МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ МАГАЗИНУ ОДЯГУ З ВИКОРИСТАННЯ YII2»**

Виконав: студент 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Д.І. Литвин
(ініціали та прізвище)
Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

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

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Литвину Данилу Ігоровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет магазину одягу з використання Yii2

керівник роботи Лісняк Андрій Олександрович, доцент, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проєктування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

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

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.02.2023	
2.	Збір вихідних даних.	20.02.2023	
3.	Обробка методичних та теоретичних	13.03.2023	
	джерел.		
4.	Розробка першого та другого розділу.	17.04.2023	
5.	Розробка третього розділу.	18.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	22.06.2023	

Студент _____ Д.І. Литвин
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет магазину одягу з використання Yii2»: 62 с., 21 рис., 13 джерел, 1 додаток.

BACKEND, COOKIE, FRONTEND, LIQPAY, MYSQL, PHP, WEB-СЕРВЕР, YII2.

Об’єкт дослідження – система, інструменти Yii2 для створення інтернет-магазинів, засоби взаємодії системи з користувачем.

Мета роботи – розробити інтернет магазину одягу з використання Yii2.

Метод дослідження – аналіз, порівняння, моделювання.

З постійним зростанням популярності електронної комерції, розробка інтернет-магазину одягу стала надзвичайно важливою для брендів і підприємств у модній індустрії. Це потужний інструмент, який дозволяє залучати нових клієнтів, розширювати ринок збуту і забезпечувати зручний спосіб покупки для споживачів.

Віртуальні магазини надають споживачам зручність покупок в будь-який зручний для них час і місце. Клієнти можуть переглядати продукти, порівнювати ціни і характеристики, здійснювати покупку всього за кілька кліків. Це особливо важливо для сучасних людей, які швидко рухаються і мають обмежений час для покупок.

Таким чином, за результатами роботи створено зручний та багатофункціональний інтернет-магазин одягу з використанням Yii2.

SUMMARY

Bachelor’s qualifying paper «Development of The Clothes Online Store Using Yii2»: 62 pages, 21 figures, 13 references, 1 supplement.

BACKEND, COOKIE, FRONTEND, LIQPAY, MYSQL, PHP, WEB-SERVER, YII2.

The object of the study is the system, Yii2 tools for creating online stores, means of system interaction with the user.

The aim of the study is to develop an online clothing store using Yii2.

The research method includes analysis, comparison, and modeling.

With the growing popularity of e-commerce, the development of an online clothing store has become extremely important for brands and companies in the fashion industry. It is a powerful tool that allows attracting new customers, expanding the market, and providing a convenient way of shopping for consumers.

Virtual stores provide customers with the convenience of shopping at any time and place that is convenient for them. Customers can browse products, compare prices and features, and make purchases with just a few clicks. This is particularly important for modern individuals who are constantly on the move and have limited time for shopping.

Thus, as a result of the work, a convenient and multifunctional online clothing store has been created using Yii2.

ЗМІСТ

ВСТУП

У сучасному цифровому світі електронна комерція набуває все більшої популярності. Інтернет-магазини стають невід'ємною частиною ринку роздрібної торгівлі, забезпечуючи зручний спосіб покупок для мільйонів людей по всьому світу. Однією з найпопулярніших категорій товарів в електронній комерції є одяг. Розробка інтернет-магазину одягу є важливим кроком для будь-якого бренду або підприємства, що спеціалізується на продажу модного одягу.

Інтернет-магазини пропонують зручний спосіб придбання товарів. Покупці можуть замовляти одяг в будь-який зручний для них час і місце, не виходячи з дому. Це особливо зручно для людей з обмеженими можливостями або тих, хто має напружений графік роботи.

Запуск інтернет-магазину вимагає значно менших витрат, ніж відкриття фізичного магазину. Ви не потребуєте фізичного приміщення, дорогого обладнання або великого персоналу. Це дозволяє зосередитися на розвитку продукту та маркетингу.

Виходячи з цього, було вирішено створити систему, котра би мала зрозумілий інтерфейс та надавала необхідний функціонал користувачам.

Актуальність дослідження: актуальність теми зумовлена зростанням популярності електронної комерції та можливості охопити ширший ринок.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження:

Мета: розробити інтернет-магазин одягу.

Задачі:

1) сформувати вимоги до системи;
2) спроєктувати та побудувати архітектуру системи;
3) реалізувати інтернет-магазин одягу;
4) протестувати роботу системи.

Об'єкт дослідження: процес взаємодії системи з користувачем, інструменти для реалізації інтернет-магазину, функціонал системи, необхідний

користувачам.

Предмет дослідження: створення системи, що дозволяє користувачам купити товар з будь-якої точки світу.

Методи дослідження: моделювання, проєктування, програмний, аналітичний.

Перший розділ присвячено збору та аналізуванню вимог до системи, огляду інструментів розробки та опису системи.

У другому розділі розглянуто етапи проєктування системи, наведено структуру бази даних.

Третій розділ присвячено реалізації та тестуванню роботи системи, наведено детальне керівництво користувача, яке описує процес роботи з інтернет-магазином одягу.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

1.1.1 Загальні терміни

Система – інтернет-магазин одягу, розроблений засобами Yii2.

Yii2 – це високопродуктивний PHP-фреймворк, який використовується для розробки веб-додатків. Він набув широкої популярності серед розробників завдяки своїй ефективності, гнучкості і простоті використання.

MySQL – це відкрите програмне забезпечення для управління базами даних, що використовує мову запитів SQL (Structured Query Language).

ДВІ – Діаграма Варіантів Використання чи Use Case Diagram.

ДК – Діаграма Класів.

ДП – Діаграма Послідовності.

ДР – Діаграма Розгортання.

Користувач – людина, котра зареєстрована у системі та має певні права доступу.

Адміністратор – користувач, який має повні права доступу до системи.

1.1.2 Технічні терміни

БД – база даних, місце збереження інформації системи.

MVC – архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

1.2 Функціональні вимоги

1.2.1 Призначення і цілі створення системи

Функціональне призначення системи – реалізувати функціональні можливості інтернет-магазину.

Експлуатаційне призначення системи: система може експлуатуватися користувачем системи.

Мета створення системи – розробка інтернет-магазину одягу.

1.2.2 Загальні функціональні можливості системи

Система має надавати всім користувачам такі можливості:

- перегляд всіх товарів;
- сортування за категоріями;
- перегляд картки товару;
- додавання товару в кошик;
- зміна кількості товару;
- видалення товару з кошику;
- оплата товару через платіжну систему;
- оформлення замовлення.

Система має надавати адміністраторам такі можливості:

- вхід/вихід до/з системи;
- дії над категоріями/товарами/замовленнями:
    - створення;
    - видалення;
    - перегляд;
    - редагування.

1.3 Нефункціональні вимоги

1.3.1 Інтерфейс користувача

Система повинна відображати коректно інтерфейс користувача на будь-якому пристрої.

1.3.2 Підтримка браузерів

Система повинна працювати для наступних браузерів останніх версій: Mozilla Firefox; Google Chrome; Safari; Opera.

1.3.3 Вимоги до продуктивності

Система повинна відображати будь-яку сторінку не довше, ніж за 2 секунди.

Система повинна відправляти повідомлення не довше, ніж 2 секунди.

1.3.4 Вимоги до безпеки

Система не повинна дозволяти вводити у поля дані, які можуть бути використані, як експлойти.

Система не повинна надавати доступ неавторизованим користувачам доступ до даних системи.

1.4 Опис предметної області

Предметною областю є розробка інтернет-магазину продажу одягу. Він повинен надавати користувачам можливість взаємодіяти з товарами електронного магазину. Процес взаємодії з системою проходить наступним чином. Користувач, повинен ввести адресу сайту в браузері, система відобразить головну сторінку.

Після переходу на сайт користувач має можливість взаємодіяти з такими розділами як: карточка товару, категорії, кошик.

Процес створення нового замовлення проходить наступним чином. Користувач натискає на кнопку «Додати до кошика під карткою товару, система додає товар до кошика. Користувач натискає на піктограму кошика та на сторінці, що відобразила система, вказує кількість товару. Після введення даних користувач натискає кнопку «Оформити замовлення та переходить до оплати, система зберігає замовлення до БД.

1.5 Опис системи

У сучасному цифровому світі електронна комерція набуває все більшої популярності. Інтернет-магазини стають невід’ємною частиною ринку роздрібної торгівлі, забезпечуючи зручний спосіб покупок для мільйонів людей по всьому світу. Однією з найпопулярніших категорій товарів в електронній комерції є одяг. Розробка інтернет-магазину одягу є важливим кроком для будь-якого бренду або підприємства, що спеціалізується на продажу модного одягу.

Розробка інтернет-магазину одягу має безліч переваг, які привертають як підприємців, так і покупців.

Глобальний ринок: інтернет дає можливість працювати на глобальному ринку. Ваш магазин буде доступний для покупців з усього світу, незалежно від їх місцезнаходження. Це розширює потенційну аудиторію і збільшує можливості

збуту.

Зручність для покупців: інтернет-магазини пропонують зручний спосіб придбання товарів. Покупці можуть замовляти одяг в будь-який зручний для них час і місце, не виходячи з дому. Це особливо зручно для людей з обмеженими можливостями або тих, хто має напружений графік роботи.

Низькі витрати: запуск інтернет-магазину вимагає значно менших витрат, ніж відкриття фізичного магазину. Ви не потребуєте фізичного приміщення, дорогого обладнання або великого персоналу. Це дозволяє зосередитися на розвитку продукту та маркетингу.

Маркетингові можливості: цифрові канали маркетингу, такі як соціальні медіа і рекламні платформи, надають безліч можливостей для просування вашого інтернет-магазину. Ви можете розробити цільовану рекламу, створити привабливий контент і налагодити взаємодію зі своїми клієнтами.

Розглянемо особливості розробки інтернет-магазину одягу.

Вибір платформи: існує безліч платформ для створення інтернет-магазину, таких як Shopify, WooCommerce, Magento тощо. Важливо вибрати платформу, яка відповідає вашим потребам і має зручний інтерфейс для управління товарами, замовленнями і оплатою.

Дизайн інтерфейсу: ваш інтернет-магазин повинен мати привабливий інтерфейс, що привертає увагу клієнтів і створює позитивне враження. Врахуйте брендову ідентичність, зручну навігацію, чітку презентацію товарів і зручність оформлення замовлення.

Функціональність інтернет-магазину: розробка функціональних можливостей, таких як пошук товарів, фільтри, рейтинги, відгуки покупців, розширені опції доставки і оплати, допоможе покращити користувацький досвід і зробить процес покупок зручним.

Інтеграція платіжних систем: для зручності покупців і покращення конверсії важливо інтегрувати різні платіжні системи, такі як PayPal, Stripe, карткові платежі і т.д. Це забезпечить безпечну оплату та збільшить довіру до вашого магазину.

Оптимізація для пошукових систем: правильна оптимізація вашого інтернет-магазину для пошукових систем (SEO) допоможе покупцям знайти ваші товари в пошукових запитах. Використовуйте ключові слова, оптимізуйте метатеги, створюйте унікальний контент для товарів і піклуйтесь про швидкість завантаження сторінок.

Аналітика і відстеження: встановіть систему аналітики, наприклад, Google Analytics, щоб отримувати важливу інформацію про трафік, конверсію та поведінку покупців. Аналізуйте ці дані і вносьте відповідні зміни для покращення ефективності вашого магазину.

Маркетинг і просування: розробіть стратегію маркетингу, використовуйте соціальні медіа, контентний маркетинг, електронну пошту та інші канали для залучення нових клієнтів і збереження старих. Пропонуйте привабливі акції, знижки та програми лояльності, щоб стимулювати покупки.

1.6 Особливості розробки інтернет-магазинів

1.6.1 Переваги

Найбільш очевидною перевагою електронної торгівлі є можливість "проходжуватися" по магазинах в Інтернет, не виходячи при цьому з будинку.

Які ж мотиви рухають людьми що визначають можливість зробити замовлення в магазині, не виходячи з будинку?

Економія часу: у Інтернет можна вибирати і порівнювати характеристики товарів серед декількох магазинів, і дана процедура займає хвилини на відміну від утомливих походів по традиційних магазинах. Покупка необхідного товару через Інтернет економить такий вельми дорогий час. Електронний магазин доступний 24 години в добу, 7 днів в тиждень. Працює без свят і вихідних, черг там немає. Покупець може у будь-який час відірватися від вибору товару, якщо в даний момент у нього немає часу, і повернутися до вибору навіть через декілька

діб (всі дані по вибраних позиціях залишаються в аккаунті користувача). Продовжити вибір можна у вільний час - на робочому місці або удома, увечері або вночі.

