**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Інтегровані інформаційні системи»**
**спеціальності 126 «Інформаційні системи та технології»**
**на тему: «Вебзастосунок для планування роботи та концентрації»**

Виконав (-ла):
студент (-ка) IV курсу, групи ІА-92
Ємельяненко Ярослава Сергіївна _____

Керівник:
асистент
Майер Ілля Сергійович _____

Рецензент:
старший викладач кафедри ОТ
Алещенко Олексій Вадимович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2023 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра інформаційних систем та технологій**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

# ЗАВДАННЯ
## на дипломний проєкт студенту
## Ємельяненко Ярославі Сергіївні

1. Тема проєкту «Вебзастосунок для планування роботи та концентрації», керівник проєкту Майер Ілля Сергійович, асистент, затверджені наказом по університету від «31» травня 2023 р. №2101-с.

2. Термін подання студентом проєкту: «12» червня 2023 р.

3. Вихідні дані до проєкту: мова програмування Java та її документація, фреймворк Spring, фреймворк Spring Boot, фреймворк Vaadin, система управління базами даних MySQL, сервіс Amazon RDS.

4. Зміст пояснювальної записки: вступ, аналіз предметної області, аналіз існуючих рішень, формування вимог, опис програмного середовища, проєктування програмного застосунку, реалізація програмного застосунку, проєктуваня бази даних, керівництво з використання, висновки, перелік використаних джерел.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): архітектура застосунку, діаграма класів та пакетів, діаграма бази даних, діаграма діяльності.

6. Дата видачі завдання: 22 лютого 2023 року

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1 | Ознайомлення з підприємством | 21.04.2023 | |
| 2 | Аналіз існуючих рішень | 28.04.2023 | |
| 3 | Опис середовища проєктування | 28.04.2023 | |
| 4 | Розробка програмних модулів | 05.05.2023 | |
| 5 | Тестування ПЗ | 12.05.2023 | |
| 6 | Створення інструкцій | 19.05.2023 | |
| 7 | Оформлення документації | 06.06.2023 | |
| 8 | Подання готового проєкту | 12.06.2023 | |

Студент Ярослава ЄМЕЛЬЯНЕНКО

Керівник Ілля МАЙЕР

# АНОТАЦІЯ

Ємельяненко Я.С. Вебзастосунок для планування роботи та концентрації. КПІ ім. Ігоря Сікорського, Київ, 2023.

Пояснювальна записка дипломного проєкту містить 68 сторінок, 4 розділи, 31 рисунок, 2 таблиці, 1 додаток, 17 джерел та 4 кресленики.

Ключові слова: вебзастосунок, дедлайн, задача, концентрація, нотатка, планування, пріоритет, таймер, тайм-менеджмент.

Об'єкт дослідження: ефективний тайм-менеджмент у сучасному світі.

Мета роботи: розробка вебзастосунку, який надає користувачам можливість планувати свої задачі, зосереджуватись на виконанні роботи та підвищувати продуктивність.

Для розробки та підтримки вебзастосунку були використані такі інструменти: Java 17, фреймворки Vaadin, Spring, Spring Security, Spring Data JPA, Spring Boot, база даних MySQL, сервіс Amazon RDS.

У дипломному проєкті було розроблено систему, головна функція якої – допомогти користувачам упорядкувати свої завдання та покращити тайм-менеджмент.

Основні характеристики та показники: простий та інтуїтивний інтерфейс, ефективне управління задачами, підвищення концентрації користувачів.

Очікуваний результат впровадження системи: збільшення ефективності та продуктивності користувачів шляхом кращого управління часом та задачами, покращення організації робочого процесу та зниження ризику пропускання дедлайнів. Розробка може бути використана в подальшому для створення більш масштабної системи, придатної до користування у великих командах на комерційній основі.

# SUMMARY

Yaroslava Yemelianenko. Web application for work planning and concentration. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2023.

The explanatory note of the diploma project contains 4 chapters, contains 68 pages of text, 31 figures, 2 tables, 1 appendicy, 17 sources and 4 drawings.

Keywords: concentration, deadline, note, planning, priority, task, timer, time-management, web application.

Research object: effective time management in the modern world.

Job Objective: develop a web application that empowers users to plan their tasks, focus on work, and increase productivity.

The following tools were used to develop and support the web application: Java 17, Vaadin frameworks, Spring, Spring Security, Spring Data JPA, Spring Boot, MySQL, Amazon RDS.

In the diploma project a system whose main function is to help users organize their tasks and improve time-management was developed.

Main characteristics and indicators: simple and intuitive interface, effective task-management, increased user concentration.

The expected result of the implementation of the system: increased efficiency and productivity of users through better time and task management, improved organization of the work process and reduced risk of missing deadlines. The development can be used in the future to create a larger system suitable for use by large teams on a commercial basis.

| Номер рядка | Формат | Позначення | Найменування | Кільк. аркушів | Номер екзем. | Примітка |
|---|---|---|---|---|---|---|
| 1 | | | Документація загальна | | | |
| 2 | | | | | | |
| 3 | | | Знову розроблена | | | |
| 4 | | | | | | |
| 5 | А4 | ІА92.050БАК.004 ПЗ | Пояснювальна записка | 68 | | |
| 6 | А3 | ІА92.050БАК.004 Д1 | Архітектура застосунку | 1 | | |
| 7 | А3 | ІА92.050БАК.004 Д2 | Діаграма класів та пакетів | 1 | | |
| 8 | А3 | ІА92.050БАК.004 Д3 | Діаграма бази даних | 1 | | |
| 9 | А3 | ІА92.050БАК.004 Д4 | Діаграма діяльності | 1 | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |
| 21 | | | | | | |
| 22 | | | | | | |
| 23 | | | | | | |
| 24 | | | | | | |
| 25 | | | | | | |
| 26 | | | | | | |
| 27 | | | | | | |
| 28 | | | | | | |

| Зм. | Аркуш | № докум. | Підпис | Дата | ІА92.050БАК.004 ТП | | |
|---|---|---|---|---|---|---|---|
| Розроб. | | Ємельяненко Я.С. | | | Вебзастосунок для планування роботи та концентрації. Відомість проєкту | Літ. | Аркуш 1 / Аркушів 1 |
| Керівн. | | Майер І.С. | | | | Т | |
| Затв. | | | | | | КПІ ім. Ігоря Сікорського Група ІА-92 | |

# Пояснювальна записка
# до дипломного проєкту
# на тему: «Вебзастосунок для планування роботи та концентрації»

Київ – 2023 року

ЗМІСТ

# ВСТУП

Одним з ключових явищ, які породжують необхідність мати вебзастосунок для тайм-менеджменту, є зростаючий обсяг роботи та завдань у сучасному світі. Завдяки швидкому технологічному розвитку, люди стикаються зі збільшеним обсягом інформації, різноманітними професійними та особистими завданнями, а також невеликим часом для їх виконання.

Проблема тайм-менеджменту виникає в результаті складнощів у керуванні та раціональному використанні часу. Основні аспекти цієї проблеми включають недостатнє планування, нездатність визначати пріоритети, відсутність концентрації на роботі та схильність до прокрастинації.

Недостатнє планування може призводити до невиконання завдань вчасно або їх некоректного деліверінгу. Без належного планування, люди можуть переоцінювати свої можливості, надмірно завантажувати себе або не враховувати неочікувані обставини, що призводить до стресу та невдач.

Погана концентрація на роботі – також проблема, з якою стикаються багато людей. Вплив сучасних технологій, соціальних мереж, постійного доступу до інформації може розсіювати увагу та заважати зосередженості на важливих завданнях. Низька продуктивність, помилки та втрата часу можуть бути наслідками недостатньої уваги до роботи.

Неуміння встановлювати пріоритети може призвести до незбалансованого розподілу часу між різними завданнями: важливі задачі залишаються невиконаними, тоді як у менш важливі вкладають більше зусиль та часу, не отримуючи суттєвих результатів.

Прокрастинація є ще одним блокером, пов'язаним з ефективним тайм-менеджментом. Її визначають як «форму невдачі самоконтролю, що характеризується нераціональним затягуванням виконання завдань, незважаючи на потенційно негативні наслідки» [1]. Прокрастинація може призводити до недовиконання термінів, погіршення соціальних відносин та результатів роботи.

Ці проблеми впливають на продуктивність, якість роботи та загальний

добробут. Люди, які не вміють ефективно управляти часом та зосереджуватися на роботі, часто відчувають стрес, втрачають мотивацію та можуть бути незадоволені своїми досягненнями.

Кінцева мета тайм-менеджменту полягає в тому, щоб виконати більше за той самий проміжок часу і зробити це з меншими зусиллями та з меншою кількістю проблем. Навички ефективного тайм-менеджменту дають відчуття звільнення та контролю над особистим життям.

В умовах такого перенавантаження та зростаючої складності завдань виникає потреба у системі, що допоможе краще організувати час, пріоритизувати завдання, створювати розклади та нагадування. Вебзастосунок для тайм-менеджменту надає зручні та ефективні інструменти для планування, управління та контролю часу, нагадування про важливі завдання, та створює сприятливу робочу атмосферу. Він також може надати корисні засоби для покращення концентрації, боротьби з прокрастинацією та підвищення продуктивності. Такі застосунки можуть надати статистику, аналітику та звіти, які допоможуть відстежувати витрачений час, оцінювати результати та вдосконалювати особисту ефективність.

Впровадження розробок та досліджень у галузі тайм-менеджменту може мати позитивний вплив на організації, працівників та особисту продуктивність. Застосування нових інструментів, методик та підходів, які базуються на дослідженнях та розробках, може допомогти покращити тайм-менеджмент, забезпечити більшу ефективність та досягнення поставлених цілей. Результати таких досліджень та розробок можуть бути використані в різних сферах, включаючи освіту, бізнес, особистий розвиток та багато інших.

Об'єктом дослідження даного проєкту є ефективний тайм-менеджмент.

Предметом дослідження є проблема неефективного управління часом, низької зосередженості на роботі та прокрастинації.

Метою роботи є створення функціонального та користувацьки-орієнтованого вебзастосунку «Anchor/Assist», який надасть зручні та ефективні інструменти для планування, керування завданнями, підвищення концентрації та запобігання прокрастинації.

Завданнями, які необхідно вирішити для досягнення поставленої мети, є:

– розробка інтуїтивно зрозумілого інтерфейсу користувача, який забезпечить зручну навігацію та взаємодію зі застосунком;

– реалізація функціоналу для планування задач, встановлення пріоритетів та дедлайнів;

– розробка механізмів для стимулювання концентрації та уникнення розсіювання уваги під час виконання завдань;

– впровадження інструментів для запобігання прокрастинації, зокрема нагадувань, мотиваційних елементів.

Ці положення відрізняються від вже відомих методик та підходів тим, що вони спрямовані на інтеграцію різних аспектів тайм-менеджменту в єдиний вебзастосунок, що дозволяє забезпечити комплексний підхід до управління часом та покращення продуктивності.

Дипломний проєкт складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 17 найменувань, 1 додаток. Графічна частина включає 4 кресленики формату А3. Загальний обсяг 68 сторінок.

# 1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

## 1.1 Аналіз предметної області

Тайм-менеджмент – це концепція та набір методик, спрямованих на ефективне управління часом з метою досягнення поставлених цілей. Історія тайм-менеджменту сягає далеко назад, а його розвиток пов'язаний з потребою людей у більш організованому та продуктивному використанні часу.

У XX столітті із зростанням швидкості життя та збільшенням обсягу роботи виникла потреба у більш систематичному підході до управління часом. У 1911 році Фредерік Вінслоу Тейлор опублікував «The Principles of Scientific Management» – видатну працю, яка відіграла важливу роль у розвитку теорії та практики управління [2]. Тейлор пропонує систематичний підхід до організації робочих процесів з метою досягнення максимальної продуктивності та ефективності.

У цій книзі Тейлор висуває ідею, що управління повинно базуватися на наукових принципах та методах, а не на досвідчених, але недостатньо систематичних, підходах. Він пропонує розглядати працю як науковий процес, що може бути вивчений, виміряний та оптимізований.

Тейлор вводить такі ключові принципи наукового управління:

– розподіл ролей — він рекомендує чітко визначати ролі та обов'язки керівників та робітників; керівники повинні бути відповідальні за планування та контроль, а робітники — за виконання роботи;

– вивчення робочого процесу — Тейлор пропонує детально аналізувати та вивчати робочі процеси з метою виявлення найбільш ефективних методів виконання роботи;

– вибір та навчання робітників — він відзначає, що керівництво повинно залучати та навчати найкращих робітників, а також створювати системи навчання та підвищення кваліфікації;

– стандартизація робочих методів — Тейлор пропонує розробляти стандартизовані методи виконання роботи, що дозволяє досягати більшої ефективності та однаковості результатів;

– співпраця між керівництвом та робітниками – він підкреслює важливість співпраці та довіри між керівництвом та робітниками для досягнення спільних цілей.

У 1950-х роках фахівці з організації праці почали активно досліджувати та розробляти методи підвищення продуктивності праці та ефективного планування часу. Закон Паркінсона є концепцією, яку висунув британський історик та письменник Сідні Паркінсон у своїй книзі «Parkinson's Law: The Pursuit of Progress». Цей закон описує явище, коли робочий обсяг часу та ресурсів, витрачений на виконання певної роботи, розширюється до повного доступного часу для її виконання, незалежно від реальної необхідності або складності завдання. Як наслідок – згаяний час, збільшення бюрократії та надмірне навантаження на ресурси.

Закон Паркінсона може бути застосований до різних сфер життя та діяльності, включаючи управління проєктами, бізнес-процеси, бюрократичні структури та організаційні системи загалом.

Важливо зрозуміти, що закон Паркінсона не є науковим законом у суворому сенсі, а скоріше представляє собою емпіричне спостереження і загальний принцип, який може пояснити деякі явища у реальному світі. Він наголошує на важливості ефективного управління часом, установленні пріоритетів та уникненні надмірної бюрократії.

Застосування принципів ефективного управління та технік тайм-менеджменту може допомогти уникнути впливу закону Паркінсона, раціоналізувати процеси роботи та забезпечити більш продуктивне використання ресурсів.

Техніки тайм-менеджменту є цінним інструментарієм для ефективного розподілу часу і управління задачами. Вони допомагають підвищити продуктивність, зменшити стрес, покращити організацію робочого часу і досягати поставлених цілей. Серед популярних методик тайм-менеджменту можна виділити матрицю Ейзенхауера, техніку «Помідор», метод Парето (80/20), техніку "Порція

на день", зворотну психологію термінів, техніку "Календарний блокувальник" та "Список завдань".

Матриця Ейзенхауера допомагає пріоритизувати завдання за двома факторами – важливістю і терміновістю. Завдання класифікуються у чотири категорії: важливі та термінові, важливі, але не термінові, термінові, але не важливі, і не важливі і не термінові. Це допомагає фокусуватися на найважливіших завданнях і уникати відволікань.

Техніка «помідора» – метод, який базується на принципі роботи протягом фіксованих періодів часу, називаних "помідорами" (зазвичай 25 хвилин), і відпочинком протягом короткого періоду між ними. Це допомагає зберегти фокус і ефективніше виконувати завдання.

Метод Парето (80/20) ґрунтується на принципі, що близько 80% результатів досягаються через 20% зусиль. Виокремлення найважливіших 20% завдань і пріоритизація їх допомагає зосередитися на найбільш важливих і впливових аспектах роботи.

Техніка "порція на день" полягає в плануванні лише обмеженої кількості завдань на кожен день. Вибір конкретних завдань допомагає уникнути перенавантаження і зосередитися на виконанні обмеженого числа важливих робіт.

Зворотна психологія термінів – техніка, яка полягає в тому, щоб надати собі суворі терміни для завершення завдань. Створення внутрішнього тиску може сприяти більш швидкому та ефективному виконанню роботи.

Техніка "Календарний блокувальник", що включає розподіл часу на блоки заздалегідь заданої тривалості для конкретних видів робіт або завдань. Це допомагає створити структуру і організувати робочий день.

Техніка "списку завдань" – простий, але ефективний метод, який полягає в складанні списку завдань, які потрібно виконати. Це допомагає зберегти загальне уявлення про завдання і контроль над ними.

Вибір технік тайм-менеджменту залежить від індивідуальних потреб і стилю роботи кожної людини. Рекомендується експериментувати з різними техніками та

адаптувати їх до своєї особистої ситуації, щоб знайти найбільш ефективний спосіб керування часом.

Для імплементації таких технік та спрощення тайм-менеджменту використовуються відповідні застосунки, що допомагають покращити продуктивність, збалансувати робочі та особисті завдання, знизити стрес і досягти більшої ефективності.

Основною метою систем тайм-менеджменту є створення систематизованого підходу до організації та планування часу. Вони надають структуру та методологію для керування завданнями, пріоритизації роботи, управління термінами та контролю продуктивності.

Системи тайм-менеджменту можуть включати різні компоненти, такі як:

– планування завдань – основна складова частина таких систем, що допомагають скласти список завдань, визначити їх пріоритети та розподілити час для виконання кожного завдання;

– організація календаря – включає розподіл завдань у часові слоти, планування зустрічей та подій, а також нагадування про важливі терміни;

– трекінг часу – компонент, який дозволяє відстежувати, як ви витрачаєте свій час, що допомагає оцінити продуктивність і знайти можливості для оптимізації;

– нагадування та сповіщення – функція, що допомагає нагадати про важливі події, дедлайни та завдання;

– керування пріоритетами – система дає змогу встановити пріоритет завдання, завдяки чому можна відсіяти менш важливе і зосередитись на тому, що треба зробити у першу чергу.

Одними з ворогів ефективного тайм-менеджменту є прокрастинація та низька концентрація, неможливість зосередитись на вже поставленій та описаній задачі. Вони можуть мати різні причини, такі як відчуття незацікавленості або недосконалості щодо завдання, страх неуспіху або перенавантаження, бажання уникнути незручностей або відчуття дискомфорту, або недостатня організація та

планування часу. Існують різні методи та практики, які можна застосовувати для покращення концентрації на завданні.

