**Автономная некоммерческая организация дополнительного профессионального образования «Санкт-Петербургский университет повышения квалификации и профессиональной переподготовки»**

**ДИПЛОМНАЯ  РАБОТА**

**по дополнительной  профессиональной  образовательной  программе**

**профессиональной переподготовки**

**«Вычислительные, программные, информационные системы и компьютерные технологии»**

**Тема:**

**«»**

Дипломную работу  подготовил:

**Халилов Руслан Нариманович**

Должность:

Старший эксперт отдела криминалистических экспертиз ЭКЦ МВД по Республике Крым

Место работы:

Министерство Внутренних Дел по Республике Крым

**г. Симферополь,  Республика Крым**

**Российская Федерация**

# CОДЕРЖАНИЕ

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
|  | Определения, термины и сокращения | 5 |
| 1 | Анализ и исследование предметной области | 7 |
| 1.1 | Анализ рынка и обоснование выбора | 8 |
| 1.2 | Цель разработки системы | 9 |
| 2 | Проектирование информационной системы | 14 |
| 2.1 | Описание диаграмм | 14 |
| 2.1.1 | Диаграмма деятельности | 14 |
| 2.1.2 | Use-Case диаграмма | 15 |
| 2.1.3 | ER – диаграмма | 16 |
| 2.2 | Проектирование UI/UX | 17 |
| 3 | Разработка информационной системы | 19 |
| 3.1 | Среда разработки | 19 |
| 3.1.1 | Java Script | 19 |
| 3.1.2 | Task-менеджер Gulp | 20 |
| 3.1.3 | Django, преимущества и функции | 20 |
| 3.1.4 | SQLite Studio | 21 |
| 3.2 | Архитектура проекта | 22 |
| 3.2.1 | Авторизация | 24 |
| 3.2.2 | User Interface | 25 |
| 3.2.3 | Функционал персонала | 28 |
|  | Заключение | 30 |
|  | Список использованной литературы | 31 |
|  | Приложение. Текст программы | 33 |

**ВВЕДЕНИЕ**

На сегодняшний день в мире каждая индустрия непосредственно связана с IT- миром. Для облегчения работы предприятий, малого или большого бизнеса люди используют различные информационные системы, позволяющие хранить, искать или обрабатывать необходимую информацию, но также и распространяющие определённую информацию пользователю. С каждым годом развитие ИС упрощает и ускоряет процесс работы организаций, и индустрия ресторанного бизнеса не является исключением.

Интернет в ресторанном деле играет крайне важную роль и не стоит пренебрегать возможностью выхода продукта во всемирную сеть. Причина его важности - это бесконечный поток числа пользователей, что делает предприятие не только источником привлечения клиентов, но и целью потенциальных бизнес- партнёров. Создание большой информационной Web-системы будет выполнять сразу две работы, на стороне пользователя и предприятия.

Проблема текущих информационных систем в том, что их установка дорого обходится, для использования нужно иметь углублённые знания системы и при возникновении проблем внутри системы, требуется вмешательство установщиков. Актуальность данного проекта состоит в том, чтобы создать автоматизированную систему - более лёгкую в использовании для персонала, содержащую информацию продукта для клиентов, что является одним из приоритетов в разработке данного проекта. Информационная система, построенная на Web-платформе, не нуждается в системных требованиях, достаточно иметь устройство доступом к интернету (это может быть даже смартфон). Кроме того, по мере появления новых требований, функций или данных, проект легко может быть расширен или модифицирован.

# Определения, термины и сокращения

В таблице 1 предоставлены все термины и сокращения, используемые при описании данного проекта и связанные с технологиями, которые были использованы в разработке.

# Таблица 1 – Сокращения, термины и их определения

|  |  |
| --- | --- |
| Сокращение или термин | Определение |
| HTML | Язык гипертекстовой разметки  (Hypertext Markup language) |
| CSS | Язык описания внешнего вида  документа (Cascading Style Sheets) |
| Фреймворк | Программное обеспечение, облегчающие разработку и объединение разных компонентов в  один общий проект |
| Task-менеджер | Программа, упрощающая и  ускоряющая рабочий процесс |
| UI**/**UX | Опыт пользователя и то, что он видит  (User Interface / User Experience) |
| JavaScript (js) | Динамический многозадачный язык программирования, более широко  применяющийся в браузерах |
| jQuery | Библиотека JavaScript, облегчающая процесс работы с языком и работающая напрямую с элементами  DOM – дерева. |
| Npm | Менеджер пакетов в составе  Node.js(node package manager) |
| Django | Фреймворк для веб-приложений, на  языке Python. |
| GitBash | Приложение, заменяющее работу  командной строки |
| DOM | Представление HTML в виде дерева  тегов (Document Object Model) |
| Node.js | Программная платформа,  использующая Node.js |
| MVC | Схема разделения данных на три  отдельных компонента(Model-View- Controller) |
| ООП | Объектно-ориентированное  программирование |
| СУБД | Система управления базами данных |
| ИС | Информационные системы |

# 

# Анализ и исследование предметной области

* 1. **Анализ рынка и обоснование выбора**

Для эффективной работы предприятий малого или большого бизнеса нужны удобные и эффективные инструменты управления. Информационная система – это платформа, внедряемая в предприятие, которая оптимизирует и ускоряет процесс работы. Современные ИС состоят из Front-офисных и Back-офисных подсистем. Front-офисные подсистемы предназначены для оптимизации обслуживания клиента, Back-офисные подсистемы автоматизируют менеджмент предприятия, считают расходы и доходы.

Автоматизация системы ресторанного бизнеса является ключевым фактором конкурентоспособности предприятия. Качество ресторана измеряется не только хорошими блюдами, но и скоростью обслуживания клиента, подробной информацией о ресторане, его меню и т.д. Хорошее программное обеспечение позволяет повышать эффективность работы персонала и уровень сервиса, увеличивать прибыль, вести отчетность о прибыли и качестве обслуживания.

Спецификация ресторанного бизнеса охватывает следующие стадии:

* Производство
* Обслуживание клиентов
* Управление персоналом
* Ведение документации

На сегодняшний день существует большое количество систем, предназначенных для оптимизации деятельности предприятий общественного питания. Самые популярные из них: R-keeper, 1C:Предприятие, iiko, ArchiDelivery, АСТОР и т.д. Все эти платформы широко используются в организациях общественного питания уже десятки лет и считаются лидерами на рынке. Причиной этого являются: многофункциональность, регулярная техническая поддержка и выгодные условия сотрудничества с создателями данных систем. Обычно для их установки и дальнейшей эксплуатации используют специальное оборудование, чаще всего это терминал или компьютер с операционной системой Windows.