Низькі ціни: більшість товарів в мережі Інтернет можна придбати за нижчу ціну, ніж в звичайному супермаркеті. Чим це обумовлено? По-перше, у електронному магазині немає необхідності орендувати торгові площі, по-друге, відсутні витрати по охороні і змісту торгових залів, немає потреби витрачати гроші на найм і навчання продавців-консультантів, по-третє, не потрібно оплачувати послуги посередників.

Безкоштовна доставка (в більшості випадків): після оформлення замовлення і передачі його в службу доставки, кур'єр безкоштовно привозить покупку додому або в офіс, в найкращий час. У покупців електронних магазинів «не болить голова» з приводу отримання замовлення, їх не обтяжує сама думка про перетягання важких сумок від магазину до будинку.

Свобода вибору: у відвідувачів інтернет – магазинів існує повна, нічим не обмежена свобода вибору. Сформована заявка поступає не на склад рядового роздрібного магазину, а на найбільший оптовий склад, де є величезний вибір товарів. У електронному магазині можна ознайомитися з інструкціями по застосуванню і подивитися фотографії товарів, не покидаючи улюбленого крісла перед монітором комп'ютера. Звичайно, не можна ознайомитися з товаром «живцем», відчути його руками, але є можливість сходити в звичайний магазин, подивитися на уподобаний товар, ще раз зважити все «за і проти», й істотно заощадити значну суму грошей при покупці через електронний магазин.

Психологічний комфорт: відвідувачі електронних магазинів не штовхаються в натовпі інших покупців, з нетерпінням чекаючи, коли підійде вільний менеджер. Там ніхто не докучає настирливими порадами і не примушує купити непотрібну річ. Відсутній будь-який тиск з боку продавця. Консультації з будь-якого питання, здійснюються через службу online підтримки: «Бажаєте замовити новинку, що ще не поступила на склад магазину? Залиште заявку і вас проінформують про надходження по електронній пошті».

Зручність оплати: клієнти можуть обрати найоптимальніший спосіб оплати вибраного товару: готівкою при доставці кур'єром, поштовий або банківський переказ, оплата кредитною картою або електронними грошима (Webmoney, та ін.). Вони можуть відмовитися від покупки, навіть не оплачуючи витрати при доставці.

Анонімність: про покупки відвідувачів електронних магазинів ніхто не дізнається, якщо тільки вони самі не захочуть поділитися цією інформацією. Тільки електронний магазин гарантує повну анонімність покупця, адже реєструватися можна під будь-яким ім'ям.

1.6.2 Недоліки

Найбільш важливими перешкодами здійснення покупки в електронному магазині виявився не достатньо добре налагоджений процес продажу товарів, куди входить:

Недосконала система доставки: покупці вимагають від електронного магазину швидкої і якісної доставки замовлень. Покупцям дуже зручно, коли покупки доставляють за вказаною адресою прямо на будинок або на роботу. Недосконаліть системи доставки може виражатися в не дотриманні терміну доставки товару або дорогій доставці товару. Оперативність служби доставки багато в чому визначає «лице» магазину. Але поки, по визнанню співробітників електронних магазинів, доставка товарів є одним з самих слабких місць компаній.

Незручна система оплати: виявилось, що не раз відмовлялися від здійснення покупки тільки тому, що електронний магазин не пропонував зручної для них форми оплати. Найпоширенішою формою оплати є оплата кур'єрові готівкою. Він же наголошується як найбезпечніший спосіб. Найбільш економним способом оплати можна віднести передоплату банківським переказом.

Складна система замовлення: складна система замовлення полягає в довгому і заплутаному процесі оформлення покупки. Покупцеві необхідно заповнити довгу і часто не зовсім зрозумілою форму, внаслідок чого витрачається багато часу в Інтернет, який не завжди дешево стоїть. Також можна відзначити повільну швидкість завантаження деяких магазинів, що може послужити причиною виходу з магазину.

Нестабільний асортимент: відповідно, отримати конкурентну перевагу зможе той магазин, який запропонує користувачам весь необхідний ним асортимент. Крім того, нестабільний асортимент деяких магазинів, тобто коли після вибору товару і оформлення покупки виявлялося, що потрібного товару немає на складі або просто магазин так і не зв'язався з покупцем.

Необхідність реєстрації: деякі магазини примушують покупця проходити дуже довгий покроковий процес реєстрації. Це відлякує потенційних покупців здійснювати покупки в даному магазині, особливо коли покупець цінує час, що витрачається ним.

Проблема віртуальності: неможливість відчути і потримати товар, що купується (на картинці він може виглядати відмінно, але в реальності зовсім інакше).

Невиконання вибору замовленн:. можуть переплутати колір товару, модель, вигляд і так далі

Втрата замовлення: рідко, але буває, замовлення взагалі втрачаються.

1.6.3 Інструменти розробки

Розробка інтернет-магазину включає в себе розробку клієнтської частини сайту, тобто створення дизайну, та розробку серверної частини сайту, а саме функціоналу. Далі розглянемо детальніше кожну зі сторін.

Для реалізації зовнішнього інтерфейсу веб-сайту застосовується мова гіпертекстової розмітки HTML, таблиці каскадних стилів CSS та мова

програмування JavaScript. Але розробка дизайну лише на цих трьох складових не закінчується, адже зараз небувалого поширення набувають фреймворки. Розглянемо найпоширеніші front-end фрейморки.

Bootstrap: авторами фреймворка є творці Twitter, які випустили його в 2011 році. Самий використовуваний open-source фреймворк в світі. Як і в будь-який інший ефективний front-end фреймворк, в Bootstrap входять компоненти HTML, CSS і JS. Фреймворк дотримується стандартів адаптивного веб-дизайну, дозволяючи вам створювати адаптивні сайти будь-якої складності і розмірів. Підходить для новачків і тих, хто вважає за краще надійні front-end фреймворки.

Semantic-UI: новачок серед фреймворків, що виділяється серед інших, і у якого є всі шанси стати найпопулярнішим front-end фреймворком. Фреймворк цінують за його простоту. У коді використовуються зрозумілі вирази. Тут відчувати себе як вдома будуть навіть ті, хто майже не знайомий з програмуванням. Ще одна помітна особливість Semantic-UI в тому, що він інтегрується з величезною кількістю сторонніх бібліотек. Їх стільки, що вам, швидше за все, не знадобляться якісь інші бібліотеки. Все це робить процес розробки простіше і зручніше. Підходить новачкам і тим, кому потрібен легкий і спритний фреймворк.

Foundation: від компанії Zurb - це високо просунутий front-end фреймворк корпоративного класу, який ідеально підходить для створення гнучких і адаптивних веб-сайтів. Фреймворк використовується на сайтах Facebook, eBay і Mozilla і через свою складність може не підійти новачкам.

Materialize: адаптивний front-end фреймворк, який відповідає специфікації Google по дизайну material design. У комплекті ви знайдете готові набори кнопок, іконок, карток, форм та інших компонентів. Є стандартна версія і версія під керуванням Sass. У Materialize є зручна сітка, за допомогою якої можна створювати макети. Також в стандартному комплекті вже присутні стилі матеріал дизайну для тіней, шрифтів, кольорів і т.д.

Крім того, в порівнянні front-end фреймворків на Google Trends видно (рис. 1.1), що Bootstrap все ще на голову випереджає своїх конкурентів за

кількістю згадок.

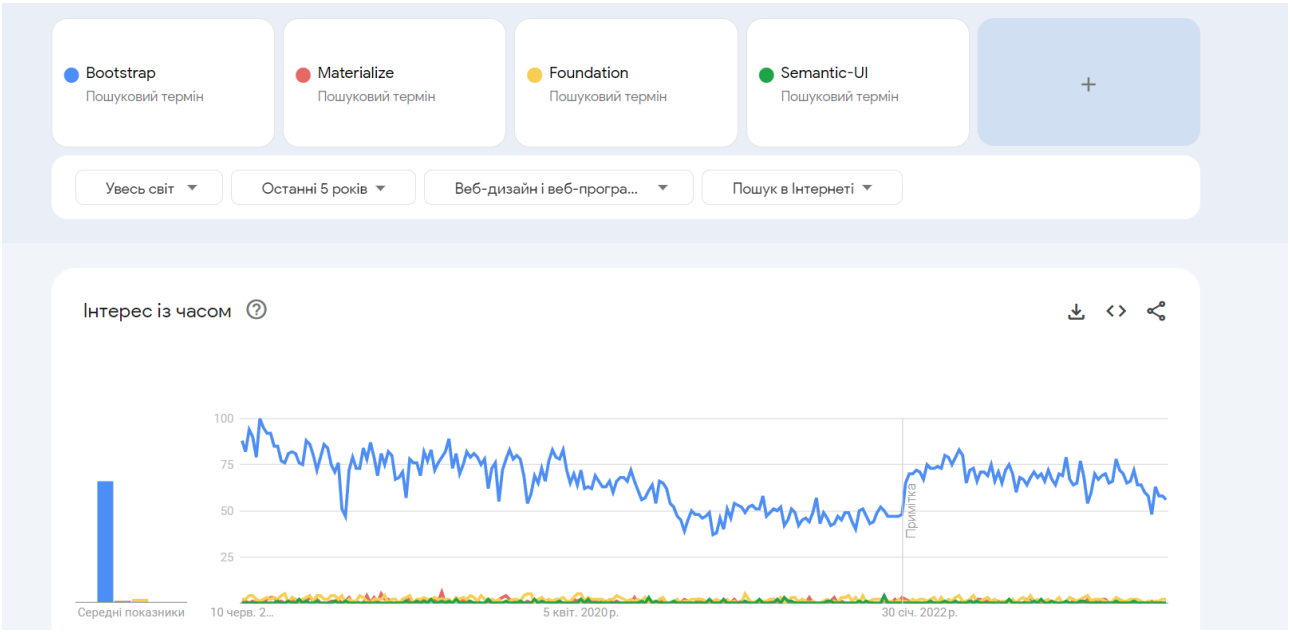

Рисунок 1.1 – Статистика front-end фреймворків

Сучасні динамічні сайти обов’язково використовують базу даних як для зберігання різного контенту, так і для реалізації своїх функцій. Найбільш поширеними системами управління базами даних є MySQL, PostgreSQL, ORACLE. Ці бази даних є полегшеними, тобто мають урізаний функціонал, достатній для задач веб розробки. Найбільш поширеною є MySQL. Дана база досить проста в використанні і має високу швидкість обробки запитів, що важливо при великій кількості звернень від користувачів до сервера, на якому розташовується сайт.

Серед серверних мов можна виділити Ruby, ASP, JSP, Python, Perl, але найбільш поширеним (більше 5 млн. серверів) є мова PHP версії 5. Особливістю даного мови, саме в 5 версії, є повна підтримка об’єктно-орієнтованої технології програмування, що спрощує роботу з повторного використання коду, покращує читаність коду і дозволяє працювати над створенням сайту команді розробників. Даною мовою і для даної мови написана досить велика кількість фреймворків (каркасів розробки структурних і функціональних елементів сайту). Серед

найбільш поширених фреймворків можна виділити Laravel, Symphony, Zend, Yii2, CakePHP, Code Igniter та інші. Серед цих фреймворків існують універсальні і спеціалізовані (Призначені для реалізації тільки деяких функцій і завдань веб розробки). Використання фреймворків дозволяє значно прискорити процес розробки, зробити код більш стандартним і зрозумілим [12].

Розглянемо фрейморки більш детально.

Laravel: один з найпопулярніших PHP-фреймворків, що володіє виразним і елегантним синтаксисом. Він дозволить максимально спростити вирішення основних і найболючіших завдань, таких як аутентифікація, маршрутизація, сесії і кешування. Laravel створювався як спроба об'єднати тільки все найкраще, що є в інших PHP-фреймворк, а також Ruby on Rails, ASP.NET MVC і Sinatra. Одне з найважливіших його переваг – наявність інтегрованої системи модульного тестування.

Symfony: фреймворк, що складається з безлічі компонентів і написаний на PHP5 фреймворк, який використовує патерн Model-View-Controller. Пропонує швидку розробку і управління веб-додатками, дозволяє легко вирішувати рутинні завдання веб-програміста. Одне з основних його переваг – підтримка безлічі баз даних (MySQL, PostgreSQL, SQLite або будь-яка інша PDO-сумісна СУБД).

Zend: написаний повністю в об'єктно-орієнтованому стилі фреймворк, який використовує всі останні нововведення PHP. Розроблено з мінімальними залежностями від інших компонентів, кожен з яких можна використовувати окремо; тим не менш, стандартний набір бібліотек робить його дуже потужним і легко розширюваним засобом розробки. Крім того, він пропонує надійну і високопродуктивну реалізацію MVC і зручну у використанні абстракцію бази даних, а також безліч інших можливостей, які в сумі роблять його одним з найбільш функціональних фреймворків.

