**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

# ВИПУСКНА РОБОТА

**на тему:**

## "Інформаційний ресурс книжкового видавництва з реалізацією маркетингового алгоритму"

**Завідувач**
**випускаючої кафедри** **Довбиш А.С.**

**Керівник роботи** **Проценко О.Б.**

**Студент гр. ІН–61** **Ткачова М.О.**

**СУМИ 2020**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

**Кафедра комп'ютерних наук**

Затверджую _____

Зав. кафедрою Довбиш А.С.

“_____” _____2020 р.

## Завдання

## до випускної роботи

Студента четвертого курсу, групи ІН—61 спеціальності “Інформатика” денної форми навчання Ткачової Маргарити Олександрівни.

**Тема: “Інформаційний ресурс книжкового видавництва з реалізацією маркетингового алгоритму”**

Затверджена наказом по СумДУ

№_____ від _____2020 р.

**Зміст пояснювальної записки**: 1) аналітичний огляд методів побудови веб–додатку; 2) постановка завдання й формування завдань дослідження; 3) огляд і опис засобів для розробки; 4) розробка інформаційного ресурсу; 5) аналіз результатів.

Дата видачі завдання “_____”_____2020 р.

Керівник випускної роботи _____ Проценко О.Б.

Завдання прийняв до виконання _____ Ткачова М.О.

# РЕФЕРАТ

**Записка:** 50 стор., 36 рис., 4 додатка, 15 джерел.

**Об'єкт дослідження** — Інформаційний ресурс книжкового видавництва з реалізацією маркетингового алгоритму.

**Мета роботи —** Розробити інформаційний ресурс книжкового видавництва, який має бути зручним та зрозумілим у використанні, привабливим для користувача. Ресурс має бути адаптивним, а також працювати у сучасних популярних веб–браузерах.

**Результати —** Виконано вибір методів вирішення поставленої задачі; на основі фреймворку Yii2 реалізовано клієнтську частину інформаційного ресурсу книжкового видавництва.

ІНФОРМАЦІЙНИЙ РЕСУРС, КНИЖКОВЕ ВИДАВНИЦТВО,
ФРЕЙМВОРК, HTML, CSS, JAVASCRIPT, YII2

# ЗМІСТ

# ВСТУП

Розвиток технологій не призупинюється ні на хвилину, вони постійно змінюються і удосконалюються. Зараз неможливо уявити своє існування без них, вони посідають невід'ємне місце у більшості сфер життя людини.

Навіть покупки тепер можна легко зробити за допомогою інтернету. Необхідно лише знайти відповідний електронний магазин, додати вподобані товари до корзини та оплатити замовлення. І для цього навіть не потрібно вставати з насидженого дивана.

Сьогодні майже кожна організація має особистий сайт, на якому можна подивитися інформацію про послуги, які вона надає. Правильно створений веб–сайт — це гарна візитівка для компанії. Коли у організації або зовсім немає сайту, або він наповнений недостатньою кількістю інформації, або ж має дуже незручний дизайн, це становиться істотною перешкодою для успішної взаємодії компанії та клієнта.

Тому для багатьох організацій, індивідуальних підприємців та інших, досить важливим пунктом в їх діяльності є розробка такої інформаційної системи, яка змогла б забезпечити максимально простий спосіб взаємодії представника компанії та клієнта через мережу інтернет.

Метою роботи є розробка веб-сайту для книжкового магазину. Об'єктом дослідження є процеси збору, зберігання, обробки та подання клієнтам інформації про магазин. Предметом є розробка веб–сайту, який дозволяє реалізувати ці процеси. Для досягнення мети були поставлено наступні завдання:

- Провести аналіз веб-сайтів схожої тематики в інтернеті.
- Розробити зручний дизайн веб–сайту.
- Обрати засоби та технології для розробки веб–сайту.

# 1 ІНФОРМАЦІЙНИЙ ОГЛЯД

## 1.1 Клієнт-серверна архітектура в розробці інформаційних ресурсів

Інформаційний ресурс — це створений у мережі Інтернет веб–додаток для відображення необхідної інформації про компанію, її взаємодії з клієнтами шляхом онлайн продажу товарів та різноманітних послуг. Це реалізований зв’язок між співробітниками магазину та покупцями, які можуть бути територіально роз’єднаними і не мати можливості спілкуватися за допомогою прямого контакту, тому співпраця здійснюється через електронні засоби зв’язку.

На даний момент існує досить велика кількість інтернет–магазинів. Зустрічаються такі критерії оцінки сайтів:

- Зміст.
- Структура.
- Дизайн.
- Функціональність.
- Інтерактивність.
- Загальне враження.

Зміст — це інформація, що представлена на сайті. Інформаційне наповнення повинне, перш за все, відповідати тематиці сайту та привертати увагу відвідувача. Дані мають бути достатньо повними, але не занадто об’ємними.

Структура характеризується організацією інформації та можливістю навігації між розділами. Вона має бути простою та ефективною, за допомогою чого користувач матиме змогу створити уявну модель змісту сайту. Якщо представлену інформацію можна легко охопити, то клієнт знайде необхідні йому дані з меншою витратою часу. Це підвищить привабливість та ефективність ресурсу.

Дизайн — це зовнішній вигляд сайту. Критеріями оцінки для нього є висока якість, доречність та відповідність заданій тематиці. Він не має бути перенасиченим елементами, але і занадто простим виглядати не повинен.

Функціональність характеризується технологічною стороною ресурсу: швидкість роботи, доречність та повна працездатність. Також сайт повинен бути універсальним: функціонувати на всіх популярних платформах та з різною роздільною здатністю екрана.

Під інтерактивністю мається на увазі можливість спілкування між клієнтом та робітником, двосторонній обмін інформацією.

Загальне враження є мало не головним критерієм, адже якщо клієнт буде задоволений сайтом загалом, він з більшою ймовірністю повернеться пізніше та/або зробить покупки. Загальне враження — це сума усіх попередніх складових, але є й деякі інші ледве помітні тонкощі, що змушують користувача залишитися або піти.

Клієнт–серверна архітектура передбачає взаємодію між двома головними її частинами — клієнтом та сервером, що обслуговує своїх клієнтів. Сервери надають інформацію та послуги, а клієнт ними користується. Їх взаємодія відбувається за рахунок мережі (наприклад, Інтернет). Правилами такої співпраці є протоколи обміну.

Є три основних рівня роботи клієнт-серверної архітектури:

- Перший — це інтерфейс користувача, де клієнт вводить керуючі команди. Іншими словами — фронт–енд.
- Другий — це бек–енд, тобто рівень, на якому відбувається обробка інформації, прийняття команд від клієнта.
- Третій — це рівень, що забезпечує управління даних, їх зберігання та ін. В більшості випадків усі ці операції здійснюються за допомогою баз даних.

«Спілкування» між рівнями відбувається за допомогою архітектурного стилю REST (Representational state transfer). Він передбачає чотири основні

операції: GET (отримання даних), POST (передача даних на сервер), PUT (зміна цих даних) та DELETE (видалення даних із серверу). Перед обробкою самих даних спочатку відбувається валідація, перевірка доступу, обробка помилок та ін.

## 1.2 Огляд існуючих рішень

Для прикладу коротко оглянемо декілька українських книжкових онлайн–магазинів.

Клуб сімейного дозвілля — досить популярний магазин та видавництво. Веб-сайт має простий, але привабливий та зручний дизайн.

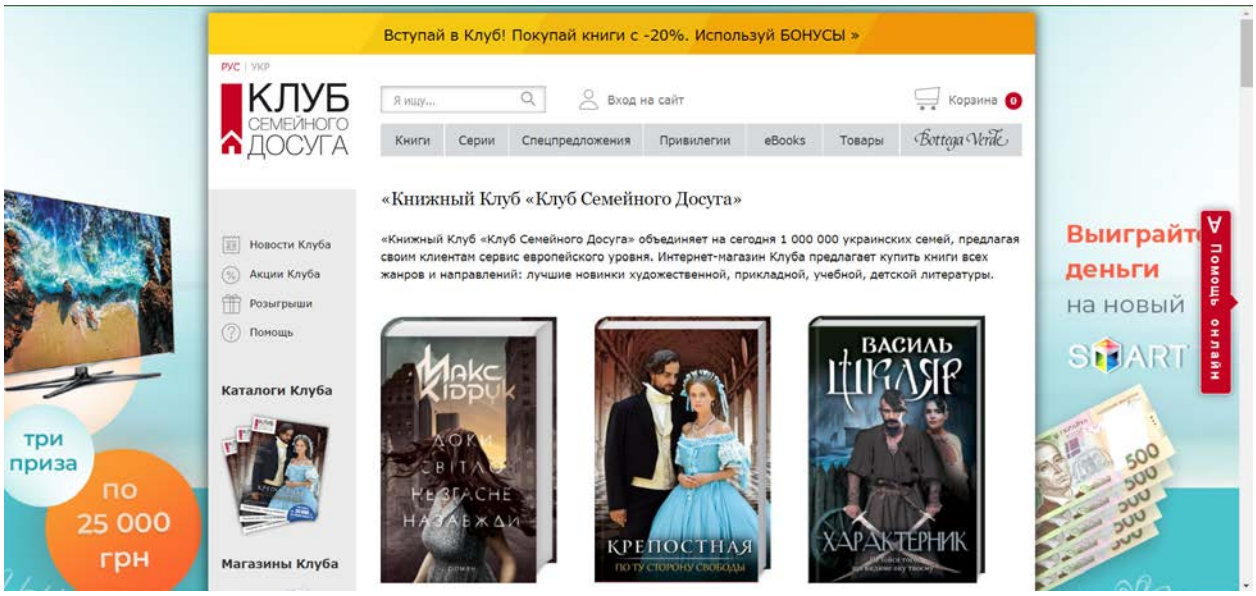


Рисунок 1.1 – Головна сторінка сайту «Клуб сімейного дозвілля»

В головному меню виділені навігація товарів, акції та новини, поле пошуку, корзина, авторизація та ідеального розміру логотип компанії. На головній сторінці користувач може знайти коротку інформацію про Клуб. Усі складоові розміщені доцільно та правильно, немає зайвих речей, погляд відвідувача може сфокусуватися на головному — на товарах, що надає видавництво.

Проте не вся інформація знаходиться на видноті. Дані про доставку, оплату та більше про сам Клуб винесли у футер.

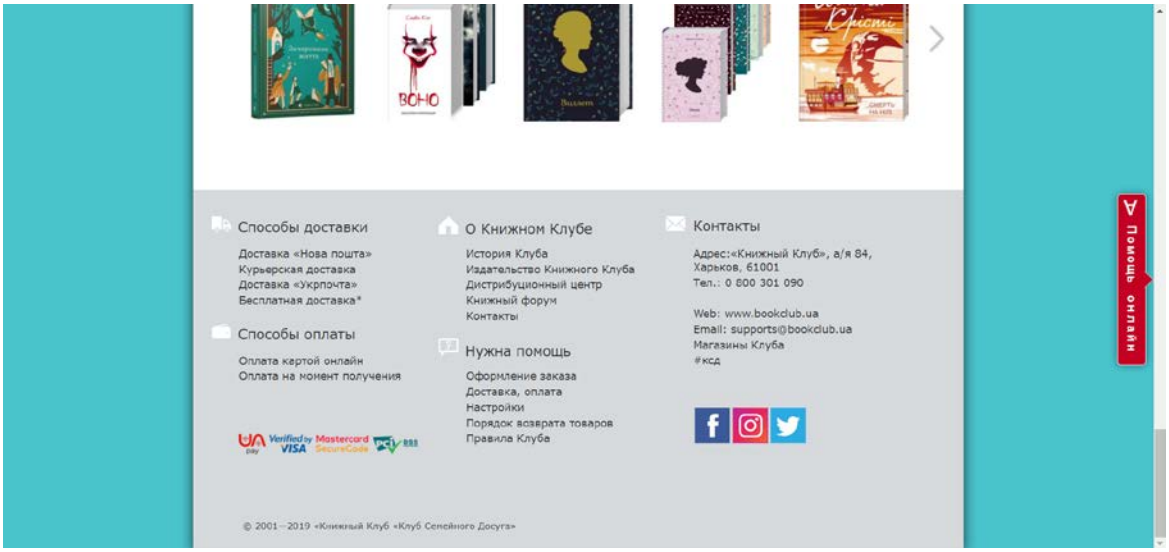


Рисунок 1.2 – Футер сайту «Клуб сімейного дозвілля»

Це не достатньо зручно, оскільки не всі прокручують сторінку до кінця. Та все же на сайті є весь обсяг потрібної інформації (про доставку з оплатою з покроковою інструкцією, контакти для зворотнього зв’язку та інше).

Сайт онлайн–магазину Yakaboo також оформлений досить просто, без зайвих деталей, в приємній кольоровій гамі. Тут більш вдала навігація, оскільки все знаходиться в верхньому меню, біля логотипу компанії. Каталогом товарів являється випадаюче меню, щоб не займати місця.

Клієнт, тільки зайшовши на сайт, матиме змогу за короткий час знайти усю необхідну йому інформацію — від списку книг до контактів, за допомогою яких можна звернутися до представника компанії.

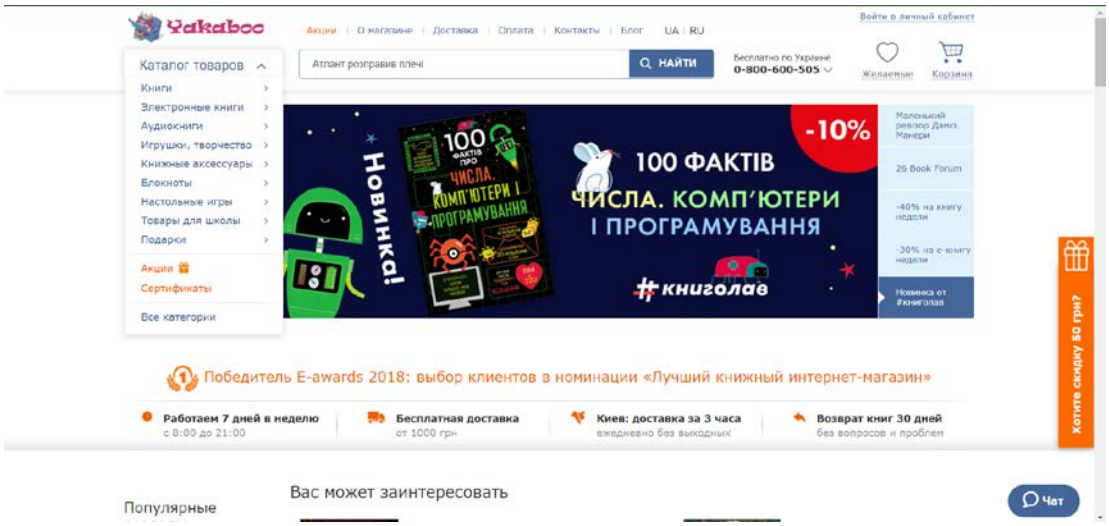


Рисунок 1.3 – Головна сторінка сайту «Yakaboo»

Під банерами розміщена така важлива інформація, як графік роботи, вимоги до безкоштовної доставки та повернення книг.

