МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП
«ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-07

Дипломний проєкт

здобувачки освіти денної форми навчання
РП.07.14.000.ДП

МИРОНЕНКО АНАСТАСІЇ ПАВЛІВНИ

м. Одеса
2024 р

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: ***121«Інженеріяпрограмного забеспечення»***

Освітньо-професійна програма: «***Розробка програмного забезпечення»***

Група: ***4РП-07***

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка інтелектуального пошуку для сайту з продажу КТ

Проєктний матеріал складається з пояснювальної записки на 58 сторінках та графічного (перзентаційного) матеріалу на **10** аркушах (слайдах).

Дипломник _____ (Мироненко А.П.)

Керівник _____ (Кунуп Т. В)

Консультанти:

з економічного розділу _____ (Іванченков В. С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н. І.)

з нормоконтролю _____ (Петрашова В. І.)

старший консультант _____ (Кривченко Ю. В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю. В.)

Завідувач відділення _____ (Скорнякова О. В.)

Захист «27» 06 2024 р. Протокол ЕК № 3

Оцінка ЕК 3 (задовільно) / 70 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення **комп'ютерних систем** Комісія **КТ та ІП**

Спеціальність 121 **«Іженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР Беркань І.В.

"15" 01 2024 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти Мироненко Анастасії Павлівні

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка ітелектуального пошуку для сайту с продажу КТ.

затверджена наказом по коледжу від "01.11" 2023р. № 244-А2-02

2. Термін здачі закінченого проекту (роботи) 10.06.24

3. Вихідні данні до проекту (роботи)

1. Проектні рішення з розробки ітелектуального пошуку для сайту с продажу КТ.
2. Алгоритм розробки програми.
3. Мови програмування для створення програмного продукту Java-script, HTML, CSS.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Вступ. Аналіз сучасних технологій розробки програмного продукту;

Конструкторський розділ. Розробка архітектури системи; Проектування бази даних; Розробка алгоритму забезпечення підтримки прийняття рішень при пошуку комплектуючих; Розробка програмного забезпечення; Економічна частина; Охорона праці.

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

Схема структурна варіантів використання інтелектуального пошуку КТ;

Схема структурна розгортання системи;

Схема структурна активності користувача ітелектуального пошуку для сайту с продажу КТ;

ER-діаграма структури бази даних;

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Технологічний	Кунуп Т.В.		
Економічна частина	Іванченков В.С.		
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		
Старший консультант	Кривченко Ю.В.		

7. Дата видачі завдання 15.01.24

Керівник Кунуп Т.В. _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	Постановка мети та задач проектування	5.05.2024	виконано
2	Аналіз теми ДП та огляд літературних джерел	7.05.2024	виконано
3	Аналіз технічного завдання	9.05.2024	виконано
4	Розробка конструкторського розділу ДП	11.05.2024	виконано
5	Постановка задачі при проектуванні інтелектуального пошуку	13.05.2024	виконано
6	Розробка структури програми	16.05.2024	виконано
7	Розробка інтелектуального пошуку	18.05.2024	виконано
8	Виконання розділу «Економічні розрахунки»	20.05.2024	виконано
9	Розрахунок економічних показників	23.05.2024	виконано
10	Виконання розділу «Охорона праці»	25.05.2024	виконано
11	Виконання пояснювальної записки ДП	27.05.2024	виконано
12	Виконання мультимедійної презентації ДП	10.06.2024	виконано
13	Малий захист	13.06.2024	виконано
15	Підготовка проекту до захисту та тестування ПП	16.06.2024	виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 07. 14 000. 00 ДП	Дипломний проект		
А4			РП 07. 14 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 07. 14 000. 00 ДП		
Розробив		Мироненкко А.П.		10.06	Розробка інтелектуального пошуку для сайту з продажу КТ	Літ.	Аркуш 4 / Аркушів
Перевірив		Кунуп Т.В.		10.06		Н Д П	
Н. контр.		Петрашова В.І.		14.06	ВСП "ОТФК ОНТУ" гр.4РП-07		
Затверд.		Кривченко Ю.В.		14.06			

ЗМІСТ

ВСТУП

В сучасному світі інформаційні технології відіграють ключову роль у багатьох сферах життя, включаючи комерцію. Інтернет-магазини, що пропонують широкий асортимент комп'ютерної техніки, стають дедалі популярнішими серед споживачів. Проте, з розширенням асортименту товарів виникає проблема ефективного пошуку необхідних продуктів, яка може впливати на користувацький досвід та рівень задоволеності клієнтів.

Забезпечення зручного та швидкого пошуку товарів на сайті є однією з найважливіших задач для будь-якого онлайн-магазину. Традиційні пошукові системи часто не в змозі адекватно обробляти складні запити користувачів, що призводить до невідповідних результатів і втрати потенційних покупців. Саме тому виникає потреба у впровадженні інтелектуальних систем пошуку, здатних аналізувати запити та надавати релевантні результати.

Цей дипломний проект спрямований на розробку інтелектуальної пошукової системи для сайту з продажу комп'ютерної техніки. Основною метою є створення пошукового механізму, який зможе ефективно обробляти запити користувачів, враховувати їхні потреби та пропонувати найбільш підходящі варіанти товарів. Використання сучасних методів машинного навчання та обробки природної мови дозволить забезпечити високу точність пошуку та покращити користувацький досвід.

У даному проекті буде досліджено існуючі технології інтелектуального пошуку, проаналізовано їхні переваги та недоліки, а також розроблено та впроваджено власну систему, яка задовольнятиме специфічні потреби інтернет-магазину з продажу комп'ютерної техніки. Результати проекту можуть бути корисними не лише для конкретного сайту, а й для інших онлайн-платформ, що прагнуть покращити свої пошукові можливості та підвищити задоволеність клієнтів.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Проектні рішення з розробки розробка інтелектуального пошуку

Розробка інтелектуального пошуку для сайту з продажу комп'ютерної техніки є актуальним завданням, що має на меті підвищити ефективність взаємодії користувачів з сайтом, забезпечуючи їм можливість швидко знаходити необхідні товари і робити покупки з максимальним комфортом.

Метою ДП є розробка інтелектуальної пошукової системи для сайту з продажу комп'ютерної техніки, яка забезпечуватиме високоточний і зручний пошук товарів, покращуючи користувацький досвід. Ця система повинна автоматично аналізувати запити користувачів, враховувати їхні потреби та пропонувати найбільш релевантні результати, що дозволить збільшити ефективність пошуку та задоволеність клієнтів, а також підвищити конкурентоспроможність сайту на ринку. Метою проекту є:

1. Реалізувати інтелектуальну функцію пошуку, яка розуміє та обробляє запити користувачів семантично.
2. Інтегрувати семантичний пошук з існуючою логікою пошуку інтернет-магазину.
3. Забезпечити повернення пошукових результатів, що включають не лише точні збіги, але й семантично схожі продукти.

У дипломному проекті реалізовано вдосконалення функціоналу пошуку в інтернет-магазині, що базується на CMS OpenCart, шляхом інтеграції інтелектуальних семантичних пошукових можливостей, використовуючи моделі OpenAI. Завдання полягає в тому, щоб функція пошуку повертала не лише точні збіги з запитом користувача, але й семантично схожі продукти.

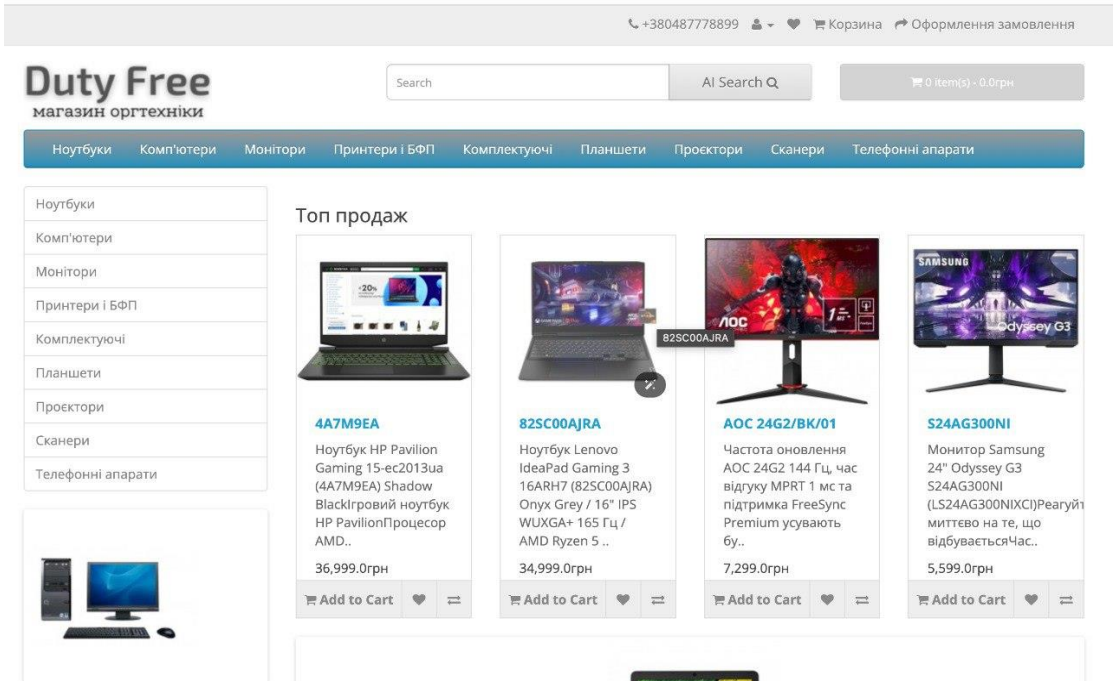


Рисунок 1.1. Скриншот головної сторінки сайту

Для створення інтелектуального пошуку необхідно сформувати базу продуктів комп’ютерної техніки, комплектуючих, ранжувати їх за різними критеріями, визначити множину класів завдань, з якими буде працювати комп'ютер у майбутньому, та вирішити задачу класифікації для ефективного підбору комплектуючих.

Розробка ітелектуального пошуку спрощить процес вибору комп’ютерної техніки та допоможе мінімізувати вплив недоліків та проблем, пов'язаних з неправильним вибором компонентів. Скриншот головної сторінки сайту представлений на рисунку 1.1.

1.2 Актуальність проблеми ітелектуального пошуку для сайту

Розвиток електронної комерції в сучасному світі став ключовим фактором для зростання бізнесу. Однак, однією з найважливіших проблем, з якими стикаються онлайн-магазини, є забезпечення ефективного пошуку товарів на сайті. Тут на допомогу приходить Smart Site Search або " ітелектуальний

пошук".

Ітелектуальний пошук - це пошукова система, що базується на інтелектуальних алгоритмах. Вона володіє розширеними можливостями, які покращують досвід користувачів, розпізнаючи та пропонуючи релевантні результати, знаходячи синоніми та визначаючи фонетичні схожості між літерами і словами, щоб показати правильні результати навіть при помилках в написанні ключових слів.

1. Задоволення клієнтів: Близько 40% онлайн-покупців відразу користуються пошуком на сайті. Відповідно, забезпечення ефективного пошуку є критично важливим для задоволення клієнтів і підвищення конверсії.
2. Підвищення конверсій: Дослідження показали, що користувачі, які використовують функцію пошуку, мають у 2.4 рази більшу ймовірність зробити покупку. Це безпосередньо впливає на збільшення доходу бізнесу.
3. Зменшення кількості незадоволених клієнтів: Погано налаштований пошук може стати причиною фрустрації клієнтів, що призведе до їхнього відходу до конкурентів.

Основні проблеми традиційного пошуку

Незважаючи на великі переваги розумного пошуку, більшість сайтів електронної комерції не забезпечують його на належному рівні. Основні проблеми включають:

Нерелевантні результати.

Відсутність відповідних товарів.

Повільна швидкість роботи.

Поганий дизайн інтерфейсу.

Відсутність візуальних елементів.

Ключові функції ітелектуального пошуку

1. Автозаповнення: Покращує точність пошуку, пропонуючи релевантні підказки під час введення запиту.
2. Уникнення сторінок без результатів: Використання аналітики для

визначення запитів без результатів та оптимізації пошуку.

3. Популярні запити: Відображення популярних товарів і запитів для полегшення навігації.
4. Семантичний пошук: Розпізнавання намірів користувачів для більш релевантних результатів.
5. Виправлення орфографічних помилок: Автокорекція та толерантність до помилок забезпечують точні результати навіть при помилкових запитах.
6. Фільтрація широких термінів: Виключення загальних термінів для покращення точності пошуку.
7. Оптимізація для мобільних пристроїв: Забезпечення зручності використання пошуку на малих екранах.

Розумний пошук є необхідним інструментом для сучасних інтернет-магазинів. Він не тільки покращує досвід користувачів, але й значно підвищує конверсії та дохід бізнесу. Впровадження передових функцій розумного пошуку допоможе вашому бізнесу залишатися конкурентоспроможним і задовольнити зростаючі потреби ваших клієнтів.

1.3 Аналіз ринку ітелектуального пошуку для сайту

Ринок розумного пошуку на сайтах є швидко розвивається сегментом у ширшій сфері електронної комерції та цифрових технологій. Цей аналіз розглядає ключові тенденції, рушійні сили, виклики та можливості, що визначають цей ринок.

Ключові тенденції ринку

Зростання прийняття електронної комерції: З глобальним зростанням онлайн-покупок платформи електронної комерції все більше впроваджують технології розумного пошуку для підвищення якості обслуговування клієнтів і збільшення продажів. Ця тенденція обумовлена необхідністю конкурувати з великими гравцями, такими як Amazon та Alibaba, які встановили високі

стандарти в обслуговуванні клієнтів.

Інтеграція штучного інтелекту та машинного навчання: Інтеграція штучного інтелекту (AI) та машинного навчання (ML) у пошукові алгоритми революціонізує спосіб генерації пошукових результатів. Ці технології дозволяють отримувати більш точні, релевантні та персоналізовані результати пошуку, покращуючи задоволеність клієнтів та коефіцієнти конверсії.

Оптимізація голосового пошуку: З популяризацією голосових пристроїв, таких як Amazon Echo та Google Home, оптимізація пошукових функцій для голосових запитів стає необхідністю. Ця тенденція стимулює розвиток більш досконалих можливостей обробки природної мови (NLP) у рішеннях розумного пошуку.

Зростання мобільної комерції: Збільшення використання мобільних пристроїв для онлайн-покупок вимагає оптимізованих для мобільних пристроїв пошукових функцій. Забезпечення безперебійного та ефективного пошукового досвіду на менших екранах є критичним напрямком для постачальників технологій розумного пошуку.

Рушійні сили ринку

Покращення досвіду клієнтів: Рішення розумного пошуку значно покращують досвід клієнтів, швидко надаючи точні та релевантні результати пошуку. Це призводить до вищої задоволеності клієнтів та лояльності.

Збільшення коефіцієнтів конверсії: Ефективні пошукові функції безпосередньо пов'язані з вищими коефіцієнтами конверсії. Клієнти, які швидко знаходять те, що шукають, частіше роблять покупки, збільшуючи загальний дохід.

Конкурентна перевага: Бізнеси, що інвестують у передові пошукові технології, отримують конкурентну перевагу над тими, хто цього не робить. Відмінні пошукові можливості можуть диференціювати бренд на насиченому ринку.

Аналітика, заснована на даних: Системи розумного пошуку часто

включають розширену аналітику, яка надає уявлення про поведінку клієнтів та пошукові запити. Ці дані можна використовувати для оптимізації інвентарю, маркетингових стратегій та загальних операцій бізнесу.

Виклики ринку

Складність впровадження: Інтеграція передових пошукових технологій у існуючі платформи електронної комерції може бути складною та ресурсомісткою. Це вимагає значної технічної експертизи та інвестицій.

Високі витрати: Розробка та підтримка складних систем розумного пошуку можуть бути дорогими, особливо для малих та середніх підприємств (МСП) з обмеженими бюджетами. Проблеми конфіденційності даних: Оскільки системи розумного пошуку часто покладаються на збір та аналіз даних користувачів, існують певні ризики, пов'язані з конфіденційністю та безпекою даних. Бізнесам необхідно враховувати ці проблеми, забезпечуючи відповідність регламентам, таким як GDPR.

Постійно змінюються технології: Швидкий темп технологічного розвитку означає, що системи розумного пошуку повинні постійно еволюціонувати, щоб залишатися ефективними. Це вимагає постійних інвестицій у дослідження та розвиток.

Глобальне розширення: Ринки, що розвиваються, представляють величезну можливість для зростання. Зі збільшенням проникнення електронної комерції в таких регіонах, як Азіатсько-Тихоокеанський регіон, Латинська Америка та Африка, очікується зростання попиту на передові пошукові технології.

