МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВИЙ ПРОЕКТ**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Web-технології»

на тему:

**«**Розробка CMS системи інтернет-магазину книг**»**

студента II курсу групи ІПЗ-20-4

спеціальності 121 «Інженерія програмного забезпечення»

Кормиша Романа Івановича

(прізвище, ім’я та по-батькові)

Керівник: Чижмотря О.Г .

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ .

(підпис) (прізвище та ініціали) \_\_\_\_\_\_\_\_\_\_\_\_\_\_ .

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ .

(підпис) (прізвище та ініціали)

Житомир – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

В.о зав кафедри ІПЗ

\_\_\_\_\_\_\_\_\_\_\_\_ А.В Морозов

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2021 р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Кормишу Роману Івановичу

1. Тема роботи: Розробка інтернет-магазину книг

керівник роботи: старший викладач кафедри ІПЗ, Чижмотря Олена Генадіївна .

1. Строк подання студентом: “ ” 2021р.
2. Вихідні дані до роботи: Розробити інтернет-магазин книг – розробити та реалізувати систему адміністрування для сайту
3. Зміст розрахунково-пояснювальної записки (перелік питань, що підлягають розробці)
   * + 1. Постановка завдання
       2. Аналіз аналогічних розробок
       3. Алгоритми роботи програми

4. Опис роботи програми

5. Програмне дослідження

1. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)

1. Презентація до КР\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Посилання на репозиторій: https://gitlab.com/2020-2024/ipz-20-4/kormysh-roman/course-work-2.git

1. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1,2 | Чижмотря О.Г., ст. викладач каф. ІПЗ |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “ ” 2021 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Примітки |
| 1 | Постановка задачі |  |  |
| 2 | Пошук, огляд та аналіз аналогічних розробок |  |  |
| 3 | Формулювання технічного завдання |  |  |
| 4 | Опрацювання літературних джерел |  |  |
| 5 | Проектування структури |  |  |
| 6 | Написання програмного коду |  |  |
| 7 | Відлагодження |  |  |
| 8 | Написання пояснювальної записки |  |  |
| 9 | Захист |  |  |

**Студент***\_\_\_\_\_\_\_\_\_\_\_\_\_\_* Кормиш Р.І

(підпис) (прізвище та ініціали)

**Керівник проекту** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чижмотря О.Г

(підпис) (прізвище та ініціали)

**РЕФЕРАТ**

Пояснювальна записка до курсового проекта на тему «Розробка інтернет магазину книг» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 40 сторінках друкованого тексту.

Пояснювальна записка має 34 сторінки додатків. Список використаних джерел містить 10 найменувань і займає 1 сторінку. В роботі наведено 25 рисунів. Загальний обсяг роботи – 74 сторінок.

У першому розділі було обґрунтовано створення сайту на тему Розробка інтернет-магазину книг.

У другому розділі проведено проектування і розробка сайту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи при створенні сайту на тему «Розробка інтернет-магазину книг».

У додатку представлений лістинг розробленого програмного продукту.

Ключові слова:

MYSQL, JAVASCRIPT, PYTHON, FAST API, VUE JS, PET-ПРОЕКТ, WEB-САЙТ, MVC, СЕРВЕР, АДМІН, АВТОРИЗАЦІЯ, МАГАЗИН, ІНТЕРНЕТ, ДАНІ.

# 

**Зміст**

[**ВСТУП** 7](#_Toc92540836)

[РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ 8](#_Toc92540837)

[**1.1** **Аналіз задачі, засобів та методів її вирішення** 8](#_Toc92540838)

[**1.2** **Аналіз існуючого програмного забезпечення за тематикою курсової роботи.** 8](#_Toc92540839)

[**Висновки до першого розділу :** 15](#_Toc92540840)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 16](#_Toc92540841)

[**2.1 Проектування загального алгоритму роботи веб-додатку** 16](#_Toc92540842)

[Загальна схема роботи програми : 17](#_Toc92540843)

[**2.2 Розробка функціональних алгоритмів роботи програми** 18](#_Toc92540844)

[**2.3 Розробка програмного забезпечення** 19](#_Toc92540845)

[РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ 23](#_Toc92540846)

[**3.1 Опис роботи з програмним додатком (Опис інтерфейсу)** 23](#_Toc92540847)

[**3.2** **Тестування роботи програмного забезпечення** 29](#_Toc92540848)

[**Висновки до 3 розділу :** 31](#_Toc92540849)

[**ВИСНОВКИ** 32](#_Toc92540850)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 33](#_Toc92540851)

[**ДОДАТКИ** 34](#_Toc92540852)

**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

БД – База даних

КП – курсовий проект

MVC – Модель–вигляд–контролер (або Модель–представлення–контролер, Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

ПЗ – Програмне забезпечення

# **ВСТУП**

У цій курсовій роботі буде наведено процес розробки інтернет-магазину книг.

Інтернет-магазин книг – це інтернет-магазин, де корситувач може замовляти книги. Взагалом останнім часом інтернет-магазини ствють все більш і більш популярними, тому що дозволяють здійснювати покупки не виходячи з дому, тому дана тема курсової є актуальною.

**Метою створення** даного курсового проекту є розробка CMS системи інтеренет магазину книжок.

**Об’єктом дослідження** є методи та засоби розробки CMS за визначеними предметними областями.

**Предмет дослідження** є використання веб-технологій для забезпечення інформаційних потреб предметної області.

# РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

## **Аналіз задачі, засобів та методів її вирішення**

Завдання полягає в тому, що потрібно:

1. Визначити потреби звичайного користувача
2. Зрозуміти основні принципи та алгротитми роботи інтернет-магазину
3. Продумати можливість додавання книжок та їх повного редагування
4. Реалізувати сортування та пошук книг
5. Реалізувати userfriendly інтерфейс
6. Продумати розподілення ролей та можливість створення нових адмінів
7. Створити корректну базу даних
8. Продумати дизайн сайту
9. Реалізувати адаптивність для різних пристроїв

Для пришвидчення роботи інтренет-магазину потрібно використовувати асинхронні методи.

## **Аналіз існуючого програмного забезпечення за тематикою курсової роботи.**

При аналізі вже існуючих *веб-сайтів* за тематикою курсової роботи було виявлено декілька проектів. Всі вони схожі і мають подібну реалізацію, адже всі вони є інтернет магазинами книжок, які призначені для продажу книжок.

Для розробки сайту використано мови програмуваня Python(+фраймворк Fast API) та JS(+фреймворк Vue js)

Aналогічні інтренет-магазини за заданою темою:

**Yakaboo**

Посилання : https://www.yakaboo.ua/

Розглянемо даний аналог:

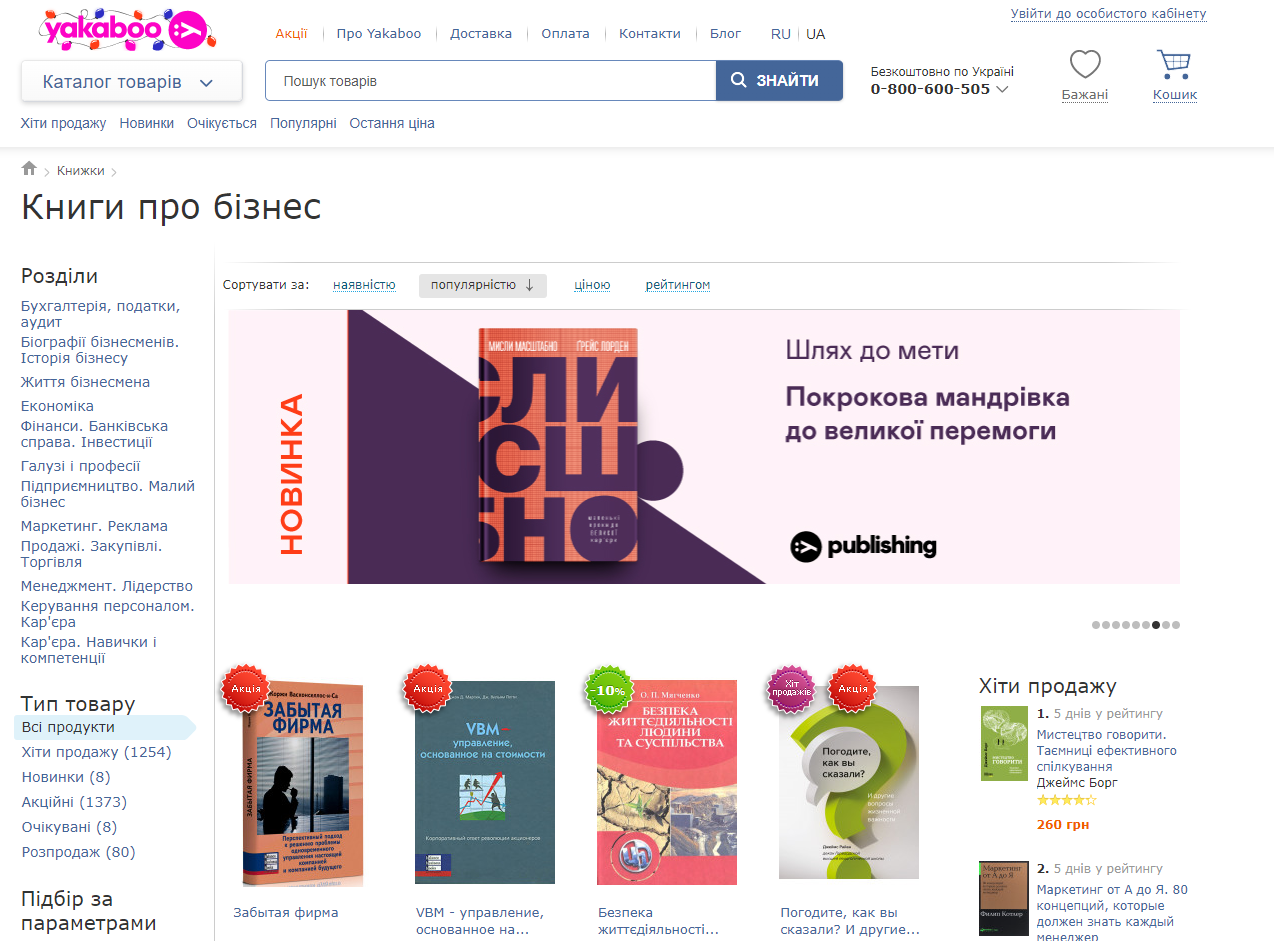


Рис 1.1 Yakaboo

Інтернет-магазин Yakaboo є одним із найпопулярніших інтернет-магазинів про продажу книг, адже має інтуїтивно зрозумілий користувацький інтріейс та широкий асортимент книг. Також має широкий вибір параметрів для сортування товарів. Серед мінусім можна виділити відсутність адаптиву.

Основні плюси:

* Інтуїтивно зрозумілий користувацький інтерфейс
* Широкий вибір параметрів для сортування товарів
* Широкий асортимент книг

Основні мінуси:

* Відсутність адаптиву

**Amazon**

Посилання : https://www.amazon.com/

Розглянемо даний аналог:

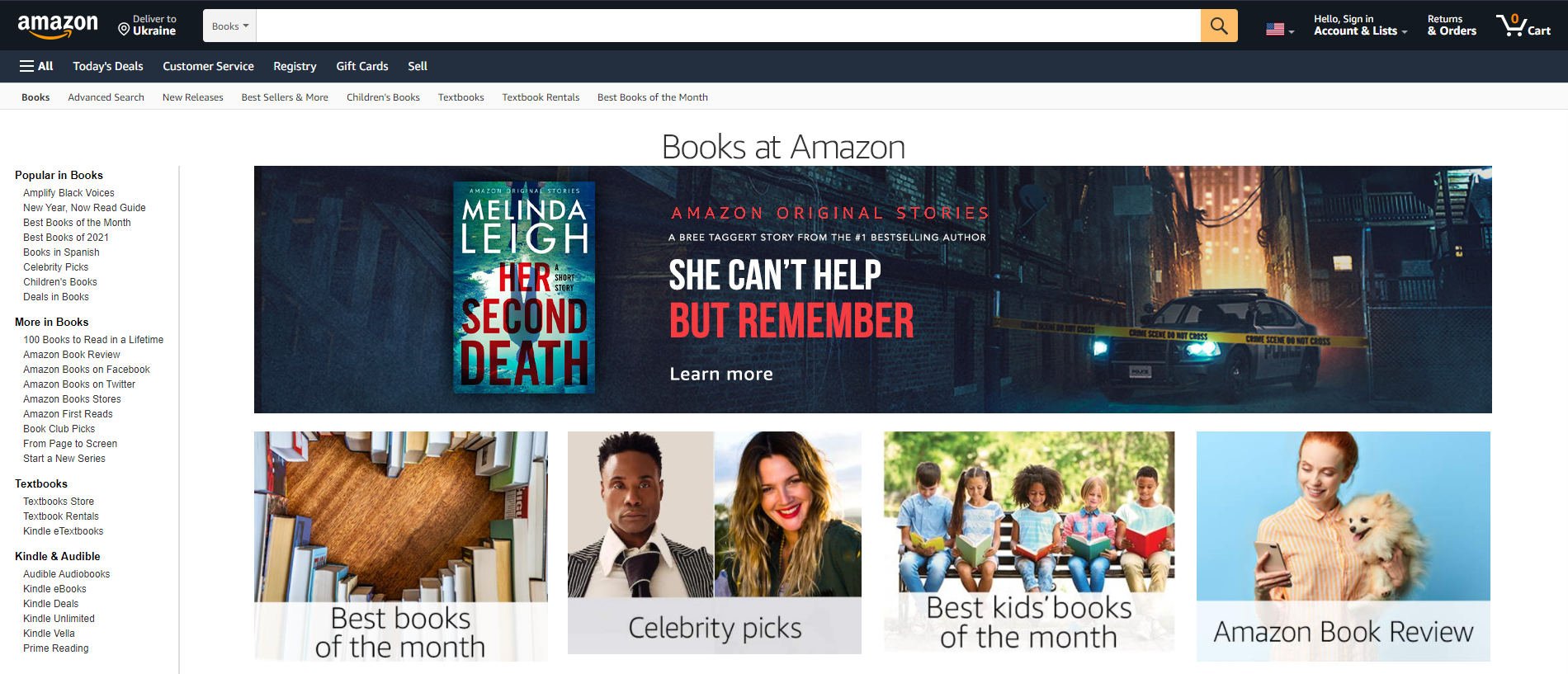


Рис 1.2 Amazon

Інтернет-магазин Amazon є одним із найперших інтернет-магазинів.

Спочатку даний сайт розроблявся саме для продажу книг, а вже згодом розширювавля для інших видів товарів. Також має широкий вибір параметрів для сортування товарів. Серед мінусім можна виділити відсутність адаптиву та незначно надмірну уількість елементів користувацького інтерфесу.

Основні плюси:

* Широкий вибір параметрів для сортування товарів
* Широкий асортимент книг
* Один з найперших інтернет-магазин книг

Основні мінуси:

* Відсутність адаптиву
* Незначно надмірний користувацький інтерфейс

**British Book**

Посилання: https://www.britishbook.ua/

Розглянемо даний аналог:



Рис 1.3 British Book

Інтернет-магазин British Book це інтернет-магаизин, який має стильний дизайн та можливості попередніх розглянутих продуктів. Навідміну від попередньо-розглянутих сайтів, має також адаптив.

