МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО–КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ПЕРСПЕКТИВНОЙ ИНЖЕНЕРИИ

ДЕПАРТАМЕНТ ЦИФРОВЫХ, РОБОТОТЕХНИЧЕСКИХ СИСТЕМ И ЭЛЕКТРОНИКИ

МЕЖИНСТИТУТСКАЯ БАЗОВАЯ КАФЕДРА

**КУРСОВОЙ ПРОЕКТ**

по дисциплине

«Междисциплинарный проектный практикум»

на тему:

«Разработка и внедрение системы управления бронированием номеров для отеля на основе фреймворка Django»

**Выполнила:**

Коновалова Виктория Николаевна

студент 4 курса

группы ПИЖ-б-о-21-1

направление подготовки 09.03.04 л «Программная инженерия»

направленность (профиль)

«Разработка и сопровождение

программного обеспечения»

очной формы обучения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Руководитель проекта:**

Свистунов И.В., ктн, доцент

межинститутской базовой кафедры

Проект допущен к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя) (дата)

Проект выполнен и

защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| Члены комиссии: |  |  |
| и.о. директора департамент цифровых, робототехнических систем и электроники |  | И. В. Азаров |
| зав. МИБК | (подпись) | Е. В. Новикова |
| доцент МИБК | (подпись) | И.В. Свистунов |

(подпись)

Ставрополь, 2024 г.

|  |  |
| --- | --- |
|  |  |

Министерство науки и высшего образования и Российской Федерации Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Департамент цифровых, робототехнических систем и электроники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление подготовки 09.03.04 «Программная инженерия» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность (профиль) «Разработка и сопровождение программного обеспечения»\_\_\_

ЗАДАНИЕ

на курсовую работу

|  |  |  |
| --- | --- | --- |
| студента | Коновалова Виктория Николаевна | |
| по дисциплине | | (фамилия, имя, отчество)  «Междисциплинарный проектный практикум» | |
| 1. Тема работы: Разработка и внедрение системы управления бронированием номеров для отеля на основе фреймворка Django | | | |
| 2. Цель: повышение уровня профессиональной подготовки путем углубления и закрепления теоретических знаний и практических навыков, приобретенных в результате изучения дисциплины «Междисциплинарный проектный практикум»; подготовка к самостоятельной разработке программного обеспечения с использованием современных информационных технологий.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
| 3. Задачи:  3.1 Анализ прикладной задачи и методов ее решения. Обоснование выбора средств, технологий и алгоритмов решения прикладной задачи.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  3.2 Разработка алгоритмов решения задачи.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  3.3 Реализация программного кода \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  3.4 Отладка и тестирование программного кода\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
| 4. Перечень подлежащих разработке вопросов:  а) теоретической части: изучение и анализ литературы, постановка условия задачи, выбор и описание методов и библиотек для ее решения и технологий, среды программирования  б) проектная часть: проектирование UML-диаграмм, IDEF, интерфейса и пр.  в) реализация: описание структуры проекта, описание файлов проекта, разработка программного кода, тестирование программы | | | |

|  |
| --- |
| 5. Исходные данные:  а) по литературным источникам: ГОСТы, международные стандарты, программные средства, используемые при разработке программного обеспечения  б) исходные данные, подготовленные для тестирования объекта профессиональной деятельности (информационной системы/приложения/программного продукта)  6. Список рекомендуемой литературы: монографии, диссертации, научные статьи, учебнометодические материалы, ссылки на официальные сайты, содержащие информацию необходимую для решения поставленных в работе задач (перечислить 4 – 5 основных литературных источников, которые применялись при выполнении КР) |

7. Контрольне сроки представления отдельных разделов курсового проекта:

25 % - представление первого раздела «30» сентября 2024 г.

50 % - представление второго раздела «28» октября 2024 г.

75 % - представление третьего раздела «25» ноября 2024 г.

100 % - представление работы на отзыв «20» декабря 2024 г.

8. Срок защиты студентом курсового проекта «26» декабря 2024 г.

Дата выдачи задания «13» сентября 2024 г.

|  |  |  |
| --- | --- | --- |
| Руководитель курсовой работы |  |  |
| К. технических н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | И. В. Свистунов |
| *(учебная степень, звание)* | *(личная подпись)* | *(инициалы, фамилия)* |
| Задание принял к исполнению студент очной формы обучения 4 курса  группы ПИЖ-б-о-21-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Коновалова В.Н. | | |
| *(личная подпись) (инициалы, фамилия)* . | | |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc186109959)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7](#_Toc186109960)

[1.1 Предметная область 7](#_Toc186109961)

[1.2 Техническое задание 7](#_Toc186109962)

[1.3 Технологии для разработки программы 9](#_Toc186109963)

[1.3.1. Django Framework 9](#_Toc186109964)

[1.3.2 SQLLite 11](#_Toc186109965)

[1.3.3 HTML и CSS 13](#_Toc186109966)

[1.3.4 Bootstrap 14](#_Toc186109967)

[1.3.5 JavaScript 16](#_Toc186109968)

[1.3.6 Git 18](#_Toc186109969)

[2 ПРОЕКТНАЯ ЧАСТЬ 20](#_Toc186109970)

[2.1 Описание архитектуры системы 20](#_Toc186109971)

[2.2 Алгоритм создания и работы приложения 21](#_Toc186109972)

[2.3 Стандарт разработки 23](#_Toc186109973)

[2.4 Структура базы данных 24](#_Toc186109974)

[2.5 Макет системы 25](#_Toc186109975)

[2.6 Диаграммы 27](#_Toc186109976)

[3 ПРАКТИЧЕСКАЯ ЧАСТЬ 30](#_Toc186109977)

[3.1 Реализация системы 30](#_Toc186109978)

[3.1.1 Модели 30](#_Toc186109979)

[3.1.2 Шаблоны 31](#_Toc186109980)

[3.1.3 Интерфейс системы 33](#_Toc186109981)

[3.2 Руководство пользователя приложения 34](#_Toc186109982)

[3.2.1 Установка и настройка окружения 34](#_Toc186109983)

[3.2.2 Создание ролей и администратора 35](#_Toc186109984)

[3.2.3 Настройка базы данных 36](#_Toc186109985)

[3.2.4 Запуск сервера 36](#_Toc186109986)

[3.2.5 Использование системы 36](#_Toc186109987)

[3.2.6 Особенности сиситемы 36](#_Toc186109988)

[3.2.7 Частные ошибки и их устранение 37](#_Toc186109989)

[3.2.8 Завершение системы 37](#_Toc186109990)

[ЗАКЛЮЧЕНИЕ 39](#_Toc186109991)

[СПИСОК ЛИТЕРАТУРЫ 40](#_Toc186109992)

[ПРИЛОЖЕНИЕ 41](#_Toc186109993)

[Приложение A 41](#_Toc186109994)

# ВВЕДЕНИЕ

Объектом исследования является веб-приложение для бронирования номеров отеля, разработанное на фреймворке Django. Цель данного проекта – создание системы, которая проста в использовании, эффективна и гарантирует надёжность данных.

Задачи:

* Изучение профильных технологий Django, HTML, CSS и SQLite.
* Проектирование структуры и базы данных.
* Разработка компонентов системы, включая шаблоны и API.

Таким образом, данная работа направлена на решение актуальной задачи доступности автоматизированной системы бронирования номеров в гостиничном бизнесе. Внедрение разработанной системы позволит уменьшить ошибки, сократит нагрузку на персонал и повысит удовлетворённость клиентов.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Предметная область

Бронирование номеров является ключевой частью операций гостиничного бизнеса. Доступность автоматизированной системы способствует уменьшению ошибок, сокращает нагрузку на персонал и повышает удовлетворённость клиентов.

Основные задачи предметной области:

* Простое и удобное бронирование номеров с использованием веб-интерфейса.
* Обеспечение надёжности и безопасности данных.
* Создание модульной системы, которая легко дорабатывается.

Предметная область включает в себя процессы управления бронированием номеров отелей. Она охватывает следующие аспекты:

1. Бизнес-процессы:

* Управление номерами (добавление, удаление, обновление статуса).
* Бронирование (выбор номера, указание дат, создание записи).

2. Основные сущности:

* Гость: пользователь, который регистрируется в системе и делает бронирование.
* Номер: объект для размещения, характеризующийся типом, ценой и доступностью.
* Бронирование: запись, связывающая гостя и номер с указанием сроков пребывания.

## 1.2 Техническое задание

Цель проекта: Разработка системы управления бронированием номеров для отеля, которая обеспечивает автоматизацию процессов регистрации гостей, управления номерами и обработки бронирований.

Функциональные требования:

1. Регистрация и авторизация пользователей:

* Пользователи должны иметь возможность создать личный аккаунт с указанием базовой информации (имя, email, пароль).
* Поддержка ролевой модели доступа: гость и администратор.

2. Работа с номерами:

* Просмотр доступных номеров с фильтрацией по типу (одноместные, двухместные) и состоянию (свободен, забронирован).
* Управление номерами для администраторов (добавление, редактирование, удаление).

