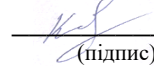


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

«До захисту допущено»

Завідувач кафедри

 В.В. Койбічук
(підпис) (Ім'я та ПРІЗВИЩЕ)

_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

(бакалавр / магістр)


зі спеціальності _____ 051 «Економіка» _____ ,
(код та назва)

_____ освітньо-професійної програми _____ «Економічна кібернетика» _____ ,
(освітньо-професійної / освітньо-наукової) (назва програми)

на тему: «Розробка автоматизованої системи менеджменту контенту економічної аналітики з використанням інструментів штучного інтелекту»

Здобувача групи ЕК.м-31 Кочережченко Романа Дмитровича
(шифр групи) (прізвище, ім'я, по батькові)


Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.



(підпис)

Кочережченко Роман
(Ім'я та ПРІЗВИЩЕ здобувача)

Керівник завідувачка кафедри, к.е.н., доцентка Койбічук В.В.
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ)



(підпис)

Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
кафедра економічної кібернетики

ЗАТВЕРДЖУЮ

Завідувач кафедри

к.е.н., доцент

_____ Віталія КОЙБІЧУК

« ___ » _____ 2024 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ
спеціальність 051 Економіка, «Економічна кібернетика», студенту 2 курсу,
групи ЕК.м-31

Кочережченко Роман Дмитрович

(прізвище, ім'я, по батькові студента)

1. Тема роботи «Розробка автоматизованої системи менеджменту контенту економічної аналітики з використанням інструментів штучного інтелекту» затверджена наказом по університету від «22» жовтня 2024 року № 1082-VI.
2. Термін подання студентом закінченої роботи «04» грудня 2024 року.
3. Мета кваліфікаційної роботи: полягає у створенні ефективного інструменту для автоматизації управління контентом та використання ШІ для генерації результатів економічної аналітики.
4. Об'єкт дослідження це створення системи управління контентом та автоматизація створення тексту з результатами економічної аналітики
5. Предмет дослідження це розробка модулів системи управління контентом та інтеграція програмного інтерфейсу провайдерів штучного інтелекту для забезпечення автоматизації результатів аналітики.
6. Кваліфікаційна робота виконується на матеріалах наукових публікацій іноземних та вітчизняних дослідників
7. Орієнтовний план кваліфікаційної роботи, терміни подання розділів керівникові та зміст завдань для виконання поставленої мети

Розділ 1 Теоретичні основи та аналіз технологій для розробки автоматизованої системи менеджменту контенту

У розділі 1 оглянути теоретичний мінімум зодо визначень CMS та ШІ, дати аргументацію актуальності роботи в контексті існуючих викликів роботи з даними та в бібліометричному аналізі. Також оглянути технічні деталі щодо технологій які будуть використовуватись під час процесу створення системи.

Розділ 2 Створення системи

У розділі 2 описати технології та процес розробки системи, включаючи розробку бази даних, команди ініціалізації необхідних бібліотек, фреймворк створення API та налаштування модулю роботи з ШІ

Розділ 3 Результати розробки системи

У розділі 3 показати результати розробки системи, приклад використання API, та приклад генерації тексту на основі реального результату аналізу даних

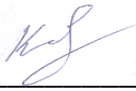
(зміст конкретних завдань до розділу, які повинен виконати студент)

8. Консультації з роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

9. Дата видачі завдання: «18»жовтня 2024 року

Керівник кваліфікаційної роботи



(підпис)

В.В. Койбічук
(ініціали, прізвище)

Завдання до виконання одержав



Р.Д.КОЧЕРЕЖЧЕНКО

АНОТАЦІЯ

кваліфікаційної роботи на тему

«Розробка автоматизованої системи менеджменту контенту економічної аналітики з використанням інструментів штучного інтелекту»

Студента Кочережченко Романа Дмитровича

У сучасному світі економічна аналітика зіштовхується із зростаючим обсягом даних, що потребують обробки та аналізу. У цій роботі запропоновано автоматизовану систему менеджменту контенту економічної аналітики, інтегровану з інструментами штучного інтелекту. Розроблена система поєднує функції традиційних CMS із можливостями генеративного штучного інтелекту (ШІ), що дозволяє створювати, організовувати та аналізувати економічний контент.

Мета роботи полягає у створенні ефективного інструменту для автоматизації управління контентом та використання ШІ для генерації результатів економічної аналітики.

Об'єктом дослідження є створення системи управління контентом та автоматизація створення тексту з результатами економічної аналітики.

Предмет дослідження: розробка модулів системи управління контентом та інтеграція програмного інтерфейсу провайдерів штучного інтелекту для забезпечення автоматизації результатів аналітики.

У межах дослідження було проведено аналіз літератури, визначено основні виклики та перспективи впровадження ШІ в управління контентом. Розроблено прототип CMS із функціоналом для генерації статей за допомогою моделей GPT, налаштованих на економічну тематику. Для цього використано передові технології, такі як Node.js, PostgreSQL та Prisma, що забезпечують масштабованість і продуктивність системи.

Практична цінність роботи полягає в можливості скорочення часу і витрат на створення економічного контенту, зокрема статей, за рахунок

автоматизації процесів. Інноваційність рішення підтверджується інтеграцією обробки природної мови (NLP) із керуванням контентом.

Отримані результати демонструють високий потенціал запропонованої системи для застосування в економічних дослідженнях, бізнес-аналітиці та автоматизації процесів управління даними.

Розроблена система вирішує ключові проблеми, пов'язані зі складністю інтеграції ШІ, якістю згенерованого контенту та взаємодією з користувачем, сприяючи ефективнішій роботі в цифровій економіці.

Результати апробації однієї з задач кваліфікаційної магістерської роботи викладені в фаховому науковому виданні Koibichuk V., Kocherezhchenko R., Hrytsenko K., Yatsenko V., Yefimenko A. (2024). Algorithms for procedural generation of game content using graphs. Infocommunication and computer technologies. No. 3. P. 77-87. <https://doi.org/10.32782/IT/2024-3-8>.

Ключові слова: штучний інтелект, система управління контентом, економічна аналітика, обробка природної мови, GPT, Headless CMS, машинне навчання, масштабованість, автоматизація.

Основний зміст кваліфікаційної роботи викладено на 36 сторінках, у тому числі список використаних джерел з 40 найменування, який розміщено на 4 сторінках. Робота містить, 14 рисунків, а також один додаток, розміщений на 24 сторінках.

Рік виконання кваліфікаційної роботи – 2024 рік.

Рік захисту роботи – 2024 рік

SUMMARY

Masters's level qualification thesis

«Development of an Automated Content Management System for Economic Analytics Using Artificial Intelligence Tools»

Kocherezhchenko Roman Dmytrovych

The master's thesis, "Development of an Automated Content Management System for Economic Analytics Using Artificial Intelligence Tools," explores a comprehensive approach to designing an efficient and scalable system for managing and generating economic analytics content. Leveraging artificial intelligence (AI), particularly natural language processing (NLP) and machine learning, the study addresses the increasing complexity and volume of digital information within the economic sector. This perspective emphasizes the significance of flexible data processing methods that can handle diverse and rapidly changing economic indicators. By adopting a strategy that integrates advanced AI models and data management solutions, the work demonstrates how a content management system (CMS) can be enhanced to manage both structured and unstructured economic data more effectively.

The theoretical foundation of the thesis underscores the necessity of automation in content management. The design and implementation of automated pipelines aim at streamlining the collection, preprocessing, and analysis of economic data, thereby reducing the manual burden traditionally associated with these tasks. The thesis proposes a versatile framework that employs robust data pre-processing and feature selection approaches. Such approaches ensure that the system focuses on the most critical factors influencing economic analytics. These factors may range from macroeconomic indicators like GDP or inflation rates, to market volatility metrics, to sector-specific trends that require careful interpretation. The resulting structured content units offer a foundation for generating insights that can inform

decision-makers, allowing them to understand the underlying forces driving economic developments and patterns.

A bibliometric analysis, conducted with tools like R Studio and Bibliometrix, pinpointed emerging technologies—AI, big data, and generative AI—within the landscape of economic research. This contextual assessment highlights how the integration of these technologies responds to current industry demands. The study also compares traditional content management systems (CMS) with the Headless CMS architecture.

The Headless CMS model, with its modular design, proves particularly well-suited for integrating advanced AI models. By separating the content repository from the presentation layer, the Headless CMS framework allows the seamless addition of AI-driven modules that can process data, generate insights, and produce tailored content. By combining the flexible Headless CMS approach with GPT-based AI models (e.g., GPT-4o and o1-preview), the system can automatically generate economic analytics content that covers standard financial indicators and adapts to a variety of data inputs. These models highlight patterns and trends in a structured manner, enhancing the clarity and usefulness of the generated output.

The technical implementation forms the backbone of this solution. Built on a Headless CMS framework and employing technologies like Node.js, TypeScript, Fastify, and Prisma, the system efficiently handles API development and content queries through PostgreSQL. The careful selection of these technologies creates a coherent and integrated environment, ensuring that each component in the pipeline supports the others. The CRUD (Create, Read, Update, Delete) architecture enables seamless management of diverse content entities—articles, categories, and user profiles—while ensuring that complex, data-driven information can be stored, retrieved, and processed without unnecessary complexity. These structured outputs may include economic forecasts derived from time series analyses, regression results that elucidate the relationships between variables, and other data-driven insights. The AI module harnesses OpenAI's GPT models for contextually accurate content

generation. System prompts guide the GPT models to select relevant features dynamically, adjusting parameters such as token limits and temperature settings to maintain a balance between detail and clarity. The careful design of these prompts and parameters ensures that the system produces content that remains aligned with economic contexts and adheres to quality standards.

Real-world test scenarios validated the system's ability to produce structured, relevant content. By applying the pipeline to various datasets and queries, it was possible to confirm that the outputs not only mirrored existing analytical methods but also introduced a degree of automation and systematic organization. This approach supports the identification of meaningful patterns in complex economic datasets, ensuring that end-users—businesses, policymakers, and researchers—receive data-driven insights that measure economic performance while revealing subtle interconnections among economic indicators.

The overall process reduces the complexity faced by analysts and decision-makers, granting them a tool that accelerates the conversion of raw economic data into actionable intelligence.

The system's adaptability is a key outcome highlighted by the research. In practice, the CMS demonstrated ease of use and flexible content management through well-defined API endpoints. The ability to integrate various data sources into the system and update content dynamically ensures that the generated analytics remain current and reflective of evolving economic conditions. The AI module generated structured insights that ranged from simple string outputs derived from raw data analysis to more specialized content that addresses unique user queries. The relevance and accuracy of these outputs, validated through iterative testing, underscore the practical applications of such an integrated system. The design encourages end-users to rely on a consistent, continuously improving source of economic information that is both timely and actionable.

In conclusion, this thesis confirms the feasibility of integrating AI-driven analytics with CMS functions to revolutionize the management and generation of economic content. By employing robust data processing methodologies and advanced AI capabilities, the system reduces time and costs associated with producing high-quality financial materials. At the same time, it maintains depth and contextual relevance by focusing on the critical variables that shape economic realities. This integrated approach fosters a more efficient and insightful economic information environment, encouraging proactive measures in the face of complex market conditions. The project's emphasis on structured outputs and dynamic data handling ensures that the system can evolve alongside the ever-shifting economic landscape.

Throughout the project, a critical challenge was ensuring that the generated content met the high standards expected in economic analysis. The quality of generated insights depended not only on the raw capabilities of GPT models but also on the precision with which system prompts were designed and tuned. This required a deep understanding of both the technical underpinnings of GPT models and the nuances of economic data interpretation.

Careful engineering of prompts and iterative testing allowed for consistent improvements in content quality. Each refinement step tightened the alignment between data inputs and the generated outputs, ensuring that the system's responses were not only contextually apt but also analytically sound.

Another significant hurdle was the integration of various technologies into a cohesive system. The decision to use Node.js, TypeScript, and Prisma proved advantageous, as these technologies offer a stable, reliable, and developer-friendly environment for building scalable APIs. PostgreSQL's advanced querying capabilities complemented this setup, enabling the handling of complex data relationships within the CMS. The use of Swagger for API documentation further streamlined both development and testing, clarifying the system's endpoints and

operations. This attention to technical detail ensured that the system, while sophisticated in its functionality, remained accessible and maintainable over time.

Future developments might include further refinement of existing models or the inclusion of additional AI techniques to deepen analysis without introducing new categories of information.

The groundwork laid by this thesis provides a stable platform that can be augmented to improve anomaly detection or enhance sentiment analysis within economic texts.

Likewise, automated responses to regulatory changes could be incorporated by refining prompts and data processing methods already in place. By weaving together economic analytics and sophisticated modeling methodologies, the system positions itself at the frontier of digital transformation, offering a consolidated resource for advancing economic research and practice. Even as it stands, the modular nature of the system allows incremental improvements without needing to overhaul the core architecture.

The system's design and implementation have practical applications across various sectors that rely on economic insights. Policymakers, for instance, can utilize the platform to generate detailed reports on market conditions, providing them with evidence-based materials that inform policy decisions.

Researchers benefit from streamlined workflows that free them from repetitive data preprocessing, allowing them to focus on higher-level analysis and interpretation. Businesses find value in the system's capacity to produce market analyses and forecast reports that are drawn from a consistent and reliable source. The ability to scale and integrate additional data streams further ensures that the system can adapt to specific organizational needs, handling higher loads or more diverse data inputs as circumstances evolve.

The integration of GPT models, such as GPT-4o and o1-preview, underscores the system's potential for producing nuanced, context-aware content. By fine-tuning

parameters such as token limits and temperature, the project optimized these models to generate financial and economic texts that remain both relevant and precise. Such capabilities bridge the gap between raw data and actionable insights, assisting users who require data-driven content that maintains analytical rigor. Each generated output stands as a piece of a larger puzzle, contributing to a mosaic of information that users can rely on for decision-making, forecasting, or comparative analysis.

The master's thesis demonstrates the transformative potential of AI-driven systems in economic analytics. By combining advanced technological solutions with innovative methodological approaches, the project created a cohesive system that meets the growing demands of data-rich environments.

The experience gained during the development process highlights the importance of interdisciplinary knowledge, combining expertise in AI technologies, software engineering principles, and economic analysis methodologies. This multidisciplinary foundation ensured that the resulting system was not only functional but also aligned with the practical needs of economic practitioners.

Ultimately, the system serves as a stepping stone toward more sophisticated applications of AI in economic contexts. It exemplifies how the integration of modern content management strategies and AI-driven analysis can streamline workflows, enhance analytical depth, and improve the overall quality of output. As technology continues to evolve, systems like this one will play a crucial role in shaping the future of economic research and practice. By providing a platform that is both flexible and rigorous, the project opens pathways for further exploration, refinement, and innovation in the field of economic analytics.

Keywords: artificial intelligence, content management system, economic analytics, natural language processing, GPT, Headless CMS, machine learning, scalability, automation.