Основні плюси:

* Всі, що і в попередньо-розглянутих інтерент-магазинів
* Адаптив для різних пристроїв

Основні мінуси:

* Не такий відомий як попередньо-розглянуті магазини

Актуальним напрямком реалізації власного інтернет-магазину є створення веб-сайту з бізнес логікою подібною до реалізованої у всіх представлених продуктах, реалізувати користувацький інтерфейс по принципам yakaboo, уникаючи надмірності Amazon, але послідувати основним аспектам інтерфейсу цього продукту. Також продумати сучасний дизай сайту, звертаючи увагу до дизайну British Book та реалізувати адаптив.

## **1.3 Технічне завдання на курсову роботу**

**1. Загальне положення**

***1.1Найменування програмного засобу***

Повне найменування програмної системи: "Розробка інтернет-магазину книг" (надалі "веб-додаток"). Коротка назва програмної системи - " Інтернет-магазин"

***1.2. Призначення розробки та область застосування***

Веб-сайт "Інтернет-магазин книг" призначений для розміщення каталогу книг, їх пошуку, та продажу. Система може бути впроваджена у роботу інтернет-магазину. Веб-сайт " Інтернет-магазин книг" дозволить швидко переглядати інформацію про книги, та забезпечить керівництву підприємства своєчасною та повною інформацією про актуальний стан замовлення покупок інтернет-магазину у повному обсязі, а також надасть користувачам можливість вибору книг різних видавництв в залежності від їх смаку.

***1.3. Найменування розробника та замовника.***

Розробник даного продукту - студент групи ІПЗ-20-4 Кормиш Роман Іванович (надалі "розробник"). Замовник програмного продукту – кафедра інженерія програмного забезпечення Житомирського державного технологічного університету в межах виконання курсової з дисципліни «Web-тееології» Чижмотря Олена Геннадіївна, Морозов Андрій Васильович **2. Підстава для розробки**

***2.1. Документ на підставі якого ведеться розробка***

Робота ведеться на підставі навчального плану за напрямом 121 «Інженерія програмного забезпечення».

**3. Вимоги до програми**

***3.1. Вимоги до функціональних характеристик.***

***3.1.1. Загальні вимоги***

Веб-додаток має забезпечувати:

• можливість дистанційної роботи з робочих станцій локальної та глобальної мережі підприємства;

• постійний доступ користувачів веб-додатку;

• оформлення замовлення;

• організацію управління сайтом;

• можливість доступ до бази даних;

***3.1.2. Склад виконуваних функцій***

Розробити інтернет-магазин книг, що підтримує виконання наступних операцій:

1. Пошук книг.
2. Зміна інформації про книгу.
3. Додавання запису про книгу:

* Назва
* Дата видання
* Ціна
* Автор
* ISBN
* Жанри
* Короткий опис

1. Фільтрація книг за параметрами:

* Назва
* Автор
* Дата видання
* Кількість
* Ціна

1. Перегляд інформації про книги

***3.1.3. Організація вхідних і вихідних даних***

Вхідними даними є інформація про книгу (Назва, Автор, ціна, кількість, дата видання, ISBN, опис)

***3.1.4. Часові характеристики і розмір пам'яті, необхідної для роботи програми.***

* Час реакції програми на дії користувача (маніпуляції з пристроями введення даних) не повинен перевищувати 0,25 с.
* Час виконання команд меню не більше 1 с.
* Відображення масивів даних за запитами не більше 3 хвилин.
* Доступність БД – 90% цілодобово.
* Операції з’єднання з БД не більше 1 хвилини.
* Обсяг оперативної пам'яті, необхідний для роботи програми не менше 1Гб.
* Дисковbй простір, необхідний для збереження програми і файлів даних не більше 300 Мбат для робочої станції та 20 ГБайт.
* Інсталяційний пакет програми, що містить у складі БД не повинні перевищувати 100 Мбайт.

Схема системи інтернет-магазину книг:

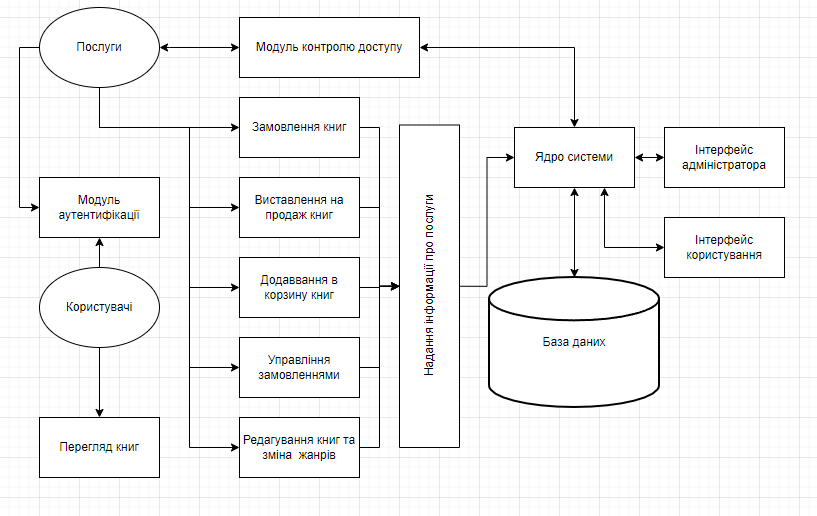
****

Рис 1.4 Структура системи інтернет-магазину книг

## **Висновки до першого розділу :**

У ході виконання першого розділу було поставлено завдання проекту, а саме який має бути функціонал і визначено актуальність проекту. Також визначено мови і фреймворки для розробки та актуальний напрямок розробки власного продукту. Розглянуто існуючі програмні продукти зі схожою тематикою та більш детально проаналізовано деякі із них, а саме : Amazon, British Book та Yakaboo.

# РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## **2.1 Проектування загального алгоритму роботи веб-додатку**

На даному етапі розробки розглянемо загальний алгоритм роботи інтернет-магазину. Загальна схема роботи приведена на рисунку 2.1.

Для початку роботи потрібно створити моделі бази даних. Потум спроектувати схеми з унаслідуванням, адже саме це потрібно для API, який буле реалізовуватися на мові програмування Python та його фреймворку Fast API.

Зголом потрібно реалізувати основні роути для виконання запитів, що дозволить протестувати отримання та обробку інформації отриманої від даного API. Потім реалівувати всі роути та сервіси для кожної сутності, з використання схем які ми описали раніше.

Для авторизації використаємо JWtoken, для забезпечення безпеки даних – хешування паролів.

Тепер спроектуємо візуальну частину. Для цього використаємо мову програмування JS та її фреймворк Vue js. Напишемо декілька компонентів та першу сторінку, з використанням vue-store, що протестувати з’єднання зі створеною API. Потім потрібно написати всі шаблони сторінок, стилі та інші компоненти з сторінками.

Для взаємодії з користувачем потрібно реалізувати “діалогові вікна” (форми). Для того, щоб гість, зареестрований користувач і адмін мали різні права, реалізуємо розділення на ролі з різними прававми. Де адмін має найбільші права (право створювати адмінів включно), користувач – набір повноцінних прав, а гісті може тільки переглядати деякі сторінки сайту.

При запуску сайту користувач буде бачити основну сторінуку, сторінку магазину. Для йому буду пропонуватися зареєструватися або аутентифікуватися. Після цих дій він отримає можливість замовляти книги та виставляти на продаж власні.

## Загальна схема роботи програми :

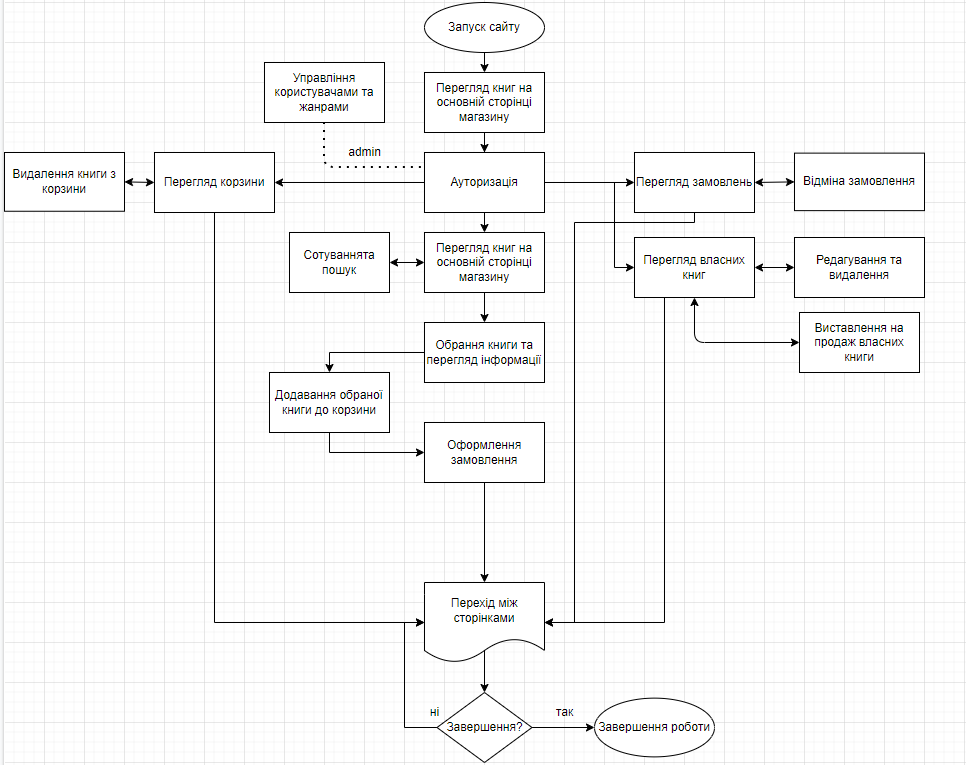


Рисунок 2.1 – Загальна схема роботи програми

## **2.2 Розробка функціональних алгоритмів роботи програми**

Залишилося розглянути детальніше алгоритми робити основних функцій.

Звичайно, основне призначення інтренет-магазину книг – продаж книг, тому розглянемо саме це. Для висталвення книги на продаж потрібно заповнити відповідну форму, після чого книга розміститься на сайті. При заповненні форми будуть з’являтися підзказки про коректність вводу. Після розміщення інший користувач зможе замовити цю книги в максимальній кількості вказаній продавцем. Користувач додає книгу в корзину, звідки можна оформити замовлення. Для цього потрібно заповнити відповідну форму, вказавши адрессу та дату прибуття, не раніше сьогоднішньої дати.

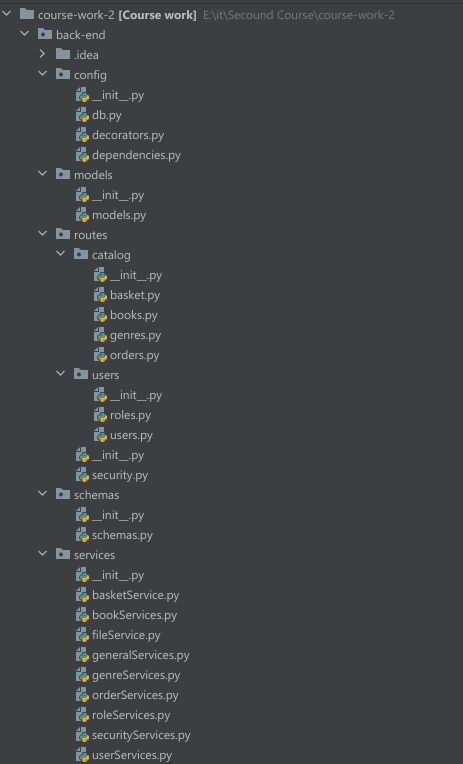
Для корректонсті обробки даних реалізована подвійна валідація, як бекенді так і на фронтенді. При будь-якій віправці форми висвітлбється вікно з оповіщення про статус успішності дії. При редагування дані автоматично заповнюють відповідні поля, для облегчення їх зміни.

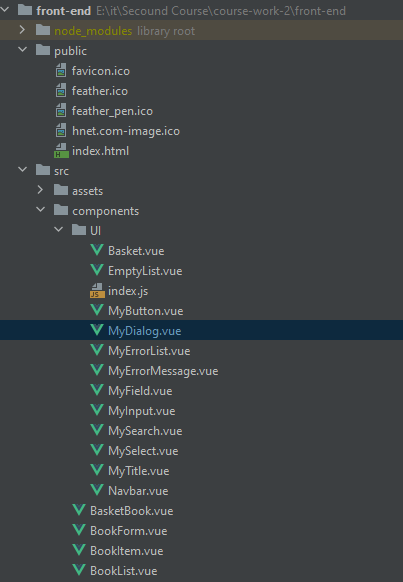
Основним алгоритмом кожної реалізованої функції є:

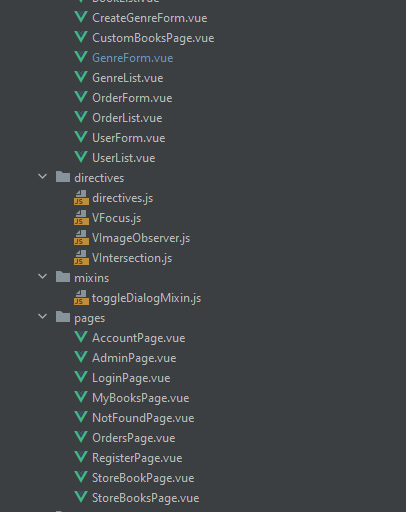
отримання даних на фронтенді -> створення запиту -> передача запиту на бекенд -> обробка отриманих даних у відповідних сервісах(взаємодія з БД) -> передача оброблених даних на фронтенд.

## **2.3 Розробка програмного забезпечення**

Структура рішення проекту :





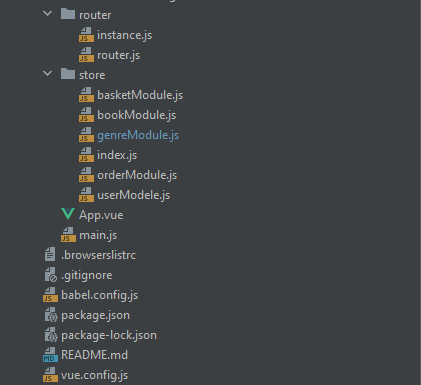


Рисунок 2.2 – Структура проекту

Більша частина програмного забезпечення розробляється відповідно до алгоритму описаного вище, тому розглянемо цей принцип на прикладі реалізованих методів книжок. Методи книжок розмістимо в файлі bookServises, що згодом будуть використовуватись в books-routes, що надають можливість приймати запити з фроненду.

Розглянемо приклад коду створення книжки:

*def* create\_book(*db*: Session, *model*: schemas.BookCreate, *current\_user*: models.User) -> Any:  
 expression = \_model.ISBN == *model*.ISBN  
 generalServices.check\_in\_use\_expression(db=*db*, model=\_model, expression=expression)  
 book = \_model(  
 name=*model*.name,  
 author=*model*.author,  
 content=*model*.content,  
 price=*model*.price,  
 owner=*current\_user*,  
 count=*model*.count,  
 publication\_date=*model*.publication\_date,  
 ISBN=*model*.ISBN  
 )  
 *db*.add(book)  
 *db*.commit()  
 *return* book.id

Маємо метод для створення книжки, який повертає id новостворенної книги та приймає базу даних, з якою ми працюємо, схему з даними та поточного користувача. Використовуємо дані, які передаються в метод, та створюємо книгу. Книга додається в базу, що дає змогу отримати її id згодом.

