**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ПП.09.02.07-1.25.221.18

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПМ.11 Разработка, администрирование и защита баз данных

|  |  |  |
| --- | --- | --- |
| Руководитель от предприятия:  М.П. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.Н. Касьяненко) |
| Руководитель от техникума: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Е.С. Фролова) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (В.М. Седых) |

Иркутск 2025

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc201144937)

[1. Предпроектное исследование 4](#_Toc201144938)

[1.1. Описание предметной области 4](#_Toc201144939)

[1.2. Анализ инструментов 4](#_Toc201144940)

[2. Проектирование веб-приложения 7](#_Toc201144941)

[2.1. Структурная схема веб-приложения 7](#_Toc201144942)

[2.2. Проектирование базы данных 10](#_Toc201144943)

[2.3. Проектирование пользовательского интерфейса 12](#_Toc201144944)

[3. Разработка программного модуля 15](#_Toc201144945)

[3.1. Разработка интерфейса программного модуля 15](#_Toc201144946)

[3.2. Создание базы данных 18](#_Toc201144947)

[3.3. Разработка программного модуля 20](#_Toc201144948)

[4. Тестирование программного модуля 24](#_Toc201144949)

[5. Документирование программного модуля 27](#_Toc201144950)

[5.1. Руководство пользователя 27](#_Toc201144951)

[6. Оценка возможности модернизации программного продукта 32](#_Toc201144952)

[7. Производственные задачи 33](#_Toc201144953)

[ЗАКЛЮЧЕНИЕ 34](#_Toc201144954)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 35](#_Toc201144955)

# **ВВЕДЕНИЕ**

В условиях цифровой трансформации сферы услуг автоматизация процессов бронирования стала критически важной для индустрии развлечений. Разработанное веб-приложение для кинотеатра решает ключевую проблему: устранение очередей и оптимизацию работы за счёт внедрения системы онлайн-бронирования билетов.

Актуальность проекта обусловлена:

* Ростом спроса на бесконтактные сервисы после пандемии
* Необходимостью снижения операционных затрат кинотеатров
* Трендом на персонализацию пользовательского опыта

Цель работы: создание полнофункциональной системы бронирования билетов с интеграцией в бизнес-процессы кинотеатра.

Задачи:

1. Реализация модуля управления сеансами и залами
2. Разработка интерактивной системы выбора мест
3. Создание механизма генерации электронных билетов
4. Обеспечение безопасности персональных данных

Технологический стек:

* Backend: Django (Python 3.12)
* Database: PostgreSQL 15
* Frontend: HTML

# **1. Предпроектное исследование**

## **1.1. Описание предметной области**

Основные компоненты системы:

* Расписание сеансов:
  + Отображение списка фильмов с датами, временем сеансов и доступными залами.
  + Учет разных кинозалов (размеры, цены).
* Выбор мест:
  + Визуализация схемы зала в реальном времени.
  + Бронирование мест с защитой от коллизий.
* Управление пользователями:
  + Регистрация, авторизация.
  + История покупок.
* Административная панель:
  + Добавление/редактирование сеансов, управление залами, настройка цен.

Риски и требования:

* Безопасность: Защита персональных данных и платежной информации.
* Производительность: Обработка одновременных бронирований без задержек.
* Удобство интерфейса: Интуитивная навигация для пользователей.

## **1.2. Анализ инструментов**

**Критерии выбора:**

* Совместимость с задачей (поддержка веб-разработки, работа с БД, генерация PDF).
* Простота использования и документация (для быстрого старта).
* Поддержка сообщества (наличие библиотек, готовых решений).

**Язык программирования: Python**

**Альтернативы:** PHP, JavaScript (Node.js), Ruby.

Таблица 1 – Сравнение языков программирования

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий\Язык | Python | Node.js | PHP |
| Совместимость | Идеален для веб-фреймворков, много библиотек для работы с PDF. | Быстрая обработка запросов, но сложнее с синхронными задачами. | Устаревшие фреймворки, меньше удобных инструментов для PDF. |
| Простота | Чистый синтаксис. | Требует знания асинхронности. | Специфический синтаксис. |
| Сообщество | Огромное количество библиотек. | Активное сообщество, но меньше решений для кинотеатров | Меньше современных решений. |

**Вывод:** Python — оптимальный выбор благодаря фреймворкам (Django/Flask), простоте интеграции платежей и генерации PDF.

**Инструмент проектирования: PyCharm**

**Альтернативы:** VS Code, Sublime Text.

Таблица 2 – Сравнение IDE

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий\IDE | PyCharm | VS Code | Sublime Text |
| Совместимость | Лучшая поддержка Python. | Требует настройки плагинов для Python. | Минималистичный, без глубокой интеграции с Python. |
| Простота | Готовое окружение для Django, PostgreSQL. | Необходимы плагины для работы с БД и Django. | Подходит для небольших проектов. |
| Сообщество | Официальная поддержка JetBrains. | Активно развивается Microsoft. | Меньше обновлений. |

**Вывод:** PyCharm — лучший выбор для Python-разработки благодаря встроенным инструментам для Django, отладки и работы с БД.

**База данных: PostgreSQL**

**Альтернативы: MySQL, SQLite.**

Таблица 3 – Сравнение СУБД

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий\СУБД** | **PostgreSQL** | **MySQL** | **SQLite** |
| **Совместимость** | Поддержка транзакций, сложных запросов. | Хорош для простых проектов, но хуже с конкурентными запросами. | Только для локальных приложений, нет масштабируемости |
| **Простота** | Сложнее в настройке, но надежнее для высоких нагрузок. | Проще для новичков. | Максимально прост, но не для крупных проектов. |
| **Сообщество** | Широко используется в enterprise-проектах. | Популярен, но уступает в функционале. | Не подходит для веб-приложений. |

**Вывод: PostgreSQL обеспечит надежное хранение данных бронирований и поддержит ACID-транзакции, что критично для избежания конфликтов при выборе мест.**

**Итоговое обоснование выбора:**

1. **Python + Django:**

* Готовые решения для аутентификации, админ-панели и работы с БД.
* Библиотеки для генерации PDF (ReportLab) и оплаты (Stripe).

1. **PyCharm:**

* Интеграция с Django и PostgreSQL, встроенный отладчик.

1. **PostgreSQL:**

* Надежность при одновременных транзакциях (бронирование мест).

# **2. Проектирование веб-приложения**

## **2.1. Структурная схема веб-приложения**

Одним из ключевых этапов разработки является создание диаграмм, которые помогут нам лучше понять структуру приложения и его общую функциональность.

Диаграмма прецедентов иллюстрирует взаимодействия между актерами и прецедентами. Она позволяет описать систему на концептуальном уровне. Прецедент представляет собой возможность моделируемой системы, позволяющую пользователю достичь конкретного, измеримого и полезного результата.

На рисунке 1 показана диаграмма Use Case View, демонстрирующая структурную схему веб-приложения «Сайт кинотеатра» для роли «Пользователь».