Yii:. високопродуктивний PHP-фреймворк, який використовує патерн MVC і призначений для швидкої розробки сучасних web-додатків. Його

можливості дозволяють в стислі терміни реалізовувати великомасштабні проєкти типу форумів, порталів, CMS, RESTful web-сервісів і інших складних систем. У нього вже закладено безліч перевірених і готових до використання рішень: конструктор запитів, ActiveRecord для реляційних і NoSQL баз даних, RESTful API, багаторівнева підтримка кешування і багато інших.

CakePHP: написаний на PHP програмний каркас для створення веб-додатків, що володіє активною і швидкозростаючою спільнотою. Як і більшість інших фреймворків, реалізує патерн MVC. Спочатку він створювався як клон популярного Ruby on Rails, і багато його ідеї були запозичені саме звідти. Від своїх конкурентів відрізняється тим, що підтримує не тільки PHP5, але і PHP4.

Code Igniter: мабуть, найбільш простий в освоєнні і використанні фреймворк. Легко розширюється, безпечний і використовує прості і зрозумілі підходи, одним словом – ідеальний для новачків. Одна з інших його переваг - швидкість роботи, цей фреймворк куди швидше справляється із завданням роботи з БД, ніж інші його побратими.

В порівнянні back-end фреймворків на Google Trends бачимо (рис. 1.2), що найпоширенішими є Laravel, інші фреймворки знаходяться більш-менш на одному рівні за кількістю запитів.

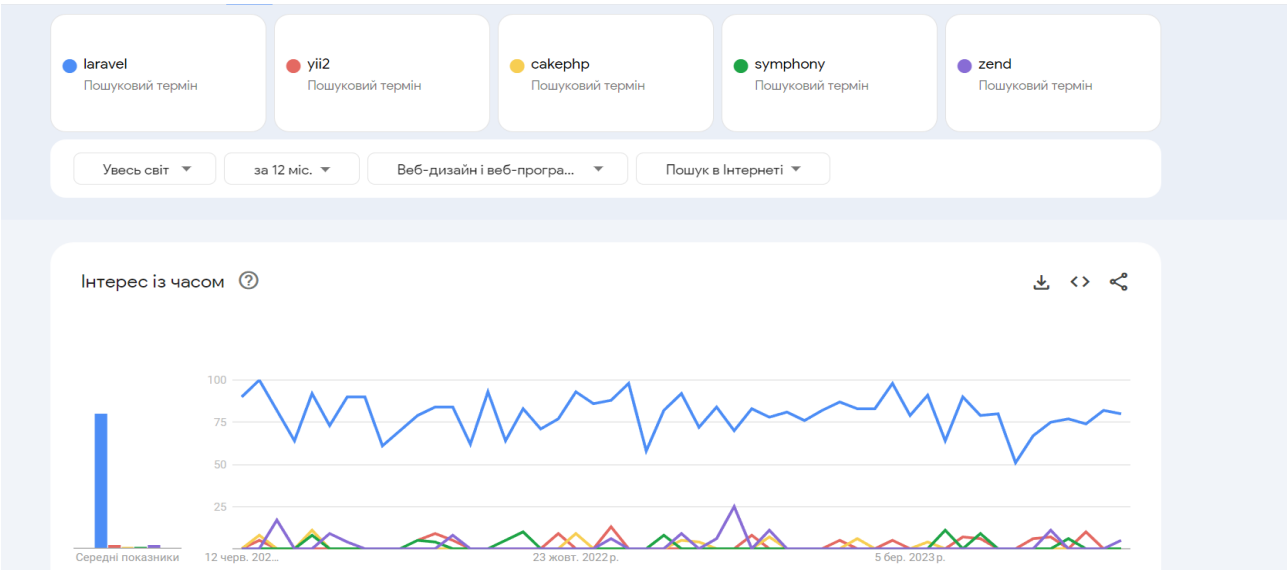

Рисунок 1.2 – Статистика back-end фреймворків

2 ПРОЄКТУВАННЯ

2.1 Проєктування бази даних

Процес проєктування бази даних є ітераційним – допускається повернення до попередніх етапів для перегляду раніше прийнятих рішень і включає наступні етапи [7]:

1) виділення сутностей і зв’язків між ними;
2) побудова діаграм ER-типу з урахуванням всіх сутностей і їх зв’язків;
3) формування набору попередніх відносин із зазначенням передбачуваного первинний ключ для кожного відношення і використання діаграм ER-типу;
4) додавання не ключових атрибутів у відносинах;
5) приведення попередніх відносин до нормальної форми;
6) вдосконалення ER-діаграми.

Перший етап проєктування – виділення сутностей і зв’язків між ними.

Виділимо наступні сутності:

- User(id, username…) – надалі сутність «Користувач»;
- Product(id, title…) – надалі сутність «Товари»;
- Image(id, image…) – надалі сутність «Зображення»;
- Category(id, title…) – надалі сутність «Категорії» ;
- Order(id…) – надалі сутність «Замовник»;
- Order_item(id, title…) – надалі сутність «Замовлення».

Виділимо зв’язки між сутностями:

- «Товари» поділяються на «Категорії»;
- «Товари» мають «Зображення»;
- «Замовлення» складається з «Товари» та «Замовник».

Другий етап проєктування – побудова діаграми ER-типу з урахуванням

всіх сутностей і зв’язків між ними (рис. 2.1).

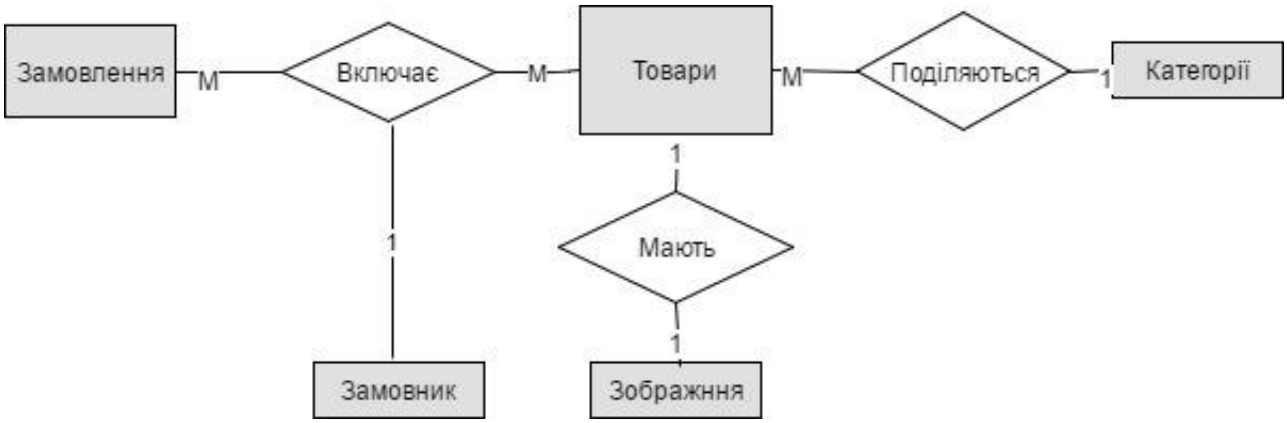

Рисунок 2.1 – Діаграма ER-типу

Зв’язок «Поділяються» є зв’язком типу «один до багатьох», так як однакову категорію можуть мати різні товари. Сутність «Товари» має обов’язковий клас належності, оскільки кожен товар має свою категорію. Сутність «Категорія» має необов’язковий клас належності, так як можливі такі значення категорії, котрі не має жоден з товарів.

Зв’язок «Мають» є зв’язком типу «один до одного», так як кожне зображення може використатися тільки один раз для одного товару. Обидві сутності в даному зв’язку мають обов’язковий клас належності, в припущені, що немає товарів без зображення, та відсутні зображення, які не відносяться до товару.

Зв’язок «Включає» є зв’язком типу «багато до багатьох» , так як замовлення може включати декілька товарів, а кожен товар входити до різних замовлень.

Зв’язок «Включає» є зв’язком типу «один до багатьох», так як однакового замовника можуть мати різні замовлення.

Всі сутності в зв’язку «Включає» мають обов’язковий клас належності, в припущені, що немає замовлень без товарів та замовників, та немає замовників і товарів які не входять до замовлення.

Третій етап проєктування – формування набору попередніх відносин із

зазначенням передбачуваного первинного ключа для кожного відношення, використовуючи діаграми ER-типу [4, 8].

На основі сформованих на другому етапі проєктування зв’язків визначаємо ключові атрибути:

- User(id, username…);
- Product(id, title, category_id…);
- Image(id, product_id, image…);
- Category(id, parent_id, title…);
- Order(id…);
- Order_item(id, order_id, title, product_id…).

Четвертий етап проєктування – додавання не ключових атрибутів, які не були вибрані в якості ключового раніше.

- User(id, username, password, email);
- Product(id, title, category_id, description, price);
- Image(id, product_id, image);
- Category(id, parent_id, title);
- Order(id, phone, address,email,note,status);
- Order_item(id, order_id, title, product_id, price, quantity).

П’ятий етап проєктування – перевірка на відповідність першій нормальній формі.

Відношення знаходиться в 1НФ, якщо всі його атрибути є простими (мають єдине значення). Початкове відношення будується таким чином, щоб воно було в 1НФ. В нашому випадку ми отримали ті ж відношення, що відповідають 1НФ [9].

Шостий етап проєктування – вдосконалення ER-діаграми згідно розробленої на попередніх етапах моделі (рис. 2.2.).

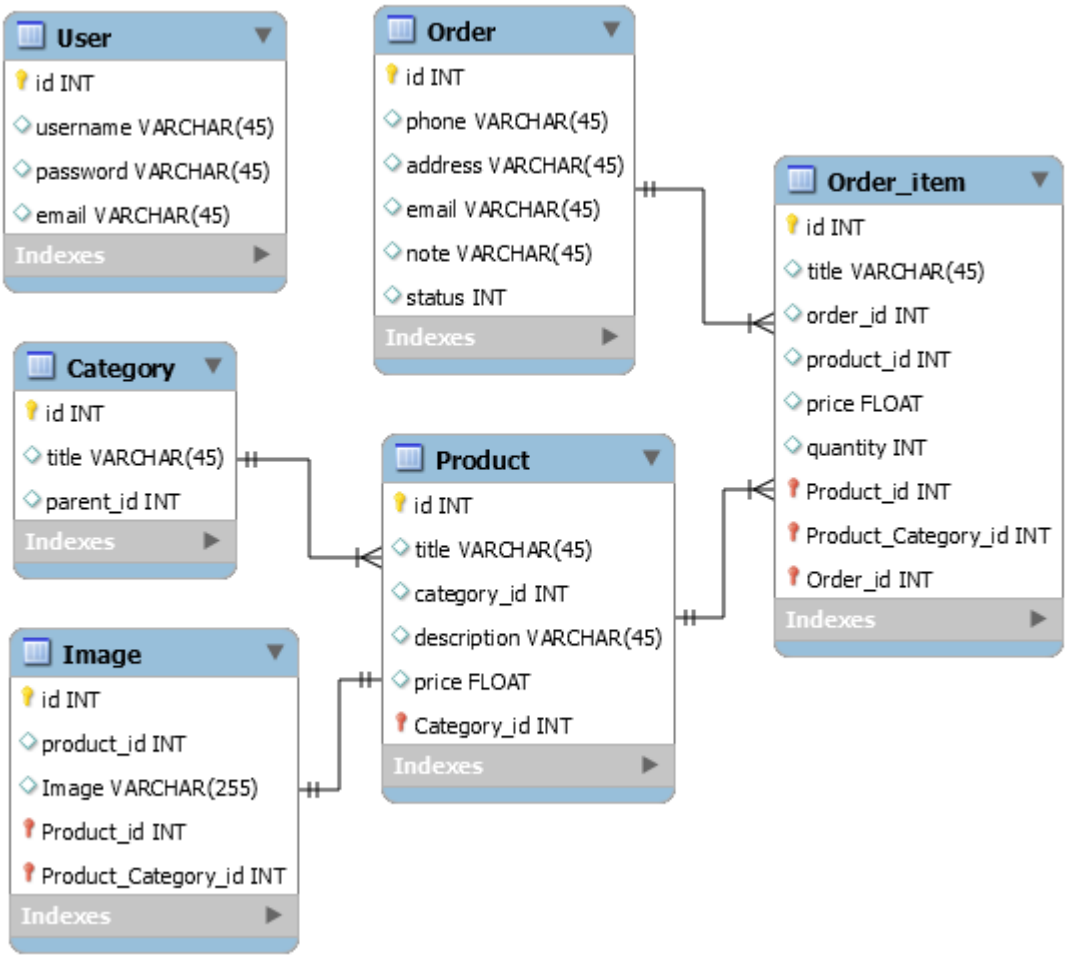

Рисунок 2.2 – Схема бази даних

2.2 Проєктування системи засобами UML

У сфері розробки програмного забезпечення проєктування системи відіграє ключову роль у створенні ефективних та функціональних програмних продуктів. Для розробки складних систем існує безліч методологій та інструментів, але одним з найпопулярніших є використання мови UML (Unified Modeling Language). UML надає стандартизовану нотацію та графічні елементи для моделювання системи, що сприяє зрозумінню, візуалізації та спілкуванню між розробниками, замовниками та іншими зацікавленими сторонами.

