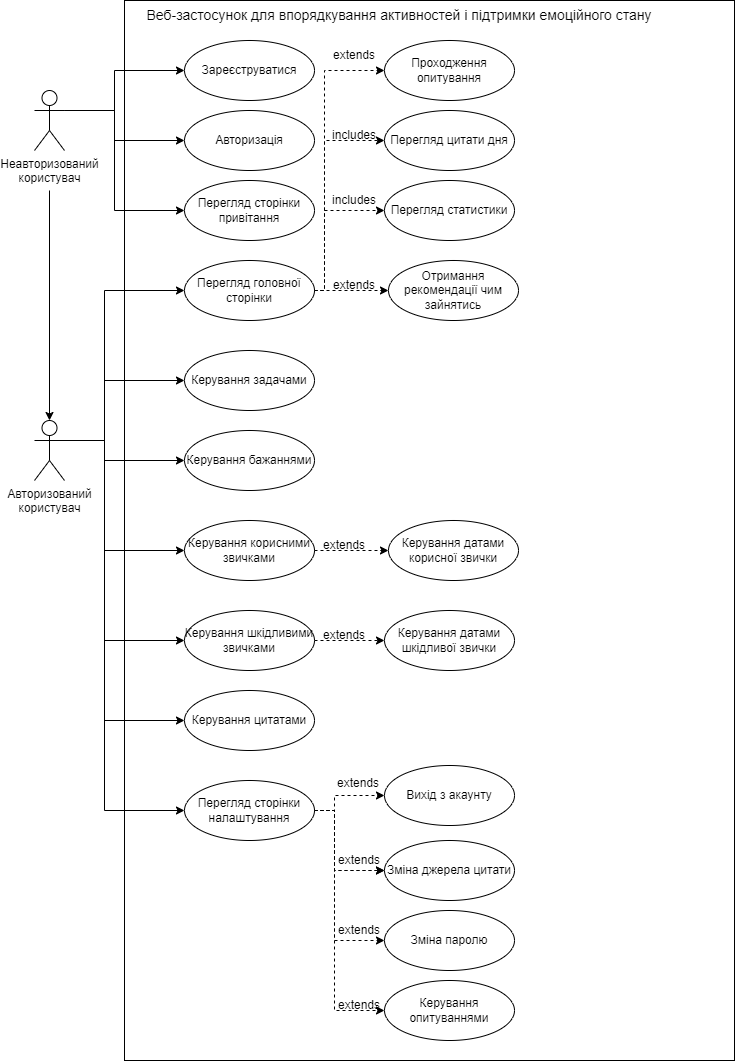


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1 - 2.20 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 - Варіант використання UC-1

|  |  |
| --- | --- |
| Use case name | Реєстрація користувача |
| Use case ID | UC-01 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Неавторизований користувач |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: логін користувача, пароль в системі, та його повтор для підтвердження. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на головну сторінку. |
| Extension | В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом. |
| Post-Condition | Створення сторінки користувача, перехід на головну сторінку |

Таблиця 2.2 - Варіант використання UC-2

|  |  |
| --- | --- |
| Use case name | Авторизація користувача |
| Use case ID | UC-02 |
| Goals | Авторизація користувача в системі |
| Actors | Неавторизований користувач |
| Trigger | Користувач бажає авторизуватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку авторизації. В поля для авторизації вводяться відповідні дані: логін користувача, та пароль в системі. Після заповнення даних користувача натискає кнопку авторизації. Після цього з’являється повідомлення про успішну авторизацію, і користувач перенаправляється на головну сторінку. |
| Extension | В випадку введення не коректних даних, кнопка авторизації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом. |
| Post-Condition | Авторизація користувача, перехід на головну сторінку |

Таблиця 2.3 - Варіант використання UC-3

|  |  |
| --- | --- |
| Use case name | Перегляд сторінки привітання |
| Use case ID | UC-03 |
| Goals | Ознайомлення із веб-застосунком і його використанням |
| Actors | Неавторизований користувач |
| Trigger | Відкриття веб-застосунку |
| Pre-conditions | - |
| Flow of Events | Користувач відкриває веб-застосунок. Йому відображається вітальна сторінка із полями для авторизації |
| Extension | - |
| Post-Condition | Користувач бачить вітальну сторінка із полями для авторизації |

Таблиця 2.4 - Варіант використання UC-4

|  |  |
| --- | --- |
| Use case name | Перегляд сторінки налаштувань |
| Use case ID | UC-04 |
| Goals | Перегляд сторінки налаштувань користувачем |
| Actors | Користувач |
| Trigger | Користувач заходить на сторінку налаштувань |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку налаштувань. |
| Extension | - |
| Post-Condition | Користувач бачить сторінку налаштувань, на якій функції вийти з акаунту, змінити пароль, перейти на сторінку з опитуваннями, змінити джерело цитат. |

Таблиця 2.5 - Варіант використання UC-5

|  |  |
| --- | --- |
| Use case name | Проходження опитування |
| Use case ID | UC-5 |
| Goals | Проходження опитування користувачем |
| Actors | Користувач |
| Trigger | Користувач натискає на кнопку «Пройти опитування» |
| Pre-conditions | Сьогодні неділя, користувач не проходив сьогодні опитування, користуач на головній сторінці. |
| Flow of Events | Користувач натискає на кнопку «Пройти опитування», відкривається вікно з опитуванням. Користувач проходить опитування. Після цього користувач натискає додати |
| Extension | Користувач вводить на яку оцінку він відчував свій емоціонал, свою продуктивність, а також три слова, якими би він описав минулий тиждень та плани на наступний тиждень. |
| Post-Condition | Користувач повертається на головну сторінку, опитування пройдене, кнопка «пройти опитування» не працює. |

Таблиця 2.6 - Варіант використання UC-6

|  |  |
| --- | --- |
| Use case name | Перегляд цитати дня |
| Use case ID | UC-6 |
| Goals | Перегляд цитати дня користувачем |
| Actors | Користувач |
| Trigger | Користувач бажає подивитися цитату дня. |
| Pre-conditions | Користувач на головній сторінці. |
| Flow of Events | Користувач переходить на головну сторінку. |
| Extension | У залежності від того що обрано в налаштуваннях у користувача, цитата з сервісу чи цитата користувача, робиться запит на цитату дня. Якщо цитата ще не надійшла, користувач бачить тробер завантаження. |
| Post-Condition | Користувач бачить цитату дня. |

Таблиця 2.7 - Варіант використання UC-7

|  |  |
| --- | --- |
| Use case name | Перегляд статистики |
| Use case ID | UC-7 |
| Goals | Перегляд статистики користувачем |
| Actors | Користувач |
| Trigger | Користувач бажає переглянути на статистику |
| Pre-conditions | Користувач на головній сторінці. |
| Flow of Events | Користувач переходить на головну сторінку. |
| Extension | У залежності від того що обрано в налаштуваннях у користувача, за тиждень, за місяць чи за 3 місяці, робиться запит на статистику. Якщо статистика ще не надійшла, користувач бачить тробер завантаження. |
| Post-Condition | Користувач бачить статистику за обраний період. |

Таблиця 2.8 - Варіант використання UC-8

|  |  |
| --- | --- |
| Use case name | Отримання рекомендації чим зайнятись |
| Use case ID | UC-8 |
| Goals | Отримання рекомендації чим зайнятись користувачем |
| Actors | Користувач |
| Trigger | Користувач натискає кнопку «Отримати рекомендацію чим зайнятись» |
| Pre-conditions | Користувач на головній сторінці |
| Flow of Events | Користувач натискає на кнопку «Отримати рекомендацію чим зайнятись» |
| Extension | - |
| Post-Condition | З’являється вікно з трьома найкращими активностями для виконання, та по три активності з кожної категорії |

Таблиця 2.9 - Варіант використання UC-9

|  |  |
| --- | --- |
| Use case name | Керування задачами |
| Use case ID | UC-9 |
| Goals | Перегляд/створення/редагування/видалення задач |
| Actors | Користувач |
| Trigger | Користувач заходить на сторінку задач |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку задач і має можливість переглянути матрицю задач. Якщо користувач хоче створити нову задачу, він натискає на кнопку «Додати задачу», заповнює дані та натискає кнопку «Додати». Якщо користувач хоче видалити задачу, він натискає на кнопку «Видалити задачу». Якщо користувач хоче відредагувати задачу, він натискає на кнопку «Відредагувати задачу». |
| Extension | Якщо користувач хоче створити задачу, він заповнює поля з дедлайном, обирає чи важлива вона та термінова, а також в чому заключається задача. Якщо користувач хоче відредагувати задачу, він редагує поля з дедлайном, важливістю, терміновістю, а також в чому заключається задача. Задачі фільтруються у порядку терміновості, важливості та готовності. |
| Post-Condition | Користувач переглянув/створив/відредагував/видалив задачі |

