Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

ІНТЕГРАЦІЯ ТА ДОСЛІДЖЕННЯ API NOVITA AI ДЛЯ СТВОРЕННЯ ТА НАВЧАННЯ КОРИСТУВАЦЬКИХ МОДЕЛЕЙ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

INTEGRATION AND RESEARCH OF THE NOVITA AI API FOR CREATING AND TRAINING CUSTOM IMAGE GENERATION MODELS

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗм-21
Качула І. М.
Керівник:
к.т.н., доцент
Суринович О. М.

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент Суринович О. М.
_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *магістр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«___» _____ 202__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ
Качулі Івану Миколайовичу

1. Тема кваліфікаційної роботи: Інтеграція та дослідження API Novita AI для створення та навчання користувацьких моделей генерації зображень
Керівник роботи: Суринович Олена Миколаївна, доцент, к.т.н.
затверджені наказом закладу вищої освіти від «29» березня 2025 року № 190/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: 04 грудня 2025 р.
3. Вихідні дані до роботи *технічне та програмне забезпечення ЕОМ*
4. Зміст розрахунково-пояснювальної записки: *аналіз сучасних технологій генерації зображень та методів навчання користувацьких моделей штучного інтелекту; обґрунтування вибору сервісу Novita AI та засобів розробки вебзастосунку; проєктування архітектури системи та реалізація вебзастосунку з підтримкою генерації зображень і створення власних моделей;оцінка продуктивності, стабільності та якості інтеграції з API Novita AI*
5. Перелік графічного матеріалу 26 рисунків, 4 таблиці, 4 лістинги коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Суринович О. М.*		
Теоретичне дослідження та практична реалізація	*Суринович О. М.*		
Експериментальне дослідження системи	*Суринович О. М.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Андрущак І. Є.*		
Показник запозичень тексту	___%		
Академічна доброчесність	*Суринович О. М.*		

7. Дата видачі завдання *«02» квітня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	Провести огляд літературних джерел по темі кваліфікаційної роботи	02.05.2025	
2	Провести аналіз загальної проблеми і вибір напрямків дослідження	24.09.2025	
3	Розробити функціональну модель та архітектуру системи	01.11.2025	
4	Описати засоби розробки об’єкта проектування	19.11.2025	
5	Практична реалізація об’єкта проектування	26.11.2025	
6	Розробити методику для проведення експерименту	05.11.2025	
7	Провести аналіз результатів експерименту	15.11.2025	
8	Здача чистового варіанту кваліфікаційної роботи на кафедру	04.12.2025	

Здобувач вищої освіти _____ Качула І. М.

Керівник кваліфікаційної роботи _____ Суринович О. М.

АНОТАЦІЯ

Качула І. М. Інтеграція та дослідження API Novita AI для створення та навчання користувацьких моделей генерації зображень. Рукопис.

Кваліфікаційна робота магістра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі здійснено аналіз сучасного стану проблеми вебзастосунків із можливістю навчання власних моделей штучного інтелекту генерування зображень, а також сформульовано завдання. В другому розділі визначено вимоги до розроблюваної системи, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено вебзастосунок із можливістю генерування зображень і навчання власних моделей. У висновках узагальнено інформацію, відображену у попередніх частинах. Результати розробки демонструють можливості ефективної взаємодії вебзастосунку з сервісом Novita AI, забезпечуючи повний цикл генерації зображень, створення користувацьких моделей та їх подальшого використання.

Ключові слова: генерація зображень, штучний інтелект, користувацькі моделі, Novita AI, вебзастосунок, Laravel, Vue.js, Nuxt, REST API, асинхронна обробка, веб-сокети.

ABSTRACT

Kachula I. M. Integration and Research of the Novita AI API for Creating and Training Custom Image Generation Models. Manuscript.

Master's Qualifying Thesis of the Educational Program «Software Engineering». Lutsk National Technical University. Lutsk, 2025.

The Master's Thesis Consists of an Introduction, Three Chapters, Conclusions, a List of References, and Appendices.

The First Chapter Presents an Analysis of the Current State of Web Applications with the Capability to Train Custom Artificial Intelligence Models for Image Generation and Formulates the Research Tasks. The Second Chapter Defines the Requirements for the Developed System, as Well as the Tools, Methods, and Algorithms Used in Its Development. In the Third Chapter, a Web Application with Image Generation and Custom Model Training Capabilities Is Developed. The Conclusions Summarize the Information Presented in the Previous Sections. The Results of the Development Demonstrate the Effective Interaction of the Web Application with the Novita AI Service, Providing a Full Cycle of Image Generation, Creation of Custom Models, and Their Further Use.

Keywords: Image Generation, Artificial Intelligence, Custom Models, Novita AI, Web Application, Laravel, Vue.js, Nuxt, REST API, Asynchronous Processing, WebSocket.

ЗМІСТ

ВСТУП

Актуальність теми. Стрімкий розвиток технологій штучного інтелекту призвів до появи нових підходів у сфері автоматизованої генерації графічних зображень. Сучасні моделі комп'ютерного зору, зокрема дифузійні нейронні мережі, здатні створювати високоякісні зображення за текстовими описами, що значно спрощує роботу дизайнерів, маркетологів та розробників цифрового контенту. Проте практичні задачі бізнесу часто потребують специфічних рішень, адаптованих до конкретного бренду, стилю або корпоративних вимог. Стандартні моделі мають обмеження щодо точності передачі фірмової стилістики, кольорової палітри та дрібних деталей.

У світовій практиці спостерігається тенденція до активного використання та тренування власних (кастомних) моделей генерації зображень, які дозволяють досягти значно точніших і стабільніших результатів у порівнянні зі стандартними моделями. Компанії впроваджують такі моделі для автоматизації створення рекламних матеріалів, адаптивного дизайну та будь-які інші специфічні задачі.

Мета роботи – розробити та дослідити навчання користувацької моделі генерації зображень на основі штучного інтелекту, здатну відтворювати графічні матеріали з високою точністю та стабільністю.

Для досягнення поставленої мети необхідно вирішити такі завдання:

— сформувати навчальний набір даних, що містить корпоративні стилі, логотипи та інші візуальні елементи бренду;

— провести тренування користувацької моделі на основі обраної базової моделі;

— виконати експериментальне порівняння стандартної та кастомної моделей при однакових умовах генерації;

— проаналізувати переваги й недоліки використання користувацької моделі;

— дослідити можливості подальшої автоматизованої обробки згенерованих зображень (видалення тексту, видалення фону, підвищення роздільної здатності);

— оцінити практичні аспекти впровадження моделі у процеси компанії.

Об'єкт дослідження – процеси генерації та обробки зображень за допомогою ШІ-моделей.

Предмет дослідження – методи та інструменти тренування користувацьких моделей генерації зображень, а також ефективність їх застосування у порівнянні зі стандартними моделями.

Новизна роботи полягає у створенні та експериментальному обґрунтуванні підходу до використання спеціально натренованої моделі, яка здатна точно відтворювати унікальні візуальні характеристики бренду та демонструє стабільніші результати генерації у порівнянні з стандартними моделями.

Практична цінність полягає у можливості впровадження розробленої моделі в реальні бізнес-процеси: автоматизацію створення брендового контенту, підготовку маркетингових матеріалів, дизайн референсів та будь-яких графічних елементів без участі спеціалістів-дизайнерів. Запропонований підхід дозволяє суттєво скоротити час та витрати на створення якісних візуальних матеріалів.

Апробація результатів дослідження. Качула І. М., Суринович О. М. Веб-платформа електронної комерції з інтеграцією фіскалізації та податкової звітності. Тези доповідей X Міжнародної науково-практичної конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025) (23-24 травня 2025 року). Луцьк: ЛНТУ, 2025. С. 332-335 [1].

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМАТИКИ ЗА ТЕМОЮ РОБОТИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд і аналіз предметної області проблеми, результатів існуючих теоретичних та експериментальних досліджень

Сьогодні штучний інтелект активно використовується не лише для обробки даних чи розпізнавання об'єктів, а й для створення нових зображень. Завдяки розвитку нейронних мереж комп'ютер може згенерувати картинку з нуля, наприклад, за коротким текстовим описом користувача.

Першими популярними підходами у цій сфері стали генеративно-змагальні мережі (GAN). Вони складаються з двох частин – одна створює зображення, а інша перевіряє, наскільки вони схожі на справжні (рис 1.1). З часом така система навчається створювати дуже реалістичні картинки. Цей метод дав початок першим моделям для генерації зображення.

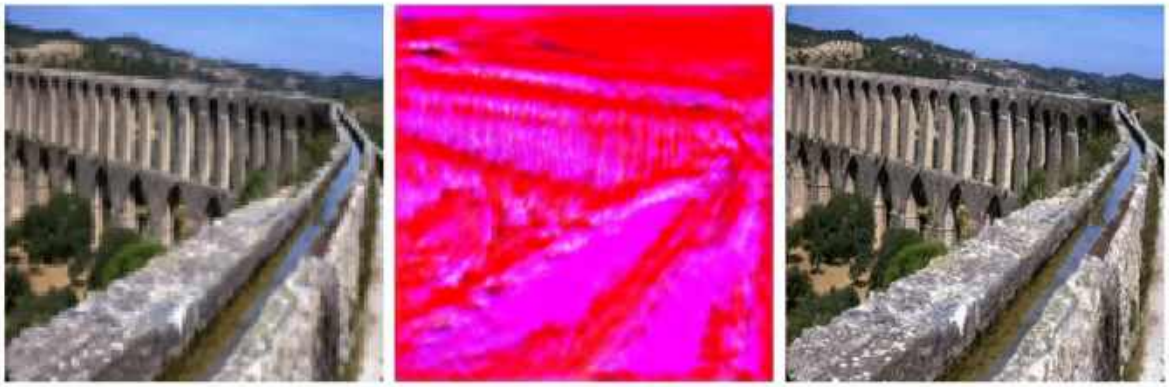


Рисунок 1.1 – Генерування картинки GAN підходом [2]

Згодом з'явилися більш сучасні технології – дифузійні моделі (Diffusion Models). Вони працюють за іншим принципом: спочатку модель створює випадкове «зашумлене» зображення, а потім поступово прибирає шум, поки не отримає готову картинку. Так працюють популярні сервіси Midjourney, DALLE 2 та інші. Ці системи здатні створювати детальні, реалістичні та навіть художні

зображення за коротким текстовим запитом. На рисунку 1.2 зображено послідовне генерування картинки дифузійною моделлю.

Рисунок 1.2 – Генерування картинки дифузійною моделлю [3]

Також у сучасних рішеннях використовуються моделі на базі трансформерів, які краще розуміють текст і контекст [4]. Це допомагає отримувати більш точні результати, коли користувач описує складні сцени або образи. На рисунку 1.3 зображення принцип генерування зображень на базі трансформерів.

Рисунок 1.3 – Генерування зображень моделю на базі трасформерів [5]

Технології генерації зображень сьогодні мають багато практичних застосувань:

- створення дизайнів і банерів;
- генерація ілюстрацій для сайтів;
- маркетинг і реклама;
- освіта, навчальні матеріали;
- творчі проєкти.

Серед найвідоміших сервісів для генерації зображень можна назвати OpenAI (DALLE), Stability AI (Stable Diffusion), Midjourney, Hugging Face, Leonardo AI та Novita AI. Ці платформи надають готові інструменти або API, через які розробники можуть інтегрувати генерацію зображень у свої вебзастосунки.

Сучасні сервіси генерації зображень дають змогу не лише користуватися вже готовими моделями, а й створювати власні. Це особливо корисно, коли потрібно отримати результати у певному стилі або з урахуванням специфічних об'єктів, яких немає у стандартних наборах даних. Наприклад, якщо компанія хоче генерувати зображення власного бренду чи продукту, їй потрібна модель, навчена саме на цих прикладах.

Процес створення такої моделі починається зі збору навчальних даних. Необхідно підготувати зображення, які відповідають бажаному стилю або темі. Усі зображення проходять попередню обробку – вирівнюються за розміром, якістю, форматом, а також часто доповнюються короткими описами, щоб модель могла краще розуміти, що на них зображено.

Далі відбувається навчання, яке зазвичай не починають з нуля, а проводять на основі вже існуючої великої моделі. Такий підхід називають донавчанням. Це дозволяє зберегти основні знання базової моделі, але доповнити їх новими прикладами, які роблять результати більш точними та відповідними до конкретного завдання.

Підготовка та структуризація навчальних даних відіграє ключову роль у процесі навчання моделі. Чим якісніше і точніше підібрані зображення та їх

описи, тим краще модель розпізнає об'єкти та стилі під час генерації. Важливо не лише мати достатню кількість прикладів, а й забезпечити їх різноманітність, щоб модель могла коректно працювати з новими ситуаціями та завданнями.

Окрім класичного донавчання, сьогодні існують більш легкі методи, які потребують менше ресурсів. Наприклад, технологія «LoRA» дозволяє швидко навчити модель на нових даних без повного перенавчання. Схема навчання зображена на рисунку 1.4.

