Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Журенка Сергія Сергійовича*
(ПІБ)

академічної групи *121-19-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованого додатку з продажу картин з використанням HTML, CSS, JavaScript, фреймворку Bootstrap та бібліотеки React*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*ст. викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19-2* *Журенка Сергія Сергійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованого додатку з продажу картин з використанням HTML, CSS, JavaScript, фреймворку Bootstrap та бібліотеки React*

затверджена наказом ректора НТУ «ДП» від *16 . 05 .2023 р.* № *350 -с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Спірінцев В.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Журенко С.С*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *11 .06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 61 с., 26 рис., 3 дод., 28 джерела.

Об'єкт розробки: веб-орієнтований додаток для комерційної діяльності.

Мета кваліфікаційної роботи: створення веб-орієнтованого додатку на основі бібліотеки React для взаємодії потенційного клієнта з предметами для продажу.

У вступі до кваліфікаційної роботи описується сучасний стан проблеми, яку буде розглянуто, конкретно визначається мета дослідження та область її застосування, надається аргументація актуальності теми, а також формулюється завдання, яке потрібно вирішити в роботі.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі розглянуто наявні варіанти реалізації задачі, обрано платформи для розробки, визначено необхідні інструменти, описано принцип роботи програми, створено структуру сайту та розроблено дизайн продукту.

У розділі, що присвячений економіці, було визначено складність процесу розробки інформаційної системи, проведено розрахунок часових витрат на створення продукту та обчислено вартість годин праці, необхідних для розробки системи.

Практичне застосування полягає у створенні онлайн-сервісу, який забезпечує можливість перегляду та придбання товарів, забезпечуючи захист приватності користувача.

Цей програмний продукт є актуальним через великий попит на дистанційний спосіб придбання товарів, який дозволяє клієнтам економити зусилля та час, також актуальність полягає в тому, що це полегшує рекламу для підприємства.

Список ключових слів: КОМП`ЮТЕР, ІНТЕРНЕТ, ДОДАТОК, БАЗА ДАНИХ, ДИЗАЙН, REACT JS, NODE JS.

ABSTRACT

Explanatory note: 61 p., 26 pict., 3 apps., 28 sources.

Object of development: web-oriented application for commercial activities.

Meta-qualification work: creation of a web-oriented add-on based on the React library for customer interaction with items for sale.

The introduction to the qualification work describes the current state of the problem that will take place, specifically the purpose of the research and the scope of its application, provides arguments for the relevance of the topic, and also formulates the task to be completed in the work.

In the first section, the subject area is analyzed, the relevance of the task and the purpose of the development is determined, the task statement is formulated, which has requirements for software implementation, technologies and software tools.

In the second section, the available options for the implementation of the task are presented, platforms for development are chosen, sufficient tools are determined, the principle of the program is described, the site structure is created, and the product design is developed.

In the section devoted to economics, the complexity of the process of developing an information system was determined, the time spent on creating a product was calculated, and the cost of labor hours was calculated, some for the development of the system.

A practical application is in the creation of an online service that provides the ability to view and purchase goods, ensuring the protection of user privacy.

This software product is relevant because of the great demand for a remote method of purchasing goods that allows customers to save effort and time, as well as the relevance of the conflict in that it facilitates advertising for the enterprise.

Keywords list: COMPUTER, INTERNET, APPLICATION, DATA BASE, DESIGN, ReactJs, Node.Js.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

HTML - HyperText Markup Language; TCP - Transmission Control Protocol;

HTTP - HyperText Transfer Protocol;

CSS – Cascading Style Sheets;

IP - Internet Protocol;

IDE - Integrated development environment; URL - Uniform Resource Locator;

DB – Data base;

ВСТУП

Дана кваліфікаційна робота присвячена створенню інтернет-магазину за допомогою бібліотеки React Js. Основною метою є побудова простої системи, щоб користувачі могли купувати картини та ілюстрації за допомогою фреймворку React Js. Інтернет-магазини стали дуже популярними, особливо під час карантину, коли люди не хочуть виходити з дому. Відмінність інтернет-магазину полягає у тому, що інфраструктура реалізована програмно, тому його володіння є вигідним і не вимагає великих витрат. Володіти інтернет-магазином може будь-яка компанія, від маленьких стартапів до великих корпорацій, тому розробка власного інтернет-магазину є важливим кроком для бізнесу в сучасному світі. Технології React Js дозволяють створити швидкий та зручний інтерфейс для користувачів, що підвищує ефективність роботи магазину та забезпечує задоволення користувачів від процесу купівлі.

Ця кваліфікаційна робота досліджує важливість наявності веб-присутності для бізнесу, оскільки багато людей шукають продукти та послуги в Інтернеті перед покупкою. Якщо бізнес не має веб-сайту, то це може призвести до втрати потенційних клієнтів, які звертаються до конкурентів з веб-присутністю. Отже, метою роботи є вивчення та вдосконалення засобів для створення веб-орієнтованих інформаційних систем з надійною та стабільною роботою, дружнім та інтуїтивним інтерфейсом та спроможністю системи адаптуватися до різних пристроїв. Результатом такої системи буде забезпечення бізнесу можливості досягати потенційних клієнтів через веб-присутність, забезпечення легкого користування користувачами та збільшення потенційного доходу бізнесу.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної області

Предметна галузь кваліфікаційної роботи це електронна комерція.

Електронна комерція (e-commerce) - це процес купівлі та продажу товарів і послуг, який здійснюється за допомогою електронних засобів зв'язку, зокрема Інтернету. У електронній комерції використовуються електронні платежі, електронні майданчики, онлайн-магазини та інші цифрові технології для проведення торговельних операцій.

Основні аспекти електронної комерції включають:

– онлайн-торгівля: це процес продажу товарів та послуг через Інтернет. Бізнеси можуть мати власні веб-сайти або використовувати платформи електронної комерції для представлення своїх продуктів та прийому замовлень в онлайн-режимі;

– електронні платежі: електронна комерція використовує електронні платіжні системи для здійснення оплати за товари та послуги. Це можуть бути кредитні картки, електронні гаманці, банківські перекази та інші електронні методи оплати;

– електронна доставка: після здійснення покупки через електронну комерцію, товари можуть бути доставлені клієнту за допомогою кур'єрських служб або поштових служб. Також існують цифрові товари, які можна завантажити або отримати електронною поштою;

– електронний маркетинг: електронна комерція включає в себе використання інтернет-маркетингу, такого як реклама в пошукових системах, соціальні медіа, електронна пошта та інші канали, для залучення клієнтів і просування товарів і послуг;

– безпека: електронна комерція також звертає велику увагу на захист особистої інформації та безпеку трансакцій. Вона використовує шифрування

даних, протоколи безпеки і сертифікати SSL для захисту конфіденційності та недопущення несанкціонованого доступу до персональних даних клієнтів;

– глобальний доступ: електронна комерція дозволяє підприємствам розширювати свої ринки та досягати клієнтів по всьому світу. Клієнти можуть здійснювати покупки з будь-якого місця з доступом до Інтернету, що дозволяє розширити базу покупців і збільшити обсяги продажів;

– аналітика і персоналізація: електронна комерція дозволяє збирати дані про покупців і їх поведінку, що дозволяє аналізувати та розуміти їх потреби та уподобання. Це дозволяє підприємствам персоналізувати пропозиції, рекомендації та маркетингові акції для кожного клієнта.

Електронна комерція стала значною частиною сучасного бізнесу, надаючи підприємствам нові можливості для продажу своїх товарів і послуг, а споживачам - зручний та доступний спосіб здійснення покупок.

Кваліфікаційна робота стосується саме онлайн-торгівлі.Предметом розробки є інтернет-магазин.

Інтернет-магазин - це електронна платформа, де ви можете здійснювати покупки через Інтернет. Ви можете відвідати цей магазин за допомогою веб-браузера або мобільного додатка і переглянути різні товари та послуги, які вони пропонують. Після вибору товарів ви можете оформити замовлення та здійснити платіж в режимі онлайн. Після цього інтернет-магазин організує доставку товару до вказаної вами адреси. Основна перевага інтернет-магазинів полягає в тому, що ви можете здійснювати покупки зручно, швидко і безпечно, не виходячи зі свого дому. Інтернет-магазини є основою електроної комерції.

1.1.1. Область електроної комерції

Область застосування інтернет-магазину включає різноманітні сектори бізнесу, включаючи:

– роздрібна торгівля: Інтернет-магазини широко використовуються в роздрібній торгівлі для продажу різних товарів, таких як одяг, електроніка,

косметика, спортивні товари, меблі та інше. Вони дозволяють покупцям переглядати асортимент, порівнювати ціни, зробити замовлення та отримати доставку безпосередньо до дому;

– бізнес-послуги: Крім фізичних товарів, інтернет-магазини можуть бути використані для продажу різних бізнес-послуг, таких як консультування, веб-дизайн, копірайтинг, програмне забезпечення та інше. Це дозволяє клієнтам замовляти та оплачувати послуги онлайн;

– харчова промисловість: Інтернет-магазини також можуть бути використані для продажу продуктів харчування, напоїв та інших продуктів харчування. Клієнти можуть замовляти продукти через Інтернет та отримувати їх доставку безпосередньо до своїх дверей;

– туризм та гостинності: У сфері туризму та гостинності інтернет-магазини використовуються для продажу туристичних пакетів, готельних послуг, авіаквитків, екскурсій та інших подорожей пов'язаних послуг. Клієнти можуть легко знайти та забронювати свої поїздки через онлайн-системи, що пропонуються в інтернет-магазинах.

Загалом, інтернет-магазини мають широке застосування у різних галузях бізнесу та надають зручну та ефективну платформу для здійснення онлайн-торгівлі та залучення клієнтів з усього світу.

1.2. Призначення розробки та область застосування

Призначенням розробки є створення веб-сервісу для продажу картин для підтримки митців та залучення клієнтів.

Основні аспекти та значення такого веб-сервісу включають:

1. Підтримка митців: Розроблений веб-сервіс створює можливість для митців представляти свої твори широкій аудиторії та залучати покупців з усього світу. Це дає митцям можливість отримати визнання, продати свої роботи та заробити на своєму мистецтві, створюючи нові можливості для їхнього професійного росту.

2. Поширення мистецтва: Веб-сервіс дозволяє митцям розповсюджувати свої витвори широкій аудиторії. Це допомагає збільшити обізнаність про їхнє мистецтво, привернути нових шанувальників та покупців з різних куточків світу. Шляхом представлення картин на веб-сервісі, митці можуть залучити увагу до своїх творів та розширити свою глобальну аудиторію.

3. Створення доступної торгівельної платформи: Веб-сервіс надає зручну платформу для покупців, де вони можуть переглядати, вибирати та купувати картини від різних митців. Це дозволяє покупцям відкрити для себе нових талановитих художників та придбати унікальні твори мистецтва безпосередньо від авторів. Крім того, веб-сервіс може забезпечувати безпеку операцій та захист інформації покупців.

В цілому, розробка веб-сервісу для продажу картин для підтримки митців та поширення їх витворів світом має на меті створити місце зустрічі митців та шанувальників мистецтва, сприяючи розвитку мистецтва, комерційному успіху митців та збагаченню культурного ландшафту.

Розробка інтернет-магазину має на меті створення функціональної та привабливої електронної платформи, яка дозволяє бізнесам продавати свої товари та послуги в Інтернеті. Основним призначенням розробки інтернет-магазину є створення зручного та ефективного середовища для електронної комерції, де клієнти можуть здійснювати покупки онлайн.

Областю застосування проекту є електронна комерція в сегменті мистецтва і декору, перегляд та продаж витворів мистецтва.

Метою розробки є інтернет-магазин з базою даних з товарами та користувачами розроблений за допомогою технологій ReactJs, NodeJs, задля комерційної діяльності та поширення ідеї діджиталізації бізнесу.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

– освітня програма спеціальності 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;

– завдання на кваліфікаційну роботу на тему «розробка інтернет-магазину на основі бібліотеки React js».

1.4. Постановка завдання

Завданням даної роботи є побудова зрозумілої системи для купівлі картин, за допомогою бібліотеки React js. Основними характеристиками розробки повинні бути:

– зрозумілий інтерфейс, щоб користувачу було зручно і він мав змогу швидко знайти та придбати потрібний йому товар;

– швидка реєстрація, вхід та вихід з облікового запису, кожний користувач має свій кошик;

– швидке завантаження сторінки, гарна продуктивність;

– швидкий запис інформації в бд, це робиться через реєстрацію користувача та додавання або видалення товарів за допомогою адмін панелі;

– зрозуміла та проста адмін панель для додавання або видалення товарів, зміна опису, ціни товару;

– розділення ролей (user та admin), user може обирати та придбати товари, admin може редагувати інформацію про товари, додавати та видаляти їх.

Поставлена задача може бути досягнута при виконанні наступних умов:

– моделювання дизайну, процес створення візуальної концепції, компонування елементів, вибір кольорів, шрифтів та інших візуальних елементів, які допомагають створити привабливий та функціональний інтерфейс;

– вибір ide залежно від мови програмування та інших переваг;

– написання програмного коду, найбільш важлива частина, код повинен бути зрозумілим та спроможнім для модифікації;

– налаштування бази даних, створення бд та прив’язка до програмного забезпечення, налаштування зв’язків між таблицями.

Кінцевим результатом має бути інтернет-магазин в якому покупець може швидко увійти в свій профіль, обрати та купити будь-яку картину.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

1. Зрозумілий інтерфейс для звичайного користувача та адміністратора: Веб-сервіс повинен мати інтуїтивно зрозумілий та дружній інтерфейс, що спрощує навігацію та взаємодію для звичайних користувачів. Це означає, що користувачі повинні легко знаходити картини, мати можливість переглядати детальну інформацію про них, додавати їх у кошик та здійснювати покупки. Адміністраторам, зі свого боку, повинно бути доступне управління контентом, налаштуваннями продуктів та управління замовленнями.