3. Процесс бронирования:

* Гости должны иметь возможность выбрать номер, указать даты заезда и выезда.
* Автоматическое обновление статуса номера при подтверждении бронирования.
* Уведомления об успешном бронировании.

4. Администрирование:

* Управление списком пользователей (блокировка, удаление).
* Генерация отчётов о занятости номеров за определённый период.

5. Удобство использования:

* Интуитивно понятный интерфейс с адаптацией для мобильных устройств.
* Быстрая загрузка страниц и отзывчивый дизайн.

Нефункциональные требования:

1. Производительность:

Обеспечение обработки до 100 одновременных пользователей без значительных задержек.

2. Надёжность:

Минимизация ошибок при бронировании через валидацию пользовательских данных.

3. Безопасность:

* Защита данных пользователей через шифрование паролей.
* Ограничение доступа к административным функциям только для уполномоченных лиц.

## 1.3 Технологии для разработки программы

### 1.3.1. Django Framework

Как и любой другой фреймворк, «Django» – это программная платформа (каркас), на который разработчик добавляет дополнительные компоненты приложения, например:

* интерфейс пользователя;
* скрипты (сценарии) работы элементов приложения/сайта;
* программные библиотеки;
* мультимедийный контент (фото, видео, аудио);
* модули безопасности;
* базы данных и т.д.

Фреймворк отвечает за базовые аспекты работы продукта. А подключаемые (сменные) модули – за специфичные функции конкретного приложения.

Использование фреймворков позволяет:

* ускорить разработку;
* упростить поддержку;
* избежать ошибок в работе приложения.

Модель – бизнес-логика, то есть совокупности методов, правил и ограничений работы с данными.

Представление – компонент, отображающий пользователю данные в зависимости от изменения модели.

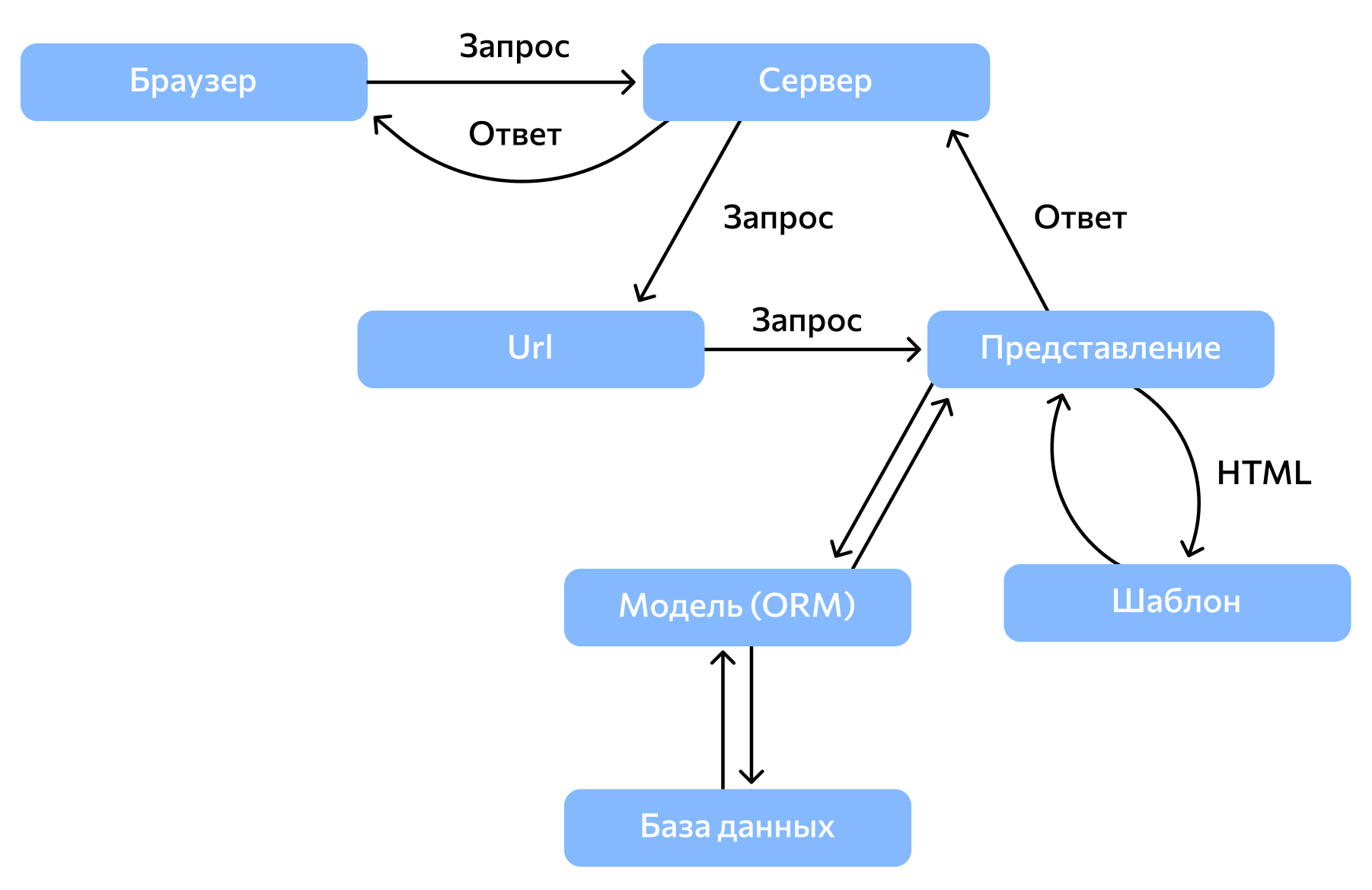
Контроллер – программный посредник, обрабатывающий действия пользователя и сообщающий модели, как она должна измениться.

Контроллер классической модели MVC примерно соответствует уровню, который в Django называется представлением (англ. View), а презентационная логика представления реализуется в Django уровнем шаблонов (англ. Template). Поэтому архитектуру Django часто называют «Модель–шаблон–представление» (MTV):

* модель по-прежнему отвечает за работу с данными (доступ, обработку, проверку и т.д.);
* шаблоны определяют, как будет отображаться информация;
* представление описывает, какие именно данные будут показываться пользователю.

Он был выбран по следующим причинам:

* Встроенная ORM: Django предоставляет удобные инструменты для работы с базой данных, упрощая создание, чтение, обновление и удаление данных.
* Безопасность: встроенная защита от SQL-инъекций, XSS-атак и CSRF-уязвимостей делает приложение надёжным.
* Масштабируемость: модульная структура Django позволяет легко добавлять новые функции и модули без значительных изменений существующего кода.

  
Рисунок 1.3 – Схема работы приложения на Django

### 1.3.2 SQLLite

SQLite – это быстрая и легкая встраиваемая однофайловая СУБД на языке C, которая не имеет сервера и позволяет хранить всю базу локально на одном устройстве. Для работы SQLite не нужны сторонние библиотеки или службы.

Большинство СУБД используют клиент-серверную архитектуру: данные хранятся и обрабатываются на сервере, а запросы к нему посылает клиент. «Клиент» – это часть программы, с которой взаимодействует пользователь. «Сервером» может быть и отдельный процесс на том же компьютере (так называемый демон), и стороннее устройство, как в случае с сайтами.

SQLite устроена иначе и не имеет сервера. Это значит, что все данные программное обеспечение хранит на одном устройстве. СУБД встраивается в приложение и работает как его составная часть. Если установить на компьютер программу, использующую SQLite, то база данных тоже будет храниться на нем же. Формат базы — один текстовый файл, который можно прочитать на любой платформе. Такой подход повышает производительность и скорость работы.

Преимущества SQLite:

1. Высокая скорость. Благодаря особенностям архитектуры SQLite работает быстро, особенно на чтение. Компоненты СУБД встроены в приложение и вызываются в том же процессе. Поэтому доступ к ним быстрее, чем при взаимодействии между разными процессами.

2. Хранение данных в одном файле. База данных состоит из табличных записей, связей между ними, индексов и других компонентов. В SQLite они хранятся в едином файле (database file), который находится на том же устройстве, что и программа. Чтобы при работе не возникало ошибок, файл блокируется для сторонних процессов перед записью. Раньше это приводило к тому, что записывать данные в базу мог только один процесс единовременно. Но в новых версиях это решается перенастройкой режима работы СУБД.

3. Минимализм. Создатели SQLite пользуются принципом «минимального полного набора». Из всех возможностей SQL в ней есть наиболее нужные. Поэтому SQLite отличают малый размер, простота решений и легкость администрирования. Для повышения базовой функциональности можно использовать стороннее программное обеспечение и расширения.

4. Надежность. Код на 100% покрыт тестами. Это означает, что протестирован каждый компонент ПО. Поэтому SQLite считается надежной СУБД с минимальным риском непредсказуемого поведения.

5. Нулевая конфигурация. Перед использованием СУБД не нужна сложная настройка или длительная установка. Для решения большинства задач ей можно пользоваться «из коробки», без установки дополнительных компонентов.