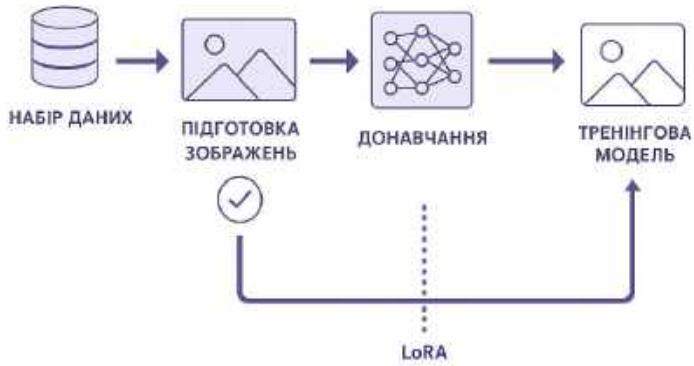


Рисунок 1.4 – Схема тренування генеративної моделі

Після навчання модель можна інтегрувати у вебзастосунок через API, що дозволяє користувачам отримувати результати генерації в реальному часі. Такий підхід робить систему гнучкою. Користувачі можуть змінювати параметри, стиль або тематику зображення, а модель підлаштовується під ці вимоги. Це особливо важливо для проєктів, де потрібно швидко отримувати унікальний контент під різні завдання.

Крім того, сучасні сервіси підтримують асинхронну роботу, що дозволяє обробляти запити паралельно і не блокувати інтерфейс додатку. Завдяки цьому навіть складні запити на генерацію великої кількості зображень виконуються ефективно та без затримок, що підвищує зручність користування системою та робить її придатною для інтеграції у комерційні вебзастосунки.

Завдяки таким технологіям створення власних моделей стало доступним навіть для користувачів без глибоких технічних знань. Сервіси, як-от Novita AI,

«Hugging Face» або «Replicate», надають прості інтерфейси для навчання моделей у хмарі. Розробнику достатньо завантажити зображення, вказати параметри, і система самостійно виконує процес навчання.

Багато сервісів генерації зображень надають можливість використовувати свої моделі через API, що дозволяє інтегрувати генерацію зображень безпосередньо у вебзастосунок чи програми. Такий підхід зручний тим, що не потрібно витрачати ресурси на навчання і підтримку власної моделі – достатньо робити запити до сервісу та отримувати готові результати.

Різні платформи пропонують схожі можливості, але мають свої особливості. Наприклад, OpenAI надає простий API для генерації зображень за текстовим описом, при цьому користувач може обмежувати розмір та стиль зображення. Stability AI пропонує більш гнучкий підхід, де можна як використовувати готові моделі, так і інтегрувати власні, що були навчанні на спеціальних наборах даних. Hugging Face надає цілий каталог моделей з відкритим кодом, що дозволяє підключати різні генеративні алгоритми до додатків. Сервіси на кшталт Midjourney і Leonardo AI більше орієнтовані на швидку генерацію креативних зображень через готові інтерфейси, але також пропонують API для інтеграції.

Особливу увагу варто приділити Novita AI, який поєднує можливість використання готових моделей та створення власних (рис. 1.5). API цього сервісу дозволяє не лише робити запити на генерацію зображень, а й керувати навчанням користувацьких моделей, налаштовувати їх параметри та отримувати результат у зручному форматі. Це робить інтеграцію більш гнучкою і дозволяє будувати власні рішення, що відповідають конкретним потребам користувачів або бізнесу.

Крім того, Novita AI вирізняється продуманою архітектурою API, що значно спрощує інтеграцію в сучасні вебзастосунки. Сервіс підтримує асинхронну обробку запитів, вебхуки та надає детальну інформацію про статус виконання задач, що особливо важливо під час роботи з великою кількістю генерацій або навчальних процесів.

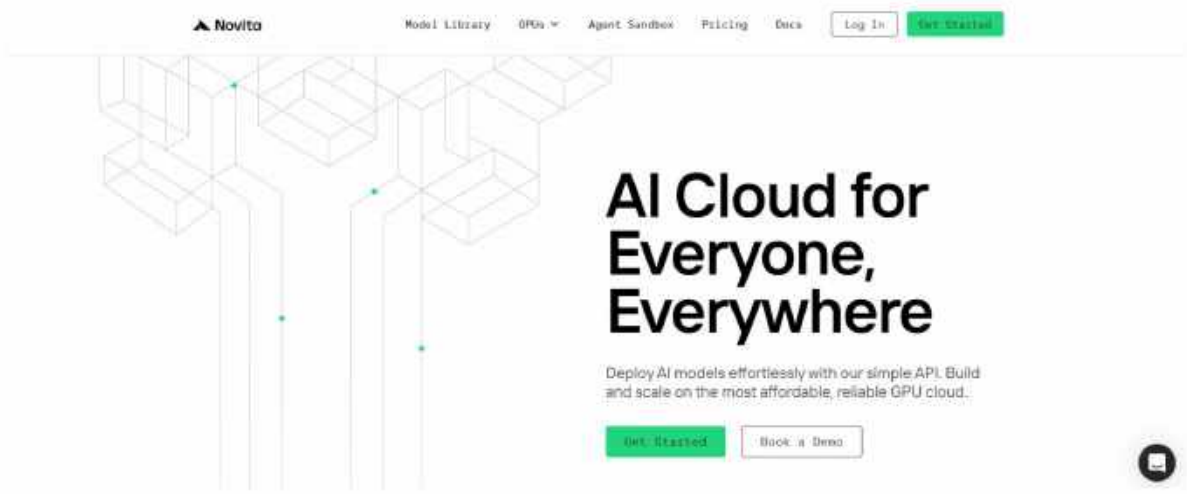


Рисунок 1.5 – Користувацький інтерфейс сервісу «Novita AI» [6]

Загалом інтеграція через API значно спрощує роботу з генеративними моделями. Вона дозволяє швидко додавати функції створення зображень у додатки, скорочує час розробки та відкриває нові можливості для персоналізації контенту. При виборі платформи важливо оцінювати доступні функції, простоту використання, можливості навчання власних моделей і вартість послуг, щоб інтеграція була ефективною і зручною для конкретного проєкту.

1.2 Огляд і аналіз методів та засобів розробки програмного забезпечення для інтеграції API Novita AI та створення користувацьких моделей генерації зображень

Поставлену задачу можна вирішити за допомогою різних видів програмного забезпечення. Це може бути, як десктопний застосунок, який встановлюється на комп'ютер, так і мобільний застосунок. Найзручнішим підходом для наших потреб є вебзастосунок. Такий додаток не потребує встановлення на пристрій, працює у браузері й дозволяє легко користуватися розробкою багатьом користувачам одночасно. Крім того, веб-застосунок дає змогу швидко підключати зовнішні сервіси та працювати з різними моделями генерації.

Проте сам вебзастосунок можна реалізувати монолітною, або мікросервісною архітектурою. У монолітному підході весь функціонал знаходиться в одному проєкті, що спрощує розгортання та розробку на початкових етапах, але може ускладнювати масштабування та розвиток системи з часом і додаванням нового функціоналу. Мікросервісна архітектура передбачає розділення додатку на окремі сервіси. У нашому випадку розділення відбудеться по бекенду на Laravel та фронтенду на Vue.js з Nuxt. Такий підхід дозволяє незалежно розробляти і підтримувати серверну і клієнтську частини.

Під час створення проєкту важливо, щоб система була простою в підтримці та могла виконувати декілька задач одночасно (працювати асинхронно, використовувати черги) [7]. Для цього логіку поділяють на окремі частини, щоб кожен модуль відповідав за свою функцію. Генерація зображень може займати більше часу, тому застосунок повинен уміти обробляти довгі операції у фоновому режимі. Це допомагає уникати «зависань» та дозволяє користувачам далі користуватись додатком, без очікувань виконання довгих операцій, таких як: надсилання електронних листів, генерація зображень, навчання моделей, тощо. Також важливо, щоб система могла розширюватися та витримувати більшу кількість запитів, якщо кількість користувачів зросте. Ще однією вимогою є можливість працювати з власними моделями, адже інколи потрібно навчати модель під конкретні задачі.

Ще одним важливим аспектом є вибір способу розгортання вебзастосунка. Застосунок може працювати на звичайному сервері, у контейнерах Docker або в хмарній інфраструктурі. Використання контейнерів спрощує налаштування середовища, адже всі залежності зберігаються всередині нього. Це дозволяє запускати застосунок на будь-якому сервері без повторної конфігурації.

Створення подібного застосунку потребує продуманого підходу до роботи з даними. Не можна допустити втрати, чи витоку даних. Потрібно визначити, де зберігатимуться зображення, результати генерації та інформація про навчання моделей. Це може бути локальне сховище, база даних або хмарні сервіси.

Важливо подбати про те, щоб доступ до цих даних був швидким і надійним, адже користувачі очікують отримувати результати без затримок та надійним захистом.

Важливо також враховувати навантаження, яке може виникати під час активної генерації зображень. Генерація зображень, сам по собі важкий процес, а якщо додатково багато користувачів одночасно надсилатимуть запити, система потребуватиме максимально продуманої структури, щоб працювати стабільно. Для цього можна використовувати черги, асинхронну обробку або масштабування сервера. Такий підхід допомагає зберегти швидкодію застосунку навіть у складних умовах і робить його більш стійким до високих навантажень.

Способи взаємодії з генеративними моделями можуть бути різними. Найчастіше використовують REST API – це простий спосіб надсилати запит на сервер і отримувати результат. У складніших випадках можуть використовувати WebSocket, якщо потрібно отримувати оновлення в режимі реального часу. Також можуть застосовуватись черги, які дозволяють ставити завдання в обробку без навантаження на основний сервер. Всі ці три підходи будуть інтегровані у нашу систему, для забезпечення максимальної продуктивності та швидкодії застосунку.

REST API підходить для швидких запитів і отримання результату після завершення операції, проте не забезпечує миттєвого оновлення статусу завдання. Саме тому, це є один із найпопулярніших концепцій створення API для інтеграцій.

Для роботи з генеративними моделями зручно використовувати API, оскільки воно дозволяє звертатися до сервісу без необхідності встановлювати або навчати складні моделі на власному сервері. В нашому випадку API Novita AI дає змогу як використовувати готові моделі для генерації зображень, так і створювати та навчати власні користувацькі моделі (рис. 1.6).

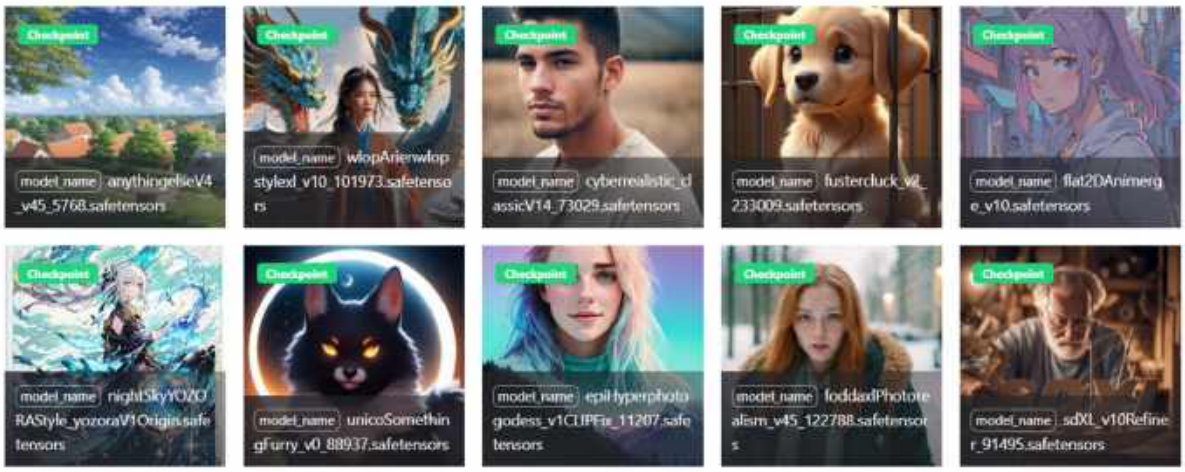


Рисунок 1.6 – Готові моделі для генерації зображень Novita AI [8]

Інтеграція з API Novita AI здійснюється за допомогою REST-запитів. REST – це стандартний спосіб обміну даними між сервером і клієнтським додатком. Коли користувач робить запит на генерацію зображення через інтерфейс вебзастосунки, фронтенд на Vue.js відправляє інформацію на серверну частину. Серверна частина, розроблена на Laravel, отримує запит, перевіряє його коректність і додає необхідні параметри для доступу до сервісу, наприклад ключ API. Потім сервер відправляє запит до Novita AI, яка обробляє його: створює зображення або виконує навчання користувацької моделі. Після завершення обробки сервіс повертає результат на сервер, де Laravel приймає дані, зберігає їх у базі даних, а потім передає назад на фронтенд. Фронтенд отримує ці дані і відображає користувачу готове зображення та статус виконання завдання, забезпечуючи інтерактивність і зручність роботи з додатком.

