МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МАГАЗИНУ ПОДІЙ З ВИКОРИСТАННЯМ LARAVEL»**

Виконав(ла): студент(ка) 2 курсу, групи 8.1210
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
А.А. Ковтун
(ініціали та прізвище)

Керівник — професор кафедри програмної інженерії, доцент, д.т.н. Чопоров С.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент — доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

(підпис)

« 09 » 06 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ(СТУДЕНТЦІ)

Ковтуну Артему Андрійовичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка магазину подій з використанням Laravel

керівник роботи (проекту) Чопоров Сергій Вікторович, д.т.н., доцент

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

затверджені наказом ЗНУ від « 09 » червня 2021 року № 851-с

2. Строк подання студентом роботи 24.11.2021

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Постановка задачі.
   2. Основні теоретичні відомості.
   3. Моделювання системи подій.

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

Презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	10.09.2021	
2.	Збір вихідних даних.	11.09.2021	
3.	Обробка методичних та теоретичних	17.09.2021	
	джерел.		
4.	Розробка першого розділу.	25.09.2021	
5.	Розробка другого розділу.	15.10.2021	
6.	Оформлення та нормоконтроль	23.11.2021	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	17.12.2021	

Студент _____ А.А. Ковтун
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.В. Чопоров
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ С.П. Швидка
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка магазину подій з використанням Laravel»: 47 с., 7 рис., 5 табл., 12 джерел, 4 додатки.

MVC, ФРЕЙМВОРК, PHP, LARAVEL, BPMN, IDEF1X, CONTROLLER.

Об’єкт дослідження – продаж подій через інтернет-магазин.

Мета роботи: розробити систему для автоматизації продажу подій.

Методи дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проектування, конструювання та тестування програмних забезпечення.

В даний час багато сервісів працюють за налагодженою схемою, не замислюючись про її ефективність та доцільність. Однак, пропозиція на ринку подій настільки зросла в останні роки, що розробити ефективну систему і вирішити, з якими компаніями в цій сфері співпрацювати, стає не так важко.

При розробці системи було проведено аналіз предметної області, розроблена функціональна модель системи, спроектована інформаційна модель системи. Автор дослідив сучасні технології розробки для реалізації поставленої задачі, що ґрунтуються на розробці систем на базі PHP-фреймворку з відкритим вихідним кодом під назвою Laravel.

SUMMARY

Master’s qualifying paper "Development of the Actions Store using Laravel": 47 pages, 7 figures, 5 tables, 12 references, 4 supplements.

MVC, FRAMEWORK, PHP, LARAVEL, BPMN, IDEF1X, CONTROLLER.

Object of the study – the sale of actions store through an online store.

Aim of the study: to develop a system for automating the sale of actions store.

Research methods – methods of collection and analysis of software requirements, modeling, design, construction and testing of software.

Currently, many online stores work according to a well-established scheme, without thinking about its effectiveness and feasibility. However, the supply of actions store has grown so much in recent years that it is not so difficult to develop an effective system and decide which companies to work with in this area.

During the development of the system the analysis of the subject area was carried out, the functional model of the system was developed, the information model of the system was designed. The author has researched advanced technology development for the implementation of the task based on the development of information systems based on PHP-framework is open source under the name Laravel.

ЗМІСТ

ВСТУП

На цей день у сфері продажу квитків на події існує багато різних автоматизованих систем, які забезпечують інтернет-торгівлю. Однак при детальному ознайомленні з основними існуючими системами продажу подій було виявлено що фунціоналу недостатньо або забагато, не завжди забезпечена підтримка найбільш популярних операційних систем, часто відсутній елементарний інтерфейс користувача або затрати на підтримку системи малими інтернет-магазинами неможлива через високу цінову політику хостингу або постачальника. Тому особливо актуальною в наш час є розробка якісної доступної широкому користувачу крос-платформенної і не ресурсозалежної системи по оформленню дистанційних покупок необхідних спортивних товарів.

Основна мета розробки такої системи полягає в збільшенні ефективності інтернет-торгівлі подіями шляхом забезпечення за допомогою розробленого додатка можливості максимально швидкої та простої взаємодії користувача з системою, а також збільшення обсягів продажу та чіткого моніторингу заказів з боку продавця.

Для досягнення цієї мети потрібно виконати наступні задачі:

- збір інформації для написання специфікацій вимог до системи (технічного завдання);
- написання плану стадій та етапів розробки;
- побудова функціональної структури для автоматизації основних-бізнес процесів та потоків даних;
- створення сценарію основних-бізнес процесів;
- побудова моделі інформаційної системи;
- розробка концепції, архітектури та платформи для реалізації інформаційної системи;
- проектування інформаційної системи;
- реалізація та тестування інформаційної системи;

- оформлення документації до проекту.

В результаті виконання роботи планується мати багатокористувацьку ефективну та просту у користуванні інформаційну систему по автоматизації основних бізнес-процесів продажу подій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження предметної області у сфері інтернет-торгівлі

Впровадження інформаційних технологій у сферу торгівлі є актуальним питанням. Інтернет-магазин допоможе користувачам всесвітнього Інтернет порталу знайти потрібну річ в області подій та заходів. Пройшовши реєстрацію на сайті, користувач становиться частиною цього сайту і має можливість віртуально не покидаючи своєї домівки замовити зацікавлений квиток. Сайт пропонує свої услуги попередньо запропонувавши пройти реєстрацію, без цієї реєстрації список запропонованих послуг дуже сильно зменшується.

Характерними рисами інтернет-магазинів є те, що вони можуть пропонувати значно більшу кількість товарів та послуг, ніж реальні магазини і забезпечувати споживачів значно більшим обсягом інформації, необхідної для прийняття рішення про покупку. Також завдяки використанню Internet-технологій є можливою персоналізація підходу до споживачів з врахуванням попередніх відвідувань магазину та зроблених в ньому покупок та використання інтернет-магазину як ефективного способу маркетингових досліджень (анкетування, конференції покупців і т.п.). Internet-магазини потребують значно менших витрат на утримання та організацію роботи, оскільки у ньому значно обмеженіша матеріально-технічна база (будівлі, споруди, приміщення) та кількість обслуговуючого персоналу. Проте Internet-магазини мають і недоліки. Основними є невизначеність реального існування товару та відповідність його основним параметрам якості, шахрайства при проведенні грошових трансакцій, проблеми з доставкою. Також можна виділити проблему платіжних систем та необхідність виконувати своєчасне адміністрування сайту.

1.2 Огляд існуючих технологій

В наш час існує багато варіантів PHP framework для веб-розробки. Найбільш популярними серед них є:

- Symfony;
- Yii;
- Laravel.

Розглянемо основні переваги та недоліки цих інструментів. Symfony володіє не тільки всесвітнім ім'ям, але і є дуже гнучким. Він включає в себе пакетну і компонентну систему, яка дозволяє вам вибирати потрібні функції PHP або просто використовувати всю інфраструктуру. Недоліками Symfony є використання страшного і могутнього ОРЗ, дуже ресурсномісткий, доволі високий поріг вступу, перенасиченість різного роду сутностями, інтегрований аннотаціонний синтаксис [1].