Таблиця 2.10 - Варіант використання UC-10

|  |  |
| --- | --- |
| Use case name | Керування бажаннями |
| Use case ID | UC-10 |
| Goals | Перегляд/створення/редагування/видалення бажань |
| Actors | Користувач |
| Trigger | Користувач заходить на сторінку бажань |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку бажань і має можливість переглянути список бажань. Якщо користувач хоче створити нове бажання, він натискає на кнопку «Додати бажання», заповнює дані та натискає кнопку «Додати». Якщо користувач хоче видалити бажання, він натискає на кнопку «Видалити бажання». Якщо користувач хоче відредагувати бажання, він натискає на кнопку «Відредагувати бажання». |
| Extension | Якщо користувач хоче створити бажання, він заповнює поле з тим в чому заключається бажання. Якщо користувач хоче відредагувати бажання, він редагує поле в чому заключається бажання. Бажання відфільтровуються у порядку виконані/не виконані. |
| Post-Condition | Користувач переглянув/створив/відредагував/видалив бажання |

Таблиця 2.11 - Варіант використання UC-11

|  |  |
| --- | --- |
| Use case name | Керування корисними звичками |
| Use case ID | UC-11 |
| Goals | Керування корисними звичками користувачем |
| Actors | Користувач |
| Trigger | Користувач заходить на сторінку корисних звичок |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку корисних звичок і має можливість переглянути список корисних звичок. Користувач може відкрити корисну звичку. Якщо користувач хоче створити нову корисну звичку, він натискає на кнопку «Додати звичку», заповнює дані та натискає кнопку «Додати». Якщо користувач хоче видалити звичку, він натискає на кнопку «Видалити звичку». Якщо користувач хоче відредагувати звичку, він натискає на кнопку «Відредагувати звичку». |
| Extension | Якщо користувач хоче створити звичку, він заповнює поле з тим в чому заключається звичка. Якщо користувач хоче відредагувати звичку, він редагує поле в чому заключається звичка. |
| Post-Condition | Користувач переглянув/створив/відредагував/видалив корисні звички |

Таблиця 2.12 - Варіант використання UC-12

|  |  |
| --- | --- |
| Use case name | Керування шкідливими звичками |
| Use case ID | UC-12 |
| Goals | Керування шкідливими звичками користувачем |
| Actors | Користувач |
| Trigger | Користувач заходить на сторінку шкідливих звичок |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку шкідливих звичок і має можливість переглянути список шкідливість звичок. Користувач може відкрити шкідливу звичку. Якщо користувач хоче створити нову шкідливу звичку, він натискає на кнопку «Додати звичку», заповнює дані та натискає кнопку «Додати». Якщо користувач хоче видалити звичку, він натискає на кнопку «Видалити звичку». Якщо користувач хоче відредагувати звичку, він натискає на кнопку «Відредагувати звичку». |
| Extension | Якщо користувач хоче створити звичку, він заповнює поле з тим в чому заключається звичка та коли вона сталася. Якщо користувач хоче відредагувати звичку, він редагує поле в чому заключається звичка. |
| Post-Condition | Користувач переглянув/створив/відредагував/видалив корисні звички |

Таблиця 2.13 - Варіант використання UC-13

|  |  |
| --- | --- |
| Use case name | Керування цитатами |
| Use case ID | UC-13 |
| Goals | Керування цитатами користувачем |
| Actors | Користувач |
| Trigger | Користувач заходить на сторінку цитат |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку цитат і має можливість переглянути список цитат. Якщо користувач хоче створити нову цитату, він натискає на кнопку «Додати цитату», заповнює дані та натискає кнопку «Додати». Якщо користувач хоче видалити цитату, він натискає на кнопку «Видалити цитату». Якщо користувач хоче відредагувати цитату, він натискає на кнопку «Відредагувати цитату». |
| Extension | Якщо користувач хоче створити цитату, він заповнює поле з цитатою та автором. Якщо користувач хоче відредагувати цитату, він редагує поле з цитатою та автором. |
| Post-Condition | Користувач переглянув/створив/відредагував/видалив цитати |

Таблиця 2.14 - Варіант використання UC-14

|  |  |
| --- | --- |
| Use case name | Керування датами корисної звички |
| Use case ID | UC-14 |
| Goals | Керування датами корисної звички користувачем |
| Actors | Користувач |
| Trigger | Користувач відкриває корисну звичку |
| Pre-conditions | Користувач на сторінці корисних звичок |
| Flow of Events | Користувач відкриває корисну звичку, йому відображається календар на якому він може побачити дні, коли він дотримувався звички. Користувач може обрати дату і якщо, тоді він не дотримався звички, він може створити дату. Якщо дотримався звички, він може видалити дату. |
| Extension | - |
| Post-Condition | Користувач створив/видалив дату корисної звички |

Таблиця 2.15 - Варіант використання UC-15

|  |  |
| --- | --- |
| Use case name | Керування датами шкідливої звички |
| Use case ID | UC-15 |
| Goals | Керування датами шкідливої звички користувачем |
| Actors | Користувач |
| Trigger | Користувач відкриває шкідливу звичку |
| Pre-conditions | Користувач на сторінці шкідливих звичок |
| Flow of Events | Користувач відкриває шкідливу звичку, йому відображається календар на якому він може побачити дні, коли він дотримувався звички. Користувач може обрати дату і якщо, тоді він дотримався звички, він може створити дату. Якщо не дотримався звички, він може видалити дату. |
| Extension | - |
| Post-Condition | Користувач створив/видалив дату шкідливої звички |

Таблиця 2.16 - Варіант використання UC-16

|  |  |
| --- | --- |
| Use case name | Вихід з акаунту |
| Use case ID | UC-16 |
| Goals | Вихід з акаунту користувача |
| Actors | Користувач |
| Trigger | Користувач бажає вийти з акаунту |
| Pre-conditions | Користувач на сторінці налаштувань |
| Flow of Events | Користувач натискає на кнопку «Вийти з акаунту», після цього з’являється вікно з пітвердженням. |
| Extension | - |
| Post-Condition | Користувач вийшов з акаунту |

Таблиця 2.17 - Варіант використання UC-17

|  |  |
| --- | --- |
| Use case name | Зміна джерела цитати |
| Use case ID | UC-17 |
| Goals | Зміна джерела цитати користувачем |
| Actors | Користувач |
| Trigger | Користувач бажає змінити джерело цитати |
| Pre-conditions | Користувач на сторінці налаштувань |
| Flow of Events | Користувач обирає джерело цитати. Є два варіанти: «З сервісу» або «Мої цитати» |
| Extension | - |
| Post-Condition | Користувач змінив джерело цитати |

Таблиця 2.18 - Варіант використання UC-18

|  |  |
| --- | --- |
| Use case name | Зміна паролю |
| Use case ID | UC-18 |
| Goals | Зміна паролю користувачем |
| Actors | Користувач |
| Trigger | Користувач натискає на «Змінити пароль» |
| Pre-conditions | Користувач на сторінці налаштувань |
| Flow of Events | Користувач натискає на «Змінити пароль». З’являється вікно з підтвердженням паролю, куди користувач має ввести актуальний пароль і натиснути кнопку «Підтвердити». Після цього з’явиться вікно зі створенням нового паролю. Користувач вводить дані та натискає: «Змінити пароль» |
| Extension | Якщо пароль не сходиться з актуальним, користувач отримує повідомлення. При зміні пароль користувач має ввести новий пароль та підтвердження паролю. |
| Post-Condition | Користувач змінив пароль |

Таблиця 2.19 - Варіант використання UC-19

|  |  |
| --- | --- |
| Use case name | Перегляд головної сторінки |
| Use case ID | UC-19 |
| Goals | Перегляд головної сторінки користувачем |
| Actors | Користувач |
| Trigger | Користувач бажає переглянути головну сторінку |
| Pre-conditions | - |
| Flow of Events | Користувач потрапляє на головну сторінку. Він бачить кнопки з проходженням опитування, з отриманням рекомендації, цитату дня та статистику. |
| Extension | - |
| Post-Condition | - |

Таблиця 2.20 - Варіант використання UC-20

|  |  |
| --- | --- |
| Use case name | Керування опитуваннями |
| Use case ID | UC-20 |
| Goals | Перегляд/створення/редагування/видалення опитування |
| Actors | Користувач |
| Trigger | Користувач заходить на сторінку опитувань |
| Pre-conditions | Користувач на сторінці налаштувань |
| Flow of Events | Користувач переходить на сторінку опитувань і бачить календар з датами, коли він проходив опитування. Користувач обирає тиждень. Якщо користувач проходив на тому тижні опитування, він може переглянути його, відредагувати, видалити. Якщо не проходив, то він може створити нове опитування. |
| Extension | Якщо користувач хоче відредагувати опитування, він редагує поля з продуктивністю, емоційністю, трьома словами, які описують минулий тиждень, а також плани на майбутній тиждень |
| Post-Condition | Користувач переглянув/створив/відредагував/видалив опитування |

## Аналіз системних вимог

Вимоги до системи користувачів:

* Сумісність системи з сучасними браузерами, такими як Google Chrome, Opera, Microsoft Edge та інші;
* Інтернет-з'єднання зі швидкістю не менше 10 мегабіт/с;

## Розроблення функціональних вимог

Веб-застосунок розділений на модулі. Кожен модуль має певний функціонал. В таблиці 2.21 наведено загальну модель вимог, а в таблиці 2.22 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.3.

