



АННОТАЦИЯ

Целью дипломной работы являлось создание веб-приложения для подбора одежды и формирования капсульных гардеробов, ориентированного на индивидуальных пользователей. В результате выполнения работы было разработано и протестировано готовое веб-приложение, соответствующее поставленным требованиям по функциональности и удобству использования.

Практическая значимость работы заключается в разработке интуитивно понятного инструмента, который упрощает процесс создания стильных образов и оптимизирует подбор одежды для повседневного использования.

Пояснительная записка содержит 46 с., 36 рис., 5 источников, 2 прил.

SUMMARY

The aim of the thesis was to create a web application for selecting clothing and forming capsule wardrobes, focused on individual users. As a result of the work, a complete web application was developed and tested, meeting the requirements for functionality and ease of use. The practical significance of the work lies in the development of an intuitive tool that simplifies the process of creating stylish looks and optimises clothing selection for everyday use. The explanatory note contains 46 pages, 36 figures, 5 sources, and 2 appendices.

СОДЕРЖАНИЕ

[Введение 5](#_Toc198538779)

[1 Анализ предметной области 6](#_Toc198538780)

[1.1 Описание предметной области 6](#_Toc198538781)

[1.2 Анализ существующих решений 7](#_Toc198538782)

[1.3 Потребности пользователей и заинтересованных лиц 8](#_Toc198538783)

[2 Постановка задачи 9](#_Toc198538784)

[2.1 Требования к разрабатываемой системе 9](#_Toc198538785)

[2.1.1 Функциональные требования 9](#_Toc198538786)

[2.1.2 Нефункциональные требования 10](#_Toc198538787)

[2.2 Обоснование выбора инструментальных средств. 10](#_Toc198538788)

[2.2.1 Выбор языка программирования. 10](#_Toc198538789)

[2.3 Определение этапов реализации проекта 11](#_Toc198538790)

[3 Проектирование приложения 13](#_Toc198538791)

[3.1 Архитектурное проектирование 13](#_Toc198538792)

[3.5 Диаграммы uml 21](#_Toc198538793)

[4 Разработка приложения 27](#_Toc198538794)

[4.1 Реализация архитектуры 27](#_Toc198538795)

[4.2 Реализация пользовательского интерфейса 30](#_Toc198538796)

[5. Тестирование 36](#_Toc198538797)

[5.1 Тест-требования 36](#_Toc198538798)

[5.2 Тест-план 36](#_Toc198538799)

[5.3 Результаты тестирования 38](#_Toc198538800)

[Заключение 40](#_Toc198538801)

[Список использованных источников 41](#_Toc198538802)

[Приложение А 42](#_Toc198538803)

[Приложение В 45](#_Toc198538804)

введение

Целью выполнения работы является создание веб-приложения по подбору одежды и аксессуаров. Приложение должно содержать функционал для загрузки фотографий, подбору капсульного и тематического капсульных гардеробов. Для достижения поставленной цели необходимо решить следующие задачи:

* провести анализ существующих решений по подбору капсульного гардероба;
* определить основные требования к веб-приложению, исходя из потребностей целевой аудитории;
* разработать архитектуру и дизайн интерфейса веб-приложения;
* провести тестирование и оптимизацию приложения для обеспечения его стабильности и эффективной работы;
* отслеживание изменений с помощью GitHub.

В ходе преддипломной практики будет разработано веб-приложение, предоставляющее пользователям удобный инструмент для подбора одежды и аксессуаров, формирования капсульных гардеробов и получения стилевых рекомендаций.

1. Анализ предметной области

1.1 Описание предметной области

Тема преддипломной практики: «Разработка веб-приложения для подбора одежды и аксессуаров».

Планируется создать приложение, которое поможет пользователям формировать индивидуальные капсульные гардеробы на основе их предпочтений, событий и имеющейся одежды. Также планируется реализовать подбор капсульного гардероба из всей имеющейся одежды.

В ходе разработки приложения будут реализованы следующие функциональные требования:

* Регистрация и авторизация пользователей.
* Управление личным гардеробом (добавление, категоризация, редактирование, удаление).
* Генерация капсульных гардеробов и индивидуальных капсул.
* Фильтрация по цвету, типу, событиям.
* Добавление в избранное готовых решений.

Планируемые входные параметры:

* Данные пользователя (логин, пароль, имя пользователя).
* Фотографии и описания предметов одежды.
* Критерии подбора (цвет, стиль, события).

Планируемые выходные параметры:

* Сгенерированные капсульные гардеробы.
  1. Анализ существующих решений

В процессе поиска аналогов были найдены следующие приложения:

Свободно распространяемые аналоги:

Примерочная одежды по фото онлайн.

Визуальное представление сайта представлено на рисунке 1:

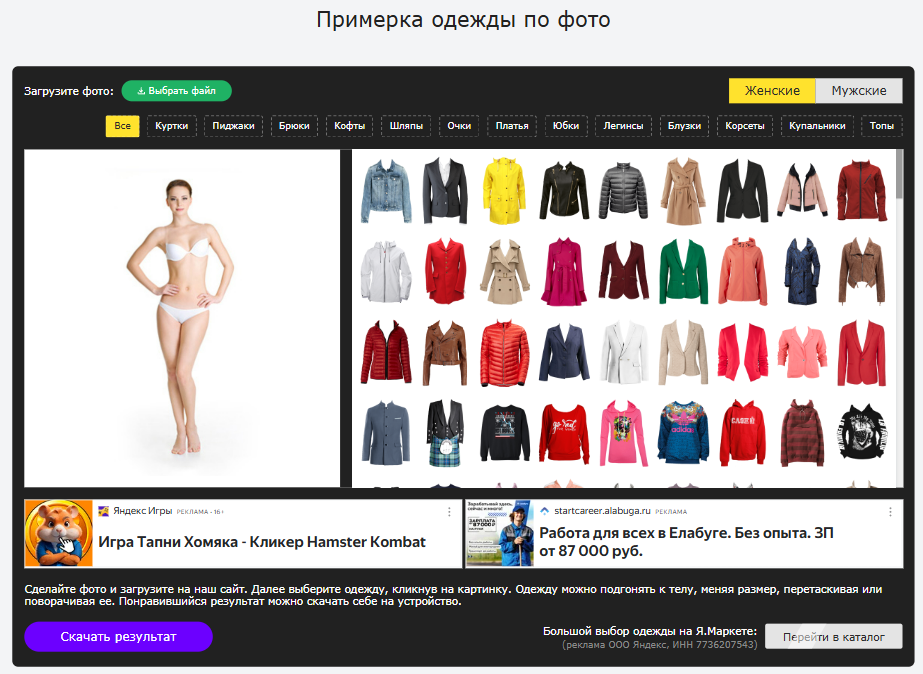


Рисунок 1 – Визуальное представление сайта «Примерочная одежды по фото онлайн»

Набор функций:

– Загрузка фотографии модели.

– Примерка одежды на модель онлайн.

– Загрузка получившейся капсулы одежды.

– Фильтрация одежды по гендеру и типу.

Достоинства:

* Большой выбор одежды и аксессуаров.
* Онлайн примерка на модель.
* Можно сохранить набор одежды себе на устройство.
* Не требует регистрации.

Недостатки:

* Точность примерки.
* Нет возможности загрузить свои фотографии одежды и аксессуаров.
  1. Потребности пользователей и заинтересованных лиц

Данное приложения будет помогать создавать для пользователя подборку капсульного и индивидуального гардеробов на основе загруженных фотографий пользователя.

Целевая аудитория:

* Люди, которые хотят обновить свой гардероб и создать стильный образ.
* Те, кто ищет индивидуальные советы по стилю и подбору одежды.
* Люди, которые хотят избегать ненужных покупок и создавать гармоничный капсульный гардероб.

Таким образом, приложение поможет пользователям легко систематизировать свой гардероб, создавать модные образы из уже имеющихся вещей, экономя время и бюджет на подбор одежды.

1. Постановка задачи

### 2.1 Требования к разрабатываемой системе

### 2.1.1 Функциональные требования

Системными характеристики:

* Разрабатываемая система является веб-приложением.
* Приложение разрабатывается на языке программирования С#.
* Приложение работает с базой данных SQLite.

Пользовательские требования:

* Регистрация и авторизация.
* Загрузка фото.
* Добавление тегов (тип одежды, цвет, сезонность, стиль и т. д.).
* Удаление и редактирование элементов гардероба.
* Создание тематических подборок (например, «офисный стиль», «вечерний лук»).
* Генерация общих капсул.
* Сохранение понравившихся капсул в избранное.
* Удаление из избранного.

### 2.1.2 Нефункциональные требования

Анализ предметной области позволил определить следующие нефункциональные требования:

* Должны быть реализованы различные уровни доступа для различных ролей пользователей (администратор, рядовой пользователь).
* Интерфейс должен быть интуитивно понятным и удобным для пользователей.
* Система должна минимизировать вероятность сбоев и свести к минимуму вероятность аварийного завершения работы.
* Пароли не должны храниться в открытом виде.
* Время загрузки фото – не более 5 секунд.
* Генерация капсул – не более 30 секунд.
* Поддержка до 10 пользователей одновременно

### 2.2 Обоснование выбора инструментальных средств.

### 2.2.1 Выбор языка программирования.

Бэкенд: ASP.NET Core (C#)

* Производительность: ASP.NET Core — высокопроизводительный фреймворк, оптимизированный для современных веб-приложений.
* Гибкость: поддержка MVC, Web API, Razor Pages.
* Кроссплатформенность.

База данных: SQLite1]

Простота: не требует отдельного сервера, данные хранятся в одном файле.

Фронтенд: (CSHTML) + CSS

* Интеграция с бэкендом: прямая работа с моделями C# (@model).

Среда разработки: Visual Studio

* Полноценная среда для С#, ASP.NET, фронтенда и работы с БД.
* Встроенные инструменты без необходимости настраивать кучу программ.

Система контроля версий: GitHub

* Надежное сохранение кода.
* Возможность отката к любой предыдущей версии.
* Визуальное отслеживание изменений в коде.

2.3 Определение этапов реализации проекта

Для успешной реализации дипломного проекта работу следует организовать поэтапно.

Подготовительный этап:

* Формулировка целей и задач проекта.
* Разработка технического задания.
* Исследование существующих аналогов.

Проектирование:

* Создание архитектурного решения.
* Проектирование структуры базы данных.

Разработка приложения:

* Создание интерфейсных компонентов.
* Настройка навигации между разделами.
* Разработка контроллеров.

Тестирование:

* Проверка функциональности системы.
* Устранение обнаруженных ошибок.
* Оптимизация производительности.

Завершение:

* Финальная доработка приложения.
* Оформление документации.
* Подготовка к защите.

3 Проектирование приложения

3.1 Архитектурное проектирование

Разрабатываемая система представляет собой веб-приложение, основанное на архитектурном шаблоне Model-View-Controller и использующее Entity Framework Core для работы с базой данных.

3.2 Проектирование базы данных

В результате исследования предметной области была разработана ER - диаграмма базы данных, выполненная в нотации Мартина. На рисунке 3.1 представлена визуализация спроектированной модели данных.

