**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](#_Toc166681065)

[1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 7](#_Toc166681066)

[1.1 Аналіз предметної області 7](#_Toc166681067)

[1.2 Аналіз існуючих рішень 9](#_Toc166681068)

[1.2.1 Аналіз відомих програмних продуктів 9](#_Toc166681069)

[1.2.2 Аналіз відомих алгоритмічних та технічних рішень 11](#_Toc166681070)

[1.3 Опис бізнес-процесів 15](#_Toc166681071)

[1.4 Постановка задачі 16](#_Toc166681072)

[Висновки до розділу 17](#_Toc166681073)

[2 РОЗРОБЛЕННЯ вимог до програмного забезпечення 18](#_Toc166681074)

[2.1 Варіанти використання програмного забезпечення 18](#_Toc166681075)

[2.2 Аналіз системних вимог 19](#_Toc166681076)

[2.3 Розроблення функціональних вимог 19](#_Toc166681077)

[2.4 Розроблення нефункціональних вимог 21](#_Toc166681078)

[Висновки до розділу 21](#_Toc166681079)

[3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 22](#_Toc166681080)

[3.1 Архітектура програмного забезпечення 22](#_Toc166681081)

[3.2 Обґрунтування вибору засобів розробки 23](#_Toc166681082)

[3.3 Конструювання програмного забезпечення 23](#_Toc166681083)

[3.4 Аналіз безпеки даних 26](#_Toc166681084)

[Висновки до розділу 26](#_Toc166681085)

[4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 27](#_Toc166681086)

[4.1 Аналіз якості ПЗ 27](#_Toc166681087)

[4.2 Опис процесів тестування 27](#_Toc166681088)

[4.3 Опис контрольного прикладу 28](#_Toc166681089)

[Висновки до розділу 28](#_Toc166681090)

[5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 29](#_Toc166681091)

[5.1 Розгортання програмного забезпечення 29](#_Toc166681092)

[5.2 Супровід програмного забезпечення 29](#_Toc166681093)

[Висновки до розділу 29](#_Toc166681094)

[ВИСНОВКИ 30](#_Toc166681095)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 31](#_Toc166681096)

[ДОДАТКИ 32](#_Toc166681097)

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. Не менш важливою проблемою також є зріст напруженності у зв’язку з війною в Україні. 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 я обрав 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;

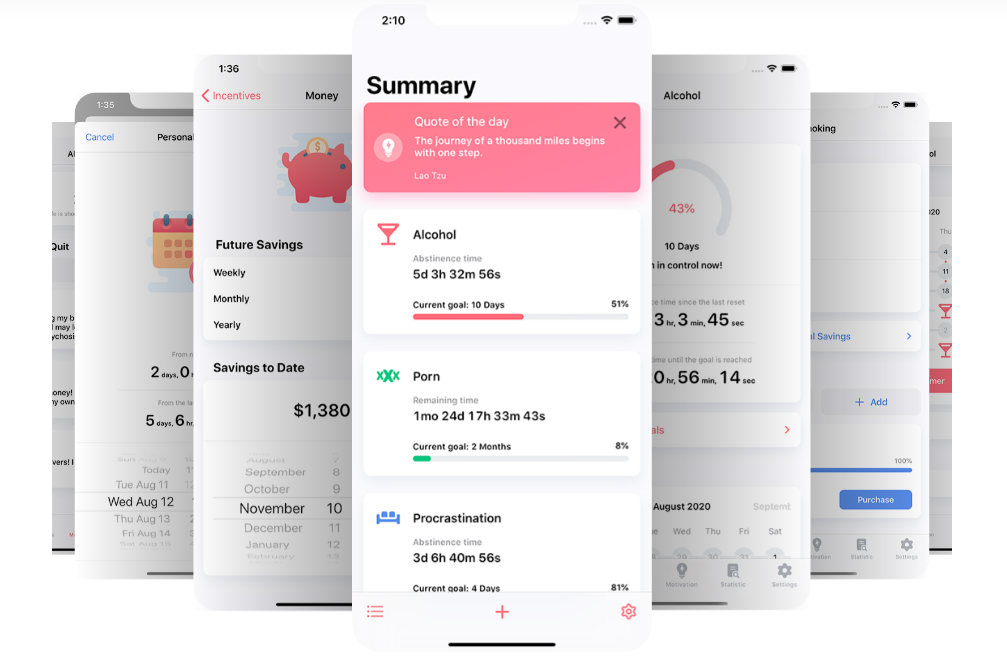


Рисунок 1.1 – Мобільний застосунок Quitzilla[5]

