Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Створення інформаційної системи для аналітичного опрацювання телефонних дзвінків засобами Django, Celery, Celery Beat, PostgreSQL, Redis

Виконав: студент VI курсу, групи СНмз-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Вовнянка Г.Р.
(підпис) (прізвище та ініціали)

Керівник _____ Пасічник В.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Стадник М.А.
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« 17 » листопада 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Вовнянці Галині Романівні
(прізвище, ім'я, по батькові)
1. Тема роботи Створення інформаційної системи для аналітичного опрацювання телефонних дзвінків засобами Django, Celery, Celery Beat, PostgreSQL, Redis

Керівник роботи Пасічник Володимир Володимирович, д.т.н., професор кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від « 27 » листопада 2025 року № 4/7-1041
2. Термін подання студентом завершеної роботи 22 грудня 2025 р.
3. Вихідні дані до роботи Наукові публікації про інформаційні системи, аналітичне опрацювання даних, контейнеризацію, опрацювання аудіо інформації телефонних дзвінків.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Предметна область спеціалізованих аналітичних систем. 1.1 Актуальність дослідження та створення спеціалізованих аналітичних систем. 1.3 Оптимізація бізнес-процесів через AI-аналіз дзвінків. 2 Засоби розробки інформаційної системи аналізу дзвінків. 2.1 Вибір програмно-алгоритмічної платформи для системи аналізу дзвінків. 2.2 Django та Celery для асинхронної обробки ресурсомістких завдань опрацювання аудіо даних телефонних дзвінків. 2.3 Контейнеризація сервісів за допомогою Docker для потокового аналітичного опрацювання аудіо дзвінків. 3 Практична реалізація інформаційної системи аналізу телефонних дзвінків. 3.1 Принципи та підходи до обробки телефонних дзвінків. 3.2 Архітектура серверної частини системи аналітики дзвінків. 3.3 Налаштування Docker контейнерів для функціонування інформаційної системи аналітичного опрацювання телефонних дзвінків. 3.4 Інтеграція інформаційної системи з Binotel для збору та обробки дзвінків. 3.5 Інтеграція з KeepinCRM для синхронізації даних клієнтів та дзвінків.
4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження. 3 Завдання дослідження. 4 Актуальність дослідження. 5 Архітектура інформаційної системи 6 Серверна частина ІС. 7. Схема БД ІС. 8. Характеристика сутностей та атрибутів БД. 9. Архітектура Celery 3 Django і Redis. 10. Gemini API. 11. Схема роботи системи аналізу дзвінків. 12. Інтерфейс користувача. 13. Висновки. 14. Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Гурик О.Я., доцент кафедри МТ		
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 17 листопада 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	17.11.2025	
2.	Аналіз науково-технічних публікацій та збір даних по	17.11.2025-24.11.2025	
	темі кваліфікаційної роботи		
3.	Виконання дослідження згідно мети кваліфікаційної	25.11.2025-08.12.2025	
	роботи		
4.	Оформлення розділу «Предметна область	09.12.2025-11.12.2025	
	спеціалізованих аналітичних систем»		
5.	Оформлення розділу «Засоби розробки інформаційної	09.12.2025-11.12.2025	
	системи аналізу дзвінків»		
6.	Оформлення розділу «Практична реалізація	09.12.2025-11.12.2025	
	інформаційної системи аналізу телефонних дзвінків»		
7.	Виконання завдання до підрозділу «Охорона праці»	25.12.2025-08.12.2025	
8.	Виконання завдання до підрозділу «Безпека в	25.12.2025-08.12.2025	
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	12.12.2025	
10.	Нормоконтроль	15.12.2025-16.12.2025	
11.	Перевірка на плагіат	17.12.2025	
12.	Попередній захист кваліфікаційної роботи	18.12.2025	
13.	Захист кваліфікаційної роботи	23.12.2025	

Студент _____ Вовнянка Г.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пасічник В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення інформаційної системи для аналітичного опрацювання телефонних дзвінків засобами Django, Celery, Celery Beat, PostgreSQL, Redis // Кваліфікаційна робота освітнього ступеня «Магістр» // Вовнянка Галина Романівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНмз-61 // Тернопіль, 2025 // С. 74, рис. – 3, табл. – 1, кресл. – , додат. – 3, бібліогр. – 58.

Ключові слова: аналітика, дзвінки, автоматизація, інформаційна система, фреймворк, база даних, інтеграція, асинхронність.

Кваліфікаційна робота присвячена розробці інформаційної системи для автоматизованого аналізу телефонних дзвінків із використанням Django, Celery, Celery Beat, PostgreSQL та Redis.

У першому розділі описані проблеми та тенденції аналізу дзвінків, сучасні технології обробки комунікацій, функціональні та нефункціональні вимоги, а також сформульовано постановку задачі автоматизованого аналізу.

У другому розділі досліджено засоби та інструменти реалізації системи, обґрунтовано вибір Django та Celery, розглянуто контейнеризацію за допомогою Docker і схему бази даних PostgreSQL.

У третьому розділі описано практичну реалізацію системи, проаналізовано принципи обробки дзвінків та архітектуру серверної частини, проведено налаштування Docker і інтеграцію з Binotel та KeepinCRM.

Об'єкт дослідження: процес автоматизованого аналізу дзвінків у компаніях.

Предмет дослідження: інформаційна система для збору, обробки та аналітики дзвінків.

ANNOTATION

Creating an Information System for Analytical Processing of Telephone Calls Using Django, Celery, Celery Beat, PostgreSQL and Redis // The educational level "Master" qualification work // Vovnianka Halyna Romanivna // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNmz-61 group // Ternopil, 2025 // P. 74, fig. – 3, tables – 1, posters – , annexes – 3, ref. – 58.

Key words: analytics, calls, automation, information system, framework, database, integration, asynchrony.

The qualification work is devoted to the development of an information system for automated analysis of telephone calls using Django, Celery, Celery Beat, PostgreSQL and Redis.

The first section describes the problems and trends of call analysis, modern technologies for processing communications, functional and non-functional requirements, and also formulates the statement of the task of automated analysis.

The second section examines the means and tools of implementing the system, justifies the choice of Django and Celery, considers containerization using Docker and the PostgreSQL database schema.

The third section describes the practical implementation of the system, analyzes the principles of call processing and the architecture of the server part, configures Docker and integrates with Binotel and KeepinCRM.

Object of research: the process of automated call analysis in companies.

Subject of research: an information system for collecting, processing and analyzing calls.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AES – симетричний метод криптографії, що застосовується для захисту інформації.

API – це програмний механізм, який сприяє обміну даними та взаємодії між програмами.

DNS – система доменних імен.

GDPR – регламент ЄС, що встановлює правила та стандарти захисту персональних даних фізичних осіб.

HTTP – це набір правил, за якими відбувається передавання даних між веб-сервером і браузером.

JSON – це текстовий формат, призначений для впорядкованого опису даних та їх обміну через мережу.

NLP – напрямок штучного інтелекту, який виконує автоматичний аналіз та обробку природної людської мови.

REST – це підхід до побудови взаємодії між клієнтом і сервером, який базується на використанні HTTP.

SSL – це протокол захисту, призначений для створення зашифрованого та безпечного каналу зв’язку між сервером і клієнтом.

URL – це унікальна адреса в Інтернеті, що вказує на розташування конкретного ресурсу.

VPS – це віртуальна машина на фізичному сервері з окремими правами користувача.

CRM – це програмне забезпечення для обліку клієнтів та управління взаємодією з ними.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному бізнесі якість комунікації з клієнтами є ключовим чинником досягнення високих результатів, насамперед у сфері продажів. Телефонні дзвінки надалі залишаються важливим каналом взаємодії, а традиційний ручний аналіз розмов вимагає значних зусиль, є суб'єктивним та недостатньо гнучким за масштабом. Це ускладнює своєчасну діагностику проблем і впливає на рівень обслуговування та лояльність клієнтів.

Сучасні технології – обробка природної мови, машинне навчання, вебтехнології та робота з великими даними – дають змогу автоматизувати й прискорити аналіз дзвінків. Розроблення серверної частини вебзастосунку для аналітичного опрацювання дзвінків із використанням Django, Celery, Celery Beat, PostgreSQL та Redis є важливим науково-прикладним завданням.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи ступеня «Магістр» є створення серверної частини інформаційної системи для автоматизованого аналізу телефонних дзвінків, що забезпечує обробку великих обсягів даних, оцінювання якості діалогу за визначеними критеріями та формування аналітичних звітів. Реалізація поставленої мети передбачає виконання таких завдань:

– провести огляд сучасного стану досліджень і рішень у сфері автоматизації аналізу комунікацій та обробки аудіоданих;

– дослідити існуючі методи обробки природної мови та підходи до оцінювання якості телефонних розмов;

– виконати порівняння архітектурних підходів до побудови серверної логіки для систем аналітичного опрацювання даних;

– розробити прототип серверної частини вебзастосунку з використанням Django, Celery, Celery Beat, PostgreSQL та Redis, що забезпечує автоматизацію аналізу телефонних дзвінків.

Об’єкт дослідження Процеси автоматизованої обробки та аналізу даних телефонних розмов у веборієнтованих інформаційних системах.

Предмет дослідження. Методи, моделі та технології аналітичного опрацювання телефонних дзвінків, а також засоби серверної реалізації відповідних програмних рішень.

Наукова новизна одержаних результатів полягає у подальшому розвитку та адаптації методів автоматизованої оцінки телефонних розмов до умов вебзастосунку, що працює з великими обсягами даних та використовує сучасні інструменти асинхронної обробки, зокрема Celery і Redis.

Практичне значення одержаних результатів. Виконано створення прототипу серверної частини інформаційної системи, яка автоматизує аналіз телефонних дзвінків, знижує навантаження на quality-менеджерів, забезпечує об’єктивність оцінювання комунікацій та створює передумови для впровадження прогнозних моделей і персоналізованих сценаріїв взаємодії з клієнтами.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на XIII науково-технічної конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2025 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додатки А).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 58 найменувань та 3 додатків. Загальний обсяг кваліфікаційної роботи складає 74 сторінки, з них 52 сторінки основного тексту, який містить 3 рисунки та 1 таблицю.

1 ПРЕДМЕТНА ОБЛАСТЬ СПЕЦІАЛІЗОВАНИХ АНАЛІТИЧНИХ СИСТЕМ

1.1 Актуальність дослідження та створення спеціалізованих аналітичних систем

У сучасному конкурентному бізнес-середовищі якість взаємодії з клієнтами є одним із ключових факторів успіху, особливо для компаній, що займаються реалізацією товарів і послуг. Телефонні звінки залишаються важливим каналом комунікації, який впливає на їхню задоволеність, лояльність та готовність до повторних покупок. Оцінка ефективності таких контактів є складним і трудомістким процесом. Зазвичай контроль якості здійснюють вручну менеджери, що супроводжується низкою проблем: витрачається багато часу на обробку, обмежені можливості масштабування та сповільнюється формування аналітичних звітів. Через це компанії втрачають шанс швидко виявляти слабкі місця у взаємодії з клієнтами, коригувати стратегії продажів і підвищувати продуктивність відділів збуту.

Відсутність автоматизації процесів для всебічного контролю телефонних взаємодій значно ускладнює процес масштабування бізнес-процесів. У той же час, сучасні технологічні рішення – зокрема методи обробки природної мови (NLP) [1], аналіз емоційного відтінку мовлення та алгоритми машинного навчання – створюють можливості для розробки спеціалізованих інструментів. Такі інструменти дають змогу не лише автоматизувати оцінку комунікацій, але й формувати глибокі аналітичні висновки щодо якості проведених діалогів. Впровадження подібних систем стає особливо актуальним для компаній, які прагнуть скоротити операційні витрати, зміцнити конкурентні позиції та забезпечити стабільно високий рівень обслуговування клієнтів. Таким чином, автоматизація оцінювання телефонних розмов набуває значущого практичного значення, оскільки її ефективна реалізація сприятиме оптимізації ключових бізнес-процесів і створенню додаткової цінності для споживачів.

Важливість проблеми зростає у зв’язку зі збільшенням попиту на цифрові інструменти для управління взаємодією з клієнтами. Дослідження демонструють, що організації, які впроваджують передові технології аналізу комунікацій, відзначаються вищим рівнем утримання клієнтів та позитивною динамікою фінансових показників. Водночас, більшість доступних на ринку систем або складні для інтеграції, або не відповідають специфічним потребам підприємств у сфері комерційної діяльності. Створення інформаційної системи [2], що забезпечує автоматизовану обробку повного обсягу телефонних розмов та генерує детальні аналітичні звіти, дасть змогу усунути існуючі прогалини, надаючи компаніям ефективний інструмент для підвищення стандартів обслуговування та оптимізації комунікаційних процесів.

1.2 Сучасні інформаційні технології аналізу телефонних дзвінків

Телефонні дзвінки продовжують відігравати ключову роль у взаємодії з клієнтами у продажах, проте їхній аналіз за допомогою традиційних методів, що здійснюються вручну quality-менеджерами, є:

– трудомістким;
– суб’єктивним;
– з обмеженим масштабом.

Сучасні інноваційні технології, зокрема обробка природної мови (NLP), машинне навчання, аналіз емоційного відтінку голосу та хмарні обчислення [3], дають змогу автоматизувати процес аналізу телефонних розмов. Це суттєво підвищує його продуктивність, точність та об’єктивність. Використання цих технологій у інформаційних системах для обробки дзвінків створює можливості для швидкого виявлення проблем у комунікації, оптимізація сценаріїв продажів і підвищення рівня якості обслуговування клієнтів. Обробка природної мови (NLP) виступає ключовим елементом автоматизованого аналізу розмов.

Автоматичні системи розпізнавання мовлення (ASR), як от Google Speech-to-Text [4] чи DeepSpeech [5], виконують первинну функцію конвертації голосових записів у письмовий (текстовий) формат. Ця початкова стадія є

критично важливою, оскільки вона уможливлює опрацювання значних масивів телефонних розмов без потреби у трудомісткому ручному прослуховуванні. Наступним кроком є лінгвістичний аналіз отриманого тексту за допомогою алгоритмів NLP, які часто використовують моделі, розроблені на основі архітектури BERT. Ці алгоритми здійснюють всебічне оцінювання діалогів за низкою критеріїв, включаючи: дотримання затверджених скриптів (шаблонів) комунікації; фіксацію присутності/відсутності заздалегідь визначених термінів і ключових виразів; ідентифікацію критичних точок у бесіді, як от висловлення клієнтом незгоди (заперечень) або вагань (сумнівів).

Крім того, методики обробки природної мови (NLP) [1] надають змогу ідентифікувати емоційну тональність письмового матеріалу, що створює передумови для оцінки емоційного контексту та загального сенсу діалогу.

Машинне навчання (ML) виступає як доповнення до NLP, пропонуючи механізми для класифікації комунікацій за критеріями якості, виявлення нетипових ситуацій та прогнозування підсумків. Моделі ML, розроблені на базі таких фреймворків як TensorFlow, PyTorch [6] або Scikit-learn [7], проходять тренування на історичних даних, що робить їх адаптивними до унікальних вимог певної компанії. Зокрема, алгоритми, що застосовують методи кластерного аналізу, можуть автоматично ідентифікувати нестандартні випадки, як от суттєві відхилення від регламентованого сценарію або виникнення конфліктних бесід, без потреби втручання людини.

Аналіз емоційного забарвлення голосу додає ще один важливий вимір до всебічної оцінки якості спілкування. Застосовуючи спеціалізовані інструменти, наприклад, бібліотеку Librosa, система проводить аналіз акустичних характеристик: частоти (висоти) голосу, інтенсивності (гучності) та швидкості (темпу) мовлення. Це дає змогу встановити емоційний стан учасників розмови, їхню ступінь впевненості та рівень професіоналізму. Таким чином, досягається більш холістична (комплексна) картина комунікаційного процесу, оскільки об'єднуються дані, отримані як із тексту, так і з аудіозапису.

Хмарні обчислювальні платформи, як от AWS, Google Cloud та Microsoft Azure [8], забезпечують необхідну інфраструктурну базу для ефективного

опрацювання великих масивів даних. Використання хмарних технологій дає змогу одночасно обробляти тисячі дзвінків, динамічно масштабувати систему відповідно до поточного навантаження та підключати готові API для розпізнавання мовлення або аналізу тексту. Це зменшує капітальні витрати на власну IT-інфраструктуру та дає змогу застосовувати систему у компаніях різного рівня.

Застосування цих технологій забезпечує швидке, об’єктивне та масштабоване проведення аналізу, що дає змогу бізнесу своєчасно реагувати на виявлені проблеми та вдосконалювати процеси продажів. Водночас виникають виклики, зокрема потреба у достовірних даних для навчання моделей, підтримка різних мов і діалектів та забезпечення конфіденційності інформації, що вимагає відповідних заходів:

– шифрування;
– мультимовних моделей;
– алгоритмів подавлення шумів.

У перспективі інформаційну систему для аналітичного опрацювання телефонних дзвінків можна розширити шляхом інтеграції з CRM-платформами, впровадження прогнозних аналітичних інструментів та розробки індивідуальних рекомендацій для менеджерів, що підвищить її практичну цінність для бізнесу.

Інноваційні технології, як от штучний інтелект (AI) та машинне навчання (ML) [9], забезпечують суттєве підвищення ефективності процесу оцінювання телефонних комунікацій. Завдяки застосуванню автоматичних систем транскрипції мовлення та алгоритмів обробки природної мови (NLP) з'являється можливість оперативно переводити голосові діалоги в текстову форму, виокремлювати найважливіші терміни, аналізувати емоційне забарвлення бесіди та встановлювати інтенції співрозмовників. Такий підхід дає змогу компаніям оперативно обробляти клієнтські запити, підвищувати ефективність обслуговування та знаходити недоліки у взаємодії.

Крім того, сучасні аналітичні засоби дають змогу в режимі реального часу:

– моніторити розмови;
– формувати звіти;

– прогнозувати дії клієнтів, спираючись на зібрані дані.

Це створює умови для:

– персоналізованого підходу;

– ефективнішого навчання персоналу;

– оптимізацію бізнес-процесів.

Застосування інноваційних рішень у сфері обслуговування телефонних комунікацій дає змогу компаніям отримати переваги над конкурентами та забезпечує зростання рівня лояльності і задоволеності споживачів.

1.3 Оптимізація бізнес-процесів через AI-аналіз дзвінків

У реаліях сучасного ринкового середовища, де боротьба за лояльність споживачів невпинно посилюється, ефективна комунікаційна стратегія відіграє критичну роль, насамперед у галузі реалізації товарів та послуг. Телефонне спілкування продовжує зберігати статус одного з провідних засобів контакту з клієнтською аудиторією, проте його оцінювання через призму класичних методик, що виконується фахівцями з контролю якості в ручному режимі, характеризується:

– повільним;

– суб’єктивним;

– обмеженим охопленням аналізу.

