

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

Факультет програмних систем та прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2026 р.

Дипломний проєкт

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інженерія програмного
забезпечення мультимедійних та інформаційно-пошукових систем»**

спеціальності 121 Інженерія програмного забезпечення

**на тему: «Програмне забезпечення з вбудованим помічником на основі
штучного інтелекту для управління відносинами з клієнтами»**

Виконав:

студент IV курсу, групи КП-21

Боньковський Ілля Віталійович

Керівник:

старший викладач кафедри ПЗКС, д-р філософії

Юсин Яків Олексійович

Консультант з нормоконтролю:

доцент кафедри ПЗКС, к.т.н., доцент

Онай Микола Володимирович

Рецензент:

доцент кафедри ПСТ ФІТ «КНУ

ім. Тараса Шевченка», д.т.н., с.н.с.

Порєв Геннадій Володимирович

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет програмних систем та прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення
мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2025 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Боньковському Іллі Віталійовичу

1. Тема проєкту «Програмне забезпечення з вбудованим помічником на основі штучного інтелекту для управління відносинами з клієнтами», керівник проєкту Юсин Яків Олексійович, асистент кафедри ПЗКС, доктор філософії, затверджені наказом по університету №1984-С від «28» травня 2026 р.
2. Термін подання студентом проєкту «12» червня 2026 р.
3. Вихідні дані до проєкту: див. Технічне завдання.
4. Зміст пояснювальної записки:
 - огляд та аналіз існуючих рішень;
 - обґрунтування вибору засобів реалізації;
 - розроблення вебсервісу;
 - аналіз розробленого сервісу.
5. Перелік обов'язкового графічного матеріалу:
 - схема бази даних (креслення);
 - функціональність програмного забезпечення (креслення);
 - узагальнена схема архітектури програмного забезпечення (плакат);
 - діаграма послідовності сценарію «Запит рекомендації щодо наступного кроку» (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2025	
2.	Розроблення та узгодження технічного завдання	22.11.2025	
3.	Розроблення архітектури системи та структури баз даних	15.12.2025	
4.	Підготовка матеріалів першого розділу дипломного проєкту	28.12.2025	
5.	Проектування інтерфейсу користувача CRM-системи	01.02.2026	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2026	
7.	Програмна реалізація серверної, клієнтської частин та інтеграція ШІ	11.03.2026	
8.	Тестування програмного забезпечення	18.03.2026	
9.	Підготовка матеріалів третього розділу дипломного проєкту	27.03.2026	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	15.04.2026	
11.	Підготовка графічної частини дипломного проєкту	22.04.2026	
12.	Оформлення документації дипломного проєкту	27.05.2026	

Студент

Ілля БОНЬКОВСЬКИЙ

Керівник проєкту

Яків ЮСИН

АНОТАЦІЯ

У даному дипломному проєкті спроектовано та реалізовано програмне забезпечення із вбудованим інтелектуальним помічником для управління відносинами з клієнтами. Метою розробки є автоматизація операційної діяльності відділів продажу та надання менеджерам інтелектуальної підтримки на основі накопиченої історії взаємодії з клієнтами.

Програмний продукт охоплює повний цикл ведення клієнтської бази: від реєстрації нових клієнтів до моніторингу етапів воронки продажів та керування робочими завданнями. Інтегрований аналітичний модуль Smart Advisor, який взаємодіє з хмарним сервісом Google Gemini API, здатний аналізувати контекст поточних угод, обчислювати ймовірність їх успішного завершення та генерувати покрокові рекомендації для користувачів без необхідності ручного збору даних.

Технічна реалізація проєкту базується на сучасній клієнт-серверній архітектурі. Серверна частина розроблена мовою C# з використанням фреймворку ASP.NET Core. Для забезпечення високої продуктивності та надійності збереження інформації застосовано комбінований підхід: реляційна база Microsoft SQL Server відповідає за строгу узгодженість бізнес-даних, а нереляційна СКБД MongoDB – за збереження системних журналів подій та телеметрію III-запитів. Користувацький вебінтерфейс побудовано у форматі SPA на базі фреймворку Angular із залученням компонентної бібліотеки PrimeNG, що гарантує високу ергономіку та адаптивність візуального середовища. Отримані результати мають високе практичне значення і готові до інтеграції у реальні бізнес-процеси підприємств.

ABSTRACT

This diploma project focuses on the design and implementation of customer relationship management software featuring an embedded intelligent assistant. The main objective of the development is to automate the operational activities of sales departments and provide managers with smart decision support based on the accumulated history of client interactions.

The software product covers the entire lifecycle of client base management, ranging from the registration of new clients to tracking sales pipeline stages and managing daily workflows. The integrated analytical module, Smart Advisor, which interfaces with the Google Gemini API cloud service, is capable of analyzing the context of ongoing deals, calculating the probability of their successful closure, and generating step-by-step recommendations for users without the need for manual data aggregation.

The technical execution of the project is grounded in a modern client-server architecture. The backend is developed in C# utilizing the ASP.NET Core framework. To ensure high performance and robust data storage, a combined persistence approach is applied: a relational Microsoft SQL Server database guarantees strict business data consistency, while a non-relational MongoDB DBMS handles system event logging and AI request telemetry. The user web interface is designed as a Single Page Application (SPA) driven by the Angular framework and the PrimeNG component library, ensuring optimal ergonomics and responsiveness of the visual layout. The obtained results demonstrate significant practical value and are ready for deployment in real-world enterprise business processes.

ДП.045440-01-90 Програмне забезпечення з вбудованим помічником на основі штучного інтелекту для управління відносинами з клієнтами.
Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Програмне забезпечення з	5	
	вбудованим помічником		
	на основі штучного		
	інтелекту для управління		
	відносинами з клієнтами.		
	Технічне завдання		
ДП.045440-03-81	Програмне забезпечення з	66	
	вбудованим помічником		
	на основі штучного		
	інтелекту для управління		
	відносинами з клієнтами.		
	Пояснювальна записка		
ДП.045440-04-51	Програмне забезпечення з	5	
	вбудованим помічником		
	на основі штучного		
	інтелекту для управління		
	відносинами з клієнтами.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмне забезпечення з	11	
	вбудованим помічником		
	на основі штучного		
	інтелекту для управління		
	відносинами з клієнтами.		
	Керівництво користувача		

[illegible]

Факультет програмних систем та прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2025 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ З ВБУДОВАНИМ ПОМІЧНИКОМ
НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ УПРАВЛІННЯ
ВІДНОСИНАМИ З КЛІЄНТАМИ

Технічне завдання

ДП.045440-02-91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Ілля БОНЬКОВСЬКИЙ

ЗМІСТ

1. Найменування та галузь застосування.....	3
2. Підстава для розроблення	3
3. Призначення розробки.....	3
4. Вимоги до програмного продукту	3
5. Вимоги до проєктної документації	4
6. Етапи проєктування	4
7. Порядок тестування розробки.....	5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення з вбудованим помічником на основі штучного інтелекту для управління відносинами з клієнтами.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації операційної діяльності відділів продажу, структурування клієнтських баз та підвищення ефективності ухвалення рішень менеджерами завдяки проактивній аналітиці.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмне забезпечення повинне забезпечувати такі основні функції:

- 1) створення, редагування та видалення профілів клієнтів і компаній;
- 2) створення, редагування та моніторинг комерційних угод;
- 3) управління воронкою продажів та зміна статусів комерційних операцій;
- 4) генерація покрокових рекомендацій для менеджера за допомогою штучного інтелекту;

- 5) прогнозування ймовірності успішного закриття угоди за допомогою штучного інтелекту;
- 6) створення, редагування та контроль виконання робочих завдань;
- 7) пошук і фільтрація даних у реєстрах клієнтів та угод;
- 8) візуалізація ключових метрик ефективності на інформаційній панелі.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

- 1) пояснювальна записка;
- 2) програма та методика тестування;
- 3) керівництво користувача;
- 4) креслення:
 - «Схема бази даних. ER-діаграма»;
 - «Функціональність програмного забезпечення. UML-діаграма прецедентів».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури та аналіз існуючих рішень.....	13.11.2025
Розроблення та узгодження технічного завдання	22.11.2025
Розроблення архітектури системи та структури баз даних	15.12.2025
Проектування інтерфейсу користувача CRM-системи	01.02.2026
Програмна реалізація системи та інтеграція ШІ	11.03.2026
Тестування програмного забезпечення	18.03.2026
Підготовка матеріалів текстової частини проєкту	15.04.2026
Підготовка графічної частини проєкту	22.04.2026
Оформлення технічної документації проєкту	27.05.2026

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет програмних систем та прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2026 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ З ВБУДОВАНИМ ПОМІЧНИКОМ
НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ УПРАВЛІННЯ
ВІДНОСИНАМИ З КЛІЄНТАМИ**

Пояснювальна записка

ДП.045440-03-81

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Ілля БОНЬКОВСЬКИЙ

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ	3
ВСТУП	6
1. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ.....	7
1.1. Salesforce	7
1.2. Pipedrive.....	10
1.3. NetHunt CRM	14
1.4. Висновки до першого розділу.....	17
2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ.....	19
2.1. Вибір засобів реалізації серверної частини.....	19
2.2. Вибір засобів реалізації бази даних	23
2.3. Вибір засобів реалізації клієнтської частини	26
2.4. Висновки до другого розділу	28
3. РОЗРОБЛЕННЯ ВЕБСЕРВІСУ	30
3.1. Загальний опис структури системи.....	30
3.2. Аналіз функціональних вимог до програми.....	32
3.3. Особливості реалізації серверної частини.....	36
3.4. Структура бази даних SQL Server	41
3.5. Структура бази даних MongoDB	49
3.6. Особливості реалізації клієнтської частини.....	51
3.7. Висновки до третього розділу	52
4. АНАЛІЗ РОЗРОБЛЕНОГО СЕРВІСУ	54
4.1. Аналіз реалізованого програмного забезпечення.....	54
4.2. Тестування програмного забезпечення.....	56
4.3. Рекомендації щодо подальшого вдосконалення системи	57
4.4. Висновки до четвертого розділу.....	58
ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	62
ДОДАТКИ.....	66

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ACID (*Atomicity, Consistency, Isolation, Durability*) – набір властивостей транзакцій баз даних (атомарність, узгодженість, ізолюваність, довговічність), що гарантують надійність і передбачуваність роботи з даними.

API (*Application Programming Interface*) – набір визначень та протоколів, що дозволяють різним програмним компонентам взаємодіяти між собою.

BI-система (*Business Intelligence*) – програмне забезпечення для перетворення сирих даних у змістовну та корисну інформацію для бізнес-аналізу та складання звітів.

C# – сучасна, об'єктно-орієнтована мова програмування зі строгою статичною типізацією, розроблена компанією Microsoft.

CORS (*Cross-Origin Resource Sharing*) – механізм безпеки, який дозволяє або забороняє вебсторінкам робити запити до іншого домену, відмінного від того, з якого було завантажено сторінку.

CRM (*Customer Relationship Management*) – система управління відносинами з клієнтами, призначена для автоматизації взаємодії із замовниками, оптимізації бізнес-процесів та збільшення продажів.

DI (*Dependency Injection*) – шаблон проєктування (впровадження залежностей), що використовується для зменшення зв'язності між компонентами програми.

DTO (*Data Transfer Object*) – шаблон проєктування, об'єкт передачі даних, який використовується для інкапсуляції даних та їх пересилання між клієнтом і сервером.

EF Core (*Entity Framework Core*) – об'єктно-реляційний відображувач (ORM) для платформи .NET, який дозволяє розробникам працювати з базами даних через об'єкти мови програмування.

JSON (JavaScript Object Notation) – легкий текстовий формат обміну даними, зручний для читання людиною та машинного аналізу.

JWT (JSON Web Token) – відкритий стандарт для безпечної передачі інформації між сторонами у вигляді об'єкта JSON, що використовується для автентифікації та авторизації без збереження стану сесії на сервері.

LINQ (Language Integrated Query) – технологія в .NET, що додає можливості виконання строго типізованих запитів до колекцій даних або баз даних безпосередньо в мові програмування C#.

MongoDB – провідна документно-орієнтована система керування базами даних (NoSQL), яка зберігає інформацію у вигляді гнучких JSON-подібних документів.

.NET – безкоштовне кросплатформне середовище розробки з відкритим вихідним кодом від Microsoft для створення різноманітних типів програмного забезпечення.

NoSQL – підхід до проєктування баз даних, який передбачає зберігання даних у форматах, відмінних від класичних реляційних таблиць.

REST API (Representational State Transfer) – архітектурний стиль побудови мережових програмних інтерфейсів, що базується на протоколі HTTP та використовує стандартні методи (GET, POST, PUT, DELETE).

RxJS – бібліотека для реактивного програмування за допомогою спостережуваних послідовностей (Observables), що використовується в Angular для правління асинхронними потоками даних.

SaaS (Software as a Service) – модель розповсюдження програмного забезпечення, за якої доступ до хмарного застосунку надається користувачам через інтернет за передплатою.

SAM (Smart Advisor Module) – розроблений у межах даного дипломного проєкту модуль розумного помічника на основі алгоритмів штучного інтелекту.

SPA (Single Page Application) – односторінковий вебзастосунок, який завантажує єдиний HTML-документ і динамічно оновлює його вміст під час взаємодії користувача з системою без перезавантаження сторінки.

SQL Server – реляційна система керування базами даних, розроблена компанією Microsoft.

Webhooks – механізм отримання сповіщень у реальному часі про певні події в системі шляхом відправлення HTTP-запитів на задану URL-адресу.

Вебформа – елемент графічного інтерфейсу вебзастосунку, призначений для введення та відправки користувачем даних на сервер.

Ворона продажів – концептуальна модель, яка відображає шлях клієнта від першого контакту до успішного закриття комерційної угоди.

Kanban-дошка – інструмент візуалізації робочого процесу, де об'єкти представлені у вигляді карток, що переміщуються між стовпцями, які відповідають етапам процесу.

Лід (Lead) – потенційний клієнт, особа або компанія, що виявила інтерес до послуг чи товарів компанії.

Prompt-інженерія – процес структурування оптимізації та налаштування текстових запитів (промптів) для максимально точної та ефективної взаємодії з великими мовними моделями.

Рендеринг – процес програмної генерації та візуалізації графічного інтерфейсу на екрані пристрою користувача.

СКБД (Система керування базами даних) – комплекс програмного забезпечення, призначений для створення, редагування, керування та адміністрування даних у базах даних.

Фреймворк (Framework) – програмний каркас, комплексний набір бібліотек та готових архітектурних рішень, що полегшує і прискорює розробку програмного забезпечення.

ВСТУП

У сучасних умовах динамічного розвитку бізнесу ефективне управління відносинами з клієнтами є ключовим фактором конкурентоспроможності будь-якої компанії. Системи управління відносинами з клієнтами стали невід'ємною частиною корпоративної інфраструктури, дозволяючи зберігати інформацію про клієнтів, відстежувати історію взаємодії та координувати комерційні угоди.

Проте більшість класичних CRM-систем функціонують переважно як пасивні бази даних, які вимагають від менеджерів значних часових витрат на ручне введення інформації, фіксацію нотаток та планування подальших кроків. У таких умовах фахівці часто пропускають важливі деталі комунікацій, а керівництво компанії позбавлене можливості оперативно отримувати аналітику щодо успішності майбутніх угод. Вирішенням цієї проблеми є інтеграція сучасних технологій штучного інтелекту, зокрема великих мовних моделей, безпосередньо в архітектуру CRM. Це дозволяє перетворити систему з інструменту простого обліку на інтелектуального помічника. Таким чином, розроблення програмного забезпечення з вбудованим помічником на основі штучного інтелекту для управління з відносинами з клієнтами є надзвичайно актуальним інженерним завданням.

Метою проєкту є розроблення та дослідження ефективного програмного забезпечення для управління відносинами з клієнтами із вбудованим інтелектуальним модулем, який автоматизує рутинні процеси обліку, структурує історію взаємодії та надає рекомендацій для підтримки прийняття рішень менеджерами з продажів.

1. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. Salesforce

Salesforce – це багатофункціональна хмарна платформа, яка є світовим лідером у сегменті програмного забезпечення для управління відносинами з клієнтами. Система орієнтована на широкий сегмент користувачів – від малого бізнесу до великих корпорацій. Salesforce реалізована на основі моделі SaaS, що допомагає компаніям уникати витрат на підтримку власної інфраструктури, отримуючи доступ до інструментів автоматизації через веббраузер [1]. Основна мета – це створення єдиного інформаційного простору для відділів продажу, маркетингу та підтримки, що дозволяє автоматизувати повний цикл взаємодії з клієнтами.

До основних функціональних можливостей платформи належать:

- управління потенційними клієнтами та активними угодами;
- ведення детальної бази контактів та організацій-контрагентів;
- автоматизація бізнес-процесів за допомогою інструменту Flow Builder;
- інтелектуальна аналітика та прогнозування на базі штучного інтелекту Einstein AI;
- можливість розширення функціональності через платформи з розширеннями AppExchange.

Однією з основних переваг системи є її архітектура, яка базується на гнучкій об'єктній моделі даних. ЦентRALЬНОЮ СУТНІСТЮ виступає «Account» (обліковий запис компанії), до якої через зв'язок «один-до-багатьох» прив'язуються контакти, угоди та задачі [1].

Процес взаємодії з системою Salesforce починається з авторизації, після чого користувач має доступ до робочого простору, де логіка управління продажами реалізована через етапи життєвого циклу угоди.

Кожна угода проходить шлях від первинної кваліфікації до фінального закриття, за рахунок чого користувач має змогу чітко

відстежувати конверсію на кожному рівні воронки продажів. Приклад такого робочого простору з візуалізацією поточного стану угоди наведено на рис. 1.1.

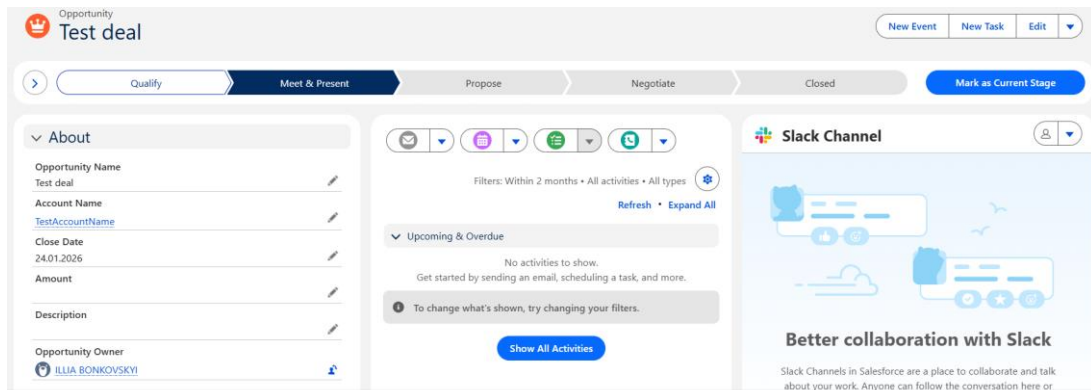


Рис. 1.1. Робочий простір керування угодою в системі Salesforce

Рішення щодо інтерфейсу платформи націлені на візуалізацію великих масивів даних. Основним інструментом для взаємодії менеджера з даними є гнучкі схеми відображення сутностей – Page Layouts, які дозволяють адаптувати інтерфейс під потреби бізнес-процесів. Робота з масивами інформації уніфікована для наявних сутностей системи і реалізована через спискові (табличні) представлення. Приклад такої таблиці наведено на рис. 1.2. Таблиці забезпечують можливість групування, фільтрації та швидкого доступу до об'єктів системи, таких як угоди, клієнти або компанії.

	Name	Title	Company	Phone	Email	Lead Status	Owner
1	Ivan Ivaniv		KPI	0683391383	iliabonkovskyi@gmail.com	Contacted	IBONK

Рис. 1.2. Табличне відображення продажів в системі Salesforce

Центральним елементом архітектури інтерфейсу є деталізована картка запису, яка зберігає всю критично важливу інформацію: від метаданих до повної історії комунікації чи пов'язаних документів [2, 3]. Даний підхід забезпечує узгодженість при роботі з об'єктами та дозволяє підтримувати цілісність даних у межах системи.

Для моніторингу активності система містить інтегровані інформаційні панелі та календарі, що відображають ключові показники ефективності у реальному часі (рис. 1.3). Дизайн системи сприяє зменшенню навантаження на користувача завдяки контекстним підказкам та автоматичним заповненню полів на основі попередньо введеної інформації.

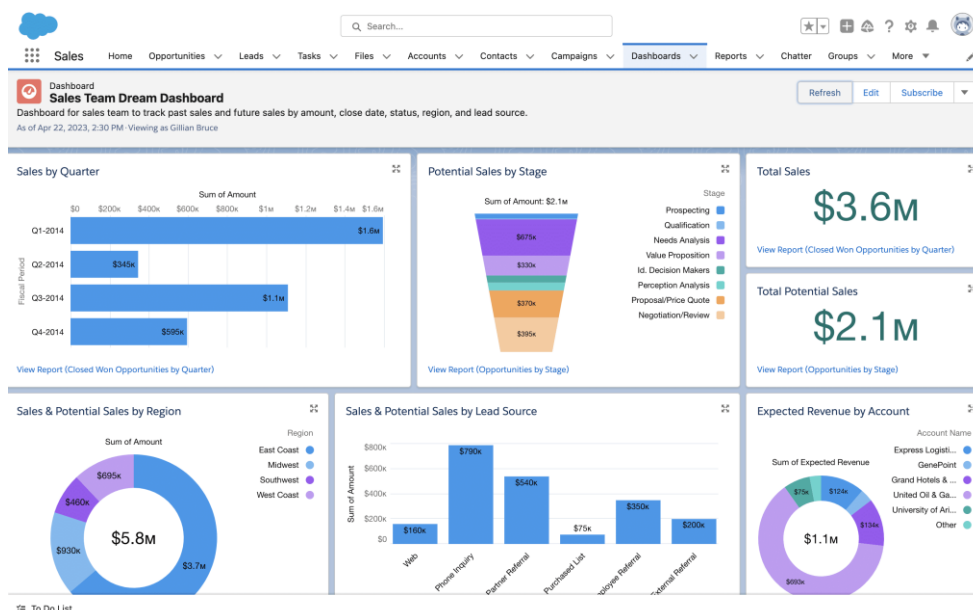


Рис. 1.3. Інформаційні панелі Salesforce

Окремої уваги в контексті розробки програмного забезпечення з модулем розумного помічника потребує інтелектуальна складова платформи на базі штучного інтелекту – Einstein AI. Цей модуль використовує алгоритми машинного навчання для аналізу даних та прогнозування ймовірності закриття угод. Штучний інтелект автоматично оцінює клієнтів та продажі, виділяючи найбільш перспективні контакти, та надає менеджеру рекомендації щодо наступних кроків [1]. Такий підхід

дозволяє автоматизувати рутинні аналітичні операції, впливає на результативність роботи менеджера з продажів, проте ефективність роботи AI-складової прямо залежить від обсягу та якості накопичених даних у системі.

Додатковою перевагою Salesforce також є її розвинена екосистема інтеграцій з іншими бізнес-платформами: месенджерами (Slack, Microsoft Teams), інструментами управління проєктами (Jira, Asana) та сервісами автоматизації маркетингу (HubSpot). Це дозволяє використовувати CRM як центральний вузол обміну даними в межах бізнесу.

Попри високу технологічність, Salesforce має низку недоліків, що роблять її не ідеальним рішенням конкретно для малого та середнього бізнесу. По-перше, система є «важкою» з технічної точки зору, що спричиняє затримки при поданні деяких елементів інтерфейсу. По-друге, система насичена надмірною функціональністю, що створює високий поріг входження. Також варто зазначити відсутність повноцінної безкоштовної версії і високу вартість підписки, що є критичним фактором для стартапів.

1.2. Pipedrive

Pipedrive є спеціалізованим рішенням, розробленим на основі моделі SaaS, створене з акцентом на потреби команд з активних продажів. На відміну від більшості платформ, дана система базується на методології «Activity-based selling», яка має фокус на дії, що призводять до закриття угод. Це визначає архітектуру системи як інструмент, що не просто зберігає дані, а активно сприяє результату користувача під час робочого процесу [4].

Основна функціональність платформи включає в собі [5]:

- налаштування воронки продажів та її візуалізація;
- робота з угодами та завданнями спільно з колегами;
- створення, фільтрування, сортування та сегментування лідів;
- перегляд історії всіх викликів, електронних листів, зустрічей, нотаток тощо;

- звітність стосовно продажів та основних завдань;
- прогнозування щодо обсягів продажів і доходу на основі воронки продажів.

