https://doi.org/10.31891/2219-9365-2026-86-53
УДК 004.8:004.912:37.018.43

ЯЦЕНКО Роман
Харківський національний економічний університет імені Семена Кузнеця
https://orcid.org/0000-0001-7968-6890
roman.yatsenko@hneu.net
ЗАМУРА Дмитро
Харківський національний економічний університет імені Семена Кузнеця
https://orcid.org/0009-0009-4436-9647
dmytro.zamura@hneu.net

РОЗРОБКА ТА ВПРОВАДЖЕННЯ ІНТЕЛЕКТУАЛЬНОГО RAG-ЧАТ-БОТА ДЛЯ ВЕБРЕСУРСІВ УНІВЕРСИТЕТУ В РАМКАХ РОЗБУДОВИ ЦИФРОВОЇ ІНКЛЮЗИВНОЇ ОСВІТНЬОЇ ЕКОСИСТЕМИ

У статті представлено архітектуру, технічну реалізацію та результати впровадження інтелектуального чат-бота на базі технології RAG (Retrieval-Augmented Generation) до вебресурсів Харківського національного економічного університету імені Семена Кузнеця. Робота виконана в рамках комплексного прикладного наукового дослідження «Розробка відкритої цифрової інклюзивної освітньої екосистеми університету для забезпечення безперервності вищої освіти в Україні», що фінансується за рахунок видатків загального фонду державного бюджету. Детально описано багатошарову архітектуру системи за моделлю C4, включаючи двоетапну систему маршрутизації запитів (Topic-based Routing), використання патерну ReAct Agent за допомогою фреймворка LangGraph, конвеєр збору та індексації даних (Data Ingestion Pipeline), а також особливості інтеграції з векторною базою даних Pinecone та великими мовними моделями OpenAI. Обґрунтовано ключові архітектурні рішення, зокрема асинхронну обробку запитів, тематичну фільтрацію семантичного пошуку та розділення відповідальності на рівні Django-додатків. Запропонований підхід забезпечує масштабованість, контрольованість генерації відповідей та зниження бар'єрів доступу до інформації для всіх категорій учасників освітнього процесу, включаючи осіб з особливими освітніми потребами.

Ключові слова: штучний інтелект в освіті, RAG-системи, великі мовні моделі, LangGraph, Pinecone, семантичний пошук, чат-бот, цифрова екосистема, інклюзивна освіта, Topic-based Routing.

YATSENKO Roman, ZAMURA Dmytro
Simon Kuznets Kharkiv National University of Economics

DEVELOPMENT AND IMPLEMENTATION OF AN INTELLIGENT RAG CHATBOT FOR UNIVERSITY WEB RESOURCES WITHIN THE FRAMEWORK OF BUILDING A DIGITAL INCLUSIVE EDUCATIONAL ECOSYSTEM

The article presents the architecture, technical implementation, and deployment results of an intelligent chatbot based on Retrieval-Augmented Generation (RAG) technology integrated into the web resources of the Simon Kuznets Kharkiv National University of Economics. The study was carried out within the framework of the comprehensive applied research project "Development of an Open Digital Inclusive Educational Ecosystem of a University to Ensure the Continuity of Higher Education in Ukraine," funded by the general state budget. The paper provides a detailed description of the system's multilayer architecture based on the C4 model, including a two-stage query routing mechanism (Topic-based Routing), the implementation of the ReAct Agent pattern using the LangGraph framework, a Data Ingestion Pipeline for data collection and indexing, and the integration of the Pinecone vector database with OpenAI large language models. Key architectural decisions are substantiated, including asynchronous request processing, topic-based filtering of semantic search results, and separation of responsibilities at the Django application level. The proposed approach ensures system scalability, controllability of response generation, and reduced barriers to information access for all categories of educational process participants, including individuals with special educational needs.

Keywords: artificial intelligence in education, RAG systems, large language models, LangGraph, Pinecone, semantic search, chatbot, digital ecosystem, inclusive education, Topic-based Routing.

Стаття надійшла до редакції / Received 07.04.2026
Прийнята до друку / Accepted 30.04.2026
Опубліковано / Published 31.05.2026

1. ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Сучасний ландшафт вищої освіти в Україні характеризується комплексом безпрецедентних викликів: повномасштабна агресія, масове переміщення студентів та науково-педагогічних працівників, необхідність забезпечення безперервності освітнього процесу в умовах постійних загроз, а також зростаючі вимоги до цифрової трансформації закладів вищої освіти (ЗВО) [1]. У цих умовах штучний інтелект (ШІ) стає не просто перспективним інструментом, а стратегічною необхідністю для забезпечення якості та доступності освіти.

Потенціал застосування ШІ в освіті є багатовимірним і охоплює кілька ключових напрямків. По-перше, *персоналізація навчання* – ШІ-системи здатні аналізувати індивідуальні патерни навчальної

активності, успішність та стиль засвоєння матеріалу для формування адаптивних освітніх траєкторій [2]. По-друге, *автоматизація рутинних процесів* – від оцінювання типових завдань до генерації зворотного зв'язку, що дозволяє викладачам зосередитися на якісно складніших аспектах педагогічної діяльності [3]. По-третє, *інтелектуальна підтримка студентів* – ШІ-асистенти забезпечують цілодобовий доступ до консультацій, знижуючи навантаження на адміністрацію та деканати [4]. По-четверте, *аналітика навчального процесу* – обробка великих масивів даних про взаємодію здобувачів освіти з інформаційними ресурсами для виявлення проблемних зон та прогнозування академічних ризиків [5].

В рамках державного бюджетного фінансування науковцями та викладачами Навчально-наукового інституту інформаційних технологій Харківського національного економічного університету імені Семена Кузнеця (ХНЕУ ім. С. Кузнеця) виконується прикладне наукове дослідження за темою «Розробка відкритої цифрової інклюзивної освітньої екосистеми університету для забезпечення безперервності вищої освіти в Україні». Проєкт спрямований на створення комплексної цифрової інфраструктури, що інтегрує інструменти ШІ, адаптивні технології та інклюзивні рішення для забезпечення рівного доступу до освіти незалежно від фізичного місцезнаходження та особливих потреб здобувачів [6].

Одним із ключових завдань проєкту є впровадження ШІ-компаньйонів для підтримки студентів та абітурієнтів у взаємодії з інформаційними ресурсами університету. Вебсайт ЗВО є основним джерелом інформації для потенційних та поточних здобувачів освіти, проте обсяг та структурна складність розміщених матеріалів (навчальні плани, нормативні документи, новини, розклад, умови вступу тощо) створюють значні бар'єри в ефективному пошуку релевантної інформації. Це є особливо критичним для осіб з особливими освітніми потребами, для яких навігація складними вебінтерфейсами може становити додаткову перешкоду [7].

Інтелектуальний чат-бот на базі технології RAG (Retrieval-Augmented Generation) [8] здатний вирішити зазначені проблеми шляхом надання природномовного інтерфейсу для взаємодії із базою знань університету. На відміну від традиційних чат-ботів, побудованих на системах правил або FAQ-базах, RAG-системи поєднують семантичний пошук в базі знань з генеративними можливостями великих мовних моделей (LLM), що забезпечує контекстуально точні та інформативні відповіді з опорою на фактичні дані [9].

Метою даної статті є опис архітектурних рішень, технічних деталей реалізації та результатів впровадження інтелектуального RAG-чат-бота *HNEU Agent System* для вебресурсів ХНЕУ ім. С. Кузнеця.

2. ОГЛЯД ПОВ'ЯЗАНИХ ДОСЛІДЖЕНЬ

2.1. RAG-системи: принципи та еволюція

Парадигма Retrieval-Augmented Generation була запропонована Lewis et al. [8] як підхід, що поєднує переваги параметричної пам'яті LLM з непараметричним пошуком у зовнішніх базах знань. Класична RAG-архітектура передбачає два етапи: (1) retrieval – семантичний пошук релевантних фрагментів документів за запитом користувача у векторному сховищі; (2) generation – формування відповіді мовною моделлю з урахуванням знайденого контексту. Порівняно з підходом fine-tuning, RAG-системи мають суттєві переваги: можливість оновлення бази знань без перенавчання моделі, прозорість джерел інформації та зменшення проблеми «галюцинацій» LLM [10].

