МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему:

«РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ

ДЛЯ ПРОДАЖУ РИБАЛЬСЬКОГО СПОРЯДЖЕННЯ»

Виконав: здобувач групи ІТ-41
спеціальності 126 «Інформаційні системи та технології»

Чудійович Н. В.
(прізвище та ініціали)

Керівник: Шувар Б.І.
(прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
“____” _____202 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Чудійович Назарій Васильович
(прізвище, ім’я, по батькові)

1. Тема роботи *«Розробка інтернет-магазину для продажу рибальського спорядження»*
керівник роботи *Шувар Б.І., к.е.н., доцент*
(наук.ступінь, вч. звання, прізвище, ініціали)
затверджені наказом Львівського НУП *від 27.11.2023 року № 641/к-с*
2. Строк подання студентом роботи *10 червня 2024 року*
3. Вихідні дані до роботи: *характеристика предметної сфери; вихідні дані та вимоги до роботи інтернет-магазину, опис використаних технологій та мов програмування, науково-технічна і довідкова література.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ
Огляд використаних технологій
Проектування інтернет-магазину
Програмна реалізація проекту
Охорона праці та безпека в надзвичайних ситуаціях
Висновки
Список використаних джерел

5. Перелік графічного матеріалу
Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3,4,5	*Шувар Б.І., к.е.н., доцент*			
6	*Городецький І. М., к.т.н., доцент*			

7. Дата видачі завдання *28 листопада 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Огляд використаних технологій:*	*28.11.2023 – 31.12.2023*	
2	*Проектування інтернет-магазину*	*01.01.2024 – 28.02.2024*	
3	*Програмна реалізація проекту*	*01.03.2024 – 30.04.2024*	
4	*Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях*	*01.05.2024 – 14.05.2024*	
5	*Завершення оформлення розрахунково-пояснювальної записки та презентаційного матеріалу*	*15.05.2024 – 31.05.2024*	
6	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*01.06.2024 – 10.06.2024*	

Здобувач _____ *Чудійович Н.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Шувар Б.І.,* _____
(підпис) (прізвище та ініціали)

Зміст роботи

ВСТУП

Електронна комерція є однією з найбільш динамічних та швидкозростаючих галузей сучасного бізнесу. З кожним роком все більше споживачів віддають перевагу здійсненню покупок в Інтернеті через зручність, доступність та розмаїття пропозицій. Відповідно, інтернет-магазини стають невід'ємною складовою успішного бізнесу, незалежно від його розміру чи галузі.

Ця рсова робота присвячена розробці інтернет-магазину на базі фреймворка Ruby on Rails. Ruby on Rails, або просто Rails, відомий своєю простотою та ефективністю в розробці веб-додатків. Вибір цього фреймворку для реалізації інтернет-магазину обумовлений його гнучкістю, широким спектром готових рішень та високою продуктивністю.

В рамках цієї роботи буде розглянуто процес розробки інтернет-магазину з нуля, включаючи проектування бази даних, створення моделей та контролерів, реалізацію функціональності пошуку товарів, каталогізації, корзини покупок та системи оплати. Також будуть розглянуті аспекти оптимізації, безпеки та масштабованості додатку.

Мета цієї роботи - детально вивчити процес розробки інтернет-магазину на базі Ruby on Rails та продемонструвати практичні навички роботи з цим фреймворком. Результатом буде готовий інтернет-магазин, який може бути використаний як основа для подальших розробок або реального використання в комерційних цілях.

Визначення цілей та об'єктів дослідження.

1. **Цілі дослідження**:
    - Розробити функціональний інтернет-магазин на базі Ruby on Rails.
    - Вивчити процес розробки веб-додатків на Ruby on Rails.
    - Дослідити можливості інтеграції платіжних систем у веб-додатки.
    - Провести аналіз ефективності та швидкодії створеного магазину.
    - Розглянути питання безпеки веб-додатків та методи їхнього захисту.

2. **Об'єкти дослідження**:
    - Ruby on Rails як фреймворк для веб-розробки.
    - Моделі, контролери та вигляди веб-додатку.
    - Система аутентифікації та авторизації користувачів.
    - Інтеграція платіжних систем у веб-додаток.
    - Оптимізація та швидкодія веб-додатку.

Обґрунтування актуальності обраної теми.

Обрана тема є актуальною з кількох причин:

Зростання електронної комерції: З кожним роком електронна комерція стає все більш популярною та загальновживаною формою торгівлі. За даними досліджень, обсяги онлайн-продажів постійно зростають, що робить розробку інтернет-магазинів важливою галуззю веб-розробки.

Популярність Ruby on Rails: Ruby on Rails залишається одним з найпопулярніших фреймворків для веб-розробки. Він пропонує зручну та ефективну розробку веб-додатків завдяки своїй простоті, гнучкості та великій кількості готових рішень.

Потреба в ефективних рішеннях: Бізнесам потрібні ефективні та швидкодіючі інтернет-магазини для привертання клієнтів та збільшення продажів. Розробка інтернет-магазину на базі Ruby on Rails може забезпечити швидкий розвиток та надійність додатку.

Можливості для інновацій: Інтернет-магазини постійно розвиваються та вдосконалюються, додаючи нові функції та сервіси для полегшення покупок клієнтів. Розробка інтернет-магазину на базі Ruby on Rails відкриває широкі можливості для впровадження інновацій та покращень.

Отже, обрана тема є актуальною та важливою в контексті сучасних вимог електронної комерції та розвитку веб-розробки.

1. РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ:

1.1. Огляд існуючих інтернет-магазинів.

Огляд існуючих інтернет-магазинів є важливим етапом при розробці власного магазину, оскільки дозволяє вивчити та аналізувати різноманітні підходи, функціональність та дизайн.

Ось деякі ключові аспекти, які можна врахувати під час огляду:

Категорії товарів та асортимент: Розгляньте різноманіття категорій товарів у різних магазинах, а також диверсифікацію асортименту. Приділіть увагу тому, як категорії організовані та як легко користувачі можуть знаходити потрібні товари.

Дизайн та користувацький інтерфейс: Оцініть дизайн інтерфейсу користувача різних магазинів. Подивіться, які елементи дизайну використовуються для привернення уваги користувачів та полегшення навігації.

Система пошуку та фільтрації: Проведіть аналіз системи пошуку та можливостей фільтрації товарів. Розгляньте, які параметри фільтрації доступні для користувачів та наскільки ефективно працює система пошуку.

Кошик та оформлення замовлення: Вивчіть процес додавання товарів до кошика, оформлення замовлення та оплати. Подивіться, які можливості надаються користувачам для управління їхніми замовленнями та доставкою.

Відгуки користувачів та рейтинги товарів: Оцініть наявність системи відгуків користувачів та рейтингів товарів. Подивіться, як ці відгуки впливають на прийняття рішення покупцями та як вони керуються власниками магазинів.

Мобільна оптимізація: Переконайтеся, що розглянуті вами магазини мають адаптивний дизайн для мобільних пристроїв. Мобільна оптимізація є важливою для забезпечення зручного користування магазином на будь-яких пристроях.

Безпека та конфіденційність: Приділіть увагу заходам безпеки та захисту конфіденційності даних користувачів. Вивчіть, які заходи приймаються для захисту від кібератак та зловмисного використання особистих даних.

1.2. Аналіз їхніх переваг і недоліків.

Після проведення огляду існуючих інтернет-магазинів можна виділити переваги та недоліки цих платформ:

Переваги:

1. Широкий асортимент товарів: Багато магазинів пропонують великий вибір товарів у різних категоріях, що задовольняє різноманітні потреби покупців.

2. Зручна навігація: Багато магазинів мають інтуїтивно зрозумілу структуру та добре організовану навігацію, що дозволяє швидко знаходити потрібні товари.

3. Система фільтрації і пошуку: Більшість магазинів пропонують потужні системи фільтрації та пошуку, які дозволяють користувачам швидко знаходити товари за різними параметрами.

4. Зручний процес оформлення замовлення: Більшість магазинів пропонують зручні та швидкі процеси оформлення замовлення та оплати, що сприяє покращенню користувацького досвіду.

5. Відгуки та рейтинги товарів: Багато магазинів мають системи відгуків користувачів та рейтингів товарів, які допомагають покупцям зробити обдуманий вибір.

Недоліки:

1. Помірна швидкодія: Деякі магазини можуть мати повільну швидкодію завантаження сторінок, що може вплинути на користувацький досвід та збільшити ймовірність відмови від покупки.

2. Неякісний дизайн інтерфейсу: Деякі магазини можуть мати неякісний дизайн інтерфейсу, що може робити навігацію складною та призводити до поганого враження від магазину.

3. Обмежена функціональність: Деякі магазини можуть бути обмежені у функціональності, наприклад, не мати підтримки мобільних пристроїв або не надавати достатньо опцій для налаштування.

4. Низька безпека: Деякі магазини можуть мати недостатньо захищені системи безпеки, що може призвести до ризику витоку особистих даних користувачів.

5. Неякісне обслуговування клієнтів: Деякі магазини можуть мати недостатньо ефективну службу підтримки клієнтів, що може ускладнити вирішення проблем або отримання допомоги.

Аналіз переваг та недоліків існуючих магазинів допоміг визначити ключові аспекти, які потрібно врахувати при розробці власного інтернет-магазину на базі Ruby on Rails.

РОЗДІЛ 2. ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ:

2.1. Rubi

Ruby — це високорівнева інтерпретована мова програмування, створена для швидкої і простої розробки програм. Ruby поєднує в собі простоту написання коду з потужними можливостями, що робить її популярною серед розробників.

Основні характеристики Ruby

1. Об'єктно-орієнтована мова (ООП):

- Все в Ruby є об'єктом, включаючи примітивні типи даних (числа, рядки, навіть null, який представлений як `nil`).

- Підтримка основних принципів ООП: наслідування, інкапсуляція, поліморфізм.

2. Динамічна типізація:

- Типи змінних визначаються під час виконання програми, а не під час компіляції.

- Це дозволяє писати більш гнучкий і менш формальний код.

3. Синтаксична гнучкість:

- Ruby дозволяє писати код у різних стилях, що може бути корисним для різних завдань.

- Підтримка блоків, які можуть бути передані як аргументи методів, що робить код більш лаконічним і зручним для роботи з ітераціями та колекціями.

4. Гармонія з природною мовою:

- Синтаксис Ruby дуже читабельний і нагадує англійську мову.

- Це робить код легким для розуміння навіть для новачків.

5. Потужні стандартні бібліотеки:

- Ruby має велику кількість вбудованих бібліотек, які охоплюють різноманітні задачі, від роботи з файлами і мережею до веб-розробки і обробки даних.