ЗМІСТ

ВСТУП	13
РОЗДІЛ 1 Теоретичні основи та аналіз технологій для розробки автоматизованої системи менеджменту контенту	16
1.1 Актуальність створення автоматизованої системи менеджменту контенту	16
1.2 Бібліометричний аналіз технологій для розробки автоматизованої системи менеджменту контенту з використанням інструментів штучного інтелекту	17
1.3 Технічні деталі щодо реалізації системи автоматичної генерації контенту	22
РОЗДІЛ 2. СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ ТА ВЗАЄМОДІЇ З ШТУЧНИМ ІНТЕЛЕКТОМ	29
2.1 Розробка системи управління контентом економічної аналітики.....	29
2.2 Розробка модулю взаємодії зі штучним інтелектом	34
РОЗДІЛ 3. РЕЗУЛЬТАТИ РОЗРОБКИ СИСТЕМИ.....	36
3.1 Результати розробки системи менеджменту контенту	36
3.2 Результати розробки модулю штучного інтелекту	37
ВИСНОВКИ.....	43
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	45
ДОДАТКИ	49
ДОДАТОК А.....	49

ВСТУП

На даний момент методи та підходи роботи з інформацією та даними, якими користуються спеціалісти різних сфер та напрямків змінюються. Процеси змін, пов'язані як і загальним розвитком рівня володіння цифровими технологіями, так і стрімким розвитком інструментів штучного інтелекту, далі ШІ, для роботи з різними типами інформації. Враховуючи зміни які відбуваються доречно розглянути можливості використання сервісів для оптимізації часових та грошових затрат при роботі з інформацією. В рамках роботи була розроблена система для управління контентом, де основний контент статті створюється за допомогою ШІ. Виходячи з цього можна зробити висновок що практична реалізація та цінність цієї роботи полягає в двох частинах: створенні системи управління контентом результатів економічної аналітики, формуванні процесу взаємодії з ШІ.

До основних проблем процесу створення подібних систем можна віднести складність розробки як модулю для взаємодії з контентом, так і модуля для взаємодії з ШІ для отримання якісного результату. Ці проблеми та процес їх вирішення є основою цієї роботи.

Актуальність теми полягає в необхідності автоматизації роботи з контентом, а саме в створенні первинного матеріалу економічної аналітики, що дозволить суттєво заощадити час та гроші при створенні статей та інших текстів. Ринок ШІ постійно перебуває в центрі уваги, в тому числі науковців та інженерів, які досліджують різні аспекти цієї сфери, та впроваджують нові рішення (McKinsey, 2023).

Мета роботи полягає у створенні ефективного інструменту для автоматизації управління контентом та використання ШІ для генерації результатів економічної аналітики.

Об'єктом дослідження є створення системи управління контентом та автоматизація створення тексту з результатами економічної аналітики.

Предмет дослідження: розробка модулів системи управління контентом та інтеграція програмного інтерфейсу провайдерів штучного інтелекту для забезпечення автоматизації результатів аналітики.

Наукова новизна одержаних результатів даної кваліфікаційної роботи полягає в розробленні системи, яка поєднує функції системи для управління контентом результатів економічної аналітики та взаємодії з ШІ, що дозволяє мати інший робочий потік при створенні тексту. Крім того, наукова новизна полягає в об'єднанні інструментів на основі штучного інтелекту, таких як обробка природної мови (Natural Language Processing, NLP) і машинне навчання, з керуванням контентом, що економить час і ресурси.

Традиційні платформи CMS обмежуються організацією статичного вмісту, тоді як запропонована система автоматизує генерацію тексту в CMS, запроваджуючи безперебійний, ефективний і масштабований робочий процес. Ця система вирішує ключові проблеми, включаючи складність інтеграції та якість контенту, створеного штучним інтелектом, за допомогою розширених інтерфейсів API та повторюваних відгуків користувачів. Контраргументи щодо відсутності нюансів у змісті штучного інтелекту та занепокоєння щодо переміщення роботи пом'якшуються завдяки забезпеченню контролю з боку людини та акцентуванню на трансформації робочих місць за допомогою розширення штучного інтелекту.

Для досягнення поставленої мети в ході дослідження потрібно вирішити такі завдання, як:

- розглянути теоретичний мінімум щодо створення та взаємодії CMS та ШІ;
- дослідити технології та інструменти для розробки;
- визначити технології та інструменти, що будуть використані для створення системи;
- створити план розробки CMS системи;
- розробити базу даних, для збереження даних CMS;
- розробити модуль управління контентом;

- розробити модуль взаємодії з ШІ;
- здійснити перевірку якості генерації контенту;
- провести аналіз результатів розробки та створення системи.

РОЗДІЛ 1 Теоретичні основи та аналіз технологій для розробки автоматизованої системи менеджменту контенту

1.1 Актуальність створення автоматизованої системи менеджменту контенту

Сьогодні економічна аналітика стикається з безпрецедентним обсягом даних, які потрібно обробляти швидко та ефективно. За даними Міжнародної корпорації даних (IDC), до 2025 року обсяг даних у світі досягне 175 зеттабайтів (International Data Corporation, 2018). Цей рівень викликів вимагає переходу до автоматизації, адже традиційні методи обробки просто не справляються з такими навантаженнями (Vi, 2023; Kanbach et al., 2024; Ooi et al., 2023). Інструменти на базі ШІ, як-от алгоритми машинного навчання та нейронні мережі, є ключовими у вирішенні цієї проблеми. Вони здатні знаходити приховані закономірності у великих наборах даних, які часто залишаються непомітними навіть для досвідчених аналітиків (Lehr, Caliskan, Liyanage, & Banaji, 2024).

Ще одна помітна перевага автоматизованих систем – це економія коштів. McKinsey & Company оцінюють, що впровадження ШІ-рішень може знизити операційні витрати компаній на 10–20% (McKinsey, 2023). При цьому важливо не забувати про мінімізацію ризиків людських помилок (Giraud et al., 2023). У секторах, де кожна помилка може призводити до значних втрат, впровадження ШІ стає перевагою (Wu, Yan, Luo, Wang, & Tang, 2024).

Найпоширеніший виклик – це безпека даних (Wach et al., 2023). Системи ШІ часто працюють з великими обсягами конфіденційної інформації, що підвищує ризик витоків чи кібератак. Однак сучасні рішення включають надійні інструменти захисту, зокрема шифрування даних і багатофакторну автентифікацію (OpenAI, 2023). Ба більше, системи ШІ часто здатні ідентифікувати кіберзагрози ще до їхньої реалізації, що додає рівень захисту, недоступний для звичайних методів (SentinelOne, 2024).

Таким чином, автоматизовані системи на базі ШІ – це не лише відповідь на сучасні виклики, а й інструмент, який формує майбутнє економічної аналітики. Впровадження таких рішень дозволяє організаціям не лише виживати, але й процвітати в умовах цифрової економіки (Boukar et al., 2024; Harrer, 2023; Nyenno et al., 2023; Yu & Gong, 2024).

1.2 Бібліометричний аналіз технологій для розробки автоматизованої системи менеджменту контенту з використанням інструментів штучного інтелекту

За пошуковим запитом «розробка автоматизованої системи менеджменту контенту з використанням інструментів штучного інтелекту» в наукометричній базі даних Скопус було знайдено масив публікацій в обсязі 306 документів, проіндексованих за період з 2019 року по 2024 рік. Бібліометричний аналіз проведено з використанням R Studio з використанням пакетів та бібліотек Shiny та Bibliometrix, та їх відповідних функцій, розроблених науковцями (Aria & Cuccurullo, 2017). Застосування зазначених бібліотек відбувається в окремому середовищі з однойменною назвою Bibliometric. Початок роботи в середовищі Bibliometrix починається з імпорту текстового масиву даних, згенерованого за відповідним запитом пошуковою системою БД Скопус, в форматі BibТех. Головна інформація щодо імпортованої вибірки відображена на рис. 1.1.

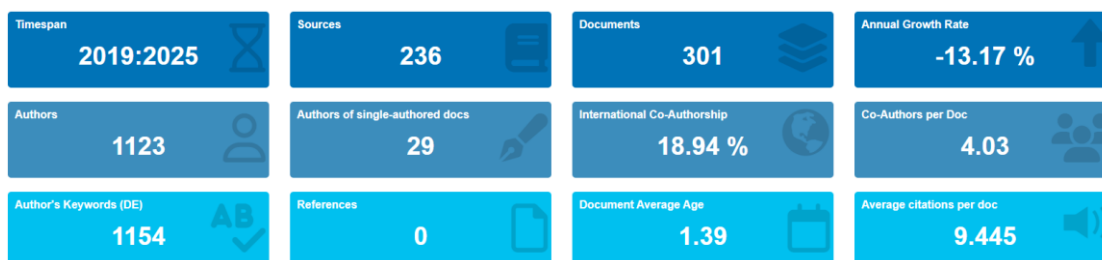


Рисунок 1.1 – Головна інформація щодо аналізованого масиву публікацій за темою дослідження.

Перш за все, потрібно зазначити що, негативний річний темп зростання -13,17% говорить про уповільнення створення подібних дослідницьких документів. Це вказує на можливість впровадити інновації.

З 4,03 співавторами на документ і 18,94% міжнародним співавторством ця сфера показує високий показник співпраці. Така тенденція відображає складність і міждисциплінарний характер штучного інтелекту та економічної аналітики, де взаємодія між доменами різних предметних областей має вирішальне значення.

На рис 1.2 відображено графік, що містить три полюси, які відображають співпрацю авторів в контексті ключових слів, анотацій та країн (права частина відображає ключові слова, середня – анотації авторів щодо теми дослідження, ліва – країни). Загальна кількість складає 15 елементів за кожним критерієм.

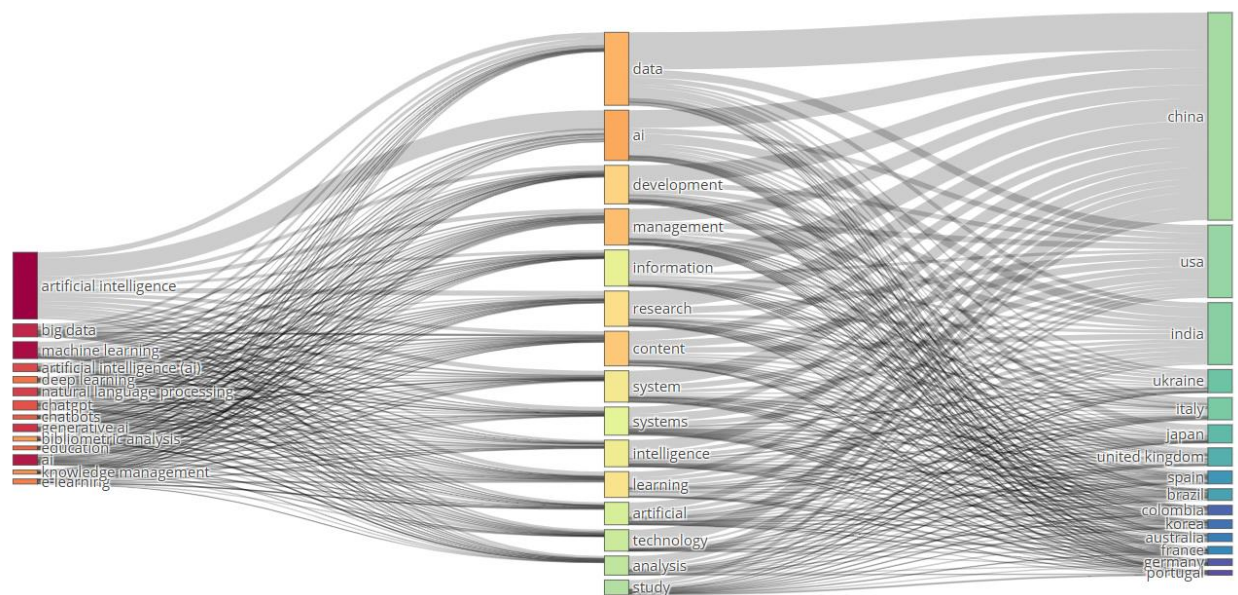


Рисунок 1.2 – Триполюсний графік

Висока частота вживання ключових термінів, таких як «штучний інтелект», «дані» та «контент», свідчить про зростаючу важливість впровадження ШІ для управління контентом у сфері економічної аналітики. Це підтверджує потенціал розробки інноваційних систем, які інтегрують

алгоритми штучного інтелекту для ефективного управління та аналізу великих масивів даних.

Фокус на таких поняттях, як «управління» та «системи», корелює з метою автоматизації процесів керування контентом. Використання штучного інтелекту в цьому контексті дозволяє оптимізувати процеси отримання, структурування та інтерпретації економічної інформації.

Технології, такі як машинне навчання, глибоке навчання та обробка природної мови (NLP), займають центральне місце в розробці інтелектуальних систем. Їхнє впровадження сприяє ефективній роботі з як структурованими, так і неструктурованими даними, що є ключовим для вирішення завдань в економічному контексті. На рис 1.2– Деревоподібна карта частоти використання ключових слів у масиві публікацій, зображена деревоподібна карта, що відображає ключові слова та частоту їх використання в масиві публікацій.



Рисунок 1.3 – Деревоподібна карта частоти використання ключових слів у масиві публікацій

Штучний інтелект (AI) виступає центральною темою досліджень, охоплюючи 16% загальної уваги, що свідчить про його критичну роль як рушійної сили сучасних інновацій у різних дисциплінах. Управління інформацією, що складає 6% досліджень, слугує важливим доповненням до AI, особливо у сфері організації та обробки даних. Увага до таких ключових

понять, як "людина" (4%), "системи навчання" (4%) і "прийняття рішень" (3%), вказує на акцент досліджень на розробці систем ШІ, орієнтованих на користувача, та технологій підтримки прийняття рішень. Теми, пов'язані з електронним навчанням, глибоким навчанням, машинним навчанням і системами обробки природної мови, свідчать про різноманітні застосування AI у галузях освіти, аналізу даних та обробки текстової інформації. Водночас, нові напрямки, як-от периферійні обчислення, генеративний штучний інтелект, системи реального часу та розробка навчальних програм, хоча і займають меншу частку, залишаються перспективними для подальшого розвитку інтелектуальних систем. Спеціальні теми, включаючи обробку даних, цифрове зберігання та управління знаннями, відображають спеціалізовані, але важливі аспекти структурування та обробки інформації.

Рис. 1.4 відображає тематичну карту в контексті дослідження кваліфікаційної роботи.

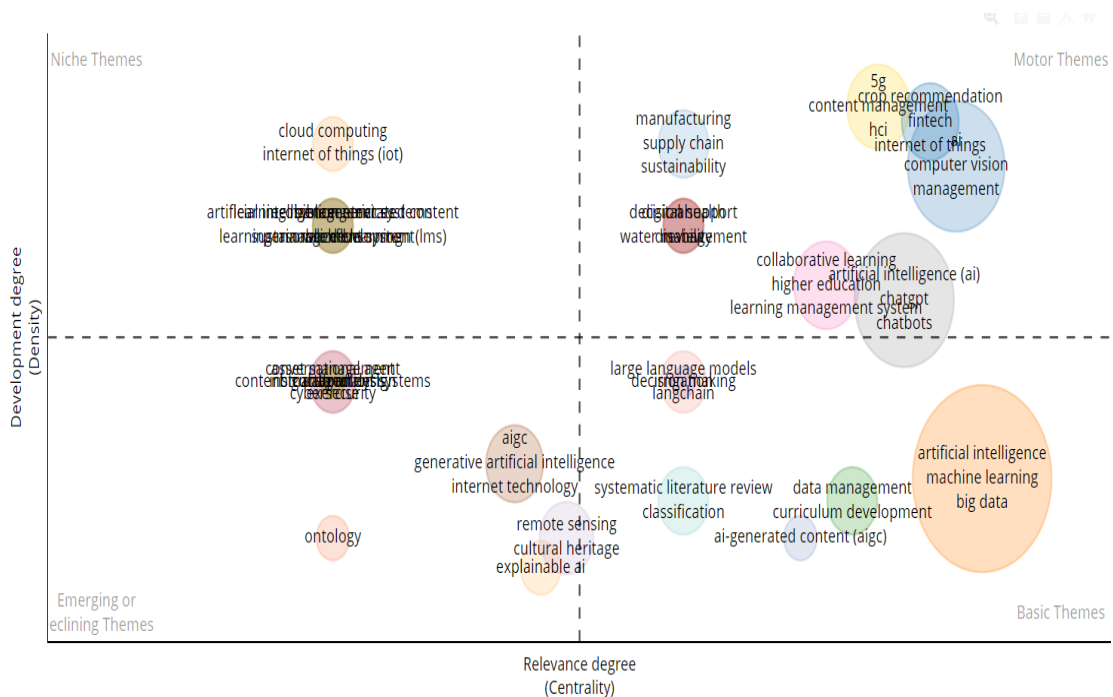


Рисунок 1.4 – Тематична карта

Теми-двигуни (Motor Themes), правий верхній квадрант, характеризує текі теми, як управління контентом, фінтех, комп'ютерне бачення та технології 5G, що активно сприяють розвитку цієї галузі та мають значний вплив на її прогрес. Вони високорозвинені та актуальні, що вказує на їхню

роль критично важливих засобів для інноваційних систем, таких як автоматизоване керування вмістом.