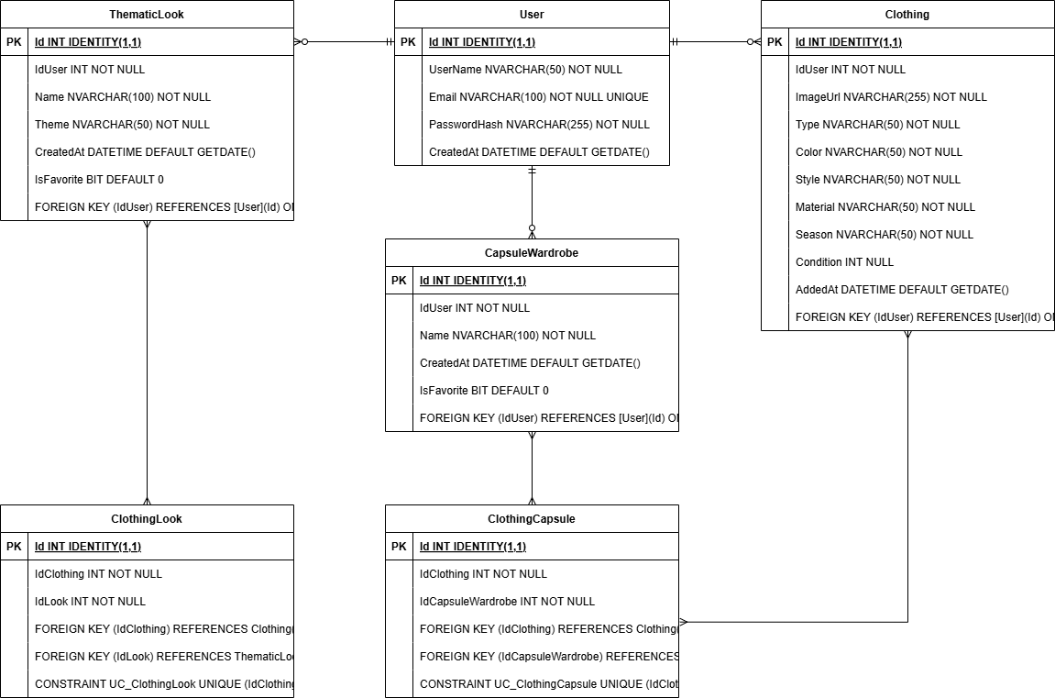


Рисунок 3. – ER-диаграмма базы данных

Эта модель наглядно отображает структуру базы данных системы, включая все основные сущности, их атрибуты и взаимосвязи между ними. Она служит основой для организации хранения информации и обеспечивает целостность данных при работе приложения. Визуальное представление в виде ER-диаграммы помогает лучше понять логику работы с данными и служит руководством при реализации физической модели базы данных.

3.3 Проектирование интерфейса пользователя

На рисунке 3.2 представлен макет интерфейса авторизации.

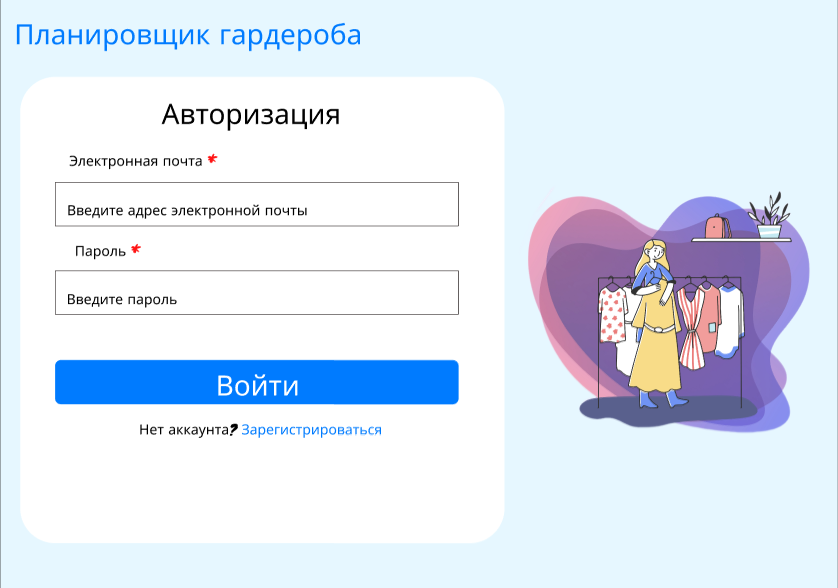


Рисунок 3. – Страница авторизации

На рисунке 3.3 представлен макет интерфейса регистрации.

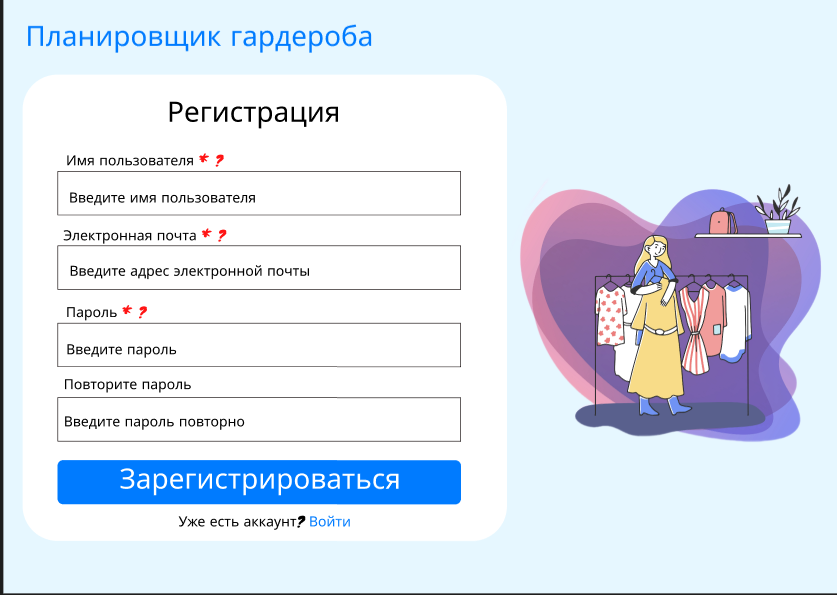


Рисунок 3. – Страница регистрации

На рисунке 3.4 представлен макет интерфейса загрузки фотографий одежды.

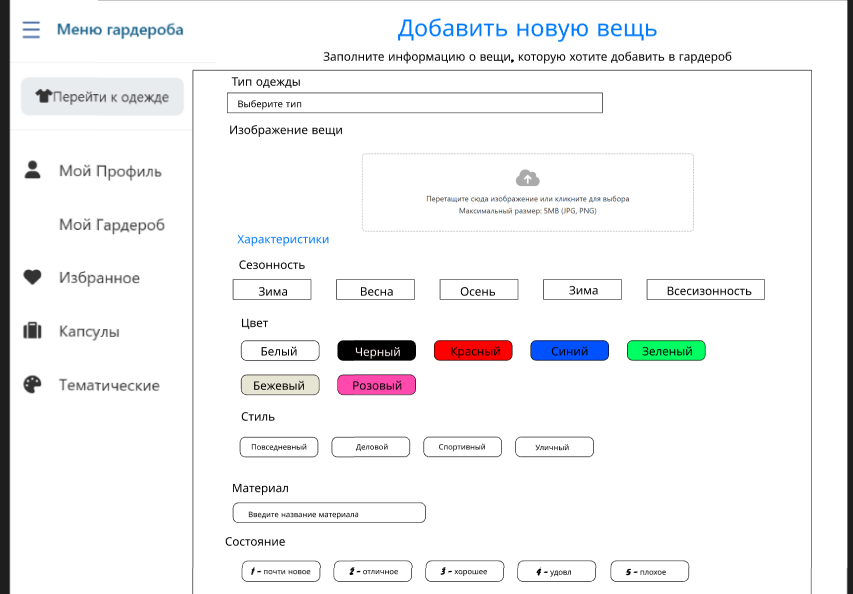


Рисунок 3. – Страница загрузки изображения одежды

На рисунке 3.5 представлен макет интерфейса главной страницы.

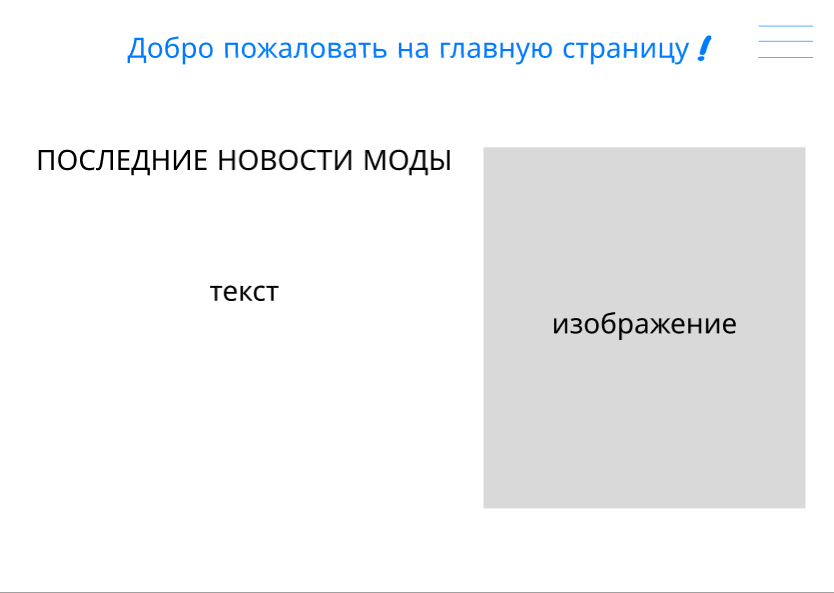


Рисунок 3. – Главная страница

На рисунке 3.6 представлен макет интерфейса страницы гардероба пользователя.

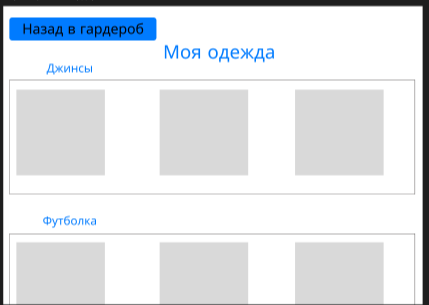


Рисунок 3. – Страница гардероба пользователя

На рисунке 3.7 представлен макет интерфейса профиля пользователя.

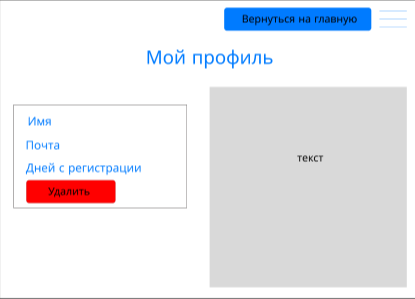


Рисунок 3. – Страница профиля пользователя

На рисунке 3.8 представлен макет интерфейса страницы капсульного гардероба.

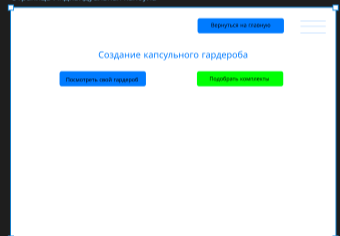


Рисунок 3. – Страница капсульного гардероба

На рисунке 3.9 представлен макет интерфейса тематической капсулы.

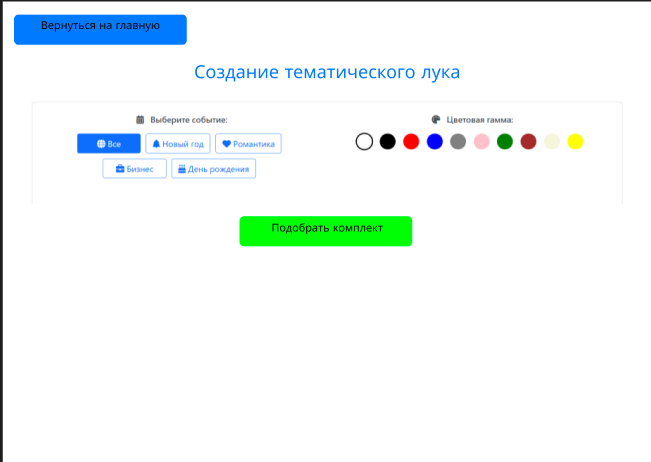


Рисунок 3. – Страница тематической капсулы

На рисунке 3.10 представлен макет интерфейса избранного.