Таблиця 2.21 – Загальна модель вимог

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | ID вимоги | Пріоритети | Ризики |
| 1 | Перегляд сторінки привітання. | FR-1 | Низький | Низький |
| 2.1 | Реєстрація користувача. | FR-2 | Високий | Високий |
| 2.2 | Авторизація користувача. | FR-3 | Високий | Високий |
| 2.3 | Вихід із акаунту. | FR-4 | Високий | Низький |
| 3 | Зміна паролю | FR-5 | Середній | Високий |
| 4 | Перегляд головної сторінки | FR-6 | Високий | Низький |
| 5 | Керування задачами | FR-7 | Середній | Середній |
| 6 | Керування бажаннями | FR-8 | Середній | Середній |
| 7 | Керування корисними звичками | FR-9 | Середній | Середній |
| 8 | Керування шкідливими звичками | FR-10 | Середній | Середній |
| 9 | Керування цитатами | FR-11 | Середній | Середній |
| 10 | Перегляд сторінки налаштування | FR-12 | Високий | Низький |
| 11 | Проходження опитування | FR-13 | Середній | Низький |
| 12 | Перегляд цитати дня | FR-14 | Низький | Низький |
| 13 | Перегляд статистики | FR-15 | Середній | Низький |
| 14 | Отримання рекомендації чим зайнятись | FR-16 | Середній | Низький |
| 15 | Керування датами корисної звики | FR-17 | Високий | Середній |
| 16 | Керування датами шкідливої звички | FR-18 | Високий | Середній |
| 17 | Зміна джерела цитати | FR-19 | Низький | Низький |
| 18 | Керування опитуваннями | FR-20 | Середній | Середній |

Таблиця 2.22 – Перелік функціональних вимог

|  |  |
| --- | --- |
| ID вимоги | Назва та опис |
| FR-1 | Перегляд сторінки привітання.  Неавторизований користувач бачить вітальну веб-сторінку, на якій також можна вибрати опцію для реєстрації або входу в систему. |
| FR-2 | Реєстрація користувача.  Неавторизований користувач повинен мати змогу створити акаунт за допомогою логіну та паролю. |
| FR-3 | Авторизація користувача.  Неавторизований користувач повинен мати можливість увійти в акаунт за допомогою логіну та паролю. |
| FR-4 | Вихід із акаунту.  Авторизований користувач повинен мати змогу вийти з акаунту. |
| FR-5 | Зміна паролю.  Авторизований користувач повинен може змінити пароль. |
| FR-6 | Перегляд головної сторінки. |
| FR-7 | Керування задачами.  Користувач може переглянути список задач, створити нову задачу, редагувати або видалити існуючу задачу. |
| FR-8 | Керування бажаннями.  Користувач може переглянути список бажань, створити нове бажання, редагувати або видалити існуюче бажання. |
| FR-9 | Керування корисними звичками.  Користувач може переглянути список корисних звичок, створити нову корисну звичку, редагувати або видалити існуючу корисну звичку. |
| FR-10 | Керування шкідливими звичками.  Користувач може переглянути список шкідливих звичок, створити нову корисну звичку, редагувати або видалити існуючу корисну звичку. |
| FR-11 | Керування цитатами.  Користувач може переглянути список цитат, створити нову цитату, редагувати або видалити існуючу цитату. |
| FR-12 | Перегляд сторінки налаштування.  Користувач мусить мати змогу потрапити на сторінку налаштування. |
| FR-13 | Проходження опитування.  Користувач може щонеділі пройти опитування з саморефлексії. |
| FR-14 | Перегляд цитати дня.  Користувач має отримувати кожен день цитату дня у залежності від його налаштувань. |
| FR-15 | Перегляд статистики  Користувач має змогу переглядати статистику продуктивності. |
| FR-16 | Отримання рекомендації чим зайнятись.  Користувач має змогу отримати рекомендацію чим зайнятись враховуючи створені ним активності. |
| FR-17 | Керування датами корисної звички.  Користувач має змогу переглянути дати, коли було дотримано корисної звички, додавати та видаляти дати. |
| FR-18 | Керування датами шкідливої звички  Користувач має змогу переглянути дати, коли було дотримано шкідливої звички, додавати та видаляти дати. |
| FR-19 | Зміна джерела цитати.  Користувач має змогу обрати джерело цитати між сервісом та його власними цитатами. |
| FR-20 | Керування опитуваннями.  Користувач має змогу переглянути створені раніше опитування, додавати нові опитування, видаляти та редагувати існуючі. |

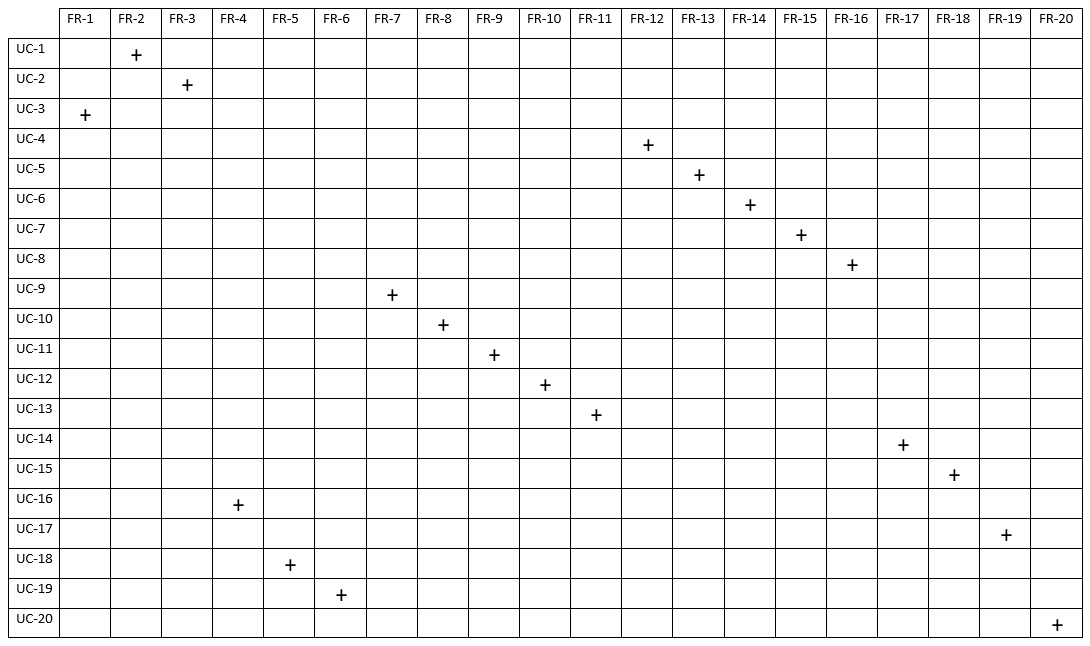


Рисунок 2.2 – Матриця трасування вимог

## Розроблення нефункціональних вимог

* Веб-застосунок повинен коректно відображатися в сучасних браузерах, що підтримуються.
* Веб-застосунок має підтримувати англійську мову як основну для зручної взаємодії з користувачами.
* Веб-застосунок слід спроектувати так, щоб він був доступний як з мобільних, так і з десктопних браузерів.
* Веб-застосунок має гарантувати конфіденційність та безпеку користувачів, щоб їхні дані не потрапили до сторонніх осіб.
* Веб-застосунок повинен працювати швидко та без затримок.

## Висновки до розділу

У цьому розділі був проведений аналіз потреб користувачів та вимог до програмного забезпечення. Завдяки опису бізнес-процесів та аналізі предметної області у першому розділі, було сформовано список варіантів використання, на основі якого було створено діаграму. Це дало змогу виділити функціональні вимоги та створити матрицю трасувань.