Визначення вимог: перший етап у проєктуванні системи – це розуміння вимог до неї. Вимоги можуть бути функціональними (що система повинна робити) та нефункціональними (якість, надійність, продуктивність тощо). UML

дозволяє моделювати вимоги за допомогою діаграм вимог, які допомагають визначити потреби користувачів та функціональні можливості системи.

Аналіз системи: на другому етапі проводиться детальний аналіз системи з метою визначення ключових об’єктів, компонентів та їх взаємозв’язків. UML-діаграми класів дозволяють моделювати структуру системи, включаючи класи, атрибути, методи та взаємини між ними. Крім того, можуть використовуватися діаграми об’єктів для більш детального представлення екземплярів класів та їх взаємодій.

Проєктування архітектури: на цьому етапі визначається архітектура системи – її основні компоненти, підсистеми, модулі та взаємозв’язки між ними. UML-діаграми компонентів та діаграми пакетів допомагають моделювати архітектурні аспекти системи. Діаграми компонентів відображають фізичну організацію системи, показуючи компоненти та їх залежності, а діаграми пакетів дозволяють групувати компоненти в логічні групи.

Проєктування поведінки: на цьому етапі визначається поведінка системи – як вона взаємодіє з користувачами та іншими системами. Для моделювання поведінки системи використовуються різні види UML-діаграм, такі як діаграми послідовності, діаграми стану, діаграми активності та інші. Наприклад, діаграма послідовності дозволяє відобразити послідовність операцій та взаємодію об’єктів у конкретному сценарії використання.

Проєктування інтерфейсу: якщо система має графічний інтерфейс користувача, то цей етап включає розробку інтерфейсу. UML-діаграми специфікації інтерфейсу допомагають визначити елементи інтерфейсу та їх взаємозв’язки з системою. Це може включати діаграми взаємодії, діаграми компонентів і діаграми стану для моделювання різних аспектів інтерфейсу.

Документація: не менш важливим етапом є створення документації проєкту, яка описує моделі UML, архітектуру системи, прийняті рішення проєктування та інше. Документація є важливим інструментом для збереження знань про систему та забезпечення спілкування між розробниками та іншими зацікавленими сторонами.

2.3 Діаграма варіантів використання

Візуальне моделювання в UML можна уявити як певний процес рівневого спуску від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування.

Розробка діаграми варіантів використання переслідує наступні цілі:

- визначити загальні межі і контекст модельованої предметної області на початкових етапах проєктування системи;
- сформулювати загальні вимоги до функціонального поведінки проєктованої системи;
- розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей;
- підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми складається в наступному: проєктована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на моделюєму систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.

Діаграма варіантів використання наведена на рисунку 2.3.

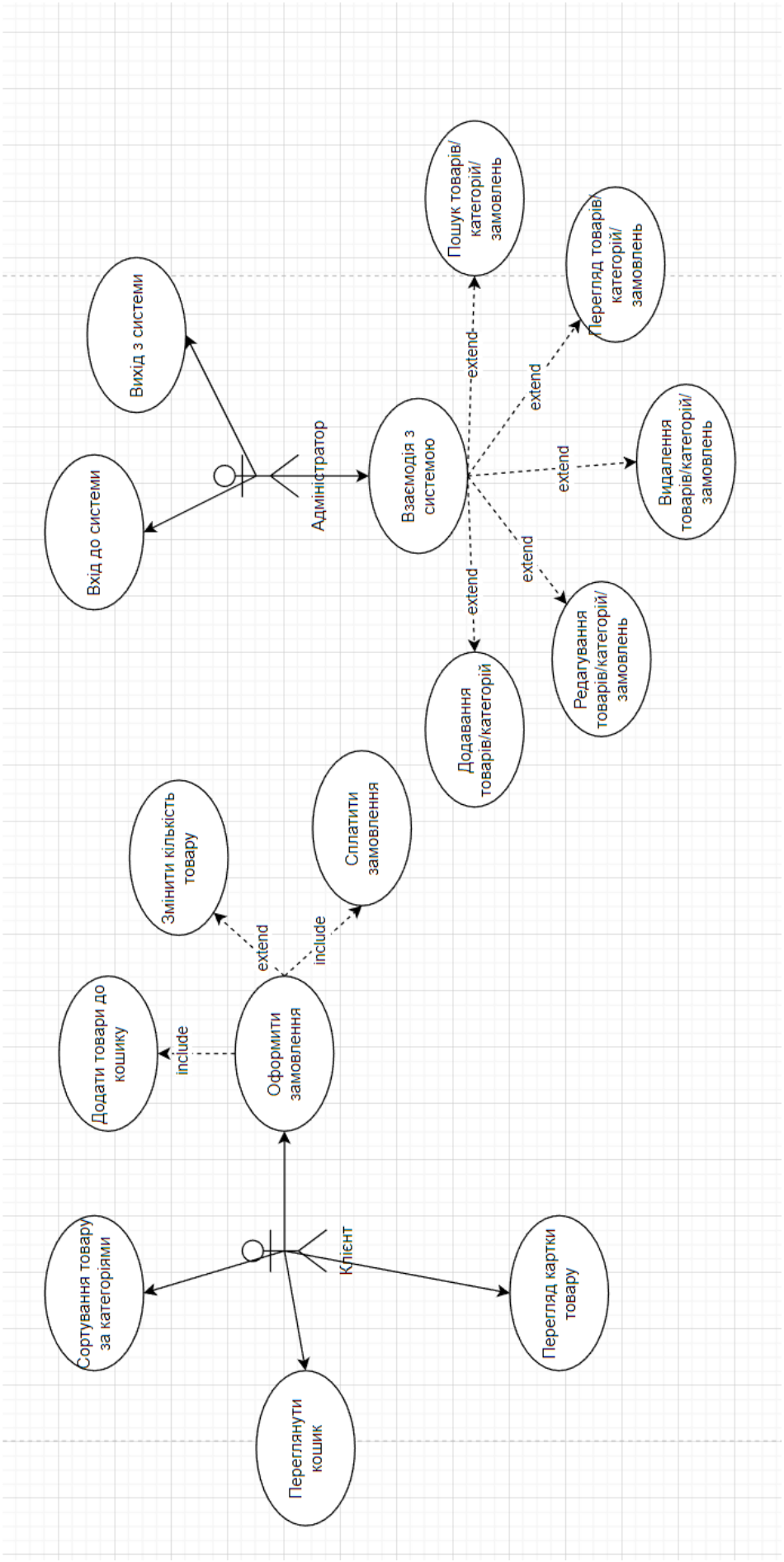

Рисунок 2.3 – Діаграма варіантів використання

2.4 Діаграма послідовності

На рисунку 2.4 зображено діаграму послідовності основного бізнес-процесу «Оформлення замовлення».

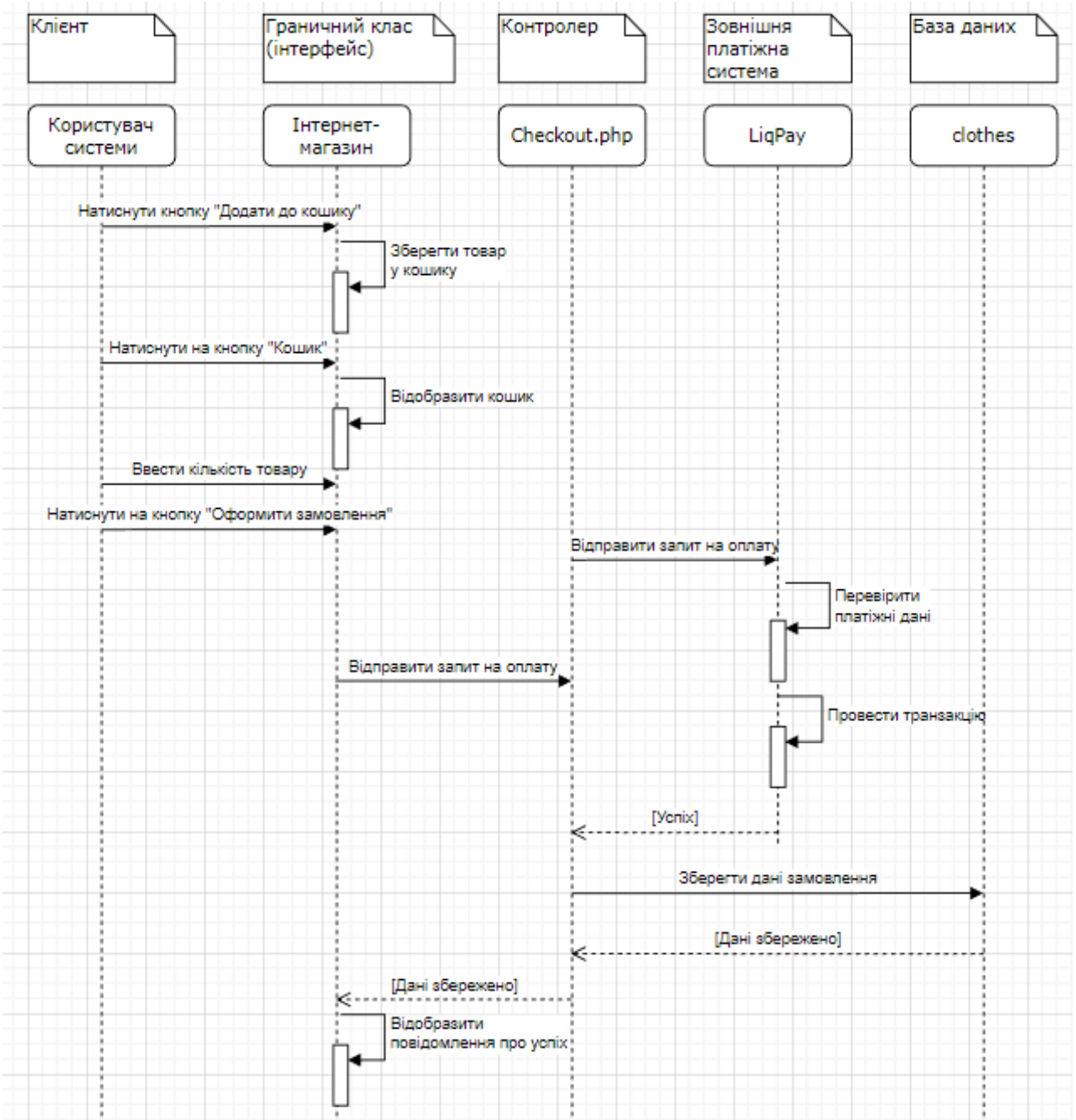

Рисунок 2.4 – Діаграма послідовності

Діаграма послідовності (Sequence Diagram) є одним з видів UML-діаграм і використовується для моделювання взаємодії між об’єктами або компонентами

системи впродовж певного часового інтервалу. Вона дозволяє візуалізувати послідовність викликів, повідомлень та обміну даними між об’єктами та компонентами системи під час виконання певного сценарію.

Основною метою діаграми послідовності є зрозуміння та візуалізація логіки взаємодії між об’єктами в рамках конкретної функціональності або сценарію використання. Вона допомагає ілюструвати порядок викликів методів, передачу повідомлень, обробку подій та інші аспекти взаємодії між об’єктами.

2.5 Діаграма класів

На рисунку 2.5 зображено діаграму класів системи.