Важливим є планування і пріоритизація: перед початком роботи треба чітко визначити мету та пріоритети завдання. Складання детального плану дій допомагає зосередитись на потрібних діях і уникнути розсіяності.

Створення робочого оточення, яке сприяє концентрації, також є важливим аспектом. Необхідно вимкнути відволікаючі фактори, такі як мобільні пристрої, соціальні медіа або шум, і створити комфортну, тиху обстановку для праці.

Використання методів організації часу, таких як «Помодоро» або техніка «Глибокої роботи», може допомогти створити структурований графік роботи, в якому періоди фокусу і перерви чергуються, що дозволяє підтримувати високу концентрацію на завданні.

Регулярне проведення вправ для розслаблення і медитаційних практик сприяє зняттю стресу, покращенню психічного спокою і зосередженості. Такі вправи можуть включати глибоке дихання, йогу або майндфулнес-практики.

Важливо усвідомлювати власні відволікаючі фактори і розробляти стратегії для їх уникнення або мінімізації. Це може включати вимкнення сповіщень на пристроях, створення режимів без доступу до соціальних медіа або встановлення чітких меж для відвідування непродуктивних веб-сайтів.

Так, існують дослідження, пов'язані з використанням таймера та його впливом на концентрацію. Одне з таких досліджень — «Turning Time from Enemy into an Ally Using the Pomodoro Technique». Досвід міланської команди Sourcesense показує, що за належного використання техніка Pomodoro може підвищити прозорість як оцінок, так і роботи, пов’язаної з проєктом, яка виконується на різних сайтах. [3].

Дослідження описує метод «помідора», який базується на використанні таймера для поділу часу на періоди роботи та перерв. Учасники дослідження використовували таймери для розподілу часу на інтервали тривалістю 25 хвилин, які відповідали періодам концентрованої праці, а потім надавали собі короткі

перерви тривалістю 5 хвилин. Після чотирьох таких інтервалів була надана довша перерва тривалістю 15-30 хвилин.

Результати дослідження показали, що використання методики Помідора з таймером допомогло учасникам поліпшити їхню концентрацію, зосередженість та продуктивність. Встановлення конкретних періодів роботи і перерв допомагало зменшити розсіювання уваги та втому, що забезпечувало більш ефективне виконання завдань.

Можуть бути використані і нестандартні техніки «Помідора». Наприклад, Даніель Левітен, професор поведінкової нейронауки, послуговується «реальним Помідором». Це використання подій, які відбуваються навколо, як таймерів [6].

Варто зауважити, що музика також може мати вплив на концентрацію та прокрастинацію. Проте, важливо зазначити, що вплив музики на кожну особу може бути індивідуальним, і деякі люди можуть краще концентруватися без музичного супроводу. Також слід враховувати, що тип музики, особливості завдання та особисті уподобання можуть впливати на результати.

Було проведено дослідження щодо впливу музики на концентрацію. Одне з таких досліджень, під назвою "The Mozart Effect: Music Listening and Cognitive Abilities", вивчало вплив класичної музики, зокрема музики Вольфганга Амадея Моцарта, на когнітивні здібності, включаючи концентрацію [4].

Дослідження включало експеримент, в якому учасники прослуховували музику Моцарта перед виконанням когнітивних завдань, таких як розв'язання логічних задач або мнемонічні тести. Результати показали, що певні типи класичної музики, зокрема музика Моцарта, можуть підвищити короткочасну когнітивну функцію, включаючи концентрацію та сприйняття інформації.

Інше дослідження, «The Impact of Background Music on Cognitive Performance», вивчало вплив музики на когнітивні здібності та задоволення від виконання завдань. Було зібрано тридцять шість учасників для трьох когнітивних тестів: вербальне мислення, digit-span і розумове обертання. Кожен тест проходив в одній із таких умов: тиша, вокальна та інструментальна музика. Це дослідження показало вплив фонової музики на виконання різних когнітивних функцій у

студентів університету. Дослідниця дійшла висновку, що фонова музика може бути корисною, але лише за певних обставин. По результатах дослідження стало зрозуміло, що фонова музика перешкоджає короткочасній пам’яті, але виконання візуально-просторових і лінгвістичних завдань було покращено завдяки інструментальній фоновій музиці [5].

Таким чином, для покращення концентрації можна використовувати сервіси з таймерами, фоновими звуками(шум дощу, музика або бінауральні ритми) тощо.

## 1.2 Огляд існуючих сервісів

Сучасний світ вимагає від нас більшої продуктивності та ефективності. Управління часом стає ключовим елементом успішного функціонування як в особистій, так і в професійній сфері. Існує безліч застосунків тайм-менеджменту та покращення концентрації, які надають корисні інструменти та методики для організації робочих процесів, пріоритизації завдань та підвищення продуктивності. Для того, щоб зрозуміти, які функції є найбільш затребуваними, необхідно провести аналіз вже існуючих рішень.

### 1.2.1 Todoist

Todoist – це популярна система тайм-менеджменту, яка дозволяє створювати списки завдань, встановлювати терміни виконання та відстежувати прогрес роботи. Todoist надає можливість організовувати завдання за проєкти, присвоювати їм пріоритети та спільно працювати з колегами. Користувачі можуть синхронізувати свої завдання з різними пристроями та використовувати його як особистий асистент у плануванні щоденних справ. Ця система є однією з найпопулярніших для тайм-менеджменту. За даними з вересня 2021 року, кількість користувачів Todoist перевищує 25 мільйонів.

Todoist розроблений з урахуванням високої навантаженості. Він здатен витримувати значну кількість користувачів та їх завдань. Базуючись на

популярності системи, можна стверджувати, що Todoist ефективно справляється з великим обсягом даних та активною взаємодією користувачів.

Цей застосунок доступний на різних платформах, включаючи веб-додаток, мобільні додатки для iOS та Android, розширення для браузерів та додаток для настільних комп'ютерів (Windows і macOS). Це дає користувачам можливість використовувати Todoist на різних пристроях та в різних ситуаціях, незалежно від місця розташування.

Пропонуються безкоштовний тарифний план з обмеженими можливостями, а також платний план Todoist Premium, який відкриває доступ до розширеного функціоналу. Перехід на платний план дозволяє користувачам отримати більше можливостей, таких як нагадування про пропущені дедлайни, пошук у нотатках, пріоритети для завдань та багато іншого.

Todoist пропонує широкий функціонал для ефективного управління завданнями, серед якого:

– створення завдань з можливістю додавання заголовків, описів, тегів, дедлайнів та пріоритетів;

– організація завдань у проєкти та мітки для кращої структуризації;

– нагадування про наближення дедлайнів через електронну пошту, сповіщення на мобільному пристрої або настільному комп'ютері;

– можливість додавати коментарі, підзадачі та вкладені файли до завдань;

– здатність ділитися проєктами та завданнями з іншими користувачами для спільної роботи та колаборації.

Todoist володіє широким функціоналом та надійністю, що робить його популярним вибором для багатьох користувачів управління часом та завданнями.

### 1.2.2 Jira

Jira – це потужна система управління проєктами і завданнями, розроблена компанією Atlassian спеціально для команд розробників програмного забезпечення. Є однією з провідних систем управління проєктами, широко використовуваною в

IT-компаніях та командах розробників. Великі компанії, такі як Airbnb, Spotify, NASA та багато інших, використовують Jira для керування своїми проєктами.

Система розроблена з урахуванням високих навантажень, її можна успішно використовувати як для невеликих команд, так і для великих організацій. Вона може ефективно обробляти значну кількість проєктів, завдань, коментарів та змін.

Jira доступна як веб-додаток та настільний клієнт, що дозволяє користувачам використовувати її на різних пристроях та операційних системах. Крім того, Jira пропонує додаткові інтеграції з іншими популярними інструментами розробки, такими як GitHub, Bitbucket, Confluence та інші.

Розробники пропонують кілька тарифних планів, включаючи безкоштовний план для невеликих команд. Однак, для повноцінного функціоналу та розширених можливостей доступні платні плани Jira Software, Jira Service Management та Jira Core, які мають різні функції та обмеження.

Jira пропонує широкий спектр функціональних можливостей, зокрема:

– створення та управління проєктами, завданнями, епіками, справами тощо;

– організація завдань у спринти, стеження за їх прогресом та розподіл роботи в команді;

– керування пріоритетами (користувачі можуть призначати пріоритети від «Низький» до «Високий» або використовувати інші власні маркери пріоритетів для кращого орієнтування.), термінами виконання та нагадуваннями про дедлайни;

– візуалізація проєктів за допомогою діаграм ганта, дошок завдань та звітів;

– широкі можливості для створення звітів та аналізу даних проекту (швидкості роботи, витрати часу, прогрес завдань тощо), що допомагає зрозуміти ефективність процесу та зробити відповідні корекції;

– дозволяє керувати ресурсами команди, можна відстежувати доступність та завантаженість кожного учасника команди, призначати їм завдання та контролювати розподіл роботи.

– створення та візуалізація завдань на гнучких дошках Kanban або Scrum, де можна переміщувати та відстежувати задачі;

– спільна робота над завданнями, обговорення, коментування та спільне редагування.

Jira забезпечує ефективне управління проєктами, завданнями та комунікацією в командах розробників. Вона надає широкі можливості для організації робочого процесу, збільшення продуктивності та забезпечення успішної доставки проєктів.

### 1.2.3 Trello

Trello – це веб-платформа для управління проєктами та завданнями, яка пропонує простий і інтуїтивно зрозумілий спосіб організації роботи та співпраці в командах. Є популярним інструментом, який використовується не тільки в IT-сфері, але і в різних інших галузях. Великі компанії, малі бізнеси, команди розробників та навіть особисті користувачі широко використовують Trello для організації своїх завдань і проєктів.

Trello є дуже масштабованим інструментом, який здатний витримувати як невеликі проєкти, так і великі команди. Завдяки його хмарній архітектурі та розподіленому збереженню даних, забезпечується швидкий доступ та ефективна робота навіть при великій кількості користувачів та завдань.

Він доступний як веб-додаток, а також має мобільні додатки для різних платформ, таких як iOS та Android. Це дає можливість користувачам використовувати Trello на будь-яких пристроях та в будь-якому місці. Крім того, Trello можна інтегрувати з різними іншими інструментами, такими як Slack, Jira, Google Drive та інші.

Застосунок пропонує безкоштовний план з базовим функціоналом, а також платні плани з розширеними можливостями. Платні плани Trello Business Class та Trello Enterprise надають додаткові функції для більших команд і організацій, такі як розширені звіти, обмеження доступу та інші.

Основою Trello є система дошок, списків і карток. Користувачі можуть створювати нові дошки для проєктів, розміщувати на них списки завдань і створювати картки для кожного окремого завдання. Картки можуть містити опис,

дедлайни, коментарі, вкладення та інші деталі. Крім того, Trello пропонує функції, такі як мітки, чек-листи, вкладеність списків, повідомлення, активності та інші. Користувачі можуть пересувати картки між списками, змінювати їх статус, призначати відповідальних за завдання, додавати мітки та кольорові етикетки для класифікації, а також встановлювати терміни виконання. Trello також підтримує спільну роботу, де кілька користувачів можуть спільно працювати над проєктом, додавати коментарі, відстежувати зміни та оновлення.

### 1.2.4 Noisli

Noisli – це популярний застосунок, який пропонує широкий функціонал для створення фонових звуків, які сприяють концентрації, розслабленню та підвищенню продуктивності. Завдяки великому вибору природних звуків, таких як шум дощу, лісова атмосфера, вуличний шум та багато інших, Noisli допомагає створювати комфортну та сприятливу обстановку для роботи, вивчення або релаксації.

Основний функціонал Noisli включає:

– можливість комбінувати різні звуки та створювати індивідуальні мікси, обрати звуки, які найбільше відповідають потребам та вподобанням;

– вбудований таймер, який дозволяє встановити певний час роботи або відпочинку, що допомагає створити ритмічний графік роботи та відпочинку, дотримуючись принципів методу Помодоро або інших технік тайм-менеджменту;

– можливість писати нотатки прямо в додатку;

– можливість встановити фон, який найбільше підходить під поточний настрій користувача (статичний або змінюваний з часом);

– збереження міксів зі звуками для майбутнього використання, що дозволяє швидко отримати потрібну атмосферу без необхідності налаштовувати звуки знову.

Noisli є фріваром з можливістю придбати платну версію. Платні опції дозволяють отримати доступ до додаткових функцій, таких як більша кількість

звуків для комбінування, можливість синхронізувати налаштування між різними пристроями та інші переваги. Користувачі можуть вибрати план підписки, який найкраще відповідає їхнім потребам.

### 1.2.5 Focus@Will

Focus@Will – це онлайн-сервіс, розроблений спеціально для підвищення концентрації та покращення продуктивності. Використовуючи принципи нейробіології та музикотерапії, Focus@Will надає користувачам підбірку музики, яка допомагає зосередитися та залишатися у стані потоку протягом тривалого часу.

Сервіс пропонує великий вибір музичних жанрів, звукових ландшафтів та інструментальних композицій. Користувачі можуть вибрати той стиль, який найкраще сприймається та стимулює їхню концентрацію.

Focus@Will дозволяє налаштовувати музичний потік відповідно до особистих потреб користувача. За допомогою алгоритмів, які враховують вікові категорії, рівень стресу та інші параметри, сервіс пропонує оптимальну музику для покращення концентрації.

Застосунок використовує спеціальну технологію «ADHD type», яка забезпечує музичні ритми та структури, що допомагають підвищити увагу та зосередженість у людей з розладами уваги.

Функція «Productivity Tracker» надає користувачам можливість відстежувати час, проведений у стані концентрації, та оцінювати свою продуктивність. Це дозволяє користувачам аналізувати свої звички та вдосконалювати свою робочу ефективність.

Focus@Will є платним сервісом, який пропонує різні підписки з різними функціональними можливостями. Він доступний як веб-додаток та мобільний додаток для iOS та Android пристроїв.

Цей сервіс користується популярністю серед студентів, професіоналів та тих, хто працює зі збільшеним обсягом інформації та потребує високого рівня концентрації та продуктивності. Він допомагає створити краще аудіовізуальне

середовище для роботи, навчання та творчості, що сприяє підвищенню уваги, зосередженості та результативності.

### 1.2.6 Be Focused

Be Focused є одним з популярних застосунків для підвищення концентрації та ефективного управління часом. Він пропонує ряд функцій та інструментів, які допомагають користувачам зосередитися на своїх завданнях та виконувати їх продуктивно.

Основний функціонал Be Focused включає можливість створення та керування списками завдань, налаштування таймера концентрації та перерв, а також відстеження часу, витраченого на кожне завдання. Застосунок дозволяє налаштувати інтервали роботи та відпочинку відповідно до методики «Помідора», де робочі блоки тривають певну кількість часу, а потім слідує коротка перерва.

Be Focused також надає звіти про продуктивність, які допомагають користувачам відстежувати свої успіхи та покращувати ефективність роботи. Застосунок працює на різних платформах, включаючи комп'ютери, смартфони та планшети, що дозволяє користувачам мати доступ до своїх завдань та таймерів у будь-який час із будь-якого пристрою.

Be Focused має як безкоштовну, так і платну версії. Платна версія надає додаткові функції, такі як синхронізація між пристроями, розширені налаштування таймера та збереження звітів. Платні опції дозволяють користувачам отримати більше функціональності та зручності в роботі з застосунком. Застосунок доступний на macOS та iOS в App Store.

## 1.3 Обґрунтування актуальності розробки

Багато існуючих рішень пропонують основні функції, такі як створення завдань, встановлення термінів та пріоритетів, сповіщення та спільну роботу у команді. Однак, деякі користувачі можуть шукати додаткові можливості для

зосередження та підвищення продуктивності, такі як засоби концентрації, аналітика використання часу та інші спеціалізовані функції. Порівняльну таблицю між аналоговими рішеннями та розроблюваним додатком наведено у табл. 1.

Таблиця 1.1 — Порівняльний аналіз застосунків

| Додаток | Anchor /Assist | Todoist | Jira | Trello | Noisli | Focus@Will | Be Focused |
|---|---|---|---|---|---|---|---|
| Безкоштовні плани | + | + | + | + | + | - | + |
| Створення списків | + | + | + | + | - | - | + |
| Дедлайни | + | + | + | + | - | - | + |
| Пріоритизація | + | + | + | + | - | - | - |
| Групування | + | + | + | + | - | - | - |
| Кросплат-форменість | - | + | + | + | + | + | - |
| Спільна робота | - | + | + | + | - | - | - |
| Таймер | + | - | - | - | + | + | + |
| Фонові звуки | + | - | - | - | + | + | + |

Сильні сторони існуючих рішень включають:

– Noisly пропонує простий та інтуїтивний інтерфейс, що дозволяє користувачам швидко створювати звуки та керувати ними;

– можливість спільної роботи у команді та обмін інформацією про завдання;

– можливість спільної роботи;

– наявність мобільних додатків для доступу до системи з різних пристроїв.

Натомість, у існуючих рішень є і обмеження:

– спеціалізовані засоби для покращення концентрації та зосередженості під час виконання завдань знаходяться в інших застосунках;

– перенавантажений інтерфейс деяких застосунків викликає розгубленість та спричиняє надлишковий інформаційний шум;

– деякі рішення можуть бути складними у використанні або мають надлишковий функціонал, який не потрібен деяким користувачам;

– вартість підписки або ліцензування деяких рішень можуть бути високими для окремих користувачів або невеликих команд;

– деякі з додатків доступні лише на певній платформі.

Враховуючи популярність аналогів та поставлені вимоги, пропонується розробити простий у використанні та функціональний вебзастосунок, що буде більш орієнтованим на сінгл-користувачів. Додатковий фокус буде зосереджено на засобах концентрації, які допоможуть користувачам зосередитися та підвищити продуктивність.

