Министерство образования Иркутской области
Государственное бюджетное профессиональное
образовательное учреждение Иркутской области
«Иркутский авиационный техникум»
(ГБПОУИО «ИАТ»)

ПП.09.02.07-1.25.221.03

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Руководитель от предприятия: _____ (А.З. Зайцева)
(подпись, дата)

М.П.

Руководитель от техникума: _____ (Е.С. Фролова)
(подпись, дата)

Студент: _____ (А.А. Березин)
(подпись, дата)

Иркутск 2025

Содержание

Введение

Веб-приложение "Ресторан LUXE" – это комплексная платформа, предоставляющая клиентам ресторана доступ к полному спектру услуг через современный цифровой интерфейс. Приложение позволяет просматривать меню, бронировать столики, заказывать банкеты и мероприятия, а также взаимодействовать с рестораном через удобный и элегантный интерфейс. В современном мире, где цифровые технологии становятся неотъемлемой частью ресторанного бизнеса, подобные решения пользуются большим спросом среди заведений премиум-класса, стремящихся обеспечить высочайший уровень сервиса и удобства для своих гостей.

Разработка качественной системы для пользователей рестораном – это сложная задача, требующая внимания к нескольким ключевым аспектам:

− Комплексный функционал: успешная система должна содержать все необходимые модули для управления рестораном, что требует продуманной архитектуры, интеграции различных компонентов и обеспечения их бесперебойной работы.

− Удобный и интуитивный интерфейс: приложение должно быть простым в использовании как для гостей ресторана, позволяя легко находить нужную информацию, делать бронирования и управлять заказами.

Преимущества веб-приложения:

− Централизованный доступ к услугам: клиенты могут быстро ознакомиться с меню, забронировать столик и заказать мероприятие без необходимости звонить в ресторан или посещать его лично.

− Автоматизация процессов: система автоматически обрабатывает бронирования, генерирует коды подтверждения, что снижает нагрузку на персонал и минимизирует человеческие ошибки.

− Повышение имиджа заведения: современный и стильный веб-интерфейс подчеркивает статус ресторана премиум-класса.

Цель производственной практики: разработка функционального веб-приложения на Django, которое обеспечит полноценное цифровое присутствие ресторана LUXE и позволит клиентам удобно взаимодействовать с заведением через интернет.

Основные задачи:

- Анализ предметной области: изучение специфики ресторанного бизнеса и определение ключевых сущностей для базы данных (меню, бронирования, события, клиенты и т.д.).
- Выбор инструментов разработки: обоснование использования Django, PostgreSQL, HTML/CSS/JavaScript и дополнительных библиотек для создания современного веб-приложения.
- Проектирование базы данных: создание ER-диаграммы и схемы таблиц с учетом связей между сущностями ресторанного бизнеса.

Разработка backend-части:

- Настройка моделей Django для хранения данных о меню, бронированиях и событиях.
- Реализация CRUD-функционала для управления контентом.

Разработка frontend-части:

- Верстка шаблонов с адаптивным дизайном, отражающим премиум-статус ресторана.
- Добавление интерактивных элементов (формы бронирования, событий).
- Тестирование: проверка работоспособности всех функций, включая обработку форм бронирования, отображение меню и валидацию пользовательского ввода.
- Документирование: описание архитектуры проекта, руководство для пользователей и администраторов системы.

Предпроектное исследование

1.1Описание предметной области

Областью применения веб-платформы «Ресторан LUXE» является создание централизованной цифровой экосистемы для управления всеми аспектами деятельности ресторана премиум-класса и обеспечения высококачественного сервиса для гостей.

Современная ресторанная индустрия требует комплексного подхода к цифровизации бизнес-процессов и клиентского сервиса. Ресторан LUXE, позиционирующийся как заведение высокой кухни, должен предоставлять своим гостям не только изысканные блюда, но и безупречный цифровой опыт взаимодействия. Система включает множество взаимосвязанных компонентов: меню с категориями блюд, систему бронирования столиков, организацию банкетов и мероприятий.

Веб-платформа предназначена для автоматизации следующих процессов:

− Система бронирования: автоматизация процесса резервирования столиков с выбором даты, времени и количества гостей;

− Организация мероприятий: прием заявок на банкеты, корпоративы и частные события;

− Административные функции: управление пользователями, мониторинг бронирований и аналитика;

Пользователи системы являются:

− Гости ресторана — посетители, использующие платформу для ознакомления с меню, бронирования столиков и заказа мероприятий;

− Администраторы системы — технические специалисты, управляющие настройками платформы, правами доступа и обеспечивающие стабильную работу системы;

Основные функции модуля:

- Интерактивное меню с детальными описаниями блюд;
- Система онлайн-бронирования с календарем доступности;
- Модуль организации мероприятий;
- Панель администрирования для управления контентом;

Требования к системе включают в себя:

- Производительность: способность обрабатывать множественные одновременные бронирования, быстрая загрузка страниц и оптимизация для пиковых нагрузок.
- Надежность: гарантированная сохранность данных о бронированиях, устойчивость к сбоям.
- Безопасность: система администрирования.