6. Малый размер. Полностью сконфигурированный SQLite со всеми настройками занимает меньше 400 Кб. Если использовать СУБД без дополнительных компонентов, размер можно уменьшить до 250 Кб. Он зависит только от количества загруженной информации. Несмотря на малый размер, SQLite поддерживает большинство функций стандарта SQL2 и имеет ряд собственных.

7. Доступность. SQLite находится в публичном доступе. На ее использование нет правовых ограничений, а владельцем считается общество. Можно открывать, просматривать и изменять исходный код установленного ПО.

8. Кроссплатформенность. СУБД подходит для UNIX-подобных систем, MacOS и Windows.

9. Автономность. Система независима от стороннего ПО, библиотек или фреймворков. Чтобы приложение с базой на SQLite работало, дополнительные компоненты не требуются. Также не обязателен доступ в интернет: вся база хранится на устройстве, получить данные можно локально.

SQLite – лёгкая реляционная база данных, которая идеально подходит для разработки и тестирования:

* Простота использования: не требует настройки сервера и сложных конфигураций.
* Надёжность: обеспечивает хранение данных в компактном формате.
* Интеграция с Django: Django поддерживает SQLite по умолчанию, что упрощает настройку.

### 1.3.3 HTML и CSS

HTML и CSS – это два основных инструмента, которые необходимы при работе с шаблонами сайта.

HTML – стандартизированный язык программирования документов. Большинство страниц содержат описание разметки на языке HTML. Иными словами, HTML определяет структуру содержимого страницы. Например, именно в HTML задаются заголовки и абзацы, изображения и т.д.

CSS – язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, к вышеупомянутым документам HTML). Проще говоря, основное назначение CSS – описывать оформление внешнего вида контента.

Таким образом, если HTML нужен для того, чтобы описать, какая именно информация и в каком порядке должна выводиться на странице, то CSS расширяет возможность HTML и позволяет менять цвета, шрифты, фон и т.д.

HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) используются для создания пользовательского интерфейса:

* HTML: отвечает за структуру страниц (заголовки, списки, формы).
* CSS: обеспечивает стиль и оформление интерфейса, включая адаптивный дизайн.

### 1.3.4 Bootstrap

Bootstrap – это открытый и бесплатный HTML-, CSS- и JS-фреймворк, который используют веб-разработчики для быстрой верстки адаптивных дизайнов сайтов и веб-приложений. Включает в себя CSS- и HTML-шаблоны оформления для веб-форм, меток, типографики, кнопок, блоков навигации и других компонентов веб-интерфейса.

Bootstrap используется, когда:

* у сайта много страниц;
* страницы собраны из простых базовых элементов – кнопок или таблиц;
* не будет глобального редизайна;
* шаблонность страниц окупается скоростью внедрения.

Из чего состоит Bootstrap

По сути, Bootstrap – это набор файлов. После их подключения к странице для верстки станет доступно большое количество готовых компонентов и классов. Они позволяют быстро и качественно создавать адаптивный дизайн сайта.

Классы в Bootstrap делятся на 3 большие группы:

* для создания сетки – адаптивного макета страницы;
* для стилизации контента – текста, изображений, кода, таблиц и прочей информации;
* служебные – для решения популярных вспомогательных задач, таких как отображение и скрытие элементов, выравнивание текста на странице, настройка цвета фона, отступов и пр.

Кроме классов, в Bootstrap есть компоненты (готовые объекты интерфейса). Это хлебные крошки (путь от начального элемента до уровня иерархии, который сейчас просматривает пользователь), кнопки, выпадающие списки и подсказки, модальные окна, формы, навигационные меню, всплывающие панели и пр.

Особенности Bootstrap:

1.Снижение количества времени на разработку

Фреймворк дает готовые решения, которые позволяют создавать макеты сайтов быстрее. Для быстрого запуска проектов есть множество примеров. Для верстки любых прототипов (альбомов, слайдеров, панели входа и т. д.) потребуются файлы index.html, которые находятся в каждой папке с примерами, и соответствующий CSS-файл. Например, carousel.css или cover.css. Изменять код можно по своему усмотрению, внося правки в текстовом редакторе. Bootstrap позволяет скопировать код из примера и вставить его в свой проект, к которому подключен фреймворк.

2. Адаптивность и кроссбраузерность

Сайт будет корректно отображаться в современных браузерах и на экранах устройств различных размеров, независимо от диагонали. Для этого не нужно вносить изменения в разметку. Вероятность непредвиденных ошибок в функционале и верстке сведена к минимуму.

3. Легкость в использовании и открытость

Bootstrap очень простой для освоения и работы. Кроме того, к фреймворку есть множество уроков и инструкций. Открытый исходный код позволяет адаптировать Bootstrap под свои потребности.

4. Понятный код

С помощью Bootstrap можно писать простой и качественный код, который будет понятен другим разработчикам. Это облегчает работу в команде.

5. Единство стилей

Элементы фреймворка гармонично сочетаются друг с другом, что позволяет создавать сайты и страницы в едином стиле.

6. Шаблонность

Сайты, созданные с помощью Bootstrap, имеют одинаковую навигацию, структуру, кнопки. Чтобы решить проблему, можно изменять шаблон в зависимости от идей дизайнеров и пожеланий заказчика.

7. Отсутствие поддержки устаревших версий браузеров

Так как Bootstrap постоянно обновляется, сайты на Bootstrap могут некорректно отображаться в старых браузерах.

Его выбор обусловлен:

* Готовыми компонентами: кнопки, формы и навигационные элементы упрощают разработку.
* Адаптивностью: страницы автоматически подстраиваются под размеры экрана.
* Совместимостью: лёгкая интеграция с HTML-шаблонами Django.

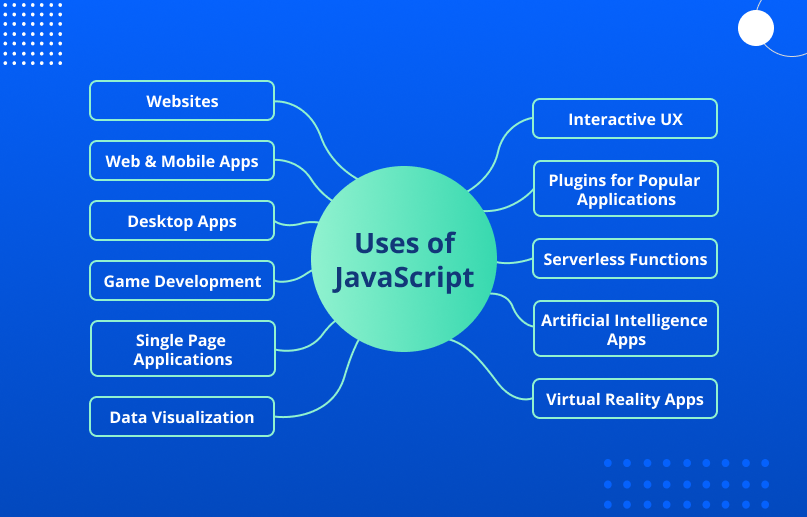
### 1.3.5 JavaScript

JavaScript сегодня используется более чем в 97% веб-сайтов, что делает его более чем основным. Он оживляет все: от маленьких блогов до огромных корпоративных сайтов. Этот язык является ключевым для добавления динамического контента и функций, которые нравятся пользователям. Это один из самых универсальных навыков в сфере технологий.

Он имеет решающее значение для создания хорошо работающих веб-приложений. С его помощью можно добавлять на сайты интересные интерактивные функции, улучшая взаимодействие пользователей с Интернетом.

JavaScript играет ключевую роль в современной веб-разработке. Он позволяет сделать веб-страницы интерактивными и интересными. Это включает в себя создание динамических веб-платформ с использованием HTML, CSS и JavaScript. Привлекательная природа этих технологий помогает привлекать и удерживать пользователей.

* Валидация данных на клиентской стороне: проверка корректности заполнения форм.
* Динамические элементы: обновление данных без перезагрузки страницы.

  
Рисунок 1.3 – Использование JavaScript

### 1.3.6 Git

Git – это специальная программа, которая позволяет отслеживать любые изменения в файлах, хранить их версии и оперативно возвращаться в любое сохранённое состояние.

Большинство других систем контроля версий хранят информацию в виде списка изменений в файлах. Git работает иначе – он хранит скорее набор снимков – полное отображение того, как выглядит файл в момент сохранения. Это позволяет всегда иметь полную информацию обо всех файлах и быстро восстанавливать любую из предыдущих версий.

Принцип работы, в проекте создаем окружение Git – все изменения файлов отслеживаются в рамках настроенных параметров и заданных фильтров. Нужно добавить в проект новую функцию, изменив или доработав существующий код. Для этого потребуется создать внутри проекта отдельную ветку – в Git они называются branch. Работа в этой ветке никак не затрагивает основной код – если с новыми изменениями что-то пойдёт не так и код станет невалидным и перестанет запускаться, основной проект не пострадает. А когда новая функция будет дописана и протестирована, ветку можно будет «наложить» на основной код.