Проанализировав данные системы, были замечены несколько недостатков. Во-первых, системы никак не взаимодействуют с клиентами. То есть, система не предоставляет ресурс, дающий какую-либо информацию гостю о предприятии. Также, гость не может никак взаимодействовать с этим платформами, доступ к функциям данных систем запрещён. Во-вторых, для эксплуатации данных систем требуется дополнительное оборудование и в дальнейшем необходимо обязательно поддерживать работоспособность этого оборудования. В-третьих, для работы с данными системами нужно быть проинструктированным. То есть, интерфейс данных систем требует определённых знаний и понимания, есть вероятность того, что можно нажать не туда, отсюда возникает риск происшествия непредвиденных обстоятельств. Также одним из самых главных минусов данных систем является то, что при внесении изменений в функционал, необходимо полностью модифицировать всю программу, что отнимает много времени.

При создании данного проекта были учтены именно эти недочёты других систем. Самым оптимальным решением данных минусов является создание информационной Web-платформы. Первое преимущество данного проекта состоит в том, что его взаимодействие распространяется не только на сотрудников заведения, но и на клиентов предприятия, что влечёт за собой несколько плюсов. Это: привлечение потока клиентов посредством выхода продукта в интернет, увеличение лояльности заведения и привлечение потенциальных партнёров. Второе преимущество, это отсутствие каких-либо системных требований. Для использования данной системы достаточно лишь устройства с доступом в интернет (это может быть даже смартфон). Третьим преимуществом является мобильность данной системы. Исходя из предыдущего пункта, можно сделать вывод, что управлять Web-системой можно находясь в любой точке мира. Управление через терминал ограничивает персонал и делает прикованным к одной локации. Четвёртое преимущество в том, что дополнить функционал данной системы проще из-за того, что она интегрирована в Web-разработку. Web-приложение легко поддаётся изменению и более широко функционально чем программа.

# Цель разработки системы

Цель разработки проекта – разработка информационной системы ресторана с интеграцией Web-приложения и реализация функций, отвечающих требованиям пользователей и необходимых персоналу в данной сфере.

Задачи, которые необходимо реализовать в проекте:

* Предоставление подробной информации о предприятии;
* Возможность отправки данных о бронировании и обратной связи;
* Просмотр меню заведения по фильтрам;
* Предоставление функционала для эффективной работы персонала. Функции, распространяющиеся только для персонала заведения:
* добавление и удаление сотрудников предприятия;
* обработка форм пользователей;
* редактирование меню (добавление и удаление);
* авторизация;
* создание онлайн чеков, для дальнейших манипуляций с ними.

# Проектирование информационной системы

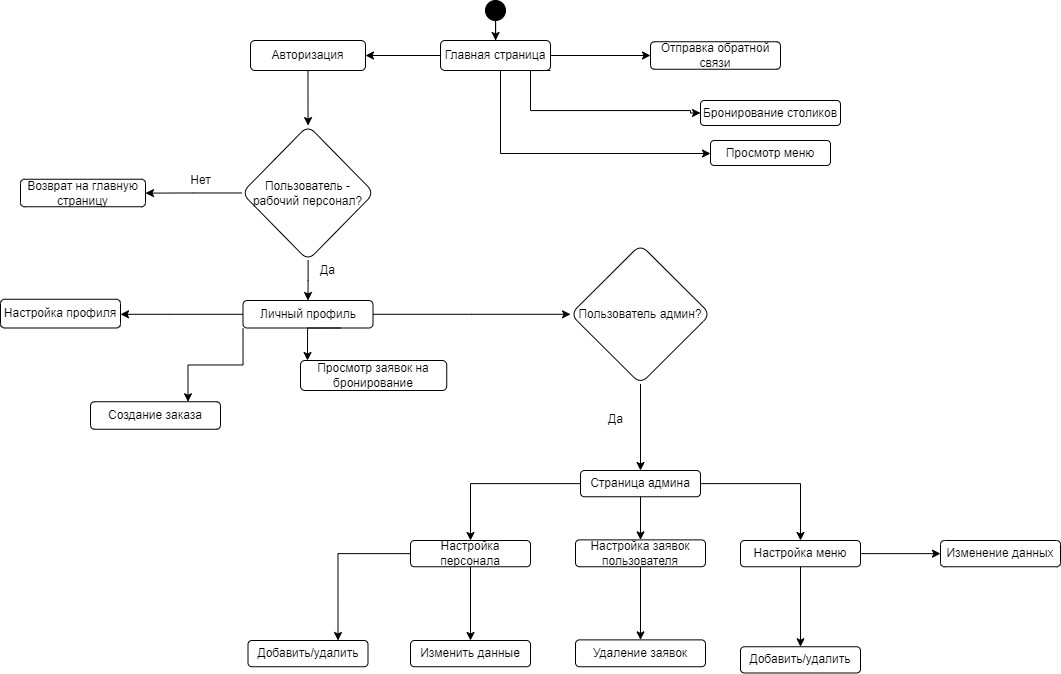
* 1. **Описание диаграмм**

# Диаграмма деятельности

Диаграмма деятельности предоставляет последовательность действий и отражает некоторые аспекты поведения системы. Благодаря созданию данной схемы можно понять: роль каждого компонента, этапы реализации и последовательность выполнения задач.

Проект начинается с того момента, как пользователь заходит на главную страницу. Для того, чтобы обычному посетителю сайта воспользоваться функционалом, регистрация и вход не нужны. Сразу доступны функции:

* бронирование столиков (указывается номер телефона, имя, желаемая дата и количество человек)
* отправки обратной связи (указывается номер телефона, имя и комментарий клиента)
* просмотр меню

Если в систему зашёл рабочий персонал, ему необходимо перейти на страницу авторизации. При успешном входе, пользователя перенаправит в его кабинет, где ему будет доступен отдельный функционал, в зависимости от занимаемой должности. Администратор может: просматривать заявки, отправленные пользователем, изменять данные персонала, удалять и добавлять новых сотрудников, редактировать меню (добавлять и удалять блюдо, менять данные блюд).