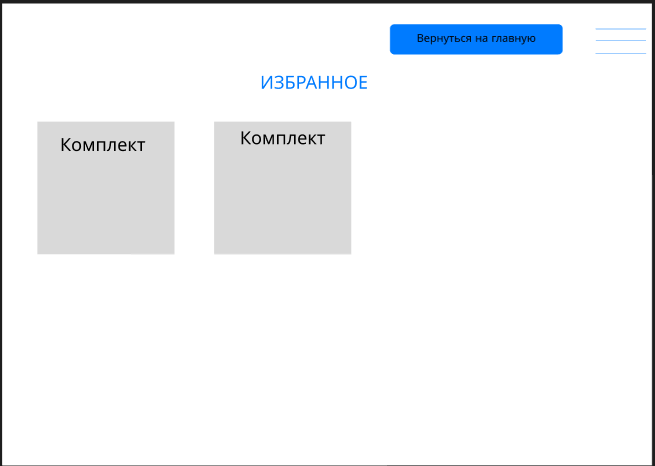


Рисунок 3. – Страница избранного

Таким образом был разработан макет пользовательского интерфейса для разрабатываемого продукта. Макет может отличаться от конечного вида интерфейса приложения, так как в процесса разработки могут быть внесены изменения.

3.4 Проектирование модулей и компонентов

Модуль аутентификации.

Функции:

* Регистрация пользователя.
* Авторизация пользователя.

Входные данные.

Для регистрации:

* Имя пользователя.
* Email.
* Пароль.

Для авторизации:

* Email.
* Пароль.

Выходные данные:

* Успешная авторизация/регистрация.
* Обработка ошибок.

Модуль управлением гардероба.

Функции:

* Добавление вещи.
* Просмотр вещей по категориям.
* Общий список вещей.

Входные данные.

* Тип одежда.
* Изображение.
* Цвет.
* Стиль.
* Материал.
* Сезон.
* Состояние.

Выходные данные:

* Успешное добавление.

Модуль генерации капсул.

Функции:

* Генерация базовых комплектов.
* Просмотр вещей для капсул.

Входные данные.

* Не требуется.

Выходные данные:

* Сгенерированный гардероб.

Модуль генерации тематического образа.

Функции:

* Генерации образа по теме.
* Просмотр доступных вещей.

Входные данные.

* Цвет.
* Событие.

Выходные данные:

* Сгенерированная тематическая капсула.

Модуль профиля пользователя.

Функции:

* Просмотр профиля.
* Удаление аккаунта.

Входные данные.

* Не требуется.

Выходные данные:

* Данные пользователя.

3.5 Диаграммы UML

Диаграмма классов – представляет описание структуры классов в системе и их взаимосвязи. Диаграмма классов представлена на рисунке 3.11.

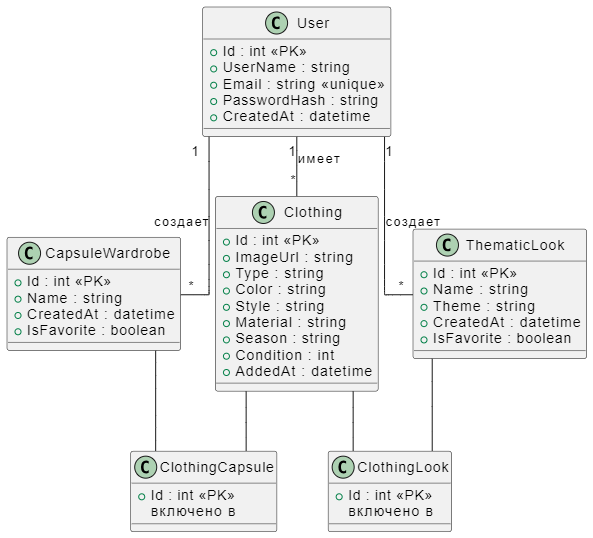


Рисунок 3. – Диаграмма классов

Таким образом диаграмма классов предоставляет четкое представление о структуре системы и взаимосвязях между ее компонентами.

Диаграмма последовательности – отображают взаимодействия между объектами, порядок выполнения операций и обмена информацией во времени.

Диаграмма последовательности регистрации представлена на рисунке 3.12.

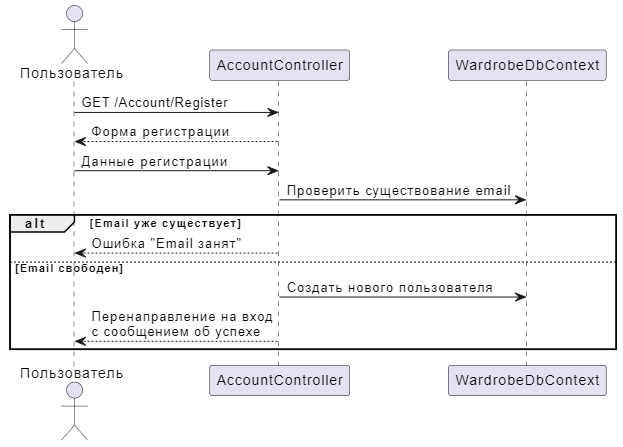


Рисунок 3. – Диаграмма последовательности для регистрации

Диаграмма последовательности авторизации представлена на рисунке 3.13.

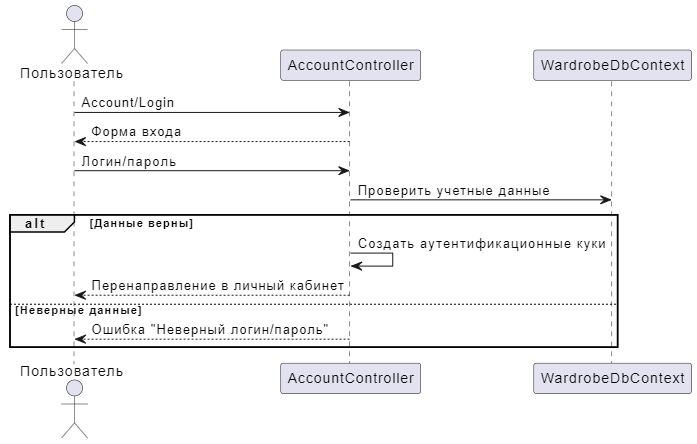


Рисунок 3. – Диаграмма последовательности для авторизации

Диаграмма последовательности добавления фотографий одежды представлена на рисунке 3.14.

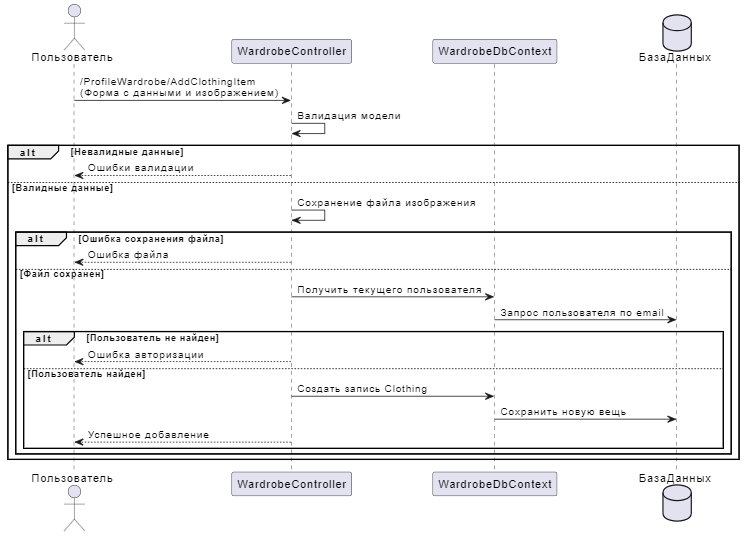


Рисунок 3. – Диаграмма последовательности по добавлению одежды

Диаграмма последовательности генерации капсульного гардероба представлена на рисунке 3.15.

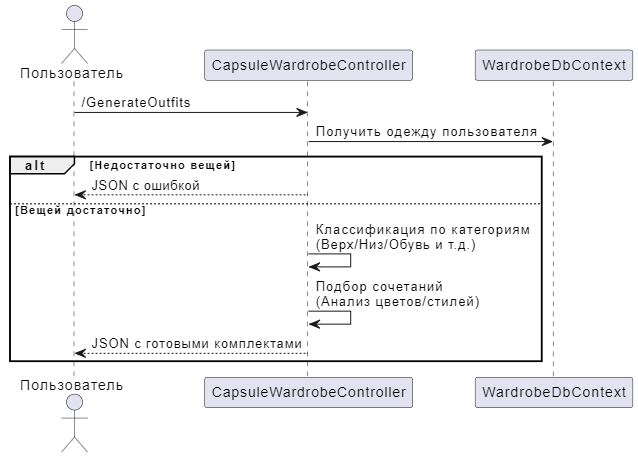


Рисунок 3. – Диаграмма последовательности для генерации капсульного гардероба

Диаграмма последовательности генерации тематической капсулы представлена на рисунке 3.16.

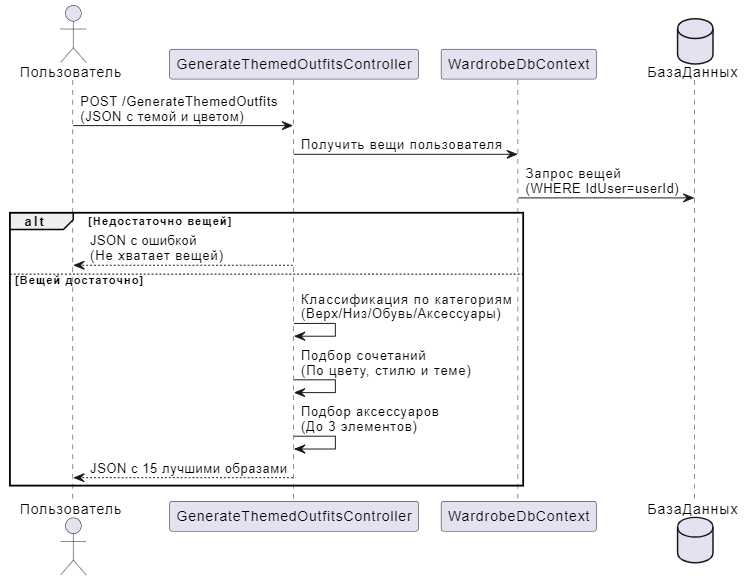


Рисунок 3. – Диаграмма последовательности для генерации тематических образов

Эти диаграммы позволяют наглядно представить последовательность действий и взаимодействий между различными компонентами системы, включая контроллеры, сервисы и базу данных.

Диаграмма прецедентов – позволяет визуализировать различные типы ролей в системе и то, как эти роли взаимодействуют с системой.

Диаграмма прецедентов для приложения представлен на рисунке 3.17.

