**Министерство просвещения Республики Казахстан**

**Колледж TOO «Astana IT University»**

**Отчет**

**по производственной практике**

|  |  |  |
| --- | --- | --- |
| Студент\_\_\_\_\_\_\_\_\_\_Оралбай Диас\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа ВТ2412  /ф.и.о./ /подпись/ | | |
| Руководитель практики от  колледжа | Преподаватель спец. дисциплин  /должность/ | \_Баумуратова Ш.Б.\_\_  /ф.и.о./ |
| Допущен к защите | «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_\_г. |  |
| Защищена | «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_\_г. | с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /письменно/ |
| Члены комиссии: | \_Бердибеков З.Б.\_\_\_\_  /ф.и.о./ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /подпись./ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  /ф.и.о./ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /подпись./ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  /ф.и.о./ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /подпись./ |

**г. Астана, 2025г.**

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
|  | **Введение**…………………………………………………………….. | 3 |
| 1 | **Теоретические основы**…………………………………………….. | 5 |
| 1.1 | Понятие информационной системы и ее роль в автоматизации ресторанного бизнеса……………………………………………….. | 5 |
| 1.2 | Обзор существующих инструментов для создания систем управления заказами в ресторанном бизнесе……………………... | 7 |
| 1.3 | Архитектура веб-приложений для системы управления заказами в ресторане…………………………………………………………... | 11 |
| 2 | **Практическая реализация**………………..………………………. | 15 |
| 2.1 | Создание и настройка базы данных для системы управления заказами в ресторане……………………...………………………… | 15 |
| 2.2 | Разработка пользовательского интерфейса………………………... | 16 |
|  | **Заключение**…………………………………………………………. | 24 |
|  | **Список литературы**……………………………………………….. | 25 |

**Введение**

Современный ресторанный бизнес развивается в условиях жесткой конкуренции, где ключевым фактором успеха является качество и скорость обслуживания клиентов. Для достижения этих целей рестораны все чаще внедряют цифровые технологии, которые помогают автоматизировать управление заказами, снижать человеческий фактор в обработке информации и повышать общую производительность. Ручны22е методы учета и обработки заказов постепенно уходят в прошлое, уступая место интегрированным информационным системам, способным обеспечивать комплексное управление бизнес-процессами.

В условиях растущих требований клиентов автоматизация становится не просто дополнительной функцией, а необходимостью. Современные системы управления заказами позволяют ресторанам быстро обрабатывать запросы, исключать ошибки, возникающие при передаче информации между сотрудниками, и эффективно использовать данные для анализа и дальнейшего улучшения сервиса. Автоматизация также способствует более эффективному распределению ресурсов, как внутри заведения, так и между несколькими точками ресторанного бизнеса, если речь идет о сетевых предприятиях.

Для решения этих задач в рамках данной работы была разработана система управления заказами ресторана с использованием языка программирования Python (среда разработки PyCharm) и базы данных PostgreSQL. Выбор данных инструментов обусловлен их надежностью, гибкостью и широкими возможностями для разработки масштабируемых приложений. Система обеспечивает управление основными элементами ресторанного бизнеса, включая прием заказов, учет информации о клиентах, управление меню и анализ статистических данных для принятия решений.

**Цель работы**

Основной целью разработки является создание удобной и функциональной платформы, которая позволит автоматизировать процесс управления заказами ресторана, повысить уровень обслуживания клиентов и сократить временные затраты на выполнение рутинных операций.

**Задачи работы**

Для достижения поставленной цели в проекте были поставлены следующие задачи:

1. **Проектирование архитектуры системы:** разработка структуры платформы, включая серверную и клиентскую части.
2. **Создание базы данных:** разработка базы данных в PostgreSQL для хранения информации о клиентах, заказах, меню и других элементах системы.
3. **Разработка пользовательского интерфейса:** создание интуитивно понятного интерфейса для работы сотрудников ресторана, который обеспечит удобное управление заказами.
4. **Автоматизация рабочих процессов:** реализация функционала для приема, обработки и передачи заказов на кухню с минимальными задержками.
5. **Интеграция аналитических инструментов:** добавление возможностей для анализа ключевых показателей, таких как объемы продаж, популярные блюда и средняя продолжительность обработки заказов.
6. **Тестирование и оптимизация системы:** проверка системы на работоспособность, устранение возможных ошибок и повышение производительности.
7. **Обеспечение масштабируемости:** проектирование системы с возможностью ее дальнейшего расширения, включая подключение новых функций или модулей.

**1 ТЕОРЕТИЧЕСКИЙ ЧАСТЬ**

**1.1 Понятие информационной системы и ее роль в автоматизации ресторанного бизнеса**

Информационные системы в ресторанном бизнесе также играют ключевую роль в обеспечении высокой конкурентоспособности заведения на рынке. В условиях постоянных изменений в предпочтениях клиентов и повышения требований к качеству обслуживания, способность ресторанов эффективно адаптироваться к этим условиям становится необходимостью. Внедрение информационных технологий позволяет ресторанам оперативно реагировать на изменения, что является важным фактором для сохранения лояльности клиентов и привлечения новых посетителей.

Одним из важных аспектов использования информационных систем является их роль в управлении запасами и контроле за расходами. С помощью специализированных модулей в ИС, ресторан может в реальном времени отслеживать наличие продуктов на складе, автоматически обновлять данные по запасам и заказывать необходимые товары. Это помогает минимизировать перерасход и дефицит продукции, а также сокращает время на выполнение закупок.

Кроме того, информационные системы могут значительно повысить уровень персонализации обслуживания. С помощью аналитики данных, ИС может отслеживать предпочтения клиентов, их заказы, а также частоту посещений. Это дает возможность ресторанам создавать индивидуальные предложения для постоянных клиентов, предлагая им персонализированные скидки, меню или специальные акции. В свою очередь, такие подходы повышают удовлетворенность гостей и способствуют их повторным посещениям.