Сучасний розвиток RAG-систем характеризується переходом від простих конвеєрів пошуку до більш складних агентних архітектур. Зокрема, Advanced RAG [11] включає техніки попередньої обробки запитів (query rewriting, HyDE), реранжування результатів та ітеративного уточнення пошуку. Agentic RAG [12] виходить за межі лінійного конвеєра, наділяючи систему здатністю самостійно приймати рішення про необхідність пошуку, вибір інструментів та стратегію відповіді.

2.2. Агентні архітектури на базі LLM

Патерн ReAct (Reasoning + Acting) [13] є одним з найбільш ефективних підходів до побудови LLM-агентів. ReAct-агент чергує етапи міркування (Thought) та дії (Action), що дозволяє моделі обґрунтовувати свої рішення і використовувати зовнішні інструменти для отримання необхідної інформації. Фреймворк LangGraph [14] реалізує цей патерн у вигляді скінченного автомата (state machine), де кожен вузол графа відповідає за конкретний етап обробки, а переходи між вузлами визначаються станом системи та рішеннями LLM.

2.3. ШІ-чат-боти у вищій освіті

Впровадження чат-ботів у ЗВО є предметом активних досліджень у світовій практиці. Okonkwo та Ade-Ibijola [4] систематизували застосування ШІ-чат-ботів в освіті, виділивши їх ключові функції: навчальна підтримка, адміністративна допомога, консультування абітурієнтів та моніторинг залученості студентів. Дзюбан [15] описав практичний досвід побудови RAG-системи для автоматизації служби підтримки з використанням OpenAI та Pinecone в умовах українського IT-ринку.

Водночас кількість досліджень щодо впровадження RAG-систем саме у вищих навчальних закладах України залишається обмеженою, що обумовлює актуальність та новизну представленої роботи.

3. АРХІТЕКТУРА СИСТЕМИ HNEU AGENT SYSTEM

Для забезпечення максимальної релевантності, достовірності відповідей та масштабованості системи було обрано архітектуру RAG з агентним оркестратором. Система, названа *HNEU Agent System*, побудована за багаторівневою модульною архітектурою з використанням сучасного стека відкритих технологій.

Для проєктування та документування архітектури було застосовано модель C4 (Context, Containers, Components, Code) [16], яка забезпечує структурований опис системи на чотирьох рівнях абстракції – від загального контексту до деталей реалізації.

3.1. Рівень системного контексту (System Context)

На найвищому рівні абстракції система взаємодіє з чотирма типами зовнішніх акторів та сервісів (рис. 1).

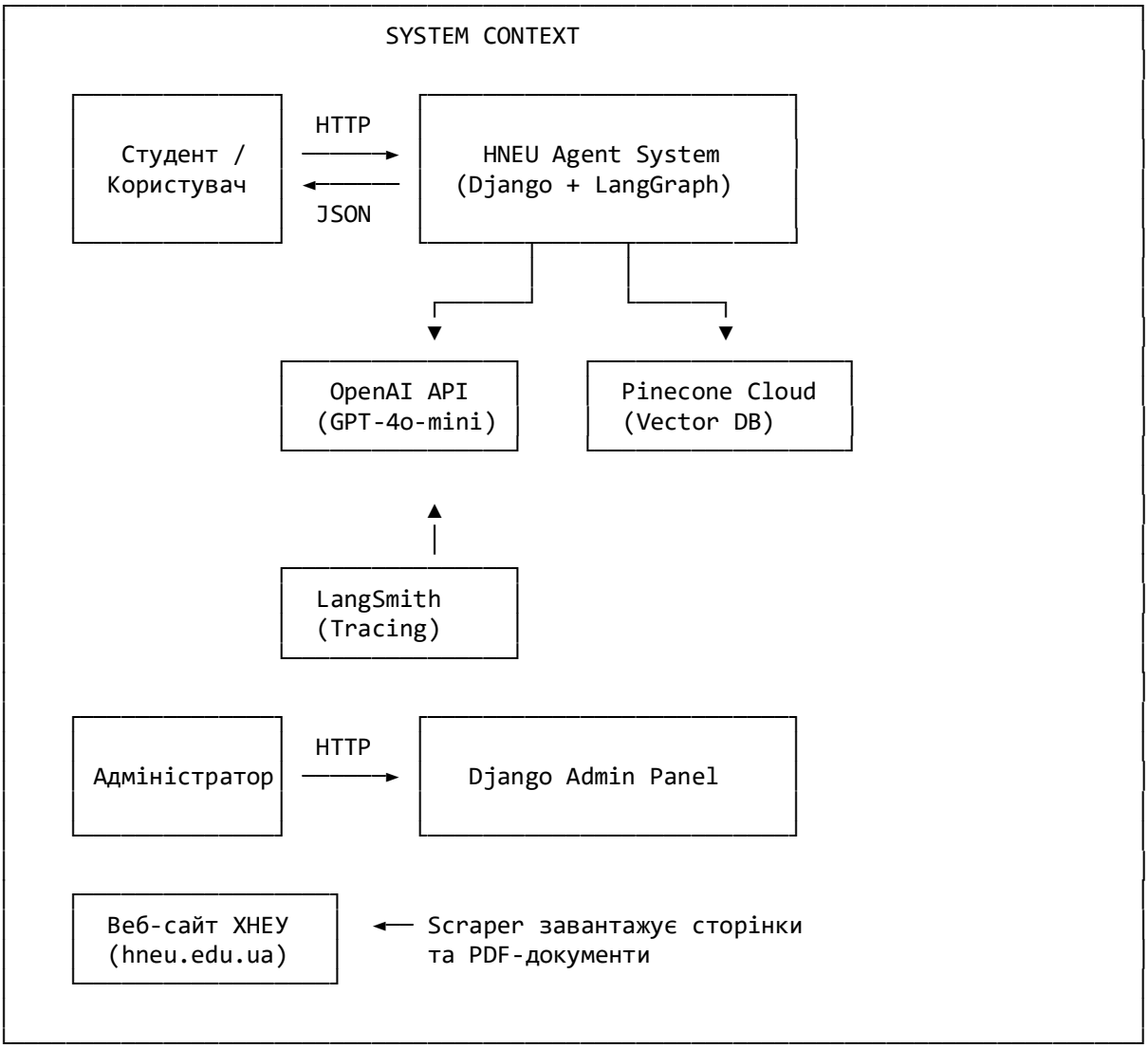


Рис. 1. Діаграма системного контексту HNEU Agent System (C4 Level 1)

Основними акторами системи є:

Студент / Користувач – взаємодіє з чат-ботом через JavaScript-віджет, інтегрований на вебсторінках університету, надсилаючи природномовні запити та отримуючи відповіді у форматі JSON через REST API.

Адміністратор – керує джерелами даних (URL-адресами вебсторінок та PDF-документів), тематичними тегами та системними промптами через панель управління Django Admin.

OpenAI API – зовнішній хмарний сервіс, що надає доступ до LLM GPT-4o-mini для генерації відповідей та класифікації тем, а також до моделі text-embedding для формування векторних представлень текстових фрагментів.

Pinecone Cloud – керована хмарна векторна база даних для зберігання та семантичного пошуку по корпусу знань ХНЕУ.

LangSmith – платформа моніторингу та трейсингу LLM-викликів для контролю якості,

відлагодження та збору аналітичних даних про взаємодію користувачів із системою.

Веб-сайт ХНЕУ (hneu.edu.ua) – основне джерело даних, з якого скрапер автоматично завантажує та індексує контент.

