НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
“ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ”
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

ПРОЕКТНА РОБОТА

з дисципліни

Комп’ютерні Мережі

на тему:

“Розробка інтернет-магазину для Дизайн-Студії ANNA-OSTROVSKA ”

Студента *III* курсу групи КА-71
Спеціальність 124 Системний аналіз.
Островський Захар Юрійович
Прийняв: Кухарєв С.О.

Київ – 2020 рік

ЗМІСТ

ВСТУП

Актуальність: з розвитком онлайн-торгівлі та сучасними викликами, такі як епідемії, висока ціна орендної плати, створення інтернет-магазину стає єдиним рішенням, яке дає бізнесу можливість працювати у цих умовах.

Мета: розробити та реалізувати MVP модель інтернет-магазину ANNA-OSTROVSKA.

Завдання: розібратися у принципі роботи і проектування бекенду сайту, інтеграції клієнтської частини із сервером і базою даних, підібрати необхідні програмні продукти для реалізації поставленої задачі з врахуванням наступних критеріїв:

- актуальність, швидкодія та перспективність програмної складової;
- гнучкість та легкість модифікацій, масштабованість;
- швидкість розробки.

Далі необхідно освоїти обраний програмний продукт і, враховуючи відсутність попереднього досвіду у написанні бекенду та повноцінного проектування баз даних, розробити MVP модель інтернет-магазину (інтеграцію фронт-, бекенду і БД) з власною CMS (Content Management System), яка дозволятиме забезпечити повноцінний процес: створення адміністратором через CMS нових товарів по категоріях, відображення товарів на стороні клієнта, можливість для клієнта оформити замовлення, відображення замовлень адміністратору в CMS.

Практичне значення одержаних результатів: розроблена модель буде використана для тестування ефективності підходу, виявлення сильних і слабких сторін її використання для подальшого розширення функціоналу, а також інших програмних аспектів, які необхідно враховувати при проектуванні такого роду систем. Ця MVP модель стане основою для першої версії інтернет-магазину, який вийде в продакшн.

Використане програмне забезпечення: при виконанні роботи було використане таке програмне забезпечення: JavaScript + Node.js, стандартне середовище розробки PhpStorm; операційна система Windows 10; веб-браузер Google Chrome для роботи з веб-сайтами; текстовий редактор Microsoft Word 2016 для підготовки та оформлення проектної роботи.

Структура роботи: робота складається зі вступу, трьох розділів, висновку, списку використаних джерел та додатків.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

1.1. Опис вимог для MVP моделі

Сайт повинен містити функціонал для адміністратора і звичайного покупця, бути швидким в роботі та мати стильний і адаптивний дизайн.

Design Studio ANNA OSTROVSKA займається професіональним пошиттям високоякісних індивідуальних танцювальних костюмів, а також має свою лінійку танцювальної форми. Крім того планується розширення асортименту товарів, доступних для загального продажу. Кожен з таких товарів має свою категорію. . За кожною категорією закріплюється своя розмірна сітка. Для клієнта також на кожен товар можуть бути доступні для вибору такі параметри: колір, розмір, довжина, кількість. Товари бувають у двох версіях чорній і кольоровій. В чорній – основний матеріал чорний і містить кольорові вставки (колір вставок на вибір клієнта). В кольоровій - основний матеріал кольоровий (колір основного матеріалу на вибір клієнта) і містить чорні вставки. Кожна версія має свою ціну та опис.

Користувач повинен по черзі додати до кошика кожен товар у необхідній для нього версії, потрібних параметрах і кількості. Після цього перейти на сторінку оформлення замовлення для надання своїх контактних даних і способу доставки. Для допомоги користувачу і уникненню введення некоректних даних повинна бути продумана система підказок, валідації введених даних у формі та автоматичного показу лише можливих для заповнення параметрів на основі попередньо обраних.

Реєстрація користувача і створення його власного профілю передбачається, але не входить у плани для реалізації найближчим часом. Таким

чином на даному етапі важливо розробити надійний спосіб реалізації кошика в умовах не ідентифікованого користувача.

У розробці архітектури системи вирішено прагнути слідувати концепції RESTful API. Згідно одному із принципів запити на сервер повинні повністю описувати зміст дії, яку вони мають виконати, і сервер не має зберігати в себе відомості про минулі запити. Тому реалізація кошика має повністю відбуватися на стороні фронтенду, а дані відправлятися лише в момент підтвердження форми оформлення замовлення. Ця проблема була елегантно і ефективно вирішена з використанням localStorage браузера, в якому зберігалися дані про товари кошика у форматі JSON. Таким чином, оскільки браузером виділяється для кожного сайту своє місце у localStorage, то якщо користувач відкриє сайт у кількох вкладках, чи навіть вікнах (чи взагалі закриє сайт і повернеться через кілька днів для завершення покупки), кожний екземпляр сайту звертатиметься до єдиного місця сховища на комп'ютері. Відповідно дії будуть синхронізуватися.

Крім того для MVP моделі сайт повинен містити головну сторінку з описовою складовою та інформацією про умови оплати і доставки.

Для адміністратора функціонал MVP моделі повинен включати можливість створення нового товару і перегляду нових замовлень. Було вирішено, що у зв'язку з малою кількістю категорій і однотипністю параметрів, можливих для товарів, не є доцільним прописувати окремий функціонал для адміністратора по їх створенню. Категорії і доступні параметри вже будуть закладені в логіку системи.

Форма для створення нового товару повинна складатися з таких полів: назва товару – input [type="text"], назва сторінки товару (фактично url сторінки) – input [type="text"], опис картці товару – textarea; вибір категорії товару – select,

вибір матеріалів виробу – select multiple; вибір параметрів, що характеризують товар – checkbox; вибір кожної з доступних версій включає в себе одночасно – версію – checkbox, ціна версії – input [type=”text”], опис версії – textarea; вибір фотографій товару – input [type=”file”] з розробленим функціоналом попереднього перегляду завантаженої фотографії і фіксацією першої фотографії, як основної; опублікувати товар одразу ж після його створення – checkbox.

Сторінка із списком замовлень містить таблицю, у якій в порядку від найостаннішого замовлення виведена інформація про кожне з них. Для функціоналу MVP моделі цього достатньо. На наступному етапі розробки сторінка замовлень має зазнати суттєвих змін. Планується додавання статусу «новий», «в обробці», «переданий у службу доставки», «завершений», «відмова». Під кожен статус планується відвести окрему сторінку. Відповідно, коли вже буде продуманий цілісно цей розділ, буде розроблений уніфікований дизайн для цього функціоналу.

1.2. Обґрунтування вибору програмного забезпечення

При виборі програмного забезпечення дуже важливу роль відігравали такі проблеми, пов’язані з цим проектом: невизначеність та необхідність у швидкій імплементації робочих компонентів.

Невизначеність пов’язана перш за все з тим, що бізнес є таким, що розвивається і пробує реалізуватися у новому для себе сегменті. Ідуть постійні експерименти і зміни в продукції і її асортименті. Тому не можливо заздалегідь на даному етапі спроектувати повністю усю систему і врахувати увесь необхідний функціонал, який має бути в неї закладений. Крім того, це мій перший досвід написання бекенду та інтеграції баз даних з реальним застосунком.

Саме тому я вирішив іти за методом розробки програмного забезпечення Scrum – на кожному етапі реалізувати лише цілісний функціонал за принципом 20% функціоналу становить 80% корисності сервісу, і який можна одразу запускати для реального використання.

Необхідність у швидкій імплементації робочих компонентів напряму пов’язана з описаною вище проблемою.

Враховуючи ці фактори, а також технічні аспекти ефективної реалізації саме веб-сревісу, було вирішено реалізувати бекенд на Node.js і базу даних на MongoDB, яка є NoSQL (не зважаючи на те, що в курсі Баз Данних ми вивчали SQL).

Проте саме така комбінація є, на мою думку, найкращою для даного проекту (веб-сервісу і по сумісносності інтернет-магазину) і її ефективність підтверджена такими компаніями як Twitter, Uber, Wallmart та інші, де критичним є час подачі контенту для великої кількості користувачів.

По-перше, іде інтеграція довкола єдиної мови програмування JavaScript, яка використовується і на фронтенді, і на бекенді, і для роботи з базою даних. Це пришвидшує швидкість і якість розробки веб-сервісів.

По-друге, в нерелятивістських базах даних не вимагається задання попередньої структури, дані можуть зберігатися в довільному форматі. NoSQL бази даних набагато зручніше модифікувати і масштабувати ніж SQL. Це саме те, що необхідно для даного проекту, в якому не можливо передбачити все наперед і постійно потрібно змінювати структуру на основі нових задач і отриманого досвіду.

По-третє, в MongoDB дані зберігаються у форматі BJSON (Binary JavaScript Object Notation), але по суті ми передаємо до бази даних і отримуємо

з неї дані у форматі JSON. Тому для вебу, де основним форматом обміну даними став JSON, це надзвичайно ефективно. Адже ми зберігаємо об’єкти в такому ж форматі, в якому і працюємо з ними, і не потрібно витрачати час на конвертацію сутності при читанні з бази даних чи записі до неї.

Таким чином, обраний функціонал закладає в основу проекту великий потенціал для нарощування потужності на майбутнє. При зростанні популярності сервісу можна буде досягти високої ефективності у регулюванні навантаження на ресурс та підключити, за необхідності, такі фреймворки як React чи Angular.