6. Метапрограмування:

- Ruby підтримує метапрограмування, дозволяючи програмам змінювати свій код під час виконання.

- Це забезпечує високий рівень гнучкості і можливості для створення DSL (доменно-специфічних мов).

Застосування Ruby

1. Веб-розробка:

- Ruby найбільше відомий завдяки фреймворку Ruby on Rails, який дозволяє швидко створювати веб-додатки.

- Rails слідує принципам DRY (Don't Repeat Yourself) і Convention over Configuration.

2. Автоматизація та скрипти:

- Ruby часто використовується для написання скриптів автоматизації завдяки своїй простоті і потужності.

3. Розробка ігор:

- Існують фреймворки, такі як Gosu, які дозволяють розробляти ігри на Ruby.

4. Обробка даних:

- Ruby використовується для аналізу і обробки даних, завдяки бібліотекам як Nokogiri для парсингу XML та HTML, і Prawn для створення PDF-файлів.

Переваги та недоліки Ruby

Переваги:

- Простий і зрозумілий синтаксис.
- Потужні вбудовані бібліотеки.
- Висока гнучкість і можливості метапрограмування.
- Велика і активна спільнота розробників.

Недоліки:

- Виконання коду може бути повільнішим порівняно з мовами на зразок C++ чи Java.
- Споживання пам'яті може бути вищим.
- Менше бібліотек та інструментів для наукових і технічних розрахунків порівняно з Python.

Отже, Ruby — це потужна і гнучка мова програмування, яка підходить для широкого спектру завдань, особливо веб-розробки. Її простота і синтаксична гнучкість роблять її відмінним вибором для початківців і досвідчених розробників, які цінують продуктивність і зручність написання коду.

1.1. Rubi on Rails

Для програмної реалізації було обрано мову програмування Ruby та фреймворк Ruby on Rails (RoR) як один з найпопулярніших веб-фреймворків для розробки веб-додатків. Ось деякі з переваг використання Ruby on Rails:

1. Швидкість розробки: RoR надає розробникам готовий набір інструментів та бібліотек, що дозволяють швидко розробляти веб-додатки. Із застосуванням конвенцій над різноманітністю виборів, розробка програмного забезпечення на Ruby on Rails є швидшою та менш складною.
2. Масштабованість: RoR дозволяє легко масштабувати веб-додатки на великі обсяги, що дозволяє розширювати існуючі системи в майбутньому.

3. Висока продуктивність: збільшена продуктивність забезпечується завдяки великій кількості готових компонентів та зробленого вибору в їхній структурі.

4. Легкість у відлагодженні: RoR має вбудовану систему відлагодження, яка дозволяє легко відстежувати та виправляти помилки в коді.

5. Підтримка баз даних: RoR підтримує різні бази даних, включаючи MySQL, PostgreSQL, Oracle та SQLite.

6. Ruby on Rails має велику та активну спільноту розробників, яка надає багато корисних інструментів та розширень, а також допомагає з вирішенням проблем.

2.2. Визначення функціональних та нефункціональних вимог.

Функціональні та нефункціональні вимоги до інтернет-магазину можуть бути досить різноманітними в залежності від специфіки бізнесу та потреб користувачів. Ось такі вимоги до мого магазину:

Функціональні вимоги:

1. Реєстрація та авторизація користувачів: Система повинна надавати можливість реєстрації нових користувачів та авторизації вже існуючих.

2. Перегляд каталогу товарів: Користувачі повинні мати можливість переглядати список доступних товарів, розділені на категорії та підкатегорії.

3. Пошук товарів: Система повинна надавати можливість швидкого пошуку товарів за назвою, категорією, ціною тощо.

4. Додавання товарів до кошика: Користувачі повинні мати можливість додавати обрані товари до кошика для подальшого оформлення замовлення.

5. Оформлення замовлення: Система повинна надавати можливість користувачам оформлювати замовлення, вказуючи адресу доставки, обираючи метод оплати тощо.

6. Система оплати: Магазин повинен мати систему прийому платежів, яка дозволяє користувачам зручно та безпечно оплачувати свої замовлення.

Нефункціональні вимоги:

1. Продуктивність: Система повинна працювати швидко та ефективно навіть при великому навантаженні.

2. Безпека: Магазин повинен забезпечувати безпеку особистих даних користувачів та фінансових транзакцій.

3. Надійність: Система повинна бути надійною та стабільною, мінімізуючи можливість виникнення помилок та збоїв.

4. Масштабованість: Магазин повинен бути готовий до масштабування, здатний обробляти зростаючий обсяг трафіку та кількість товарів.

5. Адаптивний дизайн: Система повинна мати адаптивний дизайн, який забезпечує коректне відображення на різних типах пристроїв (комп'ютери, планшети, мобільні телефони).

6. Локалізація та інтернаціоналізація: Магазин повинен підтримувати міжнародну локалізацію та інтернаціоналізацію, зокрема мови та валюти.

Це загальний список функціональних та нефункціональних вимог, які можуть бути враховані при розробці інтернет-магазину.

3. ПРОЕКТУВАННЯ СИСТЕМИ:

3.1. Розробка структури бази даних.

Таблиця для Медіа Файлів

string "mediable_type"

integer "mediable_id"

integer "photo_id"

datetime "created_at"

datetime "updated_at"

Таблиця фотографій

string "name"

string "meta_alt"

string "image_file_name"

string "image_content_type"

bigint "image_file_size"

datetime "image_updated_at"

datetime "created_at"

datetime "updated_at"

Таблиця Виробників

string "alias"

datetime "deleted_at"

datetime "created_at"

datetime "updated_at"

Таблиця перекладів для виробників

integer "manufacturer_id"

string "locale"

datetime "created_at"

datetime "updated_at"

string "name"

text "description"

Таблиця категорій

string "meta_description"

string "meta_keywords"

string "alias"

datetime "deleted_at"

integer "parent_id"

datetime "created_at"

datetime "updated_at"

Таблиця перекладів для категорій

integer "category_id"

string "locale"

datetime "created_at"

datetime "updated_at"

string "name"

text "description"

Таблиця товарів

decimal "retail_price", precision: 9, scale: 2, default: "0.0"

string "alias"

boolean "publication"

string "status"

integer "category_id"

datetime "deleted_at"

integer "manufacturer_id"

datetime "created_at"

datetime "updated_at"

string "sku"

boolean "recommended"

string "youtube_id"

integer "rating"

integer "rating_count"

Таблиця перекладів для виробників

integer "product_id"

string "locale"

datetime "created_at"

datetime "updated_at"

string "name"

text "description"

Таблиця налаштувань системи

string "var"

datetime "created_at"

datetime "updated_at"

text "value"

Таблиця користувачів

string "first_name"

string "last_name"

string "role"

datetime "created_at"

datetime "updated_at"

string "email"

string "encrypted_password"

string "reset_password_token"

datetime "reset_password_sent_at"

datetime "remember_created_at"

json "permissions"

Таблиця корзини

t.decimal "total_retail_price"

t.decimal "total_sale_price"

t.decimal "prepaid_amount"

t.integer "user_id"

t.string "aasm_state"

t.datetime "prepaid_at"

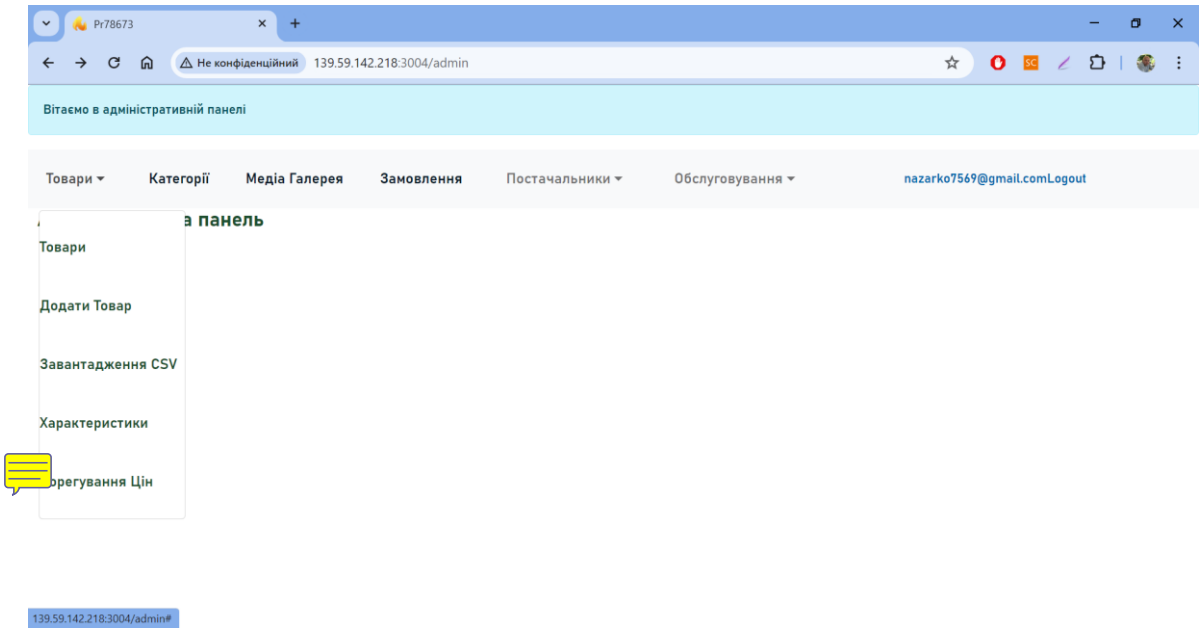
t.datetime "delivered_at"
t.datetime "created_at"
t.datetime "updated_at"
t.string "payment_method"
t.string "shipment_method"
t.string "first_name"
t.string "last_name"
t.string "phone"
t.string "email"
t.text "comments"
t.jsonb "shipment_info", default: "{}", null: false
t.integer "total_with_shipment"
t.datetime "ordered_at"
t.integer "order_number"
t.string "client_ip"

Таблиця товарів у корзинах

t.string "product_name"
t.decimal "product_retail_price"
t.decimal "product_sale_price"
t.integer "product_count"
t.integer "delivery_price"
t.integer "product_id"
t.string "basket_id"
t.datetime "created_at"
t.datetime "updated_at"
t.string "product_sku"

2. Визначення архітектури додатку.