3.2. Рівень контейнерів (Container Diagram)

На другому рівні C4-моделі відображено внутрішню структуру HNEU Agent System, що включає кілька логічних контейнерів у межах єдиного Django-додатку (рис. 2).

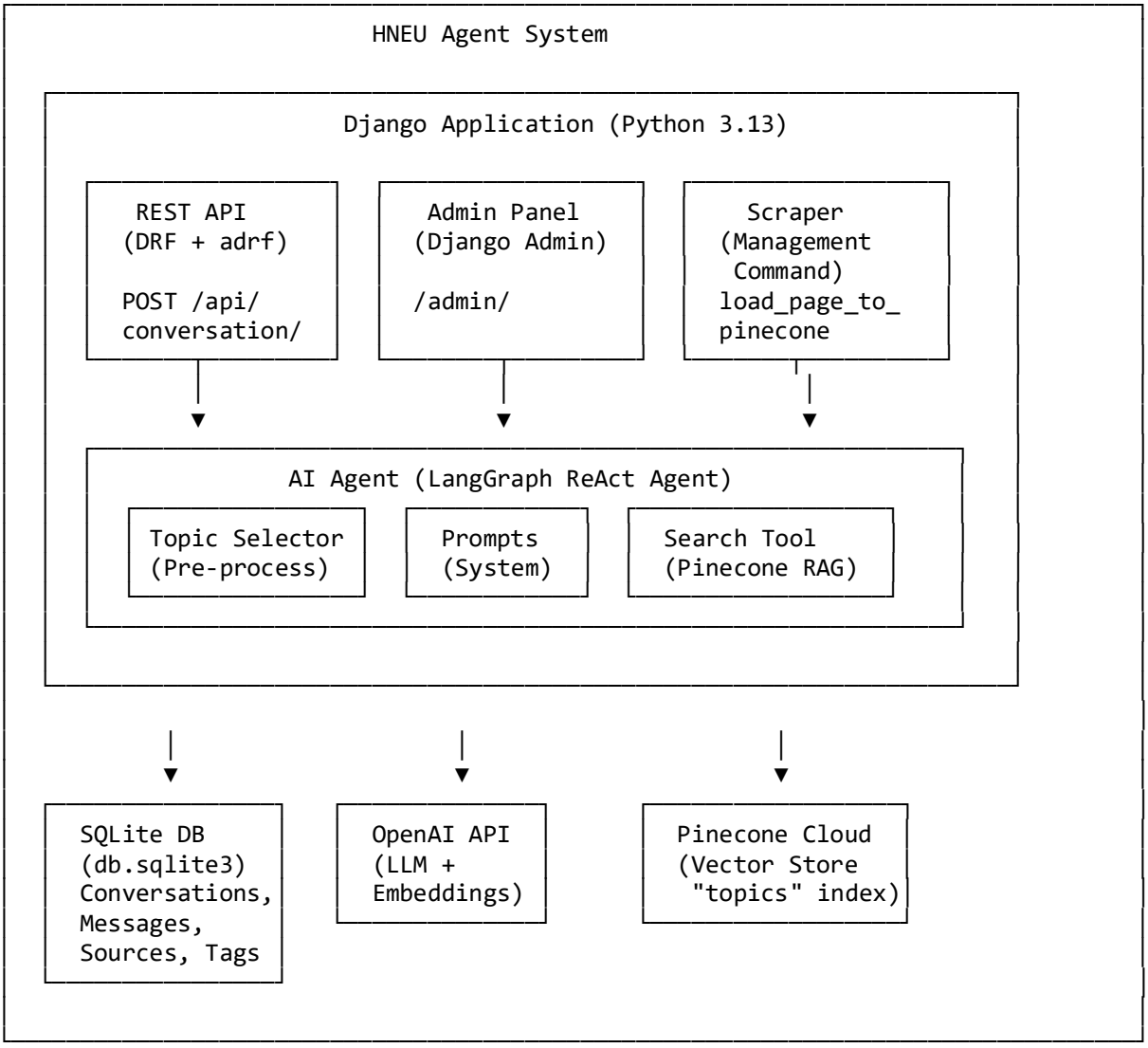


Рис. 2. Діаграма контейнерів HNEU Agent System (C4 Level 2)

Система побудована на базі Django Application (Python 3.13) і складається з наступних логічних контейнерів:

REST API (DRF + adrf) – асинхронний REST-інтерфейс на базі Django REST Framework з розширенням adrf (async DRF views). Забезпечує єдину кінцеву точку `POST /api/conversation/` для прийому повідомлень від клієнтського віджета та повернення відповідей агента. Використання асинхронних views є принциповим рішенням, оскільки обробка кожного запиту включає множинні I/O-bound операції (звернення до OpenAI API, Pinecone, SQLite), і блокуюча модель обробки суттєво обмежувала б пропускну здатність сервера [17].

Admin Panel (Django Admin) – панель адміністрування на базі стандартного Django Admin з розширенням django-import-export для масового імпорту та експорту джерел даних. Через цей інтерфейс адміністратор керує переліком URL-адрес для скрапінгу, тематичними тегами (з додатковими промптами для кожного тегу), а також іншими параметрами системи.

Scraper (Management Command) – модуль збору та індексації даних, реалізований як Django Management Command `load_page_to_pinecone`. Забезпечує автоматизований конвеєр завантаження, обробки та збереження контенту вебсайту ХНЕУ у векторній базі Pinecone.

AI Agent (LangGraph ReAct Agent) – ядро інтелектуальної системи, що включає модуль попередньої класифікації запитів (Topic Selector), набір системних промптів та інструмент семантичного пошуку в Pinecone.

SQLite DB – реляційна база даних для зберігання конверсацій (історії діалогів), повідомлень, переліку джерел, тегів з промптами та інших оперативних даних. Вибір SQLite обумовлений простотою розгортання та достатньою продуктивністю для поточних навантажень.

Таблиця 1 узагальнює технологічний стек контейнерів системи.

Таблиця 1.

Технологічний стек контейнерів HNEU Agent System

Контейнер	Технологія	Призначення
REST API	Django REST Framework 3.17 + adrf 0.1.9	Асинхронна обробка запитів користувачів
Admin Panel	Django Admin + django-import-export 4.4.0	Управління джерелами, тегами, промптами
Scraper	Django Management Command + LangChain loaders	Завантаження та індексація контенту
AI Agent	LangGraph 1.1.6 + LangChain 1.2.15 + OpenAI GPT-4o-mini	ReAct-агент із семантичним пошуком
SQLite DB	SQLite3	Зберігання діалогів, джерел, тегів
Vector Store	Pinecone 7.3.0	Семантичний пошук по базі знань

3.3. Рівень компонентів (Component Diagram)

На третьому рівні C4-моделі деталізовано внутрішню структуру Django-додатку, який складається з трьох Django Apps із чітким розділенням відповідальності (Separation of Concerns) (рис. 3).