РОЗДІЛ 2. АРХІТЕКТУРА ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ БД

2.1. Аналіз функціонування та організаційні засади підприємства

Оскільки для інтеграції бази даних з сервером використовувався Mongoose, то на основі поставленої задачі було прийнято рішення організувати базу даних з використанням схем. Таким чином, оскільки про деякі поля ми знали точно, якого формату вони повинні бути, то для них ми це прописували явно. Поля, які ж могли мати довільну природу, встановлювалися як тип Mixed.

2.2. Проектування структури БД

Для роботи з товарам вирішено розробити колекції для категорій, версій, та параметрів товару.

```
const productCategorySchema = new Schema({
    name: {
        type: String,
        required: true,
        unique: true
    },
    pageName: {
        type: String,
        required: true,
        unique: true
    },
    description: {
        type: String,
        required: true
    }
});

const productVersionSchema = new Schema({
    name: {
        type: String,
        required: true,
        unique: true
    }
});

const productParameterSchema = new Schema({
     name: {
         type: String,
         required: true,
         unique: true
     },
    type: {
        type: String,
        required: true,
```

```
    },
    values: []
});
```

Це ті значення, які повинні бути вбудованими в логіку системи. При оновленні їх значень і перезапуску серверу відбувається їх одноразове оновлення на нові значення:

```
emitter.once('dbUpdateEvent', next =>
{
    ProductCategories.deleteMany({})
        .then(res => {console.log('Drop productCategory collection res: ', res);},
                err => next(err))
        .catch(err => next(err));

    ProductCategories.create(
        [{name: "Комплект формы", pageName: "sets", description: "Полные комплекты
тренировочной формы"},
            {name: "Топики", pageName: "tops", description: "Топики по отдельности"},
            {name: "Юбки", pageName: "skirts", description: "Юбки по отдельности"}])
        .then(categories => {console.log('categories Created ', categories);},
                err => next(err))
        .catch(err => next(err));


    ProductVersions.deleteMany({})
        .then(res => {console.log('Drop productVersions collection res: ', res);},
            err => next(err))
        .catch(err => next(err));

    ProductVersions.create(
        [{name: "color line"}, {name: "black line"}])
        .then(versions => {console.log('ProductVersions Created ', versions);},
            err => next(err))
        .catch(err => next(err));

    ProductParameters.deleteMany({})
        .then(res => {console.log('Drop ProductParameters collection res: ', res);},
            err => next(err))
        .catch(err => next(err));

    ProductParameters.create([
        {name: "Цвет", type: "radio", values:
["черный","красный","салатовый","малиновый","сиреневый","белый"]},
        {name: "Размер", type: "multi-radio", values: [
                {sizeCategory: "Дети", values: [1,2,3,4,5]},
                {sizeCategory: "Взрослые", values: ["XS","S","M","L"]}
            ]},
        {name: "Длина", type: "number", values: [{min: 1}]},
        {name: "Количество", type: "number", values: [{min: 1}]}
    ])
        .then(versions => {console.log('ProductParameters Created ', versions);},
```

```
            err => next(err))
        .catch(err => next(err));
});
```

Схема товарів мала наступний формат:

```
const productSchema = new Schema({
    name: {
        type: String,
        required: true,
        unique: true
    },
    page: {
        type: String,
        required: true,
        unique: true
    },
    category: {
        type: mongoose.Schema.Types.ObjectID,
        ref: "ProductCategory"
    },
    materials: [String],
    parameters: [{
        type: mongoose.Schema.Types.ObjectID,
        ref: "ProductParameter"
    }],
    price: [{
        version: {
            type: mongoose.Schema.Types.ObjectID,
            ref: "ProductVersion"
        },
        price: Currency
    }],
    cardDescription: {
        type: String,
        required: true
    },
    description: [{
        version: {
            type: mongoose.Schema.Types.ObjectID,
            ref: "ProductVersion"
        },
        text: String
    }],
    images: [String],
    mainImage: {
        type: String,
        required: true
    },
    display: {
        type: Boolean,
        default: true
    }
},
```

```
{
    timestamps: true
});
```

Схема замовлень мала такий формат:

```
const orderSchema = new Schema({
    customer: {
        type: String,
        required: true
    },
    phone: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    certificate: String,
    country: {
        type: String,
        required: true
    },
    comment: String,
    totalSum: Currency,
    delivery: {},
    bucket: []
},
{
    timestamps: true
});
```

2.3. Життєві цикли БД

Попереднє планування: попереднє планування включало в себе аналіз, формування структури, зв’язків та моделі даних.

Перевірка здійсненності: у наявності є встановлений сервер Node.Js (та всіх компонент з package.json).

Визначення вимог: вимоги визначені в постановці задачі. База даних охоплює всі елементи, що були вказані там.

Реалізація: реалізація включала в себе написання програмного коду на JavaScript із використанням синтаксису запитів в MongoDB (за допомогою Mongoose).

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ВЗАЄМОДІЇ З БД

3.1. Інструкція користувача (адміністратора)

При переході за адмін-посиланням адміністратор потрапляє до адмін-панелі. Зліва є меню в якому він може перейти на сторінку додавання товару (Рисунок 3.1).

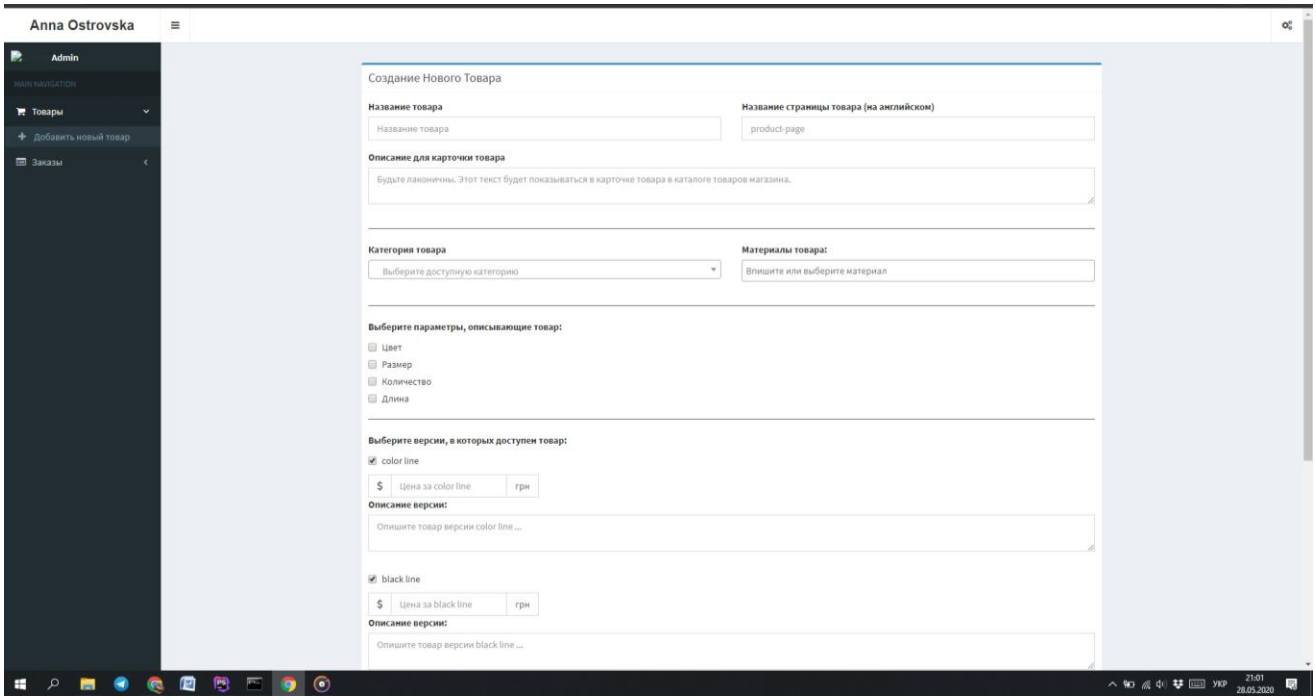

Рисунок 3.1 – Сторінка додавання товару.

Розглянемо детальніше кожну секцію. Перша секція – створення картки товару (Рисунок 3.2).