Сайти «Клубу сімейнго дозвілля» та «Yakaboo» залишають після себе приємне загальне враження. Товари розподілені за категоріями, уся необхідна інформація розміщена на сторінках, дизайн привабливий, замовлення оформляється досить просто.

Розглянемо інші, менш вдалі випадки.

Хедер у сайту веб–магазину «КнигоЛенд» занадтно великий та перенасичений інформацією. Такі основні частини сайту як корзина та поле для пошуку губляться на екрані серед надлишків даних. В іншому оформлення досить приємне, вся інформація знаходиться в полі зору.

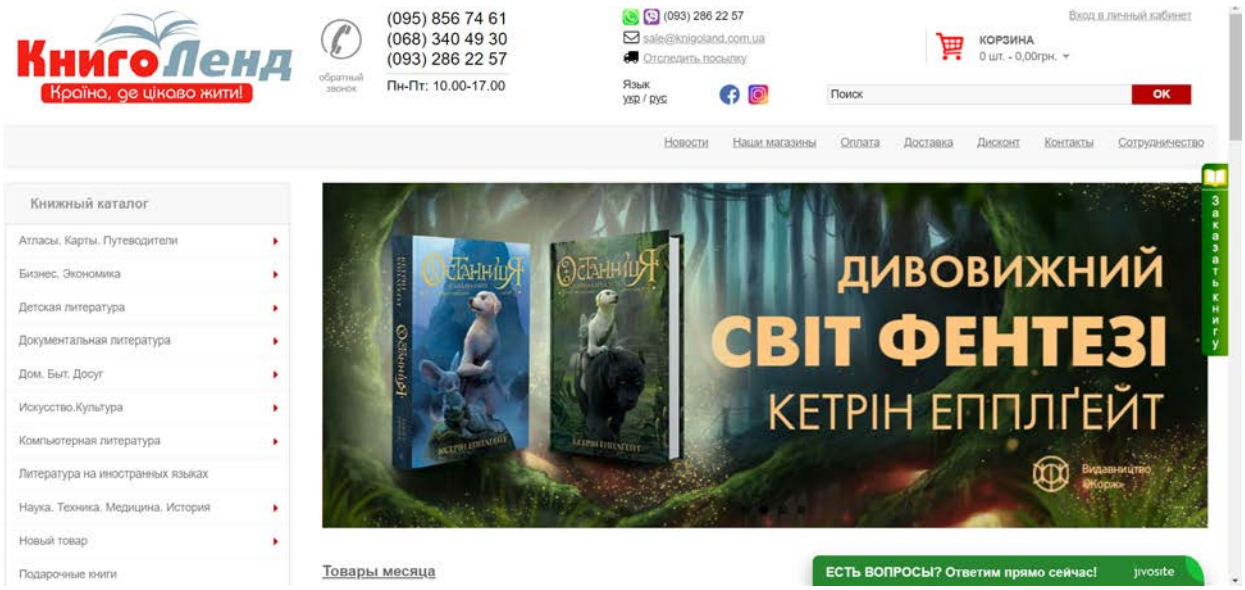


Рисунок 1.4 – Головна сторінка сайту «КнигоЛенд»

У магазину «book24» окрім проблем із кольоровою палітрою (для деяких людей цей дизайн може бути занадтно яскравим), є також труднощі з хедером, які випливають із попередньої проблеми. Колір корзини губиться на фоні, розмір було підібрано невдало. Проте у сайту є великий плюс — усе оформлено в одному стилі та є досить корисний футер — а це означає, що користувач швидко знайде усю необхідну йому інформацію.

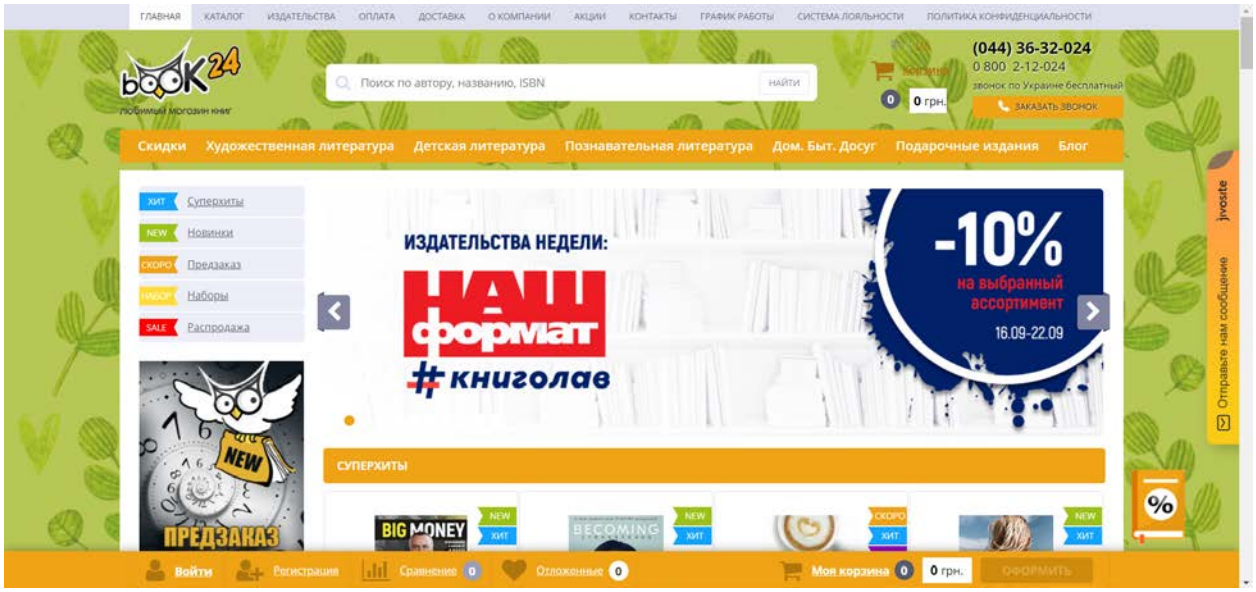


Рисунок 1.5 – Головна сторінка сайту «book24»

«Книгокрад» неправильно використовує розміщення елементів — досить багато пустого місця, що неприємно притягує погляд. Також деякі іконки дивляться занадто дешево.

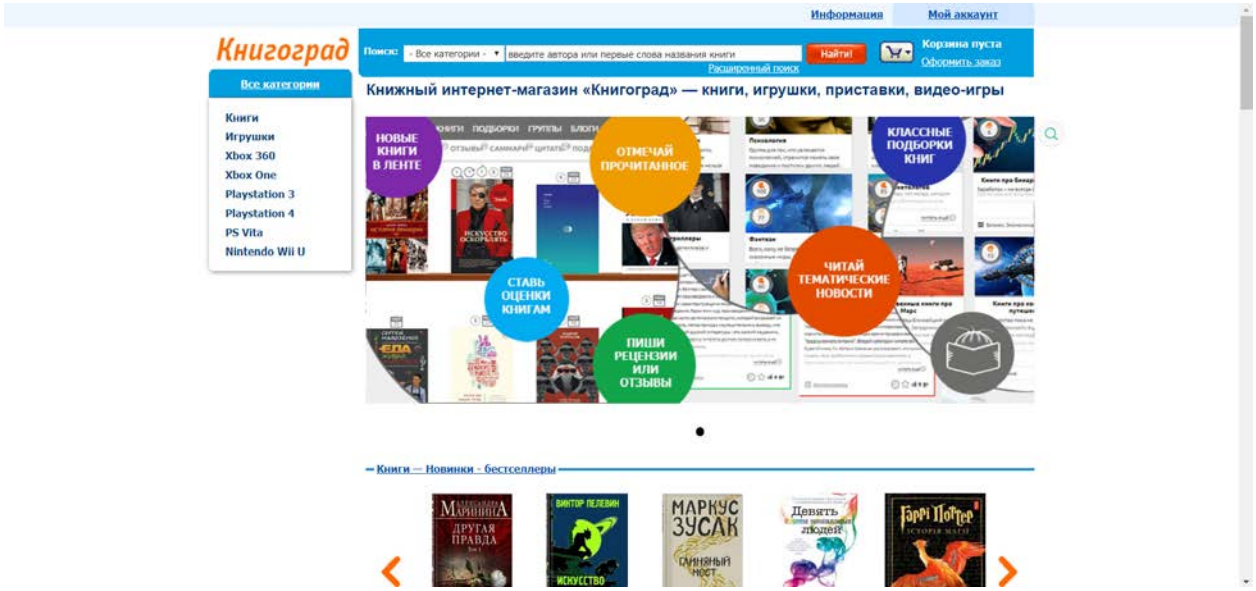


Рисунок 1.6 – Головна сторінка сайту «Книгокрад»

У «Книгарні Є» навпаки — сайт перевантажено різноманітними елементами, і погляд не може сконцетруватися на чомусь одному. До того ж використовується забагато червоного кольору — дуже яскраво.

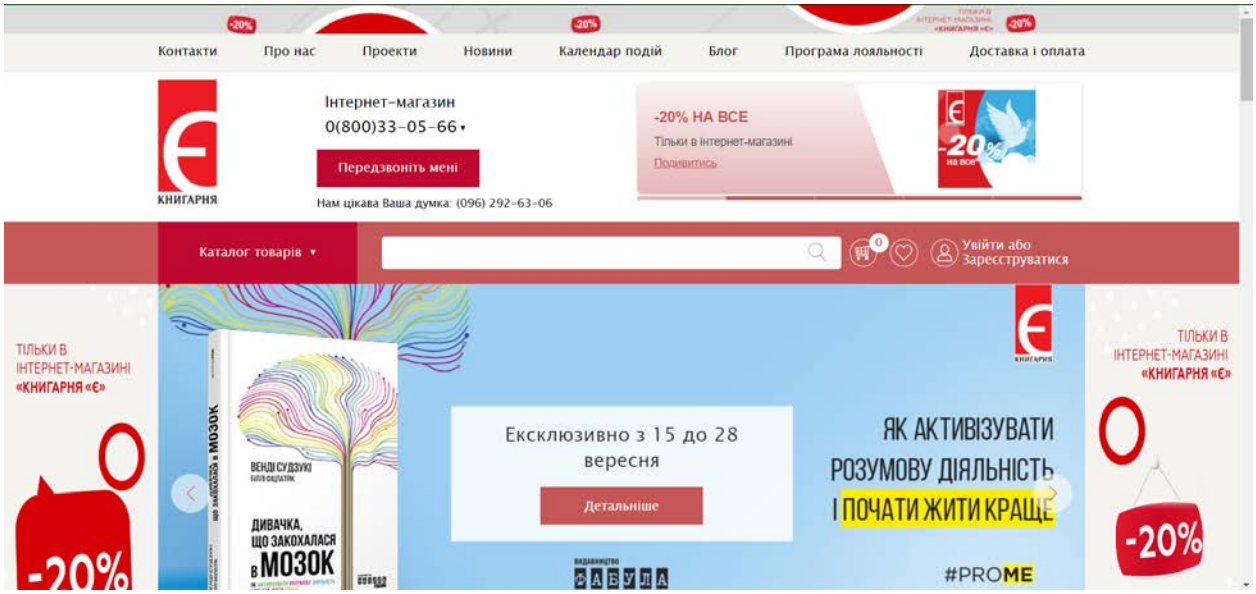


Рисунок 1.7 – Головна сторінка сайту «Книгарня Є»

У наступній таблиці представлені оцінки по всім критеріям для показаних сайтів.

Таблиця 1.1 – Порівняльна таблиця сайтів схожої тематики

| | **КСД** | **Yakaboo** | **КнигоЛенд** | **book24** | **Книгокрад** | **Книгарня Є** |
|---|---|---|---|---|---|---|
| **Зміст** | 10 | 10 | 10 | 10 | 8 | 10 |
| **Навігація** | 9 | 10 | 10 | 10 | 7 | 8 |
| **Дизайн** | 10 | 10 | 8 | 8 | 3 | 6 |
| **Функціональність** | 10 | 10 | 10 | 9 | 8 | 7 |
| **Інтерактивність** | 10 | 10 | 10 | 10 | 3 | 10 |

У результаті аналізу різних сайтів зі схожою тематикою були виявлені такі цілі:

Необхідна розробка декількох інтерфейсів: для клієнта та адміністратора. Клієнт повинен мати змогу переглядати інформацію, зв’язуватися з робітниками та робити покупки. Адміністратор може керувати правами користувача, їх обліковими записами, а також замовленнями.

Потрібно створити такий дизайн сайту, щоб він був виконаний у спокійному неагресивному стилі. Зовнішній вигляд має приваблювати, але не відволікати користувача від основного завдання — здійснення покупок.

Необхідно пам’ятати, що на сайті повинна бути розміщена уся важлива інформація — про доставку, оплату, контакти для зворотнього зв’язку. Це допоможе клієнту швидше зорієнтуватися та зважитися на покупку. Вся інформація повинна знаходитися в легкому доступі для кожного.

### 1.3 Маркетингова діяльність

Основою маркетингової діяльності є залучення клієнтів та збереження їх лояльності. Важливо знати різницю між різними категоріями клієнтів та розуміти як правильно з ними працювати. Існує три основних типи — гарячі, теплі та холодні клієнти, які відрізняються між собою діяльністю, яку необхідно провести для їх залучення до придбання товарів.

Гарячі клієнти — це клієнти, які максимально зацікавлені в купівлі послуг, що надаються компанією, тому вони являються найлегшою категорією. Такі клієнти одразу знають чого саме хочуть і хочуть придбати це якнайшвидше, тому їх не треба вмовляти, необхідно лише запропонувати бажані ними товари. Також їм можна запропонувати додаткові послуги, є деяка ймовірність (близько 30%), що вони згодяться. Але важливо чітко пам’ятати, що операцію з такими клієнтами треба закривати швидко і не задавати занадто багато питань.

Теплі клієнти — це клієнти, що зацікавлені в покупці певних товарів, але ще не вирішили точно — чи будуть брати цей товар і коли взагалі будуть. При роботі з такими клієнтами необхідно постійно підтримувати спілкування, поки клієнт не скаже залишкове «так» або «ні» на придбання послуги. Також треба робити так, щоб клієнт залишався зацікавленим в покупці. Це може бути досить складно, адже прийняття рішення для теплих клієнтів зазвичай досить довге і може постійно переноситися. Для роботи з такими покупцями

необхідно створити таку схему, яка дозволить не втрачати з ними зв’язок та залишити їх зацікавленими в покупці.

Холодні клієнти — це байдужі до покупки представлених компанією послуг клієнти. Вони можуть бути абсолютно незацікавленими, або ж можуть відмовитися через погане пояснення співробітника. Найважливіше в роботі з холодними клієнтами це знайти хоча б деяку частину, що готова йти на контакт, та не нав’язуватися ним.