1.2 Обзор инструментальных средств для разработки

Выбор инструментов разработки напрямую влияет на качество и эффективность создания программного продукта.

Для проектирования архитектуры веб-приложения удобно использовать PostgreSQL (для работы с базой данных) и Draw.io (для визуализации структуры), а проектировать интерфейсы – в Figma.

Приложение будет разделено на клиентскую и серверную части. Клиентскую сторону можно разрабатывать в PyCharm Community Edition, а серверную – с использованием стандартной СУБД PostgreSQL.

Такой набор инструментов обеспечит удобство разработки и надежность будущего продукта.

PostgreSQL – это мощная объектно-реляционная система управления базами данных (СУБД). Это самостоятельная СУБД с расширенными функциями, высокой производительностью и поддержкой сложных запросов.

Draw.io – это онлайн-инструмент для создания схем, диаграмм и блок-схем. Это бесплатное приложение, которое широко используется для визуализации процессов, архитектуры систем, ER-диаграмм баз данных и других видов диаграмм.

Figma – это онлайн-платформа для дизайна интерфейсов и прототипирования. Она позволяет создавать интерфейсы приложений, веб-сайтов и других цифровых продуктов. Создание макетов интерфейсов, прототипов и взаимодействий между элементами, командная работа в реальном времени. Поддерживает векторную графику и работу с компонентами, что удобно для разработки интерфейсов.

PyCharm Community Edition – это бесплатная интегрированная среда разработки (IDE) для языка программирования Python. Она разработана компанией JetBrains и предоставляет все необходимые базовые инструменты для написания, отладки и запуска кода.

Python – это универсальный язык программирования, широко используемый в различных областях, таких как веб-разработка, анализ данных, машинное обучение, разработка программного обеспечения и многое другое. Он известен своей простотой

и читабельностью, что делает его отличным выбором как для начинающих, так и для опытных разработчиков.

Мобильное приложение будет содержать в себе информацию её необходимо хранить, изменять, структурировать и использовать. Это реализуется благодаря базе данных. Были рассмотрены следующие варианты реализации СУБД:

- MySQL;
- MongoDB;
- PostgreSQL.

MySQL – это реляционная система управления базами данных (СУБД), разработанная в 1995 году и ныне принадлежащая Oracle. Реляционные СУБД основаны на моделировании данных в виде таблиц (сущностей) с определенными взаимосвязями между ними. В MySQL данные организованы в виде таблиц с строками и столбцами, а также использует SQL (Structured Query Language) для создания, чтения, обновления и удаления данных (CRUD-операции). SQL предоставляет богатый набор команд для работы с данными и управления структурой базы данных.

MySQL использует архитектуру клиент-сервер, где сервер обрабатывает запросы от клиентов и возвращает им результаты. Это позволяет использовать MySQL в распределенных системах, где приложения взаимодействуют с сервером базы данных по сети.MySQL предлагает несколько движков хранения, такие как InnoDB (с поддержкой транзакций и внешних ключей) и MyISAM (более быстрый, но без поддержки транзакций).

MongoDB – это система управления базами данных NoSQL, разработанная в 2009 году компанией MongoDB Inc. В отличие от реляционных СУБД, MongoDB хранит данные в виде документов BSON (бинарная версия JSON), что позволяет создавать сложные и вложенные структуры данных.

Документы в MongoDB организованы в коллекции (аналог таблиц в реляционных базах данных). Каждый документ представляет собой JSON-подобную структуру, состоящую из пар «ключ-значение». Это позволяет хранить данные с разными структурами в одной коллекции.

MongoDB не требует заранее определенной схемы, что делает её гибкой и позволяет изменять структуру данных на лету. Это упрощает работу с изменяющимися данными и быстрое прототипирование. MongoDB использует собственный язык запросов, основанный на операторах и фильтрах для поиска, обновления и агрегации данных. Это позволяет выполнять сложные операции, такие как группировка и фильтрация данных.

PostgresSQL – это реляционная система управления базами данных с открытым исходным кодом, впервые выпущенная в 1989 году. Это одна из самых мощных СУБД, поддерживающая как традиционные реляционные структуры, так и современные расширения, такие как JSON и геопространственные данные.

PostgreSQL поддерживает широкий спектр возможностей SQL, таких как сложные запросы, подзапросы, оконные функции, общие табличные выражения (CTE) и агрегатные функции, что делает его подходящим для аналитических задач и работы с большими объемами данных.

PostgreSQL поддерживает пользовательские типы данных, хранимые процедуры, и даже хранение сложных структур данных, таких как JSONB (бинарный JSON). Это делает его полезным как для традиционных реляционных задач, так и для хранения неструктурированных данных. PostgreSQL поддерживает репликацию (синхронную и асинхронную), что позволяет создавать резервные копии и настраивать отказоустойчивость. Также поддерживает распределенные транзакции и параллельные запросы.

Для наглядности сравнения вариантов реализации базы данных была составлена таблица 1.

