**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ПП.09.02.07-1.25.201.18

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

|  |  |  |
| --- | --- | --- |
| 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**

[**1.Предпроектное исследование** 3](#_Toc200089906)

[1.1. Описание предметной области 3](#_Toc200089907)

[1.2. Анализ инструментов 4](#_Toc200089908)

[**2.Проектирование веб-приложения** 7](#_Toc200089909)

[2.1. Структурная схема веб-приложения 7](#_Toc200089910)

[2.2. Проектирование базы данных 9](#_Toc200089911)

[2.3. Проектирование пользовательского интерфейса 12](#_Toc200089912)

[**3.** **Разработка программного модуля** 14](#_Toc200089913)

[3.1. Разработка интерфейса программного модуля 14](#_Toc200089914)

[3.2. Создание базы данных 16](#_Toc200089915)

[3.3. разработка программного модуля 18](#_Toc200089916)

[**4.** **Тестирование программного модуля** 21](#_Toc200089917)

[**5.** **Документирование программного модуля** 23](#_Toc200089918)

[5.1. Руководство пользователя 23](#_Toc200089919)

[**6.** **Оценка возможности модернизации программного продукта** 24](#_Toc200089920)

[**7.** **Производственные задачи** 25](#_Toc200089921)

[**Заключение** 26](#_Toc200089922)

[**Список используемых источников** 27](#_Toc200089923)

# **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, типы экранов, цены).
* Выбор мест:
  + Визуализация схемы зала в реальном времени.
  + Бронирование мест с защитой от коллизий.
* Оплата:
  + Интеграция с платежными системами (например, Stripe, PayPal).
  + Подтверждение оплаты и генерация PDF-билета (с QR-кодом и данными сеанса).
* Управление пользователями:
  + Регистрация, авторизация, восстановление пароля.
  + История покупок и возможность отмены бронирования.
* Административная панель:
  + Добавление/редактирование сеансов, управление залами, настройка цен.
  + Просмотр статистики продаж.

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 (Django, Flask), много библиотек для работы с PDF (ReportLab, PyPDF2). | Быстрая обработка запросов, но сложнее с синхронными задачами (генерация PDF). | Устаревшие фреймворки (Laravel), меньше удобных инструментов для PDF. |
| Простота | Чистый синтаксис, легко изучать. | Требует знания асинхронности. | Специфический синтаксис. |
| Сообщество | Огромное количество библиотек (Django для админ-панели, Stripe для оплаты). | Активное сообщество, но меньше готовых 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 (автодополнение, отладчик, интеграция с Django). | Универсальный, требует настройки плагинов для Python. | Минималистичный, без глубокой интеграции с Python. |
| Простота | Готовое окружение для Django, PostgreSQL. | Нужно устанавливать 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 (бронирование мест). | 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 продакшена. |
| **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).
2. **PyCharm:**
   * Интеграция с Django и PostgreSQL, встроенный отладчик.
3. **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».

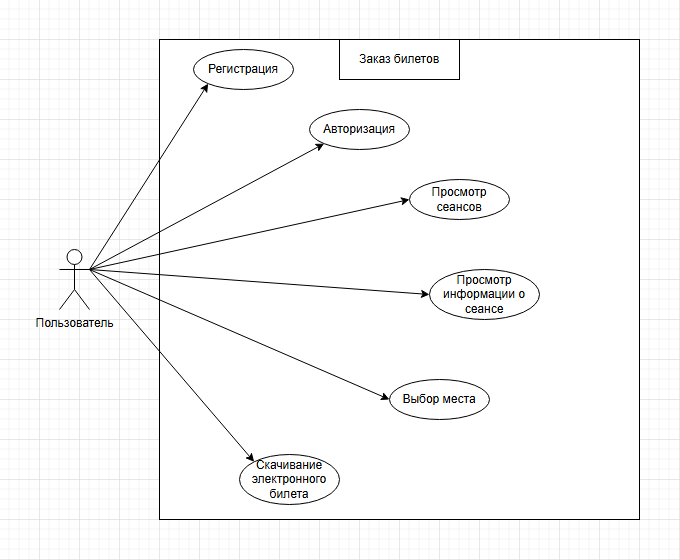


Рисунок 1 - Диаграмма прецедентов

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.