Український ринок розумних систем пошуку демонструє стабільне зростання, відображаючи глобальні тенденції. Попит на покращений користувацький досвід, підвищену онлайн-видимість та персоналізовані результати пошуку стимулює впровадження інтелектуальних рішень для пошуку в різних галузях.

Ключові фактори, що впливають на це зростання:

Зростання електронної комерції: Сектор електронної комерції в Україні швидко розширюється, спонукаючи підприємства інвестувати в технології, які підвищують задоволеність клієнтів та коефіцієнти конверсії.

Цифрова трансформація: Українські компанії все більше впроваджують ініціативи цифрової трансформації, визнаючи важливість інтелектуального пошуку у наданні персоналізованого досвіду.

Конкурентна перевага: Підприємства використовують розумний пошук на сайті, щоб отримати конкурентну перевагу, пропонуючи своїм клієнтам інтуїтивно зрозумілі та ефективні можливості пошуку.

Ключові гравці ринку

Хоча український ринок не такий насичений, як світовий, кілька компаній пропонують розумні рішення для пошуку на сайті, адаптовані до місцевих потреб:

AmoCRM. Популярна CRM-платформа в Україні, яка пропонує функцію пошуку на сайті як частину свого набору інструментів. (https://www.amocrm.com/)

Weblium. Український конструктор вебсайтів, який включає базову функціональність пошуку на сайті у свої плани. (https://weblium.com/)

Lucky Orange. Хоча це не українська компанія, Lucky Orange широко використовується в Україні для аналітики вебсайтів і пропонує функцію відстеження пошуку на сайті. (https://www.luckyorange.com/)

Виклики та можливості ринку.

Український ринок розумних систем пошуку надає як виклики, так і можливості:

Обізнаність. Хоча попит на інтелектуальний пошук зростає, багато українських підприємств досі не усвідомлюють повного потенціалу та переваг цих рішень.

Вартість. Малі та середні підприємства можуть вважати вартість впровадження передових рішень для пошуку непомірною.

Локалізація. Адаптація глобальних пошукових рішень до специфічних потреб української мови та ринку може бути складним завданням.

Однак ці виклики також створюють можливості для компаній, які можуть:

Навчати ринок: Компанії, які активно просувають переваги розумного пошуку на сайті та пропонують освітні ресурси, можуть освоїти зростаючий ринок.

Пропонувати доступні рішення: Надання економічно ефективних варіантів, адаптованих до бюджетних обмежень малих підприємств, може відкрити нові сегменти клієнтів.

Зосередитись на локалізації: Компанії, які інвестують у розуміння нюансів української мови та поведінки користувачів, можуть розробляти високорелевантні рішення для пошуку.

1.4. Вибір технологій

Вибір технологічного стеку є одним з найважливіших рішень, які приймаються на початковому етапі розробки будь-якого програмного проекту. Від цього вибору залежить успішність, ефективність і майбутня масштабованість проекту. Нижче наведено кілька ключових причин, чому вибір технологічного стеку є таким важливим.

Критерії вибору технологій

1. Сумісність: Технологія повинна бути сумісна з існуючою CMS OpenCart і базою даних MySQL. Скриншот бази даних показано на рисунку 1.2.

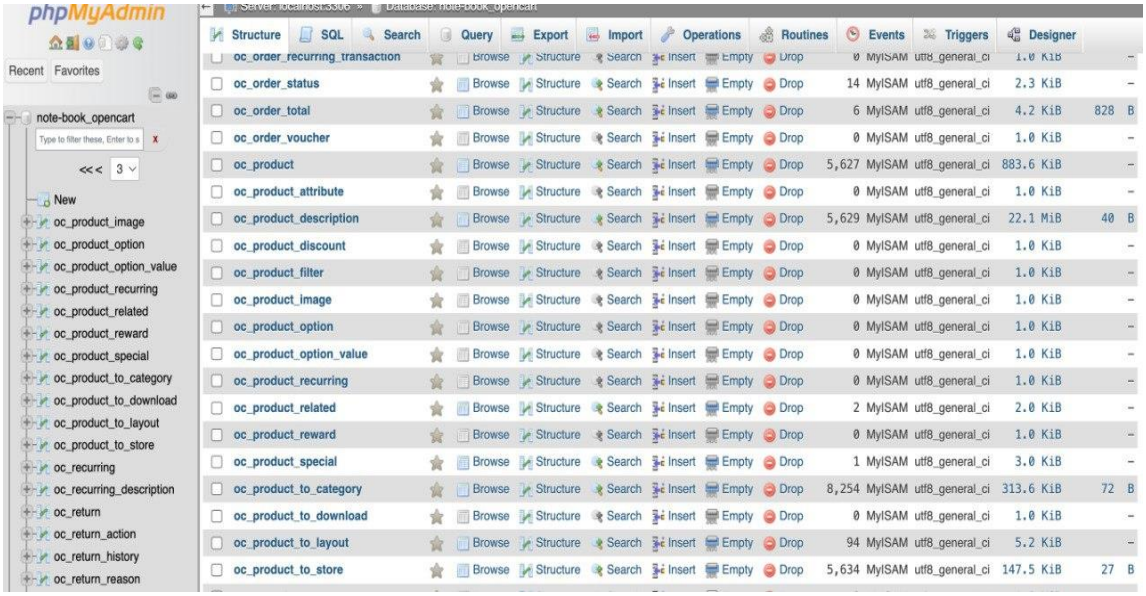
2. 

Рисунок 1.2. Скриншот бази даних сайту

3. Масштабованість: Обрана технологія повинна забезпечувати можливість обробки майбутнього зростання обсягу даних і навантаження користувачів.

4. Продуктивність: Технологія повинна забезпечувати високу продуктивність як для процесів пошуку, так і для процесів отримання даних.

5. Простота інтеграції: Технологія повинна легко інтегруватися з OpenCart і іншими компонентами.

6. Документація та підтримка спільноти: Наявність детальної документації і активної підтримки спільноти для допомоги в розробці та вирішенні проблем.

7. Передові функції: Підтримка передових функцій пошуку, включаючи семантичний пошук.

Процес вибору

Крок 1: Технологія для бекенд сервера

Розглянуті варіанти:

PHP (рідний для OpenCart):

Переваги: Пряма інтеграція з OpenCart, не потрібна додаткова серверна

настройка. Недоліки: Обмежена підтримка сучасних практик розробки і проблеми з масштабованістю.

Node.js з фреймворком Nest.js:

Переваги: Сучасний, масштабований і ефективний; відмінна підтримка асинхронних операцій, що важливо для обробки запитів в реальному часі. Nest.js додає структуру і підвищує підтримуваність.

Алє є недоліки: Потрібна настройка окремого серверного середовища.

Рішення: Node.js з фреймворком Nest.js

Причина: Node.js з Nest.js забезпечує надійне і масштабоване серверне середовище, яке підтримує сучасні практики розробки та ефективну обробку асинхронних операцій, що робить його ідеальним для реалізації інтелектуальної функції пошуку.

Крок 2: Інструмент для документації API

Розглянуті варіанти:

Swagger:

Переваги : Широко використовується і підтримується; пропонує комплексні можливості документації API; легко інтегрується з Node.js і Nest.js.

Недоліки: Відсутні значні недоліки.

Postman:

Переваги: Відмінний інструмент для тестування API, але менш підходящий для автоматичної генерації документації.

Недоліки: Переважно тестувальний інструмент, а не інструмент для документації.

Рішення: Swagger

Причина: Swagger є промисловим стандартом для документації API, пропонуючи розширені функції і безшовну інтеграцію з обраною серверною технологією.

Крок 3: Побудовник запитів до бази даних

Розглянуті варіанти:

Knex:

Переваги: Потужний побудовник запитів для SQL баз даних; підтримує декілька баз даних, включаючи MySQL; добре інтегрується з Node.js.

Недоліки: Потрібне навчання для розробників, які не знайомі з ним.

Sequelize.

Переваги: ORM з більш абстрактною обробкою запитів.

Недоліки: Трохи складніший і може створювати непотрібне навантаження для цього проекту.

Рішення: Knex

Причина: Knex пропонує гарний баланс між потужністю та простотою, забезпечуючи ефективне побудування запитів при можливості прямої взаємодії з базою даних MySQL.

Крок 4: Векторна база даних для семантичного пошуку

Розглянуті варіанти:

Pinecone:

Переваги: Оптимізована для векторного пошуку; підтримує високопродуктивний, масштабований і реальний час пошуку.

Недоліки: Потрібна додаткова настройка.

Elasticsearch:

Переваги: Потужний пошуковий та аналітичний двигун.

Недоліки: Складність і потенційне перевантаження для вимог векторного пошуку.

Рішення: Pinecone

Причина: Pinecone спеціально розроблена для високопродуктивного векторного пошуку, що робить її оптимальним вибором для реалізації можливостей семантичного пошуку.

Крок 5: Великі мовні моделі

Розглянуті варіанти:

OpenAI:

Переваги: Передові мовні моделі з сильною підтримкою семантичного пошуку; розширена документація API та підтримка спільноти.

Недоліки: Вартісні міркування.

Google BERT:

Переваги: Сильні можливості семантичного пошуку.

Недоліки: Складніший процес інтеграції.

Рішення: OpenAI

Причина: Моделі OpenAI забезпечують найпросунутіше семантичне розуміння, що робить їх ідеальними для покращення функціоналу пошуку в онлайн-магазині.

Підсумковий технологічний стек

Серверна частина: Node.js з фреймворком Nest.js

Документація API: Swagger

Побудовник запитів до бази даних: Knex

Векторна база даних: Pinecone

Великі мовні моделі: OpenAI

Обраний технологічний стек забезпечує баланс між сумісністю, масштабованістю, продуктивністю, простотою інтеграції та передовими можливостями пошуку. Node.js з Nest.js забезпечує надійну серверну інфраструктуру, тоді як Swagger гарантує повну документацію API. Knex сприяє ефективній взаємодії з базою даних, Pinecone пропонує високопродуктивний векторний пошук, а моделі OpenAI дозволяють реалізувати передові можливості семантичного пошуку. Ця комбінація значно покращить функціонал пошуку в онлайн-магазині, надаючи користувачам релевантні та різноманітні результати пошуку продуктів.

1.5 Розробка архітектури системи

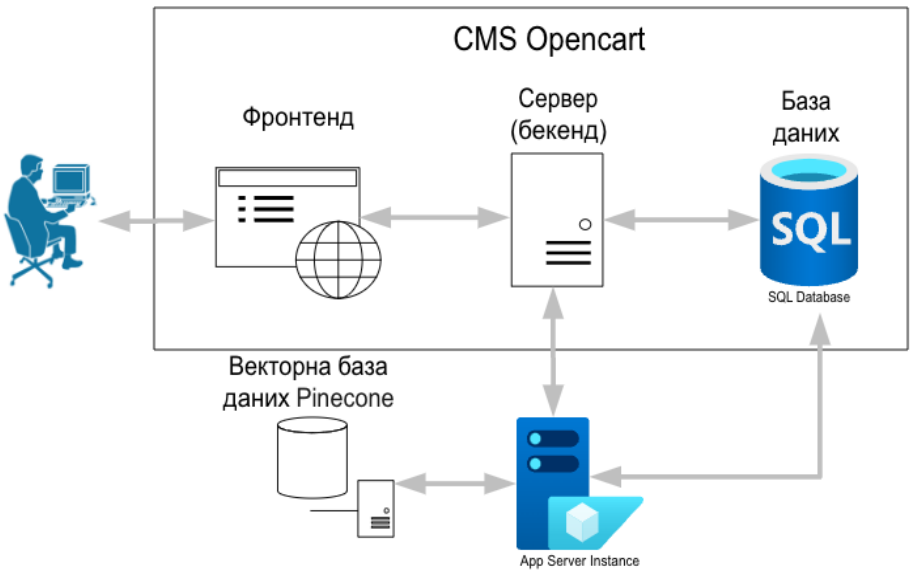


Рисунок 1.3. Архітектура системи

Архітектура системи складається з кількох компонентів, що працюють разом для забезпечення функціонування онлайн-магазину, побудованого на CMS OpenCart. Нижче наводиться опис кожного з компонентів, представлених на схемі на рисунку 1.3.

Компоненти системи:

1. Користувач.

Взаємодіє з фронтендом через веб-браузер.Надсилає запити на пошук продуктів і отримує результати пошуку.

2. Фронтенд.

Інтерфейс користувача, який відображає результати пошуку та інформацію про продукти. Відправляє запити до серверної частини (бекенду) для отримання даних.

3. Сервер (бекенд).

Основний сервер, який обробляє запити від фронтенду. Використовує Node.js з фреймворком Nest.js для обробки запитів і взаємодії з іншими компонентами системи.

Відправляє запити до бази даних MySQL для отримання інформації про продукти.

Інтегрується з векторною базою даних Pinecone для виконання семантичного пошуку.

База даних MySQL:

Основне сховище даних для продуктів та іншої інформації, що використовується OpenCart. Зберігає дані про назви, описи та інші характеристики продуктів.

8. Векторна база даних Pinecone**:**

Використовується для зберігання векторних представлень (ембеддингів) продуктів, створених за допомогою моделей OpenAI. Виконує семантичний пошук для знаходження продуктів, які відповідають запитам користувачів не тільки за точним збігом, але й за смисловою схожістю.

9. App Server Instance (Інстанція серверного додатку):

Виконує спеціалізовані завдання, пов'язані з обробкою даних та взаємодією з Pinecone і OpenAI.

Відповідає за виклик API OpenAI для створення ембеддингів продуктів.

Потік даних:

1. Користувач відправляє запит через фронтенд.
2. Фронтенд пересилає запит на серверну частину.
3. Сервер виконує запит до бази даних MySQL для отримання базової інформації про продукти.
4. Сервер також взаємодіє з Pinecone для виконання семантичного пошуку, якщо необхідно.

5. Pinecone використовує векторні представлення продуктів для пошуку семантично схожих результатів.

6. Сервер об'єднує результати з MySQL і Pinecone, форматує їх і відправляє назад на фронтенд.

7. Фронтенд відображає результати пошуку користувачу.

Ця архітектура дозволяє інтегрувати семантичний пошук в онлайн-магазин, побудований на OpenCart, забезпечуючи користувачам більш релевантні та різноманітні результати пошуку. Використання Pinecone і OpenAI значно підвищує можливості пошукової системи, дозволяючи знаходити продукти не тільки за точним збігом ключових слів, але й за їхнім змістовним значенням.

1.6 Практична реалізація

- **1.6.1 Створення структури сервера**

Для створеня сервера мною було використано NestJs. NestJs - це прогресивний фреймворк для Node.js, який дозволяє створювати ефективні, надійні та масштабовані серверні додатки. У цьому керівництві розглядається крок за кроком процес створення сервера на NestJs для інтеграції з CMS OpenCart.

Встановлення NestJs

1. Встановила Node.js

Завантажила та встановил останню версію Node.js з офіційного сайту: Node.js

2. Встановила Nest CLI:

Відкрил термінал і виконал наступну команду для встановлення Nest CLI глобально: npm install -g @nestjs/cli

3. Створила новий проект:

Створили новий проект NestJs за допомогою команди Nest CLI:

nest new project-name

Слідувала підказкам і обрала менеджер пакетів npm.

Крок 2: Налаштування проекту

1. Перейшла до створеного проекту:

   cd project-name

2. Встановив необхідні залежності:

Використали менеджер пакетів для встановлення залежностей:

npm install openai, nest-knexjs, mysql, knex, @nestjs/config

Крок 3: Налаштування бази даних

1. Створила конфігураційний файл для підключення до бази даних:

Створила файл .env у кореневій директорії проекту:

```
PORT=3000
# PINECONE DATABASE CONFIG
PINECONE_INDEX=ai-course
PINECONE_API_KEY=b5538d94-068a-4f34-a599-278ba38fb0b7
PINECONE_ENVIRONMEN=gcp-starter
# SQL DATABASE CONFIG
DB_HOST=duty-free.top
DB_USER=note-book_note-_opencart
DB_PASSWORD=admin
DB_NAME=note-book_opencart
```

2. Налаштувала Knex для з'єднання з SQL базою даних. Скриншот налаштування бібліотеки Knex на рисунку 1.4.