Таблица 1 - Сравнение средств реализации базы данных

Название БД	MySQL	MongoDB	PostgreSQL
Большое кол-во типов данных	+	-	+
Популярность	+	+	+

Отказоустойчивость	+	+	+
Гибкость в моделировании данных	-	+	-
Не требует строгой схемы	-	+	-
Поддержка ACID	+	-	+
Масштабируемость	+	+	+
Подходит для аналитики	-	-	+
Простота установки	+	+	-
Сообщество и документация	+	+	+

Таким образом, несмотря на то, что MySQL обладает преимуществами в простоте установки и скорости операций чтения, PostgreSQL лучше соответствует требованиям проекта по надежности, гибкости и функциональности. Это делает PostgreSQL разумным выбором для разработки масштабируемого, безопасного и функционально богатого веб-приложения для систематизации веб-приложения.

Для наглядности сравнения вариантов среди языков программирования была составлена таблица 2.

Таблица 2 - Сравнение языков программирования для разработки веб-приложения

Критерии	Python	Java	C#
Простота изучения	Высокая	Средняя	Средняя
Экосистема и библиотеки	Богатая, включая Django, Flask	Широкая, корпоративные фреймворки	Хорошо, интеграция с .NET

Производительность	Средняя	Высокая	Высокая
Разработка веб-приложений	Отличная	Широкая	Хорошая
Безопасность	Хорошая	Высокая	Хорошая

Python был выбран в качестве базового языка благодаря высокой простоте изучения и быстроте разработки, а также богатой экосистеме и мощной поддержке веб-приложений через фреймворк Django. Это обеспечивает эффективное создание надёжных, масштабируемых и легко сопровождаемых решений для систематизации веб-приложения.

PyCharm Community Edition — это бесплатная версия интегрированной среды разработки (IDE) PyCharm от компании JetBrains, предназначенная для инновационных разработчиков и небольших проектов на языке Python. Она предоставляет базовые инструменты для написания, отладки и тестирования кода на Python, включая поддержку таких языков и форматов, как HTML, XML, JSON, YAML и Markdown.

Android Studio — это официальная интегрированная среда разработки (IDE) для создания приложений под Android, разработанная компанией Google. Она основана на IntelliJ IDEA и предоставляет мощные инструменты для разработки, отладки и тестирования мобильных приложений.

IntelliJ IDEA — это мощная интегрированная среда разработки (IDE) от компании JetBrains, предназначенная для профессиональной разработки программного обеспечения на Java, Kotlin, а также поддерживающая множество других языков, включая JavaScript и Python. Она кроссплатформенная и работает на Windows, macOS и Linux.

Для наглядности сравнения вариантов среды разработки была составлена таблица 3.

Таблица 3 - Сравнение IDE сред для проектирования веб-приложения

Критерии	PyCharm Community Efition	Android Studio	IntelliJ IDEA
Поддержка Python	Отличная	Требует расширений	Хорошая
Удобство и функционал	Богатый функционал	Для мобильной разработки	Мощная
Бесплатно	Бесплатно	Бесплатно	Платная (есть бесплатная версия)

Для разработки на Python и работы с платформером Django наиболее распространенным является выбор среды PyCharm и Android Studio. Обеспечиваем бесплатную версию с полным набором инструментов для эффективного написания, отладки и тестирования кода. PyCharm выделяет мощный функционал, специально ориентированный на Python-разработчиков, включая интеллектуальное автодополнение и встроенный отладчик, что обеспечивает глубокую интеграцию с Python-экосистемой. Android Studio, в свою очередь, предлагает легкую и гибкую среду с поддержкой расширения Python, что делает ее удобной и быстрой для настройки. IntelliJ IDEA, хоть и мощная и универсальная IDE с поддержкой Python через плагины, является платной (за исключением ограниченной версии) и ориентирована больше на мультиязычную разработку. Таким образом, PyCharm и Android Studio предлагают бесплатные, функционально насыщенные и удобные решения для эффективной работы с Python и Django.

Для проведения анализа языков и средств разработки выбор был сделан с использованием языка Python и редактора PyCharm, поскольку они позволяют легко и удобно создавать веб-приложения.

1) Для визуального проектирования структуры базы данных можно использовать бесплатный и открытый инструмент pgAdmin (для PostgreSQL), который легко интегрируется с различными системами управления контентом (CMS) и позволяет эффективно управлять данными веб-приложений.

2) Для построения структурных схем, контекстных диаграмм и декомпозиции диаграмм применяются CASE-инструменты, такие как Draw.io, благодаря его простому и доступному понятному интерфейсу, доступности онлайн и удобству использования даже для пользователей без традиционных технических знаний.

3) Для создания прототипа веб-интерфейса был выбран онлайн-редактор Figma, который предоставляет бесплатную базовую версию, кроссплатформенность (работает в браузере и в приложении), большое количество шаблонов и возможность расширения функционала с помощью плагинов. Это позволяет быстро и эффективно разрабатывать дизайн и интерактивные прототипы веб-приложений.

Веб-приложения, разработанные с использованием Python и современных веб-фреймворков, обладают следующими преимуществами:

Удобство в использовании. Веб-приложения обеспечивают доступ к функциям и сервисам через любой браузер с адаптивным и стандартным интерфейсом.