Структура збереження даних в Pipedrive побудована на чіткій ієрархії чотирьох основних сутностей:

- 1) організація;
- 2) контактна особа;
- 3) угода;
- 4) дія («Activity»).

З інженерної точки зору, особливістю є те, що «Угода» є не статичним записом, а динамічним об'єктом, який обов'язково повинен мати заплановану подію. Якщо система не містить запланованої наступної дії для угоди, вона автоматично маркуватиме її як критичну, що вимагає негайної уваги менеджера. Така логіка забезпечує високу цілісність бізнес-процесів та запобігає втраті потенційних клієнтів із-за людського фактору, що є безперечною перевагою даного програмного забезпечення.

Візуальна концепція Pipedrive, порівняно з Salesforce, є більш мінімалістичною. Інтерфейс побудований навколо «Pipeline View» – багаторівневої воронки продажів, Kanban-дошки, яка з точки зору користувацького досвіду мінімізує кількість переходів між сторінками. Користувач взаємодіє з угодами за допомогою механіки drag-and-drop, що дозволяє інтуїтивно переміщувати об'єкти між етапами. Такий архітектурний підхід тісно пов'язаний з методологією продажів, орієнтованою на дії. Кожна картка угоди на Kanban-дошці містить візуальні індикатори стану. Крім того, деталізація інформації по конкретній угоді відбувається у спливаючих бічних панелях, що дозволяє працювати з даними клієнта, не втрачаючи загального візуального контексту всієї воронки продажів (рис. 1.4).

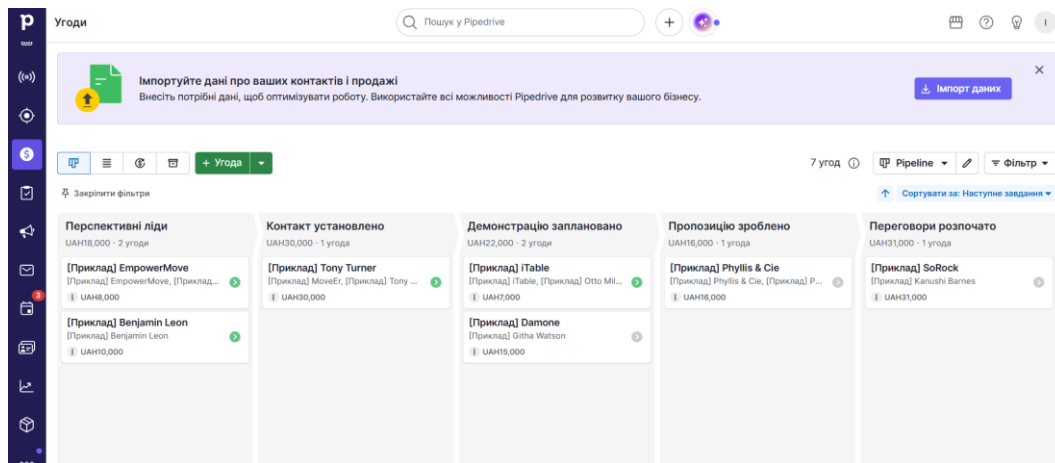


Рис. 1.4. Головний інтерфейс воронки продажів Pipedrive

Для роботи з іншими сутностями (лідери, завдання) за замовчуванням налаштоване представлення у вигляді таблиці, приклад якої продемонстровано на рис. 1.5.

Усі	Виклик	Зустріч	Завдання	Термін виконання	Ел. пошта	Діловий обід	Справи	Прострочені	Сьогодні	Завтра	Цього тижня	Наступного тижня	Оберіть період
☐	Виконано	Тема	Угода	Пріоритет	Контактна особа	Адреса електронної пошти	Номер телефону	Ори:					
☐	☐	[Приклад] Final attempt	[Приклад] Benjamin Leon		[Приклад] Benjamin Leon	benjamin.leon@gmail.com (Робочий)	785-202-7824						
☐	☐	[Приклад] Context call	[Приклад] Tony Turner		[Приклад] Tony Turner	tony.turner@moveer.com (Робочий)	218-348-8528	[Приклад]					
☐	☐	[Приклад] Showing	[Приклад] iTable		[Приклад] Otto Miller	otto.miller@itablee.eu (Робочий)	601-906-7534	[Приклад]					
☐	☐	[Приклад] Final contact atte...	[Приклад] EmpowerMove		[Приклад] Gloria Quinn	gloria.quinn@empowermove.com (Робочий)	862-252-9773	[Приклад]					
☐	☐	[Приклад] New proposal	[Приклад] SoRock		[Приклад] Kanushi Barnes	kanushi.barnes@gmail.com (Робочий)							
☐	☐	[Приклад] Send proposal	[Приклад] Damone		[Приклад] Githa Watson	githa.watson@odamoneet.com (Робочий)	260-414-1246						
☐	☐	[Приклад] Proposal feedback	[Приклад] Phyllis & Cie		[Приклад] Phyllis Yang	phyllis.yang@gmail.com (Робочий)	240-707-3884	[Приклад]					
☐	☐	[Приклад] Send proposal			[Приклад] Peru Zitan	peru.zitan@ioreannn.ee (Робочий)	110-123-1246	[Приклад]					

Рис. 1.5. Таблиця з даними «Завдання» на платформі Pipedrive

Pipedrive містить власний модуль ШІ помічника – AI Sales Assistant. Дана надбудова аналізує поведінкові шаблони користувачів та історичні дані про закриті угоди. Також наявне окреме вікно наявного менеджера (рис. 1.6), де менеджер може надсилати окремі власні запити та отримувати рекомендації. Завдяки цьому інструменту відділ продажів має змогу автоматизувати підготовку до переговорів та швидше знаходити вузькі місця у воронці. Крім того, така інтерактивність суттєво знижує когнітивне навантаження на співробітників під час обробки великих обсягів клієнтських даних.

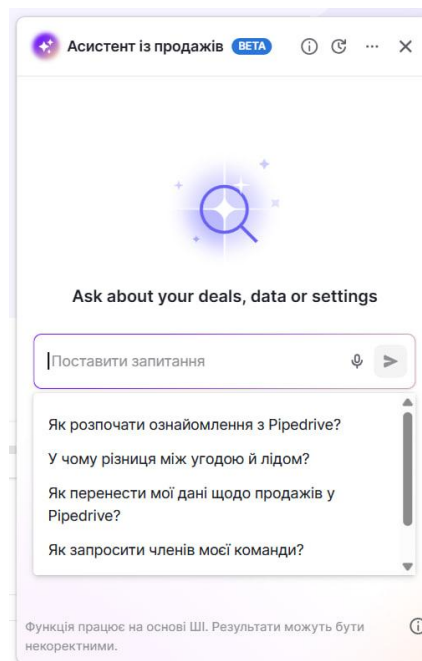


Рис. 1.6. Модульне вікно рекомендацій та запитів AI Sales Assistant

Крім того, система використовує штучний інтелект для функції «Smart Contact Data» – технологія, що дозволяє автоматично знаходити та підтягувати додаткову інформацію про клієнта з відкритих джерел лише за допомогою електронної пошти [4]. Це суттєво автоматизує рутинний процес збору даних, що є однією з цілей розробки даного дипломного проєкту.

Екосистема Pipedrive Marketplace включає понад 400 готових інтеграцій, серед яких ключове місце займають сервіси Google та Microsoft. Система підтримує двосторонню синхронізацію пошти та календарів, що дозволяє автоматично фіксувати кожну взаємодію в картці клієнта. Для розробників надається REST API, що дозволяє будувати складні автоматизації та інтегрувати CRM у специфічні умови підприємств [6].

Попри очевидні переваги у візуалізації продажів, у ході аналізу програмного забезпечення було виявлено низку обмежень. Pipedrive ідеально підходить для фази активного продажу, але майже не містить функціональності для роботи після закриття угод, що призводить до того, що менеджер часто змушений переносити дані в іншу систему. Також варто відзначити, що повний доступ до ШІ-інструментів та розширеної звітності

відкривається лише в дорогих тарифних планах. Крім того, були виявлені певні недоліки, пов'язані з інтерфейсом системи:

- під час рендерингу деяких елементів інтерфейсу спостерігаються уповільнення;
- попри наявність української мови в системі, певні елементи залишаються не адаптованими (наприклад, під час роботи з AI Sales Assistant);
- наявність неточностей в помилках валідації під час додавання сутностей у систему.

1.3. NetHunt CRM

NetHunt CRM – це сучасна українська CRM-система, яка реалізувала концепцію гібридного робочого простору. На відміну від Salesforce, яка є масивною автономною екосистемою, NetHunt побудована за принципом гнучкого конструктора сутностей. Система може функціонувати як окремий вебзастосунок (рис. 1.7) або інтегруватись усередині Google Workspace, що дозволяє користувачу керувати бізнес-процесам безпосередньо з поштової скриньки [7].

З точки зору проєктування систем управління даними, NetHunt CRM пропонує модель, де базовою структурною одиницею є «Папка». В інженерному розумінні папка виступає як сутність або ж таблиця бази даних, що містить набір записів. На рис. 1.7 продемонстровано, що система пропонує попередньо налаштовані папки для основних сутностей: угоди, контакти і компанії. Така архітектура дозволяє розробнику або адміністратору системи гнучко створювати нові типи полів без необхідності прямої маніпуляції над схемою бази даних [8]. Це суттєво знижує технічний поріг для адаптації системи та дозволяє підприємствам швидко масштабувати структуру даних відповідно до зміни своїх бізнес-процесів.

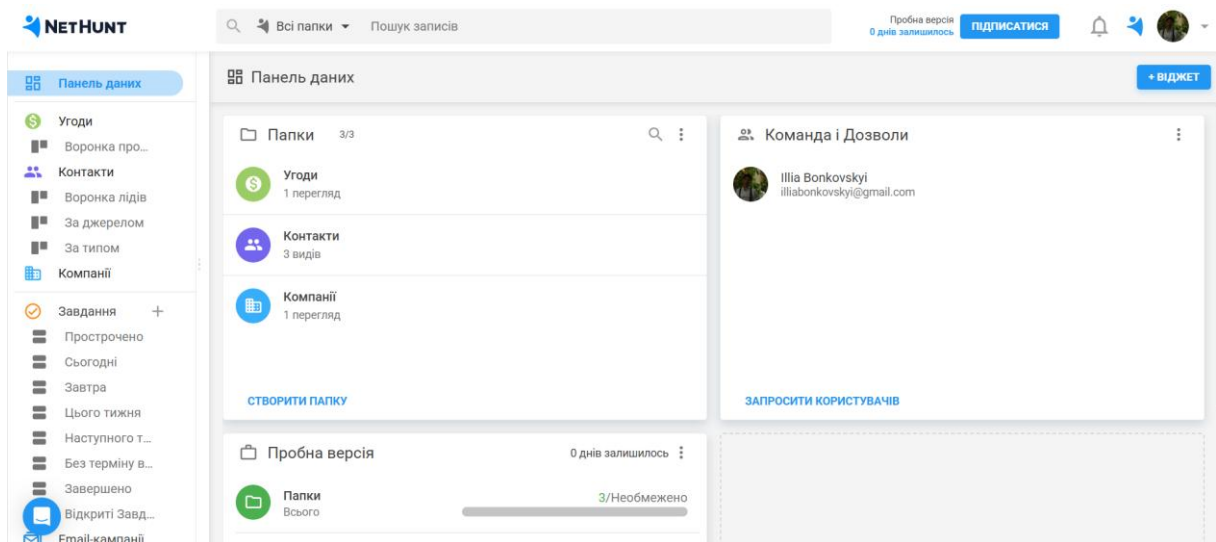


Рис. 1.7. Панель даних та структура основних сутностей NetHunt CRM

Дизайн даної системи реалізований за принципом безшовної інтеграції. Основні елементи управління CRM вбудовані безпосередньо в бічну панель Gmail [9]. На рис. 1.8 наведено приклад того, як менеджер може працювати з базою даних, не виходячи з контексту робочого листування. Крім контекстної панелі, система підтримує візуалізацію процесів у вигляді Kanban-дошок, що дозволяє органічно вписати воронку продажів у загальну стилістику продукту.

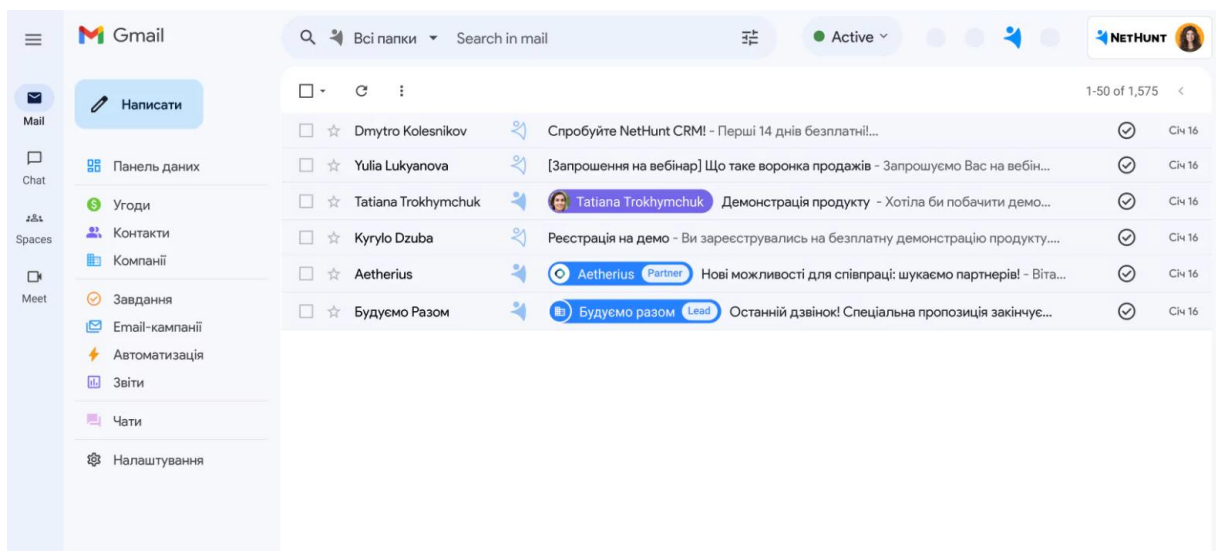


Рис. 1.8. Інтеграція інтерфейсу NetHunt CRM у бічну панель Gmail [10]

В модулі «Угоди» основним інструментом візуалізації виступає воронка продажів, яка реалізована як динамічна Kanban-дошка (рис. 1.9), де етапи відповідають логічним бізнес-процесам компанії.

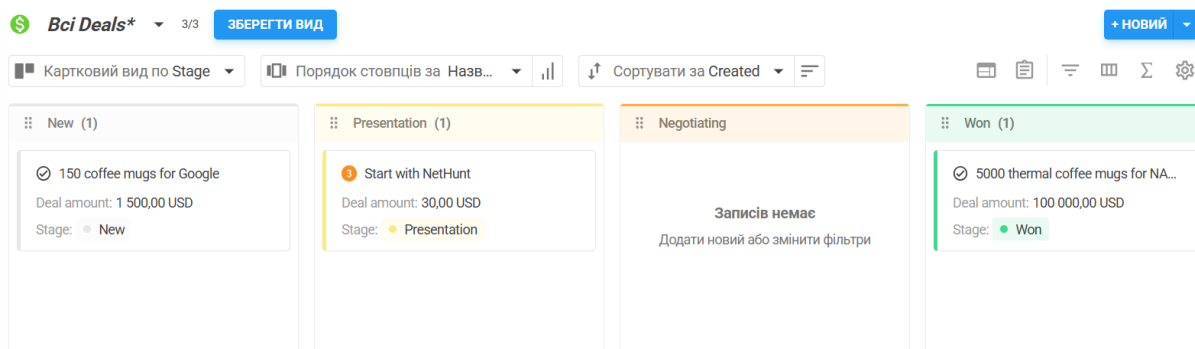


Рис. 1.9. Візуалізація воронки продажів за допомогою Kanban-дошки

Важливою складовою NetHunter CRM є вбудований рушій автоматизації, інтерфейс якого зображено на рис. 1.10. Він дозволяє налаштовувати тригерні події (наприклад, зміна етапу угоди, отримання листа, заповнення вебформи) та прив'язувати до них ланцюжки автоматичних дій (відправка електронних листів, створення задачі) [11].



Рис. 1.10. Конструктор автоматизації бізнес-процесів

Крім того, система надає відкритий REST API та підтримує Webhooks, що дозволяє інтегрувати її з іншими системами.

Аналітичний модуль системи генерує звіти в режимі реального часу, спираючись на дані з папок. Користувачу доступні звіти щодо ефективності менеджерів, прогнозування продажів та аналіз причин втрати угод [12].

Незважаючи на переваги архітектури та високий ступінь гнучкості, були спостережені певні обмеження системи NetHunt CRM:

- перевантаження інтерфейсу, що буде спричинене масштабуванням бізнесу та збільшенням кількості власних налаштованих полів через глибоку інтеграцію в Gmail;
- базові звіти системи не завжди задовольняють потреби складних бізнес-процесів, вимагаючи експорту даних у зовнішні BI-системи;
- відсутність інтелектуального асистування, у системі відсутні модулі машинного навчання, які б могли аналізувати історію комунікацій та генерувати інтелектуальні підказки щодо наступних кроків менеджера.

1.4. Висновки до першого розділу

У першому розділі було проведено огляд та детальний аналіз провідних на ринку систем управління відносинами з клієнтами, зокрема Salesforce, Pipedrive та NetHunt CRM. Дослідження показало, що існуючі рішення пропонують різноманітні архітектурні підходи – від масивних хмарних екосистем до гібридних рішень, вбудованих у поштові сервіси.

Кожна з розглянутих систем має вагомі переваги, але водночас містить критичні обмеження для певних бізнес-сценаріїв. Платформа Salesforce володіє гнучкою об'єктною моделлю та потужним інструментарієм на базі штучного інтелекту Einstein AI. Проте ця система перевантажена надмірною функціональністю, що створює високий поріг входження, є технічно важкою та має високу вартість підписки, що робить її неоптимальним вибором для стартапів та малого бізнесу.

Система Pipedrive, побудована на методології фокусування на діях, пропонує зручну візуалізацію воронки продажів та власний ШІ-помічник. Незважаючи на це, вона майже не містить інструментів для роботи з клієнтами після закриття угод, а повний доступ до її інтелектуальних інструментів відкривається лише в найдорожчих тарифних планах. Зі свого боку, гібридне рішення NetHunt CRM забезпечує зручну безшовну інтеграцію з Gmail. Однак при масштабуванні бізнесу та збільшенні кількості полів її інтерфейс стає перевантаженим. Крім того, у цій системі повністю відсутні модулі машинного навчання, які могли б проактивно аналізувати історію комунікацій та генерувати підказки менеджеру.

Отже, спільним слабким місцем багатьох існуючих систем є або висока вартість доступу до інтелектуальних функцій, або перевантаженість інтерфейсу, або повна відсутність ШІ-асистування, що залишає CRM виключно інструментом збереження даних.

На основі виявлених недоліків обґрунтовано актуальність розроблення власного програмного забезпечення для управління відносинами з клієнтами. Нова система повинна вирішувати проаналізовані проблеми шляхом поєднання інтуїтивно зрозумілого для користувача інтерфейсу із вбудованим помічником на основі штучного інтелекту, який буде здатний автоматично аналізувати історію комунікацій і надавати інтелектуальні рекомендації щодо наступних кроків, значно підвищуючи ефективність роботи відділу продажів.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

Реалізація сучасного програмного забезпечення для управління відносинами з клієнтами з інтегрованим модулем штучного інтелекту вимагає ретельного підбору технічного стеку. Зважаючи на складність бізнес-логіки, необхідність забезпечення високого рівня безпеки комерційних даних та потребу в подальшій масштабованості системи, процес розроблення було розділено на три ключові складові:

- серверну частину;
- систему управління базами даних;
- клієнтську частину.

2.1. Вибір засобів реалізації серверної частини

Серверна частина є базовим компонентом CRM-системи, оскільки вона відповідає за обробку бізнес-логіки, взаємодію з базою даних, автентифікацію користувачів та інтеграцію з великими мовними моделями. Для розробки серверної частини було обрано платформу .NET та мову програмування C#.

2.1.1. Платформа .NET та мова C#

Платформа .NET – це безкоштовне кросплатформне середовище розробки з відкритим вихідним кодом для створення різноманітних типів застосунків [13]. Сучасна платформа об'єднує найкращі риси класичного .NET Framework та інноваційного .NET Core, забезпечуючи єдиний набір бібліотек та інструментів розробки для Windows, Linux, macOS [14].

З інженерної точки зору архітектура .NET базується на двох основних компонентах:

- середовище виконання – відповідає за завантаження коду, керування пам'яттю та асинхронне виконання потоків [13];

- бібліотека базових класів – надає розробнику набір типів для роботи з файловими системами, мережевими протоколами, колекціями даних [14].

Мова програмування C# є флагманською мовою платформи .NET. Це сучасна, об'єктно-орієнтована мова зі строгою статичною типізацією, яка була спроектована для підвищення продуктивності розробника при збереженні високого рівня контролю над роботою програми [15].

Основною перевагою вибору даного стеку для розробки SAM CRM є висока продуктивність та масштабованість. За результатами незалежних тестів (наприклад, TechEmpower), .NET є однією з найшвидших серверних платформ у світі. Використання асинхронної моделі програмування дозволяє серверу обробляти тисячі одночасних запитів від менеджерів без блокування ресурсів, що є критичним для систем із великою кількістю активних угод [15].

Крім того, сувора типізація C# дає перевагу у надійності, мінімізуючи ризик виникнення помилок при роботі з фінансовими даними. Більшість логічних недоліків виявляються ще на етапі компіляція, а під час роботи користувача.

Платформа містить вбудовану підтримку сучасних стандартів автентифікації та авторизації (OAuth2, JWT), а також захист від поширених вразливостей, таких як SQL-ін'єкції або XSS-атаки [16]. У контексті зберігання бази клієнтів та комерційної таємниці цей фактор був першочерговим.

Додатковою перевагою є потужна екосистема NuGet. Менеджер пакетів NuGet надає доступ до тисяч бібліотек, що дозволяє легко інтегрувати в систему засоби для генерації PDF-звітів, експорту даних в Excel або роботи з HTTP-клієнтами для взаємодії з ШІ [17].

Важливою рисою C# є наявність технології LING – Language Integrated Query. Вона дозволяє розробнику писати запити до колекцій даних або бази даних безпосередньо у коді мовою C#, використовуючи

строгу типізацію. Це значно спрощує реалізацію фільтрації клієнтів, пошуку за угодами та аналітичної обробки даних, оскільки структура запиту перевіряється компілятором [16].

Для розробки SAM CRM використовується середовище виконання .NET 8, оскільки воно забезпечує найкращу підтримку як контейнеризації, так і хмарних технологій. Використання даного стеку дозволить створити систему, яка зможе легко масштабуватись від стартапу до великої корпорації з тисячами користувачів, забезпечуючи при цьому стабільність та швидкість відгуку інтерфейсу.

Отже, поєднання платформи .NET та мови C# створює надійний фундамент для серверної частини системи, дозволяючи реалізувати складну бізнес-логіку управління відносинами з клієнтами.

2.1.2. Фреймворк ASP.NET Core

Для безпосередньої розробки серверної частини (вебсервісу) системи SAM CRM було обрано фреймворк ASP.NET Core. Це сучасний, високопродуктивний, кросплатформний фреймворк з відкритим вихідним кодом, створений компанією Microsoft для розробки сучасних хмарних вебзастосунків та мікросервісів [18].

У контексті розробки платформи SAM CRM, даний фреймворк використовується для створення RESTful API. Архітектурна модель системи передбачає чітке розділення на клієнтську та серверну частини. Сервер не займається рендерингом HTML-сторінок, а функціонує виключно як постачальник даних. Він приймає HTTP-запити від клієнтської частини, обробляє їх, виконує необхідну бізнес-логіку або звернення до бази даних і повертає результат.

Важливою архітектурною особливістю ASP.NET Core є вбудована підтримка шаблону «Впровадження залежностей». Цей механізм є ключовим для реалізації чистої, багатошарової архітектури [19].