Застосування технологій штучного інтелекту (AI) [10] для аналізу телефонних дзвінків дає можливість автоматизувати цей процес:

– підвищуючи якість бізнес-процесів;

– оптимізуючи роботу сейлз-команд;

– підвищуючи рівень задоволеності клієнтів.

Технології штучного інтелекту, що застосовуються для аналізу телефонних розмов, включають:

– обробки природної мови (NLP);

– машинного навчання;

– визначення емоційного тону мовлення.

Сучасні технології NLP, розпізнавання мовлення Google Speech-to-Text [4] та машинне навчання, дають змогу автоматично перетворювати аудіозаписи розмов на текст і всебічно їх аналізувати. Системи оцінюють дзвінки за якістю, відповідністю сценарію, наявністю ключових слів, а також прогнозують успіх продажів на основі історичних даних. Крім того, аналіз тональності голосу допомагає визначити емоційний стан учасників та рівень професіоналізму менеджера.

Застосування такого підходу сприяє оптимізації бізнес-процесів у кількох напрямах.

По-перше, впровадження автоматизованого аналізу телефонних розмов значно скорочує час, необхідний для їх оцінки, оскільки забезпечує обробку великої кількості дзвінків у режимі реального часу. Це дає змогу фахівцям з контролю якості звільнитися від рутинних завдань та зосередитися на аналітичних і стратегічних аспектах, як от покращення сценаріїв продажів.

По-друге, застосування технологій штучного інтелекту забезпечує об'єктивність результатів аналізу, зменшуючи вплив суб'єктивних факторів. Це сприяє отриманню стандартизованих і відтворюваних оцінок, що підвищує довіру до підготовленої звітності.

По-третє, аналітичні звіти, створені на основі інтелектуальної обробки даних, дають змогу своєчасно виявляти проблемні аспекти взаємодії з клієнтами, зокрема неефективні мовні конструкції або типові заперечення. Це дає можливість швидко адаптувати та оптимізувати стратегію продажів.

Поєднання інструментів інтелектуального аналізу з хмарними сервісами, як от AWS або Google Cloud [8], забезпечує можливість гнучкого масштабування системи та ефективної обробки великих обсягів інформації без необхідності створювати та підтримувати власну апаратну інфраструктуру. Цей підхід робить інформаційно технологічні рішення вигідними для організацій різного розміру – від невеликих інноваційних проектів до великих корпоративних структур. Крім того, технології AI-аналізу можуть бути інтегровані з CRM-платформами , що надає керівництву аналітичні рекомендації щодо оптимального ведення

комунікації з клієнтами, а також інструменти для прогнозування їхньої поведінки. Це, в свою чергу, створює умови для підвищення ефективності взаємодії з клієнтами та поліпшення показників конверсії.

Інтеграція інтелектуальних систем аналізу надає підприємствам значні стратегічні переваги у довгостроковій перспективі. Виявлення стабільних закономірностей у ході діалогів дає змогу вдосконалювати не лише самі методи комунікації, а й суміжні управлінські процеси, як от підготовка персоналу та ефективне планування робочих змін. Зокрема, технології штучного інтелекту можуть визначати співробітників, які потребують додаткового підвищення кваліфікації, а також точно прогнозувати періоди максимальної активності клієнтів. Такий підхід сприяє оптимальному використанню ресурсів і дає змогу суттєво зменшувати операційні витрати [10].

Разом із тим, практична реалізація AI-рішень стикається з низкою викликів, серед яких ключовими є забезпечення наявності репрезентативних і якісних даних для навчання моделей, дотримання високих стандартів інформаційної безпеки та врахування лінгвістичного різноманіття, включно з регіональними діалектами. Для подолання цих проблем застосовуються комплексні підходи, зокрема методи криптографічного захисту даних, багатомовні лінгвістичні моделі та алгоритми очищення аудіосигналу від сторонніх шумів.

У стратегічній перспективі функціонал інтелектуального аналізу може розвиватися в напрямку предиктивного моделювання ключових змін у поведінкових патернах клієнтів або автоматичного формування сценаріїв взаємодії, що додатково підвищує економічну цінність і значущість таких інструментів для сучасного бізнесу.

1.4 Оцінка та визначення вимог до інформаційної системи для аналітичного опрацювання телефонних дзвінків

Оцінка та визначення вимог до інформаційної системи для аналітичного опрацювання телефонних дзвінків відіграє важливу роль у розробці системи. Цей процес дає можливість детально сформулювати функціональні та нефункціональні вимоги системи, врахувати особливості операційної діяльності торговельних підприємств та забезпечити створення програмного продукту, здатного ефективно автоматизувати оцінку телефонних розмов. Визначення специфікацій базується на аналізі потреб потенційних користувачів, існуючих технічних обмежень та поточного рівня розвитку технологій.

Для формування переліку вимог було проведено аналіз стандартних операційних процесів підприємств, діяльність яких значною мірою залежить від телефонних комунікацій. Головне завдання, на розв'язання якого спрямована інформаційна система для аналітичного опрацювання телефонних дзвінків, полягає у зменшенні високої ресурсомісткості та впливу людського фактору, що характерні для ручної оцінки розмов фахівцями з контролю якості. Сучасний бізнес потребує програмного рішення, яке забезпечує автоматичну обробку всього масиву дзвінків, гарантує об’єктивну перевірку за встановленими параметрами та генерує детальну аналітику для вдосконалення стратегій взаємодії з клієнтами. Основними користувачами системи є менеджери з якості, керівники відділів продажу та технічні адміністратори, відповідальні за налаштування та супровід платформи.

Функціональна специфікація програмного рішення визначається кількома ключовими напрямами.

По-перше, передбачено автоматичне перетворення аудіофайлів у текст за допомогою сервісів розпізнавання мовлення, як от Google Speech-to-Text [4].

По-друге, здійснюється семантичний аналіз тексту за допомогою NLP-алгоритмів для контролю виконання скриптів, виявлення ключових слів та заперечень клієнта.

По-третє, інформаційна система для аналітичного опрацювання телефонних дзвінків виконує акустичний аналіз для оцінки емоційного забарвлення розмови та професіоналізму менеджера.

Також передбачено формування систематизованих звітів, що включають:

– оцінку якості дзвінків;

– виявлені проблеми;

– пропозиції щодо вдосконалення процесів.

У завершальному етапі передбачено інтегрування з CRM-системами [11], що дає змогу автоматично зберігати результати аналізу та застосовувати їх у бізнес-процесах.

До нефункціональних вимог відносяться:

– продуктивність;

– безпека;

– зручність використання.

Інформаційна система має демонструвати високу продуктивність при обробці великих обсягів даних з мінімальними затримками, що потребує використання хмарних платформ, як от AWS або Google Cloud [8], для динамічного масштабування ресурсів. Захист інформаційних даних є пріоритетом, оскільки записи розмов часто містять конфіденційну інформацію. Для цього застосовуються методи криптографічного захисту під час передачі та зберігання даних, а також дотримання міжнародних стандартів, зокрема GDPR.

Серверна архітектура має бути оптимізована для безперешкодної взаємодії з зовнішніми додатками через протоколи RESTful API [12] або GraphQL [13]. Важливою характеристикою також є здатність інформаційної системи підтримувати різні мови та регіональні діалекти, що забезпечує ефективне використання продукту на глобальному ринку [14].

1.4.1 Функціональні вимоги до інформаційної системи аналізу телефонних дзвінків

Функціональні вимоги до інформаційної системи для аналітичного опрацювання телефонних дзвінків визначають її ключові можливості, які забезпечують результативну експлуатацію в межах автоматизації моніторингу переговорів та підвищення ефективності операційної діяльності у торговельній сфері. Перелік специфікацій сформовано з урахуванням потреб основних груп користувачів – фахівців із контролю якості, керівників відділів продажу та адміністраторів – з одночасним забезпеченням сумісності зі сторонніми платформами та гнучкості інструментів обробки даних.

Розглянемо перелік функціональних вимог:

1. Керування доступом та ідентифікація користувачів. Інформаційна система реалізує функціонал створення персональних профілів та перевірки повноважень через введення логіна й пароля. Під час формування облікового запису вказуються базові атрибути – електронна адреса, ім’я, посадові обов'язки. З метою посилення захисту застосовується криптографічне хешування паролів та механізми мультифакторного підтвердження особи для адміністраторів.

2. Взаємодія з телекомунікаційними сервісами. Інформаційна система забезпечує сполучення з поширеними платформами IP-телефонії, як от Twilio, Asterisk або RingCentral, через програмні інтерфейси для автоматизованого збору записів розмов. Користувачам надано інструментарій для конфігурування з’єднання за допомогою токенів та ключів доступу, що гарантує стабільне імпортування медіаданих будь-якого розширення.

3. Синхронізація з CRM-рішеннями. Інформаційна система підтримує інтеграційні зв’язки з провідними інструментами управління відносинами з клієнтами, як от Salesforce, Pipedrive або HubSpot, використовуючи відповідні API [15]. Це дає можливість:

– здійснювати автоматичне зіставлення висновків аналізу дзвінків з профілями клієнтів у CRM;

– зберігати звіти;

– надсилати менеджерам відповідні рекомендації.

Користувачі мають змогу налаштовувати параметри взаємодії через інтерфейс інформаційної системи.

4. Аналіз однієї або декількох бесід дає змогу проводити оцінювання як окремих сесій зв'язку, так і масивів діалогів за конкретний термін. Процедура дослідження охоплює:

– автоматичну трансформацію аудіозапису в текст за допомогою інструментів розпізнавання мовлення, наприклад Google Speech-to-Text;

– лінгвістичне опрацювання отриманого тексту алгоритмами NLP для моніторингу дотримання скриптів, пошуку ключових виразів та виявлення складних моментів;

– визначення емоційного забарвлення мовлення через аналіз акустичних характеристик голосу учасників.

Підсумкова інформація надається у форматі розгорнутої аналітики за кожним викликом або консолідованої звітності за сукупністю контактів.

5. Систематизація викликів. Інформаційна система має підтримувати функцію групування діалогів за різними ознаками:

– номером телефону клієнта;

– тривалістю дзвінка;

– датою.

Впорядкування реалізується через панель користувача з опцією вибору декількох фільтрів та визначенням напрямку показу інформації (за зростанням або спаданням).

6. Ознайомлення з аналітикою. Користувачам надається доступ до розгорнутої звітності, що містить:

– результати аналізу дзвінків;

– показники якості (у відсотковому еквіваленті чи за бальною шкалою);

– ідентифіковані недоліки (порушення сценарію, суперечки з боку замовника);

– статистику ключових слів;

– поради щодо оптимізації роботи.

Дані можуть відображатися у форматі таблиць, діаграм або текстових описів із можливістю вивантаження у PDF чи CSV. Також передбачено механізм створення персоналізованих шаблонів за вибраними метриками.

Викладені вимоги гарантують практичність та багатофункціональність інформаційної системи для аналітичного опрацювання телефонних дзвінків, даючи користувачам можливість якісного дослідження переговорів, синхронізації з наявним ПЗ та отримання змістовної бази для покращення торговельних операцій. Практичне втілення цих характеристик потребує застосування сучасних технологій, на кшталт архітектури RESTful API для системної інтеграції та інструментарію NLP на базі Gemini [16].

1.4.2 Нефункціональні вимоги до інформаційної системи аналізу телефонних дзвінків

Нефункціональні вимоги до інформаційної системи для аналітичного опрацювання телефонних дзвінків окреслюють перелік експлуатаційних параметрів, що гарантують:

– надійність;

– продуктивність;

– безпеку.

Даний перелік критеріїв базується на особливостях функціонування торговельного бізнесу, врахуванні технологічних лімітів та вимогах до безпечного опрацювання масштабних обсягів інформації, що має конфіденційний характер.

1. Продуктивність. Інформаційна система має здійснювати опрацювання інтенсивних потоків даних, зокрема тисяч викликів щодоби, за мінімальний проміжок часу. Тривалість обробки п'ятихвилинного аудіофайлу, що охоплює розпізнавання мовлення, семантичний аналіз та оцінку емоцій, не повинна перевищувати тридцяти секунд. Задля високої швидкодії серверна частина має базуватися на хмарних інфраструктурах, як от AWS або Google Cloud [8],

підтримуючи паралельні обчислення та автоматичне регулювання потужностей відповідно до поточного трафіку.

2. Масштабованість. Інформаційна система для аналітичного опрацювання телефонних дзвінків має бути здатною до розширення в міру зростання бази клієнтів та інформаційного навантаження. Інформаційна система повинна зберігати стабільність при одночасній активності до ста користувачів та аналізі близько десяти тисяч розмов на день, забезпечуючи нарощування ресурсів без переривання сервісу. Для цього доцільно впровадити мікросервісну архітектуру та хмарні технології з гнучким керуванням активами [17].

3. Безпека. З огляду на конфіденційний характер переговорів, інформаційна система має гарантувати надійний захист відомостей на всіх стадіях їхнього оброблення. Необхідно реалізувати криптографічне шифрування під час передачі (протоколи HTTPS, TLS 1.2+) та у сховищах (стандарт AES-256). Доступ до інформаційної системи реалізується через механізми автентифікації, а операції адміністраторів мають фіксуватися в журналах аудиту. Інформаційна система повинна функціонувати згідно з міжнародними нормативами захисту даних, зокрема регламентом GDPR.

4. Надійність та безперебійність. Інформаційна система для аналітичного опрацювання телефонних дзвінків повинна гарантувати стабільну роботу з показником доступності не нижче 99,9% та мінімізувати ймовірність виникнення технічних збоїв. Інформаційна система має підтримувати механізми автоматичного відновлення після програмних чи апаратних помилок через використання децентралізованих баз даних та систем резервування. У випадку критичної ситуації допустимий обсяг втрати даних обмежений порогом у 0,01%.

5. Сумісність та інтеграційна здатність. Серверний компонент інформаційної системи має забезпечувати підтримку поширених форматів аудіо WAV, MP3, OGG та безперешкодну взаємодію з телефоніями Twilio, Asterisk і CRM-платформами HubSpot, Salesforce на основі протоколів RESTful або GraphQL. Також інформаційна система повинна коректно функціонувати в багатомовному середовищі, застосовуючи лінгвістичні моделі NLP для аналізу різних мов та діалектів.

6. Ергономічність та зручність експлуатації. Технічний опис API серверної частини інформаційної системи має бути деталізованим та відповідати стандарту OpenAPI [18], що спрощує процес її поєднання з іншим програмним забезпеченням. При штатному навантаженні швидкість відповіді API повинна становити не більше двохсот мілісекунд. Графічна оболонка інформаційної системи для аналітичного опрацювання телефонних дзвінків у частині налаштування зовнішніх сервісів має вирізнятися високим рівнем юзабіліті та простотою виконання операцій.

7. Керування помилками та стійкість до перешкод. Інформаційна система має демонструвати коректну роботу з медіафайлами низької якості, зокрема за наявності фонового шуму, низької інтенсивності звуку або переривчастого сигналу, завдяки застосуванню алгоритмів фільтрації та цифрової корекції аудіо. У випадках, коли аналіз запису є технічно неможливим, інформаційна система повинна надсилати користувачу сповіщення з роз’ясненням причини. Інформаційна система має передбачати розширені механізми обробки виняткових ситуацій для запобігання аварійним зупинкам сервісу.

8. Універсальність розгортання. Серверний компонент інформаційної системи для аналітичного опрацювання телефонних дзвінків повинен характеризуватися незалежністю від апаратної платформи та операційного середовища. Це досягається шляхом використання технологій контейнеризації на прикладі Docker, що значно спрощує процес міграції між хмарними інфраструктурами. Такий підхід забезпечує високу гнучкість при виборі технічної бази та сприяє зниженню видатків на її супровід.

Окреслені нефункціональні параметри гарантують, що інформаційна система для аналітичного опрацювання телефонних дзвінків буде продуктивною, безпечною і зручною для використання в реальних комерційних обставинах. Вони повністю відповідають потребам сучасних підприємств щодо оперативного та результативного опрацювання значних масивів даних.

1.5 Постановка задачі створення інформаційної системи для аналітичного опрацювання телефонних дзвінків

Ключовим завданням є розробка серверної архітектури інформаційної системи для аналітичного опрацювання телефонних дзвінків, призначеної для автоматизованого моніторингу якості комунікацій. Це виступає вагомим інструментом для підприємств торговельної сфери, де рівень взаємодії з контрагентами прямо визначає успішність бізнес-результатів. Реалізація інформаційної системи для аналітичного опрацювання телефонних дзвінків покликана змінити ресурсозатратний та суб'єктивний метод експертного оцінювання, що проводиться фахівцями з контролю якості, на продуктивне, об'єктивне та масштабоване цифрове середовище. Інформаційна система має забезпечувати опрацювання звукових записів, проводити їх експертизу згідно із заданими метриками, як от відповідність алгоритмам продажу, пошук маркерних фраз або ідентифікація проблемних зон, та формувати підсумкову звітність, яка дає змогу:

– оптимізувати комунікаційні стратегії;
– підвищити ефективність сейлз-команд;
– оптимізувати досвід взаємодії з клієнтами.

Реалізація поставленого завдання передбачає впровадження інструментів реєстрації та ідентифікації користувачів із розмежуванням прав доступу для адміністраторів, quality-менеджерів та керівників. Захист відомостей забезпечується через криптографічне хешування паролів та можливість застосування двофакторного підтвердження особи задля посилення безпеки. Інформаційна система має підтримувати взаємодію з поширеними телекомунікаційними сервісами, як от Asterisk [19], Twilio [20] або RingCentral [21], за допомогою програмних інтерфейсів для автоматизованого збору аудіофайлів у форматах MP3, WAV чи OGG. Також заплановано реалізацію процесу синхронізації з CRM-платформами, як от Salesforce, HubSpot або Pipedrive, з метою:

– автоматичного збереження результатів аналізу;

– кореляції отриманих відомостей із персональними картками клієнтів;

– надання персоналу змістовних порад щодо подальшої комунікації.

Базові можливості бекенд-складової інформаційної системи для аналітичного опрацювання телефонних дзвінків здійснює поетапний автоматичний аналіз викликів.

На початковій стадії звукові дані трансформуються у текстовий формат завдяки сервісам розпізнавання мови, як от Google Speech-to-Text, з урахуванням лінгвістичних особливостей та регіональних діалектів.

На наступному кроці текстовий масив досліджується інструментами обробки природної мови, які перевіряють дотримання регламентів продажу, здійснюють пошук маркерних виразів та фіксують критичні аспекти, наприклад заперечення з боку замовника або порушення корпоративних стандартів спілкування.