Одним из заметных трендов в сфере ресторанного бизнеса является использование мобильных приложений и онлайн-услуг, интегрированных с информационными системами. В результате гости могут заказать еду через приложение, выбрать место для бронирования и оплатить заказ онлайн, что значительно ускоряет процесс обслуживания. Такие сервисы становятся неотъемлемой частью современной стратегии ресторанов, отвечающих на запросы клиентов, стремящихся к удобству и экономии времени.

Информационные системы также играют важную роль в управлении человеческими ресурсами ресторана. Использование ИС для планирования графиков работы сотрудников, учета рабочего времени, а также анализа производительности помогает оптимизировать трудовые процессы. Это не только повышает эффективность работы персонала, но и снижает риски, связанные с нарушением трудового законодательства. Система также помогает контролировать текучесть кадров и анализировать потребности в дополнительном обучении и повышении квалификации сотрудников.

Еще одним важным аспектом является интеграция с системами управления отношениями с клиентами (CRM), которая позволяет ресторану строить долгосрочные отношения с клиентами, улучшать качество обслуживания и разрабатывать более эффективные маркетинговые стратегии. С помощью CRM-систем можно собирать и анализировать данные о поведении клиентов, их предпочтениях, а также проводить различные маркетинговые мероприятия, такие как рассылка уведомлений о скидках или специальных предложениях.

Внедрение информационных систем в ресторанном бизнесе также способствует повышению уровня безопасности. Например, использование электронных систем для учета финансовых операций и кассовых транзакций позволяет исключить возможные мошенничества и повысить прозрачность бизнеса. Кроме того, такие системы могут обеспечивать контроль за соблюдением санитарных норм и стандартов безопасности на всех этапах работы ресторана, включая хранение и приготовление продуктов.

**1.2 Обзор существующих инструментов для создания систем управления заказами в ресторанном бизнесе**

Современные системы управления заказами в ресторанном бизнесе требуют использования различных технологий для обеспечения эффективности, безопасности и удобства. Рассмотрим инструменты и платформы, которые могут быть использованы для создания таких систем с учетом технологий, применяемых в вашем проекте.

**Используемые технологии в нашем проекте**

**Django** — это мощный веб-фреймворк для **Python**, который активно используется для создания надежных и масштабируемых веб-приложений, включая системы управления заказами. В вашем проекте **Django** отвечает за создание серверной части системы, обработку данных, взаимодействие с базой данных и реализацию бизнес-логики.

Особенности **Django** в контексте ресторанного бизнеса:

* Модели и базы данных. **Django** использует **ORM** (**Object-Relational Mapping**), что упрощает взаимодействие с базой данных. Все данные о заказах, бронированиях, клиентах и отзывах хранятся в базе **PostgreSQL**
* Административная панель. **Django** предоставляет встроенную административную панель, которая позволяет удобно управлять данными, такими как заказы, пользователи и отзывы, без необходимости писать дополнительный код.
* Безопасность. **Django** включает встроенные механизмы защиты данных, такие как защита от **CSRF**, защита сессий и шифрование паролей.
* Масштабируемость. **Django** позволяет легко масштабировать систему, добавлять новые функции и модули, что важно для ресторанов, которые растут по объему и услугах.

**HTML, CSS, JavaScript**   
Для создания пользовательского интерфейса вашего сайта используются основные веб-технологии: **HTML**, **CSS** и **JavaScript**. Они обеспечивают отображение страниц, интерактивность и взаимодействие пользователя с системой.

Использование в проекте:

* **HTML:** создается структура веб-страниц, таких как главная страница, страницы заказов, бронирования столиков, форма регистрации и входа.
* **CSS:** используется для стилизации страниц, обеспечения удобного интерфейса и адаптивности, чтобы сайт был удобен для пользователей на различных устройствах, будь то ПК или мобильные телефоны.
* **JavaScript:** применяется для создания динамичных элементов на сайте, таких как обработка событий на страницах, а также для взаимодействия с сервером через AJAX-запросы. JavaScript также может быть использован для валидации данных на клиентской стороне, например, проверки правильности введенного номера телефона при бронировании столика.

**PostgreSQL**  
PostgreSQL Server является базой данных, которая используется для хранения всех данных в вашем проекте. Это реляционная база данных, которая идеально подходит для задач, связанных с хранением и обработкой больших объемов данных.

Особенности PostgreSQL для ресторанного бизнеса:

* Таблицы данных. Все данные о заказах, пользователях, отзывах и бронированиях хранятся в структурированных таблицах, что облегчает доступ и обработку информации.
* Шифрование данных. Для повышения безопасности персональных данных клиентов используется шифрование данных, например, паролей и другой чувствительной информации.
* Резервное копирование. PostgreSQL поддерживает регулярное резервное копирование данных, что обеспечивает защиту информации в случае сбоев системы.
* Транзакции. Для обеспечения целостности данных, например, при создании или удалении заказа, используется механизм транзакций. Это гарантирует, что операция будет выполнена полностью, или будет отменена, если возникнут ошибки.

**JavaScript для интерактивности и AJAX**   
JavaScript играет важную роль в обеспечении интерактивности и динамичного взаимодействия с пользователем. В вашем проекте JavaScript используется для создания более удобного пользовательского интерфейса, а также для работы с сервером через AJAX.

Как это работает:

* **AJAX-запросы** позволяют JavaScript отправлять асинхронные запросы к серверу для выполнения действий, таких как добавление заказа, бронирование столика или регистрация пользователя, без необходимости перезагрузки страницы.
* **Обновление данных в реальном времени.** Например, при добавлении заказа или изменении данных о бронировании, информация сразу обновляется на странице, что улучшает взаимодействие с системой.
* **Валидация форм.** JavaScript используется для проверки введенных данных, например, правильности номера телефона при бронировании столика.

Используемые в вашем проекте технологии, такие как **Django**, **HTML**, **CSS**, **JavaScript** и **PostgreSQL**, предоставляют мощный инструментарий для создания гибкой и функциональной системы управления заказами для ресторана.