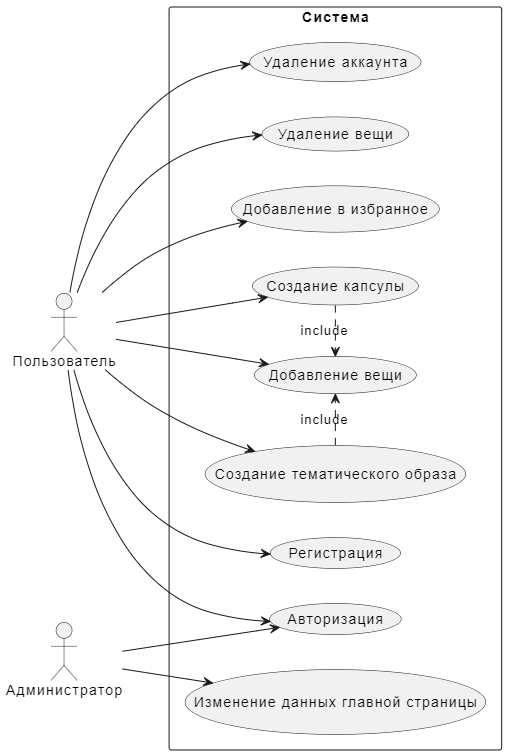


Рисунок 3. – Диаграмма прецедентов

Таким образом, разработанная диаграмма прецедентов отображает взаимодействия между различными ролями пользователей и системой, позволяя четко выделить основные функции и сценарии использования.

Диаграмма деятельности – это графическое представление процессов и задач, выполняемых в рамках определенного проекта или деятельности.

Диаграмма деятельности для регистрации представлена на рисунке 3.18.

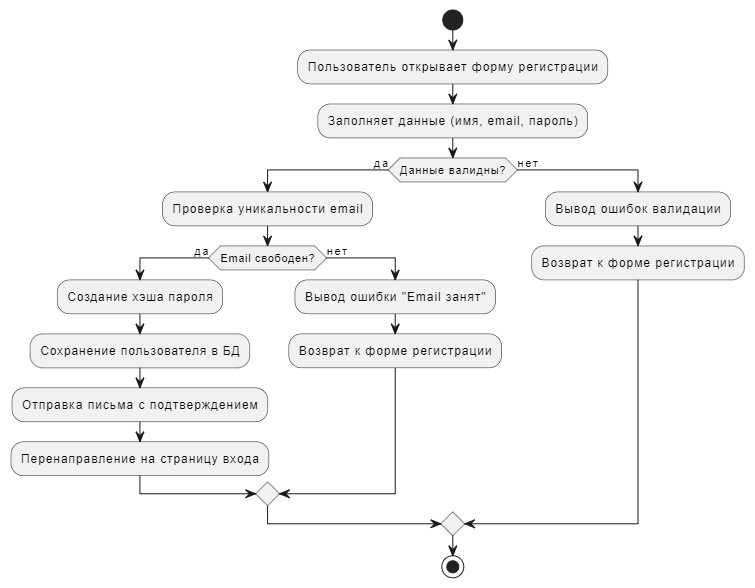


Рисунок 3. – Диаграмма деятельности для регистрации

Диаграмма деятельности для авторизации представлена на рисунке 3.19.

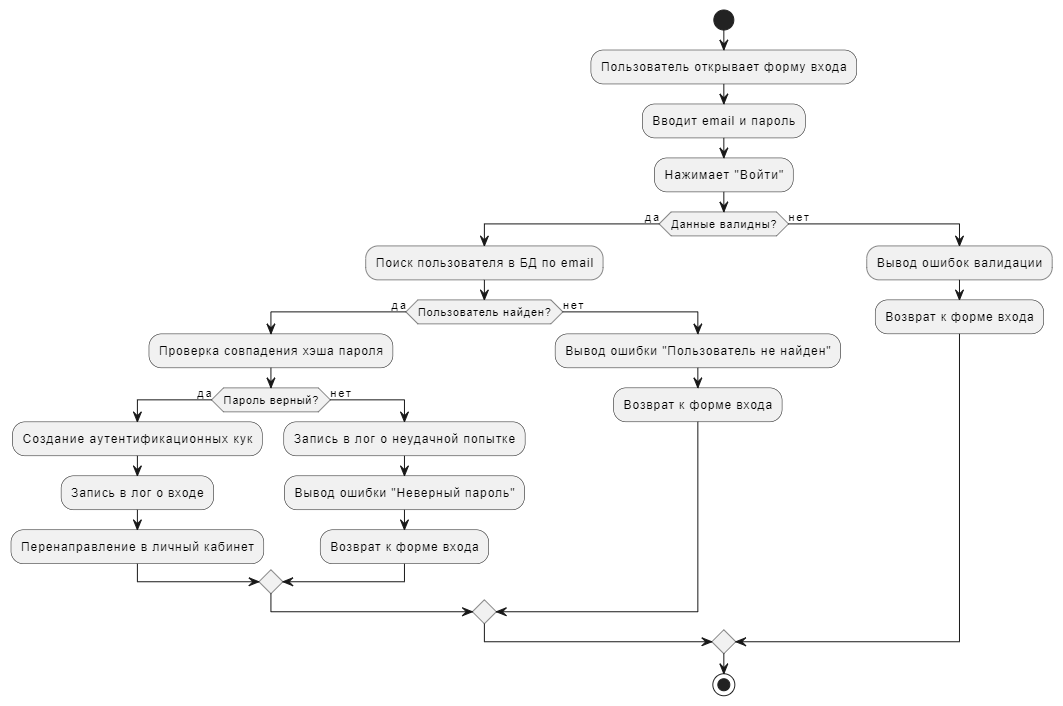


Рисунок 3. – Диаграмма деятельности для авторизации

Диаграмма деятельности для загрузки изображений одежды представлена на рисунке 3.20.

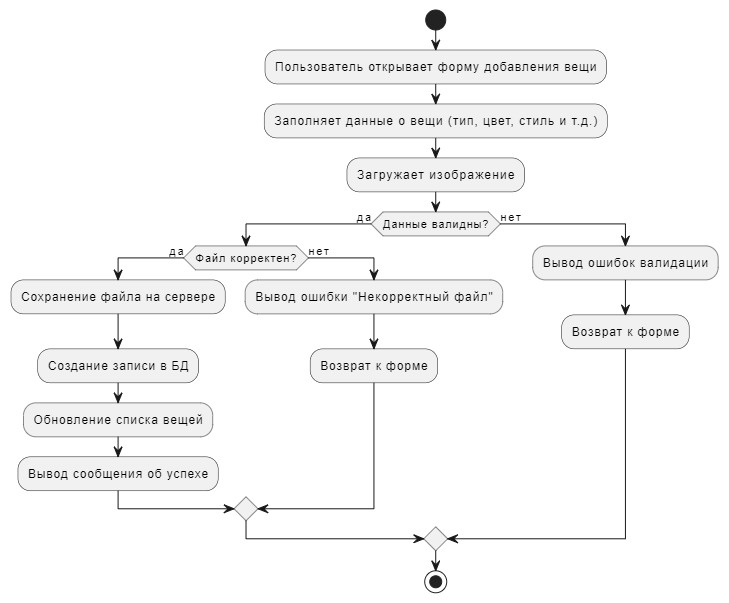


Рисунок 3. – Диаграмма деятельности для загрузки изображений

Диаграмма деятельности для генерации капсульного гардероба представлена на рисунке 3.21.

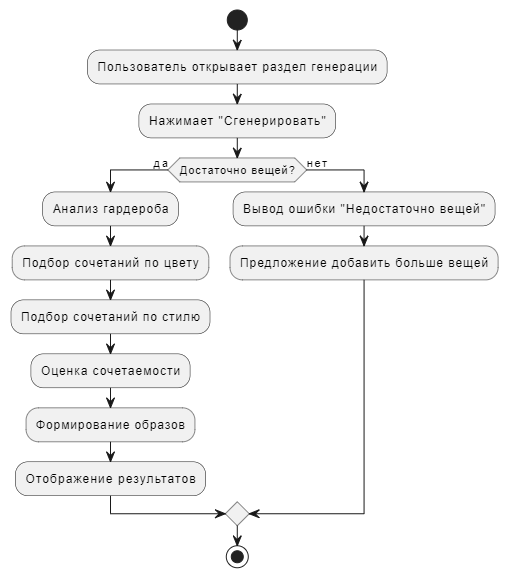


Рисунок 3. – Диаграмма деятельности для капсульного гардероба

Диаграмма деятельности для генерации тематических капсул представлена на рисунке 3.22.

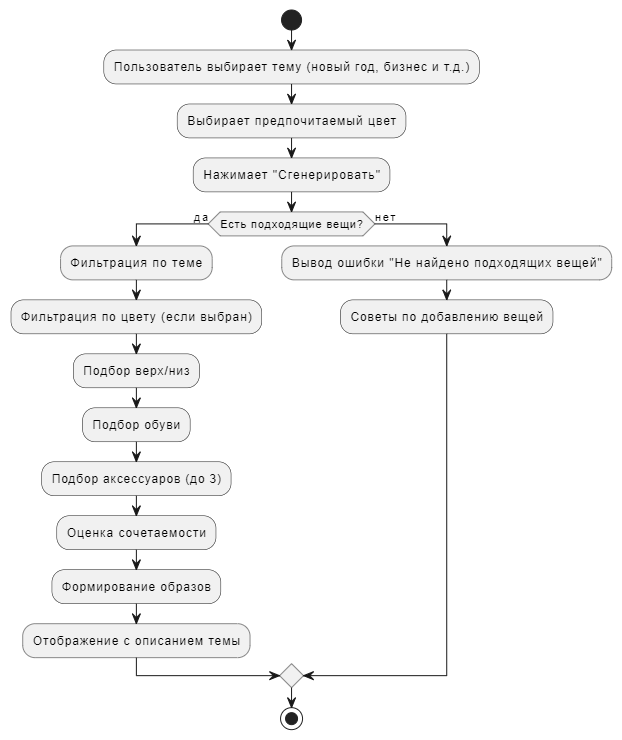


Рисунок 3. – Диаграмма деятельности для генерации тематического образа

Диаграммы деятельности наглядно отображают процессы и последовательность шагов в системе, помогая анализировать логику выполнения задач, условия переходов и взаимодействие компонентов. Это способствует лучшему пониманию и оптимизации рабочих процессов.

4 Разработка приложения

4.1 Реализация архитектуры

Было разработано 8 моделей, описывающих структуру данных и процессов:

* CapsuleWardrobe.
* Clothing.
* ClothingCapsule.
* ClothingLook.
* ThematicLook.
* User.

Листинг кода моделей представлен в приложении А.

Архитектура системы включает следующие контроллеры:

* AccountController – отвечает за управление учетными записями пользователей: регистрацию новых аккаунтов с валидацией данных и хешированием паролей, аутентификацию при входе в систему, а также выход из учетной записи.
* CapsuleWardrobeController – генерирует капсульные гардеробы, анализируя одежду пользователя по категориям (верх, низ, аксессуары) и создавая стильные комплекты на основе сочетаемости цветов, стилей и сезонности.
* ClothingController – отображает и систематизирует элементы гардероба пользователя, предоставляя удобный просмотр добавленных вещей с сортировкой по дате.
* GenerateThemedOutfitsController – специализируется на создании тематических образов для различных событий (деловых встреч, праздников и т. д.), подбирая одежду и аксессуары по заданным критериям (цвет, стиль, сезон).
* ProfileWardrobeController – управляет персональными данными пользователя: отображает профиль, позволяет удалить аккаунт и просматривать гардероб.
* WardrobeController – реализует основные операции с гардеробом: добавление новых вещей с загрузкой изображений и валидацией, а также фильтрацию одежды по категориям для удобного просмотра.

4.2 Реализация базы данных

Структура разработанной базы данных представлена на рисунке 4.1 и 4.2.