2. Швидкий та простий процес покупки: Процес покупки повинен бути швидким та простим для користувачів. Користувачі повинні мати можливість додавати бажані товари до кошика, переглядати вміст кошика, вибирати спосіб доставки та спосіб оплати, заповнювати необхідні дані для доставки та оформлювати замовлення з легкістю та зручністю.

3. Збереження бажаних покупок в особистому кошику: Користувачі повинні мати можливість зберігати бажані товари в особистому кошику, щоб вони могли повернутися до них пізніше, додавати або видаляти товари з кошика, а також зберігати замовлення для майбутньої оплати та відстеження.

Крім цього, важливо враховувати принципи безпеки, які включають захист особистих даних користувачів, безпечність оплати та захист від шахрайства. Надійна інфраструктура, така як шифрування даних, застосування безпечних протоколів зв'язку та механізми аутентифікації, також є важливими аспектами для успішної реалізації цих функціональних вимог.

1.5.2. Вимоги до інформаційної безпеки

Система оброблення покупок має деякі основні компоненти для забезпечення безпеки та розподілу доступу:

1. Реєстрація користувачів: Кожен користувач, який бажає здійснювати покупки, повинен зареєструватись на сайті. Під час реєстрації, користувачі надають свої особисті дані, включаючи електронну пошту. Ці дані зберігаються в базі даних.

2. Web Token: Після успішної реєстрації кожному користувачеві надається унікальний web token. Цей токен використовується для ідентифікації користувача під час кожного запиту до сервера. Він може бути переданий у заголовку або запиті HTTP, щоб сервер міг перевірити та ідентифікувати користувача.

3. Безпека особистих даних: Для захисту особистих даних користувачів, таких як пароль, використовувується шифрування, а саме web-

token при збереженні цих даних у базі даних. Крім того забезпечено належні заходи безпеки на серверному та клієнтському рівнях, щоб запобігти несанкціонованому доступу до особистої інформації.

4. Ролі користувачів: система має концепцію ролей, яка дозволяє розподіляти доступ до різних функцій в залежності від ролі користувача. На сайті є дві основні ролі: користувач та адміністратор. Користувачі мають обмежений доступ та можуть здійснювати покупки, переглядати свої замовлення тощо. Адміністратори, натомість, мають розширений доступ, включаючи можливість додавати товари, керувати замовленнями та взаємодіяти з системою управління.

5. База даних: Всі дані, пов'язані з користувачами, ролями та іншою інформацією, зберігаються в базі даних. База даних добре захищена, використовуються належні механізми безпеки, такі як шифрування, контроль доступу та забезпечення цілісності даних.

6. Адмін панель: Адміністраторам надається спеціальна адміністративна панель, яка надає їм можливість керувати та взаємодіяти з різними аспектами системи. Це може включати додавання та видалення товарів, перегляд замовлень, керування користувачами тощо. Доступ до адмін панелі обмежений, доступ мають лише адміністратори.

7. Заходи безпеки: Для забезпечення безпеки системи, крім вищезазначених пунктів, також використовується захист від хакерських атак, таких як SQL-ін'єкція та перехоплення сесій. Застосовується нормалізація вхідних даних.

Ці компоненти забезпечують захист особистих даних користувачів, розподіляють доступ до функцій в системі та забезпечують безпеку взаємодії з базою даних.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для нормального функціонування веб-орієнтованої інформаційної системи, повинні виконуватися певні вимоги до технічних засобів.

Для клієнтської частини:

– процесор з тактовою частотою не менш ніж 1,5 ГГц;

– оперативна пам'ять не менш ніж 2GB.

Для серверної частини:

– процесор з тактовою частотою не менш ніж 2,5 ГГц.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

Іншими словами, для ефективної роботи системи рекомендується мати клієнтську частину з процесором, що має тактову частоту не менше вказаної, а також оперативною пам'яттю не менше вказаного обсягу. Для серверної частини рекомендується мати процесор з тактовою частотою не менше вказаної. Відповідно до цих рекомендацій програмний виріб буде працювати з відповідною надійністю, швидкістю обробки даних і безпекою, які вимагаються замовником.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати веб-орієнтована підсистема, відповідало наступним вимогам:

– операційна система Unix, Linux, Microsoft Windows XP/7/8/10;

– браузер Інтернет (Microsoft Internet Explorer, MozillaFireFox, Opera, Google Chrome);

Веб-орієнтована інформаційна система повинна бути реалізована на мові програмування JavaScript, а саме на бібліотеці ReactJs, з використанням HTML, CSS для верстання сторінок, з сервером NodeJs та базою даних PostgreSQL.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної роботи має бути веб-орієнтований додаток для комерційної діяльності, а саме для продажу картин, та для звернення уваги потенційних клієнтів.

Сайт повинен працювати на будь-яких системах, пристроях, запускатись в різних браузерах, тобто бути універсальним, щоб охопити як можна більше потенційних покупців.

Призначення:

– зручний інтерфейс для купівлі-продажу товару(картин);

– зображення картин та детальний опис для огляду;

– залучення покупців за допомогою привабливого UI;

– зручність розташування та налаштування товару за допомогою адмін-панелі.

2.2. Опис застосованих математичних методів

Розробка, проектування додатку було проведено без застосування математичних методів, були використані лише прості арифметичні операції для пагінації, та поелементного виводу товарів (по 3-4 елементи в ряд).

2.3. Опис використаної архітектури та шаблонів проектування

В кваліфікаційній роботі використана трирівнева клієнт-серверна архітектура.

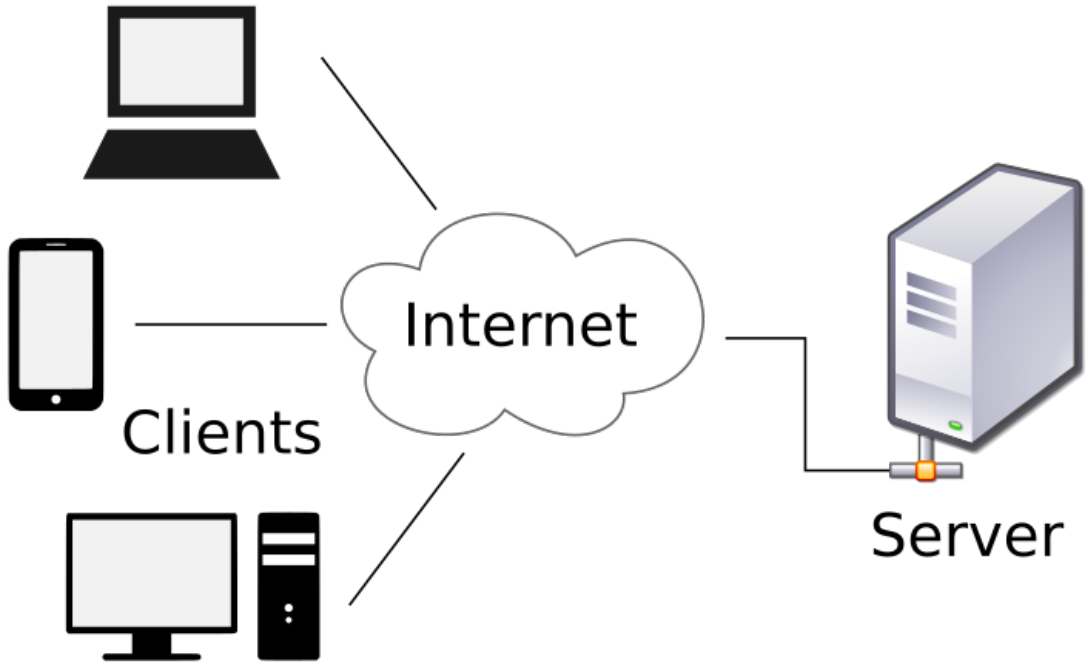


Рис 2.1. Схема роботи трирівневої клієнт-серверної архітектури

Трирівнева клієнт-серверна архітектура (англ. Three-tier client-server architecture) - це модель розподіленої системи, в якій функції програмного забезпечення розподілені між трьома рівнями: клієнтським, середнім і серверним рівнями. Клієнтський рівень: Цей рівень включає клієнтські пристрої (наприклад, комп'ютери, смартфони, планшети), на які користувачі взаємодіють з програмним забезпеченням. Клієнтський рівень надає інтерфейс для користувачів, що дозволяє виконувати операції та виконувати запити до середнього рівня. Середній рівень: Цей рівень виконує бізнес-логіку та обробку даних. Він виконує запити від клієнтського рівня, обробляє їх, виконує потрібні операції та взаємодіє з серверним рівнем для доступу до даних або збереження змін. Серверний рівень: Цей рівень містить сервери, які забезпечують доступ до даних та зберігання інформації. Він може включати бази даних, файлові сервери, сервери додатків тощо. Серверний рівень відповідає забезпеченню доступності даних та виконання запитів, які виходять із середнього рівня. Трирівнева архітектура дозволяє розділити логіку програмного забезпечення на окремі компоненти, що спрощує розробку, супроводження та масштабування

системи. Клієнтський рівень може бути реалізований у вигляді веб-інтерфейсу або десктопної добавки, середній рівень може бути реалізований як додатковий рівень логіки на сервері, а серверний рівень може використовувати різні технології та серверні ресурси в залежності від потреб системи.

Шаблоном проектування був використаний MVC Шаблон.

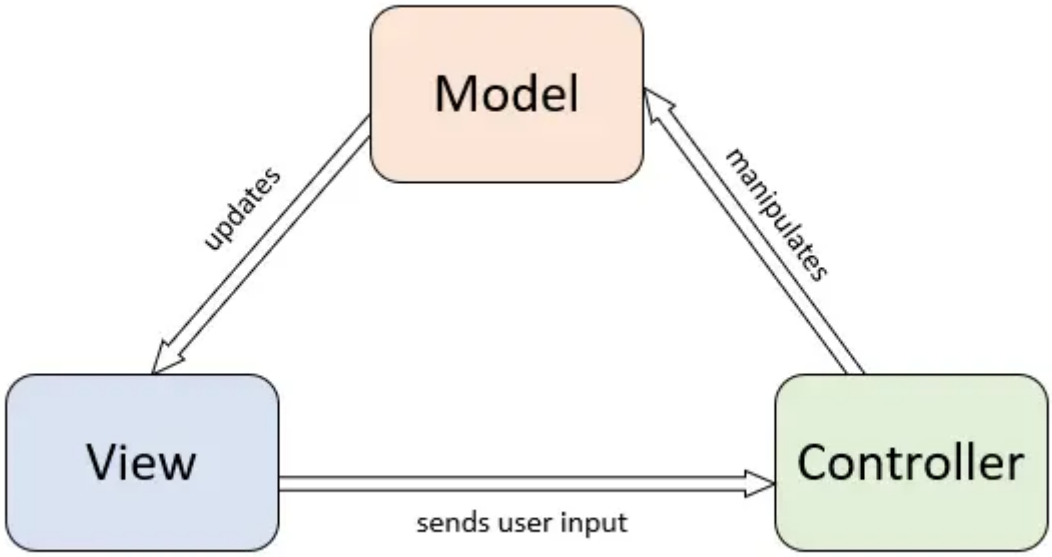


Рис 2.2. Схема роботи MVC шаблону проектування.

Модель-Вид-Контролер (Model-View-Controller, MVC) є одним з найпоширеніших шаблонів проектування для розробки програмного забезпечення. Він дозволяє розділити компоненти системи на три основні частини: модель, вид та контролер.

1. Модель (Model): Модель представляє дані та бізнес-логіку системи. Вона відповідає за збереження та обробку даних, а також за виконання операцій, пов'язаних з бізнес-правилами. Модель може включати класи або об'єкти, що представляють дані, та методи для доступу до цих даних та їх обробки.

2. Вид (View): Вид відповідає за представлення даних користувачу. Він відображає інформацію з моделі у зручному для користувача форматі. Вид може бути представленим у вигляді веб-сторінок, графічних інтерфейсів або

будь-яких інших способів візуалізації даних. Він зазвичай отримує доступ до даних через модель і може відправляти повідомлення контролеру для здійснення дій.

3. Контролер (Controller): Контролер служить посередником між моделлю та видом. Він обробляє запити користувача, виконує потрібні дії та взаємодіє з моделлю та видом. Контролер приймає вхідні дані від користувача, обробляє їх, виконує необхідні операції з моделлю та визначає, який вид буде показаний користувачу з оновленими даними.

Основна ідея MVC полягає в тому, що кожна з трьох компонентів (модель, вид та контролер) виконує свої відповідальності та має обмежений доступ до решти компонентів. Це дозволяє забезпечити розділення логіки, полегшує тестування, підтримку та масштабування системи.

Застосування MVC у проектуванні інтернет-магазинів дозволяє зберегти дані (модель) незалежними від їх представлення (вид) та логіки взаємодії з користувачем (контролер). Це сприяє більшій гнучкості, легкості розробки та підтримки системи.

2.4. Опис використаних технологій та мов програмування

Даний веб-додаток був розроблений за допомогою наступних технологій:

– React.js;
– React-router-dom;
– JavaScript;
– Node.js;
– CSS3;
– HTML5;
– PostgreSQL;
– Express.js;
– JWT.

Детальний опис кожної технології зі стеку:

React.js - це JavaScript-бібліотека з відкритим вихідним кодом, яка дозволяє розробникам створювати інтерактивні користувацькі інтерфейси для веб-сайтів і додатків. Цей фреймворк дозволяє ефективно керувати станом компонентів та автоматизувати процес створення і оновлення відображення елементів на сторінці. React дозволяє розробникам створювати складні додатки з багатим функціоналом та високою швидкодією завдяки використанню віртуального DOM та оптимізації рендерингу. Крім того, React має велику спільноту, що надає безліч додаткових компонентів та інструментів для полегшення розробки.