Розглянемо роут для даногго методу:

@router.post('', response\_model=int)  
*async def* create\_book(*bookCreate*: schemas.BookCreate,*db*: Session = Depends(get\_db),  
 *current\_user*: models.User = Depends(get\_current\_user)):  
 *return* bookServices.create\_book(db=*db*, model=*bookCreate*, current\_user=*current\_user*)

Як модемо бачити код методу не великий, тому що весь функціонал винесений в сервіс.

Далі перейдемо до створення запиту з фронтеду:

async createBook({*state*, *commit*, *rootState*, *rootGetters*}) {  
 await *commit*('setFormLoading', true)  
 *rootState*.errors = []  
 const book = *state*.book  
 book.publication\_date = new *Date*(book.publication\_date.toLocaleString()).toISOString()  
 await instance  
 .post(`${*state*.defaultRoot}`, book, {headers: *rootGetters*.getHeaders})  
 .then(*response* => {  
 book.id = *response*.data

book.image = 'default.png'  
 *commit*('pushBook', book)  
 *commit*('clearBook')  
 })  
 .catch(*error* => {  
 *rootState*.errors.push(*error*.response.data.detail)  
 })  
 await *commit*('setFormLoading', false)  
},

Тут ми можемо бачити формування запиту, де instance містить основний шлях до нашого API

export const *instance* = *axios*.create({  
 baseURL: 'http://192.168.0.105:8000/api/',  
});

, що дозволяє уникнути дубляції коду. Виконується метод post, який відпраляє post запит на сервер та приймає як параметри частину шляху, в даному випадку defaultRoot(‘/books’), book – обєкт з даними про книгу config, який містить в headers токен доступу. Далі отриманий результат виконаня обробляється в then(), де ми виконуємо даданя даних до моделі книги та додаємо її в список книг, які показуються користувачу.

Інші запити виконуються по схожому алгоритму, тому вважаю надмірним демонтрувати їх код.

Також важливим компонетом веб додатку є аутентифікація, реалізована на основі JWT. Роглянемо її код:

@router.post("/token", response\_model=schemas.Token)  
*async def* login\_for\_access\_token(*form\_data*: OAuth2PasswordRequestForm = Depends(), *db*: Session = Depends(get\_db)):  
 user = authenticate\_user(db=*db*, email=*form\_data*.username, password=*form\_data*.password)  
 *if not* user:  
 *raise* HTTPException(  
 status\_code=status.HTTP\_401\_UNAUTHORIZED,  
 detail="Incorrect username or password",  
 headers={"WWW-Authenticate": "Bearer"},  
 )  
 access\_token\_expires = timedelta(minutes=ACCESS\_TOKEN\_EXPIRE\_MINUTES)  
 access\_token = create\_access\_token(  
 data={"sub": user.email, "role": user.role.name}, expires\_delta=access\_token\_expires  
 )  
 *return* {"access\_token": access\_token, "token\_type": "bearer"}

Вище приведений код роута для аутентифікації, який приймає форму з даними. В методі authenticate\_user виконується аутентифікація і якщо вона не успіша користувачу передається повідомлення про помилку.

Розглянемо дедальніше цей метод:

*def* authenticate\_user(*db*: Session, *email*: str, *password*: str) -> Any:  
 model = models.User  
 expression = model.email == *email* user = generalServices.get\_by\_expression(db=*db*, model=model, expression=expression)  
 *if not* verify\_password(*password*, user.password):  
 *return False  
 return* user

В даному методі ми звертаємося до бази даних і перевіряємо на співпадіння паролі та повертаємо відповідний реультат.

Після того як ми отримали результат аутентифікації, при успішності ми створюємо токен доступута записуємо в нього дані про email, role та дату закінчення токену. І в фіналі повертаємо на фронтенд результат виконання методу, а саме новостворений токен доступу.

## **Висновки до 2 розділу :**

Спроектовано загальну схему програми. Визначено та детально описано основні можливості програми. Пояснено алгоритми найважливіших методів інтернет-магазину та приведено код їх основних частин. Приведено приклади коду деяких методів. Роз’яснено деякі нюанси роботи програми. Показано структуру проекта.

# РОЗДІЛ 3 ОПИС РОБОТИ З ДОДАТКОМ

## **3.1 Опис роботи з додатком (Опис інтерфейсу)**

Після запуску програми з’являється головна сторінка (Рис 3.1), де користувач бачить список книг та може тільки їх проглядати.

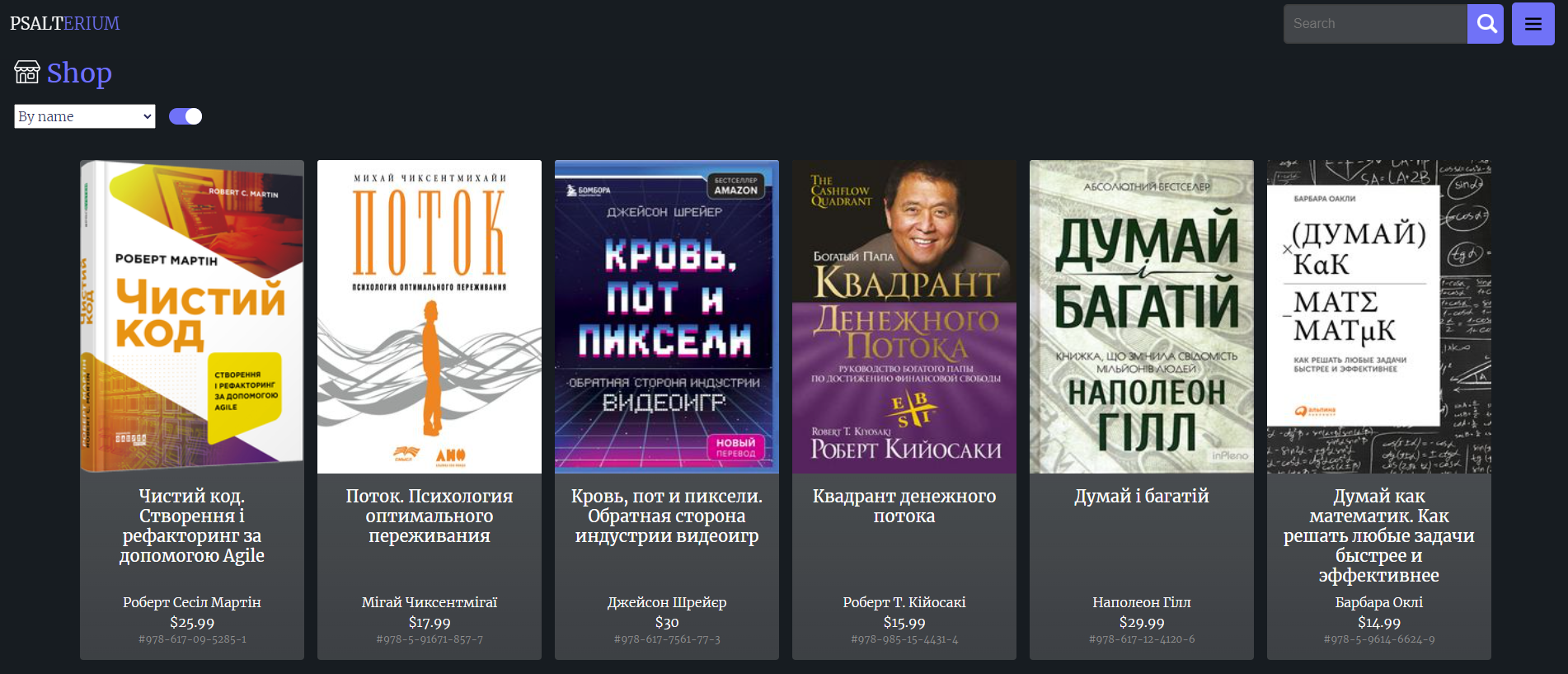


Рис 3.1 Головна сторінка

Головна сторінка має функції переходу до конкретних книжок, сортування та пошуку. Також реалізована динамічна пагінація у виді бескінечної ленти. У верхньому правому куті розміщений пошук та меню, де користувач може побачити наявні сторінки.

Перейдемо до конкретного алгоритму роботи х веб додатком.

Спочатку користувач обирає книгу, наприклад, книгу “Чистий код”. При натиску на карточку з відповідною книгою, він переходить на сторінку книги (Рис 3.3), де він може прочитати інформацію про книгу більш детально та додати деяку кількість книг до власної корзини. Після успішного додання книги до корзини, користувач бачить повідомлення:

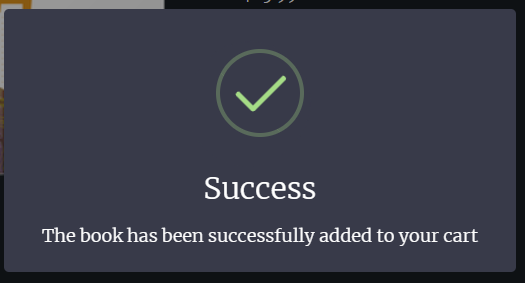


Рис 3.2 Сторінка конкретної книги

Далі користувач може переглядати інші книги та додавати до корзини або оформити замовлення.

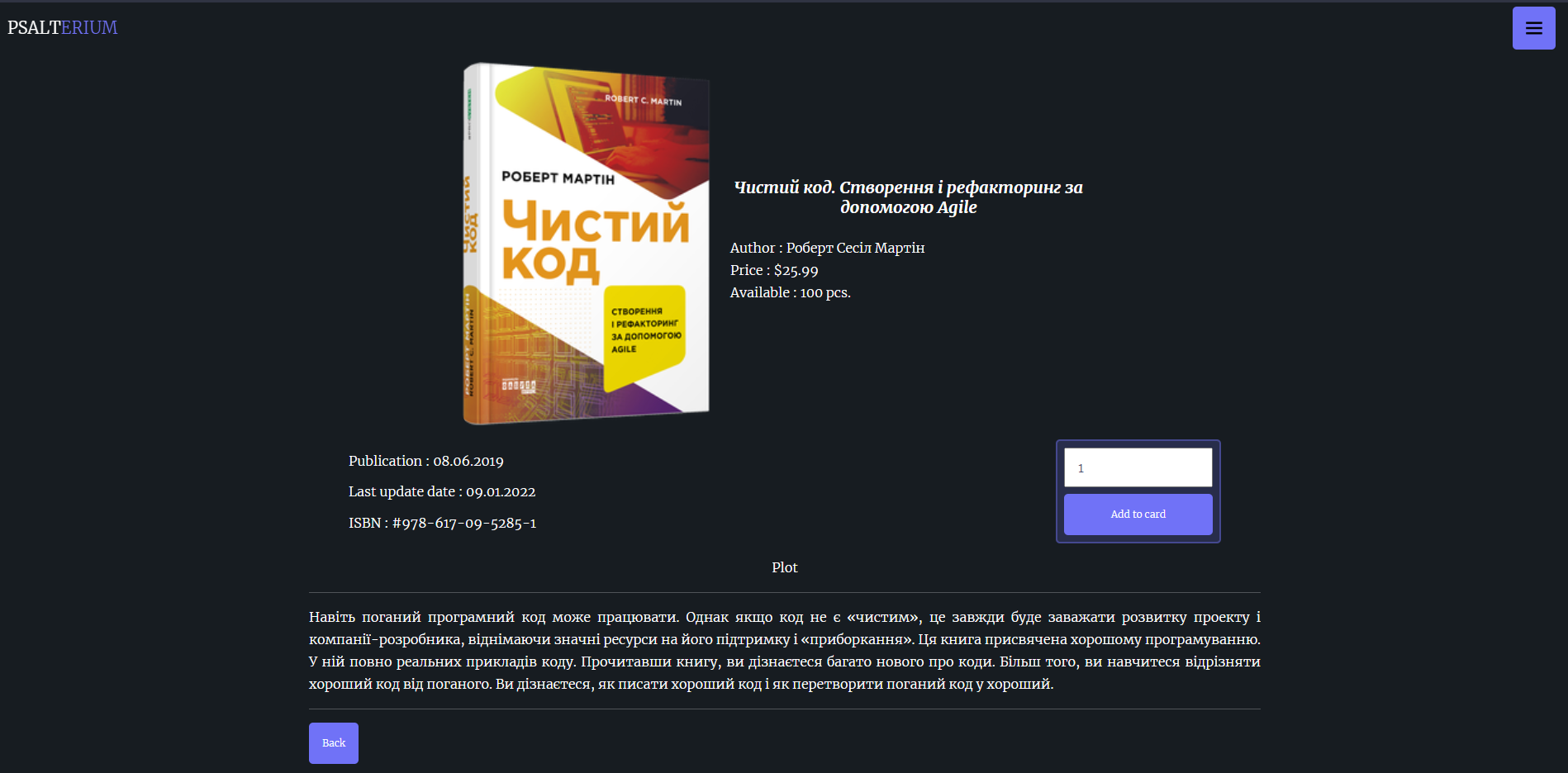


Рис 3.3 Сторінка конкретної книги

Для прикладу оберемо ще одну книгу(“1984”). Додали до корзини 3 екземпляри. Далі користувач переходить до корзини для оформлення замовлення(Рис 3.4). Як можемо бачити, тут відображаються всі нами обрані книги.

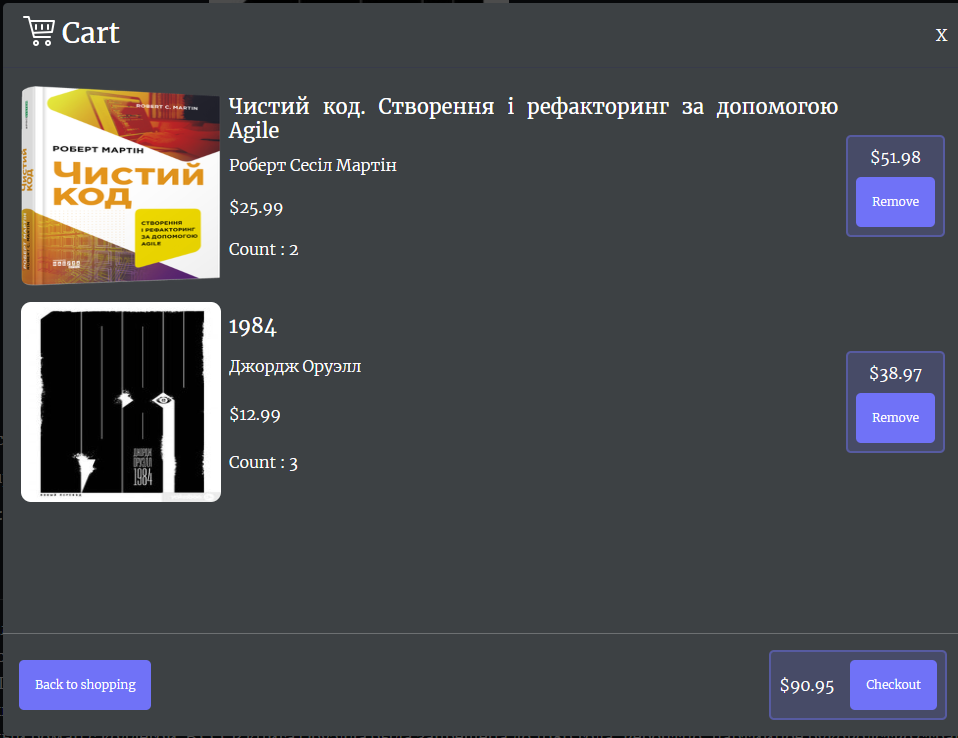


Рис 3.4 Корзина

Також користувач може видалити з корзини книгу, натиснувши на кнопку зліва від книги. Над кнопкою також відображена ціна за всі екземпляри книги. Якщо користувач впевнений, що йому саме ці книги потрібні, він переходить до оформлення замовлення(Рис 3.5). Після оформлення форми замовлення додається в список замовлень(Рис 3.6).