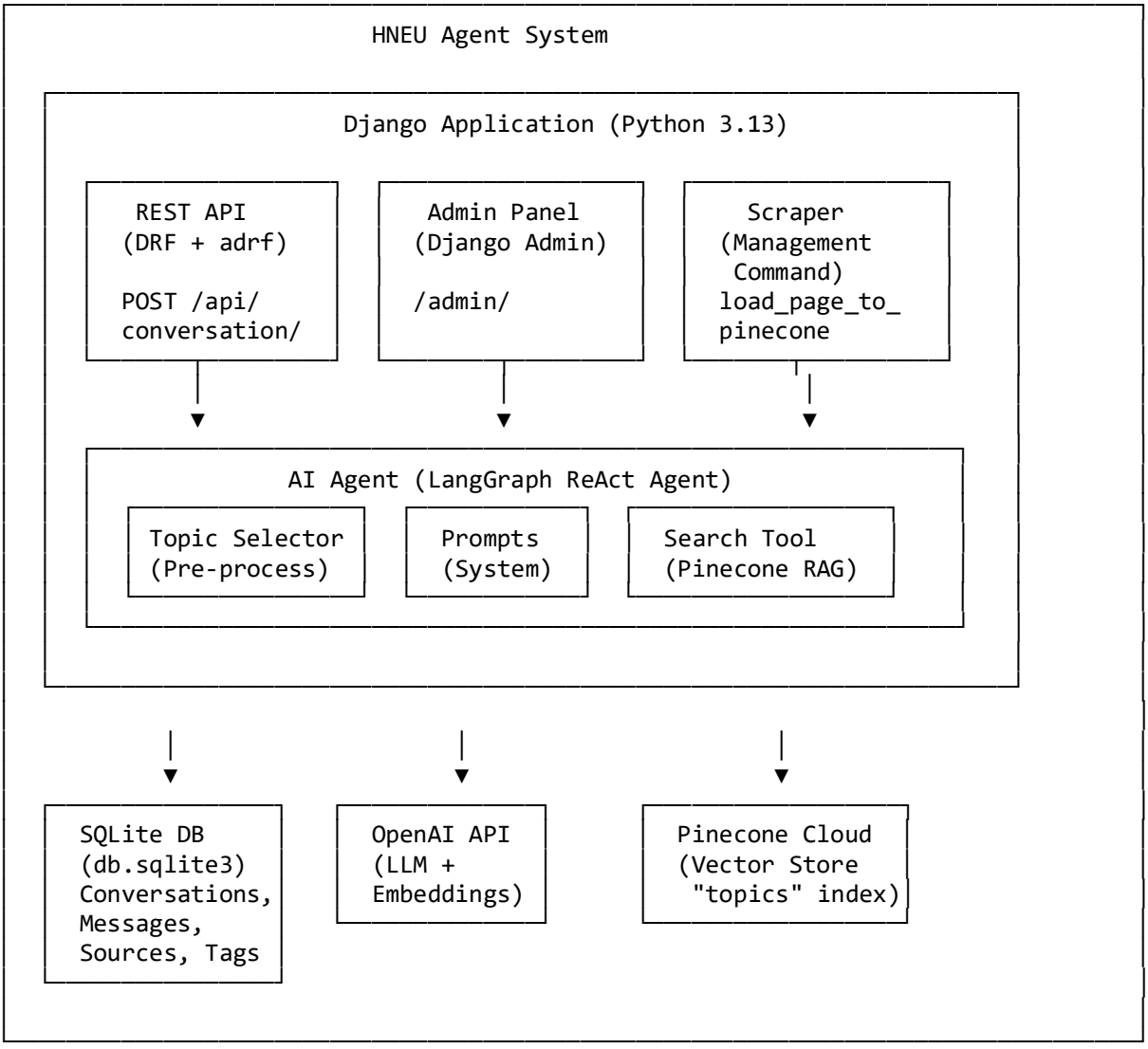


Рис. 2. Діаграма контейнерів HNEU Agent System (C4 Level 2)

Система побудована на базі Django Application (Python 3.13) і складається з наступних логічних контейнерів:

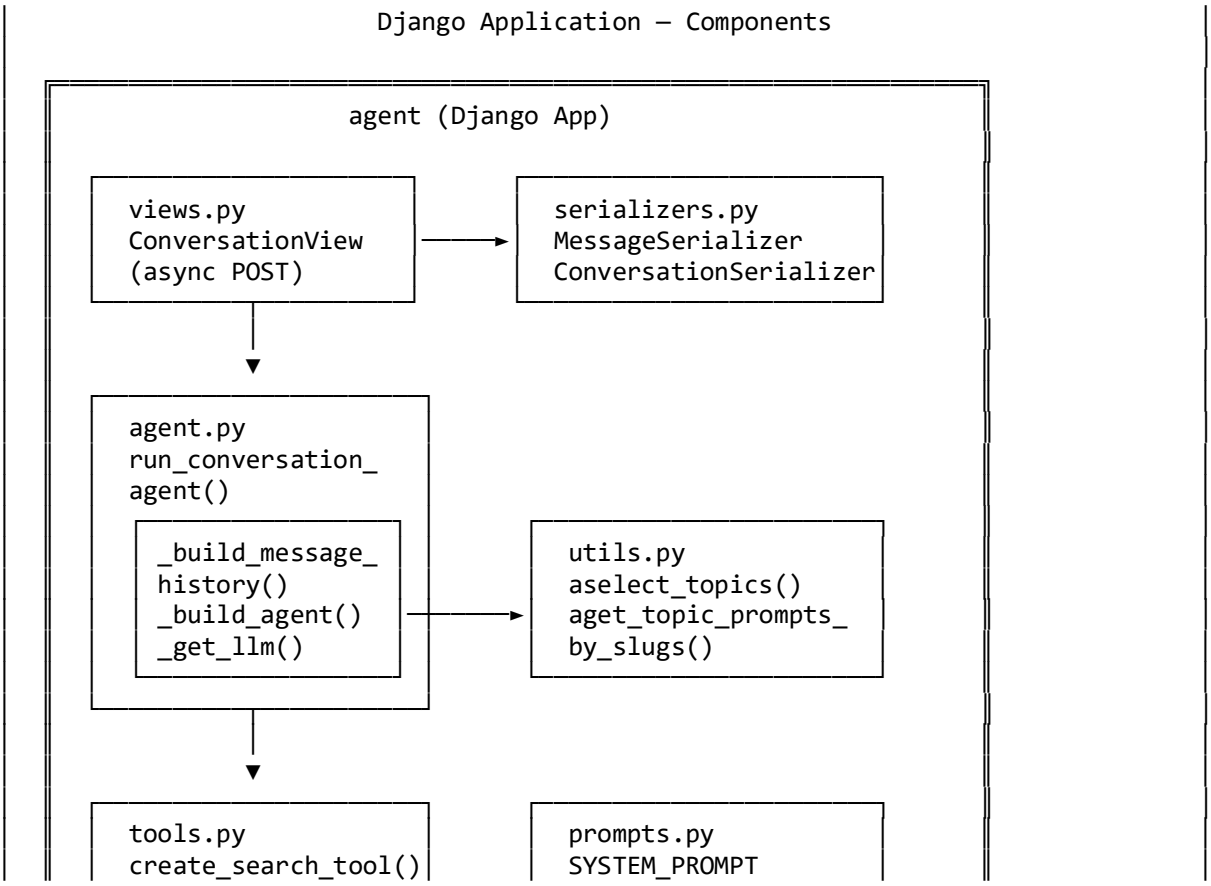
1. **REST API (DRF + adrf)** – асинхронний REST-інтерфейс на базі Django REST Framework з розширенням adrf (async DRF views). Забезпечує єдину кінцеву точку POST /api/conversation/ для прийому повідомлень від клієнтського віджета та повернення відповідей агента. Використання асинхронних views є принциповим рішенням, оскільки обробка кожного запиту включає множинні I/O-bound операції (звернення до OpenAI API, Pinecone, SQLite), і блокуюча модель обробки суттєво обмежувала б пропускну здатність сервера [17].
2. **Admin Panel (Django Admin)** – панель адміністрування на базі стандартного Django Admin з розширенням django-import-export для масового імпорту та експорту джерел даних. Через цей інтерфейс адміністратор керує переліком URL-адрес для скрапінгу, тематичними тегами (з додатковими промптами для кожного тегу), а також іншими параметрами системи.
3. **Scraper (Management Command)** – модуль збору та індексації даних, реалізований як Django Management Command load_page_to_pinecone. Забезпечує автоматизований конвеєр завантаження, обробки та збереження контенту вебсайту ХНЕУ у векторній базі Pinecone.
4. **AI Agent (LangGraph ReAct Agent)** – ядро інтелектуальної системи, що включає модуль попередньої класифікації запитів (Topic Selector), набір системних промптів та інструмент семантичного пошуку в Pinecone.
5. **SQLite DB** – реляційна база даних для зберігання конверсацій (історії діалогів), повідомлень, переліку джерел, тегів з промптами та інших оперативних даних. Вибір SQLite обумовлений простотою розгортання та достатньою продуктивністю для поточних навантажень.