Для інтеграції важливо правильно організувати кілька процесів. По-перше, це автентифікація – сервер повинен надіслати свій API-ключ, щоб сервіс Novita AI підтвердив доступ. По-друге, необхідно враховувати обробку запитів і помилок, щоб у випадку тимчасових проблем з'єднання або некоректного запиту додаток не «зависав», а показував повідомлення користувачу. По-третє, потрібно організувати обробку результатів генерації,

наприклад, зберігати зображення у базі або на сервері та передавати їх на фронтенд у відповідному форматі.

Сервіс Novita AI підтримує асинхронну інтеграцію, коли запит на генерацію зображення може оброблятися деякий час, а користувач отримує повідомлення, коли результат готовий [9]. Це важливо для великих зображень або складних моделей, де час обробки може займати кілька секунд або хвилин.

Важливо відзначити, що інтеграція через API робить додаток гнучким і масштабованим. Завдяки цьому можна підключати різні моделі, оновлювати їх без змін у коді додатка та легко розширювати функціонал, наприклад, додавати нові типи генерації або параметри для користувацьких моделей.

Існує багато сервісів для генерації зображень за допомогою штучного інтелекту. Вони відрізняються за можливістю навчання власних моделей, інтеграції через API, підтримкою різних форматів даних та налаштувань генерації. Основні сервіси генерації зображень: Novita AI, OpenAI (DALLE), Stable Diffusion, Midjourney, Hugging Face та Leonardo AI [10]. Нижче наведена таблиця 1.1 з основними характеристиками популярних платформ.

Таблиця 1.1 – Характеристиками платформ для генерації зображень

Платформа	Тип інтеграції	Підтримка кастомних моделей	Асинхронна робота	Особливості
OpenAI (DALLE)	REST API	Ні	Так	Висока якість зображень, простота інтеграції
Stable Diffusion	REST API / локально	Так	Так	Можливість локального навчання моделей, гнучкі налаштування
Midjourney	Обмежена API	Ні	Частково	Художні стилі, швидка генерація, обмежена інтеграція
Leonardo AI	Обмежена API	Ні	Ні	Креатив, простота використання
Hugging Face	REST API, Python SDK	Так	Так	Великий каталог відкритих моделей, підтримка кастомізації
Novita AI	REST API	Так	Так	Генерація та навчання власних моделей, зручна інтеграція

Як видно з таблиці 1.1, сервіси відрізняються тим, наскільки легко їх підключати та налаштовувати. Платформи OpenAI та Midjourney добре підходять для швидкої генерації красивих зображень за готовими стилями, але не дозволяють створювати або навчати свої власні моделі. Stable Diffusion та Hugging Face дають більше можливостей, а саме можна змінювати параметри генерації, навчати власні моделі та експериментувати з результатами.

Особливо зручна Novita AI, бо вона легко інтегрується через REST API, підтримує навчання користувацьких моделей, може працювати асинхронно та видає результати у зручному форматі. Тому вона є найкращим вибором для вебзастосунка, де потрібна висока якість зображень і можливість гнучко налаштовувати моделі під конкретні завдання.

Таким чином, проведене порівняння дозволяє обрати платформу, яка найкраще відповідає потребам проєкту, забезпечуючи баланс між швидкістю генерації, гнучкістю налаштувань і можливістю інтеграції у вебзастосунок.

1.3 Постановка завдання на кваліфікаційну роботу магістра

На основі аналізу сучасних технологій генерації зображень та огляду методів і засобів розробки програмного забезпечення для інтеграції API Novita AI, було сформовано основні завдання, які необхідно вирішити в рамках кваліфікаційної роботи магістра.

Метою роботи є розробка вебзастосунку, що дозволяє здійснювати генерацію зображень та створення користувацьких моделей на базі сервісу Novita AI, забезпечуючи зручний інтерфейс, надійну інтеграцію та ефективну обробку запитів.

Для досягнення поставленої мети необхідно виконати такі завдання:

— проаналізувати існуючі технології генерації зображень за допомогою штучного інтелекту, дослідити їх можливості, обмеження та сфери застосування;

— дослідити API сервісу Novita AI, його функціональні можливості, формати запитів та відповіді, підтримку кастомних моделей і асинхронної роботи;

— спроєктувати архітектуру вебзастосунку, що включає побудову схеми бази даних, бекенд на Laravel та фронтенд на Vue.js з Nuxt;

— розробити серверну частину додатку, яка забезпечує інтеграцію RESTful API Novita AI;

— створити користувацький інтерфейс, який інтегрує серверну частину;

— реалізувати підтримку асинхронної генерації зображень, включно з механізмом перевірки готовності результату за допомогою веб-сокетів;

— забезпечити збереження результатів генерації та навчання у базі даних, з можливістю їх подальшого перегляду та повторного використання;

— провести тестування та оцінку роботи системи, включно з оцінкою швидкодії, стабільності, коректності інтеграції API та зручності використання інтерфейсу.

Виконання цих завдань дає змогу створити повноцінний вебзастосунку, який охоплює повний процес роботи з генеративними моделями, від формування запиту до отримання, обробки та збереження результатів. Система забезпечує послідовність дій, необхідних для стабільної та зручної роботи з штучним інтелектом.

Також система надає можливість створювати й навчати власні моделі, які відповідають конкретним потребам користувача. Це дозволяє розв'язувати спеціалізовані задачі та налаштовувати якість генерації відповідно до вимог проєкту, чи конкретного користувача.

Очікуваним результатом виконання кваліфікаційної роботи є створення стабільного вебзастосунку, який дозволяє користувачам ефективно генерувати зображення та навчати власні моделі.

Критеріями успішності роботи є: коректна інтеграція з API Novita AI, швидкодія при обробці запитів, стабільна робота при одночасній генерації декількох зображень, збереження даних у базі та можливість повторного використання результатів. Додатково передбачається реалізація захисту даних користувачів і забезпечення масштабованості системи для роботи з великою кількістю запитів одночасно.

РОЗДІЛ 2
ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНТЕГРАЦІЇ API NOVITA AI ДЛЯ СТВОРЕННЯ ТА НАВЧАННЯ КОРИСТУВАЦЬКИХ МОДЕЛЕЙ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

2.1 Обґрунтування вибору шляхів, технологій, алгоритмів і засобів вирішення поставленого завдання

Для розробки вебзастосунка, який буде працювати з API Novita AI, я обрав Laravel для бекенду та Vue.js з Nuxt для фронтенду. Laravel – це сучасний PHP-фреймворк, який дозволяє швидко створювати серверну частину додатка (рис. 2.1). Він добре підходить для створення REST API, через яке фронтенд зможе відправляти запити і отримувати відповіді від сервера. Його перевага у тому, що фреймворк має готові бібліотеки та компоненти, які спрощують розробку і зменшують кількість помилок.

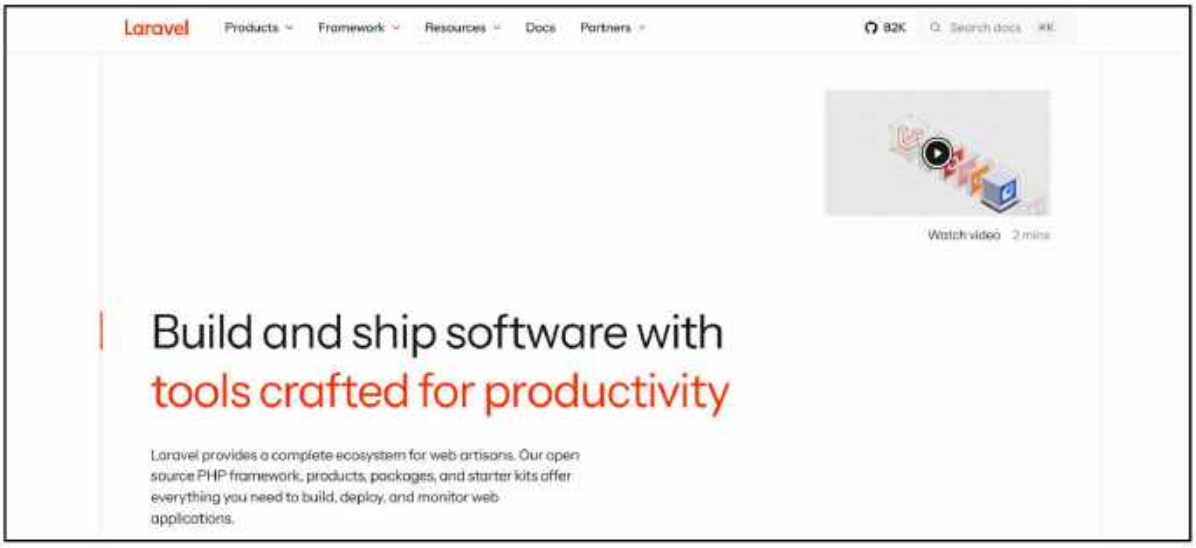


Рисунок 2.1 – Користувацький інтерфейс сервісу Laravel [11]

Для фронтенду використовується Vue.js. Це фреймворк на JavaScript, який дозволяє будувати інтерфейси користувача у вигляді компонентів. Завдяки Vue.js веб-сторінки можуть оновлюватися динамічно, без перезавантаження. Це дуже зручно, коли користувач робить запит на генерацію зображення через API

і хоче одразу побачити результат. Vue.js дозволяє організувати інтерфейс таким чином, щоб всі форми та елементи працювали швидко і логічно (рис. 2.2).

Рисунок 2.2 – Користувацький інтерфейс сервісу Vue.js [12]

Для полегшення розробки фронтенду я використовую Nuxt. Nuxt – це фреймворк, побудований на основі Vue.js, який надає готову структуру проєкту, автоматично налаштовує маршрутизацію сторінок, серверний рендеринг (ssr) і генерацію статичних сторінок, що підвищує швидкодію та SEO вебзастосунка. Він дозволяє легко організувати спільну логіку між компонентами, керувати станом додатку та інтегрувати асинхронні запити до API нашого бекенду.

Nuxt також надає зручні механізми для організації спільної логіки між компонентами, що дозволяє уникнути дублювання коду та дотримуватися принципів DRY. Вбудована система керування станом через Pinia спрощує роботу з глобальними даними, а підтримка асинхронних функцій дає змогу легко інтегрувати фронтенд із REST API бекенду.

Використання Nuxt спрощує підтримку великого фронтенду і робить його більш масштабованим та зручним для розвитку нових функцій. На рисунку 2.3 зображено інтерфейс головної сторінки сервісу Nuxt.

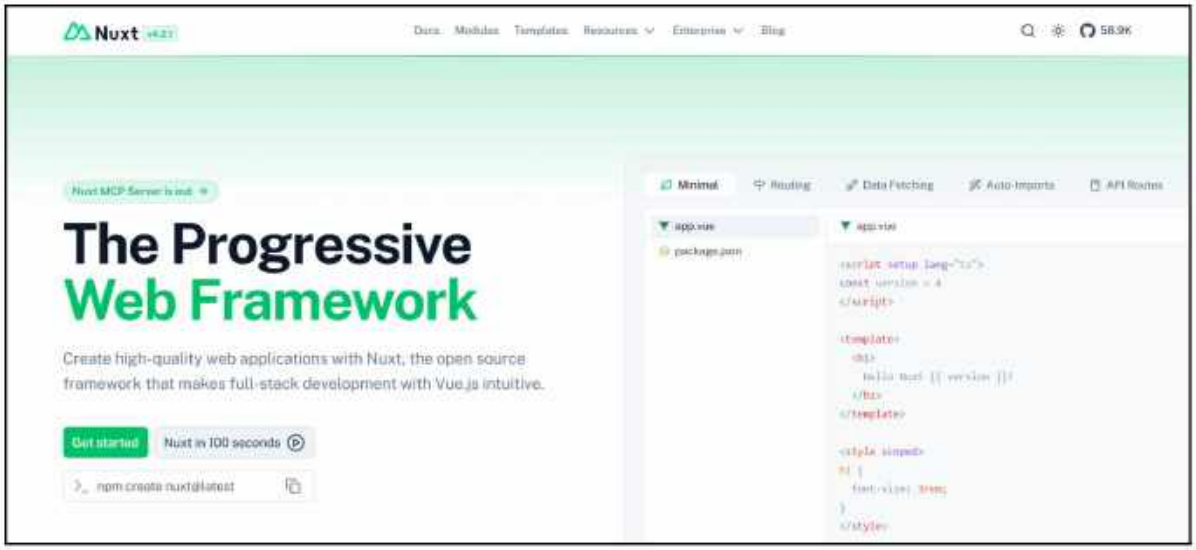


Рисунок 2.3 – Користувацький інтерфейс сервісу Nuxt [13]

Для обміну даними між фронтендом і бекендом використовується REST API, який дозволяє надсилати запити до сервісу Novita AI і отримувати готові зображення або дані про навчання моделей. Це означає, що серверна частина приймає запит, обробляє його, звертається до API Novita AI і повертає результат на фронтенд. Такий підхід робить додаток гнучким і зручним у використанні.