Базові теми (Basic Themes), правий нижній квадрант містить сфери, такі як ШІ, машинне навчання та великі дані, які домінують у цьому квадранті, підкреслюючи їх важливу роль як основу для прогресу в галузі. А також ці теми забезпечують опис технологічної інфраструктури, необхідної для створення передових систем керування контентом.

Нішеві теми (Niche Themes), лівий верхній квадрант: такі теми, як хмарні обчислення та Інтернет речей (IoT), дуже розвинені, але мають більш спеціалізоване значення. Їх присутність свідчить про те, що вони мають вирішальне значення в конкретних програмах, але ще не центральні для ширших ініціатив управління контентом.

До теми, що виникають або занепадають (Emerging or declining themes), лівий нижній квадрант увійшли такі сфери, як генеративний ШІ, онтологія та пояснюваний ШІ, все ще перебувають на ранніх стадіях розвитку або втрачають свою значимість. Однак наявність генеративного штучного інтелекту в цьому квадранті теж свідчить про зростаючий інтерес і потенціал для інтеграції в більш усталені теми в майбутньому.

Отже, діаграма (рис. 1.4) показує важливі аспекти, які слід врахувати при розробці автоматизованої системи управління контентом для економічної аналітики. Штучний інтелект, машинне навчання та великі дані, розташовані в квадранті «Основні теми», є фундаментальними технологіями, які забезпечують основу для створення системи з потужними можливостями обробки даних і прогнозного аналізу. Управління контентом, як зріла й актуальна тема, наголошує на необхідності впровадження сучасних підходів і технологій, що відповідають вимогам цієї галузі.

Генеративний штучний інтелект, розташований у квадранті «Теми, що розвиваються», пропонує перспективні можливості, наприклад, автоматичне створення висновків чи рекомендацій із економічних даних, що значно розширює функціонал майбутньої системи. У той же час такі напрямки, як

фінансові технології та комп'ютерне бачення, які знаходяться у квадранті «Моторні теми», демонструють важливість міждисциплінарного підходу. Інтеграція цих рішень може додати нових можливостей, як-от використання візуальної аналітики чи адаптація до специфіки фінансових даних.

Нішеві напрями, такі як хмарні обчислення та Інтернет речей (IoT), хоч і мають вузьку спрямованість, пропонують реальні перспективи для масштабування системи та впровадження збору даних у реальному часі, що особливо актуально для великих або розподілених систем. Прозорість, підкреслена в темах «Emerging», таких як пояснювальний штучний інтелект, є ключовим елементом для побудови довіри користувачів до рішень на основі AI (Sundarambal et al., 2023).

Таким чином, комбінація базових технологій, як штучний інтелект і машинне навчання, розширення функціональних можливостей через використання новітніх розробок, і пошук інноваційних рішень у нішевих напрямках створює цілісну концепцію для розробки автоматизованої системи управління контентом (He, 2022; Hu et al., 2021; Korinek & Stiglitz, 2021; Qin et al., 2024; Sundarambal et al., 2023).

1.3 Технічні деталі щодо реалізації системи автоматичної генерації контенту

Перш за все потрібно визначитись, який функціонал має бути реалізований через CMS. Основне визначення лежить в аббревіатурі CMS – Content Management System. Це означає, що основна частина функціоналу системи буде зосереджена на реалізації методів для керування контентом у системі (Кузнецова, 2023).

В рамках проекту була розроблена Headless CMS. На відміну від традиційного виду CMS, така система не має клієнтського UX інтерфейсу; натомість взаємодія з системою зазвичай забезпечується через HTTP-запити.

Така архітектура дозволяє зробити CMS дуже гнучкою, адже це дозволяє сприймати її як пакет більших систем (Odom, 2022).

Додатково, використання Headless CMS дає можливість розробникам створювати більш масштабовані та адаптивні рішення. Завдяки відсутності жорсткої прив'язки до конкретного фронтенду, контент можна легко інтегрувати в різні платформи та пристрої через API (AWS, 2023).

Крім того, така система сприяє більш ефективному розподілу ресурсів між командами розробників. Фронтенд та бекенд можуть розроблятися незалежно один від одного, що прискорює процес розробки та впровадження нових функцій. Це також підвищує гнучкість у виборі технологій для кожної з частин системи.

Використання Headless CMS також поліпшує безпеку, оскільки зменшується кількість потенційних вразливостей, пов'язаних з користувацьким інтерфейсом. Це особливо важливо для проєктів, де безпека даних є критичним аспектом.

Отже маючи загальні уявлення про систему яка буде розроблятися можна перейти до теоретичних основ взаємодії з ШІ, який буде використовуватись в системі. Перш за все, потрібно дати контекст визначення ШІ про який йдеться саме у цій роботі. Для виконання поставлених задач та цілей не було розроблено своєї моделі штучного інтелекту, натомість було використано готові моделі штучного інтелекту, доступ до яких здійснювався через надане API. Для цієї роботи було обрано моделі OpenAI, а саме gpt-4o, та o1-preview, так як в результаті тестування було визначено, що вони надають оптимальний варіант стабільності та якості результатів. Для того аби розуміти як воно працює хоча б на верхньому рівні потрібно визначити тип моделей, а саме GPT. Generative Pre-trained Transformers (GPT) — це клас мовних моделей, які використовують методи глибокого навчання для створення тексту, схожого на людину. Вони побудовані на архітектурі трансформатора, яка зробила революцію в обробці природної мови (NLP), дозволяючи моделям розуміти контекст і зв'язки в послідовних даних ефективніше, ніж попередні

архітектури, такі як рекурентні нейронні мережі (RNN) і мережі довготривалої короткочасної пам'яті (LSTM). Зараз це не дає корисної інформації, тому потрібно розглянути в контексті архітектури, а не загального визначення (Vaswani, et al., 2017).

Розуміння того, як працюють моделі GPT, передбачає вивчення основних принципів обробки природної мови та глибокого навчання. GPT, що означає Generative Pre-trained Transformer, є типом мовної моделі, розробленої OpenAI, яка може генерувати текст, схожий на людину, на основі вхідних даних. Ці моделі побудовані на основі трансформаторної архітектури, нейронної мережі, яка значно просунула сферу обробки природної мови (Vaswani, et al., 2017; Radford, Narasimhan, Salimans, & Sutskever, 2018).

В основі моделей GPT лежить здатність архітектури такої моделі, як трансформатор, обробляти дані непослідовним способом, що дозволяє моделі розуміти контекст і зв'язки між словами в реченні ефективніше, ніж попередні моделі, такі як рекурентні нейронні мережі. Це досягається за допомогою механізму, відомого як «самоувага», який дозволяє моделі зважувати важливість різних слів під час побудови відповіді (OpenAI, 2023). Одночасно враховуючи весь вхідний контекст, модель може фіксувати довготривалі залежності та нюанси в тексті (Sundaresan & Zhang, 2022).

Процес навчання моделей GPT складається з двох основних етапів: попереднього навчання та тонкого налаштування. На етапі попереднього навчання модель навчається, передбачаючи наступне слово в реченні в величезній кількості текстових даних, зібраних з Інтернету. Це неконтрольоване навчання дозволяє моделі осягати граматику, факти про світ і різні мовні шаблони без явного людського ярлика. Широкий доступ до різноманітного вмісту дозволяє моделі генерувати зв'язний і контекстуально релевантний текст на широкий спектр тем.

Тонка настройка – це наступний етап, на якому модель налаштовується для конкретних завдань або доменів за допомогою меншого, більш цілеспрямованого набору даних. Цей процес може включати контрольоване

навчання, коли модель навчається на прикладах, які включають як вхідні, так і бажані вихідні дані, покращуючи її здатність виконувати такі завдання, як переклад, узагальнення або відповіді на запитання. У деяких випадках для узгодження результатів моделі з уподобаннями людини використовується навчання з підкріпленням за допомогою зворотного зв'язку людини, покращуючи релевантність і відповідність її відповідей (Christiano, et al., 2017).

Під час генерації тексту модель GPT приймає підказку введення та обробляє його через кілька рівнів нейронних мереж. Вхідний текст спочатку розбивається на лексеми, які є основними одиницями, такими як слова або підслова. Ці токени перетворюються на числові представлення, які називаються вбудовуваннями, фіксуючи їхні значення у формі, з якою може працювати модель. Вбудовування проходять через рівні декодера трансформатора, де механізми самоконтролю допомагають моделі зрозуміти зв'язки між токенами в контексті всього вхідного масиву даних.

Потім модель передбачає наступний маркер у послідовності, враховуючи ймовірності можливих маркерів і вибираючи той із найвищою ймовірністю. Цей процес повторюється ітеративно, причому кожен щойно згенерований маркер стає частиною вхідних даних для наступного кроку передбачення. Результатом є послідовний і відповідний контексту фрагмент тексту, який продовжується з початкової підказки .

Моделі III, які будуть використовуватися для розробки є ітераціями цієї моделі зі збільшенням складності та можливостей (Gallardo Paredes et al., 2023). Наприклад, gpt-4o, використовувався для роботи таких програм, як ChatGPT, які можуть вести детальні та контекстні розмови з користувачами; o1-preview являє собою подальший прогрес, пропонуючи покращені можливості міркування, краще оброблення нюансованих інструкцій (Brown, et al., 2020).

Маючи уявлення про те як працюють III які будуть використовуватися в цій роботі, можна почати огляд обраних технологій.

Процес розробки програмного забезпечення потребує вибору технологій, які відповідають поточним вимогам і забезпечують можливість масштабування. Для проєкту обрано Node.js із TypeScript, Fastify, PostgreSQL та Prisma. Цей набір технологій забезпечує інтеграцію компонентів, швидкість виконання та гнучкість у розробці.

Для серверної частини проєкту було розглянуто кілька платформ, зокрема Node.js та Python (Django/FastAPI), із врахуванням критеріїв продуктивності, гнучкості, масштабованості та сучасних мовних можливостей. Python демонструє високу ефективність у задачах обробки даних і штучного інтелекту, а його фреймворки Django та FastAPI дозволяють швидко розпочати розробку завдяки структурованості та сучасній архітектурі відповідно, проте має обмеження щодо продуктивності в серверних задачах через Global Interpreter Lock (GIL) і менш ефективну обробку великої кількості одночасних запитів порівняно з Node.js. Node.js із TypeScript, у свою чергу, пропонує подієву модель, що забезпечує високу продуктивність і масштабованість, а також нативну підтримку асинхронності. TypeScript додає статичну типізацію, яка знижує ризик помилок, покращує підтримку коду, дозволяє структурувати великі проєкти та інтегрує сучасні функції JavaScript. Крім того, Node.js має багату екосистему бібліотек і модулів, що прискорює розробку та спрощує інтеграцію з іншими інструментами, а низький поріг входження завдяки базуванню на JavaScript сприяє залученню команди (Kearse et al., 2012; Ofoeda et al., 2023). Вибір Node.js із TypeScript забезпечує баланс між продуктивністю, простотою та масштабованістю, що робить його оптимальним рішенням для серверної частини проєкту, порівняно з Python (GeeksforGeeks, 2024).

Для реалізації серверної частини проєкту було проведено аналіз популярних фреймворків, таких як Express та Fastify, із врахуванням критеріїв продуктивності, інтеграції з TypeScript, масштабованості та модульності. Express, як широко використовуваний фреймворк із великою спільнотою, не забезпечує нативної підтримки TypeScript і демонструє нижчу продуктивність

у порівнянні з іншими фреймворками. Fastify вирізняється високою продуктивністю завдяки оптимізованій архітектурі, нативною підтримкою TypeScript та плагіновою системою, яка спрощує розширення функціоналу й оптимізацію окремих компонентів сервісу. У порівнянні з іншими фреймворками, Fastify демонструє кращу швидкодію, інтегрується з TypeScript без додаткових витрат на конфігурацію, має сучасний підхід до обробки запитів і активно розвивається. Ці переваги зробили Fastify оптимальним вибором для забезпечення стабільності, продуктивності та гнучкості серверної частини проєкту (The New Stack, 2023).

Для зберігання та обробки даних було розглянуто кілька варіантів баз даних, зокрема PostgreSQL, MySQL та MongoDB, із врахуванням критеріїв транзакційності, масштабованості, підтримки складних запитів і можливості розширення. MongoDB, як документоорієнтована база даних, демонструє гнучкість у роботі з напівструктурованими даними, але не забезпечує рівня транзакційної цілісності, який відповідає ACID-принципам, і менш ефективна у складних аналітичних запитах. MySQL є популярною реляційною базою даних із простою конфігурацією, але має обмежені можливості у роботі зі складними SQL-запитами порівняно з PostgreSQL. PostgreSQL було обрано завдяки її потужному SQL-движку, який підтримує Common Table Expressions, Window Functions і складні транзакції, забезпечуючи високу консистентність даних. Її розширюваність дозволяє створювати власні типи даних і адаптувати систему до специфічних потреб проєкту, а стабільна робота з великими обсягами інформації робить її оптимальним вибором для складних і масштабованих застосунків (Vercel, 2023).

Для взаємодії з базою даних було розглянуто кілька ORM-інструментів, зокрема Prisma, Sequelize та TypeORM, з урахуванням критеріїв продуктивності, інтеграції з TypeScript, простоти використання та можливостей автоматизації. Sequelize пропонує гнучкість і широку підтримку, але має складніший синтаксис і не забезпечує повної інтеграції з TypeScript, що підвищує ризик помилок у великих проєктах. TypeORM надає кращу

інтеграцію з TypeScript і підтримує декоратори, але вимагає більше часу на конфігурацію та менш ефективний у великих масштабованих системах. Prisma було обрано завдяки її автоматизації моделей, які генеруються на основі схеми бази даних, що прискорює розробку та знижує ризик помилок. Інтеграція з TypeScript забезпечує типізованість запитів, гарантуючи їх коректність, а простий API дозволяє взаємодіяти з базою даних без необхідності написання складних SQL-запитів. Ці переваги роблять Prisma оптимальним вибором для оптимізації роботи з даними та прискорення розробки (Ahmed, 2024).

Для адміністративної роботи з базою даних було обрано DataGrip, враховуючи критерії зручності використання, підтримки різних типів баз даних, функціональності та можливостей інтеграції. DataGrip, як універсальний інструмент для роботи з базами даних, забезпечує інтелектуальний редактор SQL із підтримкою автозавершення та перевірки синтаксису, що знижує ймовірність помилок. Крім того, платформа надає розширені можливості для аналізу структури бази даних, управління схемами та виконання складних запитів. У порівнянні з іншими інструментами, такими як DBeaver або pgAdmin, DataGrip пропонує інтуїтивно зрозумілий інтерфейс, кращу інтеграцію з іншими інструментами JetBrains і більш потужний функціонал для складних сценаріїв адміністрування, що робить його ефективним вибором для управління PostgreSQL у рамках проекту.