Таблиця 1 узагальнює технологічний стек контейнерів системи.

Таблиця 1.

Технологічний стек контейнерів HNEU Agent System

Контейнер	Технологія	Призначення
REST API	Django REST Framework 3.17 + adrf 0.1.9	Асинхронна обробка запитів користувачів
Admin Panel	Django Admin + django-import-export 4.4.0	Управління джерелами, тегами, промптами
Scraper	Django Management Command + LangChain loaders	Завантаження та індексація контенту
AI Agent	LangGraph 1.1.6 + LangChain 1.2.15 + OpenAI GPT-4o-mini	ReAct-агент із семантичним пошуком



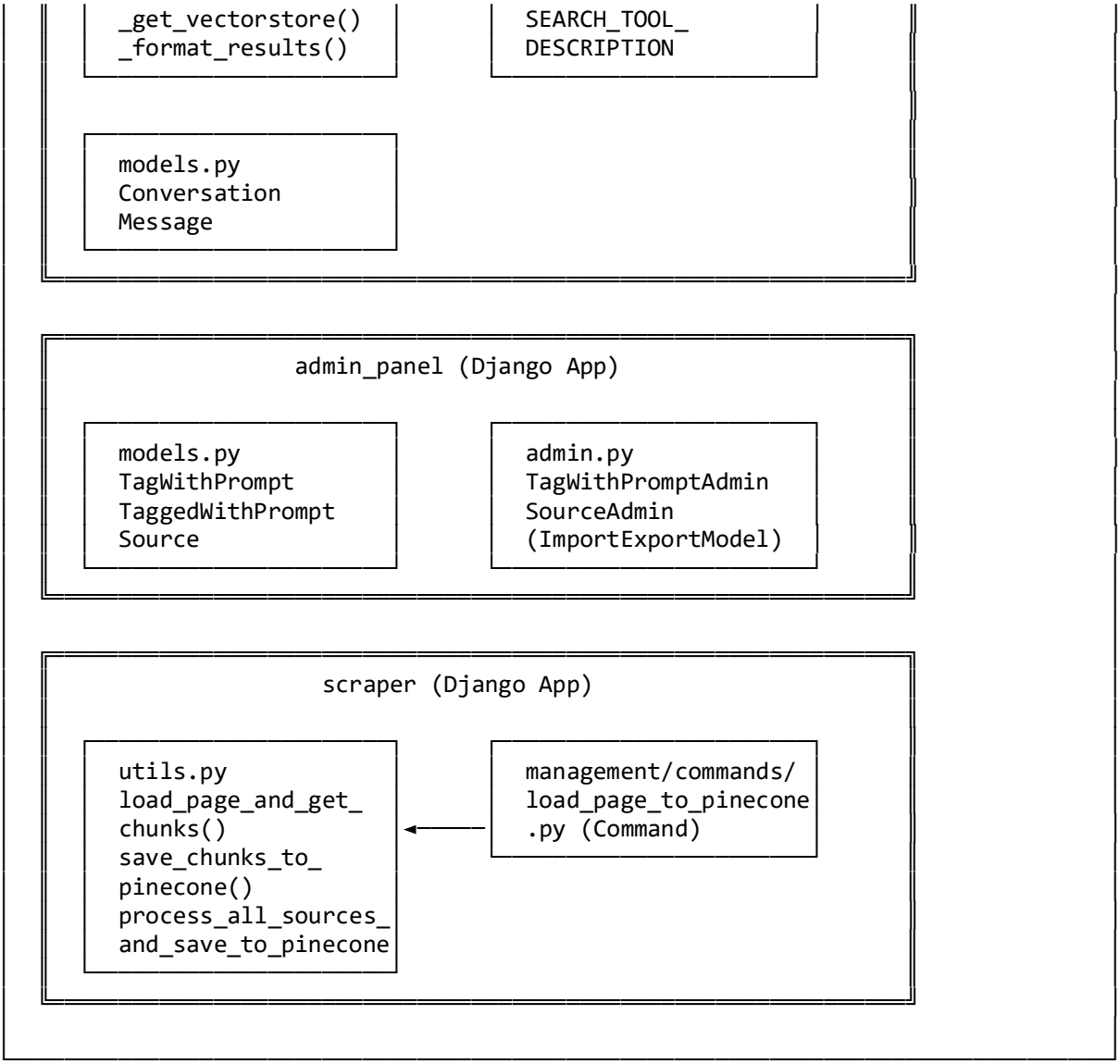


Рис. 3. Діаграма компонентів Django-додатку (C4 Level 3)

3.3.1. Agent App – ядро інтелектуальної обробки

Центральний додаток `agent` відповідає за повний цикл обробки запитів користувача та включає наступні компоненти:

ConversationView (`views.py`) – асинхронний REST endpoint, що приймає POST-запити з повідомленням користувача та ідентифікатором розмови (UUID), передає їх на обробку агенту та повертає серіалізовану відповідь через `MessageSerializer` та `ConversationSerializer`.

run_conversation_agent() (`agent.py`) – головна функція оркестрації, що координує послідовність кроків: побудова історії повідомлень → класифікація тем → компонування системного промпту → побудова та запуск ReAct-агента.

aselect_topics() (`utils.py`) – асинхронна функція попередньої класифікації запиту на тематичні категорії за допомогою швидкого LLM-виклику, результатом якого є набір slug-ідентифікаторів тем.

aget_topic_prompts_by_slugs() (`utils.py`) – асинхронне отримання додаткових промптів, прив'язаних до виявлених тем, з бази даних SQLite.

create_search_tool() (`tools.py`) – фабрична функція для створення LangChain Tool, що інкапсулює семантичний пошук у Pinecone з фільтрами за обраними тематичними тегами.

SYSTEM_PROMPT (`prompts.py`) – системний промпт, що визначає роль, обмеження та стиль відповідей ШІ-асистента ХНЕУ.

Conversation / Message (`models.py`) – ORM-моделі для персистентного зберігання історії діалогів.

3.3.2. Admin Panel App – управління базою знань

Додаток `admin_panel` забезпечує інтерфейс адміністрування:

TagWithPrompt – розширена модель тегу (на базі django-taggit), що окрім стандартних полів (name, slug) містить додатковий текстовий prompt, який автоматично додається до системного промпту агента при виявленні відповідної теми. Це дозволяє гнучко налаштовувати поведінку системи для різних тематичних доменів.

Source – модель джерела даних, що містить URL-адресу, набір тегів (через TaggedWithPrompt) та прапорець `should_be_scraped` для керування черговістю індексації.

SourceAdmin – налаштований адмін-інтерфейс з підтримкою масового імпорту/експорту через django-import-export, що спрощує пакетне управління великою кількістю джерел.

3.3.3. Scraper App – конвеєр збору та індексації даних

Додаток `scraper` реалізує автоматизований конвеєр Data Ingestion:

load_page_and_get_chunks() – функція завантаження контенту вебсторінки (через `WebBaseLoader` із BeautifulSoup4) або PDF-документа (через `PyPDFLoader`) та його розбиття на текстові фрагменти за допомогою `RecursiveCharacterTextSplitter` з параметрами chunk_size=1000 символів та chunk_overlap=200 символів.

save_chunks_to_pinecone() – функція формування ембедингів через OpenAI text-embedding та їх збереження у Pinecone з метаданими, що включають тематичні теги джерела (поле `topics`).

process_all_sources_and_save_to_pinecone() – пакетна обробка всіх джерел з прапорцем `should_be_scraped=True`.

load_page_to_pinecone – Django Management Command для запуску повного циклу скрапінгу з командного рядка.