Yii відомий завдяки тому, як легко його налаштувати. Фреймворк має досить велику продуктивність, що є величезною перевагою для продажу, використовує якісні інструменти безпеки. Але до недоліків варто віднести не дуже гнучке формування роутів, занадто склеєні бібліотеки для frontend'a з backend'ом [2].

З розвитком технологій та мережі Інтернет здобув популярності Laravel, який є сучасним фреймворком, що підходить для широкого кола завдань. Laravel дозволяє використовувати сервіс-провайдер (service provider), завдяки якому можна централізовано підключати необхідні компоненти програми. У Laravel можна легко розширювати будь-які компоненти. Також окремо можна відзначити зручну маршрутизацію і валідацію входять параметрів. Фреймворк дає можливість працювати з різними базами даних, змінювати їх структуру, відкочувати зміни і т.д. При написанні будь-якого проекту його потрібно тестувати – в Laravel представлені функціональні тести (Feature-тести), перевіряючі функціонал проекту (з точки зору користувача), і модульні тести

(Unit-тести), які перевіряють саму логіку проекту [3]. Тому я звернув увагу на цей фреймворк із за найбільшої популярності та універсальності в роботі.

1.3 Огляд аналогів

На сьогоднішній день існує багато інтернет-магазинів, які спеціалізуються на продажу квитків на події. Загалом вони мають подібний функціонал та різняться лише дизайном. Найбільш відомими є:

- Musin;
- Karabas;
- Concert.

Відповісти на питання, який інтернет-магазин краще, не вийде. Вони розвиваються, не уступаючи один одному в конкурентній боротьбі. Для підвищення інтересу до користувачів робиться зручний, унікальний та гарний зовнішній вигляд.

Усі інтернет-магазини є популярними серед користувачів, та задовольняють поставлену перед ними мету. До недоліків існуючих систем можна віднести надто нагромаджений інтерфейс та особливості оплати.

Наприклад, при відкриванні головної сторінки Concert майже на весь екран слайдер з рекламою, що на мій погляд надто виділено. Також на Karabas не дуже комфортно переглядати події.

1.4 Технічне завдання

1.4.1 Найменування і область застосування

Повне найменування системи та її умовне позначення: «Розробка магазину подій з використанням Laravel». Присвоїмо системі коротку назву

«Ukrafisha», що демонструє сферу подій. Інтернет-магазин призначений для просування квитків на заходи, збільшення обсягів продажу, залучення нових покупців.

1.4.2 Підстави для розробки

Підставою для розробки технічного завдання є завдання до кваліфікаційної роботи.

Найменування теми розробки – «Розробка магазину подій з використанням Laravel».

1.4.3 Призначення і цілі створення системи

Мета створення інтернет-магазину полягає в забезпеченні максимальної зручності для відвідувачів, наявності провокуючих до купівлі елементів, – за рахунок чого продажі помітно збільшуються. «Ukrafisha» буде являти собою повнофункціональний магазин, що відповідає всім сучасним вимогам, як з боку презентаційних можливостей, так і що має відношення до зручності роботи з товарним асортиментом.

Інтернет-магазин передбачає можливості:

- перегляд товарів;
- додавання зацікавлених товарів у кошик;
- подивитися прев’ю події;
- перегляд заказів.

Ці можливості необхідні для якісної взаємодії користувача з магазином.

1.4.4 Вимоги до системи в цілому

Робота із інтернет-магазином відбувається наступним чином. При запуску сайту відкривається вікно головної сторінки. При натисканні кнопки реєстрації

користувача відкривається форма реєстрації. При натисканні кнопки «Підтвердити» відбувається реєстрація користувача у системі. При натисканні кнопки «Скасувати» форма реєстрації закривається. При натисканні кнопки авторизації користувач може пройти аутентифікацію шляхом введення коректного логіна та пароля у відповідні поля та подальшим натисканням кнопки «Увійти». На статичної панелі меню присутні кнопки «Усі події», «Категорії», «У корзину».

При натисканні кнопки «Усі події» відображаються доступні до користувача події, при обранні певної події відкривається вікно, що містить обраний захід.

При натисканні на «Категорії» відображається сторінка з категоріями подій які доступні.

При натисканні кнопки «У корзину» відкривається сторінка, що містить інформацію про добавленні у корзину події.

1.4.5 Вимоги до структури додатка

Інтернет-магазин повинен складатися з наступних сторінок:

- головна сторінка, на якій відображаються події заохочення покупців та меню;
- сторінка кошика;
- сторінка каталогу;
- сторінка авторизації користувача;
- сторінка реєстрації користувача;
- сторінка прев’ю товару;
- сторінка заказів.

Ці сторінки чітко організовують моніторинг заказів і продуктів продавцем та процес оформлення заказу покупцем, в якому він може продивитись необхідну інформацію про подію.

1.4.6 Вимоги до клієнтської частини

Для коректної роботи інтернет-магазину потрібне виконання наступних вимог:

- комп’ютер з операційною системою або гаджет Android, iOS;
- сучасний браузер, наприклад Google Chrome;
- підключення до мережі Інтернет;
- хостинг.

Хостинг повинен забезпечити цілодобовий режим роботи системи для контролю, зберігання, оновлення та відновлення даних.

1.4.7 Стадії і етапи розробки

Розробка повинна бути проведена в 6 стадій [4]:

- вивчення аналогів;
- розробка технічного завдання;
- розробка документації технічного проекту;
- робоче проектування;
- реалізація та тестування;
- розробка документації користувача.

На етапі вивчення аналогів виконується вивчення ринку та ознайомлення з вже існуючими інтернет-магазинами, що направлені на вирішення аналогічної проблеми.

На етапі розробки виконується збір усі вимог до вихідного продукту.

На етапі проектування виконується розробка інформаційних та функціональних діаграм які описують проект.

На етапі реалізації та тестування виконується розробка та тестування інтернет-магазину.

Етап розробки документації представляє собою оформлення керівництва користувача до використання інтернет-магазином.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «UKRAFISHA»

2.1 Опис інформаційної системи

Інформаційна система «Ukrafisha» – це інтернет-магазин, що займається продажом квитків різних подій. Фірма приймає замовлення на події різного тематичного призначення.

Користувач виступає в ролі клієнта котрому треба зробити вибір.

Менеджери, які виконують безпосередньо оформлення заказів.

Бухгалтерія займається прийомом оплати на товари і видачою безпосередньо квитанцій.

Менеджери-постачальники займаються пошуком та наповненням актуальної інформації бази даних.

Адміністратор зобов’язаний займатися підтримкою цілісності даних, роблячи дампи даних та слідкувати розмеженням доступу.