Висновок до розділу

Проведений аналіз існуючих рішень у галузі тайм-менеджменту підкреслює важливість ефективного управління часом в сучасному світі. Історія тайм-менеджменту відображає його корені та еволюцію, що дало поштовх для розвитку різних методів та підходів.

Під час дослідження було виявлено, що тайм-менеджмент був реакцією на зростаючі вимоги та виклики сучасного світу. Він став невід'ємною частиною професійного та особистого життя, допомагаючи краще організовувати час, пріоритезувати завдання та досягати більшої продуктивності.

Історія тайм-менеджменту показала, що протягом років розроблялися різні методи та підходи до ефективного управління часом. Від класичних технік, таких як методи Парето та ABC-аналіз, до сучасних систем тайм-менеджменту, таких як GTD, «Помідор» та інші, люди завжди ставили собі за мету полегшити і покращити своє використання часу.

Цей підрозділ показав, що історія тайм-менеджменту постійно розвивається, а нові технології та інструменти додають нові можливості. Розуміння минулих досягнень і практик допомагає будувати майбутнє тайм-менеджменту та розробляти нові інноваційні підходи.

Історія тайм-менеджменту є цінним джерелом знань, що надає нам контекст і розуміння важливості ефективного розподілу часу, цінності навичок тайм-менеджменту, які необхідні в сучасному світі, де швидкість, конкуренція та великий обсяг інформації стають нормою. Знання про історію тайм-менеджменту надає нам базу для подальшого розвитку та впровадження нових інструментів і підходів для досягнення максимальної продуктивності та ефективності.

Дослідження, проведені в цій галузі, підтверджують, що правильний тайм-менеджмент сприяє покращенню продуктивності, зниженню стресу і підвищенню благополуччя. Застосування наукових принципів та методів дозволяє зосередитися на головних завданнях, підвищує концентрацію та забезпечує більш ефективне використання ресурсів.

Незважаючи на наявність аналогових рішень, які добре зарекомендували себе протягом багатьох років, актуальність створення нового додатку в галузі тайм-менеджменту є обґрунтованою. Швидкий темп життя, цифрові технології та зростання потреб користувачів вимагають інноваційних інструментів, які б допомагали краще організовувати час, зосереджуватися на завданнях і досягати поставлених цілей.

Необхідно підкреслити, що наявність багатьох рішень-аналогів у галузі тайм-менеджменту свідчить про важливість цієї теми та постійне прагнення людей до організації свого часу.

Отже, враховуючи історію, дослідження та потреби сучасного суспільства, створення нового додатку з тайм-менеджменту є актуальним і важливим кроком, спрямованим на поліпшення продуктивності, зниження стресу і забезпечення кращого управління часом у цифровій епосі.

# 2 ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

При виборі певних засобів технологій, бібліотек, середовищ розробки та інших інструментів необхідно ретельно обґрунтувати свої вибори, враховуючи специфіку проєкту, його потреби, обмеження та вимоги.

Метою даного розділу є забезпечення обґрунтованості та об'єктивності вибору засобів реалізації, що сприятиме успішному впровадженню та розробці програмного продукту. Детальний аналіз і обґрунтування вибору засобів допоможе зрозуміти, які переваги та можливості надають обрані технології і інструменти, а також як сприятимуть досягненню поставлених цілей і задоволенню потреб користувачів. Це дозволить упевнено просуватися вперед у процесі розробки, максимально використовуючи потенціал обраних засобів та забезпечуючи високу якість і ефективність програмного продукту.

Метою розробки є створення інноваційного і високоефективного рішення в галузі тайм-менеджменту, яке допоможе людям краще управляти своїм часом, покращити продуктивність та досягати поставлених цілей.

Основне призначення розробки полягає в наданні користувачам інструментів та можливостей для ефективного планування, організації та керування своїм часом. Цільовий продукт має бути зручним, інтуїтивно зрозумілим та потужним програмним забезпеченням, яке сприятиме оптимальному використанню часу, зменшенню стресу та підвищенню продуктивності.

Розробка має на меті забезпечити користувачам зручні та ефективні інструменти для планування задач, керування пріоритетами, відстеження прогресу та управління проєктами. Розробка має великий потенціал для покращення якості життя людей, надаючи їм інструменти для ефективного управління часом, досягнення цілей та досягнення балансу в житті.

Після ретельного аналізу конкурентних рішень, було виділено задачі для вирішення, складено перелік функціональних та нефункціональних вимог, визначено користувацький та адміністраторський модулі.

## 2.1 Загальні вимоги до додатку

### 2.1.1 Функціональні вимоги до системи

Користувацький та адміністраторський модулі мають бути реалізовані з веб-інтерфейсом та повинні відповідати наступним функціональним вимогам:

- реєстрація та авторизація користувача;
- можливість створити складну задачу;
- можливість переглянути усі задачі;
- можливість видалити задачу;
- можливість редагувати задачу;
- можливість встановити пріоритет задачі;
- можливість встановити статус задачі;
- можливість встановити дедлайн задачі;
- можливість створювати теги;
- можливість переглянути задачі за тегом;
- можливість створити план на день;
- можливість встановити таймер;
- можливість увімкнути фонові звуки для концентрації;
- можливість редагування профілю користувача;
- можливість додавати нових адміністраторів;
- можливість переглядати користувачів;
- можливість банити користувачів.

### 2.1.2 Нефункціональні вимоги до системи

Нефункціональні вимоги визначають якість, надійність та ефективність системи, а також її здатність відповідати вимогам користувачів та контексту використання. Відповідно до цього, нефункціональні вимоги описують характеристики, які не пов'язані безпосередньо з функціональністю системи, але мають велике значення для успіху проєкту.

Користувацький інтерфейс є невід'ємною складовою будь-якої системи, незалежно від її типу чи призначення. Він є мостом комунікації між користувачем та системою, дозволяючи користувачеві взаємодіяти з системою та керувати її поведінкою.

Мінімалістичний інтерфейс має свої переваги і відповідає потребам сучасного користувача. Основна причина, чому інтерфейс повинен бути мінімалістичним — він забезпечує простоту та зосередженість на основних елементах та функціоналі. Серед аргументів, що підкреслюють важливість мінімалістичного дизайну інтерфейсу, можна навести:

– забезпечення легкості сприйняття, так як має мінімальну кількість зайвих деталей і зосереджений на суттєвих елементах, що робить його зрозумілим для користувача;

– завантажений інтерфейс з великою кількістю кнопок, меню та інших елементів може заплутати користувача і викликати перевантаження інформацією, натомість мінімалістичний підхід допомагає зменшити це перевантаження, зосереджуючи увагу на основних функціях та зручному використанні;

– естетичний вигляд та елегантність, що створює враження сучасності, розуміння технологій та професіоналізму;

– краще виокремлення контенту, який є основним елементом взаємодії з користувачем.

Отже, мінімалістичний інтерфейс є ефективним рішенням для поліпшення користувацького досвіду, спрощення взаємодії та забезпечення фокусу на основних завданнях та контенті.

Важливою вимогою для будь-якого застосунку є забезпечення безпеки і конфіденційності даних, особливо в сучасному цифровому середовищі, де велика кількість особистої та конфіденційної інформації зберігається та обробляється. Існує немало причин, чому застосунок повинен дбати про безпеку та конфіденційність даних. Користувачі довіряють застосунку свої персональні дані, такі як імена, адреси, контактна інформація та інші особисті дані. Застосунок повинен забезпечувати захист цих даних від несанкціонованого доступу, втрати

або зловживання. Багато застосунків використовуються в комерційних цілях і містять конфіденційну інформацію про бізнес-процеси, клієнтів, фінансову інформацію тощо. Забезпечення конфіденційності цих даних є критичним для запобігання фінансовим втратам, порушенням довіри та іншим негативним наслідкам.

Окрім того, у багатьох країнах існують закони і нормативні акти, що регулюють зберігання та обробку персональних даних. Застосунок повинен відповідати цим вимогам, забезпечуючи правовий захист даних своїх користувачів.

Безпека та конфіденційність даних є важливими факторами для створення довіри користувачів до застосунку. Користувачі більш схильні використовувати та залишатися в застосунку, якщо вони впевнені, що їхні дані захищені.

Таким чином, розроблена система має відповідати наступним нефункціональним вимогам:

– мати зручний користувацький веб-інтерфейс;

– мати мінімалістичний інтерфейс, який буде легким для сприйняття, неперегруженим та не буде відволікати;

– забезпечити безпеку і конфіденційність даних.

## 2.2 Обґрунтування вибору засобів реалізації

Вибір інструментів для реалізації має вирішальний вплив на функціональність, ефективність, безпеку та масштабованість системи. Врахування потреб проєкту, його специфіки, масштабу та інших факторів допомагає зробити обґрунтований вибір засобів, які забезпечать досягнення бажаних результатів.

### 2.2.1 Мова програмування: Java 17

Мову Java було обрано для реалізації цього додатку. Вона була створена під основні принципи ООП, що робить її природнім вибором для реалізації об'єктно-орієнтованих концепцій і практик.

Об'єктно-орієнтоване програмування (ООП) має кілька переваг, що роблять його використання корисним у багатьох випадках:

– дозволяє розділити програму на окремі модулі (класи), які мають свою внутрішню логіку і можуть бути легко змінені або модифіковані без впливу на інші частини програми;

– підтримка концепції класів і наслідування, що дозволяє створювати класи, які можуть успадковувати властивості і функціональність від інших класів, ще сприяє повторному використанню коду;

– завдяки наслідуванню і поліморфізму, ООП дозволяє легко розширювати функціональність програми;

– об'єкти в ООП можуть представляти реальні об'єкти або концепції, а їх взаємодія може відображати реальні взаємодії, таким чином це полегшує розуміння програми і сприяє більш ефективному розв'язанню завдань.

Протягом останніх кількох років більшість розробників Java обмежувалися набором функцій Java 8 або Java 11, проте тепер доступна нова версія LTS Java. З Java 17 прийшли нові функції та покращення, що полегшують розробку. Наприклад, введення Sealed Classes, Pattern Matching, var keyword та інші дозволяють писати більш зрозумілий та компактний код. Один з прикладів використання нової функції Java 17 можна побачити на рисунку нижче.

```
public static void showResult( @NotNull UtilNotification resultNotification) {
    switch (resultNotification.getStatus()) {
        case ERROR -> show(resultNotification.getMessage()).addThemeVariants(LUMO_ERROR);
        case SUCCESS -> show(resultNotification.getMessage()).addThemeVariants(LUMO_SUCCESS);
        case UNDEFINED -> show(resultNotification.getMessage()).addThemeVariants(LUMO_CONTRAST);
    }
}
```

Рисунок 2.1 – Нова функція Extended switch expression в додатку

Кожне оновлення Java приносить покращення продуктивності, оптимізації та вдосконалення. Java 17 не є винятком і може пропонувати поліпшену швидкодію та ефективність у порівнянні зі старішими версіями.

Java має вбудовану систему управління пам'яттю та вбудовані механізми обробки помилок, що робить її стабільною та надійною платформою для розробки вебзастосунків.

До того ж, ця мова є однією з найпопулярніших мов програмування, і має велику спільноту розробників. Це означає, що є багато ресурсів, документації, бібліотек та підтримки, що значно полегшує розробку та розвиток проєкту.

Java базується на концепції «write once, run anywhere» (пиши один раз, запускай скрізь). Це означає, що програми, написані на Java, можуть запускатися на різних операційних системах без необхідності переписувати код, що робить їх більш гнучкими та масштабованими.

Ця мова має широкий вибір бібліотек, фреймворків та інструментів розробки, що сприяє підвищенню продуктивності та прискоренню розробки. Наприклад, для розробки вебзастосунків на Java є відомі фреймворки, такі як Spring, GWT, Struts та Vaadin, які дозволяють швидко створювати потужні та масштабовані додатки.

### 2.2.2 Фреймворк веб-інтерфейсу: Vaadin

Vaadin — це Java фреймворк для створення користувацьких інтерфейсів із темами та компонентами та багатьма можливостями для розширення.

Vaadin складається з клієнтської та серверної сторін, причому клієнтська сторона побудована на основі добре відомого фреймворку Google Widget Toolkit, а серверна сторона обробляється VaadinServlet. Охоплення фреймворком серверної частини означає, що кожна зміна, яка вноситься в інтерфейс користувача, негайно надсилається на сервер, тож у будь-який момент бекенд знає, що відбувається на фронтенд частині [15].

Vaadin надає високорівневий інтерфейс програмування, що дозволяє розробникам швидко створювати веб-інтерфейси без необхідності вручну кодувати HTML, CSS або JavaScript. Це значно прискорює процес розробки та дозволяє зосередитися на функціональності додатку.

Фреймворк пропонує багато готових компонентів із багатогранною функціональністю, таких як кнопки, таблиці, форми, навігаційні меню тощо. Це дозволяє будувати складні інтерфейси, просто компонуючи та налаштовуючи ці компоненти за допомогою Java-коду.

Веб-додатки, розроблені з використанням Vaadin, можуть запускатися на будь-якому пристрої та платформі. Vaadin забезпечує автоматичну адаптацію інтерфейсу до різних розмірів екранів та пристроїв, що дозволяє забезпечити зручний користувацький досвід незалежно від пристрою, на якому запущений додаток.

Vaadin надає вбудовані механізми для захисту додатків, такі як перевірка валідності даних, обробка запитів та захист від атак, таких як XSS (Cross-Site Scripting) та CSRF (Cross-Site Request Forgery). Це допомагає забезпечити безпеку вебзастосунку та захистити користувачів від потенційних загроз.

Фреймворк дозволяє розширювати функціональність за допомогою власних компонентів, тем та розширень. Можна створювати свої компоненти або використовувати ті, які розроблені спільнотою, що дозволяє розширити можливості додатку та пристосувати його до конкретних вимог проєкту.

### 2.2.3 Фреймворк Spring/Spring Boot

Spring і Spring Boot надають потужні інструменти, які спрощують процес розробки. З їх допомогою можна швидко створювати високоякісні застосунки, скорочуючи час та зусилля, які потрібно вкласти у конфігурацію та управління різними аспектами продукту.

Spring заснований на принципі інверсії управління (IoC) та інжекту залежностей (DI), що сприяє зменшенню зціплення залежностей та полегшує тестування та підтримку коду. Це робить програми більш модульними та зрозумілими. Spring займається інфраструктурою, а розробник — логікою застосунку [8].

Spring надає широкі можливості для розширення та налаштування додатків. За допомогою аспектно-орієнтованого програмування (AOP) можна легко впроваджувати додаткову функціональність, таку як логування, без зміни основного коду додатку [7].

Цей фреймворк надає контейнер управління бізнес-логікою, який дозволяє ефективно керувати життєвим циклом компонентів додатку. Це дає можливість зосередитися на розробці бізнес-логіки, не зважаючи на деталі управління.

Також має широку підтримку для інтеграції з різними технологіями та фреймворками, такими як бази даних, веб-сервери, хмарні сервіси, месенджери тощо. Це дозволяє легко і безпроблемно інтегрувати додаток з існуючими системами.

Також є велика та активна спільнота розробників, що означає, що завжди можна знайти підтримку, документацію, плагіни та інші ресурси для розвитку проєкту.

Серед інших переваг цього фреймворку:

– попередньо визначені шаблони (для технологій Hibernate, JDBC і JPA);

– слабке зчеплення;

– легке і просте тестування (завдяки інжекту залежностей);

– хороша підтримка абстракціонування (для специфікацій на основі Java EE, таких як JMS, JDBC, JPA та JTA.);

– постійне управління транзакціями;

– гарна організація веб-платформ.

Однією з ключових функцій Spring Boot є Spring Boot Autoconfiguration, яка спрощує процес конфігурації та розгортання додатків на основі Spring Framework. Ця функція дозволяє автоматично налаштовувати багато компонентів інфраструктури додатка, що забезпечує зручну та ефективну розробку.

Основна ідея Spring Boot Autoconfiguration полягає в тому, щоб додаток автоматично налаштовував своє середовище на основі наявних залежностей та конфігурацій. Spring Boot аналізує класи залежностей, які були додані до проєкту, та налаштовує відповідні компоненти для цих залежностей. Наприклад, якщо ми

додаємо залежність на базу даних, Spring Boot автоматично створює конфігурацію для підключення до бази даних, щоб можна було працювати з нею без необхідності вручну налаштовувати всі параметри.

Spring Boot Autoconfiguration використовуєся для автоматичного налаштування багатьох інших компонентів, таких як веб-сервери, безпека, логування, кешування, моніторинг, розподілена обробка повідомлень та багато інших. Завдяки цій функції розробникам не потрібно глушити конфігураційні файли або вручну налаштовувати кожен компонент окремо. Spring Boot самостійно розуміє, які компоненти необхідно налаштувати та які значення використовувати, щоб додаток працював оптимально.

### 2.2.4 Spring Security

Spring Security надає можливість здійснювати процеси автентифікації та авторизації в програмах за допомогою простих фільтрів сервлетів. У веб-програмах існує ризик загроз та атак, оскільки вони доступні для будь-якого користувача Інтернету. Деякі REST-точки можуть бути обмеженими для певних користувачів, наприклад, для доступу до оновлення записів у базі даних або для виконання адміністративних операцій. Ця бібліотека забезпечує захист URL-адрес та безпеку корпоративних додатків, побудованих на базі J2EE. Вона надає широкі можливості налаштування безпеки, такі як автентифікація та авторизація. Spring Security є стандартом де-факто для захисту додатків, розроблених з використанням фреймворка Spring. [14].

Фреймворк працює на таких основних концепціях:

- автентифікація;
- авторизація;
- зберігання паролів;
- фільтри сервлетів.

Автентифікація використовується для перевірки того, чи користувач використовує програму, надаючи дійсні облікові дані, які використовуються для

перевірки того, хто він є. Автентифікація – це встановлення особи (користувача, системи, яка може виконувати дію в додатку).