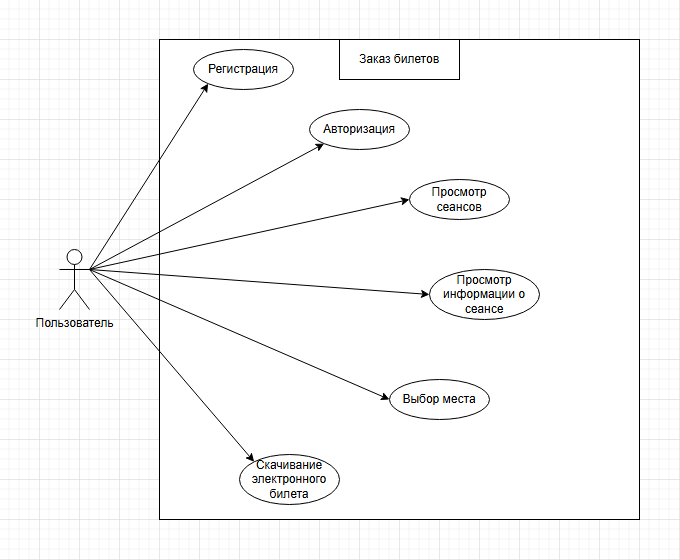


Рисунок 1 – Диаграмма прецедентов

Диаграмма деятельности – это UML-диаграмма, на которой отображаются действия, состояния которых описаны на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения, представленная в виде координированного последовательного и параллельного выполнения подчинённых элементов, включая вложенные виды деятельности и отдельные действия. Эти элементы соединены потоками, которые направляются от выходов одного узла к входам другого.

Диаграммы деятельности применяются для моделирования бизнес-процессов, технологических процессов, а также последовательных и параллельных вычислений.

На рисунке 2 представлена диаграмма деятельности.

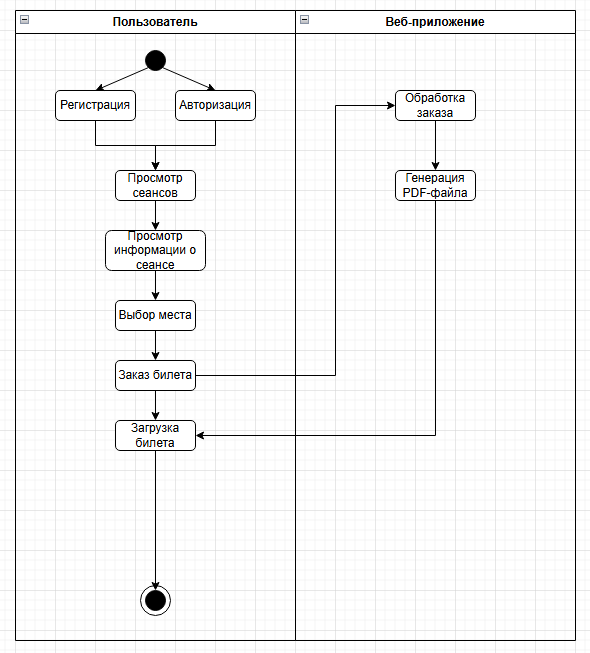


Рисунок 2 – Диаграмма деятельности

Диаграмма декомпозиций — это инструмент структурного анализа, который разбивает сложную систему на иерархию вложенных компонентов. Она демонстрирует, как высокоуровневые функции системы декомпозируются на подфункции, а те, в свою очередь, на более детальные операции. На рисунке 3 изображена диаграмма декомпозиций.

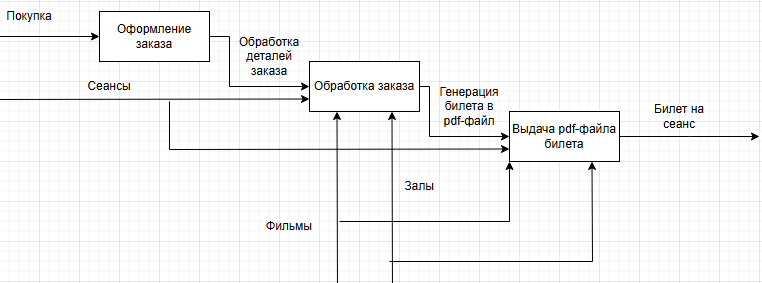


Рисунок 3 – Диаграмма декомпозиций

Контекстная диаграмма — это модель верхнего уровня, которая изображает систему как единый "чёрный ящик" и её взаимодействие с внешними сущностями. Она отвечает на вопрос: "С кем и как система обменивается данными?". На рисунке 4 изображена контекстная диаграмма.

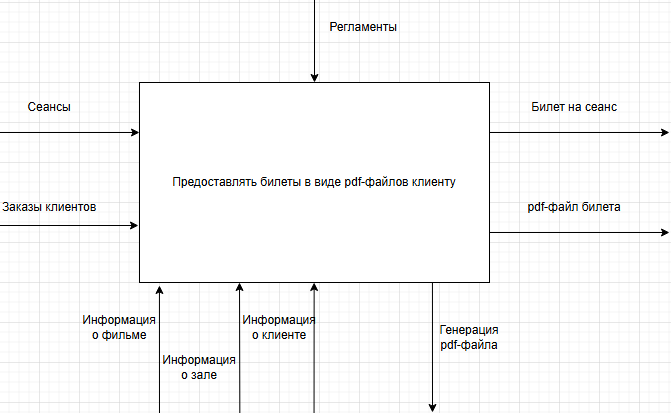


Рисунок 4 – Контекстная диаграмма

Разработанные диаграммы (прецедентов, деятельности, декомпозиций, контекстная) комплексно отображают архитектуру веб-приложения, ключевые бизнес-процессы и взаимодействие компонентов системы с внешними сущностями. Эти модели стали основой для последующей детализации и реализации функционала системы бронирования билетов.

## **2.2. Проектирование базы данных**

Перед началом разработки программного обеспечения важно спроектировать базу данных, определив, с какими данными будут работать пользователи системы и как эти данные взаимосвязаны.

Цель ER-моделирования заключается в создании удобных для восприятия человеком методов сбора и представления информации.

Основные элементы ER-моделей – сущности и их атрибуты.

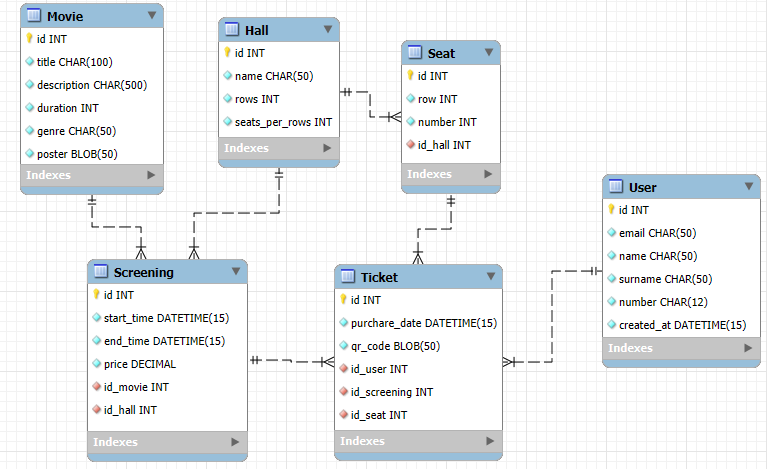


Рисунок 5 – ER-модель

База данных веб-приложения создана и реализована в СУБД PostgreSQL и состоит из 5 таблиц