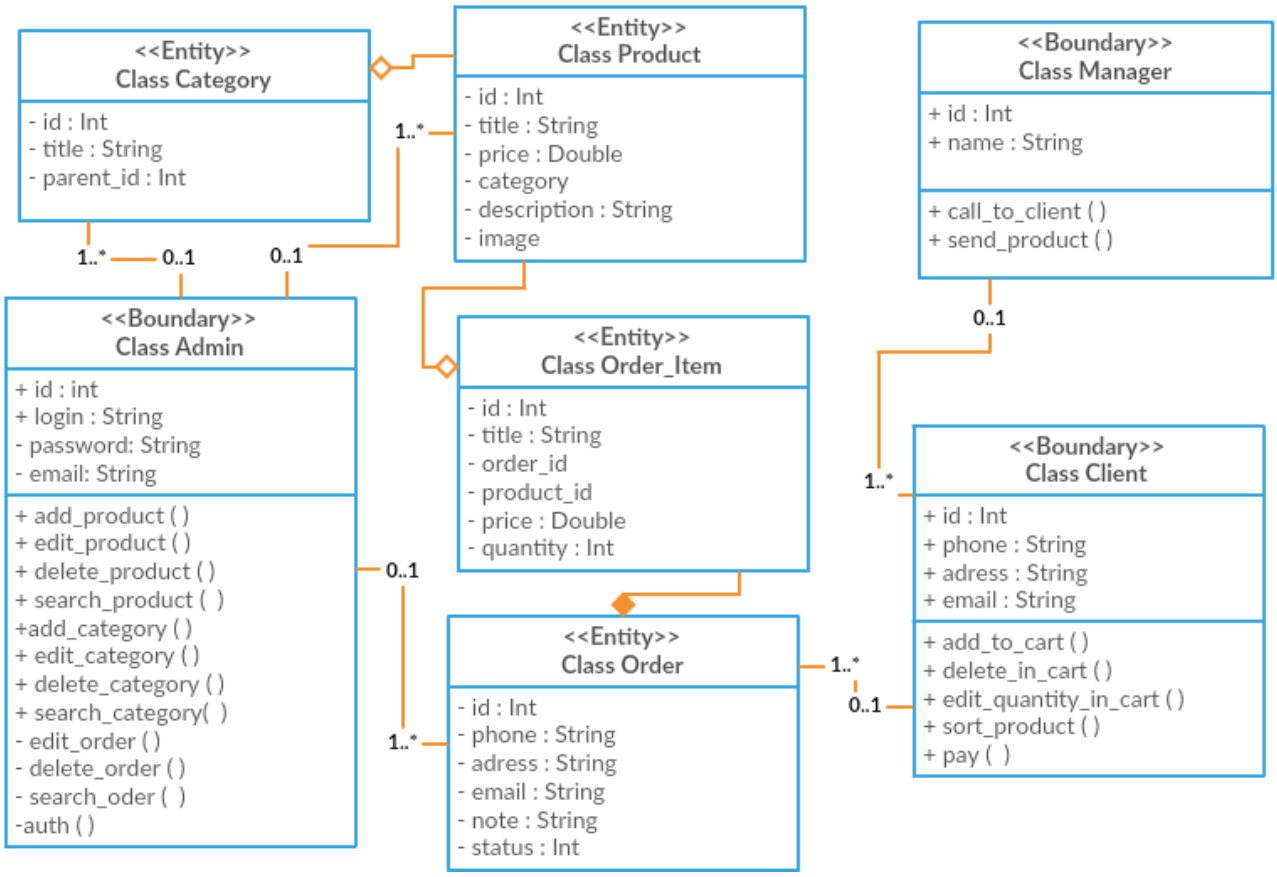

Рисунок 2.5 – Діаграма класів

Діаграма класів (Class Diagram) є одним з найбільш використовуваних видів UML-діаграм і використовується для моделювання структури системи шляхом відображення класів, їх атрибутів, методів та взаємозв'язків між ними. Діаграма класів надає загальний огляд об'єктно-орієнтованої системи та допомагає уявити, як класи взаємодіють один з одним.

Основна мета діаграми класів – візуалізувати структуру системи, включаючи класи, їх взаємозв'язки, атрибути та методи. Вона дозволяє розробникам та зацікавленим сторонам зрозуміти, як об'єкти системи взаємодіють та співпрацюють один з одним.

2.6 Діаграма розгортання

На рисунку 2.6 наведено діаграму розгортання системи.

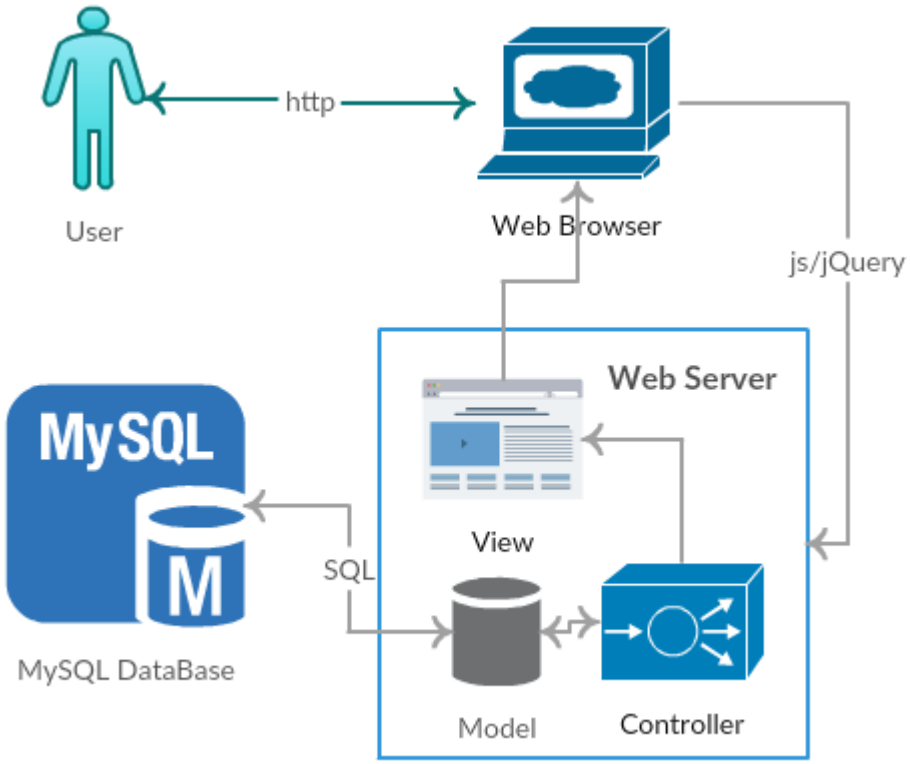

Рисунок 2.6 – Діаграма розгортання

Діаграма розгортання (Deployment Diagram) є одним з видів UML-діаграм і використовується для моделювання фізичного розташування компонентів системи та їх взаємозв’язків на апаратному та програмному забезпеченні. Діаграма розгортання дозволяє візуалізувати, як компоненти системи розподілені між фізичними вузлами та як вони комунікують між собою.

Основна мета діаграми розгортання – показати фізичне розташування компонентів системи, таких як сервери, комп’ютери, мережеві вузли або інші пристрої, а також з’єднання та комунікацію між ними. Вона допомагає розробникам та системним архітекторам уявити інфраструктуру, необхідну для роботи системи, та зрозуміти вимоги до апаратного забезпечення та середовища розгортання.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Вибір технологій розробки проєкту

Мова програмування: в якості серверної мови програмування було обрано скриптова мова програмування PHP 7.2.

База даних: в поєднанні з php7 найкращим «дуетом» визнано реляційну базу даних MySQL.

Фреймворк: згідно з теми проєкту та технічного завдання обов'язковою частиною розробку є використання Yii2 framework. Також буде застосовано фреймворк Bootstrap для реалізації розмітки сайту майбутнього дизайну.

Сервер: компіляція програмного коду сайту буде виконуватись в браузері Google Chrome на локальному веб-сервері Open Server для Windows.

Платіжна система: сплачення коштів за замовлення буде відбуватися за допомогою платіжної системи LiqPay, яка дозволяє приймати платежі і переказувати гроші за допомогою мобільного телефону, Інтернету і платіжних карт у всьому світі.

3.2 Встановлення Yii2 framework

Для встановлення Yii2 framework було використано наступні команди Composer [1, 3, 13]:

```
1   composer global require "fxp/composer-asset-plugin:^1.3.1"
2   composer create-project --prefer-dist yiisoft/yii2-app-advanced yii-application
```

Рисунок 3.1 – Команди для встановлення Yii2 framework

Далі за допомогою локального серверу запустимо встановлений фрейморк.

Результат роботи наведено на рисунку 3.2.

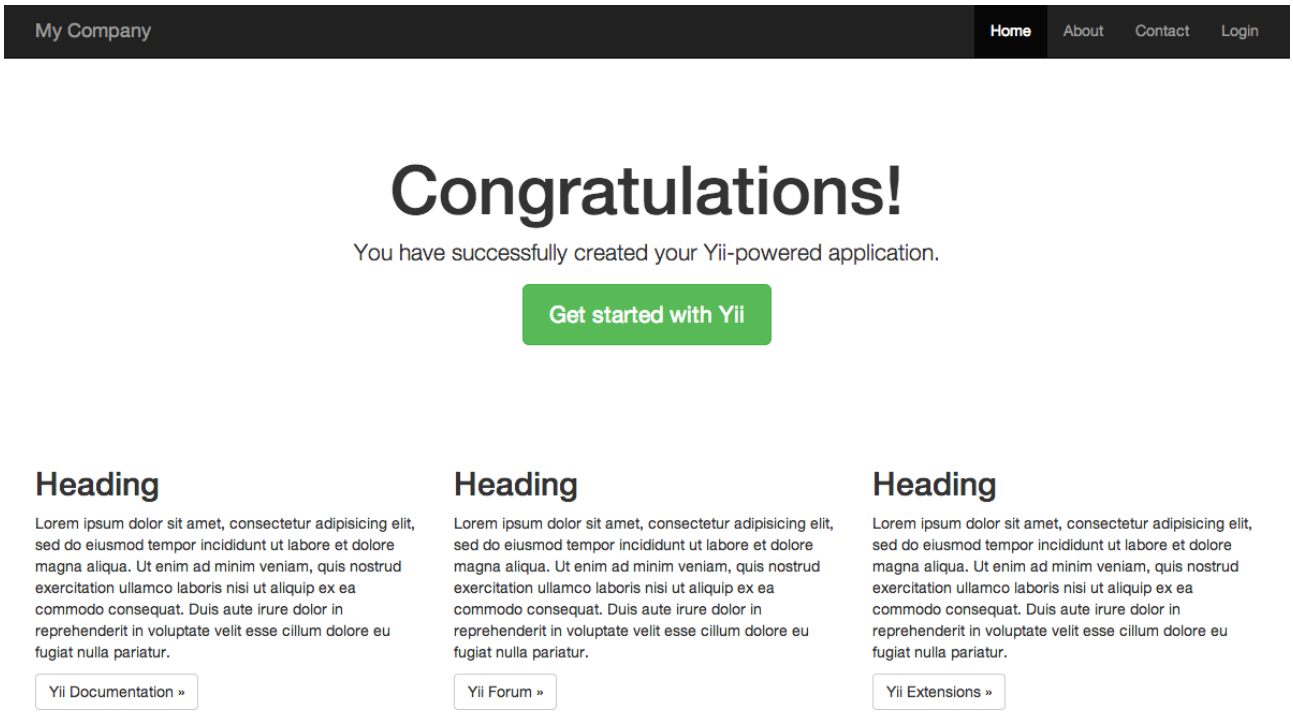

Рисунок 3.2 – Результат першого запуску Yii2 framework

3.3 Перенесення розмітки на Yii2

Шаблон майбутнього сайту було запозичено з сайту http://demo.themeum.com/html/eshopper/index.html. Завантажений код потрібно перенести на Yii2, для цього ми копіюємо теки з файлами стилів, java-скриптів, шрифти та зображення до нашого проєкту, а саме в директорію *\frontend\web*.

Наступним кроком буде перенесення розмітки до проєкту. Для цього потрібно скопіювати весь вміст файлу *index.html* з завантаженого шаблону, до файлу *frontend\views\layouts\main.php*.

Далі ми видозмінимо файл *main.php* таким чином:

– спростимо нашу розмітку, прибравши зайві частини розмітки, функціонал яких не буде далі розроблятися;

– підключення файлів реалізуємо в файлі

\frontend\assets\AppAssest.php;

– замінюємо динамічні частини коду на змінні Yii2.

Відредагований, згідно документації [6, 13], файл див. додаток А.

3.4 Створення та підключення бази даних

Згідно технічного завдання та спроєктованої бази даних у вигляді ER-моделі, sql-запит буде мати наступний вигляд (рис. 3.3):

```
CREATE DATABASE shop;
CREATE TABLE `category` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `parent_id` int(11) DEFAULT NULL,
  `
title` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
CREATE TABLE `image` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `product_id` int(11) DEFAULT NULL,
  `image` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
CREATE TABLE `order` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `phone` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `address` text COLLATE utf8_unicode_ci,
  `email` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `notes` text COLLATE utf8_unicode_ci,
  `status` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
CREATE TABLE `order_item` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `order_id` int(11) DEFAULT NULL,
  `title` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `price` decimal(19,2) DEFAULT NULL,
  `product_id` int(11) DEFAULT NULL,
  `quantity` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
CREATE TABLE `product` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `description` text COLLATE utf8_unicode_ci,
  `category_id` int(11) DEFAULT NULL,
  `price` decimal(19,2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
CREATE TABLE `user` (
```

Рисунок 3.3 – MySQL –запит

Далі необхідно підключити базу даних до проєкту [2, 8]. Для цього редагуємо файл *common\config\main-local.php* наступним чином (рис. 3.4):

```
'db' => [
        'class' => 'yii\db\Connection',
        'dsn' => 'mysql:host=localhost;dbname=shop',
        'username' => 'root',
        'password' => '',
        'charset' => 'utf8',
    ],

```

Рисунок 3.4 – Конфігурація для бази даних

Відповідно до таблиць баз даних створюємо моделі *common**models*\:

- Category.php;
- Image.php;
- Order.php;
- OrderItem.php;
- Product.php;
- User.php.

