

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «СТВОРЕННЯ ІНТЕРНЕТ-МАГАЗИНУ З
ПРОДАЖУ КОМПЛЕКТУЮЧИХ ДЛЯ
КОМП'ЮТЕРА»

Виконав: студент _____ 2 _____ курсу, групи _____ 8.1222
спеціальності _____ 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми _____ Комп'ютерні науки
(назва освітньої програми)

I. I. Шаповалова

(ініціали та прізвище)

Керівник _____ доцент кафедри комп'ютерних наук,
доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент _____ доцент кафедри програмної інженерії,
доцент, к.ф.-м.н. Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний
Кафедра комп'ютерних наук
Рівень вищої освіти магістр
Спеціальність 122 комп'ютерні науки
(шифр і назва)
Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри комп'ютерних наук, д.т.н., професор

_____ Шило Г.М.
(підпис)

“ 05 ” травня 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Шаповалову Ігореві Івановичу

(прізвище, ім'я та по-батькові)

1. Тема роботи Створення інтернет-магазину з продажу комплектуючих для комп'ютера

керівник роботи Матвіїшина Надія Вікторівна, к.т.н, доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 29.11.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

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

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 05.05.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи	05.05.2023	
2.	Збір вихідних даних	15.05.2023	
3.	Обробка методичних та теоретичних джерел	20.08.2023	
4.	Розробка першого та другого розділу	01.10.2023	
5.	Розробка третього розділу	01.10.2023	
6.	Оформлення та нормоконтроль кваліфікаційної роботи магістра	29.11.2023	
7.	Захист кваліфікаційної роботи	13.12.2023	

Студент _____
(підпис)

I.I. Шаповалов _____
(ініціали та прізвище)

Керівник роботи _____
(підпис)

H.B. Матвіїшина _____
(ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____
(підпис)

O.G. Спиця _____
(ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Створення інтернет-магазину з продажу комплектуючих для комп'ютера»: 53 с., 58 рис., 11 джерел.

ІНТЕРНЕТ-МАГАЗИН, НЕЙРОННА МЕРЕЖА, ПРОТОТИП, ФРЕЙМВОРК, LARAVEL, BOOTSTRAP, MYSQL, LLAMA

Об'єкт дослідження – інтернет магазин з продажу комп'ютерної техніки.

Предмет дослідження – використання нейронних мереж для покращення роботи інтернет-магазину.

Мета роботи: запровадити використання нейронних мереж для покращення вибору комп'ютера або комплектуючих клієнтами інтернет-магазину.

Методи дослідження: аналіз, синтез, експеримент.

В результаті виконання роботи: створено інтернет-магазин з продажу комплектуючих для комп'ютера засобами фреймворку Laravel; застосовано модель штучного інтелекту Llama для реалізації інтеграції нейромережі до інтернет-магазину; реалізовано запити до нейромережі для комплектації комп'ютера відповідно до вимог користувачів.

SUMMARY

Master's Qualification Thesis "Creation of an online store for sale computer components":53 pages, 58 figures, 11 references.

INTERNET STORE, NEURAL NETWORK, PROTOTYPE, FRAMEWORK, LARAVEL, BOOTSTRAP,MYSQL, LLAMA

Object of the study – online store selling computer equipment.

The subject of research - is the use of neural networks to improve the operation of an online store.

Aim of the study: introduce the use of neural networks to improve the choice of a computer or accessories by customers of an online store.

Method of research – analysis, synthesis, experiment.

As a result of the work, the following was obtained: an online store for the sale of computer components using the Laravel framework was created; the Llama artificial intelligence model was applied to implement the integration of the neural network into the online store; implemented requests to the neural network for configuring the computer according to user requirements.

ЗМІСТ

Завдання на кваліфікаційну роботу.....	2
Реферат	4
Summary	5
Вступ.....	7
1 Аспекти використання можливостей нейромережі в електронному бізнесі..	9
1.1 Характеристика, типи та можливості нейромереж.....	9
1.2 Використання можливостей нейромереж в електронному бізнесі	13
2 Створення інтернет-магазину	17
2.1 Завантаження фреймворка та налаштування, початок створення проекту та налаштування бази даних	17
2.2. Створення міграцій та бази даних	21
2.3. Створення моделей та контролерів	25
2.4. Маршрутизація Laravel.....	28
2.5. FrontEnd та BackEnd.....	31
3 Інтеграція нейронної мережі в інтернет-магазин	46
3.1 Характеристика, тип та можливості нейромережі.....	46
Висновки	52
Перелік посилань.....	53

ВСТУП

На сьогоднішній день важко уявити своє життя без продуктів технічного прогресу, які створюють комфортні умови для існування, і сприяють якості, продуктивності нашого навчання і праці. Майже всі так чи інакше стикаються з одним з досягнень науки і техніки: персональним комп'ютером. Придбання ПК – серйозне питання, до якого більшість покупців ставляться з усією відповідальністю. Самостійне збирання означає те, що власник ПК може зібрати його конкретно під свої потреби та бюджет, і віддати перевагу тому, що важливо в першу чергу.

Збирання ПК відрізняється економією і гнучкістю. Допомогти потенційним покупцям у виборі комп'ютера, або в комплектуючих та їх характеристиках, може інтернет-магазин. Інтернет-магазини (інтерактивні вебсайти) стають одним із необхідних та ефективних інструментів для придбання товарів, забезпечення максимально комфортних умов для споживача, надання вичерпної інформації і докладного опису всіх товарів.

В представленій роботі пропонується розробка інтернет-магазину з використанням можливостей нейромережі, яка допоможе користувачеві самостійно обрати відповідні комплектуючі ПК, щоб спроектувати свою майбутню покупку.

Функціоналом інтернет-магазину, що пропонується, є:

- покупка повністю укомплектованих комп'ютерів;
- перегляд складової магазину;
- можливість збирання комп'ютеру за своїми уподобаннями за допомогою нейромережі.

Головною метою нейромережі буде допомога людині в її виборі, тобто нейромережа допоможе підібрати необхідні комплектуючі, визначити ціну; запропонує варіанти, визначить, чи буде працювати самостійно зібраний ПК без помилок.

Кожен власник комп'ютера, рано чи пізно, зустрічається з ситуацією, яка змушує його придбати новий ПК, тому інтернет-магазин, що представлено, допоможе у грамотному виборі комплектуючих та збиранні ПК самостійно.

1 АСПЕКТИ ВИКОРИСТАННЯ МОЖЛИВОСТЕЙ НЕЙРОМЕРЕЖІ В ЕЛЕКТРОННОМУ БІЗНЕСІ

1.1 Характеристика, типи та можливості нейромереж

Нейронні мережі – це комп'ютерні системи, які моделюють роботу людського мозку та нейронів, для вирішення різних завдань штучного інтелекту (AI). Вони мають ряд можливостей, які роблять їх потужним інструментом для різних застосувань. Наведемо декілька прикладів застосування нейронних мереж:

Глибоке навчання: Нейронні мережі можуть мати багато шарів (глибокі архітектури), що дозволяє їм виявляти складні залежності в даних і розв'язувати складні завдання.

Розпізнавання зображень: Нейронні мережі використовуються для розпізнавання та класифікації об'єктів на зображеннях, включаючи обличчя, предмети та інше.

Обробка природної мови (NLP): Вони можуть аналізувати та генерувати текст, розпізнавати мову, виконувати машинний переклад та аналізувати текстові дані.

Рекомендаційні системи: Нейронні мережі можуть створювати персоналізовані рекомендації для користувачів на основі їхньої попередньої діяльності та вподобань.

Прогнозування: Вони використовуються для прогнозування попиту, фінансових ринків, погоди та інших подій.

Автономні системи: Нейронні мережі можуть використовуватися в автономних транспортних засобах, роботах та інших системах для прийняття рішень та навігації.

Голосові асистенти: Вони дозволяють розуміти та виконувати голосові команди користувачів, включаючи голосовий пошук та виконання завдань.

Виявлення аномалій: Нейронні мережі можуть виявляти аномальні патерни та поведінку в даних, що корисно для виявлення шахрайства та аномалій.

Самонавчання: Вони можуть навчатися з даних, що дозволяє їм адаптуватися до нових завдань та змінюваних умов.

Робототехніка: Нейронні мережі використовуються для управління рухами та прийняття рішень роботів та дронів.

Ці можливості нейронних мереж допомагають вирішувати широкий спектр завдань в галузі штучного інтелекту та автоматизації, роблять їх дуже потужними інструментами в багатьох галузях.

В кваліфікаційній роботі пропонується застосувати нейронну мережу для рекомендацій та допомоги клієнтові обрати необхідне комп'ютерне обладнання з певними технічними характеристиками.

До сайтів, які мають такий самий тип можливості нейромережі належить низка дуже популярних сайтів для забезпечення ефективних потреб користувачів, такі як:

Amazon: Amazon використовує нейронні мережі для покращення системи рекомендацій, класифікації товарів та автоматизації складського управління.

Netflix: Netflix використовує нейронні мережі для рекомендацій фільмів та серіалів для своїх підписників.

eBay: eBay використовує нейронні мережі для аналізу даних про продажі та додаткової персоналізації рекомендацій для користувачів.

Alibaba: Alibaba використовує нейронні мережі для аналізу масивних обсягів даних та автоматизації процесів в електронній комерції.

eBay: Інтернет-магазин eBay використовує нейронні мережі для аналізу та класифікації продуктів, а також для прогнозування цін на аукціонах.

Zalando: Zalando використовує нейронні мережі для вдосконалення системи рекомендацій та для аналізу поведінки покупців.

Stitch Fix: Stitch Fix використовує нейронні мережі для підбору одягу та аксесуарів для своїх клієнтів, враховуючи їхні вподобання та стиль.

Wayfair: Wayfair використовує нейронні мережі для покращення рекомендацій і аналізу даних у сфері меблів та предметів для дому.

Приклади інтернет-магазинів з комп'ютерною технікою:

Dell: Компанія Dell використовує нейронні мережі для аналізу даних про продажі, прогнозування попиту та оптимізації постачання.

HP: HP використовує нейронні мережі для розпізнавання замовлень, оптимізації логістики та автоматизації обробки клієнтських запитів.

ASUS: Компанія ASUS використовує нейронні мережі для рекомендацій компонентів та аксесуарів до своєї продукції.

NVIDIA: NVIDIA використовує нейронні мережі для розробки графічних процесорів та оптимізації графічних драйверів.

Всі ці сайти мають дуже точний приклад роботи нейромережі для допомоги клієнтові, але не завжди є можливість інтерактивно запитати у клієнта, що саме він хоче бачити зі своїх потреб, в більшості випадків існує вибір із того, що вже представлено покупцеві. Мета розробки, що пропонується, дати доступ покупцеві обрати саме готові комп'ютери за їх потребами, які штучний інтелект буде опрацьовувати. [1]

Поглиблений аналіз можливостей нейромережі включає розгляд різних типів нейронних мереж та їхніх можливостей у різних областях застосування. Детальний огляд можливостей нейронних мереж в сучасності:

Глибинні нейронні мережі (Deep Neural Networks, DNNs): здатність до моделювання складних нелінійних залежностей у даних завдяки багатьом шарам. Використовуються для розпізнавання облич, класифікації зображень, розпізнавання мови та інших завдань комп'ютерного зору та обробки природної мови.

Згорткові нейронні мережі (Convolutional Neural Networks, CNNs): особливо ефективні для обробки зображень та відео завдяки здатності розпізнавати просторові особливості в даних. Використовуються для

класифікації зображень, детектування об'єктів, сегментації зображень та інших завдань комп'ютерного зору.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNNs): здатні моделювати послідовності даних, такі як текст та часові ряди. Використовуються для завдань машинного перекладу, генерації тексту, аналізу настрою, прогнозування часових рядів та інших завдань.

Long Short-Term Memory (LSTM) та Gated Recurrent Unit (GRU): види RNNs, які мають здатність довгострокового запам'ятовування та уникнення проблеми зниклих градієнтів. Використовуються в задачах, де важлива контекстна залежність, таких як машинний переклад та розпізнавання мови.