Для забезпечення обміну даними в реальному часі між фронтендом та бекендом використовується технологія WebSocket. Вона дозволяє відкривати постійне з'єднання між сервером і клієнтом. Для цього бекенд повинен створити канал, на якому можна прослуховувати події, що сталися на ньому. За допомогою цієї технології сервер може надсилати повідомлення без потреби повторних запитів від користувача.

У нашому додатку WebSocket застосовується для інформування користувача про завершення генерації зображень або навчання кастомної моделі, що особливо важливо при асинхронній обробці довготривалих завдань. Такий підхід робить взаємодію користувача із системою більш швидкою та інтерактивною, зменшуючи час очікування та покращуючи загальний користувацький досвід.

Для збереження та обробки даних вебзастосунку використовується база даних PostgreSQL. Вона забезпечує надійне зберігання структурованої

інформації, підтримує складні запити, роботу з великими обсягами даних та багато різних типів даних, такі як JSON або масиви. PostgreSQL добре інтегрується з Laravel через ORM Eloquent. Крім того, вона підтримує транзакції, індекси, обмеження цілісності даних і розширені механізми оптимізації запитів, що робить додаток більш стабільним та продуктивним.

На відміну від MySQL, PostgreSQL має більш потужну систему типів даних і розширені можливості роботи з транзакціями та складними запитами. Вона краще підходить для складних проєктів. MySQL, навпаки, зазвичай швидше обробляє прості запити і добре підходить для веб-застосунків із великою кількістю читань, але має обмежені можливості роботи з розширеними типами даних і складними транзакціями. Таким чином, для нашого вебзастосунку, PostgreSQL є більш оптимальним вибором.

Для контролю версій використовується система Git. Git дозволяє відслідковувати всі зміни у файлах проєкту, повертатися до попередніх версій та керувати паралельною розробкою на різних гілках. Для зберігання репозиторію використовується GitHub. Для фронтенду та бекенду має бути створено два окремі репозиторії.

Для розгортання та тестування вебзастосунку використовується Docker – технологія контейнеризації, яка дозволяє ізолювати середовище розробки від операційної системи. Завдяки Docker можна швидко піднімати всі необхідні сервіси (бекенд, база даних, фронтенд) у вигляді контейнерів з однаковими конфігураціями на будь-якому комп'ютері. Це спрощує налаштування середовища, забезпечує повторюваність та зручність у масштабуванні, а також дозволяє уникати конфліктів залежностей між різними проєктами.

Для спрощення роботи з Docker у проєкті використовується Laradock – набір готових контейнерів для Laravel [14]. Laradock дозволяє швидко піднімати всі необхідні сервіси, такі як PHP, Nginx, база даних, Redis та інші, без потреби вручну налаштовувати кожен контейнер і їх взаємодію. Використання Laradock значно скорочує час на налаштування проєкту і зменшує ймовірність конфліктів залежностей між різними сервісами.

На рисунку 2.4 зображено весь стек технологій, який було використано при розробці системи.

Рисунок 2.4 – Стек технологій

У розробці також використовуються сучасні інструменти. Код пишеться у WebStorm та PHPStorm, що дозволяє легко редагувати файли, перевіряти помилки і працювати з проєктом.

Завдяки такому поєднанню технологій можна створити сучасний вебзастосунок, який швидко реагує на дії користувача, стабільно обробляє запити і легко інтегрується з API Novita AI. Це забезпечує зручний інтерфейс для користувача та ефективну роботу серверної частини.

2.2 Практична реалізація об’єкта проектування

Згідно із поставленими цілями в першому розділі, зараз настав час розробити структуру бази даних. Спочатку я визначив усі таблиці, які будуть потрібні для роботи бекенду, виходячи з вимог системи та максимальний упор під майбутню масштабованість. Однак просто визначити таблиці недостатньо, важливо продумати, як саме ці таблиці будуть пов’язані між собою, як вони працюватимуть всі разом, як єдиний механізм.

Для кожної сутності потрібно визначити правильний тип зв'язку. Для одних достатньо відношення «один до багатьох», для інших логічно буде «багато до багатьох», а іншим може знадобиться проміжна таблиця. Це дозволяє уникнути зайвого дублювання інформації, робить структуру даних зрозумілішою та призводить базу даних до нормальних форм [15]. Правильно налаштовані зв'язки забезпечують передбачувану поведінку системи й допомагають будувати запити без додаткових ускладнень.

Структура бази даних є фундаментом усього проєкту. Якщо фундамент буде слабким, система швидко стане складною в підтримці та розширенні, допоки не рухне остаточно. Через це етап проєктування бази даних потребує особливої уваги. Від того, наскільки якісно буде продумана структура на початку, залежить стабільність, продуктивність і масштабованість усього застосунку в майбутньому.

На рисунку 2.5 можна побачити побудовану структуру бази даних із проставленими зв'язками між таблицями.

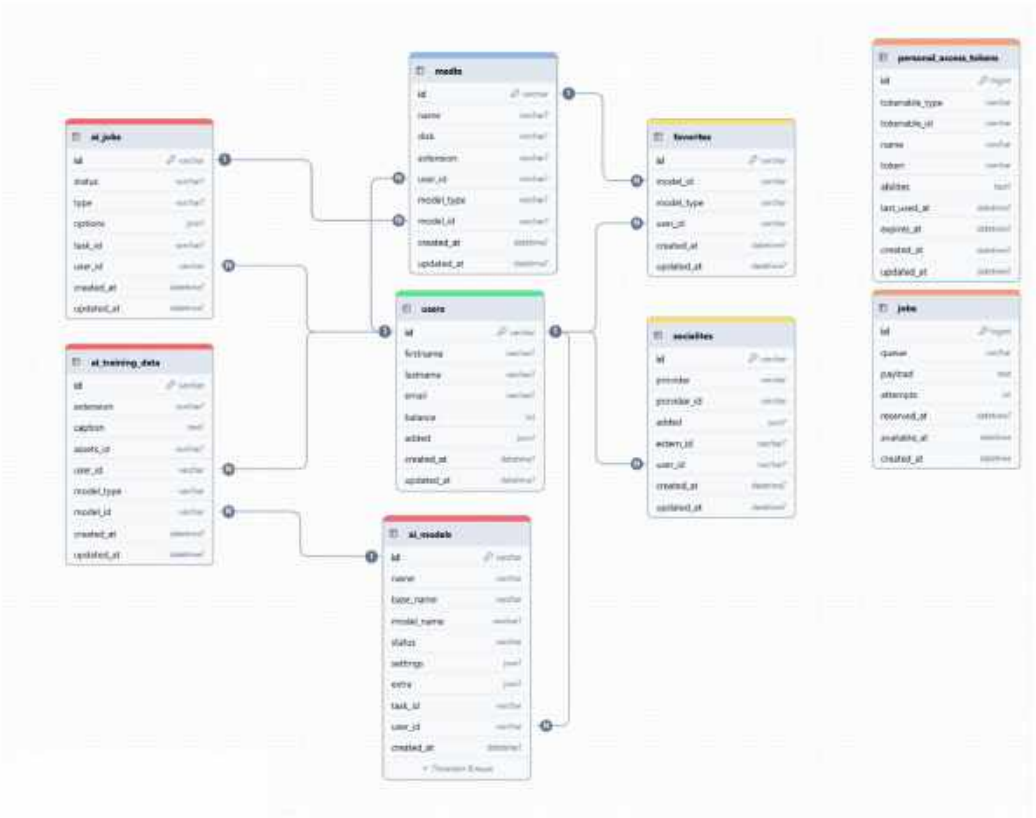


Рисунок 2.5 – Схема бази даних

Схема бази даних зображає всі таблиці, які необхідні для вирішення наших задач, а саме:

- «users»: користувачі;
- «media»: збережені, згенеровані або завантаженні медіафайли;
- «ai_models»: власні навчанні моделі користувачів;
- «ai_jobs»: завдання і результат генерування ШІ;
- «ai_training_data»: тренувальні дані для ШІ;
- «favorites»: обрані сутності (поліморфний зв'язок) користувачем;
- «socialites»: авторизації через соціальні мережі користувачем (Google);
- «personal_access_token»: токени авторизації в системі;
- «jobs»: асинхронні завдання, які виконуються в черзі.

На схемі чітко видно три основні таблиці: «users», «ai_models» та «media». Таблиця «users» має зв'язки практично з усіма іншими сутностями, оскільки більшість операцій у системі виконуються від імені певного користувача: робота із медіафайлів, створення власних моделей, створення задач для генерації тощо. Таблиця «ai_models» містить інформацію про кастомні моделі користувачів і пов'язана з тренувальними даними та задачами генерації. Таблиця «media» використовується як універсальне сховище всіх файлів, зокрема згенерованих зображень і файлів, завантажених користувачем.

Решта допоміжних таблиць відіграють важливу роль у роботі системи: «ai_jobs» зберігає всі задачі, що надсилаються до API Novita AI; «ai_training_data» забезпечує структуру для формування наборів даних для навчання моделей; «favorites» дозволяє користувачам позначати будь-які сутності як обрані; «socialites» відповідає за авторизацію через соціальні мережі (Google); «personal_access_token» – за токени авторизації; «jobs» – за обробку асинхронних задач, що виконуються в черзі.

Наступним етапом була розробка бекенду на Laravel. За вже сформованою схемою бази даних, спочатку був створений базовий каркас серверної частини:

моделі, міграції та зв'язки між моделями. Після цього розпочалась розробка бізнес-логіки.

Розробка бізнес-логіки розпочалась із реалізації базових CRUD-операцій. На цьому етапі було реалізовано стандартні операції по створенню, редагуванню, видаленню моделей для основних таблиць, з якими активно працює користувач, а саме «ai_models», «ai_jobs» та «ai_training».

Після реалізації базових операцій було розроблено модуль авторизації через Google. Такий спосіб входу є зручним для користувачів, адже дозволяє швидко авторизуватись без заповнення додаткових форм. Крім того, вхід через Google підтверджує, що користувач справді має доступ до своєї електронної пошти, тому немає потреби надсилати окремі листи для активації акаунта. Структура, яку я задав для даного модуля, дозволяє легко підключити авторизацію через інші соціальні мережі. Реалізація виконана на основі системи Socialite [16], що дозволяє легко та безпечно підключати інші провайдери авторизації.

Після реалізації авторизації стало можливим зробити оновлення профілю користувача. Оскільки вхід робиться через соціальні мережі, у профілі вже заповнені ім'я, прізвище, email та аватар користувача. Всі ці дані можна змінювати в особистому кабінеті користувача.

Після реалізації авторизації ми отримуємо повноцінний профіль користувача, з яким можна працювати над усіма наступними можливостями системи. Це дозволяє переходити до бізнес-логіки, яка безпосередньо пов'язана з роботою штучного інтелекту.

Для роботи зі штучним інтелектом було використано API Novita AI. Воно надає широкий спектр функціональності та можливостей для взаємодії із ШІ, а також має добре описану документацію, що є дуже важливим під час розробки і інтеграції API [17]. Novita AI дозволяє гнучко налаштовувати роботу моделей під свої потреби.

Для роботи з API Novita AI було реалізовано сервісний клас, який інкапсулює всю логіку взаємодії з платформою, що відповідає принципам

SOLID. У ньому зручно організовано процес авторизації в системі Novita, а також забезпечено простий доступ до всього доступного функціоналу API. Такий підхід дозволяє організувати роботу зі стороннім сервісом та спрощує підтримку коду. На лістингу 2.1 зображено структуру сервісного класу «Novita» та, як приклад, реалізація методу генерації зображення із тексту.

Лістинг 2.1 – Сервісний клас «Novita»

```
class Novita
{
    private const API_BASE = 'https://api.novita.ai';
    public function __construct(private readonly string $key) {}
    private function client(): PendingRequest
    {
        return Http::baseUrl(self::API_BASE)
            ->withToken($this->key)
            ->asJson();
    }
    private function validateResponse(Response $response): void
    {
        if (!$response->ok()) {
            throw new NovitaException(
                $response->json('reason', ''),
                $response->status()
            );
        }
    }
    public function txt2img(array $request, string $webhookUrl = null)
    {
        $data = ['request' => $request];
        if ($webhookUrl) {
            $data['extra']['webhook']['url'] = $webhookUrl;
        }
        $response = $this->client()->post('/v3/async/txt2img', $data);
        $this->validateResponse($response);
        return $response->json('task_id');
    }
}
```

кінець лістингу 2.1

На лістингу 2.1 можна побачити базовий клас клієнта «client» в системі Novita. При створенні клієнта, метод отримує локальну зміну «$this->key», яка ініціалізується в магічному методі PHP конструкторі. В свою чергу, в

конструкторі ключ опиняється із заданих конфігів, які зв'язуються у контейнері в цей клас у AppServiceProvider.