Высокая производительность. Оптимизация серверной и клиентской части позволяет быстро обрабатывать запросы и обеспечивать стабильную работу приложения.

Персонализация. Использование данных пользователя, таких как предпочтения и поведение, позволяет создавать индивидуальные пользовательские опыты и рекомендации.

Таким образом, сочетание Python и PyCharm вместе с современными инструментами проектирования и прототипирования обеспечивает ресурс и качественную разработку современных веб-приложений.

2.Проектирование программного модуля

Одним из важнейших этапов разработки является проектирование диаграмм, которые помогут лучше понять структуру нашего приложения и работоспособность в целом.

Диаграмма вариантов использования, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы, благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

Диаграмма прецедентов для веб-сайта «Ресторан LUXE» будет включать одного актера – Гость. Прецеденты включают: просмотр меню, выбор блюд, бронирование столика.

На рисунке 1 изображена Use Case View, которая показывает структурную схему веб-приложения «Ресторан LUXE» для ролей «Пользователь», «Администратор».

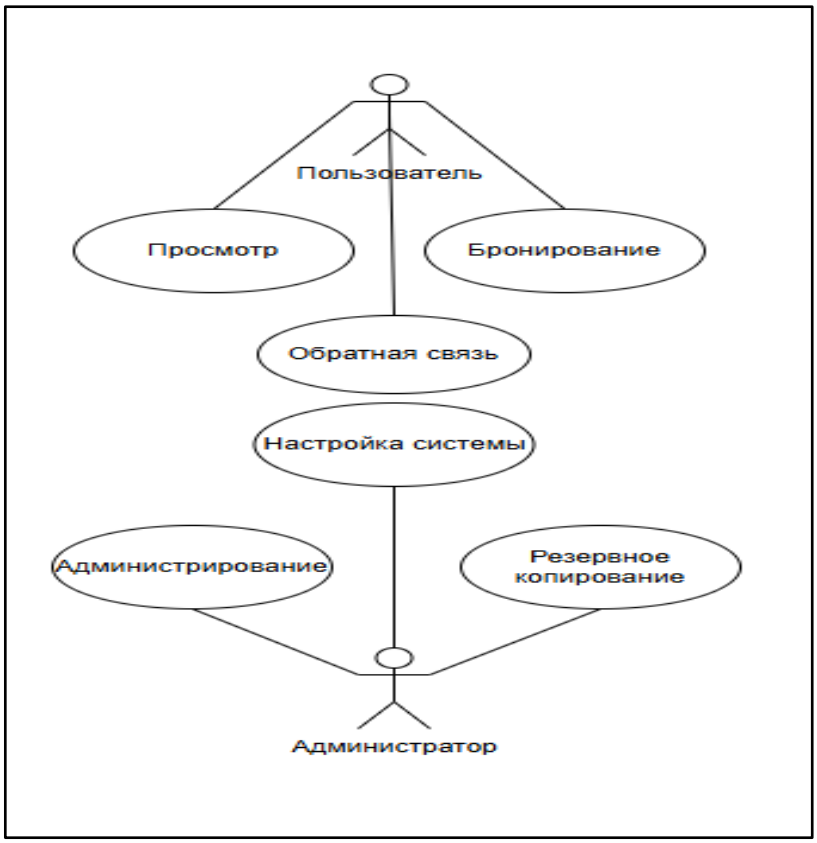

Рисунок 1 – Диаграмма прецедентов

Диаграмма активности представляет собой UML-схему, демонстрирующую операции и процессы, детали которых раскрываются в диаграммах состояний. Активность трактуется как описание выполняемого поведения через согласованное последовательное и одновременное осуществление подчиненных компонентов - встроенных типов активности и индивидуальных операций, связанных потоками данных, которые передаются от результатов одного элемента к входным параметрам другого.

Диаграммы активности применяются для моделирования рабочих процессов организации, производственных циклов, последовательных и параллельных операций обработки данных.

На рисунке 2 изображена диаграмма деятельности веб-приложения «Ресторан LUXE».

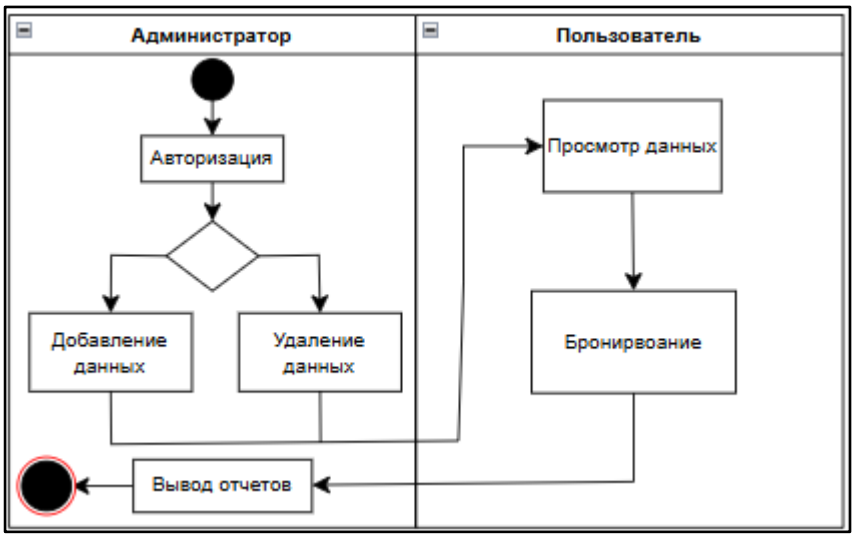

Рисунок 1 – Диаграмма деятельности

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

В итоге проектирования диаграммы деятельности были выделены основные возможные действия пользователя с программным продуктом.