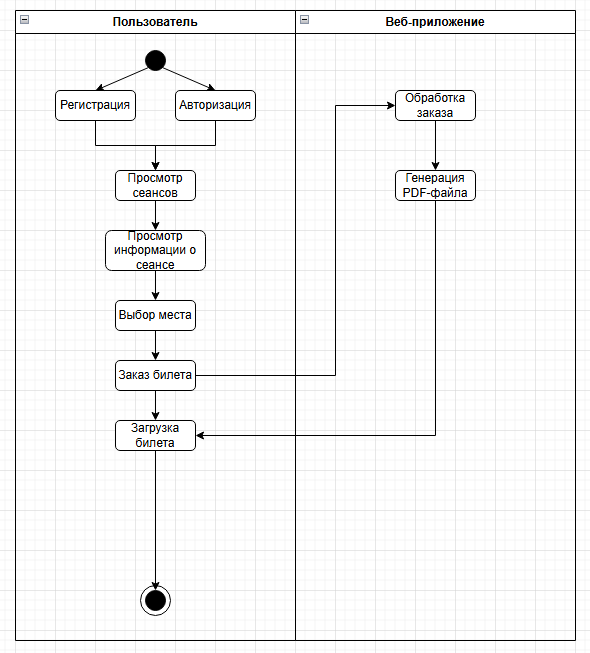


Рисунок 2 - Диаграмма деятельности

## 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.

Основными элементами ER-моделей являются сущности и их характеристики (атрибуты).

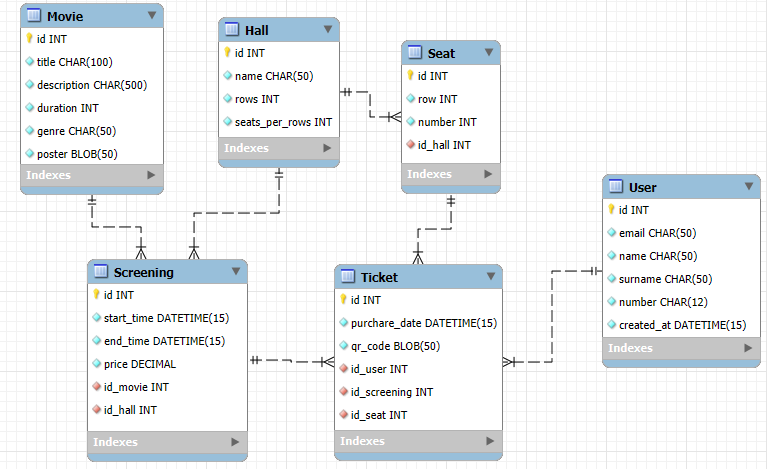


Рисунок 3 - ER-модель

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: нормализацию до первой нормальной формы и нормализацию до второй нормальной формы.

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. Существует шесть нормальных форм, и наша база данных достигла второй.

При нормализации до первой нормальной формы были выполнены условия, позволяющие считать таблицу атомарной, то есть каждое поле содержит только одно значение. Это условие было соблюдено, и таблица соответствует первой нормальной форме.

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

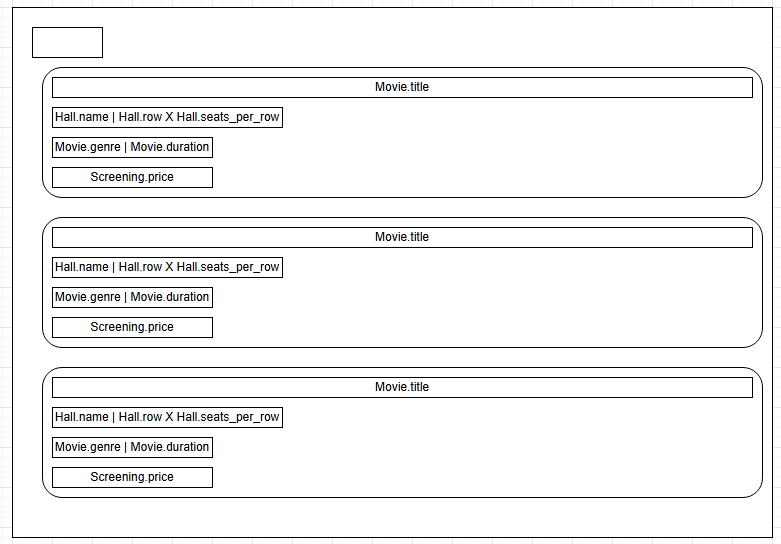


Рисунок 4 - Главная страница

На рисунке 5 изображена страница подробной информации о сеансе, который выберет пользователь на главном экране.

