

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Система для управління та організації проектів»

Виконала студентка групи ІПЗ-19-1
МАКОВСЬКА Ольга-Марія Юріївна

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **МАКОВСЬКА Ольга-Марія Юрївна**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: «Система для управління та організації проектів»,
затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Вихідні дані для роботи: використання веб-орієнтованих технологій для розробки комплексної системи обліку та планування, що враховує специфічні потреби та вимоги підприємств

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	22.02.2023р.	22.02.2023р.
2	Власенко О.В.	16.04.2023р.	16.04.2023р.
3	Власенко О.В.	02.05.2023р.	02.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	2 лютого 2023 – 15 лютого 2023	Виконано
2	Аналіз моделей аналогів	16 лютого 2023 – 28 лютого 2023	Виконано
3	Постановка технічного завдання	1 березня 2023 – 10 березня 2023	Виконано
4	Вивчення літературних джерел	11 березня 2023 – 15 березня 2023	Виконано
5	Проектування структури	16 березня 2023 – 25 березня 2023	Виконано
6	Написання програмного коду	26 березня 2023 – 14 квітня 2023	Виконано
7	Тестування роботи програми	15 квітня 2023 – 21 квітня 2023	Виконано
8	Аналіз одержаних результатів	22 квітня 2023 – 25 квітня 2023	Виконано
9	Оформлення пояснювальної записки	10 травня 2023 – 5 червня 2023	Виконано
10	Попередній захист	26 квітня 2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з комплексної програмної системи для управління та організації проектів, а також пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 71 сторінку, 33 ілюстрації, 4 таблиці, 3 додатки та 14 літературних джерел.

Основною метою даної роботи є розробка алгоритмів та реалізація веб-програмного комплексу, який покращує процеси обліку та планування, підвищує ефективність та точність, оптимізує розподіл ресурсів та покращує загальну ефективність.

В роботі визначено ключові завдання, пов'язані з розробкою системи, та проведено аналіз наявних систем, подібних до розроблюваної, а також наведено сценарії роботи програмного комплексу. Представлено огляд загальної структури системи, описано алгоритми взаємодії між окремими модулями системи, а також наведено опис об'єктної структури (діаграму класів) системи. Крім того, в статті представлено результати тестування системи.

КЛЮЧОВІ СЛОВА: ВЕБ-СИСТЕМА, CRM, УПРАВЛІННЯ ЧАСОМ, ПЛАНУВАННЯ, УПРАВЛІННЯ ЗАВДАННЯМИ, КАЛЕНДАР.

					ІПЗ.КР.Б – 121 – 23 – ПЗ								
Зм.	Арк.	№ докум.	Підпис	Дата	Система для управління та організації проектів				Літ.		Арк.	Аркушів	
Розроб.		О.Ю. Маковська											
Керівник		О.В. Власенко										2	71
									Житомирська політехніка, група ІПЗ-19-1				
Рецензент		О.Г. Чижмотря											
Зав. каф.		А.В.Морозов											

ABSTRACT

The bachelor's thesis consists of a comprehensive software system for project management and organization, as well as an explanatory note. The explanatory note to the thesis contains 71 pages, 33 illustrations, 4 tables, 3 appendices, and 14 references.

The main purpose of this thesis is to develop algorithms and implement a web-based software system that improves accounting and planning processes, increases efficiency and accuracy, optimizes resource allocation, and improves overall efficiency.

The paper identifies key tasks related to the development of the system, analyses existing systems similar to the one under development, and presents scenarios of the software complex. An overview of the general structure of the system is presented, the algorithms of interaction between individual modules of the system are described, and the object structure (class diagram) of the system is described. In addition, the article presents the results of testing the system.

KEYWORDS: WEB-BASED SYSTEM, CRM, TIME MANAGEMENT, PLANNING, TASK MANAGEMENT, CALENDAR.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	5
ВСТУП.....	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	8
1.1 Постановка задачі.....	8
1.2 Аналіз аналогів програмного продукту	11
1.3 Розробка архітектури веб-системи планування робочого часу	15
1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення	18
Висновки до розділу 1	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ПЛАНУВАННЯ РОБОЧОГО ЧАСУ ПІДПРИЄМСТВА	21
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури	21
2.2 Розробка структури бази даних системи.....	28
2.3 Проектування та реалізація алгоритмів роботи системи	34
2.4 Реалізація функціоналу системи планування робочого часу підприємства ..	40
Висновки до розділу 2	44
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ СИМУЛЯЦІЇ	45
3.1 Порядок встановлення та налаштування системи.....	45
3.2 Інтерфейс та порядок роботи.....	50
3.3 Тестування роботи системи	59
Висновки до розділу 3	62
ВИСНОВКИ.....	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	65
ДОДАТКИ.....	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVC – Model-View-Controller (Модель–представлення–контролер)

DDD – Domain-Driven Development (Предметно-орієнтоване проектування)

RBAC – Role-Based Access Control (Керування доступом на основі ролей)

ERD – Entity Relationship Diagram (Модель «сутність-зв'язок»)

DI – Dependency Injection (Впровадження залежності)

UML – Unified Modeling Language (Уніфікована мова моделювання)

CRM – Customer Relationship Management (Система управління взаємовідносинами з клієнтами)

CRUD – create, read, update and delete (створення, читання, модифікація, видалення)

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						5

ВСТУП

Актуальність теми. Належний тайм-менеджмент є критично важливим фактором, який дозволяє компаніям ефективно розподіляти ресурси, дотримуватися кінцевих термінів і досягати цілей. У сучасному конкурентному бізнес-середовищі, де час є цінним та обмеженим ресурсом, для організацій стає все більш важливим мати надійні системи для оптимізації цих процесів. Ефективне управління часом не лише сприяє успіху окремих проектів, але й має значний вплив на загальну продуктивність і прибутковість організації.

З широким розповсюдженням Інтернету та зростаючою залежністю від цифрових рішень зростає попит на веб-системи, які можуть допомогти в управлінні часом. Ці системи пропонують цілий ряд переваг, включаючи оптимізацію процесів, покращення співпраці, посилення безпеки і точне відстеження часу і витрат. Як результат, організації активно шукають рішення для своїх проблем з часом і отримання конкурентних переваг на ринку [14].

Впровадивши веб-систему для обліку та планування робочого часу на підприємстві, організації можуть досягти наступних переваг:

Підвищення продуктивності: веб-система забезпечує централізовану платформу, де працівники можуть легко реєструвати свій робочий час, відстежувати прогрес проекту та керувати завданнями.

Ефективний розподіл ресурсів: додаток забезпечує видимість робочого навантаження співробітників, що дозволяє менеджерам призначати завдання належним чином і уникати перевантаження людей.

Покращена співпраця: системи сприяють співпраці у команді, надаючи централізовану платформу для спілкування та управління завданнями. Члени команди можуть отримати доступ до інформації, пов'язаної з проектом, обмінюватися оновленнями та співпрацювати над завданнями.

Точне відстеження часу та витрат: менеджер дозволяє організаціям фіксувати оплачувані години, контролювати бюджети проектів і створювати точні звіти для виставлення рахунків клієнтам і фінансового аналізу.

Підвищена безпека даних: веб-системи забезпечують автентифікацію, шифрування даних і механізми контролю доступу для захисту від несанкціонованого доступу та витоку даних.

Метою даної кваліфікаційної роботи є розробка алгоритмів та реалізація програмного комплексу для веб-системи, яка полегшує облік та планування. Створення такої системи має на меті підвищити ефективність та точність, оптимізувати розподіл ресурсів та покращити загальну ефективність.

Цей проект спрямований на вирішення проблем, з якими стикаються підприємства в ефективному управлінні та оптимізації процесів планування та обліку робочого часу.

Встановлена мета обумовлює наступні завдання:

- аналіз завдань, пов'язаних з плануванням та обліком часу на підприємстві, а також напрямів їх автоматизації
- розробка загальної структури та визначення ключових компонентів
- вибір відповідних інструментів і технологій для реалізації системи
- впровадження програмного комплексу

Об'єктом дослідження є автоматизація управлінських процесів, пов'язаних з плануванням часу, що є життєво важливим для організацій різних галузей. Автоматизувавши ці процеси, підприємства можуть впорядкувати свою діяльність, зменшити кількість помилок, що допускаються вручну, та підвищити продуктивність.

Предметом дослідження є використання веб-орієнтованих технологій для розробки комплексної системи обліку та планування, що враховує специфічні потреби та вимоги підприємств.

Впровадження веб-системи обліку та планування робочого часу має потенціал для значного підвищення операційної ефективності, покращення використання ресурсів та оптимізації розподілу людського капіталу. Результати цього дослідження можуть призвести до підвищення продуктивності та економічної ефективності організацій, які впровадять цю систему.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

У сфері управління проектами це дослідження спрямоване на вивчення та впровадження передових технологій для розробки програмного рішення, яке покращує планування, виконання та контроль проектів. Метою проекту є створення комплексної та універсальної системи управління проектами, яка дозволить ефективно організовувати, співпрацювати та керувати проектами в різних галузях промисловості та професійних сферах.

Система надасть користувачам можливість створювати власні проекти та ефективно керувати завданнями в них. Вона надаватиме низку важливих функцій, покликаних покращити планування, виконання та моніторинг проектів. Крім того, ця система управління проектами слугуватиме централізованим центром для виконання завдань, пов'язаних з проектами, комунікації та управління документами, що дозволить користувачам залишатися сфокусованими, організованими та продуктивними протягом усього життєвого циклу проекту.

Для досягнення цієї мети проект зосередиться на наступних аспектах:

1. Аналіз проекту та збір вимог:

- 1.1. проведення ретельного аналізу практики управління проектами
- 1.2. співпраця з керівниками проектів та зацікавленими сторонами для збору вимог та визначення основних характеристик і функціональних можливостей
- 1.3. аналіз існуючих процесів управління проектами та визначення сфер, де технологія може підвищити ефективність

2. Вибір та інтеграція технологій:

- 2.1. дослідження та вибір технологій, які відповідають сучасним практикам та методологіям управління проектами

2.2. інтеграція стандартних галузевих інструментів та фреймворків управління проектами

2.3. впровадження надійних заходів безпеки для проектних даних, забезпечення їхньої конфіденційності

3. Розробка та впровадження системи:

3.1. розробка інтуїтивно зрозумілого та зручного інтерфейсу, який надає користувачам проектів цілісне уявлення про хід проекту, розподіл ресурсів та управління завданнями

3.2. розробка модулів і функцій, які дозволяють ефективно планувати терміни виконання проекту, розподіляти ресурси і відстежувати прогрес в режимі реального часу

3.3. впровадження розширених можливостей аналітики та звітності для надання цінної інформації про продуктивність

3.4. інтеграція інструментів комунікації та співпраці

3.5. побудова масштабованої та гнучкої архітектури, яка може пристосовуватися до майбутніх удосконалень та адаптуватися до мінливих потреб управління проектами

4. Тестування та забезпечення якості:

4.1. проведення ретельного тестування, щоб гарантувати, що програмне рішення працює за призначенням і відповідає вимогам

4.2. впровадження процесів забезпечення якості для виявлення та вирішення будь-яких проблем або помилок, які можуть виникнути на етапах розробки та тестування

4.3. отримання зворотного зв'язку від користувачів для підтвердження ефективності та зручності використання програмного рішення

5. Розгортання та підтримка:

5.1. розгортання програмного рішення у виробничому середовищі, забезпечення безперешкодної інтеграції з існуючими системами та процесами управління проектами

5.2. надання вичерпної документації та навчальних матеріалів для полегшення прийняття та використання програмного рішення керівниками проектів та їхніми командами

5.3. надання постійної підтримки та обслуговування для вирішення будь-яких технічних питань, впровадження оновлень та врахування відгуків користувачів

Успішна розробка та впровадження цієї системи управління проектами дасть користувачам доступ до потужного інструменту, який спростить організацію проекту, покращить співпрацю та підвищить загальну ефективність проекту. Вона слугуватиме централізованим центром для виконання пов'язаних з проектом завдань, комунікації та управління документами, що дозволить користувачам залишатися сфокусованими, організованими та продуктивними впродовж усього проекту.

Таким чином, розробка цієї універсальної системи управління проектами задовольняє потребу в комплексному рішенні, різноманітні вимоги різних типів проектів і користувачів. Завдяки використанню передових технологій і включенню необхідних функцій, система дозволить користувачам ефективно планувати, виконувати і контролювати свої проекти.

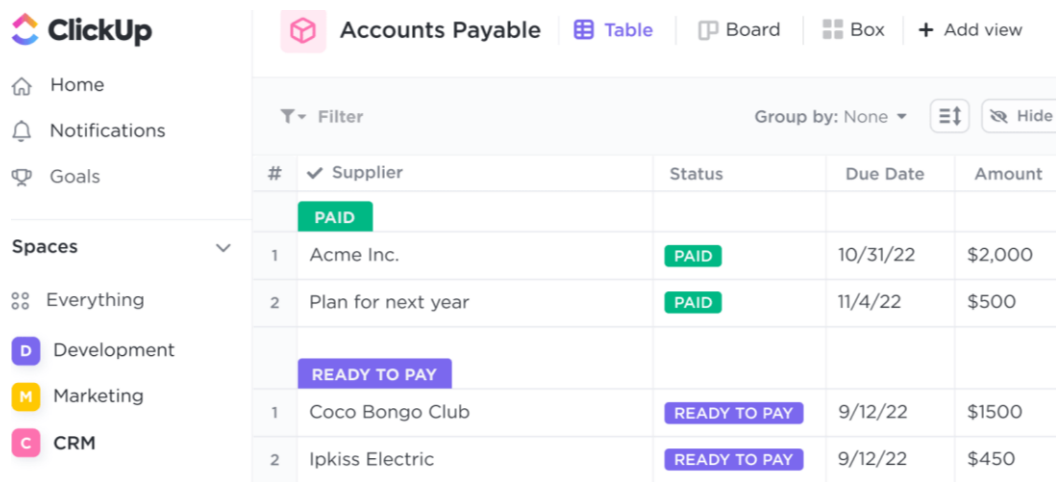
Враховуючи різноманітні вимоги різних типів проектів і користувачів, розробка цієї універсальної системи управління проектами задовольняє потребу в комплексному рішенні. Завдяки використанню передових технологій і включенню необхідних функцій, система дозволить користувачам ефективно планувати, виконувати і контролювати свої проекти.

Масштабованість, орієнтований на користувача дизайн та можливості інтеграції забезпечать довготривалу підтримку та зручність використання. Завдяки успішній реалізації цього проекту користувачі отримають покращені практики управління проектами, оптимізовані робочі процеси та розширену співпрацю, що в кінцевому підсумку призведе до підвищення продуктивності та успіху проектів у різних галузях промисловості та професійних сферах.

1.2 Аналіз аналогів програмного продукту

Існують численні програмні продукти, які допомагають командам і організаціям ефективно керувати своїми проектами. У цьому розділі ми проведемо аналіз трьох популярних інструментів: ClickUp, Asana та Jira.

ClickUp - комплексний інструмент, який пропонує широкий спектр функцій для оптимізації проекту (рис.1.1). Він має зручний інтерфейс і робочу область, що налаштовується. Основні функції включають управління завданнями, командну комунікацію, відстеження часу та постановку цілей [4].



#	Supplier	Status	Due Date	Amount
		PAID		
1	Acme Inc.	PAID	10/31/22	\$2,000
2	Plan for next year	PAID	11/4/22	\$500
		READY TO PAY		
1	Coco Bongo Club	READY TO PAY	9/12/22	\$1500
2	Ipikiss Electric	READY TO PAY	9/12/22	\$450

Рисунок 1.1 - Дошка проекту ClickUp

Переваги:

- зрозумілий і простий у використанні інтерфейс
- широкі можливості налаштування в межах різних проектів та завдань
- можливості управління завданнями (списки, дошки та календарі)
- ефективні функції відстеження часу та звітності

Недоліки:

- розширені функції та можливості можуть потребувати певного часу для освоєння новими користувачами
- можливості інтеграції, хоча і широкі, можуть не охоплювати всі конкретні програми або інструменти

Asana - платформа, що забезпечує централізований центр для співпраці, управління завданнями та відстеження проектів (рис.1.2). Asana пропонує такі функції, як призначення завдань, відстеження прогресу, спільні календарі, обмін файлами [6].