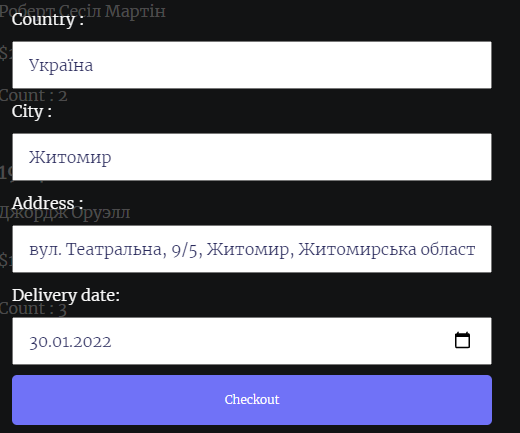


Рис 3.5 Форма оформлення замовлення

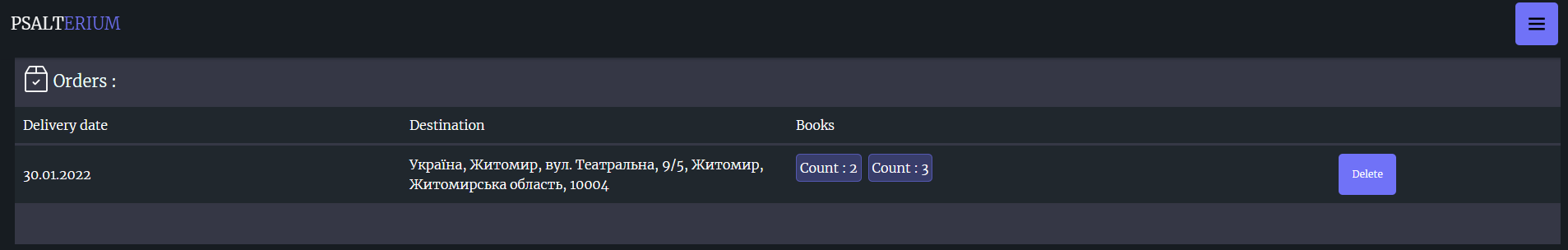


Рис 3.6 Список замовлень

На сторінці список замовлень користувач може відміняти замовлення та переглянути замовлені книги при кліку на підсвічені елементи, які ведуть на сторінки відповідних книжок. На даній сторінці адміну будуть показуватися всі замовлення, а звичайному користувачу тільки його власні.

Далі перейдемо до дадаткової можливості інтернет-магазину – виставлення на продаж власних книг.

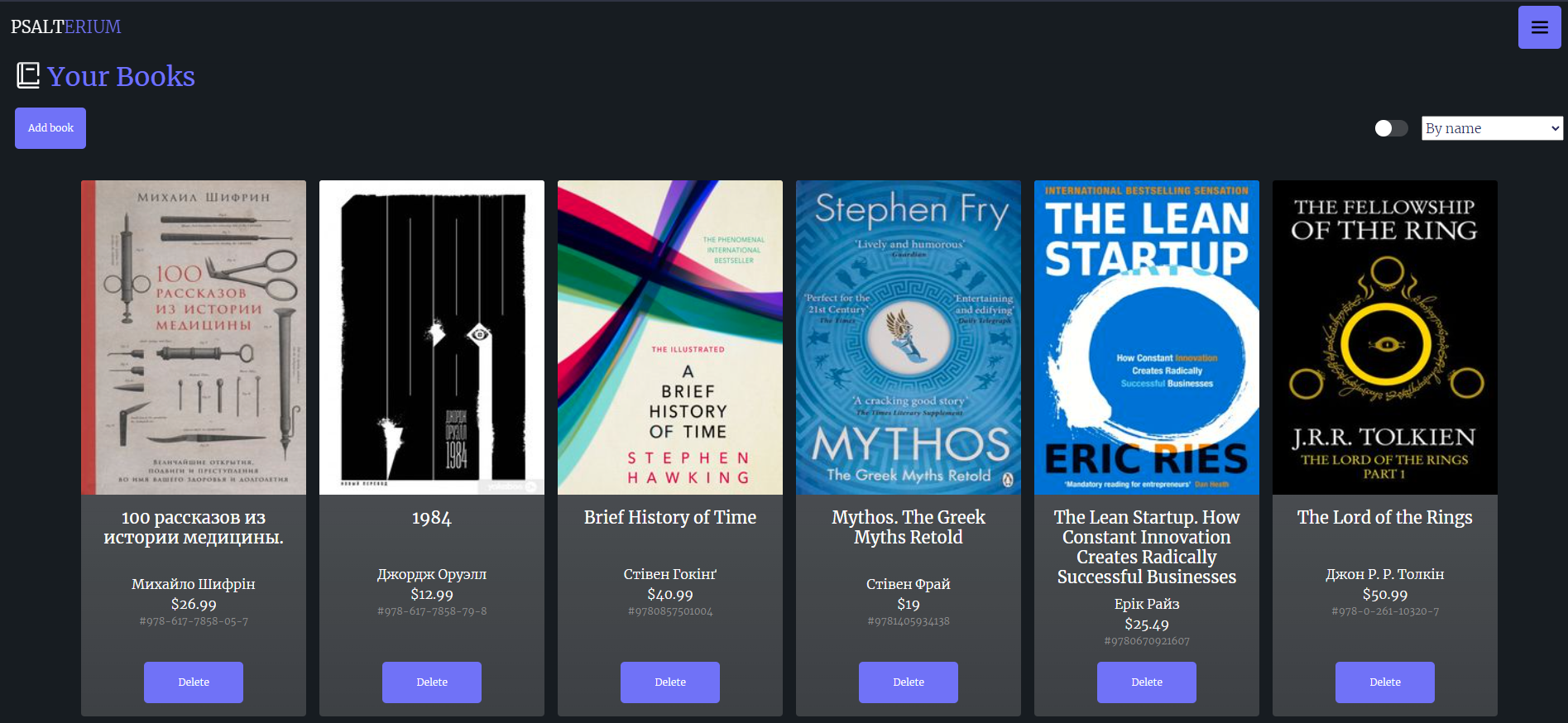


Рис 3.7 Сторінка книг виставлених на продаж користувачем

На даній сторінці у верхньому правому куті розміщена кнопка, яка відкриває форму для книги(Рис. 3.8). Коли користувач заповнить цю форму, книга відобразиться на сторінці та додасться до магазину, де інший користувач зможе її замовити.

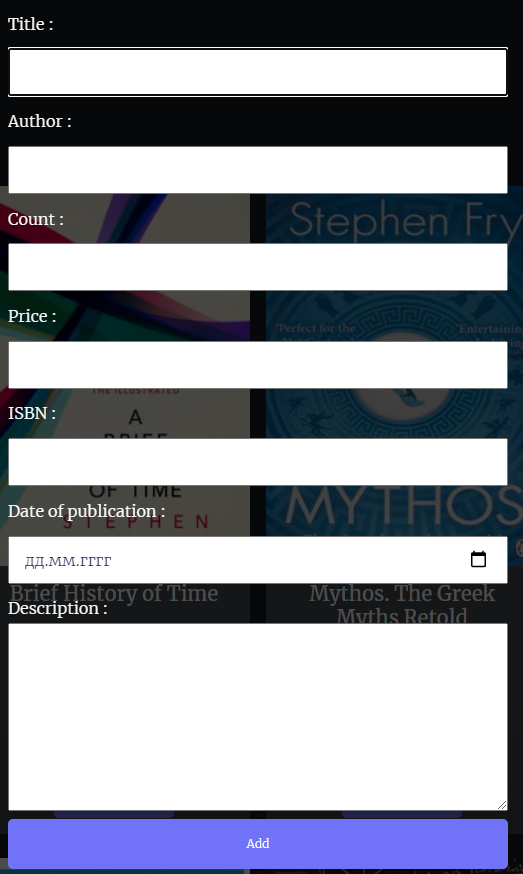


Рис 3.7 Форма для виставлення книг на продаж

Після виставлення книги, її користувач може перейти на неї і побачити додатковий функціонал, а саме можливість зміни ланих про книгу та задати їй жанри.

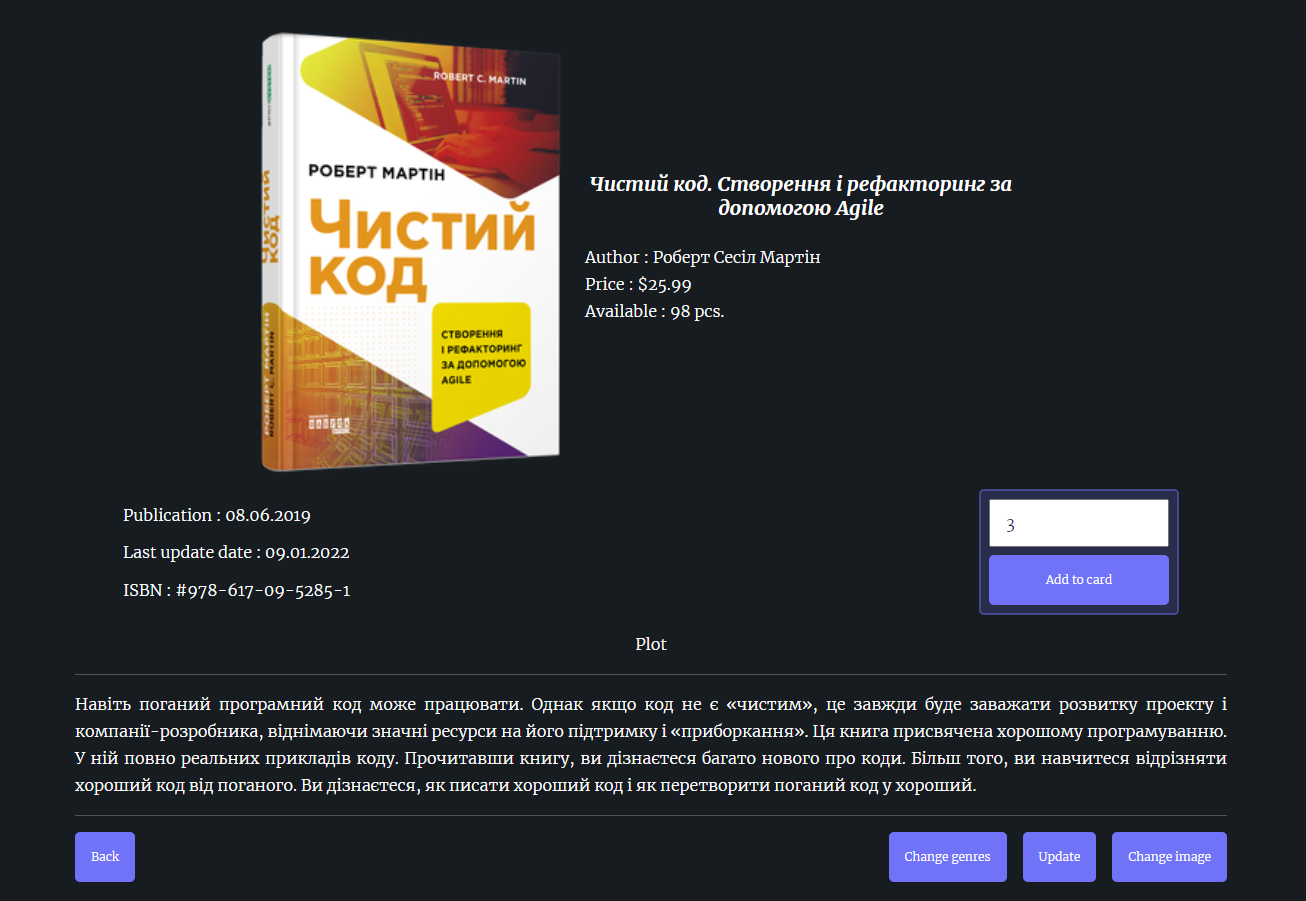


Рис 3.8 Сторінка книги з розширеним функціоналом

Користувач може перейти на власну сторінку(Рисю 3.9) і змінити про себе дані та пароль.

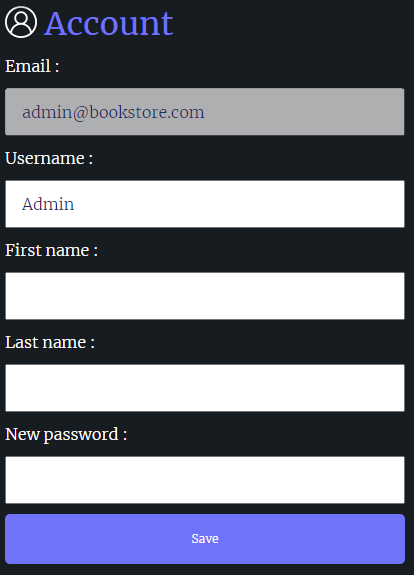


Рис 3.9 Форма для зміни інформації про користувача

Також не можна не розглянути функціонал який доступний тільки для адмінів, а саме сторінку адміністрації(Рис. 3.10).