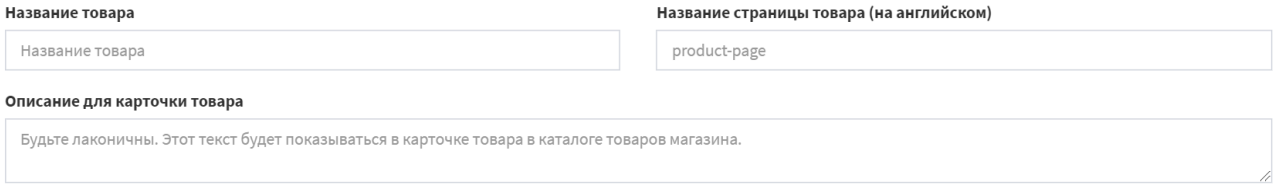

Рисунок 3.2. – Секція створення картки товару

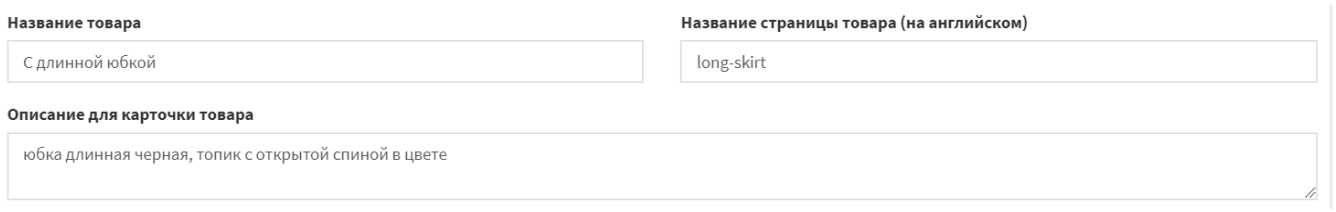

Рисунок 3.3. – Приклад заповнення картки товару

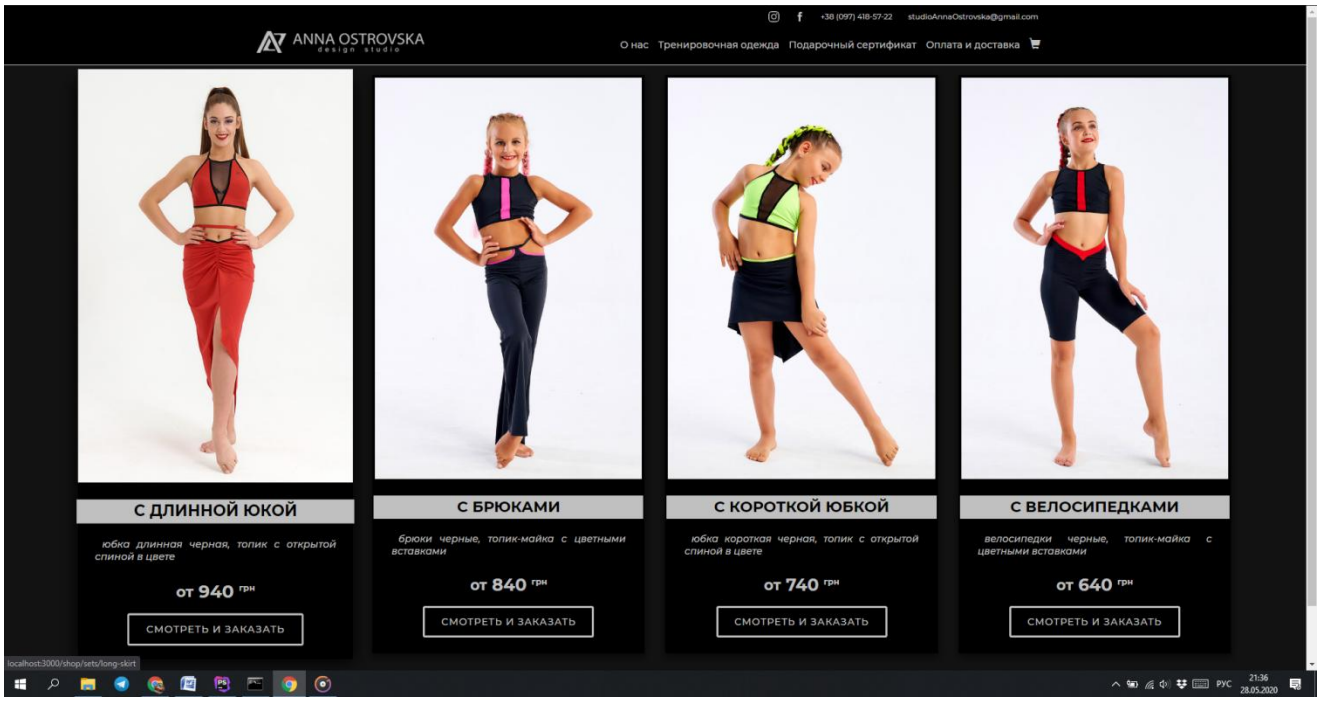

Рисунок 3.3. – Приклад вигляду картки товару

В наступній секції необхідно вибрати категорію товару та матеріал (Рисунок 3.4.). Категорія обирається одна з доступних варіантів. Матеріали можна вводити вручну. З бази даних витягуються всі унікальні значення, які були використанні раніше при створенні товарів, і пропонуються користувачу для вибору. Якщо такого немає, то користувач вводить новий матеріал.

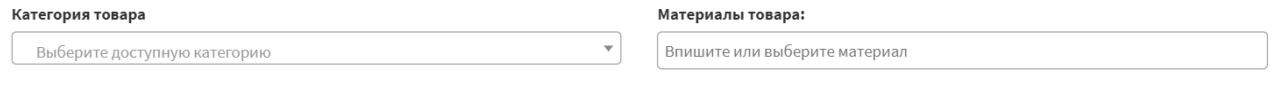

Рисунок 3.4. – Секція вибору категорії і матеріалу

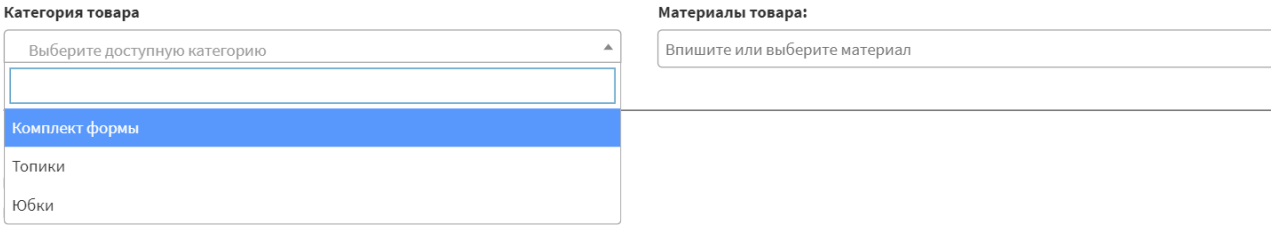

Рисунок 3.5. – Приклад вибору категорії

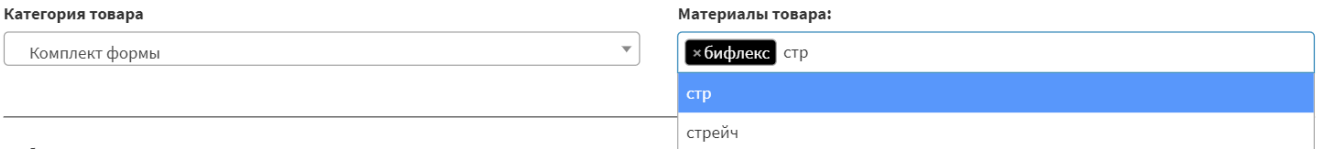

Рисунок 3.6. – Приклад вибору матеріалів

Наступна секція – вибір параметрів, які клієнт повинен заповнити для даного товару.

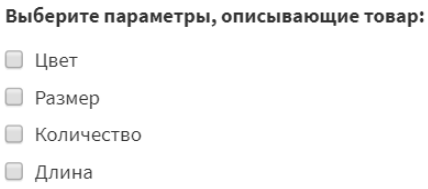

Рисунок 3.7. – Секція вибору параметрів

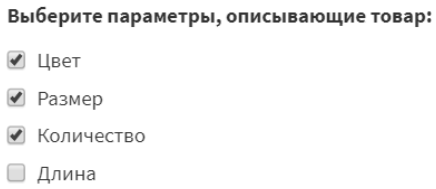

Рисунок 3.8. – Приклад вибору параметрів

Наступна секція – опис версій (Рисунок 3.9).:

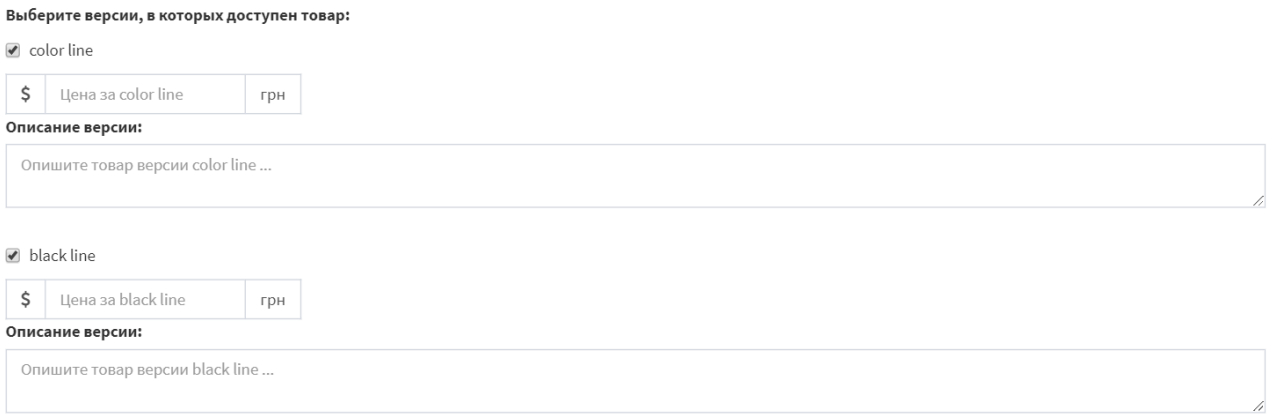

Рисунок 3.9. – Секція опису версій

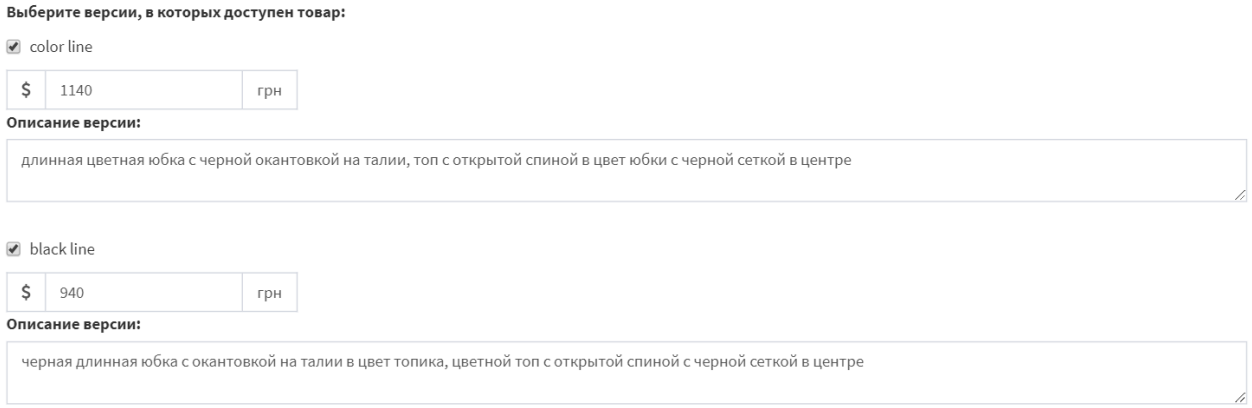

Рисунок 3.10. – Приклад заповнення опису версій

Наступна секція – вибору фотографій (Рисунок 3.11). Перша добавлена фотографія – головна і використовується для картки товару. Реалізована перевірка, яка не дасть завантажити файли не допустимого розширення. Натискаючи на кнопку, додається можливість додати ще одну картинку. Картинку можна видалити, натиснувши на хрестик.

Рисунок 3.11. – Секція вибору фотографій

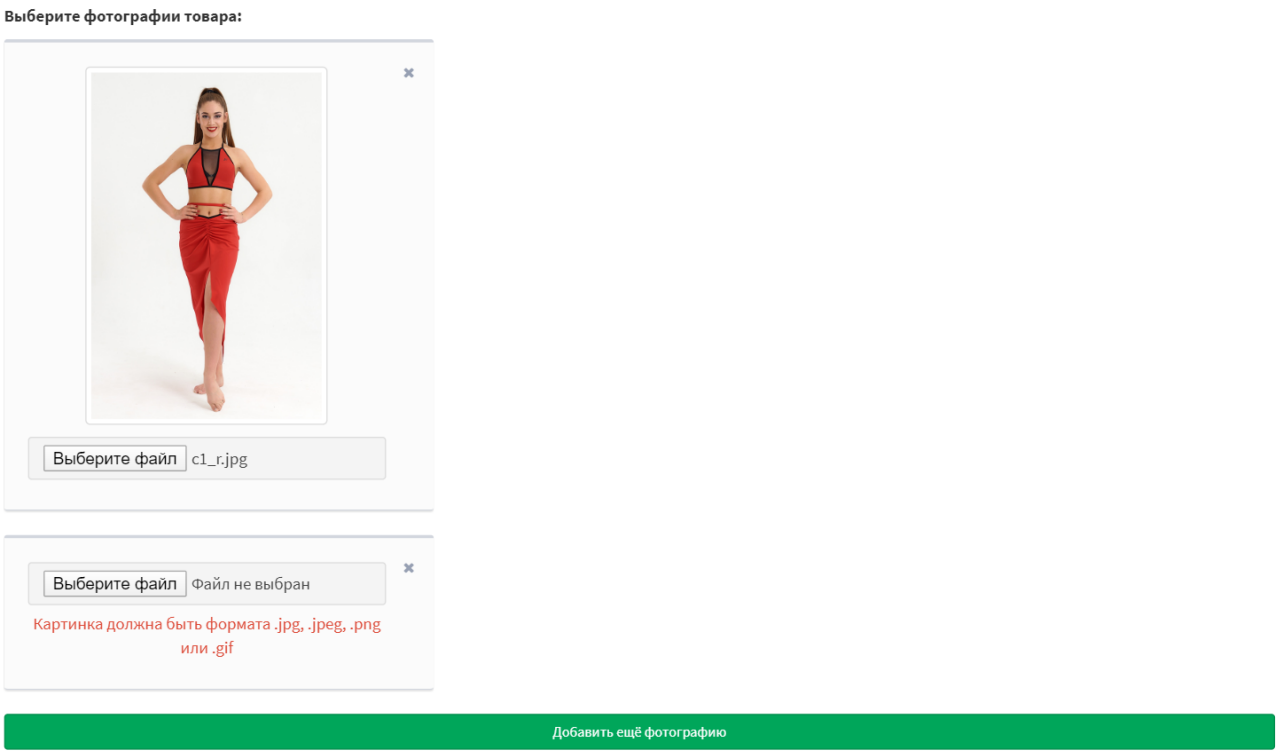

Рисунок 3.12. – Приклад вибору фотографій

В кінці натискаємо опублікувати (Рис. 3.13):

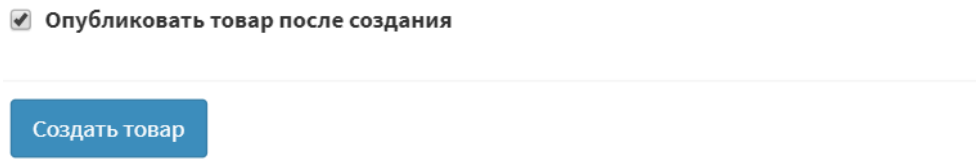

Рисунок 3.13. – Завершення створення

Після натискання кнопки відправиться AJAX запит на збереження товару без перезавантаження сторінки. Якщо запит пройшов успішно, форма оновиться.

Після створення товар на сайті буде мати такий вигляд (Рис.3.14)

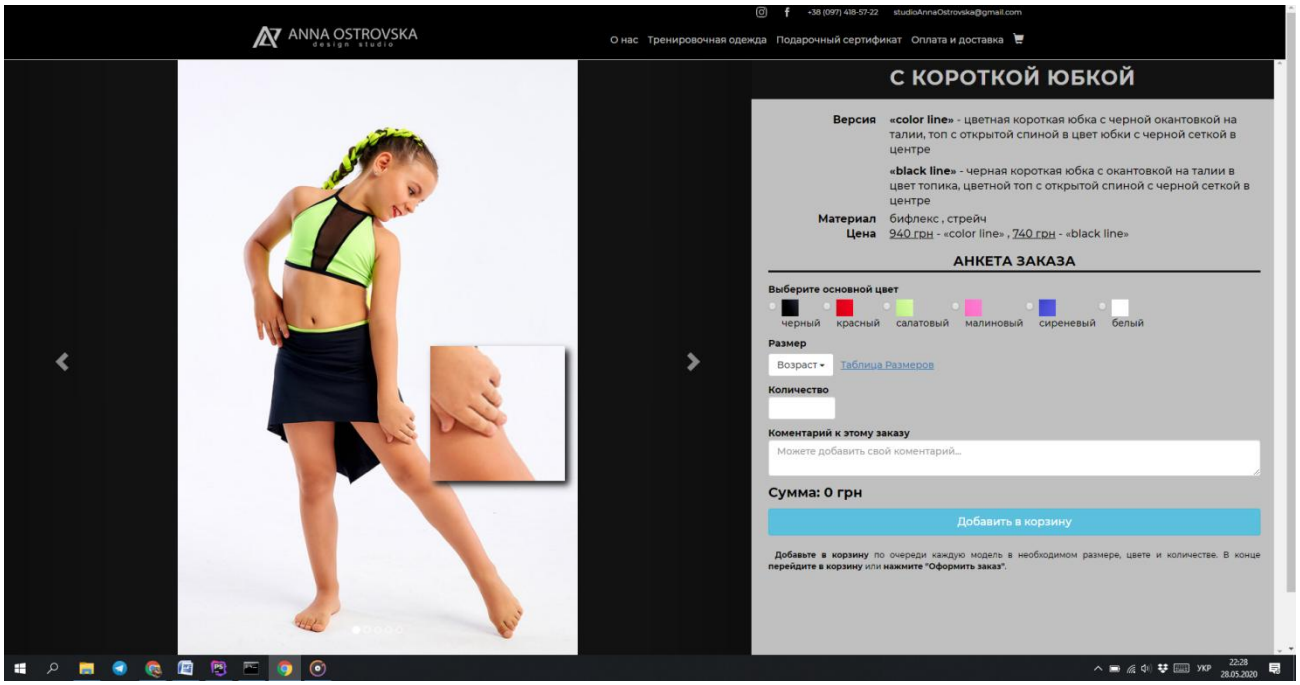

Рисунок 3.14. – Вигляд сторінки товара

Вигляд сторінки із замовленнями (чому вона має саме такий вигляд пояснено в главі 1):