РОЗДІЛ 2. СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ ТА ВЗАОЄМОДІЇ З ШТУЧНИМ ІНТЕЛЕКТОМ

2.1 Розробка системи управління контентом економічної аналітики

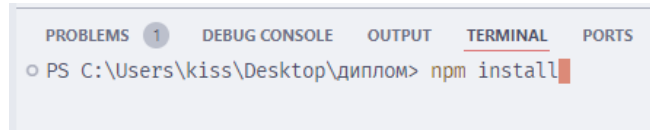
Перш за все проєкт потрібно створити, так як використовується `node.js` (Chanda et al., 2015; Doglio, 2018; Dorasamy, 2022), оптимальним варіантом буде створити в пустій директорії файл `package.json`, з такими параметрами, лістинг 2.1 – файл `package.json`.

Лістинг 2.1 – файл `package.json`

```
{
  "name": "src",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "seed": "ts-node --transpile-only prisma/seed.ts",
    "dev": "ts-node-dev --respawn src/server.ts",
    "build": "tsc",
    "start": "node build/server.js"
  },
  "prisma": {
    "seed": "ts-node --transpile-only prisma/seed.ts"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@fastify/swagger": "^9.3.0",
    "@fastify/swagger-ui": "^5.1.0",
    "@prisma/client": "^5.22.0",
    "@types/bcrypt": "^5.0.2",
    "bcrypt": "^5.1.1",
    "fastify": "^5.1.0",
    "openai": "^4.72.0"
  },
  "devDependencies": {
    "@types/node": "^22.9.0",
    "@types/swagger-schema-official": "^2.0.25",
    "prisma": "^5.22.0",
    "ts-node": "^10.9.2",
    "ts-node-dev": "^2.0.0",
    "typescript": "^5.6.3"
  }
}
```

Також в лістингу 2.1 можна побачити усі бібліотеки які використовувались під час розробки, їхні версії та інше. Детально зупинятись на таких подробицях не варто, адже ключові технології вже були описані.

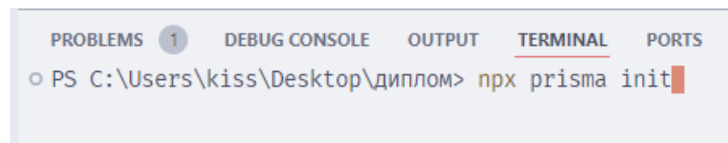
Наступним кроком буде виконання такої команди завантаження пакетів, вона завантажить зазначені бібліотеки з лістинга 2.1 локально на комп'ютер в директорію проєкта, рис. 2.2 – Команда завантаження пакетів.



```
PROBLEMS 1 DEBUG CONSOLE OUTPUT TERMINAL PORTS
PS C:\Users\kiss\Desktop\диплом> npm install
```

Рисунок 2.2 – Команда завантаження пакетів

Тепер маючи інстальовані пакети, можна виконати команду для ініціалізації інструментів Prisma в рамках проєкту, рис. 2.3.



```
PROBLEMS 1 DEBUG CONSOLE OUTPUT TERMINAL PORTS
PS C:\Users\kiss\Desktop\диплом> npx prisma init
```

Рисунок 2.3 – Команда ініціалізації Prisma

Для того, аби Prisma працювала з базою даних, потрібно змінити конфігурацію проєкту, в межах розробки використовуються локальна база даних, рис. 2.4.

```
6
7 DATABASE_URL="postgresql://postgres:4200@localhost:5432/cms?schema=public"
```

Рисунок 2.4 – Конфігурація проєкту

Тепер все готово для створення коду, який далі буде використаний для генерації бази даних. Код для генерації бази даних зображений на рис. 2.5.

```

12 enum Role {
13     ADMIN
14     EDITOR
15     AUTHOR
16 }
17
18 model User {
19     id Int @id @default(autoincrement())
20     name String
21     email String @unique
22     passwordHash String
23     role Role @default(AUTHOR)
24     articles Article[] // Articles authored by the user
25     comments Comment[] // Comments made by the user
26     createdAt DateTime @default(now())
27     updatedAt DateTime @updatedAt
28 }
29
30 model Article {
31     id Int @id @default(autoincrement())
32     title String
33     slug String @unique Add to Chat Ctrl+Shift+L Edit Ctrl+K
34     content String
35     published Boolean @default(false)
36     author User @relation(fields: [authorId], references: [id])
37     authorId Int
38     categories Category[] // Many-to-many relation with Category
39     tags Tag[] // Many-to-many relation with Tag
40     comments Comment[] // Comments on the article
41     createdAt DateTime @default(now())
42     updatedAt DateTime @updatedAt
43 }
44
45 model Category {
46     id Int @id @default(autoincrement())
47     name String
48     slug String @unique
49     articles Article[] // Many-to-many relation with Article
50 }
51
52 model Tag {
53     id Int @id @default(autoincrement())
54     name String
55     slug String @unique
56     articles Article[] // Many-to-many relation with Article
57 }
58
59 model Comment {
60     id Int @id @default(autoincrement())
61     content String
62     author User @relation(fields: [authorId], references: [id])
63     authorId Int
64     article Article @relation(fields: [articleId], references: [id])
65     articleId Int
66     createdAt DateTime @default(now())
67     updatedAt DateTime @updatedAt
68 }
69

```

Рисунок 2.5 – Код схеми

Для того, щоб виконати цей код потрібно запустити команду генерації бази даних, рис. 2.6.

```
PS C:\Users\kiss\Desktop\диплом> npx prisma generate
>> npx prisma migrate dev --name init
```

Рисунок 2.6 – Команда генерації бази даних

В результаті виконання команди генерації бази даних (рис. 2.6), отримано база даних, що схематично виглядає як на рис. 2.7.

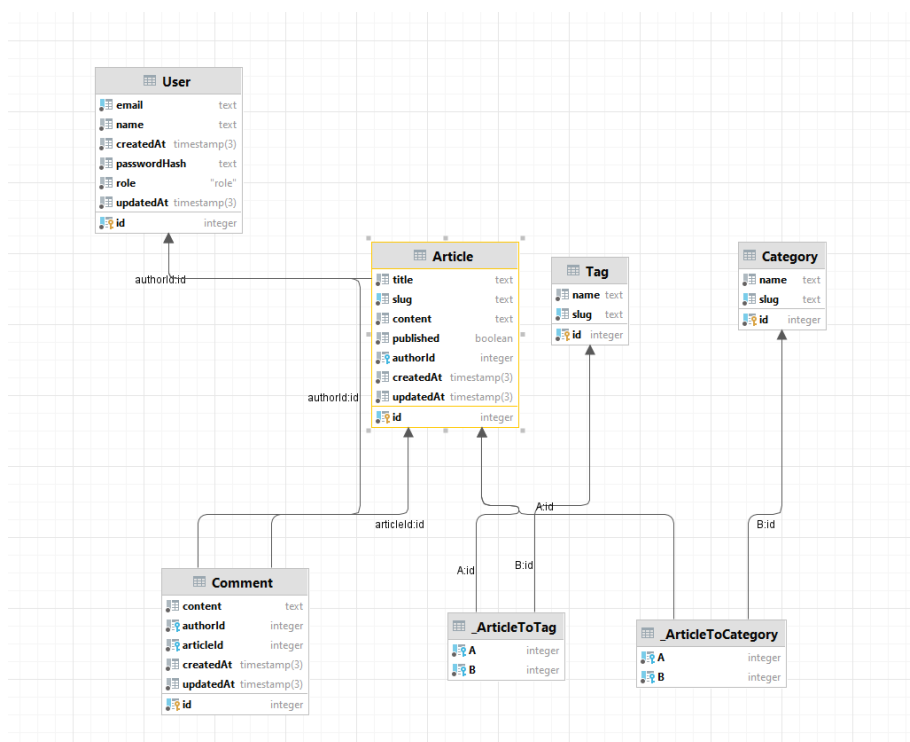


Рисунок 2.7 – Візуалізація бази даних

Маючи уявлення про базу даних можна більш детально розібратись у сутностях, що були створені:

- користувачі можуть писати статті та залишати коментарі;
- статті можуть бути пов'язані з багатьма категоріями та тегами;
- Категорії та теги допомагають організувати контент;
- Коментарі пов'язані з конкретними статтями та користувачами.

Отримана схема дозволяє:

- управляти користувачами з різними ролями та правами доступу;

- створювати та публікувати статті, організовувати їх за категоріями та тегами;
- дозволити користувачам коментувати статті, забезпечуючи взаємодію з аудиторією;
- стежити за змінами через поля `createdAt` та `updatedAt`.

Отже, маючи згенеровану базу даних, можна почати роботу над модулями API для роботи з сутностями бази даних. Для кожної сутності будуть створенні методи CRUD – Create, Read, Update, Delete операції, це дозволить мати чіткий протокол взаємодії з сутностями, і водночас бути легкою та гнучкою для того аби бути налаштованою як підсистема до будь якої системи, що дозволить розглядати систему і як самостійний застосунок, і як плагін до інших. Також до кожного модулю буде налаштована система документації Swagger яка буде генерувати з коду працюючи з можливістю виконання запитів. На етапі тесту, а також для подальшої розробки подібна річ стане в нагоді. Загалом система документації має такий вигляд, як на рис. 2.8.

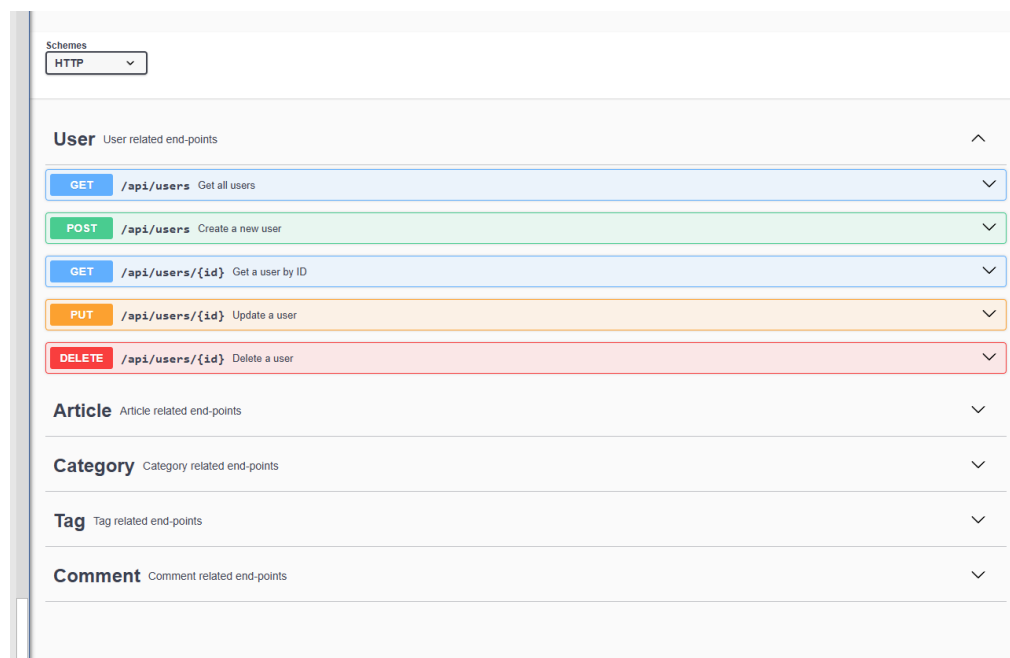


Рисунок 2.8 – Swagger документація.

Також на рис. 2.8 вже зображена готова документація, а програмний код реалізації модулів для роботи з сутностями доступний в додатку A1, A2, A3, A4, A5, A6.

2.2 Розробка модулю взаємодії зі штучним інтелектом

Наступним етапом буде налаштування параметрів ШІ для отримання бажаного результату. Серед усіх налаштувань що доступні, можна виділити наступні: кількість токенів, температура, та базова інструкція (промпт).

Кількість токенів визначає максимальну кількість слів чи символів, які модель може згенерувати за один запит. Для економічних текстів рекомендовано обирати 200–500 токенів для коротких відповідей чи пояснень, а для статей чи аналітичних матеріалів – 1000–2000 токенів. Температура є параметром, що впливає на рівень креативності відповіді: низька температура (0–0.3) забезпечує консервативні та логічні відповіді, що підходять для аналітики, середня (0.4–0.7) забезпечує баланс між фактичністю і варіативністю, ідеальна для економічних оглядів, а висока (0.8–1) використовується для створення нових ідей чи футуристичних сценаріїв. Для економічних тем точні дані краще обробляти при температурі 0.2–0.4, а для обговорення тенденцій – 0.5–0.7. Якісний промпт має починатися з чіткої теми або завдання, наприклад: "Розкажи про вплив монетарної політики на економічне зростання в Україні." Важливо вказати формат відповіді (наприклад, "Напиши структуровану статтю з основними пунктами: вступ, причини, наслідки, рекомендації"), додати контекст (наприклад, "Врахуй економічні показники України за останні 5 років") та визначити стиль написання (аналітичний, популярно-науковий чи професійний).

В ході розробки та під час тестування на реальних статтях було обрано такі параметри:

gpt-4o:

- Кількість токенів: 15000;
- Температура: 0.7

o1-preview:

- Кількість токенів: 30000;
- Температура: 0.7

Базова інструкція в них однакова, та досить проста, для демонстрації роботи «чистих» моделей було обрано такий варіант: «Based on the following context, generate a concise and relevant article in Markdown format, strictly related to economic, business, or finance subjects. Ensure the content is safe and appropriate. First line should be a topic of article. Do not rewrite tables. Focus on interpretation of data.». Він зберігає універсальність, можливість в подальшому оновленні без значних втрат якості через «перенавчання», та фокус на економічній тематиці, що звужує контекст та робить моделі більш сфокусованими на предметі контекста.

Така температура була обрана для можливості інтерпретації наданого аналізу, хоч і для «точної» генерації цей параметр вважається досить високим.

Така кількість токенів дозволяє отримати достатню кількість символів в тексті, щоб вважати цей текст – статтею.

РОЗДІЛ 3. РЕЗУЛЬТАТИ РОЗРОБКИ СИСТЕМИ

3.1 Результати розробки системи менеджменту контенту

В результаті розробки було розроблено систему менеджменту контенту, аби задовольнити потреби в гнучкості та універсальності, при розробці методів API, була дотримана CRUD архітектура, що дозволило зберегти універсальність, мінімалістичність та гнучкість. Кожен ресурс отримав свої запити для створення, читання, редагування та оновлення, та видалення. Що дозволяє не тільки створювати статті на основі ШІ, але і працювати з ними потім. Були створенні API ресурси для таких сутностей як: Article, Category, User, Tag. Всі вони необхідні для роботи системи.

Наприклад, для того, аби створити новий об'єкт сутності Category, потрібно використати POST запит з такими конфігураціями, як на рисунку 2.9.