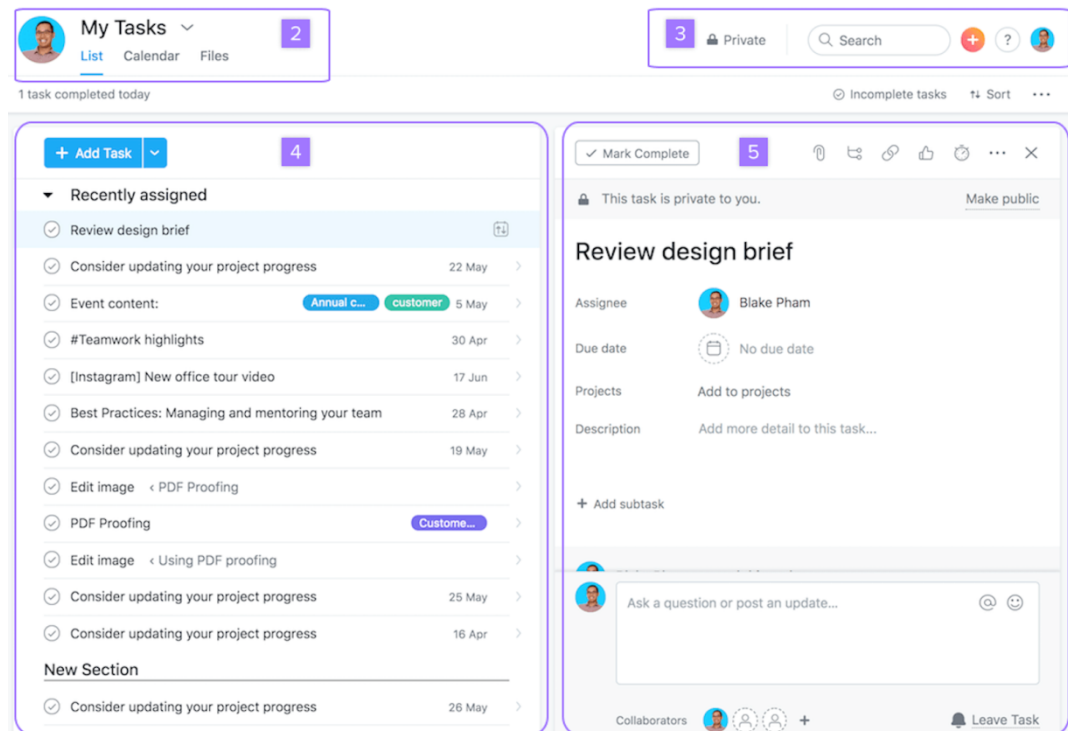


Рисунок 1.2 – Детальне зображення завдання Asana

Переваги:

- інтуїтивно зрозумілі функції співпраці, включаючи спільний доступ до документів і коментарів
- різноманітні режими перегляду (список, дошка та часова шкала)
- інтеграція з різними сторонніми інструментами
- широка підтримка мобільних додатків для доступу на ходу

Недоліки:

- розширені функції доступні лише в дорожчих планах
- обмежені можливості налаштувань представлень завдань і шаблонів
- відсутність вбудованого функціонала для відстеження часу, що вимагає інтеграції із зовнішніми інструментами

Jira - популярний інструмент серед команд розробників (рис.1.3). Він пропонує функції для планування проектів, відстеження завдань та помилок, гнучких методологій та командної співпраці. Гнучкість і масштабованість Jira роблять її придатною для різних галузей і типів проектів [5].

Key	Summary	Progress	Assignee	Priority	Story Points	Time Spent	Status
INIT-1	Core Products		Sarah Johnson	↑	119	6w 4d 2h	IN PROGRESS
INIT-3	Structure		Eugene Lebedev	↑	119	6w 4d 2h	IN PROGRESS
STR-1	Formulas		Eugene Lebedev	↑	60	1w 3d 5h	IN PROGRESS
STR-4	Support for JQL		James Stevens	↑	15		TO DO
STR-10	Design Syntax		Bob Smith	↑			TO DO
STR-5	New UX for editing of large formulas		Bob Smith	↑	26	7 2d 5h	IN PROGRESS
STR-11	Design UX for long formulas editing		Bob Smith	↑	7	2d 5h	DONE
STR-12	Implement new UI			↑	12		TO DO
STR-6	Technical Debt			↑	19	1w 1d	TO DO
STR-3	Sync Attributes to Custom Fields		Jacques Durand	↑	50	4w 1d 5h	TO DO
STR-2	Notes Column		Bob Smith	↑	9	4d	TO DO
INIT-2	Extensions		Mohammed Attar	↑	262	3w 3d 2h	IN PROGRESS
INIT-5	Structure.Gantt		Mohammed Attar	↑	166	3w 3d 2h	IN PROGRESS
GNT-1	Sync Start/Finish Dates to Jira		Ling Hong	↑	66	21	TO DO
GNT-2	Milestones		Mohammed Attar	↑	31	15 1w 4d	DONE
GNT-3	Configurable Resources		Ling Hong	↑	69	32 1w 4d 2h	IN PROGRESS

Рисунок 1.3 – Дошка завдань Jira

Переваги:

- функції, орієнтовані на гнучкість, включаючи дошки Scrum і Kanban, спринти та управління переліком можливостей
- розширені можливості відстеження проблем та управління помилками
- широкі можливості звітності та аналітики для моніторингу прогресу

Недоліки:

- більш крута крива навчання, особливо для нетехнічних користувачів
- складний процес налаштування та конфігурації, що вимагає адміністративного досвіду
- ціна може бути високою для невеликих команд або організацій і повинна відповідати їх можливостям і бюджету

Можливості управління завданнями є важливими для ефективного виконання проекту, в той час як функції спільної роботи сприяють ефективній командній комунікації. Можливості інтеграції з популярними інструментами підвищують продуктивність і спрощують робочі процеси. Цінові міркування є важливими для забезпечення доступності продукту.

Таблиця 1.1

Порівняння існуючих аналогів менеджерів проектів

	ClickUp	Asana	Jira
Простий інтерфейс	+	+	-
Можливості налаштування	+	-	+
Інтеграції	Slack, Google Drive	сторонні інструменти	продукти Atlassian
Доступність (ціна)	+	-	-
Відстеження часу	+	-	обмежені функції
Легкість в освоєнні	+	+	-
Представлення	списки, дошки, календарі	список, дошка, часова шкала	Scrum, Kanban
Спільна робота	+	+	+
Управління завданнями	+	+	+

Беручи до уваги всі аспекти, ми можемо ретельно проаналізувати та включити найкращі функції та можливості ClickUp, Asana та Jira у наш власний продукт, створивши надійне та всеохоплююче рішення для управління проектами. Такий інтегрований підхід гарантує, що наш продукт не лише задовольняє різноманітні потреби користувачів, але й використовує сильні сторони цих визнаних платформ для покращення практики управління проектами та сприяння загальному успіху організації.

Завдяки комплексному підходу наш продукт гарантує користувачам доступ до потужного та зручного інструменту, який спрощує організацію проекту, покращує співпрацю та підвищує загальну ефективність проекту.

1.3 Розробка архітектури веб-системи планування робочого часу

Архітектура системи управління проектами розроблена таким чином, щоб забезпечити масштабоване, надійне та підтримуване рішення. Архітектура побудована за шаблоном багаторівневої структури, що забезпечує розподіл завдань та сприяє повторному використанню коду.

Презентаційний рівень системи відповідає за обробку взаємодії з користувачем та візуалізація користувацького інтерфейсу. Він використовує Symfony, що слідує архітектурному шаблону MVC (рис.1.4) та шаблонізатор Twig. Рівень представлення відіграє вирішальну роль у забезпеченні зручного та зрозумілого інтерфейсу системи управління проектами. Він отримує вхідні дані, обробляє їх і представляє відповідну інформацію користувачам.

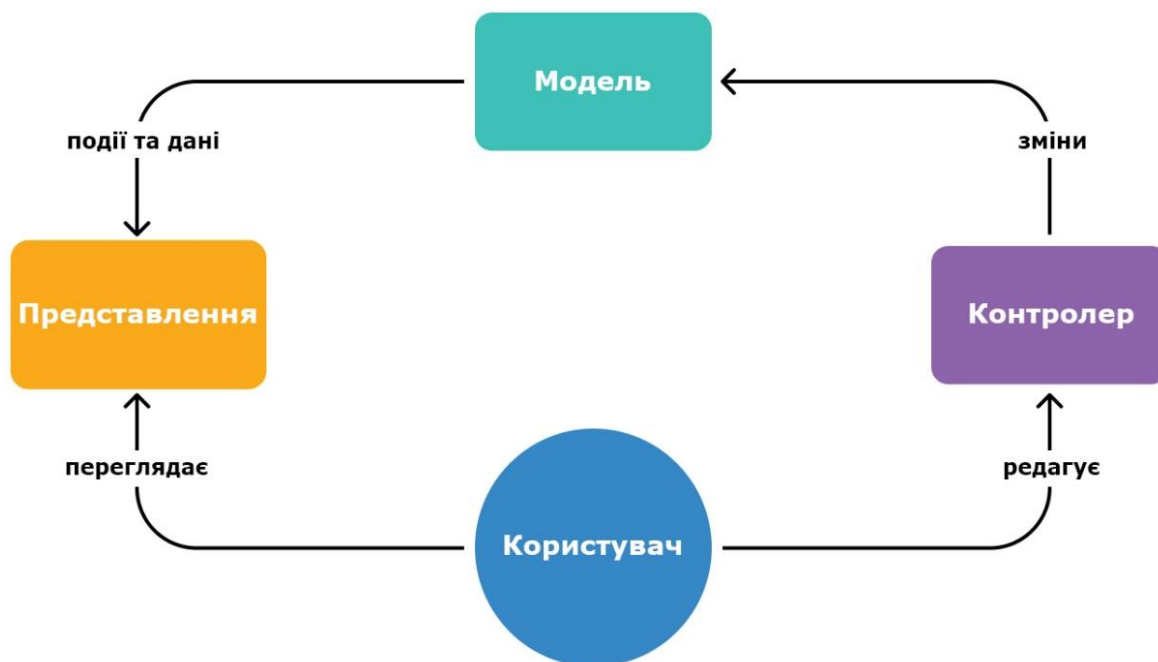


Рисунок 1.4 – представлення шаблону MVC

Рівень бізнес-логіки відповідає за обробку запитів користувачів, реалізацію бізнес-правил та координацію потоку даних між компонентами. Екосистема компонентів Symfony використовується для вирішення завдань, таких як аутентифікація та авторизація користувачів за допомогою RBAC [10].

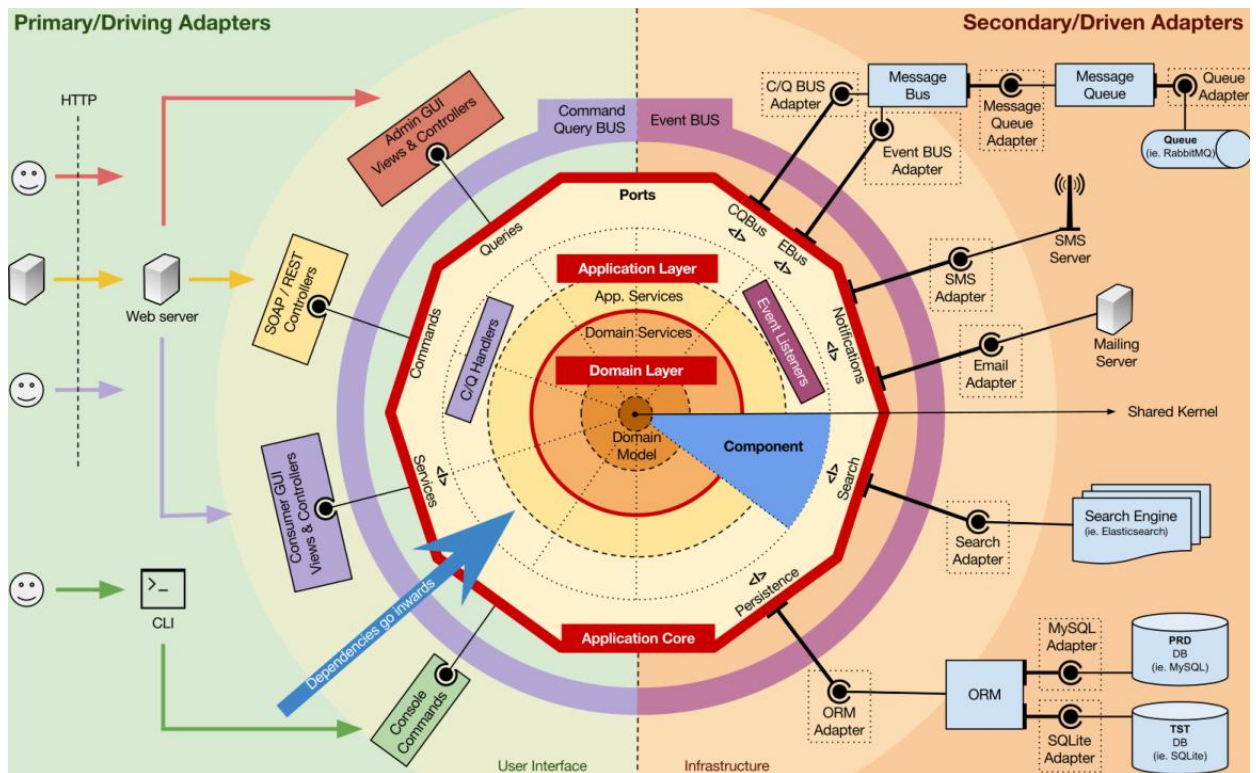


Рисунок 1.5 – представлення шаблону DDD

Також було використано принципи DDD (рис.1.5). Дана концепція передбачає розбиття додатку на різні домени і проектування кожного домену як окремого модуля. Кожен домен представляє певну сферу знань або функціональність у додатку [13].

Рівень доступу відповідає за управління зберіганням і пошуком даних у додатку. Він знаходиться між рівнем бізнес-логіки та системою зберігання даних (рис.1.6). В якості системи управління базами даних обрано PostgreSQL, яка пропонує масштабованість та підтримку повнотекстового пошуку.

Інтеграційний рівень фокусується на інтеграції зовнішніх сервісів і компонентів в систему управління проектами. RabbitMQ використовується як брокер повідомлень і система керування чергами для виконання фонових завдань і асинхронної обробки. OAuth2 використовується для автентифікації API. Centrifugo використовується для надсилання повідомлень через WebSocket, що забезпечує оновлення в режимі реального часу.

Інфраструктурний рівень охоплює вимоги до розгортання та інфраструктури системи. Docker використовується для забезпечення

узгодженого та відтворюваного середовища розробки, а також для створення образів для постановочних та виробничих середовищ.

Такий підхід спрощує процес розгортання та покращує масштабованість. Файли користувачів вивантажуються у файлові сховища, такі як хмарні сервіси зберігання даних, щоб підвищити продуктивність і забезпечити ефективну обробку завантажень і вивантажень файлів [8].

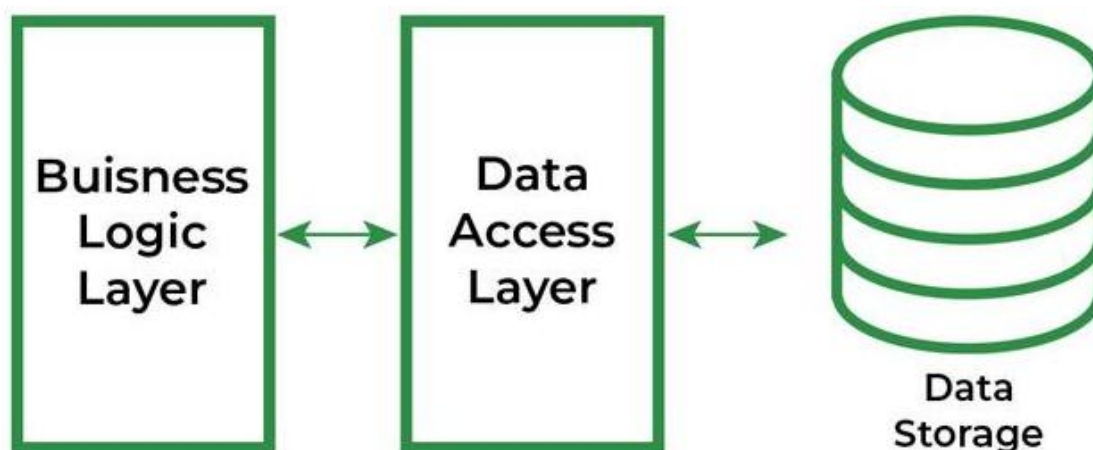


Рисунок 1.6 – Роль рівня доступу в архітектурі

Модульне та інтеграційне тестування забезпечує надійність та стабільність системи. Для автоматизації процедур тестування та перевірки функціональності системи використані фреймворки тестування, такі як PHPUnit.

Таким чином, обрана архітектура системи управління проектами не тільки забезпечує міцний фундамент, але й створює основу для розробки багатофункціонального, масштабованого та підтримуваного рішення. Дотримуючись найкращих практик, використовуючи шаблони проектування та сучасні технології, ця архітектура забезпечує ефективність, продуктивність та розширюваність системи, таким чином задовольняючи потреби користувачів, що змінюються, та створюючи умови для майбутнього зростання [2].

Такий підхід сприяє повторному використанню коду, полегшуючи додавання нових функцій, виправлення помилок та адаптацію до мінливих вимог. Крім того, така архітектура полегшує ефективну співпрацю між командами розробників.

1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення

Розробка системи управління проектами вимагає ретельного розгляду інструментів, вимог до апаратного забезпечення та архітектури системи. Цей розділ містить стислий порівняльний опис архітектурних патернів та огляд обраної архітектури, яка реалізована в додатку.

Система управління проектами розроблена з використанням **PHP** як основної мови бекенда. PHP - це широко розповсюджена і зріла мова програмування, яка пропонує багатий набір функцій і фреймворків для розробки веб-додатків. Його сумісність з фреймворком Symfony забезпечує безперешкодну інтеграцію та ефективний процес розробки [1].

Фреймворк Symfony був обраний через його широкий набір функцій та відповідність найкращим практикам. Філософія повторного використання коду, модульності та підтримки спільноти Symfony добре узгоджується з цілями проекту. Вона забезпечує надійну екосистему компонентів та інструментів, які прискорюють розробку та забезпечують високу якість коду [3].

Docker використовується для створення узгодженого та ізольованого середовища на різних машинах. Такий підхід гарантує, що всі розробники мають однакові залежності та конфігурації, що призводить до зменшення кількості проблем сумісності та покращення співпраці [12].

Doctrine ORM використовується для полегшення відображення доменних сутностей та агрегатів. Цей інструмент об'єктно-реляційного відображення забезпечує безшовну інтеграцію з фреймворком Symfony і дозволяє ефективно взаємодіяти з базами даних, дотримуючись передових практик у моделюванні та збереженні даних.