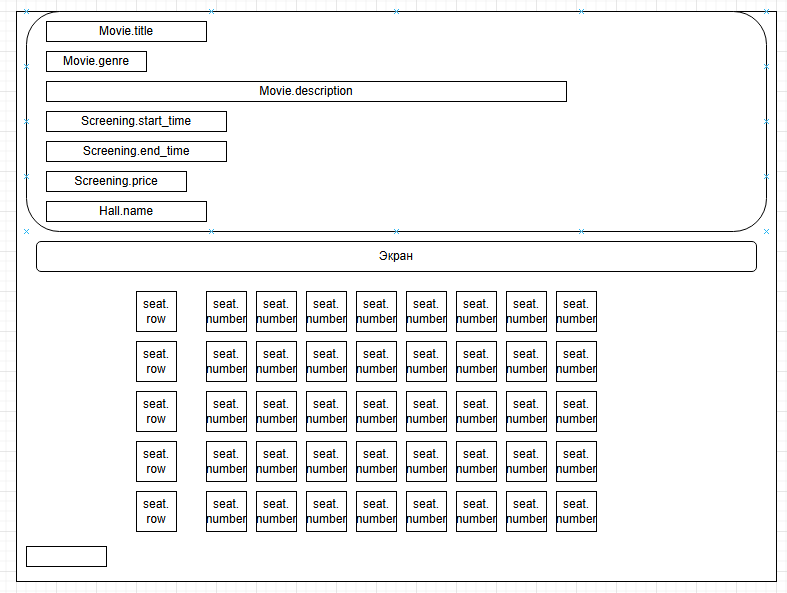


Рисунок 5 - Страница сеанса

# **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 6 и 7 изображены страницы регистрации и авторизации. Эти страницы будут самые первые для пользователя, который впервый раз зашёл на сайт.

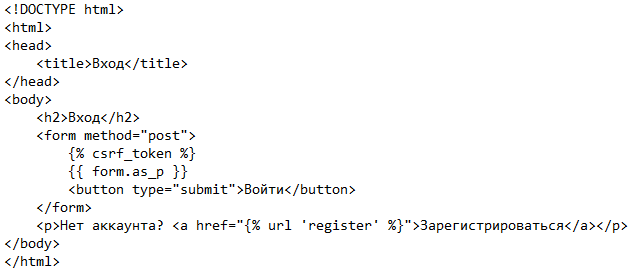


Рисунок 6 - Страница авторизации

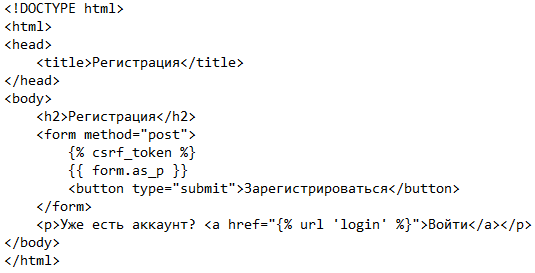


Рисунок 7 - Страница регистрации

На рисунке 8 изображена домашняя страница, где пользователь будет просматривать доступные сеансы. На эту страницу пользователь попадает после регистрации и/или авторизации.

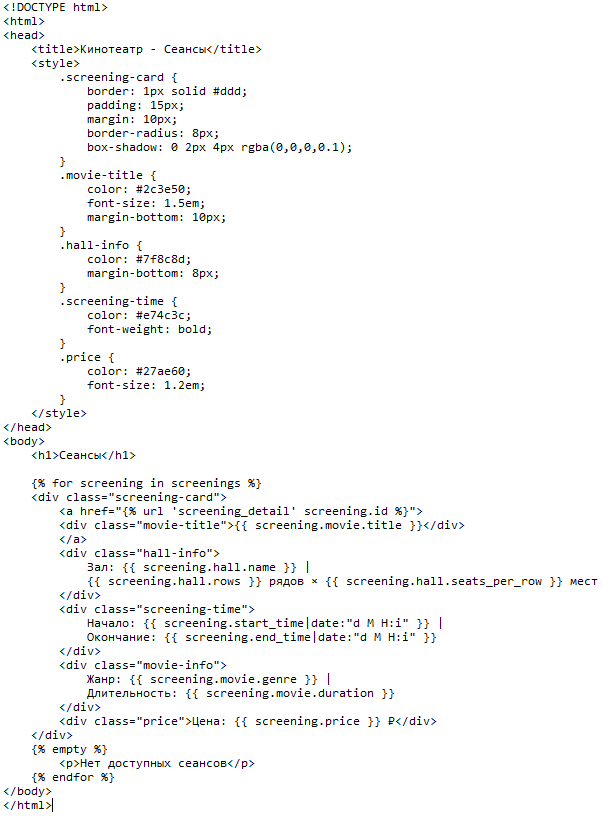


Рисунок 8 - Домашняя страница

## 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 9.

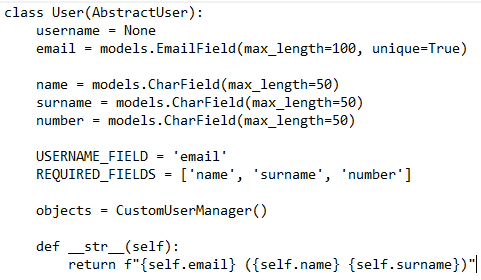


Рисунок 9 - Таблица "User"

Таблица “Hall” представлена на рисунке 10.