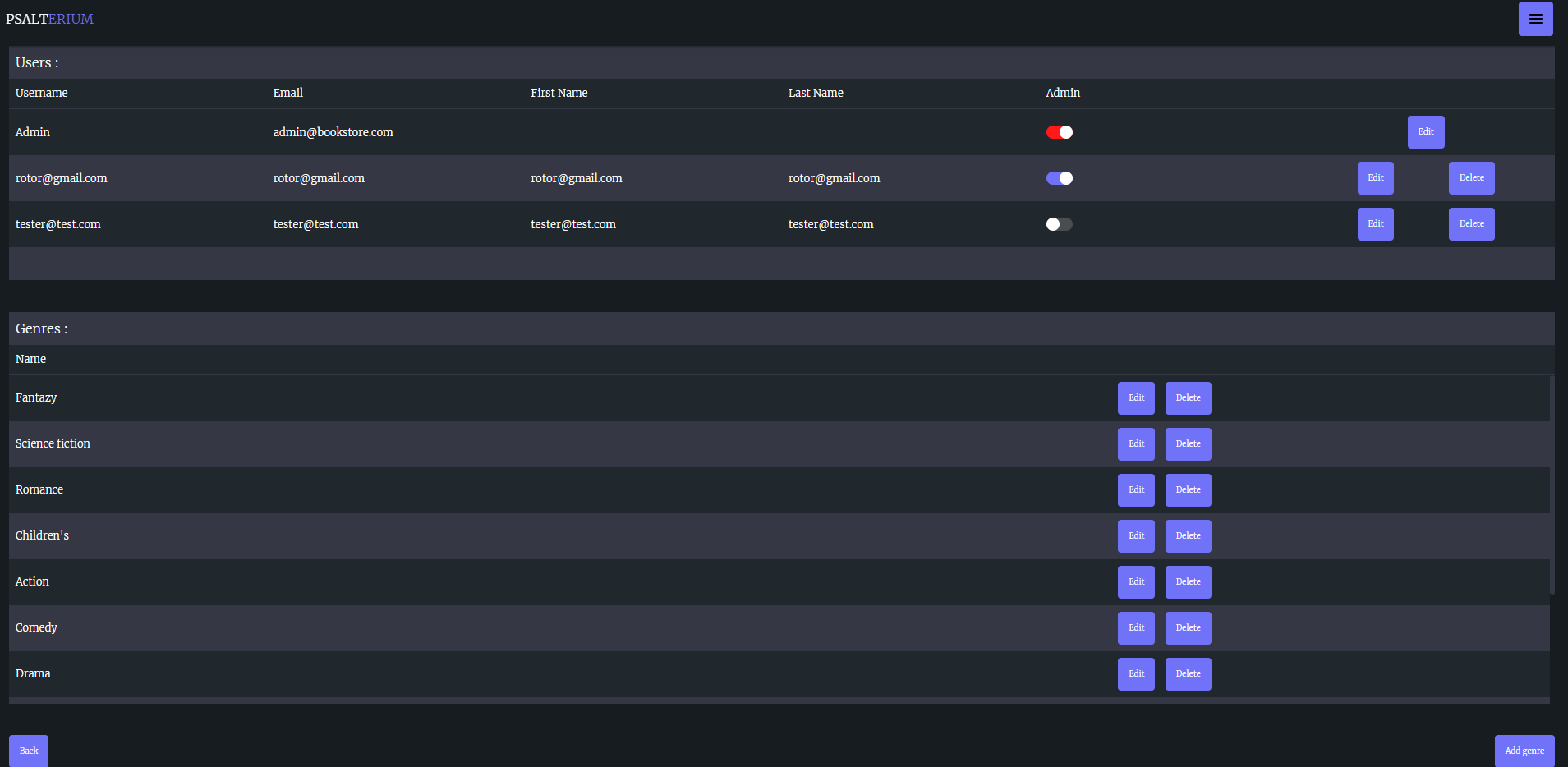


Рис 3.10 Сторінка адміністрації

На сторінку адміністрації мають доступ тільки адміністратори, як не дивно. На даній сторінці адміністратор може переглядати та змінювати дані про користувачів та жанри. Також має можливість видаляти і користувачів і жанри. Користувачі створюються при реєстрації, а для створення жанрів зліва знизу розміщена кнопка для відкриття форми створення жанру(Рис 3.11). Також адмін має можливість надати права адміна іншому користувачу.

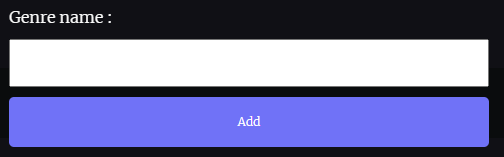


Рис 3.10 Форма створення жанру

Після кожної успішної дії користувач отримує повідомлення про успішність виконання. Розглянемо на прикладі створення нового жанру.

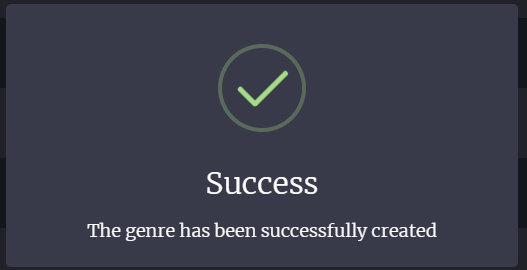


Рис 3.11 Повідомлення про успішне створення жанру

## **Тестування веб-додатку**

Веб-додаток відображає попередження про помилку при при введенні даних для зручного користування користувачем. Наприклад, при спробі створення “пустого” користувача відображаються попередження про обов’язковість полів(Рис 3.12).

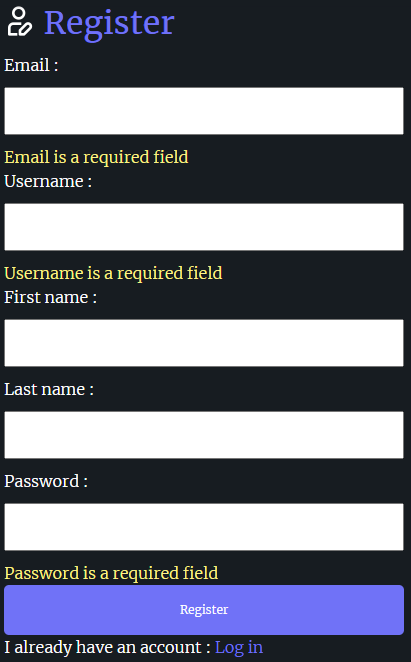


Рис 3.12 Повідомлення про необхідність заповнення

При вводі некоректного значення, з’являється попередження(Рис 3.13)

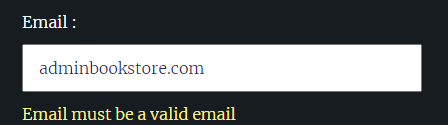


Рис 3.13 Повідомлення про некоректність вводу

При вводі даних які вже існують і не можуть дублюватися, з’являється попередження про існування(Рис 3.14)

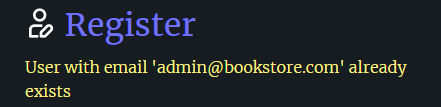


Рис 3.14 Повідомлення про існування користувача

При відправці форми з некоректними даними аутентифікації, з’являється попередження некоректність(Рис 3.15)

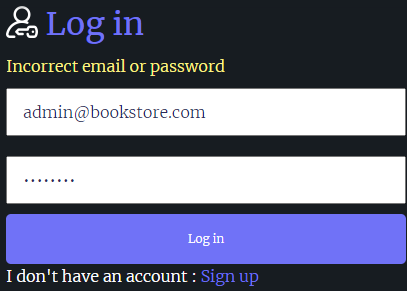


Рис 3.15 Повідомлення некоректність email або пароля

Подібна валідація реалізована на всіх формах

## **Висновки до 3 розділу :**

Було детально описано роботу програмного додатку, користувацький інтефейс, очікувані дії користувача та їх результати. Також було розписано моменти уникнення помилок завдяки допоміжним перевіркам при введенні даних або при роботі з зображеннями. Завдяки цим перевіркам вдалося уникнути частих та серйозних помилок. Наведено приклади обробки помилок.

# **ВИСНОВКИ**

Під час написання даного курсового проекту було отримано навички роботи з розробки веб-додатку.

В першому розділі курсового проекту, було проаналізовано знайдені інтернет-магазини зі схожим функціоналом та інтерфейсом, та було сформовано уявлення про інтернет-магазин, що розроблюється. Було виявлено основні потреби для написання даного веб-додатку.

В другому розділі курсового проекту, було проведено розробку функціональних і загальних частин веб-додатку. При розробці функціонального алгоритму, розробили уявлення про методи розробки програмного коду. На заключному етапі, розроблено його. В результаті, отримано готовий програмний продукт, який виконуває всі заплановані функції і задачі.

В третьому розділі курсового проекту було описана методи користування створеним інтернет-магазином, були пояснені і продемонстровані всі створені елементи інтерфейсу. Також було проведено тестування, тобто інтернет-магазин було запущено в реальних умовах її користування. В ході тестування проблем не виникало.

В результаті виконання курсового проекту отримано інтернет-магазин книжок, який повністю відповідає запланованому функціоналу, умовам та вимогам, поставленим на початку проектування та привищує його. Інтернет-магазин є зрозумілим та зручним в користуванні, адаптивним дял різних пристроїв та швидкодіючим.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Об'єктно-орієнтоване програмування [Електронний ресурс] / Освітній портал ДУ «Житомирська політехніка» – 2020. – Режим доступу до ресурсу: https://learn.ztu.edu.ua/course/view.php?id=1628
2. Технології програмування. Мова С# : навч. посібн/ В.В. Томашевський. – Житомир: ЖВІ НАУ, 2012. – 484 с.
3. Шилдт Герберт C# 4.0: полное руководство.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011. — 1056 с.: ил.
4. Эндрю Т. Язык Программирования C# 5.0 и платформа .NET 4.5 / Троелсен Эндрю., 2013. – 1168 с. – (6-е издание).
5. Павловская Т.А. Програмирование на языке высокого уровня / Т.А. Павловская – СПб.: Питер, 2011, – 461с.: ил.
6. Джеймс Глейк Хаос. Создание новой науки / Джеймс Глейк, 1987. – 416 ст.
7. Види фракталів та методи їх створення. [Електронний ресурс] – Режим доступу: <https://sites.google.com/site/fraktali22374/home/vidi-fraktaliv-ta-metodi-ieh-stvorenna>
8. Асинхронное программирование . [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/tutorial/13.3.php>
9. Введение во фракталы [Електронний ресурс] – Режим доступу: <http://algolist.ru/graphics/fracart.php>
10. OpenFileDialog и SaveFileDialog [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/windowsforms/4.20.php>

# **ДОДАТКИ**

Додаток А

**Програмний код:**

**Библиотека класів FractalClasses :**

Class Fractals.cs :

using System;

using System.Collections.Generic;

using HelperClasses;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace FractalClasses

{

public class Fractals

{

protected List<Pixel> pixels;

protected Bitmap picture;

protected Graphics g;

protected Color BackgroundColor;

public Graphics G

{

set

{

if (picture == null)

{

g = null;

}

else

{

g = value;

}

}

}

public Fractals()

{

this.picture = null;

this.pixels = null;

}

public Fractals(Bitmap picture, List<Pixel> pixels, Color BackgroundColor) : this(picture, pixels)

{

this.BackgroundColor = BackgroundColor;

}

public Fractals(Bitmap picture, List<Pixel> pixels):this(picture)

{

this.pixels = pixels;

}

public Fractals(Bitmap picture):this()

{

this.picture = picture;

G = Graphics.FromImage(picture);

}

public Fractals(Bitmap picture, Color BackgroundColor) : this(picture)

{

this.BackgroundColor = BackgroundColor;

}

public void Effects(Graphics g, Color BackgroundColor)

{

g.Clear(BackgroundColor);

g.SmoothingMode = SmoothingMode.AntiAlias;

}

public virtual void Info(TextBox info) { }

~Fractals()

{

Console.WriteLine("Class Fractals is cleared");

}

}

}

Class MBrotSet.cs :

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using HelperClasses;

namespace FractalClasses

{

public class MBrotSet:Fractals

{

protected List<Pixel> TmpPixels;

protected Size size;

protected PictureBox gradientBox;

protected int Iterations;

protected Complex c, z;

public MBrotSet(Bitmap picture, List<Pixel> pixels, Size size, PictureBox gradientBox,int Iterations):base(picture, pixels)

{

this.size = size;

this.Iterations = Iterations;

this.gradientBox = gradientBox;

TmpPixels = pixels;

}

public MBrotSet() : base(){}

public Bitmap CalculationMBrot(double hx, double hy, double x\_, double y\_, double maxZ, double SizeArea, int CountColors, ProgressBar progress)

{

progress.Invoke(new Action(() => // делегат для відображення progressBar

{

progress.Maximum = size.Width;

}));

int UserIt = Iterations;

int change;

int[] ColorIndex = new int[CountColors];

int i = 0;

for (int p = 0; p < gradientBox.Image.Width; p++)

{

if (p % (int)(gradientBox.Image.Width / ColorIndex.Length) == 0)

{

if (i >= ColorIndex.Length)

{

break;

}

ColorIndex[i] = p;

i++;

}

}

if (pixels.Count > gradientBox.Width)

{

for (int p = 0; p < gradientBox.Width; p++)

{

pixels[p].Color = TmpPixels[(int)(p \* (pixels.Count / gradientBox.Width))].Color;

}

}

for (int x = 0; x < size.Width; x++)

{

x\_ = (hx - SizeArea / 2) + x \* (SizeArea / size.Width);

for (int y = 0; y < size.Height; y++)

{

y\_ = (hy - SizeArea / 2) + y \* (SizeArea / size.Height);

c = new Complex(x\_, y\_);

z = new Complex(0, 0);

int it = 0;

do

{

it++;

z.Sqr();

z.Add(c);

if (z.Magn() > maxZ)

{

break;

}

} while (it < UserIt);

// coloring the set

if (it < UserIt)

{

change = it % ColorIndex.Length;

picture.SetPixel(x, y, pixels[ColorIndex[change]].Color);

}

else

{

picture.SetPixel(x, y, Color.FromArgb(0, 0, 0));

}

}

progress.Invoke(new Action(() => // делегат для зміни progressBar

{

progress.PerformStep();

}));

}

return picture;

}

public override void Info(TextBox info)

{

info.Text = "The Mandelbrot set is the set of complex numbers C" +

" for which the function f(z)=z^2 + c does not diverge when iterated" +

" from z = 0, i.e., for which the sequence f(0), f(f(0)) etc.," +

" remains bounded in absolute value. Its definition is credited" +

" to Adrien Douady who named it in tribute to the mathematician " +

"Benoit Mandelbrot, a pioneer of fractal geometry." +

Environment.NewLine +

Environment.NewLine +

"Visually, inside the Mandelbrot set," +

" an infinite number of elementary figures can be distinguished," +

" the largest of which is in the center - the cardioid." +

" There is also a set of ovals related to the cardioid," +

" the size of which gradually decreases, tending to zero." +

" Each of these ovals has its own set of smaller ovals," +

" the diameter of which also tends to zero, etc." +

Environment.NewLine +

Environment.NewLine +

" This process continues indefinitely, forming a fractal." +

" It is also important that these processes of figure branching" +

" do not completely exhaust the Mandelbrot set: if we consider" +

" additional “branchings” with magnification, then in them" +

" you can see your cardioids and circles that are not associated" +

" with the main figure.";

}

~MBrotSet()

{

Console.WriteLine("Class MBrotSet is cleared");

}

}

}

Class FractalTree.cs :

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using HelperClasses;

namespace FractalClasses

{

public class FractalTree : Fractals

{

protected double[] angles;

protected int minLen;

protected int BranchWidth;

private int i = 0;

public FractalTree(Bitmap picture, List<Pixel> pixels, double[] angles, int minLen, int BranchWidth, Color BackgroundColor):base(picture, pixels, BackgroundColor)

{

this.angles = angles;

this.minLen = minLen;

this.BranchWidth = BranchWidth;

}

public FractalTree() : base()

{

}

public void DrawFractalTree(int x, int y, int len, double angle, ProgressBar progress)

{

if (i == 0)

{

base.Effects(g, BackgroundColor);

progress.Invoke(new Action(() => // делегат для відображення progressBar

{

progress.Maximum = 0;

double count = 0;

double tmpCount = len;

while (tmpCount > minLen)

{

tmpCount /= 1.5;

count++;

}

if (angles.Length == 2)

{

for (int i = 0; i < count; i++)

{

progress.Maximum += (int)(Math.Pow(2, i) / Math.Pow(angles.Length, 2));

}

}

else if(angles.Length == 3)

{

for (int i = 0; i < count; i++)

{

progress.Maximum += (int)(Math.Pow(3, i) / angles.Length);

}

if(len / minLen > 40)

{

progress.Maximum \*= (int)(Math.Log(len, minLen) / 2);

}

}

else if (angles.Length == 4)

{

for (int i = 0; i < count; i++)

{

progress.Maximum += (int)((Math.Pow(4, i) / angles.Length));

}

if ((double)len / minLen <= len \* 0.05)

{

progress.Maximum \*= (int)(Math.Pow(Math.Log(len, minLen), 1.4));

}

}

}));