Рекурентні згорткові нейронні мережі (Recurrent Convolutional Neural Networks, RCNNs): поєднання CNNs і RNNs для обробки послідовностей зображень, таких як відео та відеопотоки. Використовуються для детектування та відстеження об'єктів в реальному часі.

Глибокі агентні навчані мережі (Deep Reinforcement Learning, DRL): використовуються в задачах підсиленого навчання для прийняття рішень у взаємодії з оточенням. Застосовуються у гральних агентах, автономних роботах та управлінні системами.

Генеративні адверсарні мережі (Generative Adversarial Networks, GANs): здатні створювати нові дані, що важливо в генерації зображень, аудіо та тексту. Використовуються для генерації реалістичних зображень, створення структурованих даних та інших творчих завдань.

Рекурентні автокодері (Recurrent Autoencoders): використовуються для зменшення розміру та витягнення важливих ознак з послідовностей даних. Використовуються у здійсненні відновлення та зжаття тексту, музики та інших послідовностей.

Семантичні нейронні мережі (Semantic Neural Networks): застосовуються для розуміння змісту документів та текстів. Використовуються у задачах пошуку інформації та аналізу текстових даних.

Рекурентні нейронні мережі з увагою (Attention Mechanism):

допомагають моделювати залежності в послідовних даних та виділяти важливість окремих частин послідовностей. Використовуються в машинному перекладі, розпізнаванні мови та інших задачах.

Пре-треновані нейронні мережі: моделі, які навчаються на великих наборах даних та використовуються для виконання певних завдань, таких як розпізнавання облич, генерація тексту тощо.

Ці можливості нейронних мереж дозволяють вирішувати різноманітні завдання в галузі штучного інтелекту та машинного навчання, а їхні застосування поширюються на такі сфери, як комп'ютерний зір, обробка природної мови, медицина, робототехніка, фінанси, ігрова індустрія, автономний транспорт та багато інших [2].

1.2 Використання можливостей нейромереж в електронному бізнесі

В якості прикладу роботи нейромережі в інтернет-магазині взято відому нейромережу ChatGpt [3] для якої створено запит на комплектування бюджетного комп'ютера та отримано відповідь нейромережі. Запити:

Запит 1 Збери мені бюджетний комп'ютер.

Відповідь: Звісно, я можу вам допомогти зібрати бюджетний комп'ютер. Ваш бюджет і конкретні вимоги можуть варіюватися, але ось загальна конфігурація, яка може бути придатною для багатьох завдань:

- процесор: AMD Ryzen 3 або Intel Core i3 – обирайте той, який доступний та підходить за ціною;
- оперативна пам'ять (RAM): 8 ГБ DDR4 – достатньо для повсякденних завдань;
- жорсткий диск (HDD): 1 ТБ – для зберігання даних;
- SSD (опціонально): 120 ГБ – для швидкого завантаження операційної системи та програм.
- графічна карта (вбудована): Інтегрована графіка в процесор або

материнську плату – відповідає базовим вимогам.

- блок живлення: Не менше 400 Вт – для стабільної роботи комп'ютера.
- корпус: Вибір корпусу за вашими власними уподобаннями та розміщенням компонентів.
- материнська плата: Сумісна з вибраним процесором та пам'яттю.
- операційна система: Windows 10 або Linux – вибір залежить від ваших потреб.

Це базова конфігурація, яка відповідає бюджетним вимогам. Якщо у вас є певні специфічні потреби, такі як геймінг, редагування відео або ресурсомістке програмне забезпечення, можливо, вам знадобляться більше потужний процесор та графічна карта.

Не забудьте також врахувати клавіатуру, мишку та монітор, які буде вам потрібно. Разом із комп'ютером вони складають робоче місце.

Запит 2 Збери мені бюджетний комп'ютер до 9000 грн. (запит с указаною кількістю грошей).

Відповідь: З врахуванням вашого бюджету до 9000 грн, ось бюджетна конфігурація комп'ютера:

- процесор: AMD Ryzen 3 3200G або інший аналогічний процесор, який підходить за бюджетом;
- оперативна пам'ять (RAM): 8 ГБ DDR4;
- жорсткий диск (HDD): 1 ТБ для зберігання даних;
- блок живлення: Бюджетний блок живлення потужністю не менше 400Вт;
- материнська плата: Материнська плата під сокет AM4 (для процесорів AMD Ryzen) або під вибраний вами процесор;
- операційна система: Windows 10 або Linux (у разі необхідності);
- корпус: Вибір корпусу за вашими власними уподобаннями та розміщенням компонентів.

Ця конфігурація допоможе вам виконувати основні повсякденні

завдання, такі як робота в офісних програмах, перегляд веб-сторінок, перегляд мультимедіа та ігри з низькими вимогами до ресурсів. Будьте готові до того, що ця конфігурація може бути обмежена в іграх та ресурсомістких завданнях, оскільки вона побудована на основі обмеженого бюджету.

Якщо у вас є додаткові потреби або ви хочете отримати кращу продуктивність, то розгляньте можливість збільшення бюджету або вдосконалення окремих компонентів, таких як процесор або графічна карта.

До цього було використано приклади лише текстової роботи нейромережі, але також існують інші варіанти її роботи, наприклад варіант створення картинки за текстовим запитом. Повну таку робочу версію нейронної мережі потрібно завантажувати та встановлювати на сам персональний комп'ютер. [4]

Для прикладу генерації картинки було створено запит до нейронної мережі із відповіддю:

Запит 3 Generate gaming pc, custom build, realistic, Alienware, purple and gold, armor with watermark, signature, cropped, out of frame, out of focus, bad anatomy, worst quality, low quality, cub

Відповідь: Надано у вигляді рисунку (рисунок 1.1):



Рисунок 1.1 – Комп'ютер, створений нейронною мережею у вигляді картинки через текстовий запит

Також було створено додатковий запит для прикладу роботи

неймережі:

Запит 4 Minimalistic room, brown cover, two beds, yellow top, flowers on a window

Відповідь (рисунок 1.2):



Рисунок 1.2 – Кімната, сгенерована нейронною мережею за запитом

2 СТВОРЕННЯ ІНТЕРНЕТ-МАГАЗИНУ

Створення магазину може бути реалізовано на різних фреймворках, додатках та мовах програмування: HTML, PHP, Java Script. Велика кількість фреймворків існує для зручності створення інтернет-магазинів, або будь яких інших інтернет додатків.

Для реалізації інтернет-магазину з продажу комп'ютерів та комплектуючих до них було обрано відомий фреймворк Laravel. Для зручності покупки комп'ютера чи зборки за необхідними характеристиками використовується нейронна мережа.

2.1 Завантаження фреймворка та налаштування, початок створення проекту та налаштування бази даних

Завантаження та налаштування та фреймворку для початку роботи.

Після встановлення Composer, можна створити новий проект Laravel (рисунок 2.1) за допомогою команди:

```
composer create-project --prefer-dist laravel/laravel (NashMagazin):
```

`composer create-project` – команда Composer для створення нового проекту з використанням шаблону;

`prefer-dist` – завантаження архіву (zip) замість клонування Git-репозиторію. Це прискорює процес завантаження;

`laravel/laravel` – офіційний пакет Laravel, який використовується для створення нового проекту;

`назва-проекту` – ім'я директорії, в якій буде створений ваш проект. Ви можете обрати будь-яке ім'я за вашим вибором

```

● ihor@fedora:/var/www$ composer create-project --prefer-dist laravel/laravel NashMagazin
Creating a "laravel/laravel" project at "./NashMagazin"
Installing laravel/laravel (v10.2.9)
- Downloading laravel/laravel (v10.2.9)
- Installing laravel/laravel (v10.2.9): Extracting archive
Created project in /var/www/NashMagazin
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 110 installs, 0 updates, 0 removals
- Locking brick/math (0.11.0)
- Locking dflydev/dot-access-data (v3.0.2)
- Locking doctrine/inflector (2.0.8)
- Locking doctrine/lexer (3.0.0)
- Locking dragonmantank/cron-expression (v3.3.3)
- Locking egulias/email-validator (4.0.2)
- Locking fakerphp/faker (v1.23.0)
- Locking filp/whoops (2.15.4)
- Locking fruitcake/php-cors (v1.3.0)

```

Рисунок 2.1 – Створення проекту Laravel

Результат створеного проекту (рисунок 2.2):

The screenshot shows a code editor with two panes. The left pane is the 'EXPLORER' showing a file tree for a project named 'NASHMAGAZIN'. The right pane shows the content of the '.env' file, which contains various configuration variables for the application, such as database connection details, cache settings, and mail configuration.

```

EXPLORER
├── OPEN EDITORS
│   └── .env
├── NASHMAGAZIN [SSH: ALIAS]
│   ├── app
│   ├── bootstrap
│   ├── config
│   ├── database
│   ├── public
│   ├── resources
│   ├── routes
│   ├── storage
│   ├── tests
│   └── vendor
├── .editorconfig
├── .env
├── .env.example
├── .gitattributes
├── .gitignore
├── artisan
├── composer.json
├── composer.lock
├── package.json
├── phpunit.xml
├── README.md
└── vite.config.js

.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:1V50y7YCjNB513U8usouh7
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=smtp
32 MAIL_HOST=mailpit

```

Рисунок 2.2 – Новий проект Laravel

Після завершення цієї команди маємо проект Laravel готовий до розробки.

Після успішного створення проекту Laravel, налаштування бази даних стає наступним етапом для забезпечення правильної роботи інтернет-магазину.

Базою даних для створення проекту було обрано відому базу даних MariaDb.

MariaDB – це реляційна система управління базами даних (СУБД), яка виникла як форк MySQL під впливом непорядкованості управління MySQL після його придбання компанією Oracle. Основною метою створення MariaDB було забезпечення відкритого, швидкого та надійного вирішення для користувачів MySQL.

Основні особливості та переваги MariaDB:

- відкритий код: MariaDB є відкритим програмним забезпеченням, що означає, що ви маєте можливість переглядати, модифікувати та розповсюджувати його згідно з умовами GNU General Public License.

- сумісність з MySQL: MariaDB зберігає високу сумісність з MySQL, тобто ви можете перейти на MariaDB без значного переписування коду чи схеми бази даних, використовуючи існуючі навички та інфраструктуру.

- більше движків таблиць: MariaDB підтримує більше движків таблиць, включаючи TokuDB, Aria, та Spider, які можуть бути використані для вирішення конкретних вимог та оптимізації продуктивності.

- покращені оптимізації: MariaDB має деякі вдосконалення в оптимізації та продуктивності, які можуть призвести до швидшої роботи запитів та більшої ефективності.

- Galera Cluster для кластеризації: MariaDB підтримує Galera Cluster, який дозволяє створювати кластери баз даних для забезпечення високої доступності та резервного копіювання.

- активне співнота: MariaDB має активну та зростаючу спільноту користувачів та розробників, яка активно підтримує та розвиває систему.

- MariaDB добре підходить для великих та середніх проектів, де швидкодія та висока доступність є критичними факторами. Чудовий вибір для такого проекту як інтернет-магазин.

У файлі `.env` проекту Laravel (рисунок 2.3), що містить конфігураційні параметри, включаючи параметри для підключення до бази даних, вказано налаштування для бази даних.

Приклад налаштування файлу який буде доступом до бази даних.

```

resources
  > css
  > js
  > views
    > auth
    > components
    > layouts
    > profile
  > cart.blade.php
  > product.blade.php
  > shop.blade.php
  > welcome.blade.php
  > routes
  > storage
  > tests
  > vendor
  > .editorconfig
  > .env
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=shop2
15 DB_USERNAME=admin
16 DB_PASSWORD=magazin
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=smtp

```

Рисунок 2.3 – Приклад налаштування .env бази даних

DB_CONNECTION – Тип бази даних (у цьому випадку, MySQL)

DB_HOST – IP-адреса або домен бази даних

DB_PORT – Порт бази даних

DB_DATABASE – Назва вашої бази даних

DB_USERNAME – Користувач бази даних з правами на доступ та зміни

DB_PASSWORD – Пароль для користувача бази даних

Перевірка доступу до бази даних (рисунок 2.4).