RabbitMQ використовується як черга повідомлень і брокерська система для обробки фонових завдань і асинхронної обробки. Ця технологія дозволяє ефективно планувати та розподіляти завдання, покращуючи загальну продуктивність системи та швидкість реагування [7].

Centrifugo, сервер обміну повідомленнями в режимі реального часу, інтегрована для надання сповіщень на основі WebSocket. Ця технологія дозволяє користувачам отримувати миттєві оновлення в режимі реального часу, покращуючи можливості співпраці та комунікації в системі.

Движок шаблонів **Twig** використовується для створення динамічних представлень і відображення HTML-шаблонів. Twig спрощує процес створення багаторазових і модульних шаблонів, посилюючи розподіл завдань і покращуючи ремонтпридатність коду.

Для покращення пошукової функціональності система використовує **PostgreSQL**, потужну та багатфункціональну систему управління реляційними базами даних, відому своїми розширеними можливостями пошуку та механізмами індексування. Завдяки PostgreSQL користувачі можуть скористатися алгоритмами пошуку та ефективними методами індексування, що забезпечують швидке і точне отримання результатів пошуку. Надійні функції сприяють покращенню користувацького досвіду [9].

Система управління проектами включатиме мобільний **API** для безперешкодної інтеграції з іншими додатками. API розроблено за принципами RESTful, що дозволить легко використовувати його різними клієнтськими додатками.

Архітектурні діаграми та структурні схеми забезпечують візуальне представлення компонентів системи, взаємодії та потоку даних, що ще більше покращує розуміння дизайну та архітектури системи.

Процес вибору інструментів включав ретельний аналіз варіантів з урахуванням їх особливостей, сумісності, масштабованості та простоти інтеграції. Враховуючи специфічні вимоги до веб-системи управління часом, було визначено, що обрані інструменти тісно пов'язані з бажаними функціональними можливостями та цілями проекту.

Крім того, вимоги до апаратного забезпечення були визначені на основі очікуваного робочого навантаження та продуктивності.

Висновки до розділу 1

У першому розділі цієї роботи було проведено комплексний аналіз використання інформаційних технологій для планування та обліку робочого часу на підприємстві. Розділ розпочався з чіткої постановки задачі, виокремлюючи мету розробки веб-системи для ефективного управління робочим часом.

Далі було проведено глибокий аналіз аналогів програмного продукту, зосередившись на трьох відомих інструментах управління проектами: ClickUp, Asana та Jira. Цей аналіз висвітлив особливості, функціональність, переваги та недоліки кожного інструменту, що дозволило краще зрозуміти існуючі рішення на ринку, щоб розробити максимально зручний і корисний продукт.

Крім того, розділ заглибився у важливий аспект вибору архітектури для веб-системи планування часу. Були розглянуті різні технології, такі як Symfony, Docker, RabbitMQ, PostgreSQL та інші. Обґрунтування вибору цих інструментів та вимог до апаратного забезпечення базувалося на їх сумісності, масштабованості, продуктивності та кращих галузевих практиках.

Було описано обрану архітектуру системи з акцентом на застосуванні підходу DDD, побудові багатих моделей домену та використанні компонентів Symfony. Схема архітектури забезпечила візуальне представлення рівня презентації, рівня бізнес-логіки та рівня доступу до даних, демонструючи компоненти та їхню взаємодію.

В цілому, цей розділ заклав основу для розробки інноваційної та ефективної веб-системи планування часу. Аналіз аналогів і ретельний вибір інструментів та архітектури проклали шлях до наступних розділів, де детально розглянуто проектування та реалізацію системи. Результати цього розділу зроблять значний внесок у вирішення проблем управління проектами та організаційних питань, пов'язаних з робочим часом на підприємстві.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						20

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ПЛАНУВАННЯ РОБОЧОГО ЧАСУ ПІДПРИЄМСТВА

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури

У цьому розділі представлено аналіз вимог до програмного продукту з акцентом на виявленні функціональних та нефункціональних вимог. Крім того, визначено вимоги до часових характеристик реакції системи на дії користувача, а також окреслено різні варіанти використання системи. Для забезпечення чіткого розуміння структури системи також включено узагальнену діаграму елементів (сутностей системи), їх структури та методів з описом.

Аналіз вимог передбачає всебічне вивчення потреб та очікувань різних користувачів. За допомогою інтерв'ю з користувачами, опитувань та зворотного зв'язку були визначені функціональні вимоги, які окреслюють специфічні особливості та можливості, якими повинен володіти програмний продукт. На додаток були зібрані нефункціональні вимоги, які стосуються таких аспектів, як продуктивність, безпека, надійність і зручність використання.

Функціональні вимоги:

- аутентифікація користувачів: надійні механізми автентифікації користувачів, доступ за допомогою унікальних облікових даних
- створення та управління проектами: можливість призначати завдання, встановлювати кінцевий термін, відстежувати прогрес
- система повинна дозволяти користувачам призначати завдання, полегшувати комунікацію та підтримувати спільну роботу
- можливість завантажувати, зберігати та керувати документами, забезпечуючи легкий доступ та контроль версій
- комплексні можливості звітності, генерація, аналіз проектних даних
- інтеграція зі сторонніми інструментами: продукт повинен легко інтегруватися з іншими загальноновживаними інструментами, такими як календарі, поштові клієнти та комунікаційні платформи

Нефункціональні вимоги:

- система повинна швидко реагувати на дії користувача, забезпечуючи безперебійну та чуйну роботу
- час, необхідний для завантаження сторінок, виконання операцій повинен відповідати прийнятним критеріям продуктивності
- надійні заходи безпеки для захисту даних проекту, забезпечуючи надійну автентифікацію, шифрування даних та контроль доступу
- система повинна бути надійною та доступною для використання, мати механізми резервного копіювання та відновлення
- інтерфейс користувача повинен бути зрозумілим і зручним, щоб користувачі з різним рівнем технічної підготовки могли легко орієнтуватися в системі та ефективно виконувати завдання
- масштабованість: система повинна бути масштабованою, щоб пристосовуватися до зростаючої кількості користувачів, проектів та обсягів даних без шкоди для продуктивності
- сумісність: програмний продукт повинен бути сумісним з різними веб-браузерами та операційними системами, забезпечуючи доступність для користувачів на різних платформах

Часові характеристики реакції системи:

- швидкість реакції системи: майже миттєвий час відгуку на звичайні дії користувача (завантаження сторінок, оновлення завдань та завантаження документів)
- обробка завдань: система повинна ефективно обробляти та оновлювати інформацію, гарантуючи, що зміни оперативно відображаються в системі
- генерація звітів: продукт повинен генерувати звіти в прийнятні терміни, враховуючи складність даних і розмір проекту

За допомогою тестування продуктивності та бенчмаркінгу визначаються прийнятні часові межі для різних дій та процесів у системі. Ці часові

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						22

характеристики є важливими для задоволення очікувань користувачів та уникнення розчарування, спричиненого затримками або відсутністю реакції.

Варіанти використання. Щоб задовольнити різноманітні потреби користувачів, система пропонує різні варіанти використання. Забезпечуючи гнучкість у використанні системи, користувачі можуть адаптувати програмний продукт до своїх конкретних вимог і робочих процесів, підвищуючи свій загальний досвід і продуктивність.

Ми створили діаграму варіантів використання (рис.2.1) - візуальне представлення взаємодії між користувачами системи (акторами) і самою системою та специфікацію прецедентів (дод.А).

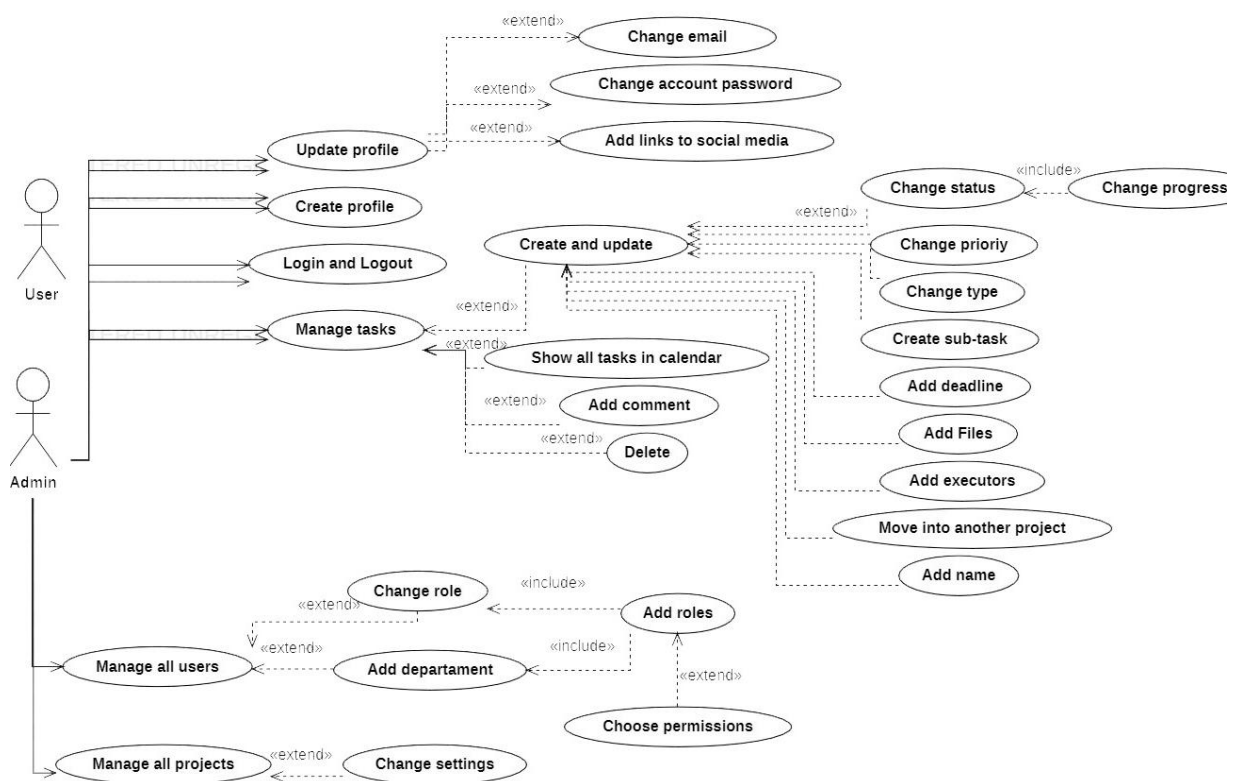


Рисунок 2.1 – Діаграма варіантів використання

Користувач:

- управління завданнями: можливість переглядати, створювати та виконувати завдання, можливість призначати завдання собі або іншим
- доступ до календаря, який відображає кінцеві термін та важливу інформацію про завдання

- комунікація завдань: спілкування з клієнтами, колегами або іншими зацікавленими сторонами, які беруть участь у конкретному проекті
- управління клієнтами та проектами: доступ до повного списку клієнтів або проектів, а також до відповідної інформації, такої як контактні дані, історія спілкування та завдання, пов'язані з проектом
- інтеграції: доступ до інтеграції із зовнішніми інструментами
- ролі користувачів: система реалізує механізми авторизації та автентифікації, щоб гарантувати, що співробітники мають доступ лише до тих функцій та інформації, які відповідають їхнім ролям
- кожен користувач матиме персоналізовану інформаційну панель, яка відображатиме відповідну інформацію, завдання та нагадування, вона слугує централізованим центром для користувачів, що дозволяє їм мати актуальну інформацію про свою роботу та пріоритети

Адміністратор:

- повноваження додавати, редагувати та видаляти облікові записи користувачів, призначати ролі та керувати правами
- можливість керувати даними в системі, включаючи інформацію про клієнтів, деталі проекту, терміни та інші важливі дані
- управління конфігурацією: можливість налаштувати параметри системи, щоб персоналізувати зовнішній вигляд системи, узгодивши його з конкретними вимогами організації
- управління інтеграцією: управління інтеграцією з іншими системами, забезпечуючи безперебійну взаємодію з інструментами
- можливість створювати звіти на основі даних, що зберігаються в системі, що надаватимуть цінну інформацію про активність клієнтів і проектів, рівень виконання завдань та інші показники ефективності
- адміністратор відповідатиме за постійне обслуговування системи, включаючи усунення несправностей, виконання оновлень та вирішення будь-яких технічних питань, що можуть виникнути

Завдяки рольовому розподілу функцій і привілеїв доступу, система гарантує, що кожен користувач може ефективно виконувати свої завдання та співпрацювати в рамках єдиної платформи. Визначені функціональні можливості для користувачів і адміністраторів створюють комплексне рішення, яке задовольняє різноманітні потреби клієнтів.

Загалом, діаграма варіантів використання забезпечує комплексне і наочне представлення функціональних можливостей. Визначаючи та документуючи варіанти використання, діаграма слугує цінним інструментом для аналізу системи, збору вимог та проектування.

Об'єктно-орієнтована структура. Структура дозволяє модульний дизайн системи, де різні елементи представлені як об'єкти з визначеними атрибутами та методами. Завдяки інкапсуляції функціональності в об'єкти, система сприяє повторному використанню, модульності та розширюваності.

Наведено узагальнену діаграму об'єктно-орієнтованої структури системи (рис.2.2), яка ілюструє взаємозв'язки та взаємодію між різними об'єктами системи. Кожна сутність представляє певний компонент або модуль програмного продукту, з власними атрибутами та методами. Діаграма слугує візуальним представленням архітектури системи, сприяючи кращому розумінню її загальної структури.

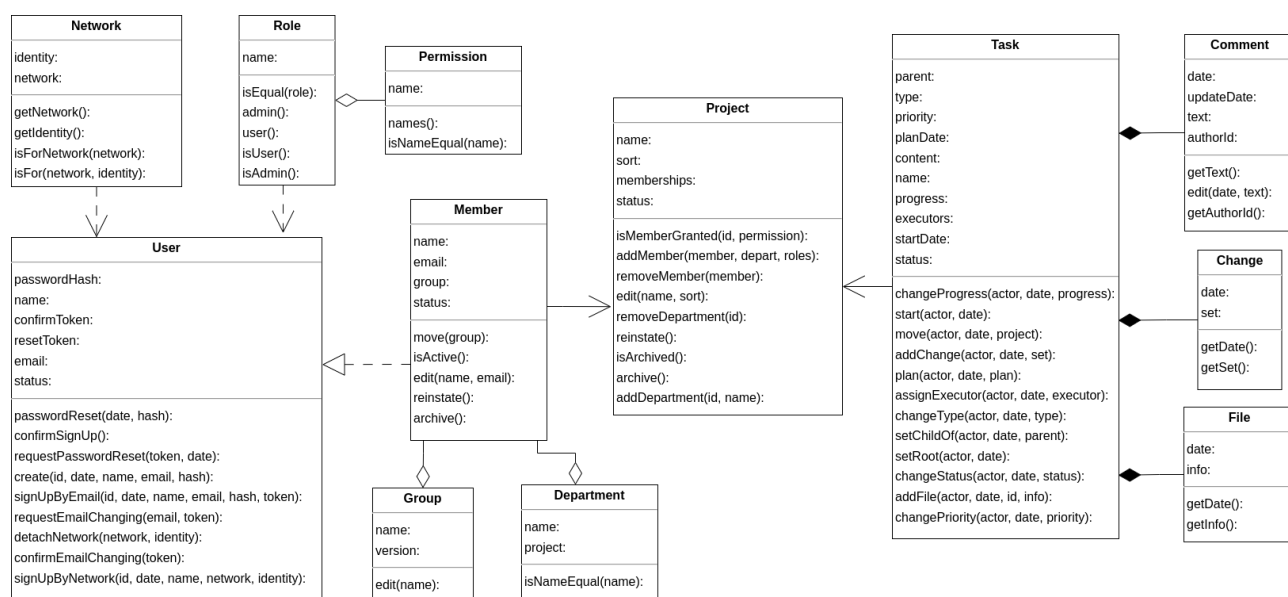


Рисунок 2.2 – Об'єктно-орієнтована структура

Клас User інкапсулює атрибути та функціональність, пов'язані з обліковим записом користувача, включаючи електронну пошту, хеш-пароль, ім'я та статус. Цей клас відповідає за керування користувачами, надаючи такі функції, як реєстрація, автентифікацію та авторизація, редагування даних, керування статусом і скидання пароля і підтвердження електронною поштою.

Клас Network відповідає за керування посиланнями на соціальні мережі користувача, назвами мереж та ідентифікаторами користувачів.

Клас Role відповідає за визначення різних ролей, які користувачі можуть мати у системі (адміністратор та користувач), надає спосіб призначати ролі і керувати ними, дозволяючи визначати привілеї користувачів у системі.

Клас Member використовується для керування учасниками, виконання таких дій, як створення нових користувачів, від'єднання від мереж, оновлення інформації, а також для виконання специфічних для користувача завдань і дій.

Клас Permission відповідає за визначення та управління різними дозволами, які можуть бути надані користувачам в системі. Дозволи можуть бути пов'язані з конкретними діями або операціями, що дозволяє здійснювати контроль доступу і гарантувати, що користувачі мають необхідні привілеї.

Клас Task інкапсулює атрибути і функціональність, пов'язані з конкретним завданням, включаючи його назву і мережу, до якої воно належить. Цей клас дозволяє створювати, керувати та відстежувати завдання в системі. Він уможливорює такі операції, як ідентифікація завдань, призначення завдань, відстеження прогресу та визначення пріоритетів завдань.