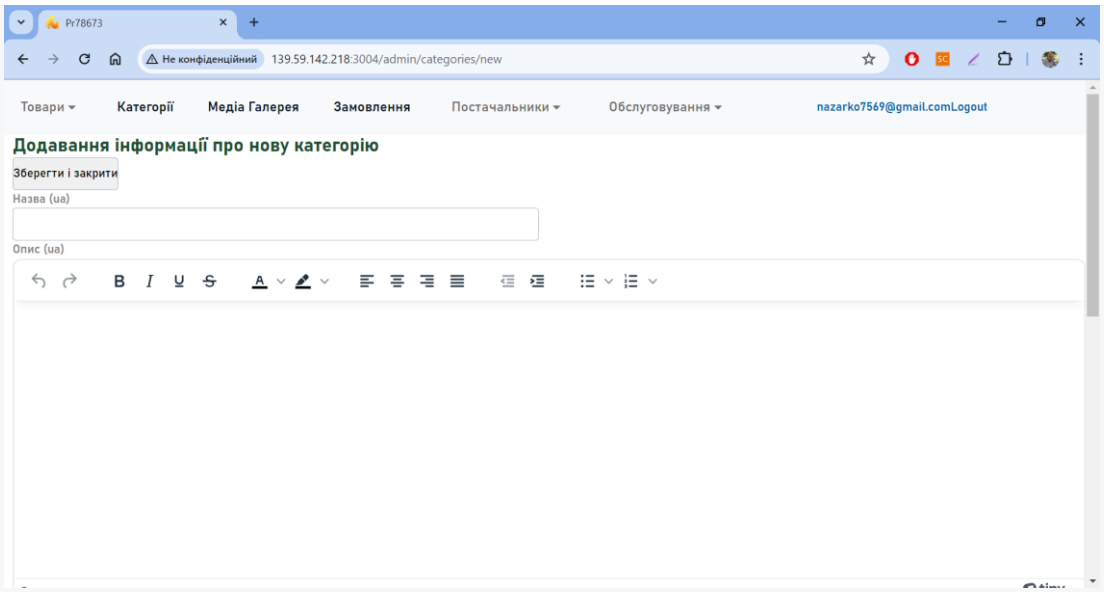
Головне меню адміністративної частини



Програмний код для головного меню адміністративної частини

```
%header
 %nav.navbar.navbar-expand-lg.navbar-light.bg-light
   .container-fluid
     #navbarSupportedContent.collapse.navbar-collapse
       %ul.navbar-nav.me-auto.mb-2.mb-lg-0
         %li.nav-item.dropdown
           %a#navbarDropdown.nav-link.dropdown-toggle{"aria-expanded" => "false",
"data-bs-toggle" => "dropdown", :href => "#", :role => "button"}
             = t(".products")
           %ul.dropdown-menu{"aria-labelledby" => "navbarDropdown"}
             %li= link_to t(".products"), admin_products_path
             %li= link_to t(".new_product"), new_admin_product_path
             %li= link_to t(".bulk_uploads"), admin_bulk_uploads_path
             %li= link_to t(".specifications"), admin_specifications_path
             %li= link_to t(".price_corection"), admin_price_services_path
         %li.nav-item= link_to t(".categories"), admin_categories_path
         %li.nav-item= link_to t(".photos"), admin_photos_path
       %ul.nav.navbar-nav.pull-right
         %li.nav
           = link_to current_user.email, admin_user_path(current_user)
         %li.nav
           = link_to('Logout', destroy_user_session_path, method: :delete)
```

Форма створення та редагування категорії



Програмний код для форми створення або редагування категорії

```
f @category.errors.any?
  #error_explanation
    %h2
      Некоректне заповнення форми: #{pluralize(@category.errors.count, "error")}
    %ul
      - @category.errors.full_messages.each do |msg|
        %li= msg
= form_for [:admin, @category] do |f|
  = render "admin/shared/actions", f: f

  = render "admin/forms/fields/text_field", form: f, field_name: :name, class_style: "text_field"
  = render "admin/forms/fields/text_area", form: f, field_name: :description, class_style: "description"
  .row
    .col-lg-6
      .row
        .col-lg-6
          .field
            = f.label t(:alias)
            %br/
            = f.text_field :alias, class: "text_field"
        .col-lg-6
          .field
            = f.label t(:meta_keywords)
            %br/
            = f.text_field :meta_keywords, class: "text_field"
      .row
        .col-lg-6
```

```
                .field
                  = f.label t(:meta_description)
                  %br/
                  = f.text_field :meta_description, class: "text_field"
              .col-lg-6
                .field
                  = f.label t(:parent_category)
                  =                                                    f.select(:parent_id,
options_from_collection_for_select(@top_categories, :id, :name, @category.parent_id), { include_blank: true }, { class: "select form-select  h-100 w-100", data: { placeholder: t(:select_parent_category) } })
          .col-lg-6
            .field
              = f.label t(:specifications)
              %br/
              =                                                f.select(:specification_ids,
options_from_collection_for_select(@specifications,              :id,              :name,
@category.specification_ids), {  }, { class: "multiple-select form-select  h-100 w-100",
multiple: true, data: { placeholder: t(:select_specifications) } })
        = render "admin/shared/image_picker", owner: @category
```

3.3. Створення схеми маршрутів.

Створення схеми маршрутів відбувається у фреймворку Ruby on Rails за допомогою файлу `routes.rb`. Цей файл містить визначення маршрутів для різних URL-адрес, які спрямовують запити від користувачів до відповідних контролерів та дій.

Ось опис створення схеми маршрутів у файлі `routes.rb`:

```ruby
Rails.application.routes.draw do
  # Маршрути для товарів
  resources :products

  # Маршрути для категорій
  resources :categories

  # Маршрути для користувачів
```

```
  devise_for :users

  # Додаткові маршрути
  get 'about', to: 'pages#about'
  get 'contact', to: 'pages#contact'

  # Маршрути для кошика
  resources :carts do
    resources :cart_items
  end
  # Інші маршрути можуть бути додані тут
end
```

У цьому прикладі:

- `resources :products` створює стандартні RESTful маршрути для ресурсу "products", які включають маршрути для індексування, створення, читання, оновлення та видалення товарів.

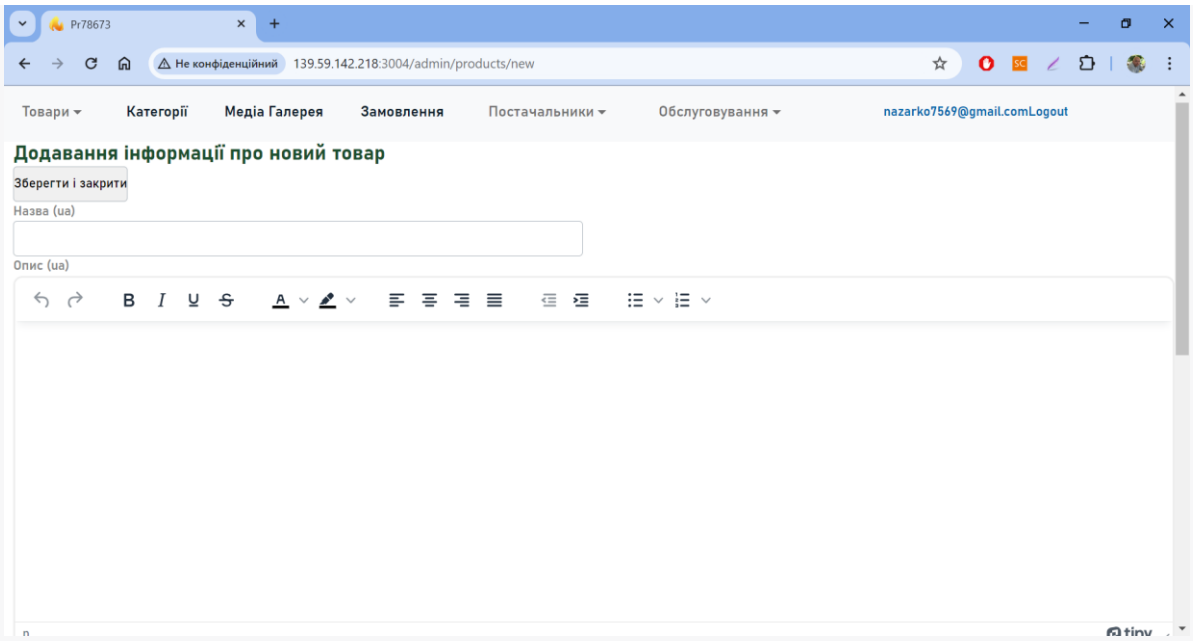
- `resources :categories` аналогічно створює маршрути для ресурсу "categories".

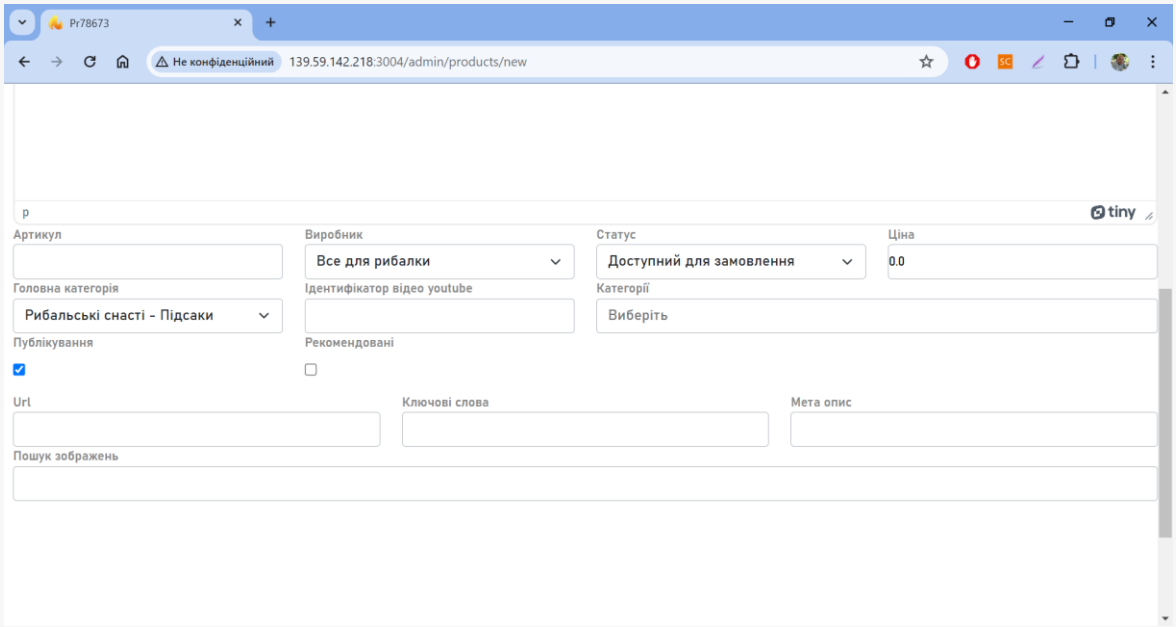
- `devise_for :users` використовується для генерації стандартних маршрутів для аутентифікації користувачів за допомогою Devise (розширення для авторизації та аутентифікації в Ruby on Rails).