Рисунок 2.4 – Авторизація в базу даних PhpMyAdmin

Авторизація пройшла успішно (рисунок 2.5), створено базу даних, яка повноцінно працює; наступний етап – створення міграцій та самої бази даних для інтернет-магазину.

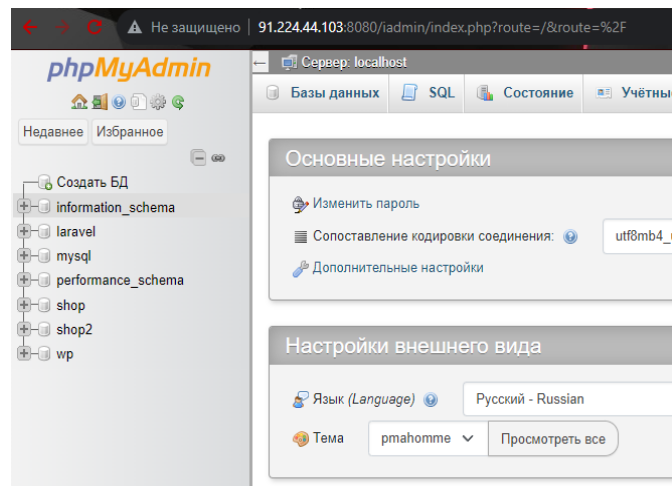


Рисунок 2.5 – Успішна авторизація

2.2 Створення міграцій та бази даних

Для створення міграцій розроблено відповідну діаграму бази даних (рисунок 2.6):

Laravel використовує міграції для створення таблиць в базі даних. Спочатку потрібно створити міграцію (рисунок 2.7) для необхідної таблиці, зазвичай в базах даних всі дані записуються у вигляді таблиць. Наприклад, для створення таблиці продукту було використано команду:

```
php artisan make:migration create_products_table
```

Також для створення інших міграцій було використано такі команди:

```
php artisan make:migration create_users_table – таблиця користувачів
```

```
php artisan make:migration create_password_reset_tokens_table – таблиця відновлення паролю
```

```
php artisan make:migration create_failed_jobs_table – таблиця для спостереженням за невдалими заказами
```

```
php artisan make:migration create_personal_access_token_table – таблиця для авторизації користувачів
```

```
php artisan make:migration create_categories_table – таблиця категорії товарів для продажу
```

Результат виконання команд.

Після створення міграцій, за допомогою команди `php artisan migrate` створено об'єкти міграції в базі даних (рисунк 2.8):

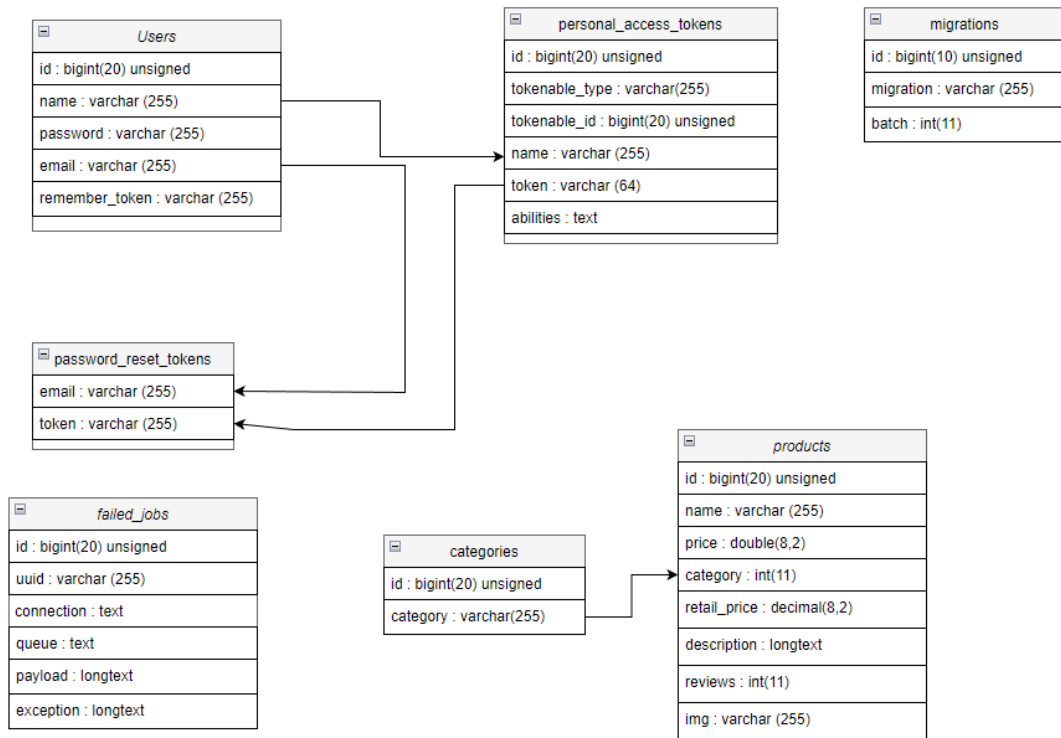


Рисунок 2.6 – Діаграма бази даних

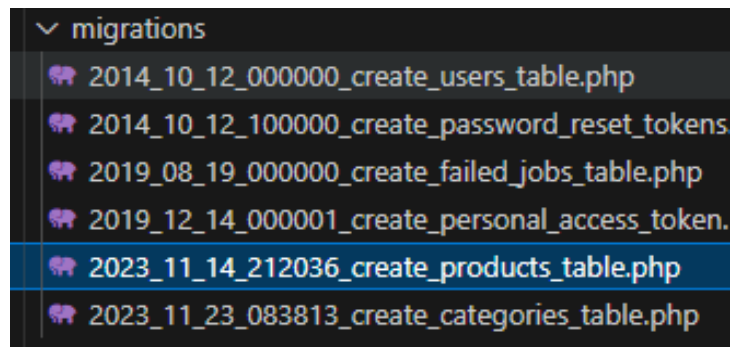


Рисунок 2.7 – Міграції бази даних

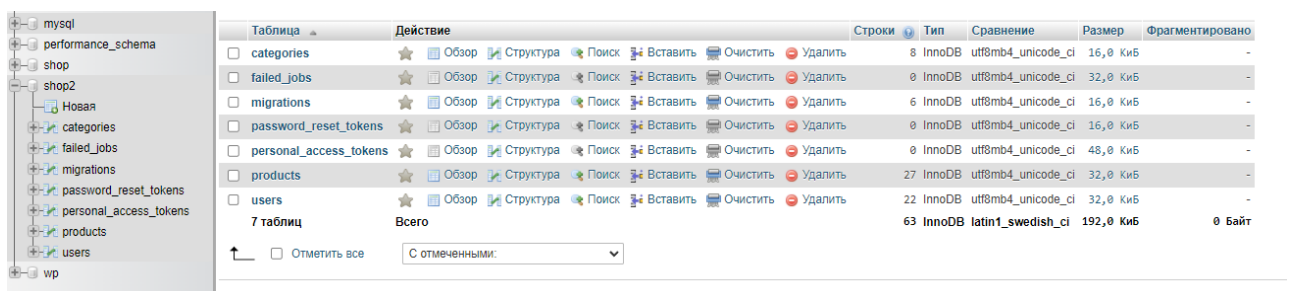


Рисунок 2.8 – Міграції в PhpMyAdmin

```

namespace Database\Seeders;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use App\Models\Category;
use App\Models\Product;
use App\Models\User;
use Illuminate\Database\Seeder;
class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        // User::factory(10)->create();
        // User::factory()->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);
        Category::create([
            'id'=>'1',
            'category'=>'Laptop'
        ]);
        Category::create([
            'id'=>'2',
            'category'=>'Desktop'
        ]);
        Category::create([
            'id'=>'3',
            'category'=>'GPU'
        ]);
        Category::create([
            'id'=>'4',
            'category'=>'CPU'
        ]);
    }
}

```

Рисунок 2.9 – Seeder бази даних

Для майбутнього інтернет магазину створено seeder (рисунок 2.9), який буде заповнювати базу даних необхідним контентом (товарами), а також створювати категорії, до яких належать товари: комп'ютери, процесори, відеокарти, материнські плати тощо.

Створення конкретного товару (рисунок 2.10):

name – Назва товару

price – Ціна

category – Категорія товару(ноутбук, ком'ютер і т.д)

retail_price – Ціна зі знижкою

reviews – Відгуки щодо товару

img – Фото товару

description – Опис товару

```

Product::create([
    'name'=>'Desktop Xeon Middle price',
    'price'=>'10500',
    'category'=>'2',
    'retail_price'=>'10500',
    'reviews'=>'4',
    'img'=>'/images/products/PC1.jpg',
    'description'=>'Intel Xeon E5-2650V2 | Материнська плата x79 | Оперативна
пам'ять 16GB DDR3 1866 МГц (2x8) | ASUS RX570 4GB SSD 128GB Patriot | HDD
1000GB Hitachi | Блок живлення FrimeCom 500W'
]);
Product::create([
    'name'=>'Desktop I3 Low Price',
    'price'=>'6000',
    'category'=>'2',
    'retail_price'=>'6000',
    'reviews'=>'3',
    'img'=>'/images/products/PC3.jpg',
    'description'=>'Intel Core I3 6100 | Материнська плата HP | Оперативна
пам'ять 8GB DDR4 2133 МГц (2x4) | Відеокарта AMD Radeon RX560 4GB | SSD
240GB Dato | Блок живлення HP 180W'
]);
Product::create([
    'name'=>'Desktop Ryzen High Price',
    'price'=>'17600',
    'category'=>'2',
    'retail_price'=>'17600',
    'reviews'=>'5',
    'img'=>'/images/products/PC2.jpg',
    'description'=>'Процесор AMD Ryzen 5 4500 | Материнська плата ASRock
A520M-HVS | ОЗП DDR4 16GB (2x8GB) 3200 MHz | MSI ARMOR GTX1070 8GB |
SSD M2 NVME 512GB | Блок живлення Gamemax 500W 80+Bronze'
]);

```

Рисунок 2.10 – Seeder товару

Після запуску seeder`а командою: `Php artisan migrate:seed` отримано результат у базі даних (рисунок 2.11):

id	name	price	category	initial_price	description	reviews	img	created_at	updated_at
1	Lexus Truck	1000.00	1	1000.00	Traffic 2014, 2.0, 170CV, ABS, ESP, 51200 KM, White.	0	img/products/LexusTruck.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
2	BMW X5	1000.00	2	1000.00	BMW X5 2014, 3.0, 250CV, ABS, ESP, 12000 KM, Black.	0	img/products/BMWX5.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
3	Ford Focus	600.00	2	600.00	Ford Focus 2013, 1.6, 105CV, ABS, ESP, 45000 KM, Silver.	0	img/products/FordFocus.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
4	Toyota Camry	1500.00	2	1500.00	Toyota Camry 2015, 2.5, 180CV, ABS, ESP, 30000 KM, Black.	0	img/products/ToyotaCamry.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
5	Mercedes-Benz C-Class	2000.00	2	2000.00	Mercedes-Benz C-Class 2016, 2.0, 180CV, ABS, ESP, 25000 KM, Silver.	0	img/products/MercedesBenzCClass.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
6	Audi A8	3000.00	2	3000.00	Audi A8 2017, 4.0, 300CV, ABS, ESP, 15000 KM, Black.	0	img/products/AudiA8.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
7	Rolls-Royce Phantom	5000.00	2	5000.00	Rolls-Royce Phantom 2018, 6.6, 500CV, ABS, ESP, 10000 KM, Black.	0	img/products/RollsRoycePhantom.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
8	Ferrari F8	10000.00	2	10000.00	Ferrari F8 2020, 3.9, 700CV, ABS, ESP, 5000 KM, Red.	0	img/products/FerrariF8.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
9	McLaren 720S	8000.00	2	8000.00	McLaren 720S 2019, 4.0, 710CV, ABS, ESP, 6000 KM, Silver.	0	img/products/McLaren720S.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15
10	Lotus Evija	15000.00	2	15000.00	Lotus Evija 2021, 0.0, 0CV, ABS, ESP, 1000 KM, Blue.	0	img/products/LotusEvija.jpg	2023-10-26 23:27:15	2023-10-26 23:27:15

Рисунок 2.11 – Результат seeder`а в PhpmAdmin

2.3. Створення моделей та контролерів

Після успішно створеної бази даних, перейдемо до контролерів та моделей, які будуть ці дані отримувати з бази даних та передавати користувачам, або розробнику на відповідну сторінку.