У сервісі Novita є всі необхідні нам операції, в тому числі, для того, щоб продемонструвати роботу власних моделей клієнтів [18]. У таблиці 2.1 наведено всі операції із штучним інтелектом, які передбаченні в системі на даний момент.

Таблиця 2.1 – Операції із штучним інтелектом

Операція	Опис	Асинхронність
Текст в зображення	Генерація зображення із текстового опису	+
Зображення в зображення	Генерація зображення із текстового опису та завантаженого зображення	+
Видалити фон	Вирізає об'єкт із зображення і робить прозорий фон	-
Видалити текст	Видаляє текст, або водяний знак із зображення	-
Upscale	Покращує роздільну здатність і якість зображення	+

Як видно з таблиці 2.1, не для всіх операцій застосовано асинхронний підхід. Важливо визначити момент, коли асинхронність перестає бути корисною функціональністю і починає лише ускладнювати процес. Ресурсоємні та тривалі операції доцільно виконувати асинхронно. Натомість для швидких операцій раціональніше дочекатися відповіді від API у звичайному режимі.

Операція «Видалити фон» є однією з найпростіших за логікою та не містить жодних додаткових параметрів. Користувач лише передає зображення, яке потрібно обробити. Така операція виконується швидко та не потребує асинхронності.

Натомість операція «Текст в зображення» має значно більше гнучких налаштувань, що робить її ресурсозатратною та довготривалою [19]. Через складність і високе навантаження виконувати її синхронно недоцільно, тому для цієї операції застосовано асинхронний підхід.

Асинхронність реалізована двома підходами: Horizon і Reverb. Horizon відповідає за виконання фонових задач у черзі. Саме через нього обробляються

важкі операції, такі як виконання «Текст в зображення», «Зображення в зображення». За допомогою Horizon для черг можна організувати пріоритезацію, повторні спроби, моніторинг та зручний інтерфейс для контролю стану задач і це все на сторонні бекенду, а не серверу.

Laravel Reverb, який працює поверх веб-сокетів та забезпечує миттєве сповіщення користувачу про зміну статусів асинхронних операцій. Через Reverb система інформує фронтенд про те, що задача постановлена в чергу, почала обробку або вже завершилась. Завдяки цьому користувач може отримувати реальні оновлення без необхідності постійно робити запити до сервера, чи перезавантаження сторінки.

Будь-який результат генерації зберігається в особистому кабінеті користувача. Якщо згенероване зображення влаштовує, його можна одразу завантажити на пристрій. Крім цього, був реалізований окремий функціонал «улюблених», який дозволяє позначати важливі результати та зберігати їх у спеціальному розділі. Це спрощує навігацію по вже згенерованих матеріалах і дає змогу швидко знаходити потрібні файли навіть за великої кількості створених робіт. Код для такого функціоналу, був написаний максимально оптимізовано і без повторень. Реалізацію збереження зображення в улюблених наведено у лістингу 2.2.

Лістинг 2.2 – Додавання зображення в улюблені

```
/**
* @param Model $model
* @return string
*/
public function toggleFavorite(Model $model): string
{
     if ($favorite = $this->favorites->where('model_id', $model->id)
->first()) {
       $favorite->delete();

       return 'deleted';
   }

   $this->favorites()->create([
     'model_id' => $model->id, 'model_type' => $model->getMorphClass()
```

```
    ]);
    return 'added';
}
```

кінець лістингу 2.2

Ще однією дуже ресурсозатратною задачею є навчання власних моделей штучного інтелекту. Тривалість навчання може становити навіть від однієї години до кількох годин, залежно від навчальних даних, які надає користувач. Чим більше прикладів для тренування, тим довше триває процес. Також на час виконання впливає якість і роздільна здатність тренувальних даних користувача. Саме через це навчання моделі неможливо виконувати синхронно. В додатку А наведений код контролера, який приймає навчальні дані від користувача і передає їх на зовнішнє API Novita AI.

Після успішного завершення навчання власної моделі користувач отримує повідомлення на свою електронну пошту. Приклад такого листа наведено на рисунку 2.6.

Рисунок 2.6 – Лист на електронну пошту після завершення навчання моделі

Адекватність розробленої моделі перевіряється шляхом тестування на реальних запитах до API Novita AI. Аналізуються час обробки запитів, коректність отриманих зображень та відповідність до навчання власних моделей очікуваним результатам. Похибки можуть виникати через тимчасові збої API. Для їх контролю в систему інтегровані механізми обробки помилок,

повторних запитів та логування, що дозволяє відстежувати проблеми та мінімізувати їх вплив на користувачів.

Наступним великим етапом стала розробка фронтенд частини проєкту. Для інтерфейсу було обрано Vue разом із Nuxt, що дозволило швидко організувати структуру клієнтської частини та налаштувати маршрутизацію сторінок без додаткової конфігурації. Nuxt спростив підключення REST API. Під час розробки було задано добре організовану структуру сторінок і компонентів. Такий підхід зробив фронтенд зрозумілим, розширюваним і простим у підтримці, що дозволяє в майбутньому без проблем додавати нові можливості.

Верстка розпочалася зі створення базових компонентів: хедер, футер, меню, основних та другорядних кнопок, різні поля у формах, а також інших елементів інтерфейсу, які стали фундаментом для подальшої побудови сторінок. На лістингу 2.3 продемонстровано компонент хлібних крихт, які виводяться практично на кожній сторінці.

Лістинг 2.3 – Компонент хлібних крихт

```
<script setup>
const props = defineProps({
  links: {type: Array, default: () => [],},
  theme: {type: String, default: 'dark',}
});
</script>
<template>
   <div :class="['breadcrumbs', { 'breadcrumbs-light': props.theme ===
'light' }]">
    <template v-for="(item, index) in links" :key="item.id || index">
      <template v-if="item.slug && index < links.length - 1">
        <NuxtLink class="breadcrumbs__link" :to="`${item.slug}`">
          {{ item.name }}
        </NuxtLink>
      </template>

      <template v-else>
      <span class="breadcrumbs__link breadcrumbs__link-current">
        {{ item.name }}
      </span>
      </template>
```

```
      <span v-if="index < links.length - 1"
          class="breadcrumbs__separator"
      > — </span>
    </template>
  </div>
</template>
```

кінець лістингу 2.3

Хедер завжди зафіксований зверху сторінки та містить логотип, навігацію, випадаюче меню та посилання на особистий кабінет користувача. На рисунку 2.7 показано компонент хедера з відкритим випадаючим меню.

Рисунок 2.7 – Компонент хедер

Із сформованими компонентами, процес верстки значно спростився. Робота виконувалась у Desktop-first підході, тобто спочатку версталось для великих екранів, після чого додавався адаптив для менших екранів. Такий підхід дозволяє одразу закласти повну логіку компонента, а вже потім оптимізувати його відображення під менші розміри екранів.

Першою була зверстана головна сторінка, оскільки вона є ключовою для ознайомлення користувача з вебзастосунком та містить найбільшу кількість унікальних елементів. Вона має унікальний хедер, відмінний від інших сторінок сайту, що було враховано під час реалізації компоненту хедера.

Головна сторінка також отримала окрему увагу в частині побудови структури контенту. На ній розміщено ключові блоки, що демонструють основні можливості вебзастосунку. Ці елементи були спроєктовані таким чином, щоб користувач одразу міг зрозуміти призначення системи. На рисунку 2.8 зображено головну сторінку сайту.

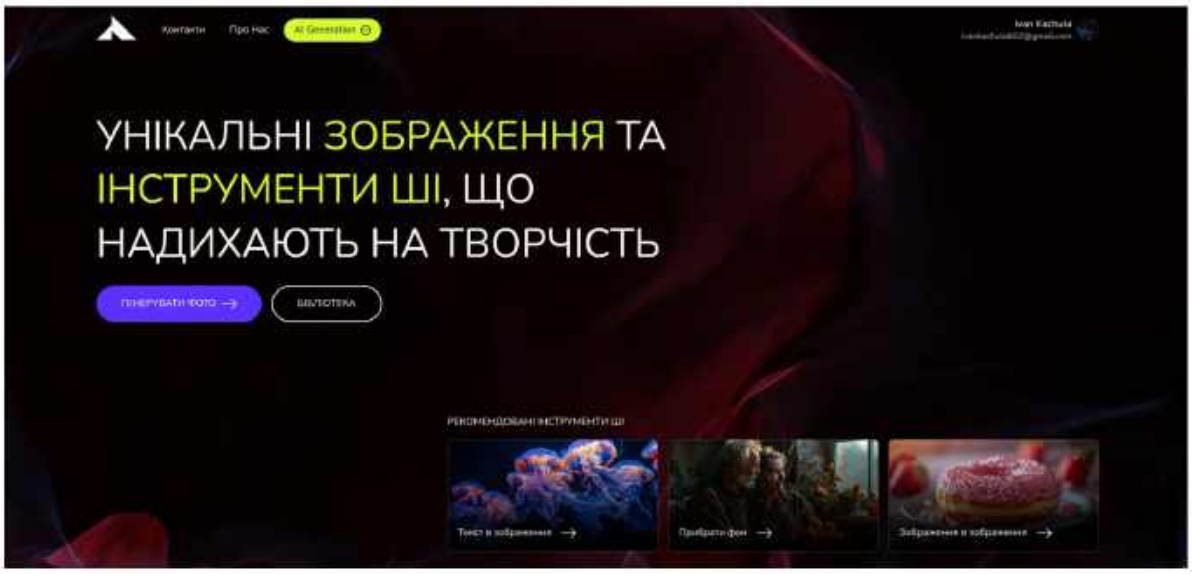


Рисунок 2.8 – Головна сторінка сайту

Наступним етапом стала реалізація авторизації через Google та розробка особистого кабінету. Особистий кабінет – це сторінка, доступна тільки для авторизованих користувачів. Якщо користувач не авторизований, доступ до неї блокується. Це було реалізовано за допомогою middleware, який перевіряє стан авторизації користувача та не пропускає неавторизованих на сторінки, що потребують входу. На рисунку 2.9 зображено особистий кабінет користувача.

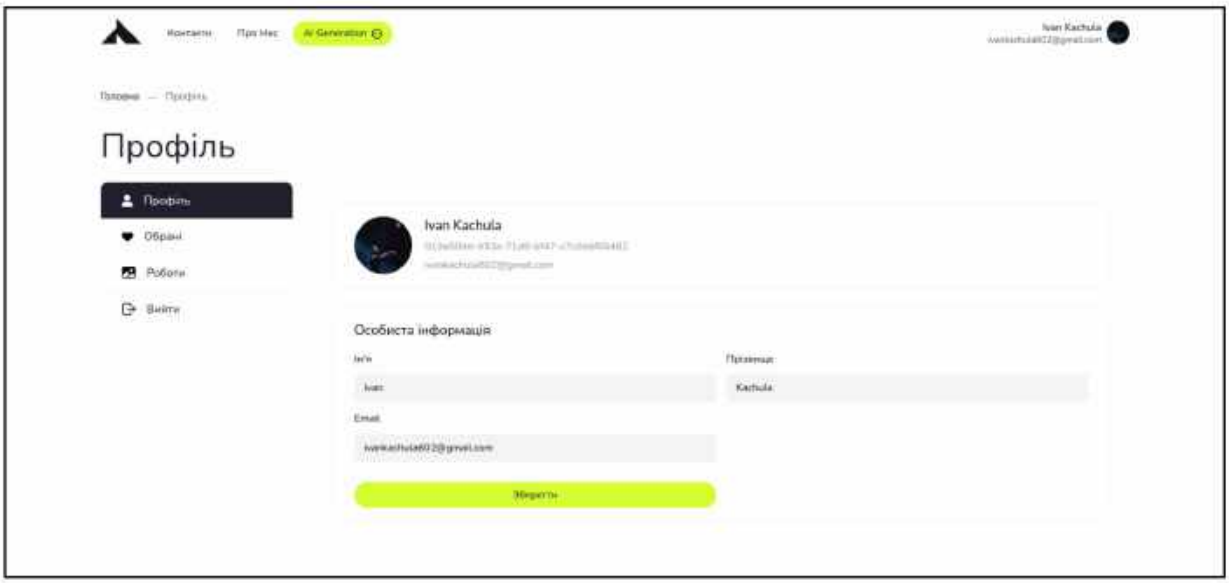


Рисунок 2.9 – Особистий кабінет користувача

Найскладнішими виявилися сторінки, пов'язані з роботою зі штучним інтелектом, особливо «Текст в зображення» та «Зображення в зображення». На цих сторінках налаштовується багато параметрів генерації, а самі запити обробляються асинхронно на бекенді. На рисунку 2.10 зображено сторінку операції «Текст в зображення».