Обробка запитів у фреймворку базується на концепції проміжного програмного забезпечення – Middleware. Життєвий цикл кожного HTTP-запиту проходить через конвеєр послідовно з'єднаних компонентів [18]. Для системи SAM CRM цей конвеєр буде спрямований виконання таких завдань, як налаштування дозволів на прийняття запитів від клієнтської частини застосунку, глобальна обробка помилок та логування.

Безпека та контроль доступу до API є критично важливими для захисту інформації в CRM-системі. В ASP.NET Core цей механізм реалізовано на базі JSON Web Tokens [20]. Цей підхід працює без збереження станів, що звільняє пам'ять сервера від обробки класичних сесій. Після успішної авторизації генерується токен із даними користувача, яким супроводжуються всі подальші запити. Для безпечного зберігання паролів використовується сучасне криптографічне хешування.

За мережеву продуктивність відповідає інтегрований асинхронний вебсервер Kestrel, здатний обробляти десятки тисяч паралельних з'єднань [21]. А маршрутизація на основі атрибутів дозволяє гнучко проєктувати шляхи до ресурсів.

Отже, використання ASP.NET Core надає надійний набір інструментів, який бере на себе базову маршрутизацію та безпеку. Це дозволяє сконцентрувати зусилля на реалізації специфічної бізнес-логіки системи та алгоритмів штучного інтелекту.

2.1.3. Інтеграція модуля штучного інтелекту

Ключовою особливістю розроблюваної системи SAM CRM, що відрізняє її від традиційних систем управління клієнтами, є наявність вбудованого інтелектуального помічника. Його завдання – трансформація CRM з пасивного інструменту збереження даних у платформу, здатну аналізувати історію комунікацій та генерувати рекомендації щодо наступних кроків.

Для реалізації цієї функціональності було прийнято рішення інтегрувати хмарний сервіс Gemini API від компанії Google. Вибір великої мовної моделі Gemini обґрунтований кількома вагомими факторами. По-перше, моделі цього покоління (зокрема, Gemini 1.5 Flash) оптимізовані для виконання високошвидкісних логічних висновків та роботи з великими обсягами контексту [22]. По-друге, платформа Google-AI Studio надає доступ до безкоштовного рівня використання API для розробників. Це робить дане рішення ідеальним для етапу розробки, тестування та запуску застосунку без залучання додаткових інфраструктурних витрат.

Крім того варто згадати простоту інтеграції з платформою .NET. Gemini API надає стандартизований RESTful інтерфейс. Це дозволяє легко інтегрувати його в серверну частину.

Вибір хмарного API для серверної інтеграції гарантує, що секретні ключі доступу ніколи не потраплять до клієнтського застосунку. Крім того такий підхід дозволяє серверу попередньо фільтрувати та анонімізувати чутливі комерційні дані перед їх відправкою до мовної моделі, забезпечуючи дотримання політик конфіденційності [23].

Підсумовуючи, вибір сервісу Gemini API є оптимальним рішенням, яке поєднує в собі доступність для розробника, потужні аналітичні можливості та легкість інтеграції з обраним серверним стеком технологій. Це дозволяє зосередитися на бізнес-логіці CRM-системи, переклавши складні обчислювальні процеси обробки природної мови на надійну хмарну інфраструктуру.

2.2. Вибір засобів реалізації бази даних

Для забезпечення високої масштабованості та продуктивності системи на рівні збереження даних було застосовано комбінований архітектурний підхід. Сучасні корпоративні системи генерують різноманітні масиви інформації, тому використання єдиної бази даних часто стає слабким місцем системи. З метою оптимізації було прийнято рішення про

використання різних типів систем керування базами даних паралельно. Всі дані в контексті розроблювальної CRM-системи умовно поділяються на дві категорії: жорстко структуровані бізнес-дані та гнучкі неструктуровані системні дані. Відповідно до цього поділу було обрано використання реляційної СКБД Microsoft SQL Server як основного сховища та документно-орієнтованої бази MongoDB як допоміжного сховища.

2.2.1. Реляційна база даних Microsoft SQL Server

Для управління основною інформацією в бізнес-інформацією системи SAM CRM було обрано реляційну систему керування базами даних – Microsoft SQL Server. Вибір реляційної моделі зумовлений природою самої предметної області управління відносинами з клієнтами. Модель даних CRM складається з великої кількості взаємопов'язаних сутностей. Наприклад, один клієнт може мати кілька контактних осіб, з якими пов'язані кілька комерційних угод, кожна з яких містить історію змін статусів та прикріплені задачі. Реляційна база даних ідеально описує такі відношення через механізми первинних та зовнішніх ключів, гарантуючи нормалізацію даних.

Головною перевагою вибору Microsoft SQL Server є глибока інтеграція з платформою .NET. Використання бази даних від Microsoft забезпечує найкращу сумісність із серверним стеком на базі ASP.NET Core. Взаємодія з базою реалізується за допомогою засобу об'єктно-реляційного відображення – Entity Framework Core (EF Core) [24]. Ця технологія дозволяє розробнику проєктувати таблиці бази даних на класи мови C#, а складні SQL-запити генерувати за допомогою запитів LINQ. Крім того, EF Core надає зручний інструментарій міграцій, що дозволяє створювати схему бази даних безпосередньо у вихідному коді проєкту.

Також SQL Server забезпечує транзакційну цілісність за рахунок відповідності принципам ACID [25]. У контексті CRM-системи це є

критичною вимогою, що унеможливилює появу пошкоджених або частково записаних даних.

Крім того, SQL Server надає потужні вбудовані механізми захисту, такі як прозоре шифрування даних на рівні файлів бази та резервних копій, а також маскування динамічних даних, що запобігає несанкціонованому витоку конфіденційної інформації навіть у разі компрометації доступу [26].

2.2.2. База даних MongoDB

Незважаючи на значну обчислювальну потужність реляційних баз даних, запис великих масивів технічної інформації (наприклад, логування кожного HTTP-запиту, фіксація помилок сервера або збереження об'ємних і змінних промптів для штучного інтелекту) у SQL-таблиці може призвести до зниження загальної продуктивності системи через блокування рядків та таблиць [27]. Для вирішення цієї проблеми архітектура SAM CRM передбачає використання NoSQL бази даних MongoDB для зберігання системних журналів подій та технічного контексту.

MongoDB – це провідна документно-орієнтована система управління базами даних з відкритим вихідним кодом. Замість класичних таблиць і рядків, вона зберігає дані у вигляді колекцій та бінарного представлення JSON документів [28].

Переваги використання MongoDB як допоміжної СУБД:

- висока швидкість запису через відсутність необхідності перевірки зовнішніх ключів та підтримки складних транзакцій між документами;
- гнучкість схем, оскільки MongoDB не вимагає попереднього оголошення жорсткої структури документа;
- розділення відповідальності через винесення системних та логічних даних у нереляційне сховище розвантажує основну базу Microsoft SQL Server.

Підсумовуючи, використання комбінованого підходу з Microsoft SQL Server та MongoDB забезпечує баланс між надійністю збереження критично важливої бізнес-інформації та продуктивністю обробки великих обсягів системних записів, що робить розроблену архітектуру стійкою до високих навантажень.

2.3. Вибір засобів реалізації клієнтської частини

Для реалізації клієнтської частини застосунку було обрано сучасні технології, які забезпечують швидку розробку та масштабованість. Основу клієнтської частини становить фреймворк Angular з використанням мови TypeScript, а для побудови інтерфейсу користувача застосовується бібліотека готових компонентів PrimeNG.

2.3.1. Фреймворк Angular та мова TypeScript

Для безпосередньої розробки клієнтської частини було обрано платформу Angular – один з найпопулярніших фреймворків корпоративного рівня, який розробляється та підтримується компанією Google [29]. На відміну від легковагових бібліотек (таких як React або Vue.js), які потребуються підключення багатьох сторонніх пакетів для базової функціональності, Angular є повноцінним фреймворком. Він «з коробки» надає стандартизовану архітектуру, систему маршрутизації, модуль для HTTP-запитів та потужний набір інструментів для роботи з формами [29]. Це дозволяє розробникам працювати в єдиній парадигмі та гарантує легке масштабування проєкту.

Основною розробки на платформі Angular є мова програмування TypeScript – надбудова над JavaScript, яка додає сувору статичну типізацію та підтримку об'єктно-орієнтованого програмування. Використання TypeScript на клієнтській частині в поєднанні з C# на серверній створює єдину надійну інженерну екосистему для проєкту SAM CRM. Завдяки типізації структури даних, які надходять від API, описуються на стороні

клієнта заздалегідь. Це дозволяє виявляти помилки невідповідності типів ще на етапі написання коду в середовищі розробки, що критично важливо при обробці складних структур даних [30].

Архітектура Angular базується на двох ключових концепціях, які ідеально підходять для розробки CRM-системи. Перш за все, це компонентний підхід. Увесь інтерфейс застосунку розбивається на незалежні, ізольовані та повторно використовувані блоку – компоненти. Кожен компонент інкапсулює власну логіку (TypeScript-клас), шаблон відображення (HTML) та стилі (CSS/SCSS) [29]. Це дозволяє ефективно керувати складним інтерфейсом та вносити точкові зміни без ризику полатати інші частини системи. По-друге, Angular глибоко інтегрований з бібліотекою RxJS, яка використовує шаблон «Спостерігач» для обробки асинхронних подій та потоків даних. У системі SAM CRM це підхід активно застосовується для опрацювання відповідей від серверної частини (наприклад, потокове отримання рекомендацій від модуля III), відстеження змін у формах та синхронізації стану застосунку в реальному часі. Оновлення інтерфейсу відбувається лише тоді, коли дані дійсно змінилися, що мінімізує навантаження на браузер користувача [31].

Таким чином, вибір фреймворку Angular та мови TypeScript забезпечує створення стабільної, легко підтримуваної та швидкодіючої клієнтської частини, яка здатна ефективно візуалізувати великі обсяги даних та безшовно взаємодіяти зі складним REST API.

2.3.2. Бібліотека UI-компонентів PrimeNG

Для побудови графічного інтерфейсу клієнтської частини було обрано PrimeNG – одну з найпопулярніших бібліотек готових UI-компонентів, спеціально розроблену для екосистеми Angular компанією PrimeTek [32]. Використання сторонньої бібліотеки компонентів замість написання власних HTML та CSS структур з нуля є стандартом при розробці

корпоративного програмного забезпечення, оскільки це суттєво зменшує час розробки та гарантує сумісність в різних браузерах [33].

Головною перевагою PrimeNG є наявність широкого набору складних компонентів, готових до використання в різних частинах розроблюваної системи. Для роботи з масивами даних (відображення бази клієнтів, контактів, історії задач тощо) застосовуються таблиці вже з вбудованою підтримкою посторінкової розбивки даних, сортування колонок, глобальної фільтрації та експорту даних, що є критично важливим для зручної роботи менеджера з великими списками.

Також наявність вбудованого модуля інтеграції з графіками – Chart.js, дозволяє швидко створювати інформаційні панелі для відображення ключових показників ефективності менеджерів та прогнозування продажів.

Крім того, PrimeNG надає десятки базових стилізованих елементів: календарі для вибору дат задач, модальні вікна для створення нових сутностей, випадаючі списки та повідомлення [32]. Використання цих компонентів гарантує єдиний, професійний і сучасний стиль усього застосунку.

2.4. Висновки до другого розділу

Вибір засобів реалізації зумовлений ключовими архітектурними та функціональними вимогами до розробки сучасної корпоративної CRM-системи.

Для серверної частини було обрано платформу .NET, мову C# та фреймворк ASP.NET Core, що гарантують високу продуктивність, надійний захист корпоративних даних та ефективну побудову RESTful API. Інтелектуальне ядро системи реалізовано на базі хмарного сервісу Gemini API, який забезпечує швидкий аналіз комунікацій та генерацію рекомендацій без інфраструктурного перевантаження сервера.

На рівні збереження даних застосовано комбінований архітектурний підхід: реляційна СУБД Microsoft SQL Server відповідає за транзакційну

цілісність бізнес-даних, тоді як документо-орієнтована MongoDB використовується для швидкого запису неструктурованих даних.

Клієнтська частина реалізована як односторінковий застосунок за допомогою фреймворку Angular і мови TypeScript, а графічний інтерфейс побудовано на базі бібліотеки компонентів PrimeNG.

Загалом, обраний стек технологій формує надійну, масштабовану та безпечну екосистему, яка повністю задовольняє архітектурні вимоги до розробки SAM CRM із вбудованим штучним інтелектом.

3. РОЗРОБЛЕННЯ ВЕБСЕРВІСУ

3.1. Загальний опис структури системи

Структура розробленої системи SAM CRM базується на сучасній розподіленій клієнт-серверній архітектурі, яка концептуально розширена за рахунок інтеграції хмарних сервісів штучного інтелекту. Графічне відображення зв'язків між програмними вузлами системи, базами даних та зовнішніми модулями наведено на рис. 3.1.

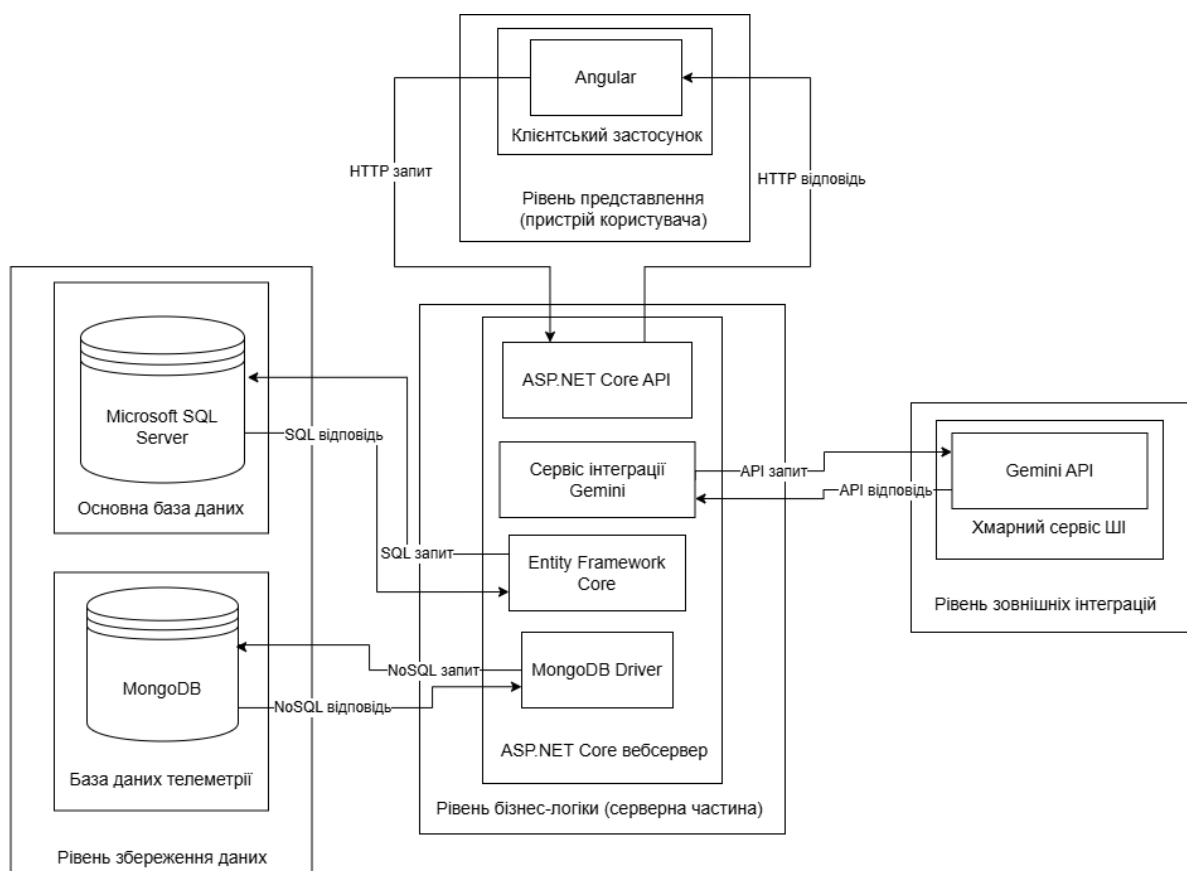


Рис. 3.1. Схема архітектури програмного забезпечення

Розроблена система концептуально поділяється на чотири основні рівні: рівень представлення (клієнтський застосунок), рівень бізнес-логіки (серверна частина), рівень збереження даних (бази даних) та рівень зовнішніх інтеграцій (сервіс ШІ).

Рівень представлення функціонує в середовищі веббраузера користувача і реалізований за концепцією односторінкового застосунку з використанням фреймворку Angular. Головна задача цього рівня – візуалізація даних, забезпечення інтерактивної взаємодії з користувачем та валідація введеної інформації на стороні клієнта. Інтерфейс побудовано з використанням компонентів PrimeNG, що забезпечує швидку роботу з великими таблицями клієнтів. Клієнтська частина не містить бізнес-логіки CRM, а функціонує як «тонкий клієнт», обмінюючись даними з сервером виключно за допомогою HTTP-запитів у форматі JSON.

Рівень бізнес-логіки представлений серверним застосунком на базі фреймворку ASP.NET Core, який виконує роль центрального оркестратора системи. Серверна частина реалізує архітектуру RESTful API та відповідає за маршрутизацію запитів, виконання складних обчислень, автентифікацію користувачів та контроль прав доступу (авторизацію) на основі JWT-токенів. Внутрішня архітектура серверної частини побудована за принципом багат шаровості: вона включає контролери для прийому запитів, сервіси для реалізації бізнес-правил та репозиторії для абстрагування роботи з даними. Це гарантує слабку зв'язність коду та спрощує його подальшу підтримку.

Рівень збереження даних спроектовано за принципом поліглотного збереження, що передбачає використання різних типів СКБД для різних задач. Фундаментом для зберігання структурованих комерційних об'єктів слугує реляційна СКБД Microsoft SQL Server. Вона забезпечує транзакційну цілісність даних завдяки дотриманню принципів ACID. Взаємодія з цією базою здійснюється через ORM-фреймворк Entity Framework Core. Водночас для зберігання неструктурованих системних подій, HTTP-логів застосунку та історії взаємодії з III використовується документно-орієнтована база MongoDB. Такий поділ дозволяє зняти навантаження з основної бази даних.

Рівень зовнішніх інтеграцій представлений модулем Smart Advisor, який взаємодіє з хмарною платформою Gemini API. Важливою архітектурною особливістю є те, що клієнтський застосунок ніколи не звертається до ШІ безпосередньо. Усі запити проходять через сервер ASP.NET Core, який виконує роль проксі-сервера. Це гарантує безпеку (ключі доступу до API залишаються прихованими на сервері) та дозволяє серверу попередньо формувати правильний системний запит, збагачуючи його актуальними даними про клієнта з SQL-бази перед відправкою до мовної моделі.

3.2. Аналіз функціональних вимог до програми

Процес збору та формалізації вимог базувався на аналізі потреб двох ключових категорій (акторів), які безпосередньо взаємодіятимуть із платформою. Першою і найактивнішою групою є менеджери з продажу. Їхня щоденна рутинна вимагає миттєвого доступу до карток угод, інструментів планування задач та модуля генерації інтелектуальних підказок. Другою групою виступають адміністратори системи, яким делеговано повноваження щодо управління обліковими записами співробітників та контролю глобальних налаштувань.

Наведений графічний опис (рис. 3.2) дозволяє чітко розмежувати зони відповідальності акторів та встановити логічні зв'язки між різними функціональними модулями застосунку. Модель візуалізує ініціацію ключових процесів: від базових тригерів автентифікації до складних аналітичних запитів, спрямованих на обробку контексту комерційних угод за допомогою інтелектуального модуля Smart Advisor. Для ролі «Авторизований користувач» пріоритетними є сценарії ведення клієнтської бази, керування об'єктами воронки продажів та отримання стратегічних рекомендацій від штучного інтелекту. У той же час, роль «Адміністратор» зосереджена на технічному керуванні правами доступу користувачів. Побудова такої моделі забезпечує однозначне трактування логіки

застосунку як розробником, так і замовником, що мінімізує ризики архітектурних помилок на етапі реалізації бізнес-логіки.

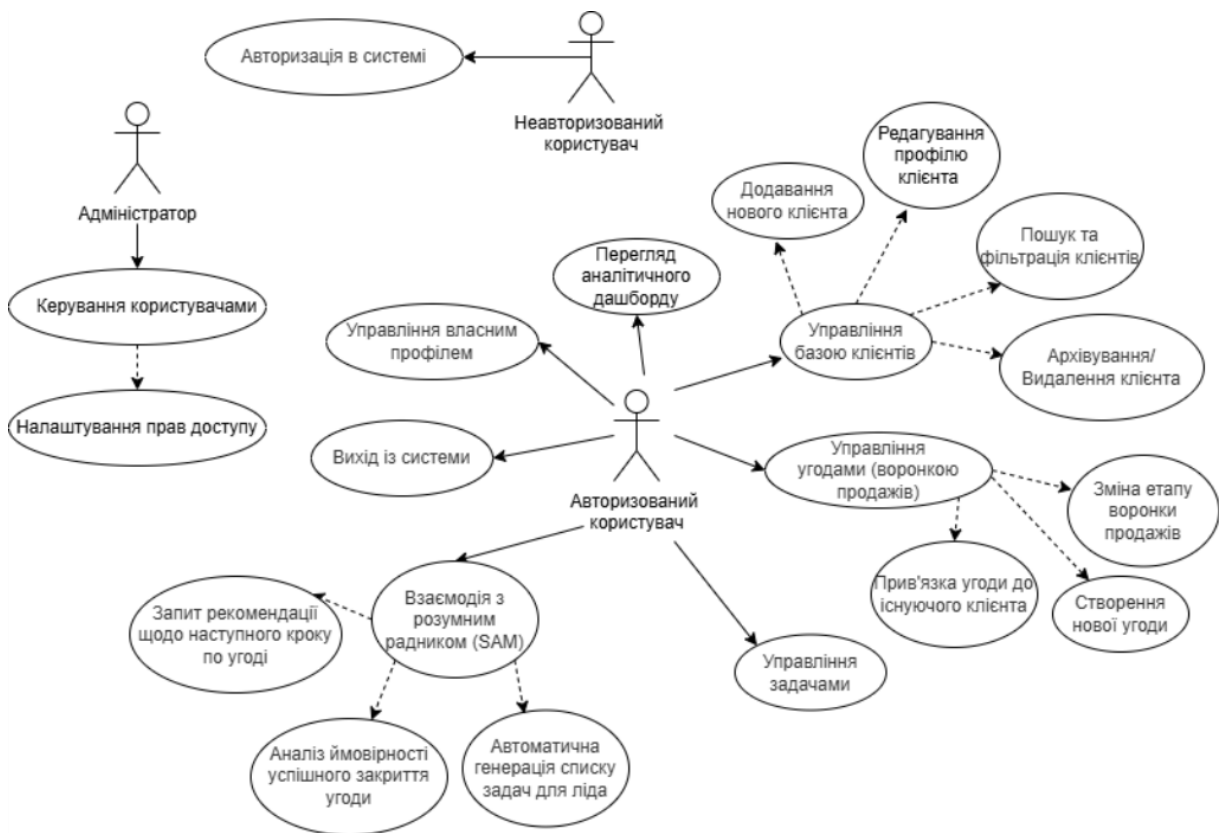


Рис. 3.2. Діаграма прецедентів програмного забезпечення

Для перетворення графічної моделі у формат, придатний для використання розробником при написанні коду, виявлені прецеденти були розбиті на конкретні технічні специфікації. Далі наведено деталізовані функціональні вимоги до системи SAM CRM, які було згруповано за логічними модулями відповідно до виділених на діаграмі прецедентів.

Робота з обліковими записами та налаштування системи – блок, що описує базові можливості доступу до застосунку, керування особистою інформацією співробітників та інструменти адміністрування платформи. Належна реалізація цього модуля гарантує безпечний розподіл прав на основі ролей та надійний захист комерційних даних компанії від несанкціонованого втручання. Вимоги наведено у таблиці 3.1.

Таблиця 3.1

Функціональні вимоги до керування доступом та користувачами

Код вимоги	Назва вимоги	Опис вимоги
F-1.1	Авторизація в системі	Забезпечення входу неавторизованого користувача до системи шляхом перевірки облікових даних та видачі токена доступу.
F-1.2	Вихід із системи	Можливість безпечного завершення сеансу авторизованим користувачем із відкликанням прав доступу поточного токена.
F-1.3	Керування власним профілем	Надання авторизованому користувачу інтерфейсу для перегляду та редагування особистої інформації.
F-1.4	Керування користувачами	Можливість для ролі «Адміністратор» створювати, блокувати або видаляти облікові записи співробітників у системі.
F-1.5	Налаштування прав доступу	Набір інструментів адміністратора для призначення специфічних ролей та рівнів доступу іншим користувачам CRM.