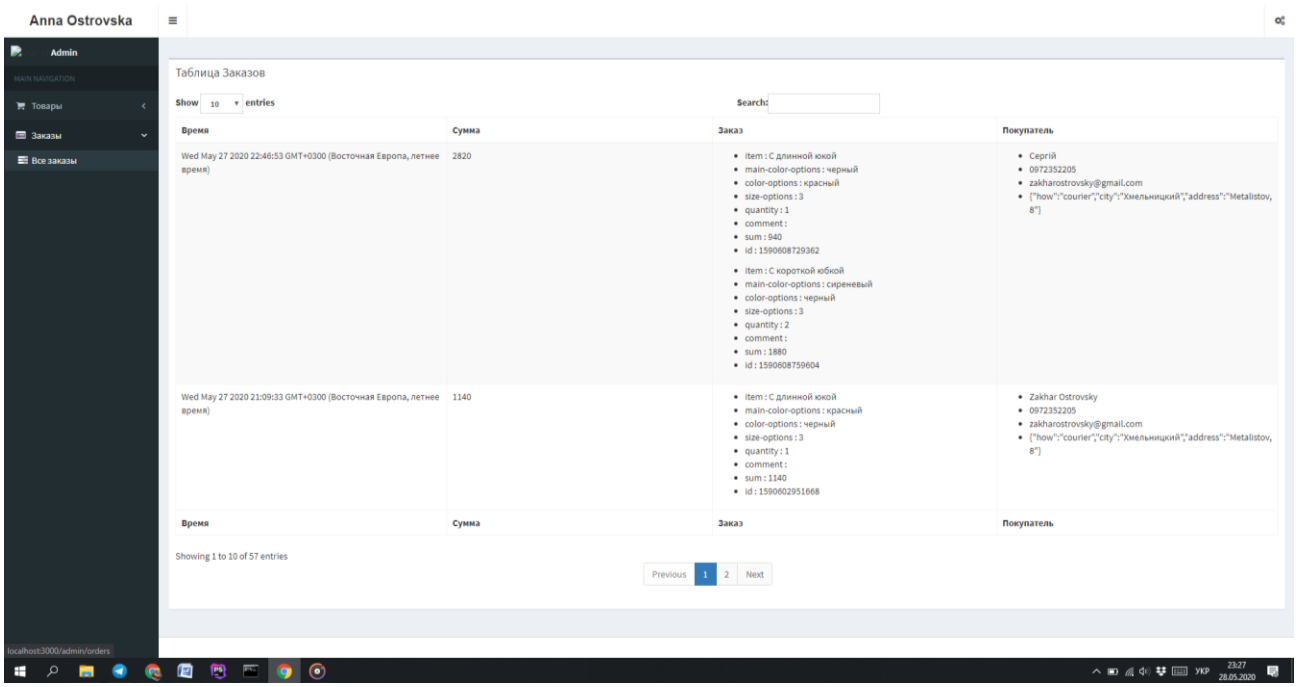

Рисунок 3.15. – Вигляд сторінки замовлень

3.2. Інструкція користувача (покупця)

Головна сторінка має такий вигляд:

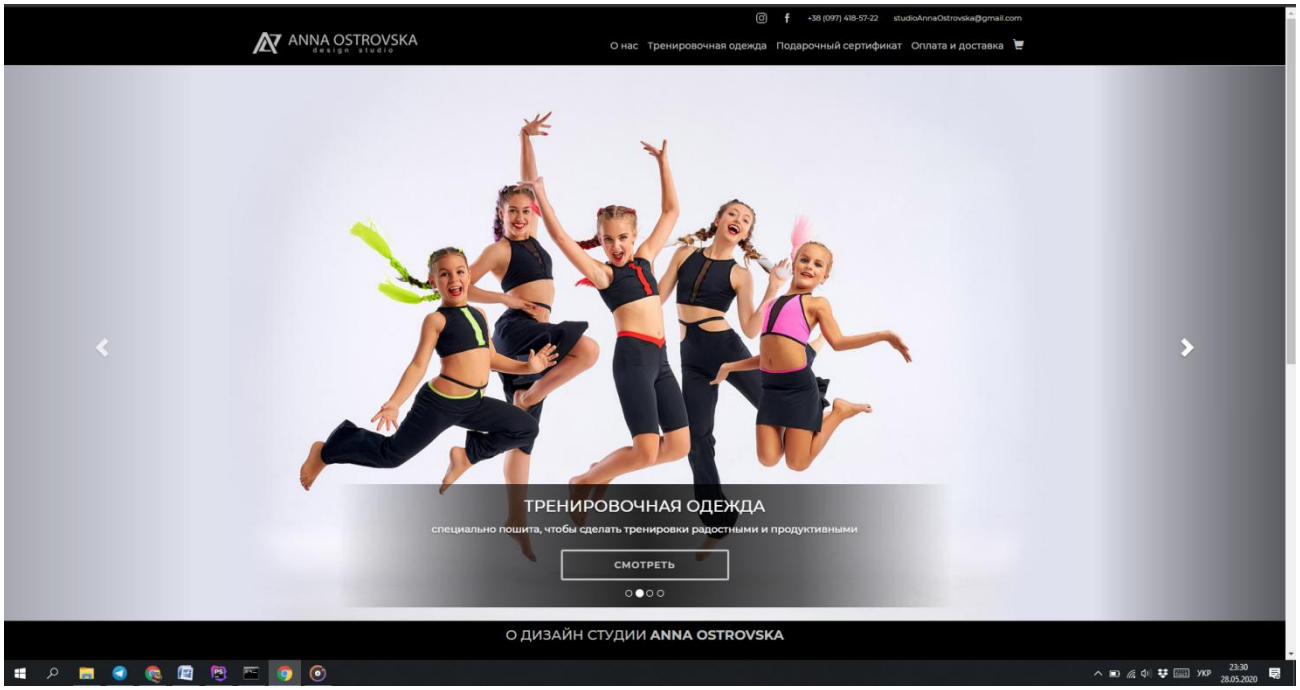

Рисунок 3.16. – Головна сторінка

При натискані на Тренировочная одежда в меню відбувається перехід на сторінку, де з бази даних завантажуються категорії товарів.

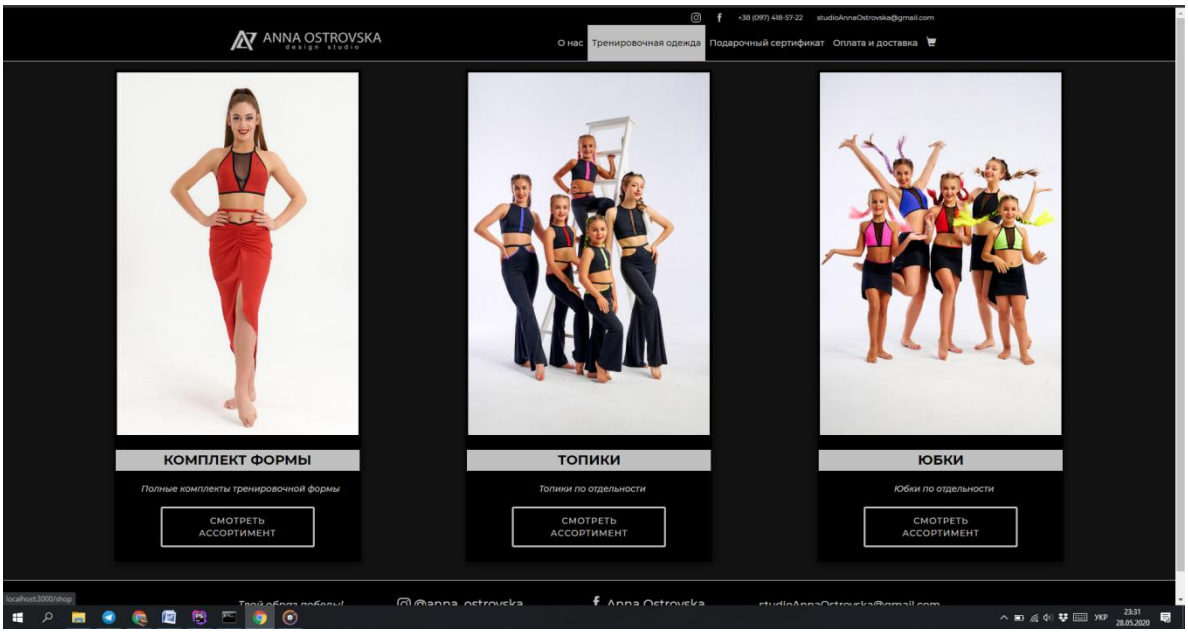

Рисунок 3.17. – Сторінка «Тренировочная одежда»

Після того як користувач обирає категорію товарів, зз бази даних формується вибірка і завантажується сторінка з товарами по обраній категорії (Рисунок 3.3.).

Користувач обирає бажаний товар і з бази даних формується вибірка усієї необхідної інформації по товару і рендериться сторінка, як наприклад Рисунок 3.14.

Далі користувачу необхідно обрати параметри товару. Якщо він пропустить параметр – система не дасть додати до кошика (Рисунок 3.18).