Визначимо основні функції системи:

- створення та підтримка актуальності товарів;
- система формування заказу;
- система автоматизації діяльності інтернет-магазину;
- система діяльності користувачів за ролями;
- система оплати бухгалтеру.

Ці функції системи будуть забезпечувати безпосередньо функціювання інтернет-магазину.

2.2 Функціональна модель ІС

Побудуємо діаграму діяльності бізнес-процесу у нотації BPMN (див. рис. 2.1).

Така діаграма ґрунтується на представлені бізнес-процесу у вигляді блок-схеми, що семантично схожа на діаграму діяльності [5].

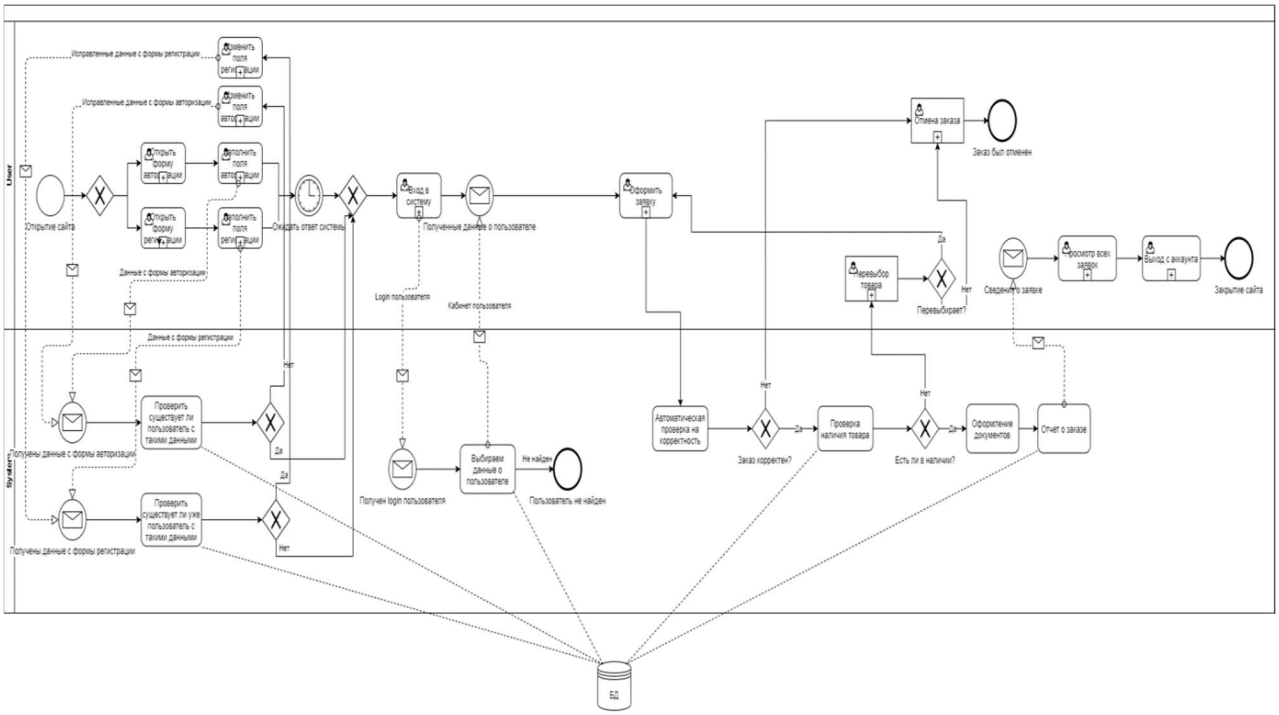

Рисунок 2.1 – Діаграма бізнес-процесу у нотації BPMN

Клієнт бажає скористатися послугами:

- клієнт заходить на сайт проходить реєстрацію або якщо він вже є користувачем сайту авторизується;
- вибирає подію;
- переглядає свій вибір;
- оформлює заказ.

Як показано на діаграмі, користувач взаємодіє з системою протягом усього бізнес-процесу.

2.3 Інформаційна модель ІС

Логічне (дата логічне) проектування це створення схеми бази даних на основі конкретної моделі даних, наприклад, реляційної моделі даних. Для

реляційної моделі даних дата логічна модель – набір схем відносин, зазвичай із зазначенням первинних ключів, а також "зв'язків" між відносинами, що представляють собою зовнішні ключі [6].

На рисунку 2.2 представлено схему бази даних, що використовується у системі.

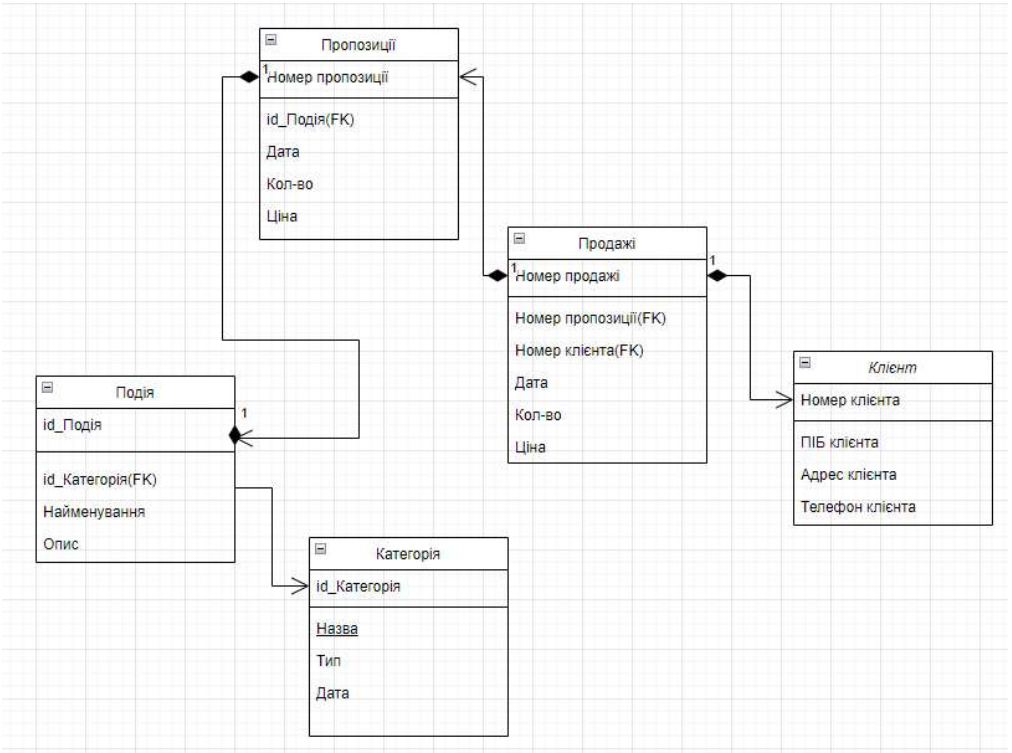

Рисунок 2.2 – IDEF1X –діаграма

База даних для системи складається з 5 таблиць, кожна з яких характеризує окрему сутність. Для кожної сутності бази даних відповідно сформовані таблиці, зображені нижче.