Также в рамках Git можно объединять разные версии кода в один. Например, над проектом трудится несколько программистов, и каждый разрабатывает или изменяет код в собственных ветках. В конце работы появится необходимость слить ветки вместе – и получается цельная программа. Это значительно облегчает совместную работу, так как не нужно ждать, пока другой разработчик допишет код, – можно работать параллельно.

Если же в одной из веток разработка пойдёт не по плану и произойдёт ошибка – всё можно просто откатить до предыдущей ветки в системе контроля версий Git, где ошибок не было. И начать разработку заново.

Данные технологии были выбраны, потому как обеспечиваю:

* Универсальность Django: этот фреймворк покрывает большинство потребностей для создания веб-приложений.
* Простота SQLite: идеальна для начального этапа разработки.
* Гибкость Bootstrap: ускоряет создание адаптивного дизайна.
* Широкая поддержка: все выбранные технологии имеют обширное сообщество, что упрощает поиск решений и документации.

  
Рисунок 1.3 – Используемые инструменты

Выводы

Проведён анализ предметной области, в рамках которой автоматизация процессов бронирования была признана важным элементом для повышения эффективности гостиничного бизнеса. Определены функциональные и нефункциональные требования к системе, что позволило задать чёткие рамки для разработки. Описаны выбранные технологии (Django, SQLite, HTML, CSS, Bootstrap), которые обеспечивают гибкость, безопасность и удобство разработки.

Теоретическая часть заложила основу для успешного проектирования и реализации системы, обозначив требования и обосновав выбор технологий.

# 2 ПРОЕКТНАЯ ЧАСТЬ

## 2.1 Описание архитектуры системы

Система использует клиент-серверную архитектуру. Серверная часть на Django обрабатывает запросы пользователей, взаимодействует с базой данных SQLite и возвращает HTML-страницы. Клиентская часть отвечает за отображение данных и обработку действий пользователей.

Ниже приведена схема архитектуры, иллюстрирующая взаимодействие между компонентами:

* Браузер отправляет HTTP-запросы к серверу.
* Сервер обрабатывает запросы с помощью URL Dispatcher, который направляет запросы к соответствующим представлениям.
* Представления взаимодействуют с моделями для получения данных из базы данных.
* Модели выполняют запросы к базе данных и возвращают данные представлениям.
* Представления используют шаблоны для генерации HTML-страниц с данными.
* HTML-страницы отправляются обратно в браузер, который отображает их пользователю.

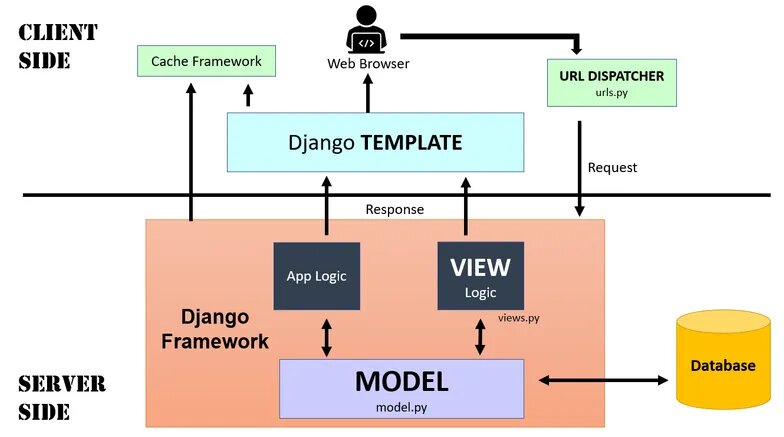


Рисунок 2.1 – Архитектура клиент-сервер

Эта архитектура обеспечивает четкое разделение задач, упрощает разработку и поддержку кода, а также улучшает масштабируемость и производительность веб-приложения.

## 2.2 Алгоритм создания и работы приложения

Алгоритм работы программы обеспечивает корректное взаимодействие пользователя с системой бронирования и управление процессами. Основные этапы:

1. Регистрация пользователя:

* Пользователь вводит данные (имя, email, пароль).
* Система проверяет корректность введённых данных и создаёт новую запись в базе данных.

2. Авторизация:

* Пользователь вводит email и пароль.
* Проверяется соответствие данных, и пользователь получает доступ к системе в зависимости от своей роли (гость или администратор).

3. Просмотр доступных номеров:

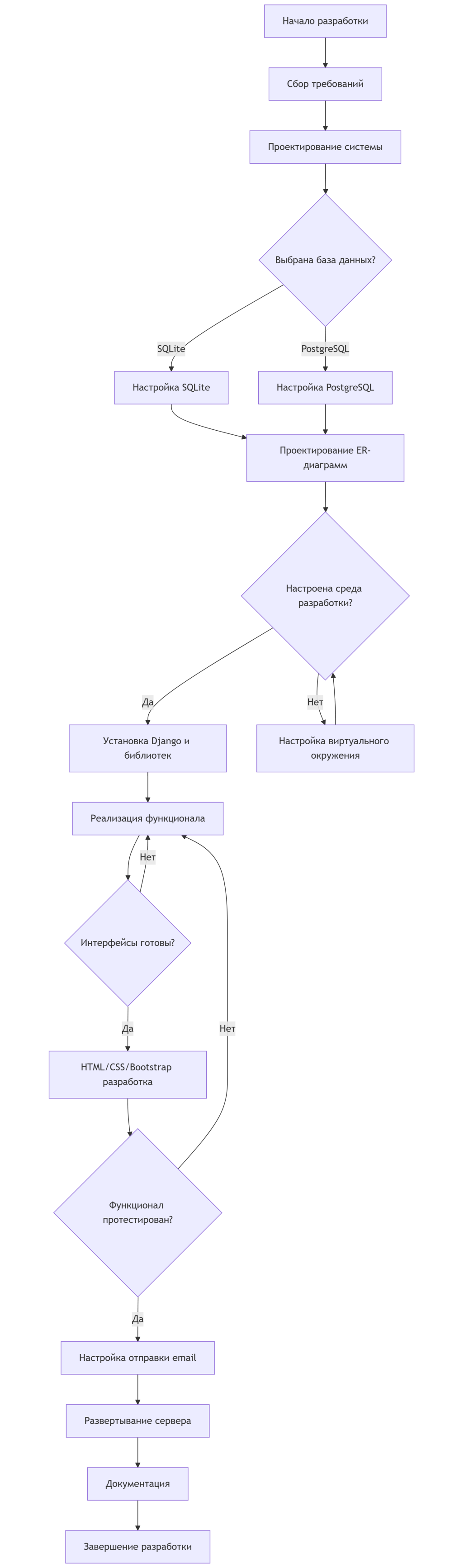
* Гость выбирает дату заезда и выезда.
* Система фильтрует номера по их доступности и возвращает результаты.

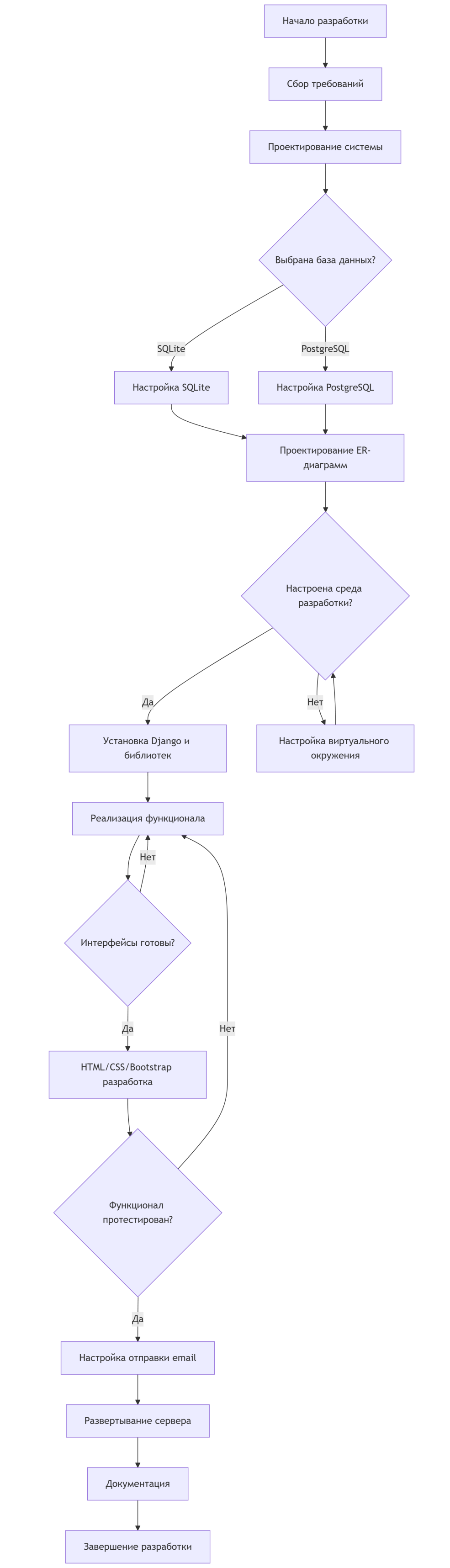
4. Процесс бронирования:

* Пользователь выбирает номер и заполняет форму бронирования.
* После подтверждения данные сохраняются в таблице Bookings, а статус номера обновляется на "занят".

