

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

**Пояснювальна записка**

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ CRM СИСТЕМИ  
ДЛЯ АВТОМАТИЗАЦІЇ ДОКУМЕНТООБІГУ У КОМПАНІЇ  
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE CRM SYSTEM  
FOR DOCUMENT FLOW AUTOMATION IN THE COMPANY

Виконав: студент 2 курсу, групи КНТ-112м

Спеціальності 121 Інженерія програмного  
забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

РИНДІН П.Ю.

(прізвище та ініціали)

Керівник ФЕДОРОНЧАК Т.В.

(прізвище та ініціали)

Рецензент ШИТІКОВА О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»  
(повне найменування закладу вищої освіти)

Факультет КНТ  
Кафедра програмних засобів  
Ступінь вищої освіти магістр  
Спеціальність 121 Інженерія програмного забезпечення  
(код і найменування)  
Освітня програма (спеціалізація) Інженерія програмного забезпечення  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф.  
Сергій СУББОТІН  
“ ” 2023 року

**З А В Д А Н Н Я**

**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА**

РИНДІНА Павла Юрійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація CRM системи для автоматизації документообігу у компанії.

Research and software implementation of the CRM system for document flow automation in the company

керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна,  
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 14 ” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | Прізвище, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-4 Основна частина | ФЕДОРОНЧАК Т.В.,<br>доцент                |                |                           |
| Нормоконтролер      | ДЕЙНЕГА Л.Ю., ст.<br>викладач             |                |                           |

7. Дата видачі завдання “04” вересня 2023 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)                                              | Строк виконання етапів проєкту (роботи) | Примітка     |
|-------|---------------------------------------------------------------------------------------|-----------------------------------------|--------------|
| 1     | Постановка завдання роботи.                                                           | 1 тиждень                               | Завдання, ТЗ |
| 2     | Аналіз та дослідження предметної області.                                             | 2-5 тижні                               | Розділ 1, 2  |
| 3     | Розробка архітектури програми.                                                        | 6 тиждень                               | Розділ 3     |
| 4     | Розробка програми.                                                                    | 7-9 тижні                               | Розділ 4     |
| 5     | Тестування та експериментальне дослідження програми.                                  | 10-11 тижні                             | Розділ 5     |
| 6     | Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування. | 12 тиждень                              | Додатки      |
| 7     | Захист роботи.                                                                        | 13 тиждень                              |              |

Студент(ка)

\_\_\_\_\_  
( підпис )      Павло РИНДІН  
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

\_\_\_\_\_  
( підпис )      Тетяна ФЕДОРОНЧАК  
(Ім'я ПРІЗВИЩЕ)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:  
94 с., 31рис., 1 табл., 3 дод., 15 джерел.

CRM СИСТЕМА, МЕНЕДЖМЕНТ ДОКУМЕНТООБІГУ,  
АДМІНІСТРУВАННЯ ДОСТУПУ, ВЕБЗАСТОСУНОК, DJANGO, MYSQL,  
CELERY, REDIS.

Об'єкт дослідження – програмне забезпечення для автоматизації документообігу в компанії.

Предмет дослідження – вебзастосунок для документообігу в компанії.

Мета роботи – розробка програмного забезпечення адміністрування та автоматизації документообігу в компанії.

Матеріали, методи та технічні засоби: мова програмування Python, система бази даних MySQL, асинхронна черга завдань Celery, NoSQL база даних Redis.

Результати. Розроблено програмне забезпечення для автоматизації бюрократичних процесів у компанії, що включає налаштування внутрішніх процесів та управління доступами. Впроваджено систему підпорядкування користувачів залежно від їх ролей та прав. Реалізована реляційна модель бази даних, відповідно до вимог сервісу.

Висновки. Реалізований вебзастосунок надає можливість автентифіковуватись, створювати нових користувачів системи виходячи з їх ролей у компанії, створювати та налаштовувати документи та задачі виходячи з нагальних юридичних або матеріальних потреб, вести комунікацію між підрозділами та виконуючими обов'язки персоналом.

Галузь використання. Автоматизація документообігу та адміністрування бізнес процесів у компаніях з розподіленою структурою підпорядкування.

## **ABSTRACT**

Explanatory note to the diploma qualifying work of the master: 94 pages, 31 figures, 1 table, 3 appendixes, 15 sources

CRM SYSTEM, DOCUMENT MANAGEMENT, ACCESS ADMINISTRATION, WEB APPLICATION, DJANGO, MYSQL.

The object of research is the process of creating a web application for automating document flow in a company. The subject of the research is the software for automating document flow in a company.

The purpose of the work is to develop software for the administration and automation of document flow in a company.

Materials, methods, and technical means: Python programming language, MySQL database system.

Results. Software has been developed to automate bureaucratic processes in the company with the ability to configure internal processes and administer access and permissions in the system. A flexible system of subordination based on the dependencies between system users according to their roles and rights was implemented. A relational database model was implemented in the system.

Conclusions. The implemented web application provides the ability to authenticate or create new system users based on their roles in the company, create and configure documents and tasks based on urgent legal or material needs, and facilitate communication between departments and executing staff.

Field of application. Automation of document flow and administration of business processes in companies with a distributed structure of subordination.

## ЗМІСТ

|                                                           |    |
|-----------------------------------------------------------|----|
|                                                           | С. |
| Перелік скорочень та умовних позначок.....                | 8  |
| Вступ.....                                                | 9  |
| 1 Аналіз предметної області .....                         | 11 |
| 1.1 Вимоги та функціонал CRM системи.....                 | 11 |
| 1.2 CRM системи.....                                      | 12 |
| 1.3 Висновки за розділом 1 .....                          | 15 |
| 2 Розробка архітектури програми.....                      | 16 |
| 2.1 Функціональні вимоги до програмного забезпечення..... | 16 |
| 2.2 Вибір засобів розробки програмного продукту.....      | 20 |
| 2.3 Проєктування бази даних .....                         | 28 |
| 2.4 Висновки за розділом 2 .....                          | 32 |
| 3 Розробка програми.....                                  | 33 |
| 3.1 Алгоритм функціонування програмного забезпечення..... | 33 |
| 3.2 Структура реалізованого програмного забезпечення..... | 36 |
| 3.3 Опис прийнятих під час реалізації рішень .....        | 38 |
| 3.4 Висновки за розділом 3 .....                          | 40 |
| 4 Експлуатація та тестування розробленої програми .....   | 41 |
| 4.1 Призначення програми.....                             | 41 |
| 4.2 Умови виконання програми.....                         | 41 |
| 4.3 Виконання програми .....                              | 41 |
| 4.4 Тестування розробленої програми .....                 | 49 |
| 4.5 Висновки за розділом 4.....                           | 50 |
| Висновки.....                                             | 51 |
| Перелік джерел посилання .....                            | 52 |
| Додаток А Технічне завдання.....                          | 54 |
| Додаток Б Текст програми.....                             | 60 |

|                                    |    |
|------------------------------------|----|
| Додаток В Слайди презентації ..... | 87 |
|------------------------------------|----|

## **ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

- CRM – Customer Relationship Management;
- API – Application Programming Interface;
- CSS – Cascading Style Sheets;
- HTML – HyperText Markup Language;
- SQL – Structured Query Language;
- WSGI – Web Server Gateway Interface;
- БД – база даних;
- СКБД – система керування базами даних.

## ВСТУП

У світі сучасного бізнесу, де ефективність та швидкість відіграють ключову роль, важливо мати в своєму арсеналі інструменти, здатні спростити та оптимізувати робочі процеси. Для цього використовують CRM системи, це потужний інструмент, що розроблюється з урахуванням потреб сучасних компаній, який дозволяє не лише вести обіг документів у зручній та ефективній формі, але й автоматизувати безліч бюрократичних процесів.

CRM-системи розроблюються з метою спростити та поліпшити взаємодію компанії з клієнтами, але їх застосування не обмежується лише цим. Ці системи стають невід'ємними інструментами для управління внутрішніми процесами та ресурсами компанії.

Такого роду системи дозволяють зберігати інформацію про клієнтів, їх історію взаємодії з компанією, вподобання та потреби. Це допомагає поліпшити якість обслуговування клієнтів та створити більш персоналізований досвід. Також вони дозволяють зберігати, організовувати та обмінюватися документами всередині компанії. Це спрощує процеси управління інформацією та зменшує ризики втрати важливих документів що у свою чергу веде до автоматизації бюрократичних процесів, таких як відправка повідомлень, обробка замовлень, планування завдань та багато іншого, що зменшує навантаження на співробітників та зменшує ймовірність помилок.

CRM системи являються надійним інструментом менеджменту різного рівня що використовуються у будь-яких типах бізнесів та компаній, допомагаючи підвищити продуктивність, поліпшити взаємодію з клієнтами та зекономити час і ресурси. Грунтуючись на структурі та матеріально-технічному забезпеченні компанії, як правило потрібен вибір типу системи залежно від її надійності та функцій.

Обираючи вебсистему CRM для компанії, створюється можливість для додаткового розвитку та масштабування складних бізнес процесів. Рішення призначене для оптимізації та спрощення багатьох аспектів діяльності.

Незалежно від розміру компанії та галузі, впровадження CRM-системи може стати ключовою складовою економічного успіху, яке підвищить продуктивність бізнесу, зробить його більш конкурентоспроможним і забезпечить ефективне управління ресурсами.

## **1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

### **1.1 Вимоги та функціонал CRM системи**

Цей розділ присвячений визначенню ключових вимог та функціональних можливостей, які повинна виконувати розроблювана CRM система. Враховуючи потреби сучасних малих та середніх бізнесів, система повинна бути гнучкою та адаптивною до різних стилів управління та організаційних структур.

Приведемо основні вимоги до системи.

Простий та інтуїтивно зрозумілий інтерфейс: CRM система повинна мати зрозумілий користувацький інтерфейс, що знижує поріг входження для нових співробітників та сприяє швидкому ознайомленню з основним функціоналом.

Закритість системи: система повинна передбачати механізми авторизації та додавання нових користувачів через спеціальний відділ або уповноважених осіб з особливими правами доступу.

Ієрархічна структура підпорядкування: система повинна відображати організаційну структуру компанії у вигляді дерева підпорядкування, де кожен співробітник в залежності від своєї посади може мати підлеглих.

Інтерфейси перегляду профілів: система повинна надавати можливість перегляду профілів користувачів для ідентифікації та уточнення їхніх даних.

Функціонал створення та управління документами: система повинна дозволяти створювати, редагувати та управляти документами, включаючи вказівку терміновості, прикріплення файлів та визначення кола осіб, які повинні бути ознайомлені з документом.

Механізм призначення та виконання завдань: система повинна надавати можливість призначення завдань співробітникам з вказівкою термінів виконання та моніторингу їх статусу.

Функціональні особливості програмної реалізації наведенні нижче.

Автоматизація бюрократичних процесів: система повинна спрощувати та автоматизувати процеси створення, обігу та зберігання документів у компанії.

Комунікація та співпраця: система повинна сприяти ефективній комунікації між співробітниками, забезпечуючи обмін інформацією та спільну роботу над завданнями та документами.

Гнучкість та масштабованість: система повинна бути гнучкою для адаптації під різні потреби бізнесу та масштабованою для зростання разом з компанією.

Ці вимоги та функціональні особливості визначають основу для розробки CRM системи, яка буде відповідати потребам сучасних малих та середніх підприємств, сприяючи їх ефективному управлінню та розвитку.

## **1.2 CRM системи**

На ринку представлено багато комерційних реалізацій CRM систем для певних чи загальних задач у компанії.

Salesforce CRM (рис. 1.1) – система є однією з провідних платформ у світі CRM. Вона відома своєю гнучкістю та широким спектром функцій, що охоплює все від управління відносинами з клієнтами до аналітики, маркетингу, і навіть штучного інтелекту [1].

Основними функціями є управління лідами та можливостями, персоналізований маркетинг, продажі та обслуговування клієнтів, аналітика та звітність, штучний інтелект (Einstein AI).

Salesforce підходить для різних типів бізнесу, від малого до великого, і особливо ефективна для компаній, що шукають високо інтегровані рішення для управління всіма аспектами відносин з клієнтами. Також Salesforce має велику екосистему інтеграцій та сумісних додатків через свій AppExchange. Модель ціноутворення є масштабованою з різними планами для різних потреб

та розмірів бізнесу.

Перевагами системи виділяють високу налаштовуваність, широкий функціонал, і сильну інтеграцію, але натомість є і недоліки у вигляді відносно висої вартісті та складності у використанні для новачків [2].

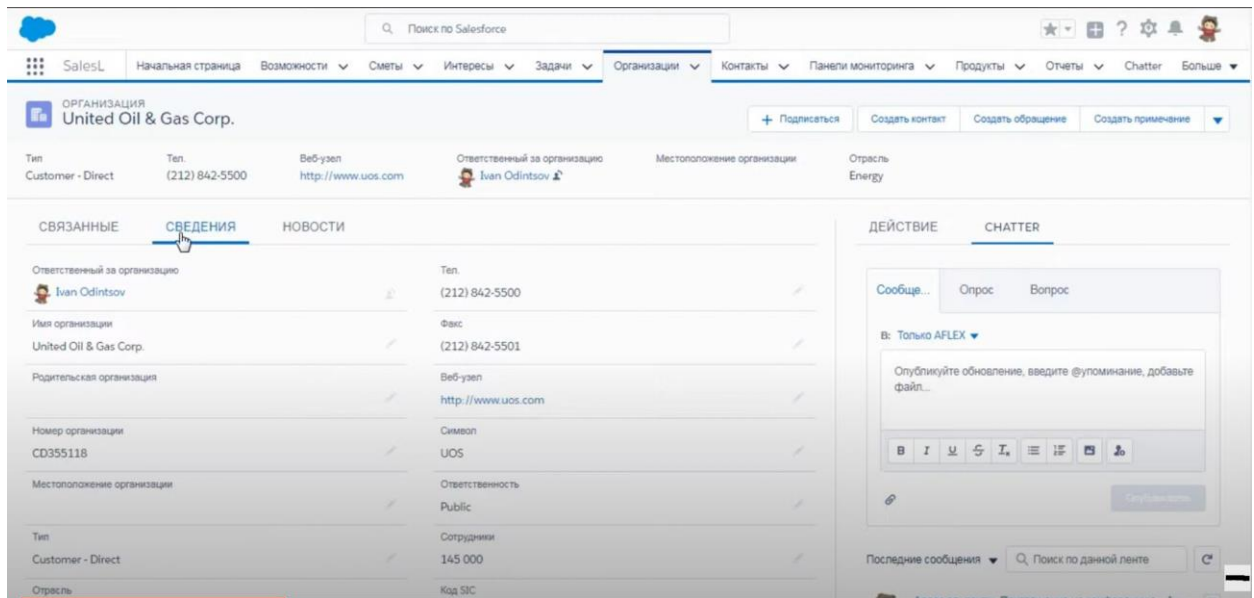


Рисунок 1.1 – CRM система Salesforce

HubSpot CRM (рис. 1.2) — є однією з найбільш популярних систем управління відносинами з клієнтами, яка славиться своєю легкістю використання та інтеграцією з іншими продуктами HubSpot для маркетингу, продажів та обслуговування клієнтів [3].

Основними функціями є управління контактами та лідами, електронна пошта та автоматизація маркетингу, аналітика продажів та звітність, інтеграція з соціальними мережами, вебформи та чат-боти. HubSpot CRM пропонує безкоштовний план з основними можливостями, а також декілька платних планів з розширеним функціоналом.

Перевагами системи є легкість використання, безкоштовний план для малих бізнесів, і сильну інтеграцію, недоліками же виділяють обмежений функціонал в безкоштовній версії та відносно високу вартість платних планів.

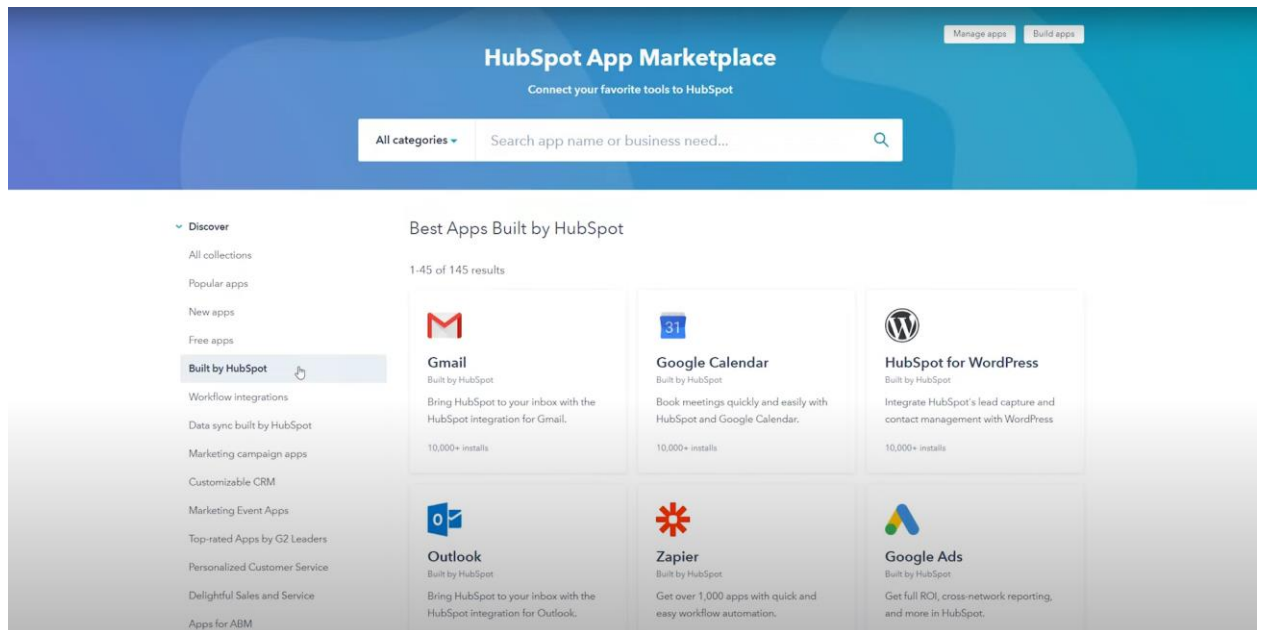


Рисунок 1.2 – CRM система HubSpot

Zoho CRM (рис. 1.3) — частина екосистеми бізнес-додатків Zoho і відрізняється своєю гнучкістю та доступністю, пропонуючи рішення для бізнесів будь-якого розміру. Zoho CRM підходить як для малих, так і для великих підприємств, особливо для тих, хто шукає масштабоване та налаштовуване рішення. Також воно інтегрується з багатьма іншими додатками Zoho, а також зі сторонніми інструментами та сервісами. Основними функціями системи є управління продажами та маркетингом, автоматизація робочих процесів, ведення клієнтської бази, аналітика та звітність, інтеграція з соціальними медіа. Перевагами системи є гнучкість, налаштовуваність, і доступність, недоліками же виділяють складність налаштування для певних бізнес-процесів та менш інтуїтивний інтерфейс [4].