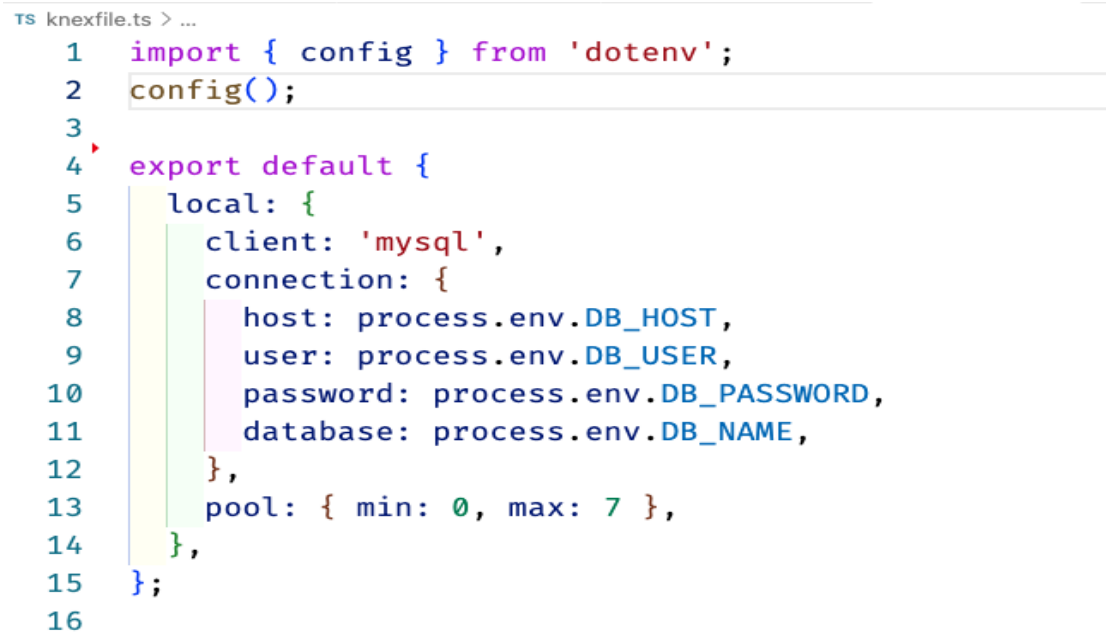


Рисунок 1.4. Скриншот налаштування бібліотеки Knex

Цей код налаштовує бібліотеку Knex для підключення до бази даних MySQL, використовуючи змінні середовища для конфігурації підключення.

Опис коду:

1. Імпорт бібліотеки dotenv:

   typescript

   Copy code

   import { config } from 'dotenv';

Імпортується функція config з бібліотеки dotenv, яка використовується для завантаження змінних середовища з файлу .env.

2. Виклик config():

   typescript

   Copy code

   config();

Викликається функція config для завантаження змінних середовища з файлу .env в process.env.

3. Експорт конфігурації Knex:

typescript

Copy code

```
export default {
  local: {
    client: 'mysql',
    connection: {
      host: process.env.DB_HOST,
      user: process.env.DB_USER,
      password: process.env.DB_PASSWORD,
      database: process.env.DB_NAME,
    },
      pool: { min: 0, max: 7 },
    },
  };
```

Конфігурація експортується як об'єкт за замовчуванням, який містить налаштування для підключення до локальної бази даних MySQL.

local:

client: 'mysql': Вказує, що використовується клієнт для бази даних MySQL.

connection: Об'єкт конфігурації підключення, що використовує змінні середовища:

host: process.env.DB_HOST: Хост бази даних.

user: process.env.DB_USER: Ім'я користувача для підключення до бази даних.

password: process.env.DB_PASSWORD: Пароль для підключення.

database: process.env.DB_NAME: Назва бази даних.

pool: Налаштування пулу підключень:

min: 0: Мінімальна кількість підключень у пулі.

max: 7: Максимальна кількість підключень у пулі.

Цей код налаштовує Knex для роботи з базою даних MySQL, використовуючи змінні середовища для конфігурації підключення. Функція dotenv.config() завантажує змінні середовища з файлу .env, що забезпечує безпечне зберігання конфіденційної інформації, такої як хост, користувач, пароль і назва бази даних. Конфігурація експортується як об'єкт, який може бути використаний для ініціалізації Knex у проекті. Скриншот конфігурації модуля в проекті показаний на рисунку 1.5.

app.module.ts > AppModule

```
import { EmbeddingModule } from './embedding/embedding.module';
import { KnexModule } from 'nestjs-knex';
import { ShopModule } from './shop/shop.module';

@Module({
  imports: [
    ConfigModule.forRoot({
      isGlobal: true,
      load: [config],
    }),
    KnexModule.forRootAsync({
      useFactory: (configService: ConfigService) => ({
        config: {
          name: 'local',
          client: 'mysql',
          version: '5.7',
          connection: {
            host: configService.get<string>('host'),
            user: configService.get<string>('user'),
            password: configService.get<string>('password'),
            database: configService.get<string>('database'),
          },
        },
      }),
      inject: [ConfigService],
    }),
    StoriesModule,
    OpenaiModule,
    EmbeddingModule,
```

Рисунок 1.5. Скриншот конфігурації модуля в проекті

Цей код демонструє конфігурацію модуля в проекті на основі NestJS з використанням кількох інших модулів, включаючи конфігураційний модуль, модуль для роботи з базою даних Knex, а також деякі користувацькі модулі.

Опис коду

1. Імпорт модулів.

EmbeddingModule з ./embedding/embedding.module

KnexModule з nestjs-knex

ShopModule з ./shop/shop.module

2. Декоратор @Module: Використовується для визначення модуля в NestJS.

3. Масив imports: Перелічує модулі, які імпортуються в цей модуль.

4. ConfigModule.forRoot.

Конфігураційний модуль, який завантажується глобально.

Параметри:

isGlobal: true - робить модуль глобальним.

load: [config] - завантажує конфігурацію з файлу config.

5. KnexModule.forRootAsync.

Асинхронна конфігурація модуля Knex для роботи з базою даних.

Використовує useFactory, яка приймає ConfigService для отримання параметрів конфігурації.

Параметри конфігурації для підключення до бази даних:

name: 'local'

client: 'mysql'

version: '5.7'

connection: включає host, user, password, database, які отримуються з ConfigService.

6. inject: [ConfigService]:

Інжектує ConfigService для використання у фабричній функції.

7. Інші модулі:

StoriesModule

OpenaiModule

EmbeddingModule

Цей код налаштовує основний модуль програми на базі NestJS з використанням глобального конфігураційного модуля і асинхронного модуля для роботи з базою даних через Knex. Він забезпечує підключення до бази даних MySQL та імпорт декількох інших модулів, необхідних для роботи програми.

Крок 4: Створення модулів, контролерів та сервісів

1. Був створений модуль для продуктів в інтернет магазині:

Виконала команду Nest CLI для створення модуля, контролера та сервісу:

nest generate module shop

nest generate service shop

nest generate controller shop

1.6.2 Налаштування модулю продуктів

Для цього відкрила файл src/shop/shop.module.ts і додали наступний код. Скриншот кода модуля для продуктів в інтернет магазині: представлено на рисунку 1.

```
op > TS shop.module.ts > ...
import { Module } from '@nestjs/common';
import { ShopController } from './shop.controller';
import { ShopService } from './shop.service';
import { EmbeddingService } from '../embedding/embedding.service';
import { OpenaiService } from '../openai/openai.service';

@Module({
  controllers: [ShopController],
  providers: [ShopService, EmbeddingService, OpenaiService],
})
export class ShopModule {}
```

Рисунок 1.6. Скриншот модуля для продуктів в інтернет магазині

Цей фрагмент коду визначає модуль ShopModule у фреймворку NestJS. Він імпортує необхідні контролери та сервіси для роботи інтернет-магазину та зв'язує їх разом у модуль. Нижче наведено детальний опис цього модуля.

Опис компонентів:

1. Імпорт модулів:

@nestjs/common - основний модуль NestJS для декораторів та інших загальних функцій.

ShopController - контролер для обробки HTTP-запитів, пов'язаних з продуктами інтернет-магазину.

ShopService - сервіс для обробки даних продуктів та взаємодії з базою даних.

EmbeddingService - сервіс для вбудовування даних у векторну базу.

OpenaiService - сервіс для взаємодії з OpenAI, ймовірно, для отримання додаткових даних або обробки запитів.

2. Декоратор @Module():

Визначає метадані для створення модуля NestJS.

Властивість controllers вказує на контролери, які будуть зареєстровані в цьому модулі.

Властивість providers вказує на сервіси, які будуть доступні в межах цього модуля для інжекції залежностей.

3. Клас ShopModule:

Це клас, який експортується і стає доступним для використання в інших частинах додатку.

Призначення модуля ShopModule об'єднує контролер та сервіси, необхідні для роботи з продуктами інтернет-магазину. Він забезпечує структуровану організацію коду, полегшуючи його масштабування та підтримку. Використання окремих сервісів для різних задач (обробка продуктів, вбудовування даних, взаємодія з OpenAI) сприяє модульності та повторному використанню коду. Наступним кроком була реалізація сервісу ShopService

для продуктів. Цей код реалізує сервіс ShopService у фреймворку NestJS для роботи з базою даних через Knex та інтеграцію зі службою EmbeddingService. В ньому реалізовано два методи: embeddingProducts() та getProducts(search).

1. Метод embeddingProducts() відповідає за отримання даних з бази даних інтернет-магазину, їх обробку та збереження у векторну базу даних.
2. Метод getProducts(search), який приймає пошуковий запит як аргумент, робить запит до векторної бази, де за допомогою семантичного пошуку знаходяться потрібні товари та їхні коди повертаються до інтернет-магазину.

```
import { Injectable } from '@nestjs/common';
import { Knex } from 'knex';
import { InjectConnection } from 'nestjs-knex';
import { EmbeddingService } from '../embedding/embedding.service';

@Injectable()
export class ShopService {
  constructor(
    private embeddingService: EmbeddingService,
    @InjectConnection() private readonly knex: Knex,
  ) {}
  async embeddingProducts() {…
  }

  async getProducts(search: string): Promise<string[]> {…
  }
}
```

Рисунок 1.7. Скриншот сервісу ShopService

Опис компонентів сервіс ShopService

1. Імпорт модулів:

@nestjs/common - основний модуль NestJS для декораторів та інших загальних функцій.

knex - модуль для взаємодії з базою даних.

nestjs-knex - модуль для інтеграції Knex з NestJS.

EmbeddingService - користувацький сервіс для вбудовування даних у векторну базу.

2. Декоратор @Injectable():

Вказує, що цей клас може бути інжектованим як сервіс у NestJS.

3. Конструктор:

Приймає два параметри:

embeddingService - екземпляр EmbeddingService для обробки даних продуктів.
knex - екземпляр Knex для взаємодії з базою даних, інжектований за допомогою декоратора @InjectConnection().

4. Методи:

embeddingProducts(): Асинхронний метод, який відповідає за отримання даних з бази даних інтернет-магазину, їх обробку та збереження у векторну базу даних.

getProducts(search: string): Promise<string[]>: Асинхронний метод, який приймає пошуковий запит як аргумент, виконує запит до векторної бази, знаходить потрібні товари за допомогою семантичного пошуку та повертає їхні коди.

Цей сервіс забезпечує інтеграцію інтернет-магазину з векторною базою даних, що дозволяє здійснювати складні пошукові запити та обробляти великі обсяги даних для покращення роботи магазину та надання кращих результатів пошуку для користувачів.

1.6.3 Розробка метода вбудуваня продуктів

Опис метода embeddingProducts:

```
const products = await this.knex
  .select(
    'oc_product.product_id as product_id',
    'oc_product.model as model',
    'oc_product.status as status',
    'oc_product_description.name as name',
    'oc_product_description.description as description',
    'oc_category_description.name as category_name',
  )
  .from('oc_product')
  .leftJoin(
    'oc_product_description',
    'oc_product.product_id',
    'oc_product_description.product_id',
  )
  .leftJoin(
    'oc_product_to_category',
    'oc_product.product_id',
    'oc_product_to_category.product_id',
  )
  .leftJoin(
    'oc_category_description',
    'oc_product_to_category.category_id',
    'oc_category_description.category_id',
  )
  .where({
    'oc_product.status': 1,
    'oc_category_description.name': categoryName,
  })
```

Рисунок 1.8. Запит до базs SQL в інтернет магазині

Цей фрагмент коду взаємодіє з базою даних за допомогою Knex. Його мета – отримати список продуктів з бази даних. Запит починається з оператора select, який вказує стовпці для отримання з трьох таблиць: oc_product, oc_product_description та oc_manufacturer. Використовується ключове слово as для перейменування стовпців у результаті для зручності.

Метод from визначає основну таблицю для запиту – oc_product, з якої починаються операції з'єднання.

Два leftJoin операції додають дані з таблиць oc_product_description і oc_manufacturer, забезпечуючи включення всіх записів з oc_product та

відповідних записів з інших таблиць. Якщо відповідностей немає, результати міститимуть null значення для стовпців з невідповідних таблиць.

Умови з'єднання вказуються відповідністю product_id з oc_product до oc_product_description та manufacturer_id з oc_product до oc_manufacturer. Умова where обмежує результати лише продуктами зі статусом 1, що означає, що продукт активний та доступний.

Для обробки результатів запиту був написаний код :

```
const pageContents = products.map((product) => {
  const productsResponse = {
    id: product.product_id,
    model: product.model,
    name: product.name,
    category_name: product.category_name,
    description: product.description.replace(/\n|<br>/g, ''),
  };
  return JSON.stringify(productsResponse);
});
```

Рисунок 1.9. Підготовка даних до вбудовування (embedding)

Цей фрагмент коду є частиною функції, яка обробляє дані про продукти перед їх збереженням у базі даних. Код використовує метод map для масиву products, щоб перетворити кожен продукт і повернути трансформований масив у змінну pageContents.

Метод map застосовує функцію до кожного елемента в масиві і повертає новий масив з результатами. У цьому випадку функція приймає аргумент product, що представляє окремий продукт з масиву products.

Усередині функції створюється новий об'єкт productsResponse з використанням синтаксису розповсюдження (...product). Цей синтаксис копіює всі властивості з product до productsResponse. Потім змінюється властивість description: видаляються всі символи нового рядка (\n) і HTML теги переносу рядка (
), використовуючи метод replace з регулярним виразом.

Модифікований description присвоюється властивості description об'єкта productsResponse.

Об'єкт productsResponse перетворюється в JSON-рядок за допомогою JSON.stringify. Результат JSON.stringify(productsResponse) для кожного продукту повертається з функції, що передається до map, утворюючи масив pageContents, який містить JSON-рядки з очищеними описами продуктів.

Підготовка даних для для масової обробки **в** EmbeddingService:

```
const page = pageContents.map((pageContent: any) => ({
  pageContent: JSON.stringify(pageContent),
  metadata: {
    type: 'shop',
    id: pageContent.id,
    model: pageContent.model,
    name: pageContent.name,
    category_name: pageContent.category_name,
  },
}));
await this.embeddingService.createBulk(page);
```

Рисунок 1.10. Скриншот вбудовування товарів у векторну базу даних.

Код трансформує масив pageContents за допомогою функції map. Кожен елемент у pageContents обробляється для створення нового об'єкта. Цей новий об'єкт зберігає оригінальний контент сторінки під ключем pageContent і додає додаткові метадані під ключем metadata, де тип контенту вказано як 'shop'. Це результує в новий масив contents, де кожен елемент є об'єктом зі структурою { pageContent, metadata: { type: 'shop' } }.

Трансформований масив contents передається методу createBulk об'єкта this.embeddingService. Цей метод викликається асинхронно, використовуючи ключове слово await, що означає, що виконання поточної функції призупиняється до завершення методу createBulk. Метод createBulk призначений для обробки масиву об'єктів, кожен з яких представляє дані для вбудовування. Реалізація createBulk включає створення об'єктів документів з наданих даних і їх

збереження у сервісі, ймовірно, PineconeStore, який обробляє індексацію та зберігання для ефективного пошуку та аналізу.

Метою методу createBulk є обробка і зберігання трансформованого контенту сторінок оптом, використовуючи PineconeStore для забезпечення функціональності, такої як пошук або аналіз даних. Ця операція важлива для ефективного управління великими обсягами контенту, особливо в контексті, де такі дані мають бути доступними та пошуковими, наприклад, на платформі інтернет-магазину.

1.6.4 Розробка метода отримання товарів

Опис метода getProducts**:**

```
async getProducts(search: string): Promise<any> {
  const queryEmbeddingDto: QueryEmbeddingDto = {
    query: search,
    metadata: { type: 'shop' },
  };
  const products = await this.embeddingService.query(queryEmbeddingDto, 4);
  const parsedProducts = products.map((product) =>
    JSON.parse(product.pageContent),
  );

  const productNames = parsedProducts.map(
    (product) => JSON.parse(product).name,
  );

  const productIds = parsedProducts.map((product) => JSON.parse(product).id);
  return { productIds, productNames };
}
```

Рис скрин метода getProducts

Наданий фрагмент коду TypeScript — це метод під назвою getProducts в межах класу сервісу, призначений для отримання інформації про продукти з векторної бази даних або зовнішнього сервісу. Цей метод був асинхронним, позначений ключовим словом async, тобто повертав Promise. Метод приймав один параметр, search, який був рядком для запиту продуктів.