Керування клієнтською базою та аналітика – модуль, що забезпечує агрегацію даних про контрагентів. Вимоги наведено у таблиці 3.2.

Таблиця 3.2

Функціональні вимоги до роботи з користувачами

Код вимоги	Назва вимоги	Опис вимоги
F-2.1	Перегляд аналітичної інформаційної панелі	Відображення зведеної статистики, ключових метрик та графіків ефективності на головному екрані авторизованого користувача.

Продовження табл. 3.2

F-2.2	Додавання нового клієнта	Створення нової картки в базі з фіксацією корпоративної та контактної інформації контрагента.
F-2.3	Редагування профілю клієнта	Можливість оновлення даних, внесення змін до існуючих атрибутів та доповнення профілю клієнта актуальною інформацією.
F-2.4	Пошук та фільтрація клієнтів	Надання інструментів швидкого пошуку по базі за заданими критеріями (назва, статус, відповідальний менеджер тощо).
F-2.5	Архівування/Видалення клієнта	Механізм приховування неактивних клієнтів з основної вибірки (архівування) або їх повного вилучення з бази даних.

Управління комерційними процесами – блок, що регламентує безпосередню операційну діяльність менеджера з продажу: ведення переговорів та виконання робочих завдань. Вимоги наведено у таблиці 3.3.

Таблиця 3.3

Функціональні вимоги до операційного модуля

Код вимоги	Назва вимоги	Опис вимоги
F-3.1	Створення нової угоди	Початок комерційного процесу зі встановленням початкових параметрів (сума, очікувана дата закриття).
F-3.2	Прив'язка угоди до існуючого клієнта	Обов'язкове встановлення реляційного зв'язку між новоствореною угодою та існуючою карткою клієнта у базі.
F-3.3	Зміна етапу воронки продажів	Можливість візуального переміщення угоди між стадіями процесу продажу.

F-3.4	Управління задачами	Створення, редагування та відстеження статусів виконання завдань (дзвінки, зустрічі, листи), пов'язаних з угодами.
-------	---------------------	--

Взаємодія з модулем штучного інтелекту містить специфічний набір вимог, що описує інтеграцію CRM-системи із зовнішнім модулем ШІ. Вимоги наведено у таблиці 3.4.

Таблиця 3.4

Функціональні вимоги до інтелектуального модуля Smart Advisor (SAM)

Код вимоги	Назва вимоги	Опис вимоги
F-4.1	Запит рекомендації щодо наступного кроку	Здатність системи агрегувати контекст угоди та передавати його до ШІ для отримання стратегічної підказки щодо подальших дій менеджера.
F-4.2	Аналіз ймовірності успішного закриття	Виклик функції рейтингу, за якої розумний радник аналізує поточний стан угоди та обчислює прогноз її успішного завершення.
F-4.3	Автоматична генерація списку задач	Можливість делегувати модулю SAM створення релевантного списку наступних завдань для опрацювання клієнта на основі історії взаємодії.

3.3. Особливості реалізації серверної частини

3.3.1. Архітектурна організація та принципи побудови сервера

Розробка серверної частини системи SAM CRM базується на використанні фреймворку ASP.NET Core, що є кросплатформним рішенням з відкритим вихідним кодом для створення сучасних високопродуктивних сервісів. Вибір цієї платформи обумовлений її високою швидкістю обробки запитів, розвиненою екосистемою та підтримкою суворої типізації мови C#,

що критично важливо для забезпечення надійності корпоративних систем зі складною бізнес-логікою.

В основі програмної архітектури серверної частини лежить принцип розділення відповідальності, реалізований через багат шарову структуру застосунку. Основні логічні рівні сервера включають:

- рівень контролерів;
- сервісний рівень;
- рівень доступу до даних.

Рівень контролерів (Web API Layer) виступає вхідною точкою для HTTP-запитів від клієнтської частини. Контролери відповідають за маршрутизацію, базову перевірку прав доступу та форматування відповідей у форматі JSON.

Сервісний рівень (Business Logic Layer) містить основну логіку функціонування CRM. Тут зосереджені класи, що описуються процеси обробки угод, управління задачами та алгоритми взаємодії з модулем штучного інтелекту.

Рівень доступу до даних (Data Access Layer) реалізує взаємодію з базами даних Microsoft SQL Server та MongoDB. Використання ORM-інструменту Entity Framework Core дозволяє уникнути написання прямих SQL-запитів, оперуючи даними на рівні об'єктів мови C#.

Фундаментальним принципом побудови сервера є широке використання механізму впровадження залежностей. ASP.NET Core має вбудований DI-контейнер, що дозволяє реєструвати компоненти системи використовуючи інтерфейси. Це забезпечує слабку зв'язність модулів, що дозволяє, наприклад, змінити реалізацію сервісу логування або провайдера ШІ-моделі без необхідності модифікації контролерів чи бізнес-сервісів.

Ефективність системи при роботі з великою кількістю одночасних користувачів забезпечується завдяки підтримці асинхронності через шаблон `async/await`. Це дозволяє потокам сервера не блокуватися під час виконання

тривалих операцій вводу-виводу, таких як запити до бази даних або звернення до зовнішнього Gemini API.

3.3.2. Реалізація механізмів безпеки та розмежування доступу

Основним механізмом перевірки автентичності користувачів у системі обрано безсесійну архітектуру з використанням стандарту JSON Web Token (JWT). На відміну від традиційної сесійної авторизації, де стан сеансу зберігається в оперативній пам'яті сервера (що створює проблеми при балансуванні навантаження), застосування JWT дозволяє серверу повністю делегувати зберігання стану клієнту. Кожен згенерований токен криптографічно підписується на сервері за допомогою секретного ключа та алгоритму хешування HMAC SHA-256 [34]. Це гарантує цілісність даних, бо будь-яка спроба модифікувати токен (наприклад, змінити ідентифікатор користувача або його роль) на стороні клієнта призведе до недійсності цифрового підпису, і сервер миттєво такий запит відхилить.

Для балансу безпеки та зручності впроваджено двоступеневу ротацію токенів. Токен доступ з коротким терміном існування (Access Token) передається через заголовки HTTP-запитів, що мінімізує шкоду від його можливого перехоплення. Для продовження сесії без повторного входу використовується інший токен – Refresh Token, який захищений особливими HTTP атрибутами. Це блокує доступ до нього через клієнтські скрипти та надійно захищає від XSS-атак.

Зберігання паролів у базі SQL Server базується на алгоритмі BCrypt. Він автоматично робить кожен запис унікальним та додає власні частини до записів, які уповільнюють обчислення, завдяки чому, система є стійкою до зламу методом прямого перебору та використання заздалегідь обчислених таблиць значень.

Керування правами доступу базується на рольовій моделі за допомогою окремого атрибуту Authorize. Це дозволяє обмежувати доступ до конкретних методів API залежно від привілеїв користувача. Додатково

налаштована політика CORS дозволяє серверу приймати запити лише від довіреного домену Angular-застосунку, автоматично блокуючи несанкціоновані звернення.

3.3.3. Забезпечення цілісності даних через DTO та валідацію

У розробленій серверній частині контроль за цілісністю та безпекою даних реалізовано за допомогою багаторівневої фільтрації та трансформації об'єктів.

Центральним архітектурним рішенням в цьому контексті є впровадження шаблону Data Transfer Object (DTO). Його роль полягає у створенні чітких обмежень між внутрішніми моделями бази даних та зовнішнім REST API. Використання DTO дозволяє регламентувати набір полів, які доступні для передачі клієнту або очікуються від нього. Це ефективно запобігає проблемі надмірного заповнення, коли зловмисник може намагатися модифікувати системні параметри об'єкта, що призначені для редагування. Застосування DTO дозволяє ізолювати структуру даних, що передаються по мережі, від схеми зберігання в базі, підвищуючи загальну гнучкість розробки [35].

Окрім ізоляції в структурі, серверна частина реалізує механізм валідації кожного вхідного запиту. Хоча клієнтський застосунок на Angular має власні перевірки форм, серверна перевірка даних є обов'язковим етапом захисту. Для цього використано поєднання вбудованих атрибутів анотацій даних та бібліотеки FluentValidation. Такий підхід дозволяє описувати складні правила: від перевірки формату електронної пошти до валідації сум угод та обов'язковості заповнення контекстних полів, необхідних для коректної роботи інтелектуального модуля Smart Advisor.

3.3.4. Програмна реалізація інтелектуального модуля Smart Advisor

Ключовою інноваційною складовою розробленої системи є інтеграція модуля розумного помічника (Smart Advisor Module, SAM), який

трансформує класичну платформу в систему підтримки прийняття рішень. З архітектурної точки хору цей модуль виконує функцію інтелектуального брокера між внутрішнім контекстом CRM-системи та зовнішньою великою мовною моделлю через Google Gemini API. Для дотримання принципів чистої архітектури та єдиної відповідальності, усю бізнес-логіку взаємодії зі штучним інтелектом інкапсульовано у спеціалізованому сервісному класі.

Для роботи із зовнішнім API Google у системі використовується фабрика `IHttpClientFactory`. На відміну від постійного створення нових об'єктів `HttpClient`, цей підхід дозволяє платформі .NET ефективніше керувати мережевими підключеннями та уникати перевантаження сервера при великій кількості запитів. Усі налаштування, зокрема секретні ключі, не зберігаються безпосередньо в коді.

Процес роботи модуля Smart Advisor поділено на чотири послідовні етапи: збір та підготовка даних, формування й виконання запиту та обробка результату. Послідовність взаємодії компонентів під час генерації рекомендацій наведено на рис. 3.4

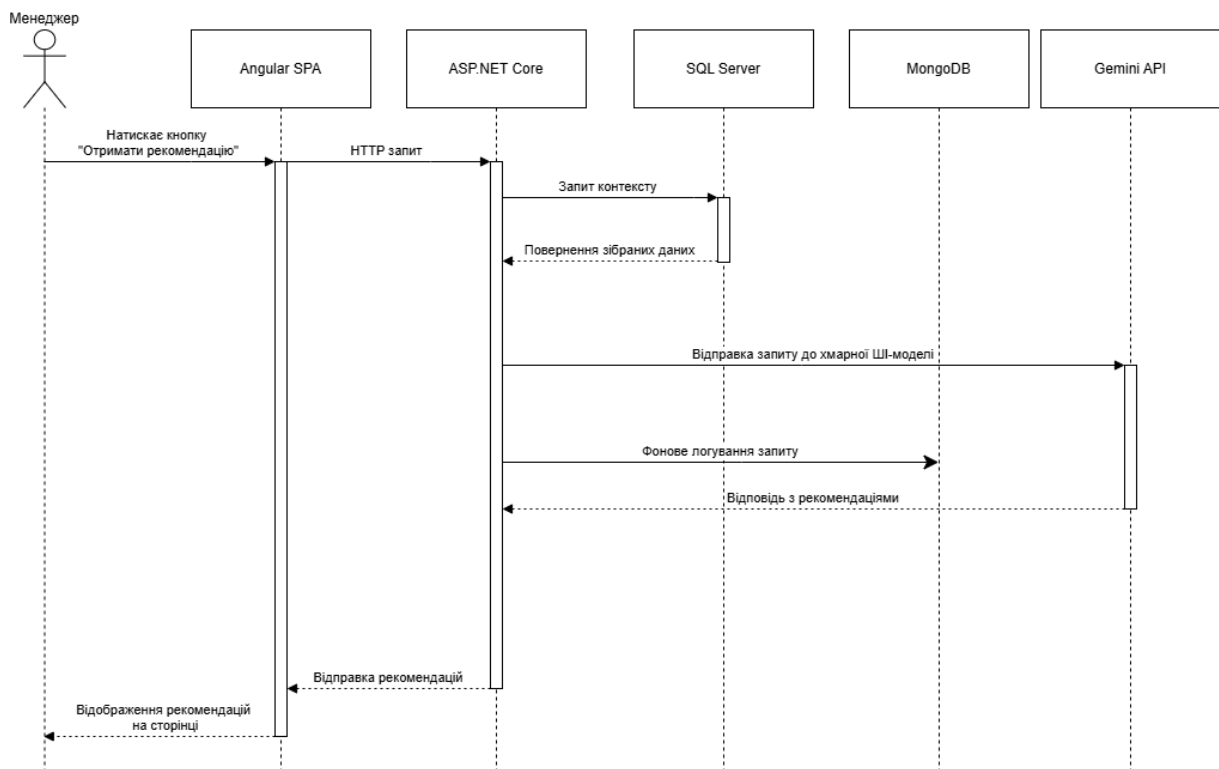


Рис. 3.4. UML-діаграма послідовності роботи інтелектуального модуля

На першому етапі, коли менеджер запитує рекомендацію для конкретної угоди, сервіс звертається до SQL Server. Сервер завантажує необхідну інформацію, наприклад: дані клієнта, історію виконання задач та нотатки. Щоб зекономити ліміт токенів при запиті до Gemini API, ці дані попередньо очищаються від технічної інформації, залишаючи лише суть для аналізу.

На другому етапі зібрані дані перетворюються на текстовий запит до ШІ. Важливо зазначити, що клієнтський застосунок не комунікує з ШІ напряду. Серверна частина самостійно формує системну інструкцію, задаючи моделі роль (наприклад, «аналітик з B2B продажів»), і передає контекст угоди. Також ШІ отримує чітку вказівку повернути результат виключно у вигляді валідного JSON-об'єкта з визначеними полями (ймовірність успіху, ризики, наступні кроки). Це суттєво зменшує ризик отримання неточної або неструктурованої відповіді.

На наступному етапі відбувається виконання запиту. Звернення до Gemini API працює асинхронно з використанням токена CancellationToken, що дозволяє перервати операцію, якщо користувач закрив сторінку. Оскільки зовнішні API можуть іноді давати збої, налаштовано механізм повторних спроб. Якщо сервер Google повертає помилку про тимчасову недоступність, сервер автоматично зачекає і спробує надіслати запит ще раз.

На останньому етапі, після отримання відповіді сервер розбирає згенерований текст і перетворює його на об'єкт C#. На цьому етапі передбачено обробку помилок. У разі відсутності помилок, об'єкт трансформується у Data Transfer Object та відправляється на клієнтську частину для візуалізації.

3.4. Структура бази даних SQL Server

Ефективність функціонування будь-якої системи управління відносинами з клієнтами безпосередньо залежить від раціональної спроектованої моделі збереження інформації. Для забезпечення

транзакційної цілісності, надійності та швидкого доступу до даних використовується реляційна система керування базами даних Microsoft SQL Server.

Для наочної візуалізації спроектованої структури, відображення всіх наявних сутностей, їхніх полів та типів реляційних взаємозв'язків між ними, було побудовано діаграму сутність-зв'язок (ER-діаграму), яку наведено на рис. 3.3.

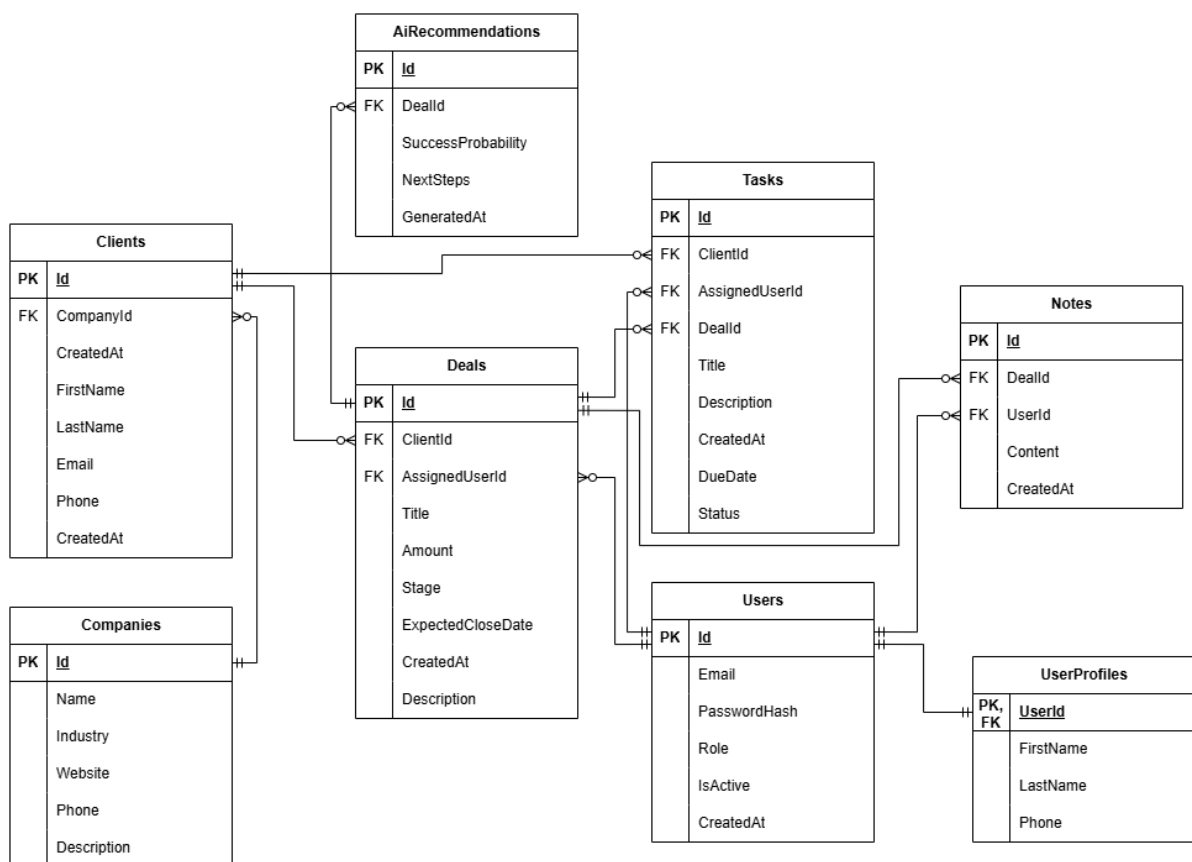


Рис. 3.3. ER-діаграма схеми бази даних

Нижче наведено деталізовані специфікації створених таблиць із зазначенням типів даних, обмежень цілісності та призначення кожного атрибута.

Для забезпечення безпеки застосунку та розмежування прав доступу використовуються таблиці Users та UserProfiles. Таблиця Users (табл. 3.5)

призначена для зберігання базових облікових даних, необхідних для автентифікації, а персональні дані співробітників винесені в окрему таблицю UserProfiles (табл. 3.6), що має зв'язок «один-до-одного» з таблицею користувачів.

Таблиця 3.5

Структура таблиці Users

Назва поля	Тип даних	Обмеження	Опис
Id	INT	PRIMARY KEY, IDENTITY	Унікальний ідентифікатор запису користувача
Email	NVARCHAR	UNIQUE, NOT NULL	Електронна пошта (використовується як логін)
PasswordHash	NVARCHAR	NOT NULL	Захешований пароль
Role	NVARCHAR	NOT NULL	Роль користувача
IsActive	BIT	DEFAULT(1), NOT NULL	Прапорець активності облікового запису
CreatedAt	DATETIME2	DEFAULT(GETUTCDATE())	Дата та час створення облікового запису

Таблиця 3.6

Структура таблиці UserProfiles

Назва поля	Тип даних	Обмеження	Опис
UserId	INT	PRIMARY KEY, FOREIGN KEY	Ідентифікатор користувача
FirstName	NVARCHAR	NOT NULL	Ім'я співробітника

Продовження табл. 3.6

LastName	NVARCHAR	NOT NULL	Прізвище співробітника
Phone	NVARCHAR	NULL	Контактний номер телефону

Інформація про контрагентів зберігається у таблиці Companies (табл. 3.7), яка є базовою сутністю клієнтського домену.

Таблиця 3.7

Структура таблиці Companies

Назва поля	Тип даних	Обмеження	Опис
Id	INT	PRIMARY KEY, IDENTITY	Унікальний ідентифікатор компанії
Name	NVARCHAR	NOT NULL	Назва компанії
Industry	NVARCHAR	NULL	Галузь або сфера діяльності
Website	NVARCHAR	NULL	Вебсайт компанії
Phone	NVARCHAR	NULL	Корпоративний номер телефону
Description	NVARCHAR	NULL	Опис або додаткова інформація

Контактні особи, які представляють ці організації, фіксуються у таблиці Clients (табл. 3.8). Взаємозв'язок реалізовано через зовнішній ключ до компанії.

Таблиця 3.8

Структура таблиці Clients

Назва поля	Тип даних	Обмеження	Опис
Id	INT	PRIMARY KEY, IDENTITY	Унікальний ідентифікатор клієнта
FirstName	NVARCHAR	NOT NULL	Ім'я контактної особи
LastName	NVARCHAR	NOT NULL	Прізвище контактної особи
Email	NVARCHAR	NULL	Робоча електронна пошта
Phone	NVARCHAR	NULL	Контактний номер телефону
CompanyId	INT	FOREIGN KEY, NULL	Посилання на компанію (зв'язок з Companies)
CreatedAt	DATETIME2	DEFAULT(GETUTCDATE())	Дата додавання клієнта до бази

Центральним елементом діяльності системи є комерційні угоди. Структуру таблиці Deals наведено у таблиці 3.9.

Таблиця 3.9

Структура таблиці Deals

Назва поля	Тип даних	Обмеження	Опис
Id	INT	PRIMARY KEY, IDENTITY	Унікальний ідентифікатор угоди
ClientId	INT	FOREIGN KEY, NOT NULL	Посилання до клієнта (зв'язок з Clients)

Продовження табл. 3.9

AssignedUser Id	INT	FOREIGN KEY, NOT NULL	Відповідальний менеджер (зв'язок з Users)
Title	NVARCHAR	NOT NULL	Коротка назва угоди
Amount	DECIMAL	DEFAULT(0), NOT NULL	Сума угоди
Stage	NVARCHAR	NOT NULL	Поточний етап воронки продажів
ExpectedCloseDate	DATE	NULL	Очікувана дата успішного закриття
CreatedAt	DATETIME2	DEFAULT(GETUTCDATE())	Дата та час створення картки угоди
Description	NVARCHAR	NULL	Детальний опис умов угоди

Для планування подальших кроків менеджера використовується таблиця завдань Tasks (табл. 3.10). Завдання мають пряму прив'язку до угод для забезпечення повноти аналітичних даних.

Таблиця 3.10

Структура таблиці Tasks

Назва поля	Тип даних	Обмеження	Опис
Id	INT	PRIMARY KEY, IDENTITY	Унікальний ідентифікатор завдання
DealId	INT	FOREIGN KEY, NULL	Прив'язка до угоди

Продовження табл. 3.10

ClientId	INT	FOREIGN KEY, NULL	Прив'язка до клієнта
AssignedUserId	INT	FOREIGN KEY, NOT NULL	Менеджер, відповідальний за завдання
Title	NVARCHAR	NOT NULL	Тема завдання
Description	NVARCHAR	NULL	Деталі завдання
DueDate	DATETIME2	NULL	Запланована дата та час виконання
Status	NVARCHAR	NOT NULL	Поточний стан виконання
CreatedAt	DATETIME2	DEFAULT(GETUTCDATE())	Дата створення завдання в системі

Історія комунікацій та текстовий контекст зберігається у таблиці нотаток Notes (табл. 3.11).

Таблиця 3.11

Структура таблиці Notes

Назва поля	Тип даних	Обмеження	Опис
Id	INT	PRIMARY KEY, IDENTITY	Унікальний ідентифікатор нотатки
DealId	INT	FOREIGN KEY, NOT NULL	Угода, якої стосується нотатка
UserId	INT	FOREIGN KEY, NOT NULL	Автор нотатки (менеджер)

Продовження табл. 3.11

Content	NVARCHAR	NOT NULL	Текстовий вміст залишеного коментаря
CreatedAt	DATETIME2	DEFAULT(GETUTCDATE())	Час фіксації коментаря

Згенеровані модулем Smart Advisor рекомендації кешуються у спеціальній таблиці AiRecommendations (табл. 3.12), що дозволяє оптимізувати кількість звернень до зовнішнього API Google Gemini.