- `get 'about', to: 'pages#about'` та `get 'contact', to: 'pages#contact'` визначають маршрути для статичних сторінок "about" та "contact".

- `resources :carts do ... end` визначає вкладені маршрути для кошика та елементів кошика.

Форма створення та редагування товару





Програмний код для форми створення або редагування товару

```
= render "admin/shared/errors", errors: @product.errors
= form_for [:admin, @product] do |f|
 = render "admin/shared/actions", f: f

 = render "admin/forms/fields/text_field", form: f, field_name: :name, class_style: "text_field"
 = render "admin/forms/fields/text_area", form: f, field_name: :description, class_style: "description"
 .row
   .col-lg-3
     .field
```

```
            = f.label t(:sku)
            %br/
            = f.text_field :sku, class: "text_field"
          .col-lg-3
            .field
              = f.label t(:manufacturer)
              %br/
              =                                                    f.select(:manufacturer_id,
options_from_collection_for_select(@manufacturers, :id, :name, @product.manufacturer_id),
{  }, { class: "select form-select  h-100 w-100" })
          .col-lg-3
            .field
              = f.label t(:status)
              %br/
              = f.select(:status, AppConfig.product_statuses.map { |k,v|  [t(k), k]}, {  },
{ class: "select form-select  h-100 w-100" })
          .col-lg-3
            .field
              = f.label t(:retail_price)
              %br/
              = f.text_field :retail_price, class: "text_field"
        .row
          .col-lg-6
            .row
              .col-lg-6
                .field
                  = f.label t(:main_category)
                  %br/
                  = f.select(:category_id, options_for_select(@categories.map{ |category|
[     "#{category.parent.try(:name)}     -     #{category.name}",     category.id]}.to_h,
@product.category_id), {  }, { class: "select form-select" })
              .col-lg-6
                .field
                  = f.label t(".youtube_id")
                  %br/
                  = f.text_field :youtube_id, class: "text_field"
            .row
              .col-lg-6
                .field
                  = f.label t(:publication)
                  %br/
                  = f.check_box :publication
              .col-lg-6
                .field
                  = f.label t(:recommended)
                  %br/
                  = f.check_box :recommended
          .col-lg-6
            .row
```

```
              .col-md-12
                .field
                  = f.label t(:categories)
                  %br/
                  = f.select(:category_ids, options_for_select(@categories.map{ |category|
[     "#{category.parent.try(:name)}     -     #{category.name}",     category.id]}.to_h,
@product.category_ids.uniq), {  }, { class: "multiple-select form-select", multiple: true
})
        - if current_user.permissions["meta_tags"]
          .row
            .col-lg-4
              .field
                = f.label t(:alias)
                %br/
                = f.text_field :alias, class: "text_field"
            .col-lg-4
              .field
                = f.label t(:meta_keywords)
                %br/
                = f.text_field :meta_keywords, class: "text_field"
            .col-lg-4
              .field
                = f.label t(:meta_description)
                %br/
                = f.text_field :meta_description, class: "text_field"
        = render "admin/shared/image_picker", owner: @product
        - if action == :edit
          =       render       "admin/forms/specifications",       specification_fields:
product_specification_fields(@product), specifications: @product.specifications, form: f
        - if action == :edit && @product.parent_id == nil
          .field
            = f.label t(:children_ids)
            %br/
            = render "admin/products/product_search", form: f
```

4. РЕАЛІЗАЦІЯ:

4.1. Створення моделей, контролерів та виглядів.

Структура моделі кошика включає такі поля:

1. id: Унікальний ідентифікатор кошика.

2. user_id: Зовнішній ключ, який посилається на користувача, який створив кошик.

3. total_retail_price: Загальна вартість всіх товарів у кошику за роздрібною ціною.

4. total_sale_price: Загальна вартість всіх товарів у кошику за зниженою ціною (якщо є).

5. prepaid_amount: Сума передоплати (якщо є).

6. aasm_state: Стан кошика, який може вказувати на поточний стан замовлення (наприклад, "новий", "в обробці", "відправлений" і т. д.).

7. prepaid_at: Дата та час, коли була зроблена передоплата.

8. delivered_at: Дата та час, коли було доставлено замовлення.

9. payment_method: Метод оплати для замовлення (наприклад, "кредитна карта", "готівка", "онлайн-переказ" і т. д.).

10. shipment_method: Метод доставки для замовлення (наприклад, "кур'єрська доставка", "самовивіз", "поштова доставка" і т. д.).

11. first_name: Ім'я клієнта, якому належить кошик.

12. last_name: Прізвище клієнта, якому належить кошик.

13. phone: Контактний телефон клієнта.

14. email: Електронна пошта клієнта.

15. comments: Додаткові коментарі чи вказівки щодо замовлення.

Зв'язки з іншими моделями такі:

- Кожен кошик належить конкретному користувачеві (багато до одного зв'язок).

- Кожен товар у кошику може посилатися на продукт (багато до одного зв'язок), тому може бути таблиця "CartItems", яка містить id кошика, id товару,

кількість товару та будь-які додаткові атрибути, які стосуються самого товару у кошику.

Ця структура моделі дозволить зберігати та керувати даними кошика покупок у моєму інтернет-магазині.

4.2. Налаштування аутентифікації та авторизації.

Процедура авторизації та аутентифікації використовується на багатьох різних веб сайтах. Це типова задача для вирішення якої є готові програмні рішення. Одним з таких рішень є Devise - це гем для аутентифікації користувачів в Ruby on Rails додатках. Ось декілька переваг використання Devise:

1. Швидкість розробки: Devise забезпечує швидку розробку автентифікації користувачів. Гем містить готові контролери, вигляди та маршрути для аутентифікації користувачів, що дозволяє розробникам зосередитися на інших аспектах додатка.

2. Гнучкість: Devise дозволяє налаштувати різні аспекти автентифікації, такі як тип аутентифікації (наприклад, з використанням електронної пошти або імені користувача), зберігання сесій, заборона зареєстрованих користувачів на певний час і т.д.

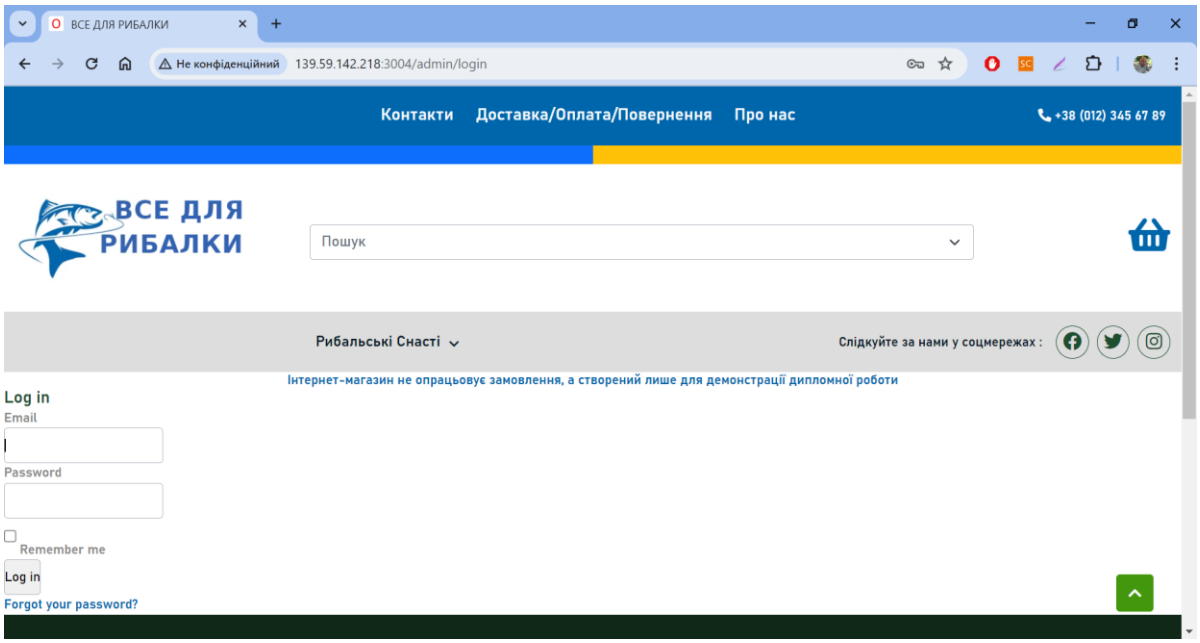
3. Розширення: Devise має велику кількість розширень (плагінів), які дозволяють налаштувати автентифікацію за власними потребами. Наприклад, можна використовувати додаткові методи аутентифікації, створювати додаткові поля для профілю користувача і т.д.

4. Безпека: Devise містить вбудовану захист від типових атак, таких як перехоплення сесій, злам паролів і т.д.

5. Підтримка: Devise має велику спільноту користувачів та розробників, яка надає допомогу та поради щодо розробки та налаштування автентифікації користувачів.

У загальному, Devise дозволяє зосередитися на основних функціях додатка, використовуючи готові рішення для аутентифікації користувачів, що полегшує та прискорює розробку.

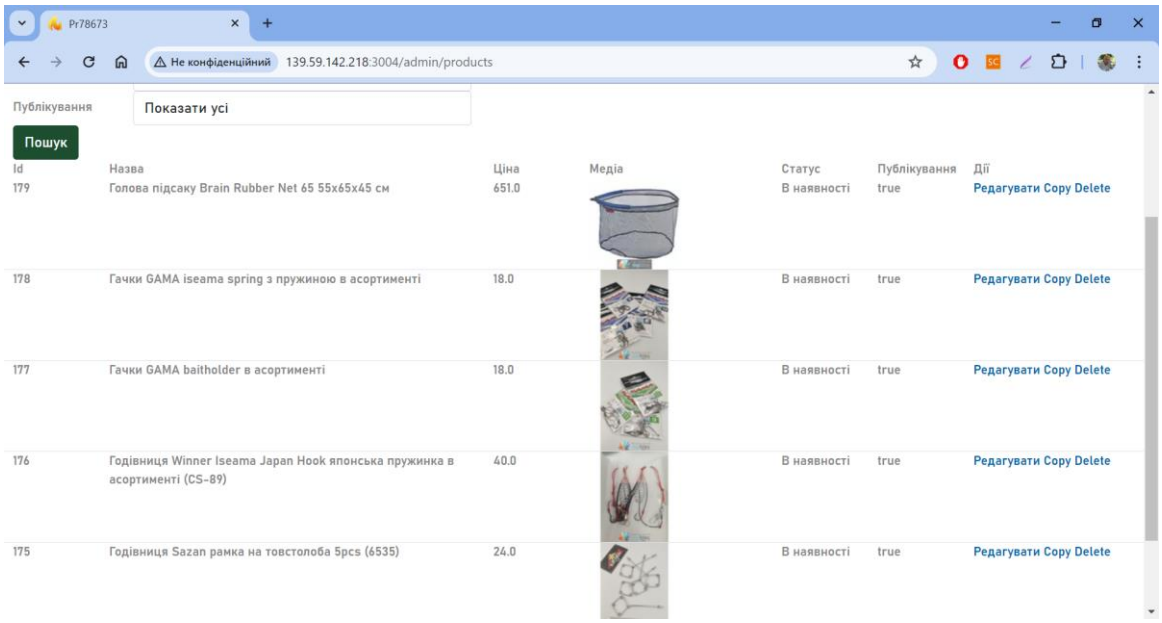
Форма входу в адміністративну частину



4.3. Реалізація кошика покупок та системи оплати.