Таблиця «Клієнти» (див. таб. 2.1) містить інформацію про користувача, такі як ім’я, адреса, телефон.

Таблиця 2.1 – Клієнти

Назва поля	Тип даних	Властивості	Опис
Номер клієнта	integer	unique	Містить номер клієнта
ПІБ	string		Містить ПІБ користувача

Продовження таблиці 2.1

Адреса	string		Містить адресу користувача
Телефон	string		Містить телефон користувача

У таблиці «Категорія» (див. таб. 2.2) міститься інформація про категорії, на які діляться події.

Таблиця 2.2 – Категорія

Назва поля	Тип даних	Властивості	Опис
Id_Категорія	integer	unique	Містить id категорії
Назва	string		Містить назву категорії
Тип	string		Містить тип
Дата	date		Містить інформацію щодо дат

Таблиця «Подія» (див. таб. 2.3) зберігає інформації про товари, які доступні у мережі.

Таблиця 2.3 – Подія

Назва поля	Тип даних	Властивості	Опис
Id_Товар	integer	unique	Містить id товару.
Id_Категорія	integer	FK на таблицю «Категорія»	Містить ключ категорії
Найменування	string		Містить назва товару
Опис	string		Містить інформацію про подію

Таблиця «Пропозиція» (див. таб. 2.4) відповідає за поставки, які були проведені.

Таблиця 2.4 – Пропозиція

Назва поля	Тип даних	Властивості	Опис
№ Пропозиції	integer	unique	Містить номер поставки
id_Подія(FK)	integer	FK на таблицю «Подія»	Містить номер товару
Дата	date		Містить дату
Кількість	integer		Містить кількість подій
Ціна	float		Містить ціну

Таблиця «Продажі» (див. таб. 2.5) містить інформацію про продажі.

Таблиця 2.5 – Продажі

Назва поля	Тип даних	Властивості	Опис
Номер продажі	integer	unique	Містить номер продажі
№ Пропозиції(FK)	integer	FK на таблицю «Пропозиції»	Містить номер пропозиції
№ Клієнта(FK)	integer	FK на таблицю «Клієнт»	Містить номер клієнта
Дата	date		Містить дату продажу
Кількість	integer		Містить інформацію про кількість
Ціна	integer		Містить інформацію про ціну

Отже, на логічному рівні виконано нормалізацію бази даних, а також виділено ключі для кожної сутності. Логічні зв'язки реалізовані за допомогою первинних і зовнішніх ключів.

2.4 Діаграма варіантів використання системи

Побудуємо діаграму варіантів використання. За допомогою цієї діаграми виконуються опис функціональних вимог до системи з метою кращого розуміння роботи системи. «Ukrafisha» можна розглядати з точки зору користувача та персоналу.

При використанні системи користувачу та персоналу доступні наступні варіанти (див. рис. 2.3).

Рисунок 2.3 – Діаграма використання

На діаграмі присутні 2 дійові особи (актори): інтернет-магазин, покупець. Вони є зовнішніми по відношенню до моделюючого бізнес-процесу системи.

2.5 Діаграма класів

Для моделювання майбутніх класів, їх атрибутів та методів, а також зв’язків використовуються діаграми класів. Для побудови діаграми класів скористаємось методологією IDEF4.

IDEF4 – це об'єктно-орієнтована метод проектування різних систем. Він був розроблений для забезпечення переходу від предметної області і вимог до об'єктно-орієнтованим моделям, де відбивається їх структура, принципи взаємодії. Об'єкти проектуються з достатньою деталізацією, що дає можливість аналізувати їх вихідну сутність [7].

Функціонально, система складається з наведених нижче класів, які є схемою роботи інтернет-магазину. Для системи було побудовано діаграму класів, що представлена на рисунку 2.4.

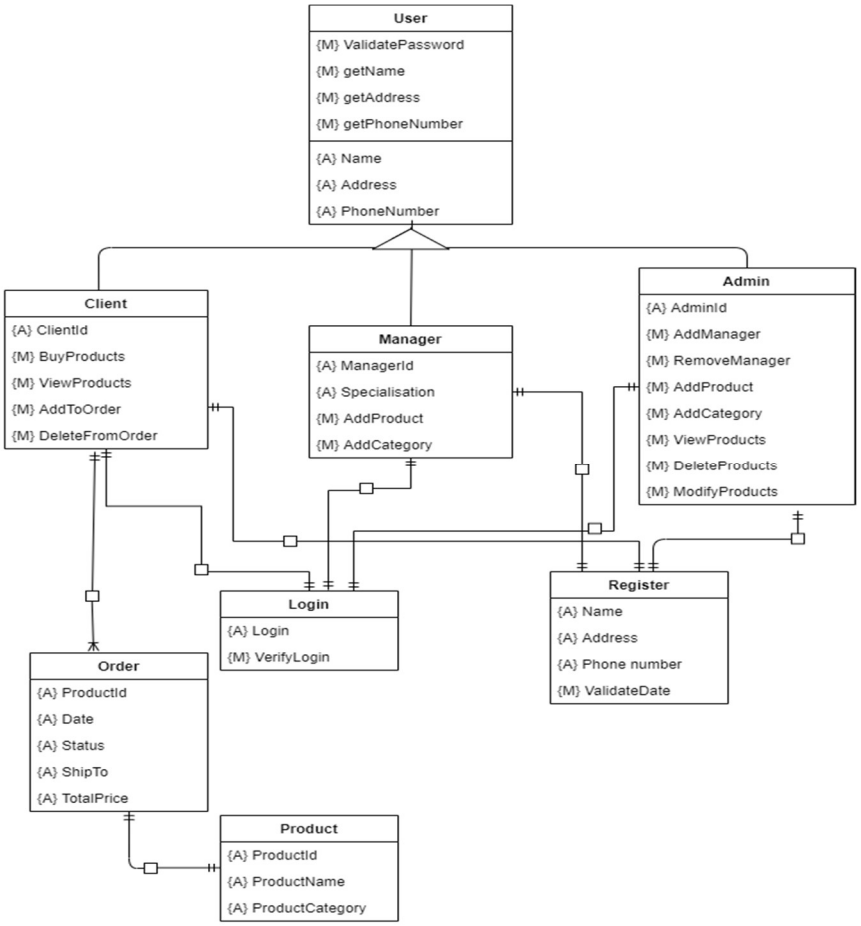

Рисунок 2.4 – Діаграма класів

Перелік основних класів-сутностей:

- User – клас, що представляє сутність користувача у додатку. Містить гетери та сетери для роботи з даними користувача. Має спадкоємців: Client, Manager, Admin.
- Login – клас, що представляє собою модель авторизації в системі.

- Register- клас, що представляє об’єкт регістрації у системі.
- Order – клас, що є моделлю заказу клієнта у системі.
- Product – клас, що є товаром.