Полегшити процес створення моделей допоможе генератор моделей присутній в Yii2 [3, 5].

Детальніше зі змістом файлів можна ознайомитися в Додатку А.

3.5 Кошик товарів

Чи не найважливішою частиною будь-якого електронного магазину є кошик. Саме він вміщує в себе товари, які покупець обрав для замовлення. Тому до його розробки потрібно підійти правильно.

Розпочнемо реалізацію зі створення контролера *frontend**controllers**CartController.php*, який буде містити функцію додавання/оновлення та видалення товару з кошика, формування списку замовлення та оформлення замовлення. Повний текст файлу див. Додаток А.

Зовнішня реалізація кошику буде наведена в Додатку А, файли \frontend\views\cart*list.php* та \frontend\views\cart*order.php*. Де *list.php* виводить

у вигляді таблиці список замовлених товарів та дві кнопки «Оформити замовлення», яка перенаправляє на *order.php* для заповнення полів клієнтом та оформлення замовлення та «Сплатити», яка перенаправляє на платіжку систему LiqPay. Уривок програми з реалізацією платіжної системи в проєкті можна переглянути на рисунку 3.5.

```
$products = $cart->getPositions();
        $total = $cart->getCost();
        $order_id = uniqid();

        $public_key = 'i7175834200';
        $private_key = 'bfiKv7fKv1Kx7KigXf4bsqqqF7upW2iVtxjr5rfk';
        $liqpay = new LiqPay($public_key, $private_key);
        $pay = $liqpay->cnb_form(array(
            'version'        => '3',
            'sandbox'        => '1',
            'amount'         => $total,
            'currency'       => 'UAH',
            'description'    => 'Оплата тестового замовлення',
            'order_id'       => $order_id
        ));

        return $this->render('list', [
           'products' => $products,
           'total' => $total,
           'pay'=>$pay,
        ]);
```

Рисунок 3.5 – Реалізація LiqPay в проєкті

3.6 Сторінка адміністратора

Зовнішній вигляд сторінки адміністратора - це також шаблон з Інтернету, перенесений на Yii2 (див. рис. 3.6). Але важливішою є розробка функціоналу, а саме розробка контролерів:

- SiteController.php;
- ProductController.php;
- ImageController.php;
- CategoryController.php;
- ProductController.php;

– OrderItemController.php;

Де SiteController відповідає за надання доступу при авторизації адміністратора. А всі інші контролери реалізують головні функції для управління даними із бази даних, а саме: створення; видалення; перегляд; редагування; пошук.

Для демонстрації в додатку А наведено реалізацію контролера ProductController.php.

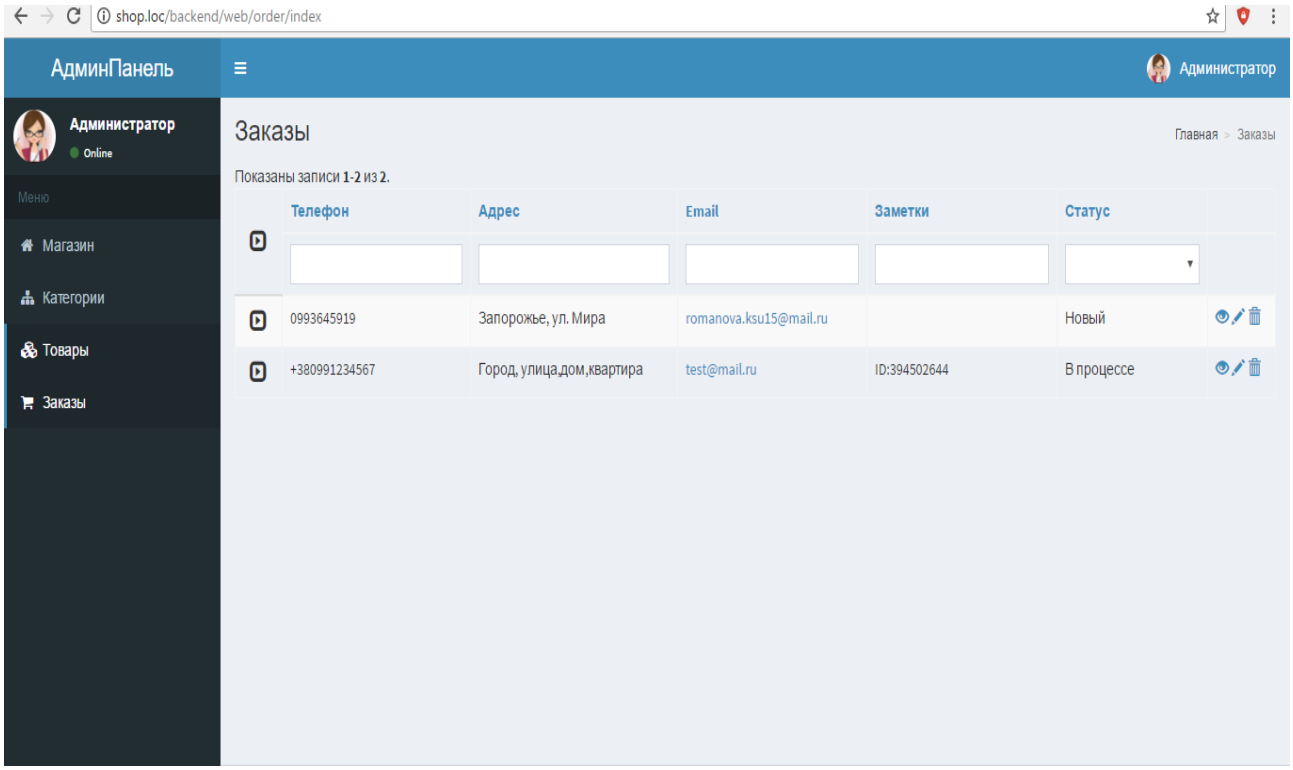

Рисунок 3.6 – Сторінка адміністратора: замовлення

3.7 Unit-тест

Unit-тести (або тести одиниць) є частиною процесу тестування програмного забезпечення, які перевіряють найменші частини коду – "одиниці" (функції, методи, класи) для переконання в їх правильному функціонуванні. У контексті контролера Yii2, unit-тести зазвичай перевіряють, чи працюють окремі

методи контроллера правильно та чи повертають очікувані результати.

Unit-тести допомагають забезпечити, що навіть найдрібніші частини програми працюють належним чином та що внесені зміни не порушують існуючу функціональність. Вони сприяють виявленню та усуненню помилок на ранніх етапах розробки та допомагають забезпечити стабільність та якість програмного забезпечення [10, 11].

На рисунку 3.7 наведено приклад тестування ProductController.

```
class ProductControllerTest extends \Codeception\Test\Unit
{
    /**
     * @var \UnitTester
     */
    protected $tester;

    protected function _before()
    {
        // Перед кожним тестом ініціалізуємо потрібні дані або середовище
    }

    protected function _after()
    {
        // Після кожного тесту очищуємо дані або відновлюємо стан
    }

    public function testActionIndex()
    {
        $controller = new ProductController('product', Yii::$app);
        $searchModel = new ProductSearch();

        $this->assertInstanceOf(ProductSearch::class, $searchModel);

        $controller->actionIndex();

        // Додати перевірки для відповідності очікуваному результату
    }

    public function testActionView()
    {
        $controller = new ProductController('product', Yii::$app);
        $id = 1;

```

Рисунок 3.7 – Тестування ProductController

3.8 Керівництво користувача

При переході на сайт, система відображає головну сторінку (рис. 3.8). З головної сторінки можна взаємодіяти з різними категоріями, кошиком, обновками, тощо.

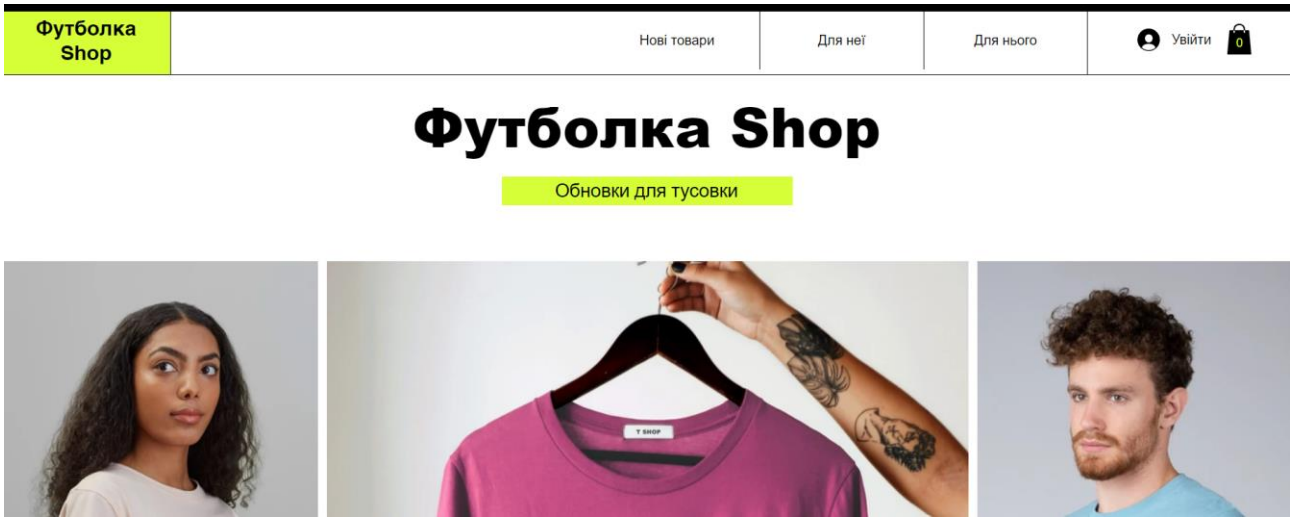

Рисунок 3.8 – Головна сторінка

Для перегляду товару за категоріями потрібно натиснути на один з трьох пунктів меню навігаційної панелі, система відобразить товари з обраної категорії (рис. 3.9).

Рисунок 3.9 – Категорія «Для нього»

Для детального ознайомлення з характеристиками та атрибутами, потрібно натиснути на зображення цього товару, система відобразить картку товару (рис. 3.10).

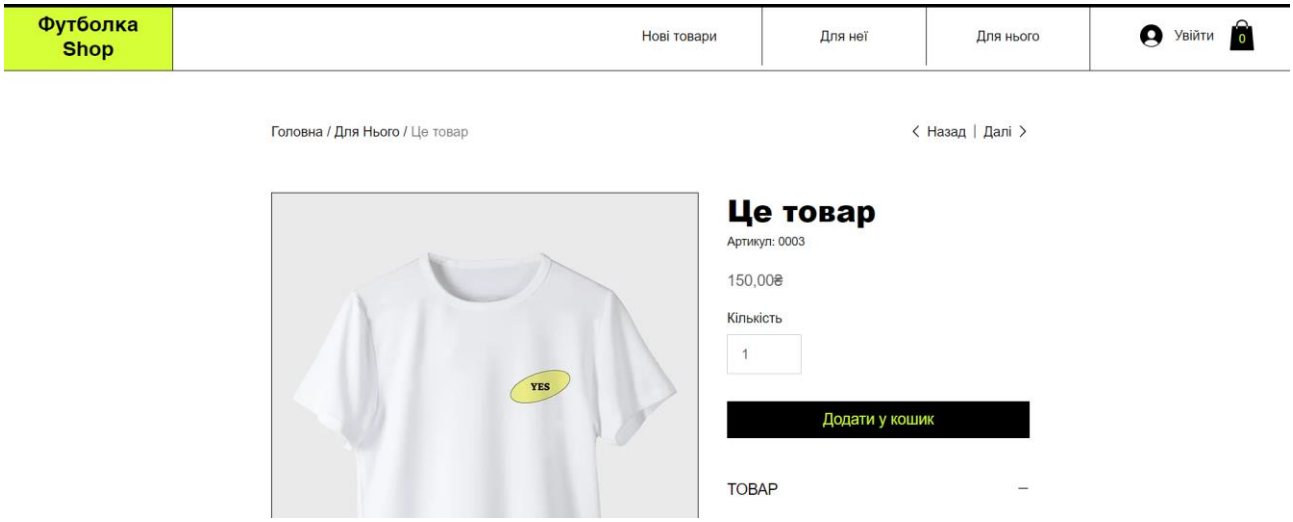

Рисунок 3.10 – Картка товару

У картці товару можна ознайомитись з описом, вказати кількість одиниць товару та ознайомитися з ціною. Для покупки товару необхідно натиснути на кнопку «Додати у кошик», система відобразить попередній перегляд товару. Для переходу до кошику, натискаємо кнопку «Переглянути» (рис. 3.11).