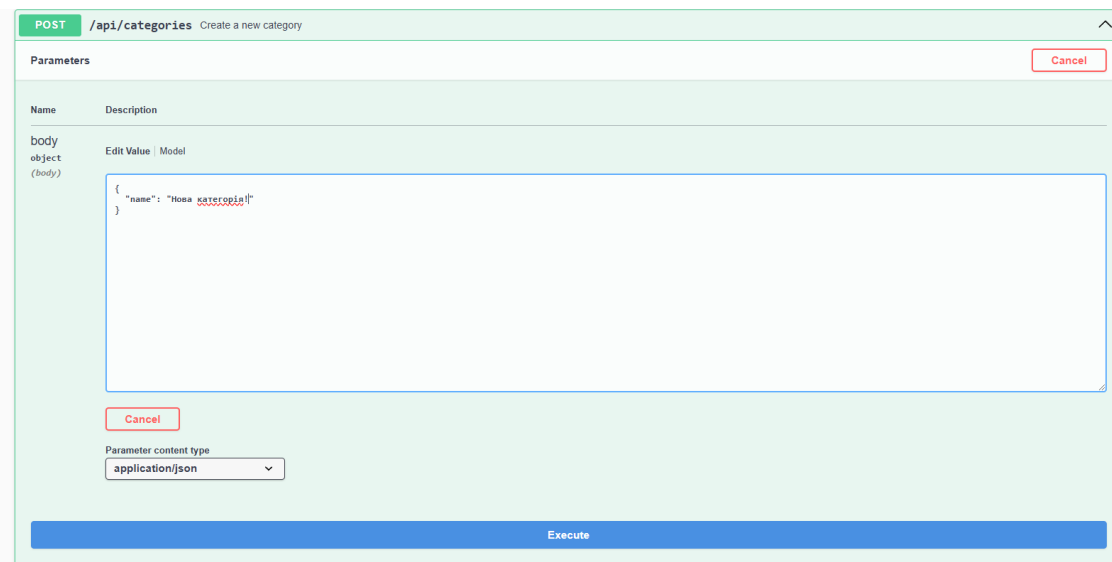


Рисунок 2.9 – Запит створення категорії.

В результаті виконання методу, буде створенна нова категорія с назвою «Нова категорія!», після кліку на кнопку «Execute», була створенна нова категорія, рис. 2.10.

WHERE		ORDER BY
id	name	slug
1	Economics	economics
2	Finance	finance
3	Business	business
4	Нова категорія!	

Рисунок 2.10 – Нова категорія

Відповідно для отримання всіх категорій потрібно використовувати запит GET, для видалення DELETE.

3.2 Результати розробки модулю штучного інтелекту

В результаті розробки модулю ШІ було розроблено метод для генерації та збереження статті в базу даних в Markdown форматі, рис. 2.11 – Метод для генерації та збереження статті.

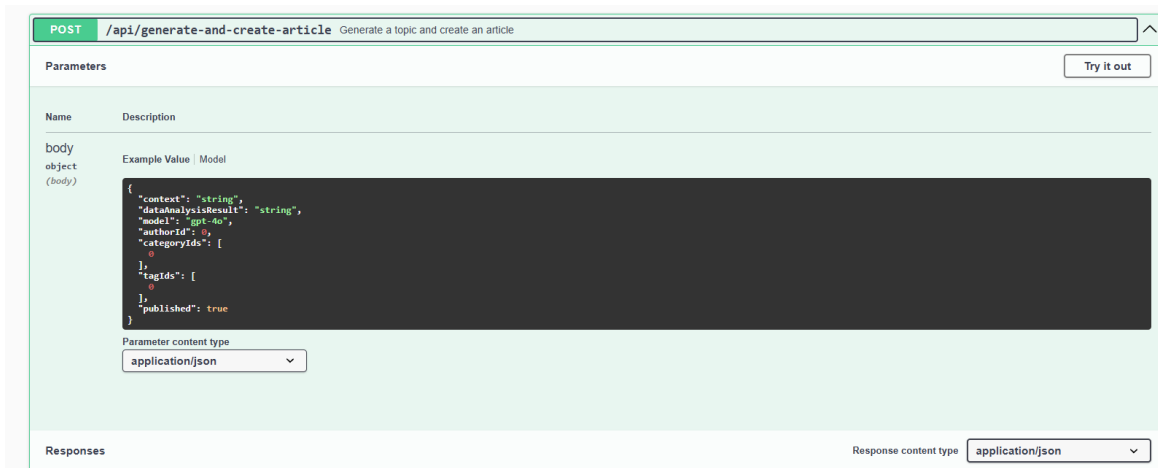


Рисунок 2.11 – Метод для генерації та збереження статті.

Загалом він має таку структуру (лістинг 2.12).

Лістинг 2.12 – структура запиту

```

{
  "context": " Sales Revenue (dependent variable, Y) as a function of Advertising Expenditure (independent variable, X)",
  "dataAnalysisResult": "Model Summary R-squared: 0.963 Adjusted R-squared: 0.958 F-statistic: 232.29 p-value (F-statistic): <0.0001 Residual Standard Error Standard Error of Estimate (SEE): 1.58",

```

```

"model": "gpt-4o",
"authorId": 1,
"categoryIds": [
  2
],
"tagIds": [
  1, 2
],
"published": true
}

```

Де,

- context – тема результатів аналізу даних, аби дати розуміння моделі про що робити статтю;
- dataAnalysisResult – результат аналізу даних;
- model – модель для запиту, в контексті цієї роботи gpt-4o та o1-preview;
- authorId – унікальне поле id автора;
- categoryIds – унікальні поля id категорій;
- tagIds – унікальні поля id тегів;
- published – флаш який буде використаний щоб відразу «опублікувати» статтю.

Для тестування було обрано результати аналізу даних зі статті «Data Mining Technology of Country’s Digital Development Level Assessing for Economic Development and Sustainable Growth: Multivariate Adaptive Regression Spline», а саме результати розрахунку VIF аналізу (Lyeonov, та ін., 2024).

Для створення статті з допомогою моделі gpt-4o, використовувалась така конфігурація метода, лістинг 2.13 – Конфігурація для gpt-4o, та конфігурація для o1-preview, лістинг 2.14 – Конфігурація для o1-preview

Лістинг 2.13 – Конфігурація для gpt-4o

```

{
  "context": "Data Mining Technology of Country\u2019s Digital Development Level Assessing for Economic Development and Sustainable Growth: Multivariate Adaptive Regression Spline. Ddl - digital development level, NCSI - National cyber security index, SEES - Easy of Doing Business",
  "dataAnalysisResult": "Regression analysis Parameter: Estimate, Standard Error, T Statistic, P-Value CONSTANT:-34.2143, 6.21907, -5.50152,0.0000 NCSI: 0.263034, 0.0476427, 5.52097,0.0000 SEES: 1.0853, 0.107924, 10.0562,0.0000 Analysis of Variance Source: Sum of Squares, Df, Mean Square, F-Ratio, P-Value Model: 27370.9, 2, 13685.4, 174.43, 0.0000 Residual: 7845.8, 100, 78.458, Total (Corr.): 35216.7, 102, Calculating VIF-test Coefficients, Standard error, t-Stat, p-value, Lower 95%, Upper 95%, Lower 95.0%, Upper 95.0%, R-squared, VIF Intercept: -34.214, 6.219, -5.502, 0.0000, -46.553, -21.876, -46.553, -21.876, NCSI: 0.2630, 0.048, 5.530, 0.0000, 0.169, 0.358, 0.169, 0.358, 0.470, 1.887 SEES: 1.0853, 0.108, 10.056, 0.0000, 0.871, 1.299, 0.871, 1.299, 0.417, 1.716",
  "model": "gpt-4o",

```

```
"authorId": 1,
"categoryIds": [1],
"tagIds": [1],
"published": true
}
```

Лістинг 2.14 – Конфігурація для o1-preview

```
{
  "context": "Data Mining Technology of Country\u2019s Digital Development Level Assessing for Economic Development and Sustainable Growth: Multivariate Adaptive Regression Spline. Ddl - digital development level, NCSI - National cyber security index, SEES - Easy of Doing Business",
  "dataAnalysisResult": "Regression analysis Parameter: Estimate, Standard Error, T Statistic, P-Value CONSTANT:-34.2143, 6.21907, -5.50152,0.0000 NCSI: 0.263034, 0.0476427, 5.52097,0.0000 SEES: 1.0853, 0.107924, 10.0562,0.0000 Analysis of Variance Source: Sum of Squares, Df, Mean Square, F-Ratio, P-Value Model: 27370.9, 2, 13685.4, 174.43, 0.0000 Residual: 7845.8, 100, 78.458, Total (Corr.): 35216.7, 102, Calculating VIF-test Coefficients, Standard error, t-Stat, p-value, Lower 95%, Upper 95%, Lower 95.0%, Upper 95.0%, R-squared, VIF Intercept: -34.214, 6.219, -5.502, 0.0000, -46.553, -21.876, -46.553, -21.876, NCSI: 0.2630, 0.048, 5.530, 0.0000, 0.169, 0.358, 0.169, 0.358, 0.470, 1.887 SEES: 1.0853, 0.108, 10.056, 0.0000, 0.871, 1.299, 0.871, 1.299, 0.417, 1.716",
  "model": "o1-preview",
  "authorId": 1,
  "categoryIds": [1],
  "tagIds": [1],
  "published": true
}
```

В результаті виконання цих запитів з конфігураціями на лістингу 2.13 та лістингу 2.14, були отримані такі результати з моделі gpt-4o:

```
# Assessing Digital Development's Impact on Economic Growth: A Data-Driven Approach
```

In a rapidly digitizing world, understanding the relationship between digital development and economic growth has become crucial for policymakers. This article explores how digital development indicators such as the National Cyber Security Index (NCSI) and the Ease of Doing Business (SEES) impact a country's economic prosperity, using a multivariate adaptive regression spline model.

```
## Key Findings from the Regression Analysis
```

```
### 1. **Model Fit and Significance**
```

The regression model demonstrates a strong fit with an R-squared value indicating that a significant proportion of the variability in economic development is explained by the independent variables, NCSI and SEES. The overall model is statistically significant (P-value = 0.0000).

```
### 2. **Impact of National Cyber Security Index (NCSI)**
```

```
- **Coefficient Estimate**: 0.263
```

- **Interpretation**: A unit increase in the NCSI corresponds to an estimated 0.263 increase in the economic development level, holding other factors constant.

- **Statistical Significance**: Highly significant (P-value = 0.0000), suggesting that improvements in cyber security are positively correlated with economic growth.

3. **Impact of Ease of Doing Business (SEES)**

- **Coefficient Estimate**: 1.085

- **Interpretation**: A unit increase in the Ease of Doing Business score leads to an estimated 1.085 increase in economic development level, ceteris paribus.

- **Statistical Significance**: This relationship is also highly significant (P-value = 0.0000), underscoring the importance of a business-friendly environment in fostering economic growth.

4. **Variance Inflation Factor (VIF) Analysis**

The VIF values for both NCSI and SEES are below 2, indicating that multicollinearity is not a concern and the estimates are reliable.

Implications for Policy and Economic Strategy

The analysis highlights two critical areas for enhancing economic development:

- **Strengthening Cyber Security**: As cyber threats continue to evolve, investing in robust cybersecurity measures can not only protect national infrastructure but also enhance economic performance by fostering trust and stability in digital transactions.

- **Improving Business Environments**: Simplifying regulations and reducing bureaucratic hurdles can dramatically improve the ease of doing business, thereby attracting investment and stimulating economic activity.

Conclusion

The findings underscore the pivotal role of digital development in economic growth. Countries aiming for sustainable economic advancement must focus on bolstering cybersecurity infrastructure and creating conducive business environments. By doing so, they can harness the full potential of digital transformation to drive economic prosperity.

Conclusion

The analysis using MARS methodology provides compelling evidence of the positive impact of digital development on economic growth. By leveraging insights from digital development metrics, countries and businesses can foster sustainable economic growth. As digital transformation continues to shape the global economy, these findings offer a roadmap for strategic investments and policy-making aimed at harnessing the full potential of digital technologies.

Та результати з моделі o1-preview:

```
# The Impact of Digital Development on Economic Growth: A Regression Analysis
# Assessing the Impact of Cybersecurity and Ease of Doing Business on Digital
Development Levels
```

The digital development level (DDL) of a country is a crucial indicator of

its ability to leverage technology for economic growth and sustainable development. Understanding the factors that influence DDL can help policymakers craft strategies to enhance their nation's digital infrastructure and competitiveness. This article interprets a regression analysis examining how the National Cyber Security Index (NCSI) and the Ease of Doing Business (SEES) affect a country's DDL.

Key Findings from the Regression Analysis

The regression model investigated the relationship between DDL (dependent variable) and two independent variables: NCSI and SEES. The results indicate a strong and significant impact of both variables on DDL.

Regression Coefficients Interpretation

- ****Intercept (-34.214):**** The negative intercept suggests that in the absence of cybersecurity measures and favorable business conditions, the digital development level would be substantially low. While unrealistic in practical terms, it underscores the foundational importance of these factors.
- ****National Cyber Security Index (NCSI) Coefficient (0.263):**** A positive coefficient signifies that higher cybersecurity readiness contributes to higher digital development levels. Specifically, a one-unit increase in NCSI is associated with a 0.263 unit increase in DDL, holding other factors constant.
- ****Ease of Doing Business (SEES) Coefficient (1.085):**** This strong positive coefficient indicates that improving the ease of doing business significantly boosts digital development. A one-unit increase in SEES leads to an approximate 1.085 unit rise in DDL, *ceteris paribus*.

Statistical Significance

All coefficients are statistically significant with p-values less than 0.001, highlighting the reliability of the relationships:

- ****T-Statistics:****
 - Intercept: -5.502
 - NCSI: 5.521
 - SEES: 10.056

These high t-values further confirm the strong influence of NCSI and SEES on DDL.

Model Evaluation

Analysis of Variance (ANOVA)

- ****Model Sum of Squares (27370.9) vs. Residual Sum of Squares (7845.8):**** A large proportion of the variability in DDL is explained by the model.
- ****F-Ratio (174.43) and P-Value (< 0.001):**** The overall model is statistically significant.

Variance Inflation Factor (VIF)

- ****NCSI VIF (1.887) and SEES VIF (1.716):**** Both VIF values are well below the threshold of 5, indicating that multicollinearity is not a concern in this model.

Implications for Economic Development

The analysis demonstrates that enhancing cybersecurity measures and simplifying business processes are vital for digital advancement:

- **Cybersecurity Investment:** Strengthening cybersecurity infrastructure builds trust in digital systems, encourages innovation, and attracts foreign investment.

- **Regulatory Reforms:** Streamlining regulations and reducing bureaucratic hurdles can stimulate entrepreneurship, foster competitive markets, and accelerate technological adoption.

Conclusion

The significant positive effects of NCSI and SEES on DDL highlight actionable areas for policymakers. By focusing on improving cybersecurity and the ease of doing business, countries can elevate their digital development levels, driving economic growth and achieving sustainable development goals.

Ці статті слугують базовими ітераціями для створення контенту, демонструючи методологічні відмінності та унікальні особливості. Обидві роботи аналізують вплив Національного індексу кібербезпеки (NCSI) та показника легкості ведення бізнесу (SEES) на цифровий розвиток країни, використовуючи регресійні моделі для інтерпретації позитивних і значущих коефіцієнтів цих показників (NCSI: 0.263, SEES: 1.085).

Вони також підкреслюють відсутність мультиколінеарності за допомогою аналізу фактору інфляції дисперсії (VIF) і роблять висновок, що покращення кібербезпеки та умов ведення бізнесу є критично важливими для стимулювання економічного зростання.

Однак gpt-4o обирає ширший підхід, акцентуючи увагу на економічних стратегіях із нарративним стилем аналізу, що включає підрозділи про придатність моделі та рекомендації для політики.