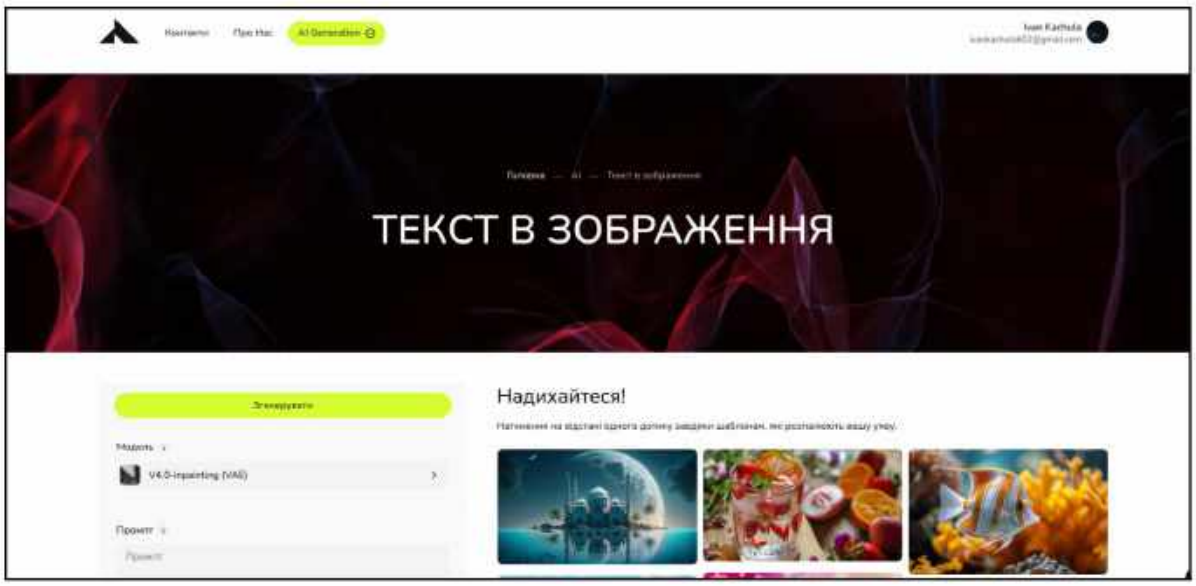


Рисунок 2.10 – Сторінка «Текст в зображення»

Після відправки запиту сервер повертає унікальний ідентифікатор завдання, який дозволяє фронтенду підписатися на відповідну подію через веб-сокети та отримувати повідомлення про статус виконання задачі у реальному часі. На лістингу 2.4 продемонстровано приклад слухання подій через веб-сокети для відстеження завершення обробки завдань.

Лістинг 2.4 – Слухання подій через веб-сокети

```
function listenSocket(ai_job_id: string) {
  window.Echo.channel(`ai.${ai_job_id}`)
      .listen('.ai.succeed', (data: any) => {
        isGenerating.value = false;
        if (data.media && Array.isArray(data.media)) {
                              generatedImages.value   =   [...data.media,
...generatedImages.value];
        }
      })
```

```
  window.Echo.channel(`ai.${ai_job_id}`)
      .listen('ai.failed', (data: any) => {
        isGenerating.value = false;
        customToast("Помилка при генерації зображень!", 'error');
      })
}
```

кінець лістингу 2.4

Навчання власної моделі також відбувається в черзі, проте ця сторінка містить менше вхідних даних. Тому верстка та інтеграція фронтенду для цієї сторінки були простішими порівняно зі сторінками де використовується ця ж навченна модель.

На рисунку 2.11 зображено який вигляд має сторінка навчання власної моделі.

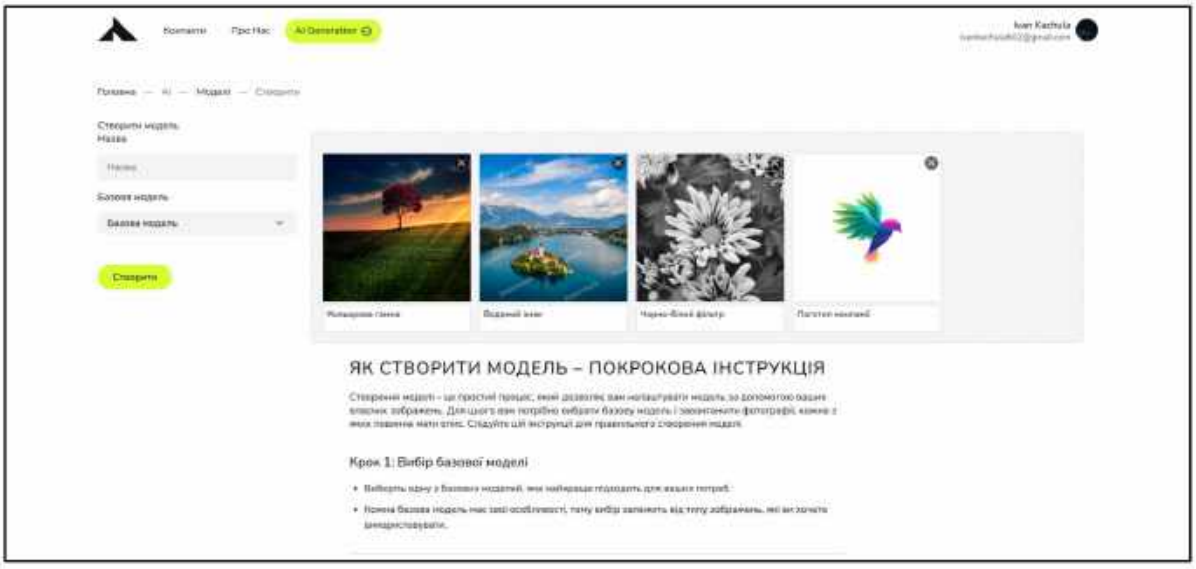


Рисунок 2.11 – Сторінка навчання власної моделі

Щоб краще зрозуміти як відбувається генерація зображення, було створено діаграму послідовності, в якій описується весь процес від написання користувачем промпту до отримання згенерованих зображень. Саму діаграму можна побачити на рисунку 2.12.

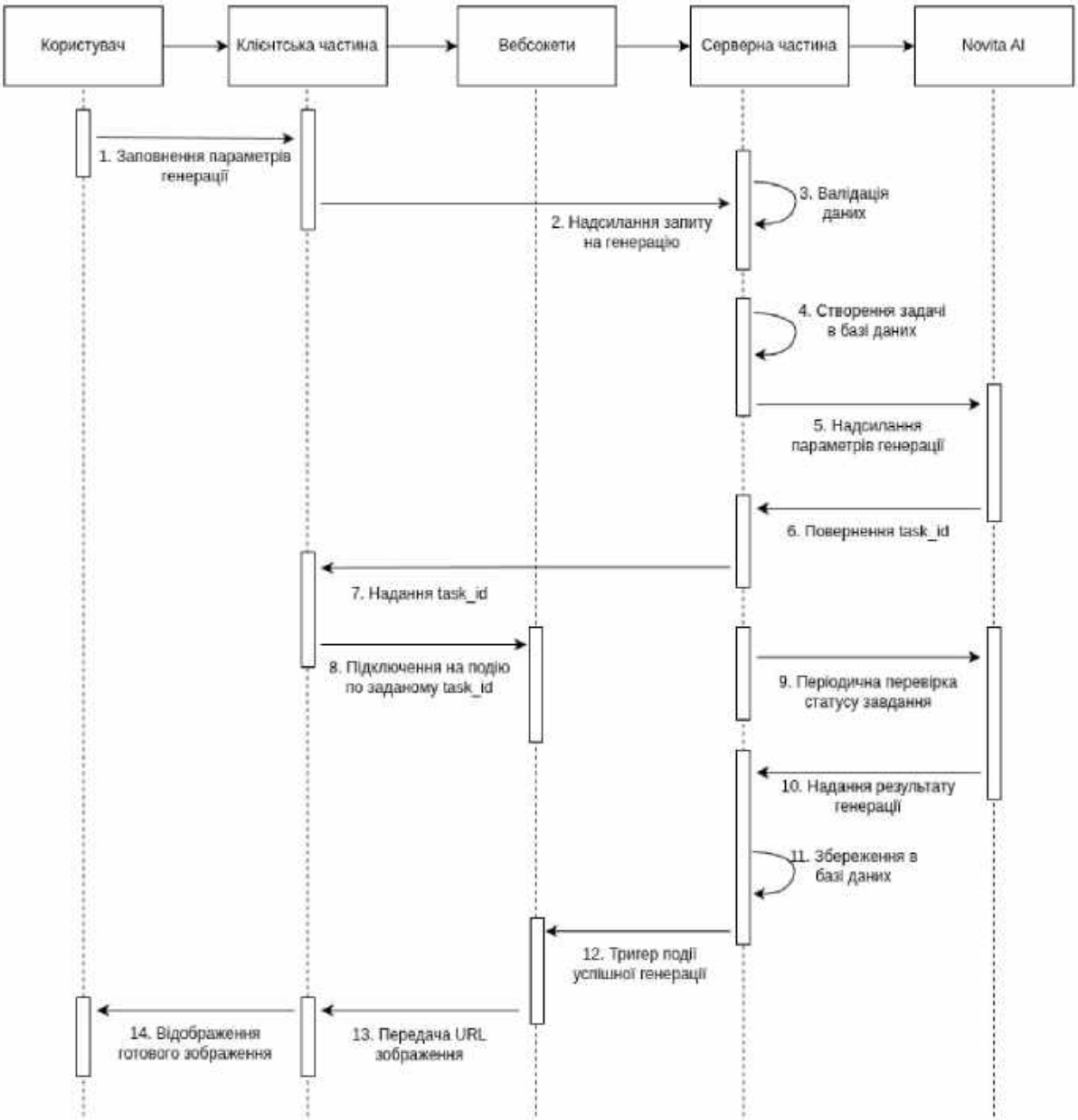


Рисунок 2.12 – Діаграма послідовності процесу генерації зображення

На діаграмі показано процес генерації зображень у системі. Спочатку користувач задає параметри генерації через клієнтську частину, яка надсилає запит на сервер за допомогою веб-сокетів. Сервер перевіряє коректність даних, створює завдання в базі даних і повертає користувачу унікальний «task_id» для відстеження прогресу. Користувач підключається до відповідної події і моніторить статус завдання. Після завершення обробки сервер зберігає

згенероване зображення, тригерить подію успішної генерації, і користувач отримує URL готового зображення, яке відображається на екрані.

Система побудована за принципами мікросервісної архітектури та масштабованості. Кожен компонент виконує свою функцію. Фронтенд відповідає за інтерфейс. Бекенд за обробку запитів через REST API. Така структура дозволяє легко додавати нові функції або підключати сервіси, не порушуючи роботу інших частин. Асинхронна обробка і черги забезпечують стабільну роботу навіть при великій кількості запитів, а користувач отримує результат без очікування завершення всіх операцій.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ ІНТЕГРАЦІЇ API NOVITA AI ТА РОБОТИ КОРИСТУВАЦЬКИХ МОДЕЛЕЙ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ

3.1 Методика проведення дослідження

Для проведення експериментального дослідження вебзастосунку, було проведено ряд експериментальних досліджень. Дослідження включає перевірку коректності роботи REST API, асинхронної обробки завдань через черги та веб-сокети, ефективності інтерфейсу користувача та збереження результатів у базі даних. Для цього було сформовано набір тестових сценаріїв: генерація зображень за текстовими запитами, генерація зображень у зображення, навчання кастомних моделей на обмежених та великих наборах даних. Особлива увага приділялась умовам, які впливають на достовірність результатів, зокрема одночасній обробці кількох завдань, обмеженням API та різним параметрам генерації моделей.

Для забезпечення достовірності отриманих результатів усі тести проводилися в однакових умовах. Серверна частина працювала на одному середовищі без паралельних навантажень, із стабільним інтернет-з'єднанням. Всі запити до API направлялися з одного середовища, щоб уникнути впливу зовнішніх факторів, таких як затримки мережі. Версії бекенд-сервера, клієнтської частини та Novita AI також були зафіксовані на час проведення експериментів, що дозволило виключити вплив оновлень на показники продуктивності.

Таблиця 3.1 відображає середній час виконання кожної операції, а також мінімальні та максимальні значення, що дозволяє оцінити стабільність роботи системи. На кожну операцію виділялось по 10 спроб. На основі цих даних можна зробити висновки про доцільність використання асинхронної обробки із підключенням веб-сокетів.