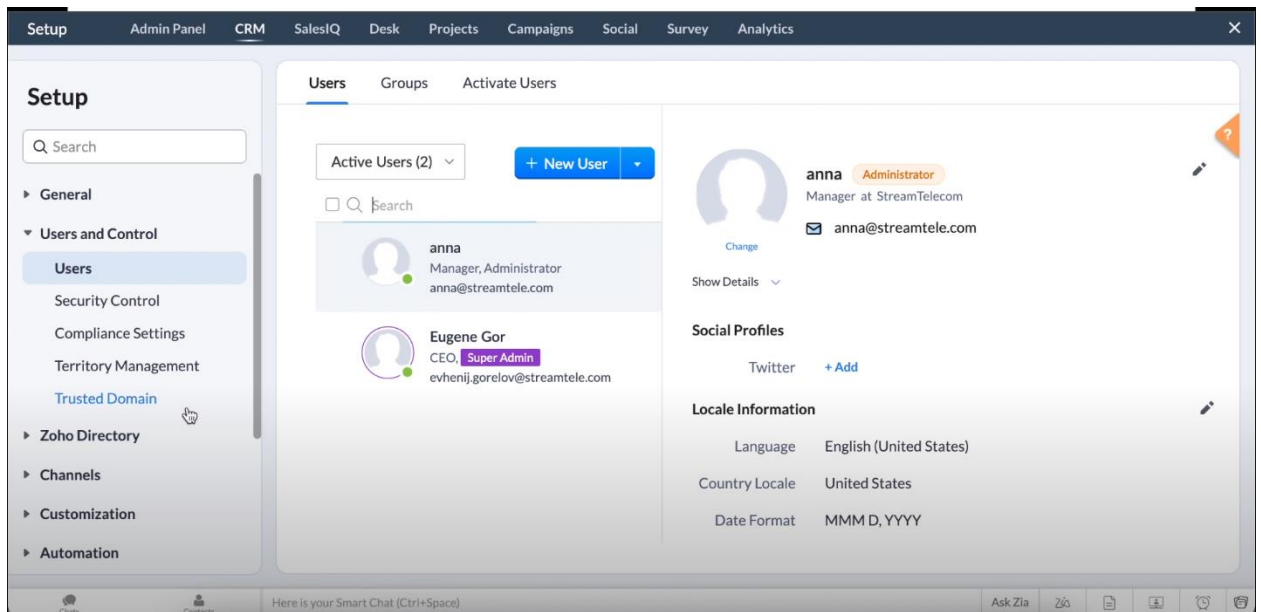


Рисунок 1.3 – CRM система Zoho

### 1.3 Висновки за розділом 1

У цьому розділі ми здійснили аналіз ринку та специфіки сфери, що дозволило нам визначити ключові напрямки для досягнення нашої цілі та розробки розподіленої CRM системи. Також були виокремлені основні засади дизайну та підтримки подібних систем, які будуть використані для удосконалення інтерфейсу користувачів та процесів адміністрування у нашому програмному продукті.

Завдання дипломної кваліфікаційної роботи полягають в наступному:

- аналіз предметної області;
- розробка концепії програмного забезпечення;
- створення програмного забезпечення.

## **2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ**

### **2.1 Функціональні вимоги до програмного забезпечення**

На основі аналізу аналогів та огляду загальноновживаних принципів побудови CRM систем в проєкті було виділено три ключові ролі, адміністратор, керуючий департаментом, працівник департаменту. Нижче наведено функціональні вимоги для кожної з цих ролей.

Сервіс повинен забезпечувати наступний функціонал адміністратора:

- можливість перегляду профілю іншого користувача;
- можливість створення, оновлення та видалення профілей користувачів;
- можливість зміни ролі користувача;
- можливість переналаштування системи підпорядковування користувачів в системі;
- можливість перегляду документів інших користувачів;
- можливість оновлення та видалення документів;
- можливість зміни прив'язки файлів до документів;
- можливість налаштування списку узгоджуючих документа;
- можливість залишати коментарі до документів;
- можливість перегляду задач інших користувачів;
- можливість створення, оновлення та видалення власних задач;
- можливість оновлення та видалення задач інших користувачів;
- можливість зміни прив'язки файлів до задач;
- можливість залишати коментарі до задач.

Функціональні вимоги керуючого департаментом:

- можливість перегляду власного профіля;
- перегляд профілей інших користувачей;
- можливість оновлення інформації власного профіля;
- можливість переналаштування системи підпорядковування користувачів у власному департаменті;

- можливість перегляду документів інших користувачів;
- можливість перегляду власних документів;
- можливість створення документів;
- можливість оновлення та видалення власних документів;
- можливість налаштування списку узгоджуваних документів

підпорядкованих користувачів;

- можливість залишати коментарі до документів;
- можливість узгодження документів;
- можливість оновлення та видалення документів підпорядкованих

користувачів;

- можливість перегляду задач інших користувачів;
- можливість створення задач;
- можливість оновлення та видалення власних задач;
- можливість оновлення та видалення задач підпорядкованих

користувачів.

Функціональні вимоги працівника департаменту включають:

- можливість перегляду власних документів;
- можливість оновлення та видалення власних документів;
- можливість створення документів;
- можливість перегляду призначених документів;
- можливість узгодження призначених документів;
- можливість залишати коментарі до призначених документів;
- можливість перегляду власних задач;
- можливість перегляду призначених задач;
- можливість оновлення та видалення власних задач;
- можливість виконувати призначені задачі;
- можливість перегляду власного профілю;
- можливість оновлення інформації власного профіля.

Адміністратор являється головною сутністю для налаштування та забезпечення ефективної роботи системи. Адміністратор має розширений

доступ до всієї інформації та процесів. Сценарій використання сервісу адміністратором указано на рис 2.1.



Рисунок 2.1 – Діаграма прецедентів адміністратора

Керуючий департаментом на відміну від адміністратора являється повноцінним співробітником що має обов'язки менеджера та керуючого у системі. У керуючого департаментом в залежності від логіки підпорядкування можуть знаходитись в управлінні інші співробітники компанії. Сценарій використання сервісу керуючим департаментом указано на рис. 2.2.

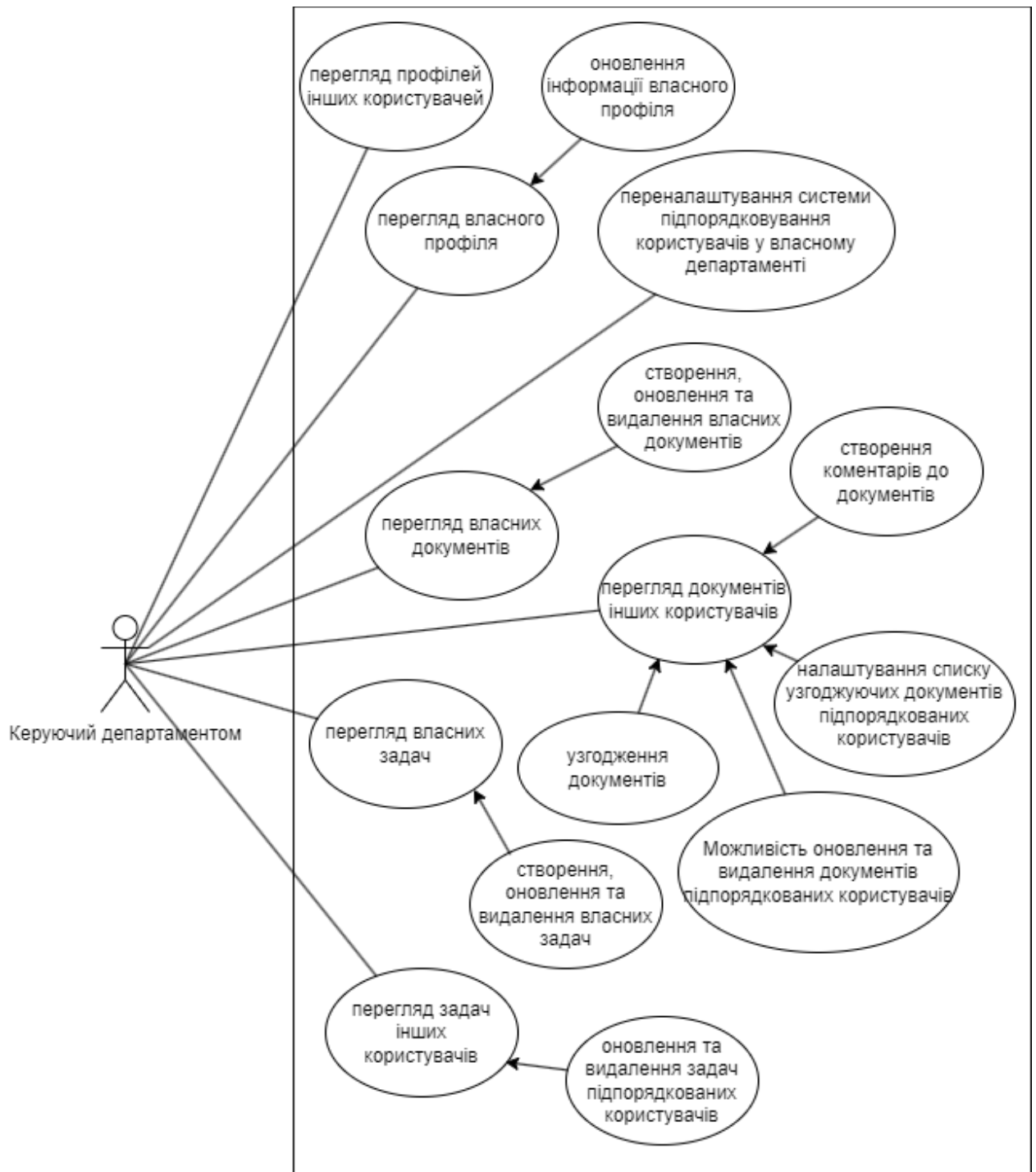


Рисунок 2.2 – Діаграма прецедентів керуючого департаментом

Працівник департаменту являється основною виконуючою сутністю. На відміну від керуючого департаментом, для цієї ролі не доступний основний керуючий функціонал системи, окрім основних обов'язків стосовно ознайомлення та виконання умов документів та задач. Сценарій використання

сервісу працівником указано на рис. 2.3.

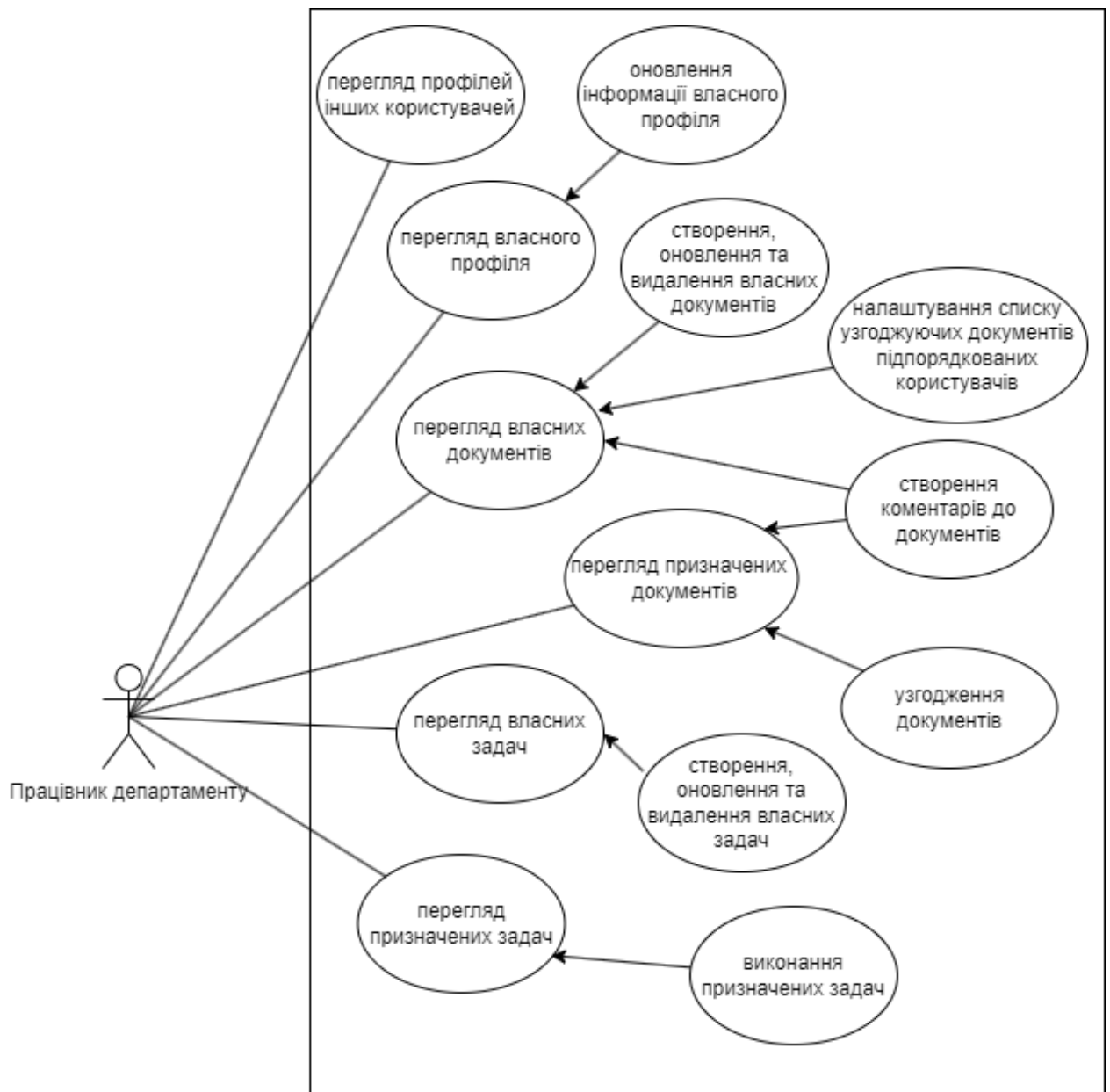


Рисунок 2.3 – Діаграма прецедентів працівника департаменту

## 2.2 Вибір засобів розробки програмного продукту

Перед тим, як розпочати розробку, я провів детальний аналіз доступних мов програмування та існуючих рішень, щоб вибрати найкращий варіант для серверної частини. Вибір припав на Python – високорівневу, універсальну та широко використовувану мову програмування. Python має широке

застосування: від веброзробки до машинного навчання, від передових технологічних проєктів до промислового програмного забезпечення.

Завдяки своїй популярності та високому рівню, Python став вибором багатьох технологічних лідерів, таких як Google, Amazon, Facebook, Instagram, Dropbox, Uber. Ця мова ідеально підходить як для початківців, так і для професіоналів, які вже мають досвід роботи з іншими мовами, такими як C++ чи Java.

Python підтримує об'єктно-орієнтоване та процедурне програмування, дозволяючи створювати компактні та читабельні програми. Його особливість вимоги до відступів значно спрощує читання коду. Однією з ключових переваг Python є велика бібліотека стандартних інструментів, що охоплює великий спектр областей: від машинного навчання до вебфреймворків, таких як Django та Flask, від обробки зображень до вебскребінгу, тестування, обробки мультимедіа, наукових обчислень та обробки тексту. Найпопулярнішими вебфреймворками, що використовуються для написання бекенду на мові Python, являються Flask та Django [5].

Flask – це мінімалістичний вебфреймворк, який надає основні інструменти для розробки вебдодатків. Його ключова особливість – відсутність прив'язки до зовнішніх бібліотек, що забезпечує простоту та гнучкість у використанні. Незважаючи на свою простоту, Flask пропонує розширення, що допомагають з перевіркою форм, об'єктно-реляційним мапінгом, системами аутентифікації, та надають інші корисні інструменти [6].

Django, навпаки, є більш комплексним вебфреймворком для Python, спрямованим на швидку та ефективну розробку вебсайтів. Цей фреймворк використовує стандартизований підхід до розробки, що сприяє створенню якісних вебдодатків і покращує продуктивність розробників. Django ідеально підходить для створення складних, керованих базами даних вебсайтів, забезпечуючи високий рівень функціональності та гнучкості [7].

Важливі різниці між цими бібліотеками зображені у таблиці 2.1.

Таблиця 2.1 – Порівняння Python фреймворків

| Характеристика                     | Flask                                     | Django                                             |
|------------------------------------|-------------------------------------------|----------------------------------------------------|
| 1                                  | 2                                         | 3                                                  |
| Тип фреймворку                     | Мікрофреймворк                            | Повноцінний фреймворк                              |
| Простота використання              | Висока (легше для новачків)               | Середня (може бути складніше для новачків)         |
| Залежності від зовнішніх бібліотек | Мінімальні                                | Багато вбудованих                                  |
| Вбудовані компоненти               | Обмежені (необхідні додаткові розширення) | Багато (адміністрування, ORM, аутентифікація тощо) |
| Підхід до розробки                 | Гнучкий, можливість вибору компонентів    | Опініоністичний, стандартизований підхід           |
| Підтримка баз даних                | Підтримка через розширення                | Потужна підтримка вбудованого ORM                  |
| Використання шаблонів              | Є, з використанням Jinja2                 | Є, з використанням Django Templates                |
| Форма розробки                     | Більш вільна, без строгих вимог           | Структурована, з дотриманням певних конвенцій      |
| Система адміністрування            | Відсутня                                  | Потужна система адміністрування                    |

Продовження таблиці 2.1

| 1                         | 2                          | 3                                       |
|---------------------------|----------------------------|-----------------------------------------|
| Розширення та плагіни     | Багато доступних розширень | Менше залежності від сторонніх плагінів |
| Виконання фонових завдань | Через додаткові розширення | Вбудовані інструменти                   |

Flask вирізняється своєю простотою, відкритістю коду та мінімалізмом у написанні програм. На відміну від цього, Django пропонує помірну швидкість розробки, має значну підтримку спільноти та відзначається легкістю в освоєнні. Виходячи з цих критеріїв, для конкретного проекту Django було визнано більш відповідним вибором, оскільки він пропонує баланс між функціональністю та зручністю розробки.