Клас на діаграмі показується у вигляді прямокутника, розділеного на 2 області. У верхній міститься назва класу, в нижній спочатку опис атрибутів (властивостей), а далі назви операцій - послуг, що надаються об'єктами цього класу.

2.6 Діаграма розгортання

Побудуємо діаграму розгортання (див. рис. 2.5). Діаграма розгортання відображає фізичні взаємозв'язку між програмними і апаратними компонентами системи, що розробляється. Одним з істотних понять даного виду діаграм є поняття вузол. Вузол є певний тип обчислювального пристрою, як правило, самостійну частину апаратури [8].

Діаграми розгортання допомагають моделювати апаратну топологію системи порівняно з іншими типами UML-діаграм, які здебільшого описують логічні компоненти системи.

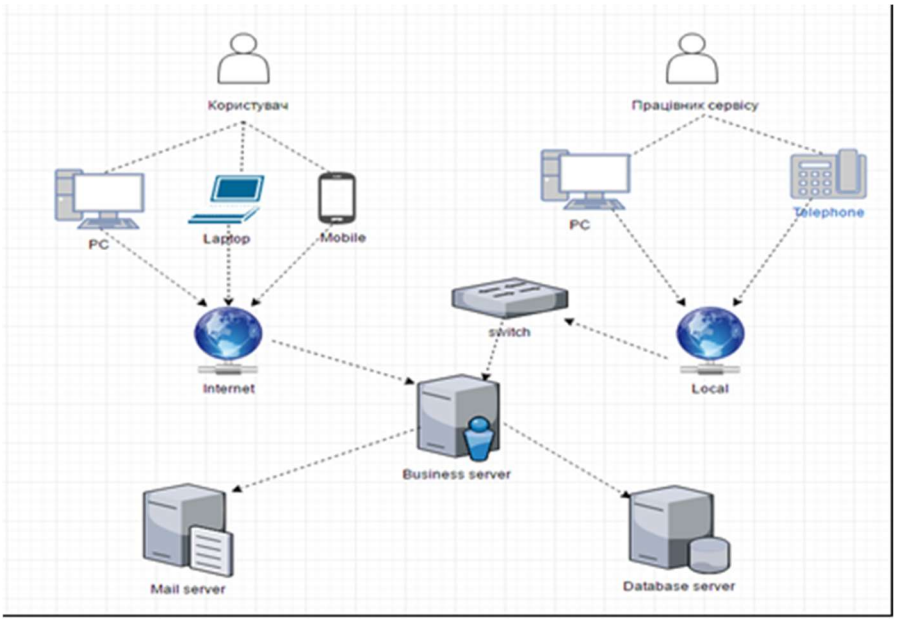

Рисунок 2.5 – Діаграма розгортання

Діаграма розгортання відображає як користувач через комп’ютер або інші гаджети за допомогою інтернету звертається до системи. Також як працівник магазину відповідає на запити. Тим часом дані зберігаються у базі даних серверу.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Для реалізації був використаний PHP фреймворк Laravel. Laravel працює за принципом MVC – model, viev, controller [9].

CRM розроблена у відповідність із цим патерном, логічна модель може бути перенесена на іншу платформу.

Компонент Bootstrap використовувався як допоміжний інструмент під час реалізації для гарного інтерфейсу сайту [10].

CRM складається з залежних частин через зв'язки в БД, які відображені в моделі, модуль користувачів йде окремо, через Auth controller.

3.2 Процес створення міграцій

Database/migrations – це директорія зберігає міграції бази даних. Для кожної таблиці створюється окремий файл міграції, в якому описуються всі поля для введення інформації. Міграції представлено в додатку А.

Таблиця «Користувачі»: реалізація міграції «create_users_table». Має поля для введення ПІБ, адреси електронної пошти, пароля, статус админа.

Таблиця «Подія»: реалізація міграції «create_products_table». Має поля номер категорії, назва, код, опис, картинка, ціна, часова мітка.

Таблиця «Закази»: реалізація міграції «create_orders_table». Має поля номер заказу, код, опис, дату, статус.

Також створені міграції для інших таблиць («Категорії», «Заказ продукта») аналогічно вищеописаним міграцій з необхідними полями, які залежать одна від одної.

3.3 Процес створення контролерів

Залежно від того, з якого url-адресою перейде користувач, буде задіяний той чи інший контролер. Контролери можуть групувати пов'язану з обробкою HTTP-запитів логіку в окремий клас. Контролери зберігаються в папці app / Http / Controllers. Контролери представлено в додатку Б.

Реалізований контролер MainController.php, клас – «MainController», в якому реалізовані всі стандартні методи:

- index, повертає представлення products;
- categories, повертає представлення categories, що містить категорії;
- category, перенаправляє на уявлення одної категорії;
- product, повертає представлення products для перегляду продуктів;

Є 2 контролери для реєстрації та авторизації користувача:

LoginController.php – перевірка користувачів, якщо адмін, то перенаправляє на панель адміна, в іншому випадку на головну.

RegisterController.php – відповідає за реєстрацію користувачів.

Кошик: контролер BasketController.php, клас – «BasketController» має функції перевірки кошика, додавання, видалення товарів з кошика.

Закази: контролер OrderController.php, клас – «OrderController» є функція представлення заказів, доступно тільки авторизованим користувачам.

Категорії: контролер CategoryController.php, клас – «CategoryController» є функція представлення, додавання, зберігання, показу, редагування, оновлення, видалення та доступно тільки адміністратору.

Продукт: контролер ProductController.php, клас – «ProductController» є функція представлення, додавання, зберігання, показу, редагування, оновлення, видалення та доступно тільки адміністратору.

Закази: контролер OrderController.php, клас – «OrderController» є функція представлення усіх заказів та доступно тільки адміністратору, для контролю над усіма заказами.

3.4 Процес маршрутизації

Всі маршрути (routes) визначені в файлі app / routes / web.php, який автоматично завантажується фреймворком. Фрагмент коду представлений в додатку В. Маршрути зіставляють url-адреса з методом певного контролера для кожного елемента.

3.5 Процес створення представлення

Уявлення (views), вони ж макети, містять HTML-код, який передається додатком. Це зручний спосіб поділу бізнес-логіки і логіки відображення інформації. Уявлення знаходяться в каталозі resources/views та відображають сторінку користувачеві. Фрагмент коду представлений в додатку Г.

Директорія Layouts відповідає за зберігання основного макета, який є каркасом для всіх сторінок і включає в себе основну структуру. Додаток має елементи, які присутні на кожній сторінці сайту (навігація, підвал і так далі), Laravel спрощує використання цих загальних функцій на всіх сторінках, використовуючи макети Blade. Це дозволяє сформувати грамотну і зручну для роботи структуру, а також домогтися гнучкості в управлінні.

Blade надає можливість імпортувати інші файли шаблонів попередньо перевіряючи їх наявність. Якщо файл відсутній, то процес формування сторінки не буде перерваний і не виникне проблем відображення сторінки.