Таблица 4 – Таблица "User"

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | Int | Идентификатор |
| email | Char(50) | Электронная почта |
| name | Char(50) | Имя |
| surname | Char(50) | Фамилия |
| number | Char(12) | Номер телефона |
| created\_at | DateTime | Дата регистрации |

Таблица 5 – Таблица "Hall"

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | Int | Идентификатор |
| name | Char(50) | Название |
| rows | Int | Количество рядов |
| seats\_per\_row | Int | Количество мест в ряду |

Таблица 6 – Таблица "Movie"

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | Int | Идентификатор |
| title | Char(100) | Название |
| description | Char(500) | Описание |
| duration | Interval | Продолжительность |
| genre | Char(50) | Жанр |
| poster | Blob | Постер |

Таблица 7 – Таблица "Screening"

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | Int | Идентификатор |
| start\_time | DateTime | Дата и время начала |
| end\_time | DateTime | Дата и время окончания |
| price | Decimal(6;2) | Стоимость |
| id\_movie | Int | Идентификатор фильма |
| id\_hall | Int | Идентификатор зала |

Таблица 8 – Таблица "Seat"

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | Int | Идентификатор |
| row | Int | Ряд |
| number | Int | Место |
| id\_hall | Int | Идентификатор зала |

Таблица 9 – Таблица "Ticket"

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | Int | Идентификатор |
| purchare\_date | DateTime | Дата покупки |
| qr\_code | Image | QR-код |
| id\_user | Int | Идентификатор пользователя |
| id\_screening | Int | Идентификатор сеанса |
| id\_seat | Int | Идентификатор места |

База данных была приведена ко второй нормальной форме, но перед этим прошла два этапа.

Нормальная форма – это свойство отношения в реляционной модели данных, которое характеризует его с точки зрения избыточности, способной привести к логическим ошибкам при выборке или изменении данных. Нормальная форма определяется набором требований, которым должно соответствовать отношение.

При нормализации до первой нормальной формы были выполнены условия, позволяющие считать таблицу атомарной, то есть каждое поле содержит только одно значение.

Для достижения второй нормальной формы необходимо, чтобы таблица уже соответствовала первой нормальной форме, а также чтобы таблицы были связаны между собой. Это предотвращает дублирование данных: если записи повторяются в разных таблицах, потребуется изменять информацию в обеих таблицах, что нежелательно.

Таким образом, представлена вся необходимая информация для понимания системы хранения данных.

## **2.3. Проектирование пользовательского интерфейса**

Пользовательский интерфейс – это средства взаимодействия между человеком и компьютером. Проще говоря, интерфейс представляет собой внешнюю часть программы или устройства, с которой работает пользователь.

Интерфейсы являются ключевым элементом взаимодействия всех современных информационных систем и приложений. Если интерфейс какого-либо объекта остаётся стабильным и стандартизированным, это позволяет изменять сам объект, не затрагивая принципы его взаимодействия с другими объектами.

Данный прототип интерфейса был создан в онлайн-конструкторе Draw.io.

На рисунке 6 изображена главная страница веб-приложения, которая будет отображаться после регистрации и/или авторизации пользователя.

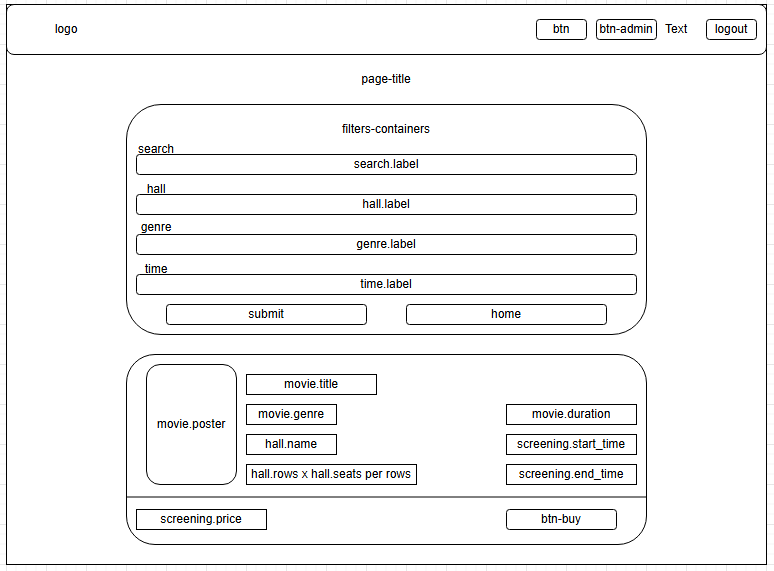


Рисунок 6 – Главная страница

На рисунке 7 изображена страница подробной информации о сеансе, который выберет пользователь на главном экране.

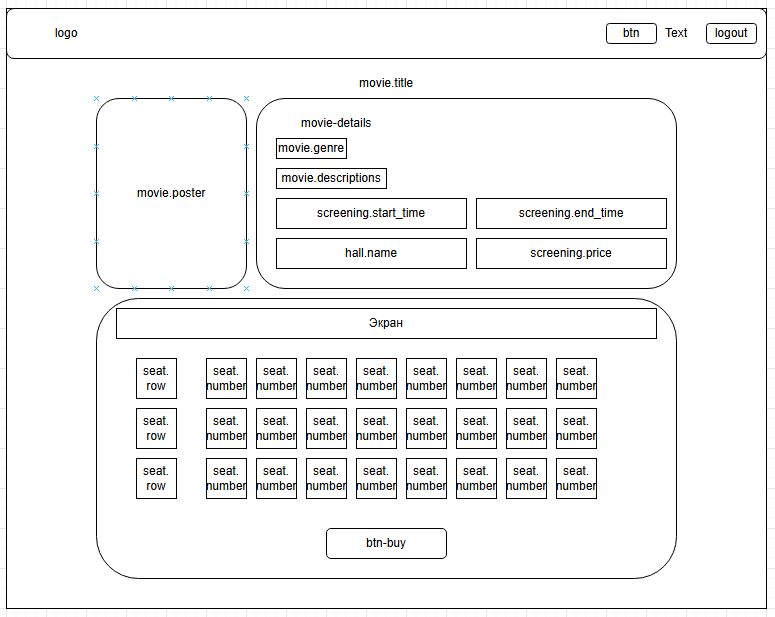


Рисунок 7 – Страница сеанса

На рисунке 8 изображён личный кабинет пользователя в который он может попасть на главной странице. В личном кабинете показаны данные и пользователе и купленные им билеты. Здесь он может поменять свои данные и скачать pdf-файлы билетов.