* **Django** управляет серверной частью и взаимодействием с базой данных,
* **HTML**, **CSS** и **JavaScript** создают интерактивные и удобные интерфейсы для пользователей,
* **PostgreSQL** служит надежным хранилищем данных с возможностью резервного копирования и защиты информации.

Эти инструменты и технологии позволяют реализовать эффективную и безопасную систему, которая автоматизирует процессы, улучшает работу ресторана и повышает уровень обслуживания клиентов.

**1.3 Архитектура веб-приложений для системы управления заказами в ресторане**

Архитектура веб-приложений для создания системы управления заказами в ресторане представляет собой набор взаимосвязанных компонентов, каждый из которых отвечает за определенные функции, обеспечивающие стабильную работу системы и удобство использования для сотрудников ресторана и клиентов. В основе архитектуры лежит разделение системы на несколько слоев, каждый из которых выполняет свою роль. Рассмотрим, как можно организовать архитектуру веб-приложения на примере вашей системы управления заказами, использующей Django, HTML, CSS, JavaScript и PostgreSQL.

**1. Клиентская сторона (Frontend)**

Клиентская сторона веб-приложения отвечает за взаимодействие с пользователем, включая отображение информации о меню, оформленные заказы, бронирования, а также систему отзывов. Это основная часть, с которой взаимодействуют как пользователи, так и сотрудники ресторана.

* **HTML** используется для создания структуры страниц: главной страницы, страницы меню, страницы с заказами и бронированиями, страницы отзывов.
* **CSS** отвечает за внешний вид сайта, его адаптивность и удобство использования на различных устройствах (ПК, планшет, мобильный телефон).
* **JavaScript** добавляет интерактивность: например, для валидации введенных данных (имя, номер телефона), отправки данных на сервер через AJAX (без перезагрузки страницы) и динамического обновления контента без необходимости обновления всей страницы.

Пример компонента, работающего на клиентской стороне:

* Форма для оформления заказа: клиент выбирает блюда из меню, добавляет их в корзину, вводит данные для доставки и оформляет заказ. Взаимодействие с сервером происходит асинхронно с помощью AJAX.

**2. Серверная сторона (Backend)**

Серверная сторона приложения, построенная на **Django**, отвечает за обработку запросов от клиента, выполнение бизнес-логики и взаимодействие с базой данных. Все важные процессы, такие как создание и обработка заказов, бронирование столиков, сохранение отзывов, выполняются именно на сервере.

* **Django** обеспечивает создание и обработку HTTP-запросов от клиента. В серверной части реализуются функции:
  + Обработка формы заказа: проверка введенных данных, создание записи в базе данных.
  + Обработка регистрации и аутентификации пользователей (например, создание аккаунта клиента).
  + Обработка отзывов и оценок от клиентов.
  + Управление данными о заказах и бронированиях (например, создание и удаление записей в базе данных).
* **Роутинг** в Django настроен так, чтобы запросы с клиента направлялись к соответствующим обработчикам (views), где выполняется необходимая логика и осуществляется работа с базой данных.

**3. База данных (Data Layer)**

**PostgreSQL** является центральным компонентом архитектуры, где хранятся все данные о заказах, клиентах, отзывах, бронированиях и других ключевых сущностях. Это реляционная база данных, которая эффективно управляет большими объемами структурированных данных.

* **Модели Django** (использующие ORM) представляют данные в виде объектов Python, и их взаимодействие с базой данных происходит через ORM (Object-Relational Mapping).
* База данных хранит следующие таблицы:
  + **Таблица заказов**, в которой содержатся данные о каждом заказе (например, блюда, их количество, стоимость, статус).
  + **Таблица пользователей** для хранения информации о клиентах, их аккаунтах и предпочтениях.
  + **Таблица отзывов**, в которой хранится информация о том, какие отзывы оставлены клиентами, с оценками и комментариями.
  + **Таблица бронирований**, которая отслеживает, какие столики зарезервированы и на какое время.
* **Шифрование** данных клиентов (например, паролей) обеспечивается через встроенные механизмы Django, такие как использование **hashing-функций**.

**4. Взаимодействие между слоями (Communication)**

Между слоями архитектуры происходит постоянное взаимодействие через HTTP-запросы и ответы:

* **Клиентская сторона (Frontend)** отправляет запросы (например, POST-запросы) на сервер для обработки данных (оформление заказа, бронирование столика, регистрация пользователя).
* **Сервер (Backend)** обрабатывает запросы, выполняет бизнес-логику, а затем взаимодействует с базой данных для получения или сохранения информации.
* **База данных** предоставляет серверу необходимые данные для формирования ответа (например, информацию о доступных столиках, статусе заказа или списке отзывов).

Обработка всех этих данных происходит в реальном времени, и изменения (например, подтвержденный заказ или забронированный столик) отображаются на клиентской стороне без необходимости перезагрузки страницы благодаря использованию **AJAX**.

**5. Безопасность и резервное копирование**

* Для обеспечения безопасности веб-приложения используется несколько технологий, включая защиту от атак CSRF, шифрование паролей и данных через SSL/TLS, а также регулярное резервное копирование данных через **PostgreSQL.**
* Ваша система настроена на автоматическое бэкапирование данных, что позволяет избежать потери важной информации в случае непредвиденных сбоев.

Архитектура веб-приложения для системы управления заказами в ресторане состоит из нескольких ключевых слоев, каждый из которых выполняет свою роль:

* **Frontend (клиентская сторона)** обеспечивает удобный и интуитивно понятный интерфейс для пользователей (клиентов и персонала ресторана) с помощью HTML, CSS и JavaScript.
* **Backend (серверная сторона)** на базе Django обрабатывает запросы, реализует бизнес-логику и взаимодействует с базой данных для управления заказами, бронированиями и отзывами.
* **Data Layer (слой данных)**, реализованный с PostgreSQL, надежно хранит все данные и предоставляет быстрый доступ к ним.

Эта многослойная архитектура обеспечивает эффективную работу системы, безопасность данных и отличное взаимодействие с пользователем.