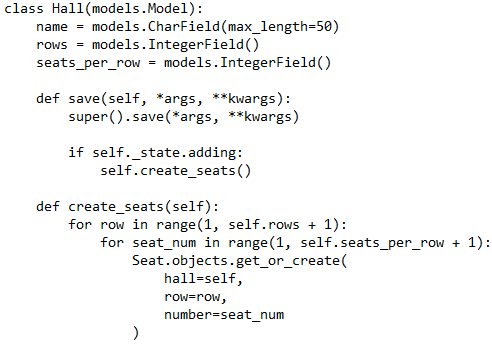


Рисунок 10 - Таблица "Hall"

Таблица “Movie” представлена на рисунке 11.

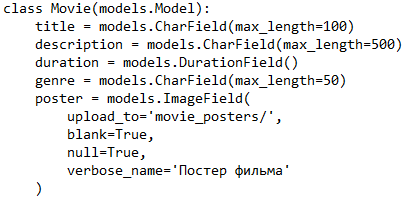


Рисунок 11 - Таблица "Movie"

Таблица “Screening” представлена на рисунке 12.

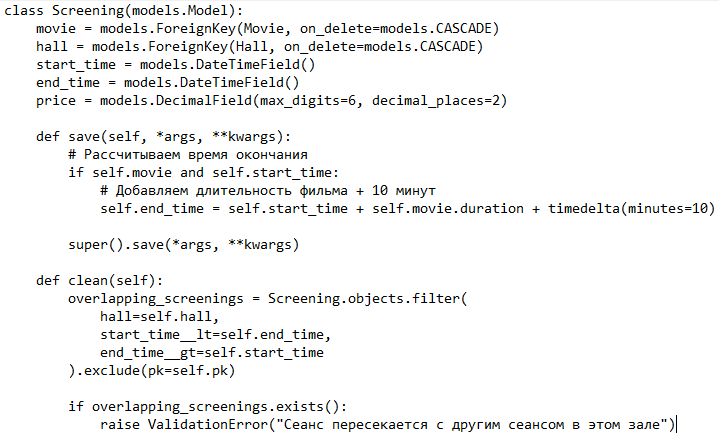


Рисунок 12 - Таблица "Screening"

Таблица “Seat” представлена на рисунке 13.

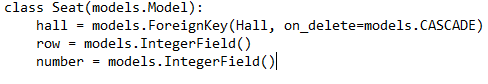


Рисунок 13 - Таблица "Seat"

Таблица “Ticket” представлена на рисунке 14.

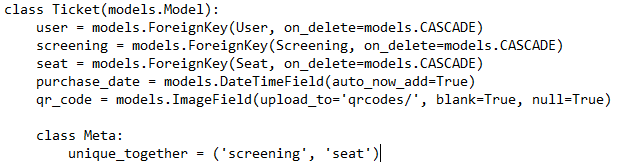


Рисунок 14 - Таблица "Ticket"