2.2 Проектирование базы данных

Прежде чем приступить разработке программного обеспечения необходимо спроектировать базу данных, а именно, определить с какими данными будут работать пользователи веб-приложения, и чем данные связаны между собой. В этом заключается процесс проектирования.

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных.

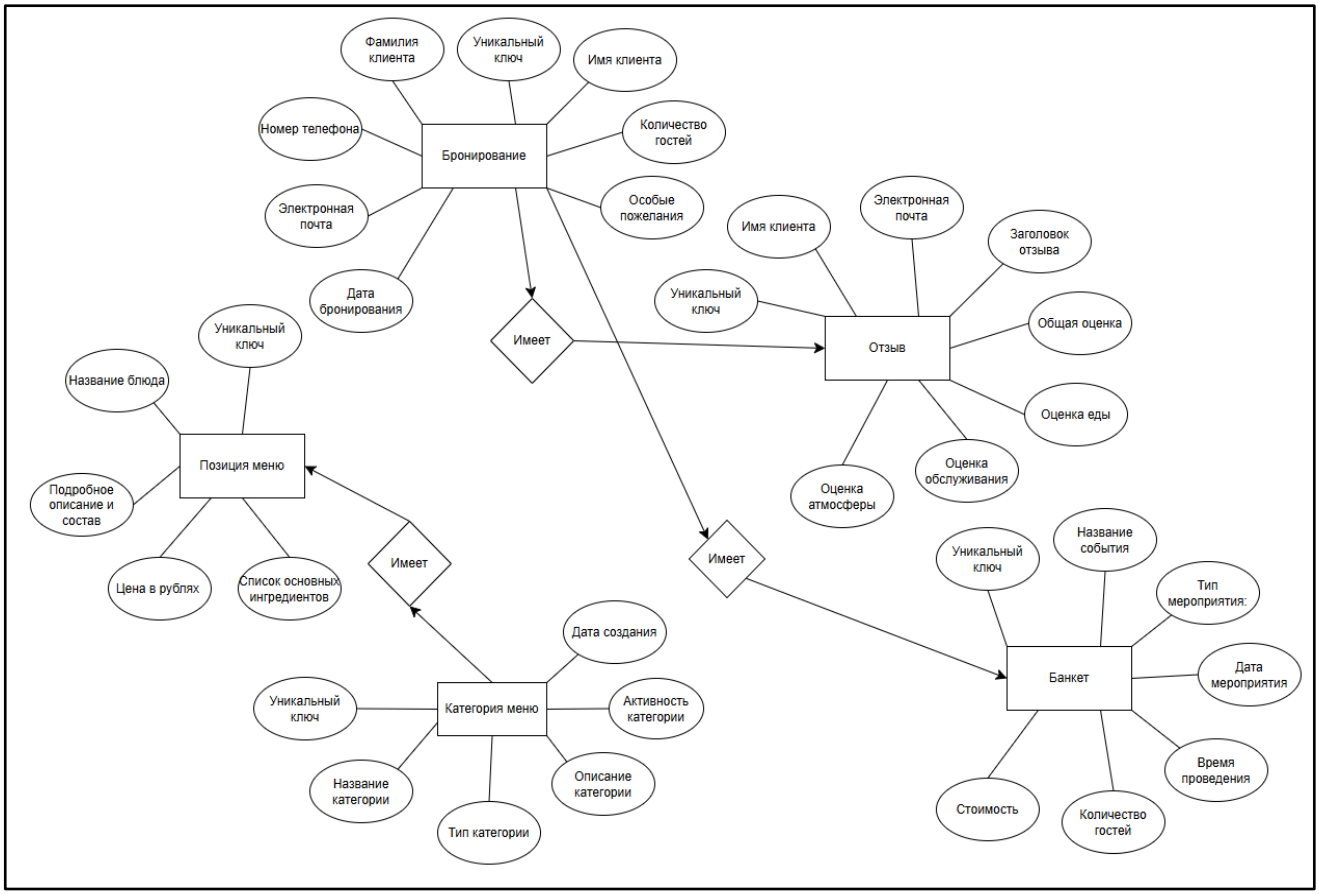

Рисунок 2 – Инфологическая модель

На представленной ER-диаграмме ресторана LUXE отображено пять основных сущностей, каждая из которых обладает своими уникальными атрибутами и связана с другими сущностями через отношения типа "Имеет", что в совокупности формирует инфологическую модель системы управления рестораном.