За допомогою простої автентифікації не можна обмежити користувачів, які ввійшли в систему, оскільки це не надає інформації про привілеї та дозволи користувача. Авторизація допомагає надати цю інформацію до того, як користувач спробує отримати доступ до ресурсу. Це процес контролю доступу, який вирішує, чи може користувач виконувати дію (контроль доступу → адміністратор, користувач, керівник, менеджер, анонім тощо) чи ні.

Інтерфейс PasswordEncoder Spring Security виконує одностороннє перетворення пароля (ми не можемо розшифрувати пароль). Spring Security надає кілька PasswordEncoder, серед яких:

- BCryptPasswordEncoder;
- Argon2PasswordEncoder;
- Pbkdf2PasswordEncoder;
- SCryptPasswordEncoder.

У розроблюваному додатку було використано BCryptPasswordEncoder. Щоб захистити від атак грубої сили, Bcrypt додає кожного разу до паролю обчислювальні витрати під час запуску. Це сповільнює процес і не дозволяє зловмисникам випробовувати сотні тисяч паролів за секунду. Наступною частиною є сіль. Сіль додає випадковий фрагмент тексту до пароля, який потрібно закодувати, що захищає від атак веселкової таблиці. Це гарантує, що навіть два користувачі з однаковим паролем матимуть різні хеші, що зберігаються в базі даних [9].

Spring Security використовує фільтри сервлетів Java, щоб почати перевірку безпеки веб-програми.

### 2.2.5 Spring Data JPA

Spring Data JPA – це фреймворк, який надає спрощений спосіб взаємодії з базами даних у додатках, що побудовані на основі Spring Framework. Він дозволяє

зменшити кількість коду, який потрібно написати для роботи з базою даних. Він надає стандартні реалізації багатьох рутинних операцій, таких як створення, оновлення, видалення та пошук записів. Це дає можливість розробникам сконцентруватись на бізнес-логіці застосунку, замість написання повторюваного коду для роботи з базою даних.

Фреймворк надає інтерфейс JpaRepository, який включає методи для стандартних операцій створення, читання, оновлення та видалення (CRUD) записів у базі даних. Розробникам не потрібно визначати ці методи вручну, оскільки вони автоматично генеруються на підставі визначення сутностей та їх взаємозв'язків.

Важливим є те, що можна визначати складні запити за допомогою аннотацій або методів з назвами, відповідним конвенціям. Це дозволяє виконувати різноманітні операції з базою даних, такі як пошук з використанням умов, з'єднання декількох таблиць та агрегація даних.

Spring Data JPA використовує механізм транзакцій, що дозволяє забезпечити цілісність та консистентність даних. Він автоматично керує транзакціями при виконанні операцій з базою даних, забезпечуючи атомарність та ізоляцію [13].

На додачу, фреймворк підтримує різні бази даних, такі як MySQL, PostgreSQL, Oracle, і багато інших, завдяки чому розробники можуть працювати з різними базами даних, не змінюючи код застосунку.

### 2.2.6 MySQL

MySQL відома своєю простотою і легкістю використання. Її синтаксис запитів є логічним та зрозумілим, що полегшує розробку та роботу з базою даних навіть для початківців. Вона використовує реляційну модель для зберігання даних у вигляді таблиць зі зв'язками між ними. Це дозволяє створити структуровану і логічну організацію даних.

Ця база даних є швидкою та ефективною, добре справляється з обробкою великого обсягу даних. Вона має оптимізовані алгоритми, що забезпечують швидкий доступ до даних та виконання запитів.

MySQL підтримує багато функцій і можливостей, включаючи транзакції, індекси, збережені процедури, тригери, реплікацію та багато іншого. Це дає можливість реалізувати складні сценарії та оптимізувати роботу з базою даних.

Вона також є однією з найбільш популярних баз даних у світі, тому має велику сумісність з різноманітними інструментами, фреймворками та мовами програмування. Це дозволяє легко інтегрувати її з різними технологіями та розширювати функціональність свого проєкту.

### 2.2.7 Amazon RDS

Amazon RDS (Relational Database Service) є хмарною послугою від Amazon Web Services (AWS), яка надає можливість легко розгортати та керувати реляційними базами даних. Amazon RDS дозволяє спростити процес налаштування, резервного копіювання, масштабування та керування реляційними базами даних, такими як MySQL, PostgreSQL, Oracle, SQL Server та іншими, у хмарному середовищі AWS [12].

Amazon RDS забезпечує керований доступ до баз даних, що означає, що AWS бере на себе завдання забезпечення доступності, резервного копіювання, автоматичного патчінгу та моніторингу бази даних. Це дозволяє розробникам та адміністраторам баз даних зосередитися на своїх додатках та бізнес-логіці, не витрачаючи багато часу на керування інфраструктурою бази даних.

Цей сервіс має наступні особливості:

– легко масштабована інфраструктура, що забезпечує зростаючі потреби проєкту, завдяки чому можна збільшувати розмір бази даних, кількість реплік та розподілити навантаження, щоб забезпечити високу доступність та продуктивність;

– Amazon RDS відповідає за управління базою даних, включаючи резервне копіювання, моніторинг, автоматичні оновлення та патчі, що звільняє від необхідності адміністрування бази даних, дозволяючи сконцентруватися на розробці додатку;

– надає можливість реплікації бази даних для забезпечення високої доступності, налаштування резервних копій та репліки бази даних в різних регіонах AWS для забезпечення надійності та безперебійної роботи системи;

– різні механізми безпеки для захисту даних, що включає в себе можливість шифрування даних в дисках та під час передачі, налаштування доступу до бази даних за допомогою різних рівнів автентифікації та управління правами доступу;

– Amazon RDS легко інтегрується з іншими сервісами AWS, такими як Amazon EC2, Amazon Lambda, Amazon VPC та багатьма іншими, що дозволяє створювати комплексні рішення та використовувати інші сервіси AWS для розширення функціональності проєкту.

Amazon RDS може мати декілька економічних переваг для підприємств і організацій, серед яких плата лише за використані ресурси, зменшення витрат на обслуговування і управління базами даних, можливість швидко розгортати нові бази даних без необхідності вручну налаштовувати апаратне та програмне забезпечення.

### 2.2.8 Lombok

Згідно документації, «Project Lombok — це бібліотека Java, яка автоматично підключається до вашого редактора та білд-інструментів, покращуючи вашу Java» [10].

Ця бібліотека надає набір зручних анотацій, які генерують код під час компіляції, допомагаючи розробникам економити час і покращуючи читабельність коду.

За допомогою Lombok можна замінити шаблонний код значущими анотаціями. Вони допомагають розробнику зосередитися на бізнес-логіці. Lombok також надає деякі анотації, які поєднують кілька інших анотацій (наприклад, @Data поєднує @ToString, @EqualsAndHashCode, @Getter/@Setter і @RequiredArgsConstructor), тому не потрібно «забруднювати» код занадто

великою кількістю анотацій. Оскільки код більш стислий, зміна та додавання нових полів не вимагає багато зусиль.

## 2.3 Вибір середовища розробки

### 2.3.1 IntelliJ IDEA

IntelliJ IDEA є одним з найпопулярніших інтегрованих середовищ розробки (IDE) для роботи з різноманітними мовами програмування, включаючи Java, Kotlin, Python, JavaScript та багато інших.

IntelliJ IDEA, як і будь-який програмний засіб має як переваги, так і недоліки, перелік яких наведено в таблиці 2.1:

Таблиця 2.1

| Переваги | Недоліки |
|---|---|
| Потужність та функціональність | Вартість |
| Підтримка різних мов програмування | Вимоги до ресурсів |
| Інтеграція з іншими інструментами | Навчання |
| Розширюваність | |

IntelliJ IDEA надає широкий набір інструментів та функціональностей, які сприяють покращенню продуктивності розробника. Вона має потужні інструменти рефакторингу, автоматичне завершення коду, аналіз коду, вбудовані інструменти для роботи з Git, дебагер та інші корисні функції, які спрощують розробку.

Окрім того, це середовище підтримує широкий спектр мов програмування та фреймворків, що дозволяє розробникам працювати з різноманітними проєктами без необхідності переключатися між різними IDE.

Вона забезпечує інтеграцію з популярними інструментами розробки, такими як системи управління версіями Git, системи збирання проєктів Maven і Gradle, сервери додатків та багато інших, що спрощує роботу з різними інструментами у контексті однієї IDE.

На додачу має потужну систему плагінів, що дозволяє розширювати функціональність IDE за допомогою сторонніх розширень, що дозволяє адаптувати робоче середовище під потреби конкретного проєкту або мови програмування.

Варто зауважити, що IntelliJ IDEA має платну ліцензію, яка може бути високою для окремих розробників або невеликих команд. Проте є безкоштовна версія Community Edition, яка надає базовий набір функцій.

Також це середовище розробки вимагає певних обчислювальних ресурсів для ефективної роботи, особливо при великих проєктах. Це може створити проблеми на менш потужних комп'ютерах або в умовах обмеженого обладнання.

Для новачків у використанні IntelliJ IDEA або розробки загалом може знадобитися певний час для ознайомлення з інтерфейсом та функціоналом IDE. Проте, надалі це може стати перевагою, оскільки користувач отримає доступ до потужних інструментів та можливостей.

Хоча IntelliJ IDEA має певні недоліки, загалом це потужний та функціональний інструмент розробки, який широко використовується професіональними розробниками для розробки різноманітних проєктів.

### 2.3.2 MySQL Workbench

MySQL Workbench є інтегрованим середовищем розробки (IDE) для роботи з базами даних MySQL. Воно надає зручний інтерфейс для створення, адміністрування та управління базами даних MySQL. MySQL Workbench включає набір інструментів, таких як графічний редактор запитів SQL, конструктор бази даних, інструменти для моделювання даних, візуальний аналізатор виконання запитів, інструменти імпорту/експорту даних, а також можливості адміністрування та керування серверами баз даних.

Це середовище розробки дозволяє розробникам зручно створювати, змінювати та виконувати SQL-запити, проєктувати схему бази даних за допомогою графічного інтерфейсу, візуально відстежувати та аналізувати виконання запитів, а також ефективно керувати базами даних MySQL.

Підсумовуючи, MySQL Workbench є потужним середовищем розробки, що дозволяє зосередитися на роботі з базою даних, забезпечує зручний інтерфейс і широкий набір функціональних можливостей для розробки і управління базами даних.

Висновок до розділу

Перш за все, правильний вибір засобів технологій, середовищ розробки та інших інструментів є ключовим етапом у розробці програмного продукту. У процесі обґрунтування вибору засобів, було детально проаналізувано різні альтернативи та порівняно їх переваги та недоліки. Враховуючи специфіку проєкту, було обрано засоби, які найбільш відповідають його потребам та вимогам.

Дослідження та обговорення вибраних засобів реалізації показали, що вони мають важливі переваги, які сприятимуть успішному впровадженню та розвитку програмного продукту. Наприклад, використання певних технологій та бібліотек дозволить забезпечити швидку розробку та високу якість коду. Використання конкретних середовищ розробки спростить роботу розробника та забезпечить зручне управління проєктом.

Java є потужною та популярною мовою програмування, що надає широкий спектр функціональності та забезпечує переносимість програмного забезпечення на різні платформи. Вона підтримує об'єктно-орієнтоване програмування, що спрощує розробку, розширення та підтримку коду.

Spring Framework є одним з найпопулярніших фреймворків для розробки програмного забезпечення на Java. Він надає широкі можливості для управління залежностями, інверсії керування, реалізації шаблонів проєктування та інтеграції з іншими технологіями.

Vaadin є потужним фреймворком для розробки веб-інтерфейсу користувача. Він надає зручні засоби для побудови інтерактивних та естетично привабливих веб-додатків. За допомогою Vaadin можна створити багатофункціональний та відзивчивий інтерфейс для зручного взаємодії з користувачем.

MySQL надає широкі можливості для зберігання, організації та маніпуляції даними, а використання Amazon RDS дозволяє забезпечити масштабованість, високу доступність та безпеку бази даних, а також спростити процес управління і підтримки.

Крім того, обрані засоби реалізації враховують важливі аспекти, такі як безпека та конфіденційність даних, що є критичними для успішної роботи програмного продукту. Також, користувацький інтерфейс та його мінімалістичний дизайн сприятимуть зручній взаємодії з продуктом та покращенню користувацького досвіду.

Обрані засоби реалізації взаємодіють між собою, утворюючи сильний та ефективний технологічний стек для розробки застосунку. Кожен інструмент вносить свій внесок у різні аспекти розробки та забезпечує важливу функціональність і переваги.

Загалом, обґрунтування вибору засобів реалізації дозволяє максимально використовувати потенціал обраних технологій та інструментів, забезпечуючи ефективну та якісну розробку програмного продукту. Правильно обрані засоби допоможуть досягти поставлених цілей проєкту та задовольнити потреби користувачів.

Таким чином, обґрунтування вибору засобів реалізації є важливим кроком у процесі розробки програмного продукту, який забезпечує надійність, ефективність та якість його реалізації.

# 3 СТРУКТУРА СИСТЕМИ

Цей розділ є ключовою складовою частиною дослідження і присвячений розгляду структурної організації системи тайм-менеджменту і концентрації. Вивчення структури системи тайм-менеджменту дозволить розкрити її функціональні можливості, ідентифікувати ключові елементи та їх взаємозв'язки, а також з'ясувати, як ці елементи сприяють досягненню поставлених цілей та оптимальному використанню ресурсів.

Дослідження структури системи тайм-менеджменту дозволить отримати глибше розуміння її функціонування та можливостей. Знання про структурні елементи та їх взаємозв'язки стане основою для подальшого розвитку та вдосконалення системи, а також сприятиме досягненню більш ефективного управління часом і досягненню поставлених цілей.

## 3.1 Основні сутності

У системі було визначено декілька основних сутностей. У Java Persistence API (JPA) сутності використовуються для відображення даних з бази даних на об'єкти в програмі. Сутність є класом, який представляє таблицю в базі даних, а об'єкти цього класу відповідають записам цієї таблиці. Основними сутностями в застосунку є Task (задача), Tag (тег), User (користувач) і Role (роль).

Сутність Task представляє окрему задачу, яку користувач може створити. Задачі можуть мати різні статуси, відображені у enum-полі TaskStatus, такі як "TODO" (ще не виконано), "INPROGRESS" (в процесі виконання) та "DONE" (виконано). Також, задачі можуть мати пріоритет, представлений у enum-полі TaskPriority, як "Low" (низький), "Medium" (середній) та "High" (високий).

Сутність Tag використовується для класифікації та групування задач. Користувач може призначити один або декілька тегів до кожної задачі, що дозволяє легко здійснювати пошук задач за певними категоріями або темами.

Сутність User представляє окремого користувача системи. Користувачі мають різні статуси, визначені у enum-полі UserStatus, такі як "Active" (активний) і

"Banned" (заблокований). Користувачі мають доступ до функціоналу системи, можуть створювати, редагувати та видаляти свої задачі. Якщо користувач заблокований, він не може користуватись системою.

Сутність Role визначає рівень доступу та повноваження користувача в системі. Ролі визначені у enum-полі RoleEntity, такі як "User" (звичайний користувач) та "Admin" (адміністратор). Різні ролі мають різний рівень доступу і можуть виконувати дозволені дії в системі, такі як управління користувачами, керування задачами та налаштування профілю.

Рівень доступу користувачів регулюється за допомогою Vaadin анотації @RolesAllowed, в якій ми вказуємо, яка саме роль дозволена для тієї чи іншої сторінки. Анотації @DenyAll, @PermitAll і @AnonymousAllowed також застосовуються до відповідних класів перегляду.

Spring Data JPA надає потужний механізм автоматичної генерації таблиць за допомогою анотацій, що дозволяє зручно визначати сутності та їх залежності. Наприклад, при використанні анотації @Entity над класом Task, Spring Data JPA розпізнає його як сутність, яка повинна бути збережена в базі даних. Крім того, при використанні таких анотацій, як @Id, @GeneratedValue і @Column, буде визначено первинний ключ, автоматично сгенеровано його значення та назви стовпців у таблиці.

Валідація даних у сутностях також зроблена за допомогою анотацій Hibernate. Серед таких анотацій — @Length (для встановлення мінімальної та макисмальної довжини значення), @NotEmpty (заборона поля бути пустим), @Email (для валідації електронної пошти). Даний механізм дозволяє не тільки слідкувати за тим, аби невалідні дані не потрапили до бази даних, а ще й «прокидувати» це повідомлення нагору, в інтерфейс, для повідомлення користувача.

Для встановлення зв'язків між сутностями, такими як Task і Tag, використовуються анотації @ManyToMany або @OneToMany, залежно від типу зв'язку. При цьому Spring Data JPA автоматично створить допоміжну таблицю з зовнішнім ключем, яка відображатиме зв'язок між цими двома сутностями. Анотації, такі як @JoinTable і @JoinColumn, дозволяють налаштовувати назви

таблиць та стовпців для зберігання інформації про зв'язки, як це показано на рисунку 3.1. Також було налаштовано автоматичну генерацію ID.

```
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(
        name = "tasks_tags",
        joinColumns = @JoinColumn(
                name = "task_id", referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(
                name = "tag_id", referencedColumnName = "id"))
private Set<Tag> tags;
```

Рисунок 3.1 — Встановлення зв’язків між сутностями Task та Tag

Загальна структура пакету з сутностями, їх полями та допоміжними enum наведена на рисунку 3.2.