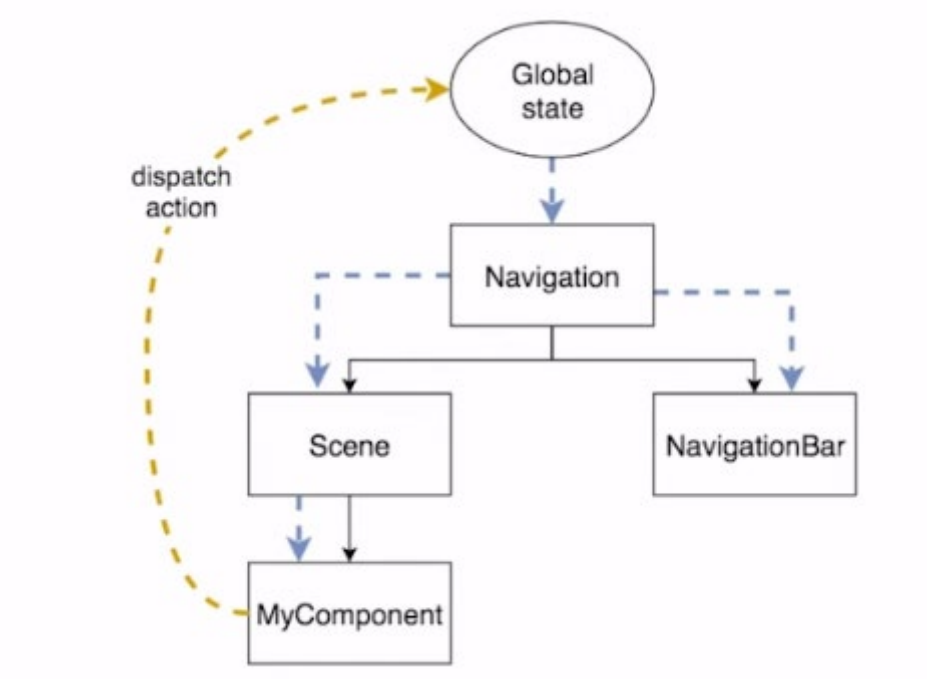


Рис 2.3. Схема реалізації react компонента

React Router DOM - це набір бібліотек для React, який надає можливість створювати односторінкові додатки зі змінним URL-адресою (routing).

Роутинг (routing) відповідає за те, яка сторінка або компонент повинні бути відображені в залежності від URL-адреси. React Router DOM дозволяє вам описувати маршрути вашого додатку з урахуванням URL-адрес, а також поведінки при переходах між маршрутами.

React Router DOM надає кілька компонентів, які можна використовувати для опису маршрутів, таких як BrowserRouter, Route, Switch та Link. Компонент

BrowserRouter дозволяє налаштувати роутинг для вашого додатку, Route визначає шлях до компонента, який повинен бути відображений, Switch використовується для відображення тільки одного маршруту за раз, а Link створює посилання на інші маршрути.

React Router DOM також дозволяє вам передавати параметри в URL-адресі, щоб динамічно відображати компоненти в залежності від URL. Також можна встановлювати захист маршрутів від несанкціонованого доступу, використовуючи компоненти PrivateRoute та Redirect.

Узагальнюючи, React Router DOM - це потужний інструмент для створення роутингу в додатках React, який дозволяє легко налаштовувати маршрути та забезпечувати їх безпеку.

JavaScript - це високорівнева, інтерпретована мова програмування, яка зазвичай використовується для розробки веб-додатків та інтерфейсів користувача. Вперше JavaScript був створений компанією Netscape в 1995 році, як скриптова мова для взаємодії з користувачем на веб-сторінках.

JavaScript зазвичай використовується для створення динамічних веб-сайтів та односторінкових додатків, а також для розробки додатків на боковій клієнтській стороні (client-side) та серверній стороні (server-side). Ця мова може бути використана для створення різноманітних функцій, таких як анімація, валідація форм, обробка подій, маніпулювання HTML-документами, взаємодія з сервером та інше.

JavaScript має багато фреймворків та бібліотек, таких як React, Angular, Vue.js, jQuery та інші, що дозволяють значно спростити та прискорити процес розробки веб-додатків. JavaScript також підтримується всіма сучасними веб-браузерами та є однією з найбільш популярних мов програмування в світі.

Node.js - це відкрите серверне середовище, яке дозволяє виконувати JavaScript код на стороні сервера. Node.js заснований на движку V8, який розробляється Google для виконання JavaScript в браузері Chrome. Однак, Node.js використовує V8 для виконання JavaScript на сервері.

Node.js відрізняється від традиційних серверних технологій, таких як PHP чи Ruby on Rails, тим, що він працює на подіях (event-driven) та не блокує виконання інших операцій до завершення попередніх. Це дозволяє досягти високої продуктивності та масштабованості, особливо в операціях вводу-виводу (I/O operations), таких як робота з мережею чи зчитування та запис до файлів.

Node.js також забезпечує широкі можливості для створення серверних додатків, включаючи розробку API, веб-серверів, чат-ботів, платіжних систем та інших. Із зростанням популярності Node.js з'явилося багато різноманітних фреймворків, таких як Express.js, Koa.js, Hapi.js, Meteor.js та інші, які допомагають розробникам швидко створювати високоякісні серверні додатки.

CSS3 Cascading Style Sheets (CSS) - мови стилів, яка використовується для оформлення веб-сторінок. CSS3 включає в себе нові функції, які покращують зовнішній вигляд веб-сторінок і роблять їх більш інтерактивними.

CSS3 дозволяє веб-розробникам створювати більш привабливі та інтерактивні веб-сторінки, що може підвищити їх ефективність та привабливість для користувачів

HTML5 (HyperText Markup Language 5) - це остання версія стандарту мови розмітки гіпертексту, яка використовується для створення веб-сторінок. HTML5 була розроблена для покращення функціональності, доступності та сумісності зі сучасними веб-браузерами.

PostgreSQL є однією з найпопулярніших відкритих реляційних баз даних з відкритим вихідним кодом. Він пропонує багато продвинутих функцій, які роблять його популярним серед розробників програмного забезпечення.

Деякі з функцій PostgreSQL включають в себе:

Розширення SQL: PostgreSQL має багато функцій, які не є стандартними для SQL, такі як можливість визначення функцій, що повертають рядки, як аргументи.

Підтримка JSON: PostgreSQL має вбудовану підтримку для JSON, що дозволяє працювати з даними в форматі JSON.

Широкі можливості розширення: PostgreSQL має підтримку плагінів, що дозволяє розширити функціональність бази даних, додавши в неї нові можливості.

Транзакції: PostgreSQL має підтримку транзакцій, що дозволяє забезпечити цілісність даних, навіть якщо сталася помилка.

Складні запити: PostgreSQL має багато функцій, що дозволяють виконувати складні запити, такі як операції злиття (JOIN), підзапити та групування даних.

Відмінна продуктивність: PostgreSQL має високу продуктивність та може обробляти великі обсяги даних з високою швидкістю.

Узагальнюючи, PostgreSQL є потужним та надійним реляційним базою даних, що забезпечує широкі можливості для розробників програмного забезпечення, які працюють з даними.

Express.js - це веб-фреймворк для Node.js, призначений для розробки веб-додатків та API. Він забезпечує високошвидкісний та мінімалістичний спосіб розробки веб-додатків на Node.js.

Express забезпечує ряд утиліт та методів, які допомагають в зручному створенні роутів, middleware-функцій, обробників помилок та інших складових веб-додатку. Фреймворк дозволяє зручно налаштовувати сервер, використовуючи маршрутизацію, шаблонізацію та інші рішення.

Один з основних принципів Express - це middleware, які дозволяють додавати логіку обробки запитів перед тим, як вони будуть передані обробникам запитів. Middleware можуть використовуватись для реалізації авторизації, обробки валідації запитів, логування та багатьох інших задач.

Express є дуже популярним веб-фреймворком для Node.js, тому він має велику спільноту користувачів та розширень, що дозволяє ефективно розробляти веб-додатки та API.

JWT (або JSON Web Token) - це стандарт відкритого формату для передачі інформації в формі токенів між двома сторонами. JWT зазвичай

використовується для аутентифікації та авторизації користувачів в різноманітних веб-додатках і API.

JWT складається з трьох частин: заголовку, пейлоаду та підпису. Заголовок містить інформацію про тип токена та алгоритм шифрування. Пейлоад містить корисну інформацію, таку як ідентифікатор користувача або роль користувача. Підпис генерується на основі заголовка та пейлоаду з використанням секретного ключа, що зберігається на сервері.

JWT має декілька переваг в порівнянні з традиційними методами аутентифікації, такими як cookies або сесії. Він є безстанним, тобто сервер не потребує зберігати стан аутентифікації на своєму боці, що полегшує масштабування додатків. Крім того, він може бути використаний для автентифікації користувачів в різних мікросервісах.

Однак, використання JWT також має свої недоліки. Оскільки токен зберігається на клієнті, він може бути вкрадений з метою атаки на стороні клієнта (наприклад, Cross-Site Scripting або XSS атака). Також важливо зберігати секретний ключ на сервері в надійному місці, щоб запобігти його витоку.

2.5. Опис структури системи та алгоритмів її функціювання

Кваліфікаційну роботу розроблено на основі стеку PERN – Postgresql, Express.js, React.js, Node.js.

Структуру веб-сайту обрано ієрархічну, тобто панель зверху(меню) на головній сторінці, де можна переміщатись між сторінками, і з кожної сторінки можна переміститись на головну.

Взагалі то існує безліч різних структур веб-сайтів, але основні з них - це:

– ієрархічна структура - це структура, у якій веб-сторінки організовані відповідно до їхньої важливості і залежності одна від одної. На вершині ієрархії зазвичай знаходиться головна сторінка, яка містить посилання

на основні розділи сайту, які в свою чергу можуть містити посилання на підрозділи і окремі сторінки;

– лінійна структура - це структура, у якій кожна сторінка містить посилання на наступну і попередню сторінки. Ця структура зручна для сайтів з невеликою кількістю сторінок і простим змістом;

– сіткова структура - це структура, у якій веб-сторінки організовані в таблиці з різними розділами і посиланнями. Ця структура зручна для сайтів з багатьма розділами і підрозділами;

– подібна до дерева структура - це структура, у якій веб-сторінки організовані як дерево з батьківськими, дочірніми і внучатими сторінками. Ця структура зручна для сайтів з багатьма взаємопов'язаними розділами і підрозділами.

Кожна з цих структур може бути комбінована або змінена відповідно до потреб користувачів та мети сайту.

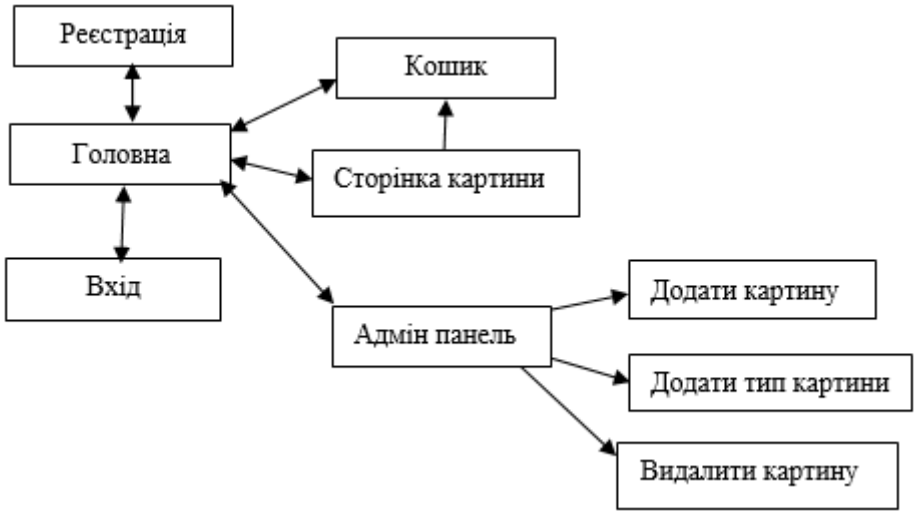


Рис 2.4. Схема структури сайту

Схема складається з п’яти основних сторінок, (шости для адміністратора), це:

– Головна.

– Реєстрація.

– Вхід.
– Сторінка окремої картини.
– Кошик.
– Адмін-панель.

Як видно зі схеми на деякі сторінки не можливо попасти з будь-якого місця на сайті, але можливо попасти з головної сторінки, бо вона найвища в ієрархії, нюансом є сторінка адмін-панелі, бо вона доступна лише для ролі адміністратора.