Отже, при роботі з клієнтами існують різні правила для кожної категорії клієнтів. Гарячим необхідно запропонувати те, що їм треба, та зробити все швидко, з теплими підтримувати постійний зв’язок та залишати їх зацікавленими, а серед холодних знайти тих, хто може зацікавитися.

## 1.4 Постановка задачі

У даній роботі необхідно створити інформаційний ресурс для книжкового видавництва з реалізацією маркетингового алгоритму. Функціонал, який необхідно реалізувати:

- Каталог з товарами;
- Можливість придбання товарів;
- Особистий кабінет користувача;
- Особистий кабінет адміністратора, де адміністратор матиме змогу управляти замовленнями;
- Реєстрація/Авторизація;
- Новинна розсилка на електронну пошту для роботи з клієнтами.

Інформаційний ресурс має бути зручним та зрозумілим у використанні, привабливим для користувача. Ресурс має бути адаптивним, а також працювати у сучасних популярних веб–браузерах.

# 2 ВИБІР МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

## 2.1 Вибір методів розроблення

Написання сайту на «чистій» мові програмування є важким, занадто довгим та не досить якісним завданням, тому існують методи для спрощення роботи над розробкою власного веб-сайту: фреймворки та CMS.

Фреймворк являє собою набір бібліотек, які злагоджено взаємодіють між собою. Для створення веб–сайтів на мові PHP існує багато різноманітних фреймворків. Найпоширеніші з них:

- Laravel;
- Symfony;
- Yii2;
- Zend.

Ці фреймворки використовують у своїй основі шаблонну модель MVC, містять велику кількість додаткових бібліотек. Підходять для розробки інтернет-магазинів або корпоративних веб–сайтів.

CMS — Content Management System, або ж Система управління контентом, — майже готовий сайт, який можна кастомізувати під себе. Являє собою інструменти для додавання, редагування, видалення даних на веб–ресурсі.

Створити сайт за допомогою CMS, по–перше, досить легко, а по–друге, набагато швидше, ніж з використанням фреймворків. Це досягається за рахунок того, що CMS надає вже готові інтерфейси для користувача та адміністратора, які необхідно лише додатково налаштувати. Але такі простота та швидкість також являються шаблонністю та обмеженістю, адже просто так розширити функціонал автор не матиме змоги. З фреймворком будь-яка ідея може бути реалізована майже без обмежень, хоча на створення сайту на основі фреймворку буде витрачено більше сил та часу.

В CMS вже було задано певну структуру, що надали розробники системи, — простіше кажучи, ми працюємо з шаблоном, тому при розробці суттєво економиться час. З фреймворком навпаки — іноді досить складно зрозуміти чому саме так виконуються написані рядки коду. До того ж, доведеться витратити багато часу на вивчення самого фреймворка.

У фреймворках зазвичай містяться лише необхідні бібліотеки, а отже, і тільки необхідний код. І якщо захочеться розширити функціонал ще більше, можна додати нові бібліотеки, створити власні віджети тощо. В CMS нерідко зустрічається ситуація, коли в ньому містяться зайві модулі, а отже і зайвий, іноді шкідливий, код.

Оскільки в CMS — це вже готовий шаблон, у ньому вже визначено який модуль з яким і як буде взаємодіяти, як це вплине на користувача. Фреймворки дозволяють не заганяти себе в рамки — можна самому вирішувати як юзер взаємодіятиме з системою. Тобто фреймворк дарує більше свободи, але для новачка, наприклад, він може бути занадто важкими, — для використання фреймворків необхідно володіти достатнім рівнем знань. Спочатку доведеться попрацювати на «чистій» мові програмування, а вже потім приступати до більш складних речей.

До основних позитивних сторін CMS можна віднести:

- Швидкість, яка досягається шаблонністю.
- Вже готові інтерфейси для користувача та адміністратора.
- Легкість та простота в створенні власного сайту. З використанням CMS не потрібно навіть володіти навичками програмування.

Серед мінусів:

- Шаблонність. Вона приносить менші витрати часу, але також і обмеження в функціоналі.
- Велика кількість модулів, яка може вам не знадобитися.
- Менша загальна продуктивність у порівнянні з якісним сайтом,

написаним на фреймворку.

- В результаті вийде досить простий веб–ресурс.

Плюси фреймворків:

- Можливість втілити будь-які ідеї, легко розширити існуючий функціонал.
- Гарна продуктивність.

Мінуси:

- Важкість в реалізації. Необхідно володіти досить високим рівнем знань для програмування на фреймворку.
- Відсутність інтерфейсів для користувача та адміністратора — необхідно писати все майже з нуля.
- На вивчення фреймворку та реалізацію продукту на ньому витрачається багато часу.

Можна зробити такі висновки: робота на основі фрейморків довга та складна, але є можливість втілити майже будь-який функціонал; CMS — це швидко, легко та зручно, але обмежено та шаблонно.

Процес створення сайту за допомогою CMS дійсно досить швидкий — будь-яка людина без спеціальної підготовки зможе зробити, наприклад, інтернет–магазин або сайт–візитівку.

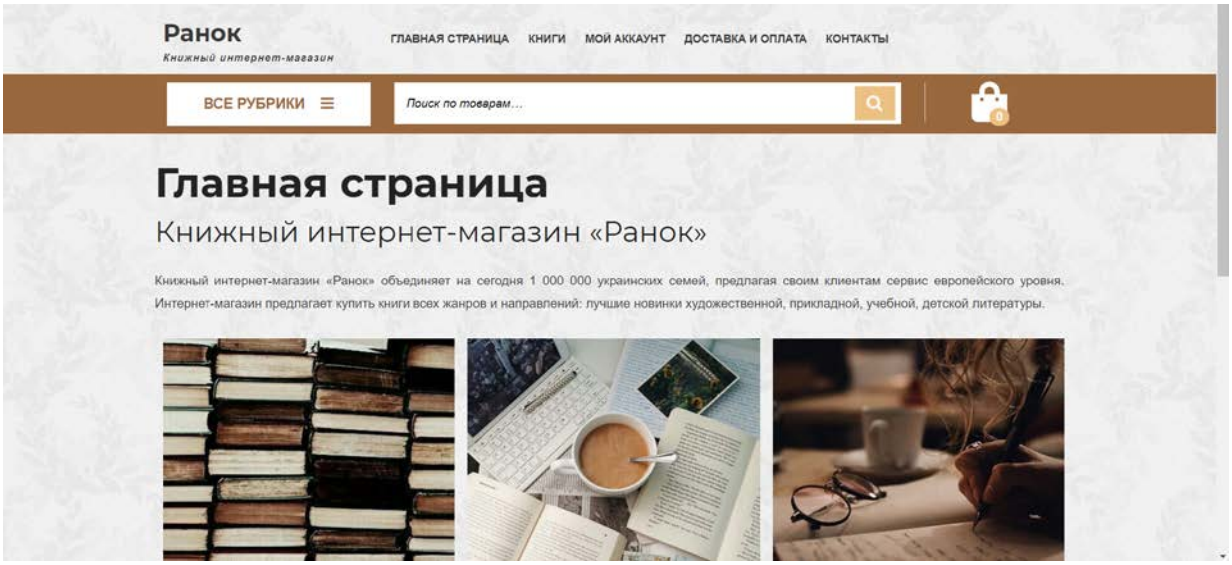


Рисунок 2.1 – Головна сторінка сайту, створеного за домогою CMS

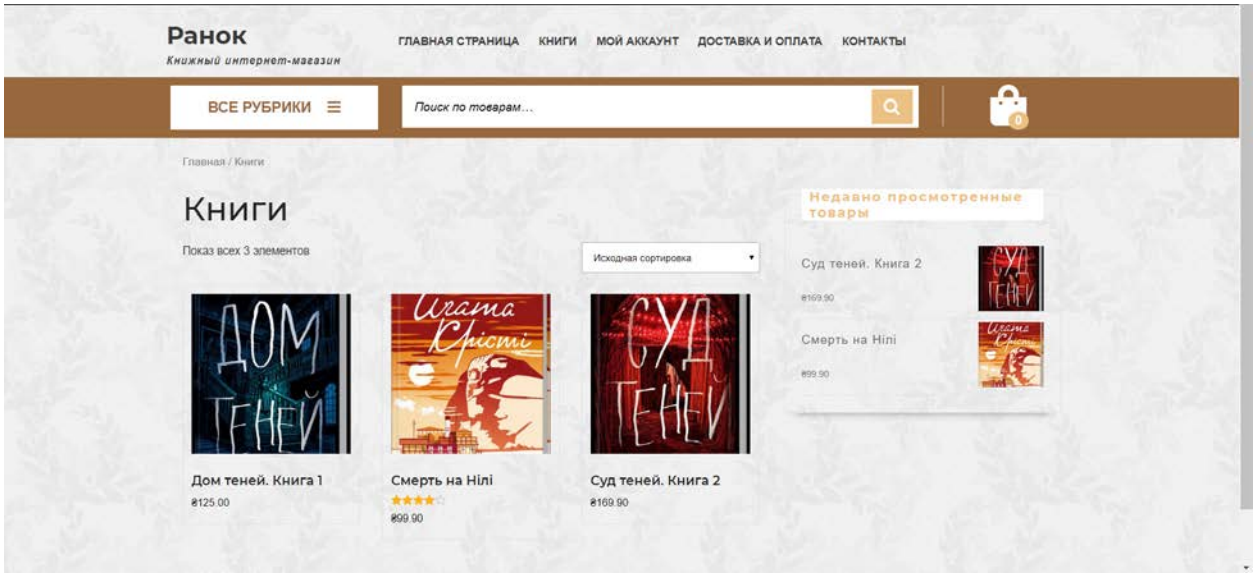


Рисунок 2.2 – Каталог товарів на сайті, створеного за допомогою CMS

Сайт було розроблено за допомогою CMS WordPress та плагіна WooCommerse для інтернет–магазинів. Дизайн був створений на основі готового шаблону VW Book Store від VW Themes.

Керування вмістом сайту, товарами, замовленнями та іншим відбувається за допомогою консолі WordPress та WooCommerce. Усі товари, що замовив клієнт, одразу відображаються на головній сторінці консолі, щоб адміністратор мав можливість побачити нове замовлення.

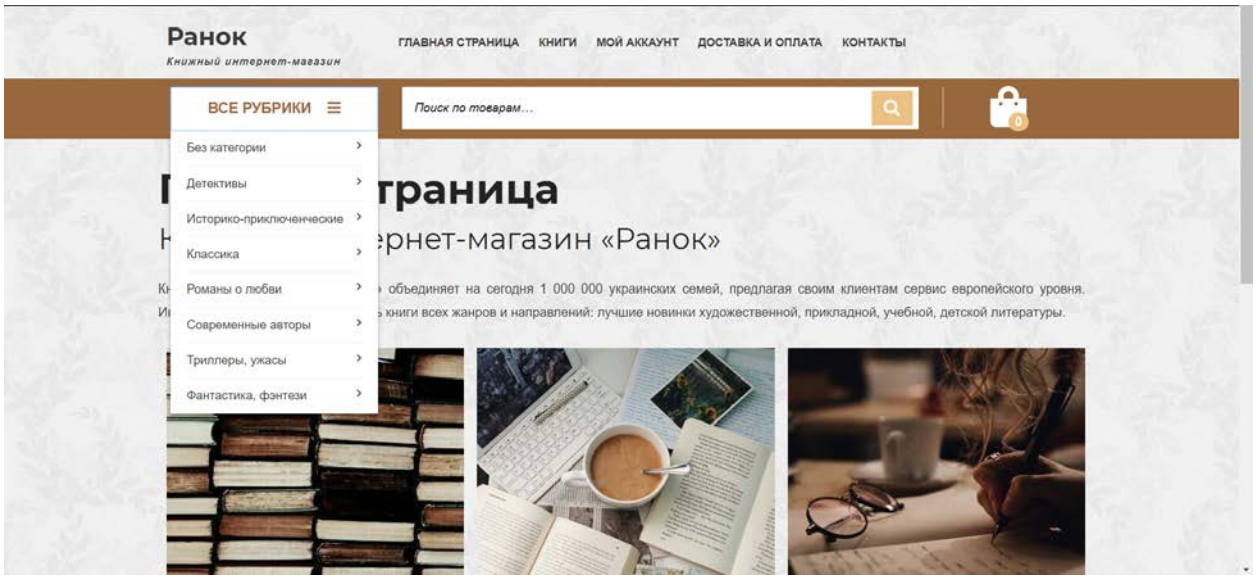


Рисунок 2.3 – Категорії товарів на сайті, створеного за допомогою CMS

Але великими мінусами стали: неадаптивність для майбутніх змін, залишкова простота та обмеження функціоналу. З фреймворками хоча і куди важче та довше працювати, але на основі них можна повністю створити свою логіку.

## 2.2 Вибір засобів програмування

У сучасному світі існує велика кількість програм та інструментів, які допомагають у створенні якісних веб–продуктів.

Програми, що дозволяють спростити технологію створення сайту:

- Photoshop — графічний редактор, що дозволяє редагувати зображення та зберігати їх. Існує спеціальний web–формат, в якому можна зберігати файли для нормального відображення на більшості екранах.
- Notepad++ — редактор з підсвічуванням коду. Зручно використовувати для створення HTML, CSS та JS–файлів.
- Open Server Panel — локальний веб–сервер для Windows. Вже містить в собі середовище для створення власної бази даних — це виконується за допомогою phpMyAdmin.
- NetBeans — IDE для розробки на таких мовах програмування, як Java, Python, PHP, C, C++ та ін. Також підсвічує код для HTML та CSS. Накраще середовище для розробки серед безкоштовних.

Засоби розробки веб–сайтів:

- HTML — HyperText Markup Language — являється основною мовою програмування для вебу. Не використовується як самостійна одиниця в безпосередньо розробці, оскільки має суттєві обмеження в функціональності уже готового продукту: сайт може бути лише статичним, без зворотнього зв’язку з користувачем. Тому HTML застосовується в поєднанні з іншими

мовами програмування. Створює структуру зовнішнього вигляду веб–системи.

- CSS — доповнення до HTML, мова для створення додаткових стилів для зміни зовнішнього вигляду тих чи інших елементів. Зручно виносити стилі у окремі файли. Так їх легше змінювати та знаходити потрібні.
- JavaScript — мова, яку зазвичай використовують разом з HTML, адже вона значно розширює функціонал. Являє собою скрипти, які інтегруються в загальний файл як підпрограма та викликаються, коли користувач починає взаємодіяти з системою, з відповідного рядка коду в HTML.
- XML — засіб для розмітки сторінки. Він допомагає контролювати правильність сформованих документів та застосованих мов програмування. Являється мовою структурування сторінок та не задає створення нового функціоналу. Правильно створена розмітка є дуже важливою складовою для правильної роботи та відображення візуалу, а це впливає на ефективність роботи в цілому та на просування/рекламу сайтів.
- PHP — мова програмування, яка задає скрипти для реалізації динамічних веб–сайтів. PHP–скрипти виконуються та обробляються на стороні сервера, клієнт бачить лише результат на інтерфейсі. Оскільки клієнт не зможе зрозуміти що саме він бачить — роботу скрипта або HTML–сторінку, то можна навіть зробити для сервера такі налаштування, щоб він на своїй стороні створював обробку усіх HTML–сторінок. Отже, за допомогою PHP розробляється бек–енд, на основі чого можна створити якісні динамічні веб–ресурси, які легко можна буде змінювати та підтримувати.