5. Управление администрацией:

* Администратор имеет доступ к функциям добавления, редактирования и удаления номеров.
* Доступен просмотр отчётов о занятости номеров.

  
Рисунок 2.2 – Блок-схема создания приложения

  
Рисунок 2.2 – Блок-схема создания приложения

2.3 Стандарт разработки

Для разработки системы бронирования соблюдались следующие стандарты:

1. Кодировка:

Все файлы сохраняются в кодировке UTF-8 для обеспечения совместимости.

2. Организация проекта Django:

Принята стандартная структура проекта Django:

* Каталог apps для приложений.
* Каталог templates для HTML-шаблонов.
* Каталог static для CSS, JavaScript и других статических файлов.

3. Соблюдение PEP 8:

Код написан с соблюдением стандартов Python (PEP 8) для повышения читаемости и удобства сопровождения.

4. Система контроля версий:

* Git используется для управления версиями и отслеживания изменений.
* Основной репозиторий хранится на GitHub с подробными комментариями к коммитам.

5. Безопасность:

* Используются встроенные механизмы Django для защиты от CSRF и XSS атак.
* Пароли пользователей хранятся в хэшированном виде с использованием PBKDF2.

## 2.4 Структура базы данных

База данных включает три основные таблицы:

1. Users:

* id (Primary Key): уникальный идентификатор пользователя.
* name: имя пользователя.
* email: адрес электронной почты.
* role: роль пользователя (гость или администратор).

2. Rooms:

* id (Primary Key): уникальный идентификатор номера.
* room\_number: номер комнаты.
* room\_type: тип комнаты (одноместный, двухместный).
* is\_available: статус доступности (True/False).

3. Bookings:

* id (Primary Key): уникальный идентификатор бронирования.
* user\_id (Foreign Key): ссылка на таблицу пользователей.
* room\_id (Foreign Key): ссылка на таблицу номеров.
* check\_in: дата заезда.
* check\_out: дата выезда. азы данных

## 2.5 Макет системы

Сформированы макеты страниц: форма регистрации, список номеров, страница создания бронирования.

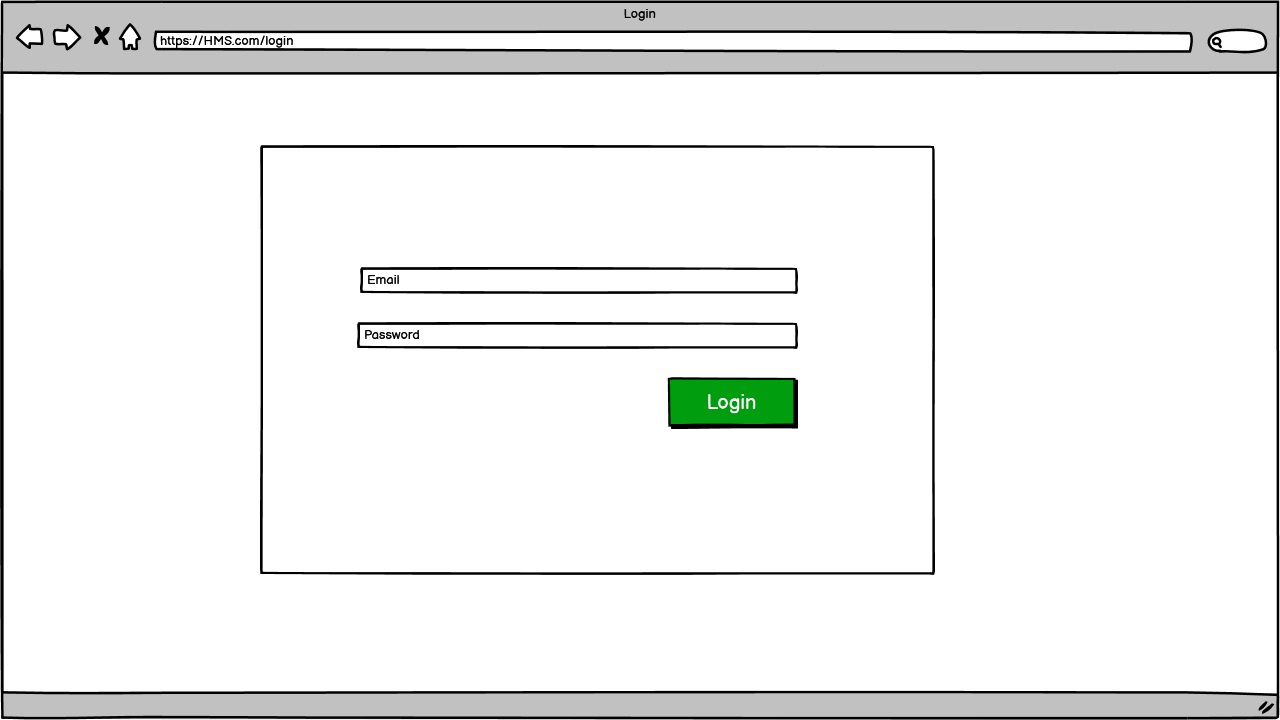
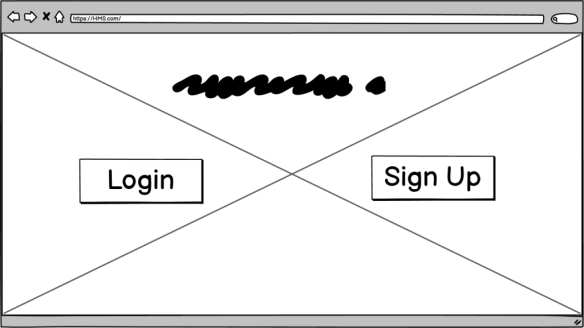
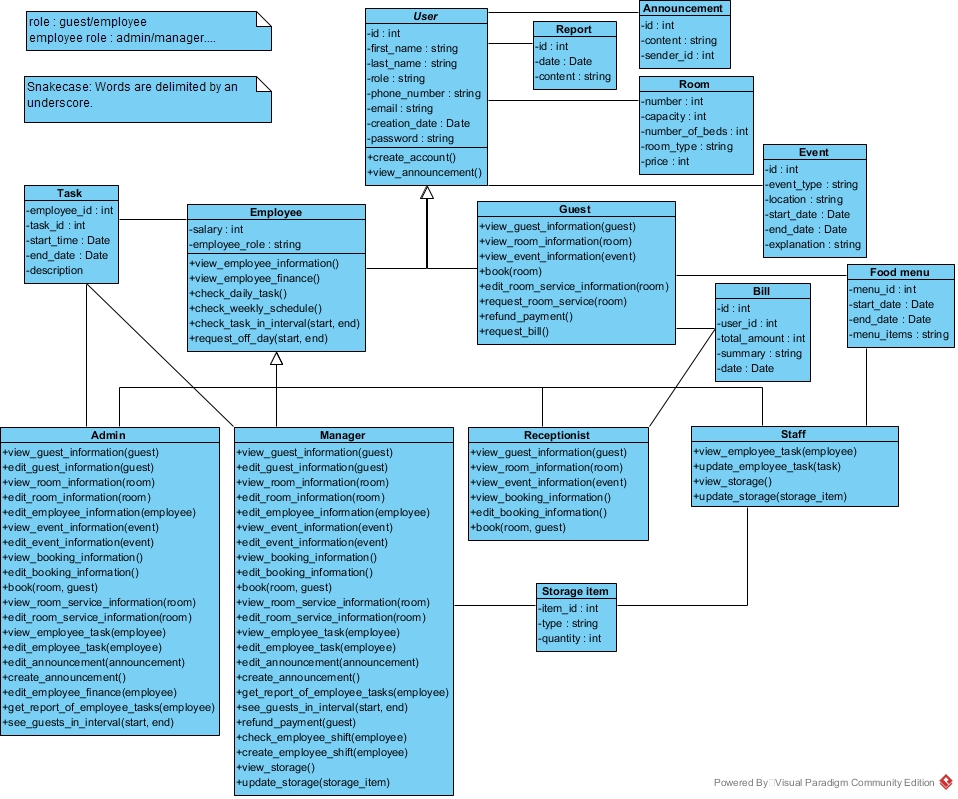
  
Рисунок 3.2.2 – Прототип страницы «Вход в аккаунт»

  
Рисунок 3.2.2 – Прототип интерфейса «Создание аккаунта» (для клиентов)

  
Рисунок 3.2.2 – Прототип страницы «Регистрация»

Интерфейс построен на HTML, CSS и Bootstrap, что обеспечивает адаптивность и удобство использования.

## 2.6 Диаграммы

Рисунок 2.6 – Диаграмма классов

Создана ER-диаграмма, включающая сущности: Users, Rooms, Bookings, на диаграмме представлены и другие сущности, которые будут реализованы в дипломной работе.