4. ІНТЕЛЕКТУАЛЬНА ОБРОБКА ЗАПИТІВ

4.1. Двоетапна маршрутизація запитів (Topic-based Routing)

Головною технологічною особливістю розробленої системи є двоетапна обробка природномовних запитів з використанням патерну *Topic-based Routing*, що забезпечує тематично-фільтрований семантичний пошук.

Етап 1: Попередня класифікація. При надходженні запиту користувача функція `aselect_topics()` виконує швидкий та економічний LLM-виклик (з мінімальним промптом та обмеженим набором вихідних токенів) для визначення тематичних категорій запиту. LLM отримує список доступних тематичних slug-ідентифікаторів (наприклад, `vstup`, `rozklad`, `stipendiyi`, `normatyvna-baza`) та повертає підмножину тих, що відповідають запиту. Наприклад, запит «Які документи потрібні для вступу на магістратуру?» буде класифіковано з тегами `["vstup", "normatyvna-baza"]`.

Етап 2: Фільтрований семантичний пошук. На основі визначених тем система створює інструмент пошуку (`create_search_tool()`), що виконує similarity search у Pinecone з фільтром `filter={"topics": {"$in": selected_topics}}`. Це обмежує простір пошуку лише чанками, що належать до релевантних тематичних кластерів, і має кілька переваг: (1) суттєве підвищення точності та релевантності результатів за рахунок виключення нерелевантного контенту; (2) зменшення кількості «шумових» фрагментів у контексті LLM; (3) зниження витрат на обробку через зменшення обсягу переданого контексту.

Цей підхід концептуально подібний до metadata filtering у RAG-системах [11], проте з використанням LLM як класифікатора замість rule-based або keyword-based підходів, що забезпечує більшу гнучкість при обробці різноманітних формулювань запитів.

4.2. ReAct Agent на базі LangGraph

У якості оркестратора системи обрано фреймворк LangGraph [14], що дозволяє реалізувати ReAct Agent Pattern [13] у вигляді графа станів. Ключовою перевагою цього підходу порівняно з лінійним RAG-конвеєром є *автономність прийняття рішень*: агент самостійно визначає, чи потрібно виконувати пошук у базі знань для відповіді на конкретний запит.

Сценарії роботи агента включають: - **Пошук у базі знань** – для фактичних запитань про університет (умови вступу, розклад, нормативні документи), тобто для запитів, що потребують актуальної та верифікованої інформації. - **Відповідь з контексту діалогу** – для уточнюючих питань, що спираються на попередні повідомлення у розмові, історія якої зберігається в SQLite та передається агенту при кожному зверненні. - **Відмова від відповіді** – для запитів, що виходять за межі компетенції ШІ-асистента ХНЕУ (відповідно до обмежень, зафіксованих у системному промпті).

У якості базової моделі генерації використовується GPT-4o-mini від OpenAI, що забезпечує оптимальне співвідношення якості генерації, швидкості відповіді та вартості API-викликів для задач інформаційної підтримки.

4.3. Потік обробки запитів

Повний цикл обробки запиту користувача від отримання повідомлення до повернення відповіді

представлено на діаграмі послідовності (рис. 4).

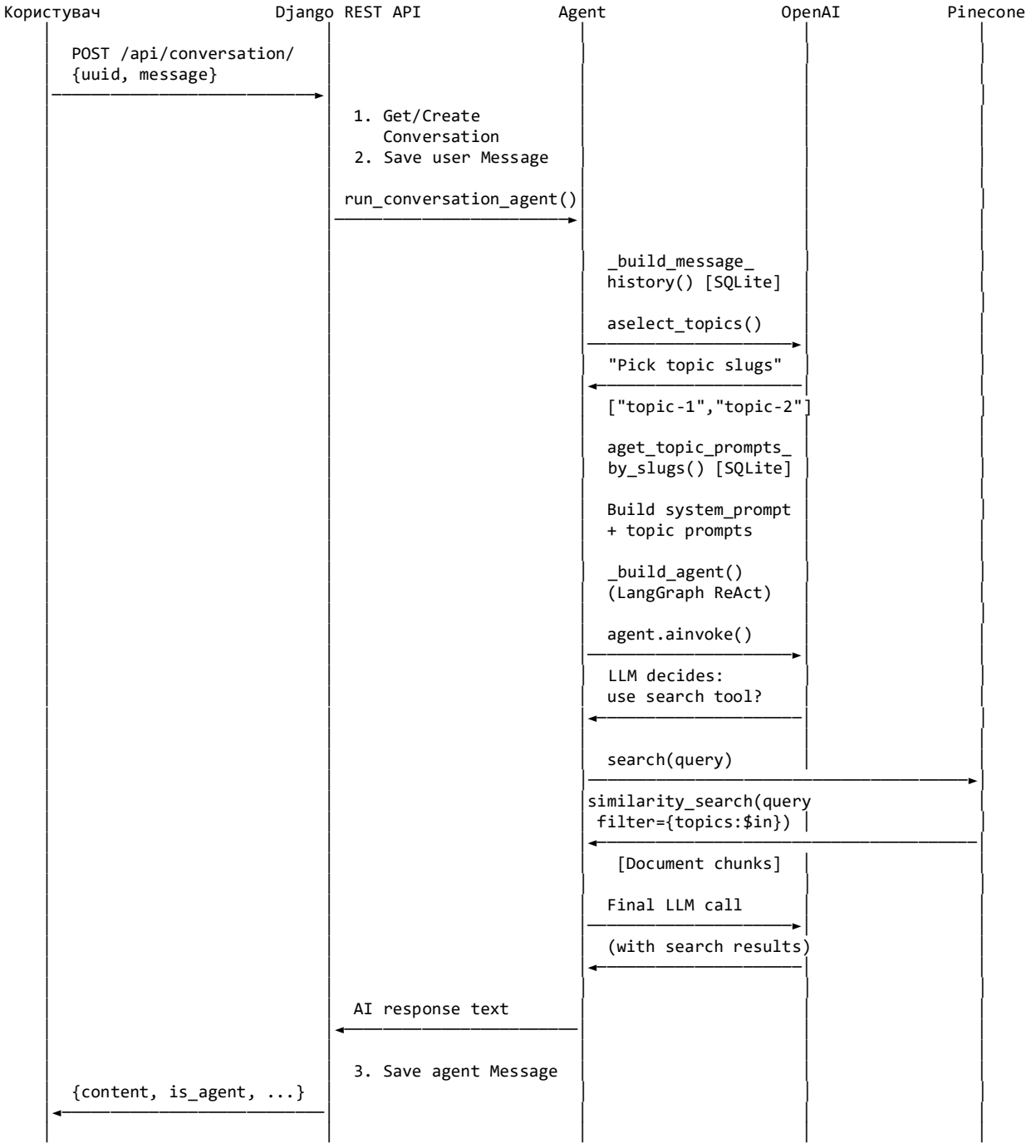


Рис. 4. Діаграма послідовності обробки запиту користувача (C4 Level 4)

Процес обробки включає наступні кроки: 1. Клієнтський JavaScript-віджет надсилає POST-запит з UUID розмови та текстом повідомлення до `ConversationView`. 2. REST API створює або знаходить існуючу розмову (Conversation) та зберігає повідомлення користувача (Message) у SQLite. 3. Функція `run_conversation_agent()` ініціює обробку: (a) побудова історії повідомлень з БД; (b) класифікація запиту на теми через LLM; (c) отримання додаткових промптів для виявлених тем; (d) компонування фінального системного промпту. 4. LangGraph ReAct Agent (`_build_agent()`) отримує сформований промпт та інструмент пошуку і запускається через `agent.ainvoke()`. 5. LLM автономно приймає рішення: (a) безпосередня відповідь з контексту або (b) виклик інструменту search для пошуку в Pinecone з тематичним фільтром. 6. У разі пошуку – Pinecone повертає релевантні Document chunks, які додаються до контексту LLM для генерації фінальної відповіді. 7. Відповідь агента зберігається як Message у SQLite та повертається клієнту у JSON-форматі.