Таблиця 3.12

Структура таблиці AiRecommendations

Назва поля	Тип даних	Обмеження	Опис
Id	INT	PRIMARY KEY, IDENTITY	Унікальний ідентифікатор рекомендації
DealId	INT	FOREIGN KEY, NOT NULL	Угода, для якої згенеровано аналітику
SuccessProbability	TINYINT	NOT NULL	Відсоток ймовірності успішного закриття
NextSteps	NVARCHAR (MAX)	NOT NULL	Згенеровані рекомендації наступних дій
GeneratedAt	DATETIME2	DEFAULT (GETUTCDATE())	Час створення прогнозу мовною моделлю

3.5. Структура бази даних MongoDB

Для забезпечення високої продуктивності вебсервісу та зняття надлишкового навантаження з основної реляційної бази даних, обробка та зберігання неструктурованих даних із високою частотою запис винесені в документно-орієнтовану СКБД MongoDB. У системі SAM CRM це NoSQL-сховище використовується як база даних для ведення системних журналів (логів) та фіксації детальної взаємодії з інтелектуальним помічником.

Для забезпечення стабільності на рівні серверної частини структура документів контролюється через класи моделей мови С#. У базі даних спроектовано дві ключові колекції.

Колекція SystemLogs призначена для глобального аудиту роботи сервера, відстеження дій користувачів та фіксації критичних помилок. Структуру документів цієї колекції наведено у таблиці 3.13.

Таблиця 3.13

Структура документа колекції SystemLogs

Назва поля	Тип даних	Опис
_id	ObjectId	Унікальний ідентифікатор документа, що містить мітку часу
Timestamp	DateTime	Точна дата та час фіксації події в системі
LogLevel	String	Тип події (Information, Warning, Error)
Message	String	Текстовий опис події
Exception	String / Null	Текст системної помилки

Продовження табл. 3.13

UserId	Int / Null	Ідентифікатор менеджера, який ініціював HTTP-запит
--------	------------	--

Колекція AiRequestLogs використовується для моніторингу роботи штучного інтелекту, аналізу якості prompt-інженерії та контролю використання лімітів токенів Google Gemini API. Структуру документів цієї колекції наведено у таблиці 3.14.

Таблиця 3.14

Структура документа колекції AiRequestLogs

Назва поля	Тип даних	Опис
_id	ObjectId	Унікальний ідентифікатор запису журналу
Timestamp	DateTime	Дата та час фіксації події в системі
DealId	Int	Ідентифікатор угоди, на основі якої формувався контекст
PromptContext	String	Повний текст згенерованого системного промпту та даних угоди
TokenCount	Int	Сумарна кількість токенів, витрачених на операцію
LatencyMs	Int	Тривалість очікування відповіді
Status	String	Фінальний статус виконання операції

Винесення цієї інформації в окреме сховище дозволяє проводити глибокий аналіз роботи системи та дій користувачів, повністю усуваючи ризики блокування таблиць в основній базі даних SQL Server під час активної роботи менеджерів.

3.6. Особливості реалізації клієнтської частини

Клієнтська частина системи SAM CRM реалізована за архітектурним шаблоном Single Page Application з використанням фреймворку Angular та строго типізованої мови програмування TypeScript. Такий підхід дозволяє перенести логіку маршрутизації та рендерингу інтерфейсу на сторону браузера користувача. Після первинного завантаження застосунку, обмін даними з сервером ASP.NET Core відбувається виключно у фоновому режимі у форматі JSON. Це усуває необхідність повного перезавантаження вебсторінок під час роботи, що забезпечує високу швидкість відгуку системи та користувацький досвід, наближений до класичних настільних застосунків.

В основі побудови клієнтського застосунку лежить компонентний підхід. Увесь графічний інтерфейс CRM-системи розділено на ієрархію незалежних модулів. Наприклад, сторінка воронки продажів складається з батьківського компонента дошки та множини дочірніх компонентів – карток окремих угод. Це дозволяє повторно використовувати код, полегшує тестування та мінімізує ризики виникнення конфліктів при внесенні змін до інтерфейсу.

Для управління асинхронними подіями застосовується бібліотека реактивного програмування RxJS. Використання шаблону Observable дозволяє ефективно обробляти потоки даних у реальному часі без блокування основного потоку браузера.

Візуальне оформлення та структура інтерфейсу побудовані з використанням бібліотеки UI-компонентів PrimeNG. Даний вибір дозволив

стандартизувати зовнішній вигляд застосунку та значно прискорити процес розробки.

Для безпечної взаємодії з сервером у клієнтському застосунку налаштовано механізм HTTP-перехоплювачів. Цей сервіс виконує рутинну роботу та автоматично додає JWT-токен доступу до заголовків кожного вихідного запиту.

Окрему увагу в інтерфейсі приділено візуалізації роботи інтелектуального модуля Smart Advisor. Завдяки реактивному підходу, рекомендації від штучного інтелекту щодо вірогідності закриття та наступних кроків по угоді підвантажуються та відображаються у відповідних компонентах карток асинхронно. Це дозволяє менеджеру миттєво отримувати аналітичні підказки без переривання основного робочого процесу.

3.7. Висновки до третього розділу

У третьому розділі дипломного проєкту було виконано повний цикл проєктування та програмної реалізації вебсервісу системи управління відносинами з клієнтами із вбудованим інтелектуальним модулем. На початковому етапі було спроектовано гнучку розподілену клієнт-серверну архітектуру та проведено детальний аналіз функціональних вимог. За допомогою методології UML побудовано діаграму прецедентів, що дозволило розмежувати зони відповідальності користувачів і закласти фундамент для логічного поділу системи на рівні представлення, бізнес-логіки та збереження даних.

Програмну реалізацію серверної частини виконано на базі фреймворку ASP.NET Core з використанням багатоваріантної структури коду, механізмів впровадження залежностей та асинхронної моделі обробки запитів. Значну увагу приділено безпеці: впроваджено дворівневу безсесійну авторизацію на основі JWT-токенів та надійне хешування паролів за алгоритмом BCrypt. Цілісність даних на етапі їх надходження до

системи забезпечується завдяки ізоляції моделей через шаблон DTO та автоматичній валідації.

Ключовим інноваційним рішенням стала розробка логіки інтелектуального модуля Smart Advisor, який взаємодіє з хмарним сервісом Google Gemini API. Модуль автоматично збирає контекст угоди та генерує структуровані рекомендації щодо подальших кроків менеджера. Для ефективного обслуговування цих процесів реалізовано комбінований підхід до баз даних. Основні бізнес-дані зберігаються у реляційній СКБД Microsoft SQL Server, тоді як запис системних даних та аудит роботи штучного інтелекту делеговано документно-орієнтованій базі MongoDB, що суттєво оптимізує навантаження на систему.

Клієнтська частина платформи розроблена за шаблоном Single Page Application з використанням фреймворку Angular та бібліотеки компонентів PrimeNG, що забезпечило швидку візуалізацію інформаційних панелей та високу інтерактивність інтерфейсу. Таким чином, розроблена архітектура та її практична реалізація повністю відповідають поставленим інженерним вимогам, утворюючи надійну та безпечну систему для ефективного управління комерційними процесами.

4. АНАЛІЗ РОЗРОБЛЕННОГО СЕРВІСУ

4.1. Аналіз реалізованого програмного забезпечення

Розроблене програмне забезпечення SAM CRM є комплексною платформою, що призначена для автоматизації та полегшення щоденної роботи менеджерів з продажів. Основна мета системи – об'єднати всі ключові етапи взаємодії з клієнтами в одному зручному цифровому середовищі, надавши фахівцям не лише інструменти для обліку, а й інтелектуальну підтримку для прийняття рішень.

Робота із системою починається з авторизації, після якої користувач потрапляє на головну інформаційну панель. Ця сторінка виконує роль центрального вузла, де зібрана зведена інформація: кількість нових клієнтів, загальний дохід, статистика успішних угод та список найближчих завдань. Це дозволяє менеджеру швидко оцінювати поточний стан справ та доцільно розставляти пріоритети.

Для організації клієнтської бази застосунок надає інструменти створення та редагування профілів компаній і конкретних контактних осіб. Користувачі можуть легко переглядати списки своїх клієнтів (рис. 4.1), застосовувати локальний пошук та фільтрацію для швидкого знаходження потрібної інформації.

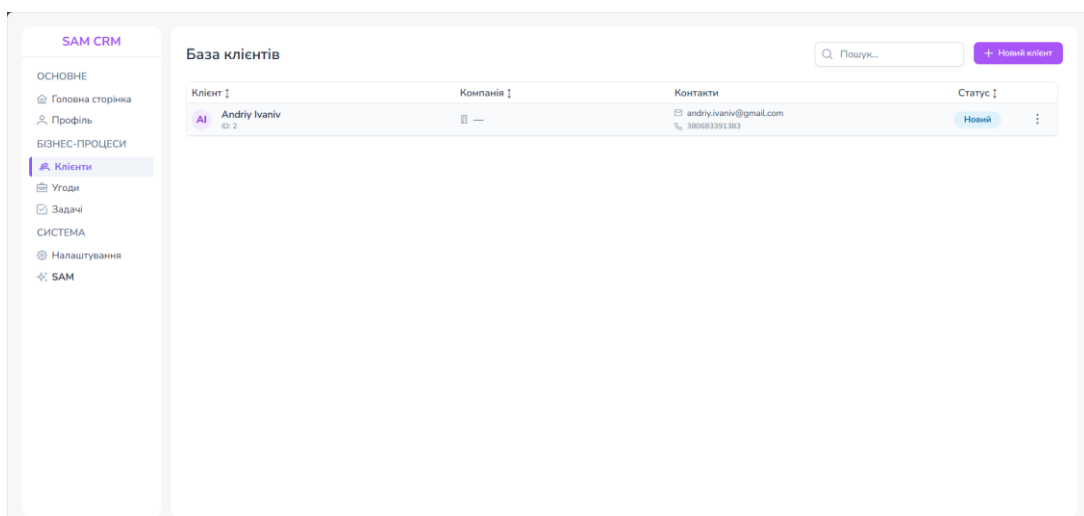


Рис. 4.1. Інтерфейс управління клієнтами

Ключовим елементом системи є модуль комерційних угод, що зображений на рис. 4.2. Він реалізований у вигляді таблиці, де кожна угода має свою назву, бюджет, поточний етап воронки продажів та очікувану дату закриття. Також зверху сторінки розташовуються три інтерактивні картки, що динамічно відображають статистику поточного списку угод.

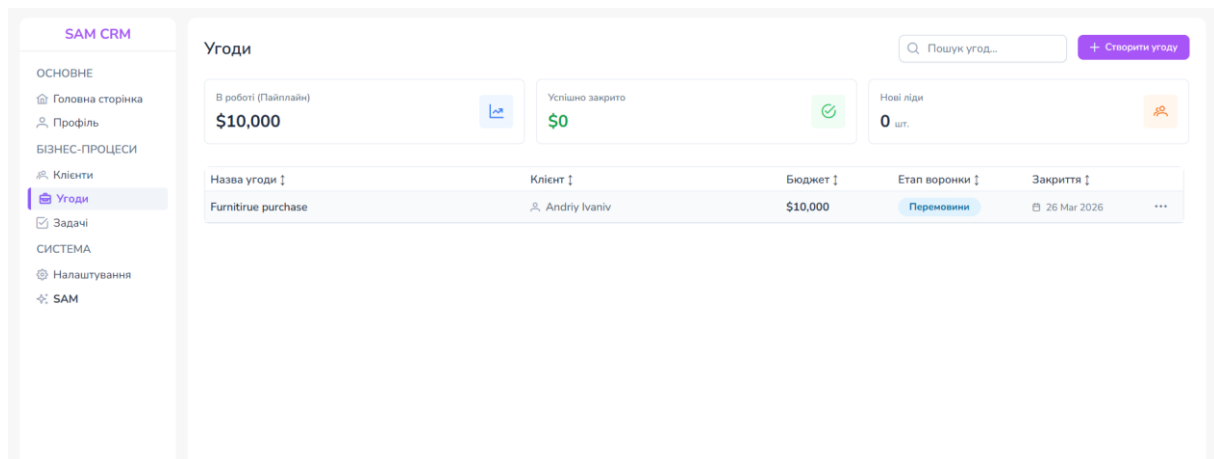


Рис. 4.2. Інтерфейс управління угодами

Для ефективного планування часу передбачено модуль керування задачами. Менеджери можуть створювати нагадування про дзвінки, зустрічі чи підготовку документів прив'язуючи їх до конкретних угод або клієнтів.

Важливою особливістю, яка відрізняє розроблений застосунок від багатьох існуючих рішень, є наявність вбудованого інтелектуального помічника. Завдяки спеціальній панелі в інтерфейсі, користувач може ініціювати аналіз обраної угоди. Система автоматично збирає контекст (попередні задачі, залишені нотатки) і за допомогою штучного інтелекту генерує рекомендації щодо наступних кроків та оцінює ймовірність успіху. Це допомагає менеджеру приймати більш обґрунтовані рішення під час ведення перемовин.

З технічного погляду, серверна частина застосунку побудована на базі сучасного фреймворку ASP.NET Core із застосуванням багаторівневої архітектури. Це забезпечує високу швидкість обробки запитів та легкість подальшого розширення системи. Для роботи з даними використано

комбінований підхід: основна бізнес-інформація зберігається у надійній реляційній базі даних Microsoft SQL Server, що гарантує їхню цілісність. Водночас системні журнали та історія запитів до штучного інтелекту зберігаються в окремій нереляційній базі MongoDB.

У підсумку, розроблене програмне забезпечення виконує поставлені завдання. Він не лише надає базові функції класичної CRM-системи, але й пропонує інноваційний проактивний підхід завдяки інтеграції мовних моделей штучного інтелекту.

4.2. Тестування програмного забезпечення

Для підтвердження надійності та відповідності розробленого програмного забезпечення заявленим технічним вимогам було проведене комплексне тестування його компонентів. Оскільки система базується на розподіленій клієнт-серверній архітектурі, процес перевірки був розділений на декілька напрямків: тестування серверного API, перевірка клієнтського інтерфейсу та оцінка інтеграції з модулем штучного інтелекту.

На рівні серверної частини першочергово здійснювалася перевірка механізмів безпеки та обробки HTTP-запитів. Важливим етапом стало тестування JWT-токенів: система успішно ідентифікувала та блокувала несанкціоновані запити до кінцевих точок. Крім того, проводилася перевірка валідації вхідних даних через об'єкти передачі даних (DTO). Було підтверджено, що спроби створення угод із некоректними числовими значеннями бюджету або відсутніми обов'язковими значеннями автоматично перехоплюються на рівні контролерів із поверненням відповідних статус-кодів помилок клієнту.

Клієнтська частина проходила перевірку на коректність рендерингу UI-компонентів та стабільність керування станом застосунку. Перевірялися сценарії маршрутизації між сторінками застосунку. Окремо приділялася увага обробці асинхронних подій: інтерфейс залишається розблокованим під час очікування відповідей від сервера. Механізми сортування,

посторінкової розбивки та фільтрації в таблицях клієнтів, компаній і угод спрацьовували без візуальних затримок.

Оцінка працездатності інтелектуального модуля Smart Advisor включала перевірку процесів збирання контексту та обробки відповідей від зовнішнього сервісу Google Gemini API. Було підтверджено, що серверна частина успішно збирає текстові нотатки та історію виконаних завдань по конкретній угоді, формує структурований запит та відправляє його до хмарної мовної моделі.

На рівні збереження інформації було перевірено транзакційну цілісність у реляційній базі SQL Server. Зокрема, перевірялося дотримання обмежень зовнішніх ключів та каскадне видалення пов'язаних завдань у разі видалення угоди. Паралельно був підтверджений фоновий запис системних журналів до сховища MongoDB.

Загалом проведене тестування довело, що розроблений програмний продукт функціонує стабільно, алгоритми обробки даних працюють без затримок, а обрані рішення розробки забезпечили надійне користування CRM-системою.

4.3. Рекомендації щодо подальшого вдосконалення системи

Хоча розроблена платформа SAM CRM успішно виконує поставлені завдання та задовольняє вимоги до систем управління відносинами з клієнтами швидкий розвиток технологій машинного навчання та бізнес-процесів відкриває широкі перспективи для її вдосконалення та модернізації.

Пріоритетним напрямком подальшого розвитку є масштабування можливостей інтелектуального модуля. Наразі ШІ працює переважно з текстовим контекстом нотаток та завдань, проте в майбутньому доцільно реалізувати алгоритми глибокого аналізу тональності комунікацій. Це дасть змогу системі автоматично аналізувати «емоційний фон» співпраці з клієнтом і завчасно давати сигнали менеджеру про ризики зриву угоди.

Також вагомим розширенням стане здатність модуля автоматично генерувати персоналізовані шаблони електронних листів або комерційних пропозицій на основі поточного етапу воронки продажів.

Іншим важливим напрямком вдосконалення є розбудова екосистеми зовнішніх інтеграцій. Ефективність роботи штучного інтелекту безпосередньо залежить від обсягу та якості вхідних даних. Тому перспективним стане підключення CRM-системи до сервісів IP-телефонії та популярних корпоративних месенджерів. Впровадження механізмів розпізнавання голосу дозволить автоматично транскрибувати телефонні розмови з клієнтами та додавати їх у вигляді тексту до карток угод. Це повністю звільнить менеджерів від рутинного ручного введення даних та забезпечить мовну модель Gemini контекстом для аналітики.

З погляду розвитку користувацького інтерфейсу та аналітики, доцільно розширити існуючу інформаційну панель повноцінною підсистемою Business Intelligence (BI). Це надасть керівникам відділів продажу інструменти для побудови власно налаштованих звітів, аналізу лідів та експорту статистики у форматах PDF та Excel.

Поступова реалізація запропонованих рішень вдосконалення дозволить поступово перетворити SAM CRM із зручного інструменту обліку на глибоко інтегрованого бізнес-асистента, який безпосередньо впливатиме на зростання фінансових показників підприємства.

4.4. Висновки до четвертого розділу

У четвертому розділі було здійснено комплексний аналіз результатів практичної реалізації програмного забезпечення управління відносинами з клієнтами. На основі огляду доведено, що спроектований користувацький інтерфейс відповідає стандартам корпоративних вебзастосунків. Завдяки компонентній архітектурі вдалося створити інтуїтивно зрозуміле робоче середовище.

Результати тестування підтвердили, що створена архітектура працює стабільно. Перевірка серверних запитів показала надійність захисту даних за допомогою JWT-токенів та коректність роботи бази даних. Також тестування довело, що система збирає необхідний контекст для розумного помічника без помилок і успішно веде фоновий запис усіх системних журналів у базу MongoDB.

Окрім цього, за підсумками аналізу було визначено кілька напрямків для подальшого розвитку платформи. Серед них – розширення функцій штучного інтелекту для генерацій текстів, інтеграція системи з IP-телефонією та популярними месенджерами для автоматичного збору інформації. У підсумку розроблена CRM-система є повністю працездатною, успішно вирішує всі поставлені завдання та має хороший потенціал для подальшого практичного використання в бізнесі.

ВИСНОВКИ

У ході виконання дипломного проєкту було досягнуто поставлену мету – розроблено, описано та проаналізовано сучасне програмне забезпечення із вбудованим інтелектуальним помічником для управління відносинами з клієнтами. Створена система успішно вирішує завдання автоматизації обліку комерційних процесів, структурування історії взаємодії з клієнтами та надання підтримки менеджерам з продажів для прийняття ефективних рішень.

На початкових етапах роботи було проведено детальний аналіз предметної області та існуючих на ринку CRM-систем. Виявлені недоліки аналогів, такі як висока рутинність дій користувача та відсутність автоматизованого аналізу накопиченої текстової інформації, дозволили сформуваати чіткі функціональні вимоги до нової платформи. На основі цих вимог було обґрунтовано вибір сучасного та надійного набору технологій, що базується на фреймворку ASP.NET Core для серверної частини та Angular з бібліотекою компонентів PrimeNG для клієнтського інтерфейсу.

Особливу увагу приділено архітектурі збереження даних, де було успішно застосовано комбінований підхід. Основна бізнес-інформація розміщена у реляційній базі даних Microsoft SQL Server, що гарантує суворий контроль цілісності даних та зв'язків між ними. Водночас для зняття навантаження з основного сервера, збереження системних журналів було винесено в документно-орієнтовану СКБД MongoDB.

Ключовою інноваційною складовою проєкту стала розробка концепції та логіки функціонування інтелектуального модуля Smart Advisor. Впроваджені алгоритми серверної prompt-інженерії дозволяють автоматично збирати контекст угод та надсилати його до хмарного сервісу Google Gemini API. Отримані результати асинхронно відображаються на клієнтській частині у вигляді списку дій та індикатора прогнозованого

успіху угоди, що допомагає мінімізувати помилки людського фактора під час ведення комерційних переговорів.

Проведено тестування готового програмного забезпечення підтвердило його стабільність, безпеку та відповідність закладеним технічним вимогам.

Також були окреслені чіткі напрямки для подальшої модернізації платформи, серед яких – розширення аналітичних функцій ШІ інструментами аналізу тональності тексту та інтеграція з IP-телефонією й корпоративними месенджерами.

Таким чином, розроблена CRM-система є завершеним, працездатним та технологічним інженерним продуктом. Вона демонструє ефективність поєднання класичних методів управління продажами з передовими можливостями генеративного штучного інтелекту.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. What is Salesforce? [Електронний ресурс] // Salesforce. – Режим доступу: <https://www.salesforce.com/products/what-is-salesforce/> (дата звернення: 19.05.2026).
2. Create a Dream Dashboard for Your Sales Team [Електронний ресурс] // Salesforce Admins. – Режим доступу: <https://admin.salesforce.com/blog/2023/create-a-dream-dashboard-for-your-sales-team> (дата звернення: 19.05.2026).
3. Create a custom lightning record page in Salesforce [Електронний ресурс] // Cloud First Labs. – Режим доступу: <https://cloudfirstlabs.com/create-a-custom-lightning-record-page-in-salesforce/> (дата звернення: 19.05.2026).
4. Pipedrive Products [Електронний ресурс] // Pipedrive. – Режим доступу: <https://www.pipedrive.com/en/products> (дата звернення: 19.05.2026).
5. About Pipedrive [Електронний ресурс] // Pipedrive. – Режим доступу: <https://pipedrive.frigat.net/about> (дата звернення: 19.05.2026).
6. Pipedrive API v1 Reference [Електронний ресурс] // Pipedrive Developers. – Режим доступу: <https://developers.pipedrive.com/docs/api/v1> (дата звернення: 19.05.2026).
7. CRM-система для Gmail [Електронний ресурс] // NetHunt. – Режим доступу: <https://nethunt.com/ua/gmail-crm> (дата звернення: 19.05.2026).
8. NetHunt Customization [Електронний ресурс] // NetHunt Help Center. – Режим доступу: <https://help.nethunt.com/en/category/customization-152e9p/> (дата звернення: 19.05.2026).
9. Воронка продажів [Електронний ресурс] // NetHunt. – Режим доступу: <https://nethunt.com/ua/features/sales-pipeline> (дата звернення: 19.05.2026).
10. Автоматизація робочих процесів [Електронний ресурс] // NetHunt. – Режим доступу: <https://nethunt.com/ua/features/workflow-automation> (дата звернення: 19.05.2026).