У контексті клієнтської частини проекту, було ухвалено рішення не використовувати спеціалізований фреймворк, а зосередитися на використанні Javascript – стандартної мови для розробки вебінтерфейсів. Цей підхід дозволяє забезпечити більш гнучку і універсальну взаємодію з користувачем, враховуючи широкі можливості Javascript у створенні динамічних вебсторінок.

JavaScript є важливою мовою програмування, яка вносить унікальні функціональні можливості як в клієнтську, так і в серверну частини вебдодатків. Ця мова служить ключем до створення інтерактивних вебсторінок, доповнюючи HTML та CSS, які забезпечують основну структуру та візуальний стиль. JavaScript приносить елементи взаємодії та динаміки, завдяки чому користувачі отримують захоплюючий вебдосвід [8].

Ця мова перетворює вебсторінки з простих статичних документів у живі, реагуючі на дії користувача середовища. JavaScript відіграє ключову роль у створенні вебдодатків, дозволяючи реалізувати складні функції, такі як анімації, асинхронні запити до сервера, обробка форм, ігри, та інші

інтерактивні можливості, що важко досягнути за допомогою інших технологій. Це робить JavaScript незамінною частиною сучасного вебдизайну та розробки.

Під час вибору системи управління базами даних були розглянуті два основних кандидати: PostgreSQL та MySQL. PostgreSQL відомий як об'єктно-реляційна система управління базами даних, яка включає унікальні можливості, такі як успадкування таблиць і перевантаження функцій, корисні для ряду специфічних задач. Важливо, що PostgreSQL дуже строго дотримується стандартів SQL, що може бути критично в деяких ситуаціях.

SQL, мова, яка лежить в основі обох систем, дозволяє з легкістю маніпулювати даними, об'єднуючи таблиці з мінімальними зусиллями та простою структурою. Це робить SQL доступним навіть для тих, хто не має глибоких технічних знань. Користувачам SQL не потрібно знати точне розташування даних на диску, щоб виконувати складні запити або аналізи.

Що стосується підтримки формату JSON (JavaScript Object Notation), який є важливим для обміну даними, обидві системи мають цю можливість. Однак, PostgreSQL вирізняється завдяки підтримці JSONB, двійкової версії JSON, яка оптимізує зберігання та обробку даних, зменшуючи повторення ключів і видаляючи непотрібні прогалини.

Обидві системи можуть похвалитися потужною підтримкою спільноти та традиційними механізмами підтримки, надаючи користувачам доступ до багатства ресурсів для ефективного використання даних, нових ідей та стратегій для прийняття рішень і виявлення нових бізнес-можливостей.

PostgreSQL вирізняється своєю універсальністю у підтримці як реляційних, так і нереляційних типів даних, що робить її однією з найбільш сумісних та надійних реляційних баз даних на сьогоднішній день. Її здатність ефективно обробляти складні запити робить її ідеальним вибором для різноманітних застосувань.

Ключові функції PostgreSQL включають:

- відновлення на час, яке дозволяє відновлювати базу даних до певного

моменту в часі;

- передбачувальний журнал змін, який реєструє всі зміни в базі даних для гарантування цілісності даних;

- збережені процедури, що дозволяють користувачам створювати та використовувати власні підпрограми для розширення функціональності.

Головні переваги PostgreSQL включають:

- високу продуктивність та масштабованість, з можливістю обробки геопросторових даних та забезпечення паралелізму;

- підтримку паралельного виконання операцій читання та запису, що покращує ефективність обробки даних;

- глибоку підтримку мов програмування, включаючи Python, Java, JavaScript, C/C++, та Ruby, що забезпечує велику сумісність;

- високу доступність та безперервність бізнесу через різні методи реплікації між серверами;

- велику гнучкість та економічну ефективність завдяки відкритому коду, що сприяє інноваціям та адаптації під специфічні потреби користувачів.

PostgreSQL, завдяки цим характеристикам, стає незамінним інструментом в світі сучасного управління даними, забезпечуючи надійність, безпеку та гнучкість для широкого спектру застосувань [9].

MySQL є швидкою, ефективною та легкою у використанні реляційною системою управління базами даних з відкритим кодом, яка ідеально підходить для високонавантажених виробничих додатків. Ця система, що відрізняється низьким споживанням ресурсів, як пам'яті, так і процесорного часу, відома своєю простотою в налаштуванні та експлуатації. MySQL Community Edition пропонує безкоштовне рішення, підтримуване активною онлайн-спільнотою [10].

Переваги MySQL на ринку включають:

- високий рівень безпеки даних, що ставить її в авангарді серед систем управління базами даних, особливо для вебдодатків, де цілісність даних має вирішальне значення;

- чудову продуктивність завдяки ефективній структурі механізму зберігання, яка включає швидкісні часткові індекси, повнотекстові індекси та оптимізовані кеші пам'яті для забезпечення високої продуктивності;
- масштабованість, дозволяючи збільшувати обсяги сховища без значних збільшень у розмірі;
- гнучкість та відкритий код, що підтримує обробку транзакцій, хоча і може бути менш гнучкою у порівнянні з нереляційними системами, такими як NoSQL.

Таким чином, MySQL представляє собою рішення, що забезпечує баланс між швидкістю, безпекою, масштабованістю та гнучкістю, що робить її привабливою для широкого спектру додатків, від малих стартапів до великих корпорацій [11].

Функціонал вебсервісу включає в себе забезпечення управління завданнями працівників, кожне з яких має визначений термін виконання. Основна мета це організувати ефективний контроль і відстеження робочих процесів, включаючи розподіл завдань, моніторинг статусів виконання та дотримання термінів. Це вимагає ефективного впровадження механізму для запланованого та асинхронного виконання завдань, що дозволяє оптимізувати робочі процеси та забезпечити дотримання встановлених термінів. Були розглянуті наступні технології для реалізації цього функціоналу:

- Celery відомий своєю високою масштабованістю, яка підтримує широкий спектр задач і взаємодіє з різними брокерами повідомлень, такими як RabbitMQ та Redis. Ця бібліотека є дуже гнучкою, дозволяючи використовувати її для різноманітних задач, від простих до дуже складних, а також надає можливість планувати виконання завдань. Проте, Celery може бути складним у налаштуванні та інтеграції, а також вимагати додаткових ресурсів сервера, особливо при роботі з великими обсягами задач.

- Redis Queue. RQ простіший у використанні та налаштуванні порівняно з Celery, особливо якщо ви вже використовуєте Redis у вашому проекті. Це більш легкий та прямолінійний варіант для асинхронного виконання завдань.

Однак, RQ не такий масштабований та гнучкий як Celery і не має деяких розширених функцій, доступних у більш складних системах [12];

- Huey є простим у налаштуванні та інтеграції, працюючи з різними типами брокерів, включаючи Redis, SQLite та in-memory storage. Він також підтримує планування задач. Однак, Huey може не відповідати потребам дуже складних або великих проєктів, та має меншу спільноту порівняно з Celery, що може ускладнити пошук ресурсів та підтримки [13].

Після детального аналізу доступних опцій для асинхронної обробки завдань у моєму Django сервісі, я вирішив вибрати Celery. Вибір Celery був зроблений через його надзвичайну масштабованість і гнучкість, які важливі для мого проєкту. Celery дозволяє ефективно управляти різноманітними задачами, які мій сервіс має виконувати, включаючи відстеження термінів виконання завдань працівниками. Ця система добре підходить для складних завдань, які можуть виникати у моєму проєкті, та пропонує потужні інструменти для планування та управління виконанням задач [14].

Celery також відзначається своєю здатністю інтегруватися з різними брокерами повідомлень. Для мого проєкту я вирішив використовувати Redis як брокер повідомлень для Celery. Вибір Redis обумовлений його високою продуктивністю, надійністю та легкістю використання. Redis відомий своєю швидкістю та ефективністю при обробці великої кількості запитів, що є критично важливим для забезпечення швидкої та ефективної обробки задач в моєму сервісі. Також Redis має хорошу підтримку у спільноті розробників, що спрощує процес налаштування та управління.

Крім того, використання Redis разом з Celery дає додаткові переваги, такі як здатність швидко обробляти дані в режимі реального часу та легкість масштабування. Це забезпечує високу гнучкість та ефективність у вирішенні задач моєї системи управління [15].

Отже, комбінація Celery та Redis в моєму Django проєкті є оптимальним рішенням, яке забезпечує потужний механізм для асинхронної обробки задач з високим рівнем продуктивності та масштабованості.

## 2.3 Проєктування бази даних

Схема бази даних складається з 14 сутностей, пов'язаних 14 зв'язками.

Сутність User (Користувач). Представляє профіль користувача в системі. Має наступні поля: id (ключ), username (ім'я користувача), first\_name (ім'я), last\_name (прізвище), email (електронна пошта), password (пароль), phone (телефонний номер), department (відділ, ключ на сутність Department), position (посада), subordinates (підлеглі, зв'язок багато до багатьох з іншими користувачами).

Сутність Department (Відділ). Представляє відділ у компанії. Має наступні поля: id (ключ), name (назва відділу).

Сутність Document (Документ). Представляє документ у системі. Має наступні поля: id (ключ), title (назва), text (текст документу), urgency\_tag (мітка терміновості), employees (список користувачів для ознайомлення, зв'язок багато до багатьох з User), created\_at (дата створення), created\_by (автор, ключ на User), status (статус документу).

Сутність DocumentComment (Коментар до Документа). Представляє коментар до документа. Має наступні поля: id (ключ), document (документ, ключ на Document), author (автор, ключ на User), text (текст коментаря), created\_at (дата створення).

Сутність Task (Задача). Представляє задачу в системі. Має наступні поля: id (ключ), title (назва), text (опис задачі), urgency\_tag (мітка терміновості), employees (список користувачів для виконання, зв'язок багато до багатьох з User), created\_at (дата створення), created\_by (автор, ключ на User), status (статус задачі).

Сутність TaskComment (Коментар до Задачі). Представляє коментар до задачі. Має наступні поля: id (ключ), task (задача, ключ на Task), author (автор, ключ на User), text (текст коментаря), created\_at (дата створення).

Сутність TaskCompletion (Виконання Задачі). Представляє інформацію

про виконання задачі користувачами. Має наступні поля: `id` (ключ), `task` (задача, ключ на `Task`), `user` (користувач, ключ на `User`), `completed` (флаг виконання).

Сутність `users_user_groups`. Системна таблиця Django для зв'язку користувачів з групами. Має поля: `id` (ключ), `user_id` (ключ на `User`), `group_id` (ключ на `auth_group`).

Сутність `auth_group`. Системна таблиця Django для груп користувачів. Має поля: `id` (ключ), `name` (назва групи).

Сутність `auth_group_permissions`. Системна таблиця Django для зв'язку груп з дозволами. Має поля: `id` (ключ), `group_id` (ключ на `auth_group`), `permission_id` (ключ на `auth_permission`).

Сутність `auth_permission`. Системна таблиця Django для дозволів. Має поля: `id` (ключ), `name` (назва дозволу), `content_type_id` (ключ на `content_type`), `codename` (кодове ім'я дозволу).

Таблиця `documents_document_employees`. Таблиця для зберігання зв'язків між документами та користувачами, які мають ознайомитися з цими документами. Має поля: `id` (ключ), `document_id` (ключ на `Document`), `user_id` (ключ на `User`).

Таблиця `tasks_taskcompletion`. Таблиця для зберігання інформації про виконання задач користувачами. Має поля: `id` (ключ), `task_id` (ключ на `Task`), `user_id` (ключ на `User`), `completed` (флаг виконання).

Таблиця `tasks_task_employees`. Таблиця для зберігання зв'язків між задачами та користувачами, які мають виконати ці задачі. Має поля: `id` (ключ), `task_id` (ключ на `Task`), `user_id` (ключ на `User`).

Таблиця `users_user_subordinates`. Таблиця для зберігання зв'язків між користувачами та їх підлеглими. Має поля: `id` (ключ), `user_id` (ключ на `User`, який є керівником), `subordinate_id` (ключ на `User`, який є підлеглим).

Схема БД представлена на рис. 2.5.

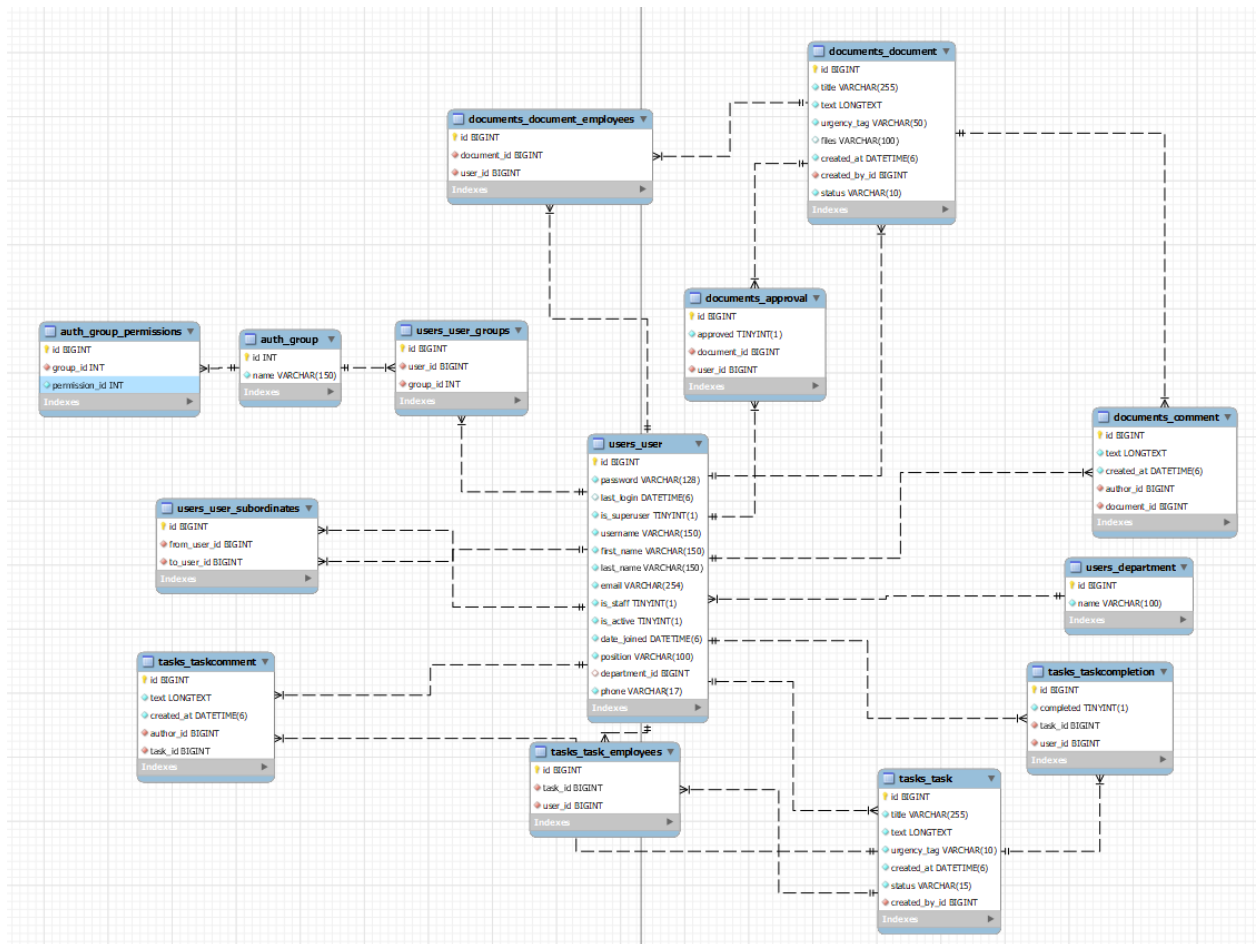


Рисунок 2.5 – Схема БД

Зв'язок між User та Department. Один-до-багатьох, оскільки один відділ може мати багатьох співробітників, але кожен співробітник належить лише до одного відділу.

Зв'язок між User та Document (автор). Один-до-багатьох, оскільки один користувач може створити багато документів, але кожен документ має лише одного автора.

Зв'язок між User та Task (автор). Один-до-багатьох, аналогічно до зв'язку між User та Document.

Зв'язок між Document та User (employees). Багато-до-багатьох, реалізований через допоміжну таблицю documents\_document\_employees, оскільки один документ може бути призначений для ознайомлення багатьом користувачам, і один користувач може ознайомлюватися з багатьма документами.

Зв'язок між Task та User (employees). Багато-до-багатьох, реалізований через допоміжну таблицю tasks\_task\_employees, оскільки одна задача може бути призначена для виконання багатьом користувачам, і один користувач може виконувати багато задач.

Зв'язок між User та DocumentComment. Один-до-багатьох, оскільки один користувач може залишити багато коментарів до документів, але кожен коментар має лише одного автора.

Зв'язок між Document та DocumentComment. Один-до-багатьох, оскільки до одного документа може бути багато коментарів.

Зв'язок між User та TaskComment. Один-до-багатьох, аналогічно до зв'язку між User та DocumentComment.

Зв'язок між Task та TaskComment. Один-до-багатьох, аналогічно до зв'язку між Document та DocumentComment.

Зв'язок між Task та TaskCompletion. Один-до-багатьох, оскільки одна задача може мати багато записів про виконання від різних користувачів.

Зв'язок між User та TaskCompletion. Один-до-багатьох, оскільки один користувач може виконувати багато задач.

Зв'язок між User та users\_user\_groups. Багато-до-багатьох, реалізований через допоміжну таблицю, оскільки один користувач може належати до багатьох груп, і одна група може включати багатьох користувачів.

Зв'язок між auth\_group та auth\_group\_permissions. Багато-до-багатьох, реалізований через допоміжну таблицю, оскільки одна група може мати багато дозволів, і один дозвіл може бути призначений багатьом групам.

Зв'язок між User та User (subordinates). Багато-до-багатьох, реалізований через допоміжну таблицю users\_user\_subordinates, оскільки один користувач може мати багатьох підлеглих, і один користувач може бути підлеглим багатьох керівників.

## 2.4 Висновки за розділом 2

У цій частині було визначено основні функціональні вимоги для створення програмного продукту. Для розробки було вибрано Python як основну мову програмування, Django як фреймворк для розробки веб-інтерфейсів, а MySQL використовувався як система управління базами даних. Також було вирішено використовувати бібліотеку Celery для обробки фонових задач і Redis як брокер повідомлень. Крім цього, була розроблена структура бази даних для проєкту.