Щодо структури папок, файлів, всередині сайту, вирішено поділити основний каталог на три підкаталоги: client, server та зображення для наповнення сайту:

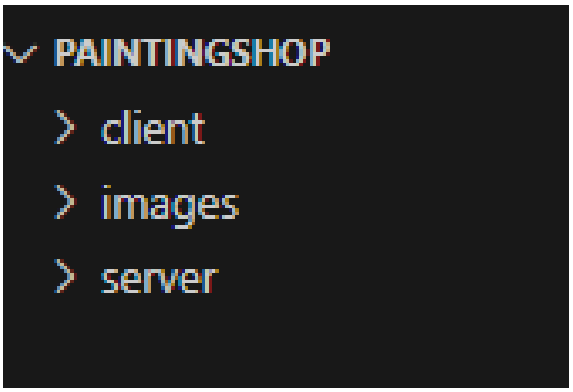
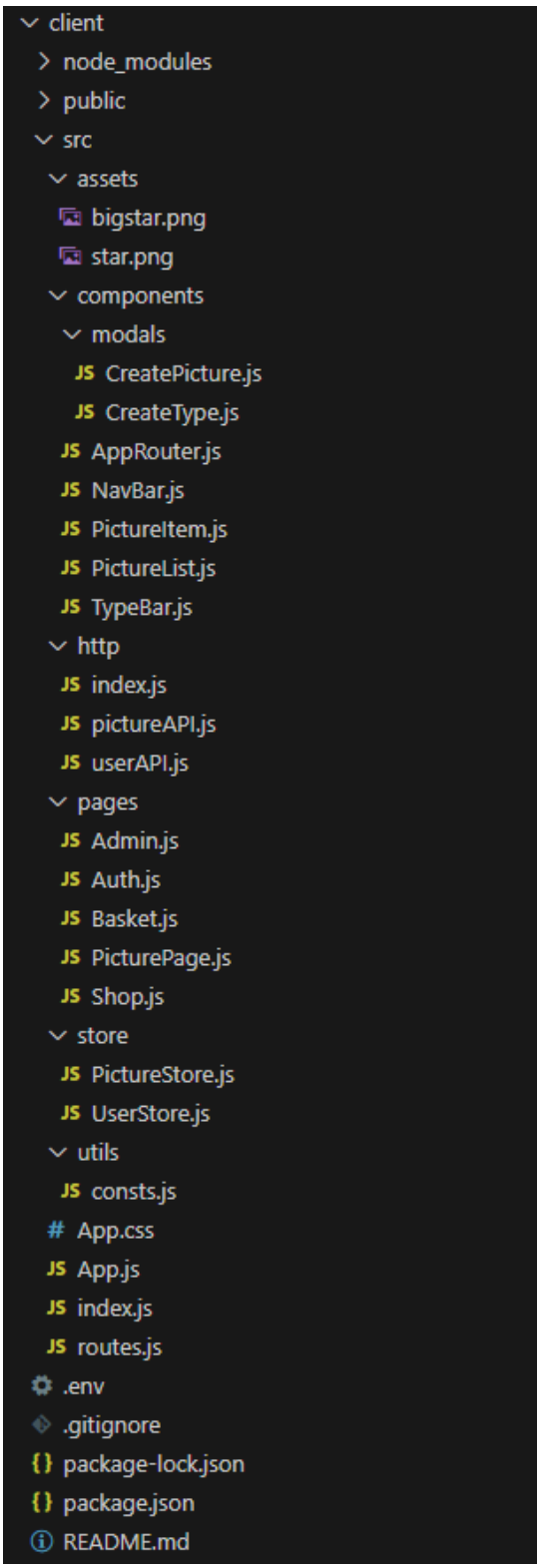


Рис 2.5. Структура проекту

Клієнтську та серверну частину роздроблено на багато підчастин, кожна з яких відповідає або описує свій процесс. Використана структура MVC – Model, View, Controll. Це дуже популярна структура, бо вона дуже чітко передає сенс побудови сайту. Model – модель бази даних, в проекті за підключення до бази даних та її створення відповідає два різних файла, що належать до каталогу server.



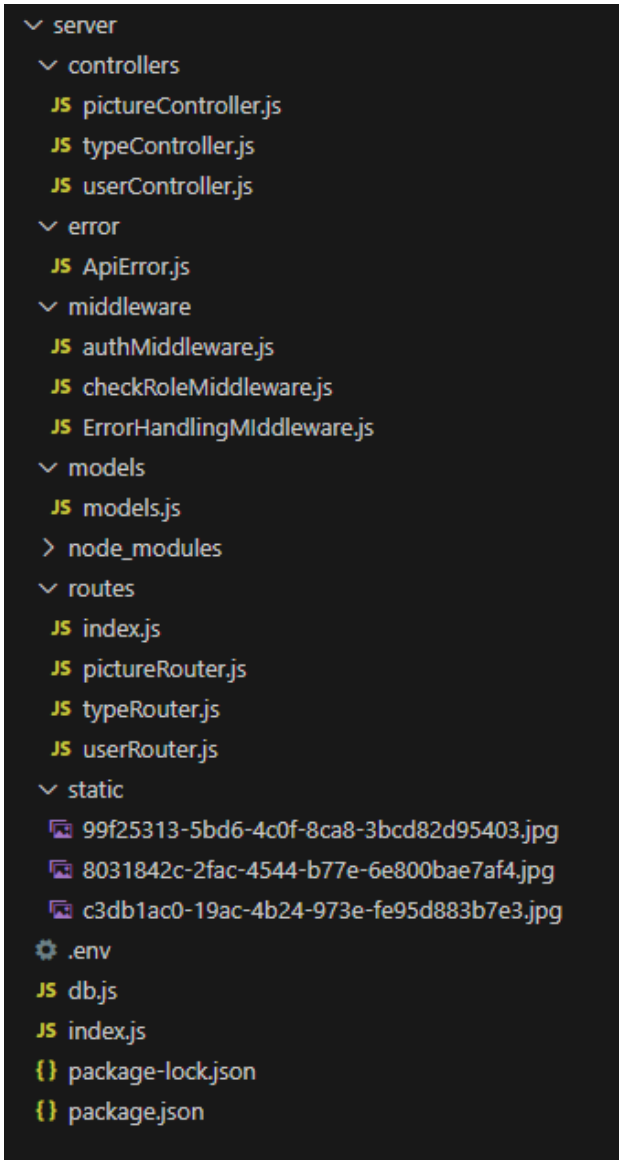
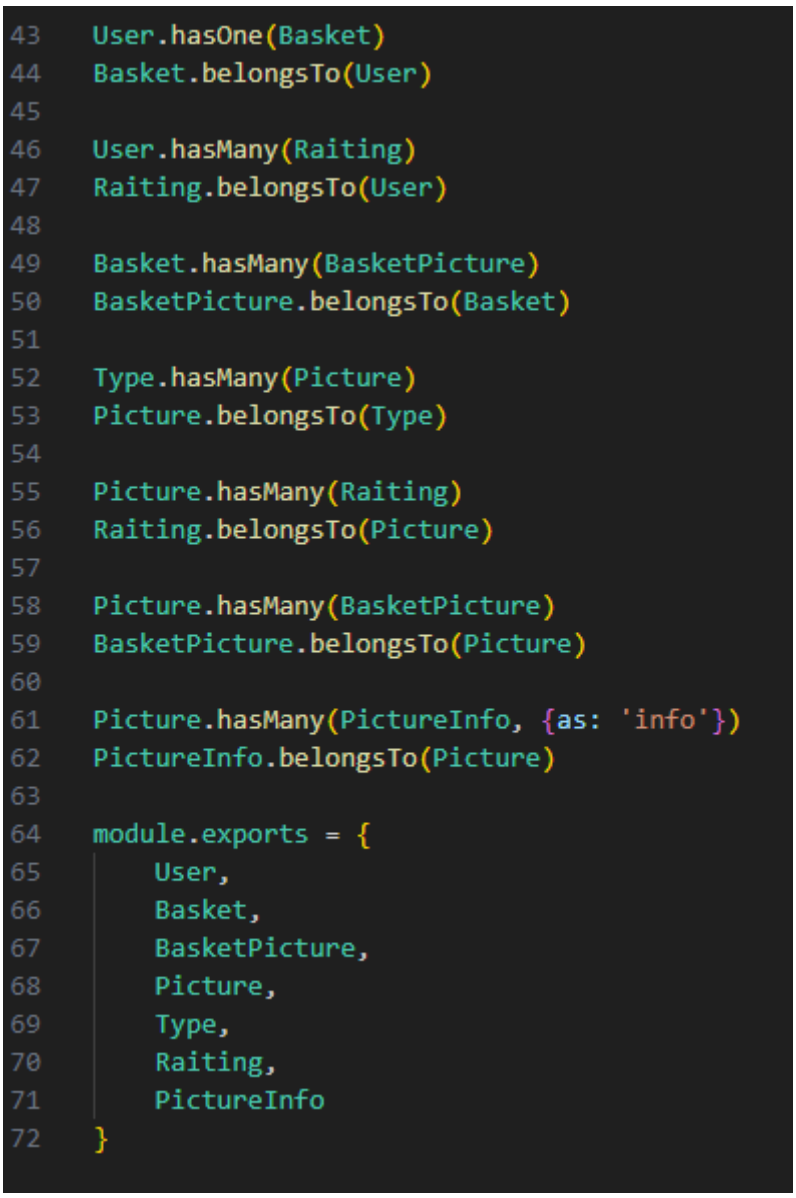


Рис 2.6. Структура клієнтської та серверної частин проекту

Вигляд створення бази даних та налаштування зв’язків між таблицями:

```
const sequelize = require('../db')
const {DataTypes} = require('sequelize')

const User = sequelize.define('user', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    email: {type: DataTypes.STRING, unique: true},
    password: {type: DataTypes.STRING},
    role: {type: DataTypes.STRING, defaultValue: "USER"},
})

const Basket = sequelize.define('basket', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
})

const BasketPicture = sequelize.define('basket_picture', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
})

const Picture = sequelize.define('picture', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    name: {type: DataTypes.STRING, unique: true, allowNull: false},
    price: {type: DataTypes.INTEGER, allowNull: false},
    rating: {type: DataTypes.INTEGER, defaultValue: 0},
    img: {type: DataTypes.STRING, allowNull: false},
})

const Type = sequelize.define('type', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    name: {type: DataTypes.STRING, unique: true, allowNull: false},
})

const Raiting = sequelize.define('raiting', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    rate: {type: DataTypes.INTEGER, allowNull: false},
})

const PictureInfo = sequelize.define('picture_info', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    title: {type: DataTypes.STRING, allowNull: false},
    description: {type: DataTypes.STRING, allowNull: false},
})
```

Рис 2.7. Створення таблиць бази даних



```
User.hasOne(Basket)
Basket.belongsTo(User)

User.hasMany(Raiting)
Raiting.belongsTo(User)

Basket.hasMany(BasketPicture)
BasketPicture.belongsTo(Basket)

Type.hasMany(Picture)
Picture.belongsTo(Type)

Picture.hasMany(Raiting)
Raiting.belongsTo(Picture)

Picture.hasMany(BasketPicture)
BasketPicture.belongsTo(Picture)

Picture.hasMany(PictureInfo, {as: 'info'})
PictureInfo.belongsTo(Picture)

module.exports = {
    User,
    Basket,
    BasketPicture,
    Picture,
    Type,
    Raiting,
    PictureInfo
}
```

Рис 2.8. Налаштування зв’язків між таблицями

Далі в структурі йде View – представлення, вигляд користувальницького інтерфейсу, за це в проекті відповідає бібліотека React.js Зручна робота з компонентами, jsx дуже полегшує роботу зі сторінками.

Саме останнє, та напевно саме головне це контролери, їх можна назвати мозком програми в контролерах було описано опрацювання процесу реєстрації, входу, роботу з адмін-панеллю додаванню та видаленню картин/типів картин.

Короткий вміст основних розділів проекту поділяється на серверний та клієнтський.

Щодо серверної частини проекту:

– контролери відповідають за основні операції на сайті;

– error відповідає за схоплення помилок та подання їх у вигляді зрозумілого тексту на сайт;

– middleware відповідає за вийняткові випадки, (наприклад коли не авторизований користувач намагається додати товар в кошик);

– models відповідає за створення таблиць бази даних, та зв’язків між ними;

– node-modules відповідає за налаштування npm, за допомогою них створюється вся система;

– routes відповідає за шляхи та орієнтування між сторінками сайту;

– static відповідає за отримання статіки (зображення картин).

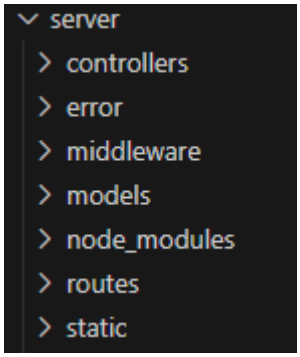


Рис. 2.9. Підрозділи серверної частини

Щодо клієнтської частини проекту:

– node-modules відповідає за налаштування npm, за допомогою них створюється вся система;

– public являє собою каталог з html-документом та іконкою сайту;

– src містить в собі всі елементи та компоненти інформаційної системи.

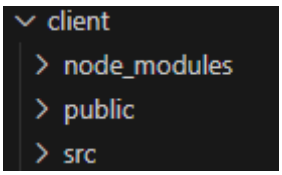


Рис. 2.10. Підрозділи клієнтської частини проекту

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідні дані які надає користувач – це електронну адресу та пароль, для реєстрації та номер телефону, відділ пошти для отримання посилки та зв'язку з клієнтом.

	id [PK] integer	email character varying (255)	password character varying (255)	role character varying (255)	createdAt timestamp with time zone	updatedAt timestamp with time zone
1	1	sergo@gmail.com	$2b$05$K0B8ELePvahkRsKJgz76xu64RgATxASbq8lVVUs68UgA1EFj88Xdy	USER	2023-05-06 19:39:40.763+03	2023-05-06 19:39:40.763+03
2	2	sergo2@gmail.com	$2b$05$r91e2OZaeKytMSZzCaAU5OMsY39cQ42f1oPPE67KIBDKTIwGoQ...	USER	2023-05-06 19:49:45.185+03	2023-05-06 19:49:45.185+03
3	3	sergo3@gmail.com	$2b$05$yrq2BJEsnpS282AoGtzQseyCEHpOHVcUsj9pZEoCuQxDXwfAO5...	USER	2023-05-06 19:52:42.785+03	2023-05-06 19:52:42.785+03
4	4	sergo4@gmail.com	$2b$05$mFfPeVIENc.b9y7JeKb7L.FzQTTE4bYwS0FSFE9b3TnYNPopQW...	USER	2023-05-06 19:58:00.258+03	2023-05-06 19:58:00.258+03
5	5	sergo5@gmail.com	$2b$05$5VID3Cai2pjDv7ITKu2BbeT1NpqMLsx/4.rfvT1pBm.cAIO6bczh6	USER	2023-05-06 19:59:26.443+03	2023-05-06 19:59:26.443+03
6	6	sergey@gmail.com	$2b$05$ktkLHDcRRBKuMylHMoCmlOkg9ZiQAyqH3GckzAMv8KCwjkcTup...	ADMIN	2023-05-09 20:20:39.899+03	2023-05-09 20:20:39.899+03
7	7	sert@gmail.com	$2b$05$J/3RFUwbnqnC1EZ1hbp5seAxKPqRhFkJ1DbwgjvuhctIGgFg6VT7u	ADMIN	2023-05-09 20:30:10.964+03	2023-05-09 20:30:10.964+03
8	8	sergo1@gmail.com	$2b$05$xfgHrcHEWQTR1zIFTLkJC.NEwE/I9hQzbwmPNz2cKpIcTUtRmqD...	ADMIN	2023-05-09 21:03:02.251+03	2023-05-09 21:03:02.251+03

Рис. 2.11. Вигляд вхідних даних користувача

Вхідні дані які надає адміністратор або розробник – це картини(товар), та назви типів картин.

	id [PK] integer	name character varying (255)	price integer	rating integer	img character varying (255)	createdAt timestamp with time zone	updatedAt timestamp with time zone	typeId integer
1	1	Краєвид	950	0	99f25313-5bd6-4c0f-8ca8-3bcd82d95403.jpg	2023-05-06 18:25:42.266+03	2023-05-06 18:25:42.266+03	2
2	2	Карпатський ранок	800	0	c3db1ac0-19ac-4b24-973e-fe95d883b7e3.j...	2023-05-06 18:51:06.867+03	2023-05-06 18:51:06.867+03	2
3	3	Тарас Шевченко	1000	0	8031842c-2fac-4544-b77e-6e800bae7af4.jpg	2023-05-06 18:52:46.579+03	2023-05-06 18:52:46.579+03	1

	id [PK] integer	name character varying (255)	createdAt timestamp with time zone	updatedAt timestamp with time zone
1	1	Портрет	2023-05-06 17:40:24.209+03	2023-05-06 17:40:24.209+03
2	2	Пейзаж	2023-05-06 17:42:05.508+03	2023-05-06 17:42:05.508+03

Рис. 2.12. Вигляд вхідних даних адміністратора

Вихідними даними є файли cookies, які зберігають деяку інформацію (наприклад токен клієнта, який створюється при реєстрації) в текстовому вигляді деякий проміжок часу.