**2 Практическая реализация**

**2.1 Создание и настройка базы данных для системы управления заказами в ресторане**

Для хранения данных в проекте была выбрана реляционная база данных PostgreSQL. Данный выбор обусловлен высокой производительностью, поддержкой ACID-транзакций, гибкостью и широкими возможностями работы с данными. В качестве графического интерфейса для управления базой данных использовался pgAdmin 4, который предоставляет удобные инструменты для работы с SQL-запросами, схемами данных и таблицами.

**Проектирование структуры базы данных**

Перед созданием базы данных был разработан концептуальный дизайн, определяющий основные сущности и их взаимосвязи. В системе были реализованы следующие модели:

* MenuItem – хранит информацию о блюдах в меню (название, описание, цена).
* Order – содержит сведения о заказах (состав заказа, клиент, статус, сумма).
* User – информация о пользователях (имя, email, роль).
* Review – отзывы клиентов о заведении (рейтинг, комментарий, дата).
* Reservation – данные о бронировании столиков (дата, время, клиент, количество мест).

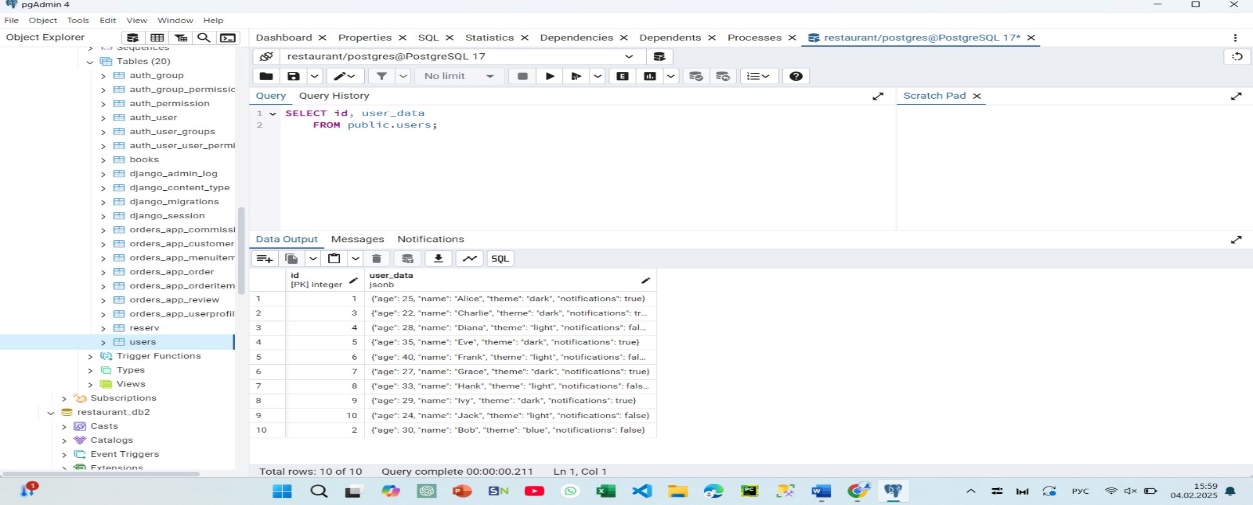


Рисунок 2.1. Таблицы база данных

**Создание базы данных**   
В **pgAdmin 4** была создана база данных под названием restaurantdb.

**Создание пользователя**   
Через командную строку был создан пользователь с паролем, чтобы обеспечить безопасное подключение.

**Настройка Django-проекта**   
В **PyCharm** был установлен Django командой:

pip install django

Затем был создан новый проект:

django-admin startproject restaurant\_project2

Далее была создана и зарегистрирована в проекте новая Django-приложение для обработки заказов:

python manage.py startapp orders\_app

**Настройка базы данных в Django**  
В файле settings.py была произведена настройка подключения к PostgreSQL(Рис.2.1):

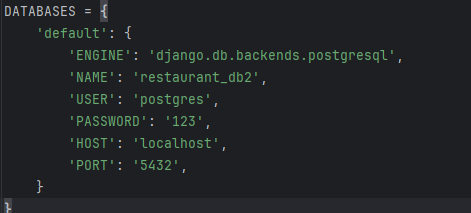


Рисунок 2.2. Настройка подключения к базе

**2.2 Разработка пользовательского интерфейса**

После настройки базы данных и создания Django-проекта, следующим этапом стала разработка пользовательского интерфейса. Для этого были созданы HTML-шаблоны, которые отвечают за отображение данных на стороне клиента.

**Создание главной страницы (home.html)**

Для главной страницы был создан файл home.html, который содержит структуру и стили для отображения информации о ресторане, меню и кнопок для взаимодействия с системой.

**Структура файла home.html:**

* **Шапка (Header):** Содержит логотип ресторана и навигационные ссылки для перехода между разделами системы (регистрация, меню, заказы, отзывы).
* **Главный блок (Hero Section):** Включает приветственное сообщение, описание ресторана и кнопки для бронирования стола и оставления отзыва.
* **Раздел меню (Menu Section):** Отображает рекомендуемые блюда с изображениями, описанием и ценами.
* **Подвал (Footer):** Содержит контактную информацию.

**Стилизация:**

* Использованы CSS-переменные для удобства управления цветовой схемой.
* Добавлены анимации для плавного появления элементов при загрузке страницы.
* Реализована адаптивная вёрстка для корректного отображения на мобильных устройствах.

**JavaScript:**

* Добавлены обработчики событий для кнопок, плавной прокрутки и мобильного меню.
* Использован Intersection Observer для анимации появления карточек с блюдами при прокрутке.