Перший крок всередині методу був для створення об'єкта з назвою queryEmbeddingDto типу QueryEmbeddingDto. Цей об'єкт включав пошуковий запит та метадані, що вказували тип запиту, у даному випадку, shop. Цей об'єкт потім передавався до методу query цього embeddingService разом з числом 4, яке, мабуть, обмежувало кількість результатів. Метод query, як описано у відповідній документації, взаємодіяв з векторною базою даних через векторний сховище, виконуючи схожий пошук на основі наданих запиту та метаданих, і повертав список продуктів, що відповідали критеріям.

Результати від методу query зберігалися у змінній products. Кожен елемент у products мав мати властивість pageContent, яка була JSON рядком, що містила деталі продукту. Функція map використовувалася для ітерації через products, розбираючи pageContent кожного продукту, щоб перетворити його з JSON рядка на об'єкт JavaScript. Цей масив об'єктів зберігався в parsedProducts.

Потім виконувалося ще два map операції на parsedProducts. Перша витягувала властивість name з кожного продуктового об'єкта, створюючи масив назв продуктів. Друга витягувала властивість id, створюючи масив ідентифікаторів продуктів. Ці масиви потім упаковувалися в об'єкт з властивостями productIds та productNames, який і повертався методом.

Декілька моментів для можливого покращення або уточнення:

- Метод парсив JSON рядок двічі для кожного продукту, один раз для назви та ще раз для ID. Це можна було б оптимізувати, парсивши один раз і витягуючи обидві частини інформації за одну ітерацію.
- Тип повернення методу був Promise<any>, що не є дуже описовим. Було б корисніше визначити інтерфейс, який описує форму поверненого об'єкта для кращої безпеки типів та зручності читання коду

Реалізація контролера.

Наступним кроком в реалізації проекту була реалізація контролера для отримання продуктів з інтернет-магазину. Для цього був створений файл src/shop/shop.controller.ts і доданий наступний код:

```
import { Controller, Get, Param } from '@nestjs/common';
import { ShopService } from './shop.service';

@Controller('products')
export class ShopController {
  constructor(private shopService: ShopService) {}

  @Get('embedding')
  async embeddingProducts() {
    return this.shopService.embeddingProducts();
  }

  @Get('get/:search')
  async getProducts(@Param('search') search: string): Promise<string[]> {
    return this.shopService.getProducts(search);
  }
}
```

Рисунок 1.11. Контролери для роботи з інтернет-магазином

Цей фрагмент коду TypeScript є частиною бекенд-додатку, який використовує фреймворк NestJS. Він визначає контролер ShopController, що обробляє HTTP-запити, пов'язані з продуктами.

1. Імпорти та декоратори: На початку файлу імпортуються необхідні модулі, включаючи Controller, Get і Param з @nestjs/common, а також ShopService. Декоратор @Controller('products') вказує, що цей контролер обробляє маршрути, які починаються з /products.
2. Визначення класу: Клас ShopController визначений з конструктором, який інжектує екземпляр ShopService.
3. Маршрут для вбудовування продуктів: Декоратор @Get('embedding') визначає обробник для GET-запитів до /products/embedding. Метод викликає embeddingProducts з ShopService для отримання і обробки даних продуктів.

4. Маршрут для отримання продуктів: Декоратор @Get('get/:search') визначає обробник для GET-запитів до /products/get/:search, де search є параметром шляху. Метод getProducts приймає параметр пошуку, викликає відповідний метод у ShopService і повертає результат клієнту.

У підсумку, цей контролер визначає два кінцевих пункти для продуктів: один для вбудовування даних продуктів, інший - для отримання продуктів на основі пошукового терміну. Він використовує ShopService для логіки між цими операціями.

1.6.5 Запуск сервера.

1. Установити всі зависимості `npm i`
2. Для запуска сервера виконала наступну команду для запуску: npm start
3. Перевірили роботу сервера:

Відкрила браузер і перейшла за адресою http://localhost:3000/api, щоб побачити дві кінцеві точки /products/embedding та /products/get/{search}.Скриншот показано на рисунку 1.12.

Ці кроки допомогли мені налаштувати сервер на основі NestJs для інтеграції з OpenCart та створити проект. Налаштувати підключення до бази даних, створити модулі, контролери та сервіси, а також запустити сервер і перевірити його роботу. Використання NestJs дозволило створити масштабований і підтримуваний серверний додаток.

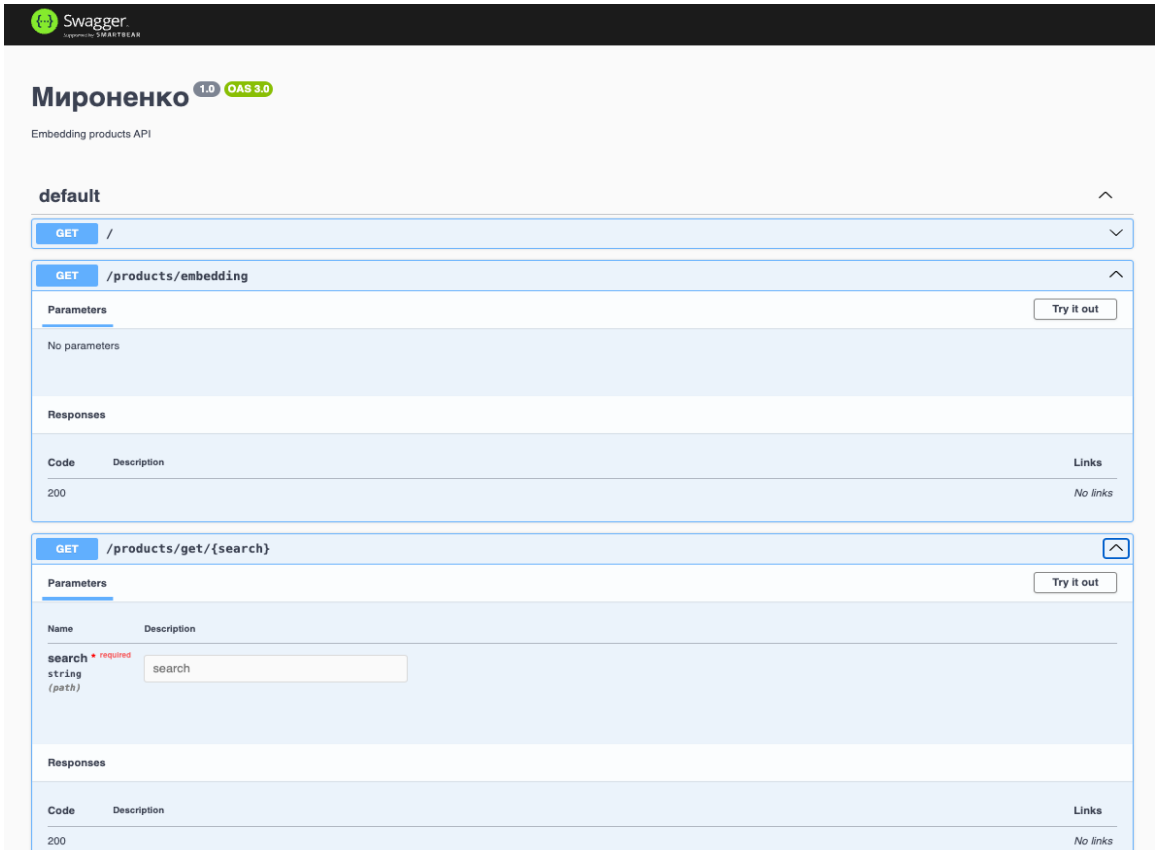


Рисунок 1.12. Скриншот кінцевих точок в swagger

1.6.6 Розгортання сервера на Azure

Наступним кроком було розгорнути цей код на Azure як Web App безпосередньо з локального ПК через сайт Azure.

Інструкція для розгортання на Azure з локального ПК

1. Підготувала проект для розгортання:

Переконалася, що всі необхідні файли проекту знаходяться у моїй локальній директорії.

Переконалася, що проект готовий до розгортання (перевірила, чи немає помилок, виконала локальні тести).

2. Увійшла в Azure Portal:

Перейшла до Azure Portal та увійшла у свій обліковий запис.

3. Створила ресурсну групу (якщо ще не створена):

В Azure Portal перейшла до розділу "Resource groups" і натиснула "Add".

Ввела назву ресурсної групи “diploma”

Натиснула "Review + create" і потім "Create".

4. Створила App Service план:

В Azure Portal перейшла до розділу "App Services" і натиснула "Create".

Вибрала "Web App" та заповнила необхідні поля:

Subscription: обрала свою підписку.

Resource Group: обрала свою ресурсну групу diploma.

Name: ввела унікальне ім'я для додатку diplomaserver.

Publish: вибрала "Code".

Runtime stack: обрала версію Node.js (Node 20 LTS).

Operating System: вибрала "Linux".

Region: обрала регіон East US.

Перейшла до розділу "App Service plan" і натиснула "Create new", заповнила поля та обрала "Free" план.

Натиснула "Review + create" і потім "Create".

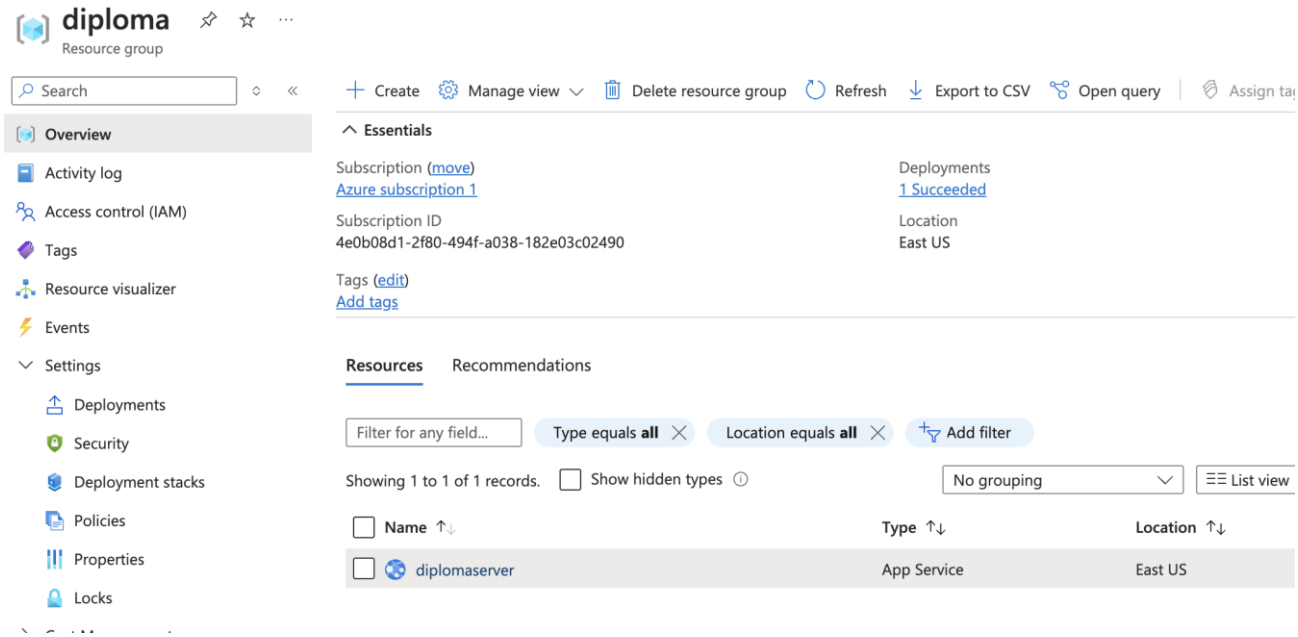


Рисунок 1.13 Огляд ресурсної групи diploma в Azure Portal

5. Встановила Azure CLI (якщо ще не встановлено):
    - Перейшла на [офіційний сайт Azure CLI] і виконала інструкції для встановлення Azure CLI на комп'ютері.
6. Увійшла в Azure CLI:
    - Відкрила термінал і виконала команду для входу в свій обліковий запис Azure:

`az login`

7. Отримала URL для Git-репозиторію:

Після виконання попередньої команди отримала URL для Git-репозиторію, який використовувався для розгортання (https://diplomaserver.scm.azurewebsites.net:443/diplomaserver.git).

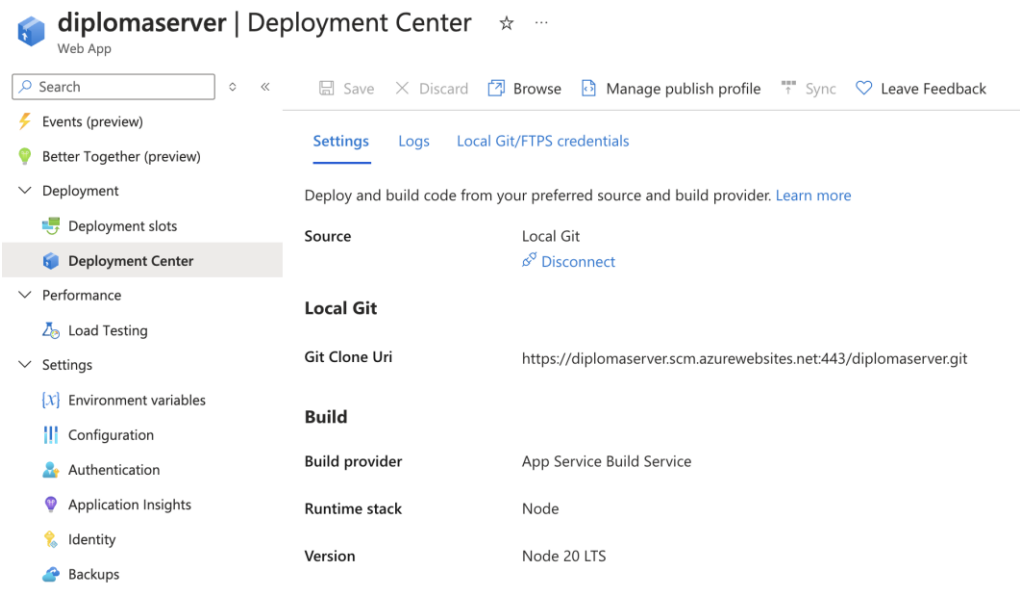


Рисунок 1.14. Огляд Deployment Center в Azure Portal

8. Додала віддалений репозиторій і виконала початковий пуш:
    - Виконала наступні команди в локальній директорії проекту:

git remote add azure https://diplomaserver.scm.azurewebsites.net:443/diplomaserver.git

git add .

git commit -m "Initial commit"

git push azure master

Перевірка розгортання

1. Перевірила статус розгортання в Azure Portal:

Перейшла до Web App в Azure Portal.

Перевірила розділ "Deployment Center" для статусу розгортання.

2. Перевірила роботу додатку:

Відкрила браузер і перейшла за URL Web App (http://diplomaserver.azurewebsites.net/) щоб переконатися, що все працює коректно.

Ці кроки дозволили мені розгорнути проект NestJs на Azure як Web App безпосередньо з мого локального комп'ютера через сайт Azure.

1.6.7 Інтеграция с CMS Opencart

Опис CMS Opencart

OpenCart - це потужна, відкрита система управління контентом (CMS) для електронної комерції, що дозволяє створювати повнофункціональні інтернет-магазини. Вона популярна завдяки своїй простоті використання, широким можливостям налаштування та підтримці безлічі модулів і тем.

Основні переваги OpenCart:

Безкоштовність і відкритий код: Вільно розповсюджується та модифікується.

Простота використання: Інтуїтивно зрозумілий інтерфейс адміністрування. Масштабованість: Підтримка великої кількості продуктів та категорій.

Багатомовність і мультивалютність: Можливість налаштування кількох мов та валют.

Широкі можливості налаштування**:** Підтримка тем та модулів для розширення функціоналу.

Архітектура

OpenCart побудована на основі традиційної архітектури Model-View-Controller (MVC), що розділяє логіку додатку на три основні компоненти: модель, подання та контролер.

1. Модель (Model):

Відповідає за взаємодію з базою даних.

Містить логіку для отримання, збереження та обробки даних.

Файли моделей зазвичай розташовані в каталозі catalog/model/.

2. Подання (View):

Відповідає за відображення даних користувачам.

Містить HTML-шаблони, стилі CSS та інший фронтенд-код.

Файли подань розташовані в каталозі catalog/view/theme/.