## **3 РОЗРОБКА ПРОГРАМИ**

### **3.1 Алгоритм функціонування програмного забезпечення**

Функціонування сервісу варіюється в залежності від ролі, яку виконує користувач у системі. Кожна роль, будь то адміністратор, керівник департаменту чи співробітник департаменту, має свої унікальні можливості та обмеження.

Головна сторінка сервісу служить вступним інтерфейсом для користувачів, надаючи загальну інформацію про систему та її можливості. Вона доступна для всіх відвідувачів, незалежно від їх статусу авторизації.

Функціонал головної сторінки включає:

- відображення назви проекту та логотипу;
- короткий опис сервісу;
- для авторизованих користувачів – відображення імені, прізвища та посади;
- навігаційна панель з посиланнями на інші розділи сервісу.

Сторінка авторизації дозволяє користувачам увійти в систему, використовуючи свій телефонний номер та пароль.

Функціонал сторінки авторизації:

- поля для введення телефону та пароля;
- автоматичне доповнення поля телефону префіксом "+380";
- кнопка для входу в систему.

Сторінка користувачів надає інформацію про всіх учасників системи, їх посади та відділи.

- функціонал сторінки користувачів:
- перелік всіх користувачів з інформацією про їх посади та відділи;
- можливість перегляду дерева підпорядкування;
- для користувачів з правами співробітника департаменту – можливість додавання та видалення підпорядкованих співробітників.

На цій сторінці організаційної структури користувачі можуть переглядати організаційну структуру компанії, включаючи інформацію про підпорядкування.

Функціонал сторінки організаційної структури:

- перелік підпорядкованих співробітників;
- перелік співробітників, яким користувач підпорядковується;
- для користувачів з правами співробітника департаменту – інтерфейс для додавання та видалення підпорядкованих співробітників.

Сторінка Документів надає доступ до всіх документів у системі, дозволяючи користувачам переглядати, створювати та коментувати їх.

Функціонал сторінки документів:

- перелік усіх документів з основною інформацією;
- відображення статусу документів;
- можливість переходу на сторінку з детальною інформацією про документ;
- кнопка для створення нового документа.

На сторінці деталей документа користувачі можуть переглядати всю інформацію про конкретний документ, включаючи текст, коментарі та статус.

Функціонал сторінки деталей документа:

- повний текст документа;
- відображення коментарів до документа;
- форма для додавання нового коментаря;
- для користувачів, вказаних у документі, кнопка для затвердження документа.

Сторінка задач дозволяє користувачам переглядати, створювати та відслідковувати виконання задач.

Функціонал сторінки задач:

- перелік усіх задач з основною інформацією;
- відображення статусу задач;
- можливість переходу на сторінку з детальною інформацією про

задачу;

- кнопка для створення нової задачі.

На сторінці деталей задачі користувачі можуть переглядати всю інформацію про конкретну задачу, включаючи текст, коментарі та статус.

Функціонал сторінки деталей задачі:

- повний текст задачі;
- відображення коментарів до задачі;
- форма для додавання нового коментаря;
- для користувачів, вказаних у задачі, кнопка для позначення задачі як виконаної.

Схема функціонування сервісу наведена на рис. 3.1.

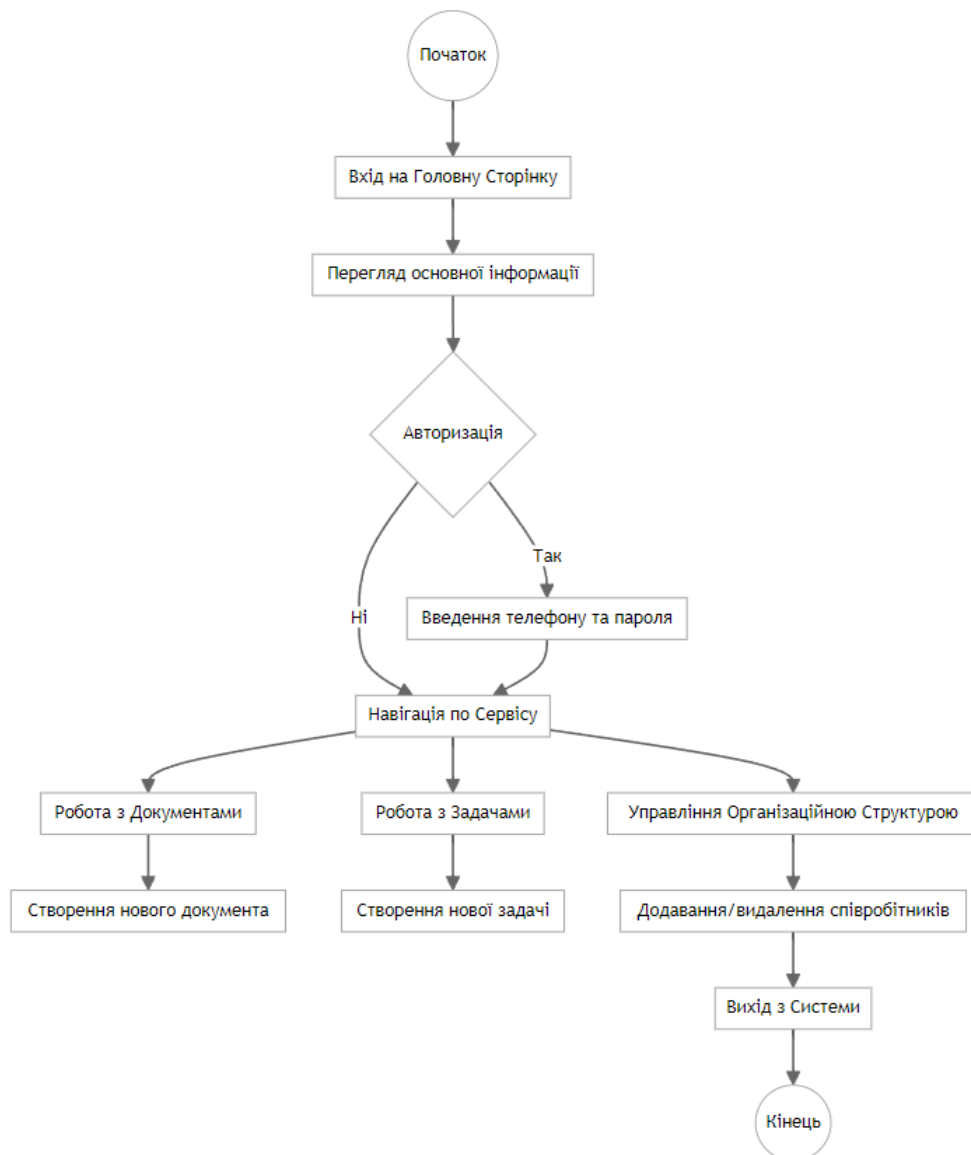


Рисунок 3.1 – Схема функціонування сервісу

## 3.2 Структура реалізованого програмного забезпечення

Структура відображає модульний підхід до розробки вебсервісу "CRMS", де кожен додаток відповідає за свою частину функціональності, спрощуючи розширення та підтримку системи.

### 1. Apps (Додатки Сервісу)

Сервіс складається з декількох додатків, кожен з яких відповідає за певний аспект функціональності.

`users` – Управління користувачами, авторизація, реєстрація, профілі користувачів.

`tasks` – Управління задачами, включаючи створення, перегляд, коментування та відмітка про виконання.

`documents` – Управління документами, включаючи створення, перегляд, коментування та процес затвердження.

Кожен додаток містить свої файли `models.py`, `views.py`, `urls.py`, `forms.py` тощо, які визначають логіку, базу даних, маршрутизацію та форми відповідно.

### 2. Celery (Асинхронні Завдання)

Celery використовується для обробки асинхронних завдань, таких як відправка повідомлень про закінчення терміну виконання задач.

`celery.py` – Конфігурація Celery для проекту.

`tasks.py` у кожному додатку – Файли з асинхронними завданнями Celery.

### 3. Static (Статичні Файли та Ресурси)

Цей розділ містить статичні файли, такі як CSS, JavaScript, зображення, які використовуються для стилізації та функціональності веб-інтерфейсу.

`css/` – Каталог з CSS-файлами.

`js/` – Каталог з JavaScript-файлами.

`images/` – Каталог з графічними ресурсами.

### 4. Templates (Шаблони Вебсторінок)

Цей розділ містить HTML-шаблони для кожної сторінки веб-сервісу.

`base.html` – Базовий шаблон.

Шаблони для конкретних сторінок кожного додатку, такі як `login.html`, `register.html`, `document_list.html`, `task_list.html` тощо.

#### 5. Project Root (Кореневий Каталог Проекту)

Кореневий каталог проекту містить конфігураційні файли та загальні ресурси.

`settings.py` – Налаштування проекту Django.

`urls.py` – Головний файл маршрутизації для всього проекту.

`wsgi.py` та `asgi.py` – Файли для розгортання проекту.

#### 6. Additional Components (Додаткові Компоненти)

Цей розділ може включати додаткові компоненти, такі як:

`utils.py` – Утиліти та допоміжні функції.

`middleware.py` – Проміжне програмне забезпечення.

Схематичне представлення структури програмного забезпечення представлено на рис. 3.2.

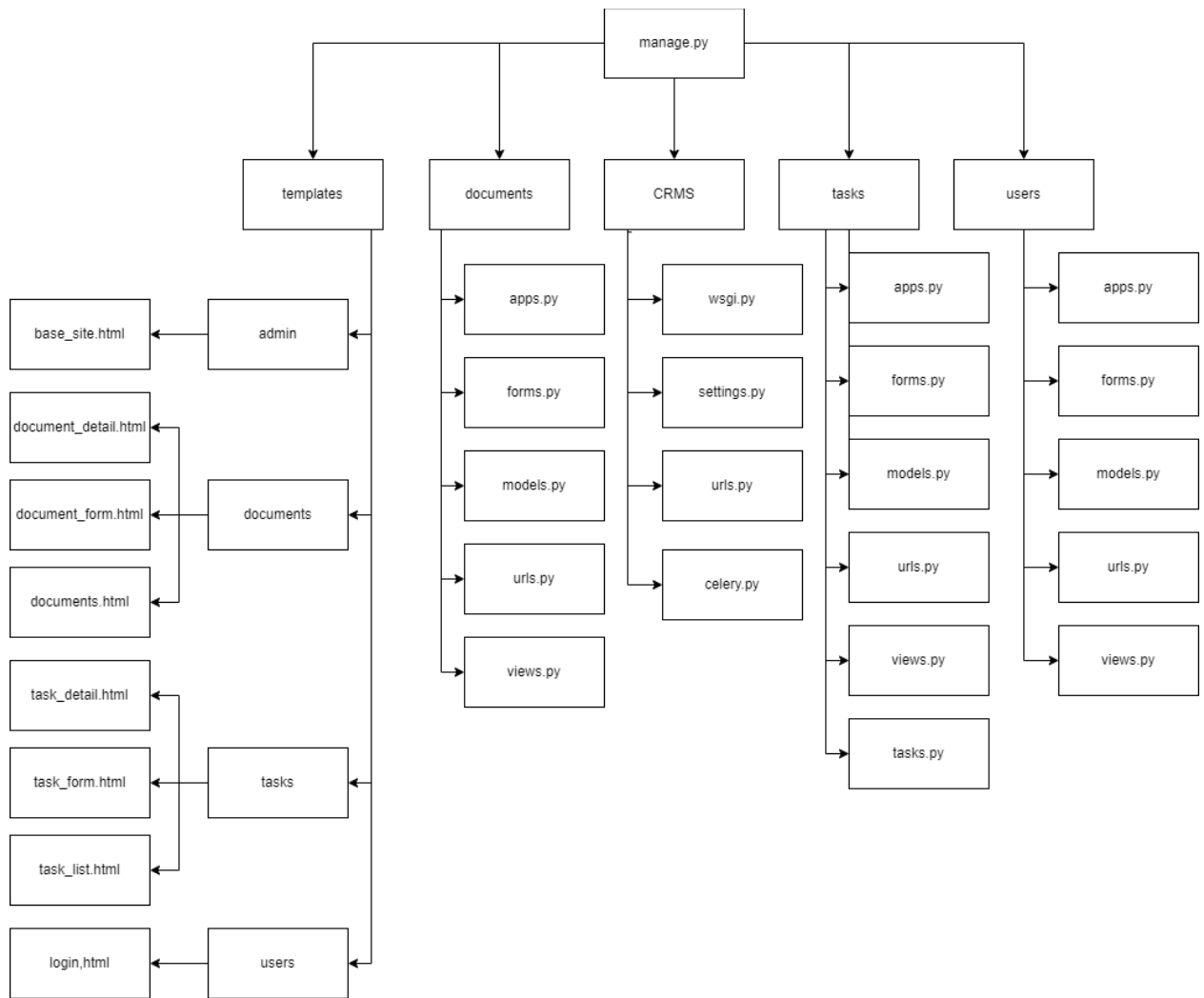


Рисунок 3.2 – Структурна схема програмного забезпечення

### 3.3 Опис прийнятих під час реалізації рішень

Ресурс: Головна сторінка (/)

GET: Повертає головну сторінку сервісу з інформацією про проект, логотипом та, у випадку авторизації користувача, його особистими даними (ім'я, прізвище, посада).

Ресурс: Авторизація (/login)

GET: Повертає сторінку для авторизації з полями для вводу телефону та пароля.

POST: Обробляє дані авторизації, перевіряє коректність введених даних та здійснює вхід користувача в систему.

Ресурс: Організаційна структура (/org)

GET: Показує списки підлеглих та керівників для поточного користувача.

POST: Дозволяє користувачам з правами is\_staff додавати або видаляти підлеглих.

Ресурс: Документи (/documents)

GET: Повертає список усіх документів з основною інформацією про кожен.

POST: Використовується для створення нових документів.

Ресурс: Створення Документа (/create\_document)

GET: Повертає форму для створення нового документа.

POST: Обробляє дані форми та створює новий документ.

Ресурс: Деталі Документа (/documents/<id>)

GET: Показує детальну інформацію про документ, включаючи коментарі та статус затвердження.

POST: Використовується для додавання коментарів до документа або зміни його статусу.

Ресурс: Задачі (/tasks)

GET: Повертає список усіх задач з основною інформацією про кожну.

POST: Використовується для створення нових задач.

Ресурс: Створення Задачі (/create\_task)

GET: Повертає форму для створення нової задачі.

POST: Обробляє дані форми та створює нову задачу.

Ресурс: Деталі Задачі (/tasks/<id>)

GET: Показує детальну інформацію про задачу, включаючи коментарі та статус виконання.

POST: Використовується для додавання коментарів до задачі або зміни її статусу.

Ресурс: Адміністративна Панель (/admin/)

GET: Надає доступ до адміністративної панелі Django, де можна керувати користувачами, документами, задачами та іншими сутностями системи.

Ресурс: Видалення Підлеглого (/remove-subordinate/)

POST: Дозволяє користувачам з правами `is_staff` видаляти підлеглих зі свого списку.

Ресурс: Вихід з Системи (`/logout/`)

GET: Виконує вихід користувача з системи та перенаправляє на головну сторінку або сторінку авторизації.

В рамках проекту було інтегровано використання Celery, яке є потужним інструментом для асинхронного виконання задач у фоновому режимі. Це дозволяє виконувати складні або тривалі операції без блокування основного потоку веб-додатку, підвищуючи його продуктивність та швидкість відгуку, за допомогою чого було реалізовано автоматичне сповіщення користувачів про закінчення терміну виконання задач у системі.

### **3.4 Висновки за розділом 3**

У даному розділі був розроблений та впроваджений алгоритм роботи програмного забезпечення. Розроблено архітектуру веб-сервісу та детально розглянуто функції кожного компонента системи. Окрім того, надано опис класів та функцій, які вони виконують.

## **4 ЕКСПЛУАТАЦІЯ РОЗРОБЛЕНОЇ ПРОГРАМИ**

### **4.1 Призначення програми**

Програма була розроблена як комплексне рішення для управління документообігом та задачами в компанії. Цей сервіс надає широкий спектр інструментів для ефективної організації робочих процесів, включаючи створення, управління та відстеження документів і задач, а також управління структурою підпорядкування співробітників. Інтеграція з Celery дозволяє автоматизувати нагадування про терміни виконання задач, підвищуючи ефективність роботи персоналу.

Серед ключових можливостей програми - управління користувачами, включаючи їх реєстрацію, авторизацію та визначення ролей у структурі компанії. Програма дозволяє створювати документи з різними рівнями екстреності, призначати відповідальних осіб для їх утвердження, а також відслідковувати статуси виконання задач. Важливою особливістю є можливість коментування документів та задач, що сприяє підвищенню комунікації та співпраці між співробітниками.

### **4.2 Умови виконання програми**

Вимоги до сервера включають:

- не менше 1,5 Гб вільного місця на жорсткому диску;
- мінімум 4 Гб оперативної пам'яті;
- процесор з 4 ядрами та частотою 2,8 ГГц.

Для коректної роботи вебсервісу на боці користувача необхідний оновлений веб браузер, який підтримує підключення до Інтернету.

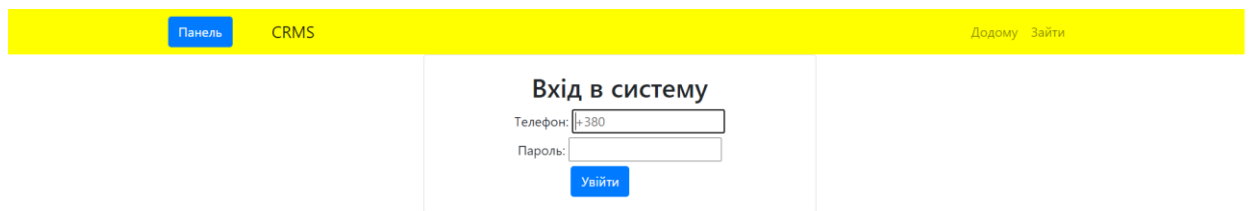
### **4.3 Виконання програми**

Сторінка входу є важливою частиною вебсервісу, вона дозволяє

користувачам, які вже мають обліковий запис, увійти в систему, використовуючи свій номер телефону та пароль. Ця сторінка має простий та інтуїтивно зрозумілий інтерфейс. На верхній частині сторінки розміщено назву сервісу, що допомагає користувачам ідентифікувати, що вони знаходяться на правильній сторінці.

Користувачі вводять свій номер телефону. Поле автоматично доповнюється кодом країни (+380). Поле для введення пароля, який був встановлений під час реєстрації.