**Пример кода home.html:**

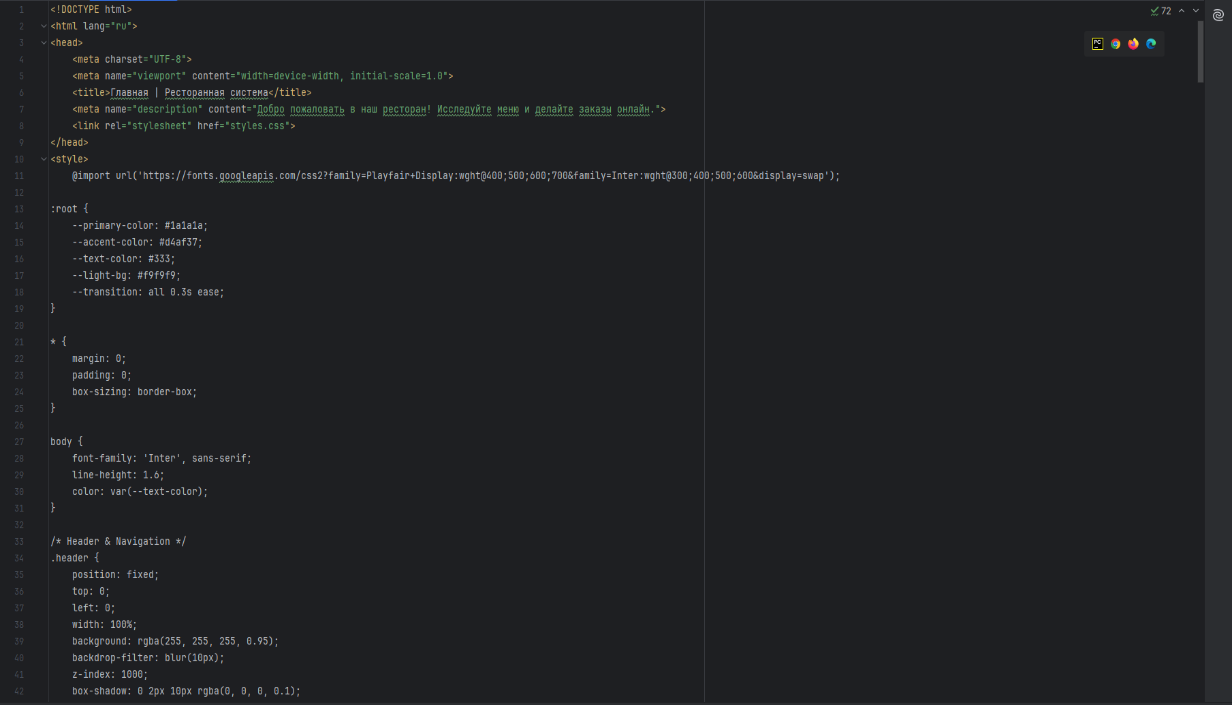


Рисунок 2.3. Код html-файла

**Создание страницы меню (menu.html)**

Для отображения меню ресторана был создан файл menu.html. Этот шаблон позволяет администраторам добавлять, редактировать и удалять блюда, а пользователям — просматривать меню и делать заказы.

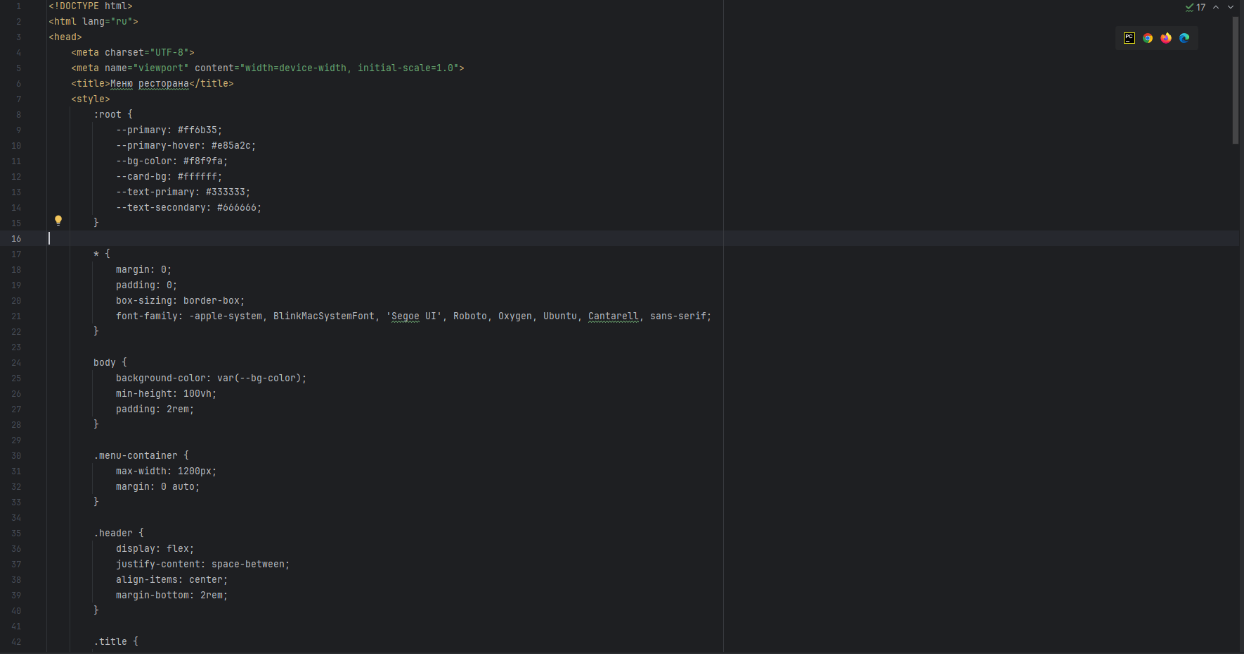
**Структура файла menu.html:**

* **Шапка (Header):** Содержит заголовок страницы и кнопку для добавления нового блюда.
* **Сетка меню (Menu Grid):** Отображает список блюд с их названиями, описаниями и ценами.
* **Кнопки действий:** Для каждого блюда предусмотрены кнопки "Заказать", "Изменить" и "Удалить".