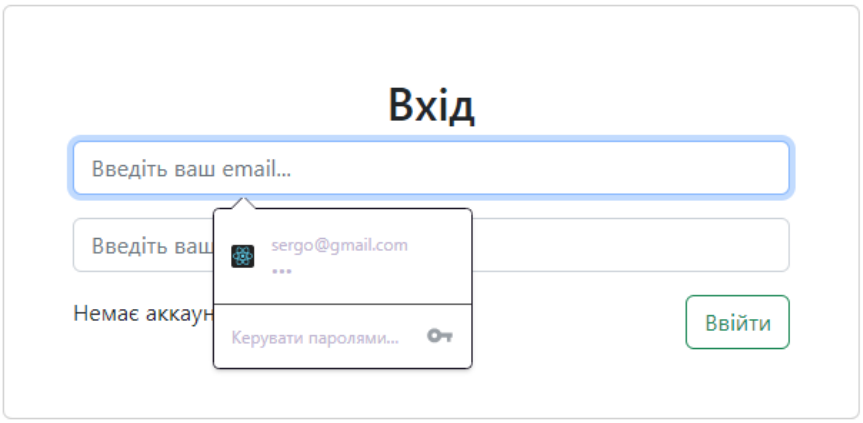


Рис. 2.13. Вигляд вихідних даних з файлів cookies

2.7. Опис розробленої системи

Для користування сайтом треба зайти на нього за посиланням URL: http://localhost:3000 з будь-якого наявного браузеру, після цього перед вами відкриється повністю готовий для використання сайт.

Рис. 2.14. Приклад посилання в браузері

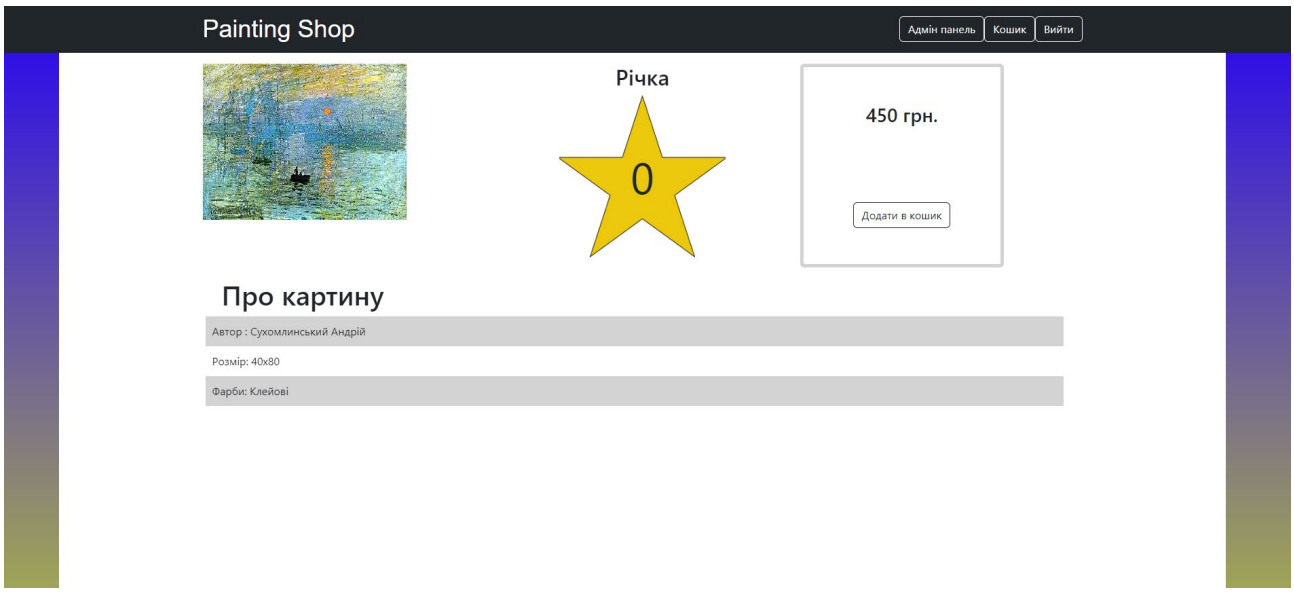


Рис. 2.15. Зовнішній вигляд сайту

2.7.1. Використані технічні засоби

Для розробки веб-додатків рекомендації сучасних розробників щодо мінімальних характеристики пк, або ноутбуків є такими:

– цп: Pentium 4;
– відеоадаптер: 3D адаптер nVidia, Intel, AMD;
– відеопам’ять: 256 МБ;
– накопичувач: 100 Гб;
– оперативна пам’ять: 2 ГБ.

З пристроїв повинні бути:

– миша;
– клавіатура;
– монітор;
– пк з мінімальними характеристиками які вказано вище.

Було використано:

– клавіатуру;
– мишу;
– монітор;

– пк з такими характеристиками:

1) ОЗУ 16 ГБ;
2) накопичувач 1 ТБ;
3) відеоадаптер: Nvidia;
4) відеопам’ять: 12 ГБ.

2.7.2. Використані програмні засоби

Для розробки даного веб-додатку були використані наступні програмні засоби:

– Visual Studio Code;
– pgAdmin 4;
– Figma;
– Diagrams.net;
– Opera GX Tools;
– Postman.

Середовище для написання коду обрано VSCode:

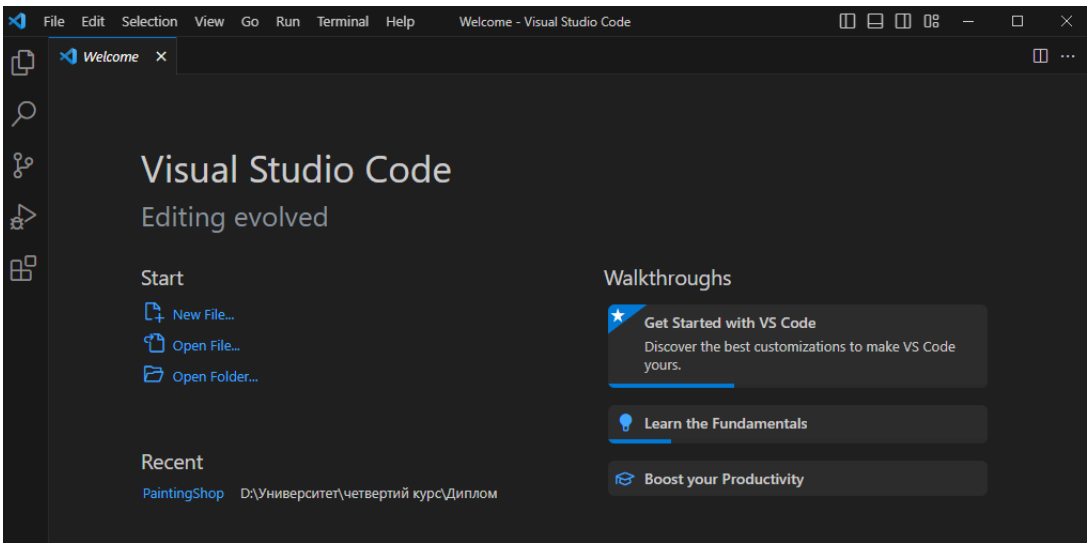


Рис. 2.16. Вигляд середовища для написання коду VSCode

Середовище для роботи з базою даних обрано pgAdmin 4:

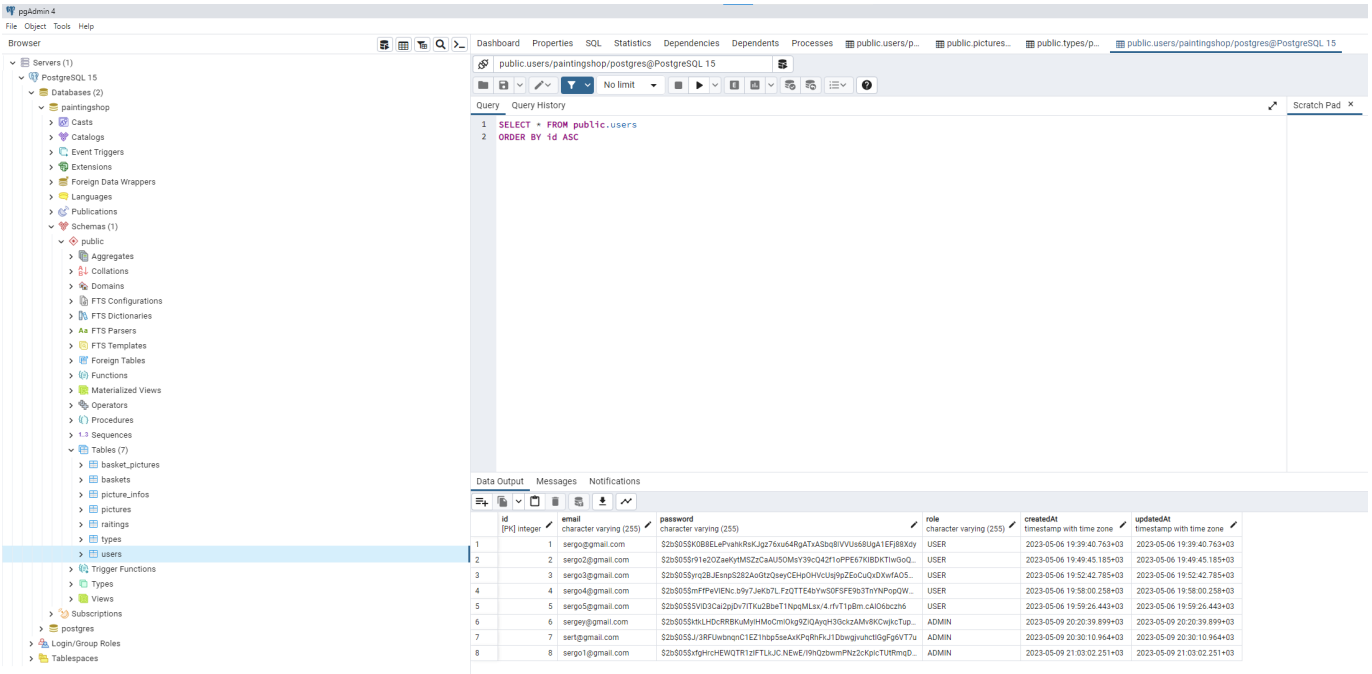


Рис. 2.17. Вигляд середовища для роботи з базою даних pgAdmin 4

Середовище для розробки дизайну сайту обрано Figma:

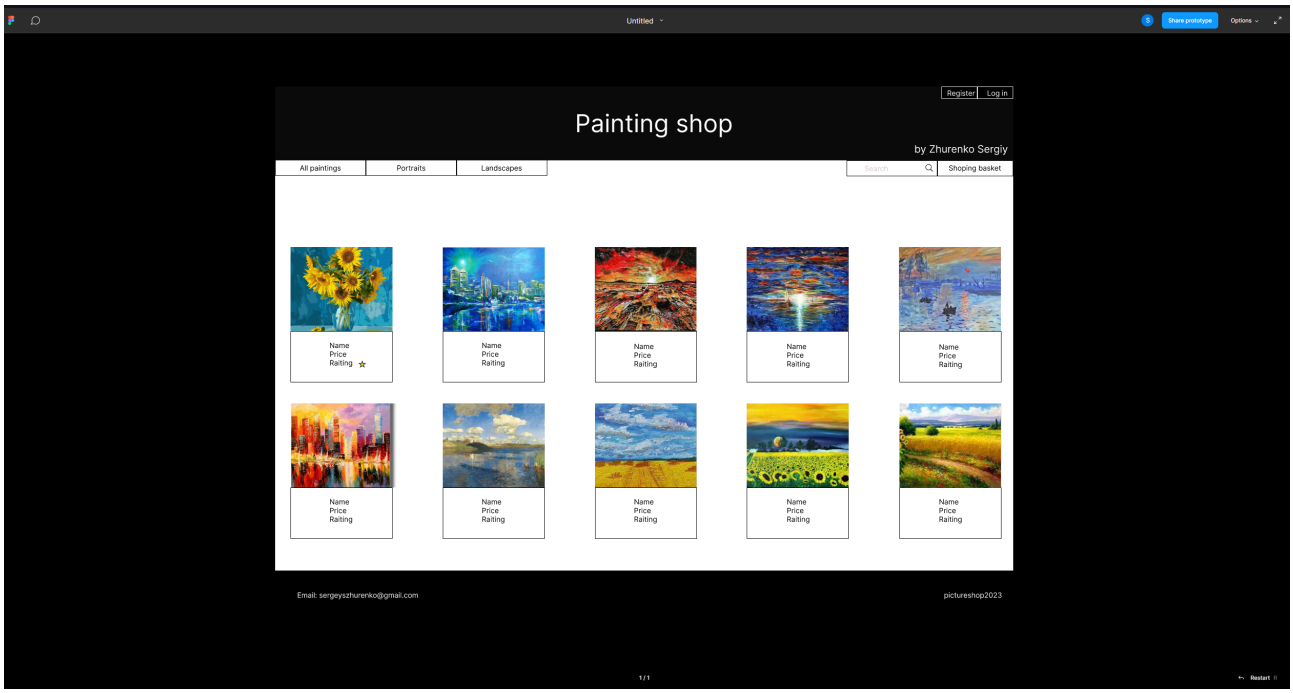


Рис. 2.18. Вигляд середовища для роботи з дизайном Figma

Веб-додаток для розробки діаграми бази даних обрано Diagrams.net:

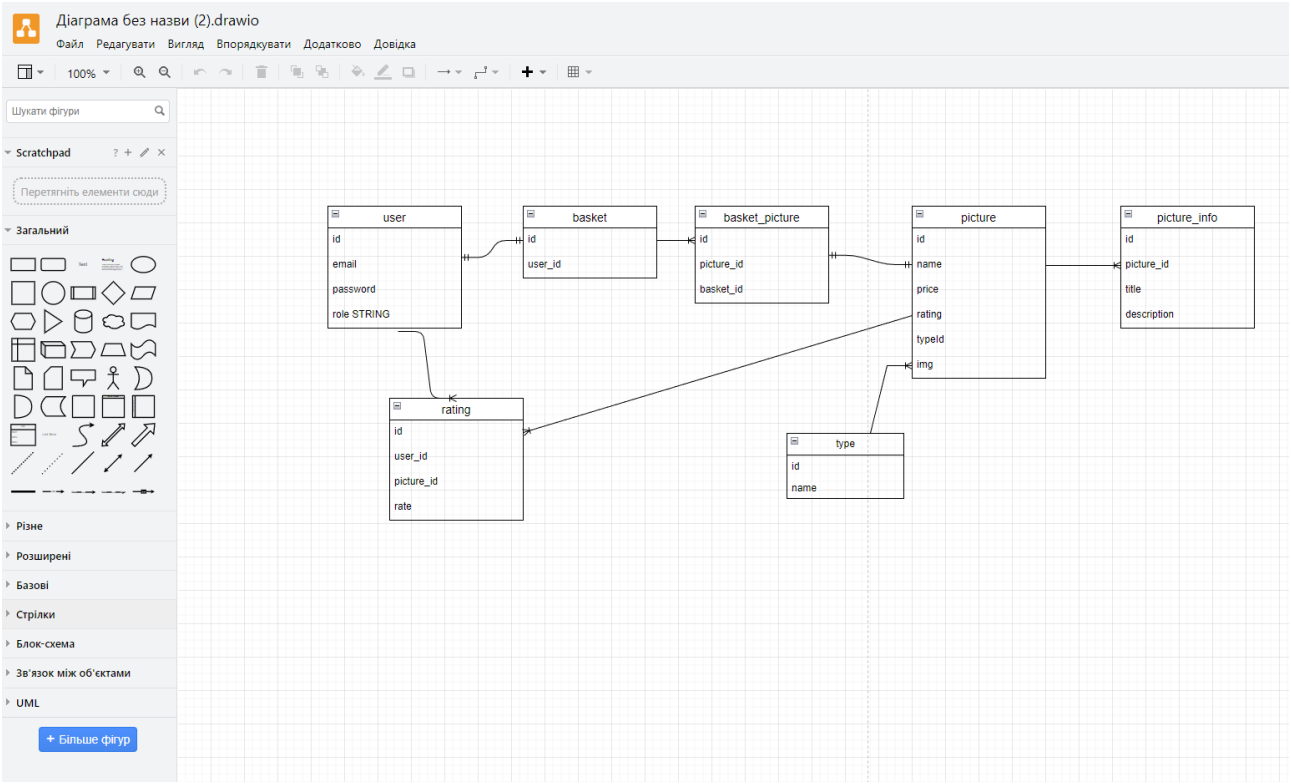


Рис. 2.19. Вигляд сайту для роботи з діаграмами для баз даних Diagrams.net

Інструмент браузеру обрано Opera GX Tools:

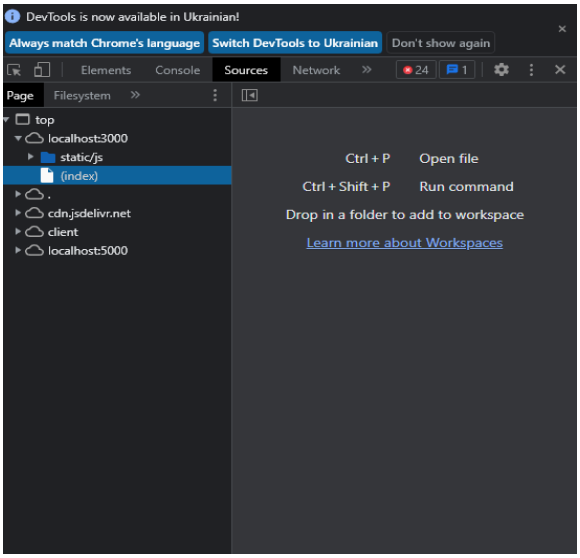


Рис. 2.20. Вигляд інструментів браузеру Opera GX Tools

Середовище для роботи з запитами, для відправки та отримання інформації обрано Postman:

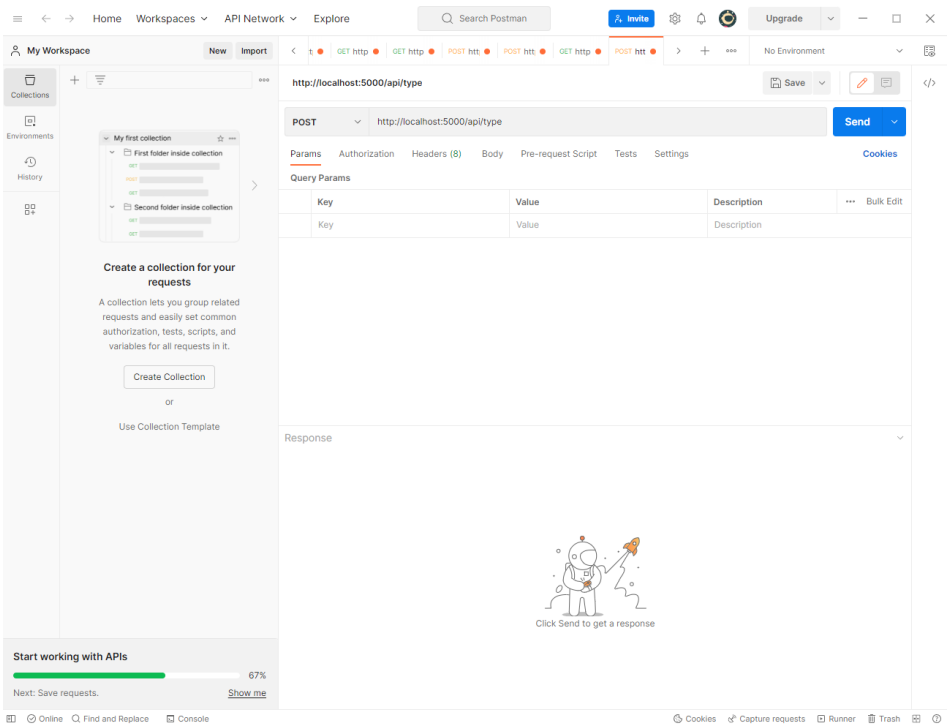


Рис. 2.21. Вигляд середовища для відправки запитів та отримання інформації

2.7.3. Виклик та завантаження програми

Розроблений веб-додаток працює на будь-якій операційній системі та в будь-якому браузері останніх версій при правильно введеному посиланні: http://localhost:3000.

2.7.4. Опис інтерфейсу користувача

2.7.4.1. Клієнтська частина

Після введення url-адреси ми опиняємось на головній сторінці (рис. 2.22):