3. Контролер (Controller):

Обробляє запити користувачів і взаємодіє з моделями.

Визначає, які дані отримувати і яке подання використовувати для їх відображення.

Файли контролерів розташовані в каталозі catalog/controller/.

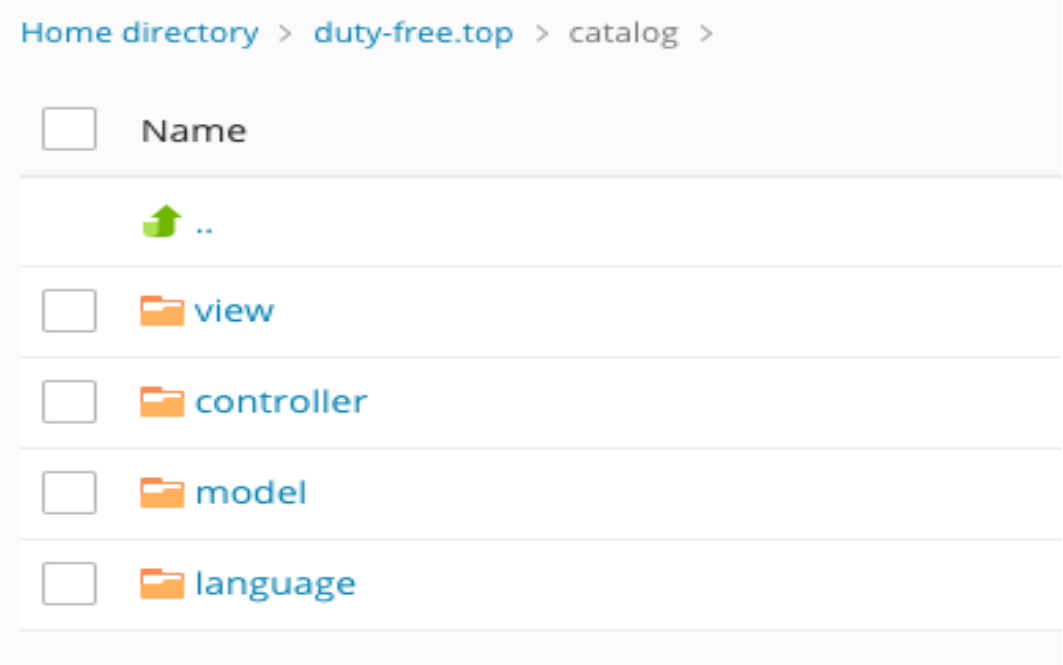


Рисунок 1.15 Архітектури Model-View-Controller (MVC) CMS Opencart

Потік виконання запиту

1. Запит користувача: Користувач здійснює запит до веб-сайту (наприклад, відкриває сторінку продукту).
2. Роутер: OpenCart роутер визначає, який контролер оброблятиме цей запит.
3. Контролер: Відповідний контролер отримує запит та взаємодіє з моделями для отримання необхідних даних.
4. Модель: Модель отримує дані з бази даних.
5. Контролер: Контролер передає дані до подання.
6. Подання: Подання формує HTML-код сторінки, яка буде показана користувачу.
7. Відповідь: Готова сторінка відправляється користувачу.

OpenCart є гнучким інструментом для створення і управління інтернет-магазинами, пропонуючи всі необхідні функції для успішної електронної комерції.

Налаштування CMS Opencart

У інтернет-магазині OpenCart, пошуковий запит передається як параметр у URL-адресі. Щоб знайти, під яким іменем передається пошуковий запит на сторінці інтернет-магазину, ви можете переглянути вихідний код сторінки або перевірити URL-адресу після введення пошукового запиту. Найдемо ім'я параметра пошукового запиту:

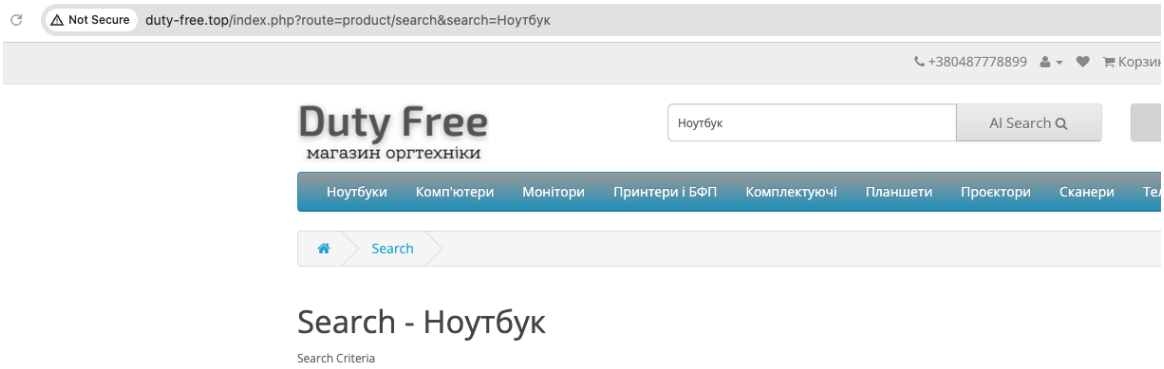
Перевірка URL-адреси:

Відкрийте інтернет-магазин і введіть пошуковий запит у поле пошуку.

Натисніть Enter або кнопку пошуку.

Подивіться на URL-адресу у браузері. Вона повинна містити параметр, який передає пошуковий запит.

Наприклад:http://duty-free.top/index.php?route=product/search&
search=Ноутбук, де search - це ім'я параметра пошукового запиту



- Рисунок1.16. Скриншот пошукового запита.

-

Щоб знайти, під яким іменем передається пошуковий запит у контролері OpenCart, виконайте наступні кроки:

1. Відкрийте файл контролера пошуку:

Зазвичай контролер, який обробляє пошукові запити, знаходиться у файлі catalog/controller/product/search.php.

2. Знайдіть метод, який обробляє пошукові запити:

Перегляньте метод, який отримує значення параметра пошуку з запиту.

3. Знайдіть рядок коду, який отримує значення пошукового запиту:

Пошуковий запит зазвичай отримується через глобальний масив $_GET або через метод $this->request->get

Code Editor

search.php

```
197         $data['products'] = array();
198
199         if (isset($this->request->get['search']) || isset($this->request->get['tag'])) {
200             $filter_data = array(
201                 'filter_name'         => $search,
202                 'filter_tag'          => $tag,
203                 'filter_description'  => $description,
204                 'filter_category_id'  => $category_id,
205                 'filter_sub_category' => $sub_category,
206                 'sort'                => $sort,
207                 'order'               => $order,
208                 'start'               => ($page - 1) * $limit,
209                 'limit'               => $limit
210             );
211
212             $product_total = $this->model_catalog_product->getTotalProducts($filter_data);
213
214             $results = $this->model_catalog_product->getProducts($filter_data);
215
```

Рисунок 1.17. Контролер search в Opencart

Нас тут цікавлять два методи в моделях, які відповідають за пошук товарів у базі.$product_total=$this->model_catalog_product-getTotalProducts($filter_data);
$results = $this->model_catalog_product->getProducts($filter_data);
$filter_data = array('filter_name' => $search,

Створили метод який робе запит на ендпоінтд http://diplomaserver.azurewebsites.net/api#/default/ShopController_getProducts для отримання даних моделей товарів.

Скриншот кода показаний на рисунку 1.18.

Рисунок 1.18. Скриншот ендпоінтда для отримання даних

Функція getProductFromAPI:

Функція приймає параметр filter_name і створює URL для запиту, кодувавши filter_name за допомогою urlencode.

Ініціалізує cURL за допомогою curl_init.

Встановлює URL для запиту за допомогою curl_setopt.

Встановлює параметр CURLOPT_RETURNTRANSFER у значення 1, щоб результат запиту було повернено як рядок.

Встановлює заголовок accept за допомогою curl_setopt та масиву заголовків.

Виконує запит за допомогою curl_exec і зберігає відповідь.

Обробляє помилки cURL за допомогою curl_errno та curl_error.

Закриває cURL-сесію за допомогою curl_close.

Декодує JSON-відповідь і повертає її як масив.

Приклад використання функції.

Визначається змінна filter_name зі значенням ноутбук.

Виклик функції getProductFromAPI з цим значенням.

Вивід результатів запиту за допомогою print_r. Цей код виконає запит до вказаного URL і поверне масив продуктів, що відповідають фільтру.

Додаемо визов функции getProductFromAPI

product.php

```
178                 $sql .= " OR LCASE(p.ean) = '" . $this->db->escape(utf8_strtolower($data['filter_name']))
179                 $sql .= " OR LCASE(p.jan) = '" . $this->db->escape(utf8_strtolower($data['filter_name']))
180                 $sql .= " OR LCASE(p.isbn) = '" . $this->db->escape(utf8_strtolower($data['filter_name']))
181                 $sql .= " OR LCASE(p.mpn) = '" . $this->db->escape(utf8_strtolower($data['filter_name']))
182
183                 // Отримуємо продукти з API
184                 $products_from_api = $this->getProductFromAPI($data['filter_name']);
185
186                 // Перевіряємо, чи відповідь не порожня
187                 if (!empty($products_from_api)) {
188                     //Додавання умов до SQL-запиту
189                     $sql .= " OR p.product_id IN (" . implode(', ', $products_from_api) . ")";
190                 }
191             }
192
```

Рисунок 1.19. Добаваня списка id продуктів у запит в базу даних Опенкарт

Отримання продуктів з API:

getProductFromAPI($data['filter_name']): Викликається метод getProductFromAPI, передаючи йому параметр filter_name з масиву $data. Цей метод надсилає запит до зовнішнього API для отримання списку продуктів, що відповідають заданому фільтру.

products_from_api: Зберігає результат виклику методу getProductFromAPI. Це буде масив, який містить інформацію про продукти, отримані з API.

Перевірка на непорожність:

!empty($products_from_api): Використовується функція empty(), щоб перевірити, чи не є масив products_from_api порожнім. Якщо масив не порожній, тобто API повернув хоча б один продукт, виконуються наступні дії.

Додавання умов до SQL-запиту:

implode(', ', $products_from_api): Функція implode() об'єднує елементи масиву products_from_api у строку, де кожен елемент розділений комою і пробілом. Цей метод використовується для створення списку product_id, який можна використовувати у SQL-запиті.

$sql .= " OR p.product_id IN (" . implode(', ', $products_from_api) . ")";: Якщо масив products_from_api не порожній, до SQL-запиту додається умова OR p.product_id IN (...), де ... - це список product_id, отриманих з API. Ця умова дозволяє SQL-запиту вибирати тільки ті продукти, product_id яких містяться у списку, отриманому з API.

1.7 Робота програми.

1.7.1 Вбудування данних в векторну базу даних

Робимо запит на кінцеву точку. програма достає усі товари з бази даних інтернет магазину показано на рисунку 1.17 і зберігає їх у векторну базу даних. Скриншот векторної бази даних показано на рисунку 1.18.

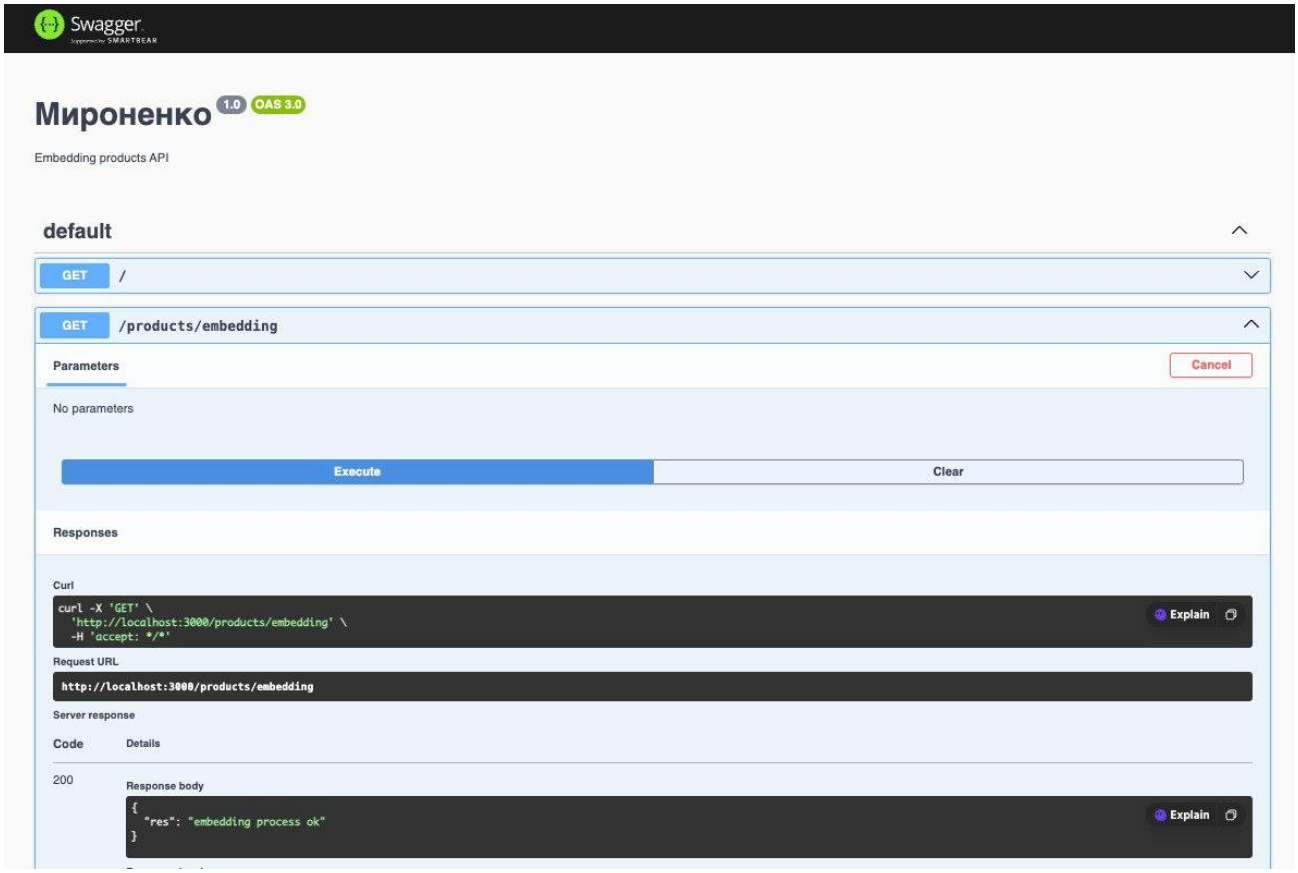


Рисунок 1.20. Робота програми

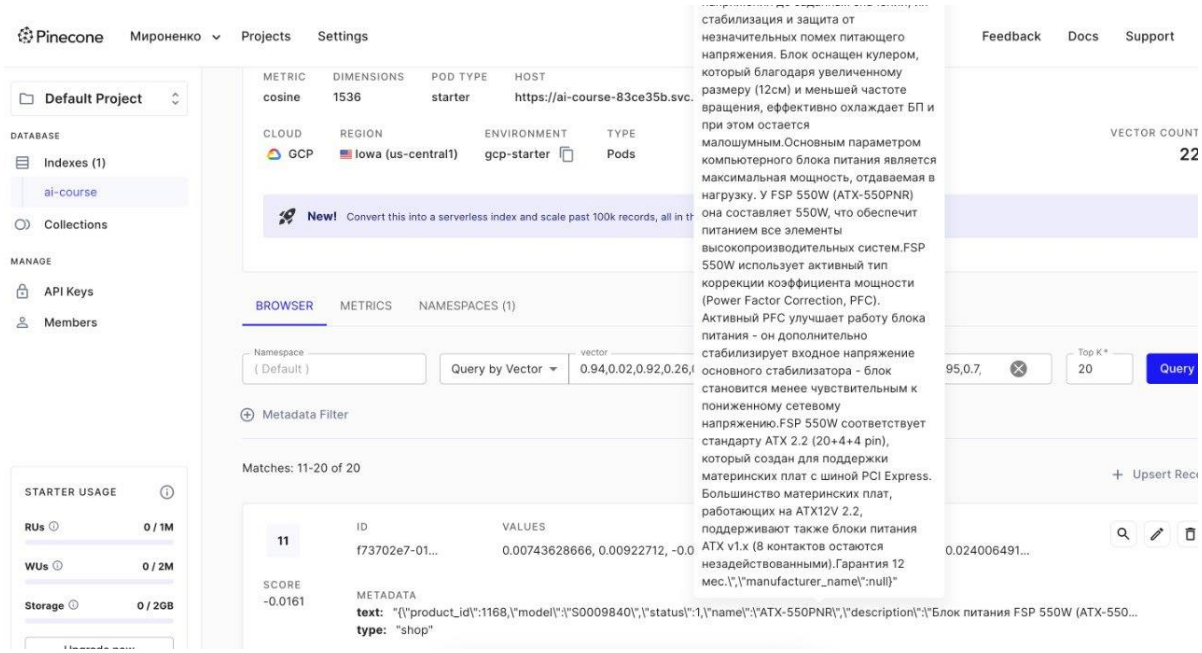


Рисунок1.21. Скриншот векторної бази даних

1.7.2 Приклад роботи інтелектуального пошуку на сайті

Користувач вводить пошуковий запит у вікно пошуку, і програма ініціює запит до векторної бази даних. У цій базі відбувається семантичний пошук товарів. Система OpenCart отримує перелік назв і моделей відповідних товарів. Наступний крок - це запит до бази даних OpenCart за знайденими товарами. Таким чином, наш інтелектуальний пошук дозволяє знаходити товари не тільки за точною відповідністю у назві, а й за семантичним змістом. Результат роботи показано на рисунку 1.19.