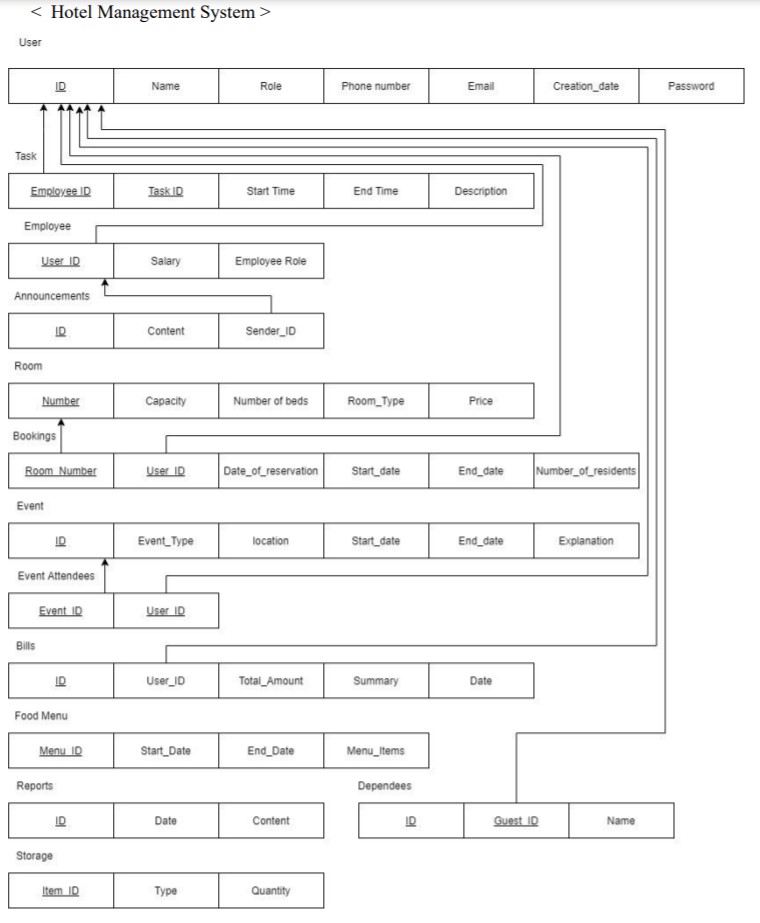


Рисунок 2.6 – ER-диаграмма



Рисунок 2.5 – Диаграмма последовательности

Вывод

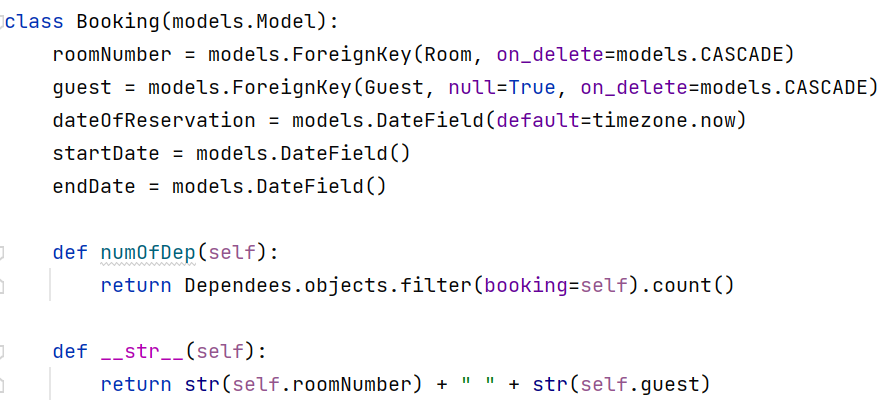
Разработан алгоритм работы системы, охватывающий ключевые операции: регистрацию, бронирование, управление номерами и генерацию отчётов. Описана структура базы данных, включая таблицы Users, Rooms, Bookings, которые обеспечивают логические связи и оптимизированное хранение данных. Интерфейс системы спроектирован с учётом интуитивности и адаптивности, что делает систему удобной как для пользователей, так и для администраторов.

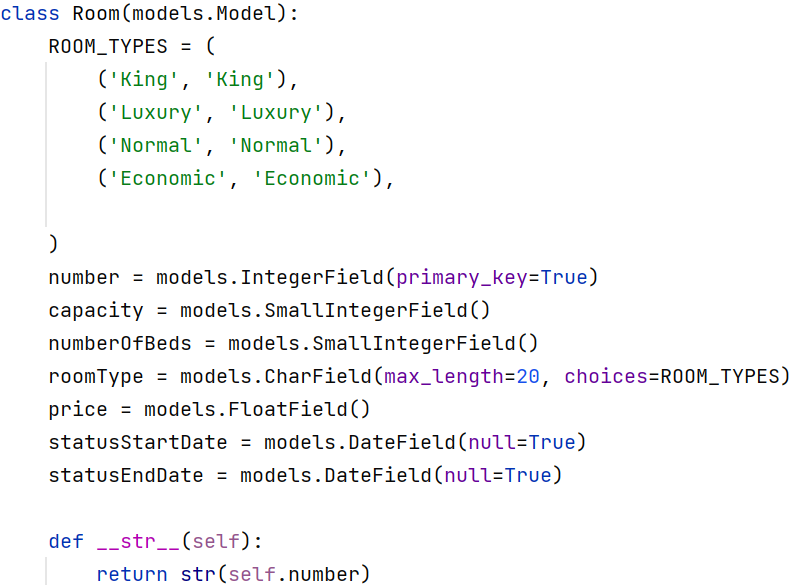
Проектная часть обеспечила создание эффективной структуры системы, логической базы данных и удобного интерфейса, что стало основой для практической реализации

# 3 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 3.1 Реализация системы

### 3.1.1 Модели

  
Рисунок 4.2.1 – Модель Booking

  
Рисунок 4.2.1 – Модель Room

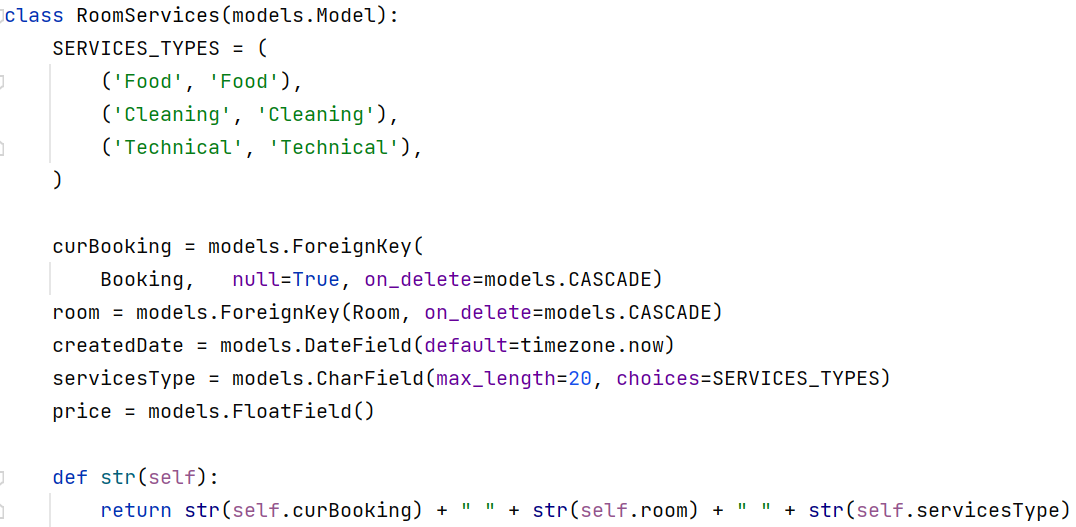


Рисунок 4.2.1 – Модель RoomServices

### 3.1.2 Шаблоны

С использованием Django Templates разработаны страницы для отображения доступных номеров, создания бронирований и управления ими.