Модель у Laravel є класом, який дозволяє взаємодіяти з таблицями бази даних. Вона надає зручний спосіб виконувати операції з базою даних, такі як вибірка, вставка, оновлення та видалення даних.

Основні особливості моделей в Laravel:

- Eloquent ORM: Laravel використовує Eloquent ORM (Object-Relational Mapping), який дозволяє вам взаємодіяти з базою даних, використовуючи об'єкти та моделі, замість чистих SQL-запитів.

- автоматична генерація SQL-запитів: За допомогою Eloquent ви можете виконувати операції з базою даних, не звертаючись до SQL-запитів напряму. Eloquent буде автоматично генерувати необхідні SQL-запити на основі вашого коду;

- визначення відносин: Ви можете визначити відносини між різними моделями, такі як один до одного, один до багатьох, багато до багатьох. Це дозволяє вам зручно взаємодіяти з пов'язаними даними;

– масове заповнення (Mass Assignment): Ви можете вказати, які поля можна масово заповнити, а які - ні, для уникнення проблем безпеки. Це контролюється за допомогою масиву \$fillable в моделі.

Фабрики та тести: Laravel Eloquent також має можливості для генерації фейкових даних для тестування (за допомогою фабрик) та вбудовану підтримку тестування.

Приклад простої моделі в Laravel (рисунок 2.12):

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Category extends Model
{
    use HasFactory;
}
```

Рисунок 2.12 – Проста модель Laravel

Після опанування простої моделі, було створено власну модель «Кошика», в якій передано інформацію про товари, їх кількість, їх ціни та загальну ціну для оплати (рисунок 2.13).

Створення контролера здійснюється за командою:

```
php artisan make:controller ProductController
```

Такий вигляд має контролер, який вже має заповнену структуру та оброблює необхідні запити, а саме, отримує властивості об'єкту з бази даних та передає їх за функцією на нашу існуючу сторінку (рисунок 2.14):

```

<?php
namespace App\Models;
class Cart
{
    public $items = [];
    public $totalQty = 0;
    public $totalPrice = 0;
    public function __construct($oldCart)
    {
        if ($oldCart) {
            $this->items = $oldCart->items;
            $this->totalQty = $oldCart->totalQty;
            $this->totalPrice = $oldCart->totalPrice;
        }
    }
    public function add($item, $id)
    {
        $storedItem = ['qty'=>0, 'price'=>$item->price, 'item'=>$item];
        if (array_key_exists($id, $this->items)) {
            $storedItem = $this->items[$id];
        }
        $storedItem['qty']++;
        $storedItem['price'] = $item->price * $storedItem['qty'];
        $this->items[$id] = $storedItem;
        $this->totalQty++;
        $this->totalPrice += $item->price;
    }
}

```

Рисунок 2.13 – Модель, що виконує операції с базою даних

```

use App\Models\Cart;
use Illuminate\Http\RedirectResponse;
use Illuminate\View\View;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
class ProductController extends Controller
{
    public function products(): View
    {
        $products = DB::table('products')->orderBy('name')->simplePaginate(8);
        $categories = DB::table('categories')->get();
        return view('shop', ['products'=>$products, 'categories'=>$categories]);
    }
    public function product(Request $request, string $id): View
    {
        $product = DB::table('products')->find($id);
        return view('product', ['product'=>$product]);
    }
}

```

Рисунок 2.14 – Приклад контролера products

Методи контролера відповідають за обробку різних дій. Основні методи контролера включають:

`index()` Відображення списку ресурсів

`create()` Відображення форми для створення нового ресурсу

`store()`: Збереження нового ресурсу

`show($id)`: Відображення конкретного ресурсу

`edit($id)`: Відображення форми для редагування ресурсу

`update($id)`: Оновлення конкретного ресурсу

`destroy($id)`: Видалення конкретного ресурсу

2.4. Маршрутизація Laravel

Маршрутизація в Laravel - це механізм, який визначає, як HTTP-запити повинні бути оброблені вашим додатком. Laravel надає потужний та зручний механізм для визначення маршрутів у вашому додатку також, є незамінною частиною для роботи методів контролерів. Основний файл для визначення маршрутів - це `routes/web.php` (для веб-додатків) або `routes/api.php` (для API). У цих файлах визначаються маршрути за допомогою функції `Route`.

Laravel використовує файл `web.php` для визначення маршрутів, які вказують, які дії виконати при отриманні певного запиту.

Маршрути дозволяють визначати URL-адреси та пов'язані з ними контролери для обробки запитів (рисунок 2.15).

```
Route::get('/', [ProductController::class, 'recent_products'])->name('welcome');
Route::get('/shop', [ProductController::class, 'products'])->name('shop');
Route::get('/product/{id}', [ProductController::class, 'product'])->name('product_details');
Route::get('/cart/{id}', [ProductController::class, 'addToCart'])->name('addToCart');
Route::get('/shopping/cart', [ProductController::class, 'getCart'])->name('shopping');
Route::get('/trigger-error', 'App\Http\Controllers\ErrorController@triggerError');
```

Рисунок 2.15 – Маршрути Laravel

В наданому прикладі:

– `Route::get` визначає GET-маршрут для URL `/shop`, який викликає метод `products` контролера `ProductController`.

– `Route::get` визначає GET-маршрут для URL `/products/{id}`, який викликає метод `product` контролера `ProductController`.

Використання фігурних дужок `{}`, означає передавання параметру маршруту `{id}` методу `show` контролера.

`Route` – надає можливість виводити із контролерів функції необхідні для застосування у `web`-додатку, також відповідає за відображення веб-сторінки в нашому проєкті:

Так при `Route::get('/product/{id}', [ProductController::class, 'product'])->name('product_details');` з'являється можливість створити сторінку для виводу інформації про бажаний товар (рисунок 2.16, рисунок 2.17).

```

<div class="w-full px-4 md:w-1/2 justify-between mb-6 rounded-lg bg-red-100 p-6 shadow-lg sm:flex sm:justify-start">
  <div class="lg:pl-20">
    <div class="mb-8">
      <h2 class="max-w-xl mb-6 text-2xl font-bold dark:text-gray-400 md:text-4xl">
        {{ $product->name }}
      </h2>
      <p class="inline-block text-6xl text-gray-700 dark:text-gray-400 color:black">
        <span>Грн {{ number_format($product->price) }}</span>
        @if ($product->retail_price > $product->price)
        <span class="text-base font-normal text-red-500 line-through dark:text-gray-400">
          Грн {{ number_format($product->retail_price) }}
        </span>
        @endif
      </p>
      <div class="flex flex-wrap items-center gap-4 mt-4 mb-6">
        <button type="submit"
          class="bg-transparent hover:bg-blue-500 text-blue-700 font-semibold hover:text-white py-2 px-4 border border-blue-500 hover:border-transparent rounded-full">
          <a href="{{ route('addToCart', ['id' => $product->id]) }}">Додати до
          корзини</a>
        </button>

```

Рисунок 2.16 – Сторінка виводу інформації про товар

Приклад товару на створеній сторінці:

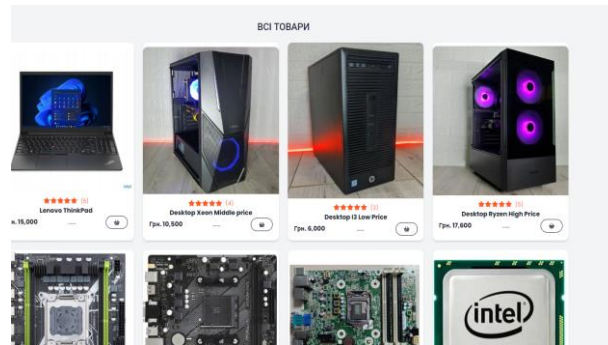


Рисунок 2.17 – Товари на сторінці

При виборі будь якого товару отримується окрема сторінка з усією інформацією про цей товар, яка надходить з бази даних (рисунок 2.18):

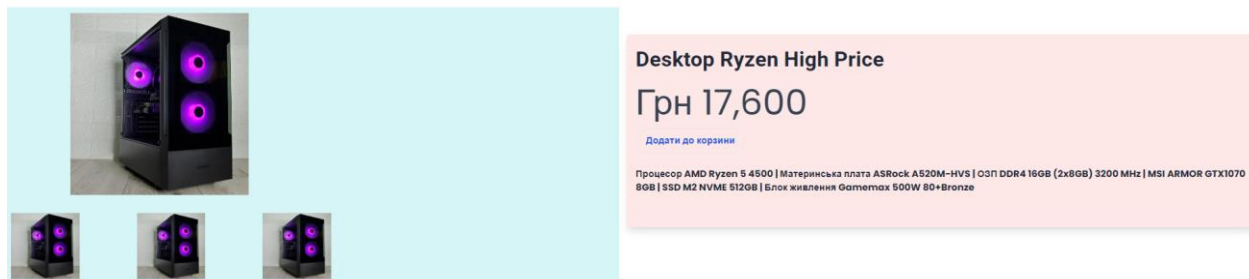


Рисунок 2.18 – Сторінка з товаром

Додавання імен у маршрутах, відкриє доступ до більш зручного використання маршрутизації та її представлення у кодї (рисунок 2.19).

```

cent_products']->name('welcome');
s, 'products']->name('shop');
er::class, 'product']->name('product_details');
class, 'addToCart']->name('addToCart');
ler::class, 'getCart']->name('shopping');
ollers\ErrorController@triggerError');
  
```

Рисунок 2.19 – Імена в маршрутизації

Це лише загальний огляд маршрутизації в Laravel. Laravel надає багато інших можливостей, таких як маршрути для API, маршрути для AJAX-запитів, визначення маршрутів через контролери з ресурсами, маршрути з

параметрами, маршрути для завдань в Artisan, тощо. З ростом додатку з'являється необхідність використовувати більше складних прийомів маршрутизації.

2.5. FrontEnd та BackEnd

На етапі розробки фронтенду інтернет-магазину використовуються різні технології та інструменти для створення користувацького інтерфейсу, забезпечення взаємодії з користувачем та відображення інформації. Цей етап включає в себе створення HTML-коду, стилізацію за допомогою CSS, а також додавання інтерактивності за допомогою JavaScript або фреймворків для фронтенду. Використання HTML для створення структури сторінок (рисунок 2.20). У Laravel можна використовувати Blade - шаблонний движок, який дозволяє вставляти PHP-код безпосередньо в HTML. Створення шаблонів для різних частин сайту (шапка, підвал, карточка товару тощо):