# Рисунок-2.1.1 – Диаграмма деятельности

* + 1. **Use-Case диаграмма**

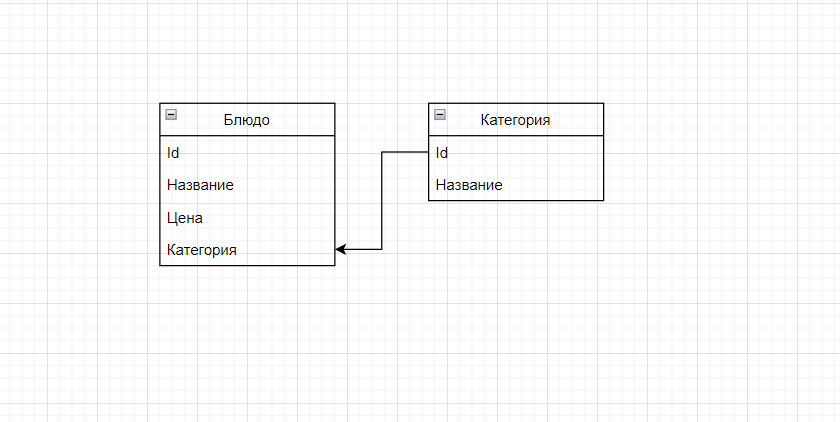
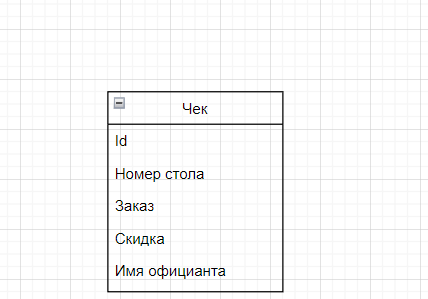
Use-case диаграмма описывает взаимодействие системы с пользователем. Пользователем может быть, как человек, так и другая система, как правило их 2 и больше. Диаграмма описывает именно внешнее поведение системы. Благодаря данной диаграмме можно увидеть роли пользователей и основной поток действий.



# Рисунок-2.1.2 – диаграмма Use-Case

* + 1. **ER – диаграмма**

ER-модель используется при проектировании баз данных. С помощью неё выделяются ключевые сущности и обозначаются связи между ними. Ниже представлена ER-диаграмма для меню ресторана.



# Рисунок-2.1.3 – ER-диаграмм

* 1. **Проектирование UI/UX**

Для правильной разработки интерфейса необходимо учитывать все тонкости продукта, то есть функционал, удобство, лёгкость использования, красоту и позиционирование. Для проектирования эффективного взаимодействия пользователей с сайтом использовалось приложение популярное приложение – Figma. Одними из основных принципов правильно созданного дизайна являются: понятность и отклик. Для начала необходимо проанализировать целевую аудиторию, так как продукт делается для людей, то основным пунктом является изучение их потребностей. После визуализации всех идей начинается тестирование. Основным важным навыком при разработке UI является способность видеть продукт глазами людей, которые будут его использовать. В результате всех действий получились:

1. понятная структура сайта;
2. узнаваемость каждого элемента;
3. отзывчивость;
4. минимум усилий при взаимодействии с интерфейсом;
5. эстетичность продукта.

# Разработка информационной системы

* 1. **Среда разработки**

Использование различных сред разработки - это прямая обязанность каждого программиста на сегодняшний день. Они упрощают и ускоряют процесс разработки проектов. Из множества текстовых редакторов (Sublime Text, Atom, Notepad++ и т.д.) выбор был остановлен на Sublime Text и Visual Studio Code. В итоге скачать и просмотрев оба редактора, был выбран VS Code. К его преимуществам можно отнести: бесплатный доступ, наличие огромного количества сниппетов, ускоряющих процесс написания кода, поддержка всех множества языков программирования, совместимость со всеми популярными операционными системами, и, что не мало важно – возможность выбора цветовой гаммы интерфейса и кода. Именно в VS code была написана структура и применена стилизация сайта, написана Front-end часть проекта (UI**/**UX). Также при написании Back-end части на фреймворке Django, использовался PyCharm community – интегрированная среда разработки, использующая язык программирования Python.

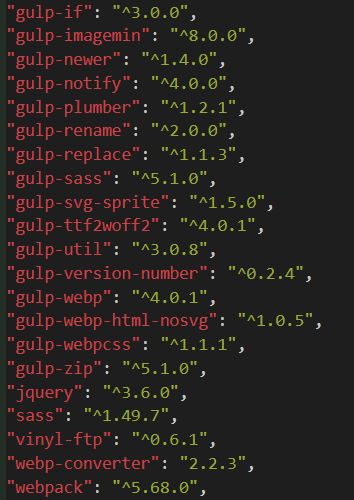
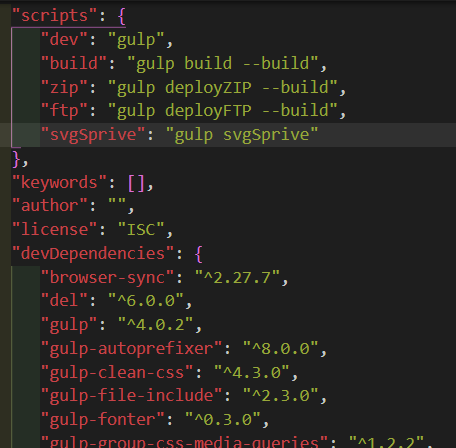
# JavaScript

На сегодняшний день JavaScript является одним из самых популярных и динамично развивающихся языков в среде программирования, особо широко используется в браузерах. Помимо этого, js используется в платформе Node.js. Именно благодаря этому языку пишется Front-end часть проектов и создаётся User Interface. Однако сейчас, одного js мало и поэтому необходимо использовать дополнительные библиотеки или Фреймворки. Одними из самых популярных на сегодняшний день являются: Angular, React, Vue.js, jQuery и другие. В разработке своего проекта я выбрал именно jQuery. jQuery – это самая популярная библиотека js на сегодняшний день, с множеством функций. Данная библиотека облегчает взаимодействие HTML с js, обеспечивает доступ к элементам DOM и легко манипулирует ими. Также в своём проекте я использовал один из плагинов jQuery - slick-carousel для корректной работы слайдеров любой сложности на сайте. Несмотря на то, что на сегодняшний день jQuery считается менее актуальной библиотекой и есть более мощные аналоги, большинство сайтов до сих пор написано с помощью неё.

# Task- менеджер gulp

Для ускорения и оптимизации процесса вёрстки был использован популярный task-менеджер Gulp. Для запуска менеджера используется командная строка или любой установленный терминал, в моём случае GitBash. Список задач, выполняемых с использованием Gulp: оптимизация HTML кода, изображений и шрифтов, минификация CSS и JavaScript файлов, запуск в режиме разработчика или production, архивация готового проекта, использование препроцессора Sass.

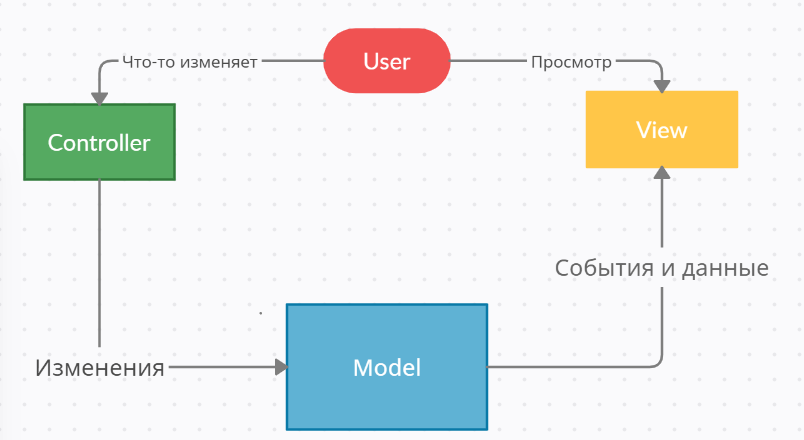
Для настройки и установки функций менеджера используется npm и терминал. Терминал уже интегрирован в VS Code, что ускоряет процесс установки, настройки и дальнейшей работы менеджера.