Одночасно інформаційна система здійснює дослідження вокальних характеристик за допомогою інструментарію на кшталт Librosa, визначаючи емоційний фон співрозмовників, інтонаційне забарвлення та рівень фаховості менеджера, що сприяє отриманню повнішої картину якості розмови.

Задля комфортної взаємодії реалізовано можливість структурувати виклики за такими критеріями:

– номером телефону клієнта;

– тривалістю розмови;

– ім’ям працівника;

– датою;

– іншими метаданими.

Інформаційна система генерує докладну звітність, яка містить показники якості комунікацій наприклад у відсотковому вираженні або за бальною шкалою, а також включає:

– оцінку якості дзвінків (наприклад, у відсотках або за шкалою);

– виявлені проблеми;

– статистику ключових слів;

– пропозиції щодо вдосконалення.

Аналітичні дані можуть подаватися у формі таблиць, діаграм або текстових описів із можливістю вивантаження у формати PDF чи CSV та адаптації шаблонів під індивідуальні запити користувачів.

Інформаційна система має демонструвати високу швидкодію, опрацьовуючи до десяти тисяч викликів щодня, при цьому тривалість аналізу одного аудіофайлу довжиною до п’яти хвилин не повинна перевищувати тридцяти секунд. Для досягнення таких показників передбачено використання хмарних середовищ, як от AWS або Google Cloud [8], що гарантують паралельне опрацювання інформації та автоматичне нарощування потужностей під навантаженням.

Пріоритетним є забезпечення безпеки:

– відомості підлягають криптографічному захисту через протоколи HTTPS та TLS версії один ціла дві десятих або вище під час передачі, а також алгоритм AES-двісті п’ятдесят шість у місцях зберігання;

– архітектура має цілковито відповідати світовим вимогам щодо приватності, передусім загальному регламенту про захист даних GDPR.

Інформаційна система повинна:

– здійснювати аналіз аудіофайлів низької якості з наявністю шумів або слабким рівнем гучності за допомогою методів цифрової фільтрації;

– інформувати персонал у випадку виникнення технічних перешкод під час обробки.

Вхідні дані:

– аудіозаписи розмов;

– супровідні відомості про виклики, зокрема контактний номер абонента, тривалість розмови, ПІБ працівника та календарна дата;

– параметри синхронізації з телекомунікаційними платформами та сервісами керування відносинами з клієнтами.

Вихідні дані:

– транскрибовані тексти;

– аналітичні висновки, що включають бальну оцінку якості, ідентифіковані критичні аспекти та емоційне забарвлення мовлення;

– звітні документи, інтегровані у CRM-платформи.

Інформаційна система має характеризуватися надійністю, гнучкістю та готовністю до подальшого масштабування, наприклад, через впровадження інструментів прогнозної аналітики чи автоматичного формування алгоритмів продажу, що суттєво підвищує її цінність для комерційної діяльності.

1.6 Висновок до першого розділу

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр» описано актуальність створення інформаційної системи для аналітичного опрацювання телефонних дзвінків та окреслено ключові аспекти використання сучасних технологій для реалізації такого інформаційного продукту. Розглянуто вплив автоматизованого аналізу дзвінків на оптимізацію бізнес-процесів, а також проаналізовано можливості застосування Django, Celery, Celery Beat, PostgreSQL та Redis для забезпечення ефективної, масштабованої та надійної обробки даних.

У межах розділу визначено функціональні й нефункціональні вимоги до майбутньої інформаційної системи, що дає змогу сформулювати чітке бачення її структури, призначення та ключових можливостей. Також подано обґрунтування необхідності створення інформаційно-технологічного інструменту такого класу та сформульовано основну постановку задачі, яка полягає у розробці комплексного рішення для автоматизованої аналітики телефонних дзвінків.

Отримані теоретичні положення першого розділу формують фундамент для подальших етапів розробки та деталізації архітектурних і технічних рішень інформаційної системи.

2 ЗАСОБИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ДЗВІНКІВ

2.1 Вибір програмно-алгоритмічної платформи для системи аналізу дзвінків

Вибір програмно-алгоритмічної платформи для розробки системи аналізу телефонних розмов виступає фундаментальним рішенням, що визначає:

– швидкість розробки;

– продуктивність;

– безпеку;

– потенціал архітектури до масштабування.

Використання мови Python спільно з фреймворком Django [22] був обраний як основна програмно-алгоритмічна платформа завдяки своїм перевагам, зокрема:

– інтеграція з телефоніями;

– CRM-системами;

– обробка звукових записів за допомогою Google Gemini API;

– генерація звітності для оптимізації корпоративних стратегій.

Python входить до переліку найбільш затребуваних інструментів розробки, що характеризується:

– простотою;

– читабельністю коду;

– широкою екосистемою бібліотек.

Для інформаційної системи, яка здійснюватиме дескриптивний аналіз аудіоданих та текстової інформації на базі Google Gemini API [23], Python гарантує безперешкодне поєднання компонентів через офіційні SDK, як от google-cloud-sdk. Це дає змогу ефективно обробляти відомості з API та впроваджувати логіку оцінювання стандартів ведення переговорів. Водночас Python спрощує взаємодію з хмарними сервісами, передусім Google Cloud, що є

критично важливим для нарощування потужностей та оперування великими за обсягом наборів та колекцій даних.

Django [22] визначено базовим інструментом розробки з огляду на його надійність, гнучкість та можливість інтенсифікувати створення архітектурно складних інформаційних систем та вебзастосунків.

– Фреймворк Django базується на концепції функціональної повноти, пропонуючи розробнику широкий набір інтегрованих інструментів:

– вбудовані інструменти для автентифікації;

– управління користувачами;

– роботи з базами даних;

– створення RESTful API.

Це дає змогу оперативно реалізувати процеси авторизації та керування повноваженнями користувачів, зокрема:

– адміністратор;

– quality-менеджер;

– керівник.

Вбудована система Django Authentication забезпечує криптографічно стійке хешування паролів та підтримує впровадження двофакторного підтвердження особи, що цілком відповідає актуальним вимогам безпеки в межах інформаційної системи.

Технологія об'єктно реляційного відображення Django ORM значно спрощує архітектурну взаємодію з базами даних, даючи змогу:

– ефективно зберігати метадані дзвінків;

– результати аналізу;

– звіти.

Фреймворк підтримує PostgreSQL [24], що є обґрунтованим рішенням для адміністрування великих за обсягом наборів та колекцій даних завдяки відмінним показникам швидкодії та розвиненим функціям пошуку за текстом. Використання цього інструментарію в межах інформаційної системи дає можливість:

– оперативно групувати записи за клієнтським номером;

– сортування за тривалістю;

– групування за ім’ям працівника, як того вимагають функціональні специфікації.

Взаємодія з телекомунікаційними сервісами, зокрема Asterisk та Twilio, а також системами управління клієнтськими відносинами, як от Salesforce або HubSpot, реалізується на базі Django REST Framework. Це спрощує розробку програмних інтерфейсів для передачі відомостей до сторонніх інформаційних систем. Вибрана методологія гарантує адаптивність та інтеграційну здатність, що є критичними для проєктованої інформаційної системи. З метою опрацювання масштабних черг звукових файлів Django залучає методи асинхронного виконання завдань за допомогою бібліотек, як от Celery [25], що дає змогу проводити аналіз через Google Gemini API з часовим лагом, який не перевищує тридцяти секунд для одного виклику. На рисунку 2.1 зображено схему архітектури асинхронної обробки задач у Django за допомогою Celery та Redis [26].

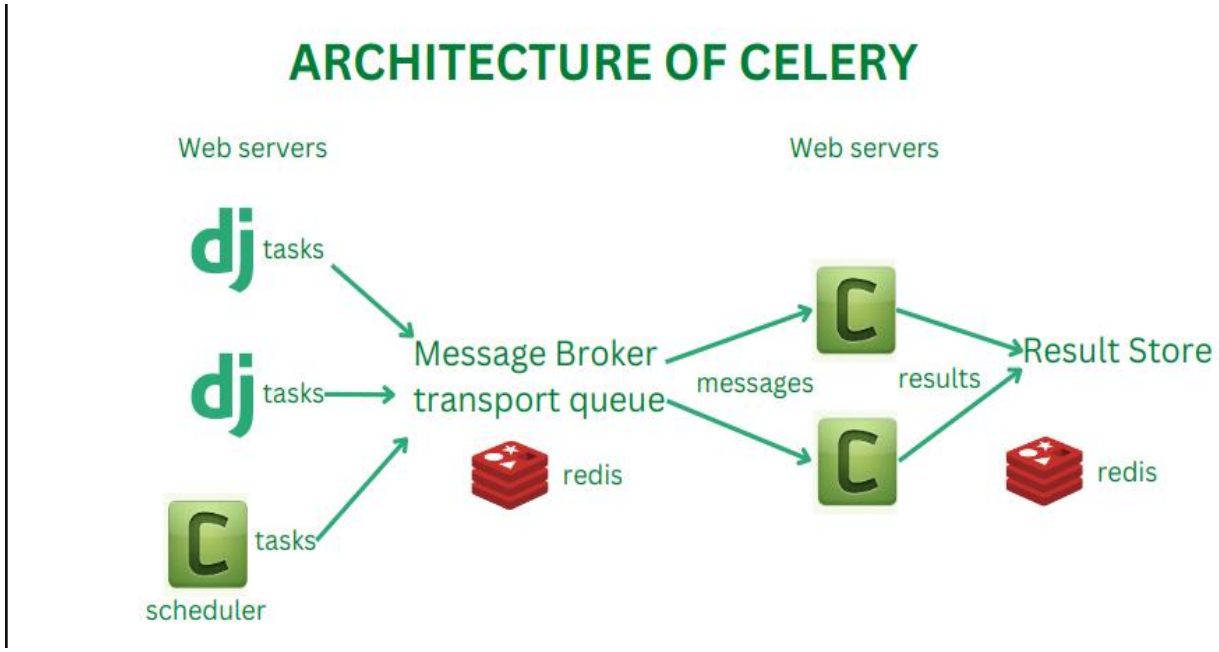

Рисунок 2.1 – Архітектура Celery з Django і Redis

Django вирізняється підвищеним ступенем захищеності. Фреймворк забезпечує надійне шифрування відомостей та відповідає нормам Загального

регламенту про захист даних GDPR, та захищає від поширених вразливостей, як от:

– SQL-ін’єкції;
– XSS-атаки;
– CSRF.

Django [22] спрощує конфігурування захищених з'єднань через HTTPS та TLS для безпечного передавання аудіофайлів і результатів їхнього аналітичного опрацювання, що є засадничим аспектом при роботі з приватними даними в межах інформаційної системи.

Окрім цього, фреймворк прискорює реалізацію проєкту завдяки використанню готових функціональних блоків, як от:

– адмін-панель;
– система шаблонів;
– інструменти для тестування.

Це дає змогу розробнику зосередити основні зусилля на взаємодії з Google Gemini API та побудові логіки бізнес-процесів, замість проєктування стандартних сервісних елементів. Оскільки для інформаційної системи для аналізу аудіо дзвінків є жорсткі вимоги до часу опрацювання, застосування Django дає змогу в стислі терміни підготувати робочий прототип і провести його апробацію на реальних масивах даних.

Зіставляючи Django з альтернативними рішеннями, як от Flask, Node.js на базі Express або Ruby on Rails, можна стверджувати, що цей фреймворк пропонує найбільш раціональне поєднання зручності розробки та широкого функціонального набору. На відміну від нього, Flask [27] вимагає значного обсягу ручного конфігурування для впровадження механізмів автентифікації та взаємодії з базами даних через об’єктно реляційне відображення, що часто призводить до затримок у реалізації проєкту. Платформа Node.js [28] демонструє високу ефективність в асинхронних середовищах, проте вона є менш адаптованою до прямої інтеграції з екосистемою Python та інструментарієм Google Gemini API. Своєю чергою, Ruby on Rails [29] має обмежену кількість

готових інструментів для взаємодії з хмарною інфраструктурою Google, що створює певні бар'єри при розробці.

Натомість Django характеризується глибокою інтеграцією з мовою Python, дозволяючи оперативно розгортати надійне програмне забезпечення без втрати стабільності. Певне підвищення споживання ресурсів, притаманне цьому фреймворку, успішно нівелюється завдяки можливостям хмарного масштабування. Для впровадження асинхронних обчислень застосовується бібліотека Celery [25], яка гарантує необхідні показники швидкодії.

Використання мови програмування Python та фреймворку Django є цілком виправданим, оскільки вони дають змогу інтенсифікувати процес розробки, забезпечують захищений зв'язок із Google Gemini API, а також підтримують масштабованість та стабільне керування даними. Це повністю задовольняє вимоги, що висуваються до інформаційної системи для аналітичного опрацювання телефонних дзвінків.

2.2 Django та Celery для асинхронної обробки ресурсомістких завдань опрацювання аудіо даних телефонних дзвінків

Комбінація фреймворку Django [30] та інструментарію Celery [25] для асинхронного виконання задач є найбільш раціональним програмно-алгоритмічним рішенням при побудові серверної логіки інформаційної системи, призначеної для аналітичного опрацювання телефонних дзвінків. Обрана архітектура забезпечує продуктивне проведення складних обчислювальних операцій, зокрема перетворення мовлення у текст, семантичне дослідження контенту та розпізнавання емоційного забарвлення через Google Gemini API, зберігаючи при цьому високу швидкість роботи та відгуку користувацького інтерфейсу.

Django, як базовий фундамент бекенд-складової інформаційної системи для аналітичного опрацювання телефонних дзвінків, надає:

– надійні інструменти для автентифікації;

– можливості синхронізації з телекомунікаційними шлюзами, наприклад Twilio;

– CRM-системами на кшталт Salesforce чи HubSpot.

Однак ресурсомісткі завдання, як от опрацювання значної кількості викликів протягом доби, здатні спричинити надмірне навантаження на головний потік обробки запитів, що зумовлює потребу у впровадженні Celery.

Celery [25] надає можливість здійснювати трудомісткі обчислювальні операції у фоновому режимі, залучаючи механізми черг та розподілені "процеси".

Під час створення інформаційної системи для аналітичного опрацювання телефонних дзвінків на Celery покладається реалізація наступних функцій:

– транскрибування аудіо в текст;

– виявлення проблемних моментів;

– аналіз емоційного забарвлення діалогу.

Задачі описуються як функції мови Python із застосуванням декоратора «@shared_task» та ініціюються в асинхронному режимі з боку Django [30]. Зокрема, аналіз звукового файлу за допомогою Google Gemini API розробляється як окреме Celery-задання, що дає змогу оперативно сповістити клієнта про зарахування запиту до черги без призупинення роботи вебінтерфейсу. Отримані дані фіксуються в бекенд-сховищі, Redis або безпосередньо у базі даних, що дає змогу користувачеві моніторити етап виконання операції через програмний інтерфейс API.

Для стабільного функціонування Celery задіюється брокер повідомлень, як от RabbitMQ [31], що здійснює менеджмент черги задач та зберігає звіти про їх реалізацію. Така архітектурна побудова гарантує масштабованість інформаційної системи, даючи змогу розгортати процеси у розподілених інфраструктурах, зокрема на базі Amazon Web Services Elastic Container Service або Kubernetes [32]. Це дає змогу виконувати нефункціональну вимогу щодо опрацювання до десяти тисяч викликів щодоби. Завдяки паралельному виконанню обчислень аналіз одного запису тривалістю до п’яти хвилин не перевищує тридцяти секунд. Фреймворк Django комунікує з Celery за допомогою

однойменної бібліотеки, що надає інструментарій для моніторингу статусу операцій через AsyncResult [33].

Належний рівень захищеності в межах інформаційної системи досягається через конфігурування брокера із залученням Transport Layer Security та процедур ідентифікації. Одночасно з цим, вбудовані механізми Django запобігають виникненню вразливостей, як от Cross-Site Request Forgery та SQL-ін’єкції, що цілком узгоджується з положеннями Загального регламенту захисту даних GDPR [34].

Додатково Celery підтримує алгоритми обробки виняткових ситуацій, наприклад, при виявленні пошкоджених аудіофайлів, шляхом логування інцидентів та сповіщення користувачів, що підвищує відмовостійкість. Синергія Django та Celery [25] забезпечує раціональний розподіл обчислювальних потужностей, високу швидкість роботи та надійність інформаційної системи. Це створює фундамент для подальшого розширення можливостей, передусім для впровадження інтелектуальної аналітики чи функціонування в режимі реального часу.

Інформаційна система, що створена на Django [30] складається з однієї або декількох структурно автономних сегментів, які доцільно проектувати у вигляді окремих модулів. Дана модульна організація є фундаментальною архітектурною особливістю Django, що забезпечує йому перевагу над іншими програмними середовищами, зокрема Ruby on Rails.

Архітектурна побудова Django базується на концепції Модель-Вигляд-Контролер, проте адаптує її під власну термінологію. У цьому фреймворку за логіку контролера відповідає «вигляд» (view), а рівень візуалізації реалізується через «шаблон» (template). Таким чином, функціонування інформаційної системи описується патерном MTV – Модель-Шаблон-Вигляд.

На рисунку 2.2 подано структурну схему, яка пояснює взаємодію між цими компонентами.

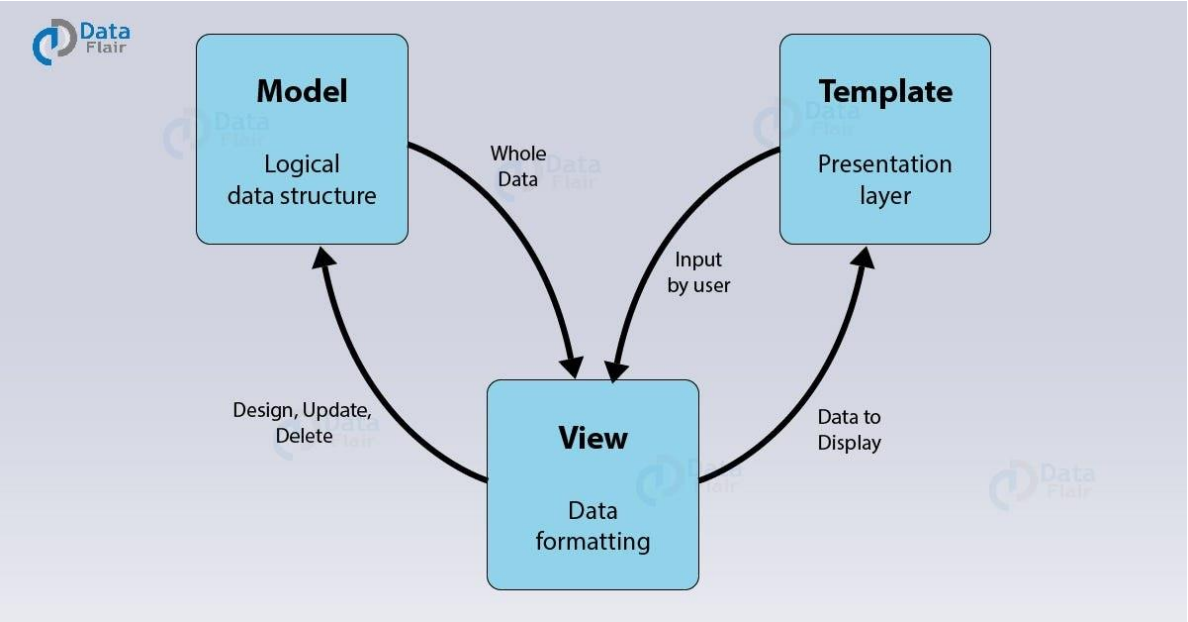

Рисунок 2.2 –Архітектура Django за моделлю MTV

У Django View виконує функції контролера, тоді як Template відповідає за формування та відображення інтерфейсу користувача. Внаслідок цього архітектурного підходу Django зберігає логіку класичної MVC-моделі, проте пропонує структурованіше та зрозуміліше розмежування обов'язків між компонентами.