Центральной сущностью является "Бронирование" с атрибутами фамилии и имени клиента, уникального ключа, номера телефона, электронной почты, количества гостей, особых пожеланий и даты бронирования, которая связана с сущностью "Отзыв", содержащей информацию о клиентских оценках еды, обслуживания и атмосферы, а также с сущностью "Банкет" для организации специальных мероприятий с указанием названия события, типа, даты, времени проведения, количества гостей и стоимости. Дополнительно модель включает сущности "Категория меню" с атрибутами названия, типа, описания и активности категории, которая связана с сущностью "Позиция меню", содержащей детальную информацию о блюдах включая название, описание, состав, цену и список ингредиентов.

Проектирование данной базы данных представляет собой непрерывный процесс, протекающий на протяжении всего цикла разработки до момента появления критически важных данных, которые нельзя потерять, а результатом этого проектирования является ER-модель (Entity-Relationship model, модель «сущность-связь») — графическое представление структуры данных, используемое для проектирования баз данных и позволяющее наглядно описать сущности системы, их атрибуты и взаимосвязи между ними.

Определение ER-модели:

ER-модель представляет собой диаграмму, состоящую из сущностей (прямоугольников) и связей (линий) между ними. Каждая сущность имеет набор атрибутов, которые описывают её характеристики.

Описание ER-модели:

Сущности (Entities):

- Сущности представляют объекты в системе, которые имеют уникальные идентификаторы.

- Каждая сущность имеет набор атрибутов, которые определяют её характеристики.

- Примеры сущностей: блюда, бронирование.

Атрибуты (Attributes):

- Атрибуты описывают характеристики сущностей.

- Каждый атрибут имеет имя и тип данных (например, строка, число, дата).

- Примеры атрибутов: имя пользователя, email, password.

Связи (Relationships):

- Связи определяют отношения между сущностями.

- Связи могут быть одного-к-одному (1:1), одного-ко-многим (1:N) или многие-ко-многим (M:N).

- Примеры связей: один пользователь может быть добавлен в несколько категорий.

Ключи (Keys):

- Ключи используются для идентификации сущностей.

- Первичный ключ уникально идентифицирует каждую сущность в базе данных.

- Внешний ключ связывает сущности через связи.

Ограничения (Constraints):

- Ограничения определяют правила для данных в базе данных.

- Примеры ограничений: пользователь должен иметь уникальный идентификатор, блюдо должен содержать информацию.

ER-модель используется для проектирования баз данных. Она помогает определить структуру данных и связи между ними. ER-модель может быть преобразована в физическую модель базы данных, которая будет использоваться для хранения и обработки данных.

ER-модель позволяет:

- Определить структуру данных.

- Определить связи между данными.

- Определить правила для данных.

- Создать физическую модель базы данных.

Она является важным инструментом для проектирования баз данных, который помогает обеспечить эффективность и надёжность хранения данных.

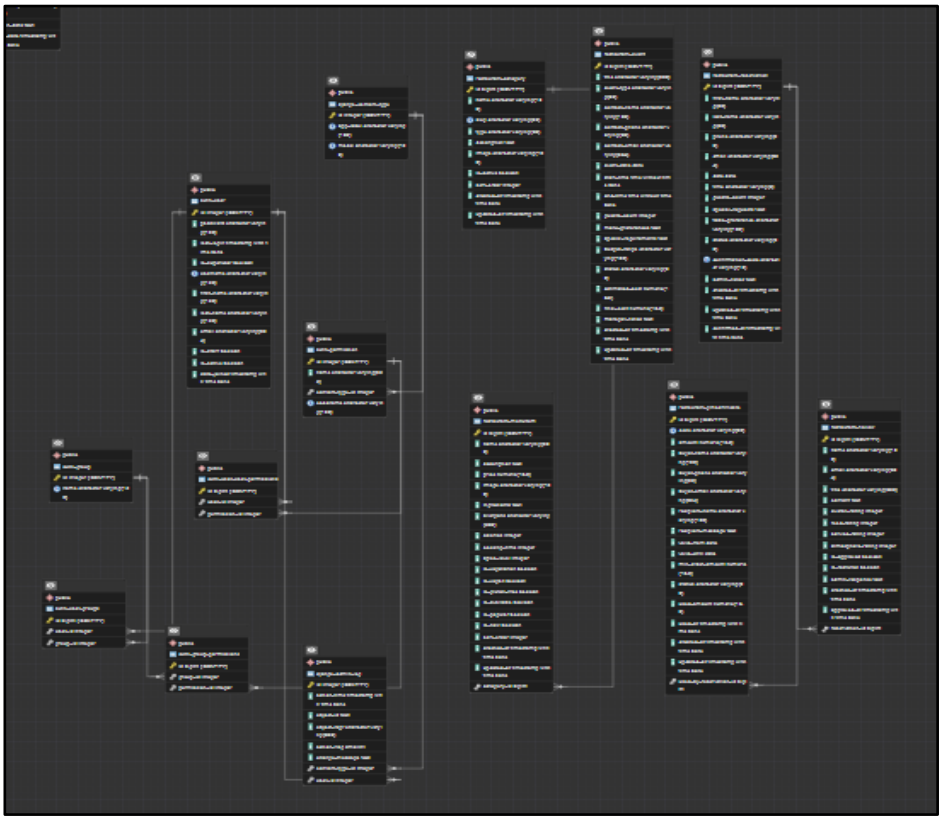

Рисунок 4 – ER-модель базы данных

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

Таблица 1 - Таблица «Category»