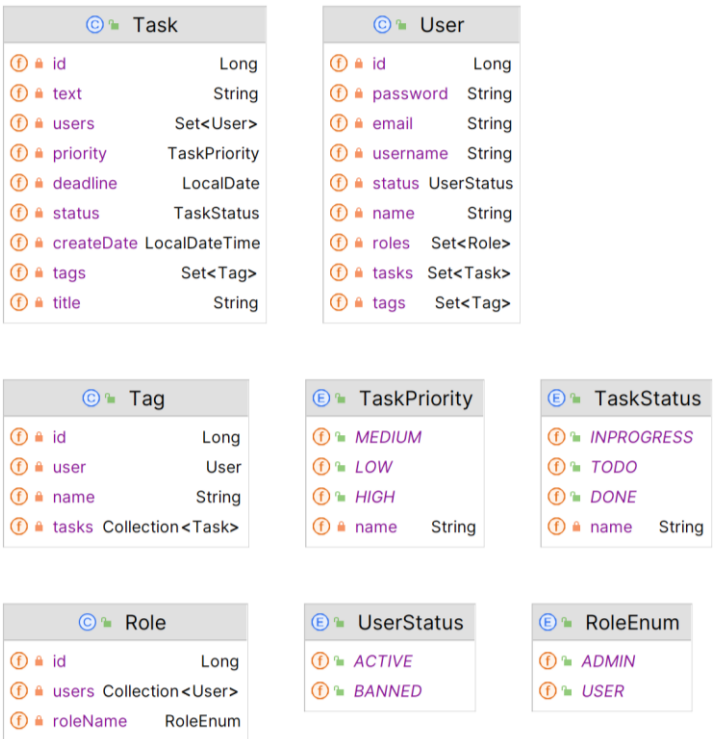



Рисунок 3.2 — Сутності пакету application.data.entity

Як видно з переліку сутностей на рисунку 3.2, кожна сутність має відповідні поля, а деякі з сутностей — для покращення безпеки та читабельності — допоміжні enum. Використання таких enum дозволить обмежити варіативність сутностей.

## 3.2 Створення бази даних

Для створення бази даних додатку було обрано хмарний сервіс Amazon Relational Database Service (Amazon RDS), який є набором керованих сервісів, який спрощує створення, використання та масштабування бази даних у хмарі. Після реєстрації акаунту було налаштовано розмір бази даних і тип інстансу, обсяг пам'яті, ім'я, параметри доступу, параметри забезпечення, такі як налаштування автоматичного резервного копіювання, моніторингу тощо. Приклад готової до використання бази можна побачити на рисунку 3.3

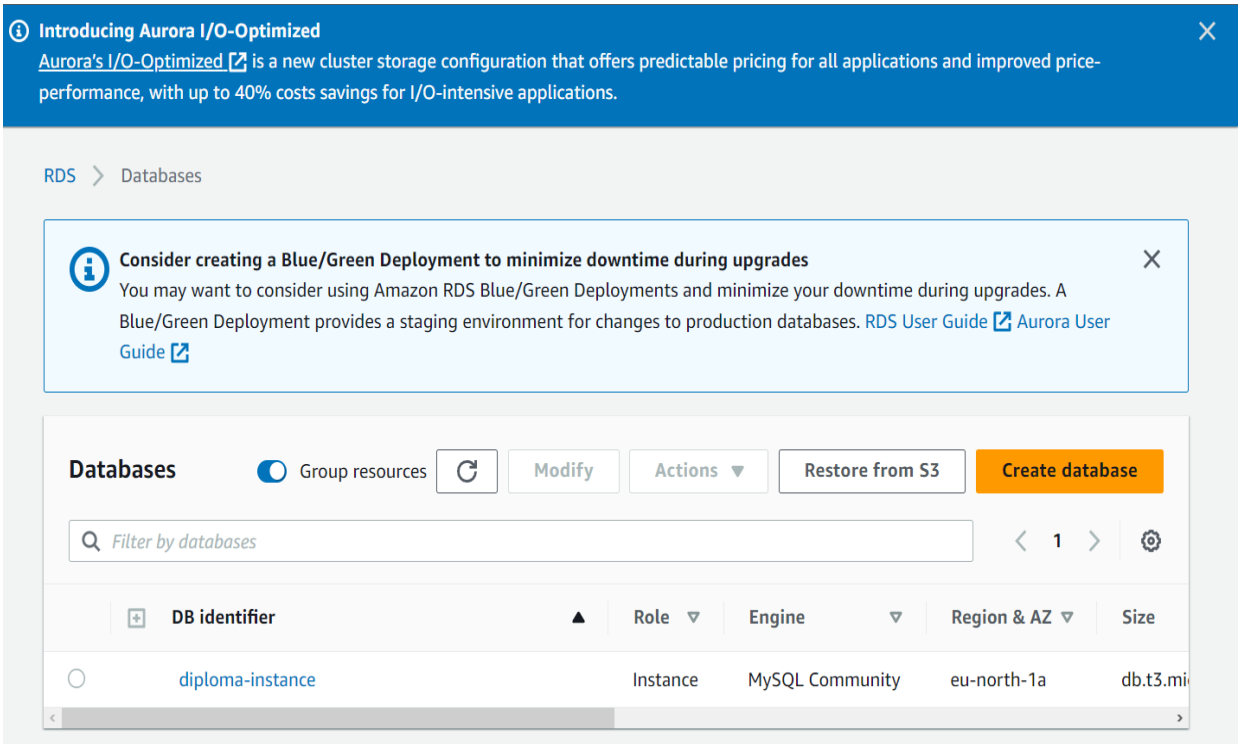



Рисунок 3.3 — Новостворена база даних у Amazon RDS

Після успішного розгортання бази даних, застосунок було підключено до неї за допомогою кредів, визначених у файлі application.properties. У цьому ж файлі було зроблено деякі інші налаштування, пов’язані з роботою бази даних.

Завдяки Hibernate основні та допоміжні (для зв’язків many-to-many) таблиці були створені автоматично з сутностей, які були описані раніше, таким чином, було збережено час, необхідний для розробки застосунку. За допомогою скрипта було

додано дві основні ролі та адміністратора. Схему створеної бази даних наведено на кресленику ІА92.05БАК.004 Д3.

### 3.3 Структура модулів, пакетів та класів

У корені проєкта знаходиться файл pom.xml у я кому підключено залежності (Spring Data JPA, Spring Security, Vaadin, Lombok тощо). З основних модулів можна виділити:

– application.components.appnav — модуль, в якому зберігаються компоненти навігації;

– application.data — модуль який містить сутності, репозиторії та сервіси, що займаються бізнес-логікою застосунку;

– application.security — модуль, що інкапсулює механізм авторизації та автентифікації;

– application.views — модуль, в якому зберігається інтерфейс застосунку та байндери між інтерфейсом та сервісами.

Модуль application.data містить пакети «entity» (зберігає сутності застосунку), «repository» (зберігає механізми доступу до даних), «service» (зберігає сервіси, які виконують бізнес-логіку) та «util» (утилітний пакет).

Модуль application.security містить наступні класи:

– SecurityConfiguration, що розширює клас VaadinWebSecurity, що налаштовує доступи до різних сторінок і встановлює представлення логіну, та містить бін PasswordEncoder, який дозволяє шифрувати паролі;

– UserDetailsServiceImpl, який імплементує UserDetailsService, перевіряє користувача та дозволяє підтягнути його ролі і, відповідно, права доступу;

– AuthenticatedUser, який дозволяє швидко віднайти користувача, що був авторизовний, а також розлогінити його.

Модуль application.views містить серверні представлення Java і також розподілений на менші пакети: «auth» (пакет з інтерфейсами та байндерами логіну та реєстрації), «noise» (пакет з інтерфейсами та байндерами фонових звуків та

таймеру), «notes» (містить інтерфейс та байндери карток задач), «task» (містить інтерфейси та байндери, пов’язані зі створеннями, редагуванням задач та тегів), «userEdits» (зберігає інтерфейси та байндери редагування користувачів та профілю) та «util» (утилітний пакет). Окремо лежить клас MainLayout, який є відправною точкою навігації і координує переміщення між тими чи іншими інтерфейсами.

Також можна виділити пакет resources, у якому лежать файли налаштувань, sql-скрипт, фонові звуки тощо. У пакеті frontend знаходяться клієнтські представлення програми на JavaScript та кастомні стилі CSS. Окремо лежить файл pom.xml, за допомогою якого підключаються залежності (Spring, Spring Security, Lombok, Vaadin тощо). Варто зауважити, що для розробки однієї з «фіч» застосунку, було обрано кастомний віджет Vaadin Simple Timer [11].

Запуск застосунку відбувається за допомогою класу Application, позначеного анотацією @SpringBootApplication. Анотація використовується для позначення класу конфігурації, який оголошує один або більше методів @Bean, а також запускає автоматичне налаштування та сканування компонентів. Ця анотація об’єднує три інші анотації: @Configuration, @EnableAutoConfiguration і @ComponentScan. SpringApplication клас використовується для завантаження та запуску програми Spring з основного методу Java. Цей клас автоматично створює ApplicationContext зі шляху до класів, сканує класи конфігурації та запускає програму.

Залежності між модулями, пакетами та класами впроваджуються за допомогою Dependency Injection — реалізації IoC принципа, основна суть якого — «інжектити» залежності там, де вони потрібні, з мінімальною участю розробника. «Інжект» залежностей може бути за допомогою сеттерів або, як у випадку розроблюваного застосунку, конструкторів. Сама конфігурація бінів може бути в XML-файлі, за допомогою Java-коду або анотацій [17]. Так як в розроблюваному застосунку немає причин писати складну конфігурацію, використано конфігурацію на анотаціях, яка є найбільш зрозумілою та читабельною. Більш детально залежності пакетів та класів один від одного та інших пакетів наведено на кресленику ІА92.050БАК.004 Д2.

## 3.4 Архітектура застосунку

Два основні фреймворки, на яких побудовано застосунок — Vaadin та Spring Boot, обидва є Java-фреймворками, але використовуються для різних цілей.

Vaadin — це фреймворк для створення веб-додатків, зокрема для створення інтерфейсів користувача. Він має набір готових компонентів інтерфейсу користувача та інструментів для створення адаптивних та інтерактивних інтерфейсів. Він також надає архітектуру на стороні сервера для обробки запитів між браузером і сервером, а також API для створення логіки на стороні клієнта.

Vaadin складається з API на стороні сервера, API на стороні клієнта, компонентів інтерфейсу користувача/віджетів, тем для керування зовнішнім виглядом і моделі даних, яка дозволяє прив'язувати компоненти на стороні сервера безпосередньо до даних. Для розробки на стороні клієнта він включає компілятор Vaadin, який дозволяє компілювати Java у JavaScript.

Spring Boot — це «надбудова» над Spring, структура для створення вебзастосунків. Він надає набір інструментів для налаштування та автоматичної конфігурації, включаючи вбудований веб-сервер, підключення до бази даних і інструменти захисту і безпеки.

Таким чином, Vaadin — це веб-фреймворк Java, який зосереджується на створенні інтерфейсу користувача та логіки на стороні клієнта, тоді як Spring Boot — це фреймворк, який має на меті покращити процес створення програм на стороні сервера за допомогою фреймворку Spring.

Програма Vaadin на стороні сервера працює як сервлет на веб-сервері Java, обслуговуючи запити HTTP. VaadinServlet зазвичай використовується як клас сервлету. Сервлет отримує клієнтські запити та інтерпретує їх як події для конкретного сеансу користувача. Події пов'язуються з компонентами інтерфейсу користувача та доставляються до Event Listener, визначених у програмі. Якщо користувач вносить зміни в компоненти інтерфейсу на стороні сервера, сервлет генерує відповідь і відтворює їх у веб-браузері,. Механізм на стороні клієнта, що

працює в браузері, отримує відповіді та використовує їх для внесення будь-яких необхідних змін на сторінці браузера.

Узагальнена структура застосунку представлена на рисунку 3.4, більш детальна структура представлена на кресленику ІА92.050БАК.004 Д1. Принцип роботи застосунку такий, що користувач надсилає запит на сервер, використовуючи інтерфейс, створений з вбудованих або додаткових компонентів Vaadin, які мають Event Listener. Зв'язок між представленням з компонентів Vaadin, написаним на Java, та сутностями на бек-енд частині застосунку відбувається за допомогою байндерів (рисунок 3.5), які також надає Vaadin. Клас Binder дозволяє встановити, як значення в бізнес-об'єкті прив'язуються до полів в інтерфейсі користувача.

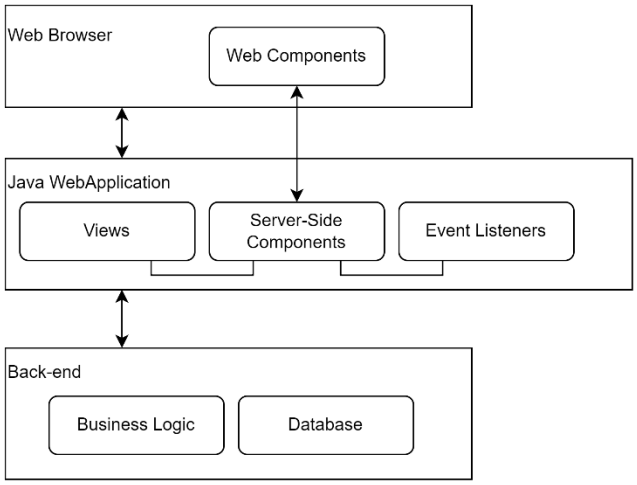



Рисунок 3.4 — Узагальнена схема застосунку

```
public void addBindingAndValidation() {
    binder.bindInstanceFields(addTagDialog);

    binder.setStatusLabel(addTagDialog.getErrorMessageField());

    addTagDialog.getSaveButton().addClickListener(event -> {
        try {
            final var bean = new Tag();
            binder.writeBean(bean);
            final var resultMessage : UtilNotification = tagService.addTag(bean);
            showResult(resultMessage);
            addTagDialog.close();
            UI.getCurrent().getPage().reload();
        } catch (Exception e) {
            showResult(new UtilNotification(UtilStatus.ERROR, UtilMessages.UNEXPECTED_ERROR_OCCURRED));
        }
    });
}
```

Рисунок 3.5 — Байндер для об'єкту тега

Після того, як дані з UI форм були збережені у сутностях, з ними можна працювати за допомогою відповідних сервісів: редагувати, зберігати, видаляти тощо.

Репозиторій (repository) та сервіс (service) є двома основними компонентами, що використовуються для доступу до даних та виконання бізнес-логіки. Репозиторій відповідає за безпосередню взаємодію з базою даних і забезпечує доступ до даних через CRUD-операції (створення, читання, оновлення, видалення). Сервіс відповідає за виконання бізнес-логіки та координацію роботи з репозиторієм. Він містить методи для обробки та маніпуляції даними перед збереженням у базі даних. Також може включати складну логіку, валідацію даних, обробку подій та інші операції, які не прямо пов'язані з доступом до бази даних. Сервіс забезпечує відокремлення бізнес-логіки від презентаційного шару та полегшує тестування та підтримку коду.

Робота у сервісах та репозиторіях побудована за допомогою Spring Data JPA, який дозволяє не писати SQL-запити до бази даних вручну, а використовувати вбудовані механізми. При цьому, якщо все ж таки потрібна складна логіка у запиті, можна використати JPQL або native запит [16]. Окрім того, Spring бере на себе управління транзакціями, достатньо лише використати анотацію @Transactional.

Висновок до розділу

У цьому розділі було розглянуто деталі щодо архітектури додатку, а саме:

– основні сутності, які представляють, валідують, пов’язують та інкапсулюють дані;

– автоматичне створення бази даних, її зв’язків, таблиць та їх обмежень завдяки Hibernate;

– структура, призначення, налаштування та взаємозв’язки модулів, пакетів та класів;

– загальна архітектура застосунку, побудованого на фреймворках Vaadin та Spring Boot.

Структура застосунку, заснована на даних сутностях, дозволяє зорієнтуватися у функціональності та даних, що обробляються. Кожна сутність відображає важливий аспект додатку і має відповідні атрибути та зв'язки з іншими сутностями.

Створення бази даних забезпечує надійне зберігання та організацію даних. Використання AWS RDS MySQL надає можливість масштабувати базу даних залежно від потреб проєкту, забезпечуючи ефективну роботу з даними, дозволяє не встановлювати на машині сервер MySQL та економити пам’ять, і — що найважливіше — делегувати роботу з підтримки, логування, забезпечення захисту та інтеграції хмарному сервісу.

Обрана структура модулів, пакетів та класів дозволяє організовано розподілити функціональність додатку та полегшує його розробку та підтримку. Кожен модуль, пакет та клас має свої чіткі обов'язки, що сприяє чистоті та структурованості коду.

Комбінація Vaadin та Spring Boot надає можливість Java-розробнику розробляти фул-стек застосунок із користувацьким інтерфейсом та ефективною обробкою бізнес-логіки, не написавши жодного рядка на JavaScript.

Загальною метою цього розділу було створення цілісної структури застосунку, яка враховує основні елементи функціоналу, бази даних та архітектуру. В цілому, розглянута структура застосунку є достатньо організованою та ефективною. Вона надає зручність у розробці, підтримці та розширенні застосунку, а також забезпечує якісну роботу з даними та користувацький інтерфейс.

# 4 АНАЛІЗ РОЗРОБЛЕНОГО ДОДАТКУ

## 4.1 Особливості реалізації

Відповідно до визначених у підрозділі 2.1.1 функціональних вимог були імплементовані наступні рішення:

– створення акаунту шляхом реєстрації з іменем, логіном, паролем та поштою;

– авторизація за допомогою логіна та пароля;

– можливість логауту;

– створення навігації між сторінками;

– сторінка з налаштуванням профілю користувача, зокрема його іменем, поштою, логіном та паролем;

– сторінка додавання нової задачі, що має обов’язкові поля, такі як заголовок та текст, та необов’язкові поля, такі як дедлайн, статус, пріоритет та список тегів;

– вікно редагування існуючої задачі;

– можливість створити новий тег;

– сторінка з усіма задачами;

– відображення задачі у вигляді картки, де присутні заповнені поля та її дата створення, а також кнопка видалення, редагування та додавання задачі у сьогоднішній список справ;

– кнопка переходу до сторінки з таймером та секундоміром;

– кнопка переходу до сторінки з фоновими звуками;

– навігація по згрупованих тегах у вигляді окремих посилань;

– можливість увімкнути темну тему для зменшення навантаження на очі вночі та усування відблиску;

– модуль адміністратора, який дозволяє банити користувачів та призначати нових адміністраторів.