Клас Change відповідає за управління та відстеження змін у завданні, додавання та редагування змін, відстеження статусу змін, а також управління файлами та діями, пов'язаними зі змінами.

Клас Project інкапсулює атрибути та функціональність, пов'язані з конкретним проектом, такі як його назва, учасники та статус. Він призначений для полегшення управління проектами, дозволяючи створювати, відстежувати та координувати проекти. Він дозволяє виконувати такі операції, як додавання і

видалення учасників проекту, редагування деталей проекту, відстеження прогресу проекту, а також керування файлами і коментарями.

Клас Group використовується для організації та управління користувачами в межах певних груп або категорій. Він надає функціональність для створення та керування групами, дозволяючи виконувати такі операції, як додавання та видалення, редагування деталей групи та сортування груп.

Клас Department призначений для полегшення управління відділами, дозволяючи створювати, редагувати та організовувати відділи. Він дозволяє виконувати такі операції, як призначення відповідальних за відділ, керування деталями відділу, а також обробку специфічних для відділу завдань.

Клас Comment дозволяє виконувати такі операції, як отримання деталей коментаря та керування діями, пов'язаними з коментарем, наприклад, отримання дати та інформації про коментар.

Клас File забезпечує операції для отримання даних про файл, керування діями, пов'язаними з файлом, і доступу до інформації про файл, такої як дата файлу та додаткова інформація.

UML-діаграма представляє об'єктно-орієнтовану структуру системи управління часом, забезпечуючи візуальне представлення компонентів системи та їх взаємозв'язків. Використовуючи діаграму UML, розробники програмного забезпечення можуть краще зрозуміти структуру і дизайн системи, гарантуючи, що вона відповідає визначеним вимогам [11].

Крім того, дана структура дозволяє створювати модульний і багаторазовий код, що сприяє гнучкості і простоті обслуговування. Інкапсуляція атрибутів і функцій у кожному класі сприяє організації коду і підвищує загальну функціональність системи.

Діаграма UML слугує основою для подальших етапів розробки та впровадження системи. Вона орієнтує розробників на реалізацію визначених варіантів використання, управління взаємозв'язками між класами та включення додаткових функцій або модифікацій у майбутньому.

2.2 Розробка структури бази даних системи

Ефективна база даних має важливе значення для безперебійної роботи системи, забезпечуючи цілісність даних, масштабованість та оптимальну продуктивність. Першим кроком у розробці є створення відповідної схеми, яка точно відображає сутності, зв'язки та атрибути системи. Процес проектування включає в себе аналіз вимог і перетворення їх в логічну модель бази даних.

Для візуалізації схеми бази даних і зв'язків між сутностями створена діаграма "сутність-зв'язок" (ERD). Вона забезпечує графічне представлення структури бази даних, ілюструючи сутності, їх атрибути та зв'язки між ними. Вона слугує планом для фактичної реалізації бази даних.

Для оптимізації продуктивності системи використане індексування стовпців, до яких часто звертаються, оптимізація планів виконання запитів та впровадження механізмів кешування для зменшення кількості обходів.

Перша схема надає огляд усіх таблиць, пов'язаних з сутністю користувача (рис.2.3). Ці таблиці призначені для збору та зберігання специфічної інформації, такої як профілі, облікові дані для автентифікації та дозволи контролю доступу. Організуючи і пов'язуючи ці таблиці, схема створює основу для ефективного управління даними, пов'язаними з користувачами.

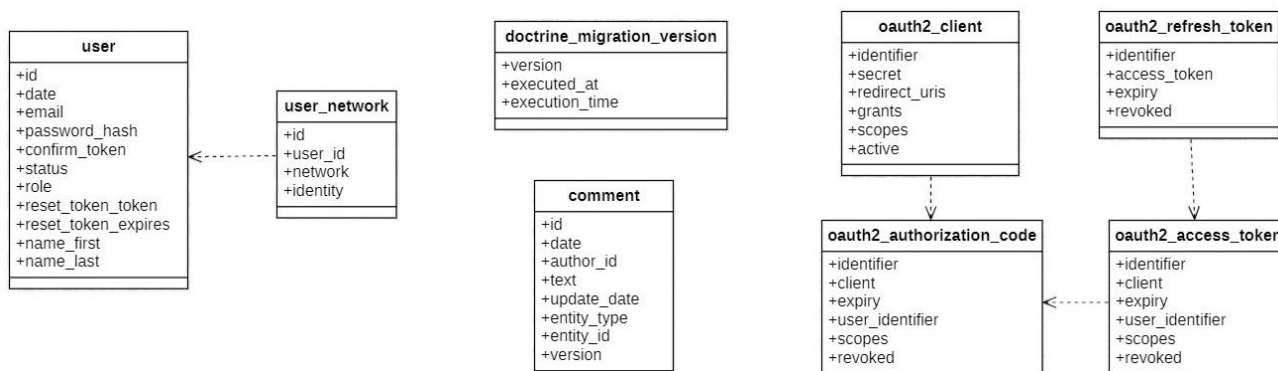


Рисунок 2.3 – Таблиці пов'язані з сутністю користувача

Друга схема фокусується на таблицях, пов'язаних із завданнями в програмній системі (рис.2.4). Ці таблиці відповідають за фіксацію деталей завдань, таких як призначення завдань, терміни, статус виконання та залежності. Визначаючи структуру і взаємозв'язки між цими таблицями, схема

дозволяє ефективно управляти і відстежувати завдання протягом усього їхнього життєвого циклу.

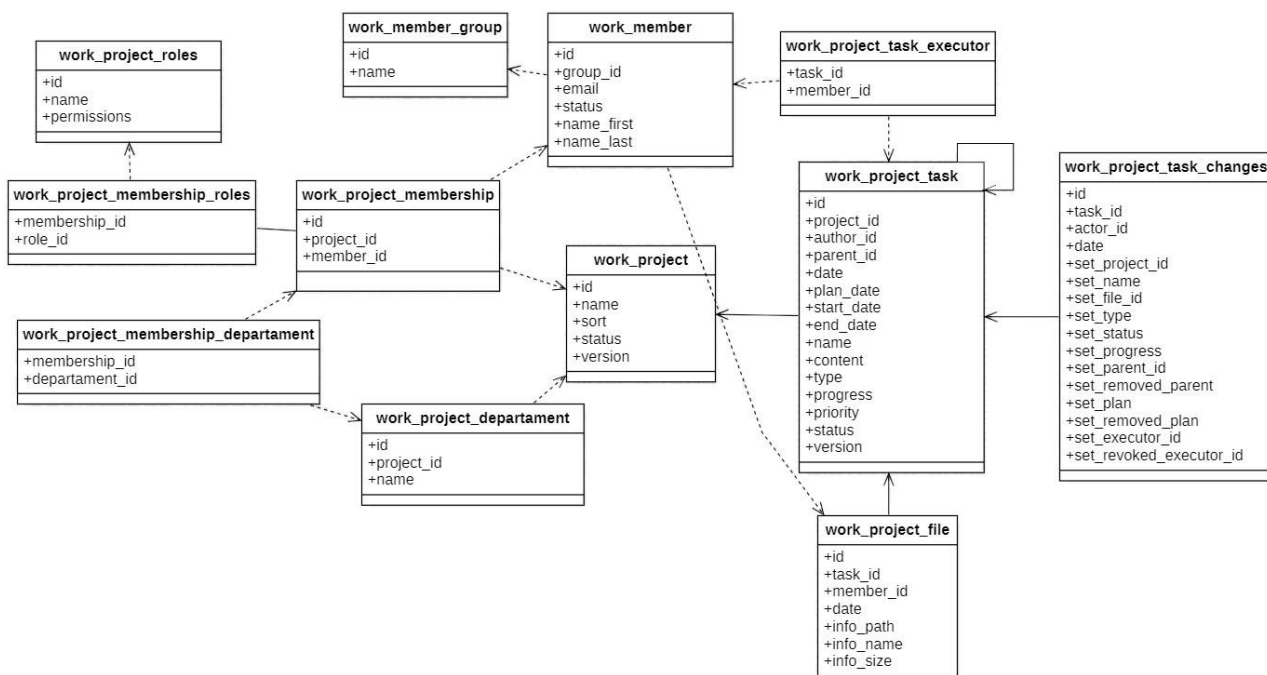


Рисунок 2.4 – Таблиці пов'язані з сутністю завдання

Разом ці схеми сприяють загальній об'єктно-орієнтованій структурі системи, забезпечуючи чітке представлення залучених сутностей та їхніх взаємозв'язків. Дотримуючись такого структурованого підходу, програмний продукт може ефективно управляти даними, пов'язаними з користувачами, та інформацією, що стосується конкретних завдань, сприяючи безперешкодній взаємодії та співпраці в рамках системи управління проектами.

user: використовується для зберігання інформації про користувачів системи. Містить дату реєстрації, адресу електронної пошти, хешований пароль, токен підтвердження реєстрації, статус облікового запису, роль або привілеї, токен для скидання пароля, термін дії токена, ім'я та прізвище.

user_network: використовується для зберігання посилань на соціальні мережі користувача, назву мережі та ідентифікатор користувача.

comment: використовується для зберігання коментарів, зроблених користувачами до сутностей у системі. Містить дату створення, зовнішнє посилання на користувача, зміст коментаря, дата останнього оновлення коментаря, тип сутності, з якою пов'язаний коментар та версія коментаря.

Опис полів таблиці «user»

Назва	Тип	PK	FK	Опис
id	int	✓		Первинний ключ
date	date			Дата користувача
email	varchar			Адреса електронної пошти користувача
password_hash	varchar			Хешований пароль
confirm_token	varchar			Токен підтвердження
status	varchar			Статус користувача
role	varchar			Роль користувача
reset_token	varchar			Токен скидання
reset_token_expires	date			Дата закінчення терміну дії токена
name_first	varchar			Ім'я користувача
name_last	varchar			Прізвище користувача

oauth2_client: зберігання інформації про OAuth2 клієнтів в системі. Містить секретний ключ, URI перенаправлення після успішної авторизації, дозволені гранти, дозволи, активний статус.

oauth2_access_token: зберігання інформації про токени доступу, згенеровані для клієнтів OAuth2. Містить асоційованого клієнта, термін дії токена, області дії або дозволи, надані токену, а також статус відкликання, що вказує на те, чи був токен відкликаний.

oauth2_refresh_token: зберігання інформації про токени оновлення, пов'язані з клієнтами OAuth2. Містить пов'язаний з ним токен доступу, термін дії токена та статус відкликання.

oauth2_authorization_code: зберігання інформації про коди авторизації. Містить асоційованого клієнта, час закінчення терміну дії коду, області дії або дозволи, надані кодом, і статус відкликання.

doctrine_migration_version: використовується для відстеження версій міграцій баз даних, виконаних інструментом міграції Doctrine. Містить інформацію про кожну версію міграції, таку як номер версії, дата і час виконання міграції, а також час, витрачений на виконання міграції.

work_project_roles: зберігання інформації про ролі в робочому проекті. Містить дані, пов'язані з кожною роллю проекту, включно з унікальним ідентифікатором, назвою ролі та дозволами, пов'язаними з цією роллю.

work_project_membership_roles: встановлює зв'язок між учасниками проекту та ролями проекту.

work_project_membership: використовується для зберігання інформації про учасника проекту в робочому проекті. Містить дані, пов'язані з кожною участю у проекті, включаючи посилання на зовнішній ключ до пов'язаного проекту та посилання на зовнішній ключ до пов'язаного учасника.

work_project_membership_department: використовується для зв'язування учасників проекту з відділами проекту.

work_project_department: використовується для зберігання інформації про відділи в межах робочого проекту.

work_project: містить дані, пов'язані з кожним робочим проектом, такі як назва проекту, порядок сортування, статус проекту і версія проекту.

Таблиця 2.2

Опис полів таблиці «work_project»

Назва	Тип	PK	FK	Опис
id	int	✓		Унікальний ідентифікатор +проекту
name	varchar			Назва проекту
sort	int			Порядок сортування проекту
status	varchar			Статус проекту
version	int			Версія проекту

work_project_task: використовується для зберігання інформації про завдання в межах робочого проекту. Містить зовнішнє ключове посилання на пов'язаний проект, на користувача, який створив завдання, батьківське завдання (якщо застосовано), дати створення, запланований початок і закінчення, ім'я або назву завдання, зміст або опис, тип завдання, статус виконання, рівень пріоритету, статус завдання і версію завдання.

Опис полів таблиці «work_project_task»

Назва	Тип	PK	FK	Опис
id	int	✓		Унікальний ідентифікатор завдання
project_id	int		✓	Зовнішній ключ (work_project)
author_id	int		✓	Зовнішній ключ (work_member)
parent_id	int		✓	Зовнішній ключ (work_project_task)
date	date			Дата створення завдання
plan_date	date			Запланована дата виконання завдання
start_date	date			Дата початку завдання
end_date	date			Дата завершення завдання
name	varchar			Ім'я або назва завдання
content	text			Зміст або опис завдання
type	varchar			Тип завдання
progress	float			Статус виконання завдання
priority	int			Рівень пріоритетності завдання
status	varchar			Статус завдання
version	int			Версія завдання

work_project_task_changes: відстеження змін, внесених до завдань.

work_member_group: зберігання інформації про групи учасників у робочому проекті.

work_member: використовується для зберігання інформації про учасників у робочому проекті. Вона містить зовнішнє посилання на пов'язану групу, адресу електронної пошти учасника, статус, а також ім'я та прізвище.

work_project_file: містить дані, пов'язані з файлом проекту, включаючи зовнішнє посилання на пов'язане завдання та учасника, дату додавання файлу, шлях або місце розташування файлу, ім'я файлу та розмір файлу.

work_project_task_executor: використовується для призначення учасників як виконавців завдань проекту. Вона встановлює зв'язок між завданнями проекту та учасниками, зберігаючи ідентифікатор завдання та ідентифікатор учасника, пов'язаний з кожним призначенням виконавця.

Після створення бази даних на основі визначених схем ми використали DBeaver, популярний інструмент управління базами даних, для перевірки структури. Підключивши його, ми змогли перевірити, що таблиці та пов'язані з ними атрибути були правильно реалізовані відповідно до заданих схем. Це включало підтвердження типів даних, довжини та зв'язків між таблицями.

Інтуїтивно зрозумілий інтерфейс DBeaver забезпечив візуальне представлення структури бази даних, що дозволило нам легко орієнтуватися і досліджувати взаємозв'язки між різними сутностями.

Використовуючи можливості DBeaver, ми отримали впевненість у точності та повноті структури бази даних, переконавшись, що вона відповідає нашому початковому дизайну та вимогам програмного продукту, що розробляється.

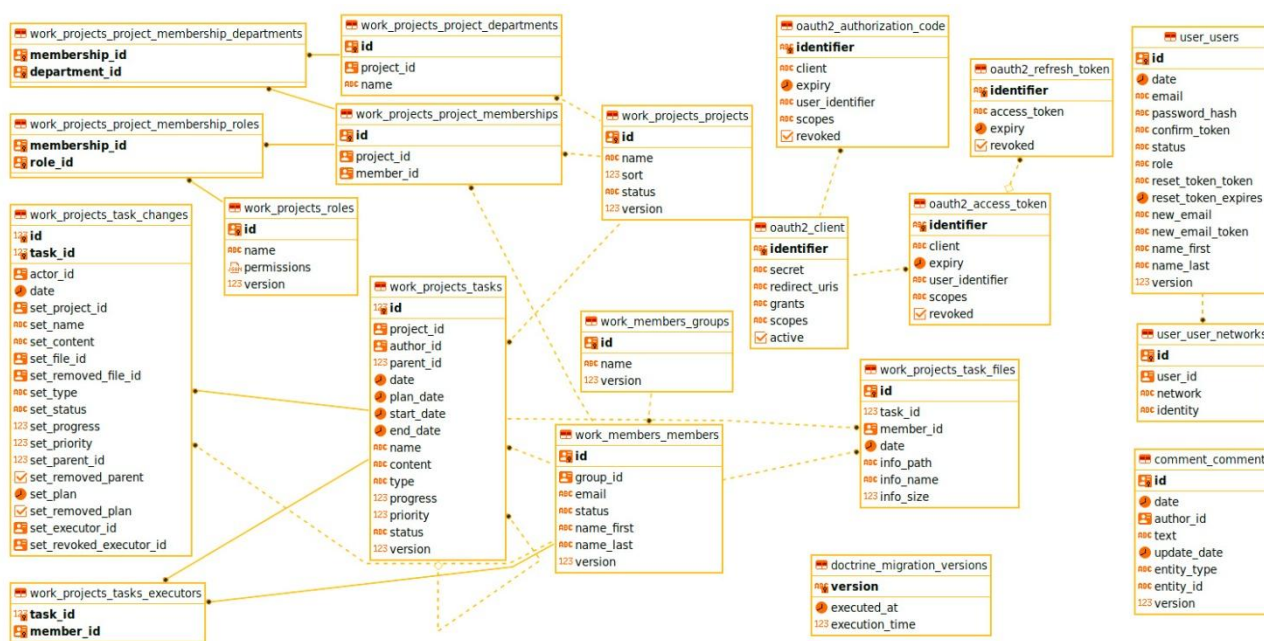


Рисунок 2.5 – Об'єктно-орієнтована структура

Зосередившись на розробці бази даних системи, ми можемо забезпечити ефективне управління даними і створити надійну основу для загальної функціональності системи. Добре спроектована та оптимізована база даних відіграє вирішальну роль в успіху системи, забезпечуючи ефективне зберігання, пошук та маніпулювання даними.