Розробка функціональних вимог допомагає зрозуміти обсяг робіт, а пріоритетність певних вимог допомагає встановити порядок розробки. Визначені функціональні вимоги сприяють систематизації розробки та виявленню нюансів, які необхідно вирішити. Розробка нефункціональних вимог, завдяки нефункціональному погляду на програмне забезпечення, допомагає краще розуміти розроблюваний продукт.

Як результат цього процесу, були детально окреслені завдання програми та її функціональні вимоги, що є важливим кроком перед початком розробки.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення. [КПІ.ІТ-0425.045440.01.91]

# КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Архітектура програмного забезпечення

Згідно з аналізом з першого розділу, архітектура «клієнт-сервер» у поєднанні з шаровою архітектурою була обрана для веб-застосунку. Переваги такої архітектури полягають у розділенні обов’язків, розділенні навантаження між клієнтськими і серверними пристроями, а також у забезпеченні безпеки даних, так як чутливі дані можуть залишатися на сервері, а не передаватися клієнту. Даний метод розробки реалізовується виключно за допомогою об'єктно-орієнтованої мови програмування та відповідних принципів. Діаграму зв’язку між рівнями програми відображено на рисунку 3.1

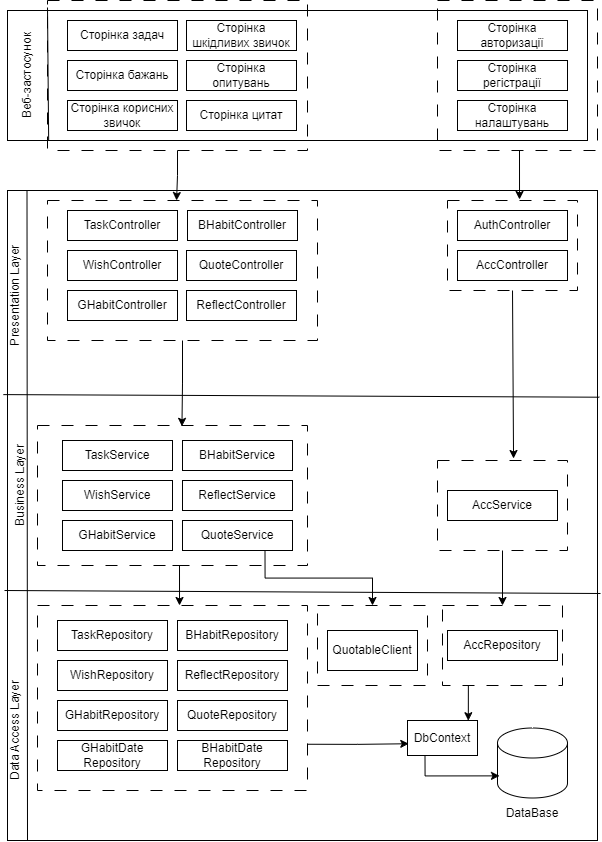


Рисунок 3.1 – Діаграма зв’язку між різними рівнями програми

Рівень PL відповідає за взаємодію з користувачем. Він включає в себе як інтерфейс користувача (UI), так і контролери та інші компоненти, які отримують дані від користувача, обробляють їх та відображають результати. Його завдання – забезпечити зручний та ефективний інтерфейс, який дозволить користувачам легко взаємодіяти з програмою та отримувати необхіднну інформацію та функціональність. У цьому контексті це відноситься до API, яке забезпечує засіб зв'язку між сервером та зовнішніми системами чи додатками.

Рівень BLL відповідає за бізнес-логіку програми. Він реалізує правила та алгоритми, що визначають поведінку програми. BLL не залежить від деталей реалізації PL або DAL. Цей рівень відіграє критичну роль у функціонуванні програми, бо він втілює всі правила та алгоритмі, які визначають її поведінку. Варто зазначити, що BLL оперує виключно з представленням даних, а не з самими даними, що знаходяться у сховищі. Це дає можливість забезпечити захист від втрати або пошкодженню інформації, зберігаючи її недоторканою у сховищі.

DAL - це шар в архітектурі програмного забезпечення, який відповідає за взаємодію з джерелом даних. Це може бути база даних, файлова система, веб-служба або будь-яке інше джерело даних. Для рівня бізнес-логіки він є безпечним способом доступу до даних, адже DAL інкапсулює логіку доступу до даних, щоб бізнес-логіка програми не залежала від деталей реалізації доступу до даних.

Управління користувачами відбувається завдяки обміну токенами: доступу та оновлення. Коли користувач авторизується, для нього формується два токени, які передаються і створюються різним чином.

Токен доступу має коротший проміжок існування та передається через запити, після чого він зберігається у локальному сховищі користувача. Формується такий токен за технологією JWT та складається з трьох частин, метод шифрування – симетричний. Це означає, що один і той же ключ використовується як для шифрування, так і для розшифрування та підпису. Зроблено це через те, що симетричне шифрування є більш швидким, зменшується ризик втрати ключа, оскільки немає необхідності управляти парою ключі, а також ключ можна легко змінити у разі потреби, забезпечуючи відмовостійкість системи.

Токен оновлення має довший проміжок існування та передається через httpOnly Cookie. Їх перевага полягає в тому, що вони недоступні для скриптів на сторінці через JavaScript, що допомагає запобігти атакам на внедрнення коду. Коли сервер встановлює куки з атрибутом HttpOnly, браузер при відправці HTTP-запитів автоматично включає ці куки у заголовок запиту.

Коли користувач заходить на сторінку веб-застосунку, а токен доступу вичерпався, робиться запит на оновлення токена доступу, що також оновлює і токен оновлення, і тому користувачу не потрібно повторно вводити логін та пароль.

Із зовнішніх систем використовується тільки публічний API Quotable[32] для отримання випадкових цитат різної довжини та різних тематик. Сервіс є безкоштовним та місить велику кількість різноманітних цитат. Використовується він тоді, коли налаштування користувача вказують на отримання цитати з сервісу, а не з даних самого користувача.

У якості провайдера бази даних була обрана СУБД MsSQL для зберігання усіх даних та інформації про користувачів. Такий вибір не є випадковим і обумовлений рядом важливих факторів. Реляційні бази даних, зокрема MsSQL, забезпечують високий рівень структурованості даних, що дозволяє зберігати інформацію у впорядкованому та логічному вигляді. Крім того, MsSQL гарантує цілісність даних, що особливо важливо для підтримки коректності і узгодженості інформації у великих та складних системах. Окрім цього, підтримка складних запитів робить MsSQL ідеальним інструментом для роботи з великими обсягами інформації, де необхідно проводити детальний аналіз та отримувати точні результати. Усі ці характеристики роблять MsSQL оптимальним вибором для систем, які мають чіткі вимоги до даних і високі стандарти надійності.

## Обґрунтування вибору засобів розробки

Важливою частиною розробки веб-застосунку є вибір програмного середовища. Для виконання поставлених задач, було розглянуто декілька варіантів:

* Visual Studio – інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Воно підтримує широкий спектр мов програмування, має глибоку інтеграцію з інструментами та службами Microsoft. Це потужний інструмент для розробки веб-додатків, мобільних додактів, ігрових проектів та інших типів програмного забезпечення [14].
* Visual Studio Code – це легкий, але потужний редактор коду, розроблений Microsoft. Він підтримує безліч мов програмування завдяки розширенням і має вбудовану підтримку для налагодження, версійного котролю та багато іншого [15].
* Rider – крос-платформенне IDE для розробки на .NET, розроблене компанією JetBrains. Воно поєднує в собі інтелектуальні можливості рефакторингу і аналізу коду з підтримкою всіх функцій, які потрібні розробникам на .NET [16].
* MonoDevelop – крос-платформенне IDE, призначене для розробки додатків на C# і інших мовах програмування за допомогою проекту Mono. В основному використовується для розробки під Linux, але підтримує і інші платформи [17].
* IntelliJ IDEA – це потужне IDE для розробки на Java і багатьох інших мовах програмування розроблене компанією JetBrains. Відоме своєю інтелектуальною підтримкою розробки, включаючи рефакторинг, автоматичне завершення коду і аналіз коду в реальному часі [18].

Для розробки веб-застосунку було обрано Visual Studio для серверної частини та Visual Studio Code для клієнтської. Вибір пов’язаний з великою кількістю доступних розширень і плагінів, наявністю потужних інструментів для діагностики і виправлення помилок, включаючи профайлінг та аналіз коду.

Серверна частина буде побудована на фреймворку ASP.NET [19] і мові програмування C#. Даний фреймворк містить дуже зручний набір інструментів, що дозволяють швидко будувати стабільні та швидкі веб-застосунки. ASP.NET пропонує широкі можливосіт для розробників, завдяки своїй модульній архітектурі та інтеграціями з іншими технологіями Microsoft. Він також дозволяє легко інтегруватися з клієнтськими додатками, забезпечуючи мастштабованість та гнучкість застосунків.