Поле	Тип данных	Описание
id	Int	Идентификатор
name	Char(100)	Название категории
type	Char(20)	Тип категории
description	Text	Описание категории
is_active	Char(20)	Активность
created_at	DateTime	Дата создания

Таблица 2 - Таблица «MenuItem»

Поле	Тип данных	Описание
id	Int	Идентификатор
name	Char(200)	Название блюда
description	Text	Описание
price	Decimal(10,2)	Цена
ingredients	Text	Список ингредиентов

Таблица 3 - Таблица «Reservations»

Поле	Тип данных	Описание
id	Int	Идентификатор
first_name	Char(50)	Имя клиента
last_name	Char(50)	Фамилия клиента
phone	Char(20)	Номер телефона
email	Char(50)	Электронная почта
date	Date	Дата бронирования
time	Char(5)	Время бронирования
guests_count	Int	Количество гостей
special_requests	Text	Особые пожелания

Таблица 4 - Таблица «Review»

Поле	Тип данных	Описание
id	Int	Идентификатор
name	Char(100)	Имя клиента
email	Char(100)	Email клиента
title	Char(200)	Заголовок отзыва
overall_rating	Int	Общая оценка
food_rating	Int	Оценка еды
service_rating	Int	Оценка обслуживания
atmosphere_rating	Int	Оценка атмосферы

Таблица 5 - Таблица «Event»

Поле	Тип данных	Описание
id	Int	Идентификатор
title	Char(200)	Название события
event_type	Char(20)	Тип события
guests_count	Int	Количество гостей
final_cost	Decimal(10,2)	Стоимость
event_date	Date	Дата события
start_time	Time	Время

База данных была приведена ко второй нормальной форме, но перед этим прошла этап: нормализацию до первой нормальной формы и нормализацию до второй нормальной формы.

Нормальная форма – это свойство отношения в реляционной модели данных, которое характеризует его с точки зрения избыточности, способной привести к логическим ошибкам при выборке или изменении данных. Нормальная форма определяется набором требований, которым должно соответствовать отношение. Существует шесть нормальных форм, и наша база данных достигла второй.

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

Для достижения второй нормальной формы необходимо, чтобы таблица уже соответствовала первой нормальной форме, а также чтобы таблицы были связаны

между собой. Это предотвращает дублирование данных: если записи повторяются в разных таблицах, потребуется изменять информацию в обеих таблицах, что нежелательно.

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

2.2Проектирование интерфейса

Проектирование пользовательского интерфейса — это процесс создания удобного и интуитивно понятного интерфейса для взаимодействия пользователя с продуктом или системой.

Интуитивно понятный дизайн: интерфейс должен быть интуитивно понятным, чтобы пользователи могли легко ориентироваться в нём и выполнять нужные действия без затруднений.

Логика и последовательность: элементы интерфейса должны быть расположены логично и последовательно. Это поможет пользователям быстро находить нужную информацию и выполнять действия.

На рисунке 5 представлена главная страница веб-приложения.