3.6 Процес створення сидерів

Директорія, яка зберігає файли с класами, що реалізують заповнення таблиці бази даних початковими даними. Для реалізації заповнення таблиці

«Події», реалізований Клас «ProductsTabelSeeder», який представлений в додатку Д.

Також сидери реалізовані для деяких таблиць: «Категорії», «Користувачі».

3.7 Ієрархія даних проекту

Як вже було зазначено у першому розділі, система створена на основі патерну Model-View-Controller. Тому дерево проекту має вигляд, наведений на рис. 3.1.

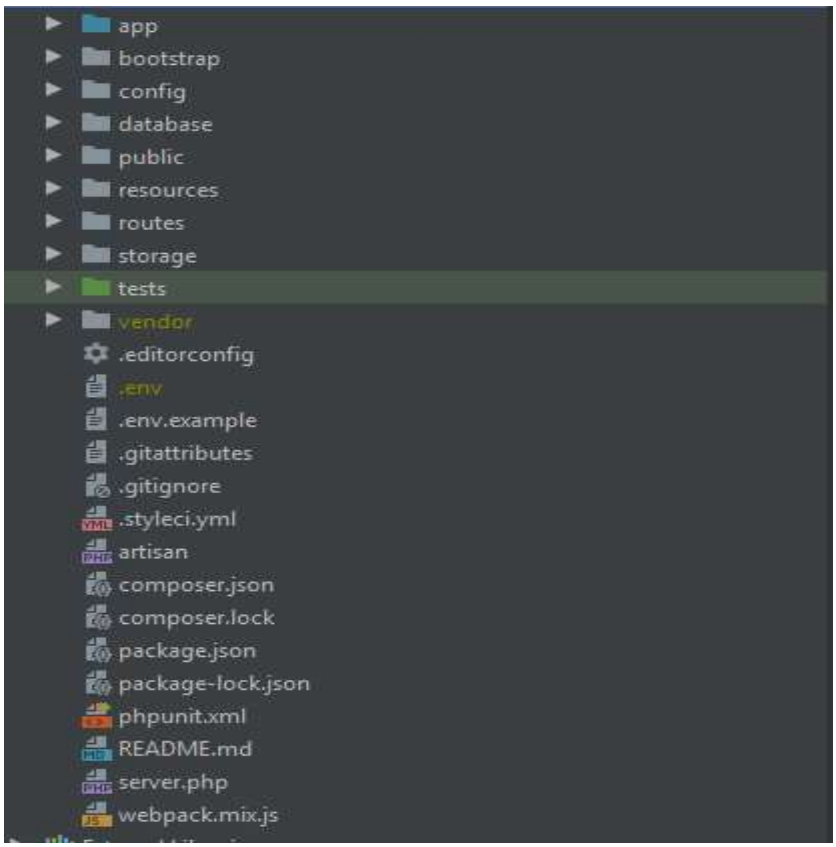

Рисунок 3.1 – Каталог проекту

Каталог проекту є складним:

- database / migrations – містить файли міграцій баз даних;

- database / seeds – зберігає файли с класами, що реалізують заповнення таблиці бази даних тестовими значеннями;
- app / http / controllers – містить контролери проекту;
- app / http / middleware – тут розташовуються посередники проекту;
- routes / web.php – файл роутінга (маршрутів користувача);
- resources / views – зберігає уявлення, що містять HTML-код,
- переданий додатком;
- app / providers – дана директорія зберігає провайдерів проекту;
- resources / images – зберігає картинки до проекту;
- public / css – містить файли стилів.

Отже, проект містить багато підкаталогів та файлів які підтримують функціювання системи в цілому.

3.8 Тестування

3.8.1 Тест-кейс

Тест-кейс – це професійна документація тестувальника, послідовність дій спрямована на перевірку будь-якого функціоналу, що описує як прийти до фактичного результату.

Тест-кейси для перевірки продукту без ознайомлення з усією документацією. Написаний один раз, зручний в підтримці тест-кейс заощадить багато часу і сил тестувальникам [11].

3.8.2 Контрольний список

Контрольний список (Check list) – це документ, який містить ряд необхідних перевірок під час тестування програмного продукту [12]. При цьому чек-лист може бути абсолютно різного рівня деталізації. Він дозволяє не

забувати про важливі тести, фіксувати результати своєї роботи і відслідковувати статистику про статус програмного продукту (див. рис. 3.2).

	IE10	FF	GChrome
Реєстрація та кабінет			
Реєстрація	Passed	Passed	Passed
Авторизація	Passed	Passed	Passed
Особистий кабінет	Passed	Passed	Passed
Функціонал користувача			
Перегляд заказов/заказа	Passed	Passed	Passed
Перегляд товаров/товара	Passed	Passed	Passed
Перегляд кошика	Passed	Passed	Passed
Додавання в кошик	Passed	Passed	Passed
Видалення с кошика	Passed	Passed	Passed
Функціонал адміністратора			
Перегляд заказів	Passed	Passed	Passed
Перегляд товарів	Passed	Passed	Passed
Перегляд категорій	Passed	Passed	Passed
Додавання, редагування, видалення категорій	Passed	Passed	Passed
Додавання, редагування, видалення товарів	Passed	Passed	Passed

Рисунок 3.2 – Чек-лист функціоналу

Як видно з результатів тестування, представлених на рис. 3.2, реєстрація, авторизація, функціонал користувача та адміністратора інтернет-магазину підтримується на основних браузерах без перешкод.

ВИСНОВКИ

В результаті виконання роботи було розроблено багатокористувацьку інформаційну систему по автоматизації основних бізнес-процесів продажу подій. У відповідності з поставленими завданнями технічного завдання були виконані наступні етапи створення технічного проекту:

- зроблен аналіз предметної області;
- обрана архітектура побудови і платформа реалізації системи;
- спроектована концептуальна модель «Ukrafisha»;
- спроектована логічна модель системи «Ukrafisha» на основі концептуальної моделі;
- розроблена діаграма BPMN 2.0, інструментом Draw.io;
- розроблені діаграми моделей даних інструментом ERwin Data Modeler (IDEF1X);
- визначена фізична структура сервера баз даних, MySQL;
- спроектована інформаційна система інструментом Rational Rose UML 2.0 в склад якої входять діаграми: використання, класів, розгортання;
- здійснене тестування системи та оформлена документації до проекту.

«Ukrafisha» хороша тим що дає можливість в простому використанні сайту з боку клієнта і з боку управління менеджером, адміністратором. Швидкий вибір товарів та оформлення заказу. Виходячи з цього розроблена система продажу подій за допомогою фреймворку Laravel.

ПЕРЕЛІК ПОСИЛАНЬ

1. Стаффер М. Laravel. Полное руководство : учебное пособие. Санкт-Петербург: O`Reilly, 2020. 512 с.