#### Todoist

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

Переваги:

* доступний а різних пристроях, що дозволяє користувачам синхронізувати свої завдання на всіх пристроях;
* має простий та зрозумілий інтерфейс;
* інтерфейс простий і зрозумілий;
* 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 зі шкідливими звичками;

деяким користувачам може не вистачати запропонованих мотиваційних інструметів для успішної відмови від звичок;

#### DailyHabits

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, архітектура, допоміжні програмні засоби та засоби розробки

#### 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 [5].

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 [6].

Для свого програмного продукту я обрав 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) [7]. Завдяки цьому токен оновлення не зможе бути викраденим.

#### 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 [8]. 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 [9]. 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 [10].

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

#### Аналіз допоміжних програмних засобів та 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 [11].
* 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 [12].
* 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 [13].
* 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 [14].
* 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 [15].

Для розробки веб-застосунку я зупинився на 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 [16] 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 архітектур 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 [17] - 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.

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.

## Опис бізнес-процесів

<У підрозділі викладають:

опис бізнес-процесів вашої розробки та їх наочне представлення за допомогою засобів моделювання (BPMN та інші).>

Для опису бізнес процесу використовується BPMN модель (рисунок 1.1).

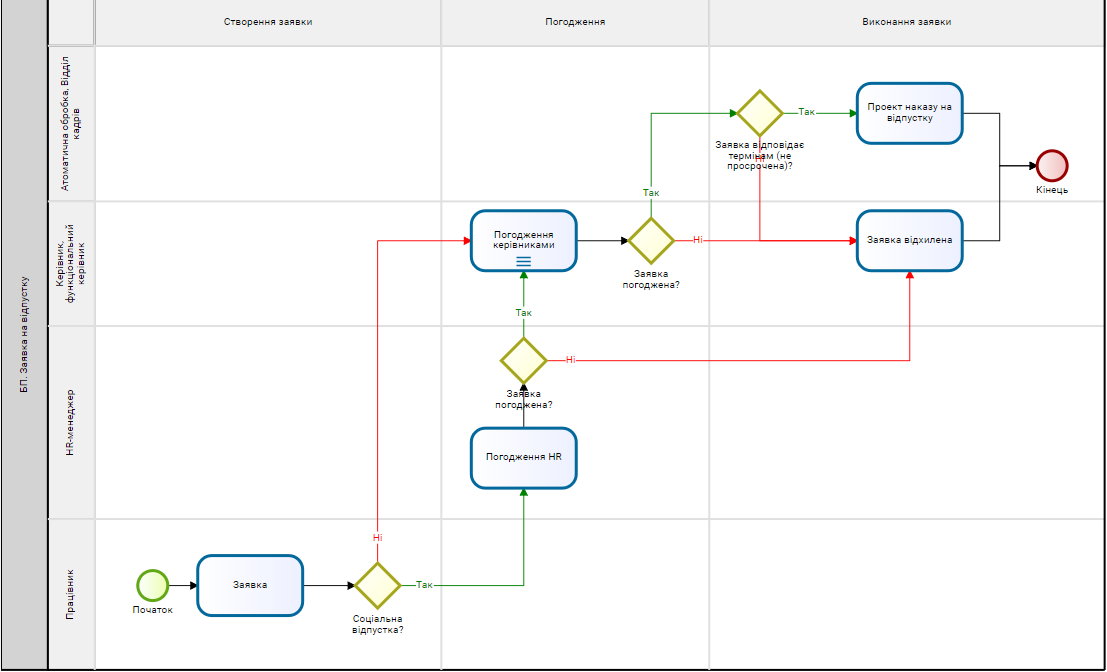


Рисунок 1.1 – Схема бізнес-процесу

Опис послідовності створення облікового запису користувача:

* користувач переходить на сторінку реєстрації;
* користувач заповнює поля реєстрації;
* якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою.