# Рисунок-3.1 – Использованные скрипты и библиотеки при вёрстке

* + 1. **Django, преимущества и функции**

Фреймворки Back-еnd занимают важное место, как в проектах, так и для самих разработчиков. Программисты всегда обращают внимание, какие задачи придётся решать и какими инструментами стоит воспользоваться. Каждый раз при разработке веб-сайтов требуются схожие компоненты и возникают однотипные проблемы. Django является бесплатным и свободным средством для веб- приложений, написанном на языке Python, а именно этот язык пользуется большой любовью разработчиков. Он предлагает нам готовые шаблоны для использования, интегрируя их в код тем самым уменьшая его размер. Также стоит подметить, что в Django имеется встроенный интерфейс администратора и кэширование, упрощает работу с формами и включает надёжную аутентификацию. Во фреймворке применяются принципы ООП, благодаря MVC разделяются пользовательский интерфейс и бизнес-логика. *Model* - содержит всю бизнес-логику приложения. *View -* отвечает за отображение данных пользователю. *Controller* - хранит код, отвечающий за обработку действий пользователя.



# Рисунок-3.2 – Схема MVC

Проекты, написанные на Django - всегда отличаются компактностью кода. Так же огромным плюсом Django является то что, столкнувшись с проблемой, можно без особых сложностей найти её решение благодаря активному обществу. Для его использования был установлен PyCharm community и установлен Python.

# SQLite Studio

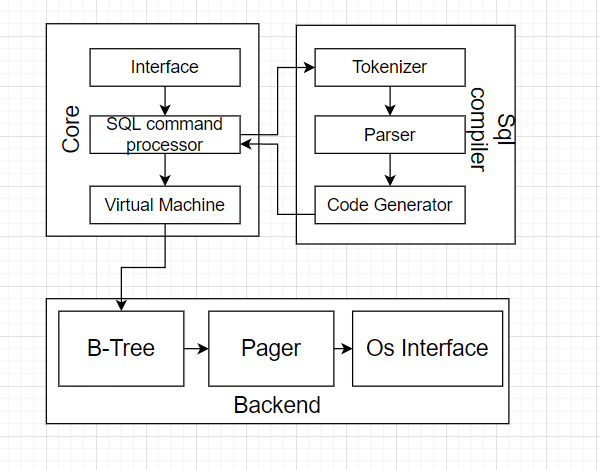
Для работы с базами данных принято использовать менеджер БД, который может управлять данными в визуальном режиме. Несмотря на то, что все СУБД выполняют одну задачу - дают возможность пользователям создавать, изменять и получать доступ к информации, хранящейся в базах данных, сама задача может выполняться по-разному. Также, функции и возможности каждой СУБД могут отличаться. Различные СУБД документированы по-разному: кратко или подробно. При сравнении различных популярных баз данных, учитывалось: удобность использования, корректная интеграция с другими продуктами, которые уже используются в проекте (Django).

К преимуществам SQLite studio можно отнести: бесплатный доступ, доступность для всех операционных систем, возможность импорта и экспорта в различных форматах, русскоязычность и различные дополнения. Интерфейс и использование программы просты, также она является локальной. Стоит отметить компактность этой СУБД, что заняло пару минут на распаковку и установку файлов.

* 1. **Архитектура проекта**

Разработка приложения включает в себя такие технологии, как jQuery, Django и SQLite Studio, как показано ниже (см. рисунок 3.1). Модели базы

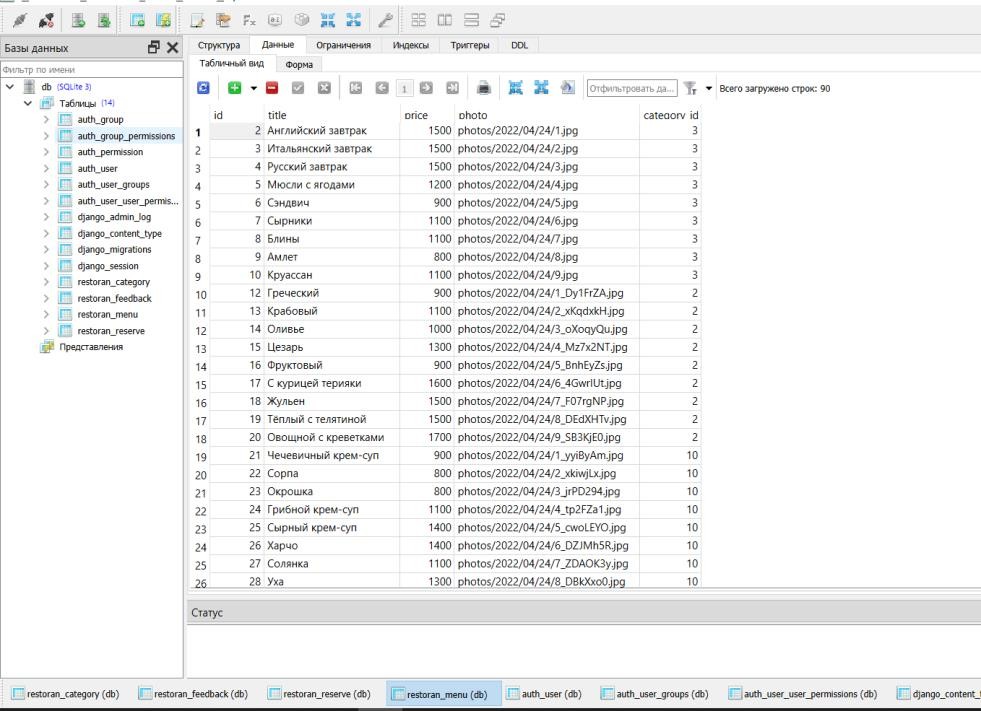
данных хранятся в файле *models* в Django, данные хранятся в СУБД SQLite Studio в виде таблиц, таблицы связаны между собой. Также редактировать данные можно в панели администратора и в СУБД. При заполнении форм пользователем, данные также отправляются в базу данных. После манипуляций с данными, они поступают на внешний уровень в User Interface, где доступ к ним имеет только администратор.