- MySQL — система для роботи з базами даних. Дозволяє працювати з даними у вигляді таблиць, обробляти їх, управляти запитами та оптимізовувати усі процеси.

Фреймворки для роботи з сайтами:

- Bootrstrap — фреймворк для спрощення роботи із зовнішнім виглядом сайту. Використовує HTML, CSS, JavaScript. Є одним із найпопулярніших та найпоширеніших фреймворків для верстування, оскільки дозволяє формувати дизайн набагато швидше. До того ж він безкоштовний, тому навіть початківці можуть створювати за допомогою нього якісні макети. Складається з набору CSS та JavaScript файлів. Для його використання необхідно підключити відповідні файли до сторінки. Для цього є два варіанти: у блоці <head> на самій HTML-сторінці помістити посилання або створити окремі папки у проекті з цими CSS та JavaScript файлами. Після
- Yii2 — фреймворк для бек–енду, який є найпростішим та найшвидшим з популярних фреймворків. Його перевагами є легка установка, наявність безлічі вбудованих рішень для інтерфейсів, можливість створювати власні віджети, генератор моделей, контролерів та CRUD. Вже має готовий користувацький інтерфейс — функції для входу, виходу, перегляду сторінок, зворотнього зв’язку. Достатньо лише застосувати власні стилі, і вже буде готовим шаблон для вашої майбутньої роботи.

Отже, для розробки було обрано фреймворк Bootstrap для фронт–енду та мову програмування PHP з фреймворком Yii2 для бек–енду.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Розробка моделі інформаційної системи

Інформаційна система реалізована на фреймворку Yii2, який передбачає структуру MVC — Model, View, Controller. Модель (Model) містить зв’язок та обробку даних. Зазвичай в моделях описуються бази даних, підключення до них, оновлення інформації в базах та ін. Вигляд (View) — це файли з верстокою, тобто HTML–коди тих сторінок, які бачить користувач. Потім ці файли підключаються у контролерах, які обробляють їх та показують користувачу. Тобто весь зовнішній вигляд сайту необхідно винести у вигляди. Контролер (Controller) — це інтерфейс між моделями та виглядами. У них обробляються усі основні операції.

MVC допомагає структурувати проект — код виділяється в окремі блоки, кожен з яких відповідає за свій функціонал.

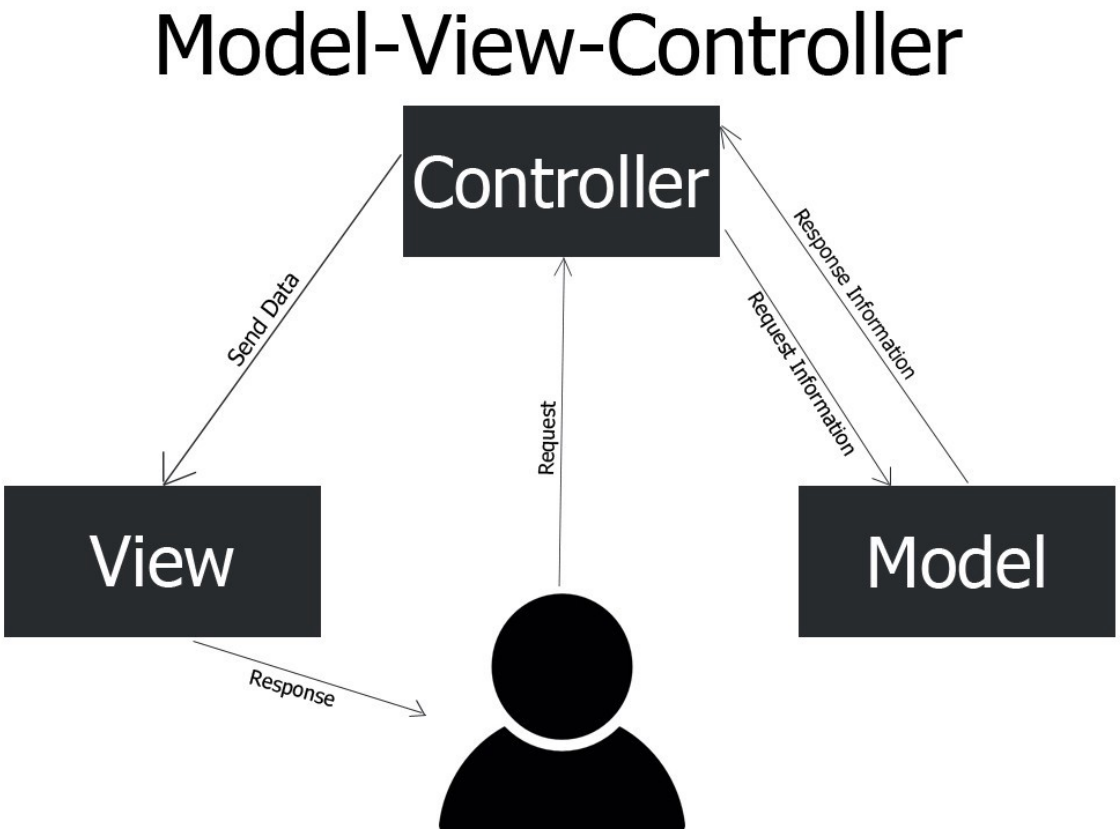


Рисунок 3.1 – Схема роботи структури MVC

Структура проекту має такий вигляд:

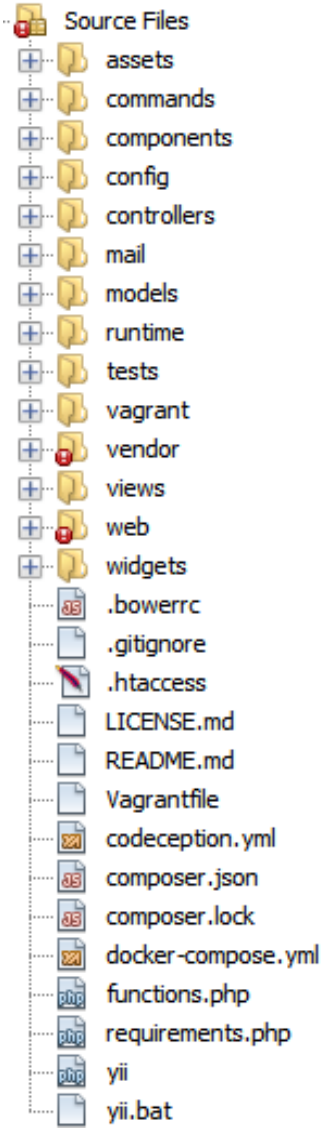


Рисунок 3.2 – Структура розробленого сайту

Відповідно у папці controllers створюються контролери, у models — моделі, а в views — вигляди. У папці web розміщені css та js файли. Config вміщує в себе конфігурацію — наприклад, дані для підключення до бази даних. У папці components можна створити власні віджети — це досить зручно, якщо, наприклад, на кожній сторінці у вас є меню категорій, але на деяких воно відрізняється.

Розглянемо схему бази даних. На ній зображені основні таблиці для сайту, в яких будуть зберігатися дані для наповнення каталогу товарів, — це таблиці Genres, Authors та Books.

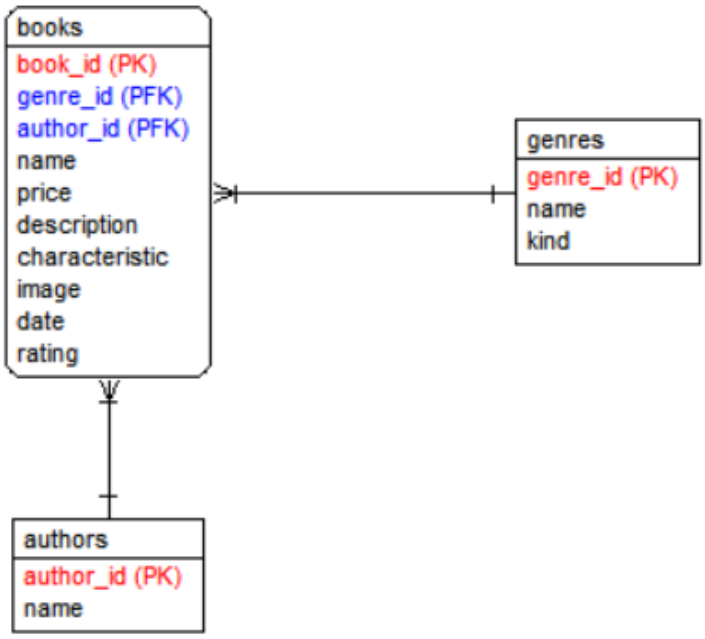


Рисунок 3.3 – ER-діаграма бази даних

В таблиці Genres представлені жанри (або ж іншими словами — категорії), за якими поділені книги. Таблиця Authors містить інформацію про авторів, а Books — про самі книги.

Таблиця 3.1 – Структура бази даних

| **Таблиця** | **Поле** | **Зміст** | **Тип** | **Ключі** |
|---|---|---|---|---|
| books | book_id | Ідентифікатор книги. | int | PK |
| | name | Назва книги. | varchar | |
| | price | Ціна книги. | float | |
| | description | Анотація до книги. | text | |
| | characteristic | Характеристика книги. | text | |
| | image | Книжкова обкладинка. | varchar | |
| | date | Дата, коли книга була викладена на сайт. | date | |
| | rating | Оцінка. | float | |
| genres | genre_id | Ідентифікатор жанру. | int | PK |
| | name | Назва жанру. | varchar | |
| | kind | Показує, чи є у жанру «батько». Існує для побудови дерева. | int | |
| authors | author_id | Ідентифікатор автора. | int | PK |
| | name | Ім’я автора. | varchar | |

Також в базі даних зберігається інформація про замовлення. Таблиці містять особисту інформацію замовника, кількість товарів, яка була замовлена, загальну суму замовлення та інформацію щодо книг.

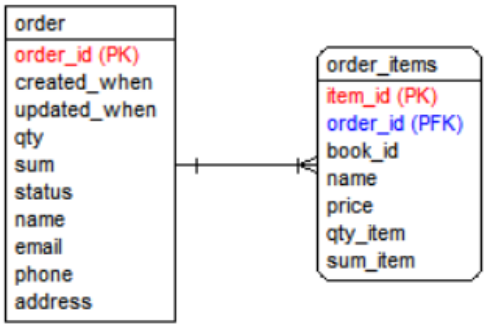


Рисунок 3.4 – ER-діаграма таблиць для інформації про замовлення

Таблиця 3.2 – Структура таблиць для замовлення

| **Таблиця** | **Поле** | **Зміст** | **Тип** | **Ключі** |
|---|---|---|---|---|
| order | order_id | Ідентифікатор замовлення. | int | PK |
| | created_when | Коли замовлення було створено. | date | |
| | updated_when | Коли замовлення було оновлено. | date | |
| | qty | Кількість замовлених товарів. | int | |
| | sum | Сума, на яку було зроблено замовлення. | float | |
| | status | Статус замовлення. | varchar | |
| | name | Ім’я замовника. | varchar | |
| | email | Електронна пошта замовника. | varchar | |
| | phone | Контактний номер телефону замовника. | varchar | |
| | address | Адреса замовника. | varchar | |
| order_items | item_id | Ідентифікатор товару. | int | PK |
| | book_id | Ідентифікатор книги. | int | |
| | name | Назва книги. | varchar | |
| | price | Ціна книги. | float | |
| | qty_item | Кількість замовлених копій. | int | |
| | sum_item | Сума за замовлені копії. | float | |

## 3.2 Програмна реалізація

На всіх сторінках сайту є меню, за допомогою якого можна перейти на головну сторінку, в каталог усіх товарів або в каталог конкретної категорії. Можна знайти інформацію про видавництво, корисну для користувача інформацію про доставку та оплату та поширені запитання. Також користувач має можливість здійснити пошук по сайту, перейти до корзини або в особистий кабінет.

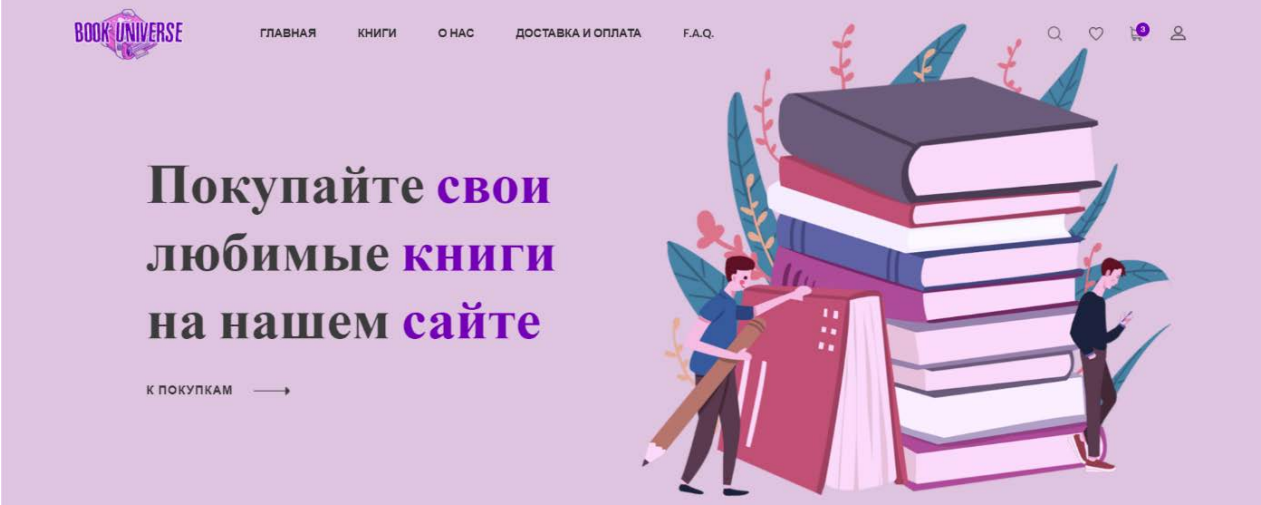


Рисунок 3.5 – Головна сторінка сайту

Головна сторінка зустрічає користувача «банером», в якому на даний момент є посилання на каталог усіх товарів. Але, наприклад, в майбутньому він може містити інформацію про акції або інші важливі новини.