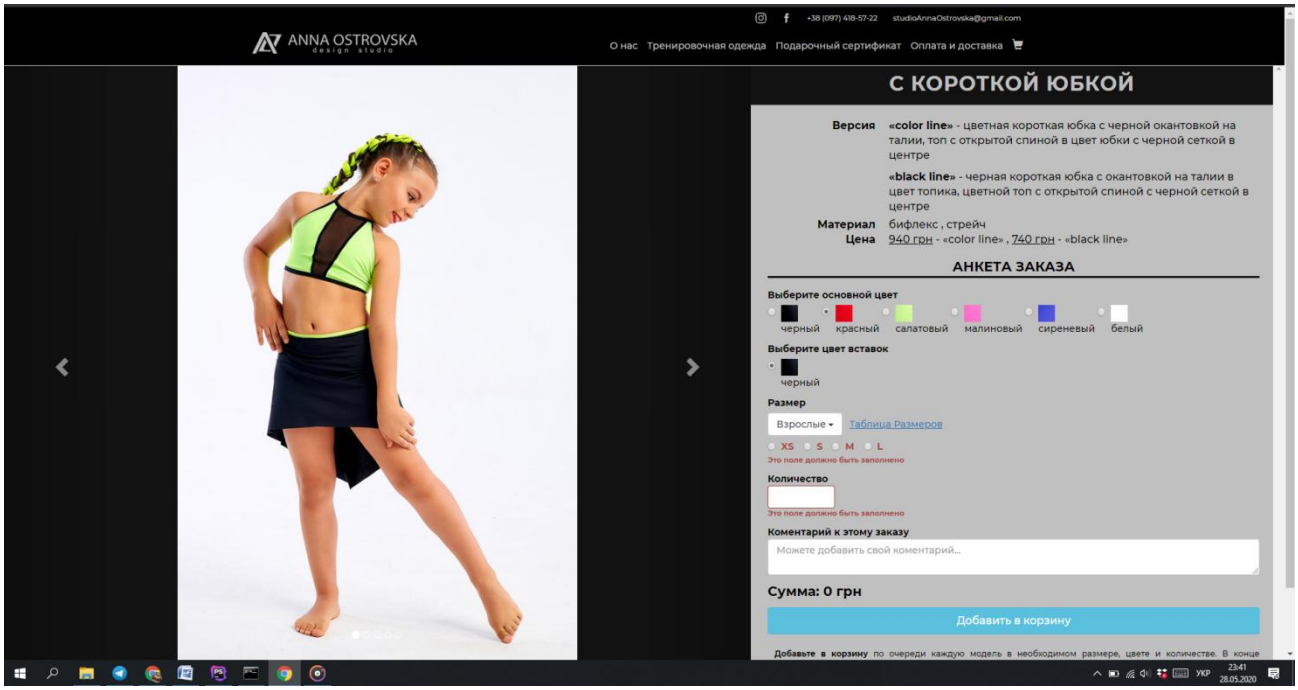

Рисунок 3.18. – Вибір параметрів

Після заповнення всіх параметрів система порахує суму. Далі натисніть додати до кошика (Рисунок 3.19).

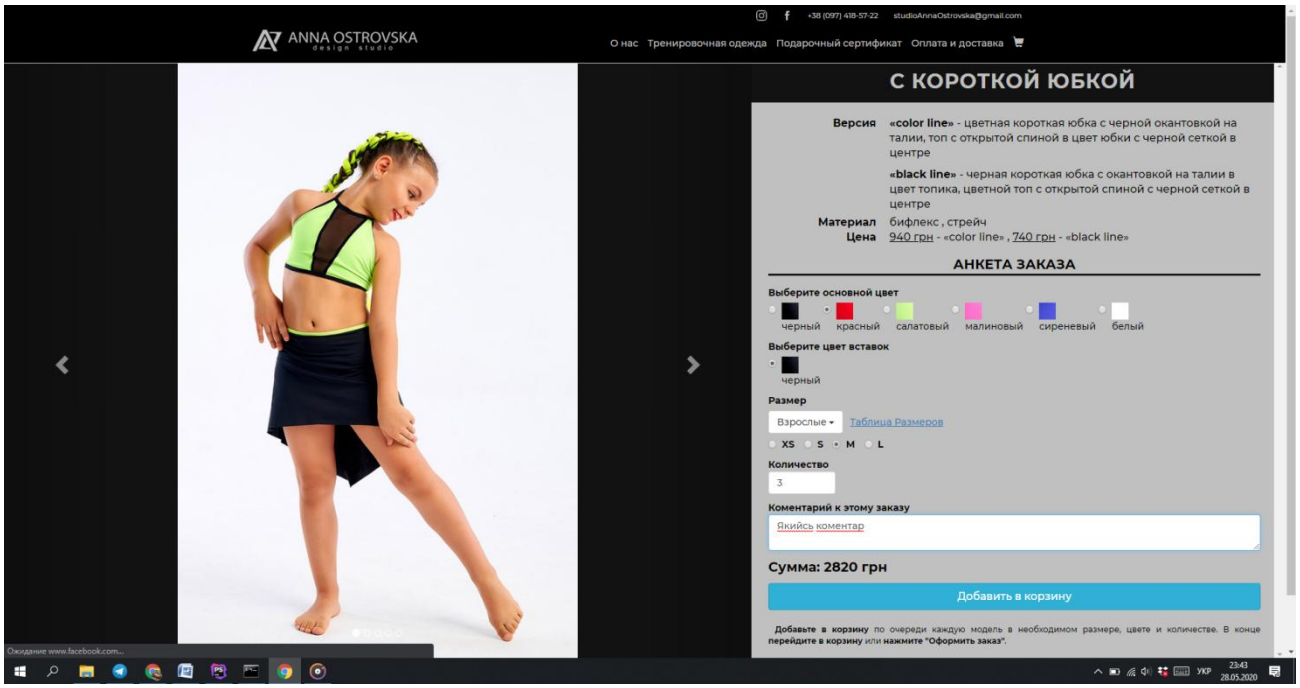

Рисунок 3.20. – Вибір параметрів (правильний)

Після натискання клавіши відбудеться анімація додавання елемента в кошик, він стане зеленого кольору і міститиме лейбу з кількістю товарів, що містяться в ньому. Це зроблено для того, щоб користувач зрозумів, що товар додався успішно і звернув увагу на місце розташування кошика, де можна переглянути своє замовлення. Крім того стає доступною кнопка «Оформить заказ». Якщо користувач виконав усі замовлення, то може нажати на неї для оформлення замовлення. Інакше – він може додати в кошик той же товар з іншими параметрами або перейти на сторінку іншого товару.

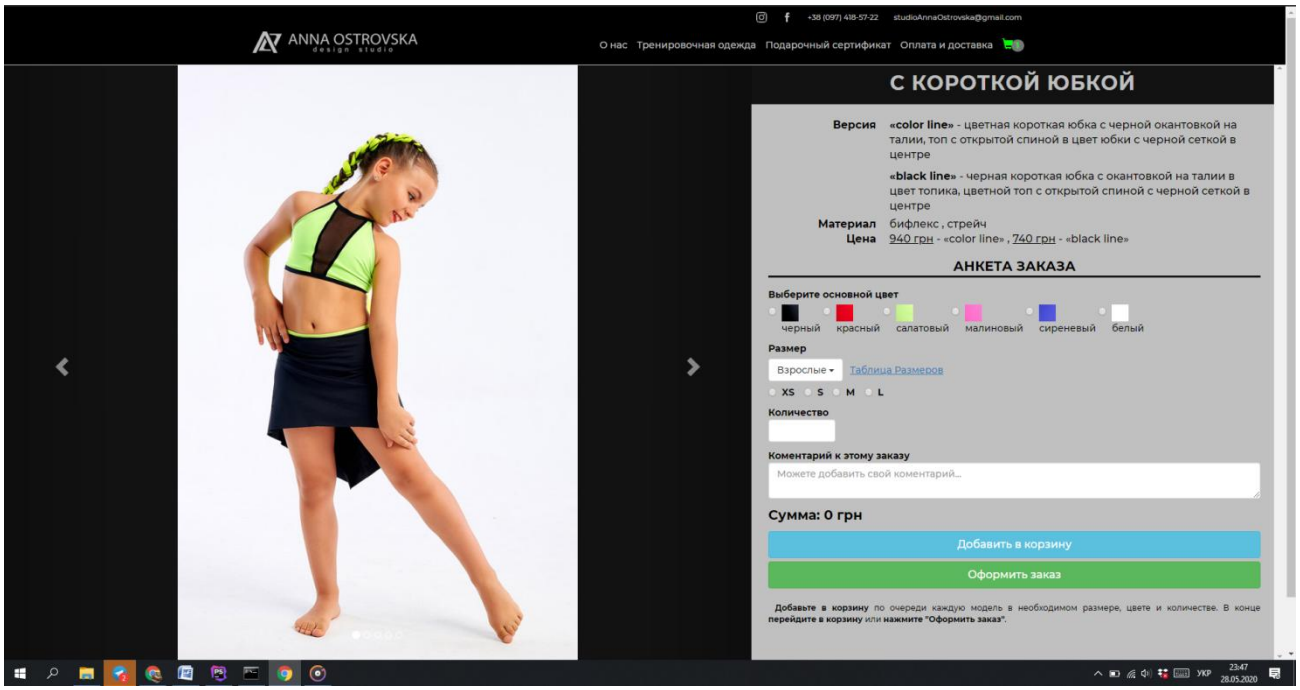

Рисунок 3.21. – Додавання до кошику

Якщо натиснути на кошик, то можна побачити обрані товари (Рисунок 3.22).