**Стилизация:**

* Использованы CSS-переменные для управления цветовой схемой.
* Добавлены анимации при наведении на карточки с блюдами.
* Реализована адаптивная вёрстка для корректного отображения на мобильных устройствах.

**Пример кода menu.html:**

****

**Рисунок 2.4. Пример кода menu.html**

**Создание моделей для меню**

Для работы с меню была создана модель MenuItem в файле models.py. Эта модель включает следующие поля:

* name — название блюда.
* description — описание блюда.
* price — цена блюда.

**Пример кода модели MenuItem:**

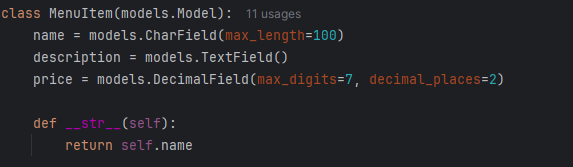
****

Рисунок 2.5. Модель для Menu

**Создание дополнительных шаблонов**

Для работы с меню были созданы дополнительные шаблоны:

* **menuitem\_confirm\_delete.html:** Страница подтверждения удаления блюда.
* **menuitem\_form.html:** Форма для добавления и редактирования блюд.

**Пример кода menuitem\_confirm\_delete.html:**

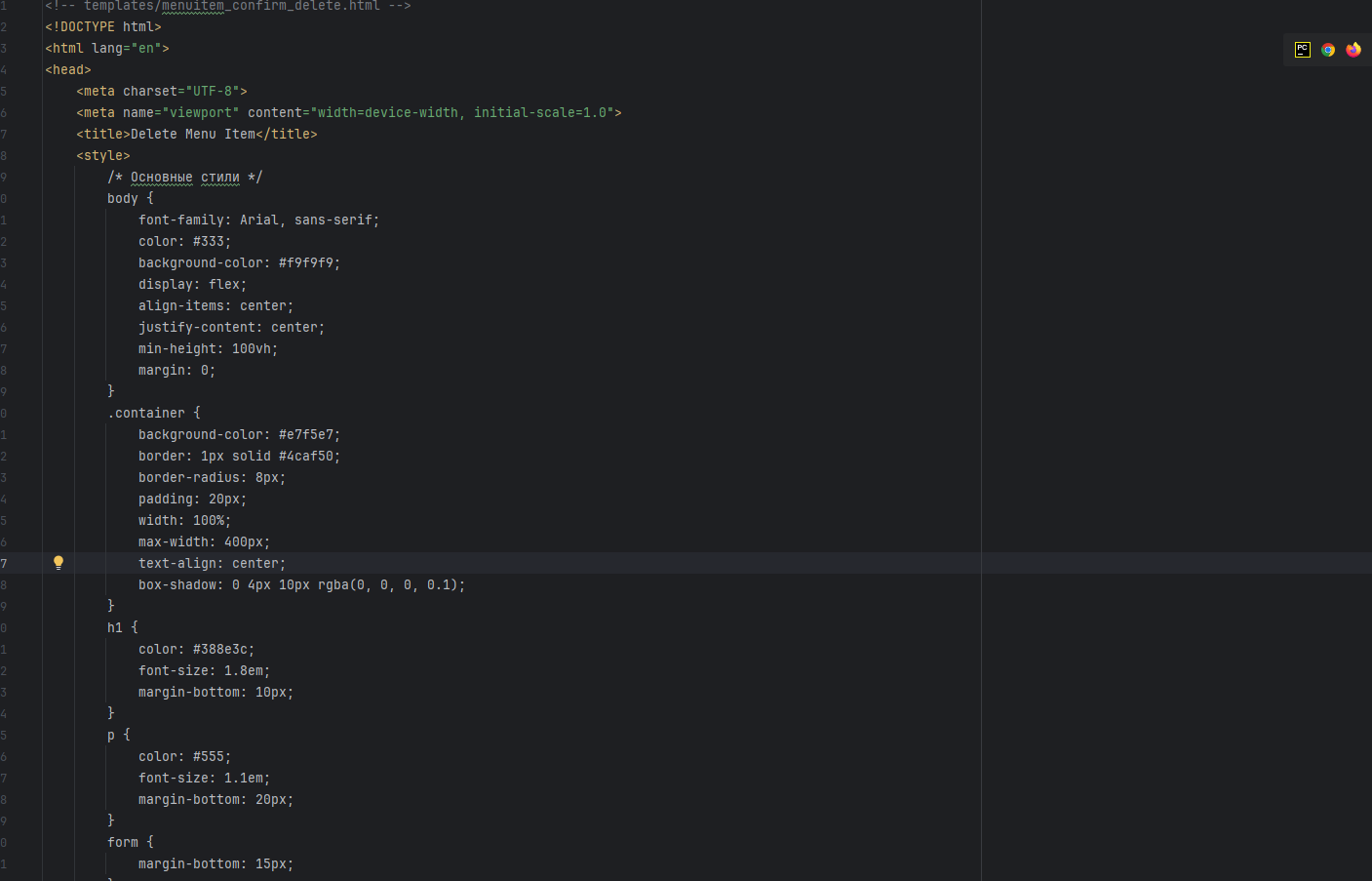
****

Рисунок 2.6. Пример кода menuitem\_confirm\_delete.html

**Пример кода menuitem\_form.html:**

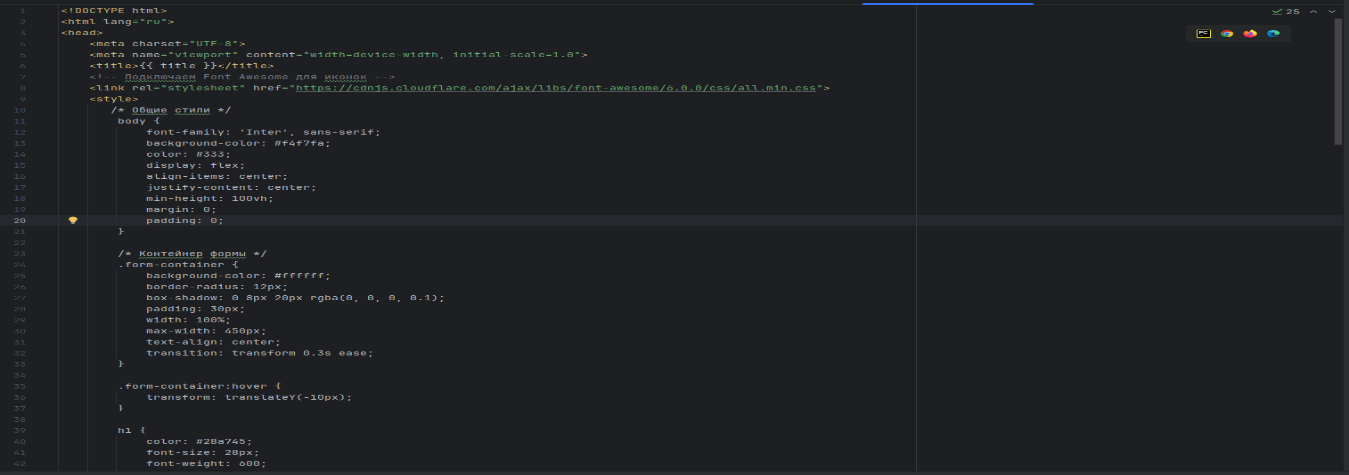
****

Рисунок 2.7. Пример кода menuitem\_form.html

**Создание страницы списка заказов (order\_list.html)**

Для отображения списка заказов был создан файл **order\_list.html**. Этот шаблон позволяет отображать все заказы, их статус и предоставляет функционал для удаления заказов.