Для клієнтської частини буде використана технологія React[20] - безкоштовна та відкрита бібліотека написана мовою JavaScript для розробки користувацьких інтерфейсів на основі компонентів. Створена та підтримувана компанією Facebook, надає розробникам потужний інструмент для побудови інтерактивних та високопродуктивних веб-додатків. Завдяки своїй компонентнів архітектурі, дозволяє розділяти інтерфейс на окремі незалежні блоки, які можна легко розробляти, тестувати та повторно використовувати.

Веб-дизайн сторінок буде зроблений із допомогою React технологій, таких як Styled Components[21], Formik[22], Axios[23], React Router[24] та React Query[25], FontAwesome[26], MaterialUI[27], Dayjs[28]. Поєднання цих технологій дозволяє створювати високоякісні веб-застосунки з гарною структурою коду, ефективним управлінням станом та даними, сучасним дизайном і відмінним користувацьким досвідом.

Для реалізації рівня доступу до даних (DAL) буде використаний фреймворк Dapper [29]. Dapper є мікро-ORM (Object-Relational Mapper) для .NET, який надає швидкий і зручний спосіб взаємодії з реляційними базами даних. Ця технологія оптимізована для високої продуктивності, не нав’язує жорстких структур чи обмежень та підтримує розширення функціоналу через додаткові пакети.

Також буде використана бібліотека ASP.NET Core Identity[31]. Дана бібліотека надає набір інструментів для менеджменту аккаунтів, безпечного зберігання та хешування паролів, автентифікації та авторизації.

Наведені вище наведені технічні рішення були обрані, оскільки вони, окрім СУБД, є частиною платформи .NET, і як наслідок мають дуже просте налаштування і інтеграцію, надійну та безперебійну роботу.

Для отримання цитат використовується сторонній сервіс Quotable [32]. Цей сервіс безкоштовно та ефективно надає велику базу цитат різних довжин та тематик.

Вибір даних технічних рішень базується на ряді причин, які стосуються продуктивності, ефективності, масштабованості та зручності розробки. Обидві технології мають широку екосистему та є кросплатформеними. Їх об'єднання створює потужний стек технологій, який забезпечить ефективний, масштабований та легко розширюваний веб-застосунок.

## Конструювання програмного забезпечення

Один з основних алгоритмів веб-застосунку впорядкування активностей та підтримки емоційного стану – це рекомендація користувачу, як провести час. Після аналізу з першого розділу було вирішено зупинитися на сортуванні й фільтрування з наперед прописаними правилами.

Завдяки аналізу предметної області, було вирішено поставити активності у порядку важливості. Правила сортування та фільтрування зазначені у таблиці 3.1

Таблиця 3.1 – Правила сортування та фільтрування

|  |  |  |
| --- | --- | --- |
| Пріоритетність | Активність | Пояснення |
| 1 | Задачі, які є важливими та терміновими | Задачі, які виділені як важливі та термінові мають бути найвищим пріоритетом, адже на виконання не так багато часу, та задача є пріоритетною для користувача. |
| 2 | Корисні звички | Згідно з аналізом предметної області, корисні звички є важливими для щоденного дотримання. |
| 3 | Задачі, які є неважливими та терміновими | Задачі, які є тільки терміновими, бажано зробити раніше ніж тільки важливі, адже для тих є більше часу. |
| 4 | Задачі, які є важливими та не терміновими | Задачі, які є тільки важливими, ще мають час для виконання, тому нижче пріоритетом аніж інші активності. |
| 5 | Бажання | Бажання стоять нижче пріоритетом ніж задачі та корисні звички, адже вони рекомендовані до виконання після вирішення поставлених задач. |
| 6 | Задачі, які є неважливими та не терміновими | Задачі, які є неважливими та нетерміновими мають найнижчий пріоритет, адже ще є час на виконання, та задача не є пріоритетною для користувача. |

Оскільки користувач може не мати бажання виконувати задачі згідно з прописаними правилами, він також отримує по три найкращі активності до виконання з кожного типу, де задачи обираються згідно з важливістю та терміновістю, а корисні звички ті, які користувач сьогодні не виконав.

Як було зазначено раніше, архітектура веб-застосунку представлена у вигляді архітектури клієнт-сервер сумісно з кількарівневою архітектурою. Для представлення даних та роботи з базою даних було розроблено 14 моделей, які представлені у таблиці 3.2

Таблиця 3.2 – Моделі

|  |  |
| --- | --- |
| Назва | Опис |
| Account | Репрезентує користувача |
| AccountActivities | Репрезентує користувача разом з усіма активностями |
| BHabit | Репрезентує шкідливу звичку |
| BHabitDate | Репрезентує дату шкідливої звички |
| GHabit | Репрезентує корисну звичку |
| GHabitDate | Репрезентує дату корисної звички |
| GHabitWithDates | Репрезентує корисну звичку разом з датами |
| Reflection | Репрезентує опитування |
| Task | Репрезентує задачу |
| UserQuote | Репрезентує цитату користувача |
| Wish | Репрезентує бажання |
| QuotableItem | Репрезентує цитату з сервісу |
| RefreshToken | Репрезентує токен оновлення |
| WhatToDoResponse | Репрезентує рекомендацію чим зайнятись |

На рівні PL (Presentation Layer) розташовані контролери, які відповідають за взаємодію з користувачами та обробку їхніх запитів. Контролери виконують ключову роль у забезпеченні користувацького інтерфейсу та зручності взаємодії з системою. Вони приймають запити від клієнтів, передають їх на відповідні сервіси бізнес-логіки, отримують оброблені результати та повертають їх користувачам у зручному форматі. Усього буде реалізовано вісім контролерів, кожен з яких призначений для роботи з конкретними модулями системи та має свої функціональні особливості. Докладний опис контролерів та їх функцій можна знайти у таблиці 3.3.

Таблиця 3.3 – Контролери

|  |  |
| --- | --- |
| Назва | Опис |
| AccController | Для роботи з акаунтами |
| AuthController | Для роботи з авторизацією, регістрацією, оновлення токену, вихід з акаунту |
| BHabitController | Для роботи з шкідливими звичками та датами шкідливих звичок |
| GHabitController | Для роботи з корисними звичками та датами корисних звичок |
| QuoteController | Для роботи з цитатами з сервісу та з цитатами користувача |
| ReflectController | Для роботи з опитуваннями |
| TaskController | Для роботи з задачами |
| WishController | Для роботи з бажаннями |

На рівні BLL розташовані сервіси, які відповідають за обробку бізнес-логіки та взаємодію з зовнішніми сервісами. Ці сервіси виконують важливу роль у функціонуванні системи. Вони не лише обробляють запити, що надходять від клієнтів, але й взаємодіють із зовнішніми системами, забезпечуючи необхідну інтеграцію та обмін даними, після чого повертають клієнтам оброблені результати. Загалом, у системі буде реалізовано сім таких сервісів, кожен з яких відповідає за взаємодію з окремими зовнішніми репозиторіями та має свої специфічні функціональні можливості. Детальний опис необхідних сервісів та їх функцій наведено у таблиці 3.4.