Розробка каталогів товарів

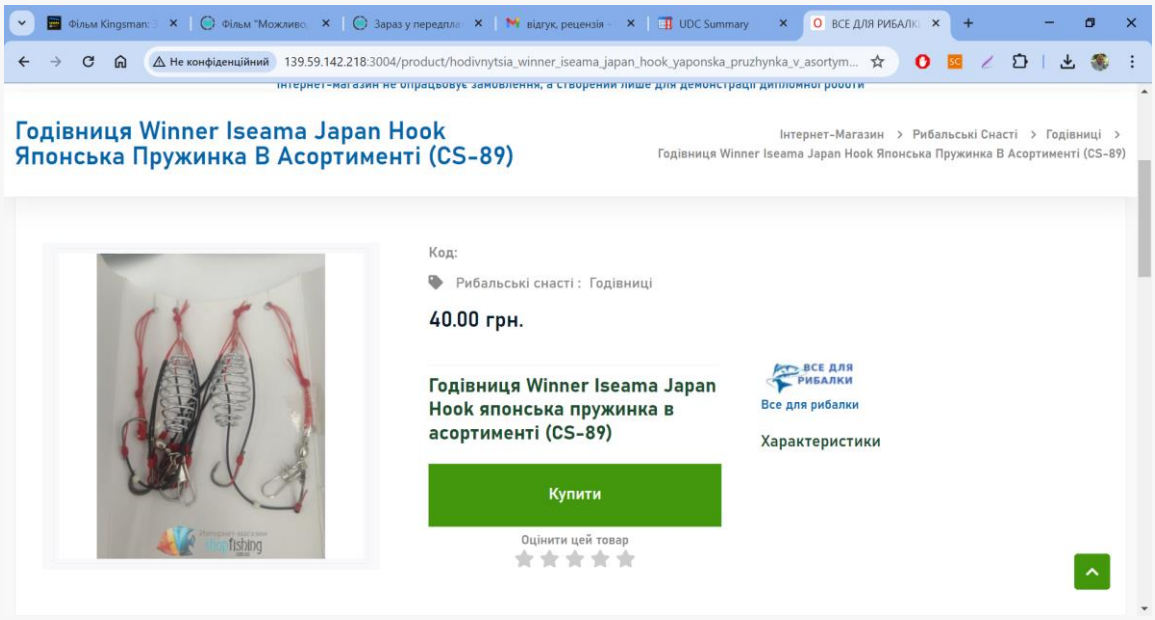
Вигляд каталогу товарів в браузері



Програмний код каталогу товарів

```
- categories = @category.parent_id == nil ? [] : [@categories.find{|category| category.id == @category.parent_id}]
%section
  .container-fluid
    = render "layouts/components/breadcrumbs", title: @category.name, categories: categories
    .row
      - @products.each do |product|
        .col-xxl-2.col-xl-3.col-lg-4.col-6
          = render "layouts/components/product/product_card", product: product, show_rate: true, specifications: @specifications
    .row.text-center.pagination.d-xs-none.d-sm-none.d-md-block
      = paginate @products, theme: 'bootstrap-5', pagination_class: "pagination-sm flex-wrap justify-content-center", nav_class: "d-inline-block"
```

Вигляд картки товару в браузері



Програмний код картки товару

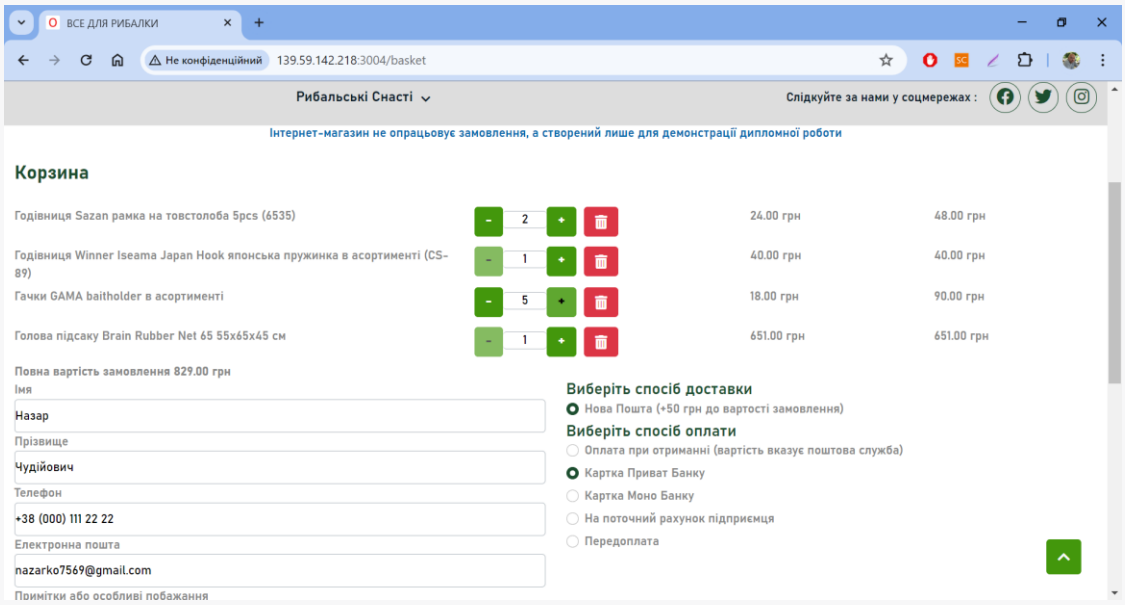
```
- child_category = @categories.find{ |category| category.id == @product.category_id}
- parent_category = @categories.find{ |category| category.id == child_category.parent_id}
```

```
      =    render    "layouts/components/breadcrumbs",    title:    @product.name,
categories: [parent_category, child_category]
      %section.item-details.section
       .container-fluid
         .top-area
           .row.align-items-center
             .col-lg-4.col-md-12.col-12
               =  render  "layouts/components/product/product_images",  product:
@product
             .col-lg-4.col-md-12.col-12
               .product-info
                 -# %h1.title
                 -#   = @product.name
                 - unless @product.youtube_id.to_s.empty?
                   .youtube-container
                     %iframe{:allowfullscreen => "", :frameborder => "0", :src =>
"https://www.youtube.com/embed/#{@product.youtube_id}?autoplay=1&mute=1", allow:
'autoplay'}
                 %p
                   = t("product_sku")
                   = @product.sku
                 %p.category
                   = icon('fas','tag')
                   =                link_to                parent_category.name,
show_category_path(parent_category)
                   = ":"
                   =                link_to                child_category.name,
show_category_path(child_category)
                 %h3.price
                   = number_to_uah(@product.retail_price)
                   -# %span $945
                 %p.info-text
                   =raw @product.description
                 - if @children_products.count > 0
                   =  render  "products/children_products",  children_products:
@children_products, specifications: @children_products_specifications, product:
@product, parent_id: @children_products[0].parent_id
                 - if @children_products.count == 0 || @product.parent_id
                   .bottom-content
                     .row.align-items-end
                       .col-lg-12.col-md-12.col-12
                         .button.cart-button
                           =   render   "shared/add_to_basket_button",   product:
@product
```

```
                = render "shared/rating_widget/rate", product_id: @product.id
            .col-lg-4.col-md-12.col-12
              .manufacturer
                = link_to show_manufacturer_path(@product.manufacturer) do
                  - if @product.manufacturer.media_files.any?
                    .text-center
                      =
image_tag(@product.manufacturer.media_files[0].photo.image.url(:thumb))
                  .text-center
                    = @product.manufacturer.name
              - unless @product.has_children?
                .product-specifications
                  %h2
                    = t "product.tabs.specifications"
                  -       unless       @product.specifications.empty?       ||
@product.specifications == "{}"
                    -   @product.specifications.each   do   |specification_id,
specification_value|
                      -   if   specification_id.split("_").length   ==   1   or
specification_id.split("_").last == I18n.locale.to_s
                        -                   specification                   =
Specification.find(specification_id.to_i)
                        .specification
                          - if specification.specification_type == "color"
                            .row
                              .col-auto
                                = "#{specification.name}:"
                              .col-auto.color{style:         "background-color:
#{specification_value}"}
                          - else
                            =   "#{specification.name}:   #{specification_value}
#{specification.unit}"
        =   render   "layouts/components/product/related_products",   products:
@related_products
```

Розробка функціоналу “Корзина”