Після введення даних, користувачі натискають кнопку «Увійти» для входу в систему (рис. 4.1).



The image shows a web interface for a login page. At the top, there is a yellow horizontal bar containing a blue button labeled 'Панель', the text 'CRMS', and two links: 'Додому' and 'Зайти'. Below this bar, the main content area is white. It features a centered box with the title 'Вхід в систему'. Inside this box, there are two input fields: 'Телефон:' with a pre-filled value of '+380' and 'Пароль:'. Below the password field is a blue button labeled 'Увійти'.

Рисунок 4.1 – Сторінка Входу (login/)

Головна сторінка є центральним вузлом вебсервісу, надаючи користувачам загальний огляд системи та доступ до ключових функцій. Ця сторінка вітає користувачів та надає корисну інформацію про сервіс.

Центральна частина сторінки включає логотип та назву сервісу, що допомагає користувачам ідентифікувати платформу. Під логотипом розміщено короткий опис сервісу, який висвітлює його основні функції та переваги. Для авторизованих користувачів відображається їх ім'я, прізвище та посада. Сторінка може включати виїзду панель з посиланнями на різні функції CRM-системи, доступні для користувача.

Залежно від прав доступу, користувачі можуть переходити до різних функцій CRM-системи, таких як управління документами, задачами та

організаційною структурою (рис. 4.2).

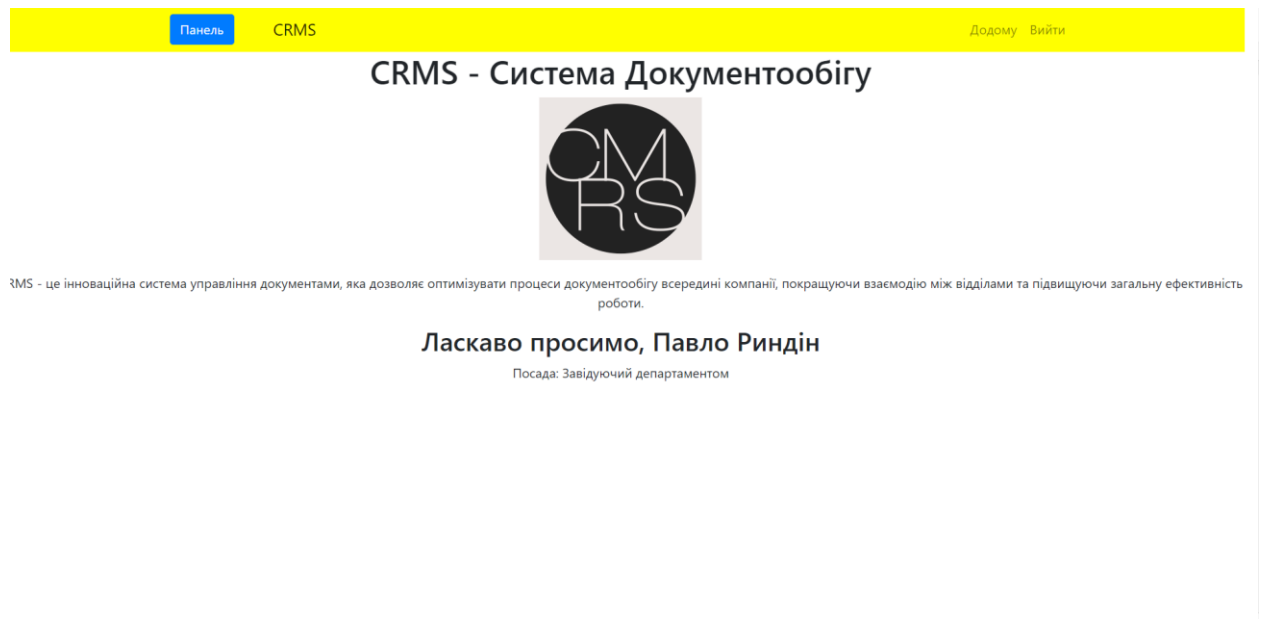


Рисунок 4.2 – Головна Сторінка (/)

Сторінка "Організаційна Структура" є важливим компонентом CRM-системи, оскільки вона відображає ієрархічну структуру компанії та взаємозв'язки між співробітниками. Ця сторінка дозволяє користувачам переглядати своїх підлеглих та керівників, а також управляти структурою підпорядкування. Користувачі можуть ознайомитися зі своїми підлеглими та керівниками. Користувачі з відповідними правами можуть додавати або видаляти співробітників зі свого списку підлеглих. Форми та інтерфейси на сторінці забезпечують зручне та інтуїтивно зрозуміле управління структурою підпорядкування (рис. 4.3).

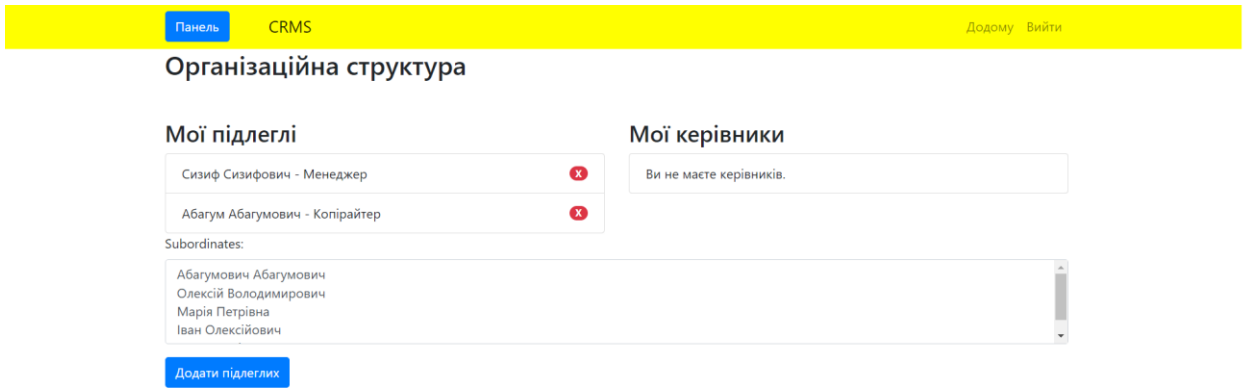


Рисунок 4.3 – Сторінка Організаційної Структури (org/)

Сторінка "Список Документів" є центральним місцем для управління та перегляду документів у CRM-системі. Вона дозволяє користувачам переглядати всі документи, до яких у них є доступ, та взаємодіяти з ними. Користувачі можуть переглядати список доступних документів та отримувати інформацію про кожен з них. На сторінці також користувачі можуть легко перейти до створення нового документа (рис. 4.4).

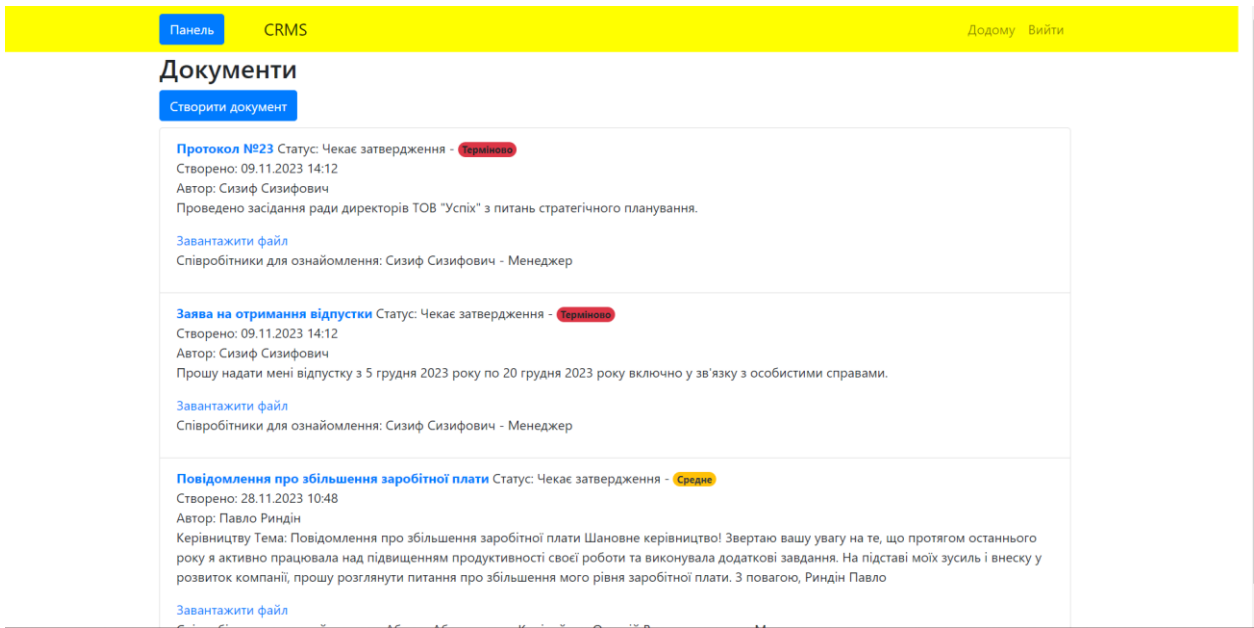


Рисунок 4.4 – Сторінка Списку Документів (documents/)

Сторінка "Створення Документа" в CRM-системі дозволяє користувачам створювати нові документи, які можуть бути використані для різних цілей в рамках організації. Ця сторінка є ключовою для додавання нового контенту та інформації в систему. Форма Створення Документа включає поля для введення назви документа, тексту, вибору тегу екстренності, прикріплення файлів та вибору співробітників для ознайомлення.

Поле Вибору Тегу Екстренності дозволяє вибрати один з трьох рівнів екстренності (срочно, середньо, не срочно). Після заповнення всіх полів, користувач може натиснути цю кнопку для створення документа (рис. 4.5).

**Створити документ**

Title:

Договір про співпрацю

Text:

Між ТОВ "Ефективність" та фірмою "Успіхова Інновація"

Предмет договору: Надання консультаційних послуг у сфері маркетингу та реклами.

Термін дії: 31 грудня 2023 року по 1 грудня 2024 року.

Строки та умови оплати: Оплата за надані послуги здійснюється щомісячно не пізніше 5 числа наступного місяця після надання послуги.

Urgency tag:

Терміново

Files: Выберите файл Скановані...писом.pdf

Employees:

« Іван Олексійович » « Анна Сергіївна »

Створити документ

Рисунок 4.5 – Сторінка Створення Документа (create\_document/)

Сторінка "Створення Задачі" у CRM-системі дозволяє користувачам створювати нові задачі, які можуть бути використані для координації та управління робочими процесами в організації. Ця сторінка є ключовою для планування та розподілу завдань серед співробітників. Форма Створення Задачі включає поля для введення назви задачі, опису, вибору тегу екстренності, встановлення дедлайну та вибору співробітників для виконання.

Поле Вибору Тегу Екстренності дозволяє вибрати один з трьох рівнів екстренності (терміново, середньо, не терміново). Поле для встановлення

дедлайну дозволяє вказати кінцевий термін виконання задачі. Вибір співробітників для виконання є мультиселектом з можливістю пошуку для вибору співробітників, які повинні виконати задачу. Після заповнення всіх полів, користувач може натиснути цю кнопку для створення задачі (рис. 4.6).

The screenshot shows a web interface for creating a task. At the top, there is a yellow header bar with a blue 'Панель' button, the text 'CRMS', and links 'Додому' and 'Вийти'. Below the header, the title 'Створити задачу' is displayed. The form consists of several fields: 'Title:' with the value 'Поручення №5'; 'Text:' with a large text area containing 'Підготувати аналіз ефективності останньої рекламної кампанії та запропонувати стратегію підвищення продажів на наступний квартал.' and a small emoji icon; 'Urgency tag:' with a dropdown menu showing 'Терміново'; 'Employees:' with a multi-select box containing 'Анна Сергіївна'; and 'Deadline:' with a date and time picker set to '2023-12-25 17:00'. At the bottom of the form is a green button labeled 'Створити задачу'.

Рисунок 4.6 – Сторінка Створення Задачі (tasks/create/)

Сторінка "Перегляд Задачі" у CRM-системі дозволяє користувачам переглядати детальну інформацію про конкретну задачу. Ця сторінка є важливою для ефективного управління задачами, оскільки вона надає всю необхідну інформацію для виконання та відстеження прогресу задачі. Інформація про Задачу: Відображає деталі задачі, включаючи назву, опис, тег екстренності, дедлайн, автора, дату створення та список сотрудників для виконання.

Статус Задачі показує поточний статус задачі (наприклад, "Чекає виконання" або "Виконано"). Є розділ для перегляду та додавання коментарів до задачі. Форма додавання коментаря дозволяє користувачам залишати свої коментарі щодо задачі. Кнопка виконання задачі з'являється для користувачів, які мають виконати задачу, знаходиться кнопка для позначення задачі як "Виконано" (рис. 4.7).

Панель

CRMS

Додому Вийти

### Поручення №12

Поручення: Підготувати фінансовий звіт за останній квартал та підготувати презентацію для наступного засідання ради директорів, запланованого на 10 грудня 2023 року.

Термін виконання: Dec. 25, 2023, 5 p.m.

Статус: Чекає виконання

#### Коментарі

Коментар:

[Залишити коментар](#)

**Павло Риндін** (Nov. 28, 2023, 9:58 p.m.):

Немає додаткових питань?

**Павло Риндін** (Nov. 28, 2023, 10 p.m.):

Немає додаткових питань?

Рисунок 4.7 – Сторінка Перегляду Задачі (tasks/<int:pk>/)

Сторінка "Список Задач" у CRM-системі представляє собою централізований перелік усіх задач, доступних для користувача. Ця сторінка є важливою для організації робочого процесу, дозволяючи користувачам переглядати, відстежувати та управляти задачами, які їм призначені або які вони створили. Список задач відображає всі задачі з основною інформацією, такою як назва, статус, дедлайн та відповідальні співробітники. При натисканні на назву задачі відкривається детальна сторінка з повною інформацією про задачу. Для задач, де користувач є відповідальним, можуть бути кнопки для оновлення статусу задачі (наприклад, "Виконано") (рис. 4.8).

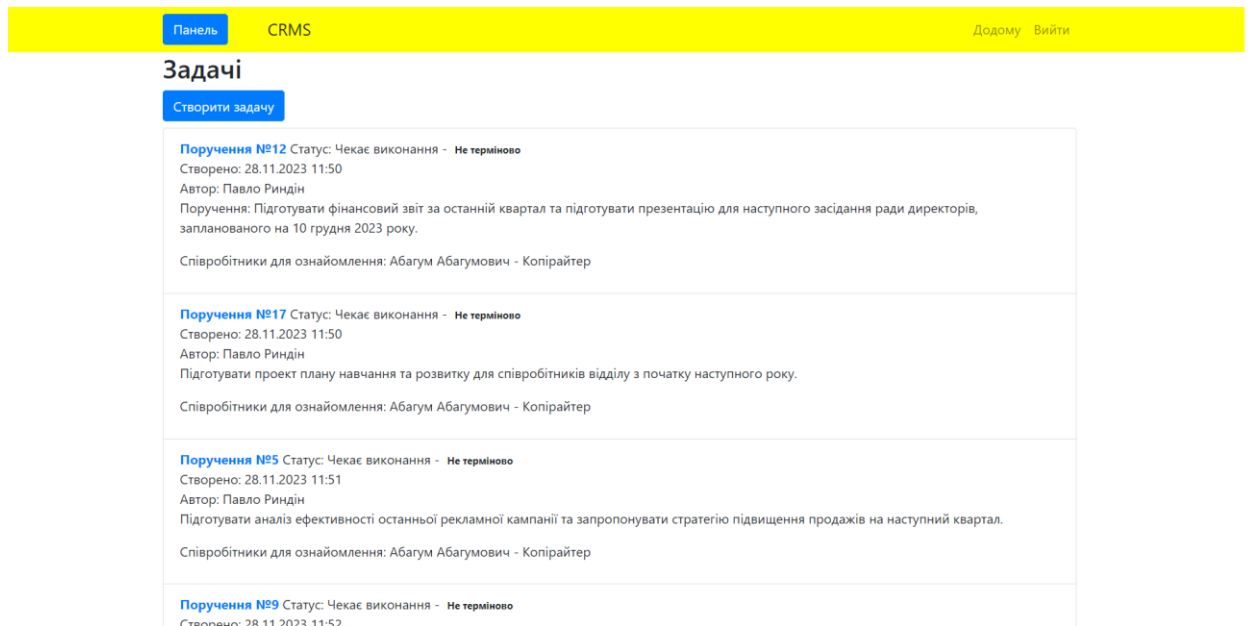


Рисунок 4.8 – Сторінка Списку Задач (tasks/)

Сторінка "Перегляд Документа" у CRM-системі дозволяє користувачам переглядати детальну інформацію про конкретний документ. Ця сторінка є важливою для роботи з документами, оскільки вона надає повний контекст та інформацію, необхідну для розуміння та взаємодії з документом. Інформація про документ відображає деталі документа, включаючи назву, текст, тег екстренності, автора, дату створення та список співробітників для ознайомлення. Статус документа показує поточний статус документа (наприклад, "Чекає затвердження" або "Затверджено"). Є розділ для перегляду та додавання коментарів до документа, що дозволяє користувачам залишати свої коментарі щодо документа. Кнопка затвердження документа для користувачів, які мають право на затвердження документа, відображається кнопка для зміни статусу документа на "Затверджено" (рис. 4.9).

Панель

CRMS

Додому Вийти

### Договір про співпрацю

Між ТОВ "Ефективність" та фірмою "Успіхова Інновація" Предмет договору: Надання консультаційних послуг у сфері маркетингу та реклами. Термін дії: 3 1 грудня 2023 року по 1 грудня 2024 року. Строки та умови оплати: Оплата за надані послуги здійснюється щомісячно не пізніше 5 числа наступного місяця після надання послуги.

Терміновість: Терміново

Створено: 28.11.2023 11:08

Автор: Павло Риндін

Співробітники для ознайомлення: Іван Олексійович, Анна Сергіївна

Статус: Чекає затвердження

Затвердити

Коментарі

Text:

Залишити коментар

Рисунок 4.9 – Сторінка Перегляду Документа (documents/<int:pk>/)

#### 4.4 Тестування розробленої програми

Тестування розробленої CRM-системи є ключовим етапом, який забезпечує її надійність, стабільність та відповідність вимогам. Тестування допомагає виявити та виправити помилки, а також переконатися у правильності реалізації функціоналу.