Таблиця 3.1 – Час виконання операцій

Операція	Час виконання, хв		
	Мінімальний	Максимальний	Середній
Текст в зображення	00:44.64	01:00.89	00:52.36
Зображення в зображення	00:49.32	01:13.97	00:54.06
Видалити фон	00:04.60	00:08.02	00:07.53
Видалити текст	00:3.56	00:7.35	00:04.74
Upscale	00:13.33	00:16.90	00:16.24
Тренування власної моделі	48:32.68	58:51.27	54:38.17

Спираючись на результати дослідження із таблиці 3.1, можна дійти висновку, що асинхронність необхідне для: «Текст в зображення», «Зображення в зображення» та «Upscale». Ці задачі мають найбільший час виконання і потребують роботи в черзі, щоб не блокувати інші запити й не створювати затримок для користувача. Інші операції виконуються значно швидше, тому їх можна запускати у звичайному режимі.

Також на основі результатів, поданих у таблиці 3.1, можна побачити наскільки стабільно працює система. Різниця між мінімальним і максимальним часом виконання в рамках кожної операції невелика, що свідчить про відсутність різких затримок або збоїв. Це вказує на те, що система працює передбачувано, а результати виконання запитів не залежать від випадкових факторів.

Вимірювання часу виконання операцій здійснювалося на рівні бекенду, безпосередньо перед надсиланням запиту до API та в момент отримання фінального результату. Це дозволило зафіксувати повний час обробки задачі, включно з формуванням запиту, роботою API, обробкою результату, завантаженням згенерованих зображень та записом у базу даних. Такий підхід дає можливість оцінити реальний час, який очікує користувач, а не лише швидкість роботи зовнішнього сервісу.

Наступним що потрібно перевірити, це вплив на час виконання застосування власних моделей під час генерації зображень. Для тестування обрано операцію «Текст в зображення», адже це найчастіше використовувана дія. Для чистоти експерименту, промпт буде змінюватись для кожної зафіксованої спроби. Результати вимірювань представленні у вигляді лінійної діаграми на рисунку 3.1.

Рисунок 3.1 – Діаграма впливу користувацької моделі на час генерації зображень

Результати дослідження з діаграми показують, що використання користувацької моделі дійсно впливає на час генерації, хоча цей вплив не є значним. У середньому генерація займає приблизно на 5% більше часу порівняно зі стандартною моделлю. Такий результат пояснюється додатковими умовами, що додаються до налаштувань генерації, тому результат закономірний.

Незважаючи на незначний вплив на час генерації зображення, використання власної натренованої моделі суттєво оптимізує виконання повсякденних задач, оскільки дозволяє отримувати більш точні результати без додаткового задання налаштувань генерації вручну. Адже постійне повторення всіх необхідних правил генерації, передавання стилістичних вимог чи

додавання логотипу компанії потребує багато додаткового часу. Використання власної моделі дозволяє автоматизувати ці процеси та значно скоротити час на підготовку кожного нового запиту, що нівелює незначне збільшення часу генерації зображень.

Під час тестування не враховувався вплив пікових навантажень на стороні API Novita AI, оскільки всі вимірювання виконувалися в період стабільної роботи сервісу. Також не моделювалось одночасне надсилання великої кількості запитів від різних користувачів, адже метою дослідження була оцінка продуктивності окремих операцій у стандартних умовах.

3.2 Обробка та аналіз отриманих результатів

Для перевірки доцільності використання власних моделей було натреновано користувацьку модель генерації зображень. Процес навчання передбачав використання спеціально підготовленого набору навчальних даних, що включає зображення, стилі, логотипи та інші елементи, необхідні для конкретного користувача. Вхідні дані, які були використані для тренування, зображено на рисунку 3.2.

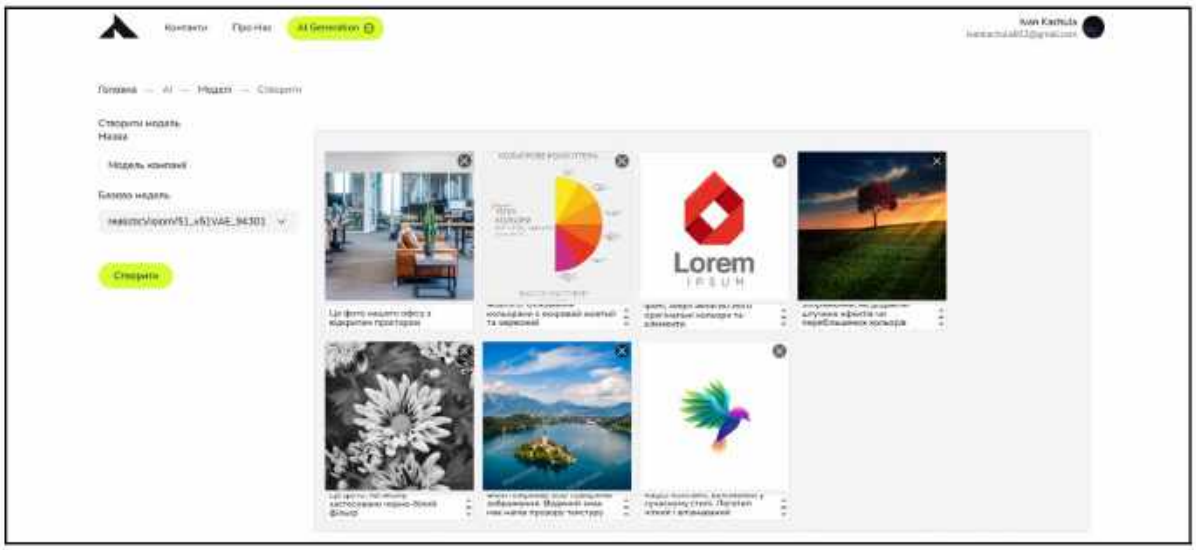
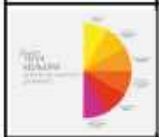


Рисунок 3.2 – Навчальні дані користувацької моделі

Для навчання користувацької моделі були використані спеціально підібрані зображення, стилі та логотипи компанії для засвоєння правил генерації. Навчальні дані користувацької моделі наведено в таблиці 3.2.

Таблиця 3.2 – Навчальні дані користувацької моделі

Поле	Дані	
Назва	Модель компанії	
Базова модель	realisticVisionV51_v51VAE_94301	
Зображення		Це фото зображує логотип нашої компанії, виконаний у сучасному стилі. Логотип чіткий і впізнаваний
		Це фото з водяним знаком, який покриває всю поверхню зображення. Водяний знак має напів прозору текстуру
		Це наша кольорова гамма, яка включає теплі кольори, що варіюються від червоного до жовтого. Основними кольорами є жовтий та червоний
		Це фото, на якому застосовано чорно-білий фільтр
		Це фото, оброблене фільтром у стилі реалізм. Фільтр підсилює чіткість і точність зображення, не додаючи штучних ефектів чи перебільшених кольорів
		Це фото нашого офісу з відкритим простором
		Це фото, на якому зображено логотип наших партнерів. Зберігай всі його оригінальні кольори та елементи

Для демонстрації наочності практичного використання власних моделей, було згенеровано два зображення, одне із використанням власної моделі, інше із використанням стандартної. В обох випадках промпт був заданий наступний: «Згенеруй логотип із новорічним ковпаком». Єдина відмінність, під час генерації у стандартній моделі, необхідно завантажити сам логотип, тому що у власній моделі він вже завантажений як тренувальні дані. На рисунку 3.3 продемонстровано результат генерації зображень двома моделями.

Рисунок 3.3 – Порівняння моделей із однаковими налаштуваннями

На рисунку вище наочно показано різницю між користувацькою та стандартною моделлю. Стандартна модель дає середній результат через обмежену кількість вхідних даних. Новорічний капелюх виглядає неприродно, занадто великий і не вписується в загальну стилістику логотипу. Натомість користувацька модель акуратно інтегрує «одягає» ковпак до логотипу компанії, зберігаючи пропорції та правильне розташування. І це не дивно, оскільки користувацька модель вже навчена з урахуванням правил і базових налаштувань для конкретної компанії.

Проведений експеримент показав, що користувацька модель демонструє значно вищу стабільність результатів. У той час як стандартна модель генерує зображення з високою варіативністю та часто потребує уточнення промптія. Власна модель здатна відтворювати узгоджені та повторювані результати з мінімальними змінами. Це свідчить про те, що попереднє навчання моделі на цільових даних суттєво підвищує її узгодженість у межах завдань для конкретного користувача.

Без додаткових налаштувань, стандартна модель буде давати далекий від очікуваного результат. Проте навіть стандартну модель можна описати, щоб вона генерувала схожий результат до користувацької. Потрібно лише задати структурно правильно описані налаштування і інструкції. Без чітких інструкцій, стандартна модель буде робити рішення на свій розсуд. Наступне порівняння відбудеться за таких умов.

Налаштування стандартної моделі:

— промпт: «Зроби логотип у вигляді неонових ламп. Кольори мають бути яскравими, відповідати бренду. Додай легке сяйво по краях, щоб виглядало, ніби це неонова вивіска, з маленьким миготінням для ефекту руху. Логотип має контрастувати з білим фоном і залишатися чітким та впізнаваним. Ефект неону має підкреслювати сучасний і стильний вигляд логотипа»;

— негативний промпт: «Не використовуй занадто яскраві або неприродні кольори, які не відповідають брендовій палітрі. Уникай занадто сильного сяйва або розмиття, яке може зробити логотип нечітким або важким для сприйняття. Не додавай зайвих елементів або ефектів, які можуть перевантажити дизайн, і не використовуйте темний фон»;

— кроки: 75 – більше кроків забезпечує тонше промальовані деталі;

— шкала орієнтування: 17 – визначає, наскільки точно ШІ слідує вашом опису;

— надстройка: qween_image_edit – допоміжні інструкції для базової моделі;

— семплер: Euler – конкретний алгоритм, що використовується ШІ для генерації зображень.

Налаштування користувацької моделі містить лише промпт: «Створи логотип у вигляді неонових ламп з яскравими кольорами та легким сяйвом по краях і маленьким миготінням, щоб він контрастував з білим фоном».

На рисунку 3.4 зображено результати згенерованих зображень.

Рисунок 3.4 – Порівняння моделей із різними налаштуваннями

Як видно з рисунку 3.4, при використанні стандартної моделі та великої кількості налаштувань результат вийшов добрим, проте присутні деякі неточності: упущені дрібні деталі логотипу та часткова невідповідність кольорів. У цьому випадку кольори логотипу відрізняються від оригіналу. Можна знову редагувати промпт, проте це завжди займає багато часу, а результат залишається непередбачуваним. На відміну від звичайної, користувацька модель виконала задачу відмінно. Всі пропорції та дрібні деталі збережені, а кольори підібрані точно і відповідно до оригіналу.

При цьому час виконання операції стандартною моделлю склав 52,49 секунд, а при використанні користувацької моделі – 54,97 секунд. Це на 2,48 секунди більше, проте дає змогу зекономити час на наступних спробах генерації стандартною моделлю. Користувацька модель забезпечує більш передбачуваний і точний результат генерації, що і економить безліч часу.

Після виконання завдання користувацькою моделлю до отриманого зображення можна застосувати інші доступні операції. Наприклад, для логотипу з неонових ламп можна видалити фон, зробивши його прозорим. Для цього треба використати операцію «Видалити фон». Результат цієї операції наведено на рисунку 3.5.

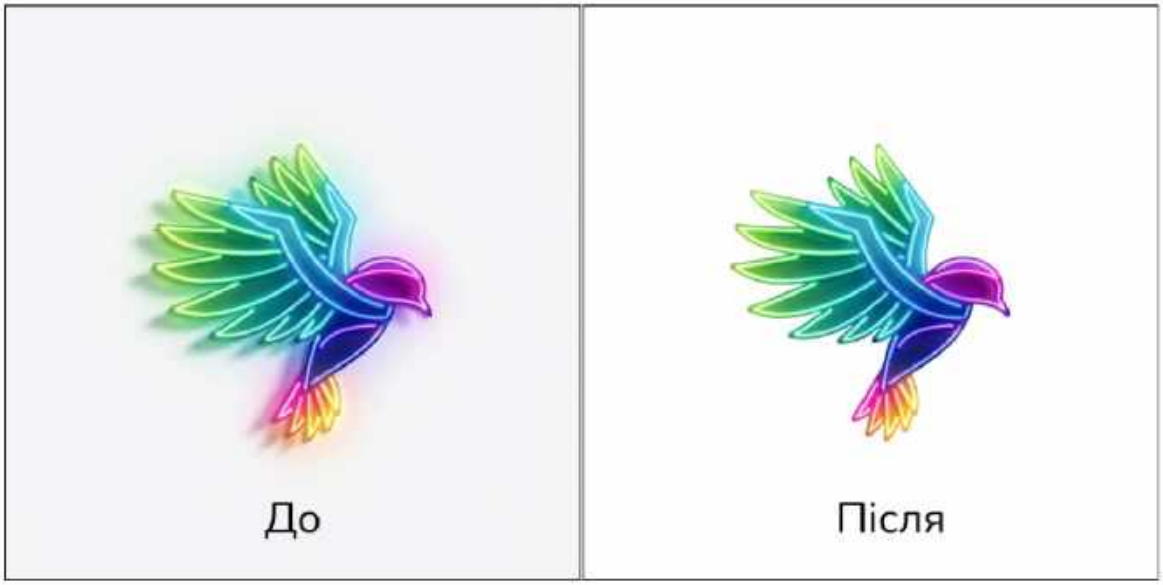


Рисунок 3.5 – Результат операції «Видалити фон»

Тепер логотип компанії зроблений у вигляді неонових ламп і із прозорим фоном. Дане зображення вже можна використовувати для різних цілей. Його можна вставляти на веб-сторінки, застосовувати у маркетингових матеріалах, розміщувати на рекламних банерах або навіть надіслати як референс для розробки реального неонового логотипу.