Також на головній сторінці є розділ з найпопулярнішими товарами. Вони визначаються за оцінками користувачів. Інший розділ — новинки на сайті — це книги, які були останніми додані в каталог.

Такі розділи допомагають користувачеві легше орієнтуватися в усіх товарах. До того ж якісь книги з цих розділів можуть зацікавити користувача, і він швидше здійснить покупки, а отже з більшою ймовірністю згодом повернеться на сайт та зробить інші покупки.

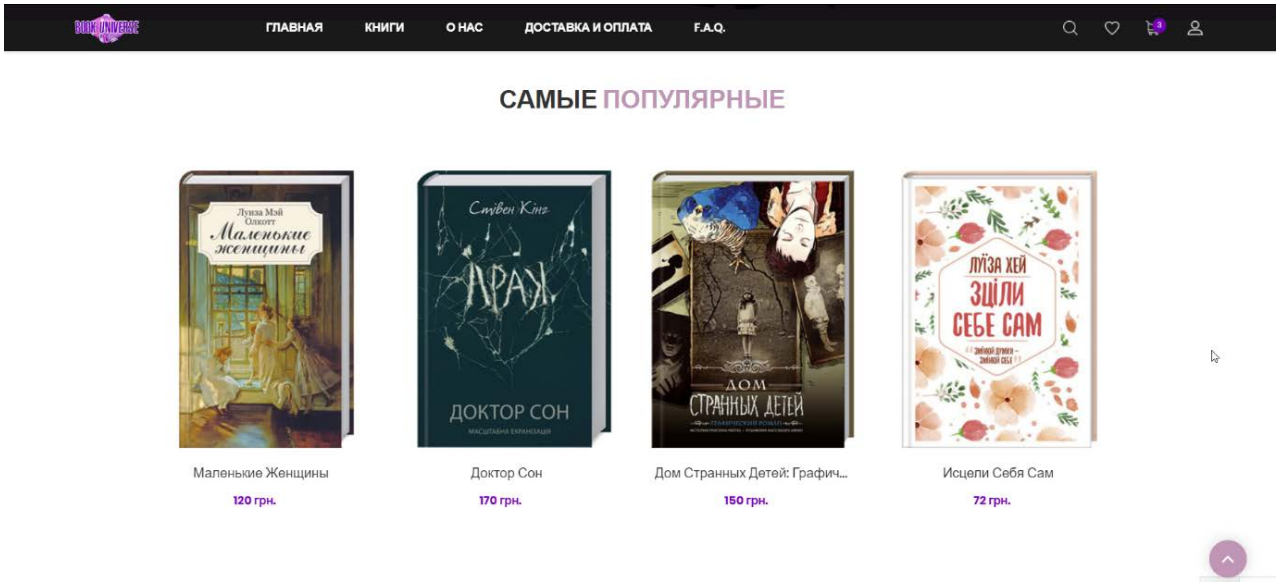


Рисунок 3.6 – Розділ з найпопулярнішими товарами на головній сторінці сайту

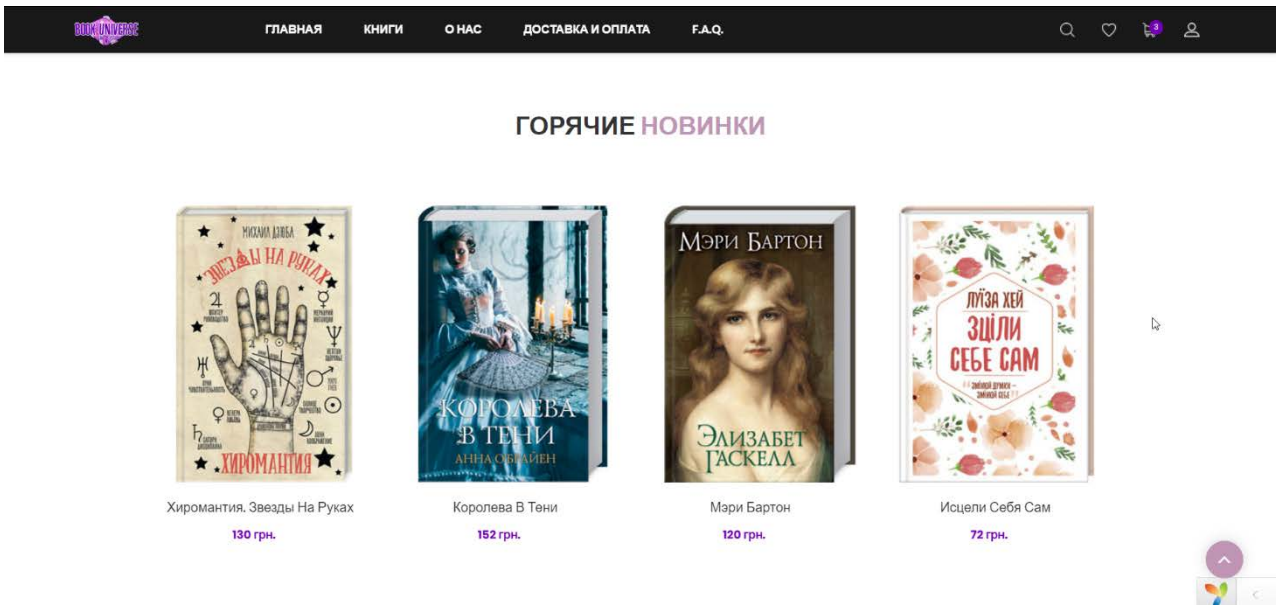


Рисунок 3.7 – Розділ з найновішими товарами на головній сторінці сайту

Футер сайту містить контактні дані видавництва та посилання та іншу корисну інформацію, яку можна знайти також у верхньому меню сайту, — це контактні дані, інформація про доставку та оплату, компанію в цілому та відповіді на запитання, які часто виникають у клієнтів.

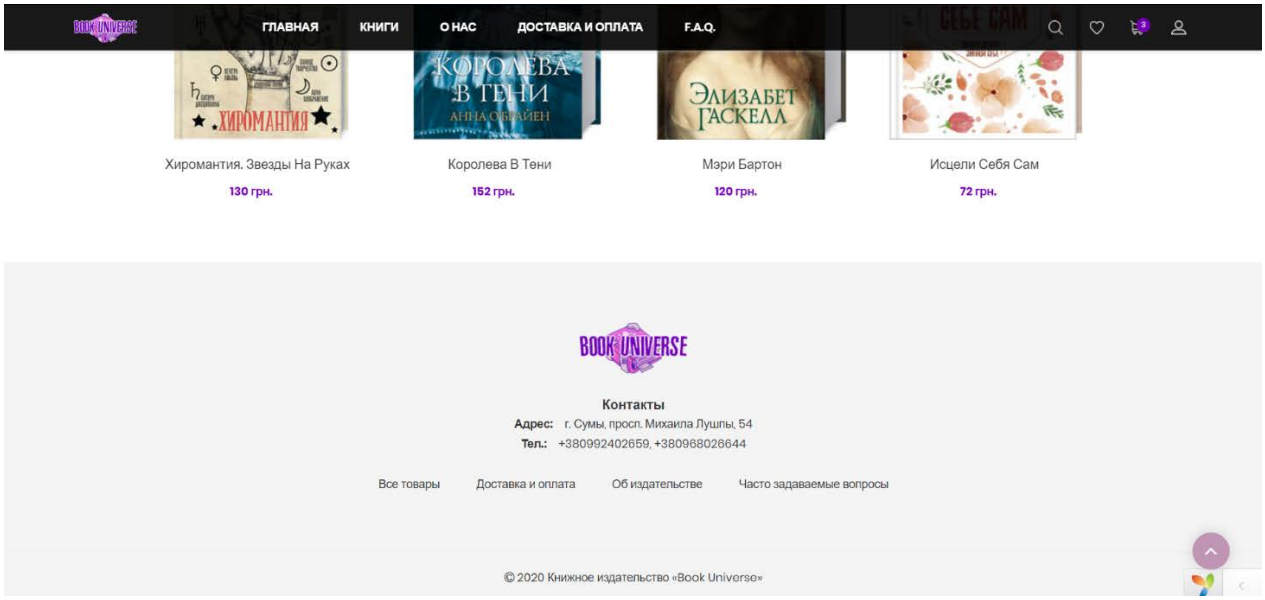


Рисунок 3.8 – Футер сайту

Обробка головної сторінки відбувається у GenresController.php:

```
class GenresController extends AppController {
    public function actionIndex() {
        $hits = Books::find()->orderBy(['rating' => SORT_DESC])-
>limit(6)->all();
        $latest = Books::find()->orderBy(['date' => SORT_DESC])-
>limit(6)->all();
        return $this->render('index', compact(['hits',
'latest']));
    }
    public function actionView($id) {
        $id = Yii::$app->request->get('id');
        $books = Books::find()->where(['genre_id' => $id])-
>all();
        $genres = Genres::find()->where(['genre_id' => $id])-
>all();
        return $this->render('view', compact(['books',
'genres']));
    }
    public function actionAllbooks() {
        $query = Books::find();
        $queryCount = $query->count();
        $pages = new Pagination(['totalCount' => $queryCount,
'pageSize' => 9]);
        $books = $query->offset($pages->offset)->limit($pages-
>limit)->all();
        return $this->render('allbooks', compact(['books',
'pages']));
    }

}
```

За допомогою фреймворка Yii2 досить легко посилати та обробляти запити в базу даних. В нашому випадку це сортування популярних та нових товарів. Цей контролер також обробляє сторінки для показу усіх товарів, показу товарів по категоріям.

У каталозі всіх товарів, власне, знаходяться усі книги. Їх можна сортувати за популярністю (спочатку популярні), часом (спочатку старі або нові) та ціною (спочатку дорогі або дешеві). Це знову ж таке допомагає користувачеві легше орієнтуватися на сайті. Особливо якщо у каталозі буде велика кількість книг — 500, 800, 1000 і т.д.

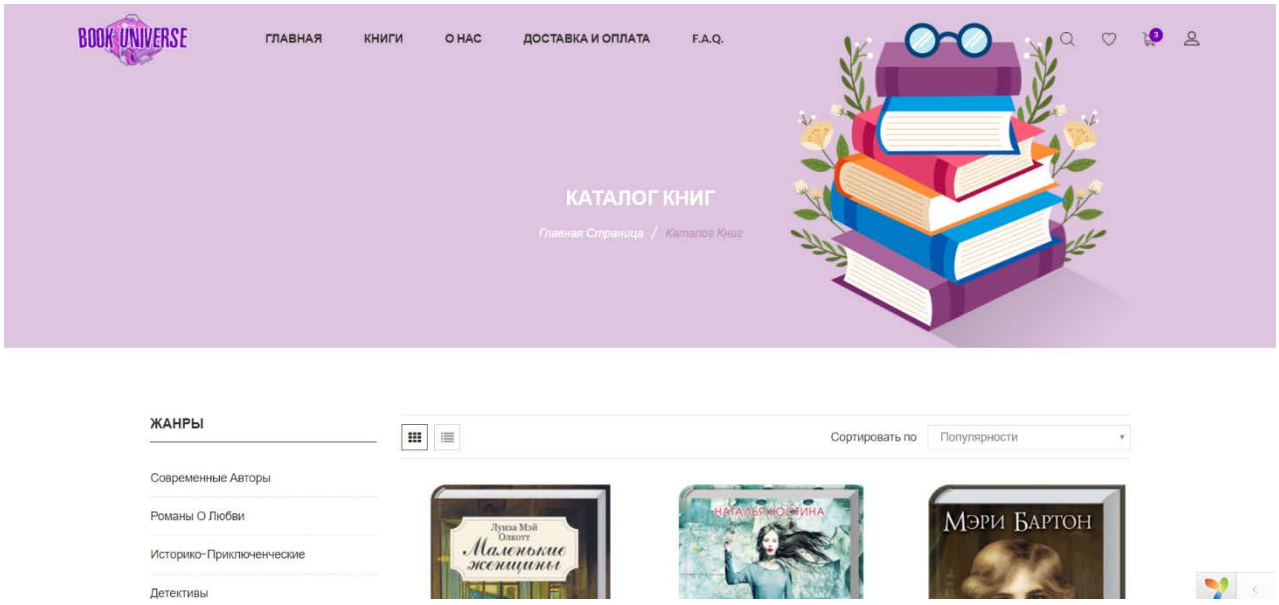


Рисунок 3.9 – Каталог усіх товарів

Є бокова панель, за допомогою якої можна перейти в будь–яку категорію за один клік. Категорії представляють собою різні книжкові жанри. Так користувачеві буде легше знайти цікаві йому товари.

Ця панель створена за допомогою кастомного віджета для меню — MenuWidget. Було вирішено зробити саме віджет, адже виведення категрій відбувається в декількох місцях — у верхньому меню та на сторінках з товарами. У view необхідно лише вставити рядок:

```
<?= app\components\MenuWidget::widget(['menutemplate' =>
'shop']) ?>
```

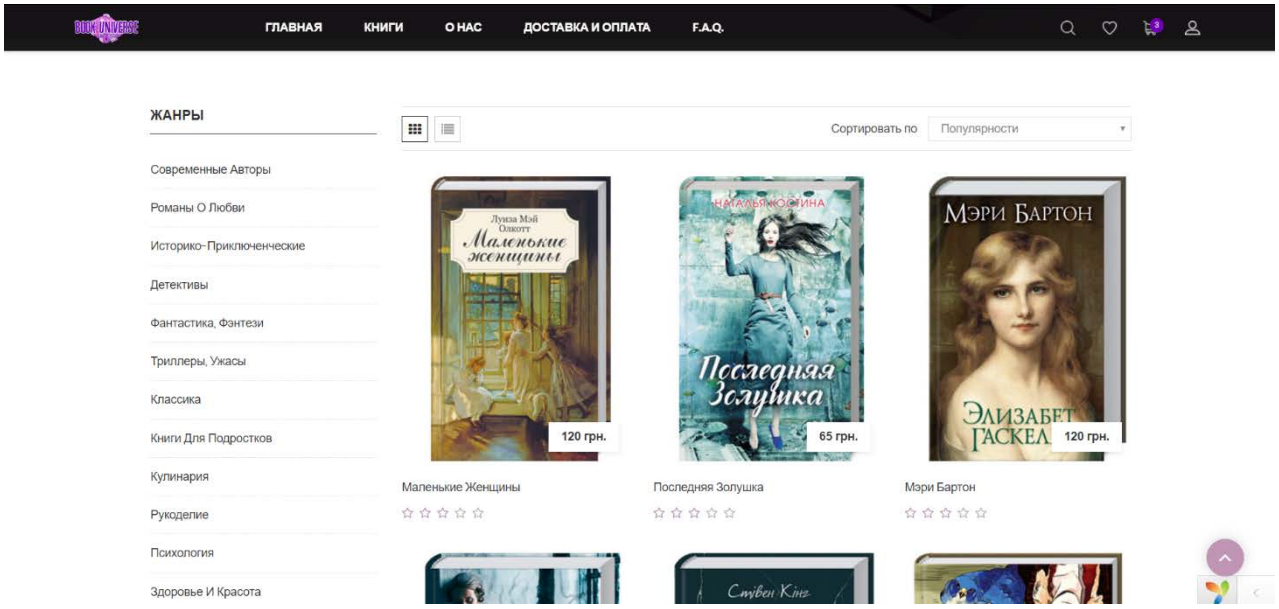


Рисунок 3.10 – Каталог усіх товарів (продовження)

Для каталогу існує два вигляда, адже різним користувачам зручно по–різному оглядати товари. Перший вигляд представляє собою «таблицю» з товарами, яка показує обкладинку, ціну, назву та оцінку. Другий — це список з товарами, який окрім попередньої інформації містить короткий опис та кнопку, що додає книгу в корзину.