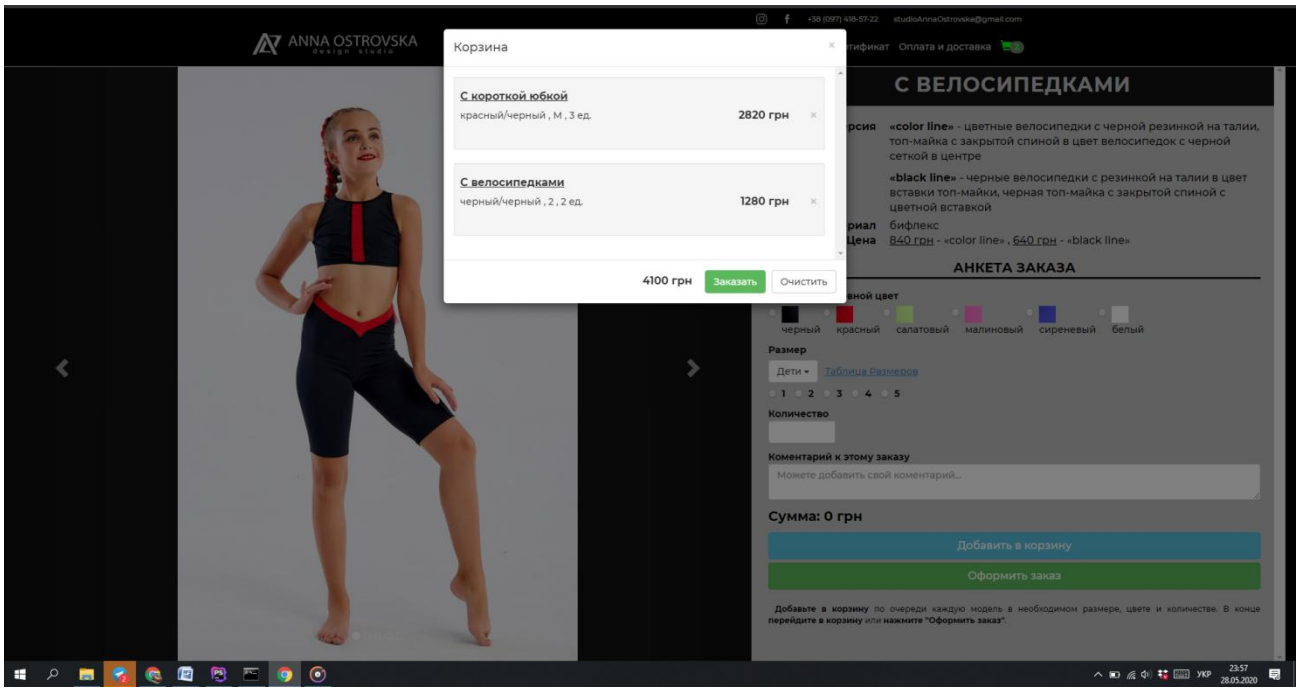

Рисунок 3.22 – Кошик

При натисканні на «Заказать» в кошику або «Оформить заказ» на сторінці товару відбувається пере адресація на сторінку Оформлення замовлення Рисунок 3.23.

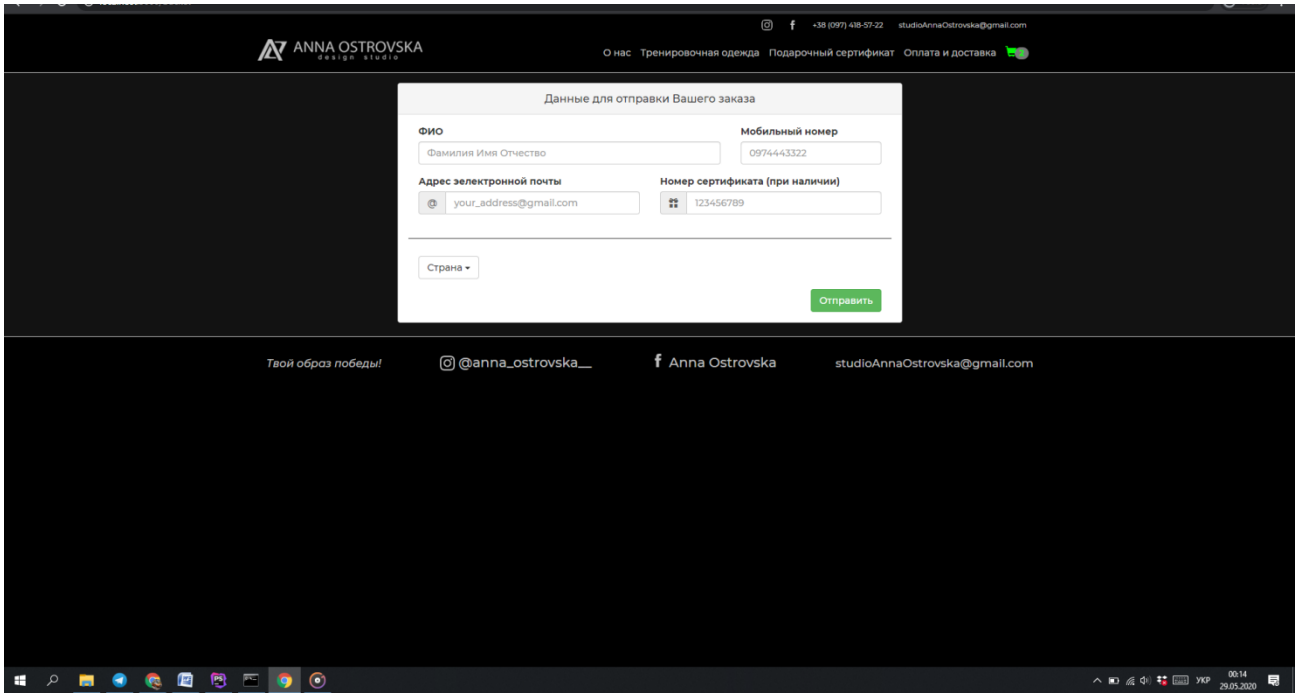

Рисунок 3.23 – Сторінку оформлення замовлення

Після заповнення контактних даних користувач заповнює анкету доставки. Якщо він спробує натиснути «Оформить», але обов’язкові поля не будуть заповненні, то система підкреслить їх і не дасть зробити відправку.