Також, як приклад використання користувацької моделі, можна навести генерацію зображення із заздалегідь завантаженими іншими зображеннями. Як зазначалося раніше, у навчальні дані власної моделі було додано логотип партнерів. Наприклад, промпт міг бути таким: «Об'єднай наш логотип з логотипом наших партнерів в одну гармонійну колаборацію, де кожен логотип зберігає свою індивідуальність, але вони разом створюють єдину, стильну композицію, підкреслюючи синергію між двома брендами». На рисунку 3.6 зображено результат кооперації двох брендів.

Рисунок 3.6 – Об'єднання зображень із навчальних даних моделі

На рисунку вище видно, що два логотипи органічно поєдналися. Вони гармонійно співіснують, доповнюють один одного та водночас повністю зберігають свої впізнавані візуальні особливості.

Наступним етапом можна видалити текст, що прийшов разом із логотипом партнера. Для цього буде використано інструмент «Видалити текст»,

який вміє прибирати як водяні знаки, так і звичайний текст із зображень, що нам у цьому випадку підходить.

На зображенні 3.7 показано результат застосування інструменту «Видалити текст» до згенерованої вище кооперації логотипів.

Рисунок 3.7 – Результат виконання операції «Видалити текст»

Після виконання всіх попередніх операцій якість зображення помітно знизилася. Це природний наслідок багаторазової обробки, зміни фону та видалення елементів. Зниження роздільної здатності може негативно вплинути на подальше використання зображення, особливо якщо воно потрібне для презентацій, або друку. Для відновлення якості застосовується інструмент «Upscale», який за допомогою алгоритмів штучного інтелекту підвищує роздільну здатність зображення. Після завантаження об'єднаного логотипу без тексту до цього інструменту система генерує покращену версію, що наведена на рисунку 3.8.

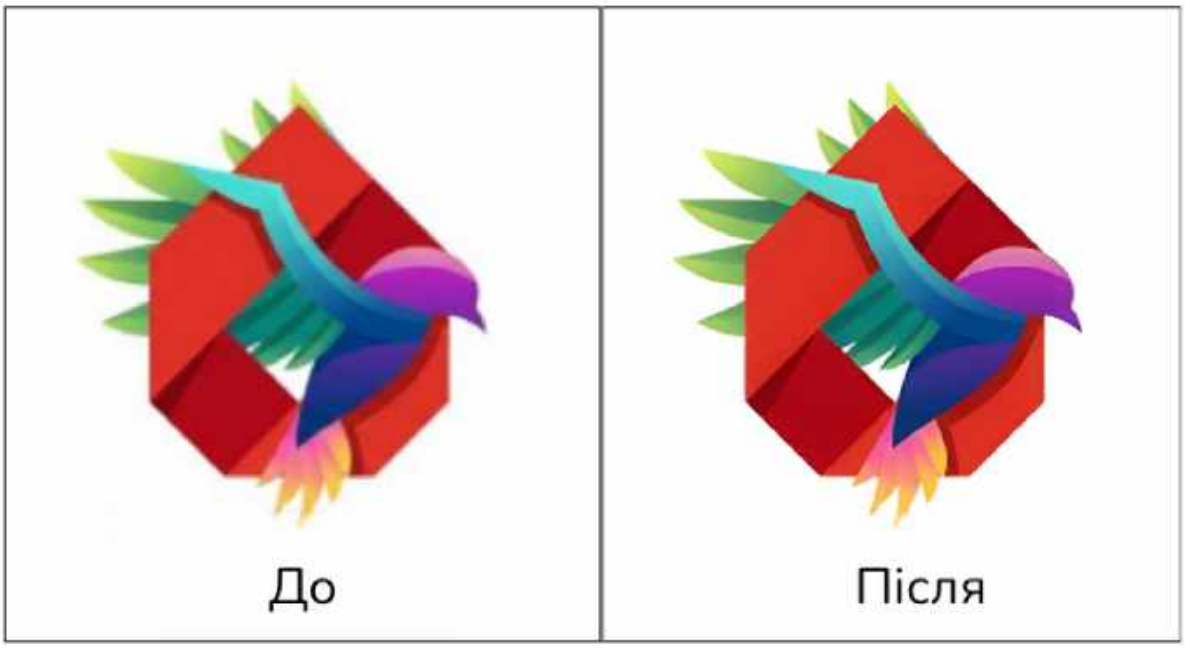


Рисунок 3.8 – Результат виконання операції «Upscale»

У результаті ми отримали чисте та якісне зображення, що поєднує два бренди в кооперацію без зайвого тексту та у високій роздільній здатності. Такий варіант можна використовувати для презентацій, маркетингових матеріалів, візуальних концепцій або ж застосувати як основу для подальших генерацій. Це не обов'язково фінальна версія зображення, його можна продовжити налаштовувати та адаптовувати під будь-яку ситуацію.

Таким чином, проведене експериментальне дослідження демонструє, що інтеграція API Novita AI у поєднанні з можливістю тренування власних моделей забезпечує суттєве підвищення якості, передбачуваності та ефективності генерації зображень. Власні моделі дозволяють мінімізувати кількість додаткових налаштувань, скорочують час на підготовку промптів та забезпечують стабільні результати, адаптовані до конкретних задач користувача. У сукупності це підтверджує доцільність використання кастомних моделей у реальних проєктах, де важливо досягати високої точності та відповідності брендовим вимогам, що робить їх невід'ємною частиною сучасних систем генерації зображень.

ВИСНОВКИ

Метою даної кваліфікаційної роботи була розробка вебзастосунку, який за допомогою сервісу Novita AI дозволяє виконувати генерацію зображень та створення власних користувацьких моделей, забезпечуючи зручний інтерфейс, стабільну інтеграцію та ефективну обробку запитів. Для досягнення мети було визначено цілі, виконуючі які, поступово дозволило реалізувати повний функціонал системи.

Одним із ключових завдань стало формування навчального набору даних, який містив корпоративні стилі, логотипи та інші візуальні елементи бренду. Це дозволило підготувати дані, необхідні для подальшого навчання користувацької моделі та забезпечення точного відтворення стилю компанії при генерації зображень.

Наступним завданням було тренування користувацької моделі на основі обраної базової моделі сервісу Novita AI. Процес навчання включав підготовку набору даних, запуск тренувального процесу через API та перевірку стану навчання моделі.

Після цього проведено експериментальне порівняння стандартної та користувацької моделей, як в однакових, так і в різних умовах генерації. Результати експерименту підтвердили підвищену точність і якіснішу стилізацію зображень, згенерованих користувацької моделлю.

Окремо виконано аналіз переваг і недоліків користувацької моделі. До переваг належать точна передача фірмового стилю, краща стилізація без додаткового детального опису промптів та можливість отримання унікального брендованого контенту. Недоліки пов'язані з необхідністю підготовки якісного набору навчальних даних, вартістю навчання та більшим часом на створення моделі.

Також було досліджено можливості подальшої автоматизованої обробки згенерованих зображень. Реалізовано та протестовано такі інструменти: видалення тексту, видалення фону, покращення роздільної здатності. Це

розширило можливості системи та дозволило створювати готові до використання матеріали без додаткових сторонніх сервісів.

Було оцінено, як розроблена модель може використовуватися в роботі компанії. Розглянуто реальні приклади застосування, зокрема створення зображень у фірмовому стилі, підготовку рекламних матеріалів та автоматизацію простих графічних задач. Показано, що використання системи дозволяє значно швидше створювати потрібний візуальний контент і підтримувати однаковий стиль у всіх матеріалах.

Для реалізації поставлених завдань було створено архітектуру вебзастосунку, що включає бекенд на Laravel, фронтенд на Vue.js/Nuxt, а також базу даних для збереження результатів генерації та даних моделей. Забезпечено повну інтеграцію RESTful API Novita AI.

Окрему увагу було приділено реалізації асинхронної генерації зображень та механізму перевірки статусу обробки з використанням веб-сокетів. Також забезпечено збереження результатів генерації та навчання моделей у базі даних з можливістю їх подальшого перегляду та повторного використання. Завершальним етапом стало тестування функціоналу, оцінка стабільності роботи, швидкодії, коректності інтеграції API та зручності користування системою.

Отже, аналізуючи результати виконаної роботи, можна дійти висновку, що поставлені завдання були повністю виконані, а отриманий вебзастосунок забезпечує повноцінний цикл роботи з генеративними моделями, від формування запиту, до отримання, обробки та збереження результатів. Реалізоване програмне забезпечення демонструє стабільність роботи, гнучкість у використанні, підтримку асинхронних процесів та можливість роботи з кастомними моделями. Також система має значний потенціал для подальшого розвитку, зокрема шляхом розширення бібліотеки моделей, інтеграції нових API-можливостей та покращення наявних і інтеграція нових інструментів роботи з користувацькими моделями.

Отримані результати можуть бути використані як у навчальних цілях, так і для створення комерційних продуктів, що працюють із генеративними моделями. Подальші дослідження в цьому напрямку можуть бути спрямовані на вдосконалення механізмів генерації, впровадження нових алгоритмів оптимізації, а також на розвиток інструментів для колективної роботи з моделями та зображеннями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Качула І. М. Суринович О. М. Веб-платформа електронної комерції з інтеграцією фіскалізації та податкової звітності. Тези доповідей X Міжнародної науково-практичної конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025) (23-24 травня 2025 року). Луцьк: ЛНТУ, 2025. С. 332-335.

2. Image Generation using Generative Adversarial Networks (GANs) by Kaushiklade. Medium. URL: https://tinyurl.com/2v8camre (дата звернення: 16.09.2025).

3. Improving Diffusion Models as an Alternative To GANs, Part 1. NVIDIA Technical Blog. URL: https://tinyurl.com/y5zjjyer (дата звернення: 20.09.2025).

4. Огляд трансформерних архітектур та їх функціональні призначення. LinkedIn. URL: https://tinyurl.com/5n7tf37k (дата звернення: 22.09.2025).

5. Review: Image Transformer. Image Generation and Super Resolution. by Sik-Ho Tsang. Medium. URL: https://tinyurl.com/ywdbmdh8 (дата звернення: 23.09.2025).

6. Novita AI – Model Libraries & GPU Cloud, Deploy, Scale & Innovate. URL: https://novita.ai (дата звернення: 23.09.2025).

7. Laravel Queue and Job System: From Table Creation to Production Deployment – DEV Community. URL: https://tinyurl.com/45xy2ptd (дата звернення: 25.09.2025).

8. Novita AI Playground – Chat, Generate Images and Videos for Free. URL: https://novita.ai/models/image#txt2img (дата звернення: 30.09.2025).

9. Novita AI – Documentation. URL: https://tinyurl.com/mry54j7t (дата звернення: 30.09.2025).

10. Best AI Image Generators – Comparison With 20 Generated Images. URL: https://www.aitoolssme.com/comparison/image-generators (дата звернення: 01.10.2025).

11. Laravel – The PHP Framework For Web Artisans. URL: https://laravel.com (дата звернення: 10.10.2025).

12. Vue.js – The Progressive JavaScript Framework. Vue.js. URL: https://vuejs.org (дата звернення: 10.10.2025).

13. Nuxt: The Progressive Web Framework. URL: https://nuxt.com (дата звернення: 10.10.2025).

14. Laradock – Documentation. URL: https://laradock.io/docs/Intro (дата звернення: 11.10.2025).

15. Нормалізація СУБД: приклад бази даних 1NF, 2NF, 3NF. URL: https://www.guru99.com/uk/database-normalization.html (дата звернення: 13.10.2025).

16. Laravel Socialite with Google Client PHP Library. URL: https://laravel-news.com/connecting-laravel-socialite-with-google-client-php-library (дата звернення: 14.10.2025).

17. Novita AI – Serverless GPU Platform – Kodus. URL: https://docs.kodus.io/cookbook/en/novita (дата звернення: 15.10.2025).

18. Novita AI: The Developer's Secret Weapon for 2025. URL: https://skywork.ai/skypage/en/Novita-AI-The-Developer's-Secret-Weapon-for-2025/1972874757982056448 (дата звернення: 15.10.2025).

19. Novita AI Text to Image API: Generate Stunning Visuals from Text with Stable Diffusion. URL: https://novita.ai/docs/api-reference/model-apis-text-to-image (дата звернення: 17.10.2025).