# Рисунок-3.3 – Архитектура SQLite

# C:\Users\sulta\AppData\Local\Microsoft\Windows\INetCache\Content.Word\архитектура.png

# Рисунок-3.4 – Технологии для разработки проекта



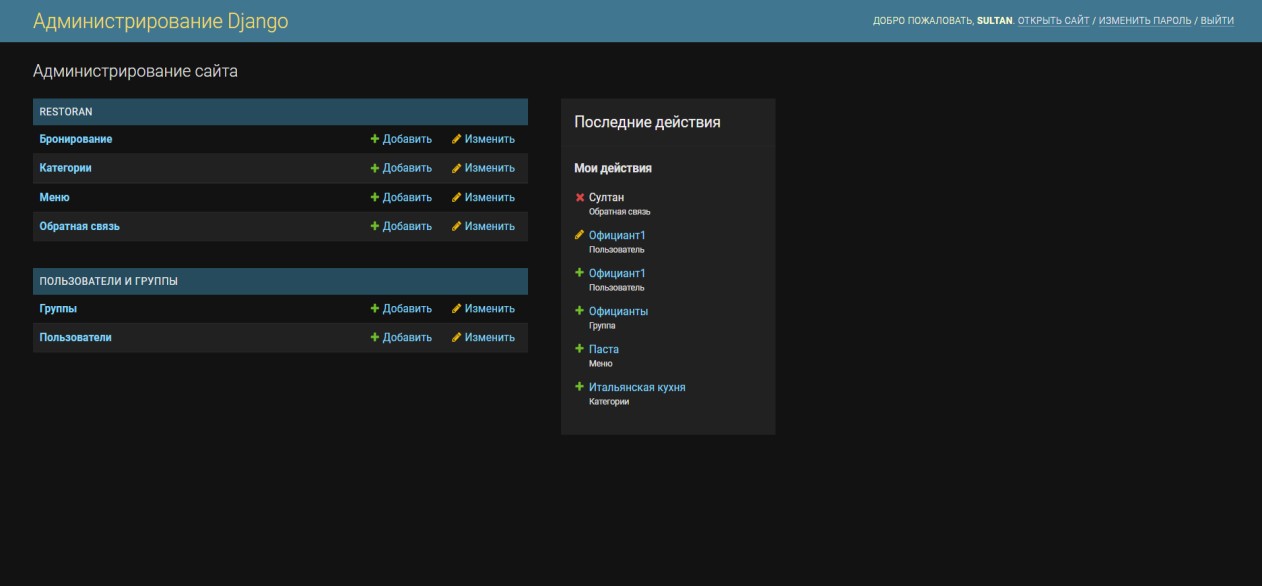
**Рисунок-3.5 – Таблицы с данными в SQLite**

# Авторизация

Авторизация на сайте осуществляется только рабочим персоналом предприятия. Для того, чтобы перейти на страницу авторизации необходимо добавить в поле адресации страницы слово “admin”. Одним из главных преимуществ фреймворка Django является встроенная система аутентификации и авторизации. Она позволяет проверять пользователей и определяет, какой пользователь может выполнить те или иные действия. Аутентификация подключается автоматически при создании скелета сайта. Создание пользователей и групп осуществляется через страницу административную панель.



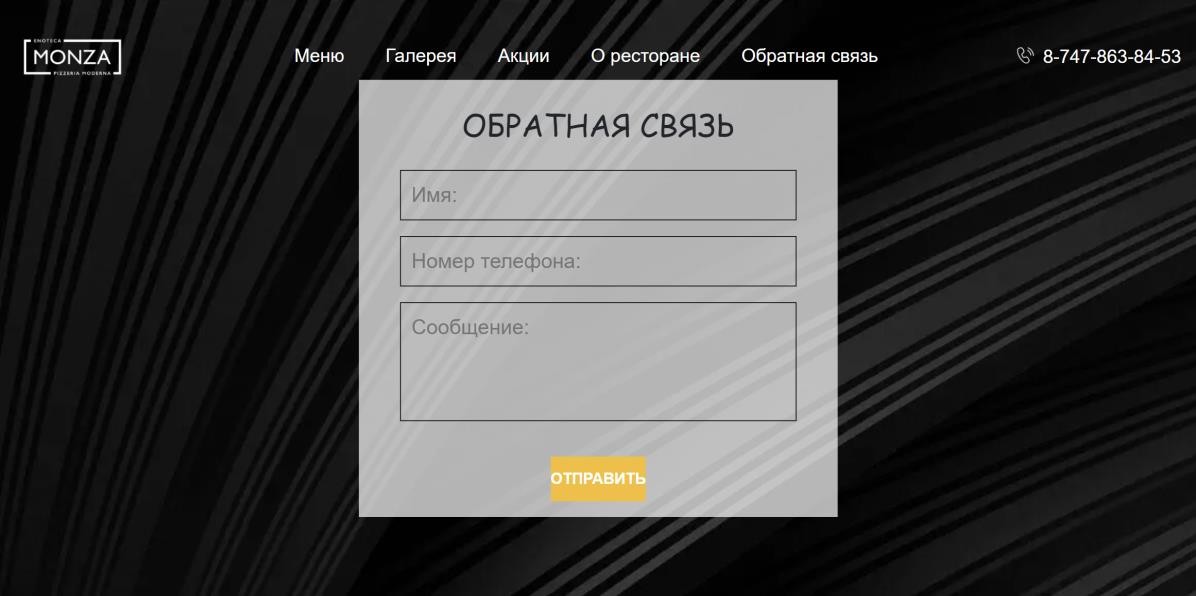
# Рисунок-3.6 – Авторизация Django



**Рисунок-3.7 – Панель администратора**

# User interface

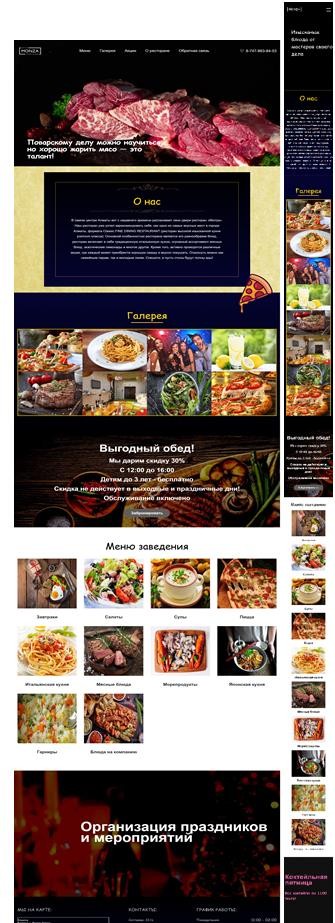
Пользовательский интерфейс разрабатываемой системы обеспечивает обмен информации между пользователем и работающим персоналом, а также, предоставляет пользователю необходимую информацию. На сегодняшний день большинство пользователей различных сайтов посещают их с помощью мобильных телефонов. Поэтому, в ходе работы над проектом были разработаны различные версии платформы под разные размеры экранов с помощью медиа-запросов в Css. В итоге сайт будет корректно отображаться как на компьютерах с маленьким разрешением монитора, так и на планшетах и мобильных устройствах. На рисунке ниже представлены первая страница сайта, которую User встречает при открытии, также вид на мобильных устройствах, отправка заявки на бронирование и страница с обратной связью:



# Рисунок-3.8 – Обратная связь



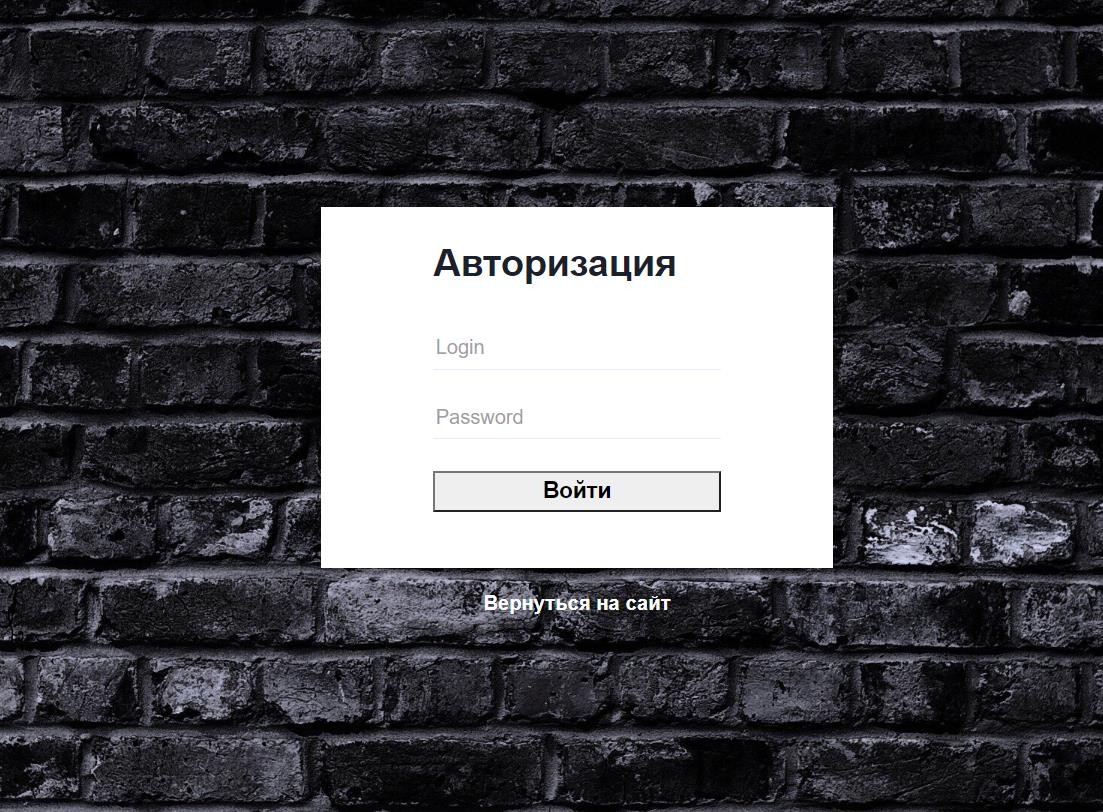
**Рисунок-3.9 – Заявка на бронирование**