2. 8 лучших PHP Framework для веб-разработчиков: керівництво. URL : https://www.hostinger.com.ua/rukovodstva/8-luchshih-php-framework-dla-web-razrabotchikov/ (дата звернення: 14.09.2021).

3. Топ-5 PHP-фреймворков: Laravel vs Yii vs Symfony, плюсы и минусы: керівництво. URL : https://mkdev.me/posts/top-5-php-freymvorkov-laravel-vs-yii-vs-zend-vs-phalcon-vs-symfony-plyusy-i-minusy (дата звернення: 15.09.2021).

4. Процес розробки програмного забезпечення: керівництво. URL : https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F (дата звернення: 10.09.2021).

5. Ввод в нотацию BPMN: керівництво. URL : https://www.elma-bpm.ru/journal/urok-1-vvod-v-notaciyu-bpmn/ (дата звернення: 25.09.2021).

6. Кузнецов С. Базы данных. Вводный курс. Семантическая модель Entity-Relationship : учебное пособие. Москва : Бином, 2009. 251 с.

7. Цуканова О. А. Методология и инструментарий моделирования бизнес-процессов : учебное пособие. Санкт-Петербург : Университет ИТМО, 2015. 101 с.

8. Грейди Б., Джеймс Р., Айвар Д. Язык UML. Руководство пользователя : учебное пособие. Москва : ДМК Пресс, 2004. 483 с.

9. Laravel. URL : https://ru.wikipedia.org/wiki/Laravel (дата звернення: 17.09.2021).

10. Bootstrap 3. Документация на русском: руководство. URL : http://bootstrap-3.ru/index.php (дата звернення: 20.09.2021).

11. Савинов Р. Тестирование дот ком: учебное пособие. Минск : Дело, 2007. 312 с.

12. Что такое чек-листы и как с ними работать: руководство. URL : https://training.qatestlab.com/blog/technical-articles/work-with-checklist/ (дата звернення: 18.10.2021).

ДОДАТОК А

Опис міграцій

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email');
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateProductsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id('id');
```

```
            $table->integer('category_id');
            $table->string('name');
            $table->string('code');
            $table->text('description')->nullable();
            $table->text('image')->nullable();
            $table->double('price')->default(0);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('products');
    }
}
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateCategoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id('id');
            $table->string('name');
            $table->string('code');
            $table->text('description')->nullable();
            $table->text('image')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('categories');
    }
}
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
class CreateOrdersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('orders', function (Blueprint $table) {
            $table->id('id');
            $table->tinyInteger('status')->default(0);
            $table->string('name')->nullable();
            $table->string('phone')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('orders');
    }
}
```

ДОДАТОК Б

Опис контролерів

```
<?php

namespace App\Http\Controllers;

use App\Category;
use App\Product;
use Illuminate\Http\Request;

class MainController extends Controller
{
    public function index() {
        $products = Product::get();
        return view('index', compact('products'));
    }
    public function categories() {
        $categories = Category::get();
        return view('categories', compact('categories'));
    }
    public function category($code) {
        $category = Category::where('code', $code)->first();

        return view('category', compact('category'));
    }
    public function product($category, $productCode) {
        $product = Product::byCode($productCode)->first();
        return view('product', compact('product'));
    }
}
```

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Artisan;
use Illuminate\Support\Facades\Storage;

class ResetController extends Controller
{
    public function reset() {
        Artisan::call('migrate:fresh --seed');

        foreach (['categories', 'products'] as $folder) {
            Storage::deleteDirectory($folder);
            Storage::makeDirectory($folder);

            $files = Storage::disk('reset')->files($folder);

            foreach ($files as $file) {
                Storage::put($file, Storage::disk('reset')->get($file));
            }
        }
```

```
        session()->flash('success', 'Проект был сброшен до начального состояния');
        return redirect()->route('index');
    }
}

<?php

namespace App\Http\Controllers;

use App\Order;
use App\Product;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class BasketController extends Controller
{
    public function basket() {
        $orderId = session('orderId');
        if (!is_null($orderId)) {
            $order = Order::findOrFail($orderId);
        }
        return view('basket', compact('order'));
    }

    public function basketConfirm(Request $request) {
        $orderId = session('orderId');
        if (is_null($orderId)) {
            return redirect()->route('index');
        }
        $order = Order::find($orderId);
        $success = $order->saveOrder($request->name, $request->phone);

        if ($success) {
            session()->flash('success', 'Ваш заказ принят в обработку!');
        } else {
            session()->flash('warning', 'Случилась ошибка');
        }
        return redirect()->route('index');
    }

    public function basketPlace() {
        $orderId = session('orderId');
        if (is_null($orderId)) {
            return redirect()->route('index');
        }
        $order = Order::find($orderId);
        return view('order', compact('order'));
    }

    public function basketAdd($productId) {
        $orderId = session('orderId');
        if ( is_null($orderId)) {
            $order = Order::create();
            session(['orderId' => $order->id]);
        } else {
            $order = Order::find($orderId);
        }

        if($order->products->contains($productId)) {
            $pivotRow = $order->products()->where('product_id', $productId)->first()->pivot;
            $pivotRow->count++;
            $pivotRow->update();
```

```
        } else {
            $order->products()->attach($productId);
        }

        if (Auth::check()) {
            $order->user_id = Auth::id();
            $order->save();
        }

        $product = Product::find($productId);
        session()->flash('success', 'Добавлен товар ' . $product->name);

        return redirect()->route('basket');
    }

    public function  basketRemove($productId) {
        $orderId = session('orderId');
        if ( is_null($orderId)) {
            return redirect()->route('basket');
        }
        $order = Order::find($orderId);

        if ($order->products->contains($productId)) {
            $pivotRow = $order->products()->where('product_id', $productId)->first()->pivot;
            if ($pivotRow->count < 2) {
                $order->products()->detach($productId);
            } else {
                $pivotRow->count--;
                $pivotRow->update();
            }
        }
        $product = Product::find($productId);
        session()->flash('warning', 'Удален товар ' . $product->name);

        return redirect()->route('basket');
    }
}
```

```
<?php

namespace App\Http\Controllers\Admin;

use App\Category;
use App\Http\Controllers\Controller;
use App\Http\Requests\CategoryRequest;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class CategoryController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $categories = Category::get();
        return view('auth.categories.index', compact('categories'));
    }

    /**
```

```
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    return view('auth.categories.form');
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(CategoryRequest $request)
{
    $params = $request->all();
    unset($params['image']);
    if ($request->has('image')) {
        $params['image'] = $request->file('image')->store('categories');

        Category::create($params);
        return redirect()->route('categories.index');
    }
}

/**
 * Display the specified resource.
 *
 * @param \App\Category $category
 * @return \Illuminate\Http\Response
 */
public function show(Category $category)
{
    return view('auth.categories.show', compact('category'));
}

/**
 * Show the form for editing the specified resource.
 *
 * @param \App\Category $category
 * @return \Illuminate\Http\Response
 */
public function edit(Category $category)
{
    return view('auth.categories.form', compact('category'));
}

/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param \App\Category $category
 * @return \Illuminate\Http\Response
 */
public function update(CategoryRequest $request, Category $category)
{
    $params = $request->all();
    unset($params['image']);
    if ($request->has('image')) {
        Storage::delete($category->image);
```

```
            $params['image'] = $request->file('image')->store('categories');
        }

        $category->update($params);
        return redirect()->route('categories.index');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param \App\Category $category
     * @return \Illuminate\Http\Response
     */
    public function destroy(Category $category)
    {
        $category->delete();
        return redirect()->route('categories.index');
    }
}
```