Натомість o1-preview пропонує більш технічний підхід, зосереджуючись на рівнях цифрового розвитку (DDL) із детальними статистичними результатами, включно з інтерпретаціями перехоплення, результатами ANOVA та t-статистиками. GPT-4o робить стратегічні висновки, орієнтовані на сталий розвиток, тоді як o1-Preview завершується практичними рекомендаціями, специфічними для покращення DDL.

Ці статті є прикладом комплементарних підходів, що поєднують стратегічну та аналітичну глибину для підтримки ітеративного створення контенту, спрямованого на широку аудиторію.

ВИСНОВКИ

Результати проведеного дослідження підтвердили високу актуальність та практичну ефективність запропонованого підходу до автоматизації управління контентом економічної аналітики за допомогою сучасних інструментів штучного інтелекту. Запропонована система є інтеграцією інноваційних технологій із традиційними підходами, що дозволяє створювати якісний, структурований та готовий до використання аналітичний контент. Зокрема, використання архітектури Headless CMS у поєднанні з мовними моделями GPT і передовими базами даних забезпечує гнучкість, масштабованість і продуктивність системи, що відповідає сучасним вимогам цифрової економіки.

У результаті виконання кваліфікаційної роботи досягнуто поставленої мети та виконано всі завдання. У першому розділі було оглянуто теоретичний мінімум щодо визначень систем управління контентом (CMS) і штучного інтелекту (ШІ), проведено аналіз актуальності автоматизованих систем управління контентом у контексті цифрової трансформації, виконано бібліометричний аналіз сучасних тенденцій і технологій у цій сфері. Також досліджено технічні деталі, включаючи архітектуру Headless CMS, мовні моделі GPT, бази даних PostgreSQL і фреймворк Fastify.

У другому розділі розроблено базу даних для збереження контенту, що включає структури для статей, категорій, тегів і користувачів, реалізовано API для CRUD-операцій із цими сутностями. Налаштовано модуль взаємодії зі ШІ, визначено параметри генерації текстів, зокрема кількість токенів, температуру

генерації та базові інструкції, а також інтегровано мовні моделі GPT-4o і o1-preview для генерації текстів на основі даних економічної аналітики.

У третьому розділі продемонстровано приклади роботи системи, включаючи API для створення та управління статтями, і результати генерації текстів на основі реальних даних економічної аналітики, що підтвердили практичну цінність розробленої системи. Проведене тестування підтвердило ефективність, гнучкість і відповідність системи сучасним вимогам.

Отже, виконані завдання довели доцільність обраного підходу до розробки автоматизованої системи менеджменту контенту, інтегрованої зі ШІ. Це сприяло підвищенню ефективності створення економічного контенту та оптимізації роботи з великими обсягами даних. Результати роботи мають значне практичне значення й відкривають перспективи для подальшого розвитку в різних галузях економіки. Практична значимість роботи полягає у можливості суттєво зменшити затрати часу та ресурсів на створення текстів для економічного аналізу, що є критично важливим у контексті стрімкого зростання обсягів даних. Це рішення також дозволяє підвищити точність і якість аналітичних матеріалів, що має велике значення для прийняття управлінських рішень на основі об'єктивних даних.

Таким чином, результати роботи мають не лише теоретичне, але й значне практичне значення, сприяючи оптимізації процесів управління контентом та підвищенню ефективності діяльності в умовах цифрової трансформації економіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ahmed, S. (2024). Popular ORMs and Their Difference: Prisma, TypeORM, and Sequelize. Отримано з Dev: <https://dev.to/shariqahmed525/popular-orms-and-their-difference-prisma-typeorm-and-sequelize-idc>
2. Aria, M., & Cuccurullo, C. (2017). bibliometrix: An R-tool for comprehensive science mapping analysis. *Journal of Informetrics*, 959-975. doi:10.1016/j.joi.2017.08.007
3. AWS. (2023). What is Headless CMS? Отримано з AWS: https://aws.amazon.com/what-is/headless-cms/?nc1=h_ls
4. Bi, Q. (2023). Analysis of the Application of Generative AI in Business Management. *Advances in Economics and Management Research*, 6(1). <https://doi.org/10.56028/aemr.6.1.36.2023>
5. Boukar, M. M., Mahamat, A. A., & Djibrine, O. H. (2024). The Impact of Artificial Intelligence (AI) on Content Management Systems (CMS): A Deep Dive. *International Journal of Intelligent Systems and Applications in Engineering*, 12(1).
6. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., . . . Ra. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901. Отримано з <https://arxiv.org/abs/2005.14165>
7. Chanda, A., Daly, A. M., Foley, D. A., Lapack, M. A., Mukherjee, S., Orr, J. D., Reid, G. L., Thompson, D. R., & Ward, H. W. (2015). Industry perspectives on process analytical technology: Tools and applications in API development. In *Organic Process Research and Development* (Vol. 19, Issue 1). <https://doi.org/10.1021/op400358b>
8. Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodio, D. (2017). Deep Reinforcement Learning from Human Preferences. doi:<https://doi.org/10.48550/arXiv.1706.03741>
9. Doglio, F. (2018). REST API Development with Node.js: Manage and Understand the Full Capabilities of Successful REST Development, Second Edition. In *REST API Development with Node.js: Manage and Understand the Full Capabilities of Successful REST Development, Second Edition*. <https://doi.org/10.1007/978-1-4842-3715-1>
10. Dorasamy, R. (2022). API Development. In *API Marketplace Engineering*. https://doi.org/10.1007/978-1-4842-7313-5_8

11. Gallardo Paredes, C. M., Machuca, C., & Semblantes Claudio, Y. M. (2023). ChatGPT API: Brief overview and integration in Software Development. *International Journal of Engineering Insights*, 1(1). <https://doi.org/10.61961/injei.v1i1.7>
12. GeeksforGeeks. (2024). Node.js vs Python: Which One is Best? Отримано з GeeksforGeeks: <https://www.geeksforgeeks.org/node-js-vs-python/>
13. Giraud, L., Zaher, A., Hernandez, S., & Akram, A. A. (2023). The impacts of artificial intelligence on managerial skills. *Journal of Decision Systems*, 32(3). <https://doi.org/10.1080/12460125.2022.2069537>
14. Harrer, S. (2023). Attention is not all you need: the complicated case of ethically using large language models in healthcare and medicine. In *eBioMedicine* (Vol. 90). <https://doi.org/10.1016/j.ebiom.2023.104512>
15. He, X. (2022). Exploration of Paths for Artificial Intelligence Technology to Promote Economic Development. *Lecture Notes on Data Engineering and Communications Technologies*, 97. https://doi.org/10.1007/978-3-030-89508-2_4
16. Hu, L., Chen, Z., & Chen, Z. (2021). Impact of Artificial Intelligence on Economic Development. *Proceedings of Business and Economic Studies*, 4(5). <https://doi.org/10.26689/pbes.v4i5.2648>
17. International Data Corporation. (2018). The digitization of the world from edge to core. Отримано з <https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf>
18. Kanbach, D. K., Heiduk, L., Blueher, G., Schreiter, M., & Lahmann, A. (2024). The GenAI is out of the bottle: generative artificial intelligence from a business model innovation perspective. In *Review of Managerial Science* (Vol. 18, Issue 4). <https://doi.org/10.1007/s11846-023-00696-z>
19. Kears, M., Moir, R., Wilson, A., Stones-Havas, S., Cheung, M., Sturrock, S., Buxton, S., Cooper, A., Markowitz, S., Duran, C., Thierer, T., Ashton, B., Meintjes, P., & Drummond, A. (2012). Geneious Basic: An integrated and extendable desktop software platform for the organization and analysis of sequence data. *Bioinformatics*, 28(12). <https://doi.org/10.1093/bioinformatics/bts199>
20. Korinek, A., & Stiglitz, J. (2021). Artificial Intelligence, Globalization, and Strategies for Economic Development. *Institute for New Economic Thinking Working Paper Series*. <https://doi.org/10.36687/inetwp146>
21. Lehr, S. A., Caliskan, A., Liyanage, S., & Banaji, M. R. (2024). ChatGPT as Research Scientist: Probing GPT's Capabilities as a Research Librarian,

- Research Ethicist, Data Generator and Data Predictor. arXiv. Отримано з <https://arxiv.org/abs/2406.14765>
22. Lyeonov, S., Bilan, Y., Koibichuk, V., Malyarets, L., Ahmad, A., & Secarǎ, C. G. (2024). Data Mining Technology of Country's Digital Development Level Assessing for Economic Development and Sustainable Growth: Multivariate Adaptive Regression Spline. *Journal of the Knowledge Economy*, 13-15. doi:<https://doi.org/10.1007/s13132-024-01958-1>
 23. McKinsey. (2023). The economic potential of generative AI: The next productivity frontier. Отримано з McKinsey Digital: <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier>
 24. Nyenno, I., Truba, V., & Tokarchuk, L. (2023). MANAGERIAL FUTURE OF THE ARTIFICIAL INTELLIGENCE. In *Virtual Economics* (Vol. 6, Issue 2). [https://doi.org/10.34021/ve.2023.06.02\(5\)](https://doi.org/10.34021/ve.2023.06.02(5))
 25. Odom, L. (09 05 2022 р.). ELI5: Що таке Headless CMS & Чому варто її використовувати? Отримано з DreamHost: <https://www.dreamhost.com/blog/uk/headless-cms/>
 26. Ofoeda, J., Boateng, R., & Effah, J. (2023). An institutional perspective on application programming interface development and integration. *Information Technology and People*. <https://doi.org/10.1108/ITP-11-2022-0826>
 27. Ooi, K. B., Tan, G. W. H., Al-Emran, M., Al-Sharafi, M. A., Capatina, A., Chakraborty, A., Dwivedi, Y. K., Huang, T. L., Kar, A. K., Lee, V. H., Loh, X. M., Micu, A., Mikalef, P., Mogaji, E., Pandey, N., Raman, R., Rana, N. P., Sarker, P., Sharma, A., ... Wong, L. W. (2023). The Potential of Generative Artificial Intelligence Across Disciplines: Perspectives and Future Directions. *Journal of Computer Information Systems*. <https://doi.org/10.1080/08874417.2023.2261010>
 28. OpenAI. (2023). GPT-4 Technical Report. OpenAI.
 29. Qin, Y., Xu, Z., Wang, X., & Skare, M. (2024). Artificial Intelligence and Economic Development: An Evolutionary Investigation and Systematic Review. *Journal of the Knowledge Economy*, 15(1). <https://doi.org/10.1007/s13132-023-01183-2>
 30. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. OpenAI. Отримано з https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

- 31.SentinelOne. (2024). AI Threat Detection: Leverage AI to Detect Security Threats. Отримано з SentinelOne: <https://www.sentinelone.com/cybersecurity-101/data-and-ai/ai-threat-detection/>
- 32.Sundarambal, B., Karthikeyini, C., Bommi, R. M., Subramanian, S., & Lokesh, M. (2023). AI Based Content Management System. 2023 *International Conference on Data Science, Agents and Artificial Intelligence, ICDSAAI 2023*. <https://doi.org/10.1109/ICDSAAI59313.2023.10452640>
- 33.Sundaresan, S., & Zhang, Z. (2022). AI-enabled knowledge sharing and learning: redesigning roles and processes. *International Journal of Organizational Analysis*, 30(4). <https://doi.org/10.1108/IJOA-12-2020-2558>
- 34.The New Stack. (2023). A Showdown Between Express.js and Fastify Web App Frameworks. Отримано з The New Stack: <https://thenewstack.io/a-showdown-between-express-js-and-fastify-web-app-frameworks/>
- 35.Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *Attention Is All You Need* (с. 5998–6008). Long Beach: Curran Associates, Inc. doi:<https://doi.org/10.48550/arXiv.1706.03762>
- 36.Vercel. (2023). Comparing MySQL, PostgreSQL, and MongoDB . Отримано з Vercel: <https://vercel.com/guides/mysql-vs-postgresql-vs-mongodb>
- 37.Wach, K., Duong, C. D., Ejdys, J., Kazlauskaitė, R., Korzynski, P., Mazurek, G., Paliszkievicz, J., & Ziemba, E. (2023). The dark side of generative artificial intelligence: A critical analysis of controversies and risks of ChatGPT. *Entrepreneurial Business and Economics Review*, 11(2). <https://doi.org/10.15678/EBER.2023.110201>
- 38.Wu, Y., Yan, L., Luo, Y., Wang, Y., & Tang, N. (2024). ChartInsights: Evaluating Multimodal Large Language Models for Low-Level Chart Question Answering. arXiv. Отримано з <https://arxiv.org/abs/2405.07001>
- 39.Yu, Z., & Gong, Y. (2024). ChatGPT, AI-generated content, and engineering management. In *Frontiers of Engineering Management* (Vol. 11, Issue 1). <https://doi.org/10.1007/s42524-023-0289-6>
- 40.Кузнецова, А. (2023). Чи чули ви про Headless CMS? Що це таке, як працює та як обрати найкращу для свого бізнесу. Отримано з [webpromo](https://webpromo.com).

ДОДАТКИ

ДОДАТОК А

Лістинг А1 – Код модулю статей

```

import { FastifyInstance, RouteShorthandOptions } from "fastify";
import { PrismaClient } from "@prisma/client";
const prisma = new PrismaClient();
export async function articleRoutes(server: FastifyInstance) {
  // Article schema
  const Article = {
    type: "object",
    properties: {
      id: { type: "number" },
      title: { type: "string" },
      slug: { type: "string" },
      content: { type: "string" },
      published: { type: "boolean" },
      authorId: { type: "number" },
      createdAt: { type: "string", format: "date-time" },
      updatedAt: { type: "string", format: "date-time" },
    },
  };
  // Options for GET /articles
  const getArticlesOpts: RouteShorthandOptions = {
    schema: {
      tags: ["Article"],
      summary: "Get all published articles",
      response: {
        200: {
          description: "List of articles",
          type: "array",
          items: Article,
        },
      },
    },
  };
  // Get all articles
  server.get("/articles", getArticlesOpts, async (request, reply) => {
    const articles = await prisma.article.findMany({
      where: { published: true },
      include: {
        author: true,
        categories: true,
        tags: true,
        comments: true,
      },
    });
    reply.send(articles);
  });
  // Options for GET /articles/:slug
  const getArticleOpts: RouteShorthandOptions = {
    schema: {
      tags: ["Article"],
      summary: "Get an article by slug",
      params: {
        type: "object",
        properties: {
          slug: { type: "string" },
        },
        required: ["slug"],
      },
    },
    response: {
      200: Article,
      404: {
        description: "Article not found",
        type: "object",
        properties: {
          error: { type: "string" },
        },
      },
    },
  };
}

```

```

    },
  },
},
},
};
// Get a single article by slug
server.get("/articles/:slug", getArticleOpts, async (request, reply) => {
  const { slug } = request.params as { slug: string };
  const article = await prisma.article.findUnique({
    where: { slug },
    include: {
      author: true,
      categories: true,
      tags: true,
      comments: true,
    },
  });
  if (article) {
    reply.send(article);
  } else {
    reply.status(404).send({ error: "Article not found" });
  }
});
// Options for POST /articles
const createArticleOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Article"],
    summary: "Create a new article",
    body: {
      type: "object",
      required: ["title", "content", "authorId"],
      properties: {
        title: { type: "string" },
        content: { type: "string" },
        authorId: { type: "number" },
        categoryIds: {
          type: "array",
          items: { type: "number" },
        },
        tagIds: {
          type: "array",
          items: { type: "number" },
        },
        published: { type: "boolean" },
      },
    },
  },
  response: {
    201: Article,
    500: {
      description: "Article creation failed",
      type: "object",
      properties: {
        error: { type: "string" },
      },
    },
  },
},
};
// Create a new article
server.post("/articles", createArticleOpts, async (request, reply) => {
  const { title, content, authorId, categoryIds, tagIds, published } =
    request.body as {
      title: string;
      content: string;
      authorId: number;
      categoryIds?: number[];
      tagIds?: number[];
      published?: boolean;
    };
  // Generate a slug from the title
  const slug = title
    .toLowerCase()