Таблиця 3.4 – Сервіси

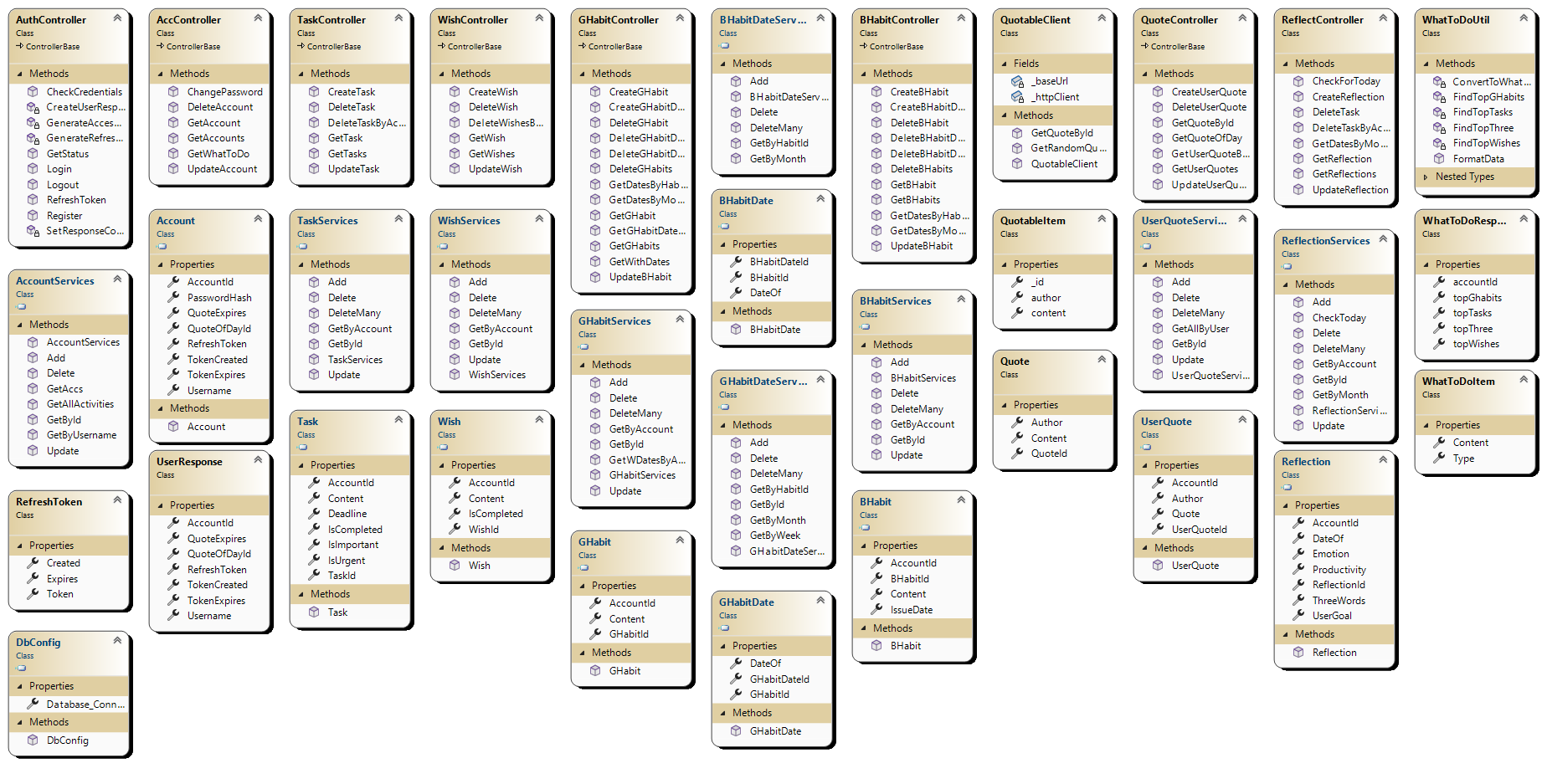
|  |  |
| --- | --- |
| Назва | Опис |
| AccService | Відповідає за операції з акаунтами |
| BHabitService | Відповідає за операції з шкідливими звичками та датами шкідливих звичок |
| GHabitService | Відповідає за операції з корисними звичками та датами корисних звичок |
| QuoteService | Відповідає за операції з цитатами з сервісу та з цитатами користувача |
| ReflectService | Відповідає за операції з опитуваннями |
| TaskService | Відповідає за операції з задачами |
| WishService | Відповідає за операції з бажаннями |

На рівні DAL (Data Access Layer) розташовані репозиторії, які відповідають за доступ до даних та їхнє управління. Репозиторії відіграють критичну роль у забезпеченні стабільного та ефективного збереження і вилучення даних. Вони виконують такі завдання, як зберігання даних, оновлення, видалення та пошук інформації у базі даних. Кожен репозиторій спеціалізується на роботі з певним типом даних або об'єктів, що дозволяє оптимізувати процеси обробки інформації. У нашій системі буде реалізовано дев’ять репозиторіїв, кожен з яких інтегрується з відповідними зовнішніми базами даних та забезпечує різноманітні функціональні можливості для роботи з ними. Детальний опис кожного репозиторію та їхніх функцій можна знайти у таблиці 3.5.

Таблиця 3.5 – Репозиторії

|  |  |
| --- | --- |
| Назва | Опис |
| AccountRepository | Відповідає за операції з таблицею акаунтів |
| BHabitRepository | Відповідає за операції з таблицею з шкідливими звичками |
| BHabitDateRepository | Відповідає за операції з таблицею з датами шкідливих звичок |
| GHabitRepository | Відповідає за операції з таблицею з корисними звичками |
| GHabitDateRepository | Відповідає за операції з таблицею з датами корисних звичок |
| UserQuoteRepository | Відповідає за операції з таблицею з цитатами користувача |
| ReflectRepository | Відповідає за операції з таблицею з опитуваннями |
| TaskRepository | Відповідає за операції з таблицею з задачами |
| WishRepository | Відповідає за операції з таблицею з бажаннями |

Структурна схема класів знаходиться в графічних матеріалах [КПІ.ІТ-0425.045440.06.99] та на Рисунку 3.2.

Рисунок 3.2 – Схема структурна класів

В якості системи управління базами даних використовується MsSQL[30]. База даних серверу призначена для зберігання користувачів, а також даних про їх задачі, бажання, корисні звички, шкідливі звички, цитати, опитування. Опис таблиць бази даних наведено у таблицях 3.6 - 3.14. Модель бази даних наведена на рисунку 3.3.

Таблиця 3.6 – Опис таблиці Accounts

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| AccountId | INT | Ідентифікаційний номер користувача |
| Username | VARCHAR(32) | Ім’я користувача |
| PasswordHash | VARCHAR(400) | Хеш пароля користувача |
| RefreshToken | VARCHAR(500) | Актуальний рефреш-токен користувача |
| TokenCreated | DATETIME | Дата та час створення токену |
| TokenExpires | DATETIME | Дата та час коли токен стане недійсним |
| QuoteOfDayId | VARCHAR(30) | Актуальна цитата дня |
| QuoteExpires | DATETIME | Час коли цитата стане недійсною |

Таблиця 3.7 – Опис таблиці Tasks

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| TaskId | INT | Ідентифікаційний номер задачі |
| AccountId | INT | Ідентифікаційний номер користувача |
| Content | VARCHAR(300) | Опис задачі |
| IsUrgent | BIT | Терміновість задачі |
| IsImportant | BIT | Важливість задачі |
| Deadline | DATETIME | Дедлайн |
| IsCompleted | BIT | Виконана задача чи ні |

Таблиця 3.8 – Опис таблиці Wishes

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| WishId | INT | Ідентифікаційний номер бажання |
| AccountId | INT | Ідентифікаційний номер користувача |
| Content | VARCHAR(300) | Опис бажання |
| IsCompleted | BIT | Виконане бажання чи ні |

Таблиця 3.9 – Опис таблиці GHabits

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| GHabitId | INT | Ідентифікаційний номер корисної звички |
| AccountId | INT | Ідентифікаційний номер користувача |
| Content | VARCHAR(300) | Опис корисних звичок |

Таблиця 3.10 – Опис таблиці GHabitDate

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| GHabitDateId | INT | Ідентифікаційний номер дати корисної звички |
| GHabitId | INT | Ідентифікаційний номер корисної звички |
| DateOf | DATETIME | Час створення дати корисної звички |

Таблиця 3.11 – Опис таблиці BHabits

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| BHabitId | INT | Ідентифікаційний номер дати шкідливої звички |
| AccountId | INT | Ідентифікаційний номер користувача |
| Content | VARCHAR(300) | Опис шкідливої звички |
| IssueDate | DATETIME | Час створення шкідливої звички |

Таблиця 3.12 – Опис таблиці BHabitDate

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| BHabitDateId | INT | Ідентифікаційний номер дати шкідливої звички |
| BHabitId | INT | Ідентифікаційний номер шкідливої звички |
| DateOf | DATETIME | Час створення дати шкідливої звички |