Вигляд корзини в браузері



Програмний код корзини

```
function BasketHandler(options) {
 var params = {
   payment_method: $('.payment_method:checked').val(),
   shipment_method: $('.shipment_method:checked').val(),
 }
 class BasketEntity {
   constructor(params) {
     this.name = params.name;
     this.count = Number(params.count) || 1;
     this.price = Number(params.price);
     this.productId = Number(params.productId)
     this.productSku = params.productSku
   }
   subTotal() {
     return Number((this.count * this.price).toFixed(2))
   }
   increment() {
     this.count ++
   }
   decrement() {
     if(this.count > 1) {
       this.count --
     }
```

```
  }
}
class Basket {
  constructor(params) {
    this.minCost = options.minCost || 250;
    this.payment_method = "post_service";
    this.shipment_method = "nova_poshta";
    this.basketEntities = [];
    this.shipmentInfo = {};
    let basketJSON = localStorage.getItem("basket");
    if (basketJSON) {
      var basketParams = JSON.parse(basketJSON);
      this.updateBasket(basketParams);
      this.createBasketEntities(basketParams.basketEntities)
    } else {

    }
  }
  findBasketEntityIndex(productId) {
    var basketEntityIndex
    this.basketEntities.findIndex(function(basketEntity, index) {
      if(basketEntity.productId == productId)
        basketEntityIndex = index
    });
    return basketEntityIndex;
  }
  addBasketEntity(params) {
    let basketEntityIndex = this.findBasketEntityIndex(params.productId)
    if(basketEntityIndex > -1) {
      this.basketEntities[basketEntityIndex].increment()
    } else {
      this.basketEntities.push(new BasketEntity(params))
    }
    this.save()
  }
  changeBasketEntityCount(productId, count) {
    let basketEntityIndex = this.findBasketEntityIndex(productId)
    if(basketEntityIndex > -1) {
      this.basketEntities[basketEntityIndex].count = count;
    }
    this.save();
  }
  changeBasketEntityPrice(productId, price) {
    let basketEntityIndex = this.findBasketEntityIndex(productId)
```

```
        this.basketEntities[basketEntityIndex].price = parseFloat(price);
        this.save();
      }
      deleteBasketEntity(productId) {
        let basketEntityIndex = this.findBasketEntityIndex(productId)
        if(basketEntityIndex > -1) {
          this.basketEntities.splice(basketEntityIndex, 1)
        }
        this.save();
      }
      save() {
        localStorage.setItem("basket", JSON.stringify(this));
      }
      delete() {
        localStorage.removeItem("basket");
      }
      create_order() {
        this.basketEntities = [];
        this.save();
      }
      updateBasket(basketParams) {
        this.payment_method = basketParams.payment_method;
        this.shipment_method = basketParams.shipment_method;
        this.first_name = basketParams.first_name || "";
        this.last_name = basketParams.last_name || "";
        this.phone = basketParams.phone || "";
        this.email = basketParams.email || "";
        this.comments = basketParams.comments || "";
        if (basketParams.city_ref) this.city_ref = basketParams.city_ref;
        if (basketParams.city_desc) this.city_desc = basketParams.city_desc;
        if (basketParams.city_desc_ru) this.city_desc_ru =
basketParams.city_desc_ru;
        if (basketParams.warehouse_ref) this.warehouse_ref =
basketParams.warehouse_ref;
      }
      createBasketEntities(basketEntities) {
        let result = [];
        basketEntities.forEach(function(basketEntity) {
          result.push(new BasketEntity(basketEntity))
        })
        this.basketEntities = result;
      }
      total() {
        let total = 0;
```

```
      this.basketEntities.forEach(function(basketEntity) {
        total += basketEntity.subTotal()
      })
      return Number(total.toFixed(2));
    }
  }
   basket = new Basket({ ...options, ...params });
  $(".add-to-basket-button").click(function() {
    let basketEntityParams = {
      name: this.dataset.productName,
      price: this.dataset.productPrice,
      productId: this.dataset.productId,
      productSku: this.dataset.productSku
    }
    basket.addBasketEntity(basketEntityParams);
  });
};
```

4.4. Впровадження функціоналу пошуку, фільтрації та сортування товарів.

У контексті інтернет-магазину, процедура пошуку лотів означає механізм пошуку товарів або "лотів", які доступні для покупки в магазині. Це може бути ключовою функцією для користувачів, які шукають конкретний товар або певну категорію товарів.

Ось основні кроки, які можуть бути включені до процедури пошуку лотів в інтернет-магазині:

1. Введення пошукового запиту: Користувач вводить ключове слово або фразу у поле пошуку на веб-сайті магазину.

2. Обробка пошукового запиту: Сервер обробляє пошуковий запит, аналізує його та визначає, які товари відповідають критеріям пошуку.

3. Відображення результатів: Знайдені товари відображаються на сторінці результатів пошуку. Кожен товар може бути представлений у вигляді зображення, назви, короткого опису та ціни.

4. Фільтрація та сортування: Користувач може використовувати фільтри або параметри сортування, щоб обмежити результати пошуку за певними критеріями, такими як ціна, категорія, рейтинг тощо.

5. Перегляд деталей товару: Користувач може клацнути на конкретний товар у списку результатів пошуку, щоб переглянути більше інформації про нього, таку як повний опис, характеристики, відгуки тощо.

6. Покупка товару: Після того, як користувач обрав певний товар, він може додати його до кошика та оформити покупку.

Процедура пошуку лотів допомагає користувачам швидко знайти потрібні їм товари у магазині, що покращує їхній досвід покупок та сприяє збільшенню продажів.

4.5. Впровадження інформування клієнтів через пошту

AMP – це технологія від Google, яка здатна збільшити ваші продажі.

Перш ніж розглянути використання AMP в ecommerce, зануримося в особливості AMP.

Accelerated Mobile Pages (AMP) — це структура з відкритим вихідним кодом, яка спочатку була розроблена для оптимізації веб-сайтів і електронних листів. Через деякий час технологія дала маркетологам можливість значно покращити користувацький досвід, додавши до листів інтерактивні елементи (каруселі, кнопки CTA, різні форми, опитування, вікторини тощо).

Функціональність AMP для електронних листів не така широка, як для веб-сайтів, але цього достатньо, щоб дивувати користувачів. Згідно з дослідженнями, 60% одержувачів емейлів, ймовірно, будуть взаємодіяти з інтерактивними листами. Для підтвердження статистики наведемо результати A/B-тестів. Отримувачів, які мали заповнити анкету, було 23 тисячі. Одна частина користувачів отримала інтерактивну форму, а інша – звичайну. Маркетологи були здивовані: взаємодія з AMP-формою була в 5,2 разу більше.

Ви повинні звернути увагу на інструменти AMP, якщо:

- AMP-емейли надають вам додаткову цінність і переваги. Використання AMP виправдане, якщо ви хочете представити різноманітні продукти, швидше і

простіше спілкуватися з аудиторією або здивувати своїх клієнтів. В інших випадках краще розглянути традиційні листи, розробка яких займає менше часу, оскільки вам не потрібно готувати дві версії листа.

- Ваші підписники використовують відповідні поштові клієнти. На жаль, не всі поштовики підтримують AMP, тому перед його використанням варто провести невелике дослідження своєї аудиторії. Gmail найкращий (він правильно відображає всі наявні теги AMP)[14].

Використовуйте зручний конструктор drag-n-drop, щоб створити HTML-версію свого листа, а потім додайте елементи AMP. Якщо деякі з поштових програм ваших клієнтів не підтримують AMP, буде показано звичайний електронний лист без інтерактивних елементів. Крім того, є користувачі з поштовиками, які не підтримують навіть HTML. Такі клієнти отримають звичайний текст – так ваше повідомлення все одно буде читабельним.

Динамічні форми

Як часто ви просите клієнтів перейти за посиланням, аби заповнити форму на вашому сайті? За допомогою AMP ви можете вставляти форми в електронні листи. Таке рішення надзвичайно зручне, якщо, наприклад, вам потрібно уточнити дані доставки або змусити покупців оцінити ваші послуги/продукти та залишити відгук.

Акордеон

Ця функція приховує певний вміст у ваших емейлах. Така можливість стане в нагоді, коли потрібно представити всі наявні пропозиції, не перевантажуючи лист. Користувачі повинні просто натиснути на заголовок, щоб побачити прихований текст, зображення, відео та навіть каруселі.

Кнопки CTA

Маркетологи часто хочуть, щоб клієнт підписався на розсилку або підтвердив певні дії (наприклад, зміна кредитної картки в обліковому записі). Кнопки AMP дозволяють робити це без переходу на будь-який вебсайт.

Ігри

Гейміфікація листів – це унікальний метод залучення користувачів і покращення утримання клієнтів. Єдиний недолік – це складність розробки таких емейлів.

Отже, які ігри можна створювати та для чого? Насправді єдине обмеження тут – це ваша уява. Ігри та вікторини в емейлах зазвичай розробляють для розваги. Коли користувачі проходять їх, вони отримують купони, знижки тощо. Цей підхід набагато ефективніший, ніж звичайні електронні листи з промокодами всередині.

AMP є чудовим способом виділитися серед тисяч інших брендів. Правильне використання згаданих інструментів підвищить ефективність ваших email-кампаній, що є ключем до збільшення доходу [14].

5. ОПТИМІЗАЦІЯ ТА ВИПРАВЛЕННЯ ПОМИЛОК:

5.1. Виявлення та виправлення помилок.

Коли ми говоримо про виявлення та виправлення помилок, важливо виконати декілька ключових завдань:

1. Моніторинг та виявлення помилок: Важливо встановити механізми моніторингу, які дозволяють відстежувати виникнення помилок у веб-додатку. Це може включати моніторинг ведення журналів, відстеження помилок JavaScript у браузері користувача, а також моніторинг процесів на сервері.

2. Аналіз та класифікація помилок: Після виявлення помилок важливо аналізувати їх та класифікувати за пріоритетом та серйозністю. Деякі помилки можуть бути критичними і потребувати негайної уваги, тоді як інші можуть бути менш важливими.

3. Виправлення помилок: Після аналізу необхідно виправити виявлені помилки. Це може включати внесення змін у код програми, виправлення конфігураційних параметрів, оновлення бібліотек та залежностей, а також виправлення помилок у базі даних.

4. Тестування виправлень: Після внесення змін важливо перевірити, що виправлення помилок працюють наочно. Це може включати проведення ручних або автоматизованих тестів, перевірку функціональності та стабільності системи.

5. Моніторинг після виправлення: Після внесення змін важливо продовжувати моніторити систему, щоб переконатися, що виправлення помилок не призвели до появи нових проблем або погіршення продуктивності.

В цілому, виявлення та виправлення помилок є важливим етапом у розробці будь-якого веб-додатку, включаючи інтернет-магазини на базі Ruby on Rails. Цей процес допомагає забезпечити стабільну та надійну роботу додатку, зменшити кількість негативних впливів на користувачів та збільшити їхнє задоволення від використання магазину.

5.2. Оптимізація швидкодії та ефективності додатку.

Оптимізація у контексті розробки програмного забезпечення, зокрема веб-додатків, полягає у покращенні ефективності, швидкодії, економії ресурсів та підвищенні продуктивності додатку. Оптимізація має на меті зробити програму більш продуктивною та ефективною, зменшуючи час відповіді, ресурсоємність та використання пам'яті.