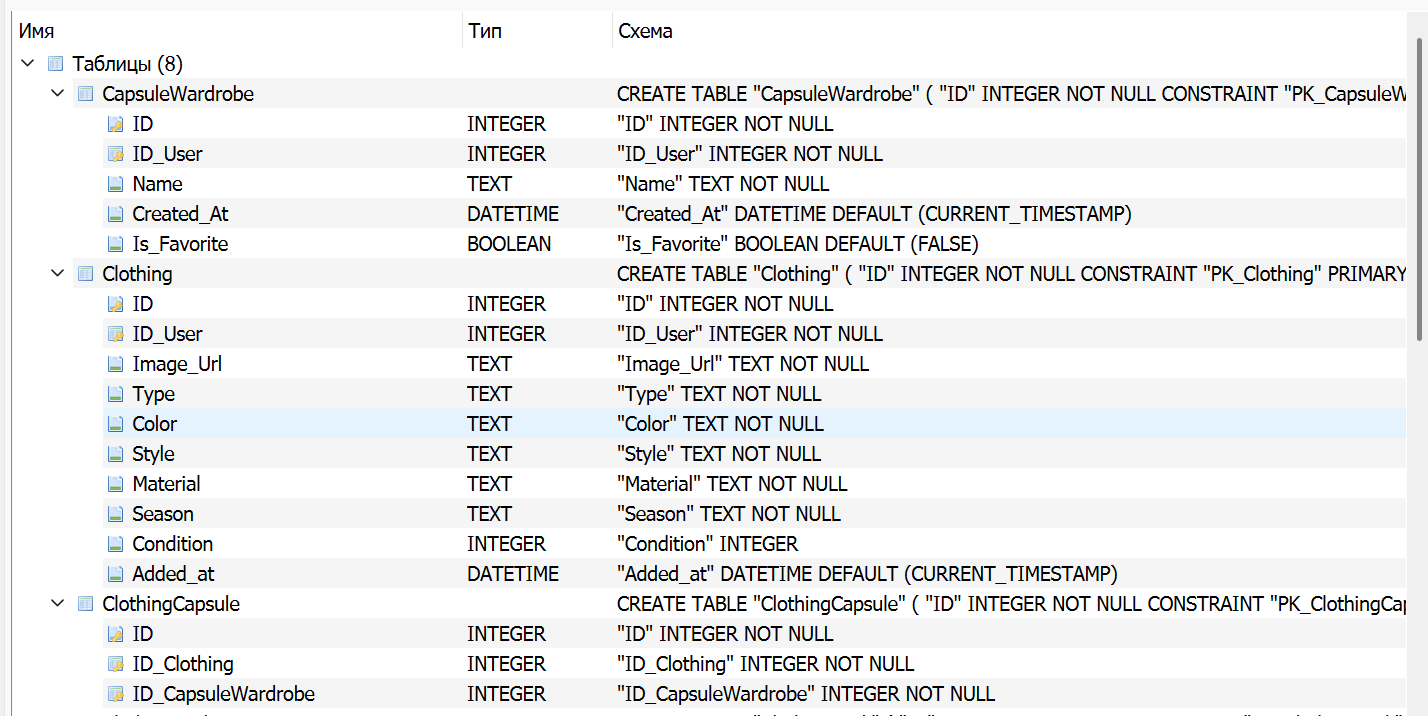


Рисунок 4. 1 – Структура базы данных

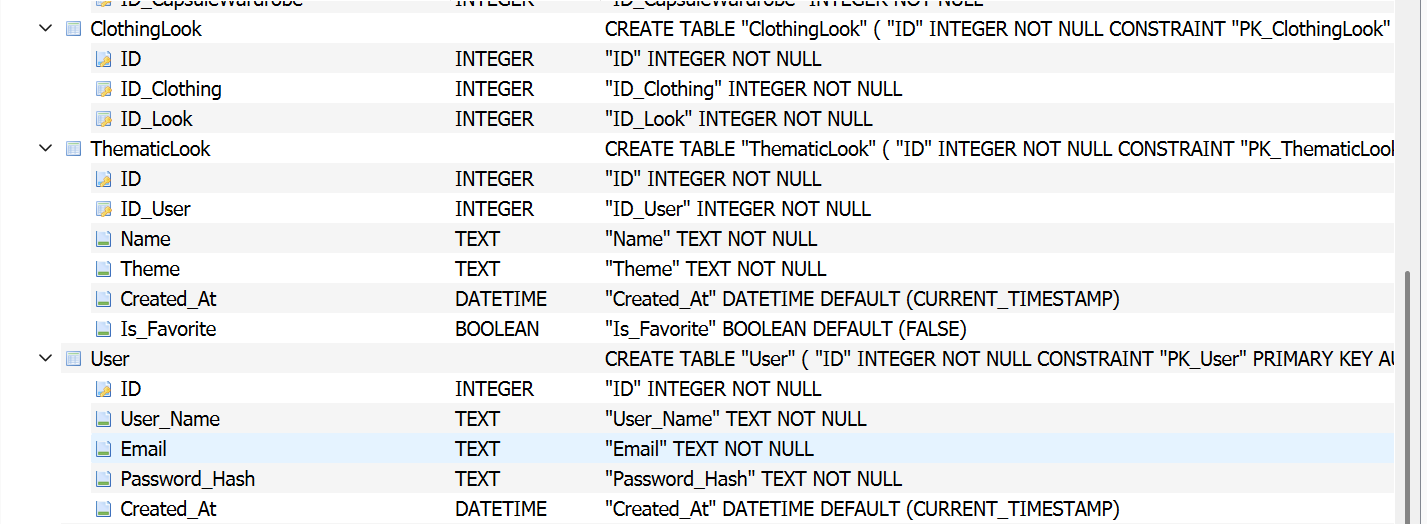


Рисунок 4. – Структура базы данных

Разработанная база данных полностью соответствует ER диаграмме, представленной во втором разделе.

4.2 Реализация модулей

Модуль аутентификации (AccountController):

* Регистрация новых пользователей с валидацией данных.
* Вход в систему с проверкой учетных данных.
* Выход из системы.
* Хеширование паролей (BCrypt).

Модуль гардероба (WardrobeController, ClothingController):

* Добавление новых вещей с загрузкой изображений.
* Валидация данных одежды (тип, цвет, стиль и др.).
* Сохранение файлов на сервере с проверкой форматов.
* Просмотр гардероба по категориям.
* Сортировка вещей по дате добавления.

Модуль генерации образов (CapsuleWardrobeController, GenerateThemedOutfitsController):

* Генерация базовых капсул одежды.
* Создание тематических образов (новый год, бизнес и др.).
* Алгоритмы подбора сочетаний по: цвету, стилю, сезонности.
* Подбор аксессуаров и обуви.
* Оценка сочетаемости элементов.

Модуль профиля (ProfileWardrobeController):

* Просмотр профиля пользователя.
* Удаление аккаунта.

4.2 Реализация пользовательского интерфейса

На рисунке 4.3 изображена страница авторизации, на которой находится форма с двумя полями:

* Электронная почта
* Пароль

Под формой расположена кнопка «Войти», а ниже — ссылка для перехода на страницу регистрации («Зарегистрироваться»).

Входные данные:

* Электронная почта
* Пароль.

Выходные данные: перенаправление на главную страницу после успешной авторизации.

Справа отображается изображение, а в верхнем левом углу указано название приложения — «Планировщик гардероба».

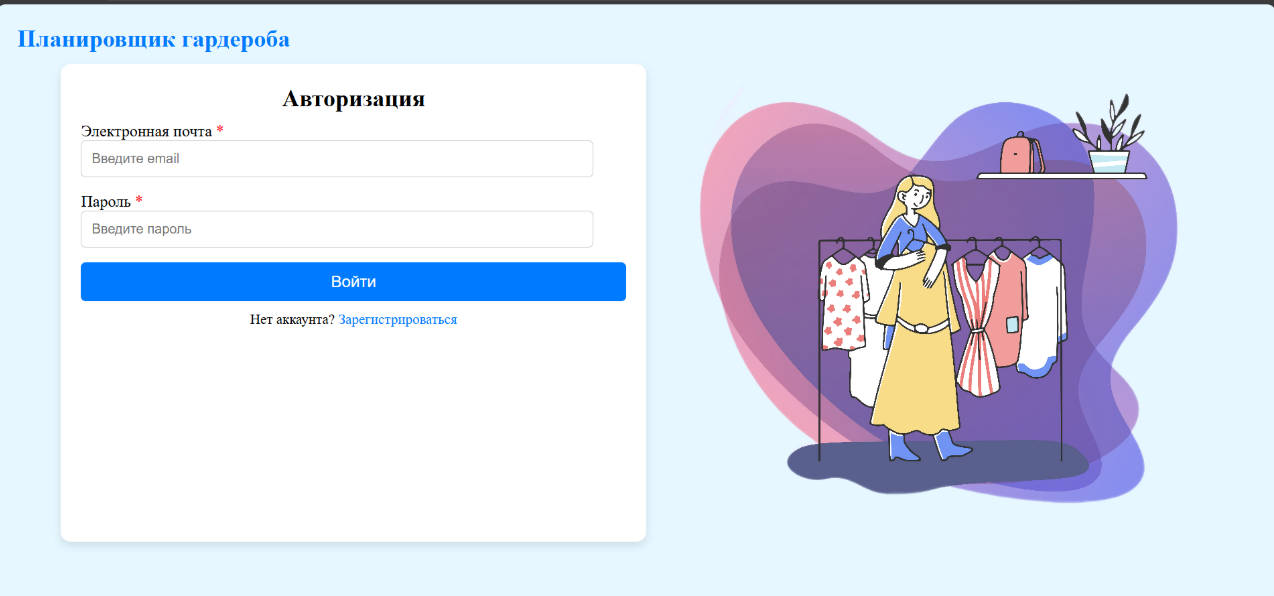


Рисунок 4. – Страница авторизации

На рисунке 4.4 изображена страница регистрации, на которой расположена форма, которая включает четыре поля:

* Имя пользователя.
* Электронная почта.
* Пароль.
* Подтверждение пароля.

Под формой находится кнопка «Зарегистрироваться», а ниже — кнопка для перехода на страницу авторизации («Войти»).

Входные данные:

* Имя пользователя.
* Электронная почта
* Пароль.

Выходные данные: перенаправление на страницу авторизации после успешной регистрации.

В правой части страницы размещено изображение, а в верхнем левом углу — название приложения — «Планировщик гардероба».