Розроблений застосунок відповідає визначеним у підпункті 2.1.2 нефункціональним вимогам, а саме є зручним та мінімалістичним, не має інформаційного шуму, який може відволікати та збивати з пантелику, є безпечним

завдяки використанню модулів захисту, зокрема механізму шифрування паролів. Окрім того, застосунок є масштабованим, так як використовує для створення інтерфейсу компоненти Vaadin, які підлаштовуються під поточний пристрій, а для зберігання даних — Amazon RDS, яка відповідає за їх безпеку, резервне копіювання, моніторинг тощо.

## 4.2 Розбір та демонстрація функціоналу додатку

Детальна діаграма діяльності наведена на кресленику ІА92.05БАК.004 Д4.

Робота з додатком починається з реєстрації. Для цього необхідно перейти на сторінку логіну та обрати пункт «Sign Up», після чого з’явиться форма реєстрації, як показано на рисунку 4.1. Можна перейти назад на сторінку авторизації за допомогою кнопки «Login». Обов'язковими для заповнення є усі поля. При спробі не заповнити поле воно помічається червоним, а натискання кнопки підтвердження не має ніякого ефекту, адже валідацію не пройдено. Результат такої поведінки можна побачити на рисунку 4.2.

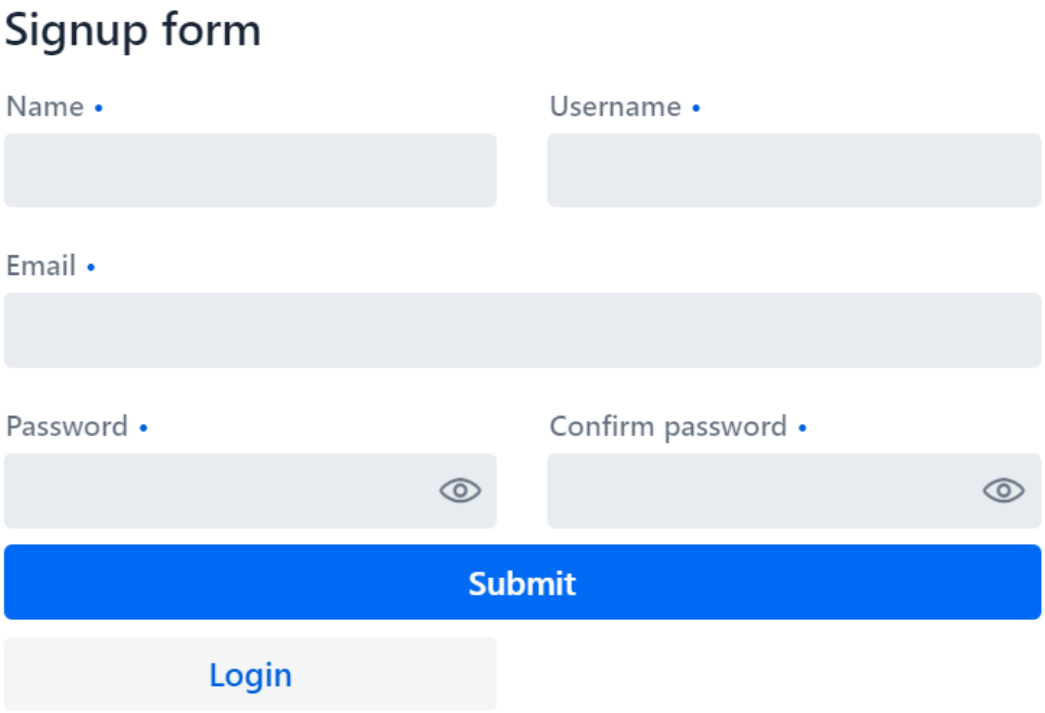



Рисунок 4.1 — Форма реєстрації

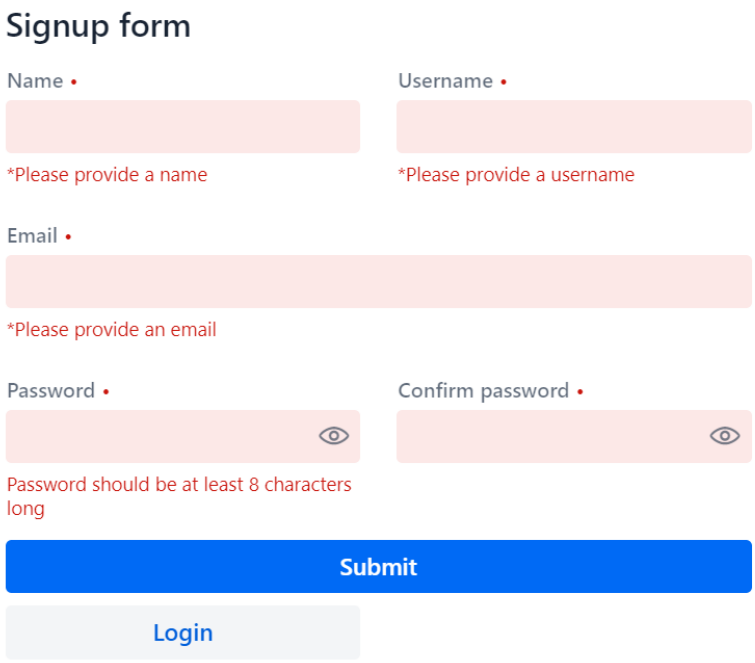



Рисунок 4.2 — Форма реєстрації при непройденій валідації

Існують і інші типи валідації: наприклад, якщо електронна пошта має некоректну форму, якщо паролі не співпадають або ж спроба зареєструвати користувача під іменем, що вже зареєстровано, як можна побачити на рисунках 4.3 та 4.4.

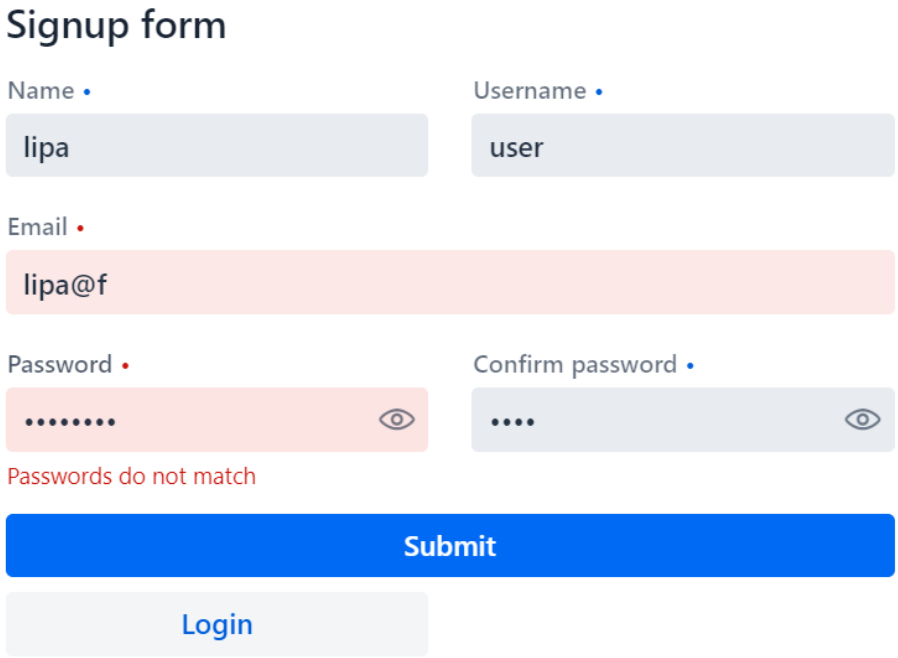



Рисунок 4.3 — Форма реєстрації при некоректному заповненні полів

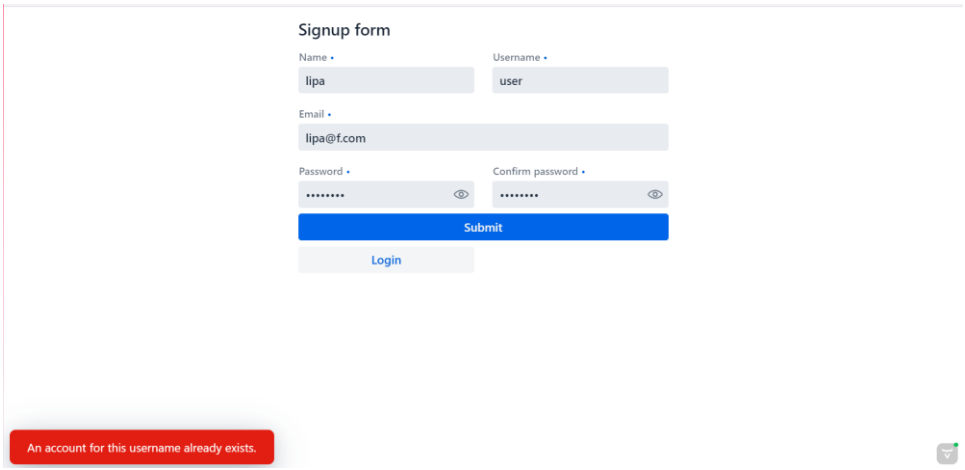



Рисунок 4.4 — Повідомлення при спробі зареєструвати той же юзернейм

Після успішної реєстрації отримуємо відповідне повідомлення (рисунок 4.5):

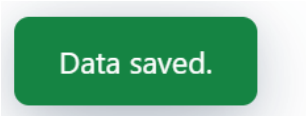



Рисунок 4.5 — Повідомлення при успішній реєстрації

Після успішної реєстрації можна перейти до сторінки авторизації (рисунок 4.6). Форма також має валідацію — при спробі авторизуватись під неправильним паролем, неіснуючим або забаненим користувачем отримуємо помилку, як показано на рисунку 4.7:

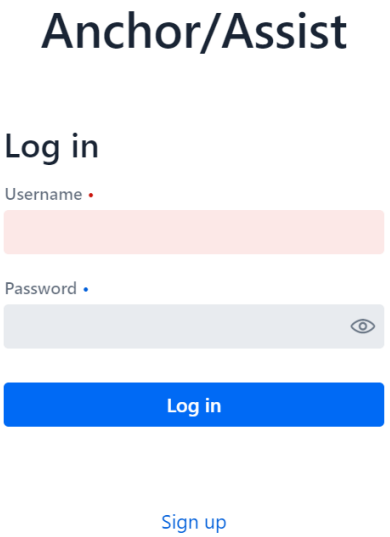



Рисунок 4.6 — Сторінка авторизації

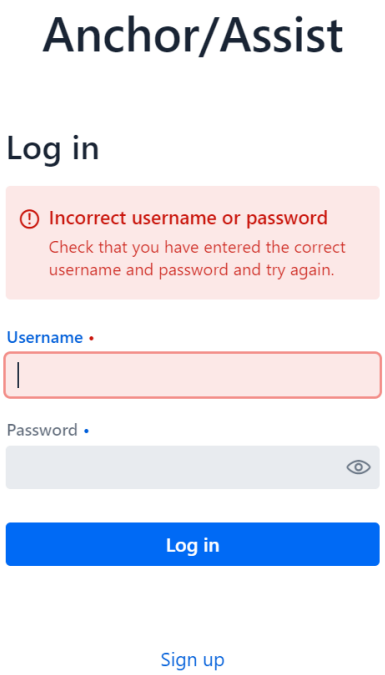



Рисунок 4.7 — Повідомлення при неуспішній авторизації

Після того, як користувач успішно авторизувався, він опиняється на сторінці усіх задач. Так як користувач не додав жодної задачі, він бачить відповідне повідомлення «You haven’t added any task yet.» на сторінці. При цьому користувач може користуватись навігацією — йому доступна кнопка усіх задач, створення задачі, таймеру та фонових звуків, а у нижній панелі — вихід з акаунту, налаштування профілю та ввімкнення темної теми. Кнопка, розташована поряд з заголовком поточної сторінка дозволяє згорнути навігацію. Вигляд головної сторінки можна побачити на рисунку 4.8:

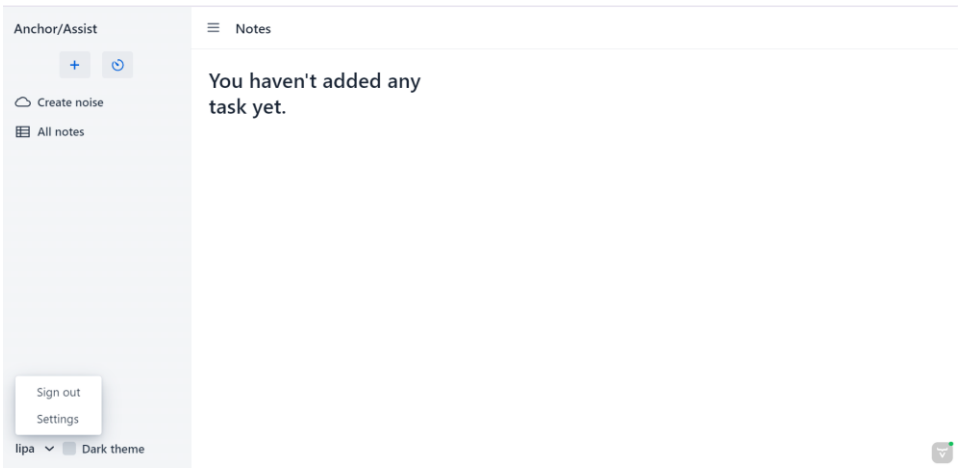



Рисунок 4.8 — Головна сторінка

Для додавання задачі користувачу необхідно натиснути кнопку «+» на панелі зліва, після чого його перекине на відповідну сторінку. Поля додавання задачі також мають валідацію — поля «Заголовок» та «Текст» є обов'язковими. Окрім того також можна заповнити поля «Статус», «Пріоритет», «Дедлайн» та «Теги». «Статус» та «Пріоритет» мають випадні списки з варіантами значень. Для «Статусу» це «TODO», «INPROGRESS» та «DONE», для «Пріоритетету» — «LOW», «MEDIUM» та «HIGH», також можна обрати пусте поле. Дедлайн обирається з випадного календаря і також є опціональним полем. Список тегів має вигляд випадного списку з існуючих, доданих користувачем тегів, при цьому тегів можна додавати один або кілька. Для додавання нового тегу необхідно натиснути «+» та заповнити відповідне поле.

Також присутня валідація полів «Title» та «Text». Ці поля не повинні бути пустими, також вони мають обмеження по довжині (для «Title» — 50 символів, для «Text» — 1500 символів).

При цьому, під час відображення поля «Text» у картці на головній сторінці, якщо довжина тексту більша за 200 символів, текст буде скорочено для перегляду, залишковий текст замінено на «…» . Повний текст можна буде побачити при натисканні на редагування картки. Валідацію полів, заповнення форми та створення тегу можна побачити на рисунках 4.9, 4.10, 4.11:

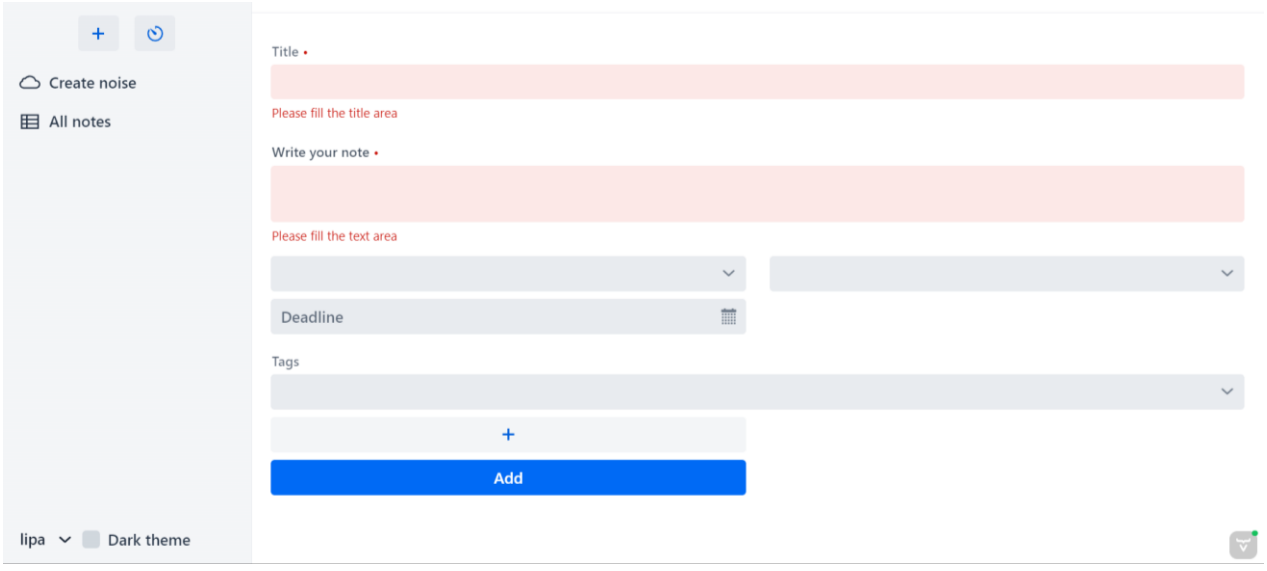
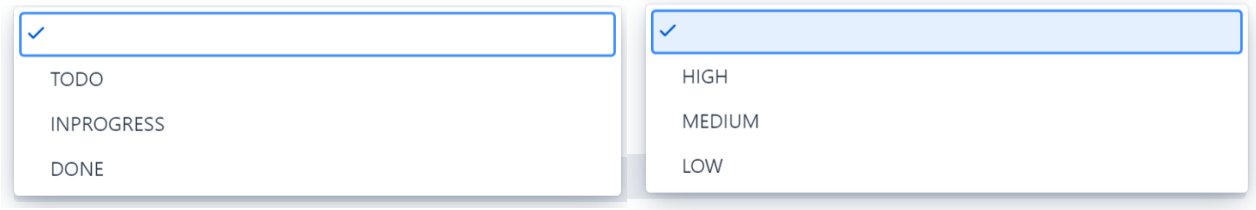
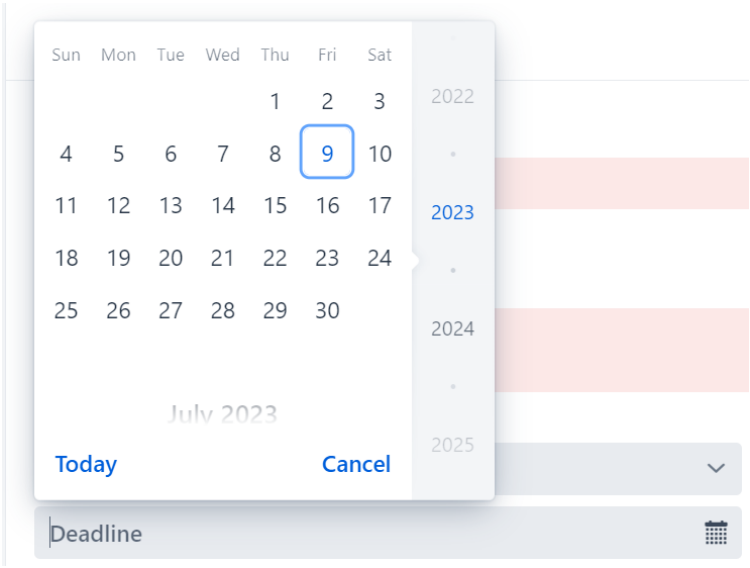