```
<div class="mb-8">
  <h2 class="max-w-xl mb-6 text-2xl font-bold dark:text-gray-400
md:text-4xl">
    {{ $product->name }}
  </h2>
  <p class="inline-block text-6xl text-gray-700 dark:text-gray-400
color:black">
    <span>Грн {{number_format($product->price)}}</span>
    @if ($product->retail_price>$product->price)
    <span class="text-base font-normal text-red-500 line-through
dark:text-gray-400">
      Грн {{number_format($product->retail_price)}}
    </span>
    @endif
```

Рисунок 2.20 – Використання HTML, Bootstrap

Наведено приклад простого коду HTML з використанням Bootstrap'a, який розміщує в колонці продукт та його вказану ціну на сайті, робота коду (рисунок 2.21):

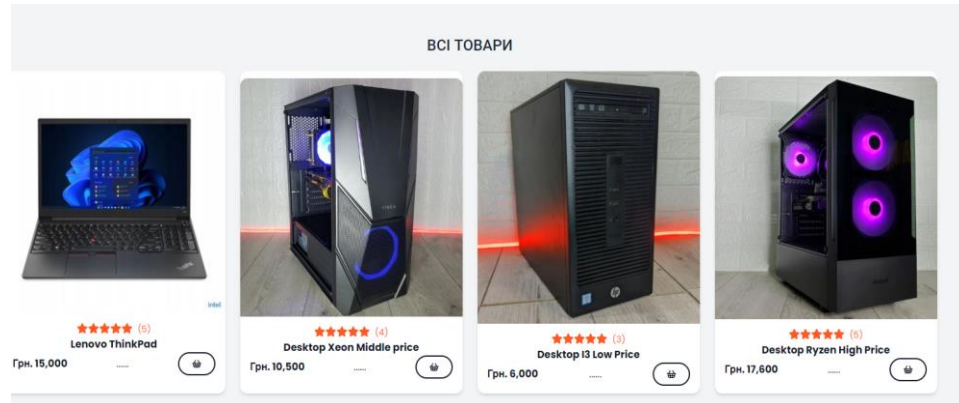


Рисунок 2.21 – Товари на сторінці

Bootstrap – це відкрита бібліотека CSS та JavaScript, що містить готові компоненти та стилі для швидкої та простої розробки інтерфейсів веб-сайтів. Bootstrap створений командою Twitter і швидко став одним із найпопулярніших інструментів для фронтенд-розробників.

Bootstrap – це потужний інструмент, який робить розробку фронтенду більш простою та ефективною. Він ідеально підходить для швидкого старту проектів або розробки привабливого та сучасного інтерфейсу для веб-додатка.

Детальніше про особливості та можливості Bootstrap:

– Система Сітки (Grid System) (рисунок 2.22):

Bootstrap використовує 12-колонкову сітку, яка дозволяє легко розташовувати елементи на сторінці:


```

<div class="container py-16">
  <div class="w-10/12 grid grid-cols-1 md:grid-cols-3 gap-6 mx-auto justify-center">
    <div class="border border-blue-600 rounded-sm px-3 py-6 flex justify-center items-center gap-5">
      
      <div>
        <h4 class="font-medium capitalize text-lg">Купуйте у нас</h4>
        <p class="text-gray-500 text-sm">Тут все дешево. От
        {{number_format(100)}}грн до {{number_format(25000)}}грн</p>
      </div>
    </div>
  </div>

```

Рисунок 2.22 – Bootstrap система сітки

Класи `col-md-`, `col-sm-`, `col-lg-` дозволяють визначити ширину елементів для різних розмірів екрану.

За приклад системи сітки на сайті взято інформаційні стенди для майбутніх покупців та користувачів сайту, які знаходяться над товарами, для того, щоб покупець одразу відчув гарантійність продукту (рисунок 2.23).

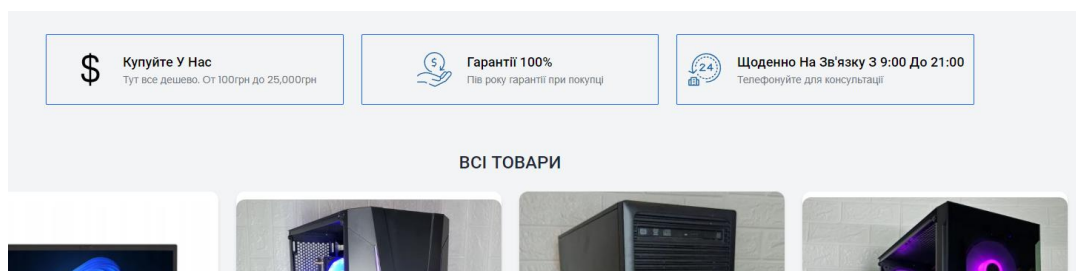


Рисунок 2.23 – Використання сітки в інтернет магазині

Прикладом використання форм вводу тексту буде реєстрація до інтернет-магазину нових користувачів (рисунок 2.24, рисунок 2.25), які в майбутньому зможуть отримувати певний відсоток знижки за купівлю в магазині, якщо вони авторизовані:

– **Navbar:** Готовий компонент для створення навігаційного меню. Можливість додавати логотип, пошукову форму та інші елементи.

Навігаційним меню в інтернет-магазині виступає панель (рисунок 2.26, рисунок 2.27), яка розташовується в `header`'і, та відповідає за зручне

переміщення по інтернет-магазину, де зручно розташовано переміщення на головну сторінку та до списку всіх товарів, а також реєстрація та вхід.

```

<div>
  <x-input-label for="name" :value="__('Ім`я')" />
  <x-text-input id="name" class="block mt-1 w-full" type="text" name="name"
:value="old('name')" required autofocus autocomplete="name" placeholder="Ваше ім`я" />
  <x-input-error :messages="$errors->get('name')" class="mt-2" />
</div>
<div>
  <x-input-label for="email" :value="__('Ваша email адреса')" />
  <x-text-input id="email" class="block mt-1 w-full" type="email" name="email"
:value="old('email')" required autocomplete="username"
placeholder="something@gmail.com" />
  <x-input-error :messages="$errors->get('email')" class="mt-2" />
</div>
<div>
  <x-input-label for="password" :value="__('Пароль')" />
  <x-text-input id="password" class="block mt-1 w-full"
type="password"
name="password"
required autocomplete="new-password" placeholder="Ваш пароль" />
  <x-input-error :messages="$errors->get('password')" class="mt-2" />
</div>
<div>
  <x-input-label for="password_confirmation" :value="__('Підтвердити пароль')" />
  <x-text-input id="password_confirmation" class="block mt-1 w-full"
type="password"
name="password_confirmation" required autocomplete="new-password"
placeholder="Підтвердити пароль" />
  <x-input-error :messages="$errors->get('password_confirmation')" class="mt-2" />
</div>
<x-primary-button>{{__('Створити новий аккаунт')}}</x-primary-button>
<div class="text-sm font-medium text-gray-900 dark:text-white">
  Вже маєте аккаунт? <a href="{{route('login')}}" class="text-blue-600
hover:underline dark:text-blue-500">Війти</a>
</div>
</form>

```

Рисунок 2.24 – Сторінка реєстрації Laravel

Рисунок 2.25 – Реєстрація за формою

```

<div class="flex md:order-2">
  <button data-collapse-toggle="navbar-search" type="button" class="inline-flex items-left p-2
w-10 h-10 justify-center text-sm text-gray-500 rounded-lg md:hidden hover:bg-gray-100
focus:outline-none focus:ring-2 focus:ring-gray-200 dark:text-gray-400 dark:hover:bg-gray-700
dark:focus:ring-gray-600" aria-controls="navbar-search" aria-expanded="false">
    <span class="sr-only">Головне меню</span>
    <svg class="w-5 h-5" aria-hidden="true" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 17 14">
      <path stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M1 1h15M1 7h15M1 13h15"/>
    </svg>
  </button>
</div>
<div class="hidden w-full md:flex md:w-auto md:order-1" id="navbar-search">
  <ul class="flex flex-col p-4 md:p-0 mt-4 font-medium border border-gray-100 rounded-lg
bg-gray-50 md:space-x-8 rtl:space-x-reverse md:flex-row md:mt-0 md:border-0 md:bg-white
dark:bg-gray-800 md:dark:bg-gray-900 dark:border-gray-700">
    <li class="text-white bg-gray-800 hover:bg-gray-900 focus:outline-none focus:ring-4
focus:ring-gray-300 font-medium rounded-full text-sm px-5 py-2.5 me-2 mb-2 dark:bg-gray-800
dark:hover:bg-gray-700 dark:focus:ring-gray-700 dark:border-gray-700">
      <a href="/" class="">Додому</a>
    </li>
    <li class="text-white bg-gray-800 hover:bg-gray-900 focus:outline-none focus:ring-4
focus:ring-gray-300 font-medium rounded-full text-sm px-5 py-2.5 me-2 mb-2 dark:bg-gray-800
dark:hover:bg-gray-700 dark:focus:ring-gray-700 dark:border-gray-700">
      <a href="{{route('shop')}}" class="">Товар</a>

```

Рисунок 2.26 – Навігаційне меню сторінки Laravel

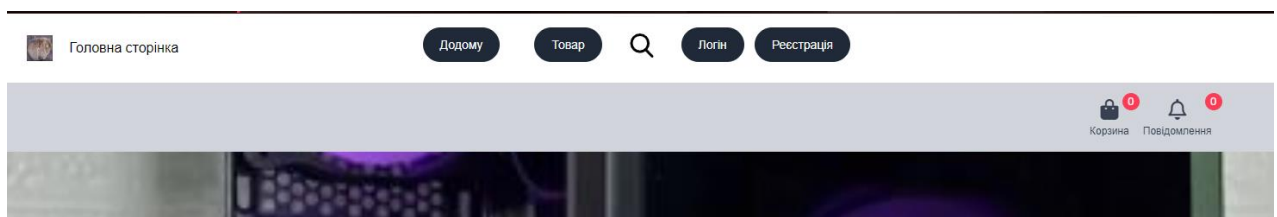


Рисунок 2.27 – Навігаційне меню на сторінці

– Breadcrumb: Вбудований компонент для створення навігації по шляху.

Навігацією інтернет-магазину слугує переміщення між домашньою сторінкою, або лицем магазину, та вибором товару, також переміщення до інтерактивного допоміжника нейромережі, при натисканні на кнопку «Товар» ми потрапляємо до списку всіх існуючих товарів (рисунок 2.28, рисунок 2.29).

```
<li class="text-white bg-gray-800 hover:bg-gray-900 focus:outline-none focus:ring-4 focus:ring-gray-300 font-medium rounded-full text-sm px-5 py-2.5 me-2 mb-2 dark:bg-gray-800 dark:hover:bg-gray-700 dark:focus:ring-gray-700 dark:border-gray-700">
  <a href="/" class="">Додому</a>
</li>
<li class="text-white bg-gray-800 hover:bg-gray-900 focus:outline-none focus:ring-4 focus:ring-gray-300 font-medium rounded-full text-sm px-5 py-2.5 me-2 mb-2 dark:bg-gray-800 dark:hover:bg-gray-700 dark:focus:ring-gray-700 dark:border-gray-700">
  <a href="{{route('shop')}}" class="">Товар</a>
</li>
```

Рисунок 2.28 – Навігація по шляху href

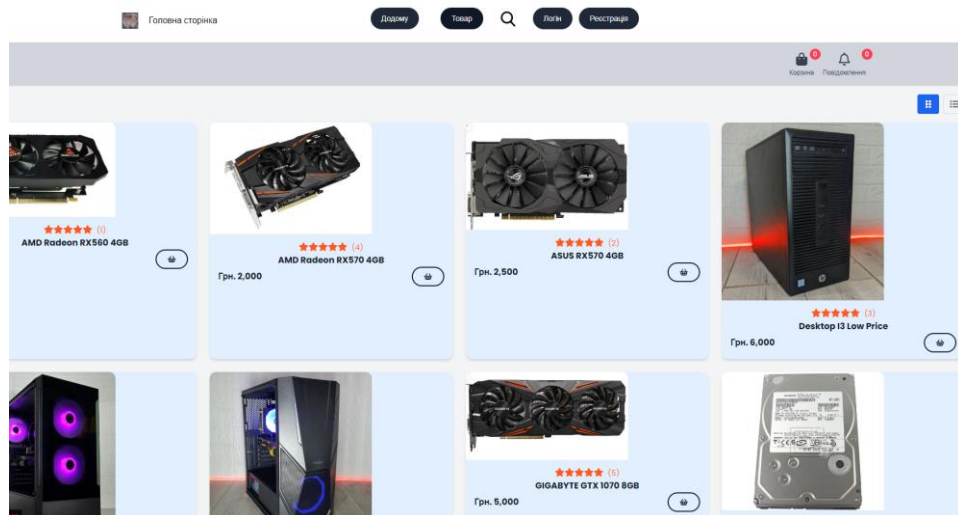


Рисунок 2.29 – Навігація по сайту

– каруселі (Carousels): Створення каруселей для відображення слайдів. Слайдер в нашому магазині розташований зручно на головній сторінці, для показу наших самих кращих товарів, або товарів які будуть зі знижкою від

розробника, також створено зручний інтерфейс для ручного переключення слайдів (рисунок 2.30, рисунок 2.31):

Bootstrap легко інтегрується з Laravel за допомогою компонента Blade. Для використання Bootstrap у проєкті Laravel, потрібно підключити стилі та скрипти Bootstrap у макеті Blade, а також використовувати класи Bootstrap для стилізації елементів інтерфейсу.