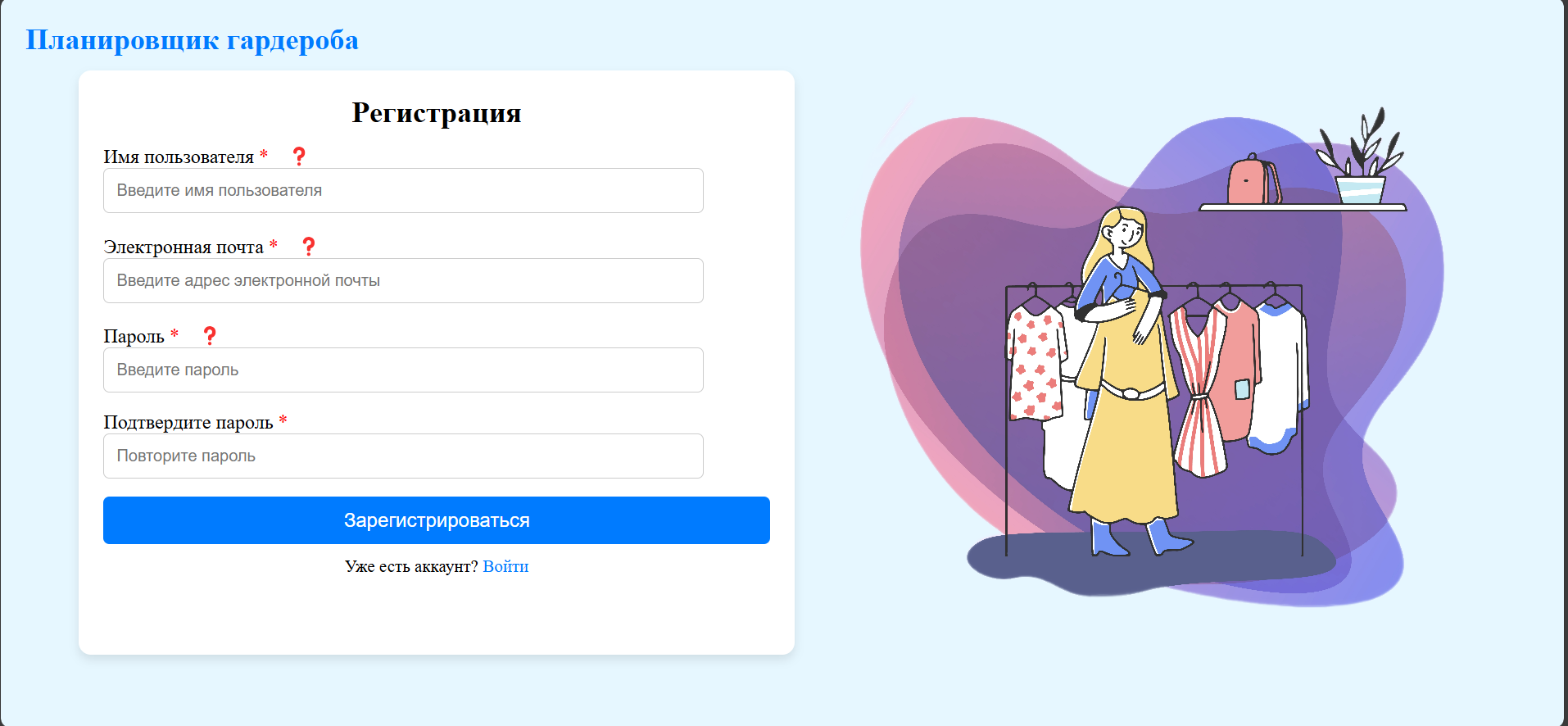


Рисунок 4. – Страница регистрации

На рисунке 4.5 изображена главная страница, в верхнем правом углу расположено выпадающее меню, позволяющее переходить между разделами приложения. В центральной части страницы отображается приветственное сообщение: «Добро пожаловать на главную страницу! Вы успешно авторизовались.»

Входные параметры:

* Данные пользователя (полученные после авторизации).

Выходные параметры: доступ к функционалу приложения через меню.

На данном этапе разработки, функционал страницы разработан не до конца.

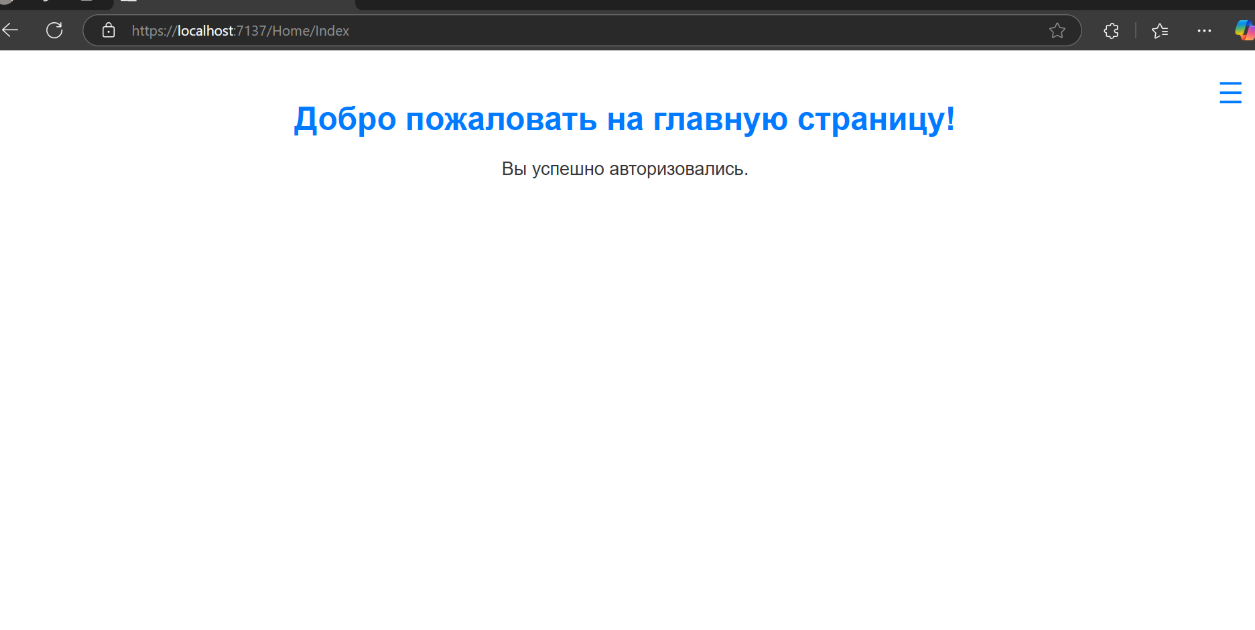


Рисунок 4. – Главная страница

На рисунке 4.6 показана страница профиля пользователя. Слева отображаются личные данные: имя, электронная почта и дата регистрации. Там же расположена кнопка «Удалить аккаунт». В верхнем правом углу находится кнопка возврата на главную страницу и дополнительное меню для навигации. В центре страницы размещён заголовок «Мой профиль».

Система получает следующие входные данные: имя пользователя, электронную почту и дату регистрации из базы данных. Выходные параметры включают: возможность удаления учётной записи при нажатии соответствующей кнопки, а также переход между разделами приложения через меню.

На данном этапе разработки, функционал страницы разработан не до конца.

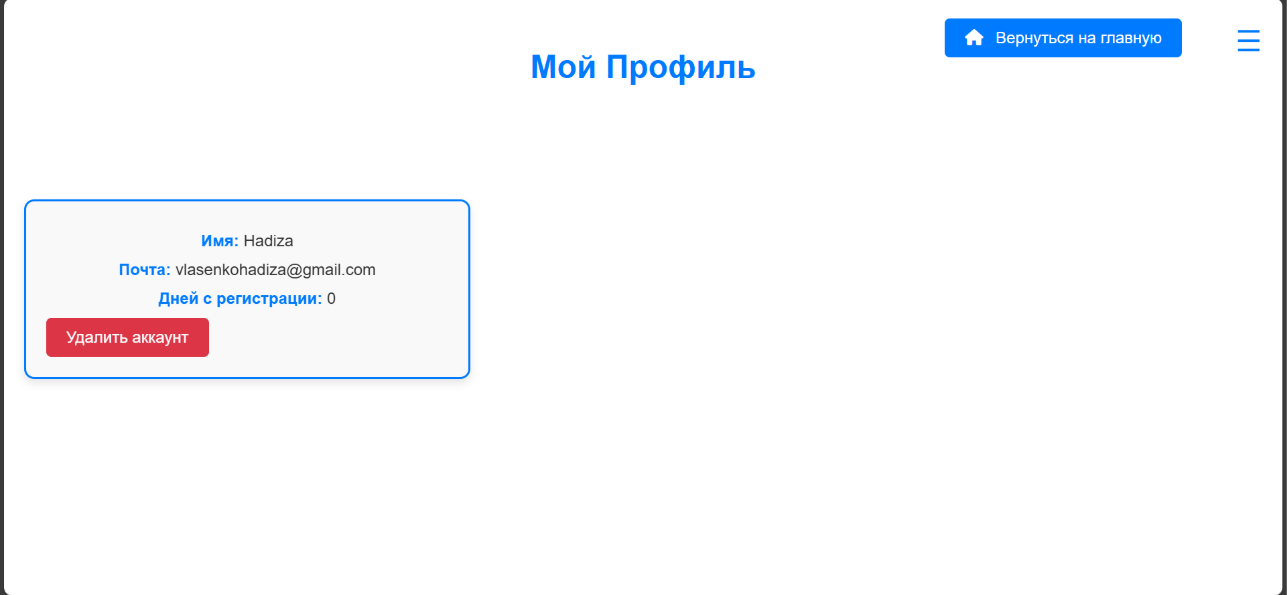


Рисунок 4. – Страница профиля пользователя

На рисунке 4.7 изображена страница добавления вещей в гардероб, представлена форма, которая позволяет загружать изображения одежды и указывать её характеристики. В форме есть поле для загрузки фотографии вещи, а также поля для указания типа одежды (например, пальто, футболка, джинсы), сезона (лето, зима, демисезон), цвета, стиля (повседневный, офисный, спортивный), материала и состояния (новое, хорошее, поношенное). Внизу формы расположена кнопка «Добавить в гардероб» для сохранения данных. Слева на странице находится меню для перехода между разделами приложения.

Входными параметрами для этой страницы являются: загружаемое пользователем изображение вещи и вводимые текстовые характеристики (тип, сезон, цвет, стиль, материал, состояние). Выходные параметры включают: сохранённую в системе вещь с прикреплённым изображением и метаданными, обновлённый список вещей в гардеробе, а также возможность навигации по другим разделам приложения через боковое меню. На данном этапе разработки, функционал страницы разработан не до конца.

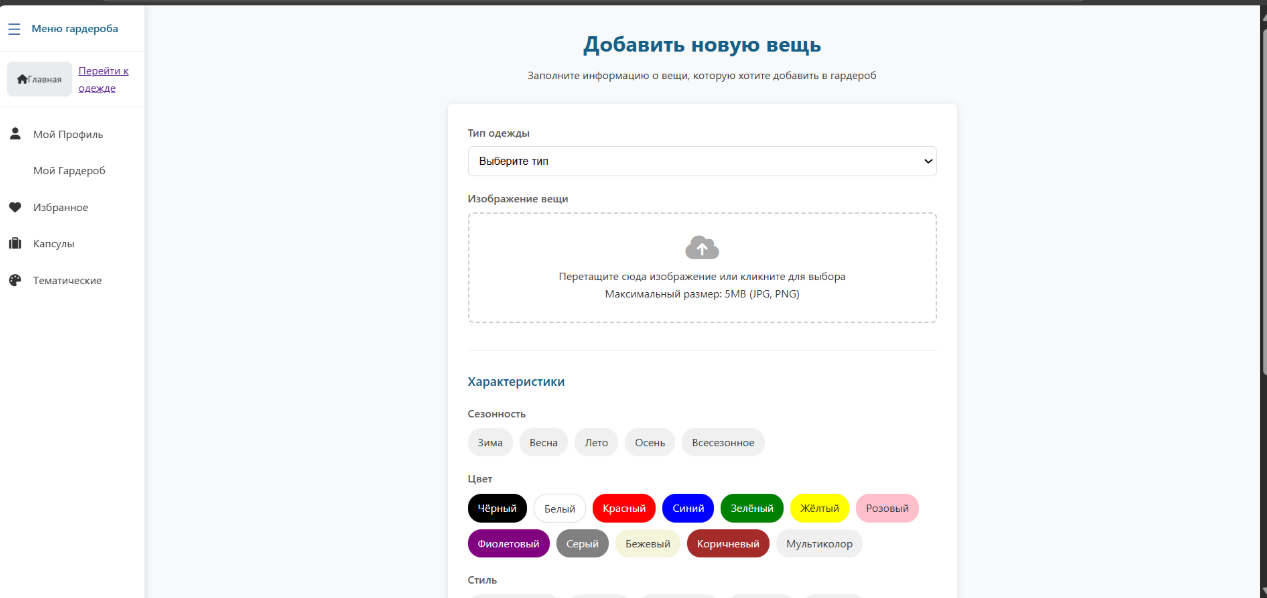


Рисунок 4. – Страница добавления одежды

На рисунке 4.8 изображена страница, где отображается добавленная одежда, представлен список вещей, сгруппированных по типам (юбки, платья, брюки, футболки и другие категории). В верхней части экрана находится информационная надпись «СКОРО ЗДЕСЬ ПОЯВИТСЯ ВАША ОДЕЖДА», которая указывает на то, что данный функционал находится в стадии разработки и будет дорабатываться. Также на странице присутствует кнопка «Вернуться назад», позволяющая пользователю перейти обратно к форме добавления новой вещи. Входными данными для этой страницы служат: загруженные в систему элементы гардероба с их характеристиками (тип одежды, цвет, сезонность, стиль и другие параметры), которые извлекаются из базы данных и автоматически сортируются по категориям. Выходными параметрами являются: возможность возврата к форме добавления одежды через соответствующую кнопку и отображение структурированного списка вещей. В будущем планируется расширить функционал этой страницы, добавив возможности фильтрации, редактирования и удаления элементов гардероба. В текущей версии интерфейс выполняет только базовые функции просмотра и навигации.