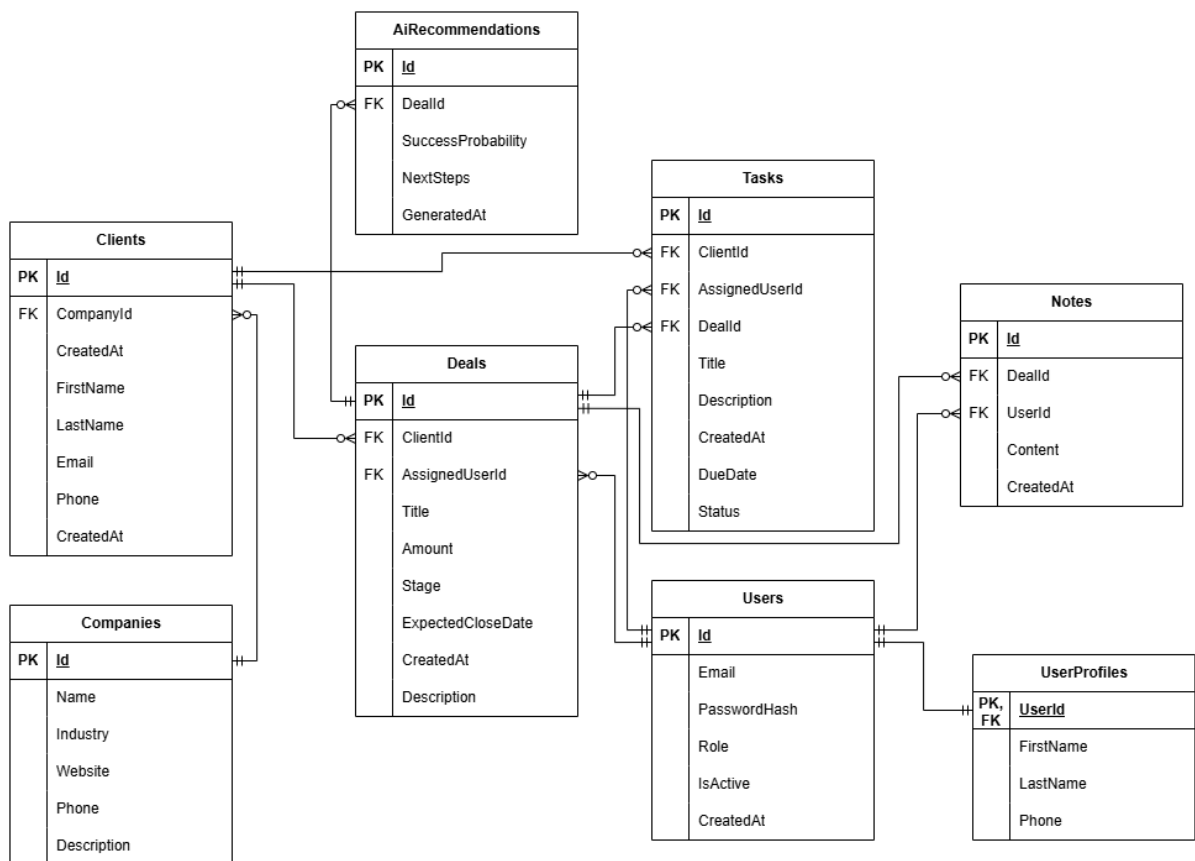
11. Звіти в CRM [Електронний ресурс] // NetHunt. – Режим доступу: <https://nethunt.com/ua/features/reports> (дата звернення: 19.05.2026).
12. Інтеграція з Gmail [Електронний ресурс] // NetHunt. – Режим доступу: <https://nethunt.ua/integrations/gmail> (дата звернення: 19.05.2026).
13. Introduction to .NET [Електронний ресурс] // Microsoft Learn. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/core/introduction> (дата звернення: 19.05.2026).
14. .NET Framework Architecture [Електронний ресурс] // GeeksforGeeks. – Режим доступу: <https://www.geeksforgeeks.org/dot-net-framework-architecture/> (дата звернення: 19.05.2026).
15. C# Documentation [Електронний ресурс] // Microsoft Learn. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/csharp/> (дата звернення: 19.05.2026).
16. Albahari J. C# 12 in a Nutshell: The Definitive Reference. – O'Reilly Media, 2023. – 1056 p.
17. NuGet Gallery [Електронний ресурс] // NuGet. – Режим доступу: <https://www.nuget.org/> (дата звернення: 19.05.2026).
18. Freeman A. Pro ASP.NET Core 7: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. – Apress, 2023. – 1200 p.
19. Dependency injection in ASP.NET Core [Електронний ресурс] // Microsoft Learn. – Режим доступу: <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection> (дата звернення: 19.05.2026).
20. JWT Authentication and Authorization In .NET 6.0 With Identity Framework [Електронний ресурс] // C# Corner. – Режим доступу: <https://www.c-sharpcorner.com/article/jwt-authentication-and-authorization-in-net-6-0-with-identity-framework/> (дата звернення: 19.05.2026).
21. Overview of ASP.NET Core [Електронний ресурс] // Microsoft Learn. – Режим доступу: <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core> (дата звернення: 19.05.2026).

22. Gemini API Documentation [Электронный ресурс] // Google AI for Developers. – Режим доступа: <https://ai.google.dev/docs> (дата звернения: 19.05.2026).
23. Use HttpClientFactory to implement resilient HTTP requests [Электронный ресурс] // Microsoft Learn. – Режим доступа: <https://learn.microsoft.com/en-us/dotnet/architecture/microservices/implement-resilient-applications/use-httpclientfactory-to-implement-resilient-http-requests> (дата звернения: 19.05.2026).
24. Entity Framework Core [Электронный ресурс] // Microsoft Learn. – Режим доступа: <https://learn.microsoft.com/en-us/ef/core/> (дата звернения: 19.05.2026).
25. Coronel C., Morris S. Database Systems: Design, Implementation, & Management. – Cengage Learning, 2018. – 816 p.
26. SQL Server Technical Documentation [Электронный ресурс] // Microsoft Learn. – Режим доступа: <https://learn.microsoft.com/en-us/sql/sql-server/> (дата звернения: 19.05.2026).
27. Sadalage P. J., Fowler M. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. – Addison-Wesley Professional, 2012. – 192 p.
28. MongoDB Manual [Электронный ресурс] // MongoDB. – Режим доступа: <https://www.mongodb.com/docs/manual/> (дата звернения: 19.05.2026).
29. Angular Overview [Электронный ресурс] // Angular. – Режим доступа: <https://angular.dev/overview> (дата звернения: 19.05.2026).
30. Freeman A. Essential Angular for ASP.NET Core MVC: Integrate Angular with ASP.NET Core. – Apress, 2021. – 412 p.
31. RxJS Overview [Электронный ресурс] // RxJS. – Режим доступа: <https://rxjs.dev/guide/overview> (дата звернения: 19.05.2026).
32. PrimeNG [Электронный ресурс] // PrimeTek. – Режим доступа: <https://primeng.org/> (дата звернения: 19.05.2026).

33. Davis A. Enterprise Angular: Best Practices for Building Large-Scale Applications. – Packt Publishing, 2022. – 380 p.
34. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519, IETF, 2015. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення: 19.05.2026).
35. Fowler M. Patterns of Enterprise Application Architecture. – Addison-Wesley Professional, 2002. – 560 p.

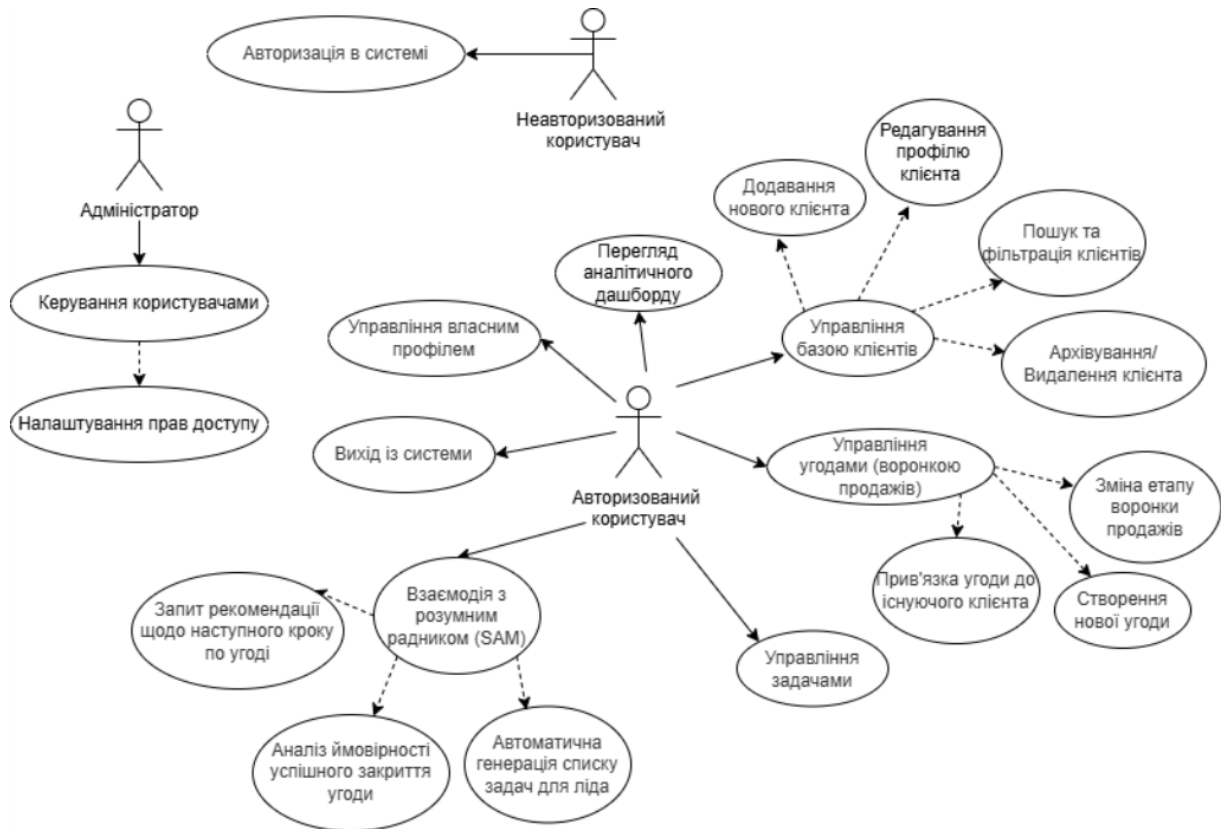
ДОДАТКИ

Додаток 1
Копії графічних матеріалів



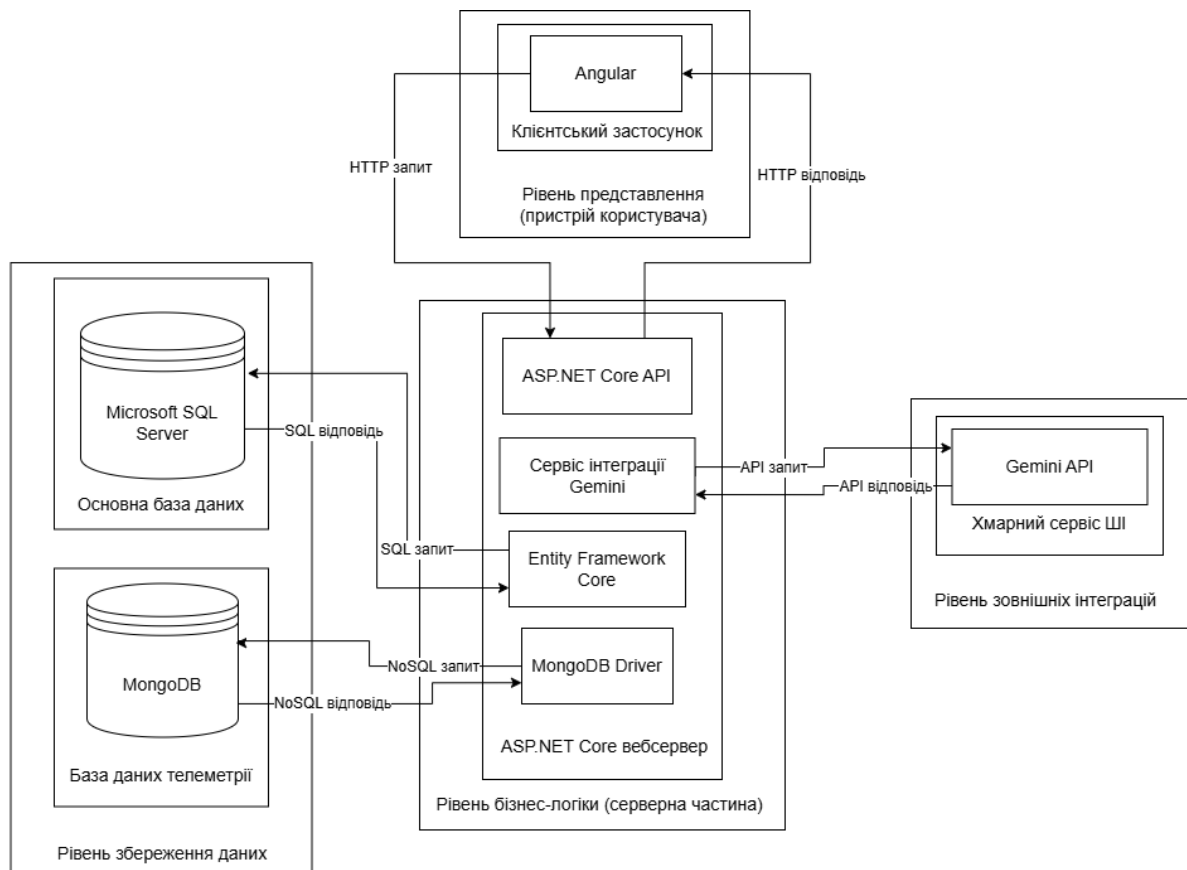
ДП.045440-06-99

Програмне забезпечення із вбудованим помічником на основі штучного інтелекту для управління відносинами з клієнтами. Схема бази даних. ER-діаграма

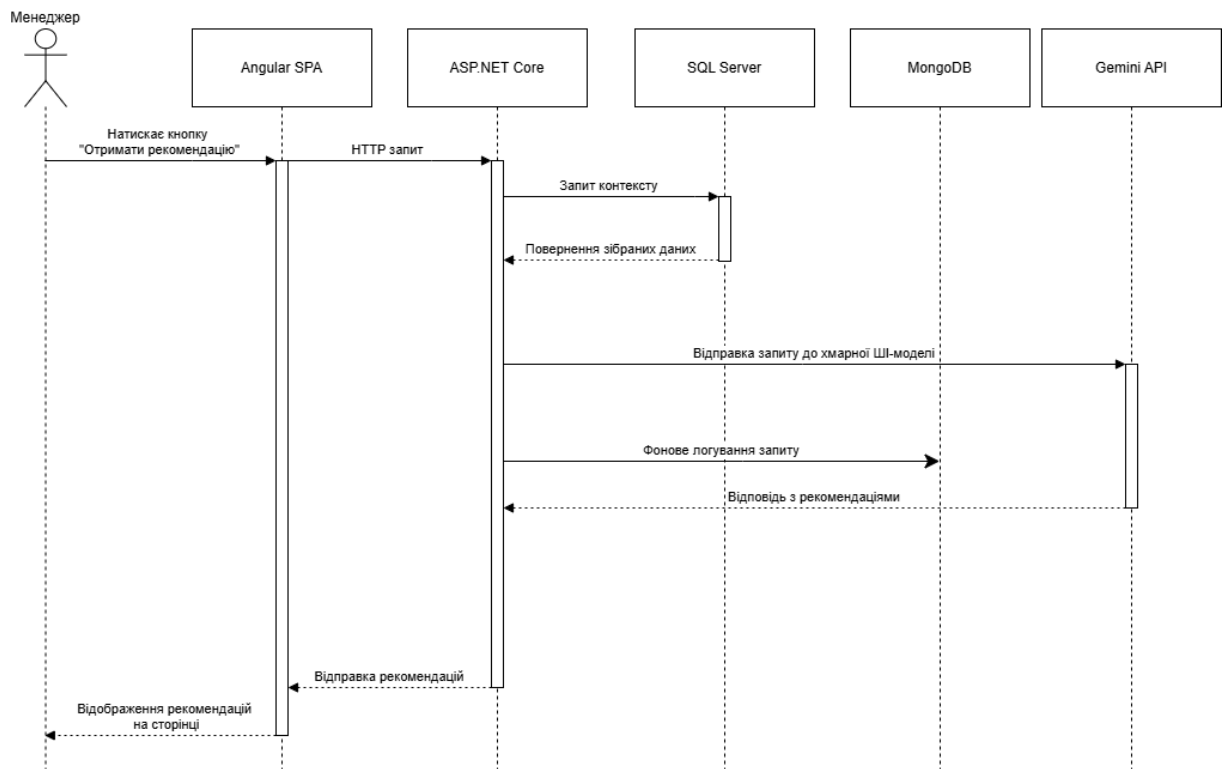


ДП.045440-07-99

Програмне забезпечення із вбудованим помічником на основі штучного інтелекту для управління відносинами з клієнтами. Функціональність програмного забезпечення. UML-діаграма прецедентів



Узагальнена схема архітектури
програмного забезпечення
Боньковський І.В., група КП-21



Діаграма послідовності сценарію
«Запит рекомендації щодо
наступного кроку»
Боньковський І.В., група КП-21

Додаток 2
Лістинг програми

Контролер керування комерційними угодами

```
public class DealsController : BaseApiController
{
    public DealsController(IMediator mediator) : base(mediator)
    {

        [HttpGet]
        public async Task<ActionResult<IEnumerable<DealDTO>>> GetAll()
        {
            return HandleResult(await _mediator.Send(new GetAllDealsQuery()));
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<DealDTO?>> Get(int id)
        {
            return HandleResult(await _mediator.Send(new
GetDealByIdQuery(id)));
        }

        [HttpPost]
        public async Task<ActionResult<DealDTO>> Create(CreateUpdateDealDTO
dto)
        {
            var result = await _mediator.Send(new CreateDealCommand(dto));

            if (result.IsSuccess)
                return CreatedAtAction(nameof(Get), new { id = result.Value!.Id
}, result.Value);

            return HandleResult(result);
        }

        [HttpPut("{id}")]
        public async Task<IActionResult> Update(int id, CreateUpdateDealDTO
dto)
        {
            return HandleResult(await _mediator.Send(new UpdateDealCommand(id,
dto)));
        }

        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(int id)
        {
            return HandleResult(await _mediator.Send(new
DeleteDealCommand(id)));
        }
    }
}
```

Конфігурація контексту бази даних

```
public class CRMDbContext : DbContext
{
    public CRMDbContext(DbContextOptions<CRMDbContext> options) :
base(options)
    {
    }

    public DbSet<Company> Companies => Set<Company>();
    public DbSet<Client> Clients => Set<Client>();
    public DbSet<User> Users => Set<User>();
    public DbSet<UserProfile> UserProfiles => Set<UserProfile>();
    public DbSet<Note> Notes => Set<Note>();
    public DbSet<AiRecommendation> AiRecommendations =>
Set<AiRecommendation>();
    public DbSet<Task> Tasks => Set<Task>();
    public DbSet<Deal> Deals => Set<Deal>();

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {

modelBuilder.ApplyConfigurationsFromAssembly(typeof(CRMDbContext).Assembly)
;

        base.OnModelCreating(modelBuilder);
    }
}

public class UserProfileConfiguration :
IEntityTypeConfiguration<UserProfile>
{
    public void Configure(EntityTypeBuilder<UserProfile> builder)
    {
        builder.ToTable("UserProfiles");

        builder.HasKey(p => p.UserId);
    }
}

public class UserConfiguration : IEntityTypeConfiguration<User>
{
    public void Configure(EntityTypeBuilder<User> builder)
    {
        builder.ToTable("Users");

        builder.HasKey(u => u.Id);
        builder.HasOne(u => u.Profile)
            .WithOne(p => p.User)
            .HasForeignKey<UserProfile>(p => p.UserId);
    }
}

public class TaskConfiguration : IEntityTypeConfiguration<TaskModel>
{
    public void Configure(EntityTypeBuilder<TaskModel> builder)
    {
        builder.ToTable("Tasks");

        builder.HasKey(t => t.Id);

        builder.HasOne(t => t.AssignedUser)
            .WithMany()
            .HasForeignKey(t => t.AssignedUserId)
```

```

        .onDelete(DeleteBehavior.SetNull);

builder.HasOne(t => t.Client)
    .WithMany(c => c.Tasks)
    .HasForeignKey(t => t.ClientId)
    .onDelete(DeleteBehavior.SetNull);

builder.HasIndex(t => t.Status);
builder.HasIndex(t => t.AssignedUserId);
    }
}

public class NoteConfiguration : IEntityTypeConfiguration<Note>
{
    public void Configure(EntityTypeBuilder<Note> builder)
    {
        builder.ToTable("Notes");

        builder.HasKey(x => x.Id);

        builder.Property(x => x.Content)
            .IsRequired();

        builder.Property(x => x.CreatedAt)
            .IsRequired();

        builder.HasOne(x => x.Deal)
            .WithMany()
            .HasForeignKey(x => x.DealId)
            .onDelete(DeleteBehavior.Restrict);

        builder.HasOne(x => x.User)
            .WithMany()
            .HasForeignKey(x => x.UserId)
            .onDelete(DeleteBehavior.Restrict);
    }
}

public class DealConfiguration : IEntityTypeConfiguration<Deal>
{
    public void Configure(EntityTypeBuilder<Deal> builder)
    {
        builder.ToTable("Deals");

        builder.HasKey(x => x.Id);

        builder.HasOne(x => x.Client)
            .WithMany()
            .HasForeignKey(x => x.ClientId)
            .onDelete(DeleteBehavior.Restrict);

        builder.HasOne(x => x.AssignedUser)
            .WithMany()
            .HasForeignKey(x => x.AssignedUserId)
            .onDelete(DeleteBehavior.SetNull);
    }
}

public class CompanyConfiguration : IEntityTypeConfiguration<Company>
{
    public void Configure(EntityTypeBuilder<Company> builder)
    {
        builder.ToTable("Companies");
    }
}

```

```

        builder.HasKey(c => c.Id);
    }
}

public class ClientConfiguration : IEntityTypeConfiguration<Client>
{
    public void Configure(EntityTypeBuilder<Client> builder)
    {
        builder.ToTable("Clients");

        builder.HasKey(x => x.Id);

        builder.HasOne(x => x.Company)
            .WithMany(c => c.Clients)
            .HasForeignKey(x => x.CompanyId)
            .IsRequired(false);
    }
}

public class AiRecommendationConfiguration :
IEntityTypeConfiguration<AiRecommendation>
{
    public void Configure(EntityTypeBuilder<AiRecommendation> builder)
    {
        builder.ToTable("AiRecommendations");

        builder.HasKey(x => x.Id);

        builder.Property(x => x.SuccessProbability)
            .IsRequired();

        builder.Property(x => x.NextSteps)
            .IsRequired();

        builder.Property(x => x.GeneratedAt)
            .IsRequired();

        builder.HasOne(x => x.Deal)
            .WithMany()
            .HasForeignKey(x => x.DealId)
            .OnDelete(DeleteBehavior.Restrict);
    }
}

```

Компонент відображення реєстру угод

```
import { Component, OnInit } from '@angular/core';
import { CommonModule, CurrencyPipe, DatePipe } from '@angular/common'; //
Пайпи для форматування грошей і дат
import { FormsModule } from '@angular/forms';

import { TableModule } from 'primeng/table';
import { ButtonModule } from 'primeng/button';
import { TagModule } from 'primeng/tag';
import { InputTextModule } from 'primeng/inputtext';
import { IconFieldModule } from 'primeng/iconfield';
import { InputIconModule } from 'primeng/inputicon';

export interface Deal {
  id: number;
  title: string;
  clientName: string;
  amount: number;
  stage: 'Lead' | 'Negotiation' | 'Proposal' | 'Won' | 'Lost';
  expectedCloseDate: Date;
}

@Component({
  selector: 'app-deals',
  standalone: true,
  imports: [
    CommonModule,
    FormsModule,
    TableModule,
    ButtonModule,
    TagModule,
    InputTextModule,
    IconFieldModule,
    InputIconModule
  ],
  providers: [CurrencyPipe, DatePipe],
  templateUrl: './deals.component.html'
})
export class DealsComponent implements OnInit {
  deals: Deal[] = [];
  searchQuery: string = '';

  ngOnInit() {
    this.deals = [
      {
        id: 101,
        title: 'Розробка корпоративного сайту',
        clientName: 'ТОВ "Альфа Тех" (Олександр)',
        amount: 4500,
        stage: 'Proposal',
        expectedCloseDate: new Date('2026-04-15')
      },
      {
        id: 102,
        title: 'Поставка 50 ноутбуків',
        clientName: 'Нова Логістика (Марія)',
        amount: 32000,
        stage: 'Negotiation',
        expectedCloseDate: new Date('2026-03-30')
      },
      {
        id: 103,
        title: 'SEO оптимізація (Q2)',

```

```

        clientName: 'Індивідуальний підприємець (Іван)',
        amount: 800,
        stage: 'Won',
        expectedCloseDate: new Date('2026-03-20')
    },
    {
        id: 104,
        title: 'Інтерпація 1С',
        clientName: 'ТОВ "МераБуд"',
        amount: 1200,
        stage: 'Lead',
        expectedCloseDate: new Date('2026-05-01')
    }
];

getStageSeverity(stage: string): "success" | "secondary" | "info" |
"warn" | "danger" | "contrast" {
    switch (stage) {
        case 'Won': return 'success';
        case 'Lost': return 'danger';
        case 'Proposal': return 'warn';
        case 'Negotiation': return 'info';
        default: return 'secondary';
    }
}

getStageLabel(stage: string): string {
    const labels: Record<string, string> = {
        'Lead': 'Новий лід',
        'Negotiation': 'Перемовини',
        'Proposal': 'КП надіслано',
        'Won': 'Успішно закрито',
        'Lost': 'Відмова'
    };
    return labels[stage] || stage;
}

get activePipelineValue(): number {
    return this.deals
        .filter(d => d.stage !== 'Lost' && d.stage !== 'Won')
        .reduce((sum, deal) => sum + deal.amount, 0);
}

get wonRevenue(): number {
    return this.deals
        .filter(d => d.stage === 'Won')
        .reduce((sum, deal) => sum + deal.amount, 0);
}

get newLeadsCount(): number {
    return this.deals.filter(d => d.stage === 'Lead').length;
}
}

```

Сервіс взаємодії сутності задач з серверним API

```
import { Injectable, inject } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Task } from '../models/task.model';

@Injectable({
  providedIn: 'root'
})
export class TaskService {
  private http = inject(HttpClient);
  private apiUrl = 'https://localhost:7066/crm/tasks';

  getTasks(): Observable<Task[]> {
    return this.http.get<Task[]>(this.apiUrl);
  }

  createTask(task: Partial<Task>): Observable<Task> {
    return this.http.post<Task>(this.apiUrl, task);
  }

  updateTaskStatus(id: number, status: number): Observable<any> {
    return this.http.patch(`${this.apiUrl}/${id}/status`, { status });
  }

  deleteTask(id: number): Observable<any> {
    return this.http.delete(`${this.apiUrl}/${id}`);
  }
}
```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”



ФАКУЛЬТЕТ ПРОГРАМНИХ СИСТЕМ ТА ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ З ВБУДОВАНИМ ПОМІЧНИКОМ
НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ УПРАВЛІННЯ
ВІДНОСИНАМИ З КЛІЄНТАМИ**

Виконав: Боньковський Ілля Віталійович, КП-21

Керівник: асистент кафедри ПЗКС, д-р філософії, Юсин Яків
Олексійович

Київ – 2026

1/16

ПОСТАНОВКА ЗАДАЧІ



Мета проєкту: розробити програмне забезпечення із вбудованим інтелектуальним модулем для автоматизації рутинних процесів обліку, структурування історії взаємодії та підтримки прийняття рішень менеджерами.