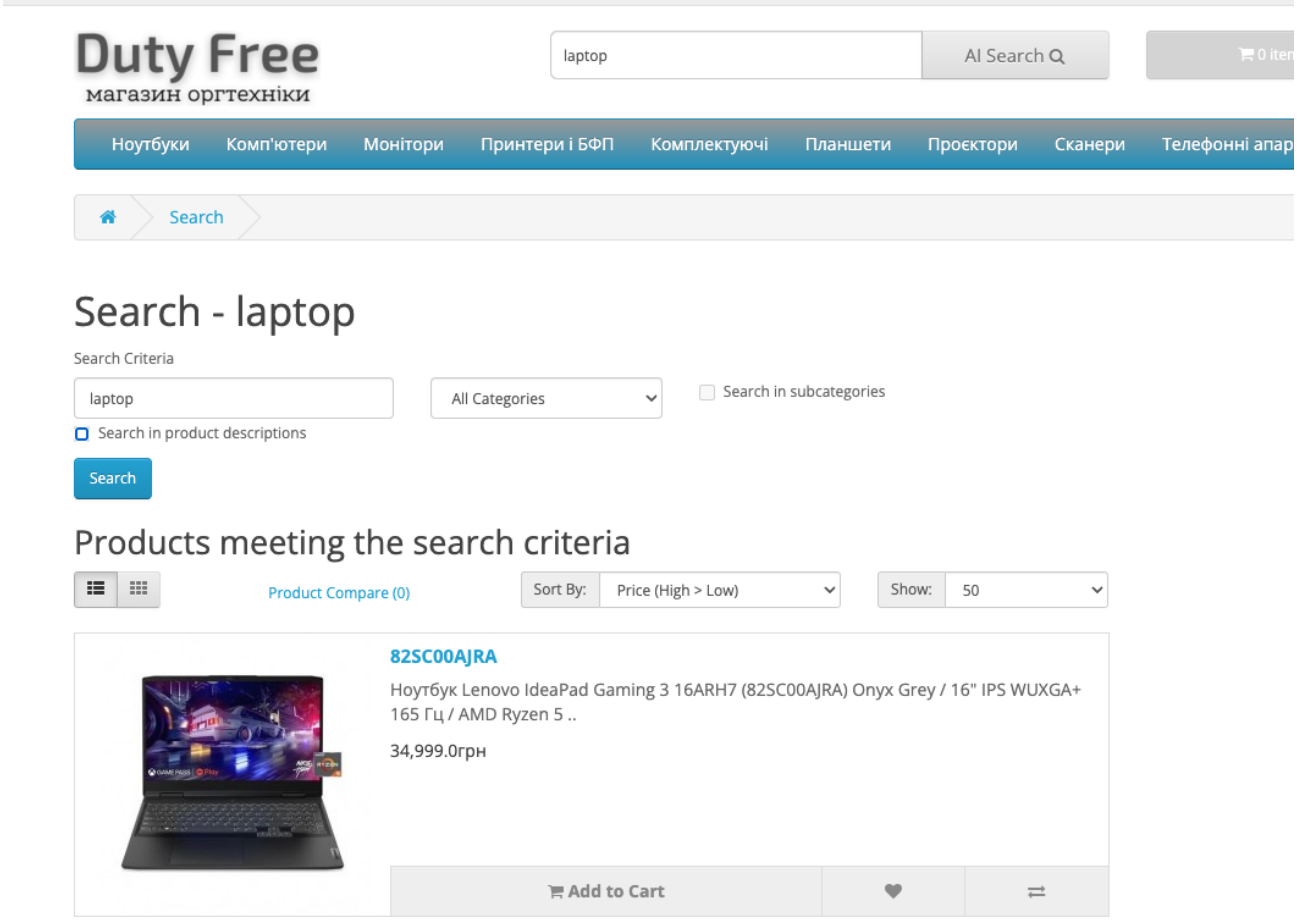


Рисунок 1.19. Скриншот результату виконання

На цьому скрині ми бачимо, що користувач зробив пошуковий запит laptop. У

назві продукту і в описі немає прямого збігу з цим словом, але завдяки семантичному пошуку по векторній базі даних система змогла знайти релевантні результати. Семантичний пошук дозволяє знаходити продукти, які можуть відповідати запиту користувача за змістом, навіть якщо конкретне слово не присутнє в назві чи описі продукту.

Ця технологія покращує результати пошуку, роблячи їх більш точними та релевантними для користувача.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Темою даного дипломного проекту є «Розробка інтелектуального пошуку для сайту с продажу КТ». Метою ДП є розробка інтелектуальної пошукової системи для сайту з продажу комп'ютерної техніки, яка забезпечуватиме високоточний і зручний пошук товарів, покращуючи користувацький досвід. Ця система повинна автоматично аналізувати запити користувачів, враховувати їхні потреби та пропонувати найбільш релевантні результати.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначаємо обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт.

Таблиця 2.1 Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП СУБД	2500 – 9800
2. Комплексні системи ведення БД	950 – 7430
3. ПП організації обчислювального процесу	13000 – 10200

Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): $T^a = 229 \times 0,8 = 183,2$ (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4.).

Таблиця 2.2.Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Кодступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а *L_1*К_н= 183,2 *0,12*0,7= 15,39 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а *L_2*К_н =183.2 *0,11*0,7 = 17,42 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Т^а *L_3*К_н*К_т =183,2 *0,61*0,7*0,7=54,76 \text{ (люд/годин)} \quad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стор), розробка ТП $N_{тп}$=18 (стор), розробка робочого проекту $N_{рп}$=25(стор), пояснювальна записка відповідно $N_{пз}$= 10 (стор) Розрахунок зведений у таблицю 2.5

Таблиця 2.5. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Ртз}$=15,39	$T_{кк}=0,7*N_{тз}=0,7*2=1,4$	$T_{нк}=0,15*N_{тз}=0,15*2=0,30$
2.Розробка ТП	$T_{Ртп}$=14,12	$T_{кк}=0,7*N_{тп}=0,7*18=12,6$	$T_{нк}=0,15*N_{тп}=0,15*18=2,7$
3.Розробка РП	$T_{Ррп}$=54,76	$T_{кк}=0,7*N_{рп}=0,7*25=17,5$	$T_{нк}=0,15*N_{рп}=0,15*25=3.8$
4.Розробка ПЗ	$T_{пз}=1,5*$ $*N_{пз}=1,5*10=15$	$T_{кк}=0,7*N_{тз}=0,7*10=7$	$T_{нк}=0,15*N_{пз}=0,15*10=1,5$
Усього, в т.ч.:	ΣT=146,1		
- на розробку	$\Sigma T_р$=99,3		
- контроль керівника		$\Sigma T_{кк}$=38,5	
- нормоконтроль			$\Sigma T_{нк}$=8,3

2.3 Розрахунок ціни програмного продукту.

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.6. Відповідно до статті 8 «Закону про Державний бюджет України на 2024» встановлено мінімальну заробітну плату у місячному розмірі з 1 квітня 2024 року - 8000 гривень; мінімальну погодинну тарифну ставку – 46,00 грн.

Таблиця 2.6 Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	99,3	46.00	4567,80
2.Контроль керівника	38,5	100.00	3850,00
3.Нормоконт-роль	8,3	100.00	830,00
Усього	-	-	ΣЗо= 9247,80

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.7

Таблиця 2.7 Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	55	4.0	220
Разом	-	-	-	$В_{м\,і}$ =220
Транспортно– заготівельні витрати (10%)				$Втр\_з = 0{,}1 \times Вм1 = 22$
Усього				$В_м = В_{мі} + В_{тр\_з} =$ 242

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.8.

Таблиця 2.8. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	242,00	$В_м$ (див. табл. 2.7)
2. Основна заробітна плата	9247,80	$З_о$ (див. табл. 2.6)
3.Додаткова заробітна плата	1387,17	$Зд = 0{,}15 \times Зо$ =0,15*9247,80
4.Відрахування до єдиного фонду соціального внеску	2339,69	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (9247,80+1387,17)
5. Накладні витрати	2774,34	$Внак. = 0{,}3 \times Зо$ =0,3* 9247,80
6. Повна собівартість	15991,00	$С_{пов} = В_м + З_о + З_д$ +Вє.с.в. + Внак.= 242,00+9247,80+1387,17+2339,69+2774 ,34

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_п*Р)/100= \quad (15991,00*10)/100=1599,10грн \qquad (2.4)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о = С_{пов}+П = \quad 15991,00+ 1599,10 \text{ грн} = 17590,10 \text{ грн} \qquad (2.5)$$

Податок на додану вартість визначаємо по наступній формулі:

$$ПДВ = 0.2 * Ц_о = 0,2*17590,10 = 3518,02 \text{ грн} \qquad (2.6)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р=Ц_о+ПДВ \ = 17590,10 + 3518,02 = \quad 21108,12 \text{ грн} \qquad (2.7)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Закон України «Про охорони праці» - є одним із найважливіших законодавчих актів. Цей закон визначає основні положення щодо реалізації конституційного права громадян на охорону їх життя і здоров'я в процесі трудової діяльності, регулює відносини між власником підприємства або уповноваженим ним органом і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок охорони праці в України.

Згідно з діючим законодавством на всіх підприємствах, в установах, організаціях власниками створюються безпечні і нешкідливі умови праці.

Умови праці на робочому місці, безпека технологічних процесів, робота машин, механізмів, устаткування, стан засобів, колективного та індивідуального захисту, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці.

В дипломному проекті використовується праця користувача ПК, тому умови праці та забезпечення безпеки його роботи здійснюються при дотриманні всіх норм та вимог безпечного використання комп'ютера і його можливостей для працівника.

Забезпечення охорони праці на виробництві завжди було надзвичайно важливим. Завдяки рекомендаціям з охорони праці, персонал на підприємстві створює алгоритм виконання робочих завдань з чітким дотриманням цих рекомендацій. Основне завдання охорони праці – розробка та впровадження заходів, спрямованих на захист життя, працездатності та здоров'я працівників під час трудової діяльності.

При роботі з комп'ютером, як і в багатьох інших галузях, повинні враховуватись нормативи освітлення, температури, відносної вологості та сили вібрації. Однак у приміщеннях з комп'ютерами найважливішим є дотримання правил пожежної безпеки, таких як вогнестійкість приміщення, а також

контроль рівня звукового шуму та характеристик електромагнітних, ультрафіолетових та інфрачервоних полів. Для аналізу охорони праці у дипломному проєкті досліджується безпека праці розробника веб-сторінок в офісному приміщенні.

3.1 Аналіз та безпека умов праці працівника на робочому місці

Під час роботи за комп’ютером на працівника можуть впливати небезпечні фактори виробничого середовища, зокрема фізичні та психофізіологічні небезпечні й шкідливі виробничі фактори.

Серед фізичних небезпечних факторів найпоширеніші такі: підвищена температура повітря робочої зони, підвищений рівень шуму, знижена вологість повітря. Ці фактори часто виникають при роботі у приміщеннях з комп’ютерами, оскільки їх робота спричиняє підвищення температури та зниження вологості повітря. Крім того, комп’ютери випромінюють електростатичні та електромагнітні поля в діапазоні від 5 Гц до 2 кГц та від 2 до 400 кГц, що призводить до підвищеного рівня електромагнітного випромінювання та статичної електрики. У офісних приміщеннях часто недостатньо природного освітлення, через що використовується штучне освітлення, яке не завжди правильно налаштоване, що може призвести до недостатньої або надмірної яскравості світла.

Психофізіологічні небезпечні фактори виробництва поділяються на фізичні та нервово-психічні перевантаження. При роботі з комп’ютером найчастіше виникають нервово-психічні перевантаження. До них відносяться перенапруга аналізаторів, монотонність праці, а також розмовна перенапруга, коли програмісту потрібно складати технічне завдання разом з клієнтом.

3.2 Розробка заходів з охорони праці

Виробниче освітлення.

Дипломним проектом враховані санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів.

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ВДТ має здійснюватися системою загального рівномірного освітлення. У виробничих та адміністративно-громадських приміщеннях, у разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення, додатково встановлюються світильники місцевого освітлення).

Мікроклімат

При роботі у приміщеннях з великою кількістю комп'ютерів, приміщення з якими класифікуються як приміщення з підвищеною небезпекою електротравм, температура повітря влітку може становити більше 35 С, що погано впливає на здоров'я людини, тож у таких приміщеннях повітря повинне охолоджуватись та понижена вологість повітря повинна регулюватись спеціальним обладнанням.

Відповідно до норм ДСН 3.3.6.042-99 температура повітря в офісі повинна становити 22-25 С, вологість повітря 40-60%, швидкість руху повітря не більше 0,1 м/с. Якщо ці норми перевищені, робочій день працівника повинен бути скорочений на 10%.

Площа приміщення має бути не менше 6,0 кв. м. із розрахунку на одне робоче місце, а об'єм – не менше 20,0 куб. м.

Віконні прорізи приміщень для роботи з персональними комп'ютерами мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки. Для внутрішнього оздоблення приміщень з персональними

комп‘ютерами слід використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі 0,7-0,8, для стін 0,5-0,6.

Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5. Поверхня підлоги має бути рівною, неслизькою, з антистатичними властивостями. Забороняється для оздоблення інтер’єру приміщень з персональними комп‘ютерами застосовувати полімерні матеріали (деревинно-стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини. Приміщення можуть обладнуватись шафами для зберігання документів, магнітних дисків, полицями, стелажами, тумбами тощо з урахуванням вимог до площі приміщень.

Розмір одного робочого місця має становити не менше 6 квадратних метрів. На столі працівника можливо розмістити допоміжні для роботи пристрої (принтери, колонки, сканери), а також місця для зберігання документів, за умови, що це не обмежуватиме видимість екрану і не заважатиме працівнику. Робочий стілець робітника має бути підйомно-поворотним, легко регульованим за висотою та забезпечувати належну підтримку та зручне положення спини і хребта особи

При початку роботи працівник має пройти лікарський огляд. Окрім того, при подальшій трудовій діяльності, працівник підлягає регулярному лікарському огляду не рідше ніж раз на 2 роки. Обов’язковим є проходження таких лікарів як терапевта, невропатолога та офтальмолога. Мають бути чітко встановлені перерви для відпочинку працівників (окрім обідньої), як правило, тривалістю 10-15 хвилин раз на годину або дві, в залежності від складності роботи. В будь-якому випадку, повинен передбачатися такий розпорядок роботи, щоб час непреривної роботи з комп’ютером був не більше ніж 4 години. Додатково, для збереження належного рівня здоров’я та професійної придатності робітників, рекомендується виділити окреме побутове приміщення.

Пожежна безпека

Пожежна безпека може бути забезпечена заходами пожежної профілактики і активного пожежного захисту. Пожежна профілактика включає комплекс заходів, спрямованих на попередження пожежі або зменшення його наслідків.

Пожежна безпека при роботі з комп'ютером передбачає обережність при обслуговуванні, ремонтних та профілактичних роботах та виконування всіх інструкцій працівником щодо пожежної безпеки. Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу. Приміщення, де розміщені робочі місця, мають бути оснащені системою автоматичної пожежної сигналізації і вогнегасниками відповідно до вимог чинного законодавства України. Проходи до засобів пожежогасіння мають бути вільними

Всі приміщення повинні бути забезпечені первинними засобами пожежогасіння: пожежним водопостачанням (пожежні крани ПК), пожежні щити з набором пожежного інструменту, вуглекислотними або порошковими

У випадку виникнення пожежі необхідно відключити електроживлення, викликати по телефону 101 пожежну команду, евакуювати людей із приміщення відповідно до плану евакуації і приступити до ліквідації пожежі.