Рисунок 4. – Страница отображения одежды

На рисунке 4.9 изображена страница генерации капсульного гардероба, где отображаются сгенерированные капсулы из всей одежды пользователя. На странице есть две кнопки: «Подобрать комплекты» и «Показать мой гардероб». Сверху есть кнопка «Вернуться на главную», которая перенаправляет на главную станицу.

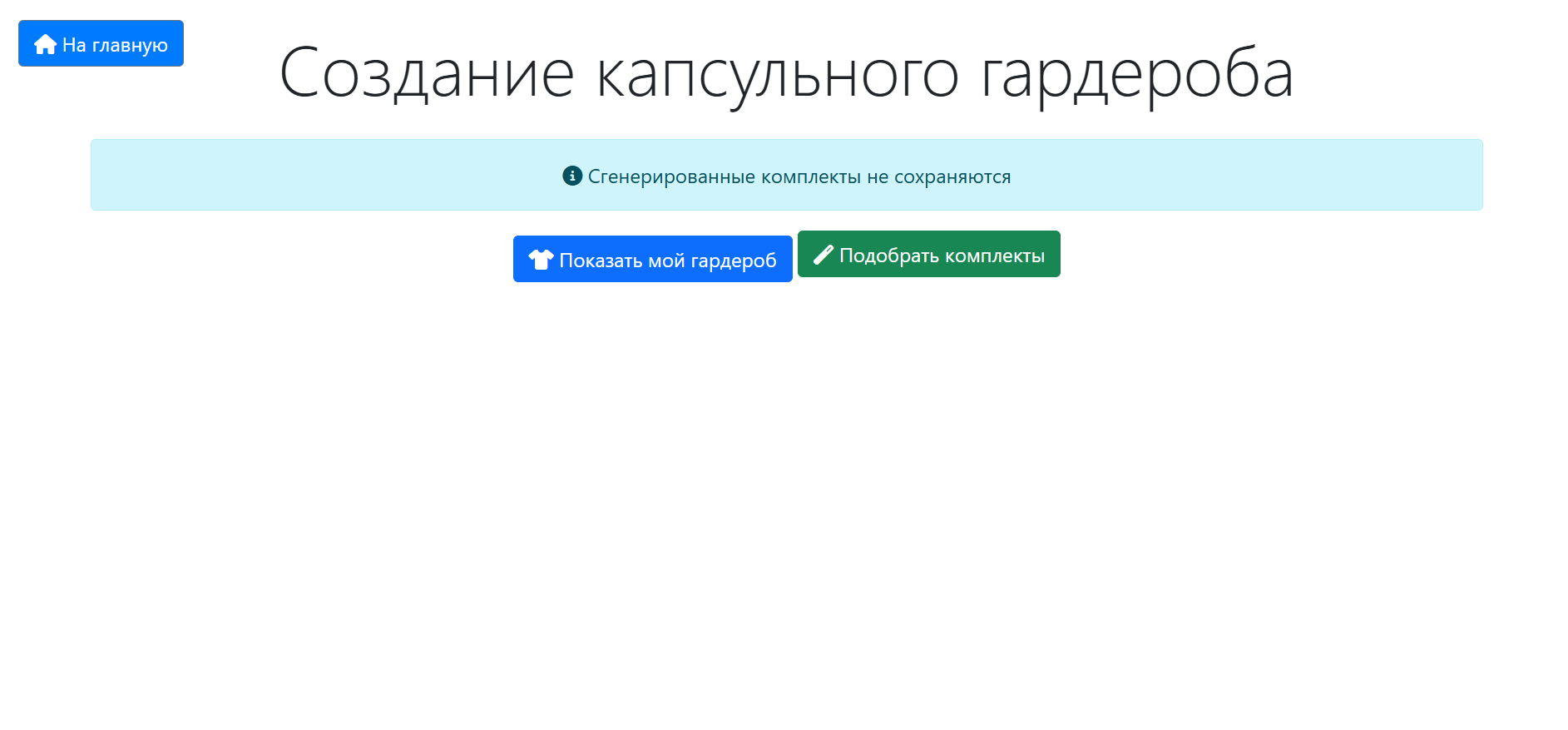


Рисунок 4. – Страница генерации капсульного гардероба

На рисунке 4.10 изображена страница генерации тематического гардероба, где отображаются сгенерированные тематические образы из всей одежды пользователя. На странице есть поля для выбора события и желаемого цвета в одежде. Сверху есть кнопка «Вернуться на главную», которая перенаправляет на главную станицу.

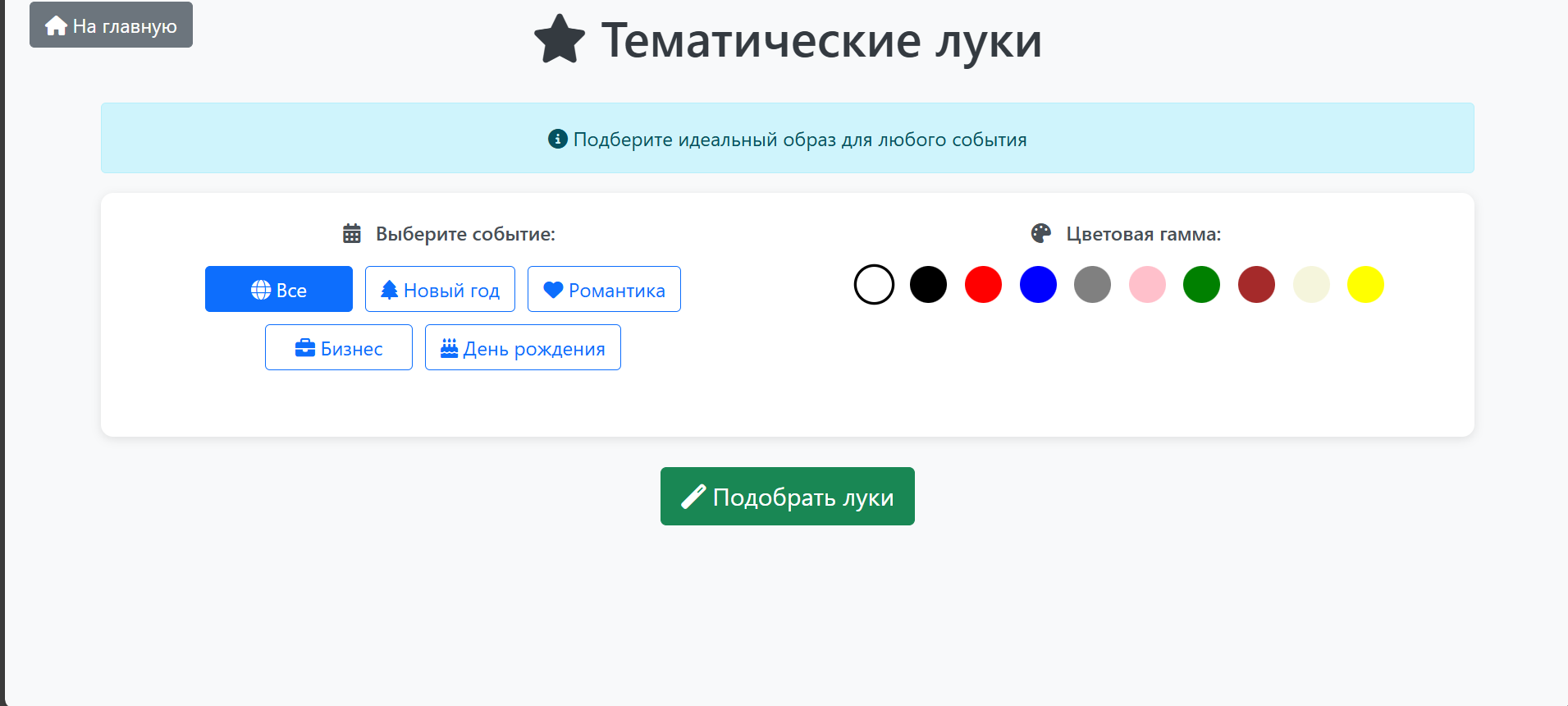


Рисунок 4. – Страница генерации капсульного гардероба

Приложение получилось удобным и простым в использовании. Оно полностью соответствует разработанным ранее схемам и макетам интерфейса. Все основные функции: авторизация, регистрация, работа с гардеробом и создание капсул реализованы. Пользователи смогут легко освоить приложение и комфортно им пользоваться благодаря интуитивно понятному интерфейсу и продуманной логике взаимодействия.

5. ТЕСТИРОВАНИЕ

5.1 Тест-требования

Авторизация пользователя.

* + Проверить, что при успешной авторизации существующего пользователя происходит перенаправление на главную страницу приложения.
  + Проверить, что при попытке авторизации с неверными учетными данными пользователь получает соответствующее сообщение об ошибке.

Регистрация нового пользователя.

* Проверить, что при корректном заполнении всех полей формы регистрации пользователь получает сообщение об успешной регистрации и перенаправляется на страницу авторизации.
* Проверить, что при вводе пароля менее 8 символов система выводит сообщение об ошибке.
* Проверить, что при вводе некорректного email-адреса (без символа @) система выводит сообщение об ошибке.
* Проверить, что при попытке регистрации с уже существующим email система выводит соответствующее сообщение.

5.2 Тест-план

Тест 1. Успешная авторизация

Описание: Проверка входа в систему с валидными учетными данными

Входные данные: Зарегистрированный email ([vlasenko2@gmail.com](mailto:vlasenko2@gmail.com)), корректный пароль (12345qQ!)

Ожидаемый результат: Перенаправление на главную страницу

Сценарий:

* + Открыть страницу авторизации
  + Ввести тестовые email и пароль
  + Нажать кнопку «Войти»

Тест 2. Неудачная авторизация (неверный пароль)

Описание: Проверка обработки неверного пароля

Входные данные: Зарегистрированный email ([vlasenko2@gmail.com](mailto:vlasenko2@gmail.com)), неверный пароль (12345678)

Ожидаемый результат: Сообщение «Неверный пароль»

Сценарий:

* + Открыть страницу авторизации
  + Ввести тестовый email и неверный пароль
  + Нажать кнопку «Войти»
  + Проверить сообщение об ошибке

Тест 3. Регистрация с коротким паролем

Описание: Проверка валидации длины пароля

Входные данные: Пароль из 6 символов (qwerty)

Ожидаемый результат: Сообщение «Пароль должен содержать минимум 8 символов»

Сценарий:

* + Открыть страницу регистрации
  + Заполнить все обязательные поля
  + Ввести короткий пароль
  + Нажать кнопку «Зарегистрироваться»
  + Проверить сообщение об ошибке

5.3 Результаты тестирования

Тест 1. Успешная авторизация

Входные данные: [vlasenko2@gmail.com](mailto:vlasenko2@gmail.com), (12345qQ!)