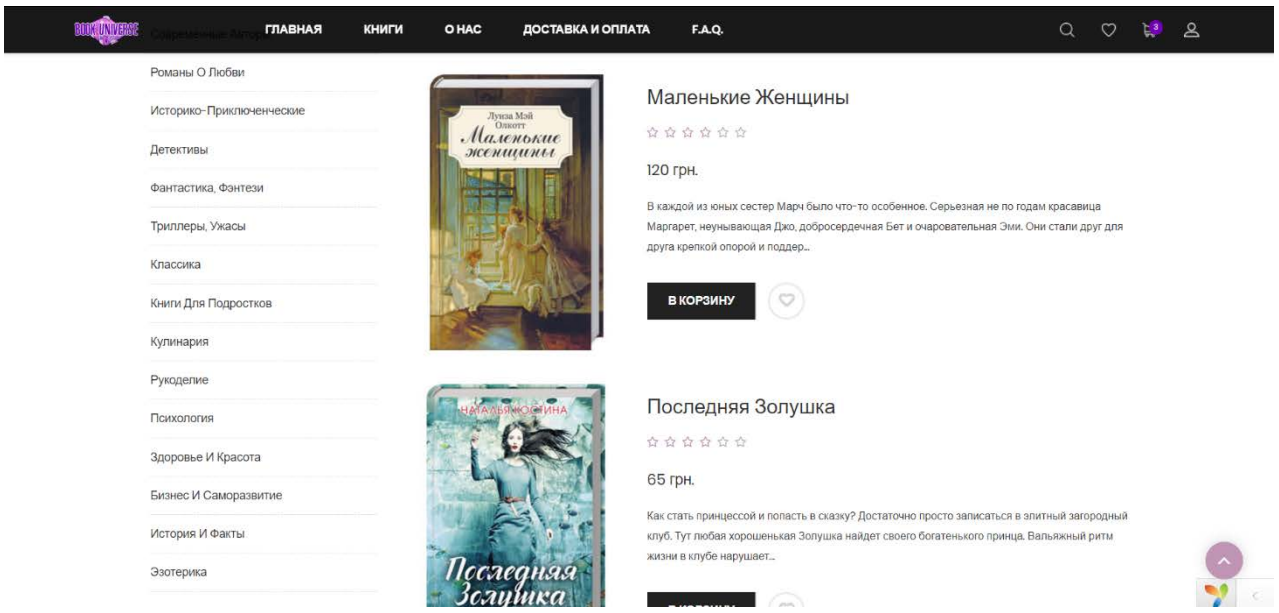


Рисунок 3.11 – Інший вигляд каталогу товарів

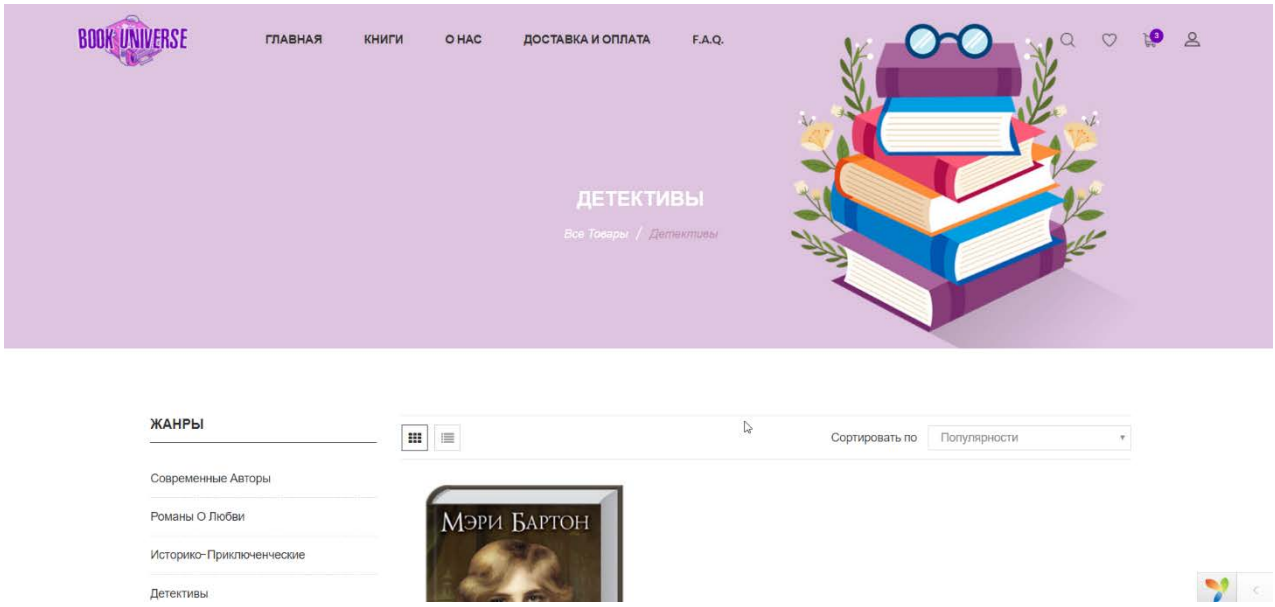


Рисунок 3.12 – Розподіл товарів за категоріями (жанрами)

Для підключення до баз даних були використані моделі Genres та Books, які віповідають за категорії та товари відповідно. Між базами даних є зв’язок — в одній категорії може міститися декілька товарів.

Модель Genres.php:

```
class Genres extends ActiveRecord {

    public static function tableName() {
        return 'genres';
    }

    public function getBooks() {
        return $this->hasMany(Books::className(), ['genre_id' =>
'genre_id']);
    }

}
```

Модель Books.php:

```
class Books extends ActiveRecord {

    public static function tableName() {
        return 'books';
    }
    public function getGenres() {
        return $this->hasOne(Genres::className(), ['genre_id' =>
'genre_id']);
    }

}
```

При перегляді одного товару також була збережена панель з категоріями. Сама сторінка містить реальний вигляд книги, таку основну інформацію як назва, автор, наявність, ціна, а також кнопку, за допомогою якої можна додати книгу до корзини.

За відображення одного товару відповідає контролер BooksController:

```
class BooksController extends AppController {

    public function actionSingle ($id) {
        $id = Yii::$app->request ->get('id');
        $single = Books::find()->where(['book_id' => $id])-
>one();
        $related = Books::find()->where([
            'and',
            ['genre_id' => $single['genre_id']],
            ['not in', 'book_id', $id],
                ])->all();
        return $this->render('single', compact(['single',
'related']));
    }

}
```

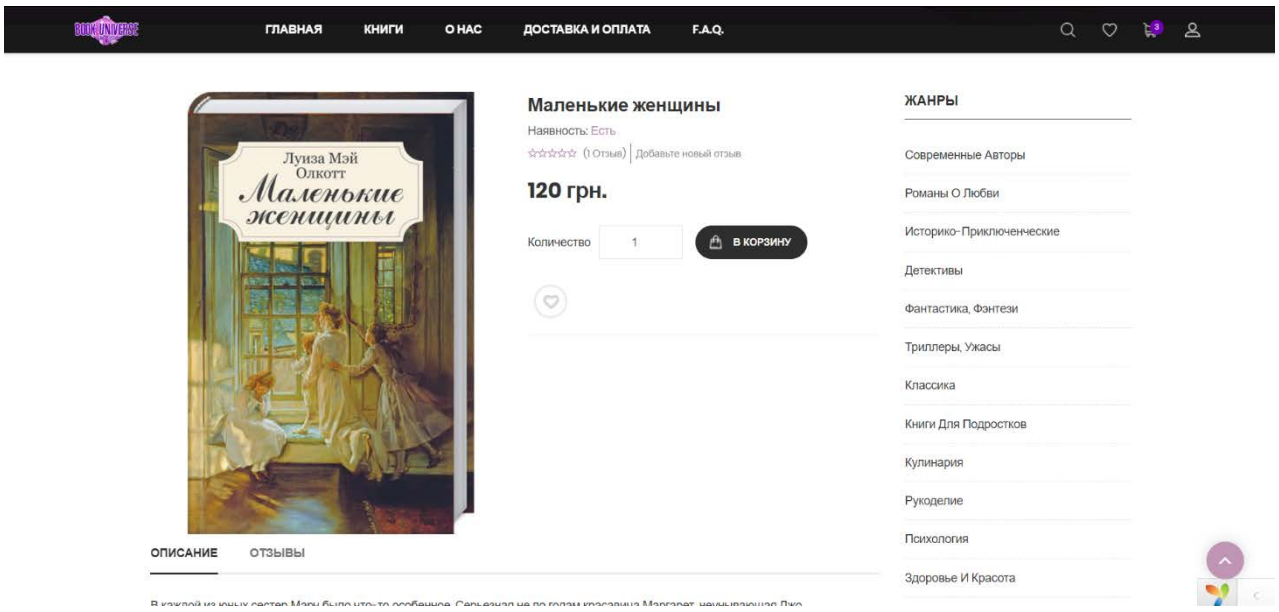


Рисунок 3.13 – Вигляд окремого товару

Знизу знаходиться опис (анотація) та характеристика книги. Чим більше інформації буде написано про товар одразу — тим менше запитань після відвідування сайту буде у користувача.

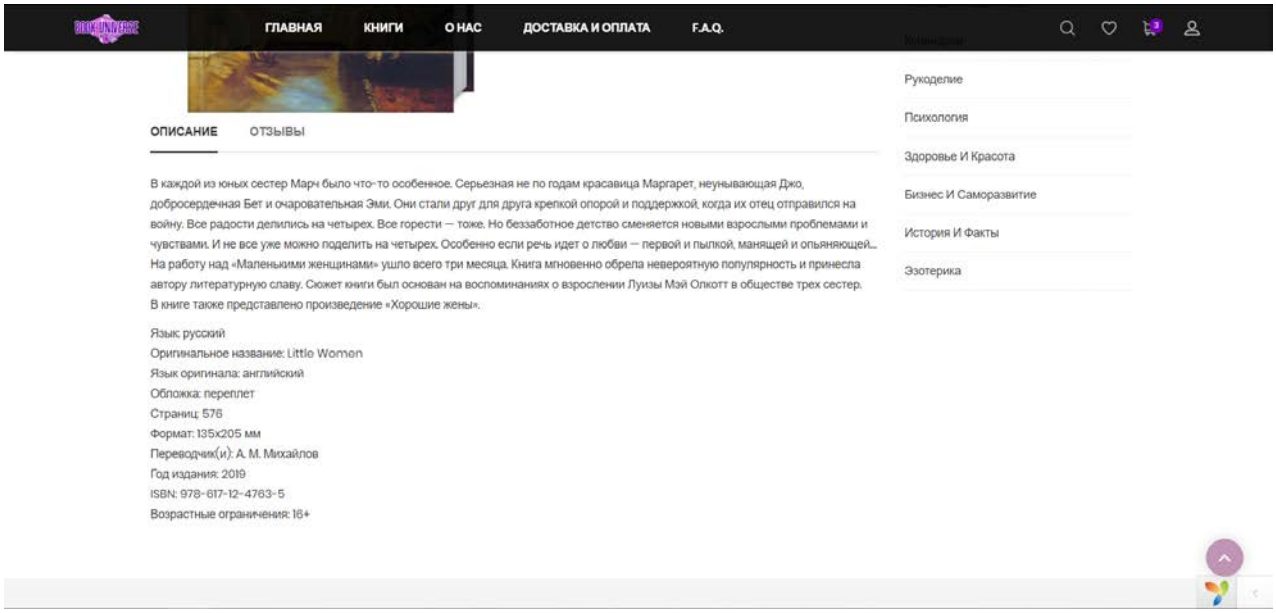


Рисунок 3.14 – Опис товару

Також на сторінці окремої книги є розділ зі схожими по жанру книгами. Це дозволяє більше зацікавити користувача і підвищує шанс, що він здійснить покупку, якщо легко знайде потрібний йому товар. Наприклад, йому сподобалася детективна книга, і знизу він бачить схожі на неї. Він перегляне їх та можливо знайде ще декілька товарів, які йому сподобаються. Придбає їх одразу або намітить майбутню покупку, а отже ще повернеться на сайт.

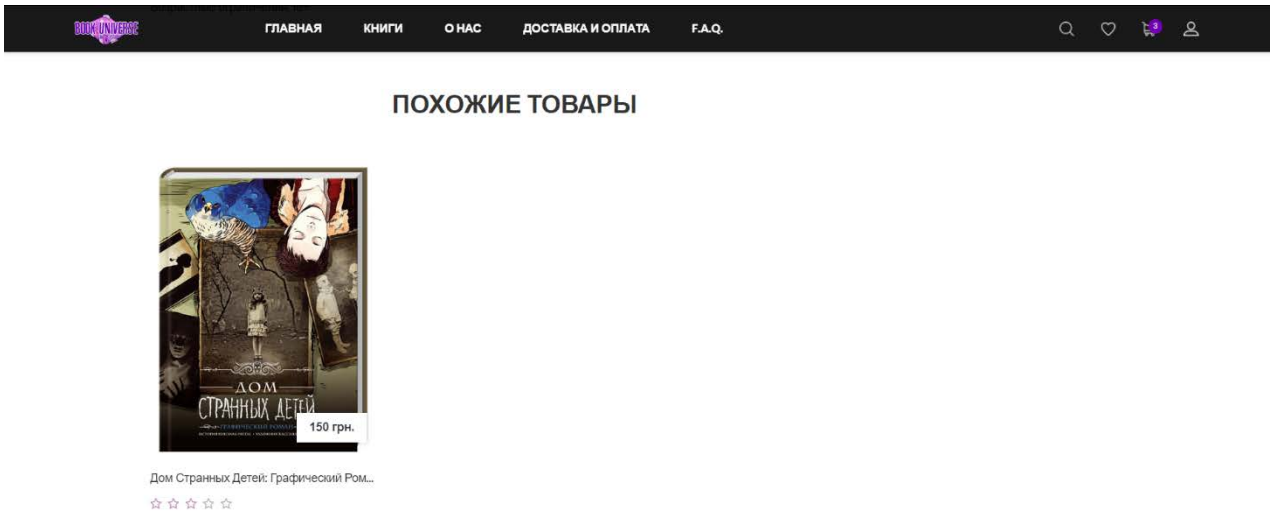


Рисунок 3.15 – Розділ схожих товарів

Щоб користувачеві було легше перейти в категорії з головної сторінки, у верхньому меню є випадаюче меню із жанрами.