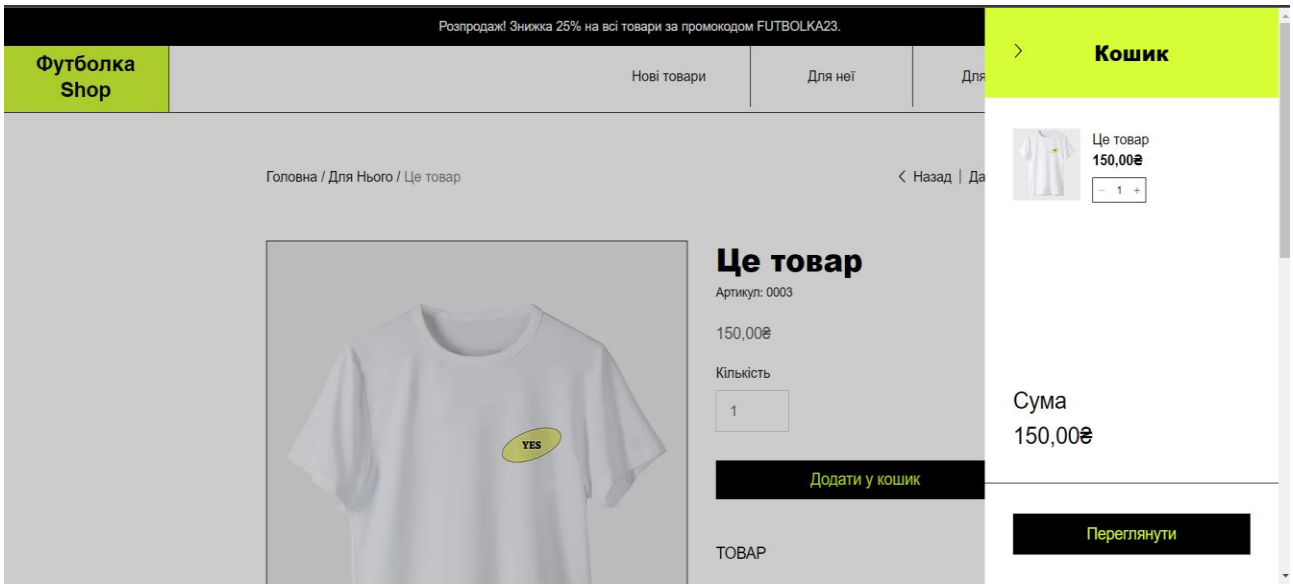

Рисунок 3.11 – Додавання товару до кошику

У кошику можна ознайомитись з найменуванням, кількістю та загальною ціною товару, також можна змінювати кількість товару не виходячи з кошика, система автоматично перераховує загальну суму. Для оформлення замовлення

необхідно натиснути на кнопку «Оформити» (див. рис. 3.12 – 3.13).

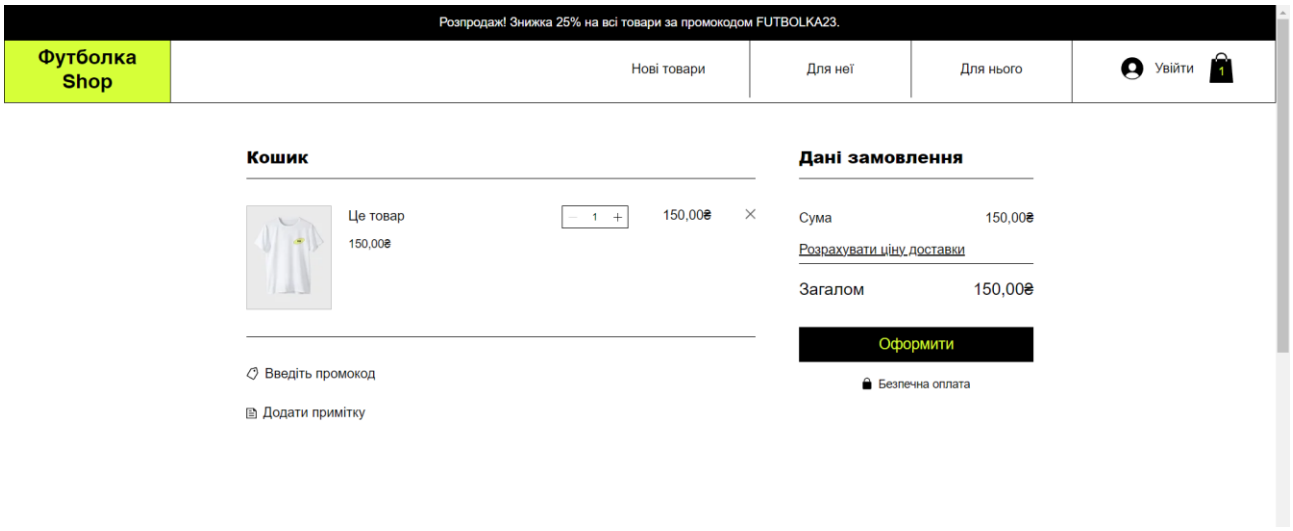

Рисунок 3.12 – Кошик

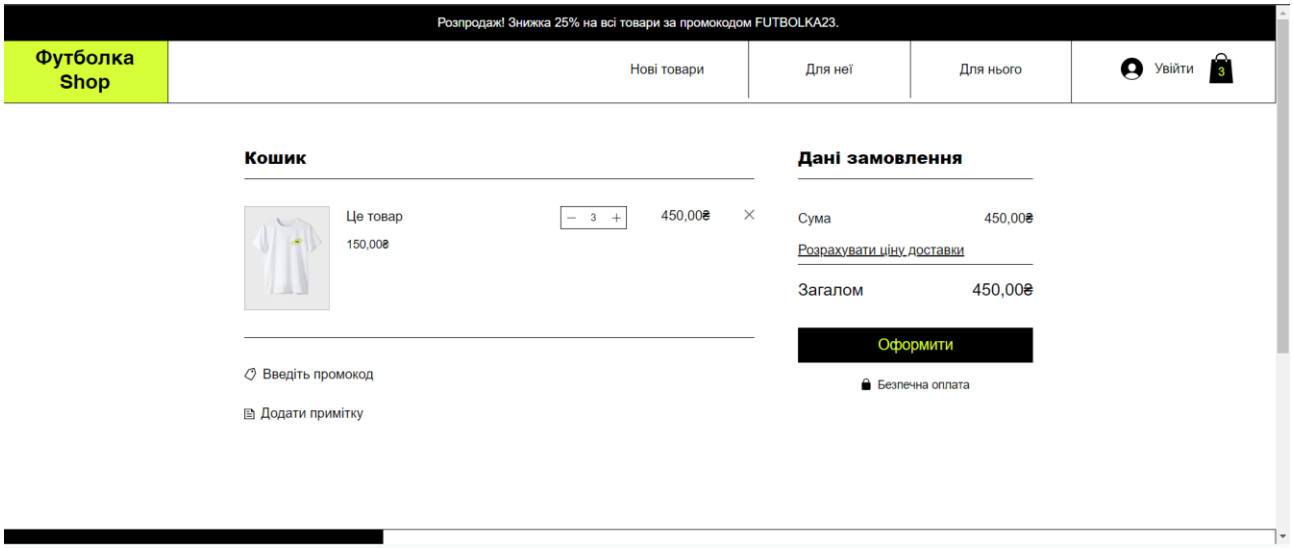

Рисунок 3.13 – Перерахунок суми

ВИСНОВКИ

В результаті роботи було написано технічне завдання на розробку інтернет-магазину одягу. Для створення цієї системи був обран фреймворк Yii2 зі сторони клієнта і сервера, за його широкі можливості у сфері створення web-систем.

У відповідності з метою кваліфікаційної роботи був розроблен інтернет-магазин одягу із застосуванням наступних технологій:

- Yii2 для реалізації клієнтської та серверної частини;
- MySQL для зберігання даних.

У відповідності з поставленими задачами були виконані наступні етапи створення системи:

- сформовані вимоги до системи (функціональні та нефункціональні (інтерфейс, кросбраузерність, безпека, продуктивність)), а також проведено огляд предметної області та інструментів розробки;
- спроєктована та побудована структура системи (побудовані діаграми прецедентів, діяльності та послідовності; надано детальний опис прецедентів);
- реалізовано інтернет-магазин одягу (наведена інструкція по створенню компонентів системи, надано керівництво користувача та структура проєкту);
- протестована робота системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Angstadt R. Yii 2 Development: Bring A Map Through The Halls Of Yii 2 Development. Independently published, 2023. 118 p.

2. Bierer D., Evans C. PHP 8 Programming Tips, Tricks and Best Practices: A practical guide to PHP 8 features, usage changes, and advanced programming techniques. Birmingham : Packt Publishing, 2021. 528 p.

3. Dicerbo A. Yii 2 Development: Bring A Map Through The Halls Of Yii 2 Development. Independently published, 2023. 236 p.

4. Duckett J. PHP & MySQL: Server-side Web Development. Wiley, 2022. 672 p.

5. Echevarria P. Yii 2 Speed: Getting Up To Speed With Yii 2. Independently published, 2023. 118 p.

6. Kenely E. Yii 2 Speed: Getting Up To Speed With Yii 2. Independently published, 2023. 236 p.

7. Murach J., Harris R. Murach’s PHP and MySQL. Fresno : Mike Murach & Associates, 2022. 848 p.

8. MySQL Developer Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 19.03.2023).

9. Nixon R. Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites (Learning PHP, MYSQL, Javascript, CSS & HTML5). O’Reilly Media, 2021. 823 p.

10. PHP Developer Documentation. URL: https://www.php.net/docs.php (дата звернення: 30.03.2023).

11. Windler C., Daubois A. Clean Code in PHP: Expert tips and best practices to write beautiful, human-friendly, and maintainable PHP. Birmingham : Packt Publishing, 2022. 264 p.

12. Yii2 Developer Documentation. URL: https://www.yiiframework.com/doc/guide/2.0/uk (дата звернення: 21.04.2023).

13. Zandstra M. PHP 8 Objects, Patterns, and Practice: Mastering OO Enhancements, Design Patterns, and Essential Development Tools. New York City : Apress, 2021. 858 p.

ДОДАТОК А

Код інтернет-магазину

main.php

```
<?php
use yii\helpers\Html;
use yii\bootstrap\Nav;
use yii\bootstrap\NavBar;
use yii\widgets\Breadcrumbs;
use frontend\assets\AppAsset;
use frontend\widgets\Alert;
/* @var $this \yii\web\View */
/* @var $content string */
AppAsset::register($this);
?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>">
<head>
    <meta charset="<?= Yii::$app->charset ?>"/>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <?= Html::csrfMetaTags() ?>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
</head>
<body>
    <?php $this->beginBody() ?>
    <div class="wrap">
        <?php
            NavBar::begin([
                'brandLabel' => 'A-LISIA',
                'brandUrl' => Yii::$app->homeUrl,
                'options' => [
                    'class' => 'navbar-inverse navbar-fixed-top',
                ],
            ]);
            $itemsInCart = Yii::$app->cart->getCount();
            $menuItems = [
                ['label' => 'КАК ЗАКАЗАТЬ', 'url' => ['/site/about']],
```

```
            //[‘label’ => ‘Контакты’, ‘url’ => [‘/site/contact’]],
            [‘label’ => ‘<span class="glyphicon glyphicon-shopping-cart"></span>‘ . ($itemsInCart ? " ($itemsInCart)" : ‘‘), ‘url’ => [‘/cart/list’]],
        ];
        echo Nav::widget([
            ‘encodeLabels’ => false,
            ‘options’ => [‘class’ => ‘navbar-nav navbar-right’],
            ‘items’ => $menuItems,
        ]);
        NavBar::end();
    ?>
    <div class="container">
    <?= Breadcrumbs::widget([
        ‘links’ => isset($this->params[‘breadcrumbs’]) ? $this->params[‘breadcrumbs’] : [],
    ]) ?>
    <?= Alert::widget() ?>
    <?= $content ?>
    </div>
  </div>
  <footer style="background: black; color: #B3B3AD; text-align: center;" class="footer">
    <div class="container">
      <div class="row">
        <div class="col-sm-4">
        <p class="pull-left">Наши контакты: </p>
        </div>
        <div class="col-sm-4">
          <p class="pull-left">Тел.: </p>
          <p class="pull-left"> +38(099)123-45-67 </p>
        </div>
        <div class="col-sm-4">
          <p class="pull-left">E-mail: </p>
         <p class="pull-left"> alisia@mail.ru </p>
        </div>
    </div>
    </div>
  </footer>
  <footer id="footer"><!--Footer-->
    <div style="background: black; text-align: center" class="footer-bottom">
      <div class="container">
        <div  class="row">
          <p >Copyright © <?= date(‘Y’) ?> A-LISIA</p>
        </div>
```

```
            </div>
        </div>
    </footer><!--/Footer-->
    <?php $this->endBody() ?>
</body>
</html>
<?php $this->endPage() ?>
```