}

int x1, y1;

x1 = (int)(x + len \* Math.Sin((2 \* Math.PI \* angle) / 360.0));

y1 = (int)(y + len \* Math.Cos((2 \* Math.PI \* angle) / 360.0));

g.DrawLine(new Pen(pixels[i \* pixels.Count \* 0.1].Color, BranchWidth), x, picture.Height - y, x1, picture.Height - y1);

i++;

if (len > minLen)

{

for (int j = 0; j < angles.Length; j++)

{

DrawFractalTree(x1, y1, (int)(len / 1.5), angle + angles[j], progress);

}

progress.Invoke(new Action(() => // делегат для зміни progressBar

{

progress.PerformStep();

}));

}

}

public override void Info(TextBox info)

{

info.Text = "Pythagoras, proving his famous theorem," +

" built a figure with squares on the sides of a right-angled triangle." +

" In our century, this figure of Pythagoras has grown into a whole tree." +

Environment.NewLine +

Environment.NewLine +

" For the first time, the Pythagorean tree was built" +

" by A.E. Bosman (1891-1961) during the Second World War," +

" using an ordinary drawing ruler. If in the classical Pythagorean" +

" tree the angle is 45 degrees, then it is also possible to build" +

" a generalized Pythagorean tree using other angles. Such a tree" +

" is often called the windswept Pythagoras tree. If you draw only" +

" the segments connecting the selected centers of the triangles in" +

" some way, you get a naked Pythagorean tree." +

Environment.NewLine +

Environment.NewLine +

"Angles (one angle - one branch)" +

" (-35 - left branch will have angle 35)" +

" or (45 right branch will have angle 45)" +

" (e.g. (-35, 40, 10) we wil have 3 branches" +

" with corresponding angles)";

}

~FractalTree()

{

Console.WriteLine("Class FractalTree is cleared");

}

}

}

Class Barnsley\_fern.cs :

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using HelperClasses;

namespace FractalClasses

{

public class Barnsley\_fern:Fractals

{

protected float minX;

protected float maxX;

protected float minY;

protected float maxY;

protected int NumberOfPoints;

protected float[] probability;

protected float[,] Coefficient;

protected int LenGradient;

public Barnsley\_fern(Bitmap picture, float maxX, float maxY, int NumberOfPoints, float[] probability, float[,] Coefficient, List<Pixel> pixels, Color backgroundColor, int LenGradient, float MinX = -3, float MinY = 0.1f):base(picture, pixels, backgroundColor)

{

this.maxX = maxX;

this.maxY = maxY;

this.minX = MinX;

this.minY = MinY;

this.NumberOfPoints = NumberOfPoints;

this.probability = probability;

this.Coefficient = Coefficient;

this.LenGradient = LenGradient;

}

public Barnsley\_fern():base()

{

}

public Bitmap DrawBransleyFern(ProgressBar progress)

{

Random random = new Random();

float x0 = 0, y0 = 0;

float x, y;

int width = (int)(picture.Width / (maxX - minX));

int height = (int)(picture.Height / (maxY - minY));

int FunctionIndex = 0;

base.Effects(g, BackgroundColor);

progress.Invoke(new Action(() =>

{

progress.Maximum = NumberOfPoints;

progress.Step = (NumberOfPoints / 100);

}));

for (int i = 1; i <= NumberOfPoints; i++)

{

// генерация числа (0;1)

double randomNum = random.NextDouble();

for (int j = 0; j < probability.Length; j++)

{

randomNum -= probability[j];

if (randomNum <= 0)

{

FunctionIndex = j;

break;

}

}

// вичислення координат

x = Coefficient[FunctionIndex, 0] \* x0 + Coefficient[FunctionIndex, 1] \* y0 + Coefficient[FunctionIndex, 4];

y = Coefficient[FunctionIndex, 2] \* x0 + Coefficient[FunctionIndex, 3] \* y0 + Coefficient[FunctionIndex, 5];

x0 = x;

y0 = y;

// перерахунок пікселів відносно форми

x = (int)(x0 \* width + picture.Width / 2);

y = (int)(y0 \* height);

picture.SetPixel(Math.Abs((int)x) % picture.Width, (int)(Math.Abs(picture.Height - (int)(y)) % picture.Height),

pixels[(int)((((x \* pixels.Count / LenGradient / (picture.Width \* 0.00195))) % pixels.Count))].Color); // розтяг градієнта на весь папоротник

if(i % (progress.Step) == 0) {

progress.Invoke(new Action(() =>

{

progress.PerformStep();

}));

}

}

return picture;

}

public override void Info(TextBox info)

{

info.Text = "The fern is one of the basic examples" +

" of self-similar sets, i.e. it is a mathematically" +

" generated pattern that can be reproducible at any" +

" magnification or reduction. Like the Sierpinski triangle," +

" the Barnsley fern shows how graphically beautiful structures" +

" can be built from repetitive uses of mathematical formulas" +

" with computers. " +

Environment.NewLine +

Environment.NewLine +

"The fern code developed by Barnsley is an" +

" example of an iterated function system(IFS) to create a fractal." +

" This follows from the collage theorem.He has used fractals to" +

" model a diverse range of phenomena in science and technology," +

" but most specifically plant structures." +

Environment.NewLine +

Environment.NewLine +

"IFSs provide models" +

" for certain plants, leaves, and ferns, by virtue of the self" +

" - similarity which often occurs in branching structures in nature." +

"But nature also exhibits randomness and variation from one level" +

" to the next; no two ferns are exactly alike," +

" and the branching fronds become leaves at a smaller scale.";

}

~Barnsley\_fern()

{

Console.WriteLine("Class Barsley\_fern is cleared");

}

}

}

Class CurveDragon.cs :

using System;

using System.Drawing;

using System.Windows.Forms;

namespace FractalClasses

{

public class CurveDragon : Fractals

{

private int i = 0;

public CurveDragon(Bitmap picture, Color BackgroundColor):base(picture, BackgroundColor)

{

}

public CurveDragon():base(){ }

public void DrawCurveDragon(int x1, int y1, int x2, int y2, int Iterations, Pen pen, ProgressBar progress)

{

int NextX, NextY;

if (i == 0)

{

base.Effects(g, BackgroundColor);

}

if (Iterations == 0)

{

g.DrawLine(pen, x1, y1, x2, y2);

}

else if(Iterations > 0)

{

i++;

NextX = (int)((x1 + x2) / 2 + (y2 - y1) / 2);

NextY = (int)((y1 + y2) / 2 - (x2 - x1) / 2);

DrawCurveDragon(x1, y1, NextX, NextY, Iterations - 1, pen, progress);

DrawCurveDragon(x2, y2, NextX, NextY, Iterations - 1, pen, progress);

//progress.Invoke(new Action(() => {

// progress.PerformStep();

//}));

}

}

public override void Info(TextBox info)

{

info.Text = "The Harter dragon, also known as" +

" the Harter-Haytway dragon, was first explored" +

" by NASA physicists John Heighway, Bruce Banks," +

" and William Harter. It was described in 1967 by" +

" Martin Gardner in the Math Games column of Scientific" +

" American. Many of the properties of a fractal have been" +

" described by Chandler Davis and Donald Knuth." +

" The fractal can be created manually. " +

Environment.NewLine +

Environment.NewLine +

"A dragon curve is any member of a family of self-similar" +

" fractal curves, which can be approximated by recursive" +

" methods such as Lindenmayer systems. The dragon curve" +

" is probably most commonly thought of as the shape that" +

" is generated from repeatedly folding a strip of paper in half," +

" although there are other curves that are called dragon curves" +

" that are generated differently." +

Environment.NewLine +

Environment.NewLine +

"For this, we take a segment, bend it in half." +

" Then we iterate over and over again. If we then" +

" unbend the resulting (folded) line again so that all" +

" angles are equal to 90 °, we get a dragon curve.";

}

~CurveDragon()

{

Console.WriteLine("Class CurveDragon is cleared");

}

}

}

**Библиотека класів HelperClasses:**

Class Complex.cs :

using System;

namespace HelperClasses

{

public class Complex

{

public double Re;

public double Im;

public Complex(double a, double b)

{

this.Re = a;

this.Im = b;

}

public void Sqr()

{

double tmp = (Re \* Re) - (Im \* Im);

Im = 2.0d \* Re \* Im;

Re = tmp;

}

public double Magn()

{

return Math.Sqrt((Re \* Re) + (Im \* Im));

}

public void Add(Complex c)

{

Re += c.Re;

Im += c.Im;

}

}

}

Class Pixel.cs :

using System.Drawing;

namespace HelperClasses

{

public class Pixel

{

public Color Color { get; set; }

}

}

**Форми FractalCreator:**

GradientForm.cs:

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using HelperClasses;

namespace FractalsCreator

{

public partial class GradientForm : Form

{

internal List<Pixel> pixels;

internal GradientForm()

{

InitializeComponent();

TopMost = true;

Focus();

this.DoubleBuffered = true;

}

private void GradientForm\_Load(object sender, EventArgs e)

{

}

///

///

/// Gradient

///

///

// getting gradient pixels

internal List<Pixel> GetPixels(Bitmap bitmap)

{

pixels = new List<Pixel>();

for (int x = 0; x < bitmap.Width; x++)

{

pixels.Add(new Pixel()

{

Color = bitmap.GetPixel(x, bitmap.Height / 2) // отримання кольору ряду пікселів з градієнту

});

}

return pixels;

}

// random gradient creation

internal void Gradient(PictureBox pictureBox)

{

Random color = new Random();

int r = color.Next(256);

int g = color.Next(256);

int b = color.Next(256);

int changer = color.Next(256);

Bitmap Gradient = new Bitmap(pictureBox.Width, pictureBox.Height);

switch (changer % 3)

{

case 0:

{

SetGradient(Gradient, r, 255 - g, b, 2);

break;

}

case 1:

{

SetGradient(Gradient, 255 - r, g, b, 1);

break;

}

case 2:

{

SetGradient(Gradient, r, g, 255 - b, 0);

break;

}

default:

break;

}

pictureBox.Image = Gradient;

}

// generation gradient

private void SetGradient(Bitmap Gradient, int r, int g, int b, int NoChangeIndex)

{

int[] change = new int[3];

change[NoChangeIndex] = 0;

for (int x = 0; x < Gradient.Width; x++)

{

for(int i = 0; i < change.Length; i++)

{

if(i != NoChangeIndex)

{

change[i] = x;

}

}

for (int y = 0; y < Gradient.Height; y++)

{

Gradient.SetPixel(x, y, Color.FromArgb(((r + change[0]) / 3) % 255, ((g + change[1]) / 3) % 255, ((b + change[2]) / 3) % 255));

}

}

}

// Random gradient button

private void RandomGradient\_Click(object sender, EventArgs e)

{

Gradient(pictureGradient);

}

// Gradient hot key

private void GradientForm\_KeyDown(object sender, KeyEventArgs e)

{

// generate gradient

if (e.Control && e.KeyCode == Keys.Z)

{

RandomGradient.PerformClick();

}

}

~GradientForm()

{

Console.WriteLine("Form GradientForm is cleared");

}

}

}

FractalForm.cs:

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Globalization;

using System.Threading;

using FractalClasses;

using HelperClasses;

namespace FractalsCreator

{

public partial class FractalForm : Form

{

///

///

/// Start values

///

///

internal Graphics g;

internal Color BackgroundColor = Color.Transparent;

internal List<Pixel> pixels;

internal readonly GradientForm gradientForm;

// start values for MBrot

internal double hx, hy, maxZ, x\_, y\_;

internal double ZoomVal = 1;

internal Size size;

internal double SizeArea;

internal DateTime start, end;

internal int selectFractal = 0;

// start values for fractal tree

internal List<NumericUpDown> Angles;

// start values for curve dragon

internal List<List<NumericUpDown>> Points;

// list of the changing values

internal readonly List<NumericUpDown> changingValues;

NumericUpDown change;

// information about fractals;

internal Fractals info;

// FractalForm FullScreen settings

internal bool FullScreen = false;

///

///

/// Main form properties

///

///

public FractalForm()

{

InitializeComponent();

TopMost = false;

CultureInfo customCulture = (CultureInfo)

Thread.CurrentThread.CurrentCulture.Clone();

customCulture.NumberFormat.NumberDecimalSeparator = ".";

Thread.CurrentThread.CurrentCulture = customCulture;

image.Image = new Bitmap(image.Width, image.Height);

size = image.Size;

g = Graphics.FromImage(image.Image);

this.DoubleBuffered = true;

gradientForm = new GradientForm();

// list of the curves dragon points

Points = new List<List<NumericUpDown>>()

{

new List<NumericUpDown>{ FirstStartPointX, FirstStartPointY, FirstEndPointX, FirstEndPointY },

new List<NumericUpDown>{ SecondStartPointX, SecondStartPointY, SecondEndPointX, SecondEndPointY },

new List<NumericUpDown>{ ThirdStartPointX, ThirdStartPointY, ThirdEndPointX, ThirdEndPointY },

new List<NumericUpDown>{ FourthStartPointX, FourthStartPointY, FourthEndPointX, FourthEndPointY },

new List<NumericUpDown>{ FifthStartPointX, FifthStartPointY, FifthEndPointX, FifthEndPointY },

new List<NumericUpDown>{ SixthStartPointX, SixthStartPointY, SixthEndPointX, SixthEndPointY },

new List<NumericUpDown>{ SeventhStartPointX, SeventhStartPointY, SeventhEndPointX, SeventhEndPointY },

new List<NumericUpDown>{ EighthStartPointX, EighthStartPointY, EighthEndPointX, EighthEndPointY }

};

// list of the angles

Angles = new List<NumericUpDown> {

FirstAngle, SecondAngle, ThirdAngle, FourthAngle

};

// list of the changing values

changingValues = new List<NumericUpDown>

{

Iterations, BranchLength, NumberPoints, DragonIterations

};

}

private void Form1\_Load(object sender, EventArgs e)

{

FractalsList.SelectedIndex = 0;

ShowImageHeight.Value = image.Height;

ShowImageWidth.Value = image.Width;

// change the coordinates of the elements

for (int i = 1; i < Angles.Count; i++)

{

if (i == 1)

{

Angles[i].Location = new Point(Angles[i - 1].Location.X + 132, Angles[i - 1].Location.Y);

}

else

{

Angles[i].Location = new Point(Angles[i - 2].Location.X, Angles[i - 2].Location.Y + 35);

}

}

labelNumberOfCurves.Location = labelZOOM.Location;

NumberOfCurves.Location = new Point(labelNumberOfCurves.Location.X + labelNumberOfCurves.Width + 30, labelNumberOfCurves.Location.Y);

LabelBranchLength.Location = labelZOOM.Location;

BranchLength.Location = ZOOMValue.Location;

labelStartX.Location = y.Location;

labelStartY.Location = LabelMaxZDegreeTwo.Location;

labelBranchWidth.Location = labelIterations.Location;

BranchWidth.Location = Iterations.Location;

StartX.Location = CenterY.Location;

StartY.Location = MaxZDegreeTwo.Location;

labelHorizontal.Location = y.Location;

Horizontal.Location = CenterY.Location;

labelVertical.Location = LabelMaxZDegreeTwo.Location;

Vertical.Location = MaxZDegreeTwo.Location;

labelNumberPoints.Location = labelIterations.Location;