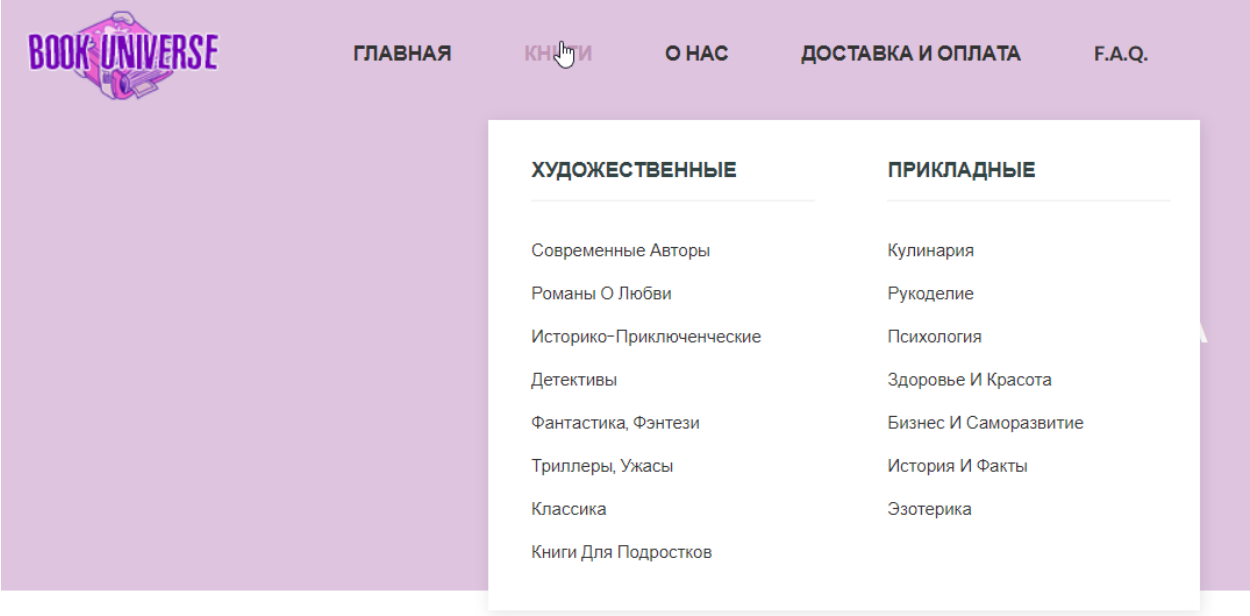


Рисунок 3.16 – Категорії товарів в головному меню

Корзина містить основну інформацію про додані товари: обкладинку, щоб наглядно було видно що саме користувач купує, назву, ціну за один товар, кількість замовлених копій та суму.

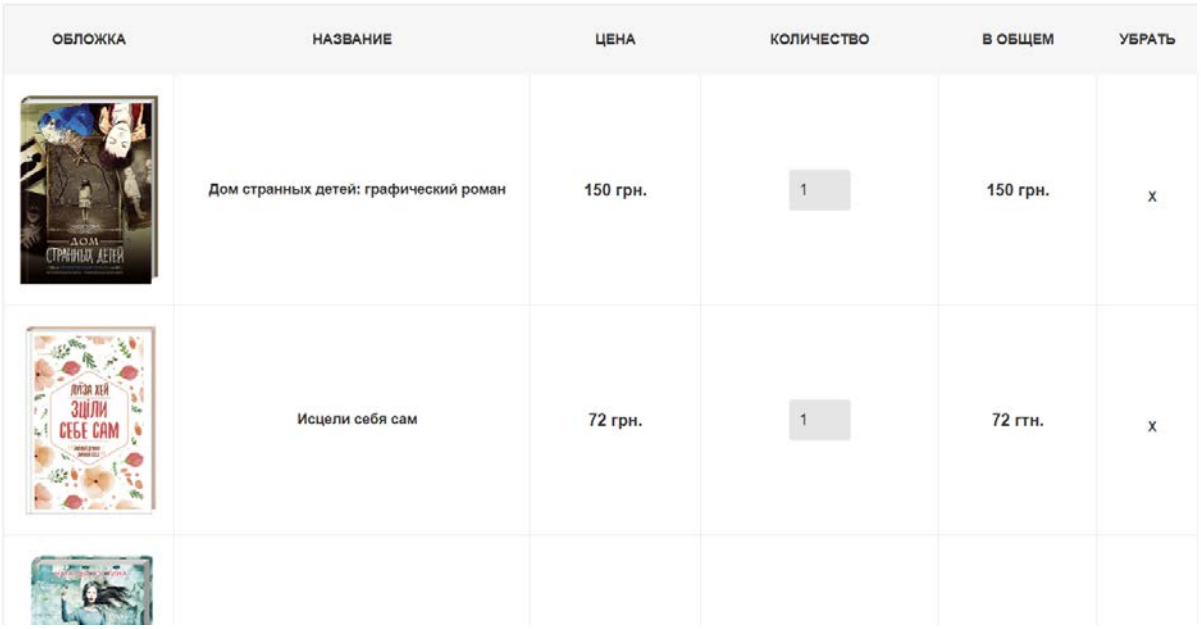


Рисунок 3.17 – Корзина з доданими товарами

А також суму за всі товари, вартість доставки та загальну суму замовлення.

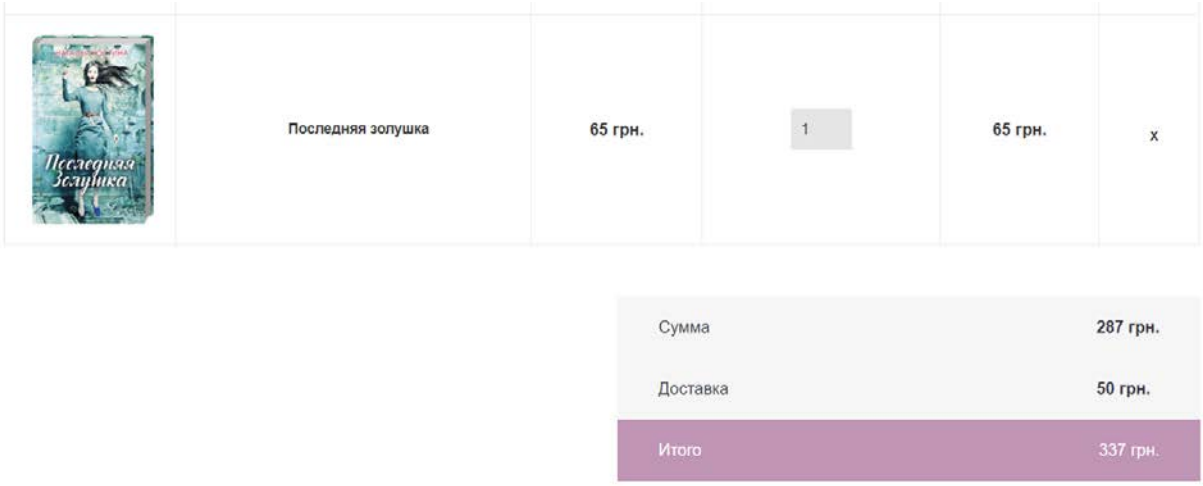


Рисунок 3.18 – Корзина з доданими товарами (продовження)

При додаванні товару в корзину висвічується модальне вікно:

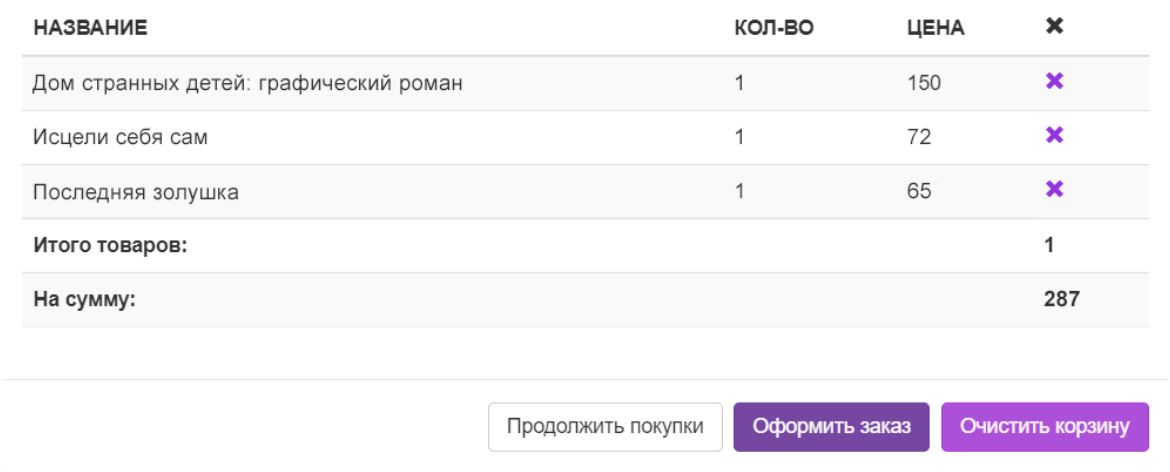


Рисунок 3.19 – Модальне вікно корзини

Вікно було сформоване за допомогою фреймворку Bootstrap:

```
<?php
        \yii\bootstrap\Modal::begin([
            'header' => '<h2>Корзина</h2>',
            'id' => 'cart',
            'size' => 'modal-lg',
            'footer' => '<button type="button" class = "btn btn-
default" data-dismiss="modal">Продолжить покупки</button>'
            . '<button type="button" class = "btn btn-
primary">Оформить заказ</button>'
            . '<button type="button" class = "btn btn-danger"
```

```
onclick="clearCart()">Очистить корзину</button>',
        ]);

        \yii\bootstrap\Modal::end();
        ?>
```

Для того, щоб оформити замовлення, необхідно авторизуватися:

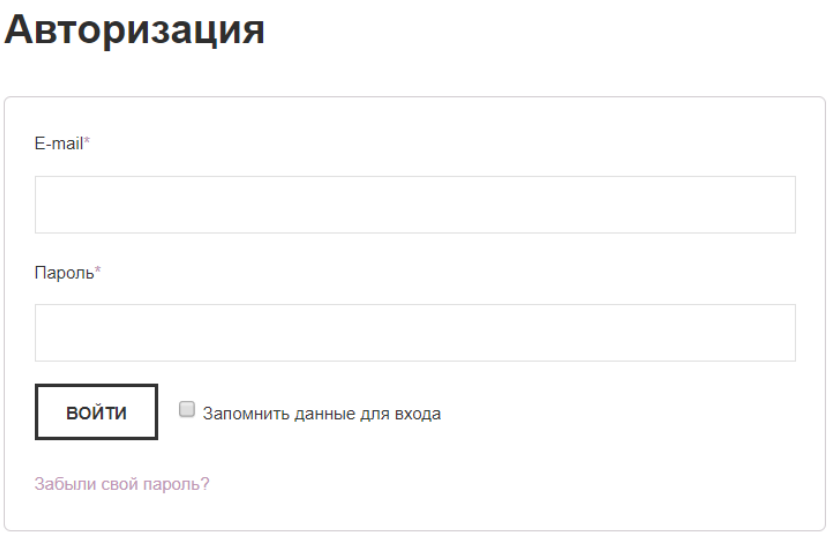


Рисунок 3.20 – Вікно авторизації

Або ж зареєструватися, якщо користувач — новий:

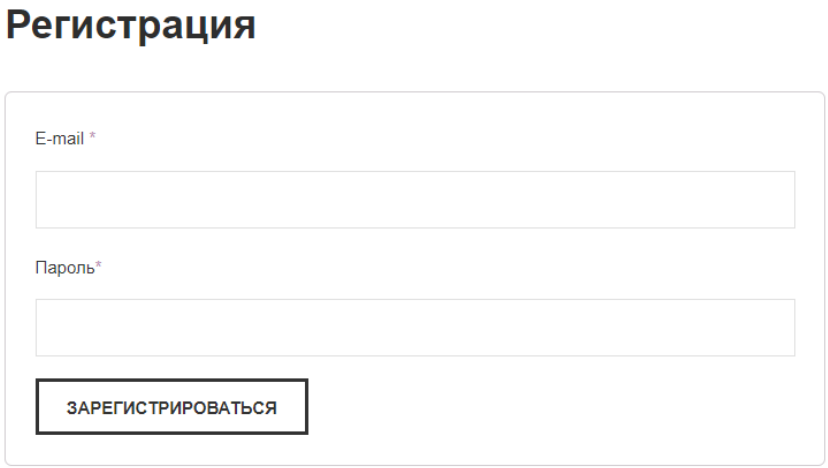


Рисунок 3.21 – Вікно реєстрації

При реєстрації користувачеві на електронну пошту надходить посилання з підтвердженням реєстрації.

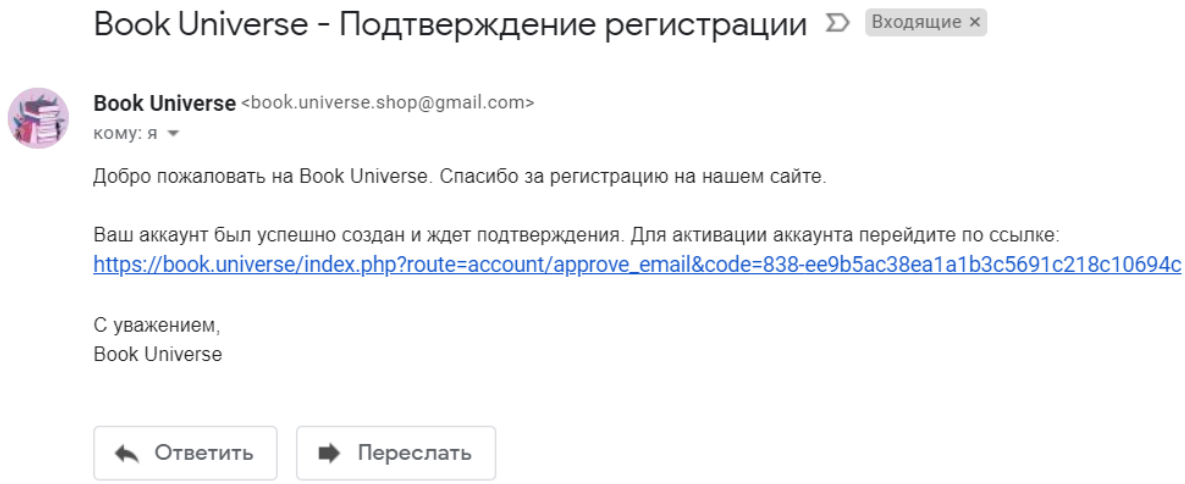


Рисунок 3.22 – Підтвердження реєстрації

На сайті є новинна розсилка: зареєстровані користувачі отримують листи з новинами та акціями. Також якщо користувач не робив покупки більше місяця, йому приходить лист з нагадуванням.