## 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 я детально розглянув та описав обрану мною 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 ми порівняли з дипломним проєктом у таблиці 1.1. Також було проаналізовано алгоритмічні рішеняя, архітектуру та допоміжні програмні засоби та засоби розроби. 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 активностями, проведеню опитувань, отриманню випадкової цитати. Завдяки цьому ми докладно вивчили та уточнили 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

<У розділі викладають:

опис головного(-х) функціоналу розробки, діаграму варіантів використання та опис варіантів використання.>

Головною функцією програмного забезпечення є …, більше функцій можна побачити на рисунку 2.1.

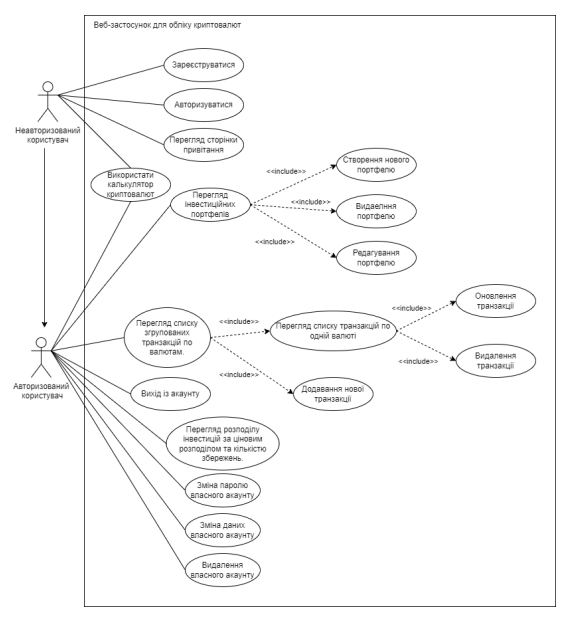


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.2 - 2.14 наведені варіанти використання програмного забезпечення.

Таблиця 2.2 - Варіант використання UC-1

|  |  |
| --- | --- |
| Use case name | Реєстрація користувача |
| Use case ID | UC-01 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Гість (незареєстрований користувач) |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: пошта користувача, пароль в системі, та його повтор для підтвердження, а також чек бокс для підтвердження умов сервісу. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу. |
| 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 сторінку входу |

## Аналіз системних вимог

<У підрозділі викладають:

системні вимоги – підрозділ опціональний, залежить від роботи.>

## Розроблення функціональних вимог

<У підрозділі викладають:

модель вимог, їх опис та матрицю трасування.>

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.15 наведено загальну модель вимог, а в таблицях 2.16 – 2.27 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.3.

Таблиця 2.15 – Загальна модель вимог

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | ID вимоги | Пріоритети | Ризики |
| 1 | Перегляд сторінки привітання. | FR-1 | Низький | Низький |
| 2 | Система авторизації користувача. | FR-2 | Високий | Високий |
| 2.1 | Реєстрація користувача. | FR-3 | Високий | Високий |
| 2.2 | Авторизація користувача. | FR-4 | Високий | Високий |
| 2.3 | Вихід із акаунту. | FR-5 | Середній | Низький |

Таблиця 2.16 – Перелік функціональних вимог

|  |  |
| --- | --- |
| ID вимоги | Назва та опис |
| FR-1 | Перегляд сторінки привітання.  Неавторизований користувач бачить веб-сторінку привітання. На ній же можна обрати можливість зареєструватися або авторизуватися. |
| FR-2 | … |