2.3 Проектування та реалізація алгоритмів роботи системи

Розробка та реалізація системних алгоритмів відповідають за виконання різних обчислювальних завдань, обробку даних і прийняття рішень в системі. Ми створили декілька діаграм діяльності для візуалізації та представлення потоку діяльності в системі. Вони пропонують чіткий спосіб зображення послідовних і паралельних дій, які відбуваються під час виконання системних процесів. Створюючи діаграми діяльності, ми прагнемо забезпечити всебічне розуміння алгоритмічного дизайну системи та її реалізації.

Діаграма активності входу в систему управління проектами (рис.2.6) включає перевірку імені користувача та пароля. Після успішного входу адміністратор отримує доступ до різних функцій і операцій, пов'язаних з управлінням проектами, завданнями, помилками і співробітниками

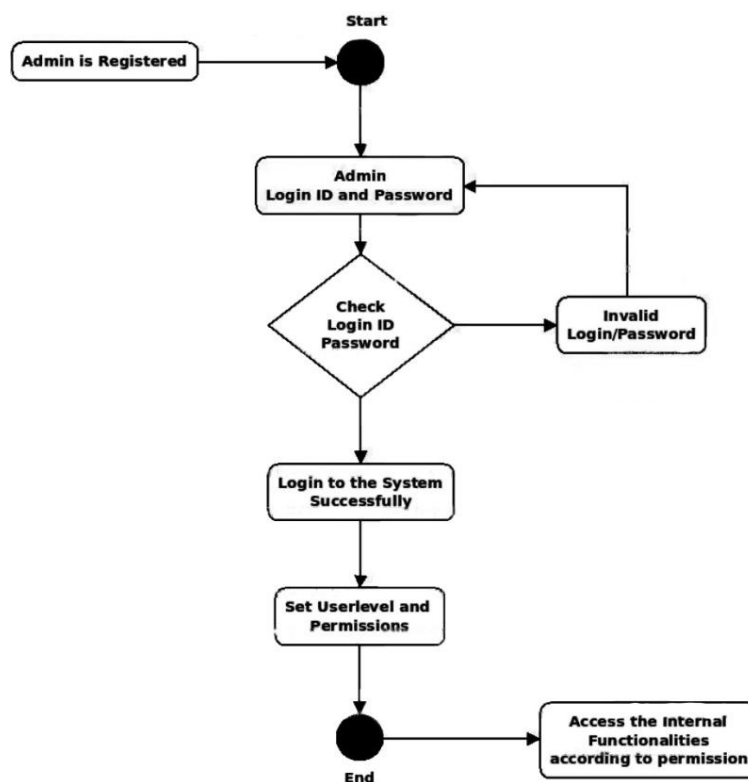


Рисунок 2.6 – Діаграма активності входу в систему

Діаграма підкреслює безпечний характер системи, оскільки аутентифікація користувача необхідна для доступу до конфіденційних сторінок. Без підтвердження особи користувачі не зможуть отримати доступ до них.

На діаграмі діяльності керування профілем (рис.2.7) зображено послідовність кроків, пов'язаних з керуванням профілем користувача в системі. Після входу система переходить до перевірки ролі користувача, щоб визначити його рівень доступу та дозволів.

Після перевірки ролі користувачеві надається можливість оновити ім'я, адресу електронної пошти, існуючий пароль, щоб забезпечити надійний і безпечний обліковий запис. На додаток до основних даних профілю, діаграма відображає можливість для користувачів додавати посилання на соціальні мережі до свого профілю.

Загалом, діяльність з управління профілем має на меті надати користувачам інструменти та гнучкість у налаштуванні та управлінні своїм профілем у системі управління часом, забезпечуючи точність, безпеку та персоналізований досвід, адаптований до їхньої ролі та уподобань.

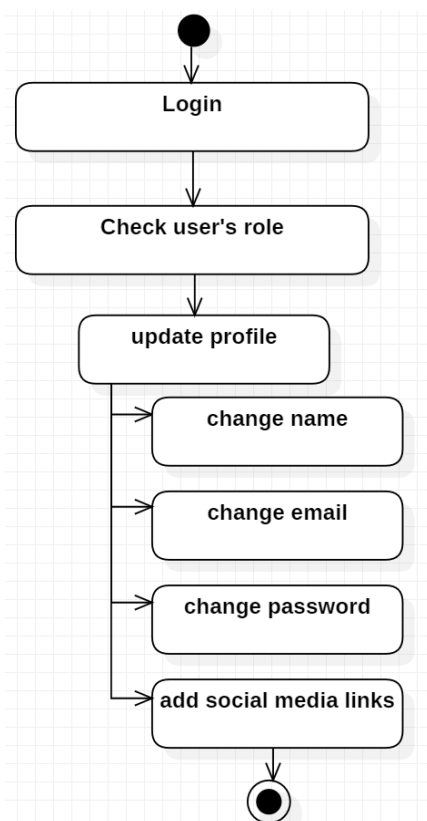


Рисунок 2.7 – Діаграма активності управління профілю

Діаграма діяльності керування завданнями (рис.2.8) описує кроки, пов'язані зі створенням, оновленням, призначенням та організацією завдань.

Діяльність починається з процесу входу в систему. Перевірка ролі користувача допомагає визначити його рівень авторизації та права доступу.

Для користувачів на діаграмі описано кілька дій з управління завданнями, які вони можуть виконувати, а саме створювати нові завдання і підзадачі, що дозволяє ієрархічно організувати роботу. Можливість призначати собі завдання дозволяє користувачам брати на себе відповідальність за виконання конкретних завдань. Додавання описів, зміна пріоритету і статусу завдань, а також встановлення кінцевих термінів. Прикріплення файлів і додавання коментарів покращує співпрацю і надає додатковий контекст завданням. Крок оновлення означає безперервний моніторинг і модифікацію завдань у разі потреби.

Для адміністраторів доступні додаткові дії. Вони мають право видаляти завдання, призначати завдання іншим користувачам і переміщати завдання між різними проектами, що дозволяє ефективно управляти завданнями і координувати роботу в системі.

Мета даної діаграми - забезпечити структурований і ефективний робочий процес для користувачів і адміністраторів для виконання різних дій, пов'язаних із завданнями, в системі управління часом.

Діаграма послідовності слугують для ілюстрації логічного потоку подій, які відбуваються під час взаємодії користувача з системою. Візуально відображаючи потік повідомлень і взаємодію між об'єктами, діаграми послідовності допомагають виявити потенційні проблемні місця і полегшують розуміння обміну інформацією.

Алгоритм взаємодії адміністратора з системою (рис.2.8). Процес починається з того, що адміністратор переходить на сторінку "Вхід" і вводить свої облікові дані. Після успішного входу система перевіряє облікові дані та надає доступ до системи.

Якщо адміністратор забув свій пароль, він може ініціювати процес "Забув пароль". Це запускає низку кроків, під час яких система перевіряє надані дані, якщо вони дійсні, система надсилає адміністратору електронного листа з

інструкціями щодо відновлення пароля. Однак, якщо дані для входу недійсні, з'являється повідомлення про помилку, яке вказує на те, що спроба не вдалася.

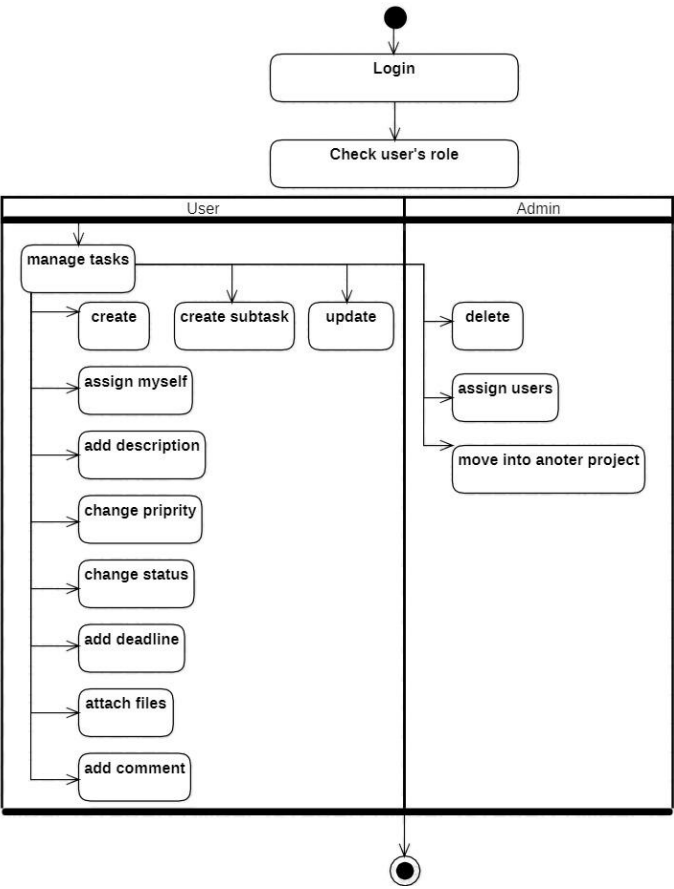


Рисунок 2.8 – Діаграма активності управління завданнями

Після того, як адміністратор увійшов в систему і його доступ підтверджено, він може перейти до таких вкладок, як проекти, користувачі та завдання. Вони дозволяють адміністратору виконувати різні операції, пов'язані з управлінням проектами, користувачами і завданнями в системі.

Зокрема, тільки адміністратор і менеджери проектів мають право додавати нових користувачів до проектів і призначати їм певні ролі. Такий контроль забезпечує належне управління.

Кожна вкладка надає основні методи для перегляду, створення, редагування та видалення записів. Ці операції дозволяють адміністратору мати повне уявлення про проекти, користувачів і завдання, а також виконувати необхідні зміни або видалення записів за потреби.

Після завершення роботи адміністратора сесія знищується разом з усіма пов'язаними з нею токенами, що забезпечує належний рівень безпеки та запобігає несанкціонованому доступу до системи.

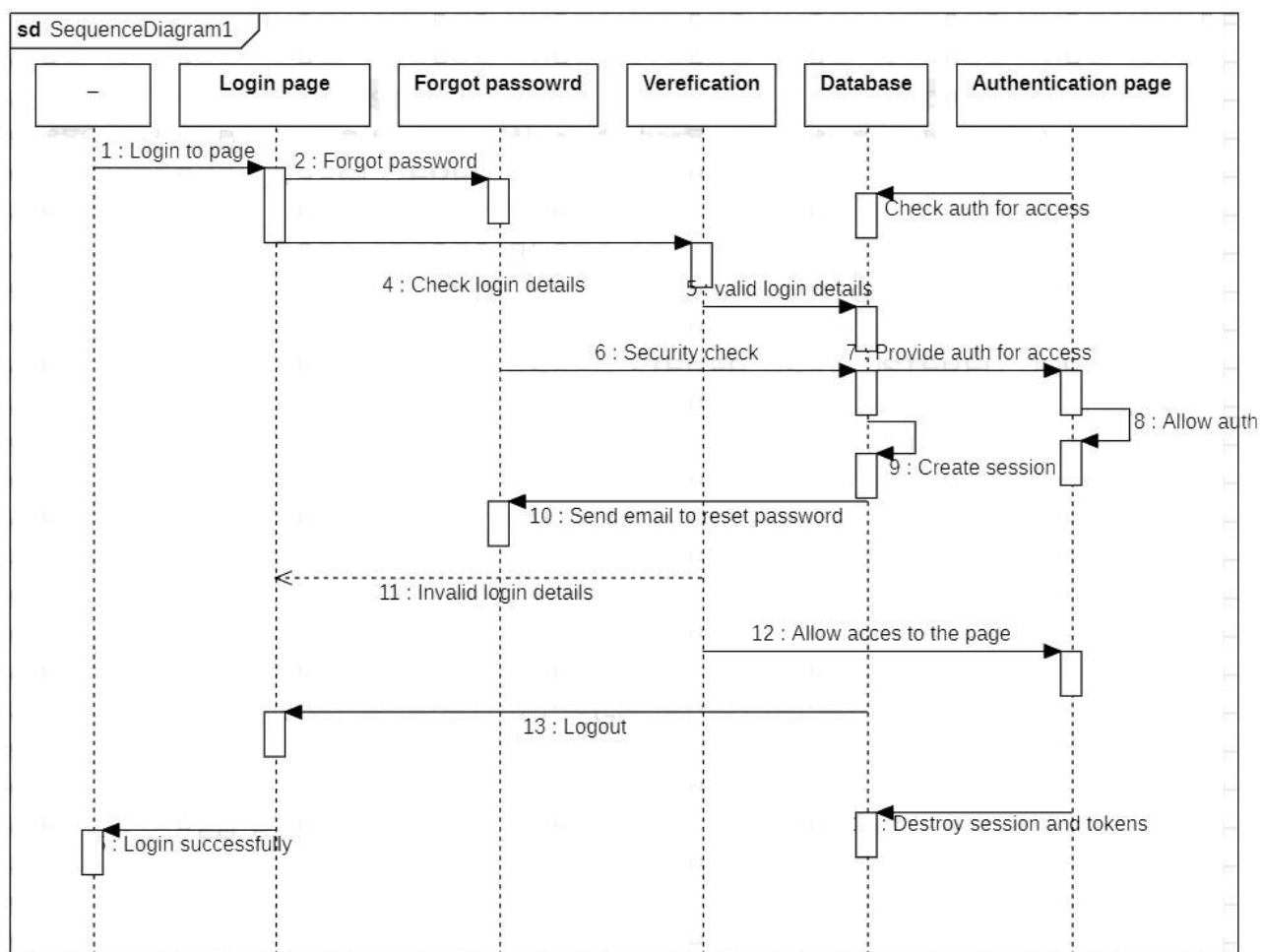


Рисунок 2.9 – Діаграма послідовності авторизації

Алгоритм роботи з проектами (рис.2.10). Процес починається з того, що користувач потрапляє на сторінку входу, де йому пропонується ввести свої облікові дані для входу.

Після успішної авторизації користувач отримує доступ до різних функціональних можливостей. У розділі "Керування проектами" користувач може виконувати такі операції, як додавання або редагування, збереження або оновлення інформації, а також додавання або видалення. Аналогічно працюють розділи "Керування учасниками", "Керування завданнями".

Протягом усього процесу система забезпечує дотримання належної цілісності даних та заходів безпеки. Авторизація користувача перевіряється на основі облікових даних, що зберігаються в базі даних, гарантуючи, що тільки авторизовані користувачі можуть отримати доступ до системи і виконувати призначені дії.

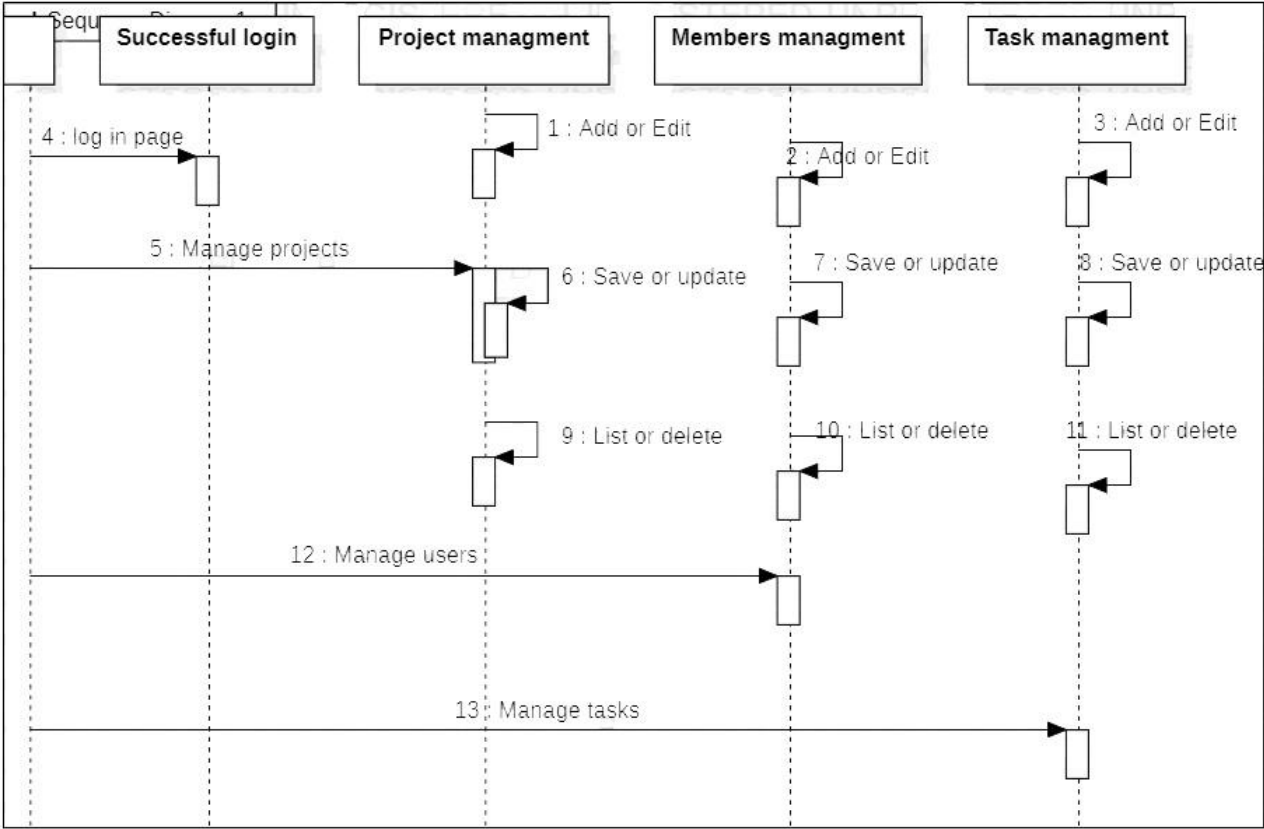


Рисунок 2.10 – Діаграма послідовності управління проектами

Загалом, діаграми послідовності пропонують стисле і наочне представлення динамічної взаємодії та потоку повідомлень у системі. Вони відіграють важливу роль у розумінні послідовності подій та обміну інформацією, дозволяючи краще аналізувати та розуміти поведінку системи.