Для тестування CRM-системи було обрано нативне тестування, яке включає в себе використання вбудованих інструментів Django та ручну перевірку функціоналу сервісу. Цей підхід дозволяє ефективно інтегрувати тестування у процес розробки та забезпечує високу точність та надійність результатів.

Кожен компонент системи, включаючи моделі, рівень репрезентації та форми, були ретельно протестовані на відповідність очікуваній функціональності. Це забезпечило впевненість у правильності роботи окремих елементів системи.

Перевірка взаємодії між різними модулями системи підтвердила їх коректну співпрацю та інтеграцію. Це включало тестування взаємодії з базою даних, роботу API та взаємодію між користувацьким інтерфейсом та

серверною частиною.

Вручну було перевірено, що всі ключові функції системи працюють згідно з вимогами. Це включало тестування процесів створення, редагування, видалення та перегляду документів та задач. Також інтерфейс користувача був протестований на предмет зручності та інтуїтивної зрозумілості, забезпечуючи легкість використання системи кінцевими користувачами.

У результаті нативного тестування всі модулі системи продемонстрували стабільну та надійну роботу. Виявлені помилки були виправлені. Тестування підтвердило, що розроблена CRM система повністю готова до впровадження та використання в реальних умовах.

#### **4.5 Висновки за розділом 4**

Під час аналізу розділу було встановлено технічні параметри для використання розробленої програми, проведено огляд ресурсів та функціонал, який передбачений цією технічною реалізацією.

## ВИСНОВКИ

За результатами розробки CRM-системи для документообігу та управління задачами в компанії були виконані наступні завдання:

- проведено детальний аналіз вимог до системи документообігу та управління задачами, визначено ключові потреби користувачів;
- створено структуру бази даних, що включає моделі для користувачів, документів, задач, коментарів та інші необхідні елементи;
- розроблено основний функціонал CRM-системи, включаючи управління користувачами, документами, задачами, коментарями та іншими важливими процесами;
- створено інтуїтивно зрозумілий веб-інтерфейс з використанням Bootstrap та jQuery, що забезпечує легкий доступ до всіх функцій системи;
- впроваджено використання Celery для автоматизації відправки повідомлень користувачам про наближення дедлайнів задач;
- підготовлено детальну документацію, що описує кожен сторінку сервісу та її функціонал, а також структуру коду.

Розроблена CRM-система є комплексним рішенням для управління документообігом та задачами в компанії, забезпечуючи ефективне взаємодія між співробітниками та автоматизацію ключових процесів. Система включає в себе гнучкість налаштувань, зручний інтерфейс користувача та можливість масштабування під різні потреби бізнесу. Завдання розробки CRM-системи в результаті повністю виконано.

Завдання дипломної кваліфікаційної роботи в результаті повністю виконано.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. About Salesforce [Electronic resource]. – Access mode: <https://help.salesforce.com/s/articleView?id=sf.overview.htm&type=5>.
2. Salesforce [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Salesforce>.
3. About Hubspot [Electronic resource]. – Access mode: <https://www.hubspot.com/products/crm>.
4. About Zoho [Electronic resource]. – Access mode: <https://www.zoho.com/ru/crm/lp/crm-software.html>.
5. Lutz, M. Learning Python: Powerful Object-Oriented Programming / M. Lutz. – Sebastopol: O'Reilly Media, Inc., 2009. – 1650 p.
6. Grinberg, M. Flask Web Development, 2nd Edition / M. Grinberg. – Sebastopol: O'Reilly Media, Inc., 2018. – 312 p.
7. Django Software Foundation [Electronic resource]. – Access mode: <https://www.djangoproject.com/foundation/>.
8. Nixon, R. Learning PHP, MySQL & JavaScript (6th Edition) / R. Nixon. – Sebastopol: O'Reilly Media, Inc., 2021. – 823 p.
9. Obe, R. O., Hsu, L. S. PostgreSQL: Up and Running / R. O. Obe, L. S. Hsu. – Sebastopol, CA: O'Reilly Media, 2012. – 147 p.
10. MySQL Community Edition [Electronic resource]. – Access mode: <https://www.mysql.com/products/community/>.
11. Kuzmichev, S., Grippa, V. M. Learning MySQL / S. Kuzmichev, V. M. Grippa. – Sebastopol: O'Reilly Media; 2nd ed., 2021. – 632 p.
12. Using Redis as a message queue [Electronic resource]. – Access mode: <https://www.oreilly.com/library/view/building-scalable-apps/9781783984480/ch05s06.html>.
13. How to Use Huey in Your Django Project for Asynchronous Task Execution [Electronic resource]. – Access mode:

<https://negativeepsilon.com/en/posts/huey-async-task-execution/>.

14. Asynchronous Tasks With Django and Celery [Electronic resource]. – Access mode: <https://realpython.com/asynchronous-tasks-with-django-and-celery/>.

15. Celery + Redis + Django [Electronic resource]. – Access mode: <https://www.codingforentrepreneurs.com/blog/celery-redis-django/>.

**ДОДАТОК А**  
**Технічне завдання**

## **Вступ**

Розроблена CRM-система призначена для ефективного управління документообігом та задачами в організаціях різного масштабу. Система надає можливість централізованого управління документами, задачами, а також взаємодії між співробітниками через організаційну структуру та систему коментарів. Основна мета системи - оптимізація процесів документообігу та управління задачами в компанії, забезпечення прозорості та ефективності робочих процесів. Система також включає інструменти для моніторингу виконання задач та управління дедлайнами, що дозволяє підвищити продуктивність роботи співробітників.

### **A.1 Підстави для розробки**

Підставою для розробки було завдання на дипломну кваліфікаційну роботу за темою «Дослідження та програмна реалізація CRM системи для автоматизації документообігу у компанії» на основі наказу №437 від 14 листопада 2022 р. за Національним університетом «Запорізька політехніка».

### **A.2 Призначення розробки**

Розроблена CRM система призначена для автоматизації процесів документообігу та управління задачами в компаніях. Система надає зручний інтерфейс для створення, зберігання, та управління документами та задачами, а також включає функціонал для відстеження статусів документів і задач, від коментування до затвердження чи виконання. Особливістю системи є інтегрована організаційна структура, яка дозволяє відображати ієрархію та взаємозв'язки між співробітниками, а також надає можливість керівникам ефективно управляти підлеглими та їх завданнями. Система також включає інструменти для моніторингу дедлайнів та нагадувань, що сприяє своєчасному

виконанню завдань та підтриманню високого рівня продуктивності в організації.

### **А.3 Основні вимоги до програми**

#### **А.3.1 Вимоги до функціональних характеристик**

Сервіс повинен забезпечувати наступний функціонал адміністратора:

- можливість перегляду профілю іншого користувача;
- можливість створення, оновлення та видалення профілей

користувачів;

- можливість зміни ролі користувача;
- можливість переналаштування системи підпорядковування

користувачів в системі;

- можливість перегляду документів інших користувачів;
- можливість оновлення та видалення документів;
- можливість зміни прив'язки файлів до документів;
- можливість налаштування списку узгоджуючих документа;
- можливість залишати коментарі до документів;
- можливість перегляду задач інших користувачів;
- можливість створення, оновлення та видалення власних задач;
- можливість оновлення та видалення задач інших користувачів;
- можливість зміни прив'язки файлів до задач;
- можливість залишати коментарі до задач.

Функціональні вимоги керуючого департаментом:

- можливість перегляду власного профіля;
- перегляд профілей інших користувачей;
- можливість оновлення інформації власного профіля;
- можливість переналаштування системи підпорядковування

користувачів у власному департаменті;

- можливість перегляду документів інших користувачів;

- можливість перегляду власних документів;
- можливість створення документів;
- можливість оновлення та видалення власних документів;
- можливість налаштування списку узгоджуваних документів

підпорядкованих користувачів;

- можливість залишати коментарі до документів;
- можливість узгодження документів;
- можливість оновлення та видалення документів підпорядкованих

користувачів;

- можливість перегляду задач інших користувачів;
- можливість створення задач;
- можливість оновлення та видалення власних задач;
- можливість оновлення та видалення задач підпорядкованих

користувачів.

Функціональні вимоги працівника департаменту включають:

- можливість перегляду власних документів;
- можливість оновлення та видалення власних документів;
- можливість створення документів;
- можливість перегляду призначених документів;
- можливість узгодження призначених документів;
- можливість залишати коментарі до призначених документів;
- можливість перегляду власних задач;
- можливість перегляду призначених задач;
- можливість оновлення та видалення власних задач;
- можливість виконувати призначені задачі;
- можливість перегляду власного профілю;
- можливість оновлення інформації власного профіля.

### **A.3.2 Вимоги до надійності**

Надійність роботи розробленої CRM системи забезпечується за рахунок стабільної роботи сервера бази даних та правильного налаштування його взаємодії з основним вебдодатком. Окрім цього, для забезпечення функціоналу асинхронних задач та планувальника завдань необхідно налаштувати та запустити сервіси Celery та Redis. Celery використовується для обробки фонових задач, таких як надсилання сповіщень про наближення дедлайнів, тоді як Redis виступає в ролі брокера повідомлень для Celery. Важливим аспектом є також налаштування періодичних завдань та моніторинг їх виконання. Система також включає статичні файли, які зберігаються на серверах, що вимагають налаштування та підтримки доступності. Загалом, надійність системи залежить від якісного налаштування всіх її компонентів та регулярного моніторингу їх стану.

### **A.3.3 Умови експлуатації**

Для забезпечення стабільної та ефективної роботи розробленої CRM системи необхідно дотримуватися певних умов експлуатації. Система вимагає наявності сервера для розміщення програмної частини програми, який повинен мати достатні ресурси для обробки запитів користувачів та виконання фонових задач. Також потрібен окремий сервер бази даних, який забезпечує зберігання та швидкий доступ до даних системи. Налаштування та оптимізація сервера бази даних є ключовими для забезпечення високої продуктивності та надійності системи.

### **A.3.4 Вимоги до складу та параметрів технічних засобів**

Вимоги до сервера включають:

- не менше 1,5 Гб вільного місця на жорсткому диску;

- мінімум 4 Гб оперативної пам'яті;
- процесор з 4 ядрами та частотою 2,8 ГГц.

Для коректної роботи вебсервісу на боці користувача необхідний оновлений веб браузер, який підтримує підключення до Інтернету.

#### **А.4 Порядок контролю та приймання**

Контроль за виконанням дипломної кваліфікаційної роботи покладається на керівника. Остаточне рішення приймається екзаменаційною комісією під час захисту роботи.

**ДОДАТОК Б**  
**Текст програми**

## Б.1 Текст файлу celery.py

```

from celery import shared_task
from django.core.mail import send_mail
from .models import Task
from django.utils import timezone

@shared_task
def check_task_deadlines():
    now = timezone.now()
    due_tasks = Task.objects.filter(deadline__lte=now, status='pending')
    for task in due_tasks:
        subject = f"Термін виконання задачі {task.title} закінчився!"
        message = f"Термін виконання задачі {task.title} закінчився. Будь ласка, перевірте статус."
        recipient_list = [user.email for user in task.employees.all()]
        send_mail(subject, message, 'from@example.com', recipient_list)

```

## Б.2 Текст файлу settings.py

```

from pathlib import Path
import os
import environ
env = environ.Env()
environ.Env.read_env()
BASE_DIR = Path(__file__).resolve().parent.parent
SECRET_KEY = 'django-insecure-zt8pml$qwfn0y6@u1w3v3fs5$)li+yf9#7^!f5&0_jz&v38co'
DEBUG = True
ALLOWED_HOSTS = ['localhost', '127.0.0.1']
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

```

```

'users',
'documents',
'tasks',
'rest_framework',
]
AUTH_USER_MODEL = 'users.User'
MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'CRMS.urls'
TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [os.path.join(BASE_DIR, 'templates')],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
]
WSGI_APPLICATION = 'CRMS.wsgi.application'
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': 'orms',
'USER': 'root',
'PASSWORD': 'totalmag',

```

```

        'HOST': 'localhost',
        'PORT': '3306',
    }
}
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.forms.AuthenticationForm',
    },
    {
        'NAME': 'django.contrib.auth.views.LoginView',
    },
]
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
LOGIN_URL = 'users:login'
LOGOUT_REDIRECT_URL = '/'
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

```

```
CELERY_BROKER_URL = 'redis://localhost:6379/0'
CELERY_ACCEPT_CONTENT = ['json']
CELERY_TASK_SERIALIZER = 'json'
CELERY_RESULT_SERIALIZER = 'json'
```

### Б.3 Текст файла CRMS/urls.py

```
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('users.urls')),
    path("", include('documents.urls')),
    path("", include('tasks.urls'))
]
```

### Б.4 Текст файла documents/urls.py

```
from django.urls import path
from .views import DocumentListView, DocumentCreateView, DocumentDetailView
from django.conf import settings
from django.conf.urls.static import static
app_name = 'documents'
urlpatterns = [
    path('documents/', DocumentListView.as_view(), name='document-list'),
    path('create_document/', DocumentCreateView.as_view(), name='create-document'),
    path('documents/<int:pk>/', DocumentDetailView.as_view(), name='document-detail'),
]
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

### Б.5 Текст файла documents/views.py

```
from django.shortcuts import render, get_object_or_404, redirect
from django.views.generic import ListView
from django.urls import reverse_lazy
from django.views.generic.edit import CreateView
from .models import Document, Approval
from .forms import DocumentForm, CommentForm
from django.views.generic.detail import DetailView
```

```

from django.http import HttpResponseRedirect

class DocumentListView(ListView):
    model = Document
    context_object_name = 'documents'
    template_name = 'documents/documents.html'

class DocumentCreateView(CreateView):
    model = Document
    form_class = DocumentForm
    template_name = 'documents/document_form.html'
    success_url = reverse_lazy('documents:document-list')

    def form_valid(self, form):
        form.instance.created_by = self.request.user
        document = form.save(commit=False)
        document.save()
        form.save_m2m()
        return super(DocumentCreateView, self).form_valid(form)

class DocumentDetailView(DetailView):
    model = Document
    template_name = 'documents/document_detail.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['comment_form'] = CommentForm()
        return context

    def post(self, request, *args, **kwargs):
        self.object = self.get_object()
        if 'approve' in request.POST:
            if request.user in self.object.employees.all():
                Approval.objects.update_or_create(document=self.object, user=request.user,
defaults={'approved': True})
                if self.object.approvals.filter(approved=False).count() == 0:
                    self.object.status = 'approved'
                    self.object.save()
                return HttpResponseRedirect(self.object.get_absolute_url())
        elif 'comment' in request.POST:
            comment_form = CommentForm(request.POST)
            if comment_form.is_valid():
                comment = comment_form.save(commit=False)
                comment.document = self.object

```

```

comment.author = request.user
comment.save()
return redirect(self.object.get_absolute_url())
return self.render_to_response(self.get_context_data(form=comment_form))

```

## Б.6 Текст файлу documents/models.py

```

from django.db import models
from django.conf import settings
class Document(models.Model):
    URGENCY_CHOICES = (
        ('urgent', 'Терміново'),
        ('medium', 'Середнє'),
        ('not_urgent', 'Не терміново'),
    )
    title = models.CharField(max_length=255)
    text = models.TextField()
    urgency_tag = models.CharField(max_length=50, choices=URGENCY_CHOICES)
    files = models.FileField(upload_to='documents/', null=True, blank=True)
    employees = models.ManyToManyField(settings.AUTH_USER_MODEL, related_name='documents')
    created_at = models.DateTimeField(auto_now_add=True)
    created_by = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE,
related_name='created_documents')
    STATUS_CHOICES = (
        ('pending', 'Чекає затвердження'),
        ('approved', 'Затверджено'),
    )
    status = models.CharField(max_length=10, choices=STATUS_CHOICES, default='pending')
    def __str__(self):
        return self.title
class Approval(models.Model):
    document = models.ForeignKey(Document, on_delete=models.CASCADE, related_name='approvals')
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    approved = models.BooleanField(default=False)
    class Meta:
        unique_together = ('document', 'user')

```

```

class Comment(models.Model):
    document = models.ForeignKey(Document, on_delete=models.CASCADE, related_name='comments')
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return f'Коментарів від {self.author.get_full_name()} на {self.document.title}'

```

## Б.7 Текст файлу documents/forms.py

```

from django import forms
from .models import Document, Comment
from users.forms import UserModelChoiceField
from django.contrib.auth import get_user_model
User = get_user_model()
class DocumentForm(forms.ModelForm):
    employees = UserModelChoiceField(
        queryset=User.objects.all(),
        widget=forms.SelectMultiple(attrs={'class': 'form-control js-example-basic-multiple'}),
        required=False
    )
    class Meta:
        model = Document
        fields = ['title', 'text', 'urgency_tag', 'files', 'employees']
        widgets = {
            'urgency_tag': forms.Select(attrs={'class': 'form-control'}),
            'title': forms.TextInput(attrs={'class': 'form-control'}),
            'text': forms.Textarea(attrs={'class': 'form-control'}),
        }
class CommentForm(forms.ModelForm):
    class Meta:
        model = Comment
        fields = ['text']
        widgets = {
            'text': forms.Textarea(attrs={'class': 'form-control'}),
        }

```

## Б.8 Текст файлу tasks/urls.py

```

from django.urls import path
from .views import TaskListView, TaskDetailView, TaskCreateView, TaskUpdateView
app_name = 'tasks'
urlpatterns = [
    path('tasks/', TaskListView.as_view(), name='task-list'),
    path('tasks/<int:pk>/', TaskDetailView.as_view(), name='task-detail'),
    path('tasks/create/', TaskCreateView.as_view(), name='task-create'),
    path('tasks/<int:pk>/update/', TaskUpdateView.as_view(), name='task-update'),
]

```

## Б.9 Текст файла tasks/views.py

```

from django.views.generic import ListView, DetailView, CreateView, UpdateView
from .models import Task, TaskComment, TaskCompletion
from .forms import TaskForm
from django.http import HttpResponseRedirect
from .forms import TaskCommentForm
from django.urls import reverse_lazy
from django.shortcuts import get_object_or_404, redirect

class TaskListView(ListView):
    model = Task
    template_name = 'tasks/task_list.html'
    context_object_name = 'tasks'

class TaskDetailView(DetailView):
    model = Task
    template_name = 'tasks/task_detail.html'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        task = self.get_object()
        context['comments'] = TaskComment.objects.filter(task=task)
        context['comment_form'] = TaskCommentForm()
        context['is_completed'] = TaskCompletion.objects.filter(task=task, user=self.request.user,
        completed=True).exists()
        return context
    def post(self, request, *args, **kwargs):
        task = self.get_object()
        if 'submit_comment' in request.POST:
            comment_form = TaskCommentForm(request.POST)

```