```

<div id="default-carousel" class="relative w-full" data-carousel="slide">
  <div class="relative h-56 overflow-hidden rounded-lg md:h-96">
    <!-- Item 1 -->
    <div class="hidden duration-10000 ease-in-out" data-carousel-item>
      
    </div>
    <div class="hidden duration-700 ease-in-out" data-carousel-item>
      
    </div>
    <div class="hidden duration-700 ease-in-out" data-carousel-item>
      
    </div>
    <div class="hidden duration-700 ease-in-out" data-carousel-item>
      
    </div>
    <div class="hidden duration-700 ease-in-out" data-carousel-item>
      
    </div>
    <div class="hidden duration-700 ease-in-out" data-carousel-item>
      
    </div>
  </div>
  <div class="absolute z-30 flex -translate-x-1/2 bottom-5 left-1/2 space-x-3
rtl:space-x-reverse">
    <button type="button" class="w-3 h-3 rounded-full" aria-current="true" aria-
label="Slide 1" data-carousel-slide-to="0"></button>
    <button type="button" class="w-3 h-3 rounded-full" aria-current="false" aria-
label="Slide 2" data-carousel-slide-to="1"></button>
    <button type="button" class="w-3 h-3 rounded-full" aria-current="false" aria-
label="Slide 3" data-carousel-slide-to="2"></button>
  </div>

```

Рисунок 2.30 – Слайдер Laravel

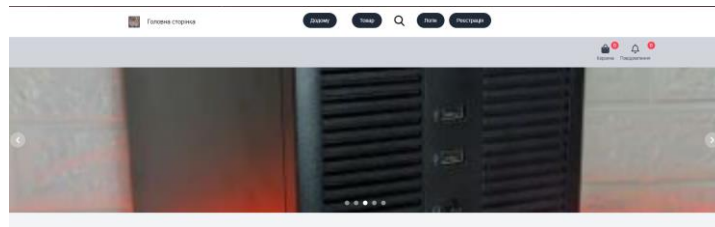


Рисунок 2.31 – Слайдер на сторінці магазину

Backend у фреймворку Laravel відповідає за обробку бізнес-логіки, взаємодію з базою даних, обробку запитів та надання відповідей фронтенду. Laravel надає ряд функцій та інструментів для зручної та продуктивної розробки на стороні сервера. Laravel забезпечує розширений інструментарій для розробки Backend-частини вебдодатків. З його допомогою можна ефективно взаємодіяти з базою даних, реалізовувати бізнес-логіку та створювати надійні та потужні серверні додатки. Laravel спрощує багато задач та дозволяє розробникам фокусуватися на логіці додатка, замість вирішення технічних деталей

Більш детально про Backend було наведено у попередніх пунктах, тому коротко приклади використання Backend'у, які не були розглянуті:

– контролери: Контролери в Laravel відповідають за обробку запитів та взаємодію з моделями та видами (views).

Контролери допомагають розділити логіку додатка на окремі компоненти (рисунок 2.32, рисунок 2.33, рисунок 2.34, рисунок 2.35).

```
public function addToCart(Request $request, $id):RedirectResponse
{
    $product = DB::table('products')->find($id);
    $oldCart = $request->session()->has('cart') ? $request->session()->get('cart') : null;
    $cart = new Cart($oldCart);
    $cart->add($product, $product->id);
    $request->session()->put('cart',$cart);
    //dd($request->session()->get('cart'));
    return back(); }

```

Рисунок 2.32 – Контролер с додаванням товару в корзину

```

<div class="mt-4 mb-4">
  <h1 class="mb-10 text-center text-2xl font-bold">Товари в корзині</h1>
  <div class="mx-auto max-w-5xl justify-center px-6 md:flex md:space-x-6 xl:px-0">
    <div class="rounded-lg md:w-2/3 mr-3 bg-gray-300">
      @if (session()->has('cart'))
        @foreach ($products as $product)
          <x-cart-products :product="$product"/>
        @endforeach
      @else
        <div class="justify-between mb-6 rounded-lg bg-teal-100 p-6 shadow-md sm:flex
sm:justify-start">
          <div class="sm:ml-4 sm:flex sm:w-full sm:justify-between">
            <div class="mt-4 flex justify-between im sm:space-y-6 sm:mt-0 sm:block sm:space-x-
6">
              <div class="flex items-center">
                <p class="text-red-600">{{__('Ви нічого не вибрали щоб придбати :()')}}</p>
              </div>
            </div>
          </div>
        </div>
      @endif
    </div>
    @if (session()->has('cart'))
      <div class="mt-6 mb-6 h-full rounded-lg border bg-teal-100 p-6 shadow-md md:mt-0
md:w-1/3">
        <div class="mb-2 flex justify-between">
          <p class="text-gray-700">Ціна товару:</p>
          <p class="text-gray-700">Грн {{number_format($totalPrice)}}</p>
        </div>
        <div class="flex justify-between">
          <!-- <p class="text-gray-700">Shipping</p>
          <p class="text-gray-700">...</p> -->
        </div>
        <hr class="my-4" />
        <div class="flex justify-between">
          <p class="text-lg font-bold mr-2">Всього до сплати:</p>
          <div class="">
            @if (session()->has('cart'))
              <p class="mb-1 text-lg font-bold color-black">Грн
{{number_format($totalPrice)}}</p>
            @else
              <p class="mb-1 text-lg font-bold color-black">Грн {{__(0.00)}}</p>
            @endif
          <!-- <p class="text-sm text-gray-700">including VAT</p> -->
        </div>
      </div>
      <button type="button" class="btn btn-danger mt-6 w-full rounded-md bg-blue-500 py-
1.5 font-medium text-blue-50 hover:bg-blue-600" id="triggerErrorBtn">Оплатити</button>
      <!-- <button class="mt-6 w-full rounded-md bg-blue-500 py-1.5 font-medium text-
blue-50 hover:bg-blue-600" button id="mvButton">Оплатити</button> -->
    </div>
  </div>

```

Рисунок 2.33 – Сторінка для виведення корзини с товаром

Якщо товар не було обрано та корзина пуста:

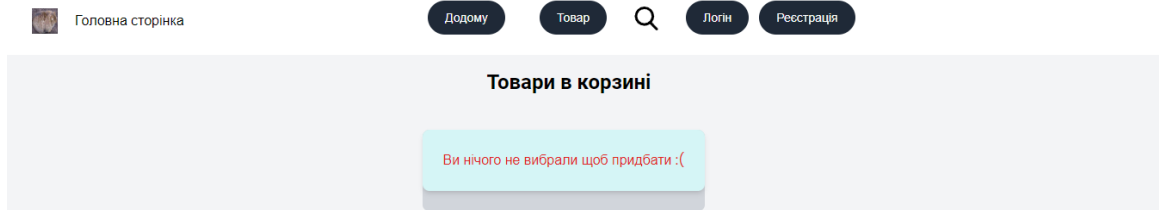


Рисунок 2.34 – Пуста корзина

Якщо товар був обраний:

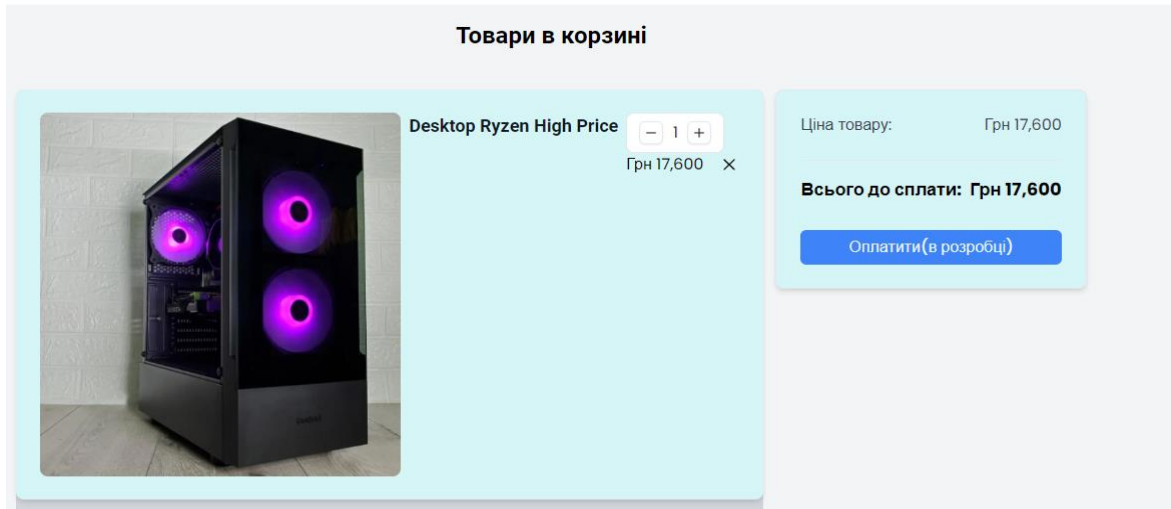


Рисунок 2.35 – Обраний товар в корзині

Також існує можливість обирати декілька товарів (рисунок 2.36):

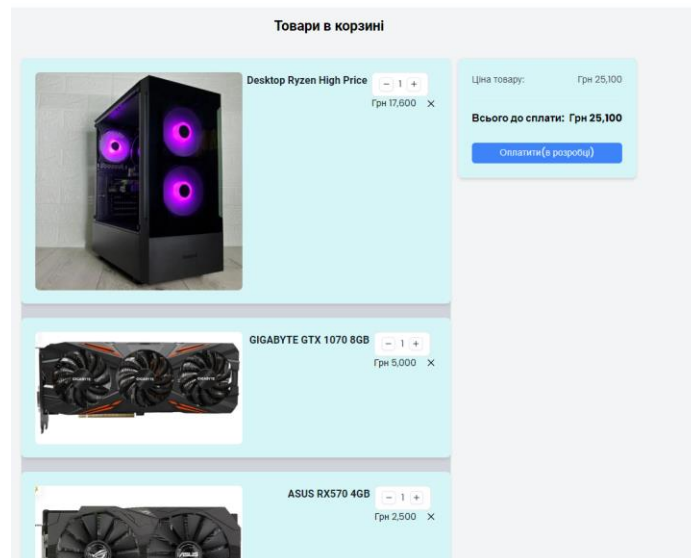


Рисунок 2.36 – Декілька товарів в корзині

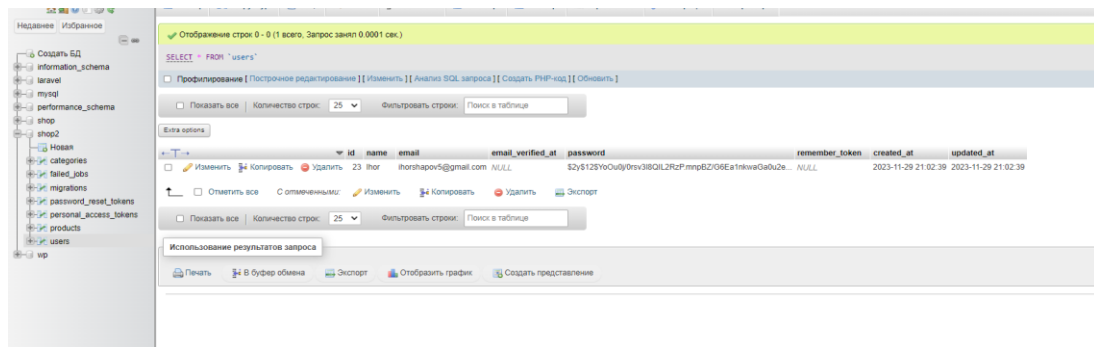
– міграції: Міграції в Laravel дозволяють створювати та змінювати схему бази даних в коді (рисунок 2.37, рисунок 2.38).

Це дозволяє легко керувати структурою бази даних та веденням версій.