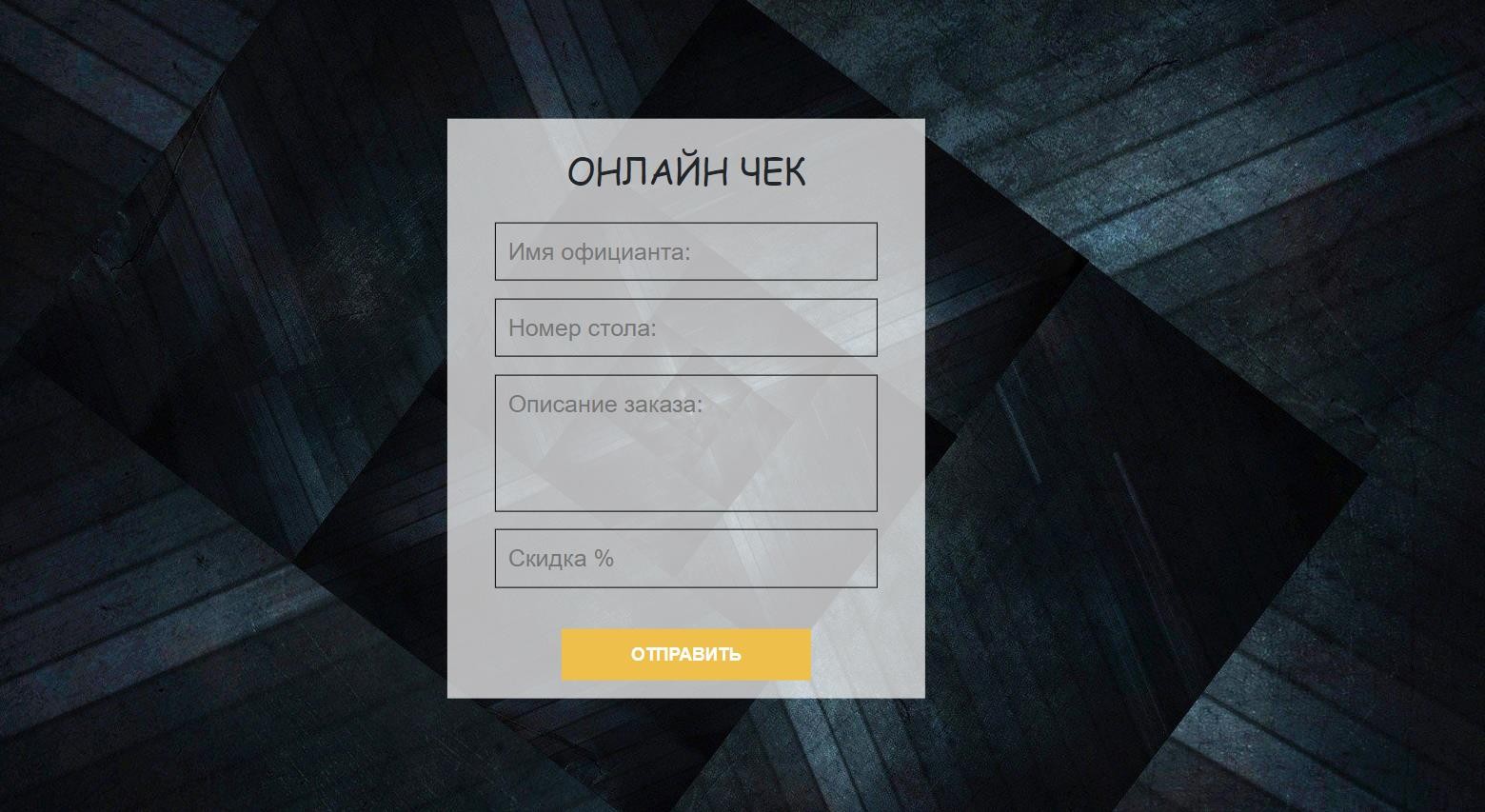
# Рисунок-3.10 – Главная страница

* + 1. **Функционал персонала**

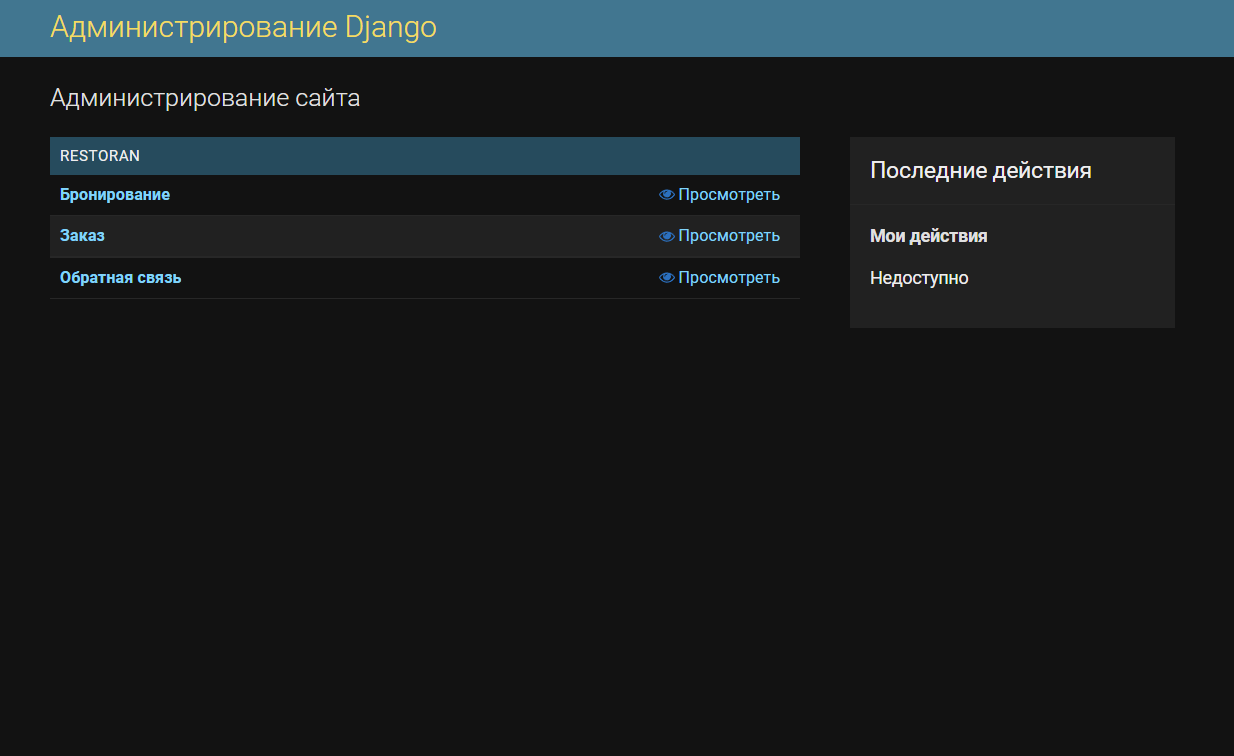
Из функций доступных персоналу – отдельная авторизация и создание онлайн чека. Авторизация доступна на отдельной странице. При успешном входе, всплывает форма отправки онлайн чека с указанием: номера стола, имени официанта, описанием заказа и наличием скидки. Далее форма отправляется в базу. На странице с данными персоналу доступны: просмотр отзывов, информация о бронировании и просмотр отправленных чеков.



# Рисунок-3.11 – Авторизация персонала



**Рисунок-3.12 – создание онлайн чека**



**Рисунок-3.13 – Панель официанта**

**ЗАКЛЮЧЕНИЕ**

В результате выполнения дипломной работы была создана готовая к использованию Web-платформа ресторана для информационной среды и рабочего персонала предприятия. Сайт предназначен для широкого круга пользователей, также присутствует функционал, отдельно предназначенный для рабочего персонала. При использовании сайта у пользователей будут возможности: просматривать актуальную информацию заведения, отправлять заявку на бронирование и отправка обратной связи для улучшения рейтинга ресторана. Возможности персонала: добавление новых сотрудников, оформление заказа, модерация заявок клиентов и редактирование меню. В ходе выполнения дипломной работы: были исследованы виды информационных систем и возможности их интеграции, изучены различные среды разработки и способы их реализации, проведён сравнительный анализ рынка, использованы методы разработки и проектирования информационных систем с применением области Web-разработки. Все выше поставленные задачи были достигнуты. Применение full-stack разработки значительно улучшило знания в области Web-программирования, особенно в Back- end разработке, например - изучение и использование фреймворка Django, написанном на языке Python. С целью улучшения эффективности сайта и внедрения в рабочую сферу, возможна интеграция новых функций, таких как: добавление личного кабинета для пользователя, введение доставки и электронных платёжных систем, а также переход с multi-page-application на single-page-application, что значительно улучшит быстродействие сайта. При акте внедрения сайта, в первую очередь необходимо использовать более многофункциональный фреймворк JavaScript, например - React, который подразумевает компонентный подход, расширить функционал и залить файлы на хостинг.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Изучение понятия информационных систем предприятий // Электронная версия на сайте <https://bstudy.net/674423/turizm/informatsionnye_sistemy_menedzhmenta_predpriyatiy_restorannogo_biznesa>
2. Классификация информационных систем // Электронная версия на сайте <https://fossdoc.com/ru/klassifikacija-informacionnyh-sistem>
3. Изучение информационных систем // Электронная версия на сайте <https://studme.org/282474/informatika/osnovnye_ponyatiya#847>
4. Анализ и сравнение информационных систем ресторанного бизнеса // Электронная версия на сайте

<https://bstudy.net/674423/turizm/informatsionnye_sistemy_menedzhmenta_predpriyatiy_restorannogo_biznesa#398>