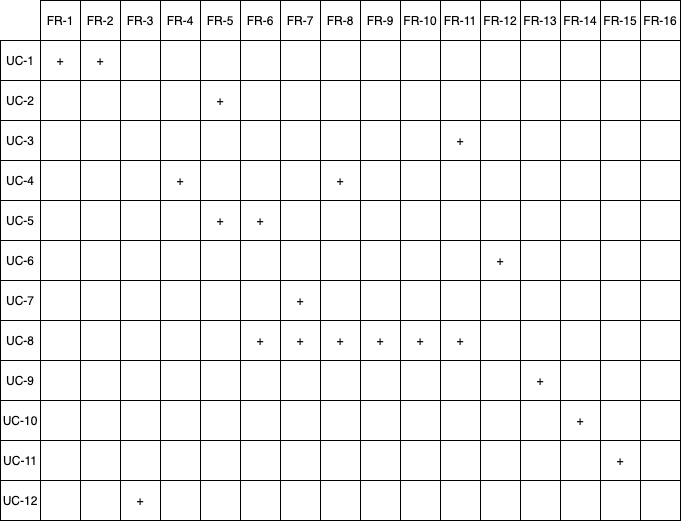


Рисунок 2.3 – Матриця трасування вимог

## Розроблення нефункціональних вимог

<У підрозділі викладають:

опис нефункціональних вимог.>

## Висновки до розділу

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

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення. <окремий документ за шаблоном «ДП\_шаблон\_ч1\_ТЗ» >

# КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Архітектура програмного забезпечення

<У підрозділі викладають:

тип архітектури (архітектурний паттерн) для вашого ПЗ та деталізація його компонентів за необхідності. Необхідно виконати повний опис архітектури. Для цього можуть бути використані (один з варіантів):

* діаграми C4 Model:
* архітектурні діаграми для представлення високорівневої архітектури (рівні 1 та 2 моделі C4 Model представлення архітектури);
* архітектурні діаграми для детального представлення архітектури (рівні 3 та 4);
* діаграми в нотації UML:
* діаграми компонентів та діаграми пакетів;
* діаграми для представлення поведінки (наприклад, діаграми послідовностей);
* діаграми класів для представлення логічної моделі даних;
* інші

Також опишіть та обґрунтуйте важливі архітектурні рішення. Наприклад:

* управління користувачами, як відбувається логін користувача в систему – за допомогою single-sign on, чи програмне забезпечення зберігає акаунти користувачів і виконує логін самостійно;
* інтеграції із зовнішніми системами, які системи та які протоколи використовуються;
* як саме забезпечується виконання нефункціональних вимог;
* які типи баз даних вибрані.

Вкажіть, який технологічний стек вибрано та обґрунтуйте рішення.>

## Обґрунтування вибору засобів розробки

<У підрозділі викладають:

опис допоміжних програмних засобів та їх порівняльний аналіз, якщо існує кілька аналогів (мова йде про сторонні бібліотеки, пакети, фреймворки, тощо). Розглядаються засоби розробки за допомогою яких можна виконати вашу розробку, робиться їх порівняльний аналіз та на його основі обираються ті, що вам підходять (маються на увазі мови програмування та IDE).>

## Конструювання програмного забезпечення

<У підрозділі викладають:

* опис оригінальних алгоритмів чи модифікацій існуючих;
* опис структур даних, програмних структур та ін.
* опис бази даних з представленням концептуальної, логічної чи фізичної моделі та з описом сутностей чи таблиць;
* опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці. >