```
public function up(): void
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('email')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Рисунок 2.37 – Міграція користувача

Зареєстрований користувач в базі даних:



The screenshot shows the PhpMyAdmin interface with a table named 'users' selected. The table structure and a single record are visible.

id	name	email	email_verified_at	password	remember_token	created_at	updated_at
23	igor	igorsharov5@gmail.com	NULL	\$2y\$12\$YoOuy0rsv38OIL2RzPmmp8Z/06EaTnkwaGa9z6...	NULL	2023-11-29 21:02:39	2023-11-29 21:02:39

Рисунок 2.38 – Зареєстрований користувач в PhpMyAdmin

– Middleware: використовуються для обробки запитів перед тим, як вони дійдуть до контролера (рисунок 2.39, рисунок 2.40, рисунок 2.41).

Middleware може виконувати різні завдання, такі як перевірка авторизації, обробка локалізації тощо, зміна користувачем його особистих даних.

```

public function update(ProfileUpdateRequest $request): RedirectResponse
{
    $request->user()->fill($request->validated());
    if ($request->user()->isDirty('email')) {
        $request->user()->email_verified_at = null;
    }
    $request->user()->save();
    return Redirect::route('profile.edit')->with('status', 'profile-updated');
}

```

Рисунок 2.39 – Middleware оновлення інформації користувача

Приклад оновлення імені користувача:

Рисунок 2.40 – Зміна імені користувача

Також змінене ім'я в базі даних:

id	name	email	email_verified_at	password	remember_token	created_at	updated_at
23	IhorShapovalov	ihorshapov5@gmail.com	NULL	S2y51281YoOu0j0rsv3l8QIL2RzP.mmpBZ/G6Ea1nkwaGa0u2e...	wAmEjruuBSNNu3XtRvWpS0F7RKZGVBl7eEek3bpwmKLLKUL...	2023-11-29 21:02:39	2023-12-03 01:37:39

Рисунок 2.41 – Зміна імені в базі даних

– авторизація та Аутентифікація:

Laravel надає готовий функціонал для реалізації авторизації та аутентифікації користувачів (рисунок 2.42, рисунок 2.43, рисунок 2.44).

Можливість визначення різних рівнів доступу для користувачів.

```

public function create(): View
{
    return view('auth.login');
}
public function store(LoginRequest $request): RedirectResponse
{
    $request->authenticate();
    $request->session()->regenerate();
    return redirect()->intended(RouteServiceProvider::HOME);
}
public function destroy(Request $request): RedirectResponse
{
    Auth::guard('web')->logout();
    $request->session()->invalidate();
    $request->session()->regenerateToken();
    return redirect('/');
}

```

Рисунок 2.42 – Авторизація та аутентифікація користувача

Наведено приклад зареєстрованого користувача, який авторизується у систему.

Рисунок 2.43 – Авторизація

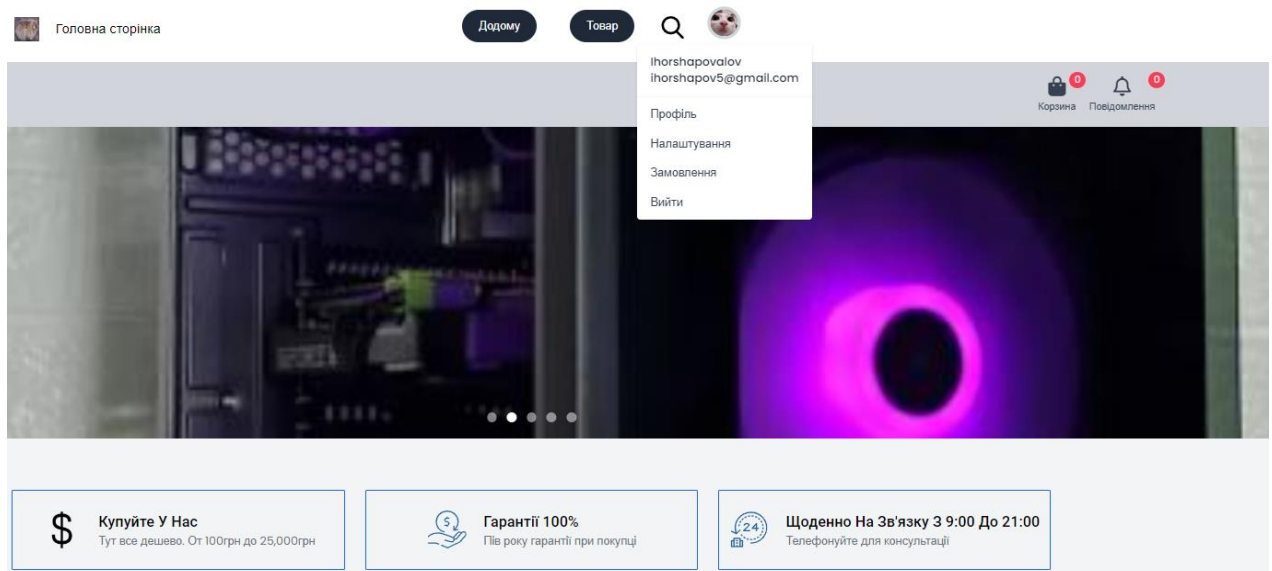


Рисунок 2.44 – Авторизований аккаунт



В налаштуваннях свого профілю є варіант зміни будь-яких внесених даних, зміна імені, паролю, пошти, а також видалення аккаунту.

– API та Ресурсні Контролери:

Laravel дозволяє легко створювати API за допомогою ресурсних контролерів.

Ресурсні контролери надають шаблонні методи для створення, читання, оновлення та видалення ресурсів.

Laravel забезпечує розширений інструментарій для розробки Backend-частини вебдодатків. З його допомогою можна ефективно взаємодіяти з базою даних, реалізовувати бізнес-логіку та створювати надійні та потужні серверні додатки. Laravel спрощує багато задач та дозволяє розробникам фокусуватися на логіці додатка, замість вирішення технічних деталей. В нашому випадку використання API в Laravel задля введення нейронної мережі як інтерактивного хелпера стане новинкою для більшої кількості інтернет-магазинів, які існують на ринку, та чимось приємним для лінивого покупця, який не відмовився би від допомоги штучного інтелекту.

Головна сторінка Додому Товар  

Profile Information

Update your account's profile information and email address.

Name

Email

SAVE

Update Password

Ensure your account is using a long, random password to stay secure.

Current Password

New Password

Confirm Password

SAVE

Delete Account

Once your account is deleted, all of its resources and data will be permanently deleted. Before deleting your account, please download any data or information that you wish to retain.

DELETE ACCOUNT

Рисунок 2.45 – Налаштування користувача

3 ІНТЕГРАЦІЯ НЕЙРОННОЇ МЕРЕЖІ В ІНТЕРНЕТ-МАГАЗИН

3.1 Характеристика, тип та можливості нейромережі

Llama 2 — це сімейство найсучасніших великих мовних моделей із відкритим доступом, випущених Meta сьогодні, і великим задоволенням є повністю підтримати запуск із комплексною інтеграцією в Hugging Face. Llama 2 випускається з відкритою ліцензією для спільноти та доступна для комерційного використання. До сьогодні розроблюється код для моделі, та підготовляються та випускаються моделі. Випуск Llama 2 представляє сімейство попередньо підготовлених і точно налаштованих LLM з параметрами в діапазоні від 7B до 70B (7B, 13B, 70B). Попередньо підготовлені моделі мають значні покращення порівняно з моделями Llama 1, включаючи навчання на 40% більшій кількості токенів, набагато більшу довжину контексту (4 тисячі токенів) і використання уваги згрупованого запиту для швидкого висновку моделі 70B!

Однак найбільш захоплюючою частиною цього випуску є налаштовані моделі (Llama 2-Chat), оптимізовані для діалогових додатків із використанням Reinforcement Learning from Human Feedback (RLHF). У широкому діапазоні тестів корисності та безпеки моделі Llama 2-Chat працюють краще, ніж більшість відкритих моделей, і досягають порівнянної продуктивності з ChatGPT згідно з оцінками людей. Більш докладніше інформація представлена в статті [5]. Цю модель було обрано для реалізації інтеграції нейромережі до інтернет-магазину, вона має всі необхідні навчені токени, для того, щоб стати інтерактивним допоміжним інструментом для покупця.

Першим кроком у пізнанні Llama є завантаження та налаштування на особистому сервері, як виконати всі необхідні умови для завантаження можна знайти в [6].

Після того, як було виконано всі необхідні кроки у встановленні неймережі було розгорнено її на особистому сервері, та отримано функціональний інтерфейс (рисунок 3.1).

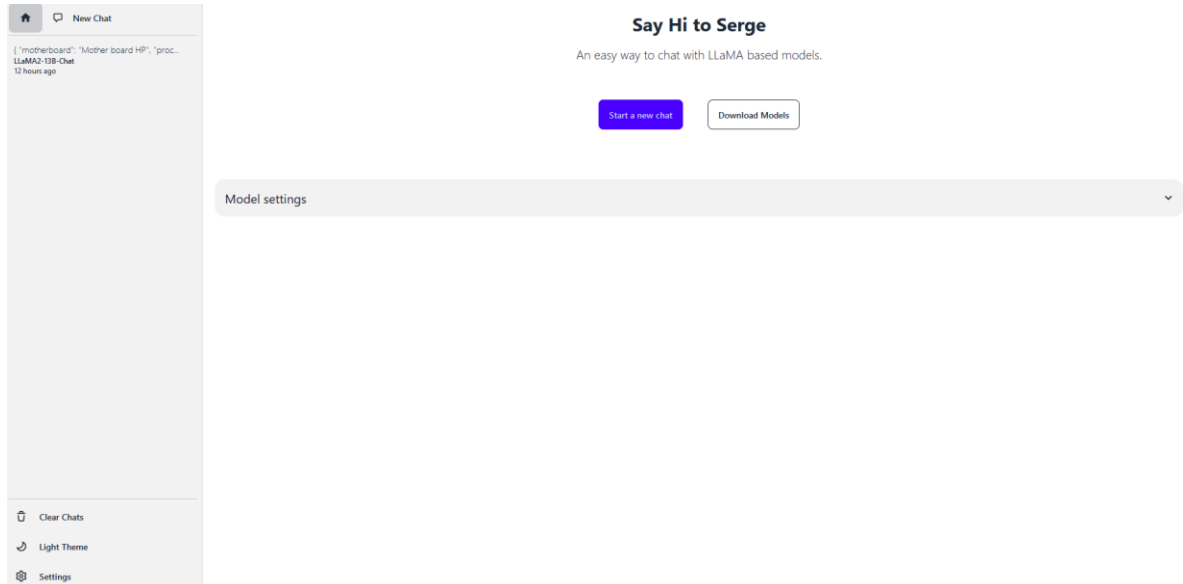


Рисунок 3.1 – Інтерфейс моделі Llama

Для взаємодії з Llama було створено окремий чат, в якому задано параметри, головним із яких є параметр “response only json” – цей параметр змушує неймережу відповідати лише у форматі json, це дозволяє найбільш точно та швидше обробляти інформацію отриману від неймережі (рисунок 3.2, рисунок 3.3):