## 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 15 и 16.



Рисунок 15 - Заполнение БД начальными данными

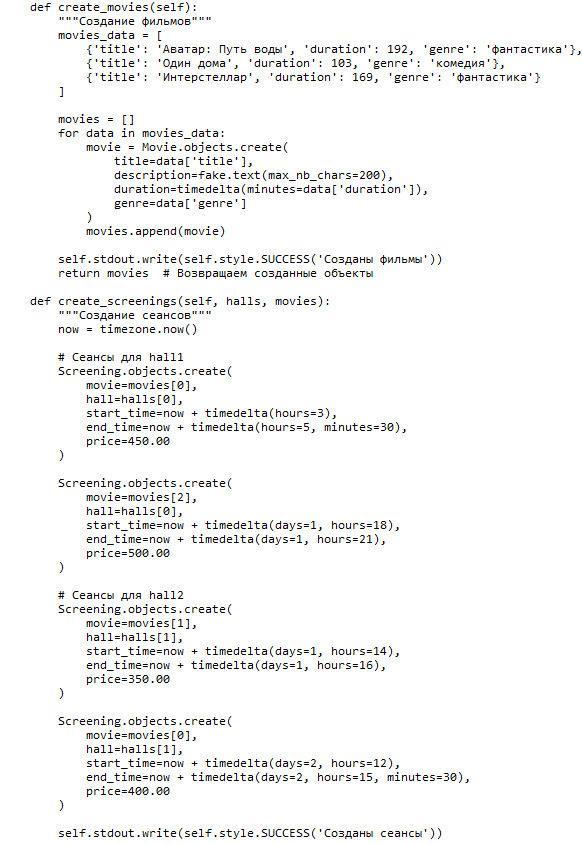


Рисунок 16 - Заполнение БД начальными данными

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 17.

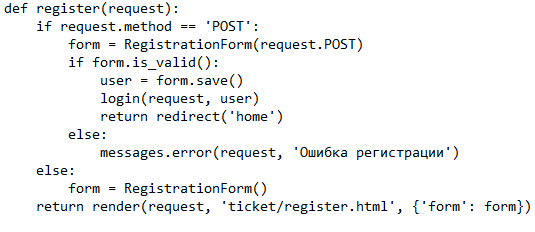


Рисунок 17 - Метод регистрации пользователей

Ну или пользователь уже был зарегистрирован на сайте, то он может авторизоваться. Код авторизации изображён на рисунке 18.

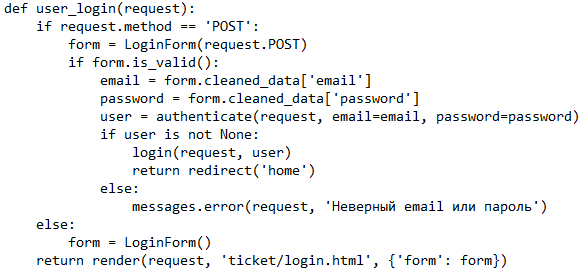


Рисунок 18 - Метод авторизации пользователей

# **Тестирование программного модуля**

Тестирование программного модуля проводилось по следующим направлениям:

1. Модульное тестирование (Unit Testing):

* Проверка корректности работы моделей Django (User, Hall, Movie, Screening, Seat, Ticket)
* Тестирование валидации данных в формах (RegistrationForm, LoginForm, ScreeningForm)
* Проверка бизнес-логики:
  + Автоматический расчет времени окончания сеанса
  + Проверка пересечения сеансов в одном зале
  + Корректное создание мест при добавлении нового зала

1. Интеграционное тестирование:

* Тестирование процесса покупки билетов:
  + Выбор мест → Бронирование → Генерация PDF
* Проверка взаимодействия между компонентами:
  + Регистрация → Авторизация → Покупка билета
  + Добавление фильма → Создание сеанса → Отображение на главной странице

1. Системное тестирование:

* Проверка работы системы под нагрузкой (10+ одновременных бронирований)
* Тестирование безопасности:
  + Защита от SQL-инъекций
  + Валидация входных данных
  + Защита CSRF-токенами

1. Тестирование пользовательского интерфейса:

* Проверка отзывчивости интерфейса на разных устройствах
* Тестирование интерактивной карты мест в зале
* Проверка генерации PDF-билетов с корректными данными

Результаты тестирования показали стабильную работу системы. Основные выявленные и исправленные проблемы:

* Конфликты при одновременном бронировании одних и тех же мест
* Некорректное отображение времени сеансов в разных часовых поясах
* Оптимизация производительности при генерации PDF для больших заказов

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

**Для администратора:**

1. Управление контентом:

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

1. Доступ:

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

# **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
* Внедрение полнотекстового поиска по фильмам

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: 29.05.2025).
2. PostgreSQL Documentation. [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата обращения: 30.05.2025).
3. ReportLab User Guide. [Электронный ресурс]. URL: <https://www.reportlab.com/docs/reportlab-userguide.pdf> (дата обращения: 02.06.2025).
4. Bootstrap Documentation. [Электронный ресурс]. URL: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> (дата обращения: 03.06. 2025).
5. MDN Web Docs. [Электронный ресурс]. URL: <https://developer.mozilla.org/ru/> (дата обращения: 04.06.2025).
6. Stack Overflow. [Электронный ресурс]. URL: <https://stackoverflow.com/> (дата обращения: 05.06.2025).
7. DigitalOcean Community Tutorials. [Электронный ресурс]. URL: <https://www.digitalocean.com/community/tutorials> (дата обращения: 06.06.2025).