```

```

.replace(/[^\a-z0-9]+/g, "-")
.replace(/(^-|-$)+/g, "");

try {
  const article = await prisma.article.create({
    data: {
      title,
      slug,
      content,
      published: published || false,
      authorId,
      categories: categoryIds
      ? {
        connect: categoryIds.map((id) => ({ id })),
      }
      : undefined,
      tags: tagIds
      ? {
        connect: tagIds.map((id) => ({ id })),
      }
      : undefined,
    },
  });
  reply.status(201).send(article);
} catch (error) {
  reply.status(500).send({ error: "Article creation failed" });
}
});
// Options for PUT /articles/:slug
const updateArticleOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Article"],
    summary: "Update an article",
    params: {
      type: "object",
      properties: {
        slug: { type: "string" },
      },
    },
    required: ["slug"],
  },
  body: {
    type: "object",
    properties: {
      title: { type: "string" },
      content: { type: "string" },
      categoryIds: {
        type: "array",
        items: { type: "number" },
      },
      tagIds: {
        type: "array",
        items: { type: "number" },
      },
      published: { type: "boolean" },
    },
  },
  response: {
    200: Article,
    404: {
      description: "Article not found",
      type: "object",
      properties: {
        error: { type: "string" },
      },
    },
  },
},
});
// Update an article
server.put("/articles/:slug", updateArticleOpts, async (request, reply) => {
  const { slug } = request.params as { slug: string };
  const { title, content, categoryIds, tagIds, published } = request.body as {
    title?: string;
  };

```

```

content?: string;
categoryIds?: number[];
tagIds?: number[];
published?: boolean;
};

// Generate new slug if title is being updated
const newSlug = title
  ? title
    .toLowerCase()
    .replace(/^[^a-z0-9]+/g, "-")
    .replace(/^(^|-)+/g, "")
  : undefined;
try {
const article = await prisma.article.update({
  where: { slug },
  data: {
    title,
    slug: newSlug,
    content,
    published,
    categories: categoryIds
    ? {
      set: categoryIds.map((id) => ({ id })),
    }
    : undefined,
    tags: tagIds
    ? {
      set: tagIds.map((id) => ({ id })),
    }
    : undefined,
  },
  include: {
    author: true,
    categories: true,
    tags: true,
    comments: true,
  },
});
reply.send(article);
} catch (error) {
  reply.status(404).send({ error: "Article not found" });
}
});
// Options for DELETE /articles/:slug
const deleteArticleOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Article"],
    summary: "Delete an article",
    params: {
      type: "object",
      properties: {
        slug: { type: "string" },
      },
      required: ["slug"],
    },
  },
  response: {
    204: {
      description: "Article successfully deleted",
      type: "null",
    },
    404: {
      description: "Article not found",
      type: "object",
      properties: {
        error: { type: "string" },
      },
    },
  },
},
};
// Delete an article
server.delete(

```

```

"/articles/:slug",
deleteArticleOpts,
async (request, reply) => {
  const { slug } = request.params as { slug: string };
  try {
    await prisma.article.delete({
      where: { slug },
    });
    reply.status(204).send();
  } catch (error) {
    reply.status(404).send({ error: "Article not found" });
  }
}
);
}

```

Лістинг А2 – Код модулю категорій

```

import { FastifyInstance, RouteShorthandOptions } from "fastify";
import { PrismaClient } from "@prisma/client";
const prisma = new PrismaClient();
export async function categoryRoutes(server: FastifyInstance) {
  // Category schema
  const Category = {
    type: "object",
    properties: {
      id: { type: "number" },
      name: { type: "string" },
      slug: { type: "string" },
      articles: {
        type: "array",
        items: {
          type: "object",
          properties: {
            id: { type: "number" },
            title: { type: "string" },
            slug: { type: "string" },
          },
        },
      },
    },
  };
  // Options for GET /categories
  const getCategoriesOpts: RouteShorthandOptions = {
    schema: {
      tags: ["Category"],
      summary: "Get all categories",
      response: {
        200: {
          description: "List of categories",
          type: "array",
          items: Category,
        },
      },
    },
  };
  // Get all categories
  server.get("/categories", getCategoriesOpts, async (request, reply) => {
    const categories = await prisma.category.findMany({
      include: { articles: true },
    });
    reply.send(categories);
  });
  // Options for GET /categories/:id
  const getCategoryOpts: RouteShorthandOptions = {
    schema: {
      tags: ["Category"],
      summary: "Get a category by ID",
      params: {
        type: "object",

```

```

    properties: {
      id: { type: "number" },
    },
    required: ["id"],
  },
  response: {
    200: Category,
    404: {
      description: "Category not found",
      type: "object",
      properties: {
        error: { type: "string" },
      },
    },
  },
},
};

// Get a single category by ID
server.get("/categories/:id", getCategoryOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  const category = await prisma.category.findUnique({
    where: { id: parseInt(id) },
    include: { articles: true },
  });
  if (category) {
    reply.send(category);
  } else {
    reply.status(404).send({ error: "Category not found" });
  }
});

// Options for POST /categories
const createCategoryOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Category"],
    summary: "Create a new category",
    body: {
      type: "object",
      required: ["name"],
      properties: {
        name: { type: "string" },
      },
    },
    response: {
      201: Category,
      500: {
        description: "Category creation failed",
        type: "object",
        properties: {
          error: { type: "string" },
        },
      },
    },
  },
};

// Create a new category
server.post("/categories", createCategoryOpts, async (request, reply) => {
  const { name } = request.body as { name: string };

  // Generate a slug from the name
  const slug = name
    .toLowerCase()
    .replace(/[a-z0-9]+/g, "-")
    .replace(/(^-|-$)+/g, "");

  try {
    const category = await prisma.category.create({
      data: {
        name,
        slug,
      },
    });
    reply.status(201).send(category);
  } catch (error) {

```

```

    reply.status(500).send({ error: "Category creation failed" });
  }
});

// Options for PUT /categories/:id
const updateCategoryOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Category"],
    summary: "Update a category",
    params: {
      type: "object",
      properties: {
        id: { type: "number" },
      },
      required: ["id"],
    },
    body: {
      type: "object",
      properties: {
        name: { type: "string" },
      },
    },
    response: {
      200: Category,
      500: {
        description: "Category update failed",
        type: "object",
        properties: {
          error: { type: "string" },
        },
      },
    },
  },
};

// Update a category
server.put("/categories/:id", updateCategoryOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  const { name } = request.body as { name?: string };

  let slug;
  if (name) {
    slug = name
      .toLowerCase()
      .replace(/[\^a-z0-9]+/g, "-")
      .replace(/(\^|-)$+/g, "");
  }

  try {
    const category = await prisma.category.update({
      where: { id: parseInt(id) },
      data: {
        name,
        slug,
      },
    });
    reply.send(category);
  } catch (error) {
    reply.status(500).send({ error: "Category update failed" });
  }
});

// Options for DELETE /categories/:id
const deleteCategoryOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Category"],
    summary: "Delete a category",
    params: {
      type: "object",
      properties: {
        id: { type: "number" },
      },
      required: ["id"],
    },
    response: {

```

```

204: {
  description: "Category deleted successfully",
  type: "null",
},
500: {
  description: "Category deletion failed",
  type: "object",
  properties: {
    error: { type: "string" },
  },
},
},
},
};
// Delete a category
server.delete(
  "/categories/:id",
  deleteCategoryOpts,
  async (request, reply) => {
    const { id } = request.params as { id: string };
    try {
      await prisma.category.delete({
        where: { id: parseInt(id) },
      });
      reply.status(204).send();
    } catch (error) {
      reply.status(500).send({ error: "Category deletion failed" });
    }
  }
);
}

```

Лістинг А3 – Код модулю коментарів

```

import { FastifyInstance, RouteShorthandOptions } from 'fastify';
import { PrismaClient } from '@prisma/client';
const prisma = new PrismaClient();
export async function commentRoutes(server: FastifyInstance) {
  // Comment schema
  const Comment = {
    type: 'object',
    properties: {
      id: { type: 'number' },
      content: { type: 'string' },
      authorId: { type: 'number' },
      articleId: { type: 'number' },
      createdAt: { type: 'string', format: 'date-time' },
      updatedAt: { type: 'string', format: 'date-time' },
      author: {
        type: 'object',
        properties: {
          id: { type: 'number' },
          name: { type: 'string' },
          email: { type: 'string', format: 'email' },
        },
      },
    },
  };
  // Options for GET /articles/:slug/comments
  const getCommentsOpts: RouteShorthandOptions = {
    schema: {
      tags: ['Comment'],
      summary: 'Get comments for an article',
    },
  };
}

```



```

params: {
  type: 'object',
  properties: {
    slug: { type: 'string' },
  },
  required: ['slug'],
},
response: {
  200: {
    description: 'List of comments',
    type: 'array',
    items: Comment,
  },
  404: {
    description: 'Article not found',
    type: 'object',
    properties: {
      error: { type: 'string' },
    },
  },
},
},
};
// Get comments for an article
server.get('/articles/:slug/comments', getCommentsOpts, async (request, reply) => {
  const { slug } = request.params as { slug: string };
  const article = await prisma.article.findUnique({
    where: { slug },
  });

  if (!article) {
    reply.status(404).send({ error: 'Article not found' });
    return;
  }

  const comments = await prisma.comment.findMany({
    where: {
      articleId: article.id,
    },
    include: {
      author: true,
    },
  });
  reply.send(comments);
});
// Options for POST /articles/:slug/comments
const createCommentOpts: RouteShorthandOptions = {
  schema: {
    tags: ['Comment'],
    summary: 'Add a comment to an article',
    params: {
      type: 'object',
      properties: {
        slug: { type: 'string' },
      },
      required: ['slug'],
    },
    body: {

```

```

    type: 'object',
    required: ['authorId', 'content'],
    properties: {
      authorId: { type: 'number' },
      content: { type: 'string' },
    },
  },
  response: {
    201: Comment,
    404: {
      description: 'Article not found',
      type: 'object',
      properties: {
        error: { type: 'string' },
      },
    },
    500: {
      description: 'Comment creation failed',
      type: 'object',
      properties: {
        error: { type: 'string' },
      },
    },
  },
};

// Add a comment to an article
server.post('/articles/:slug/comments', createCommentOpts, async (request, reply) => {
  const { slug } = request.params as { slug: string };
  const { authorId, content } = request.body as {
    authorId: number;
    content: string;
  };

  const article = await prisma.article.findUnique({
    where: { slug },
  });

  if (!article) {
    reply.status(404).send({ error: 'Article not found' });
    return;
  }

  try {
    const comment = await prisma.comment.create({
      data: {
        content,
        authorId,
        articleId: article.id,
      },
      include: {
        author: true,
      },
    });
    reply.status(201).send(comment);
  } catch (error) {
    reply.status(500).send({ error: 'Comment creation failed' });
  }
}

```

```

});
// Options for DELETE /comments/:id
const deleteCommentOpts: RouteShorthandOptions = {
  schema: {
    tags: ['Comment'],
    summary: 'Delete a comment',
    params: {
      type: 'object',
      properties: {
        id: { type: 'number' },
      },
      required: ['id'],
    },
    response: {
      204: {
        description: 'Comment deleted successfully',
        type: 'null',
      },
      500: {
        description: 'Comment deletion failed',
        type: 'object',
        properties: {
          error: { type: 'string' },
        },
      },
    },
  },
};
// Delete a comment
server.delete('/comments/:id', deleteCommentOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  try {
    await prisma.comment.delete({
      where: { id: parseInt(id) },
    });
    reply.status(204).send();
  } catch (error) {
    reply.status(500).send({ error: 'Comment deletion failed' });
  }
});
}

```

Лістинг А4 – Код модулю взаємодії з ШІ

```

// src/routes/openai.ts

import { FastifyInstance, RouteShorthandOptions } from "fastify";
import { OpenAIHelper, GenerateTopicOptions } from "../lib/openaiModule"; // Adjust the import path accordingly
import { PrismaClient } from "@prisma/client";
export async function openAIRoutes(server: FastifyInstance) {
  // Schema for the request and response
  const generateTopicOpts: RouteShorthandOptions = {
    schema: {
      tags: ["OpenAI"],
      summary:
        "Generate a topic in Markdown format based on context and optional data analysis results",
      body: {
        type: "object",
        required: ["context"],
        properties: {
          context: { type: "string" },
        },
      },
    },
  };
}

```

```

    dataAnalysisResult: { type: "string" },
    model: {
      type: "string",
      enum: ["gpt-3.5-turbo", "gpt-4o", "text-davinci-003", "o1-preview"],
      default: "gpt-4o",
    },
  },
},
},
response: {
  200: {
    description: "Successfully generated topic",
    type: "object",
    properties: {
      topic: { type: "string" },
    },
  },
  400: {
    description: "Bad Request",
    type: "object",
    properties: {
      error: { type: "string" },
    },
  },
  500: {
    description: "Internal Server Error",
    type: "object",
    properties: {
      error: { type: "string" },
    },
  },
},
},
};
// POST /api/generate-topic
server.post("/generate-topic", generateTopicOpts, async (request, reply) => {
  const { context, dataAnalysisResult, model } =
    request.body as GenerateTopicOptions;

  if (!context) {
    reply
      .status(400)
      .send({ error: "Context is required to generate a topic." });
    return;
  }

  const openAIHelper = new OpenAIHelper();

  try {
    const topic = await openAIHelper.generateTopic({
      context,
      dataAnalysisResult,
      model,
    });
    reply.send({ topic });
  } catch (error: any) {
    server.log.error("Error generating topic:", error.message);
    reply.status(500).send({ error: "Failed to generate topic." });
  }
});

// Schema for generate and create article
const generateAndCreateArticleOpts: RouteShorthandOptions = {
  schema: {
    tags: ["OpenAI"],
    summary: "Generate a topic and create an article",
    body: {
      type: "object",
      required: ["context", "authorId"],
      properties: {
        context: { type: "string" },
        dataAnalysisResult: { type: "string" },
        model: {
          type: "string",

```

```

    enum: ["gpt-3.5-turbo", "gpt-4o", "text-davinci-003", "o1-preview"],
    default: "gpt-4o",
  },
  authorId: { type: "number" },
  categoryIds: {
    type: "array",
    items: { type: "number" },
  },
  tagIds: {
    type: "array",
    items: { type: "number" },
  },
  published: { type: "boolean" },
},
},
response: {
  201: {
    description: "Successfully generated topic and created article",
    type: "object",
    properties: {
      article: { type: "object" }, // Using the existing Article schema
    },
  },
  400: {
    description: "Bad Request",
    type: "object",
    properties: {
      error: { type: "string" },
    },
  },
  500: {
    description: "Internal Server Error",
    type: "object",
    properties: {
      error: { type: "string" },
    },
  },
},
},
};
// POST /api/generate-and-create-article
server.post(
  "/generate-and-create-article",
  generateAndCreateArticleOpts,
  async (request, reply) => {
    const {
      context,
      dataAnalysisResult,
      model,
      authorId,
      categoryIds,
      tagIds,
      published,
    } = request.body as GenerateTopicOptions & {
      authorId: number;
      categoryIds?: number[];
      tagIds?: number[];
      published?: boolean;
    };

    if (!context) {
      reply
        .status(400)
        .send({ error: "Context is required to generate a topic." });
      return;
    }

    const openAIHelper = new OpenAIHelper();
    const prisma = new PrismaClient();

    try {
      // Generate the topic
      const topic = await openAIHelper.generateTopic({