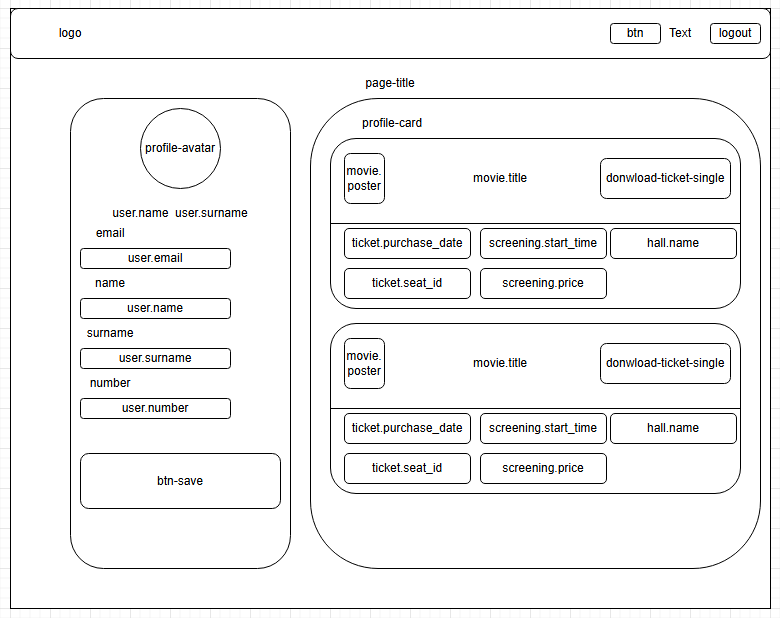


Рисунок 8 – Личный кабинет

Спроектированные прототипы интерфейса (главная страница, страница сеанса, личный кабинет) обеспечивают интуитивно понятную навигацию и покрывают все основные пользовательские сценарии работы с системой бронирования. Их разработка на этапе проектирования позволила заранее определить логику взаимодействия пользователя с приложением и минимизировать риски переделок на этапе реализации.

# **Разработка программного модуля**

## **3.1. Разработка интерфейса программного модуля**

Разработка удобного пользовательского интерфейса – это один из важнейших этапов в процессе создания моего веб-приложения.

Все страницы были написаны на HTML в PyCharm.

На рисунке 9 и 10 изображены страницы регистрации и авторизации. Эти страницы будут самые первые для пользователя, который в первый раз зашёл на сайт.

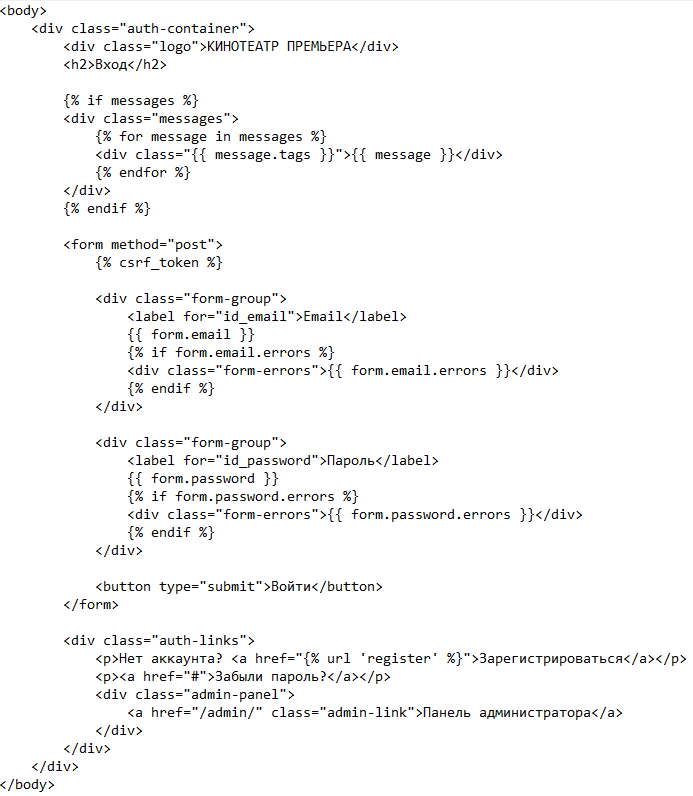


Рисунок 9 – Страница авторизации

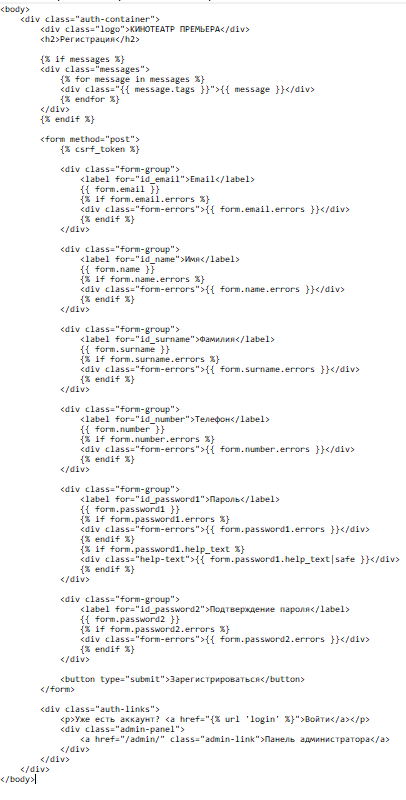


Рисунок 10 – Страница регистрации

На рисунке 11 изображена домашняя страница, где пользователь будет просматривать доступные сеансы. На эту страницу пользователь попадает после регистрации и/или авторизации.

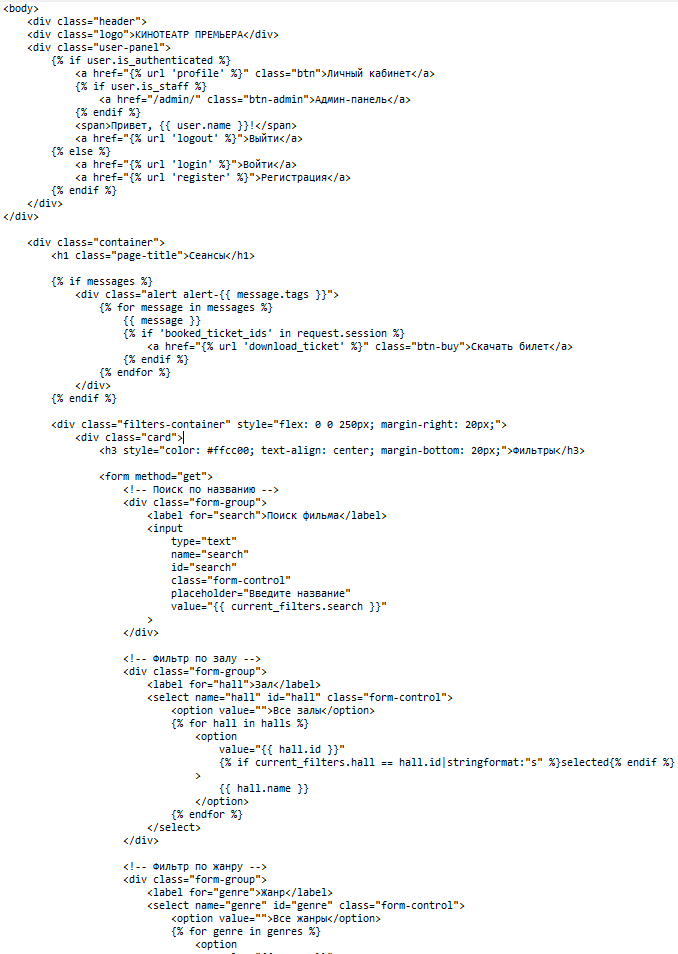


Рисунок 11 – Домашняя страница

Реализованные HTML-страницы (авторизации, регистрации, домашняя) на основе спроектированных прототипов обеспечивают базовый пользовательский интерфейс системы. Их интеграция с бэкенд-логикой Django позволила создать рабочий каркас веб-приложения для дальнейшей функциональной разработки.