Рисунок 3.1. Плакат протипожежної безпеки

ВИСНОВКИ

При створені програми-конструктора було сформувано базу даних комплектуючих, проведено ранжування їх за різними критеріями, сформована множина класів завдань, з якими буде працювати комп'ютер у майбутньому, і вирішино задачу класифікації для ефективного підбору комплектуючих. Програму-конструктор для підбору компонентів ПК з додатковою функціональністю, яка включає: структуру сторінки з навігаційним меню, область вмісту та боковим меню з компонентами та фотографіями компонентів ПК. При натисканні на пункти бокового меню відображається список компонентів з фотографіями, описом кожного та ціною.

Програма інтелектуального пошуку по підбору комплектуючих КТ. Ця інформаційна система підтримує продукти, такі як Microsoft Office та SQL Server, для розробки документації системи. Розробка програми-конструктора здійснювалася за допомогою мов програмування, таких як PHP, HTML 5, JavaScript, CSS та CMS OpenCart. Для забезпечення нашої системи інформаційними ресурсами була створена база даних MySQL, яка містить таблиці для опису та зберігання інформації про комплектуючі ПК. Також для перевірки працездатності програми використовувалися браузери Google Chrome і Mozilla Firefox.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 1. Ресурс Інтернету: ITVDN. URL: https://itvdn.com (дата звернення 02.05.2024).
2. 2. Ресурс Інтернету: Webner Solutions Blog. "Visual Studio Code: An Overview and Key Features." URL: https://blog.webnersolutions.com/visual-studio-code/ (дата звернення 04.05.2024).
3. 3. Ресурс Інтернету: Webuild.io. "Why Use Figma for Digital Product Design?" URL:https://webuild.io/why-use-figma-for-digital-product-design (дата звернення 04.05.2024).
4. 4. Ресурс Інтернету: General Assembly Blog. "What Is a JavaScript Framework?" URL: https://generalassemb.ly/blog/what-is-a-javascript-framework/ (дата звернення 04.05.2024).
5. 5. Ресурс Інтернету: Monocubed Blog. "Advantages of Vue.js." URL: https://www.monocubed.com/blog/advantages-of-vue-js/ (дата звернення 03.05.2024).
6. 6. Ресурс Інтернету: TutorialsPoint. "Laravel - Overview." URL: https://www.tutorialspoint.com/laravel/laravel_overview.htm (дата звернення 04.05.2024).
7. 7. Ресурс Інтернету: HubSpot Blog. "What Is Tailwind CSS?" URL: https://blog.hubspot.com/website/what-is-tailwind-css (дата звернення 04.05.2024).
8. 8.Ресурс Інтернету: JavaTpoint. "Git Tutorial." URL: https://www.javatpoint.com/git (дата звернення 14.05.2024).
9. 9. Ресурс Інтернету: Atomic Design by Brad Frost. "Chapter 2: Atomic Design Methodology." URL:https://atomicdesign.bradfrost.com/chapter-2 (дата звернення 26.05.2024).

ДОДАТОК А

Код реалізації сервісу для продуктів.

```
import { Injectable } from '@nestjs/common';
import { Knex } from 'knex';
import { InjectConnection } from 'nestjs-knex';
import { EmbeddingService } from '../embedding/embedding.service';

@Injectable()
export class ShopService {
  constructor(
    private embeddingService: EmbeddingService,
    @InjectConnection() private readonly knex: Knex,
  ) {}
  async getAll() {
    const categories = await this.knex
      .select(
        'oc_product.product_id as product_id',
        'oc_product.model as model',
        'oc_product.status as status',
        'oc_product_description.name as name',
        'oc_product_description.description as description',
        'oc_manufacturer.name as manufacturer_name',
      )
      .from('oc_product')
      .leftJoin(
        'oc_product_description',
        'oc_product.product_id',
        'oc_product_description.product_id',
      )
      .leftJoin(
        'oc_manufacturer',
        'oc_product.manufacturer_id',
        'oc_manufacturer.manufacturer_id',
      )
      .where({
        status: 1,
      })
      .limit(20)
      .offset(0);

    const pageContents = categories.map((category) => {
```

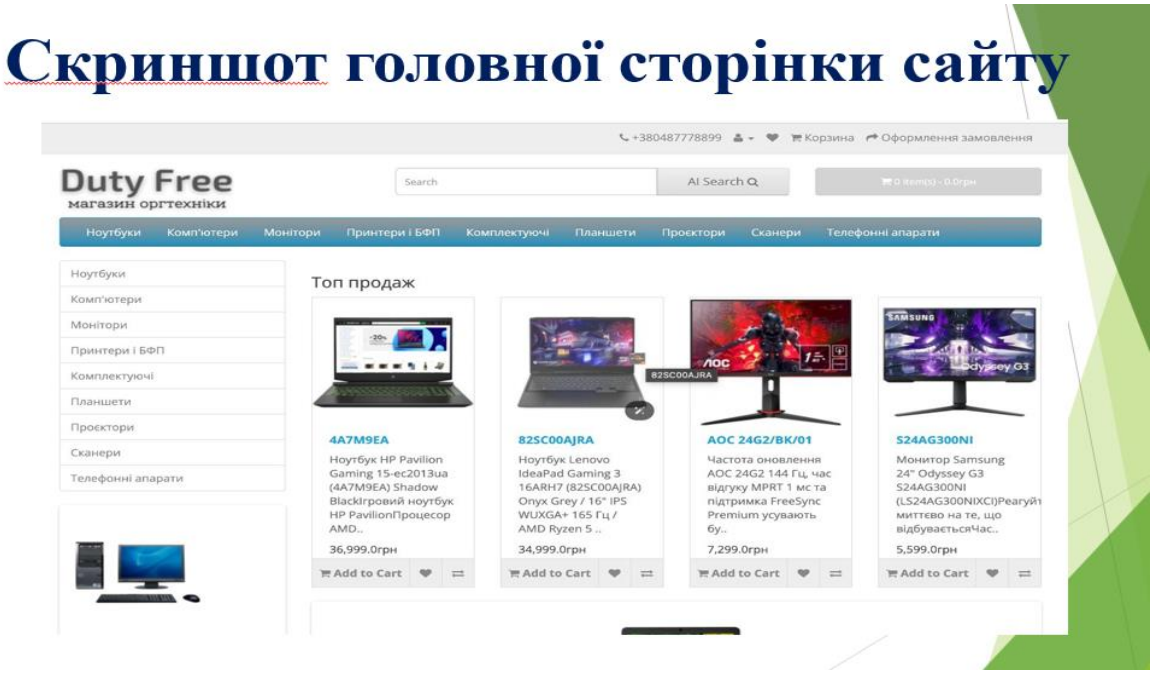
```
      const categoryResponse = {
        ...category,
        description: category.description.replace(/\n|<br>/g, ''),
      };
      return JSON.stringify(categoryResponse);
    });
    console.log(pageContents);

    const contents = pageContents.map((pageContent) => ({
      pageContent,
      metadata: {
        type: 'shop',
      },
    }));
    await this.embeddingService.createBulk(contents);
    return { res: 'embedding process ok' };
  }
}
```

Додаток А.

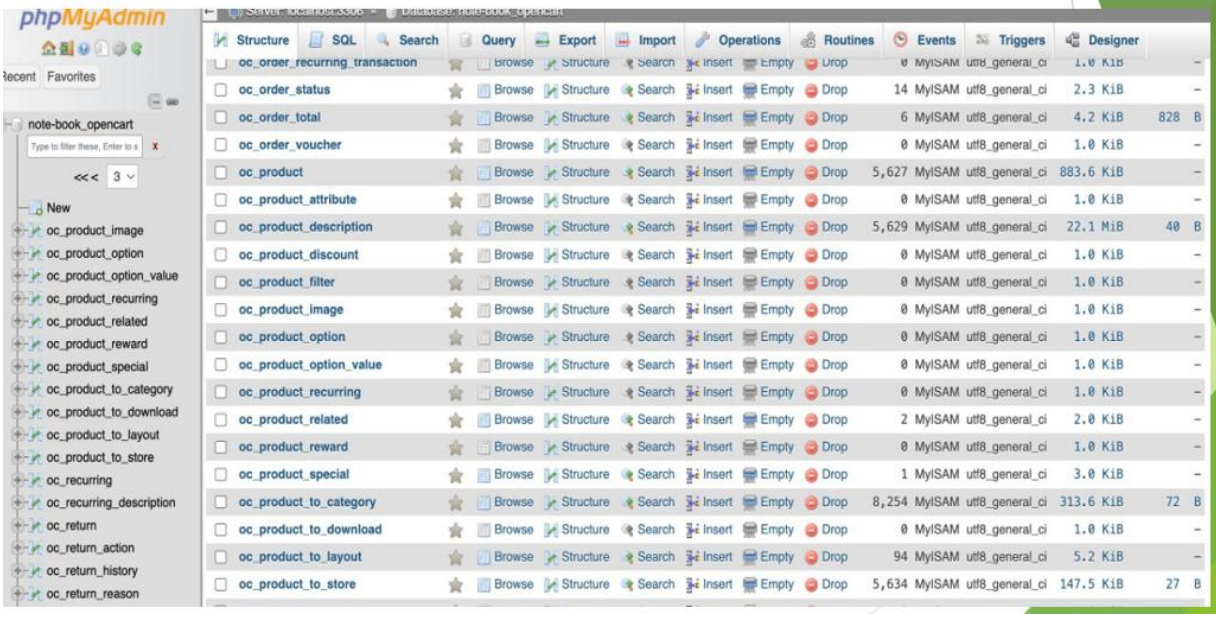
СЛАЙДИ МУЛЬТИМЕДІЙНОЇ ПРЕЗЕНТАЦІЇ

Слайд 1



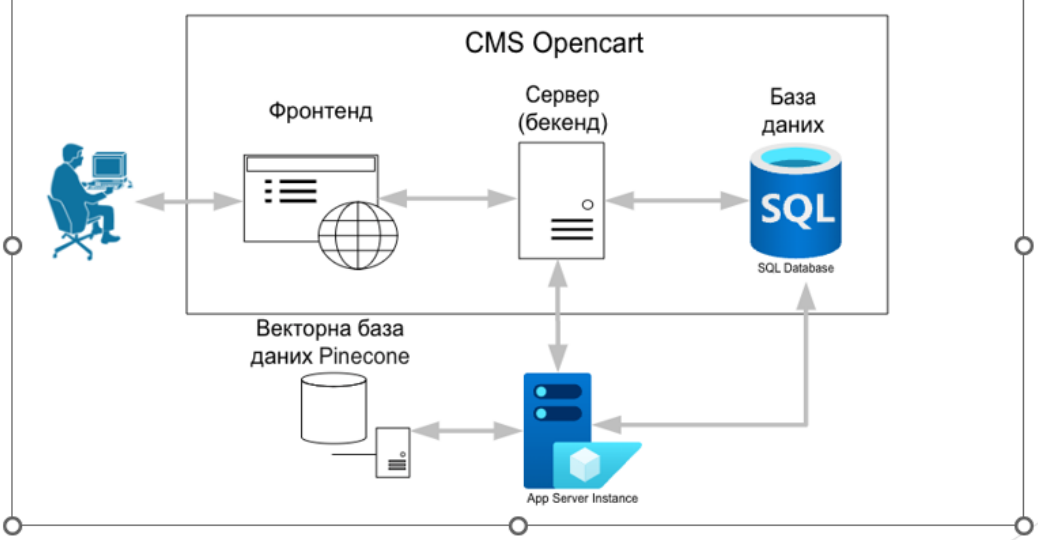
Слайд 2

Скриншот бази даних сайту



Слайд 3

Архітектура системи



Слайд 4

Скриншот налаштування бібліотеки Knex

knexfile.ts > ...

```ts
import { config } from 'dotenv';
config();

export default {
  local: {
    client: 'mysql',
    connection: {
      host: process.env.DB_HOST,
      user: process.env.DB_USER,
      password: process.env.DB_PASSWORD,
      database: process.env.DB_NAME,
    },
    pool: { min: 0, max: 7 },
  },
};

```

Слайд 5

Скриншот модуля для продуктів в інтернет магазині

op > shop.module.ts > ...

```ts
import { Module } from '@nestjs/common';
import { ShopController } from './shop.controller';
import { ShopService } from './shop.service';
import { EmbeddingService } from '../embedding/embedding.service';
import { OpenaiService } from '../openai/openai.service';

@Module({
  controllers: [ShopController],
  providers: [ShopService, EmbeddingService, OpenaiService],
})
export class ShopModule {}
```

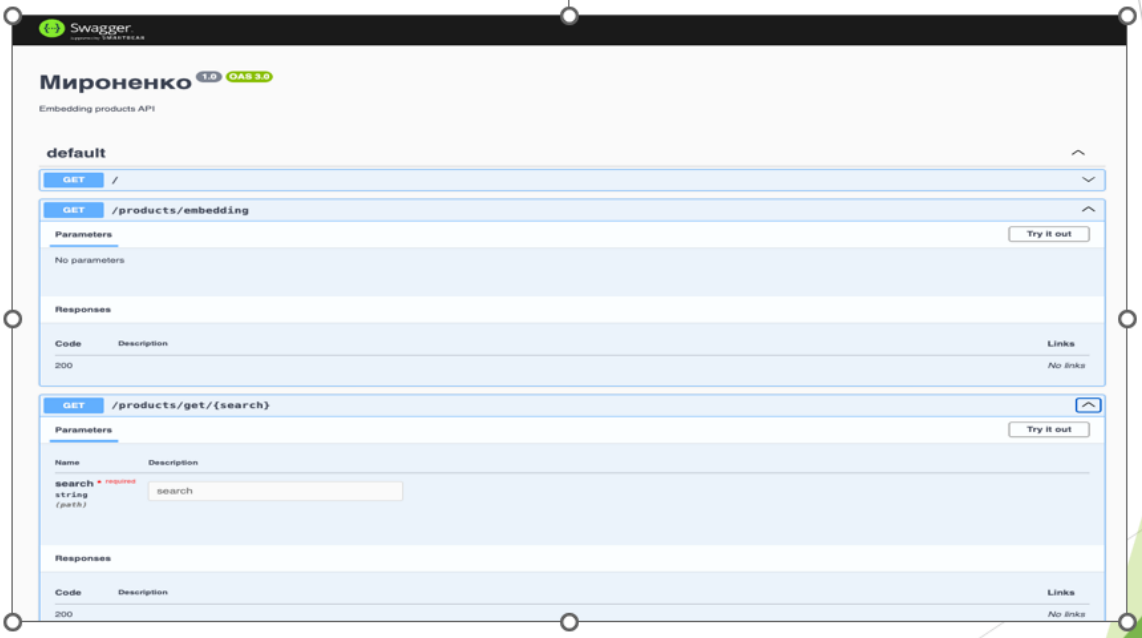
Слайд 6

Скриншот воудовування товарів у векторну базу даних

```
const page = pageContents.map((pageContent: any) => ({
  pageContent: JSON.stringify(pageContent),
  metadata: {
    type: 'shop',
    id: pageContent.id,
    model: pageContent.model,
    name: pageContent.name,
    category_name: pageContent.category_name,
  },
}));
await this.embeddingService.createBulk(page);
```