NumberPoints.Location = Iterations.Location;

labelStartPoint.Location = new Point(labelNumberOfCurves.Location.X, labelNumberOfCurves.Location.Y + 28);

labelEndPoint.Location = new Point(labelStartPoint.Location.X + labelStartPoint.Width + 30, labelStartPoint.Location.Y);

labelStartPointX.Location = new Point(labelStartPoint.Location.X, labelStartPoint.Location.Y + 23);

labelStartPointY.Location = new Point(labelStartPointX.Location.X + labelStartPointX.Width + 25, labelStartPointX.Location.Y);

labelEndPointX.Location = new Point(labelEndPoint.Location.X, labelEndPoint.Location.Y + 23);

labelEndPointY.Location = new Point(labelEndPointX.Location.X + labelEndPointX.Width + 25, labelEndPointX.Location.Y);

FirstStartPointX.Location = new Point(labelStartPointX.Location.X, labelStartPointX.Location.Y + 23);

FirstStartPointY.Location = new Point(labelStartPointY.Location.X + 5, labelStartPointY.Location.Y + 23);

FirstEndPointX.Location = new Point(labelEndPointX.Location.X, labelEndPointX.Location.Y + 23);

FirstEndPointY.Location = new Point(labelEndPointY.Location.X + 5, labelEndPointY.Location.Y + 23);

for (int i = 1; i < Points.Count; i++)

{

for (int j = 0; j < 4; j++)

{

Points[i][j].Location = new Point(Points[i - 1][j].Location.X, Points[i - 1][j].Location.Y + 28);

}

}

labelDragonBrashWidth.Location = LabelMaxZDegreeTwo.Location;

DragonBrashWidth.Location = MaxZDegreeTwo.Location;

labelDragonIterations.Location = labelIterations.Location;

DragonIterations.Location = Iterations.Location;

}

private void FractalForm\_Resize(object sender, EventArgs e)

{

if (FractalsList.SelectedIndex == 1)

{

if (StartX.Value > StartX.Minimum && StartX.Value < StartX.Maximum) { StartX.Value += (image.Width - size.Width) / 2; };

if (StartY.Value > StartY.Minimum && StartY.Value < StartY.Maximum) { StartY.Value += (image.Height - size.Height) / 2; };

}

else if (FractalsList.SelectedIndex == 3)

{

for (int i = 0; i < Points.Count; i++)

{

for (int j = 0; j < Points[i].Count; j++)

{

if (Points[i][j].Value > Points[i][j].Minimum && Points[i][j].Value < Points[i][j].Maximum)

{

if (j % 2 == 0)

{

Points[i][j].Value += (image.Width - size.Width) / 2;

}

else

{

Points[i][j].Value += (image.Height - size.Height) / 2;

}

}

}

}

}

if (FractalsList.SelectedIndex != 0)

{

if (BackgroundColor == Color.Transparent)

{

image.BackColor = Color.White;

}

else

{

image.BackColor = BackgroundColor;

}

}

size = image.Size;

ShowImageHeight.Value = image.Height;

ShowImageWidth.Value = image.Width;

}

private void FractalForm\_ResizeEnd(object sender, EventArgs e)

{

if (Progress.Value == Progress.Maximum)

{

DrawFractals();

}

}

private void FractalForm\_KeyDown(object sender, KeyEventArgs e)

{

if (Progress.Value == Progress.Maximum)

{

change = changingValues[FractalsList.SelectedIndex];

// save as

if (e.Control && e.Shift && e.KeyCode == Keys.S)

{

SaveAs.PerformClick();

}

// load gradient

else if (e.Control && e.Shift && e.KeyCode == Keys.Z)

{

LoadGradient.PerformClick();

}

// generate gradient

else if (e.Control && e.KeyCode == Keys.Z)

{

GenerateGradientToolStripMenuItem.PerformClick();

}

// create or update fractal

else if (e.Control && e.Shift && e.KeyCode == Keys.F)

{

if (tabControl.SelectedIndex == 2)

{

ButtonUpdate.PerformClick();

}

else

{

CreateFractal.PerformClick();

}

}

// previous fractal

else if (e.Shift && e.KeyCode == Keys.D1)

{

PreviousFractalToolStripMenuItem.PerformClick();

}

// next fractal

else if (e.Shift && e.KeyCode == Keys.D2)

{

NextFractalToolStripMenuItem.PerformClick();

}

// Open GradientForm

else if (e.Control && e.Shift && e.KeyCode == Keys.X)

{

GradientToolMenuItem.PerformClick();

}

else if (e.KeyCode == Keys.Oemplus && e.Shift)

{

IncreaseToolStripMenuItem.PerformClick();

}

else if (e.KeyCode == Keys.OemMinus && e.Shift)

{

DecreaseToolStripMenuItem.PerformClick();

}

// control keys

else if (FractalsList.SelectedIndex == 0)

{

if (e.Shift && e.KeyCode == Keys.Q)

{

IncreaseTheZoomValueToolStripMenuItem.PerformClick();

}

else if (e.Shift && e.KeyCode == Keys.E)

{

DecreaseTheZoomValueToolStripMenuItem.PerformClick();

}

else if (e.KeyCode == Keys.Q)

{

IncreaseZOOM.PerformClick();

}

else if (e.KeyCode == Keys.E)

{

DecreaseZOOM.PerformClick();

}

}

}

}

///

///

/// Tool Strip

///

///

// Create or update a fractal

private void CreateAndUpdateFractalToolStripMenuItem\_Click(object sender, EventArgs e)

{

CreateFractal.PerformClick();

}

// Open gradient window

private async void Gradient\_Click(object sender, EventArgs e)

{

if (gradientForm.Visible == false)

{

await Task.Run(() => {

gradientForm.ShowDialog();

gradientForm.Focus();

});

}

}

// Generate Gradient

private void GenerateGradientToolStripMenuItem\_Click(object sender, EventArgs e)

{

gradientForm.Gradient(gradientForm.pictureGradient);

}

// Zoom in

private void ZoomInMBrotToolStripMenuItem\_Click(object sender, EventArgs e)

{

IncreaseZOOM.PerformClick();

}

// Zoom out

private void ZoomOutMBrotToolStripMenuItem\_Click(object sender, EventArgs e)

{

DecreaseZOOM.PerformClick();

}

// Increase the zoom value

private void IncreaseZoomValueToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (ZOOMValue.Value < ZOOMValue.Maximum)

{

ZOOMValue.Value += ZOOMValue.Increment;

}

}

// Decrease the zoom value

private void DecreaseTheZoomValueToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (ZOOMValue.Value > ZOOMValue.Minimum)

{

ZOOMValue.Value -= ZOOMValue.Increment;

}

}

// icrease the most impotant value

private void IncreaseToolStripMenuItem\_Click(object sender, EventArgs e)

{

change = changingValues[FractalsList.SelectedIndex];

if (change.Value < change.Maximum)

{

change.Value += change.Increment;

}

}

// decrease the most impotant value

private void DecreaseToolStripMenuItem\_Click(object sender, EventArgs e)

{

change = changingValues[FractalsList.SelectedIndex];

if (change.Value > change.Minimum)

{

change.Value -= change.Increment;

}

}

// next fractal

private void NextFractalToolStripMenuItem\_Click(object sender, EventArgs e)

{

selectFractal += 1;

FractalsList.SelectedIndex = selectFractal % 4;

}

// previous fractal

private void PreviousFractalToolStripMenuItem\_Click(object sender, EventArgs e)

{

selectFractal -= 1;

if (selectFractal == -1)

{

selectFractal = FractalsList.Items.Count - 1;

}

FractalsList.SelectedIndex = selectFractal % 4;

}

///

///

/// Resize image

///

///

private void ShowImageHeight\_ValueChanged(object sender, EventArgs e)

{

this.Height += ((int)ShowImageHeight.Value - image.Height);

image.Height = (int)ShowImageHeight.Value;

}

private void ShowImageWidth\_ValueChanged(object sender, EventArgs e)

{

this.Width += ((int)ShowImageWidth.Value - image.Width);

image.Width = (int)ShowImageWidth.Value;

}

// FullScreen control

private void FractalForm\_SizeChanged(object sender, EventArgs e)

{

if (this.WindowState == FormWindowState.Maximized)

{

CreateFractal.PerformClick();

FullScreen = true;

}

else if (this.WindowState == FormWindowState.Normal && FullScreen != false)

{

FullScreen = false;

CreateFractal.PerformClick();

}

}

///

///

/// UserDialogs

///

///

// Load Gradient

private void LoadGradient\_Click(object sender, EventArgs e)

{

OpenFile.Title = "Load Gradient";

if (OpenFile.ShowDialog() == DialogResult.OK)

{

Bitmap isSufficientSize = new Bitmap(OpenFile.FileName);

int minimumImageWidth = 100; // minimum optimal pictures width for gradient to avoid some problems with division by 0

if (isSufficientSize.Width >= minimumImageWidth)

{

gradientForm.pictureGradient.Image = isSufficientSize;

MessageBox.Show("Gradient loaded successfully", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show($"Image size is too small!\nMinimum image width : {minimumImageWidth}\nCurrent width: {isSufficientSize.Width}", "Warning! Image size is too small", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

// Save image

private void SaveAs\_Click(object sender, EventArgs e)

{

if (image.Image != null)

{

SaveFileDialog SaveAs = new SaveFileDialog();

SaveAs.Title = "Save as";

SaveAs.OverwritePrompt = true;

SaveAs.Filter = "(\*.png)|\*.png|(\*.jpg)|\*.jpg|(\*.bmp)| \*.bmp|All files(\*.\*)|\*.\*";

SaveAs.ShowHelp = true;

if (SaveAs.ShowDialog() == DialogResult.OK)

{

try

{

image.Image.Save(SaveAs.FileName);

MessageBox.Show("Fractal saved successfully", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch

{

MessageBox.Show("Image can't be saved!!!", "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

// selection of the background color for the image

private void ColorButton\_Click(object sender, EventArgs e)

{

ColorDialog BackColor = new ColorDialog();

BackColor.AllowFullOpen = true;

BackColor.FullOpen = true;

BackColor.ShowHelp = true;

BackColor.Color = Color.FromArgb(117, 238, 138);

BackColor.AnyColor = true;

BackColor.CustomColors = new int[]

{

636125, 382980, 5863935, 5427317, 1566114

};

if (BackColor.ShowDialog() == DialogResult.OK)

{

BackgroundColor = BackColor.Color;

ColorButton.BackColor = BackColor.Color;

}

}

///

///

/// Fractals selection and generation

///

///

// selection of the type of fractal

private void FractalsList\_SelectedIndexChanged(object sender, EventArgs e)

{

image.Image = null;

image.BackColor = Color.White;

PointsVisibleTrue(Points);

AnglesVisibleTrue(Angles);

FractalsInfo.Text = null;

CulculationTime.Text = null;

IncreaseToolStripMenuItem.Text = null;

DecreaseToolStripMenuItem.Text = null;

Progress.Step = 1;

labelZOOM.Visible = false;

ZOOMValue.Visible = false;

DecreaseZOOM.Visible = false;

IncreaseZOOM.Visible = false;

ZoomNUM.Visible = false;

x.Visible = false;

y.Visible = false;

CenterX.Visible = false;

CenterY.Visible = false;

LabelMaxZDegreeTwo.Visible = false;

MaxZDegreeTwo.Visible = false;

LabelBranchLength.Visible = false;

BranchLength.Visible = false;

labelMinimalLength.Visible = false;

MinBranchLength.Visible = false;

labelStartX.Visible = false;

labelStartY.Visible = false;

StartX.Visible = false;

StartY.Visible = false;

labelBranchWidth.Visible = false;

BranchWidth.Visible = false;

labelAngles.Visible = false;

NumberOfAngles.Visible = false;

labelBackColor.Visible = false;

ColorButton.Visible = false;

labelIterations.Visible = false;

Iterations.Visible = false;

labelHorizontal.Visible = false;

Horizontal.Visible = false;

labelVertical.Visible = false;

Vertical.Visible = false;

labelNumberPoints.Visible = false;

NumberPoints.Visible = false;

labelNumberOfCurves.Visible = false;

NumberOfCurves.Visible = false;

labelStartPoint.Visible = false;

labelEndPoint.Visible = false;

labelStartPointX.Visible = false;

labelStartPointY.Visible = false;

labelEndPointX.Visible = false;

labelEndPointY.Visible = false;

labelDragonBrashWidth.Visible = false;

DragonBrashWidth.Visible = false;

labelDragonIterations.Visible = false;

DragonIterations.Visible = false;

GroupMouseControl.Visible = false;

zoomInMBrotToolStripMenuItem.Visible = false;

zoomOutMBrotToolStripMenuItem.Visible = false;

IncreaseTheZoomValueToolStripMenuItem.Visible = false;

DecreaseTheZoomValueToolStripMenuItem.Visible = false;

BackgroundColor = Color.Transparent;

ColorButton.BackColor = Color.White;

if (FractalsList.SelectedIndex == 0)

{

info = new MBrotSet();

info.Info(FractalsInfo); // add info about the Mandelbrot set

hx = -0.6;

hy = 0;

maxZ = 4;

CenterX.Text = hx.ToString();

CenterY.Text = hy.ToString();

MaxZDegreeTwo.Text = maxZ.ToString();

ZoomVal = 1;

SizeArea = 4;

image.Enabled = false;

DecreaseZOOM.Enabled = false;

IncreaseZOOM.Enabled = false;

labelZOOM.Visible = true;

ZOOMValue.Visible = true;

DecreaseZOOM.Visible = true;

IncreaseZOOM.Visible = true;

x.Visible = true;

y.Visible = true;

CenterX.Visible = true;

CenterY.Visible = true;

LabelMaxZDegreeTwo.Visible = true;

MaxZDegreeTwo.Visible = true;

labelIterations.Visible = true;

Iterations.Visible = true;

GroupMouseControl.Visible = true;

IncreaseToolStripMenuItem.Text = "Increase the value of the max iterations";

DecreaseToolStripMenuItem.Text = "Decrease the value of the max iterations";

zoomInMBrotToolStripMenuItem.Visible = true;

zoomOutMBrotToolStripMenuItem.Visible = true;

IncreaseTheZoomValueToolStripMenuItem.Visible = true;

DecreaseTheZoomValueToolStripMenuItem.Visible = true;

}

else if (FractalsList.SelectedIndex == 1)

{

info = new FractalTree();

info.Info(FractalsInfo); // add info about the fractal tree

NumberOfAngles.SelectedIndex = 1;

StartX.Value = image.Width / 2;

StartY.Value = image.Height / 5;

labelBackColor.Location = x.Location;

ColorButton.Location = CenterX.Location;

LabelBranchLength.Visible = true;

BranchLength.Visible = true;

labelMinimalLength.Visible = true;

MinBranchLength.Visible = true;

labelStartX.Visible = true;

labelStartY.Visible = true;

labelBranchWidth.Visible = true;

BranchWidth.Visible = true;

StartX.Visible = true;

StartY.Visible = true;

labelAngles.Visible = true;

NumberOfAngles.Visible = true;

AnglesVisibleTrue(Angles, NumberOfAngles.SelectedIndex);

labelBackColor.Visible = true;

ColorButton.Visible = true;

IncreaseToolStripMenuItem.Text = "Increase the value of the branch length";

DecreaseToolStripMenuItem.Text = "Decrease the value of the branch length";

}

else if (FractalsList.SelectedIndex == 2)

{

info = new Barnsley\_fern();

info.Info(FractalsInfo); // add info about the Barnsley fern

labelBackColor.Location = x.Location;

ColorButton.Location = CenterX.Location;

labelBackColor.Visible = true;