Завдання:

1. Проаналізувати існуючі CRM-системи та визначити їх недоліки.
2. Обрати оптимальні засоби реалізації клієнт-серверної архітектури.
3. Спроекувати та розробити серверну, клієнтську частини та бази даних.
4. Інтегрувати модуль штучного інтелекту на базі великої мовної моделі.
5. Протестувати розроблене програмне забезпечення.
6. Сформулювати напрями подальшого розвитку продукту.

АКТУАЛЬНІСТЬ



- **Перевантаження рутиною:** Менеджери витрачають критично багато часу на аналіз історії взаємодії та самостійне планування наступних кроків замість активних продажів.
- **Втрата важливих деталей:** Під час тривалих циклів продажу фахівці часто забувають або не враховують попередні домовленості з клієнтом, що призводить до зриву угод.
- **Складність прийняття рішень:** Більшість систем працюють як пасивні архіви і не підказують менеджеру оптимальних дій для успішного закриття контракту.
- **Високий поріг впровадження інновацій:** Наявні на ринку інтелектуальні CRM-рішення занадто складні та дорогі для масового використання у малому бізнесі.

ІСНУЮЧІ РІШЕННЯ



Salesforce



Pipedrive



NetHunt CRM

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ



Критерії	Salesforce	Pipedrive	NetHunt CRM
Наявність воронки продажів	+	+	+
Інтегрований ШІ-помічник	+	+	–
Доступність ШІ у базових тарифах	–	–	–
Легкогаговий/не перевантажений інтерфейс	–	+	–
Безшовна інтеграція з екосистемою пошти	–	–	+

ЗАСОБИ РЕАЛІЗАЦІЇ

- Мова програмування бекенду: **C#**
- Фреймворк серверної частини: **ASP.NET Core**
- Основна реляційна база даних: **Microsoft SQL Server**
- База даних для телеметрії та логів: **MongoDB**
- Фреймворк клієнтської частини: **Angular (TypeScript)**
- Бібліотека UI-компонентів: **PrimeNG**
- Хмарний сервіс штучного інтелекту: **Google Gemini API**



АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

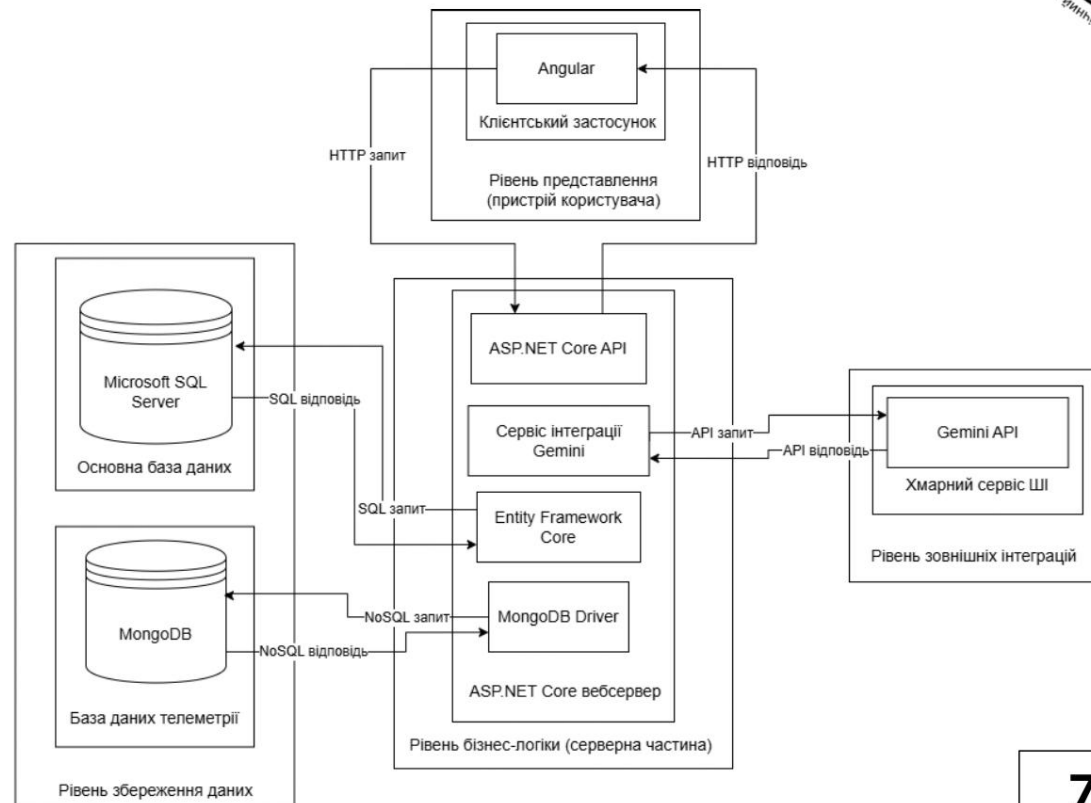
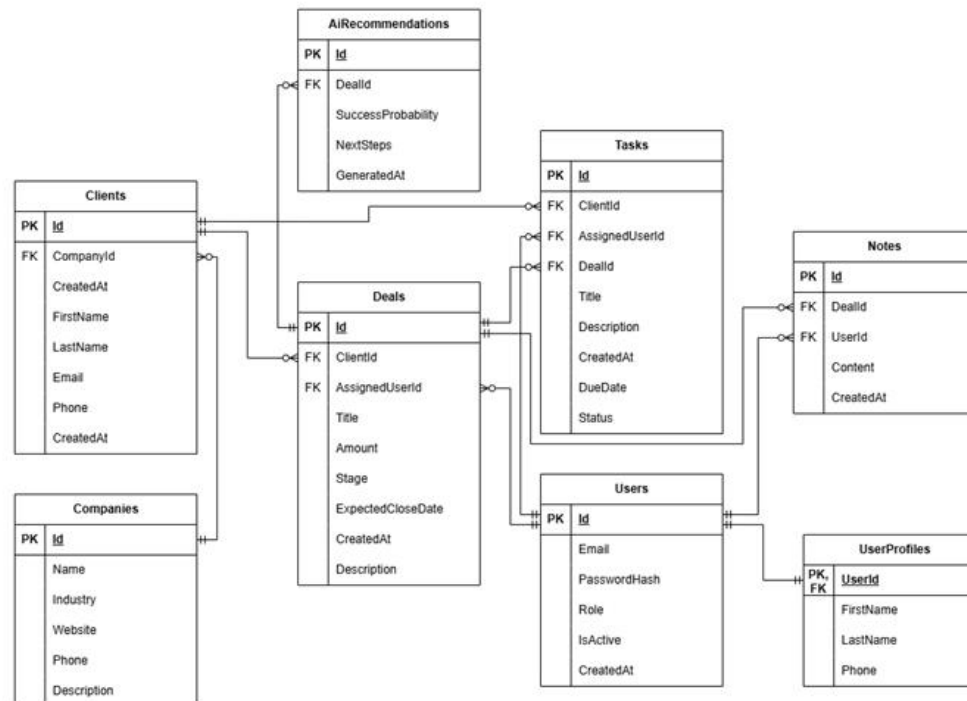
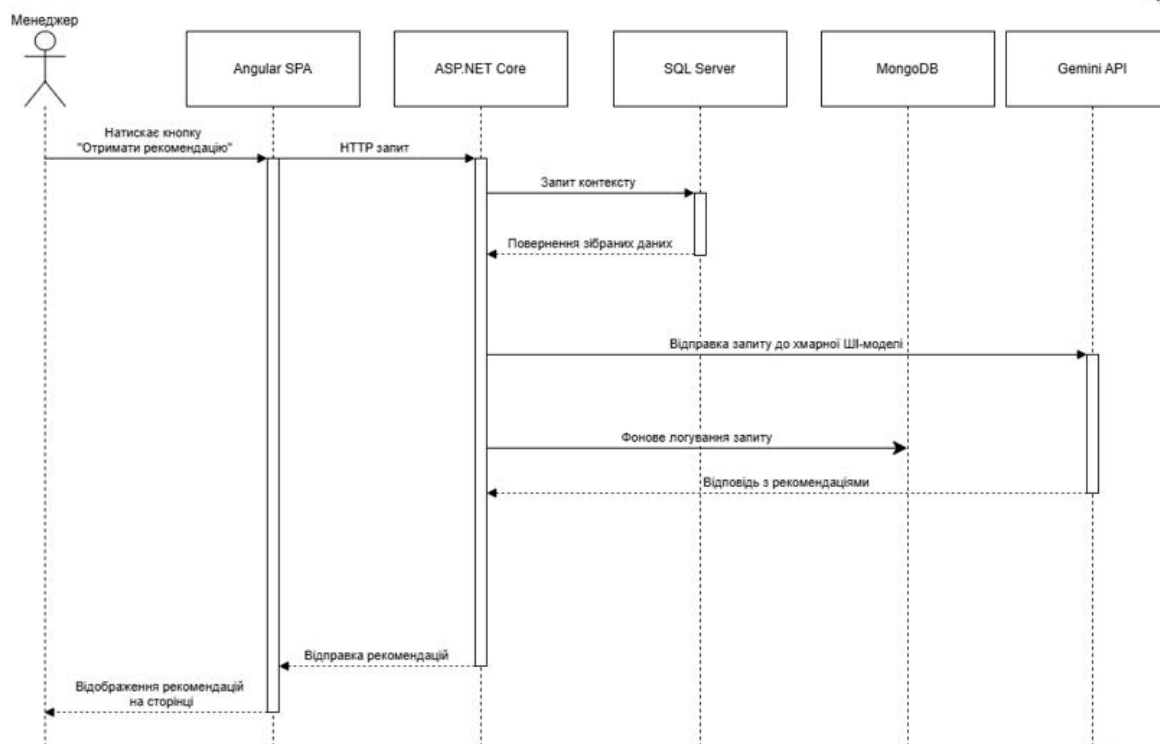


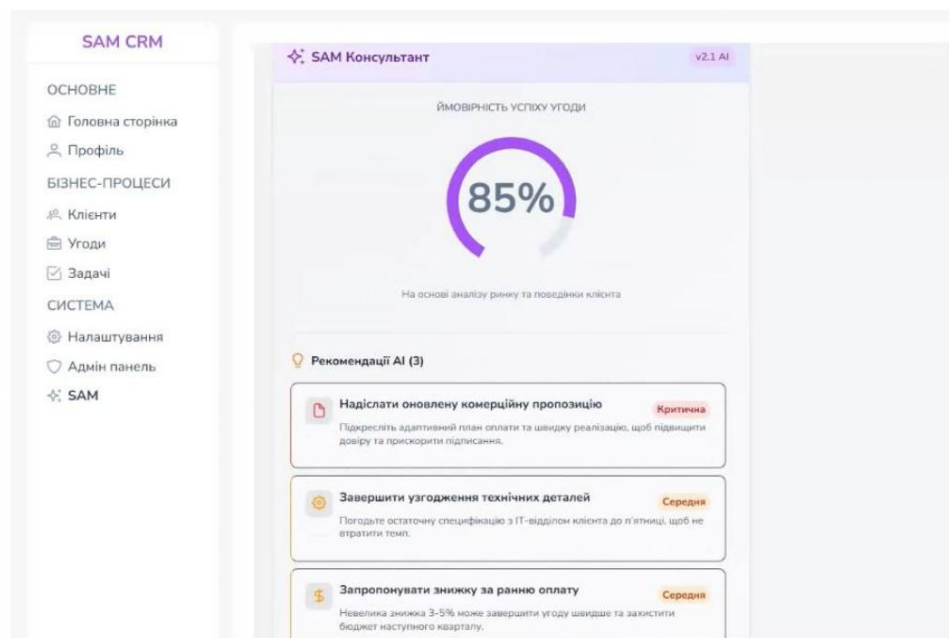
СХЕМА ОСНОВНОЇ БАЗИ ДАНИХ



ДІАГРАМА ПОСЛІДОВНОСТІ РОБОТИ ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ



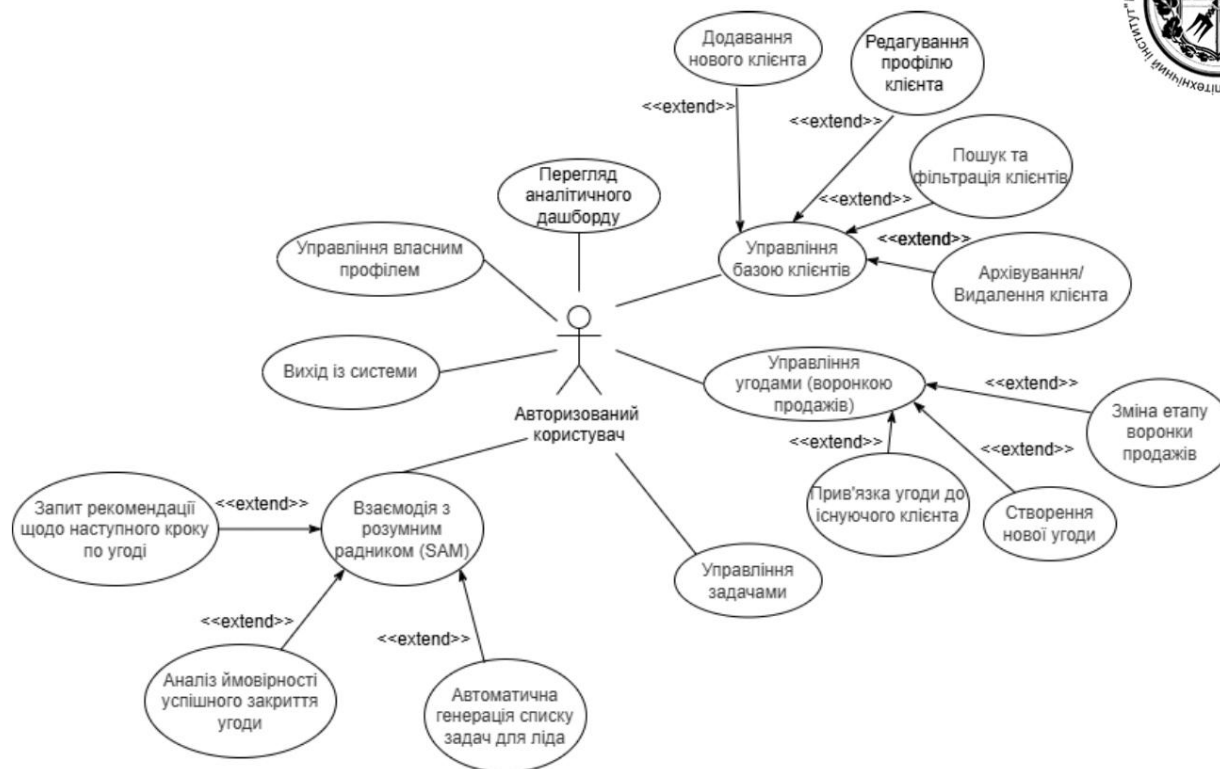
СТОРІНКА ІНТЕРФЕЙСУ СИСТЕМИ



Інтерфейс рекомендацій інтелектуального модуля

10/16

ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ



ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Процес тестування включав:

- Тестування безпеки (перевірка JWT-авторизації та обмеження доступу).
- Тестування валідації вхідних даних на рівні ASP.NET Core контролерів.
- Тестування транзакційної цілісності бази даних SQL Server.
- Асинхронне тестування інтеграції з Gemini API та паралельного запису в MongoDB.
- Тестування клієнтського інтерфейсу Angular (маршрутизація, фільтрація, відображення даних).

ПОРІВНЯННЯ З ІСНУЮЧИМИ РІШЕННЯМИ



Критерії	SAM CRM	Salesforce	Pipedrive	NetHunt
Наявність воронки продажів	+	+	+	+
Інтегрований ШІ-помічник	+	+	+	–
Низькі інфраструктурні витрати на ШІ	+	–	–	–
Розділення баз даних для оптимізації логів	+	+	–	–
Легкогаговий/швидкий SPA інтерфейс	+	–	+	–

ПОДАЛЬШИЙ РОЗВИТОК



- Впровадження ШІ для визначення «емоційного фону» комунікацій.
- Інтеграція з IP-телефонією: автоматична транскрипція дзвінків у текстові нотатки.
- Створення кастомних аналітичних звітів з експортом у PDF/Excel.
- Створення комерційних пропозицій та договорів засобами ШІ.

ВИСНОВКИ



- Проведено аналіз існуючих CRM-систем та обґрунтовано необхідність інтеграції ШІ.
- Спроектовано розподілену клієнт-серверну архітектуру з використанням стеку .NET та Angular.
- Розроблено та оптимізовано базу даних з використанням комбінованого підходу (SQL Server + MongoDB).
- Успішно реалізовано інтелектуальний модуль Smart Advisor на базі Gemini API для генерації рекомендацій.
- Протестовано програмне забезпечення, підтверджено його безпеку та продуктивність.
- Сформовано перспективні напрями подальшого розвитку продукту для бізнесу.



ДЯКУЮ ЗА УВАГУ

Факультет програмних систем та прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2025 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ З ВБУДОВАНИМ ПОМІЧНИКОМ
НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ УПРАВЛІННЯ
ВІДНОСИНАМИ З КЛІЄНТАМИ

Програма та методика тестування

ДП.045440-04-51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Ілля БОНЬКОВСЬКИЙ

ЗМІСТ

1. Об'єкт випробувань	3
2. Мета тестування	3
3. Методи тестування.....	3
4. Засоби та порядок тестування.....	4

1. ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробувань є програмне забезпечення для управління відносинами з клієнтами із вбудованим інтелектуальним помічником. Система призначена для автоматизації рутинних операційних процесів відділу продажів, обліку комерційних угод, планування робочих завдань та надання менеджерам рекомендацій за допомогою алгоритмів штучного інтелекту. Програмний продукт розроблений на основі клієнт-серверної архітектури з використанням таких технологій: C#, ASP.NET Core, TypeScript, Angular, Microsoft SQL Server, MongoDB та хмарного сервісу Google Gemini API.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Працездатність та коректне відображення UI-компонентів клієнтської частини.
2. Надійна маршрутизація та захист доступу до сторінок за допомогою механізму JWT-авторизації.
3. Коректне виконання CRUD-операцій для клієнтів, угод та завдань через серверний REST API.
4. Строга валідація вхідних даних на стороні сервера для запобігання збереженню некоректної інформації.
5. Підтримка цілісності та збереження реляційних зв'язків у базі даних Microsoft SQL Server.
6. Стабільність та безвідмовність фонових логів системних подій і телеметрії в нереляційну базу MongoDB.
7. Коректність збору текстового контексту угод та успішність двосторонньої взаємодії зі штучним інтелектом для генерації аналітичних рекомендацій.

3. МЕТОДИ ТЕСТУВАННЯ

Для оцінки якості розробленої CRM-системи застосовувався метод тестування «чорної скриньки», де увага приділялася коректності роботи функціональності з точки зору кінцевого користувача, а також інструментальне тестування кінцевих точок API, що дозволило забезпечити відповідність застосунку архітектурним вимогам.

Процес тестування базувався на ручному підході. Для кожної функції (модуль клієнтів, воронка продажів, модуль Smart Advisor) спочатку визначалися межі перевірки та критерії успішності. На основі цього створювався набір тестових випадків, які охоплювали як позитивні, так і негативні (помилкові) сценарії використання системи.

Виконання тестування полягало у послідовному відтворенні кожного тестового випадку та документуванні отриманих результатів, які порівнювалися з очікуваними. У разі виявлення невідповідностей (наприклад, помилок при збереженні об'єктів або проблем із відображенням даних на клієнті), вони аналізувалися на рівні серверних логів для встановлення причин, після чого вносилися необхідні виправлення. Останнім кроком було повторне тестування (регресійне), аби переконатися, що внесення змін не порушило роботу інших модулів CRM-системи.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Оцінка працездатності виконувалася вручну через вебінтерфейс, імітуючи реальні дії менеджера з продажів, а також за допомогою спеціалізованого програмного забезпечення (наприклад, Swagger/Postman) для перевірки серверних контролерів. Такий підхід дозволив своєчасно ідентифікувати недоліки як на рівні клієнта, так і на рівні бізнес-логіки сервера.

Процедура контролю якості включала наступні напрямки:

1. Ручне тестування стандартних бізнес-сценаріїв: створення карток клієнтів, додавання нових угод, їх переміщення по етапах воронки продажів та закриття завдань.
2. Перевірка обробки нестандартних сценаріїв на клієнті та сервері (введення від'ємних сум угод, некоректні формати електронної пошти, відсутність обов'язкових полів).
3. Перевірка генерації JWT-токенів, їх ротації та автоматичне блокування сервером запитів без достовірного токена.
4. Перевірка надійності бази даних SQL Server, тестування роботи зовнішніх ключів (наприклад, неможливість видалити клієнта, до якого прив'язані активні угоди, без каскадного підтвердження).
5. Перевірка швидкості збору контексту, відправки асинхронних запитів до хмарної ШІ-моделі та коректної десеріалізації JSON-відповідей для відображення рекомендацій у картці угоди.

Факультет програмних систем та прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2026 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ З ВБУДОВАНИМ ПОМІЧНИКОМ
НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ УПРАВЛІННЯ
ВІДНОСИНАМИ З КЛІЄНТАМИ

Керівництво користувача

ДП.045440-05-34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Ілля БОНЬКОВСЬКИЙ

ЗМІСТ

1. Опис структури програмного забезпечення	3
2. Сторінка авторизації в системі	4
3. Головна сторінка	5
4. Сторінка відображення клієнтів	5
5. Сторінка угод.....	6
6. Сторінка комерційної угоди та ІІІ-помічник	7
7. Сторінка робочих задач	8
8. Сторінка профілю співробітника.....	9
9. Сторінка панелі адміністратора та реєстрації	10

1. Опис структури програмного забезпечення

Програмне забезпечення SAM CRM є корпоративною платформою, що складається з інтерактивних вебсторінок, які динамічно перемикаються в результаті взаємодії користувача з інтерфейсом. Застосунок спроектований як односторінковий, що забезпечує швидку роботу без необхідності постійного перезавантаження сторінок у браузері. Він призначений для створення єдиного, зручного інформаційного середовища для менеджерів з продажу та адміністраторів системи.