Ефективно розробляючи та впроваджуючи ці системні алгоритми, ми можемо забезпечити безперебійну роботу системи, надаючи користувачам ефективне управління завданнями, безперебійну комунікацію, інтеграцію із зовнішніми інструментами, глибокі звіти та надійні можливості управління даними.

2.4 Реалізація функціоналу системи планування робочого часу підприємства

Впровадження веб-системи для обліку та планування робочого часу на підприємстві передбачає перетворення дизайну та функціональності системи на повнофункціональний додаток. У цьому розділі ми обговоримо етапи процесу впровадження та висвітлимо ключові аспекти розробки додатку.

В якості патерну розробки програмного забезпечення було використано DI - підхід при розробці програмного забезпечення, який дозволяє вставляти необхідні залежності в клас. Замість того, щоб клас відповідав за створення або розташування своїх залежностей, DI передбачає передачу їх у клас із зовнішнього джерела.

У проекті першим кроком до використання DI є визначення залежностей. У цьому визначенні вказуються такі деталі, як ім'я класу та будь-які аргументи конструктора. Згодом, у класі, який потребує залежностей, використовується ін'єкція конструктора, щоб прийняти залежності як аргументи.

Інкапсуляція бізнес-логіки в сервісах має кілька переваг над розміщенням логіки безпосередньо в контролерах. Використання сервісів сприяє масштабованості, ремонтпридатності та надійності програми. У проекті CRM деякі приклади служб (дод.Б), які можна використовувати замість розміщення бізнес-логіки в контролерах, включають:

1. служба управління завданнями: відповідає за операції, пов'язані із завданнями: створення, оновлення та пошук, може обробляти призначення завдань, оновлення статусів і розрахунок часу
2. служба управління клієнтами: займається управлінням інформацією про клієнтів, включаючи профілі, оновлення та пошук, може включати розрахунок рейтингів і зберігання відгуків.
3. служба управління рівнями: керує інформацією про рівні: профілі, оновлення та пошук, включає логіку управління продуктивністю: обчислення середнього часу, кількості завершених завдань

4. служба генерації звітів: створює звіти за завданнями, звіти за рівнями та інші типи звітів, може включати логіку для створення звітів у різних форматах, таких як pdf, графічні файли та діаграми

```
class ReportService
{
    private $taskRepository;
    public function __construct(TaskRepository $taskRepository)
    {
        $this->taskRepository = $taskRepository;
    }
    public function generateTaskReport()
    {
        $tasks = $this->taskRepository->findAll();
        $taskData = [];
        foreach ($tasks as $task) {
            $taskData[] = [
                'task_description' => $task->getDescription(),
                'task_status' => $task->getStatus(),
                'task_priority' => $task->getPriority(),
                'task_type' => $task->getType(),
                'task_completion_date' => $task->getCompletionDate(),
                'task_time_estimate' => $task->getTimeEstimate(),
                'task_price' => $task->getPrice(),
            ];
        }
        return $taskData;
    }
}
```

Розміщення коду бізнес-логіки в контролерах не рекомендується, оскільки це порушує принцип розподілу обов'язків. Контролери повинні в першу чергу обробляти HTTP-запити і повертати відповідні відповіді, тоді як бізнес-логіка повинна знаходитися в сервісах.

Сервіси, з іншого боку, розроблені таким чином, щоб містити бізнес-логіку, і їх можна повторно використовувати на кількох контролерах і навіть в інших програмах. Це робить код більш модульним, гнучким і його легше підтримувати, тестувати та масштабувати.

```
<?php

public function __construct(PaymentService $paymentService)
{
    $this->paymentService = $paymentService;
}
```

Змінні середовища використовуються для зберігання даних конфігурації поза кодом, що полегшує відокремлення конфіденційної інформації, такої як паролі та ключі API. Вони дозволяють використовувати різні конфігурації середовища, такі як розробка, тестування та виробництво, без зміни коду. Приклад використання змінних оточення у проекті Symfony:

```
# .env file
DATABASE_URL=mysql://db_user:db_password@127.0.0.1:3306/db_name
APP_SECRET=secretkey
```

У середовищі Docker змінні оточення особливо корисні для керування конфігураційними даними у різних контейнерах. Вони дозволяють легко змінювати конфігурацію без модифікації коду програми. Приклад використання змінних оточення в середовищі Docker:

```
# Dockerfile
FROM php:7.4-fpm

ENV DATABASE_URL="mysql://user:password@127.0.0.1:3306/db_name"

# rest of the Dockerfile
```

Приклад використання змінних оточення в конфігурації Docker Compose для проекту:

```
# docker-compose.yml
version: '3'
services:
  app:
    build:
    environment:
      DATABASE_URL: "mysql://user:password@database:3306/db_name"
```

У цьому прикладі змінну твердотільний накопичувач з середовища DATABASE_URL встановлено у файлі docker-compose.yml, який використовується для налаштування контейнерів Docker для програми. Це дозволяє легко змінювати конфігурацію для різних середовищ, таких як розробка та виробництво, без зміни коду.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						42

Інші сервіси та конфігурації: make-файли відіграють важливу роль у проектах з розробки програмного забезпечення, пропонуючи наступні переваги:

- автоматизація: make-файли автоматизують процес збірки, визначаючи залежності та команди, необхідні для збірки проекту
- відтворюваність: makefiles забезпечують відтворюваність, гарантуючи, що будь-хто, хто збирає проект, використовуючи той самий makefile, отримає ідентичні результати
- перенесення: makefiles спрощують перенесення проектів на різні операційні системи, враховуючи специфічні для платформи залежності та команди
- спрощення: makefiles абстрагуються від деталей процесу збірки, дозволяючи розробникам зосередитися на написанні коду
- налагодження: makefiles надають чіткий і стислий запис кроків збірки, що полегшує налагодження процесу збірки

Підсумовуючи, можна сказати, що make-файли - це просте та ефективне рішення для управління процесом збірки в проектах з розробки програмного забезпечення. Вони допомагають автоматизувати повторювані завдання, забезпечують відтворюваність збірок, спрощують загальний процес і допомагають на етапі налагодження.

Завдяки використанню цих практик та інструментів впровадження, система управління завданнями отримує добре структуровану та підтримувану кодову базу.

Ретельно впроваджуючи веб-систему обліку та планування робочого часу на підприємстві, організації можуть впорядкувати свої процеси управління часом, підвищити продуктивність і загальну ефективність. Успішне впровадження програми вимагає поєднання технічних знань, дотримання найкращих практик, а також постійного моніторингу та вдосконалення для задоволення мінливих потреб підприємства.

Висновки до розділу 2

У цьому розділі ми заглибилися у розробку та реалізацію веб-системи для обліку та планування робочого часу на підприємстві. По-перше, ми визначили варіанти використання та створили об'єктно-орієнтовану структуру системи. Шляхом ретельного аналізу ми визначили ключові функціональні можливості, необхідні для того, щоб система задовольняла потреби різних ролей користувачів.

Далі ми зосередилися на розробці бази даних системи. Використовуючи можливості PostgreSQL, надійної системи управління реляційними базами даних, ми створили надійну структуру для зберігання та управління важливими даними, пов'язаними з клієнтами, проектами, завданнями, кінцевими термінами та іншою важливою інформацією. Дизайн бази даних був спрямований на оптимізацію пошуку та зберігання даних, забезпечення ефективного та безпечного доступу до інформації.

Потім ми приступили до розробки та впровадження алгоритмів системи. Це передбачало розробку ефективних та дієвих алгоритмів для підтримки важливих функцій системи, як пошук, розподіл завдань, сповіщення та обробка даних. Використовуючи відповідні алгоритми та найкращі практики, ми прагнули підвищити продуктивність системи та забезпечити надійну роботу.

Ми перетворили дизайн системи та алгоритми на повнофункціональний додаток. Протягом усього процесу реалізації ми надавали пріоритет зручності використання, масштабованості та безпеці, щоб забезпечити надійне та зручне для користувача рішення.

Загалом, цей розділ був присвячений різним етапам розробки та впровадження веб-системи, включаючи визначення варіантів використання, об'єктно-орієнтовану структуру, розробку бази даних. Впроваджуючи найкращі галузеві практики та використовуючи відповідні технології, ми мали на меті створити комплексне рішення, яке задовольнить потреби підприємств в ефективному управлінні та плануванні робочого часу.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ СИМУЛЯЦІЇ

3.1 Порядок встановлення та налаштування системи

Для успішного встановлення та налаштування CRM-системи важливо враховувати серверні рекомендації та дотримуватися покрокової процедури, описаної нижче.

Серверні рекомендації для системи:

Процесор: сервер з багатоядерним процесором для забезпечення кращої масштабованості та продуктивності (для досягнення оптимальних результатів щонайменше 4-6, залежно від розміру проекту та очікуваного навантаження).

Оперативна пам'ять: достатній обсяг оперативної пам'яті для підтримки безперебійної роботи системи (мінімум 8 ГБ, ідеальний обсяг може змінюватися залежно від робочого навантаження та обсягу даних).

Зберігання даних: твердотільний накопичувач забезпечить швидший ввід/вивід порівняно з традиційними жорсткими дисками HDD.

Мережа: для надійної роботи або при обробці великих обсягів даних, необхідно швидке і стабільне підключення до Інтернету.

Безпека: використання брандмауерів, оновлення операційної системи і програмного забезпечення, по SSH і SSL-сертифікати для безпечних з'єднань.

Моніторинг та ведення журналів: система моніторингу для відстеження продуктивності сервера та додатків, інструменти збору та аналізу логів для ефективного виявлення та вирішення потенційних проблем.

Процедура встановлення та налаштування системи:

Версія Ubuntu: Використовуйте останню стабільну версію Ubuntu, яка підтримується Docker, наприклад, Ubuntu 20.04 LTS.

Встановлення Docker: Дотримуйтесь офіційної документації Docker, щоб встановити Docker у вашій системі Ubuntu. Це передбачає додавання офіційного репозиторія Docker і встановлення пакунка Docker Engine.

Налаштування доступу до Docker: Додайте користувача, який запускає процеси Docker, до групи докерів, щоб увімкнути запуск команд Docker без підвищених привілеїв.

Налаштування ресурсів контейнера: Використовуйте конфігураційні файли Docker для визначення обмежень ресурсів для кожного контейнера, зокрема пам'яті, дискового простору, процесорних обмежень тощо. Це дозволяє ефективно розподіляти ресурси для кожної служби.

Моніторинг та ведення журналів: Розширюйте можливості моніторингу системи та ведення логів, встановивши такі інструменти, як Prometheus, Grafana, ELK (Elasticsearch, Logstash, Kibana) або інші відповідні опції. Ці інструменти дозволяють відстежувати продуктивність та аналізувати операції з контейнерами Docker.

Заходи безпеки: Забезпечте безпеку вашої системи, регулярно оновлюючи як Ubuntu, так і Docker, встановлюючи брандмауери для обмеження доступу до портів Docker і використовуючи SSL-сертифікати для безпечного з'єднання.

Сервіси в системі:

CRM-система включає в себе різні сервіси, які полегшують її функціональність. До таких сервісів відносяться

manager-nginx: Розгортає веб-сервер Nginx в якості проксі, перенаправляючи трафік на процес PHP-FPM.

manager-php-fpm: виконує код Symfony 4 і має залежності від таких сервісів, як PostgreSQL, Redis, Robot Queue тощо.

manager-php-cli: Запускає PHP-CLI, що дозволяє виконувати команди Symfony з командного рядка. Він також покладається на PostgreSQL, Redis, Robot Queue та інші сервіси.

manager-queue-worker: Запускає працівників черги, відповідальних за обробку асинхронних завдань у CRM-системі, використовуючи команду Symfony messenger:consume.

manager-node-watch: Відстежує зміни в JavaScript-кодi за допомогою Nodemon, автоматично перезавантажуючи додаток при зміні файлів.

manager-node: Виконує команди Node.js, релевантні для проекту.

manager-postgres: Запускає базу даних PostgreSQL і дозволяє конфігурувати дані доступу, такі як користувач, пароль та ім'я бази даних.

manager-redis: Запускає Redis, який слугує кешем і сховищем даних для CRM-системи.

manager-queue-redis: Ініціює окремий екземпляр Redis, який використовується як черга для асинхронних завдань.

storage: Розгортає окремий веб-сервер Nginx, що надає доступ до файлового сховища CRM-системи.

storage-ftp: Запускає FTP-сервер, який полегшує доступ до файлів за протоколом FTP.

mailer: Запускає поштовий сервер MailHog, перехоплюючи та відображаючи електронні листи, надіслані з CRM-системи.

centrifugo: Запускає Centrifugo, сервер реального часу, що дозволяє інтерактивне спілкування з клієнтами. Налаштований з такими заходами безпеки, як секретний ключ та ключ API.

volumes: Визначає томи даних для постійного зберігання даних PostgreSQL, Redis і Composer.

У контексті нашої системи управління часом ми можемо використовувати діаграми компонентів UML для зображення статичних і структурних властивостей системи (рис.3.1). Ці діаграми слугують візуальним представленням фізичних частин та організації системи, ілюструючи різні компоненти, їхню взаємодію та взаємозв'язки в межах більшої системи.

Метою використання діаграм компонентів в нашій системі управління часом є документування архітектури системи та моделювання проектування розподілених систем. Ці діаграми надають абстрактне уявлення про систему, фокусуючись на самих компонентах та їхніх взаємозв'язках між собою і системою в цілому.

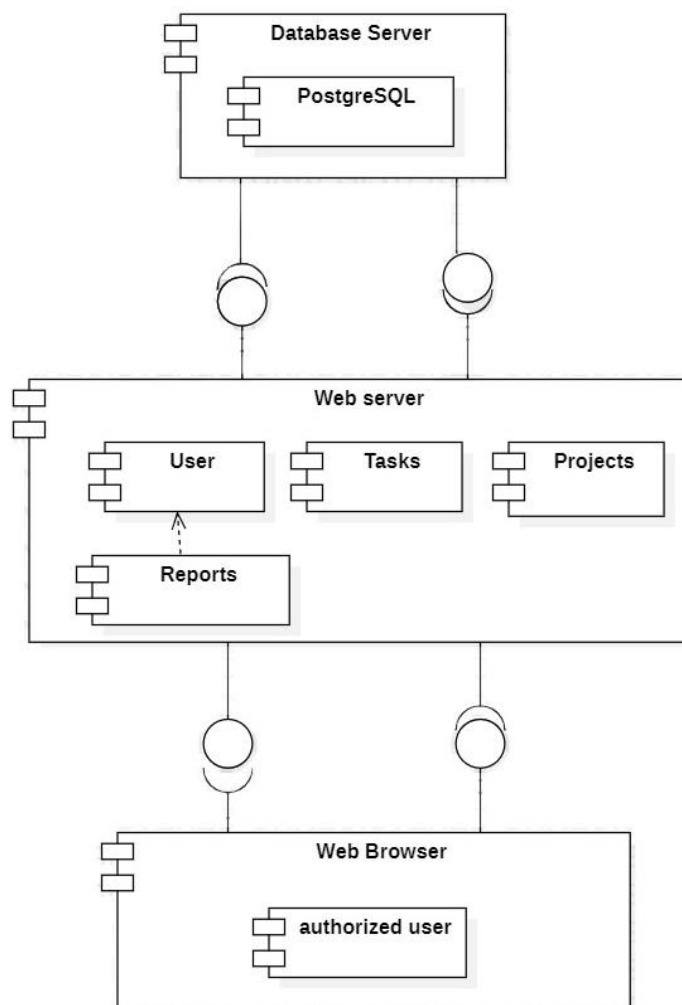


Рисунок 3.1 – Діаграма компонентів

У сфері розробки програмного забезпечення діаграми розгортання UML відіграють вирішальну роль у представленні того, як система розгортається на фізичних апаратних компонентах. Ці діаграми ілюструють взаємозв'язок між апаратними вузлами та програмними компонентами, такими як процеси, компоненти та об'єкти, які розгортаються на них.

Основна мета діаграми розгортання (рис.3.2) - продемонструвати архітектуру виконання системи. Вона охоплює апаратні компоненти, програмні компоненти та проміжне програмне забезпечення, яке полегшує їх з'єднання. Дотримуючись правил проектування, які визначають елементи, топології та зв'язки, ці діаграми ефективно відображають фізичне розміщення артефактів на вузлах.

Використовуючи діаграми розгортання, ми можемо отримати ряд переваг.

Серед них - можливість представляти складні взаємозв'язки між компонентами, що полегшує швидке виявлення джерел помилок і допомагає визначити вузькі місця в системі.

Загалом, використання діаграм компонентів і діаграм розгортання дозволяє нам ефективно візуалізувати і передавати взаємодію компонентів системи в нашій системі управління часом. Ці діаграми покращують наше розуміння архітектури системи, підтримують рішення щодо проектування системи, а також дозволяють ефективно вирішувати проблеми та оптимізувати роботу.