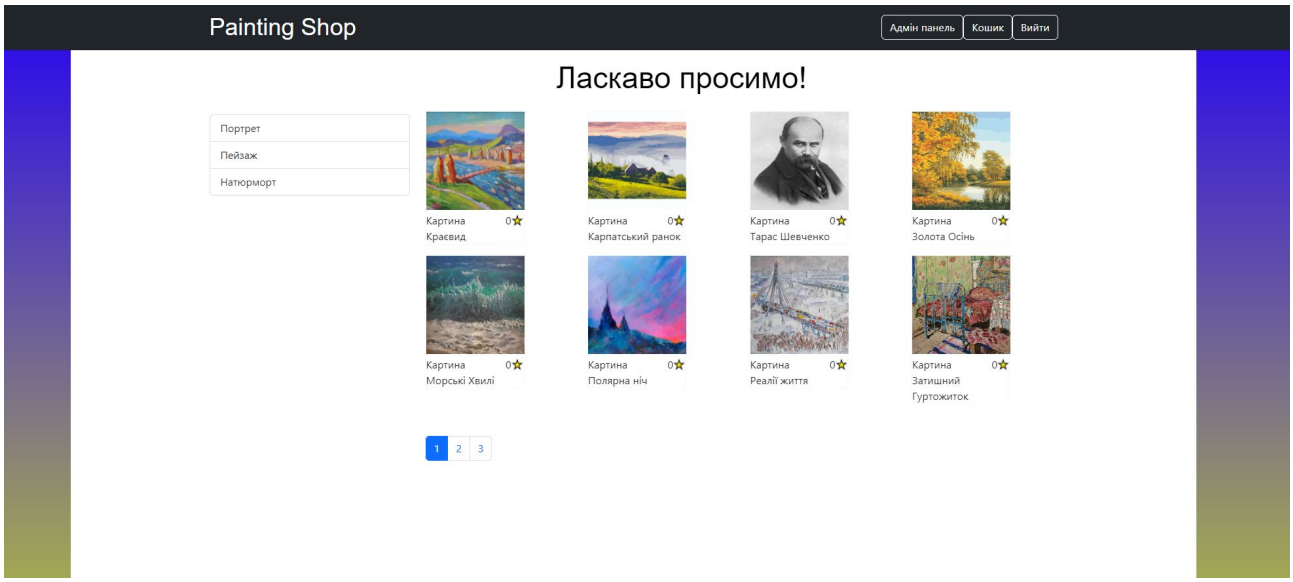


Рис. 2.22. Головна сторінка сайту

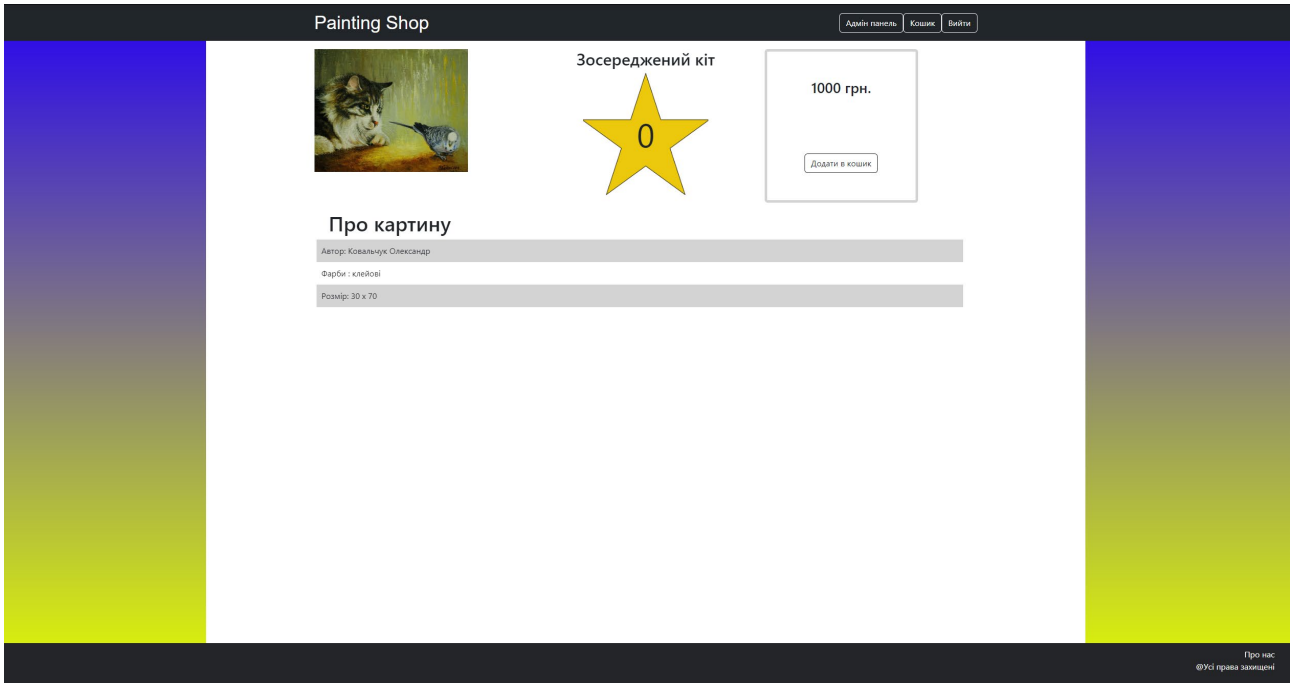


Рис. 2.23. Сторінка картини

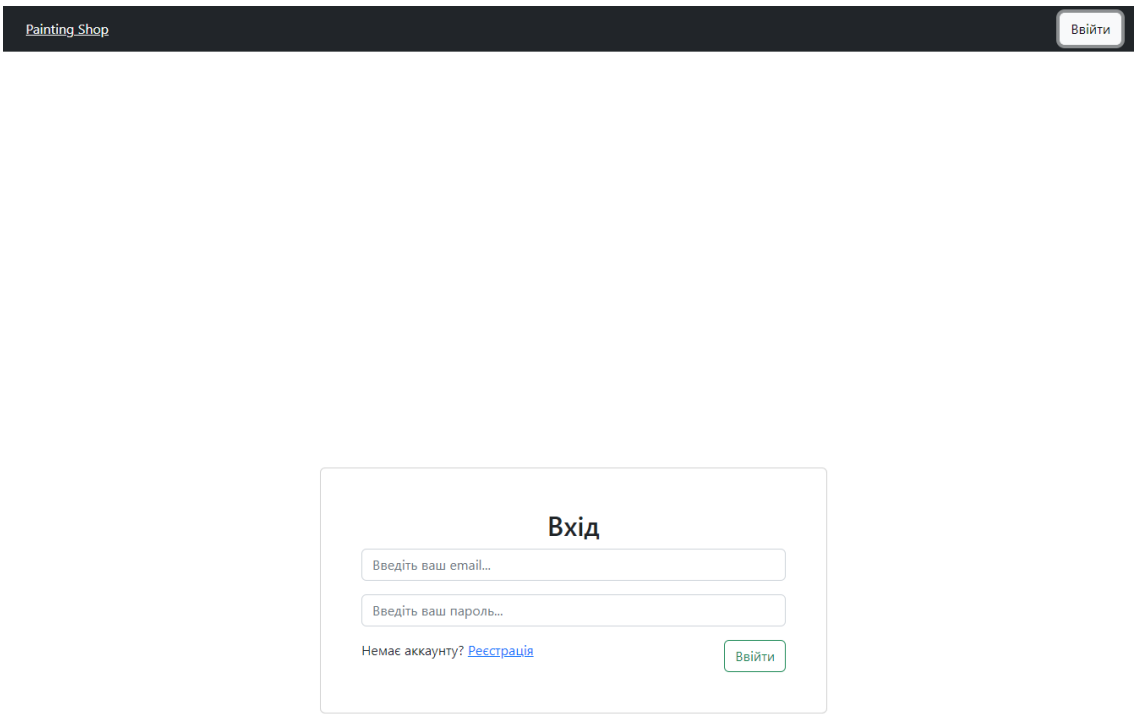


Рис. 2.24. Сторінка входу

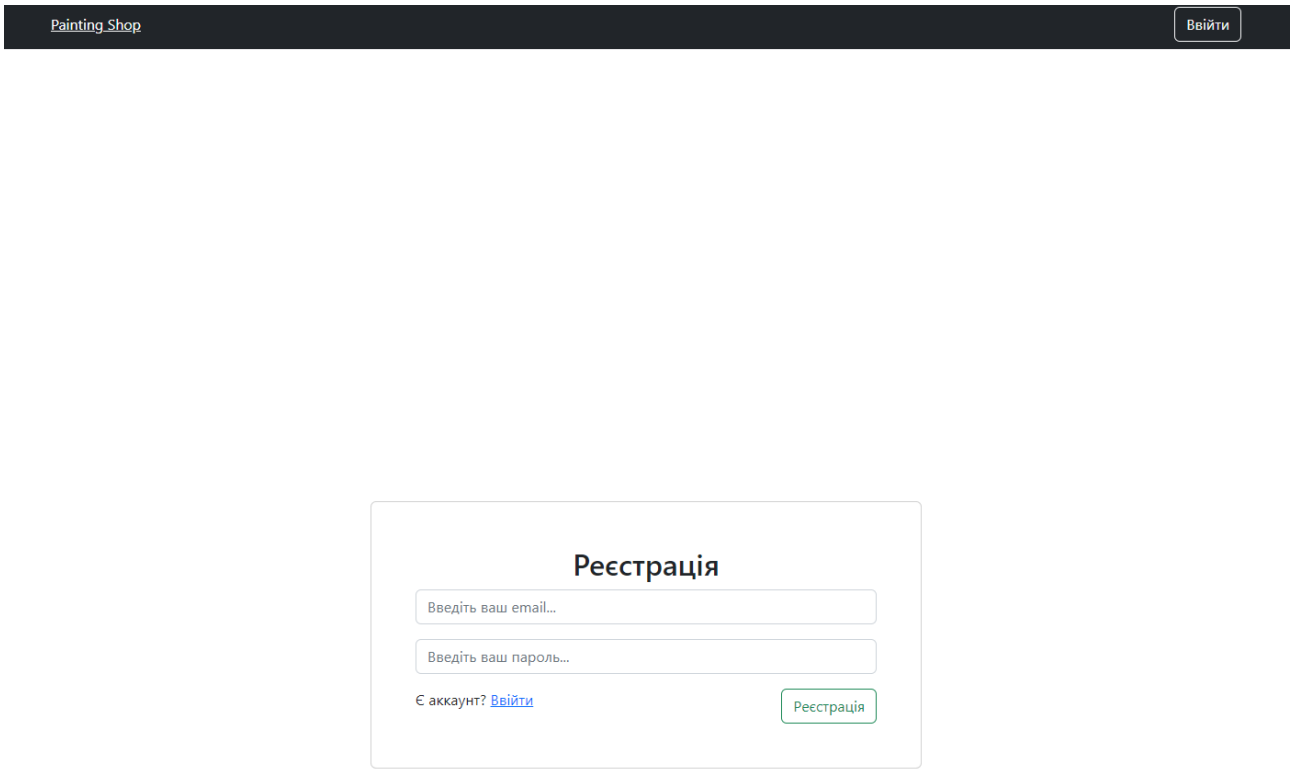


Рис. 2.25. Сторінка реєстрації

2.7.4.2. Адміністрування системою

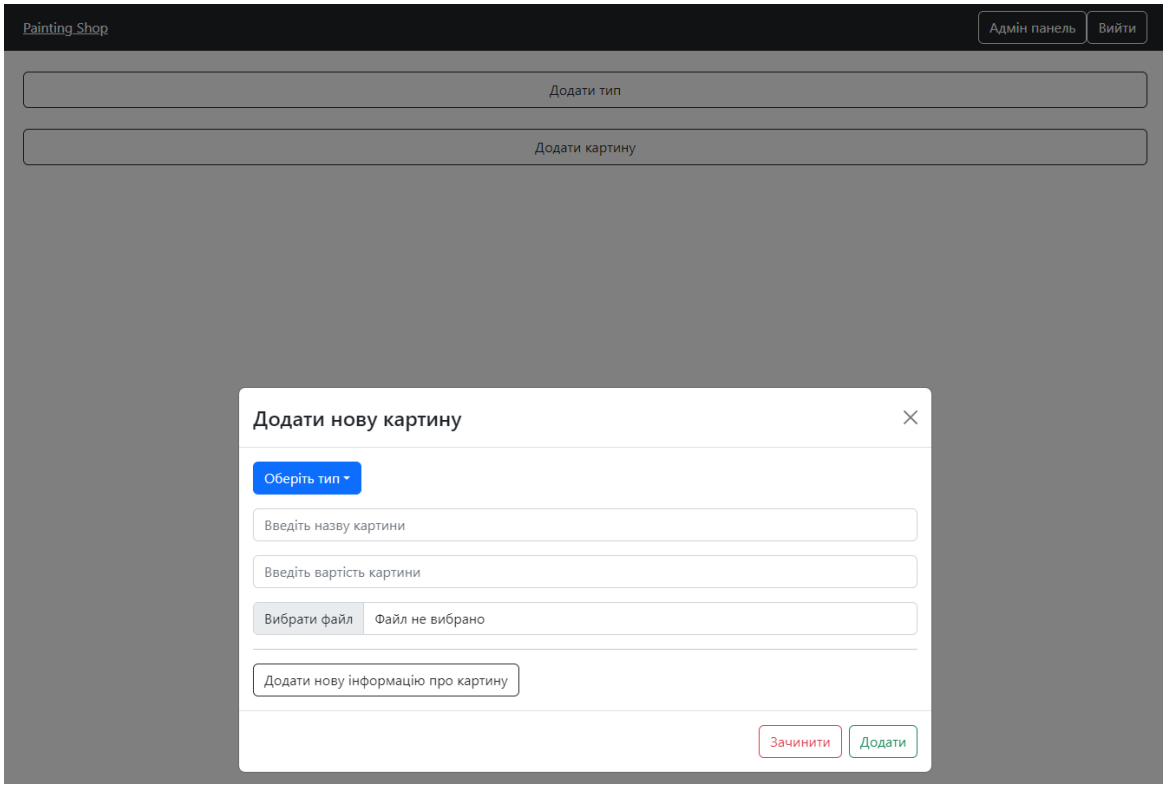


Рис. 2.26. Адмін-панель

Кожна сторінка виконує свою індивідуальну функцію, всі сторінки взаємодіють з базою даних.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вихідні дані:

1. передбачуване число операторів програми – 2000;

2. коефіцієнт складності програми – 1,3;

3. коефіцієнт корекції програми в ході її розробки – 0,2;

4. годинна заробітна плата розробника – 147 грн/год (за версією сайта DOU.ua) - https://jobs.dou.ua/salaries/?period=2022-12&position=Junior%20SE;

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;

7. вартість машино-години ЕОМ – 23 грн/год (1 грн – е/е, 17 грн – ПЗ, 5 грн -амортизація).

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_{д}$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q – передбачуване число операторів (2000);

C – коефіцієнт складності програми (1,3);

p – коефіцієнт корекції програми в ході її розробки (0,2).

$$Q = 2000 \cdot 1{,}3 \cdot (1 + 0{,}2) = 3120;$$

Витрати праці на вивчення опису задач t_{u} визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85) \cdot K}, \text{ людино-годин} \tag{3.3}$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2);

К – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1,1);

$$t_u = \frac{3120 \cdot 1{,}2}{85 \cdot 1{,}1} = 40{,}04 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot K}; \tag{3.4}$$

$$t_a = \frac{3120}{20 \cdot 1{,}1} = 141{,}81 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot K}; \quad (3.5)$$

$$t_n = \frac{3120}{25\cdot 1,1} = 113,45 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)\cdot K}; \quad (3.6)$$

$$t_{отл} = \frac{3120}{5\cdot 1,1} = 567,27 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,2 \cdot t_{отл}; \quad (3.7)$$

$t_{отл}^{к} = 1,2 \cdot 567,27 = 680,72$ людино-годин.

Витрати праці на підготовку документації:

$$t_д = t_{дp} + t_{до}; \quad (3.8)$$

де $t_{дp}$ – трудомісткість підготовки матеріалів і рукопису;

$$t_д = \frac{Q}{(15...20)\cdot K}; \quad (3.9)$$

$$t_{дp} = \frac{3120}{20\cdot 1,1} = 141,81 \text{ людино-годин.}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації;

$$t_{до} = 0{,}75 \cdot t_{др};$$ (3.10)

$t_{до}$ = 0,75 · 141,81 = 106,35 людино-годин.

$t_{д}$ = 141,81 + 106,35 = 248,16 людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 40,04 + 141,81 + 113,45 + 567,27 + 248,16 = 1160,73 людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1160,73 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $Кпо$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,}$$ (3.11)

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$ (3.12)

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата розробника становить 147 грн/год, то отримаємо:

$$З_{ЗП} = 1160,73 \cdot 147 = 170\ 627,31 \text{ грн}$$

Вартість машинного часу $З_{МВ}$, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_М \text{ , грн,} \tag{3.13}$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_М$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 567,27 \cdot 23 = 13047,21 \text{ грн}$$

Звідси витрати на створення програмного продукту:

$$Кпо = 170\ 627,31 + 13047,21 = 183\ 674,52 \text{ грн}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ , мес.} \tag{3.14}$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

Витрати на створення програмного продукту:

$T = \frac{1160,73}{1 \cdot 176}$ = 6,59 міс.

Вартість розробки інтернет-магазину становить 183 674,52 грн. Час розробки очікується приблизно 6,59 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Цей термін включає час, необхідний для дослідження, розробки алгоритму, дизайну і документування. Загальна кількістю людино-годин, яку буде витрачено на розробку – 1160,73.

ВИСНОВОК

Під час виконання даної кваліфікаційної роботи був розроблений веб-орієнтований додаток, були використані сучасні підходи та інструменти для його створення.

Веб-додаток призначений для продажу картин та комерційної діяльності однак він розроблений лише для кваліфікаційної роботи, тому не буде використовуватись для комерційної діяльності. Також цей проект відіграє велику роль в сфері маркетингу (приваблює більше користувачів й потенційних покупців до реально розташованого магазину).

Даний веб-додаток має практичну цінність для користувачів. Через те, що економить їх час. Їм не потрібно відвідувати реальний магазин, щоб подивитись асортимент, або придбати щось, бо вони можуть це зробити прямо вдома.

Під час виконання даної кваліфікаційної роботи були поставлені та виконані наступні задачі:

– аналіз предметної області;
– знайдено підстави для розробки даного веб-додатку;
– обрано раціональні та сучасні структуру та технології створення;
– написано програмний код веб-додатку;
– розроблено рекомендації щодо використання системи.

Сайт надає можливість користувачу:

– реєструватися (дані зберігаються в базу даних);
– дивитись товари, інформацію про товари (ціну, опис);
– є можливість входу та виходу з облікового запису;
– додавати товари до кошику та здійснювати покупки;
– також є адмінпанель для адміністраторів котра дозволяє редагувати товари, ціни та опис товару, додавати та видаляти товари.

Поставлена задача є дуже актуальною, оскільки створення інтернет-магазину є ефективним рішенням для підприємства, що прагне розширити свою

онлайн присутність і привернути увагу більшої кількості клієнтів. Зараз спостерігається зростання популярності інтернет-магазинів серед споживачів, завдяки їх зручності та доступності.

Основні характеристики інтернет-магазину включають:

– інформація про кожен товар: назва, опис, фото та ціна;

– зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє легко знаходити потрібні товари та здійснювати інші операції в магазині;

– онлайн оформлення замовлення з доставкою до заданої адреси;

– захищена авторизація (ваш пароль знаходиться в базі даних у вигляді веб токену);

– швидке та безпечне придбання потрібних товарів.