```

    if comment_form.is_valid():
        TaskComment.objects.create(
            task=task,
            author=request.user,
            text=comment_form.cleaned_data['text']
        )
        return redirect(task.get_absolute_url())
    elif 'complete_task' in request.POST:
        TaskCompletion.objects.update_or_create(
            task=task,
            user=request.user,
            defaults={'completed': True}
        )
        return redirect(task.get_absolute_url())
    return super().post(request, *args, **kwargs)

class TaskCreateView(CreateView):
    model = Task
    form_class = TaskForm
    template_name = 'tasks/task_form.html'
    success_url = reverse_lazy('documents:document-list')
    def form_valid(self, form):
        form.instance.created_by = self.request.user
        return super().form_valid(form)

class TaskUpdateView(UpdateView):
    model = Task
    fields = ['title', 'text', 'urgency_tag', 'employees']
    template_name = 'tasks/task_form.html'

```

## Б.10 Текст файла tasks/forms.py

```

from django import forms
from .models import Task
from django import forms
from users.forms import UserModelChoiceField
from django.contrib.auth import get_user_model
User = get_user_model()
class TaskForm(forms.ModelForm):
    employees = UserModelChoiceField(

```

```

        queryset=User.objects.all(),
        widget=forms.SelectMultiple(attrs={'class': 'form-control js-example-basic-multiple'}),
        required=False
    )
class Meta:
    model = Task
    fields = ['title', 'text', 'urgency_tag', 'employees', 'deadline']
    widgets = {
        'urgency_tag': forms.Select(attrs={'class': 'form-control'}),
        'title': forms.TextInput(attrs={'class': 'form-control'}),
        'text': forms.Textarea(attrs={'class': 'form-control'}),
    }
class TaskCommentForm(forms.Form):
    text = forms.CharField(
        widget=forms.Textarea(attrs={'rows': 4, 'cols': 40}),
        label='Коментар',
        required=True
    )

```

### Б.11 Текст файлу tasks/models.py

```

from django.db import models
from django.conf import settings
from django.urls import reverse
class Task(models.Model):
    title = models.CharField(max_length=255)
    text = models.TextField()
    urgency_tag = models.CharField(max_length=10, choices=[('high', 'Терміново'), ('medium', 'Середнє'), ('low', 'Не терміново')])
    employees = models.ManyToManyField(settings.AUTH_USER_MODEL, related_name='tasks')
    created_at = models.DateTimeField(auto_now_add=True)
    created_by = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='created_tasks')
    status = models.CharField(max_length=15, choices=[('pending', 'Чекає виконання'), ('completed', 'Виконано')], default='pending')
    deadline = models.DateTimeField(null=True, blank=True)
    def get_absolute_url(self):
        return reverse('task_detail', kwargs={'pk': self.pk})

```

```

def __str__(self):
    return self.title
class TaskComment(models.Model):
    task = models.ForeignKey(Task, on_delete=models.CASCADE, related_name='comments')
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return f'Коментарій от {self.author.get_full_name()} на задачу {self.task.title}'
class TaskCompletion(models.Model):
    task = models.ForeignKey(Task, on_delete=models.CASCADE, related_name='completions')
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    completed = models.BooleanField(default=False)
    class Meta:
        unique_together = ('task', 'user')

```

## Б.12 Текст файлу tasks/tasks.py

```

from celery import shared_task
from django.core.mail import send_mail
from .models import Task
from django.utils import timezone
@shared_task
def check_task_deadlines():
    now = timezone.now()
    due_tasks = Task.objects.filter(deadline__lte=now, status='pending')
    for task in due_tasks:
        subject = f"Термін виконання задачі {task.title} закінчився!"
        message = f"Термін виконання задачі {task.title} закінчився. Будь ласка, перевірте статус."
        recipient_list = [user.email for user in task.employees.all()]
        send_mail(subject, message, 'from@example.com', recipient_list)

```

## Б.13 Текст файлу users/urls.py

```

from django.urls import path
from .views import MainView, CustomLoginView, OrgChartView, remove_subordinate
from django.contrib.auth.views import LogoutView

```

```

app_name = 'users'
urlpatterns = [
    path("", MainView.as_view(), name='main'),
    path('login/', CustomLoginView.as_view(), name='login'),
    path('logout/', LogoutView.as_view(), name='logout'),
    path('org/', OrgChartView.as_view(), name='org-chart'),
    path('remove-subordinate/', remove_subordinate, name='remove-subordinate'),
]

```

### Б.14 Текст файла users/views.py

```

from django.views.generic import TemplateView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib.auth.views import LoginView
from django.shortcuts import redirect
from .forms import AddSubordinateForm
from .models import User

class MainView(TemplateView):
    template_name = 'main.html'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        if self.request.user.is_authenticated:
            context['full_name'] = self.request.user.get_full_name()
            context['position'] = self.request.user.position
        return context

class CustomLoginView(LoginView):
    template_name = 'users/login.html'

class OrgChartView(LoginRequiredMixin, TemplateView):
    template_name = 'org_chart.html'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        user = self.request.user
        context['subordinates'] = user.subordinates.all()
        context['managers'] = User.objects.filter(subordinates=user)
        if self.request.user.is_staff:
            available_subordinates = User.objects.exclude(id=user.id).exclude(id__in=user.subordinates.all())
            context['add_subordinate_form'] = AddSubordinateForm(user=user)
        return context

```

```

def post(self, request, *args, **kwargs):
    if not request.user.is_staff:
        return redirect('org-chart')
    user = request.user
    form = AddSubordinateForm(request.POST, user=user)
    if form.is_valid():
        subordinates = form.cleaned_data['subordinates']
        for subordinate in subordinates:
            request.user.subordinates.add(subordinate)
        return redirect('users:org-chart')
    else:
        return self.render_to_response(self.get_context_data(form=form))
from django.http import JsonResponse
from django.views.decorators.http import require_POST
from django.views.decorators.csrf import csrf_exempt
@require_POST
@csrf_exempt
def remove_subordinate(request):
    if not request.user.is_staff:
        return JsonResponse({'success': False})

    subordinate_id = request.POST.get('subordinate_id')
    subordinate = User.objects.get(id=subordinate_id)
    request.user.subordinates.remove(subordinate)
    return JsonResponse({'success': True})

```

### **Б.15 Текст файла users/forms.py**

```

from django import forms
from .models import User
class UserModelChoiceField(forms.ModelMultipleChoiceField):
    def label_from_instance(self, obj):
        return f"{obj.first_name} {obj.last_name}"
class AddSubordinateForm(forms.Form):
    def __init__(self, *args, **kwargs):
        user = kwargs.pop('user', None)
        super().__init__(*args, **kwargs)

```

```

if user is not None:
    self.fields['subordinates'].queryset = User.objects.exclude(
        id=user.id
    ).exclude(
        id__in=user.subordinates.all()
    )
subordinates = forms.ModelMultipleChoiceField(
    queryset=User.objects.none(),
    widget=forms.SelectMultiple(attrs={'class': 'form-control'}),
    required=False
)

```

### Б.16 Текст файла `users/models.py`

```

from django.db import models
from django.db import models
from django.contrib.auth.models import AbstractUser, BaseUserManager
class Department(models.Model):
    name = models.CharField(max_length=100, unique=True)
    def __str__(self):
        return self.name
class UserManager(BaseUserManager):
    def create_user(self, phone, password=None, **extra_fields):
        if not phone:
            raise ValueError('Users must have a phone number')
        user = self.model(phone=phone, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user
    def create_superuser(self, phone, password, **extra_fields):
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)
        if extra_fields.get('is_staff') is not True:
            raise ValueError('Superuser must have is_staff=True.')
        if extra_fields.get('is_superuser') is not True:
            raise ValueError('Superuser must have is_superuser=True.')
        return self.create_user(phone, password, **extra_fields)

```

```

class User(AbstractUser):
    phone = models.CharField('phone', unique=True, max_length=17)
    department = models.ForeignKey(Department, on_delete=models.SET_NULL, null=True,
related_name='employees')
    position = models.CharField(max_length=100)
    subordinates = models.ManyToManyField('self', symmetrical=False, related_name='supervisors',
blank=True)
    objects = UserManager()
    USERNAME_FIELD = 'phone'
    REQUIRED_FIELDS = []
    def __str__(self):
        return self.username
    def add_subordinate(self, subordinate):
        self.subordinates.add(subordinate)
    def remove_subordinate(self, subordinate):
        self.subordinates.remove(subordinate)

```

### Б.17 Текст файла templates/admin/base\_site.html

```

{% extends "admin/base_site.html" %}
{% load static %}

{% block extrastyle %}
<link rel="stylesheet" type="text/css" href="{% static 'css/admin_custom.css' %}" />
{% endblock %}

{% block branding %}
<a href="/admin/" class="navbar-brand">
    
</a>
{% endblock %}

```

### Б.18 Текст файла templates/documents/document\_detail.html

```

{% extends 'base.html' %}
{% block content %}
<div class="container">
    <h2>{{ document.title }}</h2>
    <p>{{ document.text }}</p>

```

```

<p>Терміновість: {{ document.get_urgency_tag_display }}</p>
<p>Створено: {{ document.created_at|date:"d.m.Y H:i" }}</p>
<p>Автор: {{ document.created_by.get_full_name }}</p>
<p>Співробітники для ознайомлення:
    {% for employee in document.employees.all %}
        {{ employee.get_full_name }}{% if not forloop.last %}, {% endif %}
    {% empty %}
        Нема співробітників для ознайомлення.
    {% endfor %}
</p>
<p>Статус: {{ document.get_status_display }}</p>
<form method="post">
    {% csrf_token %}
    <button type="submit" name="approve" class="btn btn-success">Затвердити</button>
</form>
<h4>Коментарі</h4>
<form method="post">
    {% csrf_token %}
    {{ comment_form.as_p }}
    <button type="submit" name="comment" class="btn btn-primary">Залишити
коментар</button>
</form>
</div>
{% endblock %}

```

## Б.19 Текст файлу templates/documents/document\_form.html

```

{% extends 'base.html' %}
{% block content %}
<div class="container">
    <h2>Створити документ</h2>
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit" class="btn btn-success mb-3">Створити документ</button>
    </form>
</div>
{% endblock %}

```

```
{% block javascript %}
<script>
$(document).ready(function() {
    $('.js-example-basic-multiple').select2({
        placeholder: "Выберите сотрудников",
        allowClear: true,
        minimumInputLength: 1
    });
});
</script>
{% endblock %}
```

## Б.20 Текст файла templates/documents/documents.html

```
{% extends 'base.html' %}
{% block content %}
<div class="container">
    <h2>Документы</h2>
    <div class="mb-2">
        <a href="{% url 'documents:create-document' %}" class="btn btn-primary">Создать
документ</a>
    </div>
    <ul class="list-group">
        {% for document in documents %}
            <li class="list-group-item">
                <a href="{% url 'documents:document-detail' document.id %}"><strong>{{ document.title
}}</strong></a>
                <span>Статус: {{ document.get_status_display }}</span>
                - <span class="badge rounded-pill
{% if document.urgency_tag == 'urgent' %}bg-danger
{% elif document.urgency_tag == 'medium' %}bg-warning text-dark
{% elif document.urgency_tag == 'not_urgent' %}bg-success
{% endif %}">
                    {{ document.get_urgency_tag_display }}
                </span>
            </li>
        </ul>
        <div>Создано: {{ document.created_at|date:"d.m.Y H:i" }}</div>
        <div>Автор: {{ document.created_by.get_full_name }}</div>
```

```

<p>{{ document.text }}</p>
{% if document.files %}
    <a href="{{ document.files.url }}">Завантажити файл</a>
{% endif %}
<p>Співробітники для ознайомлення:    {% for employee in document.employees.all %}
    {{ employee.get_full_name }} - {{ employee.position }}{% if not forloop.last %}, {% endif
%}

    {% empty %}
    Нема співробітників для ознайомлення.
    {% endfor %}</p>
</li>
{% empty %}
    <li class="list-group-item">Нема документів для відображення.</li>
{% endfor %}
</ul>
</div>
{% endblock %}

```

## Б.21 Текст файлу templates/tasks/task\_detail.html

```

{% extends 'base.html' %}
{% block content %}
<div class="container">
    <h2>{{ task.title }}</h2>
    <p>{{ task.text }}</p>
    <p>Термін виконання: {{ task.deadline }}</p>
    <p>Статус: {{ task.get_status_display }}</p>
    {% if not is_completed %}
    <form method="post">
        {% csrf_token %}
        <button type="submit" name="complete_task" class="btn btn-success">Виконати</button>
    </form>
    {% endif %}
    <h3>Коментарі</h3>
    <form method="post">
        {% csrf_token %}
        {{ comment_form.as_p }}
        <button type="submit" name="submit_comment" class="btn btn-primary">Залишити

```

```

коментар</button>
</form>
{% for comment in comments %}
<div class="comment">
  <p><strong>{{ comment.author.get_full_name }}</strong> ({{ comment.created_at }}):</p>
  <p>{{ comment.text }}</p>
</div>
{% endfor %}
</div>
{% endblock %}

```

## Б.22 Текст файлу templates/tasks/task\_form.html

```

{% extends 'base.html' %}
{% block content %}
<div class="container">
  <h2>Створити задачу</h2>
  <form method="post" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit" class="btn btn-success mb-3">Створити задачу</button>
  </form>
</div>
{% endblock %}
{% block javascript %}
<script>
$(document).ready(function() {
  $('<div>.js-example-basic-multiple</div>').select2({
    placeholder: "Виберіть співробітників",
    allowClear: true,
    minimumInputLength: 1
  });
});
</script>
{% endblock %}

```

## Б.23 Текст файлу templates/tasks/task\_list.html

```
{% extends 'base.html' %}

{% block content %}
<div class="container">
  <h2>Задачі</h2>
  <div class="mb-2">
    <a href="{% url 'tasks:task-create' %}" class="btn btn-primary">Створити задачу</a>
  </div>
  <ul class="list-group">
    {% for task in tasks %}
      <li class="list-group-item">
        <a href="{% url 'tasks:task-detail' task.id %}"><strong>{{ task.title }}</strong></a>
        <span>Статус: {{ task.get_status_display }}</span>
        - <span class="badge rounded-pill
          {% if task.urgency_tag == 'urgent' %}bg-danger
          {% elif task.urgency_tag == 'medium' %}bg-warning text-dark
          {% elif task.urgency_tag == 'not_urgent' %}bg-success
          {% endif %}">
          {{ task.get_urgency_tag_display }}
        </span>
        <div>Створено: {{ task.created_at|date:"d.m.Y H:i" }}</div>
        <div>Автор: {{ task.created_by.get_full_name }}</div>
        <p>{{ task.text }}</p>
        <p>Співробітники для ознайомлення: {% for employee in task.employees.all %}
          {{ employee.get_full_name }} - {{ employee.position }}{% if not forloop.last %}, {% endif %}
        </p>
      {% empty %}
        Нема співробітників для ознайомлення.
      {% endfor %}
    </li>
    {% empty %}
      <li class="list-group-item">Нема задач для відображення.</li>
    {% endfor %}
  </ul>
</div>
{% endblock %}
```

## Б.24 Текст файлу templates/users/login.html

```
{% extends 'base.html' %}
{% block content %}
<div class="container-fluid site-wrapper">
  <div class="row text-center justify-content-md-center">
    <div class="col-md-4 col-sm-8">
      <div class="card">
        <article class="card-body">
          <h2>Вхід в систему</h2>
          <form method="post">
            {% csrf_token %}
            <label for="id_username">Телефон:</label>
            <input type="text" name="username" autofocus placeholder="+380"><br>
            <label for="id_password">Пароль:</label>
            <input type="password" name="password"><br>
            <button class="btn btn-primary mr-5" type="submit">Увійти</button>
          </form>
        </article>
      </div>
    </div>
  </div>
</div>
{% endblock %}
{% block extra_js %}
<script>
  document.addEventListener('DOMContentLoaded', function() {
    var phoneInput = document.querySelector('input[name="phone"]');
    phoneInput.addEventListener('focus', function() {
      if (!phoneInput.value) {
        phoneInput.value = '+380';
      }
    });
    phoneInput.addEventListener('blur', function() {
      if (phoneInput.value === '+380') {
        phoneInput.value = '';
      }
    });
  });
</script>
```

```
});
</script>
{% endblock %}
```

## Б.25 Текст файлу base.html

```
{% load static %}
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{% block title %}Документооборот{% endblock %}</title>
  <link rel="stylesheet" href="{% static 'css/style.css' %}">
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
rel="stylesheet">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <link href="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.13/css/select2.min.css" rel="stylesheet"
/>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.13/js/select2.min.js"></script>
  {% block extra_css %}
  {% endblock %}
</head>
<body>
  <div class="d-flex" id="wrapper" style="width:100%">
    <div class="bg-light border-right" id="sidebar-wrapper">
      <div class="list-group list-group-flush">
        <a href="{% url 'users:org-chart' %}" class="list-group-item list-group-item-action bg-
light">Підпорядкування</a>
        <a href="{% url 'documents:document-list' %}" class="list-group-item list-group-item-action
bg-light">Документи</a>
        <a href="{% url 'tasks:task-list' %}" class="list-group-item list-group-item-action bg-
light">Задачі</a>
      </div>
    </div>
    <div id="page-content-wrapper" style="width:100%">
```

```

<nav class="navbar navbar-expand-lg navbar-light" style="background-color: #ffff00;">
  <div class="container">
    <button class="btn btn-primary mr-5" id="menu-toggle">Панель</button>
    <a class="navbar-brand" href="{% url 'users:main' %}">CRMS</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav ml-auto">
        <li class="nav-item">
          <a class="nav-link" href="{% url 'users:main' %}">Додому</a>
        </li>
        {% if user.is_authenticated %}
          <li class="nav-item">
            <a href="{% url 'users:logout' %}" class="nav-link">Вийти</a>
          </li>
        {% else %}
          <li class="nav-item">
            <a class="nav-link" href="{% url 'users:login' %}">Увійти</a>
          </li>
        {% endif %}
      </ul>
    </div>
  </div>
</nav>
{% block content %}
{% endblock %}
</div>
<script>
  $("#menu-toggle").click(function(e) {
    e.preventDefault();
    $("#wrapper").toggleClass("toggled");
  });
</script>
{% block javascript %}
{% endblock %}

```