## **3.2. Создание базы данных**

Разработка базы данных веб-приложения создавалась в PyCharm, а реализовывалась в PostgreSQL. База данных веб-приложения состоит из 5 таблиц.

Структуры таблиц соответствуют схеме базы данных из пункта 2.2.

Таблица “User” представлена на рисунке 12.

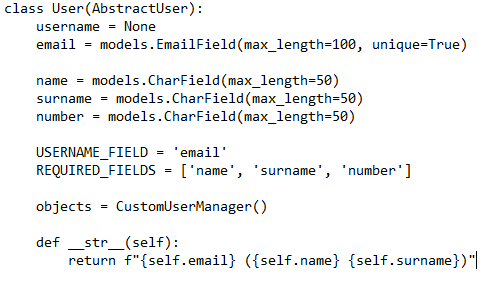


Рисунок 12 *–* Таблица "User"

Таблица “Hall” представлена на рисунке 13.

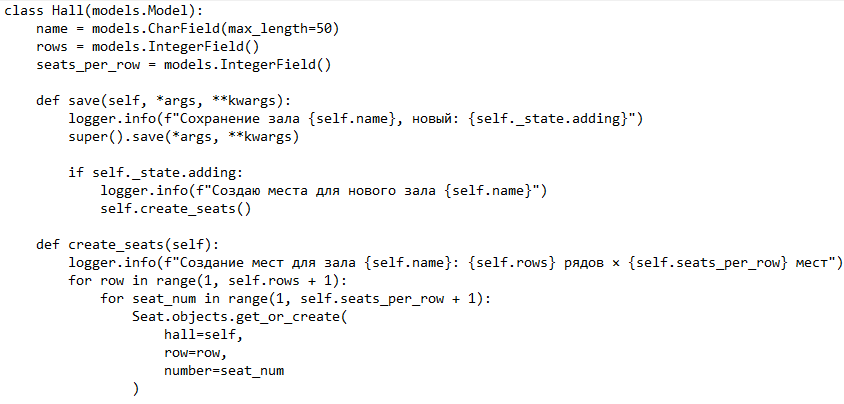


Рисунок 13 *–* Таблица "Hall"

Таблица “Movie” представлена на рисунке 14.

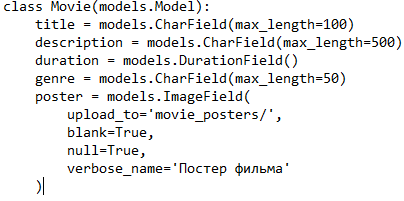


Рисунок 14 *–* Таблица "Movie"

Таблица “Screening” представлена на рисунке 15.

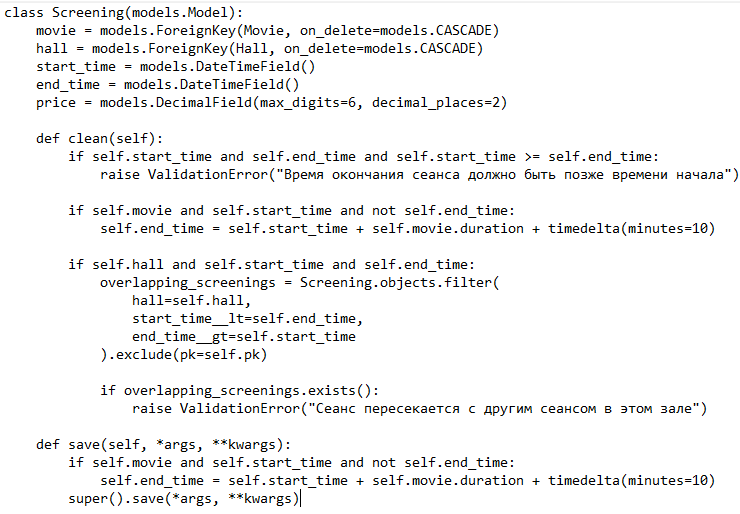


Рисунок 15 *–* Таблица "Screening"

Таблица “Seat” представлена на рисунке 16.

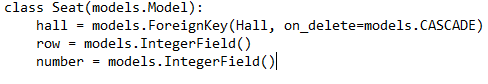


Рисунок 16 *–* Таблица "Seat"

Таблица “Ticket” представлена на рисунке 17.

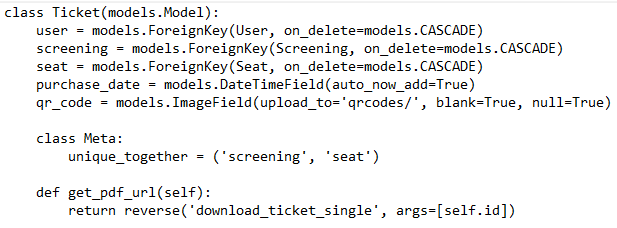


Рисунок 17 *–* Таблица "Ticket"

База данных успешно реализована в PostgreSQL в строгом соответствии с разработанной ER-моделью и приведена ко второй нормальной форме. Структура таблиц (User, Hall, Movie, Screening, Seat, Ticket) обеспечивает целостное хранение данных, необходимых для функционирования системы бронирования билетов кинотеатра.

## **3.3. Разработка программного модуля**

Таблицы в базе данных создаются через модели Django. Заполнение начальных данных происходит через кастомную команду populate\_db и код этой команды изображён на рисунках 18 и 19.