Вона реалізована за допомогою фреймворка Yii2 та розширення до нього yii2–swiftmailer.

Приклад функції для підтвердження реєстрації:

```
public function sentConfirmationLetter(User $user)
    {

        $email = $user->email;

        $letter = Yii::$app->mailer
            ->compose(
                ['html' => 'confirmationLetterHtml', 'text' => '
confirmationLettertext'],
                ['user' => $user])
            ->setTo($email)
            ->setFrom(Yii::$app->params['bookuniverseshop'])
            ->setSubject('Book Universe – Подтверждение
регистрации')
            ->send();

    }
```

# ВИСНОВКИ

У даній роботі на основі аналізу найпоширеніших методів рішення завдання були обрані HTML, CSS, JavaScript та фреймворк Bootstrap для фронт–енду; мова програмування PHP з фреймоврком Yii2 для бек–енду.

Було створено інформаційний ресурс для книжкового видавництва з реалізацією маркетингового алгоритму. Увесь функціонал, що заявлявся, був реалізований. Ресурс є зручним та зрозумілим у використанні, привабливим для користувача. Він адаптивний, працює у всіх сучасних популярних веб–браузерах. Маркетинговий алгоритм для залучення та збереження клієнтів було застосовано.

# СПИСОК ЛІТЕРАТУРИ


1. Браун И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. – Санкт-Петербург: Питер, 2017. – 214 с.
2. Гринберг С. UX-дизайн. Идея – эскиз – воплощение. – Санкт-Петербург: Питер, 2015. – 250 с.
3. Хопкинс К. PHP Быстрый старт. – Москва: Эксмо, 2015. – 110с.
4. Скляр Д. Изучаем PHP 7: руководство по созданию интерактивных веб-сайтов. – Санкт-Петербург: ООО «Альфа-книга», 2017. – 464 с.
5. Гвоздева Т., Баллод Б. Проектирование информационных систем. – Санкт-Петербург: ООО «Издательство Лань», 2018. – 156 с.
6. Кириченко А., Хрусталев А. HTML5 + CSS3. Основы современного WEB-дизайна. – Санкт-Петербург: Наука и Техника, 2018. – 354 с.
7. МакГрат М. PHP7 для начинающих с пошаговыми инструкциями. – Москва: Издательство «Э», 2018. – 256 с.
8. Шварц Б., Зайцев П., Ткаченко В. MySQL по максимуму. 3-е издание. – Санкт-Петербург: Питер Пресс, 2018. – 864 с.
9. Фаррелл Б. Веб-компоненты в действии. – Москва: Пресс, 2020. – 462 с.
10. Мейер Э. CSS. Карманный справочник. 5-е издание. – Москва: Издательство «Вильямс», 2020. – 208 с.
11. Круг С. Не заставляй меня думать. Веб-юзабилити и здравый смысл. 3-е издание. – Москва: Эксмо, 2019. – 256 с.
12. Макдональд М. Веб-разработка. Исчерпывающее руководство. – Санкт-Петербург: Питер Пресс, 2017. – 640 с.
13. Дунаев В. HTML, скрипты и стили. – СпБ: БХВ, 2015. – 816 с.
14. Критерії оцінки сайтів [Електронний ресурс]. – Режим доступу: http://www.proview.ru/criteria_evaluating_sites/
15. Полное руководство для фреймворка Yii2 [Електронний ресурс]. – Режим доступу: https://yiiframework.com.ua/ru/doc/guide/2/

# ДОДАТОК

## Додаток А

CartController.php:

```
<?php

namespace app\controllers;

use app\models\Genres;
use app\models\Books;
use app\models\Cart;
use app\models\Order;
use app\models\OrderItems;
use Yii;

class CartController extends AppController {


    public function actionAdd() {
        $id = Yii::$app->request->get('id');
        $books = Books::findOne($id);
        if (empty($books))
            return false;
        $session = Yii::$app->session;
        $session->open();
        $cart = new Cart();
        $cart->addToCart($books);
        $this->layout = false;
        return $this->render('cart', compact(['session']));
    }


    public function actionClear() {
        $session = Yii::$app->session;
        $session->open();
        $session->remove('cart');
        $session->remove('cart.qty');
        $session->remove('cart.sum');
        $this->layout = false;
        return $this->render('cart', compact(['session']));
    }


    public function actionDelete() {
        $id = Yii::$app->request->get('id');
        $session = Yii::$app->session;
        $session->open();
        $cart = new Cart();
```

```
        $cart->recalc($id);
        $this->layout = false;
        return $this->render('cart', compact(['session']));
    }


    public function actionShow() {
        $session = Yii::$app->session;
        $session->open();
        $this->layout = false;
        return $this->render('cart', compact(['session']));
    }


    public function actionView() {
        $session = Yii::$app->session;
        $session->open();
        $this->setMeta('Корзина');
        $order = new Order();
        return $this->render('view', compact(['session',
'order']));
    }

}
```

JavaScript-код для корзини:

```
    function showCart(cart) {
        $('#cart .modal-body').html(cart);
        $('#cart').modal();
    };


    function getCart() {
        alert(123);
    };


    $('#cart .modal-body').on('click', '.del-item',  function()
{
        var id = $(this).data('id');
        $.ajax({
            url: 'cart/delete',
            data: {id: id},
            type: 'GET',
            success: function (res) {
                if(!res) alert('Error');
                showCart(res);
            },
            error: function () {
```

```
            alert('Error');
        }
    });
});


function clearCart(){

    $.ajax({
        url: 'cart/clear',
        type: 'GET',
        success: function (res) {
            if(!res) alert('Error');
            showCart(res);
        },

        error: function () {
            alert('Error');
        }
    });
};


$('.add-to-cart').on('click', function (e) {

    e.preventDefault();

    var id = $(this).data('id');

    $.ajax({
        url: '/web/index.php?r=cart%2Fadd',
        data: {id: id},
        type: 'GET',
        success: function (res) {
            if(!res) alert('Error');
            //console.log(res);
            showCart(res);
        },

        error: function () {
            alert('Error');
        }

    });

});
```

# Додаток Б

SiteController.php:

```
<?php

namespace app\controllers;

use Yii;
use yii\filters\AccessControl;
use yii\web\Controller;
use yii\web\Response;
use yii\filters\VerbFilter;
use app\models\LoginForm;
use app\models\ContactForm;

class SiteController extends Controller {

    public function behaviors() {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['logout'],
                'rules' => [
                    [
                        'actions' => ['logout'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'logout' => ['post'],
                ],
            ],
        ];
    }

    public function actions() {
        return [
            'error' => [
                'class' => 'yii\web\ErrorAction',
            ],
            'captcha' => [
                'class' => 'yii\captcha\CaptchaAction',
                'fixedVerifyCode' => YII_ENV_TEST ? 'testme' :
null,
            ],
```

```
        ];
    }

    public function actionIndex() {
        return $this->render('index');
    }

    public function actionLogin() {
        if (!Yii::$app->user->isGuest) {
            return $this->goHome();
        }

        $model = new LoginForm();
        if ($model->load(Yii::$app->request->post()) && $model-
>login()) {
            return $this->goBack();
        }

        $model->password = '';
        return $this->render('login', [
                    'model' => $model,
        ]);
    }

    public function actionLogout() {
        Yii::$app->user->logout();

        return $this->goHome();
    }

    public function actionContact() {
        $model = new ContactForm();
        if ($model->load(Yii::$app->request->post()) && $model-
>contact(Yii::$app->params['adminEmail'])) {
            Yii::$app->session-
>setFlash('contactFormSubmitted');

            return $this->refresh();
        }
        return $this->render('contact', [
                    'model' => $model,
        ]);
    }

    public function actionSignup() {
        $form = new SignupForm();
        if ($form->load(Yii::$app->request->post()) && $form-
>validate()) {
            $signupService = new SignupService();

            try {
```

```
                $user = $signupService->signup($form);
                Yii::$app->session->setFlash('success',
'Проверьте вашу почту для подтверждения регистрации');
                $signupService->sentEmailConfirm($user);
                return $this->goHome();
            } catch (\RuntimeException $e) {
                Yii::$app->errorHandler->logException($e);
                Yii::$app->session->setFlash('error', $e-
>getMessage());
            }
        }

        return $this->render('signup', [
                    'model' => $form,
        ]);
    }

    public function actionAbout() {
        return $this->render('about');
    }

}
```

SignupService.php:

```
<?php

namespace common\services\auth;

use Yii;
use common\entities\User;
use common\forms\auth\SignupForm;

class SignupService
{

    public function signup(SignupForm $form)
    {
        $user = new User();
        $user->username = $form->username;
        $user->generateAuthKey();
        $user->setPassword($form->password);
        $user->email = $form->email;
        $user->email_confirm_token = Yii::$app->security-
>generateRandomString();
        $user->status = User::STATUS_WAIT;

        if(!$user->save()){
            throw new \RuntimeException('Saving error.');
        }
```

```
        return $user;
    }


    public function sentEmailConfirm(User $user)
    {
        $email = $user->email;

        $sent = Yii::$app->mailer
            ->compose(
                ['html' => 'user-signup-comfirm-html', 'text' =>
'user-signup-comfirm-text'],
                ['user' => $user])
            ->setTo($email)
            ->setFrom(Yii::$app->params['adminEmail'])
            ->setSubject('Confirmation of registration')
            ->send();

        if (!$sent) {
            throw new \RuntimeException('Sending error.');
        }
    }


    public function confirmation($token): void
    {
        if (empty($token)) {
            throw new \DomainException('Empty confirm token.');
        }

        $user = User::findOne(['email_confirm_token' =>
$token]);
        if (!$user) {
            throw new \DomainException('User is not found.');
        }

        $user->email_confirm_token = null;
        $user->status = User::STATUS_ACTIVE;
        if (!$user->save()) {
            throw new \RuntimeException('Saving error.');
        }

        if (!Yii::$app->getUser()->login($user)){
            throw new \RuntimeException('Error
authentication.');
        }
    }

}
```

# Додаток В

MenuWidget.php:

```
class MenuWidget extends Widget {

    public $menutemplate;
    public $data;
    public $tree;
    public $menuHtml;
    public $menuchilds;

    public function init() {
        parent::init();
        if ($this->menutemplate === null)
            $this->menutemplate = 'fiction';
        $this->menutemplate .= '.php';
    }
    public function run() {
        $this->data = Genres::find()->indexBy('genre_id')-
>asArray()->all();
        $this->tree = $this->getTree();
        $this->menuHtml = $this->getMenuHtml($this->tree);
        return $this->menuHtml;
    }
    protected function getTree() {
        $tree = [];
        foreach ($this->data as $id => &$node) {
            if (!$node['kind'])
                $tree[$id] = &$node;
            else
                $this-
>data[$node['kind']]['childs'][$node['genre_id']] = &$node;
        }
        return $tree;
    }
    protected function getMenuHtml($tree) {
        $str = '';
        foreach ($tree as $genre) {
            $str .= $this->genToTemplate($genre);
        }
        return $str;
    }
    protected function genToTemplate($genre) {
        ob_start();
        include __DIR__ . '/menutemplates/' . $this-
>menutemplate;
        return ob_get_clean();
    }
    protected function getMenuHtmlChilds($tree) {
```

```
        $this->menuchilds = 'menuchilds.php';
        $str = '';
        foreach ($tree as $genre) {
            $str .= $this->genToTemplateChilds($genre);
        }
        return $str;
    }
    protected function genToTemplateChilds($genre) {
        ob_start();
        include __DIR__ . '/menutemplates/' . $this->menuchilds;
        return ob_get_clean();
    }

}
```

Template fiction.php:

```
<ul class="item item02">
    <li class="title"><?= $genre['name'] ?></li>
    <?php if (isset($genre['childs'])): ?>
    <?= $this->getMenuHtmlChilds($genre['childs']) ?>
    <?php endif; ?>
</ul>
```

Template menuchilds.php:

```
<li>
    <a href="<?= \yii\helpers\Url::to(['genres/view',
'id'=>$genre['genre_id']]) ?>">
        <?= $genre['name'] ?>
    </a>
</li>
```

Template mobilemenu.php:

```
<?php if (isset($genre['childs'])): ?>
    <li><a><?= $genre['name'] ?></a>
        <?php if (isset($genre['childs'])): ?>
            <ul>
               <?= $this->getMenuHtmlChilds($genre['childs']) ?>
            </ul>
        <?php endif; ?>
    </li>
<?php endif; ?>
```

Template shop.php:

```
<?php if (isset($genre['childs'])): ?>
    <?= $this->getMenuHtmlChilds($genre['childs']) ?>
<?php endif; ?>
```

# Додаток Г

GenresController.php:

```
class GenresController extends AppController {

    public function actionIndex() {
        $hits = Books::find()->orderBy(['rating' => SORT_DESC])-
>limit(6)->all();
        $latest = Books::find()->orderBy(['date' => SORT_DESC])-
>limit(6)->all();
        return $this->render('index', compact(['hits',
'latest']));
    }

    public function actionView($id) {
        $id = Yii::$app->request->get('id');
        $books = Books::find()->where(['genre_id' => $id])-
>all();
        $genres = Genres::find()->where(['genre_id' => $id])-
>all();
        return $this->render('view', compact(['books',
'genres']));
    }

    public function actionAllbooks() {
        $query = Books::find();
        $queryCount = $query->count();
        $pages = new Pagination(['totalCount' => $queryCount,
'pageSize' => 9]);
        $books = $query->offset($pages->offset)->limit($pages-
>limit)->all();
        return $this->render('allbooks', compact(['books',
'pages']));
    }
}
```

Genres.php:

```
class Genres extends ActiveRecord {

    public static function tableName() {
        return 'genres';
    }

    public function getBooks() {
        return $this->hasMany(Books::className(), ['genre_id' =>
'genre_id']);
    }
}
```

# Додаток Д

BooksController.php:

```
<?php

namespace app\controllers;

use app\models\Genres;
use app\models\Books;
use Yii;

class BooksController extends AppController {

    public function actionSingle ($id) {

        $id = Yii::$app->request ->get('id');

        $single = Books::find()->where(['book_id' => $id])-
>one();

        $related = Books::find()->where([
            'and',
            ['genre_id' => $single['genre_id']],
            ['not in', 'book_id', $id],
                ])->all();
        return $this->render('single', compact(['single',
'related']));

    }

}
```

Books.php:

```
<?php
namespace app\models;
use yii\db\ActiveRecord;

class Books extends ActiveRecord {

    public static function tableName() {
        return 'books';
    }

    public function getGenres() {
        return $this->hasOne(Genres::className(), ['genre_id' =>
'genre_id']);
    }

}
```