ColorButton.Visible = true;

labelHorizontal.Visible = true;

Horizontal.Visible = true;

labelVertical.Visible = true;

Vertical.Visible = true;

labelNumberPoints.Visible = true;

NumberPoints.Visible = true;

IncreaseToolStripMenuItem.Text = "Increase the value of the Numbers points";

DecreaseToolStripMenuItem.Text = "Decrease the value of the Numbers points";

}

else if (FractalsList.SelectedIndex == 3)

{

info = new CurveDragon();

info.Info(FractalsInfo); // add info about the curve of dragon

NumberOfCurves.SelectedIndex = 3;

PointsVisibleTrue(Points, NumberOfCurves.SelectedIndex);

// start point of the dragon curves

for (int i = 0; i < Points.Count; i++)

{

if (i == 0)

{

Points[i][0].Value = image.Width / 2;

Points[i][1].Value = image.Height / 2;

Points[i][2].Value = image.Width / 2 - 300;

Points[i][3].Value = image.Height / 2 - 200;

}

else

{

Points[i][0].Value = Points[i - 1][0].Value;

Points[i][1].Value = Points[i - 1][1].Value;

Points[i][2].Value = Points[i - 1][2].Value - 200;

Points[i][3].Value = Points[i - 1][3].Value - 300;

}

}

labelBackColor.Visible = true;

labelBackColor.Location = y.Location;

ColorButton.Visible = true;

ColorButton.Location = CenterY.Location;

Progress.Visible = true;

labelNumberOfCurves.Visible = true;

NumberOfCurves.Visible = true;

labelStartPoint.Visible = true;

labelEndPoint.Visible = true;

labelStartPointX.Visible = true;

labelStartPointY.Visible = true;

labelEndPointX.Visible = true;

labelEndPointY.Visible = true;

labelDragonBrashWidth.Visible = true;

DragonBrashWidth.Visible = true;

labelDragonIterations.Visible = true;

DragonIterations.Visible = true;

IncreaseToolStripMenuItem.Text = "Increase the value of the Numbers points";

DecreaseToolStripMenuItem.Text = "Decrease the value of the Numbers points";

}

DrawFractals();

}

// button for generating a fractal

private void GenerateFractal\_Click(object sender, EventArgs e)

{

DrawFractals();

}

// drawing Fractals

internal void DrawFractals()

{

Progress.Invoke(new Action(() => {

Progress.Minimum = 0;

Progress.Value = Progress.Minimum;

}));

pixels = gradientForm.GetPixels((Bitmap)gradientForm.pictureGradient.Image);

size = image.Size;

if (FractalsList.SelectedIndex == 0)

{

bool er\_x, er\_y, er\_z;

ZoomNUM.Text = ZoomVal.ToString("F0") + " X";

ZoomNUM.Visible = true;

image.Enabled = true;

DecreaseZOOM.Enabled = true;

IncreaseZOOM.Enabled = true;

CenterX.ForeColor = Color.Black;

CenterY.ForeColor = Color.Black;

MaxZDegreeTwo.ForeColor = Color.Black;

er\_x = double.TryParse(CenterX.Text, out hx);

er\_y = double.TryParse(CenterY.Text, out hy);

er\_z = double.TryParse(MaxZDegreeTwo.Text, out maxZ);

if (!er\_x)

{

CenterX.ForeColor = Color.Red;

MessageBox.Show("Error entering value x!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (!er\_y)

{

CenterY.ForeColor = Color.Red;

MessageBox.Show("Error entering value y!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (!er\_z)

{

MaxZDegreeTwo.ForeColor = Color.Red;

MessageBox.Show("Error entering value max |z| ^ 2!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

DrawMBrot();

}

}

else if (FractalsList.SelectedIndex == 1)

{

image.Image = null;

DrawFractalTree();

}

else if (FractalsList.SelectedIndex == 2)

{

image.Image = null;

DrawBarnsleyFern();

}

else if (FractalsList.SelectedIndex == 3)

{

image.Image = null;

DrawCurveDragon();

}

}

///

///

/// Mandelbrot set

///

///

// Draw MBrot set

private async void DrawMBrot()

{

image.Cursor = Cursors.WaitCursor;

image.Enabled = false; ///

IncreaseZOOM.Enabled = false; ///

DecreaseZOOM.Enabled = false; /// disabled access to change the image

CreateFractal.Enabled = false; ///

FractalsList.Enabled = false; ///

start = DateTime.Now;

ZoomNUM.Width = (int)(ZoomNUM.Text.Length \* ZoomNUM.Font.Height \* 0.4);

image.Invalidate();

pixels = gradientForm.GetPixels((Bitmap)gradientForm.pictureGradient.Image);

Bitmap pictureMBrotSet = new Bitmap(image.Width, image.Height);

MBrotSet MBrotSet = new MBrotSet((Bitmap)pictureMBrotSet, pixels, size, gradientForm.pictureGradient, (int)Iterations.Value);

int CountColors = 100;

await Task.Run(() =>

{

pictureMBrotSet = MBrotSet.CalculationMBrot(hx, hy, x\_, y\_, maxZ, SizeArea, CountColors, Progress);

image.Image = pictureMBrotSet;

});

end = DateTime.Now;

CulculationTime.Text = (end - start).TotalMilliseconds.ToString("F2") + " ms";

image.Enabled = true; ///

IncreaseZOOM.Enabled = true; ///

DecreaseZOOM.Enabled = true; /// added access to change the image

CreateFractal.Enabled = true; ///

FractalsList.Enabled = true; ///

image.Cursor = Cursors.Default;

}

// Control buttons for MBrot set

private void DecreaseZOOM\_Click(object sender, EventArgs e)

{

SizeArea \*= (double)ZOOMValue.Value;

CenterX.Text = hx.ToString();

CenterY.Text = hy.ToString();

ZoomVal /= (double)ZOOMValue.Value;

ZoomNUM.Visible = true;

ZoomNUM.Clear();

ZoomNUM.Text = ZoomVal.ToString("F2") + " X";

DrawFractals();

}

private void IncreaseZOOM\_Click(object sender, EventArgs e)

{

SizeArea /= (double)ZOOMValue.Value;

CenterX.Text = hx.ToString();

CenterY.Text = hy.ToString();

ZoomVal \*= (double)ZOOMValue.Value;

ZoomNUM.Visible = true;

ZoomNUM.Clear();

if (ZoomVal < 1)

{

ZoomNUM.Text = ZoomVal.ToString("F4") + " X";

}

else

{

ZoomNUM.Text = ZoomVal.ToString("F2") + " X";

}

DrawFractals();

}

private void Image\_MouseClick(object sender, MouseEventArgs e)

{

size = image.Size;

int X = e.X, Y = e.Y;

if (FractalsList.SelectedItem == FractalsList.Items[0])

{

ZoomNUM.Visible = true;

if (e.Button == MouseButtons.Left)

{

hx = (hx - SizeArea / 2) + X \* (SizeArea / size.Width);

hy = (hy - SizeArea / 2) + Y \* (SizeArea / size.Height);

SizeArea /= (double)ZOOMValue.Value; // increase zoom by ZOOMValue value

CenterX.Text = hx.ToString();

CenterY.Text = hy.ToString();

ZoomVal \*= (double)ZOOMValue.Value;

ZoomNUM.Clear();

ZoomNUM.Text = ZoomVal.ToString("F2") + " X";

DrawFractals();

}

else if (e.Button == MouseButtons.Middle)

{ // back to default

SizeArea = 4;

hx = -0.6;

hy = 0;

CenterX.Text = hx.ToString();

CenterY.Text = hy.ToString();

ZoomVal = 1;

ZoomNUM.Clear();

ZoomNUM.Width = 80;

ZoomNUM.Text = ZoomVal.ToString("F2") + " X";

DrawFractals();

}

else if (e.Button == MouseButtons.Right)

{

x\_ = (hx - SizeArea / 2) + X \* (SizeArea / size.Width);

y\_ = (hy - SizeArea / 2) + Y \* (SizeArea / size.Height);

SizeArea \*= (double)ZOOMValue.Value; // decrease zoom by ZOOMValue value

CenterX.Text = x\_.ToString();

CenterY.Text = y\_.ToString();

ZoomVal /= (double)ZOOMValue.Value;

ZoomNUM.Clear();

if (ZoomVal < 1)

{

ZoomNUM.Text = ZoomVal.ToString("F4") + " X";

}

else

{

ZoomNUM.Text = ZoomVal.ToString("F2") + " X";

}

DrawFractals();

}

}

}

///

///

/// Fractal tree

///

///

// selection of the number of angles

private void NumberOfAngles\_SelectedIndexChanged(object sender, EventArgs e)

{

AnglesVisibleTrue(Angles);

AnglesVisibleTrue(Angles, NumberOfAngles.SelectedIndex);

}

// Draw fractal tree

private async void DrawFractalTree()

{

image.Cursor = Cursors.WaitCursor;

FractalsList.Enabled = false;

CreateFractal.Enabled = false;

start = DateTime.Now;

image.Invalidate();

Bitmap pictureTree = new Bitmap(image.Width, image.Height);

int branchLength = (int)BranchLength.Value;

double[] angles = new double[NumberOfAngles.SelectedIndex + 1];

for (int i = 0; i < angles.Length; i++)

{

angles[i] = (double)Angles[i].Value;

}

FractalTree tree = new FractalTree(pictureTree, pixels, angles, (int)(MinBranchLength.Value), (int)(BranchWidth.Value), BackgroundColor);

await Task.Run(() =>

{

tree.DrawFractalTree((int)(StartX.Value), (int)(StartY.Value), branchLength, 0, Progress);

image.Image = pictureTree;

});

if (NumberOfAngles.SelectedIndex == 0)

{

Progress.Maximum = 10;

}

Progress.Value = Progress.Maximum;

end = DateTime.Now;

CulculationTime.Text = (end - start).TotalMilliseconds.ToString("F2") + " ms";

CreateFractal.Enabled = true;

FractalsList.Enabled = true;

image.Cursor = Cursors.Default;

}

// changing the number of visible branches

private void AnglesVisibleTrue(List<NumericUpDown> Angles, int endIndex = -1)

{

if (endIndex == -1)

{

for (int i = 0; i < Angles.Count; i++)

{

Angles[i].Visible = false;

}

}

else

{

for (int i = 0; i <= endIndex; i++)

{

Angles[i].Visible = true;

}

}

}

// calculation of the changing mininum value of the branch length

private void BranchLength\_ValueChanged(object sender, EventArgs e)

{

if (NumberOfAngles.SelectedIndex == 3)

{

MinBranchLength.Minimum = (int)(Math.Floor(BranchLength.Value / 40));

}

else if (NumberOfAngles.SelectedIndex == 2)

{

MinBranchLength.Minimum = (int)(Math.Floor(BranchLength.Value / 60));

}

else

{

MinBranchLength.Minimum = (int)(Math.Floor((double)BranchLength.Value \* 0.005)) + 1;

}

if (MinBranchLength.Value < MinBranchLength.Minimum)

{

MinBranchLength.Value = MinBranchLength.Minimum;

}

}

///

///

/// Fern Barnsley

///

///

// Draw fern Barnsley

private async void DrawBarnsleyFern()

{

image.Cursor = Cursors.WaitCursor;

start = DateTime.Now;

image.Invalidate();

CreateFractal.Enabled = false;

FractalsList.Enabled = false;

float[] probability = new float[4] { 0.01f, 0.06f, 0.08f, 0.85f };

int NumbersOfPoint = (int)NumberPoints.Value;

float maxX = 3 + (float)Horizontal.Value, maxY = 10.1f + (float)Vertical.Value;

float[,] Coefficient = new float[4, 6]

{

{0, 0, 0, 0.16f, 0, 0},

{-0.15f, 0.28f, 0.26f, 0.24f, 0, 0.44f},

{0.2f, -0.26f, 0.23f, 0.22f, 0, 1.6f},

{0.85f, 0.04f, -0.04f, 0.85f, 0, 1.6f}

};

Bitmap pictureFern = new Bitmap(image.Width, image.Height);

Barnsley\_fern Fern = new Barnsley\_fern(pictureFern, maxX, maxY, NumbersOfPoint, probability, Coefficient, pixels, BackgroundColor, gradientForm.pictureGradient.Width);

await Task.Run(() =>

{

pictureFern = Fern.DrawBransleyFern(Progress);

image.Image = pictureFern;

});

end = DateTime.Now;

CulculationTime.Text = (end - start).TotalMilliseconds.ToString("F2") + " ms";

CreateFractal.Enabled = true;

FractalsList.Enabled = true;

image.Cursor = Cursors.Default;

}

///

///

/// Dragon curve

///

///

// Draw dragon curve

private async void DrawCurveDragon()

{

image.Cursor = Cursors.WaitCursor;

start = DateTime.Now;

image.Invalidate();

CreateFractal.Enabled = false;

FractalsList.Enabled = false;

int brushWidth = (int)(DragonBrashWidth.Value);

int CountIterations;

Bitmap pictureCurveDragon = new Bitmap(image.Width, image.Height);

List<int> indexes = new List<int>();

List<List<int>> Coords = new List<List<int>>();

List<int> coordinates;

for (int i = 0; i <= NumberOfCurves.SelectedIndex; i++)

{

coordinates = new List<int>();

for (int j = 0; j < 4; j++)

{

coordinates.Add((int)Points[i][j].Value);

}

Coords.Add(coordinates);

}

// obtaining color for colloring

for (int i = 0; i < Coords.Count; i++)

{

indexes.Add((int)((i + 1) \* gradientForm.pictureGradient.Image.Width / (Coords.Count + 1)));

}

CurveDragon Dragon = new CurveDragon(pictureCurveDragon, BackgroundColor);

CountIterations = (int)(DragonIterations.Value);

Progress.Maximum = 0;

for (int i = CountIterations; i >= 0; i--)

{

Progress.Maximum += (int)Math.Pow(2, i);

}

Progress.Maximum \*= (Coords.Count);

await Task.Run(() => {

for (int i = 0; i < Coords.Count; i++)

{

Dragon.DrawCurveDragon(Coords[i][0], image.Height - Coords[i][1], Coords[i][2], image.Height - Coords[i][3], CountIterations, new Pen(pixels[indexes[i]].Color, brushWidth), Progress);

}

image.Image = pictureCurveDragon;

Progress.Invoke(new Action(() =>

{

Progress.Value = Progress.Maximum;

}));

});

end = DateTime.Now;

CulculationTime.Text = (end - start).TotalMilliseconds.ToString("F2") + " ms";

CreateFractal.Enabled = true;

FractalsList.Enabled = true;

image.Cursor = Cursors.Default;

}

// selection of the number of dragon curves

private void NumberOfCurves\_SelectedIndexChanged(object sender, EventArgs e)

{

PointsVisibleTrue(Points);

PointsVisibleTrue(Points, NumberOfCurves.SelectedIndex);

}

// changing the number of visible curves

internal void PointsVisibleTrue(List<List<NumericUpDown>> Points, int endIndex = -1)

{

if (endIndex == -1)

{

for (int i = 0; i < Points.Count; i++)

{

for (int j = 0; j < Points[i].Count; j++)

{

Points[i][j].Visible = false;

}

}

}

else

{

for (int i = 0; i <= endIndex; i++)

{

for (int j = 0; j < Points[i].Count; j++)

{

Points[i][j].Visible = true;

}

}

}

}

~FractalForm()

{

Console.WriteLine("Class FractalForm is clear");

}

}

}