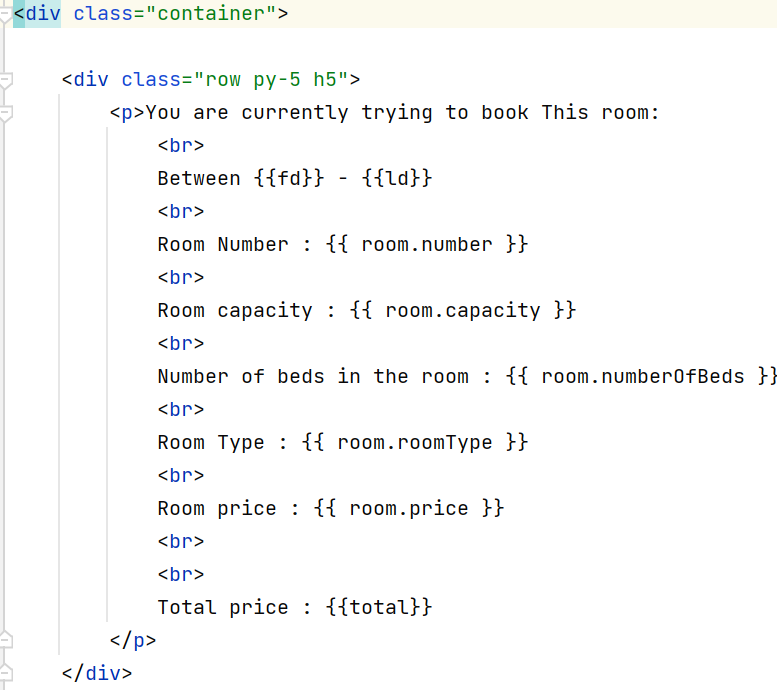
  
Рисунок 4.2.2 – booking-make.html



Рисунок 4.2.2 – guest-profile.html

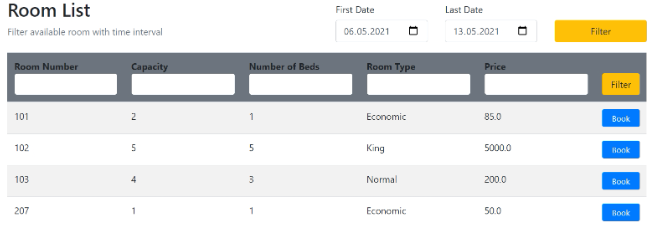
  
Рисунок 4.2.2 – guest-edit.html

  
Рисунок 4.2.2 – room-edit.html



Рисунок 4.2.2 – room-profile.html

### 3.1.3 Интерфейс системы

  
Рисунок 4.2.1 – Список комнат

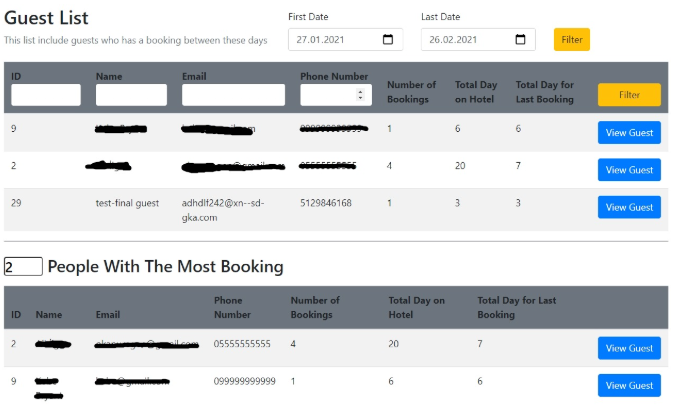
  
Рисунок 4.2.1 – Список гостей



Рисунок 4.2.1 – Форма входа

## 3.2 Руководство пользователя приложения

### 3.2.1 Установка и настройка окружения

Для работы системы бронирования требуется предварительная установка необходимых компонентов и настройка окружения. Выполните следующие шаги:

1. Убедитесь, что установлен Python 3.

Проверьте, установлена ли нужная версия, командой:

python3 --version

2. Установите Django и дополнительные зависимости.

Откройте терминал и выполните команды:

pip install Django==3.1.4

pip install django-phonenumber-field[phonenumbers]

3. Скачайте проект с GitHub.

Клонируйте репозиторий с проектом:

git clone https://github.com/VikaKon/Django-Hotel-System.git

Перейдите в папку проекта:

cd Django-Hotel-System /HMS

### 3.2.2 Создание ролей и администратора

Для корректной работы системы необходимо настроить роли пользователей и создать администратора:

1. Запустите Django оболочку:

python3 manage.py shell

2. Создайте роли пользователей и администратора:

Выполните команды по одной:

from django.contrib.auth.models import Group, User

from accounts.models import Employee

# Создание ролей

Group.objects.create(name='admin')

Group.objects.create(name='manager')

Group.objects.create(name='receptionist')

Group.objects.create(name='staff')

Group.objects.create(name='guest')

# Создание администратора

user = User.objects.create\_user('admin', password='admin123')

group = Group.objects.get(name='admin')

user.groups.add(group)

admin = Employee(user=user, salary=0)

admin.save()

3. Закройте оболочку:

exit()

### 3.2.3 Настройка базы данных

Выполните миграции для создания таблиц в базе данных:

python3 manage.py makemigrations

python3 manage.py migrate

Проверьте корректность работы базы данных, запустив приложение.

### 3.2.4 Запуск сервера

Для запуска сервера выполните:

python3 manage.py runserver

Откройте браузер и перейдите по адресу:

http://127.0.0.1:8000

### 3.2.5 Использование системы

1. Для гостей:

* Зарегистрируйтесь через форму регистрации, указав email и пароль.
* Авторизуйтесь и выберите доступный номер из списка.
* Укажите дату заезда и выезда, чтобы оформить бронирование.

2. Для администратора:

* Войдите в систему с логином admin и паролем admin123.
* Управляйте номерами через административную панель:
* Добавляйте новые номера.
* Редактируйте существующие.
* Удаляйте устаревшие номера.
* Просматривайте список бронирований и отчёты о занятости номеров.

### 3.2.6 Особенности сиситемы

1. Безопасность:

* Пароли пользователей шифруются и хранятся в безопасной форме.
* Административные функции доступны только пользователям с соответствующими правами.

2. Удобство использования:

* Интерфейс адаптирован для работы как на ПК, так и на мобильных устройствах.
* Простая и понятная структура навигации.

### 3.2.7 Частные ошибки и их устранение

1. Ошибка: Не найден модуль Django.

Проверьте, установлены ли зависимости, командой:

pip list

Если Django отсутствует, повторите установку:

pip install Django==3.1.4

2. Ошибка: Сервер не запускается.

Проверьте корректность выполненных миграций:

python3 manage.py showmigrations

Выполните миграции повторно, если необходимо.

3. Ошибка доступа:

Убедитесь, что пользователь с нужной ролью создан и добавлен в соответствующую группу.

### 3.2.8 Завершение системы

После окончания работы с системой остановите сервер, нажав Ctrl+C в терминале.

Для внесения изменений или обновления проекта используйте Git.

Вывод

Реализована функциональная часть системы, включающая модели, маршруты, представления и шаблоны.

Проведены тестирования системы: функциональное и нагрузочное. Выявленные ошибки были оперативно исправлены, что улучшило стабильность работы приложения.

Разработано руководство пользователя, которое детализирует процесс взаимодействия с системой для обеих ролей.

Практическая часть подтвердила работоспособность системы и её соответствие поставленным требованиям, предоставив готовый инструмент для гостиничного бизнеса.

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была разработана система управления бронированием номеров для отеля на основе фреймворка Django. Система включает функционал регистрации и авторизации пользователей, бронирования номеров, управления данными номеров, а также предоставляет удобный интерфейс для взаимодействия. Решение адаптировано для малых и средних гостиниц, где требуется экономичное и функциональное ПО.

Использование Django позволило эффективно реализовать ролевой доступ и надёжное хранение данных. База данных структурирована с учётом минимизации дублирования информации и обеспечения логических связей между данными. Разработанный интерфейс обеспечивает простоту и удобство использования как для гостей, так и для администраторов системы.

# СПИСОК ЛИТЕРАТУРЫ

1. Django documentation // Django URL: https://docs.djangoproject.com/en/5.1/ (дата обращения: 04.11.2024).
2. Основы HTML // developer.mozilla.org URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/HTML\_basics (дата обращения: 9.011.2024).
3. Архитектура и структура сайта на фреймворке DJANGO // studbooks.netURL:https://studbooks.net/2015118/informatika/arhitektura\_struktura\_sayta\_freymvorke\_django (дата обращения: 12.11.2024).
4. Использование моделей // djangodoc.ru URL: https://djangodoc.ru/3.2/topics/db/models/ (дата обращения: 11.11.2024).
5. Хант, Э., Томас, Д. Пограммирование на Python. Введение. / Хант, Э., Томас, Д . — Москва: Диалектика, 2019 — 260 c.
6. Документация Django / [Электронный ресурс] // : [сайт]. — URL: https://docs.djangoproject.com/en/stable/ (дата обращения: 31.11.2024).
7. Байтков, А. М., Громов, В. В. Основы веб-разработки: HTML, CSS, JavaScript / Москва: Издательство Питер, 2020 — 180 c.
8. Почему Django — лучший фреймворк для разработки сайтов // ru.hexlet.io URL: https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-sayto (дата обращения: 16.11.2024).
9. Особенности и плюсы использования Django в web'е // Хабр URL: https://habr.com/ru/sandbox/156526/ (дата обращения: 20.11.2024).
10. Четыре шага для запуска системы бронирования // Хабр URL: https://habr.com/ru/articles/842888/ (дата обращения: 01.10.2024).