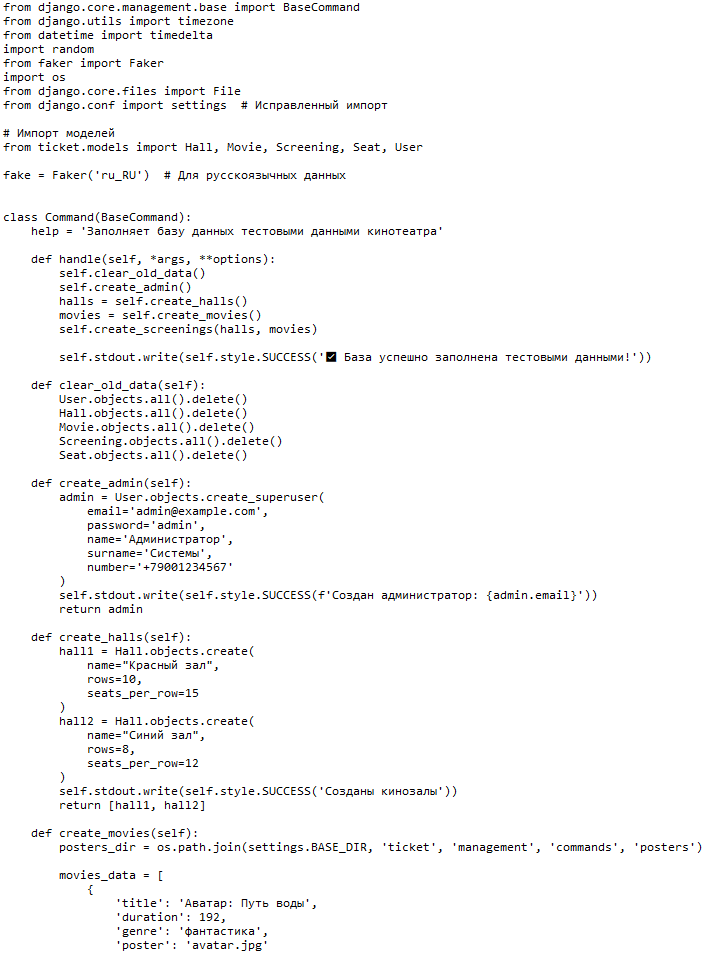


Рисунок 18 *–* Заполнение БД начальными данными

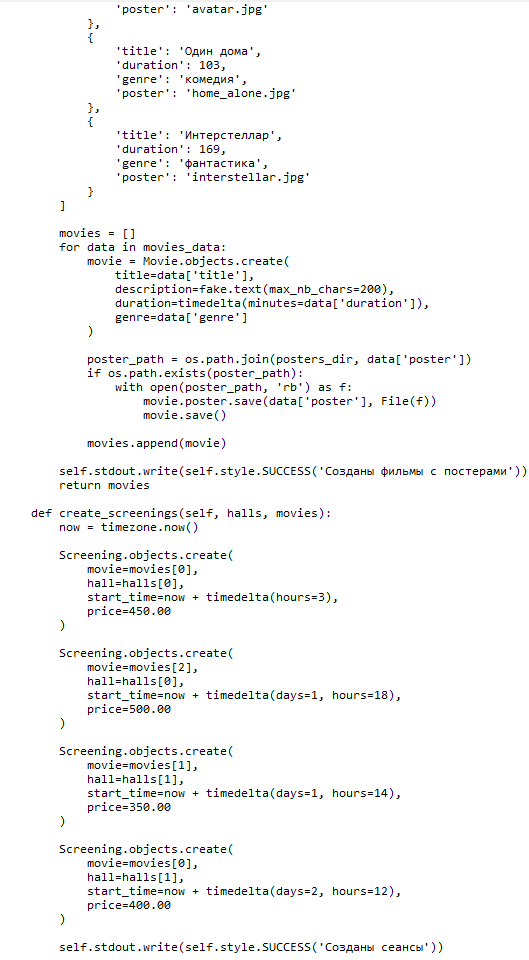


Рисунок 19 *–* Заполнение БД начальными данными

После заполнения БД начальными данными у нас появляется 2 зала, 3 фильма и 4 сеанса, так же создаётся учётная запись системного администратора, который может полностью взаимодействовать со всеми данными из БД. После этого мы переходим на сайт и перед нами страница регистрации. Код регистрации изображён на рисунке 20.

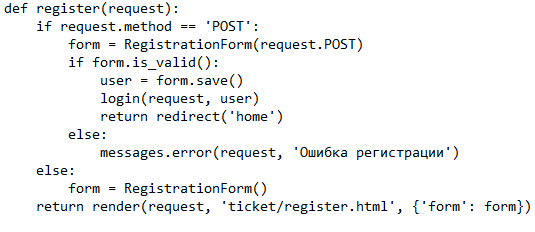


Рисунок 20 *–* Метод регистрации пользователей

Ну или пользователь уже был зарегистрирован на сайте, то он может авторизоваться. Код авторизации изображён на рисунке 21.

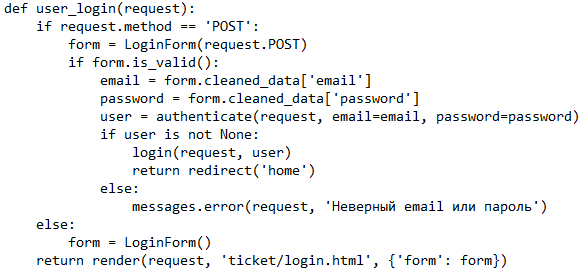


Рисунок 21 *–* Метод авторизации пользователей

На рисунке 22 изображён метод построения домашней странице

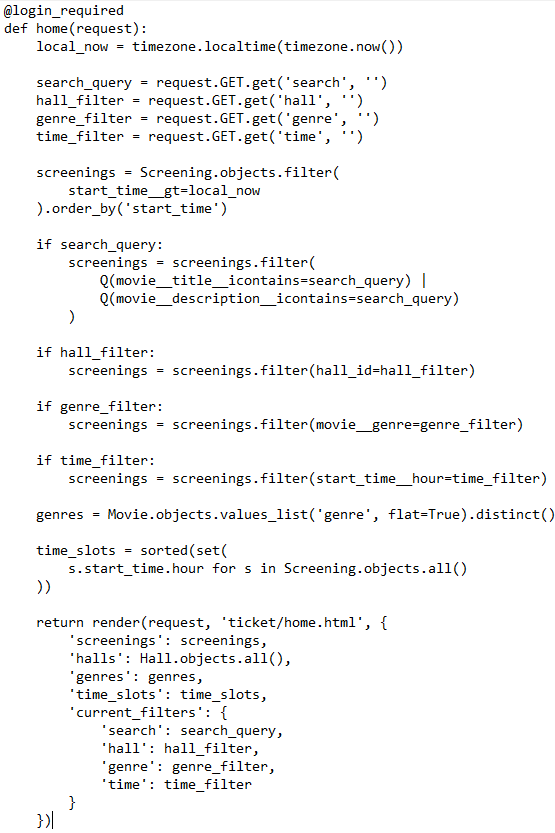


Рисунок 22 – Метод построения Домашней страницы

Ключевые модули системы (регистрация, авторизация, формирование главной страницы) реализованы на Django с использованием ORM для работы с PostgreSQL. Заполнение базы начальными данными через кастомную команду populate\_db обеспечило готовность системы к тестированию базовых сценариев работы пользователей и администратора.

# **Тестирование программного модуля**

Таблица 10 – Сценарий тестирования валидации почты

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 09.06.2025 |
| Приоритет  тестирования | Средний |
| Название теста | Регистрация пользователя |
| Этапы теста | Пользователь регистрируется с неправильной почтой |
| Тестовые данные | User.com, user, user, 88005553535, qwe123asd, qwe123asd |
| Ожидаемый  результат | Появится ошибка: «Введите корректную почту» |
| Фактический  результат | Появилась ошибка «Введите корректную почту» |

Таблица 11 – Чек-лист успешной авторизации пользователя

|  |  |
| --- | --- |
| Текстовый пример # | № 1 |
| Приоритет  тестирования | Средний |
| Заголовок/название теста | Авторизация пользователя |
| Краткое изложение теста | Пользователю нужно ввести свою почту и пароль |
| Этапы теста | Пользователь заходит на страницу авторизации.  Пользователь вводит почту и пароль и нажимает «Войти».  Пользователь попадает на главную страницу. |
| Тестовые данные | [User@gmail.com](mailto:User@gmail.com), user228 |
| Ожидаемый результат | Успешная авторизация |
| Статус | Зачёт |
| Предварительное условие | Пользователь должен иметь учётную запись |
| Постусловие | После авторизации перенаправляется на главную страницу |