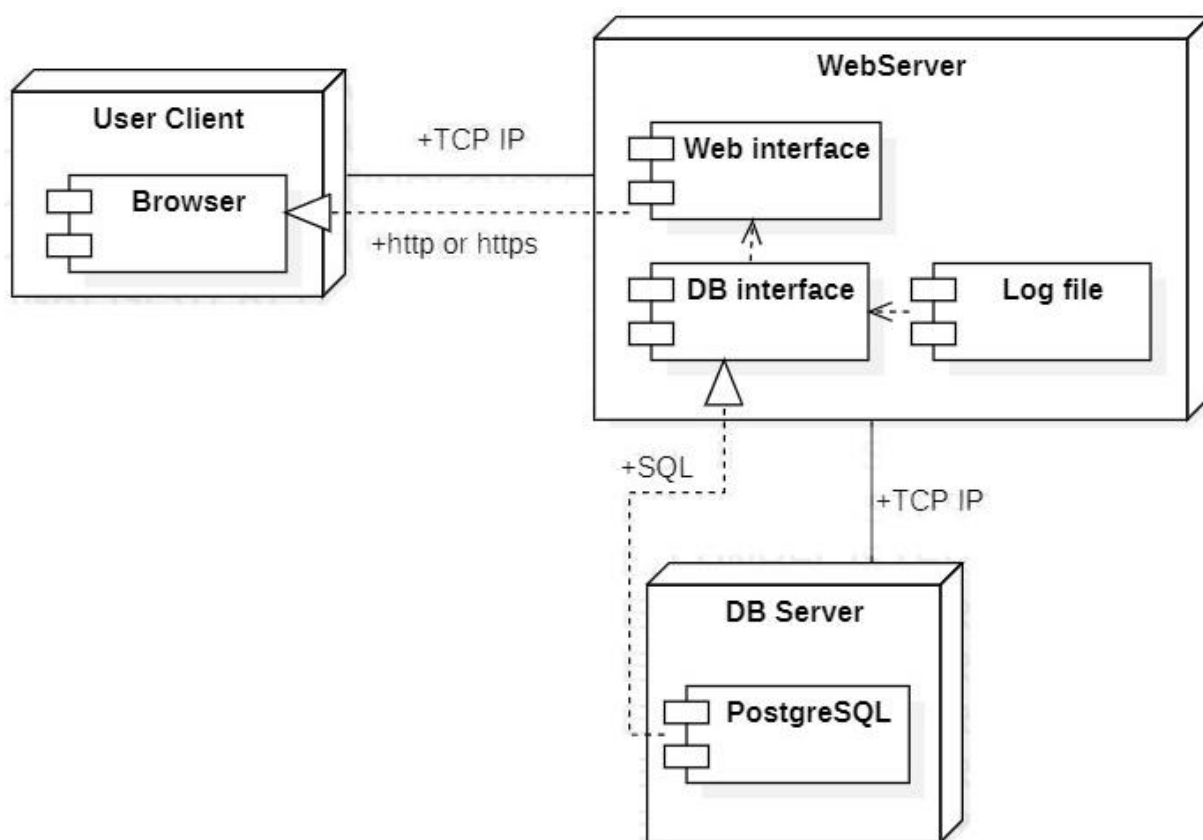


Рисунок 3.2 – Діаграма розгортання

Успішне встановлення та налаштування системи передбачає дотримання рекомендацій щодо серверів, виконання описаних процедур, а також забезпечення належного розподілу ресурсів і заходів безпеки. Використовуючи відповідні інструменти для моніторингу, реєстрації та захисту системи, ви можете оптимізувати її продуктивність і надійність.

3.2 Інтерфейс та порядок роботи

Перейдемо до огляду інтерфейсу та робочого процесу нашої системи управління часом. Ми розглянемо різні скріншоти і надамо короткі описи до кожного з них, підкреслюючи ключові особливості та функціональні можливості системи. Ці скріншоти демонструють елементи користувацького інтерфейсу, з якими користувачі взаємодіятимуть під час роботи.

Сторінка входу відображає інтерфейс (рис.3.3), де користувачі можуть ввести свої облікові дані для доступу до системи. Це безпечна та персоналізована точка входу для авторизованих користувачів, що забезпечує їхню конфіденційність та безпеку даних. Також наявне посилання на реєстрацію користувача.

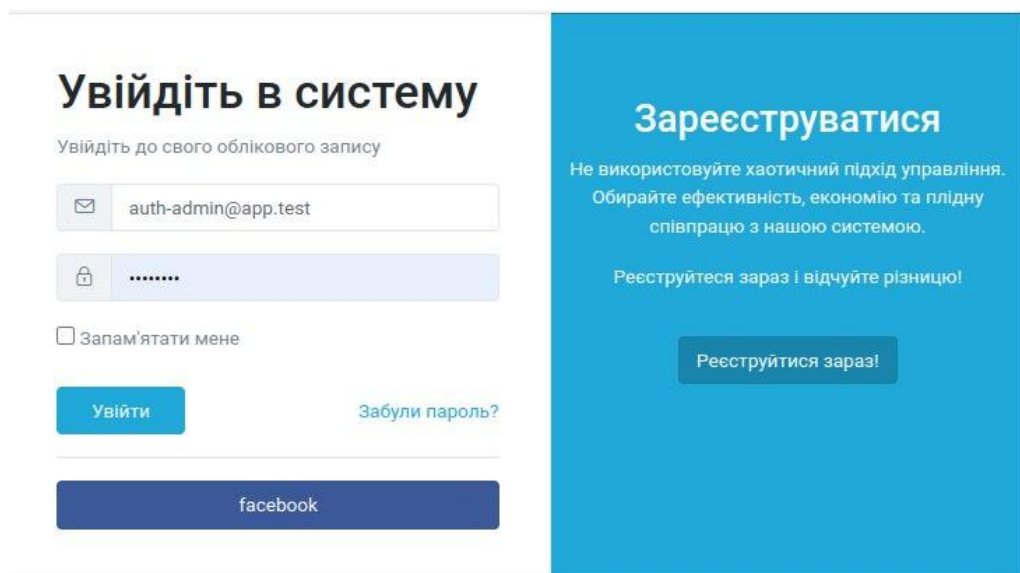


Рисунок 3.3 – Вхід в систему

Реєстрація демонструє користувацький інтерфейс для створення облікового запису новими користувачами (рис.3.4). Він містить поля для введення імені користувача, пароля, адреси електронної пошти. Це дозволяє користувачам зареєструватися і отримати доступ до системи. Після успішного завершення реєстрації користувачу надходить повідомлення на електронну пошту. Воно слугує підтвердженням реєстрації і містить інструкції з активації облікового запису та вітальне повідомлення.

Реєстрація

Створіть свій обліковий запис

Маоквська

Ольга

@ avrora758@gmail.com

.....

Створити акаунт

facebook

Рисунок 3.4 – Реєстрація

Сторінка авторизації також містить користувацький інтерфейс для скидання забутого або втраченого пароля (рис.3.5). Він дозволяє користувачам ввести адресу електронної пошти та ініціює процес скидання пароля, гарантуючи, що вони зможуть відновити доступ до свого облікового запису.

Далі користувачу буде надіслано повідомлення про скидання пароля на електронну пошту. Воно містить інструкції та безпечне посилання для скидання пароля, що гарантує захист облікового запису.

Скинути пароль

@ avrora758@gmail.com

Скинути пароль

Рисунок 3.5 – Скидання паролю

Скріншот головного меню демонструє інтерфейс (рис.3.6), за допомогою якого користувачі можуть переміщатися по різних розділах і функціях системи управління часом та логотип системи. Це централізований центр для доступу до різних функцій та опцій, доступних у додатку. Для користувача доступні такі

вкладки, як: головна, проекти та профіль користувача, а для адміністратора додається список користувачів та учасників.

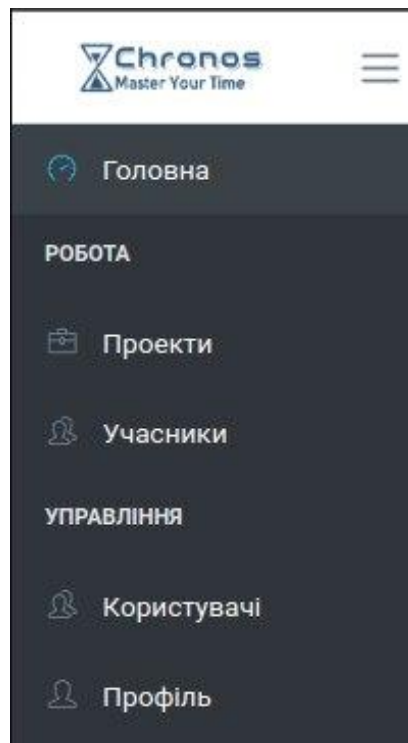


Рисунок 3.6 – Головне меню

Скріншот домашньої сторінки (рис.3.7) надає візуальне представлення інтерфейсу користувача для головного екрану системи. Домашня сторінка слугує централізованим центром, де користувачі можуть зручно отримати доступ до важливої інформації та ефективно керувати своїми завданнями.

Однією з особливостей домашньої сторінки є віджет календаря, який пропонує комплексний перегляд розкладу користувача на наступний тиждень. Він дозволяє користувачам легко планувати та організовувати свої завдання.

Крім того, головна сторінка містить список проміжних етапів завдань, призначених користувачеві, і завдань, створених користувачем. Цей список відображає важливу інформацію, включаючи номери завдань, назви проектів, назви завдань і статуси. Завдяки стислому та організованому представленню цієї інформації на головній сторінці користувачі можуть відстежувати свій прогрес, визначати пріоритетність завдань і залишатися в курсі своїх обов'язків.

Завдання						Вийти
05	06	07	08	09	10	11
▶ #44 Русі. Горе		✚ #46 Ловці	📅 #43 І тут монгольські	📅 #39 Значить	▶ #46 Ловці	
			✚ #47 Максим	▶ #45 І знов	▶ #47 Максим	
				✚ #48 Військо наше жде		

Мої завдання				Завдання для мене			
#	Проект	Назва	Статус	#	Проект	Назва	Статус
100	Кузня	Тільки Максим	Нове	100	Кузня	Тільки Максим	Нове
99	CodeCrafters		Нове	99	CodeCrafters		Нове
98	Кузня	Вовка	Готово	97	Кузня	Мирослава виросла	В процесі
97	Кузня	Мирослава виросла	В процесі	96	Кузня	Кождий дбає	Допомога

Рисунок 3.7 – Головна сторінка

Профіль користувача представляє інтерфейс, за допомогою якого користувачі можуть переглядати та керувати інформацією свого профілю (рис.3.8). Вона може включати особисті дані, контактну інформацію, налаштування та можливості оновлення профілю. Це дозволяє користувачам налаштовувати свій профіль відповідно до своїх уподобань і підтримувати інформацію в актуальному стані.

Також користувач може змінювати своє ім'я або, а також пов'язану з ними електронну пошту, гарантуючи, що інформація в їхньому профілі залишається точною і актуальною.

Профіль	
Ім'я	Маоквська Ольга 
Пошта	avro758@gmail.com 
Створено	2023-06-07 16:10:50
Роль	Користувач
Статус	Активний

Рисунок 3.8 – Профіль користувача

Список всіх користувачів демонструє інтерфейс (рис.3.9), за допомогою якого адміністратори можуть переглядати список усіх користувачів системи. Він забезпечує основну функціональність управління користувачами, дозволяючи адміністраторам контролювати і наглядати за обліковими записами.

Список включає у себе дату створення користувача, його ім'я та адреса електронної пошти, роль (користувач або адміністратор) та статус (активний, очікуючий або заблокований).

Створити користувача

Ім'я

Поштова скринька

Всі ролі

Всі статуси

Фільтрувати

Скинути

Дата	Ім'я	Пошта	Роль	Статус
2023-06-07	Маюкська Ольга	avvora758@gmail.com	Користувач	Активний
2023-06-07	Ганна Гордієнко	user-projects-index@app.test	Користувач	Активний
2023-06-07	Матвій Корнієнко	admin-projects-index@app.test	Адміністратор	Активний
2023-06-07	Софія Кравченко	admin@app.test	Адміністратор	Активний
2023-06-07	Мілана Приходько	profile-user@app.test	Користувач	Активний
2023-06-07	Юрій Радченко	auth-user@app.test	Користувач	Активний

Рисунок 3.9 – Список користувачів

Список всіх груп представляє інтерфейс (рис.3.10), який дозволяє користувачам отримувати доступ до різних груп у системі та керувати ними. Адміністратори можуть створити нові групи, додати до них користувачів та призначити певні дозволи.

Учасники

Групи

Додати групу

Назва	Учасники	
Користувачі	1	 
Наш персонал	1	 
Наш персонал	1	 
Наші клієнти	1	 
Тестовий персонал	1	 

Рисунок 3.10 – Список груп

Вкладка всіх проектів демонструє інтерфейс (рис.3.11), за допомогою якого користувачі можуть отримати доступ до всіх проектів у системі та керувати ними. Він надає огляд поточних проектів, їх статусу та пов'язаних з ними деталей.

Фільтрація дозволяє користувачам фільтрувати проекти на основі їхнього статусу або ім'я. Це зручний спосіб класифікувати і переглядати проекти на основі таких критеріїв, як "активний", "архівований".

Назва	Активний	Фільтрувати	Скинути
Ім'я			
Code Verse			Активний
CodeCrafters			Активний
Tech Blaze			Активний
Користувацький проект			Активний
Кузня			Активний

Рисунок 3.11 – Список проектів

Календар на місяць ілюструє відображає щомісячний календар завдань користувача (рис.3.12). Це допомагає користувачам візуалізувати і планувати свої кінцеві терміни та роботу на місяць, забезпечуючи ефективне управління.

29	30	31	01	02	03	04
✚ #43 І тут монгольські			✚ #44 Русі. Горе		▶ #43 І тут монгольські	✚ #45 І знов
05	06	07	08	09	10	11
▶ #44 Русі. Горе		✚ #10000002 Завдання	📅 #43 І тут монгольські	📅 #39 Значить	▶ #46 Ловці	
		✚ #46 Ловці	📅 #10000003 Завдання	▶ #45 І знов	▶ #47 Максим	
		✚ #10000001 Завдання	✚ #47 Максим	✚ #48 Військо наше жде		
		✚ #10000003 Завдання				
12	13	14	15	16	17	18
✚ #49 Ліпше би було	📅 #48 Військо наше жде		▶ #49 Ліпше би було	📅 #37 Життя «він»		
	✚ #50 Направо		✚ #51 Страшенно	▶ #50 Направо		
				✚ #52 Тугар Вовк з		

Рисунок 3.12 – Список проектів

Список усіх завдань надає організований огляд, що дозволяє користувачам визначати пріоритети та ефективно відстежувати їхній прогрес (рис.3.13). Список включає у себе номер завдання, дату створення, автора, назву проекту, в якому він знаходиться, назву завдання, тип (відсутній, нова функція, баг), рівень пріоритетності, список виконавців, кінцевий термін, статус (нове, готово, в архіві, потребує допомоги, в процесі, на перевірці) та відсоток готовності.

Всі типи

Всі статуси

Всі пріоритети

Всі автори

Всі виконавці

Всі рівні

Фільтрувати

Очистити

Всі завдання

Завдання для мене

Мої завдання

ID	Дата	Автор	Проект	Ім'я	Тип		Виконавці	Кінцевий термін	Статус	Готовність
10000003	2023-06-07	First Last	Користувацький проект	Завдання	Нова функція	■ ■ ■ ■		2023-06-08	Нове	
10000002	2023-06-07	First Last	Проект без користувачів	Завдання	Нова функція	■ ■ ■ ■			Нове	
10000001	2023-06-07	First Last	Користувацький проект	Завдання	Нова функція	■ ■ ■ ■			Нове	
100	2023-09-17	Софія Кравченко	Кузня	Тільки Максим	Нова функція	■ ■ ■ ■	Софія Кравченко		Нове	
99	2023-09-16	Софія Кравченко	CodeCrafters		Помилка	■ ■ ■ ■	Софія Кравченко		Нове	
98	2023-09-15	Софія Кравченко	Кузня	Вовка	Помилка	■ ■ ■ ■			Готово	
97	2023-09-12	Софія Кравченко	Кузня	» Мирослава виросла	Нова функція	■ ■ ■ ■	Владислав Бойко Софія Кравченко		В процесі	
96	2023-09-11	Владислав Бойко	Кузня	» Кожний дбає	Нова функція	■ ■ ■ ■	Софія Кравченко	2023-09-13	Допомога	

Рисунок 3.13 – Список завдань

Вкладка дії відображає всі дії та зміни, зроблені у завданнях поточного проекту (рис.3.14). Він містить журнал оновлень, коментарів та інших дій, пов'язаних із завданням, забезпечуючи прозорість і підзвітність.

З його допомогою ми можемо визначити автора зміни, дату та саму дію що була виконана над завданням (зміна назви, додання коментаря, зміна властивостей завдання, зміна виконавців, створення або видалення старих завдань та таке інше).

Проекти	Дії	Завдання	Календар	Ролі
Дата	Проекти	Завдання	Дії	
2023-09-22 21:09:27	Кузня	Тільки Максим	Владислав Бойко змінив(-ла) пріоритет на ■■■■	
2023-09-20 21:06:27	CodeCrafters		Софія Кравченко змінив(-ла) пріоритет на ■■■■	
2023-09-20 21:00:27	Кузня	Мирослава виросла	Владислав Бойко встановив(-ла) як дочірнє завдання #45	
2023-09-19 21:09:27	Кузня	Тільки Максим	Владислав Бойко призначив(-ла) Софія Кравченко	
2023-09-19 21:03:27	Кузня	Вовка	Владислав Бойко змінив(-ла) пріоритет на ■■■■	
2023-09-18 21:03:27	Кузня	Вовка	Владислав Бойко змінив(-ла) статус на Готово	
2023-09-18 21:03:27	Кузня	Вовка	Владислав Бойко змінив(-ла) прогрес на 100%	

Рисунок 3.14 – Список дій

Налаштування проекту включають у себе три вкладки: загальні, департаменти та учасники (рис.3.15). Доступ до даних змін має лише адміністратор.