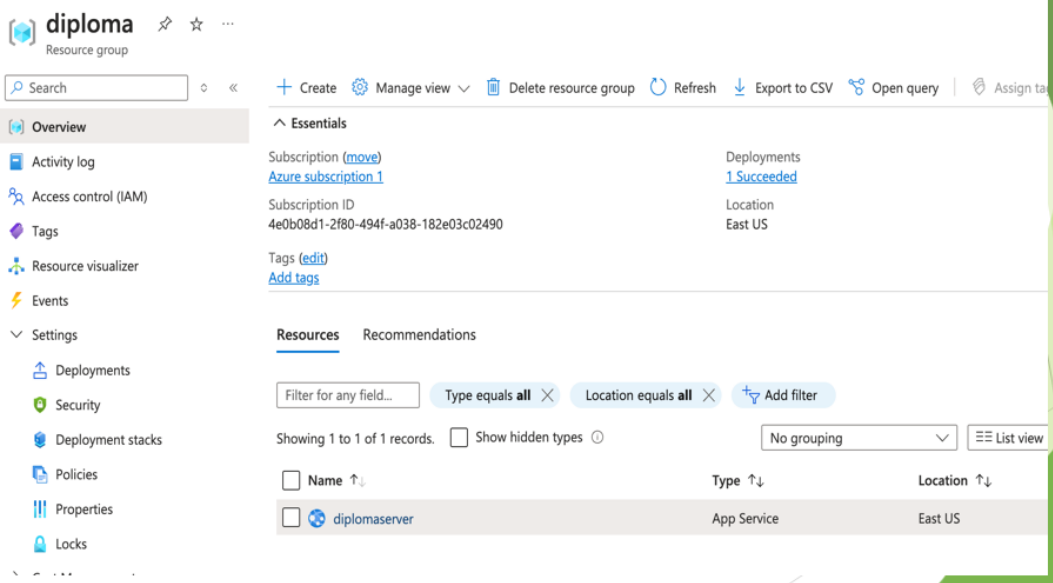
Слайд 7

Скриншот кінцевих точок в swagger



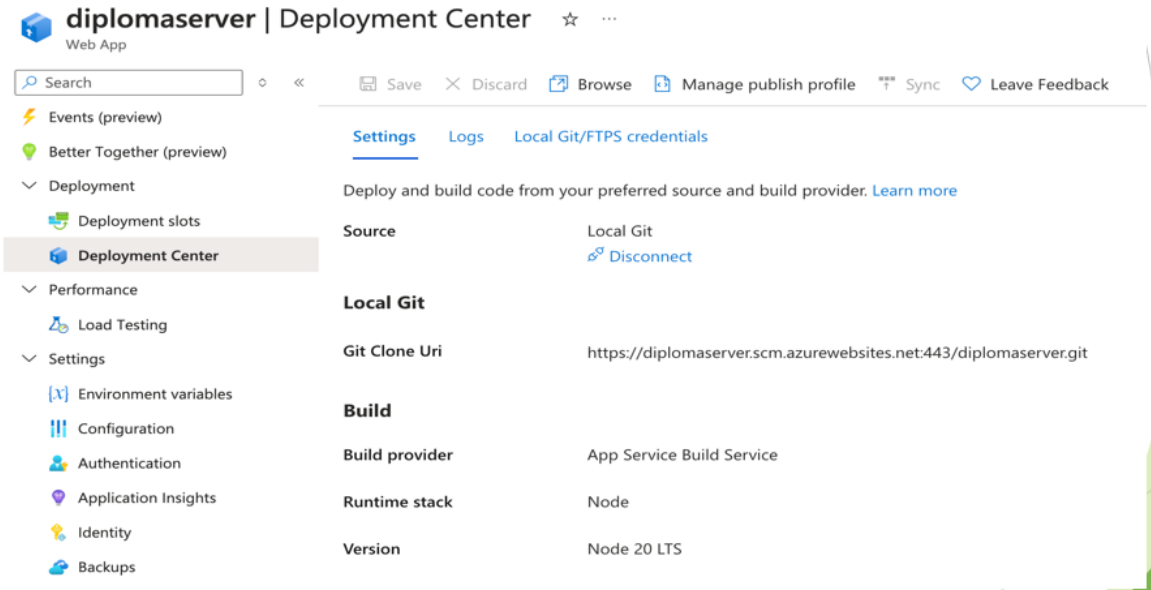
Слайд 8

Огляд ресурсної групи diploma в Azure Portal



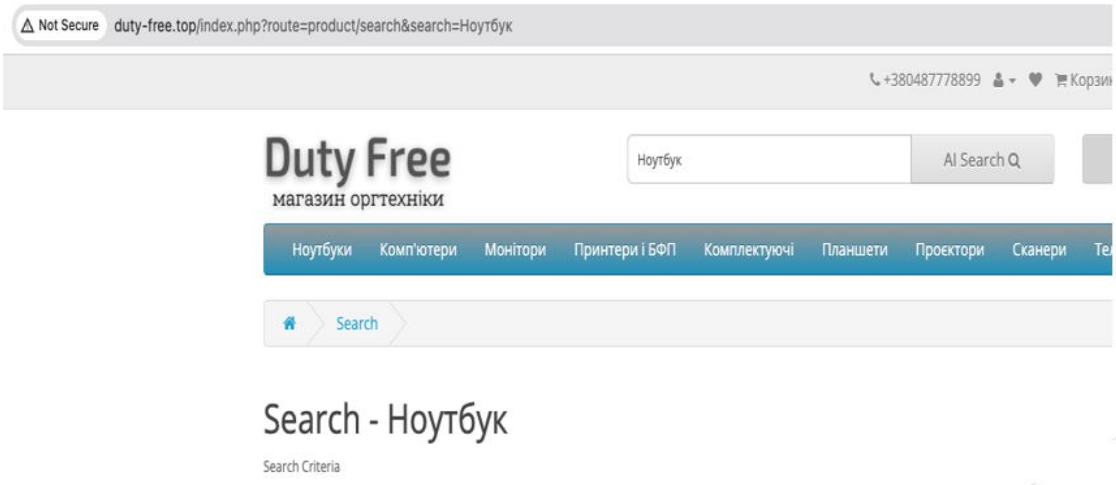
Слайд 9

Огляд Deployment Center в Azure Portal



Слайд 10

Скриншот пошукового запита



Слайд 11

Контролер search в Opencart

Code Editor

search.php

```
197         $data['products'] = array();
198
199         if (isset($this->request->get['search']) || isset($this->request->get['tag'])) {
200             $filter_data = array(
201                 'filter_name'         => $search,
202                 'filter_tag'          => $tag,
203                 'filter_description'  => $description,
204                 'filter_category_id'  => $category_id,
205                 'filter_sub_category' => $sub_category,
206                 'sort'                => $sort,
207                 'order'               => $order,
208                 'start'               => ($page - 1) * $limit,
209                 'limit'               => $limit
210             );
211
212             $product_total = $this->model_catalog_product->getTotalProducts($filter_data);
213
214             $results = $this->model_catalog_product->getProducts($filter_data);
215
```

Слайд 12

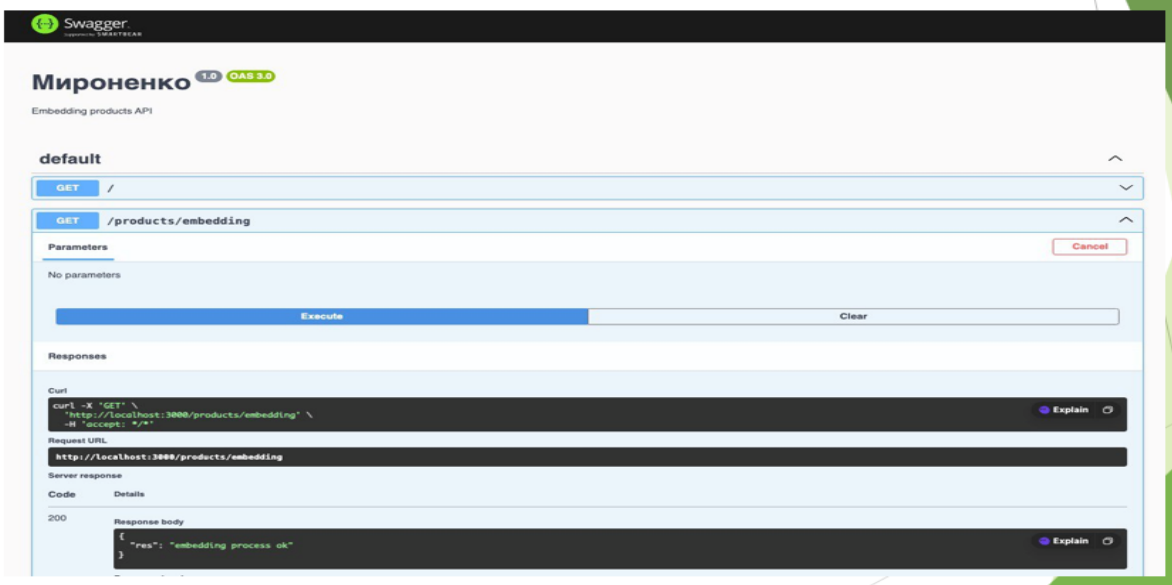
Добаваня списка id продуктів у запит в базу даних Опенкарт

product.php

```
178             $sql .= " OR LCASE(p.ean) = '" . $this->db->escape(utf8_strtolower($data['filter_name']))
179             $sql .= " OR LCASE(p.jan) = '" . $this->db->escape(utf8_strtolower($data['filter_name']))
180             $sql .= " OR LCASE(p.isbn) = '" . $this->db->escape(utf8_strtolower($data['filter_name']))
181             $sql .= " OR LCASE(p.mpn) = '" . $this->db->escape(utf8_strtolower($data['filter_name']))
182
183             // Отримуємо продукти з API
184             $products_from_api = $this->getProductFromAPI($data['filter_name']);
185
186             // Перевіряємо, чи відповідь не порожня
187             if (!empty($products_from_api)) {
188                 //Додавання умов до SQL-запиту
189                 $sql .= " OR p.product_id IN (" . implode(', ', $products_from_api) . ")";
190             }
191         }
192
```

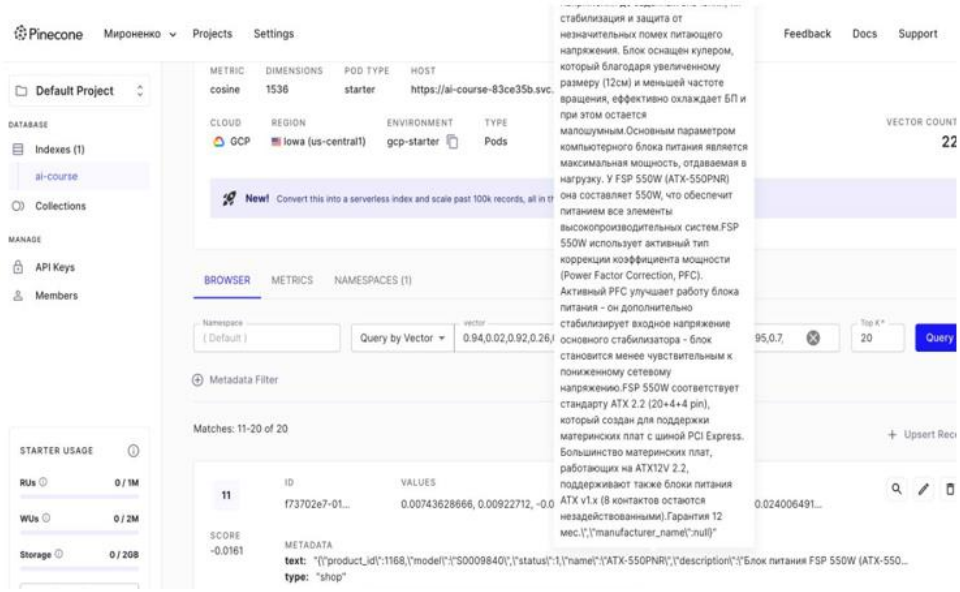
Слайд 13

Робота програми



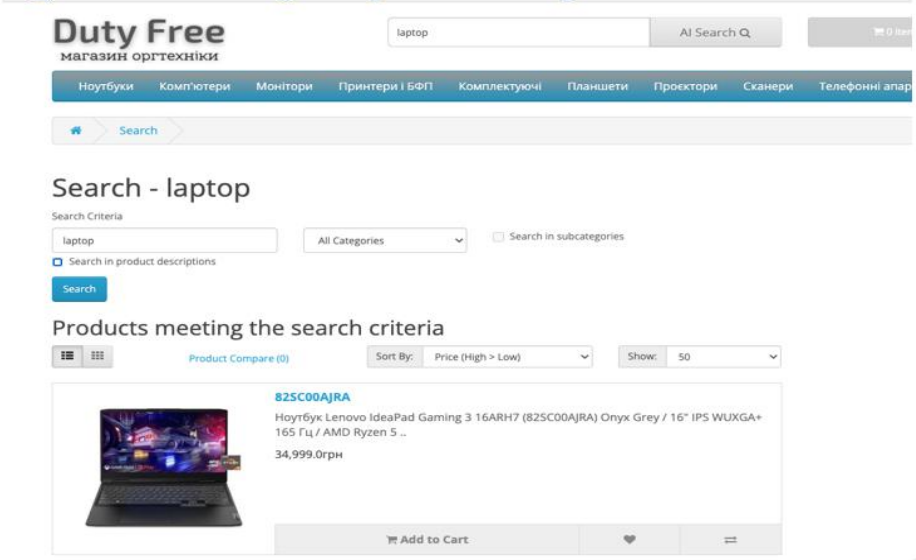
Слайд 14

Скриншот векторної бази даних



Слайд 15

Скриншот результату виконання



Слайд 16

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувачки освіти

відділення комп'ютерних систем

Мироненко Анастасії Павлівни

(прізвище, ім'я та по батькові)

Спеціальність: 121 "Інженерія програмного забезпечення"

Освітня програма: «Розробка програмного забезпечення»

Тема дипломного проекту: Розробка інтелектуального пошуку для сайту з продажу КТ

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки)

Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 64 сторінки. У пояснювальній записці виконано опис етапів розробки сайту а також технології розробки. Графічна частина складається з 19 слайдів мультимедійної презентації, які також містять пункти, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом:

Протягом всього строку дипломного проектування та переддипломної практики здобувачка освіти Мироненко А.П. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувачка освіти виконувала самостійно, з оглядом на рекомендації керівника

в) теоретична підготовка випускника (випускниці):

Здобувачка освіти Мироненко А.П. під час роботи над дипломним проектом вивчала достатню кількість літературних джерел та матеріалів за даною тематикою. Вважаю, що теоретична підготовка дипломника добра і вона готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання _____

Під час дипломного проектування здобувачка освіти Мироненко А.П. мала змогу самостійно приймати окремі рішення з реалізації програмного забезпечення та показала вміння організовано працювати над поставленим завданням, розробляти логіку та дизайн застосунку за допомогою сучасних комп'ютерних програмних засобів та мов програмування.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту _____

к.т.н. Кунуп Тетяна Василівна

Місце роботи і посада керівника дипломного проекту _____

Підпис _____

«10» 06 2023 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Мироненко Анастасії Павлівни

(прізвище, ім'я та по батькові)

Спеціальність **121 Інженерія програмного забезпечення**

Освітня програма **« Розробка програмного забезпечення»**

Керівник дипломного проекту (роботи) к.т.н Кунуп Т.В.

(прізвище, ім'я та по батькові)

Тема дипломного проекту (роботи) **Розробка інтелектуального пошуку для сайту з продажу КТ**

Обсяг розрахунково-пояснювальної записки 58 сторінок

Обсяг графічної (презентаційної) частини 12 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію робота відповідає затверджений темі та виконаний відповідно технічному завданню. Дипломний проект є актуальним з погляду останніх рекомендацій розробки програмного забезпечення для створення інтелектуального пошуку для сайту с продажу КТ

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Пояснювальна записка складається з технологічної частини, розробки структури програми та засобів програмування програми для створення інтелектуального пошуку для сайту с продажу КТ, економічної частини, розділу охорони праці та додатку. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічна частина проекту містить розрахунок затрат на виконання програми

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи)

Графічна частина складається з 12 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять структуру проекту, скріншоти роботи програми, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки висока, розробку виконано у повному обсязі

г) перелік позитивних якостей дипломного проекту (роботи)

У роботі розроблено програму інтелектуального пошуку на базі штучного інтелекту.

д) основні недоліки дипломного проекту (роботи)

1. Етапи розробки недостатньо структуровано, немає блок-схем алгоритмів та діаграм, пов'язаних з реалізацією інтелектуального пошуку.
2. У тексті ПЗ використання штучного інтелекту не достатньо обґрунтовано.
3. Наявні деякі помилки оформлення пояснювальної записки

Оцінка розрахункової частини Добре

Оцінка графічної частини Добре

Загальна оцінка Добре

Прізвище, ім'я, по батькові рецензента Царьов Роман Юрійович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, ст. викладач, зав. кафедри комп'ютерної інженерії та інформаційних систем

Підпис:

«18» 06 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016378587

Дата перевірки:
20.06.2024 17:23:08 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
20.06.2024 18:01:42 EEST

ID користувача:
100011688

Назва документа: 4РП-07 Мироненко

Кількість сторінок: 53 Кількість слів: 8664 Кількість символів: 63649 Розмір файлу: 1.95 MB ID файлу: 1016187147

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

19%
Схожість

Найбільша схожість: 6.95% з Інтернет-джерелом (https://card-file.ontu.edu.ua/server/api/core/bitstreams/3438b41f-a69..

19% Джерела з Інтернету | 717 ..... Сторінка 55

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 35

Підозріле форматування | 22 сторінки

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Мироненко Анастасії Павлівни
здобувач освіти гр. 4РП-07, та

Кунуп Тетяна Василівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи фахового молодшого бакалавра на тему:

«Розробка інтелектуального пошуку для сайту з продажу КТ». (автор роботи – Мироненко А.П., керівник роботи – Кунуп Т.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Мироненко А.П./

Керівник _____ / Кунуп Т.В./

«10» 06 2024 р.