Таблица 12 – Сценарий тестирования добавления постеров к фильму формата не .jpg

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 09.06.2025 |
| Приоритет  тестирования | Средний |
| Название теста | Добавления постера иного формата |
| Этапы теста | Админ добавляет постер формата не .jpg |
| Тестовые данные | Я легенда, Без описания, 01:32:00, боевик, ya\_legenda.png |
| Ожидаемый  результат | «Фильм успешно добавлен» |
| Фактический  результат | «Постер имеет не корректный формат» |

Таблица 13 – Чек-лист создания сеансов на одно время в одном зале

|  |  |
| --- | --- |
| Текстовый пример # | № 2 |
| Приоритет  тестирования | Средний |
| Заголовок/название теста | Создание двух сеансов на одно время в одном зале |
| Краткое изложение теста | Админ создаёт два сеанса в одном зале на одно время |
| Этапы теста | Создание первого сеанса.  Создание второго сеанса |
| Тестовые данные | Movie(35), Hall(25), 2025-06-09 07:58:49, 500  Movie(34), Hall(25), 2025-06-09 07:58:49, 350 |
| Ожидаемый результат | Ошибка «Зал уже занят» |
| Статус | Не зачёт |
| Предварительное условие | Зал должен быть пустой до тестирования |
| Постусловие | После попытки создания второго сеанса, система должна выдать ошибку |

Проведенное тестирование программного модуля системы онлайн-бронирования билетов подтвердило корректность реализации ключевых функциональных требований и выявило области для дальнейшей оптимизации.

Программный модуль соответствует основным заявленным функциональным требованиям для пользовательской части. Однако выявленная уязвимость в логике планирования сеансов требует немедленного устранения до ввода системы в промышленную эксплуатацию для обеспечения целостности данных и бесперебойной работы сервиса бронирования. Тестирование подтвердило, что система готова к дальнейшей доработке и интеграции после исправления указанных недостатков.

# **Документирование программного модуля**

## **Руководство пользователя**

**Для посетителей кинотеатра:**

* + 1. Регистрация и вход:
* На главной странице нажмите "Регистрация"
* Заполните форму: email, имя, фамилия, телефон, пароль
* Для входа используйте email и пароль

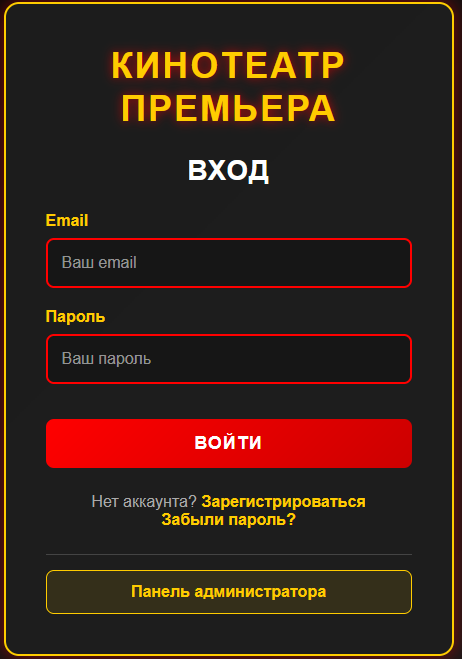


Рисунок 23 – Авторизация



Рисунок 24 – Регистрация

* + 1. Покупка билетов:
* Выберите сеанс на главной странице
* На странице сеанса выберите места на схеме зала
* Нажмите "Купить выбранные билеты"
* Скачайте PDF-билет по ссылке в сообщении об успешной покупке



Рисунок 25 – главная страница



Рисунок 26 – Схема зала

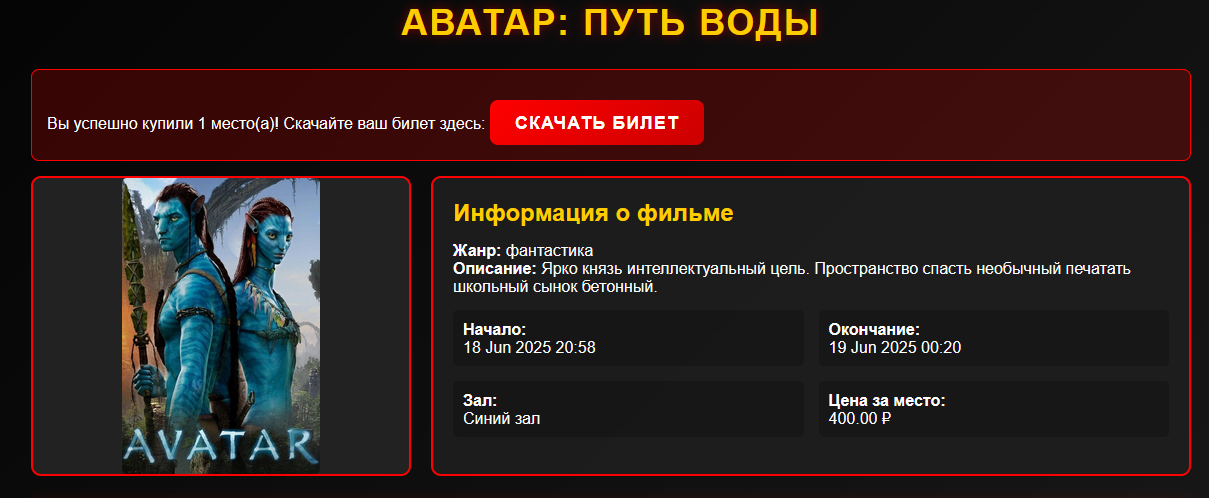


Рисунок 27 – Ссылка на PDF-билет

* + 1. Личный кабинет:
* Просмотр истории покупок
* Скачивание ранее купленных билетов
* Редактирование личных данных

