**Общая информация о проекте**

**Название проекта:** VideoGameStore  
**Тип приложения:** Десктопное приложение-магазин цифровых видеоигр  
**Аналог:** Steam  
**Целевая платформа:** PC XBOX Playstation Mobile  
**Цель проекта:** Создание минималистичного и функционального приложения для управления и просмотра цифровых видеоигр с разделением прав на администратора и пользователя.

**Технологический стек**

* **Платформа:** .NET 8.0 — современная кроссплатформенная среда разработки от Microsoft, обеспечивающая высокую производительность и широкую поддержку библиотек.
* **Интерфейс пользователя:** WPF (Windows Presentation Foundation) — технология построения UI на базе XAML и C#.
* **Архитектура приложения:** MVVM (Model-View-ViewModel) — шаблон проектирования, обеспечивающий четкое разделение логики, представления и данных. Упрощает масштабирование и поддержку кода.
* **База данных:** SQLite, реализована через пакет Microsoft.Data.Sqlite. Легковесное и встраиваемое решение для локального хранения данных.
* **Стилизация:** XAML-ресурсы, включая глобальные стили и анимации для более привлекательного визуального оформления.

**Функционал административной панели**

Администратор имеет доступ к панели управления играми.

Доступные функции:

**Управление списком игр:**

* **Просмотр списка игр:**
  + Реализован через DataGrid, где каждая строка отображает информацию об одной игре.
  + Колонки: ID, Название, Жанр, Цена, Платформа, Ключ активации.
* **Добавление и редактирование игр:**
  + Открывается отдельное модальное окно AddEditGameWindow.xaml.
  + Форма включает поля с валидацией (проверка на пустоту, корректность цены и уникальность ключа).
* **Удаление игр:**
  + Подтверждение удаления и удаление выбранной записи из базы данных.
* **Навигация:**
  + Кнопка "Открыть магазин" позволяет перейти в интерфейс покупателя, представленный в ShopWindow.xaml.

**Интерфейс магазина (пользовательская часть)**

Интерфейс магазина предназначен для просмотра доступных игр без возможности редактирования. Реализовано:

* Отображение списка игр с фильтрацией по жанрам и платформам (в перспективе).
* Просмотр подробной информации о выбранной игре.
* Минималистичный пользовательский опыт с фокусом на визуальную привлекательность.

**Особенности дизайна и визуального оформления**

* **Цветовая палитра:**  
  Основные цвета — фиолетовые оттенки:  
  #FF3700B3, #FF6200EE, #FFBB86FC — применяются для фона, подсветки, кнопок.
* **Элементы интерфейса:**
  + Все кнопки и поля ввода имеют **закругленные углы**, создавая современный внешний вид.
  + Анимации добавлены для обратной связи с пользователем (например, визуальное выделение при наведении на кнопку "Сохранить").
* **Стилизация таблиц:**
  + Выделение активной строки.
  + Заголовки столбцов оформлены в соответствии с общей темой интерфейса.
* **Глобальные стили:**
  + Определены в App.xaml, что упрощает изменение внешнего вида и повторное использование компонентов.

**Структура проекта**

**Модели:**

* Game.cs:  
  Содержит свойства:
  + ID (уникальный идентификатор),
  + Название,
  + Жанр,
  + Цена,
  + Платформа,
  + Ключ активации.

**Окна (Views):**

* MainWindow.xaml — административная панель с таблицей и кнопками управления.
* AddEditGameWindow.xaml — окно для добавления или редактирования игр.
* ShopWindow.xaml — витрина магазина для обычного пользователя.

**ViewModels:**

* Классы ViewModel для каждого окна реализуют бизнес-логику, связывают интерфейс и данные, обрабатывают команды, взаимодействуют с базой.

**Ресурсы:**

* Стили кнопок, текстов, полей ввода и анимаций задаются в App.xaml.

**Логика работы приложения**

1. **Запуск приложения** 🡪 открывается MainWindow с таблицей всех игр.
2. **Добавление/редактирование** 🡪 по нажатию на соответствующую кнопку открывается AddEditGameWindow, где происходит работа с данными.
3. **Сохранение данных** 🡪 через ViewModel информация обновляется в базе SQLite.
4. **Навигация** 🡪 по нажатию кнопки "Открыть магазин" происходит переход к пользовательскому окну ShopWindow.

**Преимущества архитектурного решения**

* **Масштабируемость:** благодаря паттерну MVVM легко добавлять новые функции без изменения существующего кода.
* **Поддерживаемость:** четкое разделение логики, представления и модели данных упрощает внесение изменений.
* **Повторное использование:** XAML-ресурсы и ViewModel-команды можно использовать в других проектах.
* **Простота тестирования:** логика приложения отделена от интерфейса, что позволяет легко проводить модульное тестирование.