5. КОНВЕЄР ЗБОРУ ТА ІНДЕКСАЦІЇ ДАНИХ (DATA INGESTION PIPELINE)

Якість відповідей RAG-системи безпосередньо залежить від повноти та актуальності бази знань. Для цього реалізовано автоматизований конвеєр збору даних, що включає наступні етапи (рис. 5).

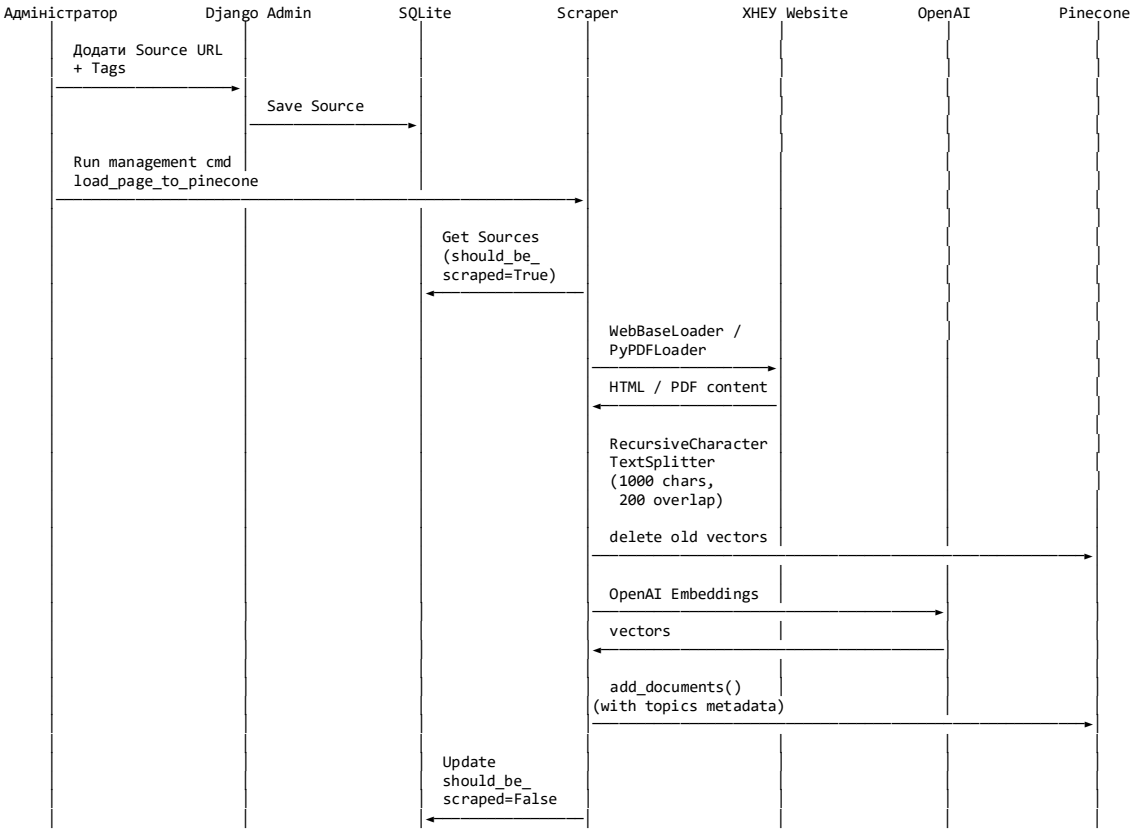


Рис. 5. Діаграма конвеєра збору та індексації даних (Data Ingestion Flow)

5.1. Управління джерелами

Адміністратор системи через панель Django Admin реєструє джерела даних (Source), вказуючи URL-адресу вебсторінки або PDF-документа та присвоюючи набір тематичних тегів (наприклад, `vstup`, `fakultety`, `stipendiyi`). Кожне джерело має булевий прапорець `should_be_scraped`, що визначає необхідність (повторної) індексації. Підтримка масового імпорту/експорту через django-import-export дозволяє ефективно керувати сотнями джерел.

5.2. Завантаження та фрагментація контенту

При запуску Management Command `load_page_to_pinecone` система послідовно обробляє всі помічені джерела:

Визначення типу контенту. За URL-адресою визначається тип завантажувача: `WebBaseLoader` (із BeautifulSoup4) для HTML-сторінок або `PyPDFLoader` для PDF-документів.

Завантаження. Контент завантажується та перетворюється на текстове представлення з видаленням розмітки та нерелевантних елементів.

Фрагментація (Chunking). Текст розбивається на чанки за допомогою `RecursiveCharacterTextSplitter` з параметрами `chunk_size=1000` та `chunk_overlap=200`. Рекурсивна стратегія розбиття забезпечує збереження семантичної цілісності фрагментів, послідовно намагаючись розділити текст за абзацами, реченнями та словами [18].

5.3. Векторизація та збереження

Текстові фрагменти перетворюються на числові вектори (ембедінги) за допомогою моделі OpenAI text-embedding. Перед збереженням нових векторів система видаляє попередні записи для даного джерела, що забезпечує актуальність індексу при повторному скрапінгу. Вектори зберігаються у Pinecone разом з метаданими, що включають тематичні теги (`topics`) джерела, URL-адресу та ідентифікатор чанка. Тегування метаданих є критичним для роботи Topic-based Routing, описаного у Розділі 4.1.

Після успішної обробки прапорець `should_be_scraped` для джерела скидається у `False`, що запобігає повторній індексації до наступного оновлення контенту.

6. КЛІЄНТСЬКИЙ ІНТЕРФЕЙС

Для взаємодії користувачів із чат-ботом розроблено легковагий JavaScript-віджет (рис. 6), що вбудовується на сторінки офіційного вебсайту ХНЕУ ім. С. Кузнеця шляхом додавання єдиного `<script>`-тегу. Віджет реалізує мінімалістичний чат-інтерфейс з полем вводу повідомлення, областю відображення історії діалогу та кнопкою відправки.

Рис. 6. Макет клієнтського JavaScript-віджета чат-бота

Архітектурно віджет є тонким клієнтом, що взаємодіє з серверною частиною виключно через REST API (`POST /api/conversation/`), передаючи UUID сесії та текст повідомлення та отримуючи JSON-відповідь з текстом агента. Такий підхід забезпечує простоту інтеграції з будь-яким вебресурсом університету без необхідності модифікації серверної частини сайту та підтримує доступність для асистивних технологій відповідно до вимог WCAG 2.1 [19].

7. Ключові архітектурні рішення та обґрунтування

У процесі проєктування та реалізації HNEU Agent System було прийнято ряд архітектурних рішень, що суттєво впливають на якість, продуктивність та підтримуваність системи. Таблиця 2 систематизує ці рішення.

Таблиця 2.

Ключові архітектурні рішення HNEU Agent System

Рішення	Обґрунтування
RAG замість fine-tuning	Можливість оновлення бази знань без перенавчання моделі; прозорість джерел; зменшення галюцинацій [10]
Topic-based Routing	Підвищення точності семантичного пошуку через тематичну фільтрацію; зменшення шуму в контексті LLM
ReAct Agent Pattern	Автономність прийняття рішень агентом щодо необхідності пошуку; гнучкість обробки різнотипних запитів [13]
Async-first (adrf)	Оптимальна продуктивність при множинних I/O-bound API-викликах (OpenAI, Pinecone) [17]
Separation of Concerns (3 Django Apps)	Незалежне масштабування та розвиток модулів agent, admin_panel, scraper
GPT-4o-mini	Оптимальне співвідношення якості, швидкості та вартості для інформаційної підтримки
SQLite для діалогів	Простота розгортання; достатня продуктивність для поточних навантажень; легкість міграції на PostgreSQL при масштабуванні
Pinecone (managed)	Керована хмарна інфраструктура; відсутність потреби у самостійному адмініструванні кластера векторної БД

8. МОНІТОРИНГ ТА АНАЛІТИКА

Для забезпечення контролю якості та збору аналітичних даних система інтегрована з платформою LangSmith [20]. Усі LLM-виклики (класифікація тем, рішення агента, генерація відповідей) автоматично логуються з повним трейсом, що включає: вхідні промпти, параметри моделі, вихідні відповіді, використані інструменти, латентність кожного кроку та вартість токенів.