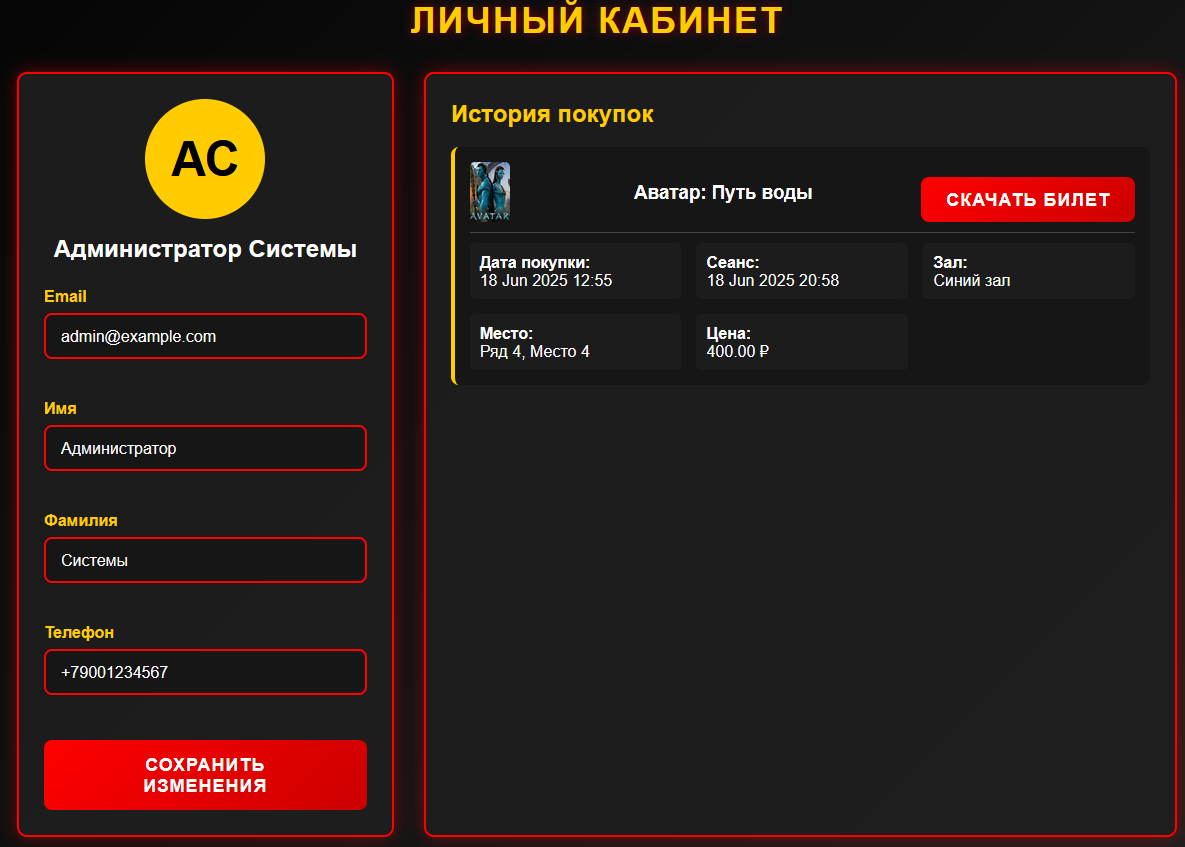


Рисунок 28 – Личный кабинет

**Для администратора:**

1. Управление контентом:

* Добавление/редактирование фильмов (с постерами)
* Создание кинозалов с указанием рядов и мест
* Планирование сеансов с автоматическим расчетом времени окончания

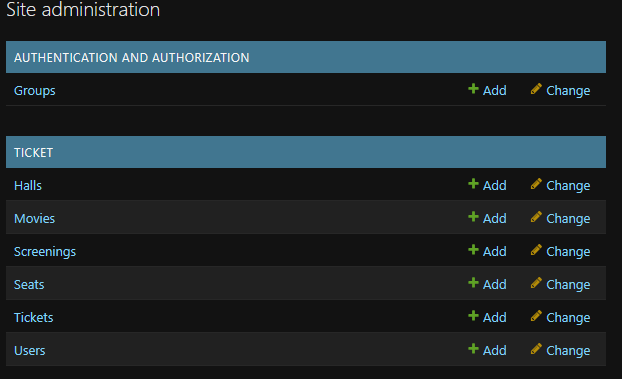


Рисунок 29 – Админ-панель

1. Доступ:

* Используйте кнопку "Админ-панель" в шапке сайта
* Логин: [admin@example.com](https://mailto:admin@example.com), Пароль: admin

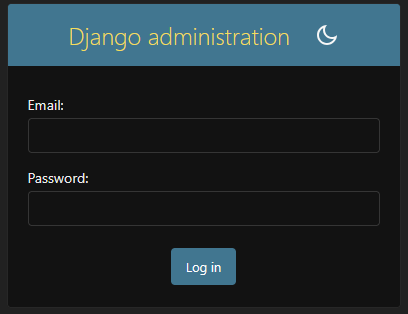


Рисунок 30 – Админ-панель

Предоставленное руководство содержит исчерпывающие визуальные и текстовые инструкции по использованию всех основных функций системы как для посетителей кинотеатра, так и для администраторов.

# **Оценка возможности модернизации программного продукта**

Программный продукт обладает значительным потенциалом для модернизации:

1. Функциональные улучшения:

* Внедрение онлайн-оплаты (Stripe, CloudPayments)
* Система лояльности и скидочные программы
* Интеграция с системой попкорн-баров и дополнительных услуг
* Модуль аналитики посещаемости и финансовой отчетности

1. Технические улучшения:

* Оптимизация производительности с помощью кэширования
* Переход на асинхронную обработку запросов (Celery + Redis)
* Реализация REST API для мобильных приложений

1. Масштабирование:

* Поддержка сети кинотеатров (мультикинотеатральность)
* Геораспределение нагрузки между серверами
* Репликация базы данных для отказоустойчивости

1. Безопасность:

* Внедрение двухфакторной аутентификации
* Регулярные аудиты безопасности
* Шифрование персональных данных в базе

Архитектура приложения (Django + PostgreSQL) позволяет легко внедрять новые функции без переписывания основной логики.

# **Производственные задачи**

В ходе производственной практики были решены следующие задачи:

1. Разработка архитектуры системы:

* Проектирование структуры базы данных
* Создание ER-диаграмм и схемы взаимодействия компонентов
* Оптимизация запросов к базе данных

1. Реализация бизнес-логики:

* Система бронирования мест с защитой от коллизий
* Алгоритм проверки пересечения сеансов
* Генерация PDF-билетов с QR-кодами

1. Разработка пользовательского интерфейса:

* Адаптивный дизайн для мобильных устройств
* Интерактивная карта выбора мест в зале
* Система уведомлений для пользователей

1. Тестирование и отладка:

* Написание тестов для критически важных функций
* Оптимизация производительности ключевых операций
* Рефакторинг кода для улучшения читаемости

1. Документирование:

* Создание руководства пользователя
* Написание технической документации
* Составление отчета о проделанной работе

# **ЗАКЛЮЧЕНИЕ**

Данная работа посвящена разработке веб-приложения. Была поставлена и достигнута цель работы, состоявшая в разработке веб-приложения.

Цель работы предполагала решение следующих задач:

* Проведён анализ уже существующих сайтов кинотеатров.
* На основе собранных данных создано своё веб-приложение.
* Спроектирован пользовательский интерфейс.
* Разработана база данных.
* Реализовано веб-приложение.

За время выполнения работы были улучшены навыки владения такими программами как PyCharm и PostgreSQL.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Django Documentation. [Электронный ресурс]. URL: <https://docs.djangoproject.com/> (дата обращения: 09.06.2025).
2. PostgreSQL Documentation. [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата обращения: 11.06.2025).
3. ReportLab User Guide. [Электронный ресурс]. URL: <https://www.reportlab.com/docs/reportlab-userguide.pdf> (дата обращения: 12.06.2025).
4. Bootstrap Documentation. [Электронный ресурс]. URL: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> (дата обращения: 12.06. 2025).
5. MDN Web Docs. [Электронный ресурс]. URL: <https://developer.mozilla.org/ru/> (дата обращения: 16.06.2025).
6. Stack Overflow. [Электронный ресурс]. URL: <https://stackoverflow.com/> (дата обращения: 16.06.2025).
7. DigitalOcean Community Tutorials. [Электронный ресурс]. URL: <https://www.digitalocean.com/community/tutorials> (дата обращения: 17.06.2025).