Загальні налаштування дозволяють користувачам змінити назву, архівувати або видалити проект. Список відділів проекту ілюструє інтерфейс, який дозволяє користувачам керувати та організовувати відділи в рамках проекту. Список учасників дозволяє керувати та призначати учасників проекту та редагувати їх. Він надає огляд учасників проекту та їхніх ролей у проекті.

Дії	Завдання	Календар	Налаштування
Загальні	Департаменти	Учасники	
Призначити учасника	Ролі		
Назва	Департаменти	Ролі	
Софія Кравченко	Розробники	Менеджер	
Владислав Бойко	Маркетинг	Гість	

Рисунок 3.15 – Налаштування проекту

Скріншот завдання демонструє інтерфейс, за допомогою якого користувачі можуть переглядати і керувати окремими завданнями (рис.3.16). Він надає детальну інформацію про конкретне завдання, включно з його

описом, призначеними учасниками, терміном виконання і пріоритетом та іншими властивостями.

Також користувач може завантажити певні файли, переглянути дії, пов'язанні із завданням та залишити коментар.

Смерека була

Розпочати

Призначити виконавців

Додати підзавдання

Редагувати

Більше...

Завдання

ID	86		
Дата створення	2023-08-23 20:24:27	Кінцевий термін	
Тип	Нова функція	Статус	Нове
Пріоритет	Найвищий	Прогрес	0

Файли

Кровавий відблиск огнів, що освічував околицю, робив його лице страшним і диким і меркотів на його грудях! - Ні! Коли вже таке твоє над ним стояв кровавий медвідь, вискаливши свої страшні зуби і.

Робоча область

2023-08-23 20:24 Софія Кравченко створив(-ла) в CodeCrafters як Нова функція з пріоритетом

2023-08-26 20:27 Софія Кравченко призначив(-ла) Софія Кравченко

Автор

Софія Кравченко

Виконавці

Софія Кравченко

Рисунок 3.16 – Налаштування проекту

Створення завдання демонструє інтерфейс для створення нового завдання. Він дозволяє користувачам вводити деталі завдання, такі як назва, опис, термін виконання і призначати виконавців, що полегшує ефективне створення і призначення завдань.

У цьому підрозділі ми розглянули інтерфейс і робочий процес нашої системи управління часом за допомогою серії скріншотів. Кожен із них представляв важливий аспект системи, від автентифікації та реєстрації користувача до управління завданнями та співпраці в межах проекту, та налаштування профілю. Ці візуальні матеріали дають уявлення про зручний дизайн і функціональні можливості, які пропонує наша система, дозволяючи користувачам ефективно організовувати свої завдання та оптимізувати продуктивність.

3.3 Тестування роботи системи

Тестування роботи системи управління часом має вирішальне значення для забезпечення її функціональності, надійності та якості. У цьому розділі ми обговоримо різні типи тестів, які можна проводити для перевірки поведінки системи. Виконуючи їх, ми можемо оцінити різні аспекти програмного забезпечення і переконатися, що система відповідає бажаним вимогам.

Перш ніж зануритися у фазу тестування, важливо врахувати рекомендації щодо сервера для системи управління часом. Сюди входить наявність надійної та масштабованої серверної інфраструктури, яка може впоратися з очікуваним навантаженням і забезпечити оптимальну продуктивність.

Функціональні тести перевіряють, як різні компоненти, модулі або системи управління часом взаємодіють один з одним. Вони гарантують, що окремі частини програми працюють як цілісна система. Наприклад, тест `SignUpTest` перевіряє, чи можна успішно зареєструвати нового користувача через API.

```
public function testSuccess(): void
{
    // Arrange
    $user = (new UserBuilder())->viaEmail()->build();

    // Act
    $response = $this->client->post('/api/signup', [
        'json' => [
            'email' => $user->getEmail(),
            'password' => 'password123',
        ],
    ]);

    // Assert
    self::assertEquals(200, $response->getStatusCode());
    // Additional assertions to verify the user is successfully registered
}
```

Юніт-тести зосереджені на тестуванні окремих функцій, методів або класів системи управління часом. Вони гарантують, що ці компоненти працюють коректно і незалежно від залежностей. Наприклад, тест `ActivateTest` перевіряє успішну активацію користувача і перевіряє статус його активності.

```

public function testSuccess(): void
{
    // Arrange
    $user = (new UserBuilder())->viaEmail()->build();

    // Act
    $user->block();
    $user->activate();

    // Assert
    self::assertTrue($user->isActive());
    self::assertFalse($user->isBlocked());
}

```

Інтеграційні тести оцінюють взаємодію між різними компонентами системи управління часом, включаючи базу даних, зовнішні сервіси або API. Вони гарантують, що ці компоненти працюють разом правильно. Наприклад, ProfileFixture створює тестові дані користувача в базі даних, щоб перевірити інтеграцію з профілем користувача.

```

public function testCreateProfile(): void
{
    $user = (new UserBuilder())->viaEmail()->build();
    $profileData = [
        'first_name' => 'John',
        'last_name' => 'Doe',
        // Additional profile data
    ];
    $this->client->request('POST', '/api/profile', [
        'headers' => ['Authorization' => 'Bearer ' . $user->getToken()],
        'json' => $profileData,
    ]);
}

```

Навантажувальні тести оцінюють продуктивність, масштабованість і надійність системи управління часом в умовах високого навантаження або стресу. Вони імітують сценарії, коли система піддається численним запитам одночасно. Наприклад, AuthTest оцінює поведінку системи, коли користувач здійснює автентифікацію через зовнішній сервіс.

```

public function testSuccess(): void
{
    $user = User::signUpByNetwork(
        $id = Id::next(),
        $date = new DateTimeImmutable(),
        $name = new Name('First', 'Last'),
        $network = 'facebook',
        $identity = '0000001'
    );
    self::assertTrue($user->isActive());
}

```

Розглянемо детальніше тест для перевірки, чи вдається успішно зареєструвати нового користувача через API (дод.В).

У методі `testSuccess()` створюється новий користувач за допомогою класу `UserBuilder`. Метод `viaEmail()` використовується для створення користувача з адресою електронної пошти. Тест імітує процес реєстрації користувача шляхом надсилання POST-запиту на шлях `/api/signup`. Запит містить пошту та пароль користувача у форматі JSON.

Тест стверджує, що код статусу відповіді, отриманої від сервера, дорівнює 200, що означає успішну реєстрацію. Можна додати додаткові твердження для додаткової перевірки успішної реєстрації користувача, наприклад, перевірити, чи присутній створений користувач у базі даних, або вивчити вміст відповіді.

Клас `SignUpTest` може також включати інші методи тестування для покриття різних сценаріїв і граничних ситуацій, пов'язаних з реєстрацією користувача:

- `testGet`: перевіряє, що метод GET для шляху `/api/auth/signup` повертає код статусу 405 (Метод не дозволений), це гарантує, що кінцева точка реєстрації користувачів підтримує тільки метод POST
- `testNotValid`: тестує поведінку процесу реєстрації при наданні недійсних облікових даних користувача, надсилає POST-запит на шлях `/api/auth/signup` з недійсними даними і стверджує, що відповідь має код статусу 400 (Недійсний запит)

Проводячи комбінацію функціональних тестів, модульних тестів, інтеграційних тестів і навантажувальних тестів, ми можемо переконатися, що система управління працює ефективно, відповідає необхідним функціональним можливостям і оптимально працює в різних умовах. Ці тести допомагають виявити та усунути будь-які потенційні проблеми або помилки, забезпечуючи надійність та якість системи.

Висновки до розділу 3

Загалом третій розділ був присвячений інтерфейсу та порядку роботи з веб-системою обліку та планування робочого часу на підприємстві. Він охоплює три основні напрямки: встановлення та налаштування параметрів системи, структуру інтерфейсу та тестування роботи програмного комплексу.

У першій частині було розглянуто процедуру встановлення та налаштування параметрів системи. Це передбачало налаштування необхідних компонентів і залежностей для забезпечення безперебійного функціонування системи. Дотримуючись наданих інструкцій, користувачі можуть ефективно встановити та налаштувати параметри системи, що дозволить їм адаптувати її до конкретних вимог підприємства.

Друга частина розділу була присвячена структурі інтерфейсу веб-системи. У ній було представлено огляд різних компонентів і модулів, що складають інтерфейс системи, пропонуючи користувачам комплексне розуміння того, як вони взаємодіють з системою. Структура інтерфейсу забезпечує зручний та інтуїтивно зрозумілий досвід роботи, що сприяє ефективному обліку та плануванню робочого часу на підприємстві.

Нарешті, було розглянуто такий важливий аспект, як тестування роботи програмного комплексу. Було підкреслено важливість проведення різних типів тестів для забезпечення функціональності, надійності та продуктивності системи. Серед згаданих тестів були функціональні, модульні, інтеграційні та навантажувальні. Ретельне тестування роботи системи дозволяє виявити та вирішити потенційні проблеми, гарантуючи якісний та надійний програмний комплекс для точного обліку та планування робочого часу на підприємстві.

ВИСНОВКИ

Даний проект охопив комплексний аналіз, розробку та впровадження веб-системи для обліку та планування робочого часу на підприємстві. Проект був розділений на три основні розділи, кожен з яких стосувався конкретних аспектів процесу розробки і завершувався цінними висновками.

Перший розділ був присвячений первинному аналізу інформаційних технологій планування та обліку робочого часу на підприємстві. Розділ розпочався з чіткої постановки завдання, окреслення цілей та завдань проекту. Було проведено глибокий аналіз аналогічних програмних продуктів з урахуванням їх особливостей, функціональних можливостей та придатності до конкретних вимог підприємства.

Крім того, в розділі було розглянуто критично важливе рішення щодо вибору архітектури для веб-системи. Було оцінено різні архітектурні варіанти, враховуючи такі фактори, як масштабованість, продуктивність і зручність обслуговування. Було пояснено обґрунтування вибору конкретної архітектури, підкреслено її сумісність з потребами підприємства та майбутнім зростанням.

Другий розділ був присвячений розробці та впровадженню веб-системи. Розділ розпочався з визначення варіантів використання, виявлення різних сценаріїв і функціональних можливостей, які повинна підтримувати система. Було визначено об'єктно-орієнтовану структуру системи, висвітлено взаємозв'язки між різними компонентами та їх ролі у виконанні сценаріїв використання. Крім того, була розроблена база даних системи, яка забезпечує ефективне зберігання та управління даними, необхідними для обліку та планування робочого часу.

Одним з ключових аспектів розділу була розробка та реалізація алгоритмів системи. Ретельна увага була приділена алгоритмам, необхідним для виконання таких завдань, як облік робочого часу, розподіл ресурсів і генерація звітів. Алгоритми були розроблені з метою оптимізації ефективності, точності та гнучкості відповідно до конкретних потреб підприємства.

Третій розділ був присвячений інтерфейсу та процедурі роботи з веб-системою. Він охоплював рекомендації щодо встановлення та налаштування параметрів системи, що забезпечило плавний і безпроблемний процес налаштування. Структура інтерфейсу системи була ретельно розроблена, щоб запропонувати користувачам інтуїтивно зрозумілу та зручну роботу. Було враховано такі фактори, як навігація, введення даних та відображення інформації, що сприяють простоті використання та мінімізації помилок.

На додаток до дизайну інтерфейсу, тестування роботи програмного пакету було підкреслено як вирішальний крок у забезпеченні функціональності, надійності та продуктивності системи. Для перевірки поведінки системи в різних умовах були проведені різні типи тестів, такі як функціональні, модульні, інтеграційні та навантажувальні тести. Тести перевіряли як окремі компоненти, так і їх взаємодію, забезпечуючи цілісність та надійність системи.

Таким чином, цей проект продемонстрував комплексний підхід до розробки веб-системи для обліку та планування робочого часу на підприємстві. Завдяки ретельному аналізу, розробці надійної архітектури та алгоритмів, а також впровадженню зручного інтерфейсу, проект успішно вирішив проблеми, пов'язані з плануванням та обліком робочого часу.

В результаті проведеної роботи ми вирішили поставлену нами задачу та зуміли розробити систему планування часу.

Рекомендації та висновки, отримані в результаті цього проекту, надають цінну інформацію для організацій, які прагнуть оптимізувати управління робочим часом. Дотримуючись викладених методологій, організації можуть ефективно використовувати веб-систему, що призведе до підвищення продуктивності, оптимізації операцій та вдосконалення процесів прийняття рішень. Проект слугує основою для майбутніх досягнень у сфері обліку та планування робочого часу, надаючи підприємствам можливість досягати своїх цілей та максимально використовувати свій потенціал.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джош Локхарт. Сучасний PHP. Нові можливості та передовий досвід / Джош Локхарт., 2016 — O'Reilly
2. Роберт К. Мартін Чистий код / Роберт К. Мартін., 2008 - 413с — O'Reilly
3. Sohail Salehi. Mastering Symfony / Sohail Salehi., 2016. – 290 с — Packt Publishing
4. ClickUp (n.d.): Один додаток, який замінить їх усі. [Електронний ресурс] – Режим доступу до ресурсу: <https://clickup.com/>
5. Програмне забезпечення Jira. (n.d.). Атласіан. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.atlassian.com/software/jira>
6. Asana. (n.d.): Інструмент управління роботою. [Електронний ресурс] – Режим доступу до ресурсу: <https://asana.com/>
7. Габріеле Сантомаджо. Кулінарна книга RabbitMQ / Габріеле Сантомаджо, Сігісмондо Боскі. - 290 с - Packt Publishing
8. Білл Вайлдер. Моделі хмарної архітектури / Білл Вайлдер. 100с – O'Рейлі
9. Юрген Шеніг. PostgreSQL 11. Майстерність розробки / - Юрген Шеніг. 354с - ДМК Пресс
10. Немчинський С. Що таке “бізнес логіка”. [Електронний ресурс] – https://foxminded.ua/shcho_take_biznes_lohika_i_yak_pochaty/
11. Важливість uml діаграм [Електронний ресурс] – Режим доступу до ресурсу: <https://gisuser.com/2022/04/why-is-uml-important-in-software-engineering/>
12. Adrian Mouat. Using Docker: Developing and Deploying Software with Containers / Adrian Mouat — O'Reilly
13. Vlad Khononov. Learning Domain-Driven Design / Vlad Khononov., 2021. — O'Reilly
14. Швальбе, К. Управління проектами інформаційних технологій / Швальбе, К., 2019. - Cengage Learning

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	
						66

Специфікація прецедентів

Таблиця А.1

CRUD профілю користувача

Ім'я варіанту використання: CRUD профілю користувача		Рівень: високий
Основний актор: користувач		
Короткий опис:	Створення, редагування, перегляд та зміна даних користувача	
Передумови:	Ініціалізація в системі	
Відносини:		
	Включення:	Ініціалізація
Основний потік:	Авторизований користувач може змінювати лише інформацію про себе (ім'я, пошта, пароль). Адміністратор може редагувати профілі інших та змінювати їх ролі	
Альтернативні/виключні потоки:	Повідомлення помилки	

Таблиця А.2

CRUD задач

Ім'я варіанту використання: CRUD задач		Рівень: високий
Основний актор: користувач		
Короткий опис:	Створення, редагування, перегляд та зміна даних завдання	
Передумови:	Ініціалізація в системі	
Відносини:		
	Включення:	Ініціалізація
Основний потік:	Авторизований користувач, який має доступ до завдань у певному проекті, може редагувати основні дані завдання: його назву, опис, статус, дату створення, кінцевий термін, відсоток прогресу, пріоритетність, тип та виконавець	
Альтернативні/виключні потоки:	Повідомлення помилки	

1. Впровадження служби:

```
// src/Service/EmailSender.php
class EmailSender
{
    public function send($to, $subject, $body)
    {
        // send email code
    }
}

// src/Controller/DefaultController.php
class DefaultController
{
    private $emailSender;

    public function __construct(EmailSender $emailSender)
    {
        $this->emailSender = $emailSender;
    }

    public function index()
    {
        $this->emailSender->send('user@example.com', 'Welcome!', 'Hello and
welcome to our site!');
    }
}
```

2. Впровадження репозиторію:

```
// src/Repository/TaskRepository.php
class TaskRepository
{
    public function findAll()
    {
        // database query to retrieve all tasks
    }
}

// src/Controller/TaskController.php
class TaskController
{
    private $taskRepository;

    public function __construct(TaskRepository $taskRepository)
    {
        $this->taskRepository = $taskRepository;
    }

    public function list()
    {
        $tasks = $this->taskRepository->findAll();
        // return a response with the list of tasks
    }
}
```

Лістинг тесту для перевірки реєстрації

```
<?php

use App\Tests\Builder\User\UserBuilder;
use PHPUnit\Framework\TestCase;

class SignUpTest extends TestCase
{
    public function testSuccess(): void
    {
        // Arrange
        $user = (new UserBuilder())->viaEmail()->build();

        // Act
        $response = $this->client->post('/api/signup', [
            'json' => [
                'email' => $user->getEmail(),
                'password' => 'password123',
            ],
        ]);

        // Assert
        self::assertEquals(200, $response->getStatusCode());
        // Additional assertions to verify the user is successfully
        // registered
    }
}
```