**Структура файла order\_list.html:**

* **Шапка (Header):** Содержит заголовок страницы "Список заказов".
* **Таблица заказов:** Отображает все заказы в таблице с такими колонками, как ID заказа, название блюда, имя клиента, дата заказа, статус заказа и действия (кнопка удаления).
* **Сообщение о пустом списке:** В случае отсутствия заказов отображается строка "Нет заказов".

**Стилизация:**

* Для таблицы использован стиль с четкими границами и цветовой акцент на строках с четными номерами.
* Добавлена кнопка удаления с красным фоном, который меняется при наведении.
* Использована адаптивная вёрстка для корректного отображения на мобильных устройствах.

**Пример кода order\_list.html:**

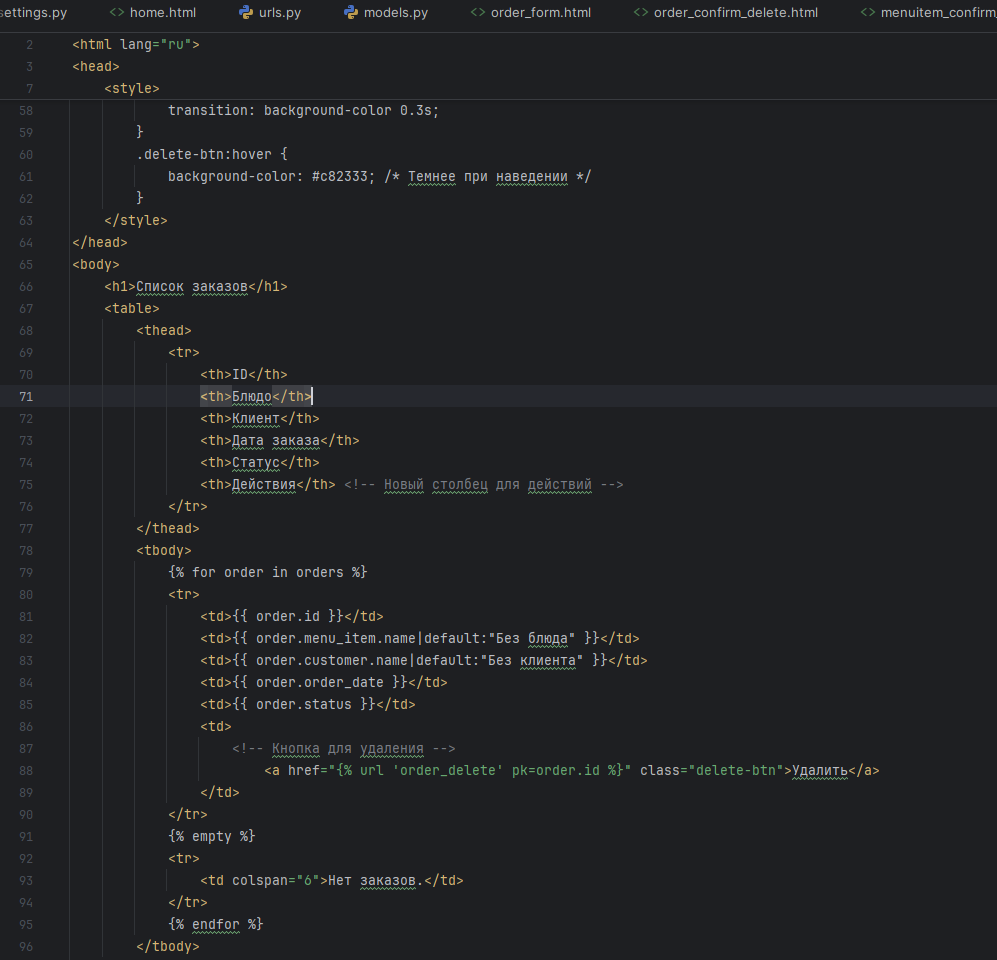


Рисунок 2.8. Код order\_list.html

**Создание формы для заказа (order\_form.html)**

Для создания и редактирования заказов был создан файл order\_form.html. Этот шаблон позволяет администратору и пользователю создавать и редактировать заказы, выбирая блюда и указывая информацию о клиенте.

Структура файла order\_form.html:

* Шапка (Header): Заголовок, который меняется в зависимости от контекста: "Создать заказ" или "Редактировать заказ".
* Форма: Содержит поля для ввода информации о заказе: выбор блюда и клиента.
* Кнопка отправки: Кнопка для сохранения или создания нового заказа.