1. Изучение системы R-keeper // Электронная версия на сайте <https://docs.rkeeper.ru/?roistat_visit=2759336>
2. Изучение системы 1С:Предприятие // Электронная версия на сайте https://v8.1c.ru/
3. Изучение системы ArchiDelivery // Электронная версия на сайте <http://www.archidelivery.ru/>
4. Изучение системы iiko // Электронная версия на сайте <https://ru.iiko.help/home/ru-ru/>
5. Документация Gulp // Электронная версия на сайте

<https://gulpjs.com/>

1. Анализ Back-end Фреймворков // Электронная версия на сайте:

<https://habr.com/ru/company/ruvds/blog/519478/>

1. Django введение // Электронная версия на сайте: <https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction>
2. Описание фреймворка Django // Электронная версия на сайте: <https://tutorial.djangogirls.org/ru/django/>
3. Документация Django // Электронная версия на сайте: <https://www.djangoproject.com/>
4. SQLite Studio tutorial // Электронная версия на сайте: <https://progtips.ru/bazy-dannyx/menedzher-baz-dannyx-sqlitestudio.html>
5. Анализ баз данных // Электронная версия на сайте: <https://drach.pro/blog/hi-tech/item/145-db-comparison>
6. Составление правильной архитектуры проекта // Электронная версия на сайтах:

<https://sales-generator.ru/blog/arkhitektura-sayta/>

<https://tproger.ru/translations/web-architecture-101/>

1. Требования и правила UI/UX // Электронная версия на сайтах: <https://www.purrweb.com/ru/blog/pravilnyj-podhod-k-sozdaniyu-ux-ui-dizajna-dlya-startapa/>

<https://ux.pub/editorial/10-zolotykh-pravil-ui-dizaina-3j5o>

1. UML-диаграммы // Электронная версия на сайте:

<https://coderlessons.com/tutorials/akademicheskii/uchit-uml/uml-diagrammy->[deiatelnosti](https://coderlessons.com/tutorials/akademicheskii/uchit-uml/uml-diagrammy-deiatelnosti)

# Приложение

# Текст программы

// код созданных моделей:

from django.db import models

from django.core.validators import MinValueValidator, MaxValueValidator

class feedback(models.Model):

phone\_number = models.CharField(max\_length=50,verbose\_name='Номер телефона')

name = models.CharField(max\_length=100,verbose\_name='Имя')

comment = models.CharField(max\_length=100, verbose\_name='Коментарий')

class Meta:

verbose\_name = 'Обратная связь' verbose\_name\_plural ='Обратная связь'

def str (self): return self.name

class reserve(models.Model):

phone\_number = models.CharField(max\_length=50, verbose\_name='Номер телефона')

name = models.CharField(max\_length=100, verbose\_name='Имя') amount = models.IntegerField(verbose\_name='Количество')

time\_create = models.DateTimeField(auto\_now\_add=True, verbose\_name='Время создания')

date = models.DateTimeField(verbose\_name='Время бронирования')

class Meta:

verbose\_name = 'Бронирование' verbose\_name\_plural = 'Бронирование'

def str (self):

return self.name

class category(models.Model):

category\_name = models.CharField(max\_length=100,verbose\_name='Название категории')

class Meta:

verbose\_name = 'Категории' verbose\_name\_plural ='Категории'

def str (self):

return self.category\_name

class menu(models.Model):

title = models.CharField(max\_length=100, verbose\_name='Название') price = models.IntegerField(verbose\_name='Цена')

category = models.ForeignKey(category, on\_delete= models.PROTECT,verbose\_name='Категория')

photo = models.ImageField(upload\_to="photos/%Y/%m/%d/",verbose\_name='Фотография',bla nk=True)

class Meta:

verbose\_name = 'Меню' verbose\_name\_plural = 'Меню'

def str (self): return self.title

class zakaz(models.Model):

name = models.CharField(max\_length=100, verbose\_name='Имя') numberofstola = models.IntegerField(verbose\_name='Номер стола') chek = models.CharField(max\_length=100, verbose\_name='Чек') skidka = models.CharField(max\_length=100, verbose\_name='Скидка')

time\_create = models.DateTimeField(auto\_now\_add=True, verbose\_name='Время создания')

class Meta: verbose\_name = 'Заказ'

verbose\_name\_plural = 'Заказ'

def str (self): return self.name

// Созданные представления view:

from unicodedata import category from django.shortcuts import render, redirect

from django.http import HttpResponse from .models import \*

from django.contrib.auth.models import User from django.contrib.auth import authenticate, logout from django.contrib.auth import login as auth\_login

from rest\_framework.views import APIView

class ModelView(APIView):

def get(self,request,\*args,\*\*kwargs):

return render(request, 'restoran/feedback.html', {'title': 'Отзывы'})

def post(self,request,\*args,\*\*kwargs): name = request.POST.get('name')

phone\_number = request.POST.get('tel') comment = request.POST.get('comment')

new\_model = feedback() new\_model.name = name

new\_model.phone\_number = phone\_number new\_model.comment = comment new\_model.save()

return render(request, 'restoran/feedback.html', {'title': 'Отзывы'})

class FormOtpr(APIView):

def get(selfself,request,\*args,\*\*kwargs):

return render(request, 'restoran/monza.html', {'title': 'О сайте'})

def post(self,request):

phone = request.POST.get('phone') name = request.POST.get('username') amount = request.POST.get('person') date = request.POST.get('trip-start') new\_model = reserve() new\_model.phone\_number = phone new\_model.name = name new\_model.amount = amount new\_model.date = date new\_model.save()

return render(request, 'restoran/monza.html', {'title': 'О сайте'})

def men(request):

menus = menu.objects.all() categoryes = category.objects.all()

return render(request, 'restoran/menu.html',{'title': 'Меню', 'menus':menus , 'categoryes':categoryes})

def cat(request, id):

menus = menu.objects.filter(category=id) categoryes = category.objects.all()

return render(request, 'restoran/menu.html', {'title': 'Меню','menus': menus, 'categoryes':categoryes})

class auth(APIView):

def get(self, request , \*args , \*\*kwargs): return render(request, 'restoran/login.html')

def post(self, request, \*args, \*\*kwargs): username = request.POST.get('username') password = request.POST.get('password')

user = authenticate(request, username=username, password=password)

print(user if user else 'oshibka')

print(authenticate(request, username=username, password=password)) if user is not None:

auth\_login(request, user) return redirect('officiant')

else:

return redirect('auth') class oficiant(APIView):

def get(selfself,request,\*args,\*\*kwargs):

return render(request, 'restoran/oficiant.html')

def post(self,request):

name = request.POST.get('name') number = request.POST.get('number') chek = request.POST.get('zakaz') skidka = request.POST.get('discount')

new\_model = zakaz() new\_model.name = name new\_model.numberofstola = number new\_model.chek = chek new\_model.skidka = skidka new\_model.save()

return render(request, 'restoran/oficiant.html')