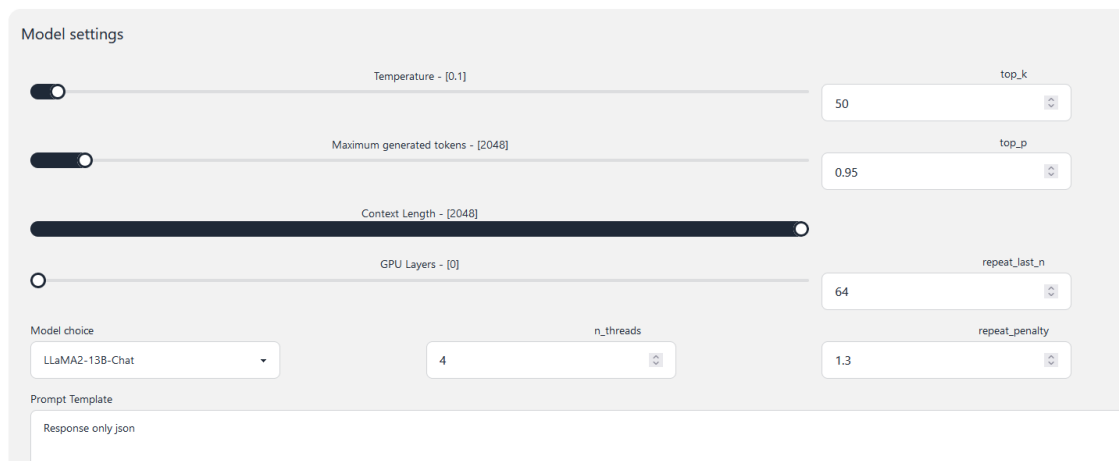


Рисунок 3.2 – Налаштування моделі Llama

12/10/2023, 1:01:54 AM
Response only json

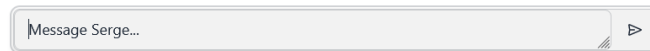


Рисунок 3.3 – Створений чат для взаємодії с нейромережею

Наступний крок це інтеграція нейромережі до інтернет-магазину. Інтеграція проводиться через отримання запиту post через Javascript (рисунок 3.4).

```

$(#neuralNetwork).click(function () {
  var input1 = "" + $('#text-input1').val() + "";
  var input2 = "";
  var url = 'http://91.224.44.103:8008/api/chat/e6e0996a-d89d-4be6-a2d3-d88532a70550/question?prompt=' + encodeURIComponent(input1)
+encodeURIComponent(input2);
  var data = { input1: input1, input2: input2 };
  var response = 0
  $.post(url, function(data) {
    response = data.split('choices')[1].split('text')[1].split("index")[0].split(":")
    [1].split(",")[0];

    $('#resultNeural').text(data.split('choices')[1].split('text')[1].split("index")[0].split(":")
    [1].split(",")[0]);
  });
});

```

Рисунок 3.4 – Взаємодія з нейромережею

Var input 1 – Введення перших вхідних даних користувачем

Var input 2 – Введення других даних системою

Var url – Посилання на опрацювання запиту нейромережею

Var data – Отримання даних

\$.post(url function(data) {response} – Виведення даних у результат на сторінці інтернет магазину

Для взаємодії с нейромережею було створено окрему сторінку інтернет-магазину, для більш зручного користування нейромережею та вводу текстових запитів для опрацювання нейромережі (рисунок 3.5, рисунок 3.6).

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset = "UTF-8">
  <meta name = "viewport" content= "width=device-width, initial-scale=1.0">
  <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
</head>
<body>
  <label for="text-input1">Введіть ваш запит сюди:</label>
  <input type="text" id="text-input1" name="input1" required>
  <button id="neuralNetwork">Натисніть на цю кнопку для запиту-> Запит
</button>
  <div id="resultNeural"> Результат запиту </div>

```

Рисунок 3.5 – Сторінка для взаємодії з нейромережею

Введіть ваш запит сюди: Натисніть на цю кнопку для запиту-> Запит

Результат запиту

Рисунок 3.6 – Текстове поле запиту

Після отримання запиту та проведення опрацювання нейромережею запиту, було отримано відповідь у форматі Json (рисунок 3.7, рисунок 3.8):



Рисунок 3.7 – Відповідь нейромережі

Та відповідно результат на сторінці:

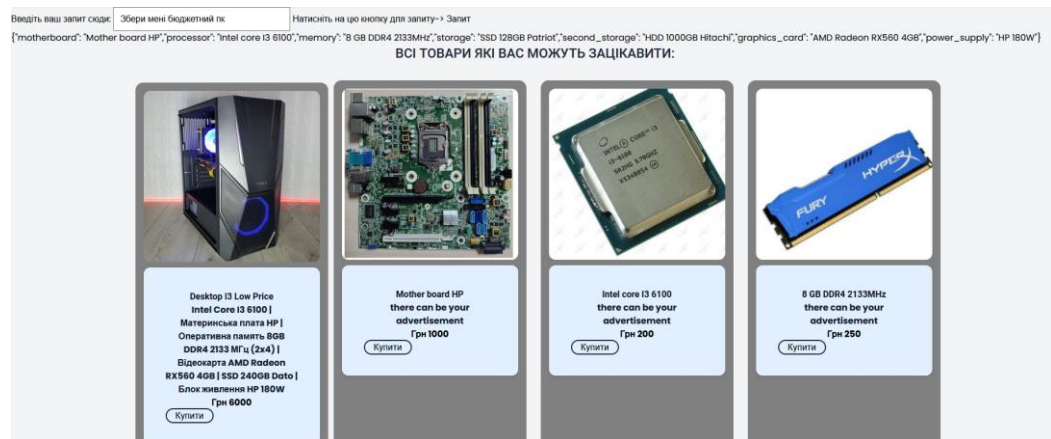


Рисунок 3.8 – Результат запиту на сторінці

Після успішного опрацювання запити, нейромережа передає данні результату на сторінку, після опрацювання результату покупцеві підбираються товари, які можуть входити в категорію «Бюджетні» та можуть зацікавити покупця, також пропонується вже зібраний комп'ютер який такою входить в категорію бюджетний. Для ефективною результативності було створено ще 2 запити, для категорії «Високобюджетний» (рисунок 3.10) та «Середньобюджетний» (рисунок 3.9):

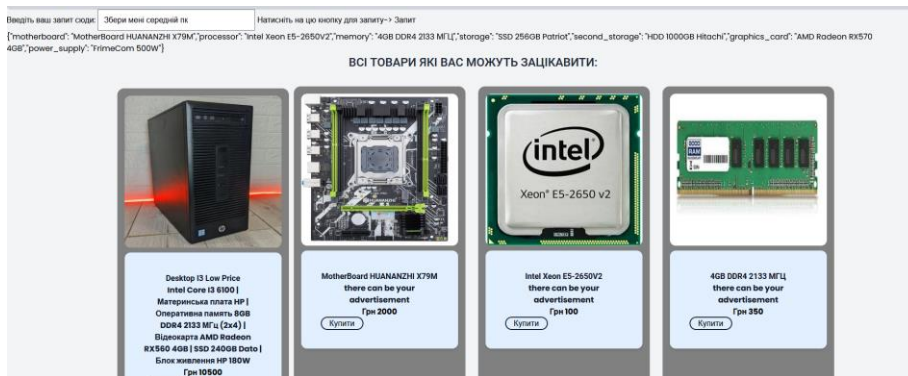


Рисунок 3.9 – Середньобюджетний ПК

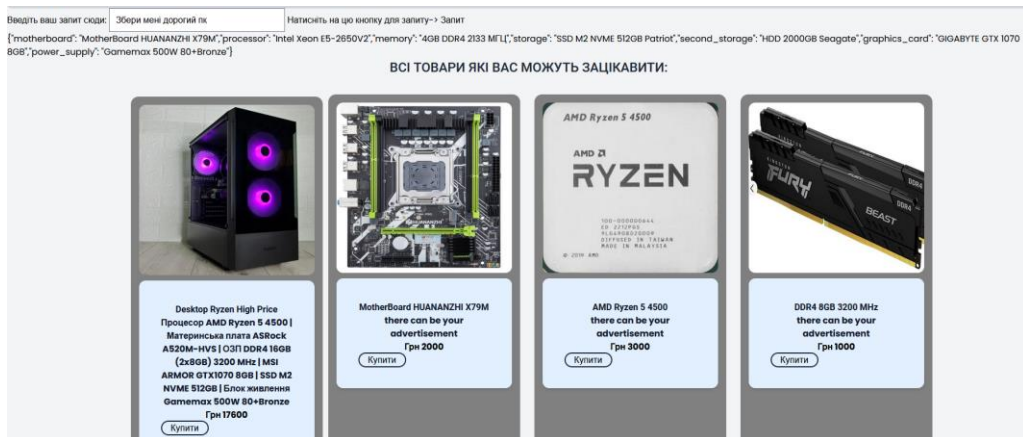


Рисунок 3.10 – Високобюджетний ПК

Ця це нова модель, яка створена компанією Meta, вона по цей час навчається та отримує нові данні для запитів, опрацювання, тому варто зазначити, що вона все ще не ідеальний прототип нейромережі, як GPT-3.5,4,5, але Ця вже має базовий набір функціоналу, який вже може використовуватись для роботи простого інтернет-магазині, а також, легкий спосіб налаштування, швидку взаємодію з інтерфейсом, що робить її привабливим варіантом. Але в цій моделі є головний недолік – модель не має стану, в якому вона «пам'ятає» попередні фрагменти розмови, користувач повинен завжди надавати їй весь контекст, щоб розмова могла продовжуватися. Ось чому довжина контексту є дуже важливим параметром, який потрібно максимізувати, оскільки він дозволяє довше розмовляти та використовувати більші обсяги інформації.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було створено інтернет-магазин с продажу комплектуючих та вже готових робочих станцій. Для створення інтернет-магазину було обрано відомий фреймворк Laravel 10.x. Головними перевагами Laravel перед більшістю фреймворків стала його зручна структура коду; Artisan, який допомагає автоматизувати більшість завдань; зручна міграція бази даних; створення сідерів; вбудована система шаблонів, яка дозволяє розробляти красиві та ефективні сторінки; Middleware, який дозволяє виконувати фільтрацію HTTP-запитів. Офіційна документація Laravel є детальною та добре написаною, що полегшує вивчення та використання фреймворку [7]. Використання Composer дозволяє легко управляти залежностями та встановлювати різноманітні пакети для розширення функціоналу додатку.

Цікавим доповненням стало використання нейромережі Serge с моделлю Llama, для створення інтернет-магазину. Модель Llama, має не досконалий навчений інтелект, але велику привабливість у своєму розвитку та зручності використання. Для клієнтів, які не мають бажання обирати самостійно комплектуючі чи готову станцію в інтернет-магазині, застосування моделі Llama стане зручною допомогою – вона виконує функцію поміжника, який лише по одному запиту може видати клієнтові необхідні комплектуючі. Так за запитом «Збери мені дешевий ПК», Llama видає всі можливі варіанти, які можуть його задовільнити покупця. Це лише невеликий приклад можливостей нейромережі під час користування інтернет-магазином – є широкий спектр функцій, які може виконувати штучний інтелект: голосовий пошук, пошук по фото, завантаження нових даних до бази, пошук необхідної інформації з інтернету тощо. Нейромережа ще має довгий спектр розвитку, в майбутньому інтеграція може стати дуже корисною новинкою для більшості продавців.

ПЕРЕЛІК ПОСИЛАНЬ

1. Можливості використання штучних нейронних мереж. Транспортні системи та технології перевезень. URL: <http://tstt.diit.edu.ua/article/view/35983> (дата звернення: 25.05.2023).
2. Нейромережа – що це таке, як працює та навіщо потрібна. Termin.in.ua. URL: <https://termin.in.ua/neuromerezha/> (дата звернення: 04.06.2023).
3. ChatGpt. URL: <https://chat.openai.com/> (дата звернення: 10.06.2023).
4. GitHub - AUTOMATIC1111/stable-diffusion-webui: Stable Diffusion web UI. GitHub. URL: <http://surl.li/ocybz> (дата звернення: 20.08.2023).
5. Paper page - Llama 2: Open Foundation and Fine-Tuned Chat Models. Hugging Face – The AI community building the future. URL: <https://huggingface.co/papers/2307.09288> (дата звернення: 10.11.2023).
6. Llama 2 - Meta AI. AI at Meta. URL: <https://ai.meta.com/llama/>. (дата звернення: 13.11.2023).
7. Laravel - The PHP Framework For Web Artisans. Laravel - The PHP Framework For Web Artisans. URL: <https://laravel.com/docs/10.x/readme> (дата звернення: 1.09.2023).
8. Using phpMyAdmin Domain.com. URL: <http://surl.li/ocybp> (дата звернення: 30.08.2023).
9. The Executive Guide to Neural Networks and Deep Learning. Infopulse. URL: <https://www.infopulse.com/blog/the-executive-guide-to-neural-networks-and-deep-learning-for-businesses> (дата звернення: 09.10.2023).
10. Stauffer M. Up & Running: A Framework for Building Modern PHP Apps. 3rd ed. California: O'Reilly Media, 2023. 559 p.
11. Aggarwal C. Neural Networks and Deep Learning: A Textbook 2nd ed. Berlin: Springer, 2023. 553 p.