Стилизация:

* Использована простая, но элегантная стилизация с зелеными оттенками, создающими приятную атмосферу для пользователей.
* Форма выравнивается по центру экрана, обеспечивая удобный интерфейс для взаимодействия.

**Пример кода order\_form.html:**

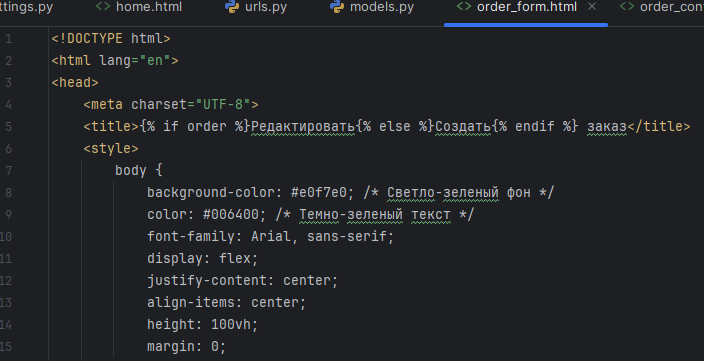
****

Рисунок 2.9. Пример кода order\_form.html

Создание страницы подтверждения удаления заказа (order\_confirm\_delete.html)

Для подтверждения удаления заказа был создан файл **order\_confirm\_delete.html**. Этот шаблон позволяет пользователю подтвердить или отменить удаление заказа, отображая его ID.

**Структура файла order\_confirm\_delete.html:**

* **Шапка (Header):** Заголовок "Удалить заказ".
* **Сообщение о подтверждении:** Текст с ID заказа и вопросом о подтверждении удаления.
* **Кнопки:** Кнопка "Да, удалить" для удаления и ссылка "Отмена" для возврата.

**Стилизация:**

* Цветовая палитра включает голубые оттенки для кнопок и текста.
* Форма аккуратно центрируется на экране, создавая удобный интерфейс для принятия решения.

Пример кода **order\_confirm\_delete.html**:

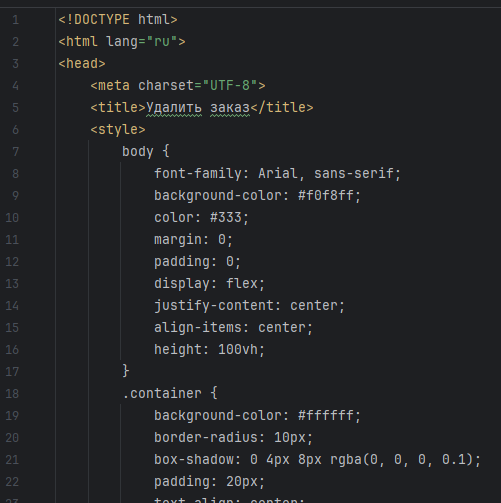


Рисунок 3.0. Пример кода order\_confirm\_delete.html

Создание моделей для заказов

Для работы с заказами была создана модель **Order** в файле models.py. Модель включает следующие поля:

* **customer:** Внешний ключ, связывающий заказ с клиентом.
* **order\_date:** Дата и время создания заказа.
* **status:** Статус заказа (например, "Pending" или "Completed").
* **menu\_item:** Внешний ключ, связывающий заказ с блюдом.

**Пример кода модели Orders:**

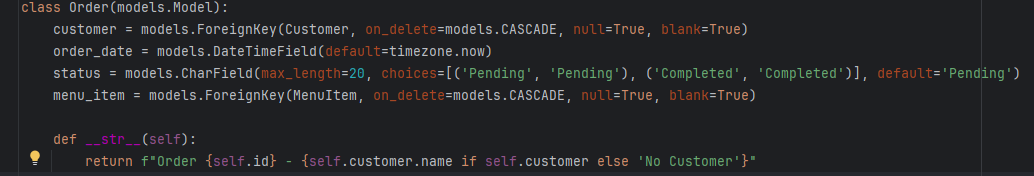
****

Рисунок 3.1. Модель Orders

**Заключение**

В ходе разработки ресторанной системы был успешно выполнен весь процесс от настройки базы данных до создания интуитивно понятного интерфейса для пользователя. Все ключевые этапы работы были интегрированы и позволили создать систему, которая эффективно решает задачи управления рестораном и взаимодействия с клиентами.

Процесс разработки включал создание моделей данных для хранения информации о блюдах, заказах и пользователях, что позволило построить гибкую и масштабируемую архитектуру базы данных. Реализация пользовательского интерфейса обеспечила удобство работы с системой, с адаптивной версткой для корректного отображения на разных устройствах. Были созданы страницы для регистрации, меню, а также функционал для управления заказами и блюдами. Важно отметить, что интерфейс был разработан с учетом современного подхода к удобству и простоте использования.

Также были внедрены элементы безопасности, включая авторизацию пользователей для доступа к административным функциям. Всё это позволило создать надежную и функциональную систему, которая сочетает в себе как удобство для пользователей, так и удобство для администраторов, улучшая эффективность работы ресторана.

Проект продвигается в сторону полной автоматизации процессов, предоставляя ресторану и его персоналу эффективные инструменты для управления заказами, блюдами и пользователями.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Грин Р. Современный Python: написание кода на Python 3. — Москва: Вильямс, 2021, 320 стр.
2. Ицик Бен-Ган. PostgreSQL. Высокопроизводительный код SQL. — Москва: Русская Редакция, 2020, 177 стр.
3. Бэнкс С., Ламберт А. Полное руководство по PostgreSQL. — Москва: ДМК Пресс, 2020, 456 стр.
4. Вильямс Р. Python для работы с базами данных. — Москва: ДМК Пресс, 2022, 368 стр.
5. Джонсон Б., Макинтош Г. PostgreSQL 2019: Руководство для профессионалов. — Санкт-Петербург: Питер, 2021, 640 стр.