2.3 Контейнеризація сервісів за допомогою Docker для потокового аналітичного опрацювання аудіо дзвінків

Контейнеризація за допомогою Docker [35] виступає фундаментальним методом впровадження бекенд-складової інформаційної системи, призначеної для оцінки якості телефонних комунікацій. Використання цієї технології забезпечує високу мобільність коду, спрощує масштабування та оптимізує адміністрування інфраструктури. Завдяки інкапсуляції застосунку разом із його залежностями в ізольовані контейнери, гарантується ідентичність роботи інформаційної системи в будь-якому оточенні – від етапу розроблення до серверів промислової експлуатації. Такий підхід повністю задовольняє нефункціональні вимоги до інформаційної системи, зокрема здатність

опрацьовувати до десяти тисяч дзвінків щодоби, забезпечуючи при цьому надійність і портативність серверних компонентів.

Під час створення інформаційної системи для аналітичного опрацювання телефонних дзвінків Docker використовується для пакування у окремі контейнери:

– Django-застосунку;
– Celery-процесів;
– брокера повідомлень, зокрема Redis або RabbitMQ;
– бази даних, зокрема PostgreSQL.

Кожна ізольована одиниця містить виключно ті програмні залежності, що необхідні для стабільного функціонування конкретного модуля. Це дає змогу нівелювати ризик виникнення системних конфліктів та полегшує процедуру модернізації окремих частин інформаційної системи. Зокрема, Django-застосунок разом із бібліотеками для комунікації з Google Gemini API інтегрується в образ, що охоплює середовище виконання Python, необхідні пакети та конфігурацію WSGI-сервера Gunicorn. Водночас Celery-процеси, які обробляють ресурсомісткі завдання, як от перетворення мовлення в текст або аналітичне дослідження текстових масивів, функціонують у власних контейнерах. Це забезпечує можливість їхнього автономного масштабування незалежно від базових модулів інформаційної системи.

Для координації контейнерів на етапах розробки та тестування застосовується Docker Compose [35], що дає змогу інтегрувати всі елементи інформаційної системи, зокрема Django, Celery, Redis та PostgreSQL, у єдиному локальному середовищі. У промислових масштабах управління здійснюється за допомогою Kubernetes або Amazon Web Services Elastic Container Service [36]. Це забезпечує автоматизоване масштабування та адміністрування контейнерів у розподіленій інфраструктурі, що гарантує стабільну роботу на рівні 99,9% та ефективну реакцію на пікові навантаження, задовольняючи високим вимогам продуктивності.

Docker [35] сприяє спрощенню розгортання інформаційної системи для аналітичного опрацювання телефонних дзвінків, даючи змогу оперативно

переміщувати програмне забезпечення між різними хмарними провайдерами, як от Amazon Web Services, Google Cloud або локальними серверами, без необхідності коригування коду.

Контейнери водночас підвищують безпеку, ізолюючи компоненти інформаційної системи та дають можливість застосовувати обмеження на ресурси, зокрема CPU чи оперативну пам’ять, що запобігає системним збоям. Наприклад, для Celery-обробників можна встановити чіткі ліміти споживання пам’яті, щоб уникнути дефіциту ресурсів для інших вузлів під час аналізу значних за розміром аудіофайлів.

Для забезпечення стабільної роботи архітектури застосовується технологія контейнеризації, де кожен компонент має власний сценарій Dockerfile [37], а загальна логіка їхньої взаємодії регламентується через конфігураційний файл «docker-compose.yml». Зокрема, Dockerfile для компонента Django, що є частиною інформаційної системи для аналітичного опрацювання телефонних дзвінків, забезпечує повний цикл розгортання: від підготовки Python-середовища та перенесення вихідного коду до встановлення бібліотек за допомогою менеджера пакетів pip і налаштування вебсервера Gunicorn. Використання інструментарію Docker Compose дає змогу автоматизувати створення ізольованих мереж та томів, необхідних для ефективної комунікації контейнерів інформаційної системи для аналітичного опрацювання телефонних дзвінків із базою даних PostgreSQL. Такий підхід гарантує високий рівень відтворюваності середовища, що є критично важливим під час тестування інтеграцій інформаційної системи для аналітичного опрацювання телефонних дзвінків із зовнішніми сервісами телефонії, як от Twilio [38] та Asterisk [39], а також у процесі взаємодії з CRM-системами, зокрема Salesforce.

Впровадження Docker суттєво оптимізує методології Continuous Integration та Continuous Delivery – безперервної інтеграції та безперервної доставки. Це дає змогу автоматизувати цикли збірки, апробації та впровадження інформаційної системи для аналітичного опрацювання телефонних дзвінків за допомогою таких інструментів, як GitHub Actions або Jenkins. Будь-яка оновлена версія інформаційної системи для аналізу телефонних дзвінків інкапсулюється в

окремий Docker-образ, який проходить ретельну перевірку в ізольованому середовищі. Такий підхід мінімізує ризики виникнення збоїв у промисловій експлуатації. Крім того, контейнеризація спрощує керування залежностями, наприклад, оновлення бібліотек для Google Gemini API, не створюючи конфліктів для інших модулів інформаційної системи для аналітичного опрацювання телефонних дзвінків.

Використання Docker гарантує високий рівень портативності, можливості масштабування та захищеності серверних компонентів інформаційної системи для аналітичного опрацювання телефонних дзвінків. Вона дає змогу раціонально розподіляти обчислювальні ресурси, спрощує інтеграцію у хмарну інфраструктуру та забезпечує стабільний фундамент для роботи архітектури, що обробляє великі за обсягом набори та колекції даних і відповідає вимогам."

2.4 Структура сховища даних для зберігання інформації щодо телефонних дзвінків та клієнтів

Для створення інформаційної системи для аналітичного опрацювання телефонних дзвінків застосуємо PostgreSQL [40], яка використовуватиме декілька реляційних таблиць, що взаємодіють між собою через класичні типи зв’язків: «один до багатьох», «один до одного» та «багато до багатьох». На рисунку 2.3 подано структуру таблиць сховища даних інформаційної системи для аналітичного опрацювання телефонних дзвінків та візуалізовано архітектуру їхніх взаємозв’язків.

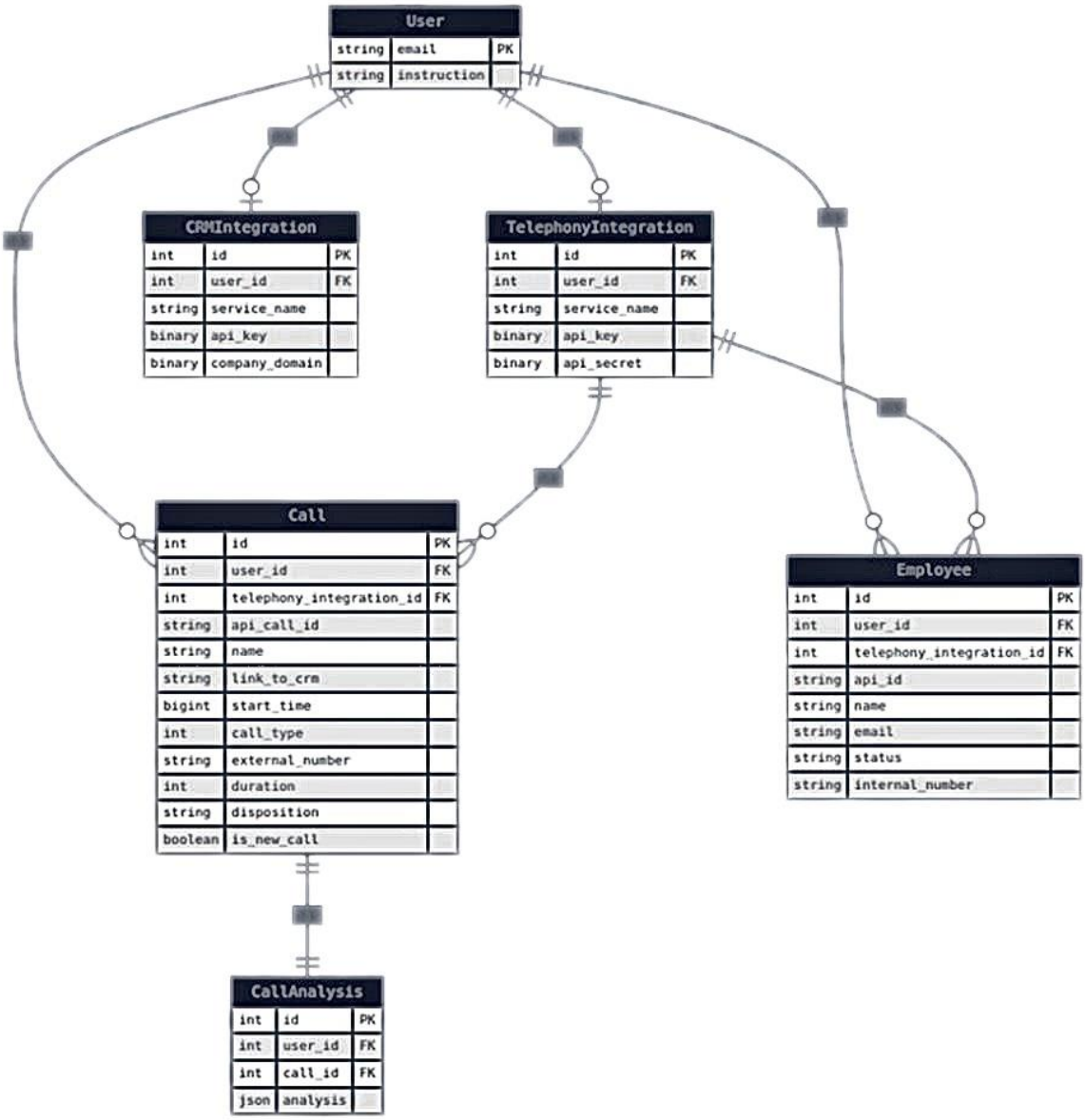

Рисунок 2.3 – Структура таблиць сховища даних інформаційної системи для аналітичного опрацювання телефонних дзвінків

Варто зауважити, що більшість таблиць сховища даних інформаційної системи для аналітичного опрацювання телефонних дзвінків мають логічне поєднання з однією або декількома іншими структурами через спільні атрибути. Ці атрибути виконують функцію унікальних ключів для ідентифікації кожного об’єкта, наприклад, call_id. Більш розгорнуті характеристики цих даних подані в таблиці 2.1.

Таблиця 2.1– Відношення між таблицями сховища даних інформаційної системи для аналітичного опрацювання телефонних дзвінків

Таблиця	**Дані**	**Зв'язки з іншими таблицями**
User	Записи, email address, instruction	TelephonyIntegration через user_id CRMIntegration через user_id Call через user_id CallAnalysis через user_id Employee через user_id
TelephonyIntegration	Service name, api_key, api_secret	User через user_id Call через telephony_integration_id Employee через telephony_integration_id
CRMIntegration	Service name, api_key, company_domain	User через user_id
Call	API call ID, name, link_to_crm, start_time, call_type, numbers, duration, disposition	User через user_id TelephonyIntegration через telephony_integration_id CallAnalysis через call_id
CallAnalysis	Метадані аналізу, JSON data	User через user_id Call через call_id
Employee	API ID, name, email, status, internal_number	User через user_id TelephonyIntegration через telephony_integration_id

Структура сховища даних забезпечує ефективне зберігання та організацію інформації про:

– телефонні дзвінки;

– користувачів системи;

– компанії;

– інтеграційні сервіси.

Використання різних типів відношень між таблицями:

– дає можливість реалізувати гнучкі запити для отримання даних;

– забезпечує цілісність інформації;

– зменшує дублювання записів.

Також слід зазначити, що така структура сховища даних спрощує масштабування інформаційної системи для аналітичного опрацювання телефонних дзвінків та інтеграцію нових модулів у майбутньому, наприклад для розширеної аналітики, статистики або автоматичного формування звітів. Всі відношення й обмеження на рівні сховища даних гарантують коректність введених даних і їх відповідність бізнес-логіці інформаційної системи.

2.5 Висновок до другого розділу

В другому розділі кваліфікаційної роботи досліджено засоби та інструменти, необхідні для створення інформаційної системи аналітичного опрацювання телефонних дзвінків. Обґрунтовано вибір фреймворку Django як основи для серверної частини інформаційної системи, що забезпечує гнучкість, швидкість розробки та високий рівень безпеки. Розглянуто інтеграцію Django з Celery та Celery Beat для виконання ресурсоємних, асинхронних та періодичних завдань, що є критично важливим для обробки великих масивів даних і забезпечення стабільності роботи інформаційної системи [22].

Також у розділі проаналізовано переваги використання Docker для контейнеризації компонентів системи, що дає змогу забезпечити ізольованість сервісів, просте масштабування та переносимість інформаційної системи.

Окрему увагу приділено розробці структури сховища даних на основі PostgreSQL та Redis, яке формує основу для ефективного зберігання, обробки та кешування аналітичної інформації.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ТЕЛЕФОННИХ ДЗВІНКІВ

3.1 Принципи та підходи до обробки телефонних дзвінків

Розробка серверної частини інформаційної системи [41] для аналітичного опрацювання телефонних дзвінків має на меті реалізацію програмного продукту, який:

– автоматизує обробку аудіозаписів;

– їхній аналіз;

– підготовка аналітичних звітів для компаній, що займаються продажами.

Інформаційна система [41] зменшує застосування ручного аналізу даних, що виконується quality-менеджерами. Це дає можливість краще оцінити комунікації та надати якісні рекомендації для покращення бізнес-процесів. Серверна частина інформаційної системи для аналітичного опрацювання телефонних дзвінків розробляється з використанням:

– Django REST Framework.

– PostgreSQL.

– Celery.

– Celery Beat.

– Docker.

З інтеграцією:

– KeepinCRM.

– Pipedrive CRM.

– Binotel.

– Google Gemini API.

В першу чергу, для розробки інформаційної системи для аналітичного опрацювання телефонних дзвінків обрано мікросервісну архітектуру, що інтегрує декілька компонентів. Зокрема Django REST Framework [42] відповідає за:

– побудова API-інтерфейсу для перевірки користувачів;

– управління інтеграціями;

– сортування розмов;

– надання звітів.

PostgreSQL [40] використовується як основа сховища даних для зберігання:

– метаданих дзвінків, зокрема номерів телефонів, тривалості розмов, імен працівників;

– транскрибованих текстів;

– результатів аналізу.

Celery [25] обробляє ресурсомісткі завдання, як от аналіз масивів даних та перетворення аудіо у текст за допомогою Google Gemini API. Синхронізацію процесів у часі, зокрема регулярний імпорт свіжих даних із Binotel, покладено на Celery Beat. Стабільність роботи всієї інфраструктури – від бази даних PostgreSQL та кешу Redis до прикладного рівня на Django [42] – забезпечується шляхом їх ізоляції в Docker-контейнерах. Такий підхід гарантує:

– ізоляцію;

– портативність;

– оперативне масштабування.

Інформаційна система інтегрується з Binotel [43] шляхом взаємодії з REST API платформи для вилучення аудіофайлів та супровідної інформації про виклики. На базі Django [42] реалізовано механізми перевірки прав доступу, зокрема робота з API-токенами, а завдяки планувальнику Celery Beat ініціюються регулярні запити на отримання свіжих даних для подальшої обробки в Celery. Взаємодія інформаційної системи для аналітичного опрацювання телефонних дзвінків із CRM-системами KeepinCRM та Pipedrive базується на використанні їхніх програмних інтерфейсів, даючи можливість передавати результати аналізу, наприклад звіти про оцінку якості чи виявлені недоліки, безпосередньо в картки клієнтів. Спеціальні точки доступу endpoints, розроблені на Django REST Framework [42], дають змогу конфігурувати параметри інформаційної системи для аналітичного опрацювання телефонних дзвінків.

Зокрема, система верифікує введені користувачем API-ключі та перевіряє стабільність з’єднання перед записом налаштувань у базу PostgreSQL [40].

Алгоритм дослідження діалогів охоплює кілька кроків. Файл із записом голосу, завантажений із Binotel, надходить у чергу Celery, де через звернення до Google Gemini API [23] виконується його розпізнавання та перетворення у текстовий формат. Готова транскрипція фіксується в PostgreSQL, після чого Gemini API проводить її поглиблений аудит: зіставляє розмову зі скриптами продажів, ідентифікує специфічні терміни та фіксує критичні точки, зокрема клієнтські заперечення. Крім того, ШІ-інструментарій досліджує акустичні параметри для визначення емоційного фону учасників та рівня компетентності персоналу. Підсумкові дані формуються у звіти, що включають якісні показники, статистичні виклади й поради, та транслюються через API як:

– таблиць;
– графіків;
– експортованих файлів, зокрема PDF, CSV.

Функціональні вимоги до інформаційної системи для аналітичного опрацювання телефонних дзвінків розроблені з використанням фреймворку Django REST [42]. Користувачі мають можливість для реєстрації та підключення до системи, як:

– адміністратор;
– quality-менеджер.

Водночас користувачі інформаційної системи можуть здійснювати конфігурацію зв’язків із Pipedrive, KeepinCRM [44] та Binotel, використовуючи для цього безпечні точки доступу. За швидке сортування масивів дзвінків за часом, тривалістю, ідентифікатором менеджера або номером абонента відповідає Django ORM, що формує оптимізовані звернення до бази PostgreSQL [40]. Застосування Celery для фонового створення звітів дає змогу системі опрацьовувати складні аналітичні запити без втрати швидкодії інтерфейсу. Регулярність процесів, як от добовий імпорт нових аудіофайлів із Binotel, гарантується планувальником Celery Beat [45].