Рисунок 4.9 — Загальний вигляд форми створення задачі та її валідація




а) б)




в)

Рисунок 4.10 — Випадні списки:

а) статусу; б) пріоритету; в) дедлайну

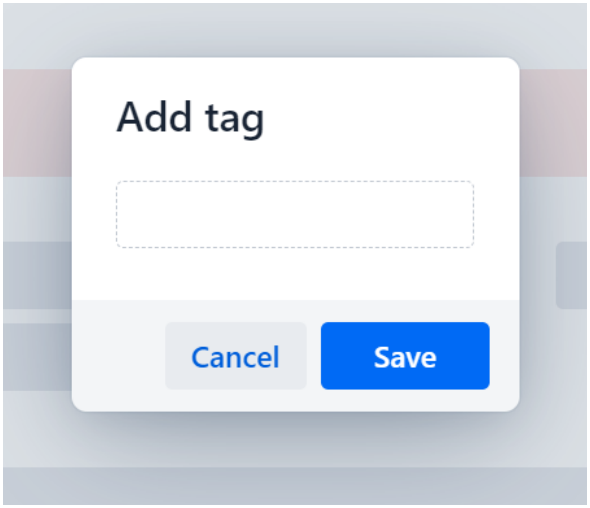



Рисунок 4.11 — Додавання тегу

Після додавання задачі можна побачити відповідне повідомлення (рисунок 4.12), якщо ж при додаванні сталася якась помилка, буде отримане повідомлення про помилку (рисунок 4.13).

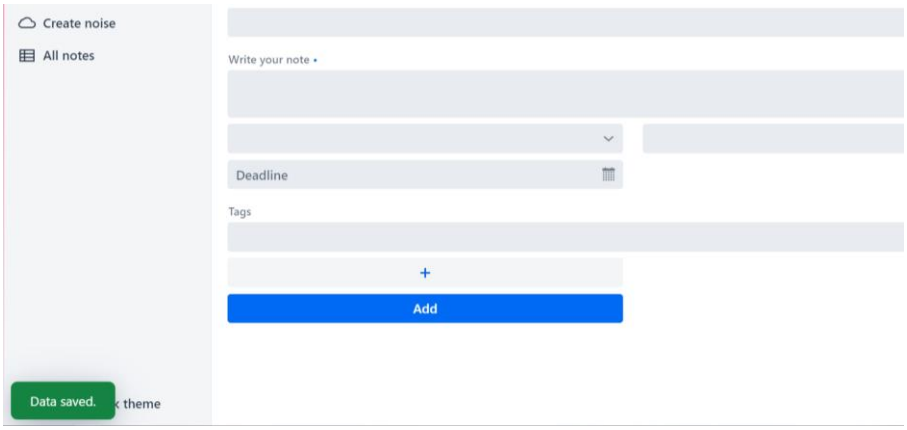



Рисунок 4.12 — Успішне повідомлення додавання задачі

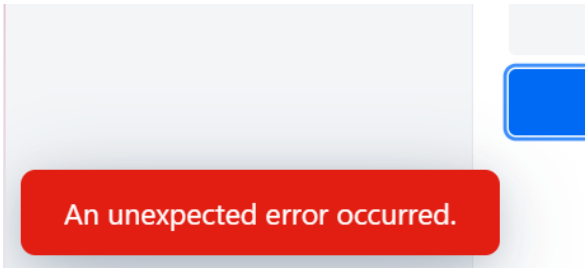



Рисунок 4.13 — Повідомлення про помилку під час додавання задачі

Після створення задача буде відображатись на головній сторінці. На рисунку 4.14 представлено просту задачу, без тегів, дедлайну, статусу та пріоритету, та складну задачу з усіма елементами. В залежності від пріоритету задачі, поле пріоритету може бути сірого (для «Low»), синього (для «Medium») та червоного кольору (для «High»). Внизу картки, біля тегу, розташована дата створення задачі. Також зліва можна побачити, що з’явилась навігація по задачам тегу (рисунок 4.15).

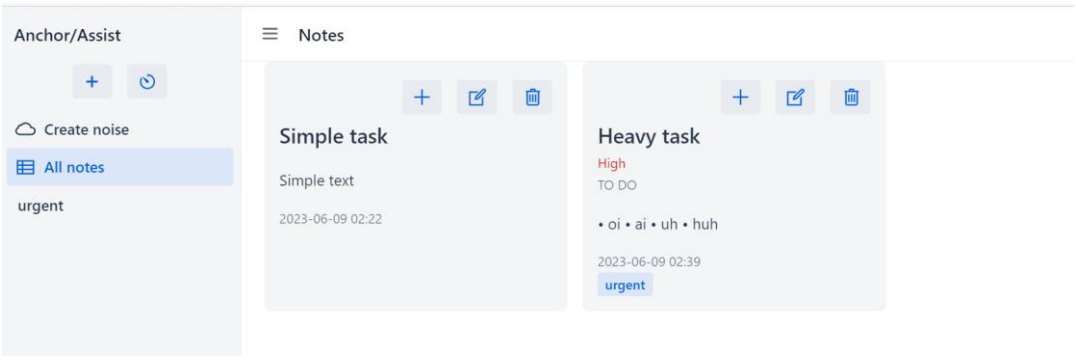



Рисунок 4.14 — Головна сторінка з кількома задачами

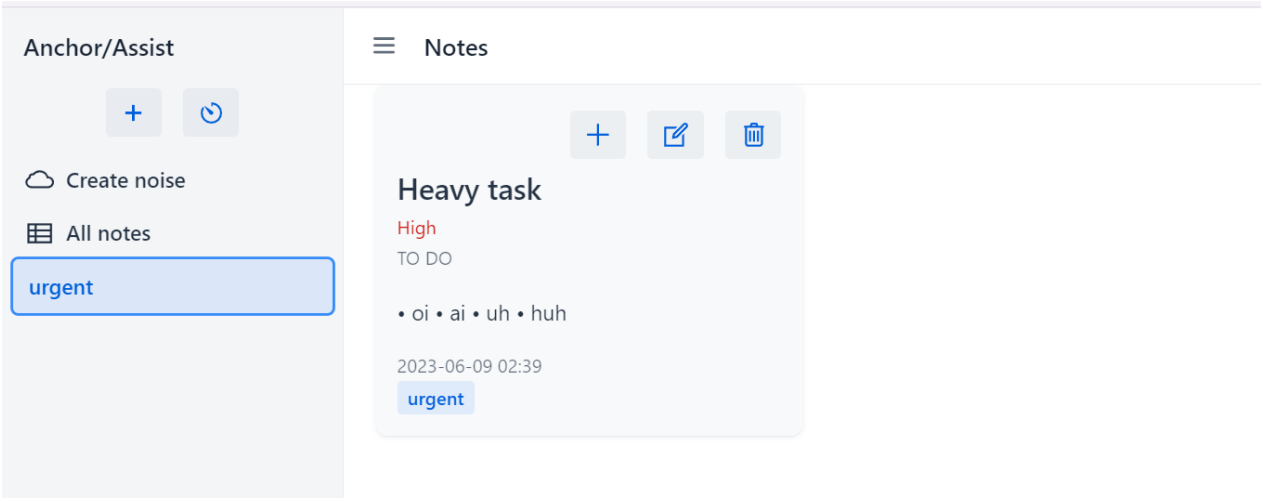



Рисунок 4.15 — Головна сторінка з задачами тегу

Для того, щоб відредагувати задачу, необхідно натиснути кнопку редагування з олівцем, після чого з'явиться відповідна форма та зелене повідомлення (рисунок 4.16).

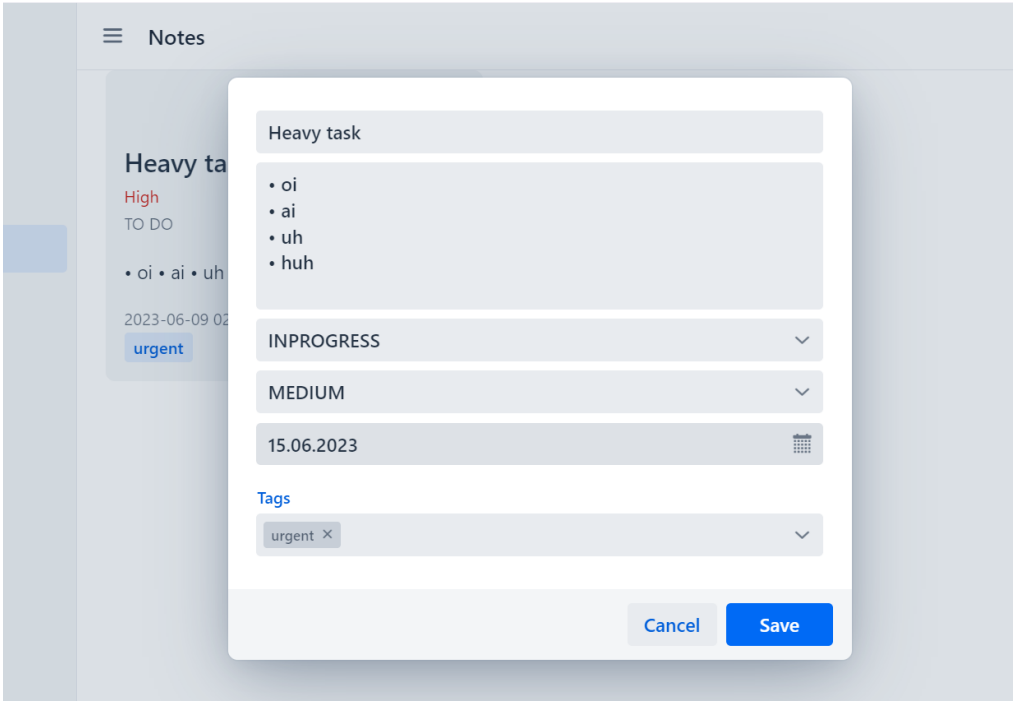
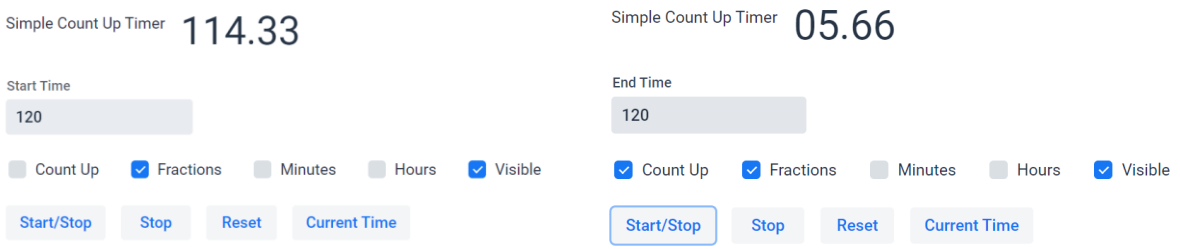



Рисунок 4.16 — Сторінка редагування задачі

Для переходу на сторінку таймера необхідно натиснути кнопку у вигляді годинника, після чого користувач потрапить на відповідну сторінку, де він зможе встановити час як для таймера, так і для секундоміра (рисунок 4.17):




а) б)

Рисунок 4.17 — Сторінка таймера:

а) таймер; б) секундомір

Для створення фонового шуму використовується відповідна кнопка у вигляді хмари. На сторінці звуків (рисунок 4.18) ми можемо запускати різні звуки та комбінувати їх. Для зупинки звуку необхідно ще раз натиснути на відповідний символ.

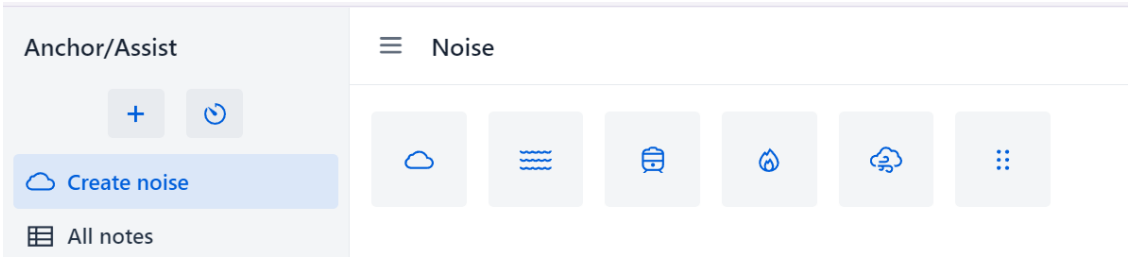
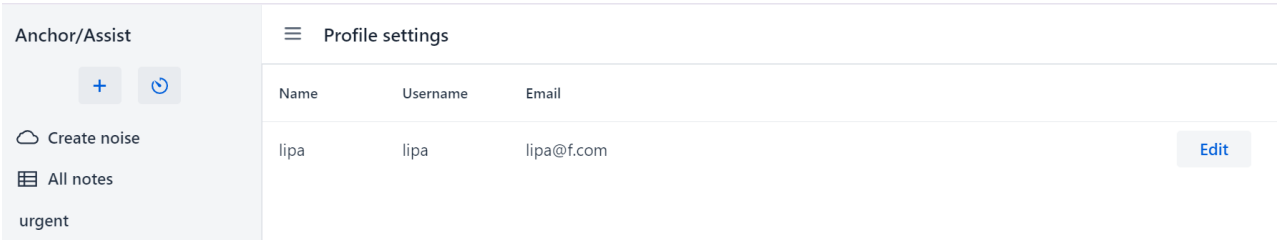
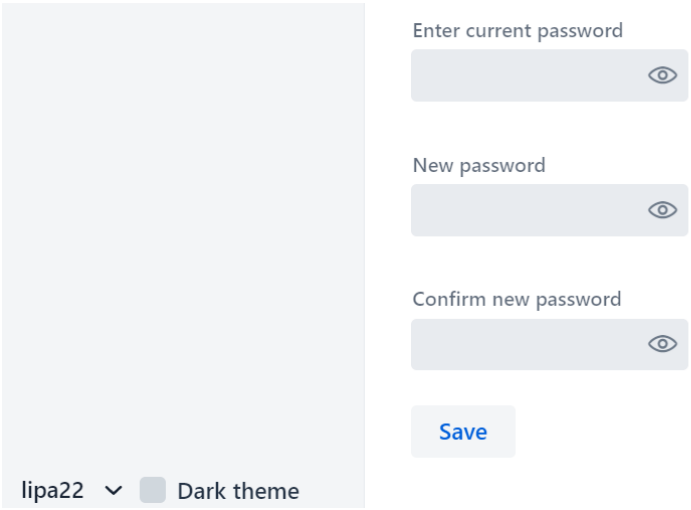



Рисунок 4.18 — Сторінка фонових звуків

Користувач також може редагувати свій профіль, для цього необхідно в панелі нижньої навігації обрати команду «Settings», після цього відкриється сторінка (рисунок 4.19), на якій можна змінити своє ім’я, пошту, логін та пароль. Для створення нового паролю необхідно ввести старий пароль.




а)




б)

Рисунок 4.19 — Сторінка налаштувань профілю:
а) звичайні налаштування; б) пароль

Під час редагування профілю також діє валідація полів (рисунок 4.20), ім’я, логін та пошта не можуть бути порожніми, окрім того пошта повинна мати коректну форму.