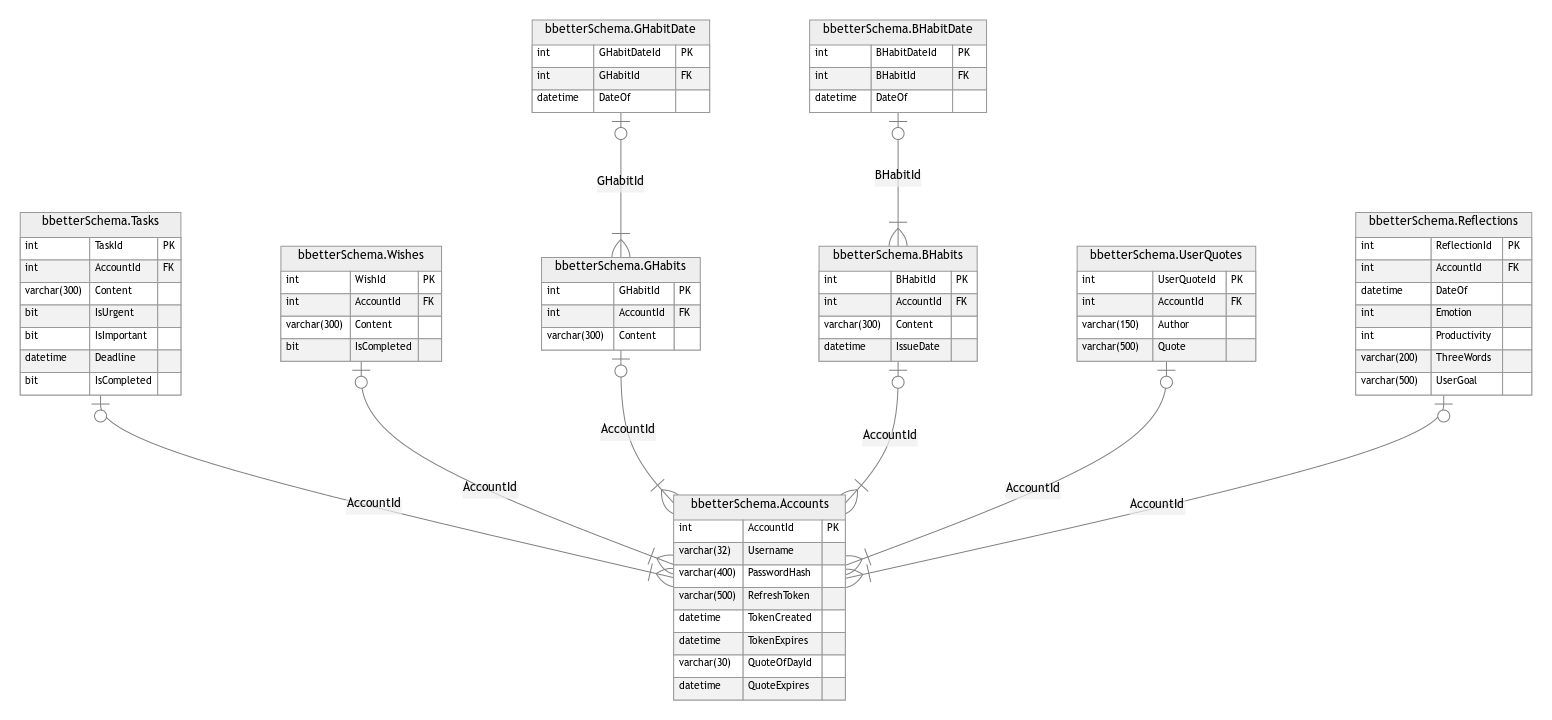
Таблиця 3.13 – Опис таблиці UserQuotes

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| UserQuoteId | INT | Ідентифікаційний номер цитати користувача |
| AccountId | INT | Ідентифікаційний номер користувача |
| Author | VARCHAR(150) | Ім’я та прізвище автора |
| Quote | VARCHAR(500) | Цитата |

Таблиця 3.14 – Опис таблиці Reflections

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| ReflectionId | INT | Ідентифікаційний номер цитати користувача |
| AccountId | INT | Ідентифікаційний номер користувача |
| DateOf | DATETIME | Час проходження опитування |
| Emotion | INT | Оцінка емоційності |
| Productivity | INT | Оцінка продуктивності |
| ThreeWords | VARCHAR (200) | Три слова, які описують минулий тиждень |
| UserGoal | VARCHAR(500) | Плани користувача на наступний тиждень |

Структурна схема бази даних знаходиться в графічних матеріалах [КПІ.ІТ-0425.045440.06.99] та на Рисунку 3.2.

Рисунок 3.3 – Схема бази даних

Перелік утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці, представлений у таблицях 3.15-3.16.

Таблиця 3.15 – Опис утиліт

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва утиліти | Опис застосування |
| 1 | Visual Studio 2022 | Головне середовище розробки програмного забезпечення серверної частини курсової роботи. |
| 2 | Google Chrome | Програмне забезпечення необхідне для використання веб-застосунку. |
| 3 | Azure Data Studio | Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних. |
| 4 | Visual Studio Code | Головне середовище розробки програмного забезпечення клієнтської частини курсової роботи. |
| 5 | Draw.io | Веб-застосунок, використаний при створенні та редагуванні діаграм та схем. |
| 6 | React.js | Бібліотека, що використовувалась для розробки програмного забезпечення клієнтської частини курсової роботи. |

Таблиця 3.16 – Опис бібліотек

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва бібліотеки | Опис застосування |
| 1 | AutoMapper[34] | Бібліотека для .NET, яка допомогає автоматизувати відображення даних між об’єктами |
| 2 | BCrypt.Net-Next[33] | Бібліотека для C#, яка надає можливість хешування паролів за допомогою алгоритму хешування BCrypt. |
| 3 | Dapper[29] | Легковагова бібліотека, яка дозволяє виконувати операції бази даних з використанням об’єктно-орієнтованого підходу. |
| 4 | Microsoft.AspNetCore.  Authentication.JwtBearer[35] | Пакет для роботи з аутентифікацією на основі токенів JSON Web Token (JWT). |
| 5 | Newtonsoft.Json[36] | Бібліотека для серіалізації та десералізації об’єктів JSON. |
| 6 | Microsoft.IdentityModel.  Tokens[37] | Частина бібліотеки для роботи з аутентифікацією та авторизацією. Надає функціональність для роботи з токенами безпеки. |
| 7 | Swashbuckle.AspNetCore[38] | Бібліотека для роботи з документацією та тестування API у ASP.NET Core. Надає можливість автоматично створювати документацію для веб-сервісу на основі коду API та атрибутів маршрутизації |
| 8 | Microsoft.Data.SqlClient[39] | Бібліотека для роботи з базами даних SQL Server в .NET |

Тексти програмного коду наведені в окремому документі «Текст програми». [КПІ.ІТ-0425.045440.03.12]

## Аналіз безпеки даних

Для передачі інформації буде застосовуватися захищений протокол HTTPS замість стандартного HTTP. Це забезпечить користання технології SSL для захисту даних, що включає шифрування інформації під час передачі між сервером і клієнтом, підвищуючи конфіденційність і безпеку обміну даними. Такий підхід допоможе запобігти несанкціонованому доступу та крадіжці даних у мережі.

Бібліотека ASP.NET Core Identity[31] забезпечує автоматизоване та безпечне хешування паролів користувачів за допомогою криптографічних хеш-функцій. При реєстрації нового користувача пароль спочатку перетворюється на хеш-значення, яке потім зберігається в базі даних. Як криптографічна хеш-функція використовується BCrypt[33] для генерації хешу паролю. До паролю користувача додається «сіль» - унікальне значення, яке робить хеш більш стійким до атак перебором. Потім використовується певна кількість ітерацій (зазвичай 12 або більше), щоб зробити хешування більш обчилслювально складним. Пароль з сіллю багаторазово шифрується за допомогою алгоритму Blowfish. Це забезпечує стійкість до розкриття оригінального паролю навіть у випадку злому бази даних.

Для захисту від SQL-ін'єкцій використовується Dapper[29] за рахунок використання параметризованих запитів. При використанні Dapper для виконання запитів до бази даних, зазвичай передається SQL-команда разом з параметрами, які відокремлені від тексту запиту. Це дозволяє Dapper автоматично обробляти ці параметри без ризику виконання шкідливого коду.

## Висновки до розділу

У даному розділі було проведене детальне моделювання та конструювання програмного забезпечення з урахуванням різноманітних аспектів. Спочатку була розглянута архітектура програмного забезпечення, включаючи аналіз та обговорення структурних компонентів системи. Проведено опис принципу роботи тришарової архітектури та архітектури «клієнт-сервер» разом з використанням відповідних схем, що допомогли у розумінні взаємодії між їх компонентами.

Крім того, був детально обгрунтований вибір засобів розробки, таких як інтегроване середовище розробки (IDE), мова програмування, фреймворки та бібліотеки, що найкраще відповідали вимогам та потребам проекту. Проведено аналіз та описано структуру бази даних, включаючи зв’язки між даними та розміщені в ній таблиці.

Окремо були розглянуті таблиці бази даних, які були створені за допомогою відповідних бібліотек, що допомогли у зручному та ефективному взаємодії з даними. Також була створена таблиця, яка включала у себе повний перелік усіх використаних утиліт та бібліотек під час розробки та проектування.

Під час аналізу безпеки даних детально розглядалося питання захисту від SQL-ін’єкцій, методи хешування паролів користувачів та забезпечення безпечної передачі інформації між клієнтом та сервером. Особлива увага була приділена використанню відповідних методів та технологій для забезпечення надійності та конфіденційності обробки даних у системі.

# АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Аналіз якості ПЗ

<У підрозділі викладають:

аналіз якості ПЗ за певними метриками.>

Метою тестування є наступне:

* перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
* перевірка збереження даних;
* перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, …);
* перевірка сумісності застосунку з різними операційними системами (Windows, Linux, …);
* знаходження проблем, помилок і недоліків з метою їх усунення;
* перевірка зручності графічного інтерфейсу.

Метриками для оцінки якості ПЗ обрано наступні:

* швидкість …

## Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

<У підрозділі викладають:

опис процесів тестування та приклади тестів, опишіть не більше 15 тестів.>

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.3 – 4.30.

Таблиця 4.3 – Тест 1.1 Реєстрація користувача

|  |  |
| --- | --- |
| Початковий стан системи | Користувач знаходиться на сторінці реєстрації |
| Вхідні дані | Електронна пошта, пароль, підтвердження паролю |
| Опис проведення тесту | У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 10 до 64 символів, який містить хоча б з одну англійську літеру, одне число і один спеціальний символ, і який не входить у топ 10000 найпопулярніших паролей, підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього натискається кнопка підтвердження реєстрації. |
| Очікуваний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації. |
| Фактичний результат | Збігається з очікуваним. |

## Опис контрольного прикладу

<У підрозділі викладають:

повний опис контрольного прикладу з усіма можливими розгалуженнями та особливостями. Кроки доповнюють ілюстраціями.>

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розгортання програмного забезпечення

<У підрозділі викладають:

Аналіз засобів розгортання (за потреби) і повний опис покрокового розгортання ПЗ. Кроки доповнюють ілюстраціями (діаграма розгортання).>

## Супровід програмного забезпечення

Інструкція користувача / програміста / системного адміністратора наведена в окремому документі.

<У підрозділі викладають:

опис того, як буде виконуватись підтримка програмного забезпечення. Ілюстрації.>

Користувачі повинні мати можливість отримати нову версію консольного застосунку з кожною версією. До того ж кожна нова версія консольного застосунку повинна бути опублікована в npm. Для автоматизації цього процесу був використаний сервіс GitHub Actions.

Створення нового випуску починається, коли нова версія консольного застосунку доставляється у репозиторій у гілку main, тобто коли commit має tag формату “v\*.\*.\*.”, де замість “\*” знаходиться число. Тоді у середовищі GitHub Actions встановлюється NodeJS. Після цього для проекту встановлюються залежності і проект збирається. Bash скрипт за допомогою бібліотеки pkg генерує виконувані файли (executables) для Linux і для Windows, та пакує файл для Linux у .deb пакет. Після цього .deb пакет і файл для Windows архівуються,

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# ВИСНОВКИ

У висновках викладають найважливіші наукові й практичні результати роботи та наводять:

* оцінку одержаних результатів і їх відповідність сучасному рівню наукових і технічних знань;
* ступінь впровадження та можливі галузі або сфери використання результатів роботи;
* наукову, науково-технічну, соціально-економічну значущість роботи;
* доцільність продовження досліджень за відповідною тематикою тощо.

Також у висновках необхідно відобразити стан вирішення усіх поставлених в дипломному проєктуванні задач.

В результаті виконання дипломного проєкту було спроєктовано …

В якості середовища розробки обрано …

У якості БД використано …

Після реалізації застосунку він був протестований на пристроях з різними версіями Android, з різними розмірами екранів щоб переконатися, що додаток акуратно відображається на різних пристроях.

Наукова новизна роботи (якщо вона є) полягає в наступному (достатньо вказати щось одне).

Вперше:

* реалізовано можливість запитів від пацієнта до лікаря;
* використано те-то, що дозволило те-то.

Модифіковано:

* те-то, що дозволило те-то.

Набуло подальший розвиток:

* те-то, що дозволило те-то.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Use of Filmed Reflection to Explore Themes of Struggle, Meaning, and Growth at Harvard Dental and Medical Schools / Jeffrey Herralla – 2020 – [Електронний ресурс] – Режим доступу до ресурсу: <https://dash.harvard.edu/bitstream/handle/1/37364911/HERRALA-SCHOLARLYPROJECT-2020.pdf?sequence=1&isAllowed=y>
2. The Effect of Tracking and Reflecting on Study Habits on Study Behavior and Grades / Donghee Yvette Wohn – 2020 – [Електронний ресурс] – Режим доступу до ресурсу: <https://www.researchgate.net/publication/352801064_The_Effect_of_Tracking_and_Reflecting_on_Study_Habits_on_Study_Behavior_and_Grades>
3. Atomic Habits: An Easy & Proven Way to Build Good Habits & Break Bad Ones / James Clear – 2018 – 27 c.
4. The Sung Diagram: Revitalizing the Eisenhower Matrix / David Millman, Hannah Bratterud, Mac Burgress, Britanny Terese Fasy – 2020 – [Електронний ресурс] – Режим доступу до ресурсу: <https://www.researchgate.net/publication/343686873_The_Sung_Diagram_Revitalizing_the_Eisenhower_Matrix>
5. Quitzilla [Електронний ресурс] – Режим доступу до ресурсу: <https://www.quitzilla.com/>
6. Todoist [Електронний ресурс] – Режим доступу до ресурсу: <https://app.todoist.com/auth/login>
7. Daily Habits [Електронний ресурс] – Режим доступу до ресурсу: <https://www.dailyhabits.xyz/>
8. JSON Web Tokens Introduction [Електронний ресурс] – Режим доступу до ресурсу: <https://jwt.io/introduction>
9. OAuth 2.0 [Електронний ресурс] – Режим доступу до ресурсу: <https://oauth.net/2/>
10. OWASP HttpOnly [Електронний ресурс] – Режим доступу до ресурсу: <https://owasp.org/www-community/HttpOnly>
11. MDN Glossary: MVC [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>
12. InterviewBit: Client-Server Architecture [Електронний ресурс] – Режим доступу до ресурсу: <https://www.interviewbit.com/blog/client-server-architecture/>
13. University of Waterloo: Layered Architecture PDF [Електронний ресурс] – Режим доступу до ресурсу: <https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1171/Arch_Design_Activity/Layered.pdf>
14. Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: <https://visualstudio.microsoft.com/>
15. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: <https://code.visualstudio.com/>
16. JetBrains Rider [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/rider/>
17. MonoDevelop [Електронний ресурс] – Режим доступу до ресурсу: <https://www.monodevelop.com/>
18. JetBrains IntelliJ IDEA [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/idea/>
19. .NET Learning: Back-End Web Development [Електронний ресурс] – Режим доступу до ресурсу: <https://dotnet.microsoft.com/en-us/learn/back-end-web-dev>
20. React Learning [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/learn>
21. Styled Components [Електронний ресурс] – Режим доступу до ресурсу: <https://styled-components.com/>
22. Formik Field API [Електронний ресурс] – Режим доступу до ресурсу: <https://formik.org/docs/api/field>
23. Axios API Introduction [Електронний ресурс] – Режим доступу до ресурсу: <https://axios-http.com/docs/api_intro>
24. React Router [Електронний ресурс] – Режим доступу до ресурсу: <https://reactrouter.com/en/main>
25. TanStack Query [Електронний ресурс] – Режим доступу до ресурсу: <https://tanstack.com/query/latest/docs/framework/react/overview>
26. Font Awesome [Електронний ресурс] – Режим доступу до ресурсу: <https://fontawesome.com/>
27. Material-UI [Електронний ресурс] – Режим доступу до ресурсу: <https://mui.com/material-ui/>
28. Day.js [Електронний ресурс] – Режим доступу до ресурсу: <https://day.js.org/>
29. Learn Dapper [Електронний ресурс] – Режим доступу до ресурсу: <https://www.learndapper.com/>
30. Microsoft Learn: SQL Server [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16>
31. Microsoft Learn: ASP.NET Core Identity [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio>
32. GitHub: Quotable [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/lukePeavey/quotable>
33. GitHub: bcrypt.net [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/BcryptNet/bcrypt.net>
34. AutoMapper [Електронний ресурс] – Режим доступу до ресурсу: <https://automapper.org/>
35. Microsoft Learn: JWT Bearer Authentication [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/ru-ru/dotnet/api/microsoft.aspnetcore.authentication.jwtbearer?view=aspnetcore-8.0>
36. Newtonsoft JSON [Електронний ресурс] – Режим доступу до ресурсу: <https://www.newtonsoft.com/json>
37. Microsoft Learn: IdentityModel.Tokens [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/api/microsoft.identitymodel.tokens?view=msal-web-dotnet-latest>
38. GitHub: Swashbuckle.AspNetCore [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/domaindrivendev/Swashbuckle.AspNetCore>
39. Microsoft Learn: SQLClient Namespace [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/sql/connect/ado-net/introduction-microsoft-data-sqlclient-namespace?view=sql-server-ver16>

# ДОДАТКИ