# ПРИЛОЖЕНИЕ

## Приложение A

Листинг A1. urls.py

from django.contrib import admin  
from django.urls import path  
from accounts.views import \*  
from room.views import \*  
from hotel.views import \*  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', home, name="home"),  
  
 path('login/', login\_page, name="login"),  
 path('logout/', logout\_user, name="logout"),  
 path('register/', register\_page, name="register"),  
  
 path('guests/', guests, name="guests"),  
 path('employees/', employees, name="employees"),  
 path('events/', events, name="events"),  
 path('bookings/', bookings, name="bookings"),  
 path('rooms/', rooms, name="rooms"),  
 path('room-services/', room\_services, name="room-services"),  
 path('announcements/', announcements, name="announcements"),  
 path('refunds/', refunds, name="refunds"),  
 path('storage/', storage, name="storage"),  
  
 path('tasks/', tasks, name="tasks"),  
 path('current-room-services/', current\_room\_services,  
 name="current-room-services"),  
 path('request-refund/', request\_refund, name="request-refund"),  
 path('event-profile/<str:id>/', event\_profile, name="event-profile"),  
 path('event-edit/<str:pk>/', event\_edit, name="event-edit"),  
 path('add-room/', add\_room, name="add-room"),  
  
 path('employee-profile/<str:pk>/', employee\_details, name="employee-profile"),  
 path('employee-edit/<str:pk>/', employee\_details\_edit, name="employee-edit"),  
 path('employee-add/', add\_employee, name="add-employee"),  
  
 path('guest-edit/<str:pk>', guest\_edit, name="guest-edit"),  
 path('guest-profile/<str:pk>', guest\_profile, name="guest-profile"),  
 path('room-profile/<str:id>/', room\_profile, name="room-profile"),  
 path('room-edit/<str:pk>/', room\_edit, name="room-edit"),  
 path('error/', error, name="error"),  
  
 path('booking-make/', booking\_make, name="booking-make"),  
 path('payment/', payment, name="payment"),  
 path('verify/', verify, name="verify"),  
  
 path('deleteStorage/<str:pk>/', deleteStorage, name="deleteStorage"),  
 path('deleteFoodMenu/<str:pk>/', deleteFoodMenu, name="deleteFoodMenu"),  
 path('food-menu/', food\_menu, name="food-menu"),  
 path('food-menu/<str:pk>/', food\_menu\_edit, name="food-menu-edit"),  
  
 path('createEvent/', createEvent, name="createEvent"),  
 path('deleteEvent/<str:pk>/', deleteEvent, name="deleteEvent"),  
 path('deleteAnnouncement/<str:pk>/',  
 deleteAnnouncement, name="deleteAnnouncement"),  
 path('deleteBooking/<str:pk>/', deleteBooking, name="deleteBooking"),  
 path('completeTask/<str:pk>/', completeTask, name="completeTask"),  
]

Листинг А2. Account/models.py

from django.db import models  
from phonenumber\_field.modelfields import PhoneNumberField  
from django.contrib.auth.models import User  
class Guest(models.Model):  
 user = models.OneToOneField(User, null=True, on\_delete=models.CASCADE)  
 phoneNumber = PhoneNumberField(unique=True)  
  
 def \_\_str\_\_(self):  
 return str(self.user)  
  
 def numOfBooking(self):  
 return Booking.objects.filter(guest=self).count()  
  
 def numOfDays(self):  
 totalDay = 0  
 bookings = Booking.objects.filter(guest=self)  
 for b in bookings:  
 day = b.endDate - b.startDate  
 totalDay += int(day.days)  
  
 return totalDay  
  
 def numOfLastBookingDays(self):  
 try:  
 return int((Booking.objects.filter(guest=self).last().endDate - Booking.objects.filter(guest=self).last().startDate).days)  
 except:  
 return 0  
  
 def currentRoom(self):  
 booking = Booking.objects.filter(guest=self).last()  
 return booking.roomNumber  
  
class Employee(models.Model):  
 user = models.OneToOneField(User, null=True, on\_delete=models.CASCADE)  
 phoneNumber = PhoneNumberField(unique=True)  
 salary = models.FloatField()  
  
 def \_\_str\_\_(self):  
 return str(self.user)  
class Task(models.Model):  
 employee = models.ForeignKey(  
 Employee, null=True, on\_delete=models.CASCADE)  
 startTime = models.DateTimeField()  
 endTime = models.DateTimeField()  
 description = models.TextField()  
  
 def str(self):  
 return str(self.employee)

Листинг А3. Rooms/models.py

from django.db import models  
from django.utils import timezone  
from accounts.models import Guest  
class Room(models.Model):  
 ROOM\_TYPES = (  
 ('King', 'King'),  
 ('Luxury', 'Luxury'),  
 ('Normal', 'Normal'),  
 ('Economic', 'Economic'),  
  
 )  
 number = models.IntegerField(primary\_key=True)  
 capacity = models.SmallIntegerField()  
 numberOfBeds = models.SmallIntegerField()  
 roomType = models.CharField(max\_length=20, choices=ROOM\_TYPES)  
 price = models.FloatField()  
 statusStartDate = models.DateField(null=True)  
 statusEndDate = models.DateField(null=True)  
  
 def \_\_str\_\_(self):  
 return str(self.number)  
  
  
class Booking(models.Model):  
 roomNumber = models.ForeignKey(Room, on\_delete=models.CASCADE)  
 guest = models.ForeignKey(Guest, null=True, on\_delete=models.CASCADE)  
 dateOfReservation = models.DateField(default=timezone.now)  
 startDate = models.DateField()  
 endDate = models.DateField()  
  
 def numOfDep(self):  
 return Dependees.objects.filter(booking=self).count()  
  
 def \_\_str\_\_(self):  
 return str(self.roomNumber) + " " + str(self.guest)  
  
  
class Dependees(models.Model):  
 booking = models.ForeignKey(Booking, null=True, on\_delete=models.CASCADE)  
 name = models.CharField(max\_length=100)  
  
 def str(self):  
 return str(self.booking) + " " + str(self.name)  
  
  
class Refund(models.Model):  
 guest = models.ForeignKey(Guest, null=True, on\_delete=models.CASCADE)  
 reservation = models.ForeignKey(Booking, on\_delete=models.CASCADE)  
 reason = models.TextField()  
  
 def \_\_str\_\_(self):  
 return str(self.guest)  
  
  
class RoomServices(models.Model):  
 SERVICES\_TYPES = (  
 ('Food', 'Food'),  
 ('Cleaning', 'Cleaning'),  
 ('Technical', 'Technical'),  
 )  
  
 curBooking = models.ForeignKey(  
 Booking, null=True, on\_delete=models.CASCADE)  
 room = models.ForeignKey(Room, on\_delete=models.CASCADE)  
 createdDate = models.DateField(default=timezone.now)  
 servicesType = models.CharField(max\_length=20, choices=SERVICES\_TYPES)  
 price = models.FloatField()  
  
 def str(self):  
 return str(self.curBooking) + " " + str(self.room) + " " + str(self.servicesType)

Листинг А4. Hotel/models.py

from django.db import models  
from django.utils import timezone  
from django.contrib.auth.models import User  
from accounts.models import Guest, Employee  
class Announcement(models.Model):  
 content = models.TextField()  
 sender = models.ForeignKey(Employee, null=True, on\_delete=models.CASCADE)  
 date = models.DateField(default=timezone.now)  
  
 def \_\_str\_\_(self):  
 return str(self.sender)  
  
  
class Event(models.Model):  
 EVENT\_TYPES = (  
 ('Movie', 'Movie'),  
 ('Theater', 'Theater'),  
 ('Conference', 'Conference'),  
 ('Concert', 'Concert'),  
 ('Entertainment', 'Entertainment'),  
 ('Live Music', 'Live Music'),  
 )  
  
 eventType = models.CharField(max\_length=20, choices=EVENT\_TYPES)  
 location = models.CharField(max\_length=100)  
 startDate = models.DateField()  
 endDate = models.DateField()  
 explanation = models.TextField()  
  
 def \_\_str\_\_(self):  
 return str(self.eventType)  
  
  
class EventAttendees(models.Model):  
 event = models.ForeignKey(Event, on\_delete=models.CASCADE)  
 numberOfDependees = models.IntegerField(default=0)  
 guest = models.ForeignKey(Guest, null=True, on\_delete=models.CASCADE)  
  
 def \_\_str\_\_(self):  
 return str(self.event) + " " + str(self.guest)  
  
  
class Bills(models.Model):  
 guest = models.ForeignKey(Guest, null=True, on\_delete=models.CASCADE)  
 totalAmount = models.FloatField()  
 summary = models.TextField()  
 date = models.DateTimeField(default=timezone.now)  
 def \_\_str\_\_(self):  
 return str(self.guest) + " " + str(self.summary) + " " + str(self.totalAmount)  
  
  
class FoodMenu(models.Model):  
 startDate = models.DateField()  
 endDate = models.DateField()  
 menuItems = models.TextField()  
  
 def \_\_str\_\_(self):  
 return str(self.menuItems) + " " + str(self.startDate)  
  
  
class Report(models.Model):  
 date = models.DateField(default=timezone.now)  
 content = models.TextField()  
  
 def \_\_str\_\_(self):  
 return str(self.content) + " " + str(self.date)  
  
  
class Storage(models.Model):  
 ITEM\_TYPES = (  
 ('Kitchen', 'kitchen'),  
 ('Cleaning', 'cleaning'),  
 ('Electronic', 'Electronic'),  
 ('Textile ', 'textile '),  
 ('Other', 'other'),  
 )  
 itemName = models.CharField(max\_length=100)  
 itemType = models.CharField(max\_length=20, choices=ITEM\_TYPES)  
 quantitiy = models.IntegerField()  
  
 def \_\_str\_\_(self):  
 return str(self.itemName)