Програмне забезпечення складається із таких основних сторінок та модулів:

- 1) сторінка входу в обліковий запис (авторизації);
- 2) головна сторінка (аналітична інформаційна панель);
- 3) сторінка відображення клієнтів;
- 4) сторінка воронки продажів;
- 5) сторінка деталей комерційної угоди з ШІ-помічником;
- 6) сторінка управління робочими завданнями;
- 7) сторінка панелі адміністратора;
- 8) сторінка профілю співробітника.

На кожній робочій сторінці присутня навігаційна бічна панель, через яку можна здійснювати переміщення між модулями системи.

До проходження процедури авторизації до облікового запису користувачу доступні лише сторінка входу. Після успішної авторизації користувач (залежно від його ролі у системі) переходить на головну сторінку та отримує повний доступ до інших функціональних розділів платформи. Такий механізм захисту надійно оберігає внутрішні бази даних компанії від несанкціонованого доступу ззовні. Крім того, автоматичне розмежування прав дозволяє сформувати безпечний та персоналізований робочий простір, де кожен співробітник взаємодіє лише з тією інформацією, яка безпосередньо стосується його посадових обов'язків.

2. Сторінка авторизації в системі

Сторінка авторизації (рис. 1) слугує єдиною точкою входу до платформи. Оскільки система містить комерційну таємницю, функція самостійної реєстрації для нових користувачів відсутня. Облікові дані для доступу генеруються та надаються співробітникам системним адміністратором.

У центрі екрана розташована форма входу. Для отримання доступу співробітник повинен ввести свою корпоративну електронну адресу та пароль у відповідні текстові поля, після чого натиснути кнопку «Увійти». У випадку введення неправильних даних система блокує вхід та виводить повідомлення про помилку перевірки.

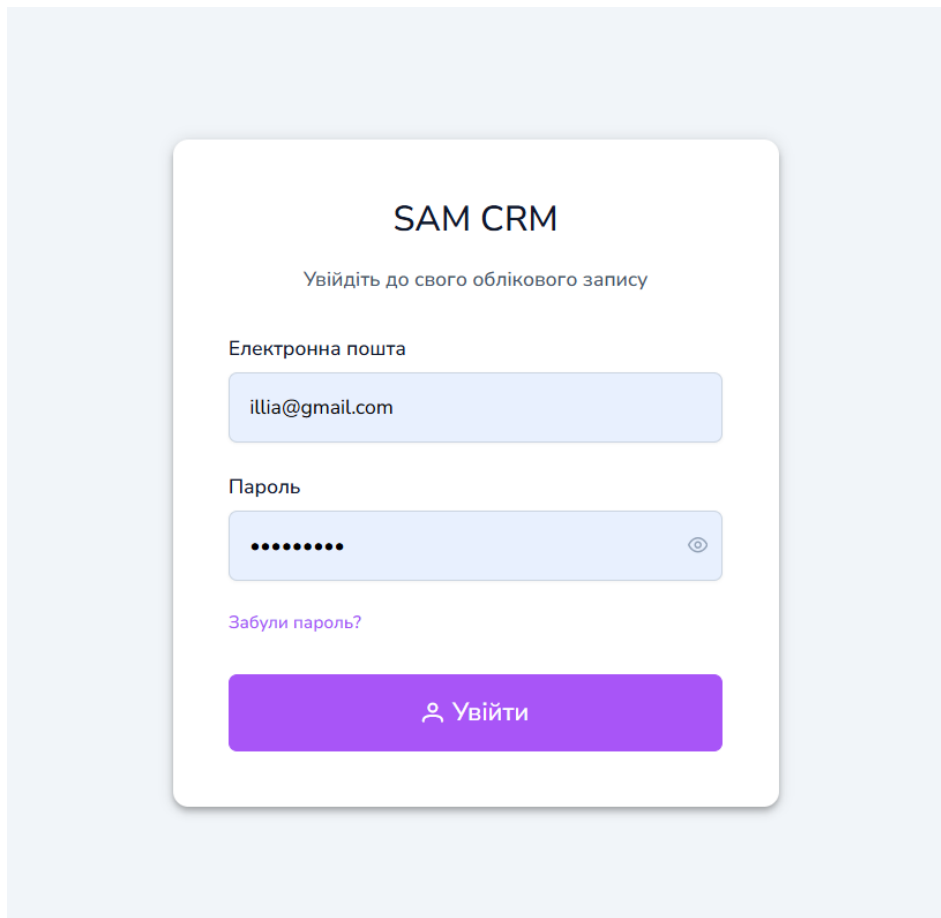


Рис. 1. Форма входу в обліковий запис

3. Головна сторінка

Після входу в систему користувач автоматично спрямовується на головну сторінку (рис. 2). Її ключова функція – надання швидкого зрізу аналітичних даних щодо поточної успішності менеджера.

Інтерфейс сторінки містить інформаційні віджети з метриками (наприклад, кількість відкритих угод, загальна сума очікуваного прибутку) та графіки, що візуалізують статистику продажів. Також на інформаційній панелі виводиться скорочений перелік пріоритетних завдань на поточний день, що допомагає співробітнику одразу сфокусуватися на термінових робочих процесах.

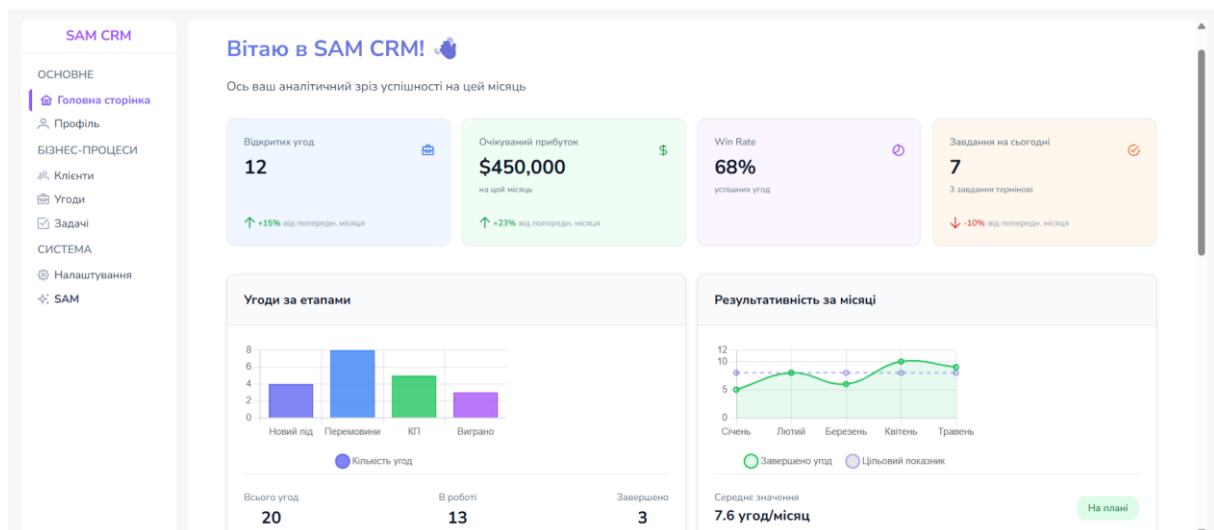


Рис. 2. Аналітична інформаційна панель

4. Сторінка відображення клієнтів

Керування контрагентами здійснюється через єдиний модуль – сторінку відображення клієнтів.

Сторінка побудована на базі інтерактивної таблиці. У ній відображається основна інформація про контактних осіб: імена, назви компаній, номери телефонів, електронні адреси та посади (рис. 3). Інтерфейс підтримує функції швидкого текстового пошуку та сортування за

колонками. За допомогою кнопок управління користувач може додати новий запис до бази, викликавши модальне вікно з формою, або відкрити існуючий запис для редагування контактних даних.

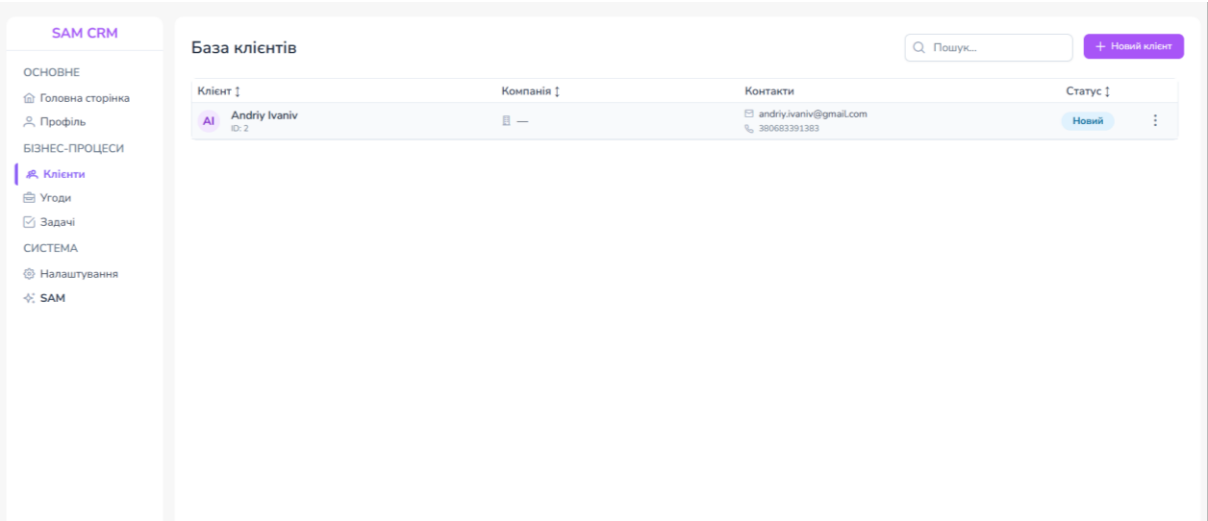


Рис. 3. Сторінка відображення бази клієнтів

5. Сторінка угод

Сторінка угод (рис. 4) є основним інструментом менеджера для відстеження поточного стану комерційних контрактів та воронки продажів.

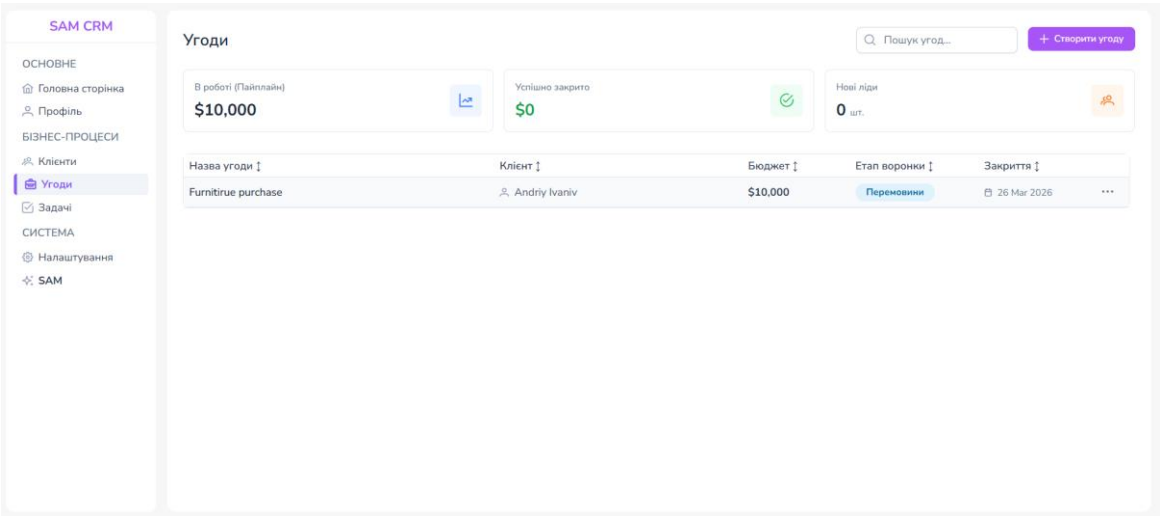


Рис. 4. Сторінка угод

У верхній частині робочої області розташована панель з інформаційними картками, які відображають ключові фінансові та кількісні метрики: загальну суму угод, що знаходяться в роботі, суму успішно закритих угод та кількість нових лідів. Над картками праворуч розміщено поле для швидкого текстового пошуку угод, а також фіолетову кнопку додавання нового запису.

Основну площу сторінки займає інтерактивна таблиця. У ній виведено ключову інформацію по кожному контракту: назву угоди, ім'я прив'язаного клієнта, орієнтовний бюджет, поточний етап воронки (відображається у вигляді графічного індикатора, наприклад, «Перемовини») та дату очікуваного закриття.

Заголовки колонок підтримують функцію сортування даних, а в кінці кожного рядка передбачено меню додаткових дій (іконка з трьома крапками), що дозволяє перейти до детального перегляду або редагування конкретної угоди.

6. Сторінка комерційної угоди та ШІ-помічник

Натискання на картку угоди відкриває детальну сторінку контракту (рис. 5). Тут зберігається вся історія комунікації, прикріплені компанії, відповідальні особи та текстові нотатки. Праворуч на сторінці інтегровано модуль ШІ-помічника. Менеджер має змогу натиснути кнопку для ініціалізації інтелектуального аналізу поточної угоди. Після обробки контексту застосунок виводить відсоткову ймовірність успішного закриття контракту та генерує список персоналізованих рекомендацій щодо того, який крок варто зробити наступним для просування угоди. Такий підхід мінімізує ризик прийняття помилкових рішень на ключових етапах переговорів та підвищує загальну продуктивність відділу продажів.

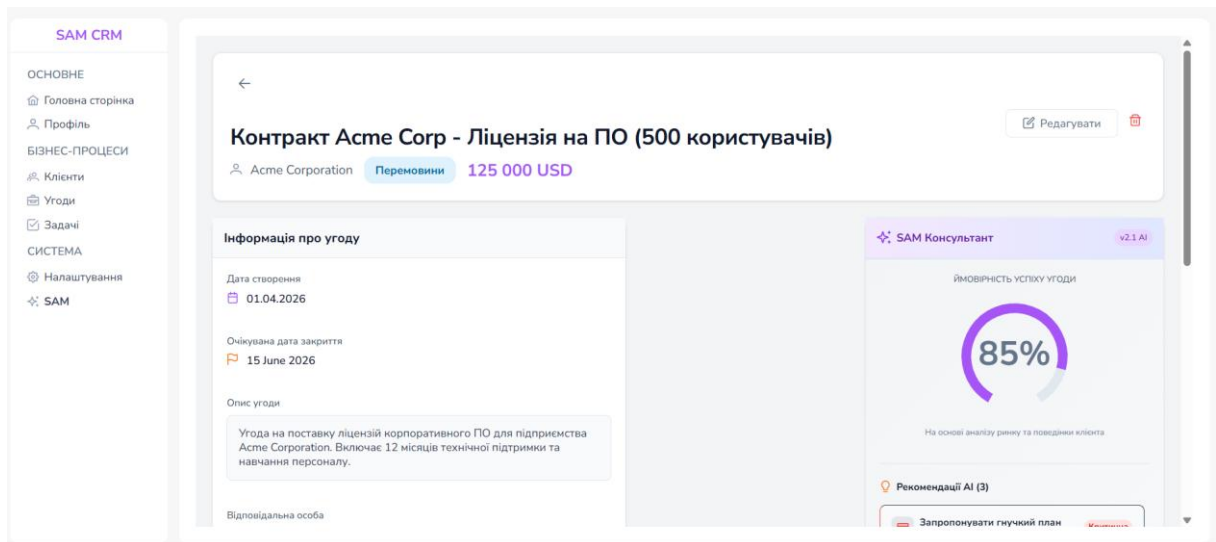


Рис. 5. Сторінка деталей угоди з модулем штучного інтелекту

7. Сторінка робочих задач

Сторінка задач (рис. 6) призначена для організації щоденної роботи менеджера та зручного ведення списку справ.

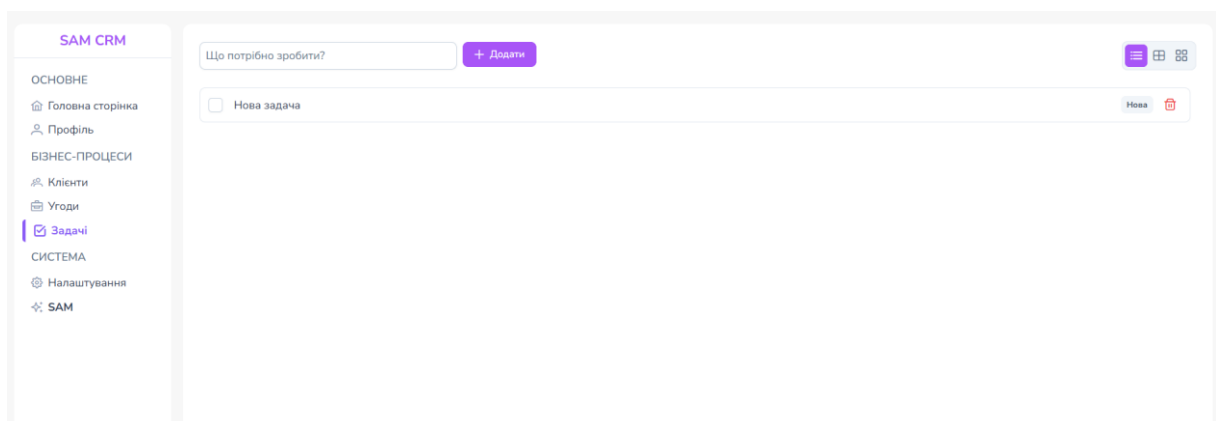


Рис. 6. Інтерфейс управління задачами

У верхній частині робочої області розташовано поле введення з текстовою підказкою «Що потрібно зробити?». Для створення нового завдання користувачу достатньо ввести його короткий опис у це поле та натиснути фіолетову кнопку «Додати». Праворуч від поля введення розміщено групу іконок для перемикавання режимів відображення:

користувач може обрати вигляд звичайного списку, Kanban-дошки або сітки залежно від власних уподобань.

Нижче розташований сам перелік доданих завдань. Кожен рядок репрезентує окрему задачу та містить інтерактивний прапорець ліворуч для швидкої відмітки про її виконання. Далі йде безпосередньо текст завдання та графічний бейдж поточного статусу (наприклад, «Нова»). У кінці кожного рядка передбачена іконка кошика (червоного кольору), натискання на яку дозволяє миттєво видалити запис із системи.

8. Сторінка профілю співробітника

На цій сторінці (рис.7) поточний користувач може переглядати свою персональну інформацію, статистику активності в системі та керувати налаштуваннями власного облікового запису (наприклад, зміна пароля або оновлення контактного номера). Це забезпечує підтримання актуальності робочих даних та дозволяє співробітнику самостійно контролювати безпеку свого доступу до корпоративної системи.

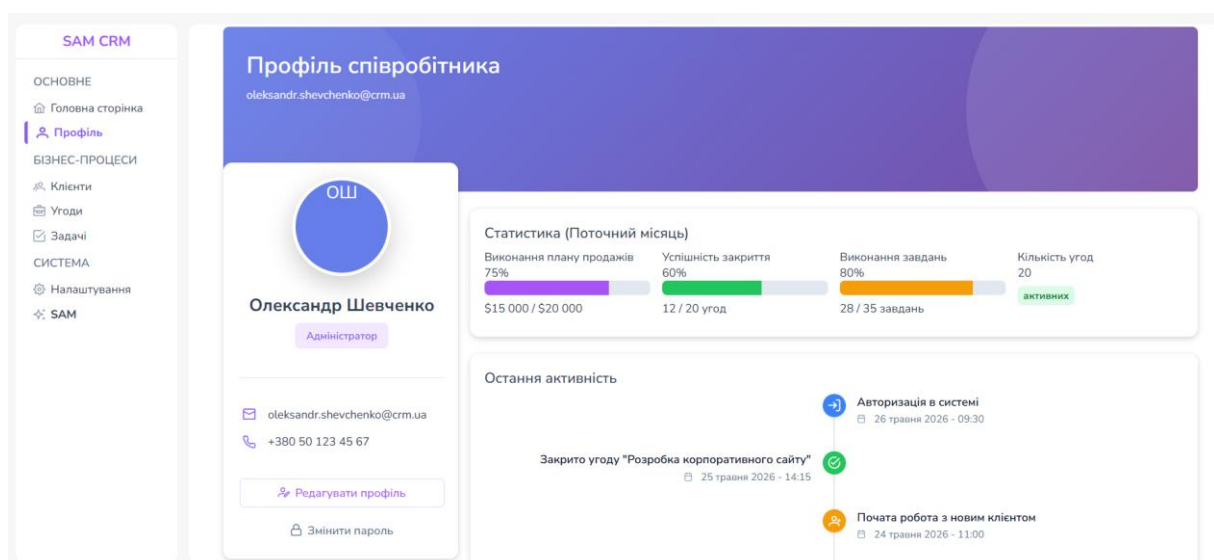


Рис. 7. Сторінка профіля користувача

9. Сторінка панелі адміністратора та реєстрації

Панель адміністратора є захищеним розділом, доступним виключно адміністраторам системи. Тут зосереджені інструменти управління доступом (рис. 8).

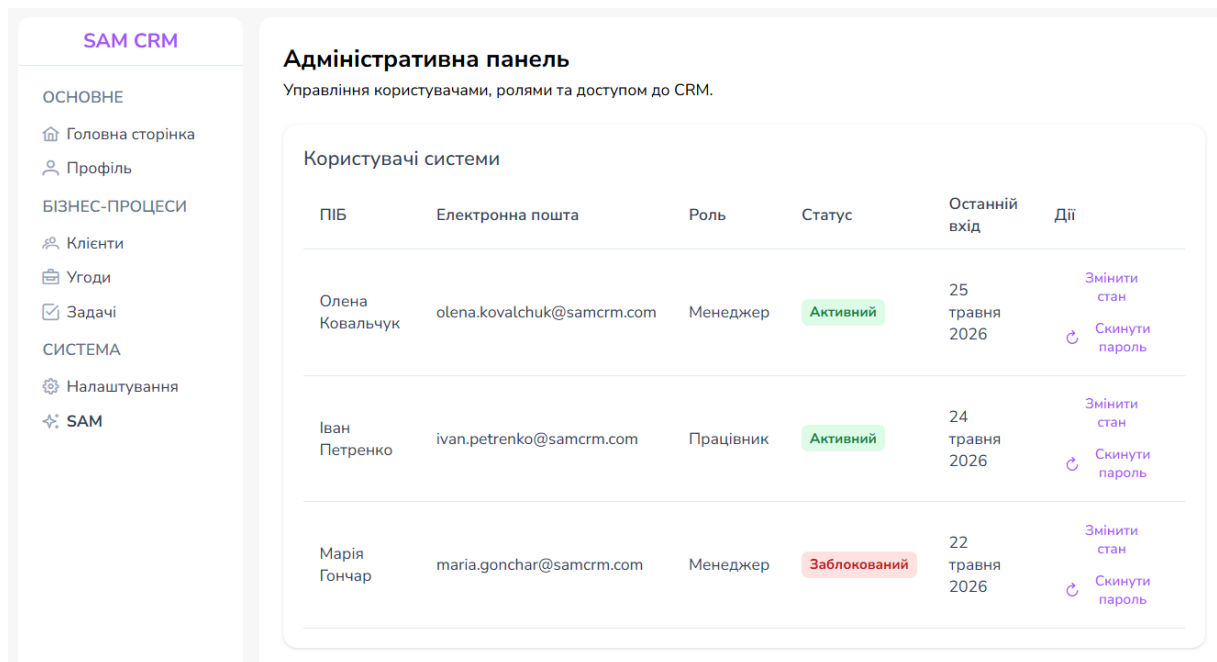


Рис. 8. Сторінка панелі адміністратора

Сторінка відображає таблицю з усіма співробітниками компанії, які мають доступ до CRM. Саме в цьому модулі адміністратор використовує окрему форму реєстрації нового користувача (рис. 9) для створення облікових записів новим співробітникам. Також адміністратор може блокувати неактивні акаунти, скидати паролі та призначати рівні прав доступу, гарантуючи корпоративну безпеку. Такий централізований підхід значно спрощує процеси адаптації та звільнення персоналу, зводячи всі дії з управління доступом до єдиного зручного інтерфейсу. Крім того, чітке розмежування ролей унеможливорює несанкціоноване втручання в конфіденційну інформацію клієнтів, що є важливим для збереження комерційної таємниці підприємства.

Новий користувач

Ім'я

Прізвище

Email

Телефон

Роль

Менеджер

Створити користувача

Будь ласка, заповніть усі обов'язкові поля.

Рис. 9. Форма реєстрації нового користувача до системи