Системна архітектура враховує критичні нефункціональні показники, зокрема захищеність та ефективність. Інформаційна система для аналітичного опрацювання телефонних дзвінків має можливість обробляти до десяти тисяч дзвінків щодня, з часом аналізу одного п’ятихвилинного запису до тридцяти секунд. Таких результатів досягнуто завдяки паралелізації потоків у Celery та використанню Docker для гнучкого масштабування ресурсів. Рівень безпеки підкріплений впровадженням протоколів HTTPS [46] та алгоритмів AES-256, суворим дотриманням регламентів GDPR та використанням токенів для авторизації API-запитів. Для захисту черг повідомлень у брокері Redis передбачено шифрування через TLS. Крім того, інформаційна система для аналітичного опрацювання телефонних дзвінків демонструє стійкість до низької якості звуку: інструменти Gemini API допомагають нівелювати сторонні шуми, а вбудовані механізми обробки винятків забезпечують логування збоїв та оперативне сповіщення клієнтів.

Процес створення інформаційної системи для аналітичного опрацювання телефонних дзвінків потребує поетапної реалізації:

– створення Django API [42] з автентифікацією та інтеграціями;
– зміна параметрів Celery і Celery Beat [45] для обробки завдань;
– інтеграція з Gemini API;
– контейнеризація через Docker.

Тестування інформаційної системи для аналітичного опрацювання телефонних дзвінків проводилось на користувацьких даних із Binotel, і включало:

– продуктивність;
– точність аналізу;
– стабільність інтеграцій.

Застосування Docker Compose [30] дає змогу організувати ізольоване середовище для попередніх перевірок та налагодження коду. Натомість, у промисловій експлуатації інформаційна система для аналітичного опрацювання телефонних дзвінків розгортається на хмарній платформі, зокрема в інфраструктурі Google Cloud або AWS, де процеси управління контейнерами та

їх масштабування покладено на Kubernetes [32]. Стратегія подальшого розвитку продукту передбачає інтеграцію модулів для прогнозування трендів, а також перехід до опрацювання даних в режимі реального часу, що підвищить цінність проєктованої інформаційної системи для бізнесу.

3.2 Архітектура серверної частини системи аналітики дзвінків

Архітектура бекенду інформаційної системи для аналітичного опрацювання телефонних дзвінків вимагає врахування наступних особливостей:

– масштабованості;

– продуктивності;

– безпеки;

– інтеграції з зовнішніми системами, як от Binotel, KeepinCRM, Pipedrive CRM [47] і Google Gemini API [23].

Бекенд інформаційної системи для аналітичного опрацювання телефонних дзвінків має в основі мікросервісну складову і базується на Django REST Framework, PostgreSQL, Celery, Celery Beat, Redis і Docker [42], що забезпечує:

– ефективну обробку ресурсомістких завдань;

– асинхронну роботу;

– оперативне розгортання.

Основні компоненти архітектури:

– Django REST Framework [42] – фундамент серверної частини архітектури. Саме цей інструмент виступає основою інформаційної системи для аналітичного опрацювання телефонних дзвінків, беручи на себе маршрутизацію HTTP-запитів, контроль доступу та логіку API. Засобами фреймворку реалізовано ключовий функціонал: від процедур реєстрації та авторизації до механізмів підключення зовнішніх сервісів, як от Binotel, Pipedrive, KeepinCRM. Крім того, він відповідає за генерацію звітів та гнучке впорядкування масивів розмов за різними критеріями: часом, тривалістю або відповідальним менеджером. Взаємодія з базою даних оптимізована через вбудований ORM, що гарантує високу швидкість вибірки інформації. Захист API базується на JWT-

токенах із чітким розмежуванням ролей, наприклад, quality-менеджер чи адміністратор. Для розгортання використовується зв’язка Gunicorn [24] у ролі WSGI-сервера та Nginx для обробки статичного контенту й проксіювання запитів у продуктивному середовищі.

– База даних PostgreSQL [40]. Ця система керування базами даних слугує центральним сховищем усіх відомостей: від метаданих дзвінків до текстових розшифровок та аналітичних висновків, а також параметрів конфігурації. Застосування індексів у поєднанні з ефективними запитами Django ORM дає змогу швидко фільтрувати та сортувати масштабні за обсягом наборів та колекцій даних, забезпечуючи стабільну обробку тисяч дзвінків щодоби. Висока надійність зберігання досягається завдяки підтримці транзакцій, а спеціалізовані розширення PostgreSQL значно спрощують повнотекстовий пошук по транскрибаціях.

– – Celery [25] – система керування розподіленими чергами. На цей інструмент покладено виконання найбільш трудомістких операцій, як от перетворення голосу в текст та інтелектуальний аналіз контенту за допомогою Google Gemini API. Технічна реалізація передбачає, що кожна окрема описується як Python-функція зі спеціальним декоратором «@shared_task» і запускається ізольовано у відповідних процесах. Celery дає змогу ефективно розпаралелювати потоки даних, охоплюючи тисячі викликів одночасно. Завдяки цьому досягається висока швидкість, що скорочує час дослідження п’ятихвилинного аудіо запису до тридцяти секунд. Процеси підтримують горизонтальне масштабування шляхом розгортання додаткових контейнерів, повністю задовольняючи критерії швидкодії системи.

– Celery Beat [25] – планувальник циклічних операцій. Цей компонент відповідає за автоматизацію регулярних дій, зокрема систематичний імпорт нових файлів із платформи Binotel. Працює це наступним чином: за розкладом, наприклад, щогодини, ініціюється звернення до зовнішнього API, завантажуються свіжі записи розмов і перенаправляються в чергу виконання Celery. Такий механізм гарантує постійну актуалізацію бази даних без

необхідності ручного керування, виступаючи ключовим елементом інтеграції інформаційної системи для аналітичного опрацювання телефонних дзвінків;

– Redis [26] – інструмент для маршрутизації повідомлень та збереження станів Celery. Він виступає посередником у чергах завдань і слугує бекендом для фіксації підсумків їхнього виконання. Завдяки винятковим швидкісним характеристикам та підтримці процесів асинхронного обміну даними дають змогу продуктивно оперувати великою кількістю фонових операцій. Для відповідності стандартам GDPR та внутрішнім протоколам безпеки, доступ до Redis захищається через механізми автентифікації та шифрування трафіку за допомогою TLS.

– Google Gemini API [23] – сервіс для інтелектуальної обробки медіа та текстових масивів. Цей зовнішній ресурс залучається для розпізнавання мовлення в аудіофайлах та подальшої змістовної інтерпретації отриманих розшифровок. У межах роботи Celery-задач записи спрямовуються до API, де після транскрибування ініціюється серія аналітичних перевірок: контроль дотримання мовних скриптів, пошук термінів-маркерів, фіксація заперечень, а також аудит акустичних параметрів – визначення емоційного фону та рівня кваліфікації менеджера. Опрацьована інформація передається назад у Django для архівації в базі PostgreSQL та візуалізації у формі аналітичних звітів.

– Docker – контейнеризація [35]. Усі компоненти, як от Django, Celery, Celery Beat, PostgreSQL, Redis, створюються окремо в контейнерах. Такий підхід встановлює:

1. Ізоляцію.
2. Портативність.
3. Оперативне масштабування.

– Завдяки Docker Compose [48] реалізується ізоляція процесів під час тестування. Інструмент координує мережеві зв'язки між контейнерами та керує Persistent Volumes, що необхідно для персистентності даних у PostgreSQL. У промисловому середовищі експлуатації інформаційної системи для аналітичного опрацювання телефонних дзвінків використання інструментів оркестрації, як от AWS ECS або Kubernetes, дає можливість автоматично масштабувати процеси

серверної частини на Django та обробників Celery відповідно до поточного рівня навантаження. Це дає змогу підтримувати високу відмовостійкість і гарантувати рівень доступності сервісу на позначці 99,9%.

Схема передачі даних:

– отримання даних: Celery Beat [25] за розкладом, зокрема щогодини, ініціює через Django [43] звернення до Binotel API. Отримані відомості про дзвінки – тривалість, ідентифікатор оператора, номер та аудіофайли фіксуються в PostgreSQL;

– обробка звуку: через Django запускається Celery-завдання, що спрямовує аудіо до Google Gemini API для перетворення в текст. Результат розшифровки повертається та архівується в PostgreSQL [40];

– аналітичний етап: Celery ініціює вторинні запити до Gemini API [23] для перевірки діалогів на дотримання скриптів, виявлення конфліктів, пошуку тегів та аудиту емоційного стану. Дані зберігаються в БД;

– підготовка звітності: за зверненням клієнта Django формує звіти на основі агрегації даних із PostgreSQL. Використання Celery [25] дає змогу у фоновому режимі створювати складну аналітику у форматах PDF, CSV або як візуальні графіки;

– інтеграція інформаційної системи для аналітичного опрацювання телефонних дзвінків з CRM: через програмні інтерфейси KeepinCRM [44] та Pipedrive підсумки аналізу синхронізуються з картками замовників. Налаштування цих зв’язків реалізовано в Django через захищені точки доступу;

– безпека забезпечується шифруванням даних за допомогою алгоритмів AES-256 та протоколу HTTPS. Захист доповнюється використанням JWT для API, шифруванням каналів Redis через TLS та дотриманням стандартів GDPR.

Безпека платформи гарантується засобами Django, які нівелюють поширені загрози, як от SQL-ін’єкції [49] та CSRF [50]. Висока продуктивність забезпечується комбінацією асинхронних задач у Celery, ефективних запитів до PostgreSQL та можливостей Docker. Система здатна опрацьовувати навіть аудіо низької якості завдяки шумопоглинанню через Gemini API, а прозорість роботи підтримується детальним логуванням помилок.

Ключові переваги обраної архітектури:

– Можливість розширення: Масштабованість: Celery і Docker [48] дають можливість додавати процеси та нові контейнери, що дає змогу витримувати пікові навантаження інформаційної системи для аналітичного опрацювання телефонних дзвінків.

– Гнучкість: мікросервісний підхід значно спрощує оновлення окремих модулів та підключення нових сервісів.

– Відмовостійкість: ізольоване середовище контейнерів і надійність транзакцій БД суттєво знижують ризик збоїв інформаційної системи для аналітичного опрацювання телефонних дзвінків.

– Темпи розробки: використання Django REST Framework [45] та наявних бібліотек дає змогу швидко реалізувати API та інтеграції інформаційної системи для аналітичного опрацювання телефонних дзвінків.

Такий підхід покриває як функціональні, так і нефункціональні вимоги, створюючи надійний фундамент із перспективою подальшого розвитку, зокрема впровадження прогнозної аналітики.

3.3 Налаштування Docker контейнерів для функціонування інформаційної системи аналітичного опрацювання телефонних дзвінків

Для Dockerfile [37] застосовано підхід багатоетапної збірки, що дає змогу зменшити розмір образу та підвищити рівень безпеки. На початковій фазі розгортання інформаційної системи для аналітичного опрацювання телефонних дзвінків, використовується базовий імідж python:3.12-slim. У робочому каталозі /qualityai встановлюються параметри

```
PYTHONDONTWRITEBYTECODE=1
PYTHONUNBUFFERED=1
```

що дає змогу уникнути створення зайвих файлів байт-коду та гарантує потік журналювання в режимі реального часу.

Інсталяція залежностей із Pipfile та Pipfile.lock відбувається за допомогою pipenv з прапорцями --deploy --system та --no-cache-dir, що гарантує ідентичність середовищ.

На завершальному етапі розгортання інформаційної системи для аналітичного опрацювання телефонних дзвінків у фінальний образ переносяться необхідні бібліотеки. Додатково інсталюється libmagic1, а для запуску коду інформаційної системи для аналітичного опрацювання телефонних дзвінків створюється окремий користувач із обмеженими правами. Стартовою точкою слугує скрипт entrypoint.sh, який відкриває порт 8000 для Django -сервера [30].

Оркестрація через docker-compose.yml об’єднує такі сервіси:

– db (PostgreSQL);
– redis;
– api – Django REST Framework;
– celery – процеси для асинхронної обробки;
– celery-beat – планувальник задач.

Сервіс бази даних працює на образі інформаційної системи для аналітичного опрацювання телефонних дзвінків postgres:17, зберігаючи інформацію в томі postgres_data та отримуючи конфігурацію з env/.env. У ролі брокера повідомлень виступає redis на базі Alpine-образу інформаційної системи для аналітичного опрацювання телефонних дзвінків. API-сервіс розгортається через Dockerfile [37], відкриває порт 8000 і запускає Django [30] після виконання міграцій.

Воркери та планувальник також створюються на основі Dockerfile інформаційної системи для аналітичного опрацювання телефонних дзвінків: celery виконує транскрибацію та аналіз через Gemini API [51], а beat – імпорт даних із Binotel [43]. Усі сервіси інформаційної системи для аналітичного опрацювання телефонних дзвінків мають налаштування restart: always для автоматичного відновлення.

Така архітектура інформаційної системи для аналітичного опрацювання телефонних дзвінків гарантує ізоляцію компонентів, що спрощує масштабування. Локально використовується Docker Compose [30], а у

продакшені інформаційна система може керуватися через Kubernetes або AWS ECS. Використання TLS і некритичного користувача забезпечує безпеку, а контейнеризація прискорює CI/CD та оновлення інформаційної системи для аналітичного опрацювання телефонних дзвінків, підтримуючи високу продуктивність при обробці даних.

У лістингу 3.1 наведено конфігураційний файл Docker Compose, у якому визначено перелік контейнерів, що підлягають запуску в межах системи.

Лістинг 3.1 – Докер контейнери інформаційної системи для аналітичного опрацювання телефонних дзвінків

```
services:
  db: <4 keys>
  redis: <1 key>
  api: <7 keys>
  celery: <4 keys>
  celery-beat: <4 keys>
volumes: postgres_data
```

В лістингу 3.2 подано dockerfile для застосунку Django.

Лістинг 3.2 – Dockerfile для Django-застосунку інформаційної системи для аналітичного опрацювання телефонних дзвінків

```
FROM python:3.12-slim AS builder ...
FROM python:3.12-slim
RUN apt-get update && apt-get install -y libmagic1
RUN useradd -m -r user && mkdir /qualityai && chown -R user
/qualityai ...
COPY    --from=builder    /usr/local/lib/python3.12/site-packages/
/usr/local/lib/python3.12/site-packages/
COPY --from=builder /usr/local/bin/ /usr/local/bin/

WORKDIR /qualityai

COPY --chown=user:user . .

RUN chmod +x /qualityai/entrypoint.sh
RUN chown user:user /qualityai/entrypoint.sh

ENV PYTHONDONWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
USER user
EXPOSE 8000
```

3.4 Інтеграція інформаційної системи з Binotel для збору та обробки дзвінків

Інтеграція з АТС Binotel [43] є фундаментом бекенду, що дає змогу автоматично отримувати аудіозаписи та метадані дзвінків – тривалість, номери, відповідальних менеджерів. Технічна реалізація базується на стеку Django REST Framework [30], PostgreSQL [24] та Docker. Для авторизації адміністратор вносить облікові дані – API Key, Secret, які зберігаються в env/.env та валідуються системою.

Оновлення даних в режимі реального часу забезпечується через механізм вебхуків, активований підтримкою Binotel [43]. Ресурсомісткі операції, як от завантаження файлів та їх транскрибація через Google Gemini API, делеговані асинхронним процесам Celery. Система зберігає аудіо в тимчасове сховище, проводить аналіз – ключові слова, тональність, та записує результати в БД. Використання Redis [26] гарантує високу продуктивність, скорочуючи час обробки п’ятихвилинного запису до тридцяти секунд.

Фундамент захищеності інтеграції базується на суворих протоколах шифрування та безпечного зберігання конфіденційної інформації. Облікові дані, як от API Key [52] та Secret, не зберігаються у відкритому вигляді, а управляються через бібліотеку python-decouple, тоді як уся комунікація з серверами Binotel відбувається через захищений протокол HTTPS [46]. Для захисту від підробки запитів, вхідні вебхуки проходять обов’язкову валідацію цифрового підпису. Система обробки помилок побудована так, щоб мінімізувати втрату даних: у разі збоїв, наприклад, при недоступності аудіофайлу, Celery фіксує деталі інциденту в Redis [25] та через Django API сповіщає адміністратора. Архітектурна стійкість інформаційної системи для аналітичного опрацювання телефонних дзвінків досягається завдяки чіткій ізоляції контейнерів Django, Redis [25], Celery та використанню транзакційної моделі PostgreSQL, що гарантує цілісність записів навіть у випадку аварійного завершення процесів.

Результати аналітики автоматично експортуються до зовнішніх систем, як от KeepinCRM [44] та Pipedrive [47]. Цей процес реалізовано асинхронно через

Celery, що дає змогу оновлювати клієнтські профілі без затримок основного інтерфейсу. Зокрема, до KeepinCRM додається запис в історію комунікацій із оцінкою якості розмови, а в картки Pipedrive [47] вносяться відповідні коментарі. Перед початком роботи спеціальні Django-ендпоінти перевіряють валідність ключів доступу до CRM. Поточна архітектура демонструє високу продуктивність, успішно опрацьовуючи до сорока великих за обсягом наборів та колекцій даних завдяки оптимізації Docker та Celery. Це створює надійну базу для майбутніх вдосконалень, включаючи впровадження прогнозної аналітики та моніторингу в режимі реального часу.

Отримання інформації про дзвінки реалізовано за допомогою двох стратегій: регулярного опитування API та використання вебхуків. Для періодичної синхронізації планувальник Celery Beat [25] щогодини ініціює задачу, яка звертається до ресурсу /calls/list системи Binotel, використовуючи бібліотеку requests, API Key та Secret. У відповідь надходить JSON-структура зі списком розмов та посиланнями на записи. Технічні параметри, такі як «call_id», «phone_number», «duration, employee_id», «call_date», «audio_url», записуються до таблиці calls у PostgreSQL. Паралельно запускається асинхронний процес Celery, який через requests.get завантажує MP3-файли у хмарне сховище для подальшого аналізу засобами Google Gemini API [51].

Інший метод забезпечує оперативність: вебхуки дають змогу отримувати дані в режимі реального часу. Спеціальний Django-ендпоінт /binotel/webhook опрацьовує вхідні POST-запити від АТС, перевіряючи їхню автентичність за допомогою цифрового підпису та API Secret. Після валідації система створює задачу в Celery для збереження метаданих та аудіофайлу, що гарантує значно швидшу синхронізацію порівняно з погодинним розкладом. У лістингу 3.3 показано код для взаємодії із API Binotel. Повний код відображено у додатку Б.