Фактический результат: произошло перенаправление на главную страницу, отображается приветственное сообщение

На рисунке 4.1. представлен результат тестирования. Тестирование прошло успешно.

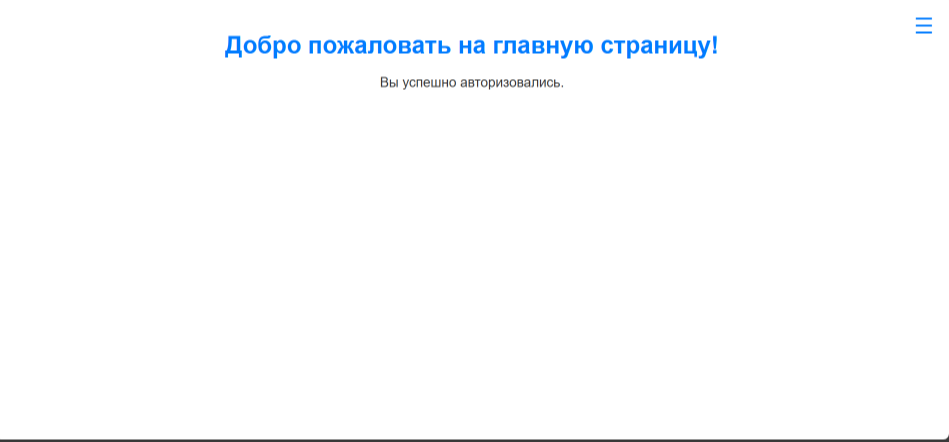


Рисунок 5. – Тестирование авторизации

Тест 2. Неудачная авторизация (пароль не ввели)

Входные данные: [vlasenko2@gmail.com](mailto:vlasenko2@gmail.com),

Фактический результат: отобразилось сообщение «Заполните поле».

На рисунке 4.2. представлен результат тестирования. Тестирование прошло успешно.

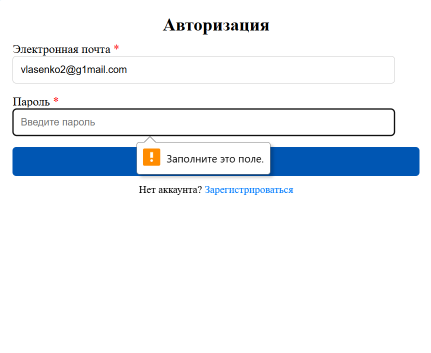


Рисунок 5. – Тестирование авторизации

Тест 3. Регистрация с коротким паролем

Входные данные: Пароль - qwerty

Фактический результат: отобразилось сообщение о необходимости пароля длиной от 6 символов.

На рисунке 4.3. представлен результат тестирования. Тестирование прошло успешно.

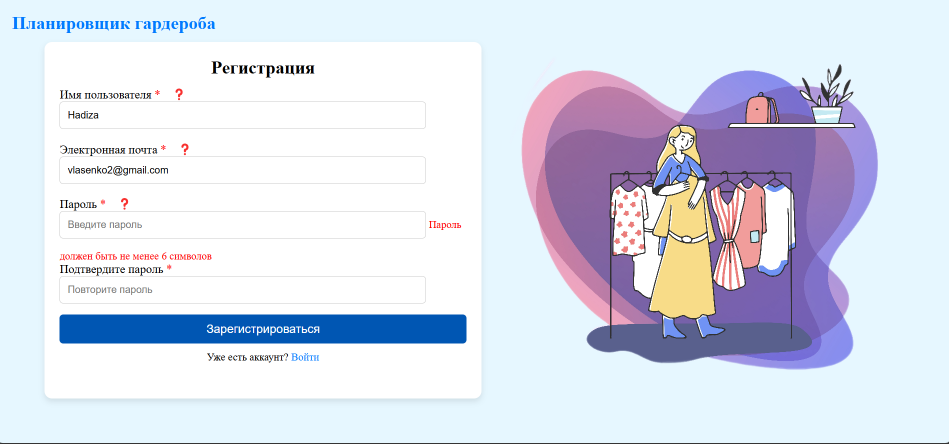


Рисунок 5. – Тестирование регистрации

Все тесты пройдены успешно.

заключение

В ходе преддипломной практики было разработано веб-приложение для подбора одежды и формирования капсульных гардеробов. В рамках работы:

* Проведен анализ существующих решений, что позволило определить ключевые функции и преимущества разрабатываемого приложения.
* Сформулированы требования на основе потребностей целевой аудитории, включая загрузку фотографий, подбор сочетаний и тематических образов.
* Реализована архитектура приложения с использованием ASP.NET Core и Entity Framework Core, обеспечивающая стабильную работу с базой данных (SQLite).
* Разработан интуитивно понятный интерфейс для управления гардеробом, включая добавление вещей, просмотр по категориям и генерацию образов.
* Проведено тестирование функционала.
* Организован контроль версий через GitHub для отслеживания изменений и совместной работы.

В результате практики удалось создать работоспособное веб-приложение с базовым функционалом для управления гардеробом. Полученные знания и навыки в области веб-разработки, работы с базами данных и тестирования программного обеспечения будут полезны в дальнейшей разработке.

список использованных источников

* 1. «Using SQLite: Small. Fast. Reliable. Choose Any Three». (2010)  
     Автор: Jay A. Kreibich
  2. «The Little ASP.NET Core Book». Автор: Nat Barbettini.
  3. [Сочетание цветов в одежде с примерами: палитры, образы, схемы](https://star-tex.ru/article/notes/kak-sochetat-cveta-v-odezhde-prostaya-teoriya-i-prakticheskie-sovet/?ysclid=m9biyzavmk168888794). URL: <https://star-tex.ru/article/notes/kak-sochetat-cveta-v-odezhde-prostaya-teoriya-i-prakticheskie-sovet/?ysclid=m9biyzavmk168888794>
  4. Учебное пособие по диаграмма прецедентов (Руководство с примерами). URL: <https://creately.com/blog/ru>
  5. [Диаграмма деятельности. Руководство для начинающих.](https://itonboard.ru/analysis/664-diagramma_dejatelnocti_rukovodstvo_dlja_nachinajushhih/) URL: https://itonboard.ru/analysis/664diagramma\_dejatelnocti\_rukovodstvo\_dlja\_nachinajushhih/

приложение А

Листинг кода модели CapsuleWardrobe

using System;

using System.Collections.Generic;

namespace OutfitPlaner\_Applcation.Models;

public partial class CapsuleWardrobe

{

public int Id { get; set; }

public int IdUser { get; set; }

public string Name { get; set; } = null!;

public DateTime? CreatedAt { get; set; }

public bool? IsFavorite { get; set; }

public virtual ICollection<ClothingCapsule> ClothingCapsules { get; set; } = new List<ClothingCapsule>();

public virtual User IdUserNavigation { get; set; } = null!;

}

Листинг кода модели Clothing

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace OutfitPlaner\_Applcation.Models;

public class Clothing

{

[Key]

[Column("ID")]

public int Id { get; set; }

[Required]

[Column("ID\_User")]

public int IdUser { get; set; }

[Required]

[Column("Image\_Url")]

public string ImageUrl { get; set; } = null!;

[Required]

[Column("Type")]

public string Type { get; set; } = null!;

[Required]

[Column("Color")]

public string Color { get; set; } = null!;

[Required]

[Column("Style")]

public string Style { get; set; } = null!;

[Required]

[Column("Material")]

public string Material { get; set; } = null!;

[Required]

[Column("Season")]

public string Season { get; set; } = null!;

[Column("Condition")]

public int? Condition { get; set; }

[Column("Added\_at")]

public DateTime? AddedAt { get; set; }

// Навигационное свойство

public virtual User User { get; set; } = null!;

public virtual ICollection<ClothingCapsule> ClothingCapsules { get; set; } = new List<ClothingCapsule>();

public virtual ICollection<ClothingLook> ClothingLooks { get; set; } = new List<ClothingLook>();

}

Листинг кода модели ClothingCapsule

using System;

using System.Collections.Generic;

namespace OutfitPlaner\_Applcation.Models;

public partial class ClothingCapsule

{

public int Id { get; set; }

public int IdClothing { get; set; }

public int IdCapsuleWardrobe { get; set; }

public virtual CapsuleWardrobe IdCapsuleWardrobeNavigation { get; set; } = null!;

public virtual Clothing IdClothingNavigation { get; set; } = null!;

}

Листинг кода модели ClothingLook

using System;

using System.Collections.Generic;

namespace OutfitPlaner\_Applcation.Models;

public partial class ClothingLook

{

public int Id { get; set; }

public int IdClothing { get; set; }

public int IdLook { get; set; }

public virtual Clothing IdClothingNavigation { get; set; } = null!;

public virtual ThematicLook IdLookNavigation { get; set; } = null!;

}

Листинг кода модели ThematicLook

using System;

using System.Collections.Generic;

namespace OutfitPlaner\_Applcation.Models;

public partial class ThematicLook

{

public int Id { get; set; }

public int IdUser { get; set; }

public string Name { get; set; } = null!;

public string Theme { get; set; } = null!;

public DateTime? CreatedAt { get; set; }

public bool? IsFavorite { get; set; }

public virtual ICollection<ClothingLook> ClothingLooks { get; set; } = new List<ClothingLook>();

public virtual User IdUserNavigation { get; set; } = null!;

}

Листинг кода модели User

using System;

using System.Collections.Generic;

namespace OutfitPlaner\_Applcation.Models;

public class User

{

public int Id { get; set; }

public string UserName { get; set; }

public string Email { get; set; }

public string PasswordHash { get; set; }

public DateTime CreatedAt { get; set; }

public virtual ICollection<Clothing> Clothing { get; set; } = new List<Clothing>();

}

приложение B

Код главной формы:

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.EntityFrameworkCore;

using OutfitPlaner\_Applcation.Data;

var builder = WebApplication.CreateBuilder(args);

var dbPath = Path.Combine(builder.Environment.ContentRootPath, "Data", "OutfitPlanner.db");

builder.Services.AddDbContext<WardrobeDbContext>(options =>

options.UseSqlite($"Data Source={dbPath}"));

// Настройка аутентификации через Cookie

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

.AddCookie(options =>

{

options.LoginPath = "/Account/Login";

options.AccessDeniedPath = "/Account/AccessDenied";

options.ExpireTimeSpan = TimeSpan.FromDays(30);

options.SlidingExpiration = true;

options.Cookie.HttpOnly = true;

options.Cookie.SecurePolicy = builder.Environment.IsDevelopment()

? CookieSecurePolicy.None

: CookieSecurePolicy.Always;

options.Cookie.SameSite = SameSiteMode.Lax;

});

builder.Services.AddControllersWithViews();

builder.Services.AddSession();

builder.Services.Configure<Microsoft.AspNetCore.Http.Features.FormOptions>(options =>

{

options.MultipartBodyLengthLimit = long.MaxValue;

});

var app = builder.Build();

using (var scope = app.Services.CreateScope())

{

var db = scope.ServiceProvider.GetRequiredService<WardrobeDbContext>();

db.Database.Migrate();

}

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseSession();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();