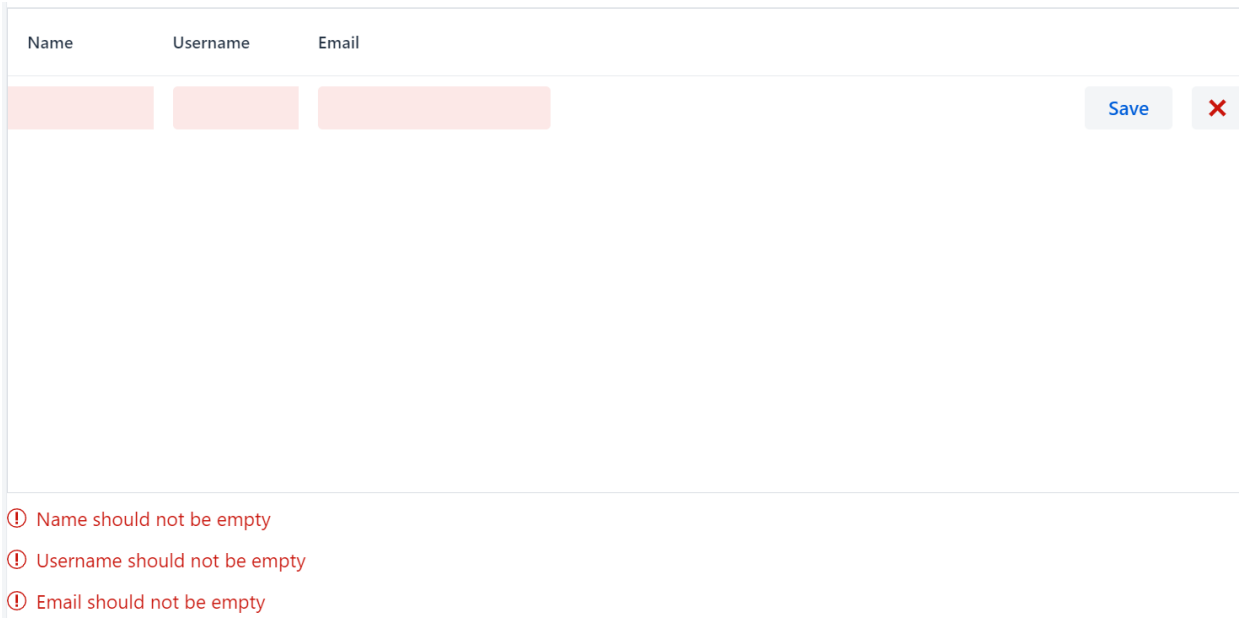



Рисунок 4.20 — Валідація налаштувань профілю

Після зміни імені можна перевірити, що воно дійсно змінилось — у нижній панелі відображається нове ім’я, як показано на рисунку 4.21:

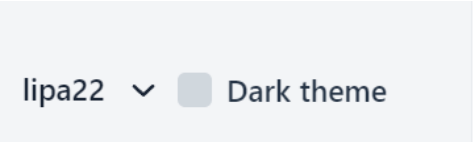



Рисунок 4.21 — Результати редагування

Для того, щоб перевірити роботу адмін панелі, авторизуємось під адміністратором. Адміністратор може банити користувачів та призначати нових адміністраторів, також йому доступний увесь функціонал звичайних користувачів. На рисунку 4.22 можемо побачити панель адміністратора зліва та список користувачів по середині.

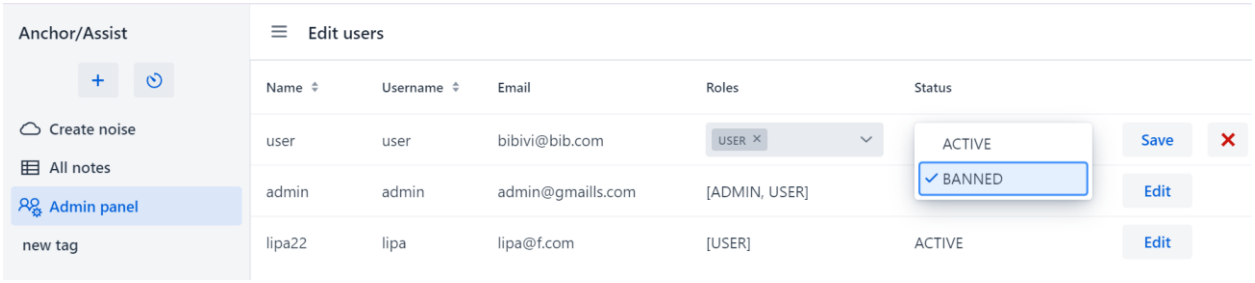



Рисунок 4.22 — Панель адміністратора

Однією з функцій застосунку є зміна теми на темну (рисунок 4.23) за допомогою відповідного чекбокса внизу сторінки біля профілю користувача. На цьому ж рисунку можна побачити, що адміністратор може змінити роль користувача а також валідацію, адже роль не може бути порожньою:

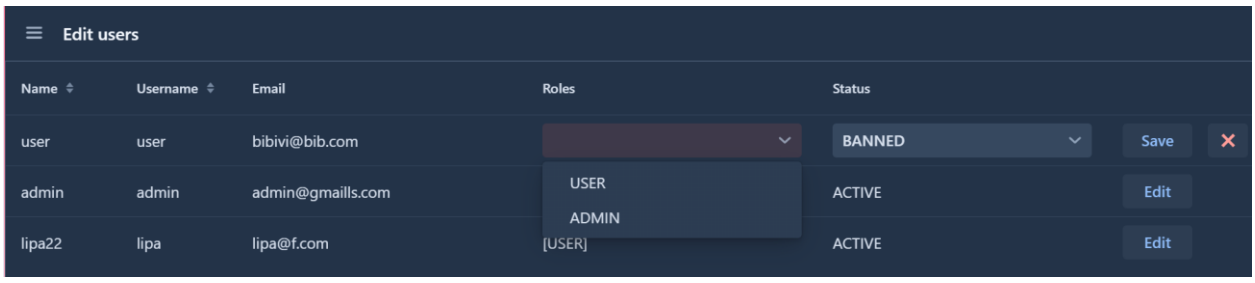
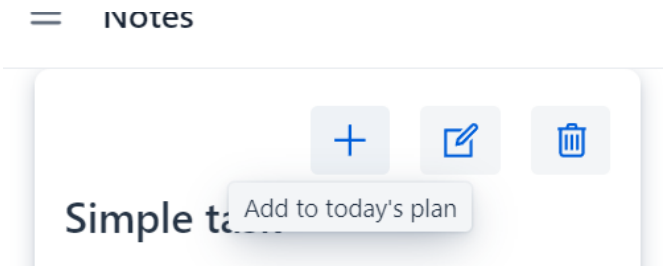


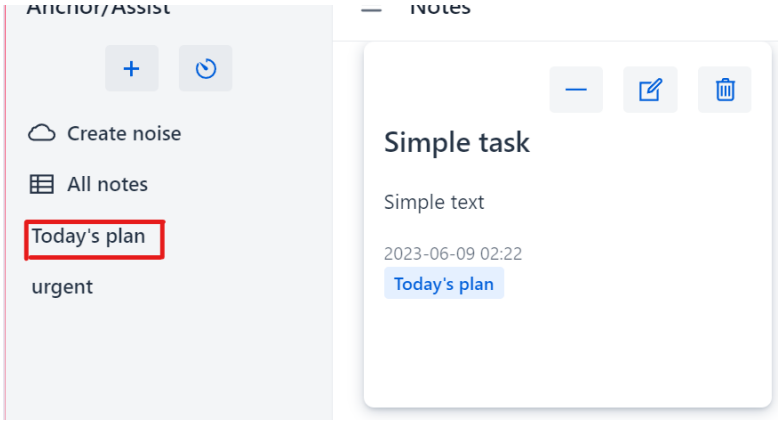

Рисунок 4.23 — Панель адміністратора в темній темі

Також задачу можна додати або видалити з сьогоднішнього плану задач за допомогою відповідної кнопки «+», як це показано на рисунку 4.24. Після того, як

задача була додана в сьогоднішній план, знак «+» зміниться на «—». За допомогою цієї ж кнопки задачу можна видалити з плану.




а)




б)

Рисунок 4.24 — Сьогоднішній план:

а) додавання задачі; б) видалення задачі

Окрім того, саму задачу можна видалити за допомогою відповідної кнопки (рисунок 4.25):

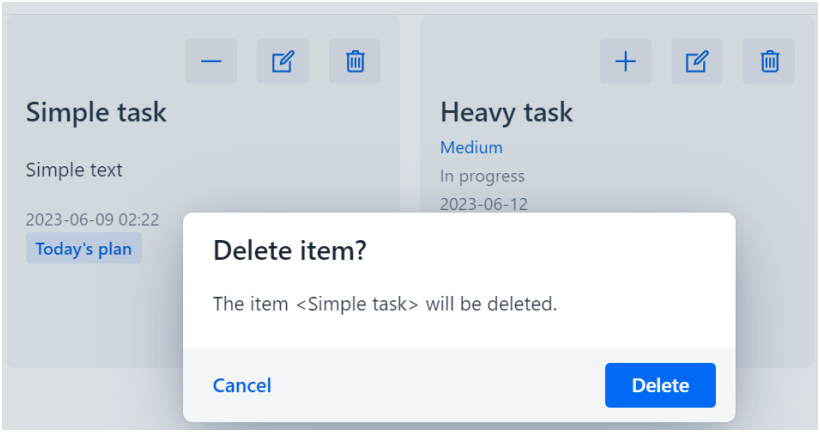



Рисунок 4.25 — Видалення задачі

## 4.3 Порівняння застосунку з аналогами

Проведемо порівняльний аналіз з існуючими аналогами, що допоможе визначити унікальні переваги застосунку, його конкурентоспроможність та привабливість для користувачів.

У порівнянні з конкурентами застосунок має такі переваги:

– простота та інтуїтивність інтерфейсу — на відміну від інших «наворочених» застосунків, розроблювана система виглядає куди більш простою та зрозумілою, не має відволікаючих факторів та інформаційного шуму;

– все в одному місці — користувачу не потрібно шукати той чи інший сайт, де він міг би увімкнути фонові звуки або таймер, тепер це все знаходиться поряд у навігації розроблюваного застосунку;

– можливість коротко описати найголовніші аспекти задачі (статус, дедлайн тощо);

– підтримка існуючого функціоналу не є складною.

Підсумовуючи, застосунок має суттєві переваги, які можуть допомогти йому створити конкуренцію вже існуючим програмам, особливо привабливим він буде для індивідуальних користувачів.

## 4.4 Рекомендації щодо подальшого вдосконалення

Застосунок має потенціал розвиватись далі та імплементувати нові «фічі». Завдяки простоті написання коду, це не повинно стати складним завданням. Можна виділити наступні пропозиції:

– групова робота над задачами — не зважаючи на те, що додаток з самого початку був орієнтований на сінгл-користувачів, він має потенціал до подібного функціоналу, адже ще на самому старті під час проєктування бази даних було додано механізм зберігання цілого списку користувачів, що працюють над задачами, а не лише одного;

– нагадування — потужний функціонал, який ще більше підвищив би концентрацію над роботою та результативність;

– можливість додавання фото до задачі;

– поради з концентрування на роботі, які завжди були б під рукою;

– сторінка Board View, на якій задачі розподілялись по статусам;

– фільтрування та сортування задач;

– аналітика витрати часу на завдання;

– зберігання налаштувань музики, створення плейлистів;

– локалізація додатку різними мовами.

Висновок до розділу

У даному розділі було проаналізовано розроблюваний додаток, його функціонал та можливості покращення у майбутньому.

Розроблена програма успішно втілює основні функціональні та нефункціональні вимоги, включаючи створення, відстеження та керування завданнями, управління тегами, аутентифікацію та авторизацію користувачів, концентрацію на роботі.

Було продемонстровано функціонал користувача та адміністратора, в першу чергу керування задачами, налаштування профілю, засоби концентрації та управління з боку адміністратора. Застосунок також відповідає принципам мінімалістичного інтерфейсу, безпеки даних та ефективності.

У цілому, розроблений додаток виконує свою основну функцію тайм-менеджменту і має потенціал для розвитку та оптимізації. Завдяки подальшому вдосконаленню та врахуванню рекомендацій, застосунок може стати ще більш привабливим для користувачів і краще відповідати їхнім потребам.

# ВИСНОВКИ

Метою проєкту є розробка та впровадження інноваційного застосунку, що надає користувачам ефективні інструменти для кращого управління своїм часом та задачами. Проєкт спрямований на надання зручного та інтуїтивно зрозумілого інтерфейсу, який допоможе користувачам планувати, організовувати та виконувати свої завдання, підвищуючи їх результативність. Мета полягає у створенні рішення, яке допоможе забезпечити більшу концентрацію, організованість та успішне виконання робочих завдань. Проєкт прагне зробити процес управління часом більш простим, структурованим та приємним для користувачів, сприяючи їхньому особистому та професійному зростанню.

Аналіз існуючих рішень та порівняння з аналогами дозволили зрозуміти, які «фічі» застосунку були б найбільш затребуваними, і вказали, в якому напрямі необхідно робити розробку. Повторний аналіз конкурентів показав, що розроблений застосунок відповідає потребам і вимогам проєкту. Він вирішує основні завдання тайм-менеджменту, надаючи користувачам зручні інструменти для планування, відстеження та організації їх завдань та концентрації на них. Крім того, застосунок підтримує безпеку та конфіденційність даних, а також надає користувачам зручний інтерфейс для взаємодії з системою.

У процесі розробки застосунку було проведено детальний аналіз вимог, враховуючи як функціональні, так і нефункціональні аспекти. В пояснювальній записці було детально розглянуто інструменти реалізації застосунку, серед яких мова, якою велось програмування, фреймворки для серверної та клієнтської частини, засоби створення, підтримки та роботи з базою даних, середовища розробки.

Java була обрана як мова програмування, оскільки вона є потужним інструментом для створення розширених та масштабованих додатків. Spring і Vaadin використовувалися для побудови надійної та ефективної архітектури, що забезпечує зручний інтерфейс користувача та легку роботу з базою даних.

MySQL та Amazon RDS були використані для створення надійної та масштабованої бази даних, що забезпечує безпеку та доступність інформації.

Lombok допоміг у спрощенні розробки, зменшивши кількість надлишкового коду та підтримуючи чистоту та читабельність проєкту.

У ході розробки було визначено ключові сутності, їх атрибути, типи; для користувачів було виділено дві ролі (звичайного користувача та адміністратора) зі своїми правами. На основі цих сутностей була створена база даних за допомогою Amazom RDS. Клієнтська та серверна частина написані на чистій Java, з використанням фреймворків Spring та Vaadin.

По результатам роботи можна зауважити, що обидва фреймворки Spring та Vaadin мають потужний функціонал, і хоча вони непогано працюють у комбінації, при розробці іноді виникали незручності саме з Vaadin. Не зважаючи на те, що його функціонал дозволяє не замислюватись над реалізацією фронтенд-частини, він іноді може мати баги або бути просто недостатньо зручним у порівнянні з тим же JavaScript. Також серед недоліків — дещо повільна робота застосунку, яка найімовірніше за все також пов’язана з Vaadin. Таким чином, можна зробити висновок, що для розробки невеликих, ненагружених складним інтерфейсом пет-проєктів Vaadin підходить, але для більш складних застосунків, тим паче серйозної комерційної розробки, не можна його порекомендувати.

Не зважаючи не деякі недоліки у розробці, застосунок має значний потенціал до покращення, імплементування якого може зробити додаток популярним.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Procrastination in Daily Working Life: A Diary Study on Within-Person Processes That Link Work Characteristics to Workplace Procrastination. Frontiers. URL: https://www.frontiersin.org/articles/10.3389/fpsyg.2018.01087/full (дата звернення: 12.05.2023).
2. Scientific management: Frederick Winslow Taylor's gift to the world?. Boston : Kluwer Academic Publishers, 1996. 191 c.,
3. Wang X., Gobbo F., Lane M. Turning Time from Enemy into an Ally Using the Pomodoro Technique. Agility Across Time and Space. Berlin, Heidelberg, 2010. P. 149–166. URL: https://doi.org/10.1007/978-3-642-12442-6_10 (дата звернення: 28.05.2023).
4. Music, Health, and Wellbeing / ed. by R. MacDonald, G. Kreutz, L. Mitchell. Oxford University Press, 2012. 568 p.
5. Davisson K. The Effects of Background Music on Cognitive Performance. Music & Science Lab. URL: https://musicsciencedurham.files.wordpress.com/2021/10/davisson.pdf (дата звернення: 01.06.2023).
6. Cooper B. B. The best productivity system for procrastinators is to work with your natural tendencies. Quartz. URL: https://qz.com/752614/the-best-productivity-system-for-procrastinators-is-to-work-with-your-natural-tendencies (дата звернення: 10.06.2023).
7. Chapter 6. Aspect Oriented Programming with Spring. Spring | Home. URL: https://docs.spring.io/spring-framework/docs/2.5.5/reference/aop.html (дата звернення: 18.05.2023).
8. Spring - Difference Between Inversion of Control and Dependency Injection - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/spring-difference-between-inversion-of-control-and-dependency-injection/ (дата звернення: 02.06.2023).

9. Bcrypt: An Overview. URL: https://mattfsewell.medium.com/bcrypt-an-overview-e3ef89950189 (дата звернення: 02.06.2023).
10. Project Lombok. Project Lombok. URL: https://projectlombok.org (дата звернення: 08.06.2023).
11. GitHub — FlowingCode/SimpleTimerAddon: Vaadin Flow integration of https://github.com/annsonn/simple-timer. GitHub. URL: https://github.com/FlowingCode/SimpleTimerAddon (дата звернення: 10.06.2023).
12. Lutkevich B. What is Amazon RDS (Relational Database Service)?. SearchAWS. URL: https://www.techtarget.com/searchaws/definition/Amazon-Relational-Database-Service-RDS (дата звернення: 11.05.2023).
13. Transactions with Spring and JPA | Baeldung. Baeldung. URL: https://www.baeldung.com/transaction-configuration-with-jpa-and-spring (дата звернення: 05.05.2023).
14. Spring Security. Spring | Home. URL: https://docs.spring.io/spring-security/reference/index.html (дата звернення: 15.05.2023).
15. Overview | Vaadin Architecture | Framework | Vaadin 8 Docs. Vaadin | Discover the Web App Framework Built for Java. URL: https://vaadin.com/docs/v8/framework/architecture/architecture-overview (дата звернення: 27.05.2023).
16. Spring Data JPA @Query | Baeldung. Baeldung. URL: https://www.baeldung.com/spring-data-jpa-query (дата звернення: 01.06.2023).
17. Multiple ways to configure Spring. A Java geek. URL: https://blog.frankel.ch/multiple-ways-configure-spring/ (дата звернення: 07.06.2023).

# ДОДАТОК А

https://github.com/YashaBlendeer/anchor-assist