```
</body>
</html>
```

## Б.26 Текст файлу main.html

```
{% extends 'base.html' %}
{% load static %}
{% block title %}Головна сторінка{% endblock %}
{% block content %}
<div class="text-center">
  <h1>CRMS - Система Документообігу</h1>
  
  <p>CRMS - це інноваційна система управління документами, яка дозволяє оптимізувати
процеси документообігу всередині компанії, покращуючи взаємодію між відділами та підвищуючи
загальну ефективність роботи.</p>
  {% if user.is_authenticated %}
  <div class="user-info">
    <h2>Ласкаво просимо, {{ full_name }}</h2>
    <p>Посада: {{ position }}</p>
  </div>
  {% endif %}
</div>
{% endblock %}
```

## Б.27 Текст файлу org\_chart.html

```
{% extends 'base.html' %}
{% block content %}
<div class="container">
  <h2>Організаційна структура</h2>
  <div class="row mt-5">
    <div class="col-md-6">
      <h3>Мої підлеглі</h3>
      <ul class="list-group">
        {% for subordinate in subordinates %}
        <li class="list-group-item d-flex justify-content-between align-items-center">
          {{ subordinate.first_name }} {{ subordinate.last_name }} - {{ subordinate.position }}
          {% if user.is_staff %}
            <span class="badge badge-danger badge-pill" style="cursor:pointer;"
```

```

onclick="removeSubordinate('{{ subordinate.id }}');">X</span>
    {% endif %}
</li>
{% empty %}
    <li class="list-group-item">У вас немає підлеглих.</li>
{% endfor %}
</ul>
</div>
<div class="col-md-6">
    <h3>Мої керівники</h3>
    <ul class="list-group">
        {% for manager in managers %}
            <li class="list-group-item">
                {{ manager.first_name }} {{ manager.last_name }} - {{ manager.position }}
            </li>
        {% empty %}
            <li class="list-group-item">Ви не маєте керівників.</li>
        {% endfor %}
    </ul>
</div>
</div>
{% if user.is_staff %}
<form method="post">
    {% csrf_token %}
    {{ add_subordinate_form.as_p }}
    <button type="submit" class="btn btn-primary">Додати підлеглих</button>
</form>
{% endif %}
</div>
{% endblock %}
{% block javascript %}
<script>
function removeSubordinate(subordinateId) {
    if(confirm('Вы уверены, что хотите удалить этого подчинённого?')) {
        $.ajax({
            url: '{% url 'users:remove-subordinate' %}',
            type: 'post',
            data: {

```

```
        'subordinate_id': subordinateId,  
        'csrfmiddlewaretoken': '{{ csrf_token }}'  
    },  
    success: function(response) {  
        if(response['success']) {  
            location.reload();  
        }  
    }  
});  
}  
}  
</script>  
{% endblock %}
```

**ДОДАТОК В**  
**Слайди презентації**

Національний університет "Запорізька політехніка"

Кафедра програмних засобів

**Дипломна кваліфікаційна робота  
на тему:**

**ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ CRM СИСТЕМИ  
ДЛЯ АВТОМАТИЗАЦІЇ ДОКУМЕНТООБІГУ У КОМПАНІЇ  
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE CRM  
SYSTEM FOR DOCUMENT FLOW AUTOMATION IN THE COMPANY**

Виконав  
Студент групи КНТ-112м

П.Ю. Риндін

Керівник  
к.т.н., доцент

Т. В. Федорончак

2023

Рисунок В.1 – Слайд № 1

**Об'єкт дослідження** – процес адміністрування CRM системи для автоматизації документообігу у компанії.

**Предмет дослідження** – програмне забезпечення автоматизації документообігу у компанії.

**Мета роботи** – розробка програмного забезпечення адміністрування та продажу цифрових товарів.

**Галузь використання** – адміністрування процесу документообігу у компанії.

**Завдання роботи:**

- Аналіз предметної області
- Розробка архітектури програмного забезпечення
- Реалізація програмного забезпечення
- Тестування розробленого програмного забезпечення

2

Рисунок В.2 – Слайд № 2

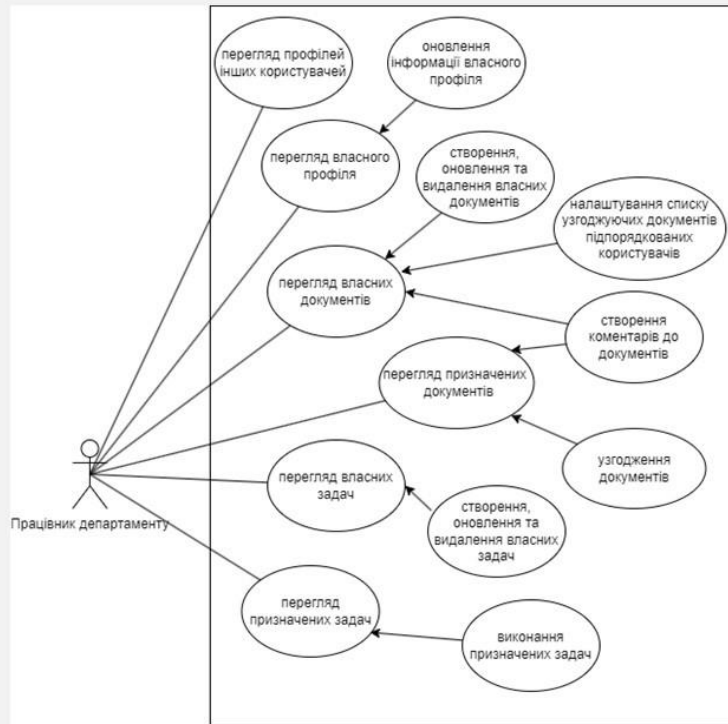


Рисунок В.3 – Слайд № 3



Рисунок В.4 – Слайд № 4

# ДІАГРАМА ПРЕЦЕДЕНТІВ ПРАЦІВНИКА ДЕПАРТАМЕНТУ



5

Рисунок В.5 – Слайд № 5

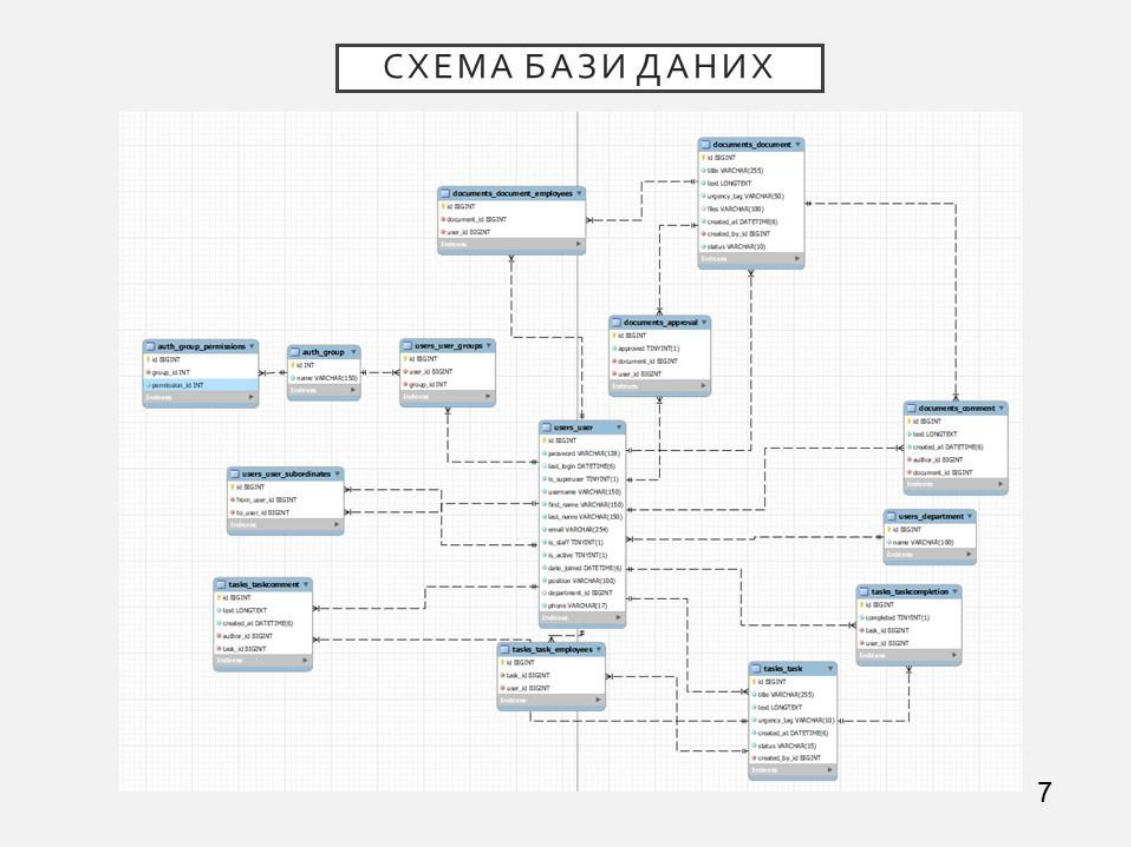
## Порівняння Python Web фреймворків

| Характеристика                     | Flask                                                   | Django                                             |
|------------------------------------|---------------------------------------------------------|----------------------------------------------------|
| 1                                  | 2                                                       | 3                                                  |
| Тип фреймворку                     | Мікрофреймворк                                          | Повноцінний фреймворк                              |
| Простота використання              | Висока (легше для новачків)                             | Середня (може бути складніше для новачків)         |
| Залежності від зовнішніх бібліотек | надає підтримку API (Application Programming Interface) | Django не підтримує API                            |
| Підтримка Visual Debug             | Мінімальні                                              | Багато вбудованих                                  |
| Вбудовані компоненти               | Обмежені (необхідні додаткові розширення)               | Багато (адміністрування, ORM, аутентифікація тощо) |

| 1                       | 2                                      | 3                                             |
|-------------------------|----------------------------------------|-----------------------------------------------|
| Підхід до розробки      | Гнучкий, можливість вибору компонентів | Опініоністичний, стандартизований підхід      |
| Підтримка баз даних     | Підтримка через розширення             | Потужна підтримка вбудованого ORM             |
| Використання шаблонів   | Є, з використанням Jinja2              | Є, з використанням Django Templates           |
| Форма розробки          | Більш вільна, без строгих вимог        | Структурована, з дотриманням певних конвенцій |
| Система адміністрування | Відсутня                               | Потужна система адміністрування               |
| Розширення та плагіни   | Багато доступних розширень             | Менше залежності від сторонніх плагінів       |

6

Рисунок В.6 – Слайд № 6



7

Рисунок В.7– Слайд № 7



8

Рисунок В.8– Слайд № 8



Рисунок В.9– Слайд № 9

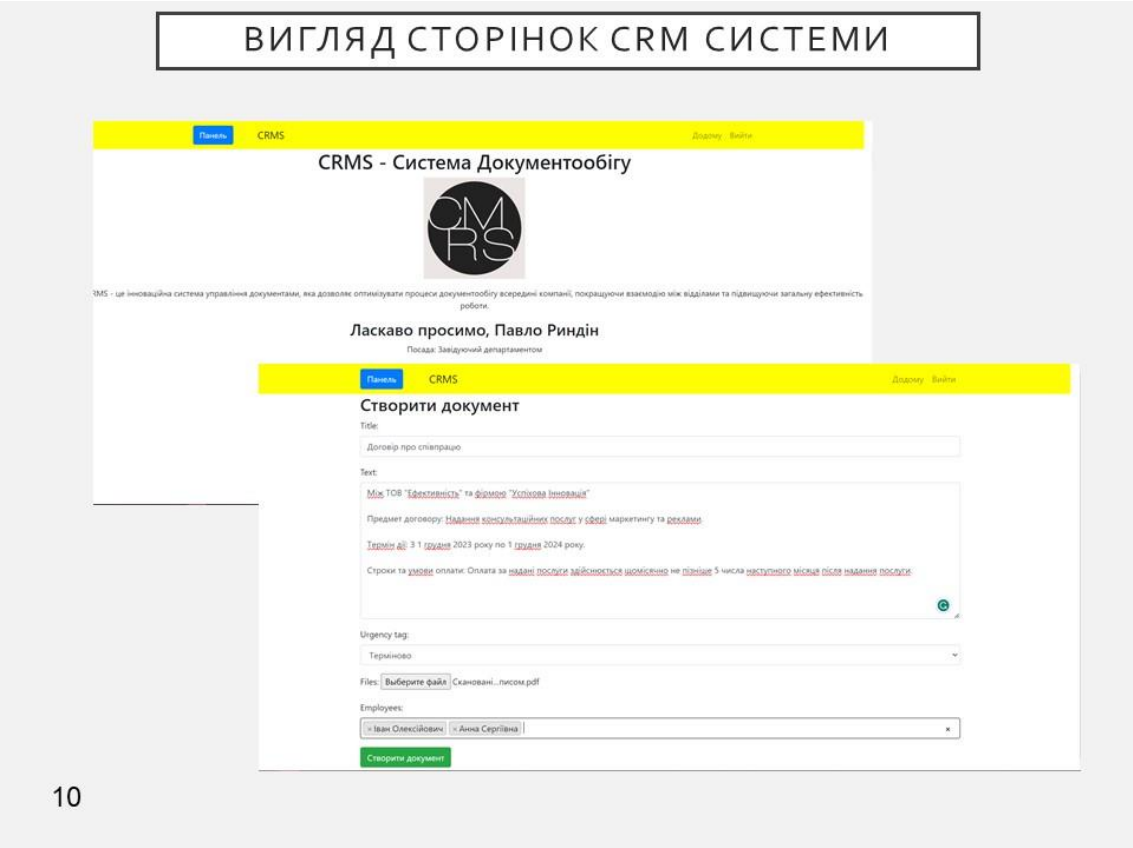
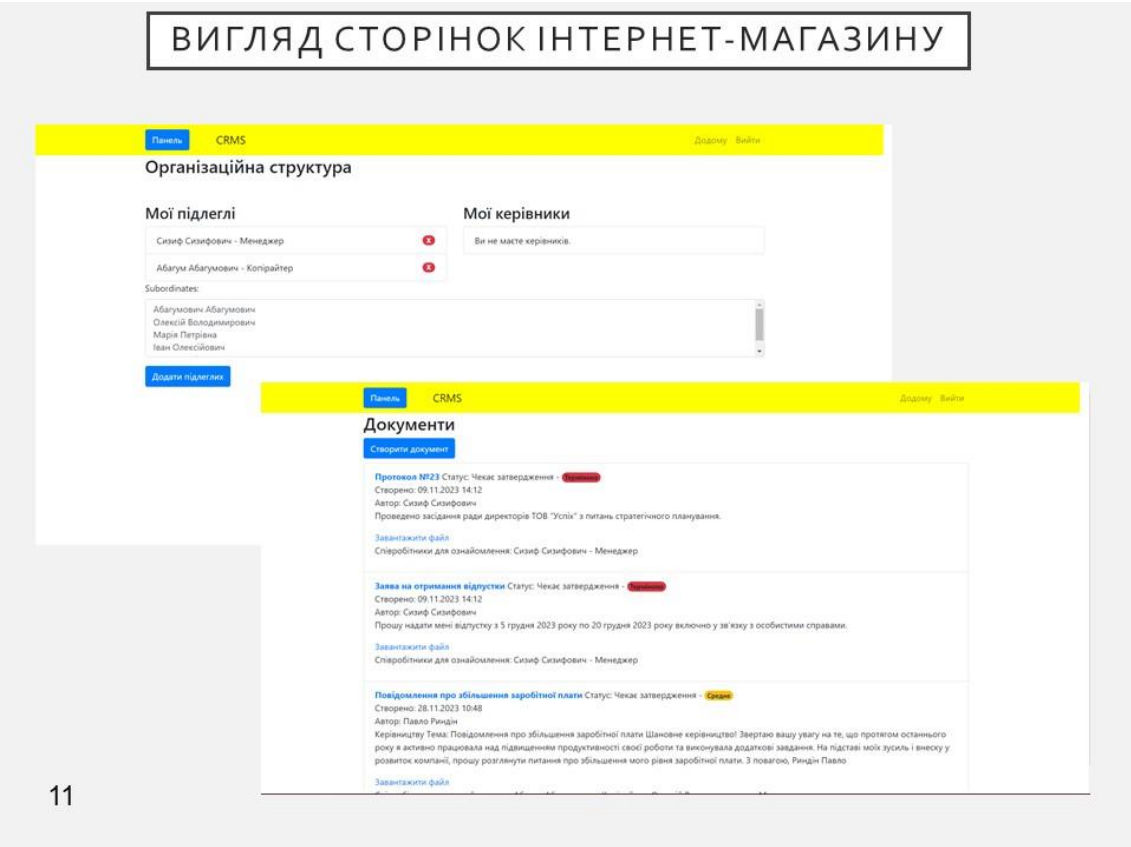


Рисунок В.10– Слайд № 10



11

Рисунок В.11– Слайд № 11

## РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Для тестування CRM-системи було обрано нативне тестування, яке включає в себе використання вбудованих інструментів Django та ручну перевірку функціоналу сервісу.

У результаті тестування всі модулі системи продемонстрували стабільну та надійну роботу. Виявлені помилки були виправлені. Тестування підтвердило, що розроблена CRM система повністю готова до впровадження та використання в реальних умовах.

12

Рисунок В.12– Слайд № 12

## ВИСНОВКИ

За результатами виконання дипломної кваліфікаційної роботи виконано такі задачі:

проведено глибокий аналіз вимог до систем управління документообігом та задачами, враховано особливості робочих процесів у компаніях;

встановлено ключові функціональні вимоги до системи, обрано мову програмування Python та фреймворк Django, використано СУБД MySQL для створення структури бази даних.

Успішно розроблено та впроваджено всі заплановані функції системи, включаючи управління документами, задачами, коментарями, організаційною структурою, а також інтеграцію з Celery та Redis для асинхронних задач;

створено структуру вебсервісу та описано кожний модуль системи. Також описані призначення класів та функцій та їх реалізацій;

Виконано нативне тестування всіх компонентів системи, забезпечивши їх надійність та стабільність. Усі модулі системи були ретельно протестовані та показали високу ефективність.

Розроблена CRM-система є комплексним рішенням, яке відповідає сучасним вимогам до управління документообігом та задачами в компаніях. Система включає в себе інтуїтивно зрозумілий інтерфейс, гнучкість налаштувань, можливість масштабування та високий рівень безпеки. Завдання проекту були виконані повністю, а розроблена система готова до впровадження та ефективного використання.

13

Рисунок В.13– Слайд № 13