Така інтеграція надає наступні можливості:

- **Відлагодження** – детальний аналіз ланцюжка дій агента при некоректних відповідях.
- **Оптимізація промптів** – ітеративне вдосконалення системних промптів та промптів тематичних тегів на основі реальних взаємодій.

- **Моніторинг якості** – виявлення тенденцій у типах запитів, частоті звернень до бази знань та випадках, коли агент не зміг надати відповідь.

- **Аналіз інформаційних потреб** – збір статистики щодо тематики запитів для подальшого вдосконалення інформаційного наповнення вебресурсів університету.

9. ОБГОВОРЕННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

Запропонована архітектура HNEU Agent System має кілька суттєвих переваг порівняно з альтернативними підходами до побудови інформаційних чат-ботів для ЗВО. По-перше, використання RAG замість fine-tuning забезпечує можливість оперативного оновлення бази знань (додавання нових сторінок через Admin Panel та повторний скрапінг) без витрат на перенавчання моделі. По-друге, Topic-based Routing з LLM-класифікатором є більш гнучким, ніж rule-based маршрутизація, та масштабується на нові тематичні домени додаванням тегів без модифікації коду. По-третє, агентна архітектура на базі LangGraph ReAct дозволяє системі адекватно реагувати як на прості, так і на складні запити, автономно обираючи стратегію відповіді.

Водночас поточна реалізація має певні обмеження, що визначають напрямки подальшого розвитку:

- **Масштабування бази даних діалогів.** При зростанні навантаження SQLite буде замінено на PostgreSQL з підтримкою з'єднань через пул.

- **Розширення модальностей.** Інтеграція потокової генерації відповідей (Server-Sent Events) для покращення користувацького досвіду, а також підтримка мультимовних запитів.

- **Розширення бази знань.** Додавання нових типів джерел: внутрішні системи LMS (Moodle), база стипендій та грантів, актуальний розклад занять.

- **Оцінювання якості.** Впровадження систематичної методології оцінювання (RAGAS, human evaluation) для кількісного вимірювання точності, повноти та релевантності відповідей [21].

- **Інклюзивні функції.** Розширення віджета засобами мовного вводу (Speech-to-Text) та голосового виведення (Text-to-Speech) для забезпечення доступності для осіб з порушеннями зору та моторики.

10. ВИСНОВКИ

У статті представлено архітектуру, технічну реалізацію та результати впровадження інтелектуального RAG-чат-бота *HNEU Agent System* для вебресурсів Харківського національного економічного університету імені Семена Кузнеця. Система розроблена в рамках прикладного наукового дослідження «Розробка відкритої цифрової інклюзивної освітньої екосистеми університету для забезпечення безперервності вищої освіти в Україні» і є практичним кроком у розбудові цифрової інклюзивної інфраструктури ЗВО.

Основні наукові та практичні результати роботи: 1. Запропоновано та реалізовано багатошарову архітектуру RAG-системи на базі моделі C4, що включає асинхронний REST API (Django 6.0 + adrf), агентний оркестратор (LangGraph ReAct), векторне сховище (Pinecone) та автоматизований конвеєр збору даних. 2. Розроблено оригінальний механізм двоетапної тематичної маршрутизації запитів (Topic-based Routing), що поєднує LLM-класифікацію з тегованою фільтрацією у векторній базі даних, суттєво підвищуючи точність та релевантність семантичного пошуку. 3. Реалізовано модульну архітектуру з чітким розділенням відповідальності (agent, admin_panel, scraper), що забезпечує незалежне масштабування та розвиток компонентів. 4. Інтегровано систему трейсингу (LangSmith) для моніторингу якості, відлагодження та збору аналітичних даних про інформаційні потреби учасників освітнього процесу.

Впровадження HNEU Agent System на вебресурсах ХНЕУ ім. С. Кузнеця знижує бар'єри доступу до інформації для всіх категорій здобувачів вищої освіти, включаючи осіб з особливими освітніми потребами та вступників, сприяючи розбудові інклюзивного та доступного цифрового освітнього середовища відповідно до сучасних стандартів цифровізації вищої освіти в Україні.

References

1. Marinoni G., van't Land H., Jensen T. The impact of COVID-19 on higher education around the world: IAU Global Survey Report. International Association of Universities, 2020.
2. Zawacki-Richter O., Marín V. I., Bond M., Gouverneur F. Systematic review of research on artificial intelligence applications in higher education — where are the educators? International Journal of Educational Technology in Higher Education, 2019, vol. 16, no. 1, pp. 1–27. DOI: 10.1186/s41239-019-0171-0.
3. Chen L., Chen P., Lin Z. Artificial intelligence in education: A review. IEEE Access, 2020, vol. 8, pp. 75264–75278. DOI: 10.1109/ACCESS.2020.2988510.
4. Okonkwo C. W., Ade-Ibijola A. Chatbots applications in education: A systematic review. Computers and Education: Artificial Intelligence, 2021, vol. 2, 100033. DOI: 10.1016/j.caeai.2021.100033.
5. Romero C., Ventura S. Educational data mining and learning analytics: An updated survey. WIREs Data Mining and Knowledge Discovery, 2020, vol. 10, no. 3, e1355. DOI: 10.1002/widm.1355.
6. Стратегія розвитку вищої освіти в Україні на 2022–2032 роки. Кабінет Міністрів України, 2022.
7. Seale J. K. E-learning and disability in higher education: accessibility research and practice. Routledge, 2014.
8. Lewis P., Perez E., Piktus A. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 2020, vol. 33, pp. 9459–9474.

9. Gao Y., Xiong Y., Gao X. et al. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997, 2024.
10. Ovadia O., Brief M., Mishaeli M., Elisha O. Fine-tuning or retrieval? Comparing knowledge injection in LLMs. arXiv preprint arXiv:2312.05934, 2023.
11. Wang L., Yang N., Huang X. et al. From RAG to Rich: Improving Retrieval-Augmented Generation for Knowledge-Intensive Tasks. arXiv preprint arXiv:2407.12873, 2024.
12. Singh A., Martin K. Agentic RAG: A Survey on Retrieval-Augmented Generation with Autonomous Agents. arXiv preprint arXiv:2501.09136, 2025.
13. Yao S., Zhao J., Yu D. et al. ReAct: Synergizing Reasoning and Acting in Language Models. International Conference on Learning Representations (ICLR), 2023.
14. LangGraph Documentation. LangChain, Inc. URL: https://langchain-ai.github.io/langgraph/ (accessed: 01.04.2026).
15. Дзюбан Ю. Автоматизуємо службу техпідтримки за допомогою RAG: порівнюємо OpenAI + Pinecone й OpenAI Assistants API. DOU, 2023.
16. Brown S. The C4 Model for Visualising Software Architecture. URL: https://c4model.com/ (accessed: 01.04.2026).
17. Django Documentation. Asynchronous support. URL: https://docs.djangoproject.com/en/5.0/topics/async/ (accessed: 01.04.2026).
18. LangChain Documentation. Text Splitters. URL: https://docs.langchain.com (accessed: 01.04.2026).
19. Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation, 2018. URL: https://www.w3.org/TR/WCAG21/.
20. LangSmith Documentation. LangChain, Inc. URL: https://docs.smith.langchain.com/ (accessed: 01.04.2026).
21. Es S., James J., Espinosa-Anke L., Schockaert S. RAGAS: Automated Evaluation of Retrieval Augmented Generation. arXiv preprint arXiv:2309.15217, 2023.