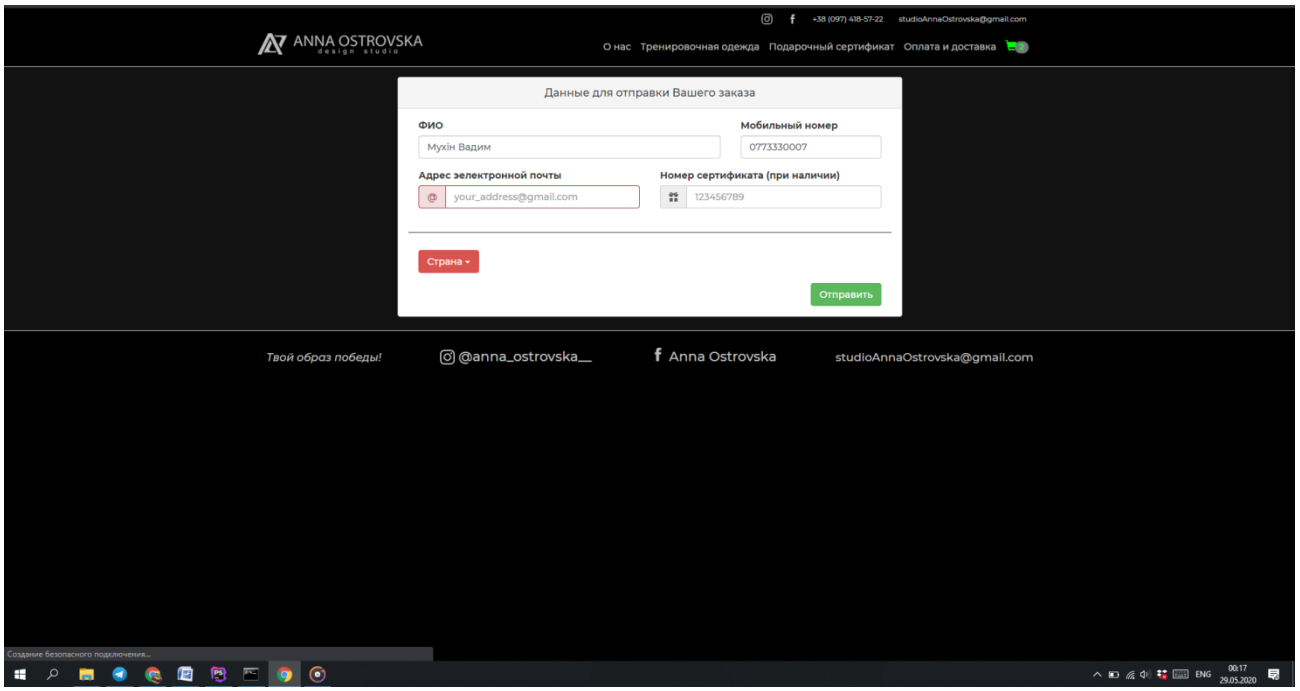

Рисунок 3.24 – Неповна інформація

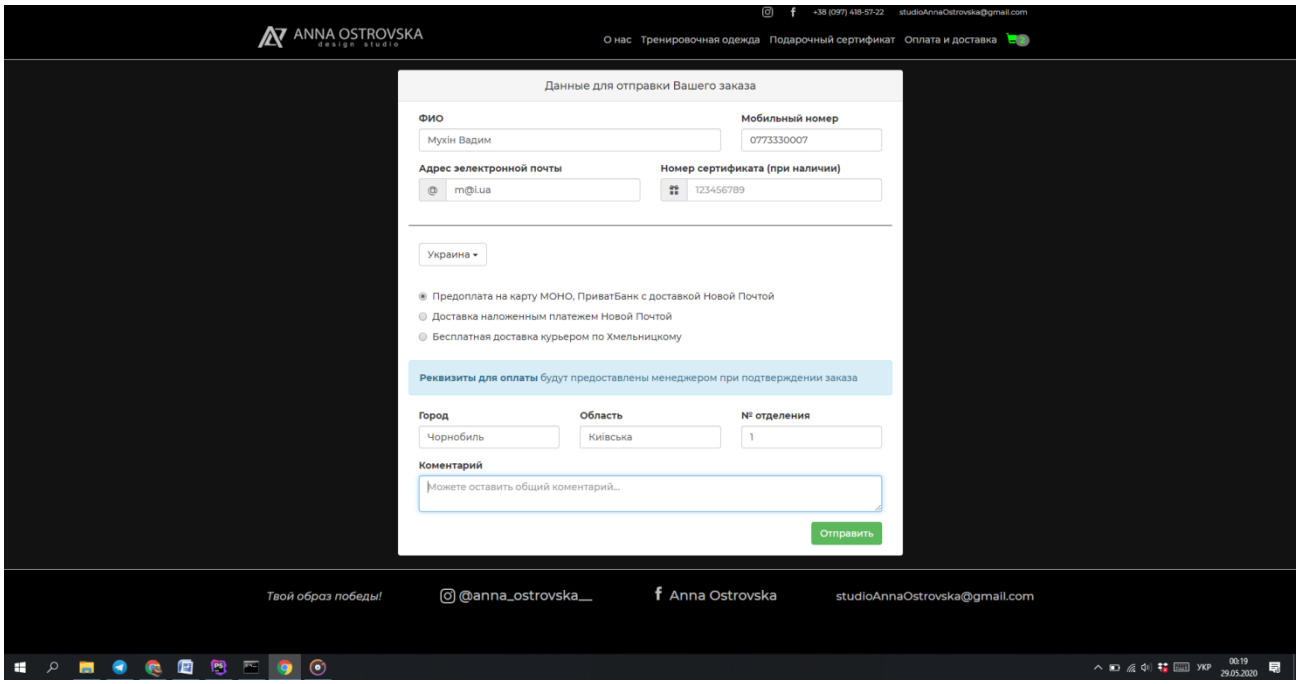

Рисунок 3.25 – Повна інформація

Після натискання «Отправить» посилається AJAX запит на сервер. Якщо успішний, то відбувається очищення кошика і видається наступне повідомлення

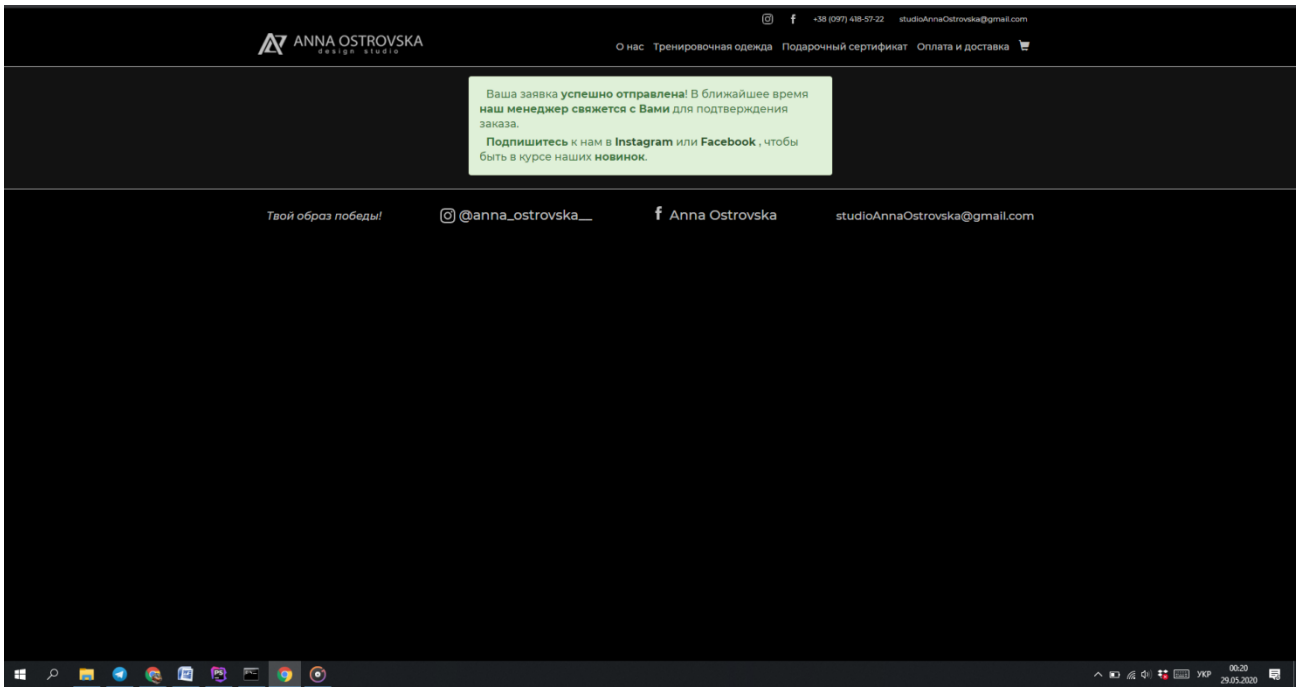

Рисунок 3.26 – Успішне замовлення

3.3. Реалізація механізмів запитів

Наведемо приклад роботи серверу при отриманні GET запиту на сторінку при натискані кнопки «Тренировочная одежда».

```
router.route('/')
    .get( (req, res, next) =>
    {
        const renderParams = {
            head: {
                og: {
                    title: 'Тренировочная форма от Анны Островской',
                    url: 'shop',
                    description: 'специально пошита, чтобы сделать тренировки радостными
и продуктивными',
                    image: 'og_main.jpg'
                },
                title: 'Тренировочная одежда - ANNA OSTROVSKA',
                description: '4 вида тренировочной одежды Anna Ostrovska и 6 цветовых
комбинаций. Выберите свою неповторимую форму для танца!',
            }
        };

        ProductCategories.find({})
            .then(categories =>
            {
```

```
                renderParams.categories = categories;

                res.render('pages/shop', renderParams);
            },
                err => next(err))
            .catch(err => next(err));
    });
```

І також фрагмент коду, з.ejs файлу, який відповідає за рендеринг сторінки на основі отриманих даних

```
<% categories.forEach(category => { %>
<div class="col-xs-12 col-sm-6 col-md-4">
    <div class="item-bg"">
        <div class="thumbnail item-wrapper">
            <a class="image-wrapper" href="/shop/<%= category.pageName %>"
rel="details">
                <img class="im" src="/images/categories/<%= category.pageName %>.jpg"
alt="Категория товаров <%= category.name %> для танца Anna Ostrovska">
            </a>
            <div class="caption">
                <h2 class="costume-name text-uppercase text"><%= category.name %></h2>
                <p class="buy-costume-info"><i><%= category.description %></i></p>
                <p><a href="/shop/<%= category.pageName %>" class="transparent-button
text-uppercase" rel="details">смотреть ассортимент</a></p>
            </div>
        </div>
    </div>
</div>
<% }); %>
```

3.4. Випробування розробленої програми

Програма пройшла успішне тестування закладеного у неї функціоналу.

ВИСНОВОК

Розроблена MVP модель є робочою і її функціонал відповідає задачам, які ставилися на початку роботи.

Завдяки розробленій MVP моделі було отримано перший досвід повноцінної фулстек розробки веб-сервісу та цінне розуміння того, що саме потрібно враховувати з самого початку при проектуванні системи і що необхідно додати для того, щоб можна було вийти з системою в продакшн. Завдяки правильно обраному підходу до розробки проекту, я на ранній стадії виявив деякі проблемні місця, які породжує розроблена структура мого серверу. Тому тепер зможу переробити її до того, як було вже написано багато функціоналу прив’язаного до попередньої структури. Наступний етап розробки планую зробити у більш спокійній обстановці і з достатньою кількістю вільного часу, для того щоб провести повторний аналіз потреб і способів їх реалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Server-side Development with Node.js, Express and MongoDB. [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.org/learn/server-side-nodejs
2. Mozilla Developer Network [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org
3. Node.js Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/docs/latest-v13.x/api
4. Mongoose Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://mongoosejs.com/docs/api
5. Bootstrap 3 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/3.3/
6. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура й правила оформлення; введ. 1996—01—01 – 6 с. – (Державний стандарт України).