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Order;
use Illuminate\Http\Request;

class OrderController extends Controller
{
    public function index()
    {
        $orders = Order::where('status', 1)->get();
        return view('auth.orders.index', compact('orders'));
    }

    public function show(Order $order) {
        return view('auth.orders.show', compact('order'));
    }
}
```

```
<?php

namespace App\Http\Controllers\Admin;

use App\Category;
use App\Http\Controllers\Controller;
use App\Http\Requests\ProductRequest;
use App\Product;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class ProductController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $products = Product::paginate(10);
```

```
    return view('auth.products.index', compact('products'));
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    $categories = Category::get();
    //$properties = Property::get();
    return view('auth.products.form', compact('categories'));
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(ProductRequest $request)
{
    $params = $request->all();

    unset($params['image']);
    if ($request->has('image')) {
        $params['image'] = $request->file('image')->store('products');
    }

    Product::create($params);
    return redirect()->route('products.index');
}

/**
 * Display the specified resource.
 *
 * @param  \App\Product  $product
 * @return \Illuminate\Http\Response
 */
public function show(Product $product)
{
    return view('auth.products.show', compact('product'));
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  \App\Product  $product
 * @return \Illuminate\Http\Response
 */
public function edit(Product $product)
{
    $categories = Category::get();
    //$properties = Property::get();
    return view('auth.products.form', compact('product', 'categories'));
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
```

```
     * @param  \App\Product  $product
     * @return \Illuminate\Http\Response
     */
    public function update(ProductRequest $request, Product $product)
    {
        $params = $request->all();
        unset($params['image']);
        if ($request->has('image')) {
            Storage::delete($product->image);
            $params['image'] = $request->file('image')->store('products');
        }

        $product->update($params);
        return redirect()->route('products.index');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Product  $product
     * @return \Illuminate\Http\Response
     */
    public function destroy(Product $product)
    {
        $product->delete();
        return redirect()->route('products.index');
    }
}

<?php

namespace App\Http\Controllers\Person;

use App\Http\Controllers\Controller;
use App\Order;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class OrderController extends Controller
{
    public function index()
    {
        $orders = Auth::user()->orders()->where('status', 1)->get();
        return view('auth.orders.index', compact('orders'));
    }

    public function show(Order $order) {
        if (!Auth::user()->orders->contains($order)) {
            return back();
        }
        return view('auth.orders.show', compact('order'));
    }
}
```

ДОДАТОК В

Опис маршрутів

```
<?php

use Illuminate\Support\Facades\Route;

Auth::routes([
    'reset' => false,
    'confirm' => false,
    'verify' => false,
]);

Route::get('reset', 'ResetController@reset')->name('reset');

Route::get('/logout', 'Auth\LoginController@logout')->name('get-logout');

Route::middleware(['auth'])->group(function () {
    Route::group([
        'prefix' => 'person',
        'namespace' => 'Person',
        'as' => 'person.',
    ], function () {
        Route::get('/orders', 'OrderController@index')->name('orders.index');
        Route::get('/orders/{order}', 'OrderController@show')->name('orders.show');
    });

    Route::group([
        'namespace' => 'Admin',
        'prefix' => 'admin',
    ], function () {
        Route::group(['middleware' => 'is_admin'], function () {
            Route::get('/orders', 'OrderController@index')->name('home');
            Route::get('/orders/{order}', 'OrderController@show')->name('orders.show');
        });

        Route::resource('categories', 'CategoryController');
        Route::resource('products', 'ProductController');
    });
});


Route::get('/', 'MainController@index')->name('index');
Route::get('/categories', 'MainController@categories')->name('categories');

Route::post('/basket/add/{id}', 'BasketController@basketAdd')->name('basket-add');

Route::group([
    'middleware' => 'basket_not_empty',
    'prefix' => 'basket',
], function () {
    Route::get('/', 'BasketController@basket')->name('basket');
    Route::get('/place', 'BasketController@basketPlace')->name('basket-place');
    Route::post('/remove/{id}', 'BasketController@basketRemove')->name('basket-remove');
    Route::post('/place', 'BasketController@basketConfirm')->name('basket-confirm');
});
```

```
Route::get('/{category}', 'MainController@category')->name('category');
Route::get('/{category}/{product?}', 'MainController@product')->name('product');
```

ДОДАТОК Г

Опис відображень

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">

   <title>Интернет Магазин: @yield('title')</title>

   <link href="https://fonts.googleapis.com/css?family=Raleway:100,600" rel="stylesheet" type="text/css">
   <script src="/js/jquery.min.js"></script>
   <script src="/js/bootstrap.min.js"></script>
   <link href="/css/bootstrap.min.css" rel="stylesheet">
   <link href="/css/starter-template.css" rel="stylesheet">
</head>
<body>
<nav class="navbar navbar-inverse navbar-fixed-top">
   <div class="container">
      <div class="navbar-header">
         <a class="navbar-brand" href="{{route('index')}}">Интернет Магазин</a>
      </div>
      <div id="navbar" class="collapse navbar-collapse">
         <ul class="nav navbar-nav">
            <li @routeactive('index')><a href="{{route('index')}}">Все события</a></li>
            <li @routeactive('categor*')><a href="{{route('categories')}}">Категории</a>
            </li>
            <li @routeactive('basket*')><a href="{{route('basket')}}">В корзину</a></li>
            <li><a href="{{route('reset')}}">Сбросить проект в начальное состояние</a></li>
         </ul>

         <ul class="nav navbar-nav navbar-right">
            @guest
               <li><a href=" {{ route('login') }}">Войти</a></li>
            @endguest

            @auth
               @admin
                  <li><a href="{{ route('home') }}">Панель админа</a></li>
               @else
                     <li><a href="{{ route('person.orders.index') }}">Мои заказы</a></li>
               @endadmin
               <li><a href=" {{ route('get-logout') }}">Выйти</a></li>
            @endauth
         </ul>
      </div>
   </div>
</nav>

<div class="container">
   <div class="starter-template">
      @if(session()->has('success'))
         <p class="alert alert-success">{{ session()->get('success') }}</p>
```

```
        @endif
        @if(session()->has('warning'))
            <p class="alert alert-warning">{{ session()->get('warning') }}</p>
        @endif
        @yield('content')
    </div>
</div>
</body>
</html>
```