```

```

    context,
    dataAnalysisResult,
    model,
  });

  // Create the article using the generated topic
  const slug = topic
    .split("\n")[0] // Get the first line as title
    .toLowerCase()
    .replace(/[\^a-z0-9]+/g, "-")
    .replace(/(\^|-)+/g, "");

  const article = await prisma.article.create({
    data: {
      title: topic.split("\n")[0],
      slug,
      content: topic,
      published: published || false,
      authorId,
      categories: categoryIds
        ? {
            connect: categoryIds.map((id) => ({ id })),
          }
        : undefined,
      tags: tagIds
        ? {
            connect: tagIds.map((id) => ({ id })),
          }
        : undefined,
    },
    include: {
      author: true,
      categories: true,
      tags: true,
      comments: true,
    },
  });

  reply.status(201).send({ article });
} catch (error: any) {
  server.log.error("Error generating topic and creating article:", JSON.stringify(error));
  reply.status(500).send({ error: "Failed to generate topic and create article." });
}
}
);
}

```

Лістинг А5 – Код допоміжних функцій для взаємодії з ШІ

```

import OpenAI from "openai";

export interface GenerateTopicOptions {
  context: string;
  dataAnalysisResult?: string;
  model?: string; // e.g., 'gpt-3.5-turbo', 'gpt-4', 'text-davinci-003', 'gpt-4'
}

export class OpenAIHelper {
  private openai: OpenAI;

  constructor(apiKey?: string) {
    this.openai = new OpenAI({
      apiKey: apiKey || process.env.OPENAI_API_KEY,
    });
  }

  async generateTopic(options: GenerateTopicOptions): Promise<string> {
    const { context, dataAnalysisResult, model } = options;

    if (!context) {
      throw new Error("Context is required to generate a topic.");
    }
  }
}

```

```

const prompt = this.buildPrompt(context, dataAnalysisResult);

try {
  if (model?.startsWith("o1-preview")) {
    console.log("Using o1-preview");
    // Use o1-preview
    const response = await this.openai.chat.completions.create({
      model: 'o1-preview',
      messages: [
        {
          role: "user",
          content:
            "You are an assistant that generates safe topics in Markdown format, strictly about economic, business, or
finance subjects.",
        },
        { role: "user", content: prompt },
      ],
      max_completion_tokens: 30000,
      // temperature: 0.7,
      n: 3,
    });

    const topic = response.choices[0].message!.content!.trim();
    if (!topic) {
      throw new Error("No topic was generated by the model.");
    }

    return topic;
  } else if (model?.startsWith("gpt-")) {
    console.log("Using gpt-");
    const response = await this.openai.chat.completions.create({
      model: model || "gpt-4o",
      messages: [
        {
          role: "system",
          content:
            "You are an assistant that generates safe topics in Markdown format, strictly about economic, business, or
finance subjects.",
        },
        { role: "system", content: prompt },
      ],
      max_completion_tokens: 15000,
      temperature: 0.7,
      n: 3,
    });

    const topic = response.choices[0].message!.content!.trim();
    if (!topic) {
      throw new Error("No topic was generated by the model.");
    }

    return topic;
  } else {
    throw new Error("Invalid model specified.");
  }
} catch (error: any) {
  console.error("Error generating topic:", error.message);
  throw error;
}
}

private buildPrompt(context: string, dataAnalysisResult?: string): string {
  let prompt = `Based on the following context, generate a concise and relevant article in Markdown format, strictly
related to economic, business, or finance subjects. Ensure the content is safe and appropriate. First line should be a topic
of article. Do not rewrite tables. Focus on interpretation of data.

Context:
${context}`;

  if (dataAnalysisResult) {
    prompt += `

```

```
Data Analysis Result:
${dataAnalysisResult}`;
}
```

```
prompt += `
```

```
Topic (in Markdown):`;
```

```
return prompt;
}
}
```

Лістинг А6 – Код модулю тегів

```
import { FastifyInstance, RouteShorthandOptions } from "fastify";
import { PrismaClient } from "@prisma/client";
const prisma = new PrismaClient();
export async function tagRoutes(server: FastifyInstance) {
  // Tag schema
  const Tag = {
    type: "object",
    properties: {
      id: { type: "number" },
      name: { type: "string" },
      slug: { type: "string" },
      articles: {
        type: "array",
        items: {
          type: "object",
          properties: {
            id: { type: "number" },
            title: { type: "string" },
            slug: { type: "string" },
          },
        },
      },
    },
  },
  },
  },
  },
};
// Options for GET /tags
const getTagsOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Tag"],
    summary: "Get all tags",
    response: {
      200: {
        description: "List of tags",
        type: "array",
        items: Tag,
      },
    },
  },
};
// Get all tags
server.get("/tags", getTagsOpts, async (request, reply) => {
  const tags = await prisma.tag.findMany({
    include: { articles: true },
  });
  reply.send(tags);
});
// Options for GET /tags/:id
const getTagOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Tag"],
    summary: "Get a tag by ID",
    params: {
      type: "object",
      properties: {
        id: { type: "number" },
      },
    },
    required: ["id"],
  },
};
};
```



```

    },
    response: {
      200: Tag,
      404: {
        description: "Tag not found",
        type: "object",
        properties: {
          error: { type: "string" },
        },
      },
    },
  },
};

// Get a single tag by ID
server.get("/tags/:id", getTagOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  const tag = await prisma.tag.findUnique({
    where: { id: parseInt(id) },
    include: { articles: true },
  });
  if (tag) {
    reply.send(tag);
  } else {
    reply.status(404).send({ error: "Tag not found" });
  }
});

// Options for POST /tags
const createTagOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Tag"],
    summary: "Create a new tag",
    body: {
      type: "object",
      required: ["name"],
      properties: {
        name: { type: "string" },
      },
    },
  },
  response: {
    201: Tag,
    500: {
      description: "Tag creation failed",
      type: "object",
      properties: {
        error: { type: "string" },
      },
    },
  },
};

// Create a new tag
server.post("/tags", createTagOpts, async (request, reply) => {
  const { name } = request.body as { name: string };
  // Generate a slug from the name
  const slug = name
    .toLowerCase()
    .replace(/[\^a-z0-9]+/g, "-")
    .replace(/(\^|-|$)+/g, "");

  try {
    const tag = await prisma.tag.create({
      data: {
        name,
        slug,
      },
    });
    reply.status(201).send(tag);
  } catch (error) {
    reply.status(500).send({ error: "Tag creation failed" });
  }
});

// Options for PUT /tags/:id
const updateTagOpts: RouteShorthandOptions = {

```

```

schema: {
  tags: ["Tag"],
  summary: "Update a tag",
  params: {
    type: "object",
    properties: {
      id: { type: "number" },
    },
    required: ["id"],
  },
  body: {
    type: "object",
    properties: {
      name: { type: "string" },
    },
  },
  response: {
    200: Tag,
    500: {
      description: "Tag update failed",
      type: "object",
      properties: {
        error: { type: "string" },
      },
    },
  },
};
// Update a tag
server.put("/tags/:id", updateTagOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  const { name } = request.body as { name?: string };

  let slug;
  if (name) {
    slug = name
      .toLowerCase()
      .replace(/[\^a-z0-9]+/g, "-")
      .replace(/(\^|-|$)+/g, "");
  }

  try {
    const tag = await prisma.tag.update({
      where: { id: parseInt(id) },
      data: {
        name,
        slug,
      },
    });
    reply.send(tag);
  } catch (error) {
    reply.status(500).send({ error: "Tag update failed" });
  }
});
// Options for DELETE /tags/:id
const deleteTagOpts: RouteShorthandOptions = {
  schema: {
    tags: ["Tag"],
    summary: "Delete a tag",
    params: {
      type: "object",
      properties: {
        id: { type: "number" },
      },
      required: ["id"],
    },
    response: {
      204: {
        description: "Tag deleted successfully",
        type: "null",
      },
      500: {
        description: "Tag deletion failed",

```

```

        type: "object",
        properties: {
          error: { type: "string" },
        },
      },
    },
  },
};
// Delete a tag
server.delete("/tags/:id", deleteTagOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  try {
    await prisma.tag.delete({
      where: { id: parseInt(id) },
    });
    reply.status(204).send();
  } catch (error) {
    reply.status(500).send({ error: "Tag deletion failed" });
  }
});
}

```

Лістинг А7 – Код модулю користувачів

```

import { FastifyInstance, RouteShorthandOptions } from 'fastify';
import { PrismaClient, Role } from '@prisma/client';
import { hash } from 'bcrypt';
const prisma = new PrismaClient();
export async function userRoutes(server: FastifyInstance) {
  // Schema for the user model
  const User = {
    type: 'object',
    properties: {
      id: { type: 'number' },
      name: { type: 'string' },
      email: { type: 'string' },
      role: { type: 'string', enum: Object.values(Role) },
      createdAt: { type: 'string', format: 'date-time' },
      updatedAt: { type: 'string', format: 'date-time' },
    },
  };
  // Options for GET /users
  const getUsersOpts: RouteShorthandOptions = {
    schema: {
      tags: ['User'],
      summary: 'Get all users',
      response: {
        200: {
          description: 'Successful response',
          type: 'array',
          items: User,
        },
      },
    },
  };
  // Get all users
  server.get('/users', getUsersOpts, async (request, reply) => {
    const users = await prisma.user.findMany();
    reply.send(users);
  });
}

```

```

});
// Options for GET /users/:id
const getUserOpts: RouteShorthandOptions = {
  schema: {
    tags: ['User'],
    summary: 'Get a user by ID',
    params: {
      type: 'object',
      properties: {
        id: { type: 'number' },
      },
      required: ['id'],
    },
    response: {
      200: User,
      404: {
        description: 'User not found',
        type: 'object',
        properties: {
          error: { type: 'string' },
        },
      },
    },
  },
};

// Get a single user by ID
server.get('/users/:id', getUserOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  const user = await prisma.user.findUnique({
    where: { id: parseInt(id) },
  });
  if (user) {
    reply.send(user);
  } else {
    reply.status(404).send({ error: 'User not found' });
  }
});

// Options for POST /users
const createUserOpts: RouteShorthandOptions = {
  schema: {
    tags: ['User'],
    summary: 'Create a new user',
    body: {
      type: 'object',
      required: ['name', 'email', 'password'],
      properties: {
        name: { type: 'string' },
        email: { type: 'string', format: 'email' },
        password: { type: 'string' },
        role: { type: 'string', enum: Object.values(Role) },
      },
    },
  },
};

```

```

response: {
  201: User,
  500: {
    description: 'User creation failed',
    type: 'object',
    properties: {
      error: { type: 'string' },
    },
  },
},
};

// Create a new user
server.post('/users', createUserOpts, async (request, reply) => {
  const { name, email, password, role } = request.body as {
    name: string;
    email: string;
    password: string;
    role?: Role;
  };

  const passwordHash = await hash(password, 10);

  try {
    const user = await prisma.user.create({
      data: {
        name,
        email,
        passwordHash,
        role: role || Role.AUTHOR,
      },
    });
    reply.status(201).send(user);
  } catch (error) {
    reply.status(500).send({ error: 'User creation failed' });
  }
});

// Options for PUT /users/:id
const updateUserOpts: RouteShorthandOptions = {
  schema: {
    tags: ['User'],
    summary: 'Update a user',
    params: {
      type: 'object',
      properties: {
        id: { type: 'number' },
      },
      required: ['id'],
    },
    body: {
      type: 'object',
      properties: {

```

```

    name: { type: 'string' },
    email: { type: 'string', format: 'email' },
    role: { type: 'string', enum: Object.values(Role) },
  },
},
response: {
  200: User,
  500: {
    description: 'User update failed',
    type: 'object',
    properties: {
      error: { type: 'string' },
    },
  },
},
},
};

// Update a user
server.put('/users/:id', updateUserOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  const { name, email, role } = request.body as {
    name?: string;
    email?: string;
    role?: Role;
  };

  try {
    const user = await prisma.user.update({
      where: { id: parseInt(id) },
      data: { name, email, role },
    });
    reply.send(user);
  } catch (error) {
    reply.status(500).send({ error: 'User update failed' });
  }
});

// Options for DELETE /users/:id
const deleteUserOpts: RouteShorthandOptions = {
  schema: {
    tags: ['User'],
    summary: 'Delete a user',
    params: {
      type: 'object',
      properties: {
        id: { type: 'number' },
      },
      required: ['id'],
    },
  },
  response: {
    204: {
      description: 'User deleted successfully',
      type: 'null',
    },
  },
};

```

```

    },
    500: {
      description: 'User deletion failed',
      type: 'object',
      properties: {
        error: { type: 'string' },
      },
    },
  },
},
};

// Delete a user
server.delete('/users/:id', deleteUserOpts, async (request, reply) => {
  const { id } = request.params as { id: string };
  try {
    await prisma.user.delete({
      where: { id: parseInt(id) },
    });
    reply.status(204).send();
  } catch (error) {
    reply.status(500).send({ error: 'User deletion failed' });
  }
});
}

```

Лістинг А8 – Код серверу

```

import Fastify from "fastify";
import { PrismaClient } from "@prisma/client";
import { userRoutes } from "./routes/users";
import { articleRoutes } from "./routes/articles";
import { categoryRoutes } from "./routes/categories";
import { tagRoutes } from "./routes/tags";
import { commentRoutes } from "./routes/comments";
import swagger from "@fastify/swagger";
import swaggerUi from "@fastify/swagger-ui";
import { openAIRoutes } from "./routes/openai";
const server = Fastify({ logger: true });
const prisma = new PrismaClient();

// Register Swagger
server.register(swagger, {
  swagger: {
    info: {
      title: "CMS API",
      description: "API documentation for the CMS",
      version: "1.0.0",
    },
  },
  host: "localhost:3000",
  schemes: ["http"],
  consumes: ["application/json"],
  produces: ["application/json"],
  tags: [
    { name: "User", description: "User related end-points" },
  ],
});

```

```

    { name: "Article", description: "Article related end-points" },
    { name: "Category", description: "Category related end-points" },
    { name: "Tag", description: "Tag related end-points" },
    { name: "Comment", description: "Comment related end-points" },
    { name: "OpenAI", description: "OpenAI related end-points" },
  ],
},
});
// Register Swagger UI
server.register(swaggerUi, {
  routePrefix: "/documentation",
  uiConfig: {
    docExpansion: "list",
    deepLinking: false,
  },
  uiHooks: {
    onRequest: function (request, reply, next) {
      next();
    },
    preHandler: function (request, reply, next) {
      next();
    },
  },
  staticCSP: true,
  transformStaticCSP: (header) => header,
});
// Register routes
server.register(userRoutes, { prefix: "/api" });
server.register(articleRoutes, { prefix: "/api" });
server.register(categoryRoutes, { prefix: "/api" });
server.register(tagRoutes, { prefix: "/api" });
server.register(commentRoutes, { prefix: "/api" });
server.register(openAIRoutes, { prefix: "/api" });
// Start the server
const start = async () => {
  try {
    await server.listen({ port: 3000 });
    console.log("Server is running at http://localhost:3000");
    console.log(
      "Documentation available at http://localhost:3000/documentation"
    );
  } catch (err) {
    server.log.error(err);
    process.exit(1);
  }
};
start()

```