Лістинг 3.3 – Програмний код, призначений для взаємодії з API Binotel

```
class BinotelApi:
      def     __init__(self,     key,     secret,     api_host=None,
api_version=None, api_format=None):
        self.key = key
        self.secret = secret

        self.api_host = 'https://api.binotel.com/api/'
        self.api_version = '4.0'
        self.api_format = 'json'
        self.disable_ssl_checks = False
        self.debug = False

        if api_host is not None:
            self.api_host = api_host
        if api_version is not None:
            self.api_version = api_version
        if api_format is not None:
            self.api_format = api_format

     def send_request(self, url, params):
         """
        Sends a POST request with JSON data to the constructed API
endpoint.
         """
         if self.debug:
                                print("[BINOTEL CLIENT] Send request:
{}".format(json.dumps (params)))

        params['key'] = self.key
        params['secret'] = self.secret

        post_data = json.dumps (params)
           full_url = f" {self.api_host} {self.api_version}/{url}.
{self.api_format}"
```

Для організації взаємодії з платформою Binotel [43] розроблено клас BinotelApi. Він виконує роль клієнтської обгортки, що інкапсулює логіку формування HTTP-запитів, автоматизує додавання авторизаційних даних та уніфікує обробку відповідей і виключних ситуацій. Це дає змогу централізувати комунікацію із зовнішнім сервісом, уникаючи дублювання коду. Повний код відображено у додатку В.

Конструктор __init__ забезпечує початкову конфігурацію екземпляра класу. Він приймає обов’язкові облікові дані (key та secret). Також передбачено гнучке налаштування параметрів з’єднання: api_host, api_version та api_format. У разі відсутності явних аргументів застосовуються налаштування за

замовчуванням: URL https://api.binotel.com/api/, версія 4.0 та формат JSON. Додатково ініціалізуються службові опції: disable_ssl_checks – керування перевіркою сертифікатів та debug – режим розширеного логування для налагодження.

Функціонал обміну даними зосереджено в методі send_request, який оперує назвою ендпоінту та набором вхідних параметрів. Перед ініціацією з'єднання метод автоматично доповнює запит авторизаційними ключами, необхідними для доступу до Binotel.

Процес підготовки включає конвертацію даних у JSON, генерацію повного URL, залежно від версії API та формування HTTP-заголовків із зазначенням типу контенту. Для фізичної передачі даних використовується POST-запит бібліотеки requests. Система містить вбудований механізм відловлювання помилок: фіксуються як мережеві збої, так і коди відповіді, відмінні від 200. Якщо комунікація пройшла успішно і отримано валідний JSON, метод повертає результат у вигляді словника.

3.5 Інтеграція з KeepinCRM для синхронізації даних клієнтів та дзвінків

Інтеграція з KeepinCRM [44] виступає ключовим елементом серверної архітектури інформаційної системи для аналітичного опрацювання телефонних дзвінків, гарантуючи автоматизований експорт аналітичних даних безпосередньо у картки клієнтів, що суттєво оптимізує процеси продажів. Використання цієї вітчизняної платформи дає змогу централізувати історію комунікацій, що повністю задовольняє функціональні специфікації інформаційної системи для аналітичного опрацювання телефонних дзвінків засобами Django [30], а також супутніх технологій: Celery, Celery Beat, PostgreSQL та Redis. Технічно взаємодія побудована на базі Django REST Framework, Docker, PostgreSQL і Celery [25], використовуючи API CRM-системи для обміну даними та фонового виконання операцій.

Для активації з'єднання користувач, зокрема адміністратор, отримує APIключ у налаштуваннях особистого кабінету KeepinCRM або через запит до технічної підтримки. З метою безпеки цей токен розміщується у конфігураційному файлі env/.env і зчитується бібліотекою python-decouple. На стороні Django [30] створено ендпоінт API, який дає змогу вводити облікові дані через графічний інтерфейс та верифікувати їх шляхом надсилання тестового запиту до серверів CRM. Параметри конфігурації зберігаються в окремій таблиці PostgreSQL, що забезпечує гнучкість архітектури та підтримку мультиакаунтінгу для різних користувачів.

Процес передачі інформації до KeepinCRM ініціюється після завершення обробки аудіо, яку виконує Google Gemini API [51]. Підсумкові метрики, як от текстова розшифровка, рейтинг якості розмови у відсотках чи балах, а також виявлені інсайти, формуються окремим завданням Celery та фіксуються в базі PostgreSQL [40]. Після цього запускається асинхронний процес, який за допомогою бібліотеки requests надсилає POST-запит на відповідні ресурси API KeepinCRM /calls або /interactions. Цей запит включає метадані – номер, дата, тривалість, відповідальний менеджер, транскрипцію та аналітичні коментарі, що збагачують історію взаємодії з клієнтом. У результаті, наприклад, у KeepinCRM генерується запис у модулі «Дзвінки» з прикріпленою оцінкою ефективності та посиланням на розширений звіт.

Архітектура підтримує двосторонній обмін даними: система на базі Django здатна імпортувати відомості з KeepinCRM, як от картки клієнтів чи етапи угод, для забезпечення контекстного аналізу розмов. Технічно це реалізовано через запити до ресурсів /clients або /deals API CRM-системи, після чого отримана інформація кешується в PostgreSQL для кореляції з телефонними дзвінками. Планувальник Celery Beat відповідає за регулярну актуалізацію бази, що гарантує свіжість даних без необхідності ручного втручання. Роль диспетчера черг та сховища для Celery виконує Redis [26], що дає змогу ефективно опрацьовувати тисячі завдань і дотримуватися жорстких часових рамок (обробка п'ятихвилинного запису займає до 30 секунд).

Захист інформаційного обміну базується на криптографічних стандартах: токени доступу зберігаються у зашифрованому вигляді, а трафік до KeepinCRM API передається через захищений канал HTTPS [46]. Для верифікації кожного запиту застосовується технологія Bearer Token, що унеможливлює несанкціонований доступ. Система обробки виключень налаштована так, що при збоях, наприклад, недійсний ключ чи відмова API, Celery фіксує інцидент у Redis та сповіщає адміністратора через засоби Django, що відповідає регламенту надійності. Ізоляція сервісів Django [45], Celery, Redis [26] досягається завдяки Docker-контейнерам, а транзакційна модель PostgreSQL гарантує цілісність збережених даних, підвищуючи загальну стійкість платформи.

Взаємодія з KeepinCRM суттєво розширює можливості комплексу, забезпечуючи автоматичне збагачення CRM-системи результатами аудиту, що спрощує роботу відділів продажів та контролю якості. Наявність спеціального Django-ендпоінту робить налаштування підключення простим та зрозумілим, а асинхронна архітектура Celery [25] підтримує стабільну продуктивність при масивах даних великого обсягу. Такий підхід формує гнучку основу для масштабування, даючи змогу в майбутньому впроваджувати поведінкову аналітику або налаштовувати миттєві сповіщення про критичні діалоги. На лістингу 3.4 показано код для взаємодії із API KeepinCRM.

Лістинг 3.4 –Код для інтеграції KeepinCRM

```
class KeepinCRMAPI:
    def __init__(self, api_key):
        """
        Initialize the KeepInCRM API
           clientwith an API key.
        Args:
            api_key (str):
                   Your KeepinCRM API key
        """
        self.api_key = api_key
        self.api_host = "https://api.keepincrm.com/v1/"
        self.headers = {
            "X-Auth-Token":
                   f"{self.api_key}",
            "Accept": "application/json",
            "Content-Type": "application/json"
        }
```

```
    def send_request(self, method, endpoint, data=None,
params=None):
        """
        Send a request to the KeepinCRM API and handle the response.
        Args:
          method (str): HTTP method (GET, POST, PUT, DELETE, etc.)
          endpoint (str): API endpoint (e.g., 'clients')
         data (dict, optional): Request body data for POST/PUT
requests
         params (dict, optional): Query parameters for the request
        Returns:
            dict: JSON response from the API

        Raises:
           Exception: If there's an HTTP error, connection error, or
JSON
decode error
        """
```

Клас KeepinCRMAPI виступає програмною обгорткою для комунікації із зовнішньою платформою KeepinCRM, яка забезпечує автоматизацію роботи з клієнтською базою, замовленнями та операційними задачами. Ключове призначення цього модуля – надати стабільний та уніфікований інтерфейс для генерації запитів до API, коректної інтерпретації відповідей сервера, а також для реалізації прикладних функцій, таких як прикріплення нотаток до профілів або ідентифікація особи за контактним номером [44].

Процес налаштування екземпляра, а саме ініціалізація класу (init) потребує передачі унікального ключа доступу, який є обов’язковим для проходження автентифікації. У рамках конструктора також визначається кореневий хост сервісу api_host та формується набір HTTP-заголовків. Останні містять токен доступу та специфікацію формату обміну даними JSON і автоматично додаються до кожного вихідного запиту.

Фундаментальний механізм обміну даними зосереджено у функції send_request. Вона оперує такими аргументами, як тип HTTP-методу та цільовий шлях (endpoint), а також приймає необов’язкові параметри для тіла запиту (data) або URL-параметрів (params). Для запобігання синтаксичним помилкам адреса нормалізується шляхом видалення дубльованих слешів. Безпосереднє виконання звернення здійснюється через універсальний виклик requests.request із встановленим лімітом очікування у десять секунд, що захищає програму від

зависання при проблемах з мережею. У випадках, коли сервер повертає помилку або некоректний формат відповіді, генерується виключення з деталізацією причини збою.

Функція add_note_to_client призначена для фіксації текстових приміток у картці конкретного замовника. Алгоритм її роботи включає генерацію коректного URL та підготовку словника з текстом повідомлення. Після цього ініціюється виклик send_request із використанням методу POST. Якщо операція завершується невдало, повертається значення False, у протилежному випадку – розпарсена структура JSON від API.

Для ідентифікації користувача за його телефонним номером, джерелом походження, наприклад, веб-форма та часом реєстрації застосовується метод get_client_id. Він конструює набір фільтрів для API-запиту: ідентифікатор джерела повинен мати точний збіг source_id_eq, дата реєстрації перевіряється на відповідність мінімальному порогу registered_at_gteq, а пошук телефону виконується через триграмний індекс trigram_idx_cont, що дає змогу знаходити часткові співпадіння. Якщо система не знаходить відповідностей, повертається False, інакше функція віддає ID останнього знайденого запису.

3.6 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи описано процес створення системи для аналітичного опрацювання телефонних дзвінків, а також розглянуто практичні аспекти розробки її серверної частини. Спроєктовано архітектуру бекенду, а саме: використання Django як основного фреймворку, Celery та Celery Beat для асинхронного виконання задач і планування періодичних процесів, а також інтеграцію PostgreSQL і Redis для ефективного управління даними та кешування. Досліджено методи побудови масштабованої та надійної системи, здатної працювати з великими масивами інформації у режимі реального часу.

У межах розділу було створено Dockerfile та Compose-файли, які забезпечують контейнеризацію сервісів та спрощують розгортання всієї інфраструктури. Особливу увагу приділено інтеграції з телефонією Binotel та

системою KeepinCRM, що дає змогу автоматизувати збір, обробку та подальший аналіз дзвінків у єдиному програмному середовищі. Подано опис обчислювального експерименту, який підтверджує працездатність розроблених компонентів та демонструє можливості системи щодо обробки та аналітики даних.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Вимоги щодо охорони праці при роботі з комп’ютерами. Інструкція для програміста

Тема кваліфікаційної роботи рівня «Магістр» присвячена розробці інформаційної системи для аналітичного опрацювання телефонних дзвінків із використанням сучасних технологій програмування й баз даних. Оскільки реалізація систем такого класу потребує тривалої роботи за комп’ютерною технікою, доцільно розглянути вимоги щодо охорони праці, спрямовані на забезпечення комфортних умов, зменшення фізичного й зорового навантаження, а також підтримання працездатності програміста під час виконання професійних обов’язків.

Працюючи перед екраном, програміст має перебувати у зручному, ергономічно продуманому робочому середовищі. Монітор слід розмістити так, щоб відстань від очей до екрану становила близько 50–70 см, а верхній край монітора перебував на рівні очей або трохи нижче – це сприяє природному положенню шиї та зменшує навантаження на очі. Екран бажано відрегулювати під невеликим кутом, що зменшує можливі відблиски і дозволяє зменшити втому зорового апарату. Робочий стіл має бути достатньо просторим, щоб забезпечувати зручне розташування клавіатури, миші та іншої периферії без скупчення, а крісло – мати регулювання висоти та спинки з підтримкою поперекового відділу хребта, що дозволяє зберігати правильну поставу протягом тривалого часу. Ноги працівника мають стояти рівно – на підлозі або підставці, щоб уникнути зайвого навантаження на ноги і суглоби [53].

Освітлення приміщення, де ведеться робота, мусить бути комфортним та безпечним: рівень освітлення має бути достатнім для чіткого розрізнення символів на екрані та документах, а при цьому джерела світла не повинні створювати відблисків чи тіней, які дратують зір. Найбільш оптимальним є поєднання природного та штучного світла, з урахуванням розташування вікон і позиції монітора, щоб уникнути прямого потрапляння сонячних променів на

екран. Також важливо підтримувати комфортний мікроклімат у приміщенні: температура повітря, вологість та вентиляція повинні відповідати нормам, що забезпечують тривале перебування за комп'ютером без дискомфорту.

Тривала інтелектуальна праця за ПК пов'язана з ментальним і фізіологічним навантаженням, яке може виявлятися у втомі, зниженні продуктивності, зоровому напруженні чи болях у м'язах. Тому програміст повинен організувати свою роботу так, щоб давати тілу можливість на відпочинок і відновлення. Регулярна зміна позиції, короткі паузи для відпочинку очей і виконання легких фізичних вправ – усе це сприяє зменшенню негативного впливу статичної роботи. Змінюючи діяльність протягом робочого дня (наприклад, чергуючи кодування, планування, документування), можна знизити ризик монотонного перенавантаження та підтримувати високу якість виконання завдань [54].

Дотримання гігієнічних норм також є невід'ємною частиною безпечної та здорової праці. Підтримка чистоти робочого місця, монітора, клавіатури, зручне розташування периферії, використання сучасних пристроїв із зменшеним мерехтінням або з опціями зниження синього світла – усе це дає змогу зменшити зорове навантаження та покращити самопочуття. Крім того, важливо балансувати навантаження: навіть при великому обсязі задач регулярний режим роботи, адекватні перерви, правильна організація робочого часу та планування сприяють підтриманню здоров'я та ефективності.

Ця інструкція є базовим орієнтиром, який програміст має враховувати при організації свого робочого процесу. Вона покликана забезпечити такі умови, за яких тривала робота з комп'ютером не буде шкідливою, а навпаки – продуктивною і безпечною для здоров'я. Виконання цих рекомендацій – важлива складова відповідального ставлення до професійної діяльності, яка підвищує шанси на довгу, плідну та стабільну роботу над створенням інформаційних систем [55].

4.2 Забезпечення безпеки життєдіяльності при роботі з ПК

Тема кваліфікаційної роботи рівня «Магістр» присвячена розробці інформаційної системи для обробки телефонних дзвінків. При реалізації такого проєкту програміст проводить тривалий час за комп’ютером, що зобов’язує не лише дотримуватися ергономічних та гігієнічних норм, але й забезпечити безпечні умови життєдіяльності під час роботи з електронним обладнанням. Тому доцільно розглянути питання, які гарантують захист здоров’я, запобігають травмам, аваріям або пожежам, а також передбачають адекватну реакцію у разі небезпечних ситуацій.

Перш за все, робоче місце має бути обладнане з урахуванням правил електробезпеки [56]. Вся комп’ютерна техніка, монітор, системний блок, периферійні пристрої повинні підключатися до справної, заземленої розетки. Важливо забезпечити надійне заземлення і захист від перепадів напруги, а також уникати перевантажень електромережі – не підключати до однієї розетки занадто багато приладів одночасно. Перед початком роботи необхідно переконатися, що кабелі не пошкоджені, вилки справні, корпус обладнання не має тріщин або ознак перегрівання. Будь-які дії з внутрішніми компонентами – наприклад, відкриття корпусу системного блоку чи заміну комплектуючих – допускаються лише при повному знеструмленні обладнання. Такий підхід мінімізує ризики ураження електричним струмом або короткого замикання.

Окрім електробезпеки, важливим є забезпечення належної пожежної безпеки. Комп’ютерна техніка під час тривалої роботи може нагріватися, особливо у випадках інтенсивного навантаження (довге компілювання, робота з віртуальними машинами чи контейнерами, багато відкритих процесів), що – за відсутності відповідного охолодження – здатне спричинити перегрів або відмову вентиляторів. Відтак робоче місце має передбачати достатню вентиляцію, щоб техніка могла охолоджуватися природнім чи примусовим способом, без перегрівання корпусу. Важливо, щоб навколо техніки не було горючих матеріалів (паперів, тканин, коробок), які підвищують ризик пожежі у разі іскріння чи перегрівання компонентів.

Крім того, програміст повинен бути обізнаний з алгоритмом дій у разі технічної несправності, перегріву пристроїв, запаху гару, диму або тріскоту – ознак, які можуть передувати аварії або пожежі. У таких випадках необхідно негайно припинити роботу, вимкнути електроживлення, повідомити відповідальних осіб і, при потребі, застосувати первинні засоби пожежогасіння або евакуації. Своєчасна реакція і обізнаність – важлива складова безпеки життєдіяльності [57].

Також варто подбати про безпечну організацію кабелів та простору навколо робочого місця. Кабелі харчування та підключення повинні бути акуратно укладені, не перехрещуватися, не створювати перешкод і не лежати на проходах – це мінімізує ризик спіткнутися, зачепити шнур або пошкодити його. Така організація простору зменшує ймовірність механічних ушкоджень обладнання та особистих травм.

Не менш важливо забезпечити умови, які запобігають стресу або паніці у випадку аварійних ситуацій. Програміст має чітко знати, як діяти при несподіваних подіях: коли вимикати обладнання, куди звертатися, як покинути робоче місце, не наражаючи себе на додаткову небезпеку.

Окрім внутрішніх чинників, на безпеку життєдіяльності впливає зовнішнє середовище. В умовах воєнної агресії Росії екосистема та інфраструктура країни зазнають значних змін, що може створювати додаткові ризики, включаючи перебої в електропостачанні та загрозу об’єктам критичної інфраструктури. Тому особлива увага має приділятися плануванню дій у випадку надзвичайних ситуацій та забезпеченню безпечної організації робочого процесу [58] .

Нарешті, важливо враховувати, що життя та здоров’я працівника – найвища цінність. Надійне та безпечне забезпечення робочого середовища, правильне технічне підключення, адекватне охолодження техніки, запобігання перегріву, акуратне укладання кабелів, відповідальна реакція на ознаки несправності – усе це створює умови, які забезпечують безпеку життєдіяльності навіть у разі несподіваних подій. Такий підхід не лише захищає здоров’я, але й підтримує працездатність, зменшує ризики аварій і створює надійний фундамент для якісної тривалої роботи над проєктом.