Category.php

```
<?php
namespace common\models;
use Yii;
use yii\behaviors\SluggableBehavior;
class Category extends \yii\db\ActiveRecord
{
    public function behaviors()
    {
        return [
            [
                'class' => SluggableBehavior::className(),
                'attribute' => 'title',
                'slugAttribute' => 'slug'
            ],
        ];
    }
    public static function tableName()
    {
        return 'category';
    }
    public function rules()
    {
        return [
            [['parent_id'], 'default', 'value' => null],
            [['parent_id'], 'integer'],
            [['title'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'parent_id' => 'Родительская категория',
```

```
            'title' => 'Категория',
            'slug' => 'Slug',
        ];
    }
    public function getParent()
    {
        return $this->hasOne(Category::className(), ['id' => 'parent_id']);
    }
    public function getCategories()
    {
        return $this->hasMany(Category::className(), ['parent_id' => 'id']);
    }
    public function getProducts()
    {
        return $this->hasMany(Product::className(), ['category_id' => 'id']);
    }
}
```

Image.php

```
<?php
namespace common\models;
use Yii;
use yii\web\UploadedFile;
class Image extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'image';
    }
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'image' => 'Фото',
            'product_id' => 'Товар',
        ];
    }
  public function getProduct()
    {
        return $this->hasOne(Product::className(), ['id' => 'product_id']);
    }
    protected function getHash()
```

```
    {
        return md5($this->product_id . '-' . $this->id);
    }
    public function getPath()
    {
        return Yii::getAlias('@frontend/web/images/' . $this->getHash() . '.jpg');
    }
    public function getUrl()
    {
        return Yii::getAlias('@frontendWebroot/images/' . $this->getHash() . '.jpg');
    }
    public function afterDelete()
    {
        unlink($this->getPath());
        parent::afterDelete();
    }
}
```

Order.php

```
<?php
namespace common\models;
use Yii;
use yii\behaviors\TimestampBehavior;
class Order extends \yii\db\ActiveRecord
{
    const STATUS_NEW = 'Новый';
    const STATUS_IN_PROGRESS = 'В процессе';
    const STATUS_DONE = 'Выполнен';
    const StatusZero = 0;
    const StatusOne = 1;
    const StatusTwo = 2;
    public function behaviors()
    {
        return [
            TimestampBehavior::className(),
        ];
    }
    public static function tableName()
    {
        return 'order';
    }
    public function rules()
```

```
{
    return [
        [[‘phone’, ‘email’, ‘address’], ‘required’],
        [[‘notes’, ‘status’], ‘string’],
        [[‘phone’, ‘email’], ‘string’, ‘max’ => 255],
        [[‘email’], ‘email’],
    ];
}
public function attributeLabels()
{
    return [
        ‘id’ => ‘ID’,
        ‘created_at’ => ‘Добавлен’,
        ‘updated_at’ => ‘Обновлен’,
        ‘phone’ => ‘Телефон’,
        ‘address’ => ‘Адрес’,
        ‘email’ => ‘Email’,
        ‘notes’ => ‘Заметки’,
        ‘status’ => ‘Статус’,
    ];
}
public function getOrderItems()
{
    return $this->hasMany(OrderItem::className(), [‘order_id’ => ‘id’]);
}
public function beforeSave($insert)
{
    if (parent::beforeSave($insert)) {
        if ($this->isNewRecord) {
            $this->status = self::StatusZero;
        }
        return true;
    } else {
        return false;
    }
}
public static function getStatuses()
{
    return [
        self::STATUS_DONE => ‘Выполнен’,
        self::STATUS_IN_PROGRESS => ‘В процессе’,
        self::STATUS_NEW => ‘Новый’,
```

```
        ];
    }
    public function getStatusName()
    {
        $values = array(
            self::StatusZero => ‘Новый’,
            self::StatusOne => ‘В процессе’,
            self::StatusTwo => ‘Выполнен’
        );
        if(isset($values[$this->status])) {
            return $values[$this->status];
        }
    }
    public function sendEmail()
    {
        return Yii::$app->mailer->compose(‘order’, [‘order’ => $this])
            ->setTo(Yii::$app->params[‘adminEmail’])
            ->setFrom(Yii::$app->params[‘adminEmail’])
            ->setSubject(‘New order #’ . $this->id)
            ->send();
    }
}
```

OrderItem.php

```
<?php
namespace common\models;
use Yii;
class OrderItem extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return ‘order_item’;
    }
    public function rules()
    {
        return [
            [[‘quantity’], ‘number’],
        ];
    }
    public function attributeLabels()
    {
        return [
```

```
            'id' => 'ID',
            'order_id' => 'Order ID',
            'title' => 'Товар',
            'price' => 'Цена',
            'product_id' => 'Product ID',
            'quantity' => 'Количество',
        ];
    }
    public function getProduct()
    {
        return $this->hasOne(Product::className(), ['id' => 'product_id']);
    }
    public function getOrder()
    {
        return $this->hasOne(Order::className(), ['id' => 'order_id']);
    }
}
```

Product.php

```
<?php
namespace common\models;
use Yii;
use yii\behaviors\SluggableBehavior;
use yz\shoppingcart\CartPositionInterface;
use yz\shoppingcart\CartPositionTrait;
class Product extends \yii\db\ActiveRecord implements CartPositionInterface
{
    use CartPositionTrait;
    public static function tableName()
    {
        return 'product';
    }
    public function behaviors()
    {
        return [
            [
                'class' => SluggableBehavior::className(),
                'attribute' => 'title',
            ]
        ];
    }
    public function rules()
```

```
    {
        return [
            [['description'], 'string'],
            [['category_id'], 'integer'],
            [['price'], 'number'],
            [['title'], 'string', 'max' => 255]
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'title' => 'Товар',
            'slug' => 'Slug',
            'description' => 'Описание',
            'category_id' => 'Категория',
            'price' => 'Цена',
        ];
    }
    public function getImages()
    {
        return $this->hasMany(Image::className(), ['product_id' => 'id']);
    }
    public function getOrderItems()
    {
        return $this->hasMany(OrderItem::className(), ['product_id' => 'id']);
    }
    public function getCategory()
    {
        return $this->hasOne(Category::className(), ['id' => 'category_id']);
    }
    public function getPrice()
    {
        return $this->price;
    }
    public function getId()
    {
        return $this->id;
    }
}
```

User.php

```
<?php
namespace common\models;
use Yii;
use yii\base\NotSupportedException;
use yii\behaviors\TimestampBehavior;
use yii\db\ActiveRecord;
use yii\web\IdentityInterface;
class User extends ActiveRecord implements IdentityInterface
{
    const STATUS_DELETED = 0;
    const STATUS_ACTIVE = 10;
    public static function tableName()
    {
        return '{{%user}}';
    }
    public function behaviors()
    {
        return [
            TimestampBehavior::className(),
        ];
    }
    public function rules()
    {
        return [
            ['status', 'default', 'value' => self::STATUS_ACTIVE],
            ['status', 'in', 'range' => [self::STATUS_ACTIVE, self::STATUS_DELETED]],
        ];
    }
    public static function findIdentity($id)
    {
        return static::findOne(['id' => $id, 'status' => self::STATUS_ACTIVE]);
    }
    public static function findIdentityByAccessToken($token, $type = null)
    {
        throw new NotSupportedException('"findIdentityByAccessToken" is not implemented.');
    }
    public static function findByUsername($username)
    {
        return static::findOne(['username' => $username, 'status' => self::STATUS_ACTIVE]);
    }
    public static function findByPasswordResetToken($token)
    {
```

```
    if (!static::isPasswordResetTokenValid($token)) {
        return null;
    }
    return static::findOne([
        'password_reset_token' => $token,
        'status' => self::STATUS_ACTIVE,
    ]);
}

public static function isPasswordResetTokenValid($token)
{
    if (empty($token)) {
        return false;
    }
    $expire = Yii::$app->params['user.passwordResetTokenExpire'];
    $parts = explode('_', $token);
    $timestamp = (int) end($parts);
    return $timestamp + $expire >= time();
}
public function getId()
{
    return $this->getPrimaryKey();
}
public function getAuthKey()
{
    return $this->auth_key;
}
public function validateAuthKey($authKey)
{
    return $this->getAuthKey() === $authKey;
}
public function validatePassword($password)
{
    return Yii::$app->security->validatePassword($password, $this->password_hash);
}
public function setPassword($password)
{
    $this->password_hash = Yii::$app->security->generatePasswordHash($password);
}
public function generateAuthKey()
{
    $this->auth_key = Yii::$app->security->generateRandomString();
```

```
    }
    public function generatePasswordResetToken()
    {
        $this->password_reset_token = Yii::$app->security->generateRandomString() . '_' . time();
    }
    public function removePasswordResetToken()
    {
        $this->password_reset_token = null;
    }
}
```

CartController.php

```
<?php
namespace frontend\controllers;
use common\models\Order;
use common\models\OrderItem;
use common\models\Product;
use yz\shoppingcart\ShoppingCart;
use delagics\liqpay\LiqPay;
class CartController extends \yii\web\Controller
{
    public function actionAdd($id)
    {
        $product = Product::findOne($id);
        if ($product) {
            \Yii::$app->cart->put($product);
            return $this->goBack();
        }
    }
    public function actionList()
    {
        $cart = \Yii::$app->cart;
        $products = $cart->getPositions();
        $total = $cart->getCost();
        $order_id = uniqid();
        $public_key = 'i7175834200';
        $private_key = 'bfiKv7fKv1Kx7KigXf4bsqqqF7upW2iVtxjr5rfk';
        $liqpay = new LiqPay($public_key, $private_key);
        $pay = $liqpay->cnb_form(array(
            'version'        => '3',
            'sandbox'        => '1',
            'amount'         => $total,
```

```
        'currency'       => 'UAH',
        'description'    => 'Оплата тестового заказа',
        'order_id'       => $order_id
    ));
    return $this->render('list', [
      'products' => $products,
      'total' => $total,
      'pay'=>$pay,
    ]);
}
public function actionRemove($id)
{
    $product = Product::findOne($id);
    if ($product) {
        \Yii::$app->cart->remove($product);
        $this->redirect(['cart/list']);
    }
}
public function actionUpdate($id, $quantity)
{
    $product = Product::findOne($id);
    if ($product) {
        \Yii::$app->cart->update($product, $quantity);
        $this->redirect(['cart/list']);
    }
}
public function actionOrder()
{
    $order = new Order();
    $cart = \Yii::$app->cart;
    $products = $cart->getPositions();
    $total = $cart->getCost();
    if ($order->load(\Yii::$app->request->post()) && $order->validate()) {
        $transaction = $order->getDb()->beginTransaction();
        $order->address;
        $order->save(false);
        foreach($products as $product) {
            $orderItem = new OrderItem();
            $orderItem->order_id = $order->id;
            $orderItem->title = $product->title;
            $orderItem->price = $product->getPrice();
            $orderItem->product_id = $product->id;
```

```
            $orderItem->quantity = $product->getQuantity();
            if (!$orderItem->save(false)) {
                $transaction->rollBack();
                \Yii::$app->session->addFlash('error', 'Cannot place your order. Please contact us.');
                return $this->redirect('catalog/list');
            }
        }
        $transaction->commit();
        \Yii::$app->cart->removeAll();
        \Yii::$app->session->addFlash('success', 'Спасибо за вашу покупку! Мы свяжемся с вами в ближайшее время.');
        $order->sendEmail();
        return $this->redirect('/frontend/web/catalog/list');
    }
    return $this->render('order', [
        'order' => $order,
        'products' => $products,
        'total' => $total,
    ]);
  }
}
```

ProductController.php

```
<?php

namespace backend\controllers;
use common\models\Category;
use Yii;
use common\models\Product;
use backend\models\ProductSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
class ProductController extends Controller
{
  public function behaviors()
  {
    return [
      'verbs' => [
        'class' => VerbFilter::className(),
        'actions' => [
          'delete' => ['post'],
```

```
            ],
        ],
    ];
}
public function actionIndex()
{
    $searchModel = new ProductSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

public function actionView($id)
{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}
public function actionCreate()
{
    $categories = Category::find()->all();
    $model = new Product();
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
            'categories' => $categories,
        ]);
    }
public function actionUpdate($id)
{
    $categories = Category::find()->all();
    $model = $this->findModel($id);
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
            'categories' => $categories,
```

```
            ]);
        }
    }
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();
        return $this->redirect([‘index’]);
    }
    protected function findModel($id)
    {
        if (($model = Product::findOne($id)) !== null) {
            return $model;
        } else {
            throw new NotFoundHttpException(‘The requested page does not exist.’);
        }
    }
}
```