Ось деякі аспекти оптимізації у контексті веб-додатків на Ruby on Rails:

1. Швидкодія відповідей сервера: Оптимізація коду контролерів та виглядів для зменшення часу відповіді сервера на запити.
2. База даних: Оптимізація запитів SQL, використання індексів, кешування результатів запитів для покращення швидкодії.
3. Кешування: Використання кешування для збереження результатів часто використовуваних запитів або фрагментів сторінок для швидкого доступу до них у майбутньому.
4. Оптимізація завантаження ресурсів: Мінімізація розміру та кількості запитів для завантаження CSS, JavaScript, зображень тощо.
5. Асинхронність: Використання асинхронних запитів та обробників для паралельного виконання завдань та зменшення часу очікування користувача.[6].
6. Масштабованість: Розробка з урахуванням можливості масштабування додатку, щоб він ефективно працював при збільшенні навантаження.
7. Безпека: Застосування належних заходів безпеки для запобігання атакам та збільшення стійкості додатку.

В цілому, оптимізація має на меті покращити якість та продуктивність веб-додатку, забезпечити більш комфортний та ефективний досвід користувача, а також зменшити навантаження на сервер та ресурси.

Кешування - це техніка збереження результатів попередньо обчислених або отриманих запитів у пам'яті або на диску з метою подальшого швидкого

доступу до них при наступних запитах. У контексті веб-додатків на Ruby on Rails кешування може бути корисним для покращення швидкодії та зменшення навантаження на сервер у випадках, коли певні дані або фрагменти сторінок не змінюються часто.

Ось кілька сценаріїв застосування кешування, які я використав в своєму інтернет-магазині

1. Кешування сторінок: Певні сторінки, які не змінюються часто (наприклад, домашня сторінка або сторінки категорій товарів), можуть бути кешовані для швидкого доступу. При запиті на ці сторінки сервер спочатку перевіряє, чи є кешована версія, і використовує її, якщо так.

2. Кешування фрагментів сторінок: Частини сторінок, які включають дорогоцінні або часто виконувані операції (наприклад, список товарів або банери), можуть бути кешовані окремо. При побудові сторінки ці фрагменти вставляються з кешу.

3. Кешування запитів до бази даних: Результати запитів до бази даних, які виконуються часто та не змінюються часто, можуть бути кешовані. Наприклад, список товарів у певній категорії може бути кешований для покращення швидкодії.

4. Кешування API викликів: Якщо ваш інтернет-магазин співпрацює з іншими сервісами через API, результати цих викликів також можуть бути кешовані для зменшення затримок та навантаження на зовнішні системи.

Застосування кешування допомагає зменшити час відповіді сервера та покращити загальну швидкодію додатку, зокрема в інтернет-магазині на Ruby on Rails. Однак потрібно ретельно розглядати, які дані кешувати та як часто оновлювати кеш, щоб уникнути відображення застарілих або неточних даних користувачам.

6.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній з них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 5.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0,2+ 0,4 – 0,2 \cdot 0,4 = 0,0592$.

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0{,}2 + 0{,}15 - 0{,}2 \cdot 0{,}15 = 0{,}264.$$

$$P_{14} = P_{11} \cdot P_5 = 0{,}118 \cdot 0{,}2 = 0{,}0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0{,}252 \cdot 0{,}1 = 0{,}0252.$$

$$P_{17} = P_{13} + P_{14} - P_{13} \cdot P_{14} = 0{,}592 + 0{,}0236 - 0{,}0592 \cdot 0{,}0236 = 0{,}0814.$$

$$P_{18} = P_{15} \cdot P_{16} = 0{,}264 \cdot 0{,}0252 = 0{,}0065.$$

$$P_{19} = P_{17} + P_{18} - P_{17} \cdot P_{18} = 0{,}0065 + 0{,}0814 - 0{,}0065 \cdot 0{,}0814 = 0{,}0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0{,}0873$.

6.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;

- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;

- покращання естетичних показників, раціональне компонування робочих місць і впорядкування робочих приміщень;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;

- зниження рівня виробничого травматизму;

- зменшення кількості випадків професійних захворювань;

- зменшення плинності кадрів через незадовільні умови праці;

- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

6.3. Безпека в надзвичайних ситуаціях

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами. Ризик надзвичайних ситуацій природного та техногенного характеру невпинно зростає, тому питання захисту цивільного населення від надзвичайних ситуацій на сьогодні є дуже важливе.

У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином:

Укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення у позаміській зоні.

Висновки:

Робота була спрямована на створення інтернет-магазину рибальського спорядження, і ця мета була досягнута. Була розроблена функціональна система, яка дозволяє користувачам переглядати товари, додавати їх до кошика, оформляти замовлення та здійснювати оплату.

Були успішно реалізовані наступні функції у вигляді функціоналу магазину, такі як

5.3. Додавання категорій,
5.4. додавання товарів,
5.5. оформлення замовлень,
5.6. можливості оплати, в т.ч. через Liqpay.
5.7. доставка через Нову пошту

Переваги та недоліки:

5.8. переваги розробленого магазину, такі як
5.9. простота використання,
5.10. зручний інтерфейс користувача,
5.11. а також недоліки, такі як
5.12. можливість подальшого вдосконалення шляхом додавання нового функціоналу або виправлення помилок.

Подальший розвиток магазину. Це може включати в себе додавання нового функціоналу (наприклад, реалізацію системи знижок, відгуків користувачів), а також вдосконалення існуючого функціоналу (наприклад, оптимізація швидкості роботи сайту, поліпшення інтерфейсу користувача тощо).

Використані джерела

1. Ruby on Rails Guides. Available from : https://guides.rubyonrails.org

2. Help and documentation for the Ruby programming language. Available from : https://ruby-doc.org/

3. GitHub - heartcombo/devise: Flexible authentication solution for Rails with Warden. Available from : https://github.com/heartcombo/devise

4. GitHub - ruby-i18n/i18n: Internationalization (i18n) library for Ruby. Available from : https://github.com/ruby-i18n/i18n

5. GitHub - globalize/globalize: Rails I18n de-facto standard library for ActiveRecord model/data translation. Available from : https://github.com/globalize/globalize

6. GitHub - puma/puma: A Ruby/Rack web server built for parallelism. Available from : https://github.com/puma/puma

7. GitHub - aasm/aasm: AASM - State machines for Ruby classes (plain Ruby, ActiveRecord, Mongoid, NoBrainer, Dynamoid. Available from : https://github.com/aasm/aasm

8. GitHub - huacnlee/rails-settings-cached: Global settings for your Rails application. Available from : https://github.com/huacnlee/rails-settings-cached

9. Портал для розробників Нова Пошта. – Режим доступу: https://developers.novaposhta.ua

10. Портал для розробників УкрПошта https://dev.ukrposhta.ua

11. Усі платіжні та інформаційні APIs LiqPay. – Режим доступу: https://www.liqpay.ua/documentation/api/home

12. Sandi Metz Practical Object-Oriented Design in Ruby - Addison-Wesley Professional, 2013. - 247 pages

13. Avdi Grimm Confident Ruby: 32 Patterns for Joyful Coding - ShipRise Media, 2013. - 280 pages

14. Технологія AMP в email: час дивувати своїх клієнтів https://esputnik.com/uk/blog/tehnologiya-amp-v-email-chas-divuvati-svoyih-kliyentiv

Додатки:

Код програмного забезпечення.