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи узагальнено основні вимоги охорони праці під час роботи з комп’ютером та особливості організації безпечного робочого місця програміста. Окреслено ключові аспекти раціонального мікроклімату, освітлення, режиму праці й відпочинку, а також фактори, що впливають на здоров’я та працездатність фахівця. Наведені положення формують підґрунтя для забезпечення комфортних умов праці та мінімізації професійних ризиків.

Також у розділі розглянуто питання безпеки життєдіяльності в надзвичайних ситуаціях, що можуть виникати у приміщеннях з комп’ютерною технікою. Підкреслено важливість знання алгоритмів дій під час пожежі, аварій електромережі та інших потенційно небезпечних подій. Поданий матеріал має практичний характер і забезпечує цілісне розуміння вимог безпеки, необхідних для стабільної та безпечної роботи у сфері інформаційних технологій.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи освітнього рівня «Магістр» було проведено комплексне дослідження теоретичних і практичних аспектів створення системи аналітичного опрацювання телефонних дзвінків. Особливу увагу приділено вибору технологій, архітектурним рішенням та інтеграції сервісів, необхідних для забезпечення точності, надійності та швидкодії системи.

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр»:

– Подано актуальність дослідження та створення спеціалізованих аналітичних систем.

– Розглянуто сучасні інформаційні технології аналізу телефонних дзвінків.

– Висвітлено процес оптимізації бізнес-процесів через AI-аналіз дзвінків.

– Проаналізовано та визначено вимоги до інформаційної системи для аналітичного опрацювання телефонних дзвінків.

– Сформовано постановку задачі створення інформаційної системи для аналітичного опрацювання телефонних дзвінків.

В другому розділі кваліфікаційної роботи:

– Обґрунтовано вибір програмно-алгоритмічної платформи для системи аналізу дзвінків.

– Досліджено можливості Django та Celery для асинхронної обробки ресурсомістких завдань опрацювання аудіо даних телефонних дзвінків.

– Описано контейнеризацію сервісів за допомогою Docker для потокового аналітичного опрацювання аудіо дзвінків.

– Сформовано структуру сховища даних для зберігання інформації щодо телефонних дзвінків та клієнтів.

В третьому розділі кваліфікаційної роботи:

– Розроблено принципи та підходи до обробки телефонних дзвінків.

– Спроєктовано архітектуру серверної частини системи аналітики дзвінків.

– Налаштовано Docker контейнери для функціонування інформаційної системи аналітичного опрацювання телефонних дзвінків.

– Інтегровано інформаційну систему з Binotel для збору та обробки дзвінків.

– Здійснено інтеграцію з KeepinCRM для синхронізації даних клієнтів та дзвінків.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» проаналізовано вимоги щодо охорони праці при роботі з комп’ютерами. Описано забезпечення безпеки життєдіяльності при роботі з ПК.

ПЕРЕЛІК ДЖЕРЕЛ

1 Joshi, Aditya, et al. "Natural language processing for dialects of a language: A survey." ACM Computing Surveys 57.6 (2025): 1-37.

2 Qatawneh, Adel M. "The role of artificial intelligence in auditing and fraud detection in accounting information systems: moderating role of natural language processing." International Journal of Organizational Analysis 33.6 (2025): 1391-1409.

3 Luchkevych, M., I. Shakleina, and O. Duda. "The impact of modern cloud technologies on the efficiency of DevOps processes." ВІСНИК ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ Учредители: Тернопольский национальный технический университет им. Ивана Пулюя 117.1 (2025): 112-122.

4 El Bahri, Jalal, Mohamed Kouissi, and Mohammed Achkari Begdouri. "Sustainable Speech Recognition: Energy, Carbon, and Performance Comparison of Whisper (Base and Large) and Google Speech-to-Text V2 (Chirp/USM)." Energy-Efficient Algorithms and Systems in Computing: Optimizing Performance and Sustainability Through Advanced Computational Methods. Cham: Springer Nature Switzerland, 2025. 213-226.

5 Schwarzer, Will, et al. "Are Deep Speech Denoising Models Robust to Adversarial Noise?." arXiv preprint arXiv:2503.11627 (2025).

6 Chen, Yunqi, et al. "Understanding the OSS Communities of Deep Learning Frameworks: A Comparative Case Study of PyTorch and TensorFlow." ACM Transactions on Software Engineering and Methodology 34.3 (2025): 1-30.

7 Lemenkova, Polina. "Automation of image processing through ML algorithms of GRASS GIS using embedded Scikit-Learn library of Python." Examples and Counterexamples 7 (2025): 100180.

8 NOUHAS, Hasnae, Abdessamad BELANGOUR, and Mahmoud NASSAR. "A Weighted Scoring Model Analysis of Cloud Storage Services: Comparing AWS, Azure, and Google Cloud Platform."

9 DUDA, OLEKSII, IRYNA SHAKLEINA, and MYKHAILO LUCHKEVYCH. "INCREASING THE EFFICIENCY OF DEVOPS THROUGH

THE USE OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING." Herald of Khmelnytskyi National University. Technical sciences 351.3.1 (2025): 143-149.

10 Judijanto, Loso, Alim Hardiansyah, and Opan Arifudin. "Ethics And Security In Artificial Intelligence And Machine Learning: Current Perspectives In Computing." International Journal of Society Reviews (INJOSER) 3.2 (2025): 374-380.

11 Goswami, Debashish, and Tahmina Rainy. "Mechanisms by which AI-Enabled CRM Systems Influence Customer Retention and Overall Business Performance: A Systematic Literature Review of Empirical Findings." ASRC Procedia: Global Perspectives in Science and Scholarship 1.01 (2025): 10-63125.

12 Patni, Sanjay. "Fundamentals of RESTful APIs." Pro RESTful APIs with Micronaut: Build Java-Based Microservices with REST, JSON, and XML. Berkeley, CA: Apress, 2025. 1-14.

13 Tsai, Omar, et al. "GraphQLer: Enhancing GraphQL Security with Context-Aware API Testing." arXiv preprint arXiv:2504.13358 (2025).

14 Lypak, H., et al. "Розроблення підходів до вибору методології проєктування інтерфейсу смартсистеми." COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION 60 (2025): 190-199.

15 Rudenko, Maksym, and Svetlana Sotnik. "Classification of CRM systems." (2025).

16 Vijayakumar, P., M. Pyingkodi, and S. Devi. "Comparative Analysis of AI Chatbot for Assessing Gemini AI, DeepSeek AI, and Qwen AI via OpenRouter API Integration." 2025 6th International Conference on Data Intelligence and Cognitive Informatics (ICDICI). IEEE, 2025.

17 Kunanets, Nataliia, et al. "DESIGNING THE STRUCTURE AND ARCHITECTURE OF SITUATION-AWARE SECURITY INFORMATION SYSTEMS FOR RESIDENTIAL COMPLEXES." Eastern-European Journal of Enterprise Technologies 133.9 (2025).

18 Sundberg, Edwin, Thea Ekmark, and Workneh Yilma Ayele. "Validating API Design Requirements for Interoperability: A Static Analysis Approach Using OpenAPI." arXiv preprint arXiv:2511.17836 (2025).

19 Macron, Tolu. "Enhancing VoIP network security: An Asterisk-based system approach." (2025).

20 Hasan, Rakib, et al. "WhatsApp Based Smart Home Automation System." 2025 2nd International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM). IEEE, 2025.

21 Houghton, Zachary Nicholas, et al. "A Novel Dataset for Testing Anti-spoofing Models in a Telephony Environment." 2025 IEEE International Conference and Expo on Real Time Communications at IIT (RTC). IEEE, 2025.

22 Bednarz, Bartłomiej, and Marek Miłosz. "Benchmarking the performance of Python web frameworks." Journal of Computer Sciences Institute 36 (2025): 336-341.

23 Singh, Ajit. "Google Gemini's Game-Changer: AI That Calls APIs & Runs Code." Available at SSRN 5213335 (2025).

24 Babali, Tunar, and Nail Mammadov. "Optimizing High-Concurrency Access to Conditions Data: A Kubernetes Orchestrated Solution with PostgreSQL and Django." SCIENTIFIC WORK Учредители: Azerbaijan Science Center 19.1 (2025): 111-116.

25 John, Beauden. "Integrating a Unified Threat Detection System Using Celery for Browser and Email Protection." (2025).

26 Zhu, Yunkai, et al. "RAPO: An Automated Performance Optimization Tool for Redis Clusters in Distributed Storage Metadata Management." IEEE Access (2025).

27 LADO, MARK JOHN. Flask Web Framework Building Interactive Web Applications with SQLite Database: A Practical, Hands-on Guide for Beginners to Intermediate Developers, Including Real-World Projects and Step-by-Step Instructions for Creating Dynamic and Engaging Web Experiences. Amazon Digital Services LLC-Kdp, 2025, 2025.

28 Springer, Sebastian. Node. js: the comprehensive guide. Packt Publishing Ltd, 2025.

29 Scott, Christopher, and Brandon Lewis. "Advancements in Web Application Development: An Analytical Review of Ruby on Rails, Python Frameworks, and Cloud-Centric Solutions."

30 Jha, Abhishek. "Node. js vs. Django: A Performance and Scalability Comparison." (2025).

31 Putra, Sipky Jaya, et al. "Comprehensive Benchmarking of Message Brokers: Evaluating Performance and Security Metrics for Reliable Messaging Systems." Jurnal Locus Penelitian dan Pengabdian 4.11 (2025): 10829-10838.

32 Aqasizade, Hossein, Ehsan Ataie, and Mostafa Bastam. "Kubernetes in action: Exploring the performance of kubernetes distributions in the cloud." Software: Practice and Experience (2025).

33 Divakaran, Adarsh. "Multiprocessing." Deep Dive Python: Techniques and Best Practices for Developers . Berkeley, CA: Apress, 2025. 471-503.

34 Дуда, Олексій, Микола Орлов, and Ігор Павлів. "ІНТЕГРАЦІЯ ЗАСОБІВ АНАЛІЗУ ВИХІДНОГО КОДУ У ІННОВАЦІЙНІЙ МЕТОДОЛОГІЇ DEVSECOPS." (2025).

35 Lee, Janghun, and Daejin Park. "Docker based Embedded Software Management and Update with Dynamic Library Techniques." 2025 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan). IEEE, 2025.

36 Bolshakov, Sergey. "Lightweight Deployment of AWS ECS Without Configuration Drift." Emerging Frontiers Library for The American Journal of Engineering and Technology 7.09 (2025): 195-202.

37 Ksontini, Emna, et al. "Refactoring for Dockerfile Quality: A Dive into Developer Practices and Automation Potential." 2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR). IEEE, 2025.

38 Balasaheb, Borse Pradnya. "IOT-DRIVEN SMART CITIES: ENHANCING ATTACK DETECTION VIA CLOUD-BASED ANALYTICS AND MULTIFACTOR AUTHENTICATION." International Journal of Applied Mathematics 38.2s (2025): 966-984.

39 Antonova, V. M., et al. "Application of Virtualization and Containerization Technologies in IP Telephony." 2025 Systems of Signals Generating and Processing in the Field of on Board Communications. IEEE, 2025.

40 Zapata, Jannette G. "MySQL VS PostgreSQL: A Comparative Analysis of Relational Database Management Systems (RDBMS) Technologies Response Time in Web-based E-commerce."

41 Липак, Галина, et al. "Побудова інтерфейсів користувача вебсайту бібліотеки на засадах UX-дизайну." Цифрова платформа: інформаційні технології в соціокультурній сфері 8.1 (2025): 172-192.

42 Simplifying, A. P. I., and Ganeshkumar Patil. "Django REST APIs Demystified."

43 Войтенко, Анна Сергіївна. "Розробка автоматизованої CRM-системи для оптимізації процесів у сервісі ремонту мобільної техніки."

44 Stupak, Victoria. "Digital solutions for client relation management in real estate companies." (2025).

45 Deolino, Júlio César Maia. "Developing an IoT system using Django and an MQTT broker to perform automation in agriculture." (2025).

46 Liew, Han Hui, Julia Juremi, and Nipuna Hiranya Weeratunge. "Strategies to Enhance Web Applications Security." Innovations in Communication Networks: Sustainability for Societal and Industrial Impact: Proceedings of 5th International Conference on Data Engineering and Communication Technology (ICDECT 2024), Volume 4. Vol. 1365. Springer Nature, 2025.

47 Kochevoy, Maxim, and Oleksandr Panaiet. "COMPARISON MARKETING STRATEGIES OF CRM SYSTEMS IN UKRAINE AND ABROAD: THE CASE OF SNOV. IO AND PIPEDRIVE." Scientific Bulletin of the Odessa National Economic University (2025): 163-170.

48 Taha, Taha Adel, and Ayad Hussain Abdulqader. "The importance of using docker containers in building a web-based system: Activstaff as a case study." AIP Conference Proceedings. Vol. 3264. No. 1. AIP Publishing LLC, 2025.

49 Neupane, Sagar. "Detecting and Mitigating SQL Injection Vulnerabilities in Web Applications." arXiv preprint arXiv:2506.17245 (2025).

50 Andreiev, Anton, and Svetlana Sotnik. "“Web application security: protection against modern cyber threats” Overview of key vulnerabilities (XSS, CSRF, SQL injections), protection methods, use of HTTPS, authentication, and authorization." (2025).

51 Rekha, V., and G. L. Prakash. "Comparative Study and Analysis of Prompting Techniques Using Gemini API Model and Reasoning Frameworks." 2025 International Conference on Computing for Sustainability and Intelligent Future (COMP-SIF). IEEE, 2025.

52 Drofa, Denys. "Integrating Advanced API Solutions into Full-Stack Web and Mobile Applications to Optimise User Experience." International Journal of Current Science Research and Review 8.05 (2025).

53 Казюра, А. В., and І. В. Віштак. ЕРГОНОМІЧНІ АСПЕКТИ ОРГАНІЗАЦІЇ РОБОЧИХ МІСЦЬ ЯК ФАКТОР ЗАБЕЗПЕЧЕННЯ ОХОРОНИ ПРАЦІ ТА ЗБЕРЕЖЕННЯ ЗДОРОВ'Я ПРАЦІВНИКІВ. Diss. Львівський державний університет безпеки життєдіяльності, 2025.

54 Герасимчук, О. В., and О. В. Кобилянський. Вплив тривалої роботи за комп’ютером на здоров’я студентів: шляхи мінімізації ризиків. Diss. ВНТУ, 2025.

55 Гурик, Олег Ярославович, et al. "Навчально-методичний посібник до практичних заняття з дисципліни «Безпека життєдіяльності, основи охорони праці» для студентів освітнього ступеня, бакалавр" усіх спеціальностей та форм навчання." (2025).

56 Бойко, С. М., and О. В. Кобилянський. Формування компетентності з електробезпеки майбутніх фахівців у процесі навчання. Diss. ВНТУ, 2025.

57 Курепін, Вячеслав Миколайович. "Безпека життєдіяльності." (2025).

58 Гурик, Олег Ярославович, et al. "Вплив воєнної агресії Росії на екосистему та безпеку життєдіяльності." Збірник тез V Міжнародної наукової конференції „Воєнні конфлікти та техногенні катастрофи: історичні та психологічні наслідки “ (2025): 87-89.

ДОДАТКИ

Додаток А

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

17-18 грудня 2025 року

ТЕРНОПІЛЬ
2025

УДК 001
М34

ПРОГРАМНИЙ КОМІТЕТ

Голова: Микола Приймак– професор кафедри комп'ютерних систем та мереж, д.т.н., професор;

Співголови: Павло Марущак– докт. техн. наук, професор, проректор з наукової роботи.
Ігор Баран– канд. техн. наук, доцент, декан факультету ФІС.

Науковий секретар: Галина Семенишин– старший викладач.

Члени: докт. фіз.-мат. наук, професор Василь Кривень; докт. техн. наук, професор Ярослав Литвиненко; докт. техн. наук, професор Микола Карпінський; докт. фіз.-мат. наук, професор Михайло Петрик; канд. техн. наук, доцент Галина Осухівська; канд. пед. наук, доцент Жанна Баб'як; канд. техн. наук, доцент Наталія Загородна.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова: Юрій Скоренький– канд. фіз.-мат. наук, доцент кафедри фізики

Члени: канд. техн. наук, доцент Вячеслав Никитюк; канд. техн. наук, доцент Дмитро Михалик; канд. техн. наук, доцент Марія Стадник; канд. техн. наук Євгенія Тиш; ст. викладач Ліліана Джиджора.

М34 Матеріали XIII науково-технічної конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 17–18 грудня 2025 р.). – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2025. –242 с.

Адреса оргкомітету: ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. (0352) 52-41-33, факс (0352) 254983.
E-mail: conffis2025@gmail.com

Редагування, оформлення та верстка: Галина Семенишин

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- Математичне моделювання;
- Інформаційні системи та технології;
- Комп'ютерні системи та мережі;
- Програмна інженерія та моделювання складних розподілених систем;
- Новітні фізико-технічні та освітні технології.

В збірнику надруковано тези доповідей XIII науково-технічної конференції «Інформаційні моделі, системи та технології» (Тернопіль, 17–18 грудня 2025 р.) за такими науковими напрямками: математичне моделювання; інформаційні системи та технології; комп'ютерні системи та мережі; програмна інженерія та моделювання складних розподілених систем; новітні фізико-технічні та освітні технології.

Розрахований на науковців, викладачів та студентів вузів.

За зміст тез та дотримання норм академічної доброчесності відповідальність несе автор.

УДК 004.03
М. Боднар; Г. Вовнянка; В. Дуда
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПЕРЕДОВІ ТЕХНОЛОГІЇ ОБРОБКИ ДАНИХ У РОЗУМНИХ МІСТАХ

UDC 004.03
M. Bodnar; H. Vovnianka; V. Duda

ADVANCED DATA PROCESSING TECHNOLOGIES IN SMART CITIES

Швидкий прогрес інформаційних та комунікаційних технологій забезпечив нові методи обробки та аналізу джерел даних у «розумних містах». Для цього застосовуються різноманітні технології для дистрибуції міст, щоб зробити їх більш придатними для життя, ефективними, інтелектуальними, сталими та стійкими. Інтелектуальні та ефективні моделі обробки даних для застосунків машинного зору в «розумних містах» впроваджують в інтелектуальні системи виявлення для моніторингу швидкості та потоку транспортних засобів, а також руху пішоходів на дорогах та тротуарах [1]. Застосування моделей манинного навчання в «розумних містах»:

– Моделі зберігання та пошуку для обробки масивних відеоданих у хмарі для моніторингу транспортних засобів та дотримання правил дорожнього руху в системах відеоспостереження та транспорту.