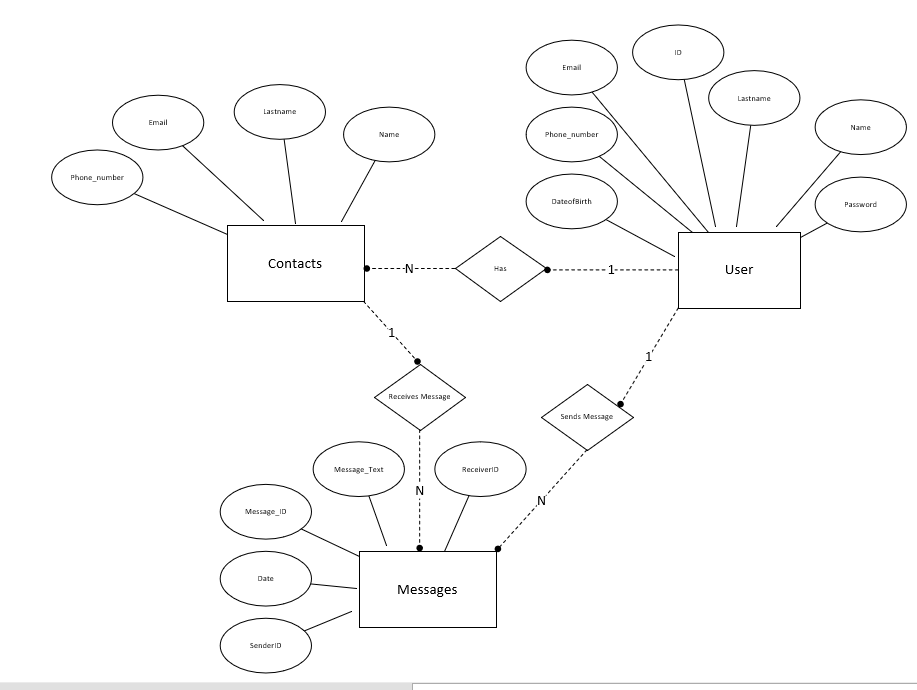


Рисунок 3.2 – ER діаграма сутностей User, Messages, Contacts

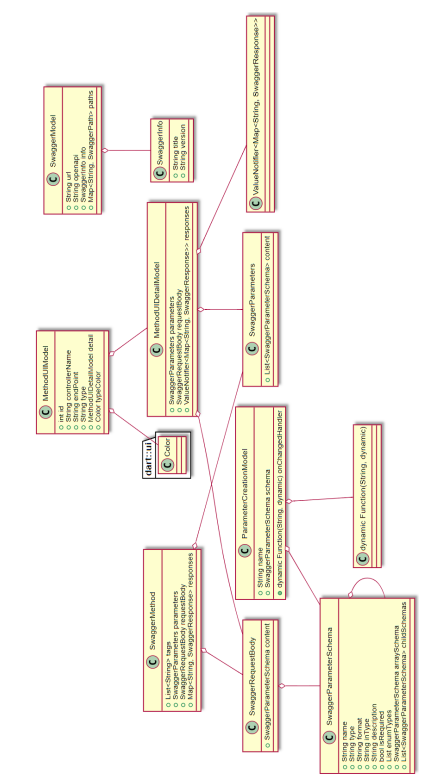


Рисунок 3.3 – Схема структурна класів

Бібліотека xpring, яка виконує запити в мережу блокчейн – однопоточна, тобто не може оброблювати паралельні запити. Через таку специфіку бібліотеки, довелось розробляти рішення яке б не блокувало запити з різних потоків, або інших частин коду. За основу ідеї алгоритму була взята багатопоточність самої мови програмування java. Java для вирішення проблем з потоками представляє ділянки коду які можна синхронізувати, що унеможливлює одночасний доступ до нього з різних потоків. Ця синхронізація проходить за допомогою передачі управління об’єкта-монітора. Алгоритм вирішення проблеми з доступом до однопоточної бібліотеки наведено на рисунку 3.4.

В якості системи управління базами даних використовується Postgres. База даних серверу призначена для зберігання користувачів, а також даних про їх …. Опис таблиць бази даних наведено у таблицях 3.11 - 3.14. Модель бази даних наведена на рисунку 3.12.

Таблиця 3.11 – Опис таблиці user

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | serial | ідентифікаційний номер користувача |
| email | varchar | електронна пошта користувача |
| password\_id | int | посилання на запис у таблиці password, де зберігається пароль користувача |

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.22.

Таблиця 3.22 – Опис утиліт

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва утиліти | Опис застосування |
| 1 | IntelliJ IDEA | Головне середовище розробки програмного забезпечення серверної частини курсової роботи. |
| 2 | Postman | Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів. |
| 3 | MySQL Workbench | Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних. |

Тексти програмного коду наведені в окремому документі «Текст програми».

## Аналіз безпеки даних

<У підрозділі викладають:

аналіз вразливостей ПЗ та будь-які питання пов’язані з безпекою даних.>

## Висновки до розділу

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

# 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. [Оформлення бібліографії ДСТУ 7.1:2006](https://drive.google.com/file/d/1VUr7fwKHOk8t1u8sh-Sv2dlEc_6OBoIS/view?usp=sharing)
2. [Оформлення бібліографії ДСТУ 8302-2015](https://drive.google.com/file/d/17RYEYVnSXolL0S8D1k85VfReYd2yC6WO/view?usp=sharing) з прикладами

# ДОДАТКИ