Розробка системи інформування через email

```
class BasketMailer < ActionMailer::Base
   # include ApplicationHelper
   # include include ActionView::Helpers::NumberHelper

   default from: "mailer@pirolev.com.ua"
   helper :application

   def order_email_to_manager(basket, to)
       @basket = basket
       attachments["order_#{@basket.order_number}.pdf"]                    =
WickedPdf.new.pdf_from_string(
         render_to_string(pdf: 'todo', template: 'baskets/order_pdf.html.haml',
layout: 'pdf.html'), { encoding: 'utf8' }
       )
       mail(:to => to, :subject => subject(basket)) # TODO create names for
order based on date and order number
   end

   def order_email_to_customer(basket)
       @basket = basket
       mail(:to => basket.email, :subject => subject(basket)) # TODO create
names for order based on date and order number
   end

   def subject(basket)
       "#{ t("order") } #{ @basket.order_number }, #{ @basket.first_name }
#{                  @basket.last_name                    },                    #{
view_context.number_to_uah(@basket.total_with_shipment) }"
   end
end



!!!
%html
 %head
   %meta{:content => "text/html; charset=UTF-8", "http-equiv" => "Content-
Type"}/
```

```
      %body
        -#    =    link_to    (image_tag    "https://leor.com.ua/assets/leor_logo-
50add2020c7866d2d956e176e586800c2a6c9d23ac8f67f963ee449c6c583310.webp"),
"https://leor.com.ua"
        %h1
          = t("mailer.customer.title")
        %h2
          = t("mailer.customer.description", "first_name": @basket.first_name)
        = render "shared/order_details", basket: @basket


     !!!
     %html
      %head
        %meta{:content => "text/html; charset=UTF-8", "http-equiv" => "Content-
Type"}/
      %body
        -#    =    link_to    (image_tag    "https://leor.com.ua/assets/leor_logo-
50add2020c7866d2d956e176e586800c2a6c9d23ac8f67f963ee449c6c583310.webp"),
"https://leor.com.ua"
        %h1
          = t("mailer.manager.title")
        = render "shared/order_details", basket: @basket
```

```
def search
   products = Product.includes({ media_files: :photo }).published.search(params[:term]).limit(10).order(:status)
   search_result = products.map do |product|
     product.media_files[0] ? product.attributes.merge("image_url" => product.media_files[0].photo.image.url(:search)) : product.attributes
   end
   respond_to do |format|
     format.json {
       render json: search_result
     }
   end
 end
```

```
function autocompleteSearchForCustomers(options) {

   // var locale = I18n.locale

   var url = "/search_product"

   $('#autocomplete_search').select2({
     minimumInputLength: 3,
       placeholder: I18n.t("search"),
     language: options.locale,
       theme: "bootstrap-5",
     delay: 500,
     width: "100%",
     ajax: {
       url: url,
       dataType: 'json',
       processResults: function (data) {
         return {
           results: $.map(data, function (item) {
               var getUrl = window.location;
               var row = '<span><img src="' + item.image_url + '" /> ' + item.name + " - " + numberToUAH(Number(item.retail_price)) + " - " + I18n.t(item.status) + '</span>';
               // return {
               //     text: item.name,
               //     id: item.alias
```

```
                // }
                return {
                    text: $.parseHTML(row),
                    id: item.alias
                }
            })
          };
        },
        cache: true
      }
    });

    $('#autocomplete_search').on('select2:select', function (e) {
      window.location.href = "/product/" + $('#autocomplete_search').val();
        $('#autocomplete_search').text("");
    });
}
```

```
function ChildrenProductsHandler(params) {

    var canClick = params.can_click;
    $(".specification-item").click(function () {
        if (canClick) {
            var containerId = parentContainerId(this);
            specificationId = $(this).attr('id').split("_")[0];
            itemId = $(this).attr('id').split("_")[1];
            // var elementId = $(this).attr('id');

specifications[containerId].findSpecificationById(specificationId).setActiveItem
(itemId);
            if (specifications[containerId].isAllChecked()) {
                //                var                value                =
specifications[containerId].findItemValueByElementId(elementId);
                var                          product                          =
specifications[containerId].findProductByCheckedSpecifications();
                //                           product                          =
specifications[containerId].findProductBySpecification(specificationId, value);
                reloadProduct(product);
            }
        }
    });

    function parentContainerId(selector) {
        return $(selector).closest(childrenProductsContainer).attr('id');
    }

    function imgContainer(item) {
        var forProductPage = $(".main-img");
        if (forProductPage.length > 0) {
            return forProductPage;
        }
        else {
            return $(item).closest(".product-card")
        }
    }

    $(".specification-item.color").hover(function () {
        var item = this;
```

```
            var containerId = parentContainerId(this);
            var elementId = $(this).attr('id');
            var                              value                              =
specifications[containerId].findItemValueByElementId(elementId);
            var specificationId = elementId.split("_")[0];
            var                             product                             =
specifications[containerId].findProductBySpecification(specificationId, value);
            if (imgContainer(item).length > 0 && product.image_url) {
                specifications[containerId].tempImgUrl                          =
imgContainer(item).find("img")[0].attributes.src.value;
                imgContainer(item).find("img")[0].attributes.src.value          =
product.image_url;
            }
        }, function () {
            var containerId = parentContainerId(this);
            var item = this;
            if         (imgContainer(item).length         >         0         &&
specifications[containerId].tempImgUrl) {
                imgContainer(item).find("img")[0].attributes.src.value          =
specifications[containerId].tempImgUrl;
                specifications[containerId].tempImgUrl = null;
            }
        });

        function reloadProduct(product) {
            window.location.href = "/product/" + product.alias;
        };

    }

    var specifications = {};

    var childrenProductsContainer = ".children-products-container"

    class SpecificationItem {
        constructor(params) {
            this.value = params.value;
            this.checked = params.checked || false;
            this.type = params.type;
        }

        renderItem(id) {
            switch (this.type) {
                case "color":
```

```
                    return this.renderColorItem(id);
                    break;
                case "string":
                    return this.renderStringItem(id);
                    break;
                default:
                    return this.renderStringItem(id);
            }
        }

    renderColorItem(id) {
        var cssClass = "col-auto specification-item color";
        if (this.checked == true) {
            cssClass += " active";
        }
        var html = "<div class='" + cssClass + "' id='" + id + "'
style='background-color: " + this.value + "'></div>";
        return html;
    }

    renderStringItem(id) {
        var cssClass = "col-auto specification-item string";
        if (this.checked == true) {
            cssClass += " active";
        }
        var html = "<div class='" + cssClass + "' id='" + id + "'>" + this.value
+ "</div>"
        return html
    }

    }

    class Specification {
    constructor(params, childrenProducts, defaultValue) {
        this.id = params.id;
        this.name = params.name;
        this.buildSpecificationItems(childrenProducts,            defaultValue,
params.specification_type);
    }

    isChecked() {
        var checked = false;
        $.each(this.specificationItems, function(index, specificationItem) {
            checked = checked || specificationItem.checked;
```

```
        })
        return checked;
    }

    buildSpecificationItems(childrenProducts, defaultValue, type) {
        var specificationItems = [];
        $.each(this.specificationValues(this.id,            childrenProducts),
function(index, value) {
            var checked = false;
            if (defaultValue == value) {
                checked = true;
            }
            specificationItems.push(new  SpecificationItem({  'value':  value,
'checked': checked , 'type': type}))
        });
        this.specificationItems = specificationItems;
    }

    specificationValues(specificationId, childrenProducts) {
        let specificationValues = new Set();
        childrenProducts.forEach((childProduct) => {
            var keyWithLocale = specificationId + "_" + self.I18n.locale;
            if (keyWithLocale in childProduct.specifications) {

specificationValues.add(childProduct.specifications[keyWithLocale]);
            }
            else {

specificationValues.add(childProduct.specifications[specificationId]);
            }
        });
        return Array.from(specificationValues);
    }

    render() {
        var id = this.id
        var html = "<div class='row title'>" + this.name + "</div><div class='row
buttons'>";
        $.each(this.specificationItems, function(index, item) {
            html += " " + item.renderItem(id + "_" + index);
        });
        html += "</div>";
        return html;
    }
```

```
    setActiveItem(itemId) {
       var specificationId = this.id;
       $.each(this.specificationItems, function(index, item) {
           item.checked = false;
           $("#" + specificationId + "_" + index).removeClass("active");
       });
       this.specificationItems[itemId].checked = true;
       $("#" + specificationId + "_" + itemId).addClass("active");
    }

    checkedValue() {
       var value;
       $.each(this.specificationItems, function(index, specificationItem) {
           if (specificationItem.checked) {
               value = specificationItem.value;
           }
       });
       return value;
    }
    }

    class Specifications {

    constructor(params) {
       this.childrenProducts = JSON.parse(params.children_products);
       this.buildSpecifications(JSON.parse(params.specifications),
this.childrenProducts, JSON.parse(params.product_specifications));
    }

    isAllChecked() {
       var checked = true;
       $.each(this.specifications, function(index, specification) {
           checked = checked & specification.isChecked();
       });
       return checked;
    }

    findSpecificationById(id) {
       var response;
       $.each(this.specifications, function(index, specification) {
           if (specification.id == parseInt(id)) {
               response = specification;
           }
```

```
        })
        return response;
    }

    findItemValueByElementId(elementId) {
        var specificationId = elementId.split('_')[0]; //only without
translations
        var itemIndex = elementId.split('_')[1];
        var specification = this.findSpecificationById(specificationId);
        var item = specification.specificationItems[itemIndex];
        return item.value;
    }

    findProductBySpecification(specificationId, value) {
        var findedProduct
        $.each(this.childrenProducts, function(index, product) {
            if (product.specifications[specificationId] == value) {
                findedProduct = product;
            }
        });
        return findedProduct;
    }

    findProductByCheckedSpecifications() {
        var findedProduct;
        var params = this.findParams();
        $.each(this.childrenProducts, function(index, product) {
            var equal = true;
            $.each(params, function(specificationId, value) {
                equal = equal && (product.specifications[specificationId] ==
value);
            })
            if (equal) {
                findedProduct = product;
            }
        });
        return findedProduct;
    }


    buildSpecifications(params, childrenProducts, defaultValues) {
        let specifications = [];
        $.each(params, function(index, specification_params) {
            var key = specification_params.id;
```

```
            if (specification_params.specification_type == "string") {
                key += "_" + self.I18n.locale;
            }
            var defaultValue = defaultValues[key];
            specifications.push(new         Specification(specification_params,
childrenProducts, defaultValue));
        });
        this.specifications = specifications;
    }

    render(containerId, onlyColors = false) {
        $.each(this.specifications, function(index, specification) {
            if    (specification.specificationItems[0].type    ==    'color'    ||
!onlyColors) {
                $("#"               +               containerId               +
childrenProductsContainer).append(specification.render());
            }
        });
    }

    findParams() {
        var params = {};
        $.each(this.specifications, function(index, specification) {
            var key = specification.id;
            if (specification.specificationItems[0].type == "string") {
                key += "_" + self.I18n.locale;
            }
            params[key] = specification.checkedValue();
        })
        return params;
    }
    }
```

```
class SetLanguageController < ApplicationController
  def ukraine
   I18n.locale = :ua
   set_session_and_redirect
 end

 def english
   I18n.locale = :en
   set_session_and_redirect
 end

 private

 def set_session_and_redirect
   session[:locale] = I18n.locale
   redirect_to :back
   rescue ActionController::RedirectBackError
     redirect_to :root
 end
end
```

```
.language-selector
   .language-button
       = link_to_unless I18n.locale == :ua, (image_tag "ua.png"), "/ukraine"
   .language-button
       = link_to_unless I18n.locale == :en, (image_tag "en.png"), "/english"
```

```
module Admin
   class NewPostServicesController < AdminController
       def index

       end

       def update_sities_and_warehouses
           begin
               update_cities(JSON.parse(get_cities))
               update_warehouses(JSON.parse(get_warehouses))
               redirect_to admin_new_post_services_path, notice: "success"
           rescue
               redirect_to admin_new_post_services_path, notice: "error"
           end
       end

       def update_cities(cities)
           cities = cities["data"].map {|city| {
               "id": city["Ref"],
               "description": city["Description"],
               "description_ru": city["DescriptionRu"]}}
           if cities.any?
               City.delete_all
               City.import cities, validate: false
           else
               raise "error"
           end
       end

       def update_warehouses(warehouses)
           warehouses = warehouses["data"].map {|warehouse| {
               "id": warehouse["Ref"],
               "description": warehouse["Description"],
               "description_ru": warehouse["DescriptionRu"],
               "city_id": warehouse["CityRef"]}}
           if warehouses.any?
               Warehouse.delete_all
               Warehouse.import warehouses, validate: false
           else
               raise "error"
           end
       end
```

```
            def get_cities
                url = "http://api.novaposhta.ua/v2.0/json/Address/getCities"
                request_params = {
                        "modelName": "AddressGeneral",
                        "calledMethod": "getCities",
                        "methodProperties": {
                                            },
                        "apiKey": Setting.nova_poshta_api_key
                        }.to_json
                RestClient.post  url,  request_params,  {content_type:  :json,
accept: :json}
            end

            def get_warehouses
                url                                                          =
"http://api.novaposhta.ua/v2.0/json/AddressGeneral/getWarehouses"
                request_params = {
                        "modelName": "AddressGeneral",
                        "calledMethod": "getWarehouses",
                        "methodProperties": {
                                            },
                        "apiKey": Setting.nova_poshta_api_key
                        }.to_json
                RestClient.post  url,  request_params,  {content_type:  :json,
accept: :json}
            end
         end
      end


      %b
         = t(:nova_poshta)
      %br/
      = shipment_info["city_name"]
      %br/
      = shipment_info["warehouse_description"]
```

```
def thank
       if flash[:payment_method] == "liq_pay"
           language = I18n.locale == :ru ? "ru" : "uk"
           language = "ru"
           liqpay = Liqpay::Liqpay.new(
               public_key: ENV['LIQPAY_PUBLIC_KEY'],
               private_key: ENV['LIQPAY_PRIVATE_KEY']
           )
           @liqpay_request = liqpay.cnb_form({
               amount:      flash[:amount],
               currency:    flash[:currency],
               order_id:    flash[:order_id],
               description: flash[:description],
               action:      'pay',
               version:     '3',
               language:    language
           })
       end
   end
```