– Моделі на основі зображень та розпізнавання для аналізу поширених даних зображень у містах, зокрема системи розпізнавання пожеж на основі глибокого навчання для розумної профілактики пожеж та системи виявлення кришок люків у моніторингу проїжджої частини та вулиць.

– Моделі прогнозування потокових даних у «розумних містах» для завчасного визначення заторів на дорогах та тенденцій руху пішоходів.

– Моделі оптимізації в «розумних містах», зокрема моделі інтелектуальної маршрутизації та планування шляху для навігації транспортних засобів у логістичних, дистрибуційних та постачальних системах [1].

Вищезазначена родина моделей в основному адаптована до сфер застосування «розумної» мобільності та «розумного» пішохідного руху. Однак, також є багато дослідників, які працюють над моделями обробки даних в інших міських доменах, або незалежно, або в інтегрованому режимі з транспортом. В [1] зроблено огляд численних платформ «розумного міста» на ринках для управління даними та інтеграції з різноманітними застосунками. Перспективним є надання відносно комплексного огляду моделей обробки міських даних з різними методами для різних прикладних доменів.

«Розумні міста» зазнали швидкого зростання і тому стали головною тенденцією у розвитку «розумних» середовищ. «Розумні міста» прагнуть покращити придатність міських поселень для життя, підвищити сталість та зменшити експлуатаційні витрати. Очікується, що концепція «розумного міста» принесе значну користь, і вона була прийнята великими містами по всьому світу. Однак цінність бачення «розумних міст» може бути повністю реалізована лише через успішну інтеграцію передових технологій та перспективних моделей міського управління.

Література

1. khaleel Khlewee, Ismael, Kassem Hamze, and Tirus Muya. "AI in smart cities: A review of urban data processing, prediction, and optimization techniques." ESTIDAMAA 2025 (2025): 29-38.

УДК 004.03

Г. Вовнянка; Д. Оныськів; А. Кривецький
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПЕРСПЕКТИВИ АНАЛІТИЧНОГО ОПРАЦЮВАННЯ ДАНИХ РОЗУМНИХ МІСТ

UDC 004.03

H. Vovnianka; D. Onyskiv; A. Kryvetskyi

PROSPECTS OF ANALYTICAL PROCESSING OF SMART CITIES DATA

Поява «розумних міст» призвела до утворення надлишку даних із різних міських джерел, що вимагає надійних методів для ефективної обробки цієї інформації. Розпізнавання іменованих інформаційних сутностей є критично важливим для вилучення значущої аналітичної інформації із цих потоків даних. Взаємодія з користувачами відіграє критично важливу роль у процесах дослідження даних, особливо в контекстах «розумних міст», де необхідно враховувати різноманітні потреби та наміри користувачів.

Останні досягнення в галузі мовних моделей діають змогу моделям швидко адаптуватися до запитів користувачів без великих наборів даних для конкретних завдань [1]. Однак, саме лише збільшення розміру моделі не гарантує точної інтерпретації запитів користувачів. Узгодження моделей з людськими запитами шляхом тонкого налаштування із використанням зворотного зв'язку від користувачів може значно підвищити задоволеність запитів та якість результатів. Проте, у контексті аналітики міського середовища існують значні виклики щодо приватності та інтеграції даних. Застосунки для аналітичного опрацювання даних у режимі реального часу повинні вирішувати проблеми конфіденційності, при цьому ефективно використовуючи цінні дані для вдосконалення застосувань великих даних орієнтованим на запити способом [2]. Безпека та конфіденційність у контекстно-залежних системах охорони здоров'я є невід'ємною частиною створення надійних та ефективних рішень у межах «розумних міст». Тому для вирішення цих взаємопов’язаних завдань необхідні програмно-алгоритмічні рішення, які надають пріоритет орієнтованим на користувача перспективам та ефективно інтегрують різноманітні системи.

Взаємопов'язані системи у міському середовищі мають на меті покращити управління через систематичне використання даних «розумних міст» [2]. Вирішення проблем масштабованості та доступності даних у межах цифрових двійників «розумних міст» може бути досягнуто за допомогою підходів із використанням синтетичних даних. Методи прогнозного моделювання, як от нейронні мережі, слугують вирішальними інструментами для прогнозування споживання електроенергії містом, інформуючи електроенергетичну промисловість про ключові тенденції споживання [3]. Крім того, системи реального часу дають змогу безпечно та ефективно управляти біометричними даними, підвищуючи можливості розподілених мереж.

Література

1. Naveed, Humza, et al. "A comprehensive overview of large language models." ACM Transactions on Intelligent Systems and Technology 16.5 (2025): 1-72.
2. Dahmane, Walid Miloud, Samir Ouchani, and Hafida Bouarfa. "Smart cities services and solutions: A systematic review." Data and Information Management 9.2 (2025): 100087.
3. Alhasnawi, Bilal Naji, et al. "The rising, applications, challenges, and future prospects of energy in smart grids and smart cities systems." Energy Conversion and Management: X (2025): 101162.

Додаток Б

Django Views для підключення телефоній і CRM

```
class ConnectKeepincrmView(APIView):
    permission_classes = [IsAuthenticated]

    def post(self, request, *args, **kwargs):
        if hasattr(request.user, 'crm_integration'):
            return Response(
                {
                    "status": "error",
                    "message": "Integration with this or another
CRM already exists. Please delete it first."
                },
                status=status.HTTP_400_BAD_REQUEST
            )

        data = request.data
        api_key = data.get('api_key')
        headers = {'X-Auth-Token': api_key}
        url = 'https://api.keepincrm.com/v1/clients'
        response_api = requests.get(url, headers=headers)

        if response_api.status_code == 200:
            integration = CRMIntegration(user=request.user,
service_name='keepincrm')
            integration.set_api_key('api_key', api_key)
            integration.save()
            return Response(
                {"status": "success", "message": "Integration
successful"},
                status=status.HTTP_200_OK
            )
        else:
            return Response(
                {
                    "status": "error",
                    "message": f"Invalid API key. KeepinCRM
responded with status {response_api.status_code}"
                },
                status=status.HTTP_400_BAD_REQUEST
            )


class ConnectPipedrivecrmView(APIView):
    permission_classes = [IsAuthenticated]

    def post(self, request, *args, **kwargs):
        if hasattr(request.user, 'crm_integration'):
            return Response(
                {
```

```
                    "status": "error",
                    "message": "Integration with this or another
CRM already exists. Please delete it first."
                },
                status=status.HTTP_400_BAD_REQUEST
            )

        data = request.data
        api_token = data.get('api_token')
        company_domain = data.get('company_domain')

        if not api_token or not company_domain:
            return Response(
                {
                    "status": "error",
                    "message": "Both API token and company domain
are required."
                },
                status=status.HTTP_400_BAD_REQUEST
            )

        url =
f'https://{company_domain}.pipedrive.com/api/v1/deals'
        params = {'api_token': api_token}

        try:
        response_api = requests.get(url, params=params)

        if response_api.status_code == 200:
            integration = CRMIntegration(user=request.user,
service_name='pipedrive')
            integration.set_api_key('api_key', api_token)
            integration.set_api_key('company_domain',
company_domain)
            integration.save()

            return Response(
                {"status": "success", "message": "Pipedrive
integration successful"},
                status=status.HTTP_200_OK
            )
        else:
            return Response(
                {
                    "status": "error",
                    "message": f"Invalid API credentials.
Pipedrive responded with status {response_api.status_code}"
                },
                status=status.HTTP_400_BAD_REQUEST
            )
        except requests.exceptions.RequestException as e:
            return Response(
                {
                    "status": "error",
                    "message": f"Connection error: {str(e)}"
```

```
                },
                status=status.HTTP_500_INTERNAL_SERVER_ERROR
            )


class ConnectBitrixcrmView(APIView):
    permission_classes = [IsAuthenticated]

    def post(self, request):

        return Response(
            {
                "status": "error",
                "message": "Integration with this or another CRM
already exists. Please delete it first."
            },
            status=status.HTTP_400_BAD_REQUEST
        )

        webhook_url_api = request.data.get('webhook_url_api')
        if not webhook_url_api:
            return Response(
                {'status': 'error', 'message': 'No webhook_url_api
provided'},
                status=status.HTTP_400_BAD_REQUEST
            )

        api_client = BitrixCRMAPI(webhook_url_api)

        if not api_client.check_api_creds():
            return Response(
                {"status": "error", "message": "Invalid webhook
URL or it do not have rights to access CRM API"},
                status=status.HTTP_400_BAD_REQUEST
            )

        integration = CRMIntegration(user=request.user,
service_name='bitrixcrm')
        integration.set_api_key('webhook_url_api',
webhook_url_api)
        integration.save()
        return Response(
            {"status": "success", "message": "BitrixCRM
integration successful"},
            status=status.HTTP_200_OK
        )


class ConnectHubspotcrmView(APIView):
    permission_classes = [IsAuthenticated]

    def post(self, request):
        if hasattr(request.user, 'crm_integration'):
            return Response(
                {
```

```
                    "status": "error",
                    "message": "Integration with this or another
CRM already exists. Please delete it first."
                },
                status=status.HTTP_400_BAD_REQUEST
            )

        code = request.data.get('code')
        if not code:
            return Response(
                {'status': 'error', 'message': 'No code
provided'},
                status=status.HTTP_400_BAD_REQUEST
            )

try:
            tokens = api_client.oauth.tokens_api.create(
                grant_type='authorization_code',
                redirect_uri=settings.HUBSPOT_REDIRECT_URI,
                client_id=settings.HUBSPOT_CLIENT_ID,
                client_secret=settings.HUBSPOT_CLIENT_SECRET,
                code=code
            )
        except ApiException as e:
            return Response(
                {"status": "error", "message": f"Unable to get
tokens"},
                status=status.HTTP_400_BAD_REQUEST
            )

        integration = CRMIntegration(user=request.user,
service_name='hubspotcrm')
        integration.set_api_key('access_token',
tokens.access_token)
        integration.set_api_key('refresh_token',
tokens.refresh_token)
        integration.save()
        return Response(
            {"status": "success", "message": "HubspotCRM
integration successful"},
            status=status.HTTP_200_OK
        )

class ConnectBinotelView(APIView):
    permission_classes = [IsAuthenticated]

    def post(self, request, *args, **kwargs):
        if hasattr(request.user, 'telephony_integration'):
            return Response(
                {
                    "status": "error",
                    "message": "Integration with this or another
telephony already exists. Please delete it first."
                },
                status=status.HTTP_400_BAD_REQUEST
```

```
            )

        data = request.data
        api_key = data.get('api_key')
        api_secret = data.get('api_secret')
        binotel_client = BinotelApi(api_key, api_secret)

        one_hour_ago = int(time.time()) - 3600
        response_api = binotel_client.send_request('stats/all-
outgoing-calls-since', {'timestamp': one_hour_ago})

        if response_api and isinstance(response_api, dict) and
response_api.get('status') == 'success':
            integration = TelephonyIntegration(user=request.user,
service_name='binotel')
            integration.set_api_key('api_key', api_key)
            integration.set_api_key('api_secret', api_secret)
            integration.save()

            # Upload calls and employees to db
            stop_time = int(time.time())

start_time = stop_time - settings.CALL_MAX_AGE
            interval_of_requests = 90 * 86400  # 90 days
            tasks.upload_binotel_calls_to_db.delay(
                request.user.id,
                api_key,
                api_secret,
                start_time,
                stop_time,
                interval_of_requests
            )

            tasks.upload_binotel_user_employees.delay(
                request.user.id,
                api_key,
                api_secret
            )

            return Response(
                {"status": "success", "message": "Integration
successful"},
                status=status.HTTP_200_OK
            )
        else:
            return Response(
                {
                    "status": "error",
                    "message": f"Invalid API keys"
                },
                status=status.HTTP_400_BAD_REQUEST
            )

class ConnectRingostatView(APIView):
    permission_classes = [IsAuthenticated]
```

```
    def post(self, request, *args, **kwargs):
        if hasattr(request.user, 'telephony_integration'):
            return Response(
                {
                    "status": "error",
                    "message": "Integration with this or another
telephony already exists. Please delete it first."
                },
                status=status.HTTP_400_BAD_REQUEST
            )
        data = request.data
        api_key = data.get('api_key')
        project_id = data.get('project_id')
        ringostat_client = RingostatAPI(api_key, project_id)

        if ringostat_client.check_api_key():
            integration = TelephonyIntegration(user=request.user,
 service_name='ringostat')
            integration.set_api_key('api_key', api_key)
            integration.set_api_key('project_id', project_id)
            integration.save()

            # Upload calls and employees to db

start_time = stop_time - settings.CALL_MAX_AGE
        tasks.upload_ringostat_calls_to_db.delay(
            request.user.id,
            start_time,
            stop_time,
        )

        tasks.upload_ringostat_user_employees.delay(
            request.user.id,
        )

        return Response(
            {"status": "success", "message": "Integration
successful"},
            status=status.HTTP_200_OK
        )
    else:
        return Response(
            {
                "status": "error",
                "message": f"Invalid API keys"
            },
            status=status.HTTP_400_BAD_REQUEST
        )

class TelephonyIntegrationView(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request, *args, **kwargs):
```

```
        integration = getattr(request.user,
'telephony_integration', None)
        if integration:
            return Response(
                {
                    "status": "success",
                    "data": {
                        "id": integration.id,
                        "service_name":
integration.get_service_name_display(),
                        "api_key":
integration.get_api_key('api_key')
                    }
                },
                status=status.HTTP_200_OK
            )
        return Response(
            {
                "status": "error",
                "message": "No telephony integration found"
            },
            status=status.HTTP_404_NOT_FOUND
        )

    def delete(self, request, *args, **kwargs):
        integration = getattr(request.user,
'telephony_integration', None)

        if integration:
            integration.delete()

           {"status": "success", "message": "Telephony integration
deleted"},
                        status=status.HTTP_200_OK
                    )
        return Response(
            {
                "status": "error",
                "message": "No telephony integration found"
            },
            status=status.HTTP_404_NOT_FOUND
        )


class CRMIntegrationView(APIView):
    def get(self, request, *args, **kwargs):
        integration = getattr(request.user, 'crm_integration',
None)
        if integration:
            return Response(
                {
                    "status": "success",
                    "data": {
                        "id": integration.id,
                        "service_name":
```

```
integration.get_service_name_display(),
                        "api_key":
integration.get_api_key('api_key'),
                        "webhook_url_api":
integration.get_api_key('webhook_url_api'),
                    }
                },
                status=status.HTTP_200_OK
            )
        return Response(
            {
                "status": "error",
                "message": "No CRM integration found"
            },
            status=status.HTTP_404_NOT_FOUND
        )

    def delete(self, request, *args, **kwargs):
        integration = getattr(request.user, 'crm_integration',
None)
        if integration:
            integration.delete()
            return Response(
                {"status": "success", "message": "CRM integration
deleted"},
                status=status.HTTP_200_OK
            )
        return Response(
            {
                "status": "error",
                "message": "No CRM integration found"
            },
            status=status.HTTP_404_NOT_FOUND
        )


class UserInstructionView(APIView):

def get(self, request, *args, **kwargs):
        user = request.user
        if not user.instruction:
            return Response(
                {
                    "status": "error",
                    "message": "No instruction added"
                },
                status=status.HTTP_404_NOT_FOUND
            )

        return Response(
            {
                "status": "success",
                "data": {
                    "instruction": user.instruction
                }
```

```
            },
            status=status.HTTP_200_OK
        )

    def post(self, request, *args, **kwargs):
        file_url = request.data.get('file_url')
        user = request.user

        google_docs_client = GoogleDocsAPI()
        check_result = google_docs_client.check_file_url(file_url)
        if not check_result['valid']:
            return Response(
                {
                    "status": "error",
                    "message": check_result['error']
                },
                status=status.HTTP_400_BAD_REQUEST
            )

        user.instruction = file_url
        user.save()
        return Response(
            {
                "status": "success",
                "message": "Instruction added"
            },
            status=status.HTTP_200_OK
        )

    def delete(self, request, *args, **kwargs):
        user = request.user
        user.instruction = None
        user.save()
        return Response(
            {
                "status": "success",
                "message": "Instruction deleted"
            },
            status=status.HTTP_200_OK
        )
```

Додаток В

Django Views для роботи із дзвінками

```
      class UserCallsView(generics.ListAPIView):
          permission_classes = [IsAuthenticated]
          serializer_class = CallSerializer
          pagination_class = CallsPagination
          filter_backends      =      [filters.DjangoFilterBackend,
AliasOrderingFilter]
          filterset_class = CallFilter
          ordering_fields = ['duration', 'date']
          ordering = ['-start_time']

          def get_queryset(self):
              analyzing_call_ids = [int(call_id) for call_id in

redis_client.smembers('analyzing_calls')]
              user_calls                                          =
Call.objects.filter(user=self.request.user)

              has_analysis = Exists(
                  CallAnalysis.objects.filter(call=OuterRef('pk'))
              )

              queryset = user_calls.annotate(
                  status=Case(
                      When(has_analysis, then=Value('analyzed')),
                      When(id__in=analyzing_call_ids,
then=Value('analyzing')),
                      default=Value('not_analyzed'),
                      output_field=CharField(),
                  )
              )

              return queryset


      class CallsAnalyzeView(APIView):
          def post(self, request, *args, **kwargs):
              calls_ids = request.data.get('calls_ids')
              user                                                =
User.objects.select_related('telephony_integration').get(id=reques
t.user.id)
              telephony_integration         =         getattr(user,
'telephony_integration', None)

              if not telephony_integration:
                  return Response(
                      {
                          "status": "error",
```

```
                        "message": "No telephony integration
found"
                    },
                    status=status.HTTP_400_BAD_REQUEST
                )
            if not calls_ids:
                return Response(
                    {
                        "status": "error",
                        "message": "calls_ids is required"
                    },
                    status=status.HTTP_400_BAD_REQUEST
                )

            user_calls = Call.objects.filter(
                user=user,
                id__in=calls_ids
            )

            analyzed_call_ids = CallAnalysis.objects.filter(
                call__in=user_calls,
                user=user
            ).values_list('call_id', flat=True)

            valid_calls                                          =
user_calls.exclude(id__in=analyzed_call_ids)

            if not valid_calls:
                return Response(
                    {
                        "status": "info",
                        "message": "No valid calls to analyze"
                    },
                    status=status.HTTP_400_BAD_REQUEST
                )

            telephony_services   =   [key   for   key,   _   in
TelephonyIntegration.SERVICES]
            if       telephony_integration.service_name       in
telephony_services:
                for call in valid_calls:
                    analyze_call.delay(
                        user.id,
                        call.id,
                        call.api_call_id,                )
                return Response({"status": "success"},
                                status=status.HTTP_200_OK)
            else:
                return Response(
                    {
                        "status": "error",
                        "message":   "Not   supported   telephony
integration"                  },
                    status=status.HTTP_400_BAD_REQUEST
                )
```