Ці характеристики є базовими для будь-якого успішного інтернет-магазину, сайт забезпечує безпечні покупки та зручний інтерфейс з сортуванням, для швидкого пошуку потрібного товару.

Структура сайту складається з двох частин backend та frontend, серверної та клієнтської частин, серверна частина написана на nodejs, клієнтська на Reactjs, їх зв’язано з базою даних під назвою «painting_shop». База даних PostgreSQL.

Згідно з економічним розділом проекту, загальна трудомісткість розробки програмного забезпечення оцінена в 1160,73 людино-годин, а витрати на створення становлять 183 674,52 грн. Прогнозований період розробки складає 6,59 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Офіційна документація ReactJS. / URL: https://reactjs.org/. дата звернення: 12.05.2023.

2.Офіційна документація Node.js. / URL: https://nodejs.org/. дата звернення: 12.05.2023.

3.Create React App. / URL: https://create-react-app.dev/. дата звернення: 13.05.2023.

4.Express.js (фреймворк для розробки серверної частини проекту).

/ URL: https://expressjs.com/. дата звернення: 14.05.2023.

5.React Router (маршрутизація в React). / URL: https://reactrouter.com/. дата звернення: 14.05.2023.

6.Axios (робота з HTTP-запитами). / URL: https://axios-http.com/. дата звернення: 14.05.2023.

7.Material-UI (компоненти UI на основі Material Design). / URL: https://mui.com/. дата звернення: 14.05.2023.

8.Styled Components (стилізація компонентів). / URL: https://styled-components.com/. дата звернення: 15.05.2023.

9.JWT (робота з JSON Web Tokens). / URL: https://jwt.io/. дата звернення: 15.05.2023.

10.Bcrypt.js (шифрування паролів). / URL: https://www.npmjs.com/package/bcryptjs/. дата звернення: 16.05.2023.

11.PostgreSQL (реляційна база даних). / URL: https://www.postgresql.org/. дата звернення: 16.05.2023.

12.GitHub (репозиторій та керування версіями). / URL: https://github.com/. дата звернення: 16.05.2023.

13. NPM (пакетний менеджер для Node.js). / URL: https://www.npmjs.com/. дата звернення: 17.05.2023.

14. Представлення UI перед початком розробки. / URL: https://www.figma.com/. дата звернення: 17.05.2023.

15. Представлення діаграми бази даних. / URL: https://www.diagrams.net/. дата звернення: 18.05.2023.

16. Стоян Стефанов. React Up & Running: Building Web Applications. — O'Reilly Media , 2016. — 230 с.

17. Нідерст Роббінс. HTML5 Pocket Reference. — O'Reilly Media, 2013. — 184 с.

18. Веру, Леа. CSS Secrets: Better Solutions to Everyday Web Design Problems. — O'Reilly Media, 2015. — 392 с.

19.Даглас Марк. Learning Node.js: A Hands-On Guide to Building Web Applications in JavaScript. — Addison-Wesley Professional, 2016. — 412 с.

20. Бендер, Алекс та Порцелло, Ева. Learning React: Functional Web Development with React and Redux. — O'Reilly Media, 2017. — 350 с.

21. Олдхем, Девон Х. Node.js Web Development: Server-side Development with Node 10 and JavaScript. — Packt Publishing, 2018. — 548 с.

22. Пілгрім, Марк. HTML5: Up and Running. — O'Reilly Media, 2010. — 224 с.

23. Джон Дакетт. HTML and CSS: Design and Build Websites. — O'Reilly Media, 2011. — 230 с.

24. Майк Кантелон та Марк Харбер. Node.js in Action. — Manning Publications, 2014. — 230 с.

25. Каccio де Соуза Сантос та Аарон Бевінс. Pro React. — O'Reilly Media, 2019. — 230 с.

26. Фреймворк react bootstrap детально про стилі. / URL: https://react-bootstrap.netlify.app/. дата звернення: 24.05.2023.

27. Для вдосконалення навичків javascript SoloLearn. / URL: https://www.sololearn.com/Course/JavaScript/. дата звернення: 24.05.2023.

28.Для вдосконалення навичків HTML, CSS SoloLearn (HTML & CSS). / URL: https://www.sololearn.com/Course/HTML/. дата звернення: 24.05.2023.

ДОДАТОК А

КОД ПРОГРАМИ

```
import { BrowserRouter } from 'react-router-dom';
import './App.css';
import { observer } from 'mobx-react-lite';
import { useContext, useEffect, useState } from 'react';
import { Context } from './index';
import { Spinner } from 'react-bootstrap';
import { check } from "./http/userAPI";

  if (loading) {
    return <Spinner animation={"grow"} />;
  }

  return (
    <BrowserRouter>
      <NavBar />
      <AppRouter />
      <Footer />
    </BrowserRouter>
  );
});

export default App;


class PictureController {
    async create(req, res, next) {
        try {
            let {name, price, typeId, info} = req.body
            const {img} = req.files
            let fileName = uuid.v4() + ".jpg"
            img.mv(path.resolve(__dirname, '..', 'static', fileName))
            const picture = await Picture.create({name, price, typeId, img:
fileName})

            if (info) {
                info = JSON.parse(info)
                info.forEach(i =>
                    PictureInfo.create({
                        title: i.title,
                        description: i.description,
                        pictureId: picture.id
                    })
                )
            }

            return res.json(picture)

        }
        if (typeId) {
            pictures = await
Picture.findAndCountAll({where: {typeId}, limit, offset})
        } else {
            pictures = await
Picture.findAndCountAll({limit, offset})
        }
        return res.json(pictures)
```

```
    }

    async getOne(req, res) {
        const {id} = req.params
        const picture = await
Picture.findOne(
            {
                where: {id},
                include: [{model: PictureInfo, as: 'info'}]
            },
        )
        return res.json(picture)
    }
}

module.exports = new
PictureController()



const Router = require('express')
const router = new Router()
const pictureController = require('../controllers/pictureController')
const checkRole = require('../middleware/checkRoleMiddleware')

router.get('/', pictureController.getAll)
router.get('/:id', pictureController.getOne)
router.post('/', checkRole('ADMIN'),
pictureController.create)

module.exports = router


import { observer } from "mobx-react-lite";
import React, { useContext } from "react";
import { Context } from "../index";
import PictureItem from "./PictureItem";
import { Row } from "react-bootstrap";

const PictureList = observer(() => {
  const { picture } = useContext(Context);

  return (
    <Row className="d-flex">
      {picture.pictures.map((picture) => (
        <PictureItem key={picture.id} picture={picture} />
      ))}
    </Row>
  );
});

export default PictureList;

</Modal.Body>
</Button>
<Button variant="outline-success" onClick={addPicture}>
Додати
</Button>
</Modal.Footer>
</Modal>
);
});
```

```
export default CreatePicture;
import React from 'react';

const Footer = () => {
  return (
    <footer style={{ position: 'fixed', bottom: 0, width: '100%',
backgroundColor: '#242528', color: 'white', textAlign: 'center', padding: '10px 0'}}>
        <a align="right" style={{ color: 'white', float: 'right',marginRight:
'30px', textDecoration: 'none' }} href='/about'>Про нас</a><br/>
        <p align="right" style={{ marginRight: '30px' }}>@Усі права захищені</p>
    </footer>
  );
}

export default Footer;

import { $authHost, $host } from "./index";

export const createType = async (type) => {
    const {data} = await $authHost.post('api/type', type)
    return data
}

export const fetchTypes = async () => {
    const {data} = await $host.get('api/type')
    return data
}

export const createPicture = async (picture) => {
    const {data} = await $authHost.post('api/picture', picture)
    return data
}

export const fetchPictures = async (typeId, page, limit = 5) => {
    const {data} = await $host.get('api/picture', {params: {
        typeId, page, limit
    }})
    return data
}

export const fetchOnePicture = async (id) => {
    const {data} = await $host.get('api/picture/' + id)
    return data
}

import { $authHost, $host } from "./index";

REACT_APP_API_URL='http://localhost:5000/'
body {
  position: relative;
  margin: 0;
  padding: 0;
}

body::before,
body::after {
  content: "";
  position: fixed;
  top: 0;
  bottom: 0;
  width: 400px;
```

```
  background: linear-gradient(to bottom, #2602f3, #e4fd00);
  z-index: -1;
}

body::before {
  left: 0;
}

body::after {
  right: 0;
}

export const BASKET_ROUTES = '/basket'
export const PICTURE_ROUTES = '/picture'
export const ABOUT_ROUTES = '/about'
BasketPicture.belongsTo(Basket);

Type.hasMany(Picture);
Picture.belongsTo(Type);

Picture.hasMany(Rating);
Rating.belongsTo(Picture);

Picture.hasMany(BasketPicture);
BasketPicture.belongsTo(Picture);

Picture.hasMany(PictureInfo, { as: 'info' });
PictureInfo.belongsTo(Picture);

module.exports = {
  User,
  Basket,
  BasketPicture,
  Picture,
  Type,
  Rating,
  PictureInfo,
};
      req.user = decoded;
      next();
    } catch (e) {
      res.status(401).json({ message: "Не авторизований" });
    }
  };
};

import React, { useContext } from "react";
import { Context } from "..";
  return (
    <Navbar bg="dark" variant="dark">
      <Container>
        <NavLink
          style={{
            color: "white",
            fontFamily: "Arial, sans-serif",
            fontSize: "36px",
            textDecoration: "none",
          }}
          to={SHOP_ROUTE}
        >
          Painting Shop
            <div>
```

```
                    <Button
                      variant={"outline-light"}
                      onClick={() => navigate(ADMIN_ROUTE)}
                      style={{ marginLeft: "30px" }}
                    >
                      Адмін панель
                    </Button>
                  </div>
                  <div>
                    <Button
                      variant={"outline-light"}
                      onClick={() => navigate(BASKET_ROUTE)}
                      style={{ marginLeft: "30px" }}
                    >
                      Кошик
                    </Button>
                  </div>
                  <div>
                    <Button
                      variant={"outline-light"}
                      onClick={() => logOut()}
                      style={{ marginLeft: "30px" }}
                    >
                      Вийти
                    </Button>
                  </div>
                </Nav>
              ) : (
                <Nav className="ml-auto" style={{ color: "white" }}>
                  <Button variant={"outline-light"} onClick={() =>
navigate(LOGIN_ROUTE)}>
                    Ввійти
                  </Button>
                </Nav>
              )}
            </Container>
          </Navbar>
        );
      });

      export default NavBar;



      import React, { useEffect, useState } from "react";
      import { Col, Card, Row, Form, Container, Image, Button } from "react-bootstrap";
      import bigstar from "../assets/bigstar.png";
      import { useParams } from "react-router-dom";
      import { fetchOnePicture } from "../http/pictureAPI";

      const PicturePage = () => {
        const [picture, setPicture] = useState({ info: [] });
        const { id } = useParams();

        useEffect(() => {
          fetchOnePicture(id).then((data) => setPicture(data));
        }, []);

        const description = [
          { id: 1, title: "Автор картини", description: "Григорій Троян" },
          { id: 2, title: "Тип фарби", description: "Олійна" },
          { id: 3, title: "Розмір", description: "600x800" },
```

```
  ];

  return (
    <Container className="mt-3">
      <Row>
        <Col md={4}>
          <Image
            width={300}
            height={300}
            src={process.env.REACT_APP_API_URL + picture.img}
          />
        </Col>
       </Row>
   </Container>


export default PicturePage;

import React, { useContext, useEffect } from "react";
import { Container, Col, Row } from "react-bootstrap";
import TypeBar from "../components/TypeBar";
import PictureList from "../components/PictureList";
import Pages from "../components/Pages";
import { observer } from "mobx-react-lite";
import { Context } from "..";
import { fetchTypes, fetchPictures } from "../http/pictureAPI";

const Shop = observer(() => {
  const { picture } = useContext(Context);
  const { user } = useContext(Context);

  useEffect(() => {
    fetchTypes().then((data) => picture.setTypes(data));
    fetchPictures(null, 1, 3).then((data) => {
      picture.setPicture(data.rows);
      picture.setTotalCount(data.count);
    });
  }, []);

  useEffect(() => {
    fetchPictures(picture.selectedType.id, picture.page, 3).then((data) => {
      picture.setPicture(data.rows);
      picture.setTotalCount(data.count);
    });
  }, [picture.page, picture.selectedType]);

  return (
    <Container>
      <Row className="mt-2">
        <Col md={3}>
          <TypeBar />
        </Col>
        <Col md={9}>
          <div
            style={{
              color: "black",
              fontFamily: "Arial, sans-serif",
              fontSize: "46px",
              marginLeft: "200px",
            }}
          >
            Ласкаво просимо!
```

```
          </div>
          <PictureList />
          <Pages />
        </Col>
      </Row>
    </Container>
  );
});

export default Shop;

import { makeAutoObservable } from "mobx";

export default class PictureStore {
  constructor() {
    this._types = [];
    this._pictures = [];
    this._selectedType = {};
    this._page = 1;
    this._totalCount = 0;
    this._limit = 8;
    this._isAuth = false;
    this._user = {};

    makeAutoObservable(this);
  }

  setTypes(types) {
    this._types = types;
  }

  setPicture(pictures) {
    this._pictures = pictures;
  }
  get types() {
    return this._types;
  }

  get pictures() {
    return this._pictures;
  }
  get totalCount() {
    return this._totalCount;
  }

  get isAuth() {
    return this._isAuth;
  }


  get page() {
    return this._page;
  }
  get user() {
    return this._user;
  }

  get limit() {
    return this._limit;
  }
. .  .
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна_робота_ Журенко.docx	Пояснювальна записка до кваліфікаційного проекту. Документ Word.
Кваліфікаційна_робота_ Журенко.pdf	Пояснювальна записка до кваліфікаційного проекту в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Журенко.ppt	Презентація кваліфікаційного проекту