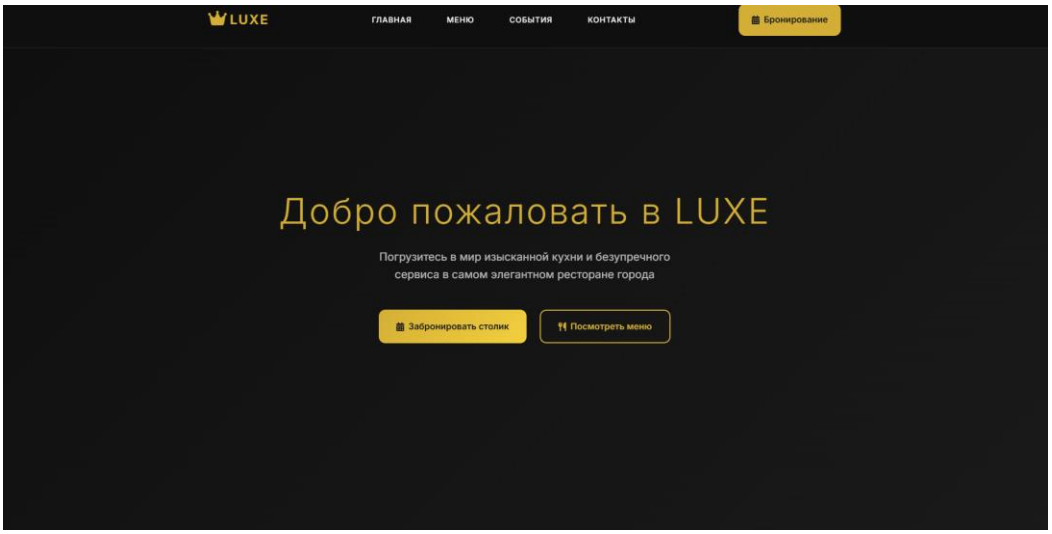

Рисунок 5 – Главная страница сайта

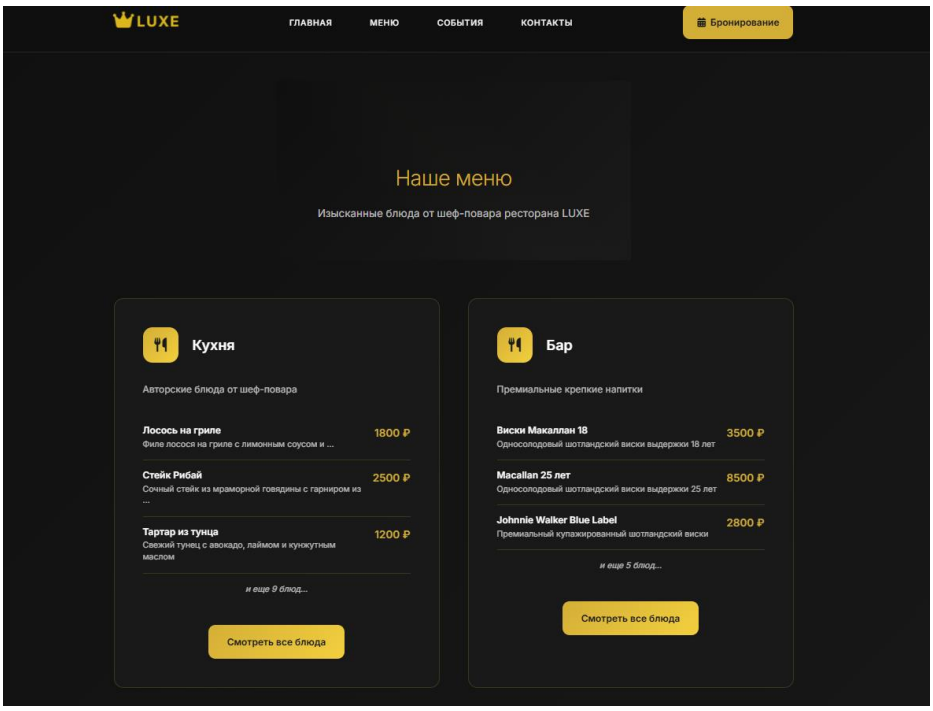

Рисунок 6 – Страница меню

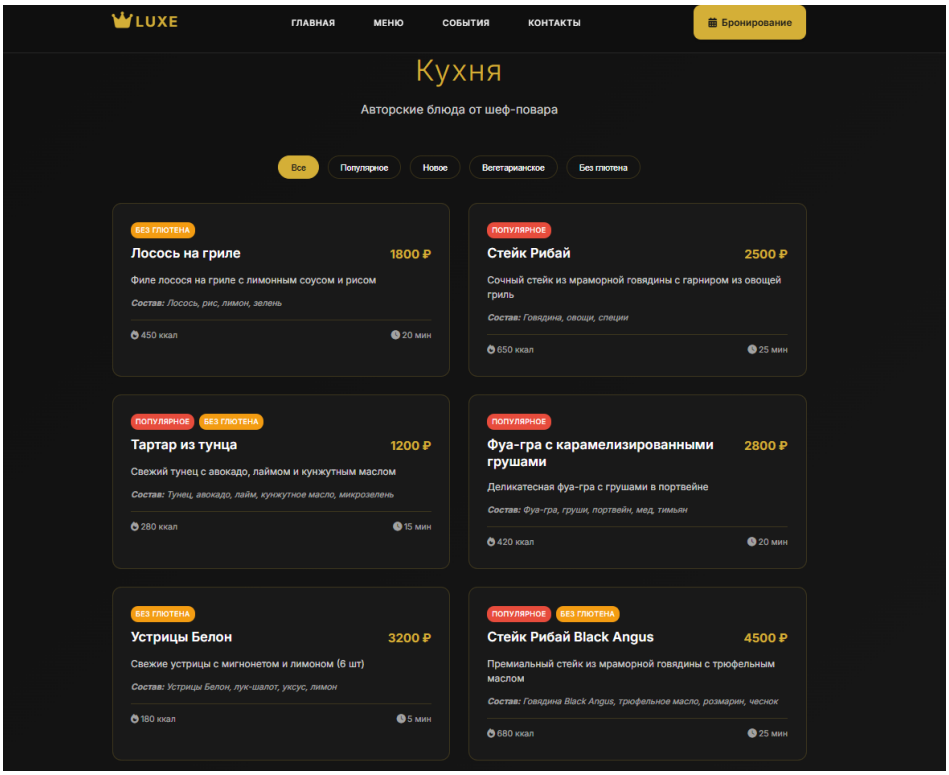

Рисунок 7 – Страница блюд ресторана

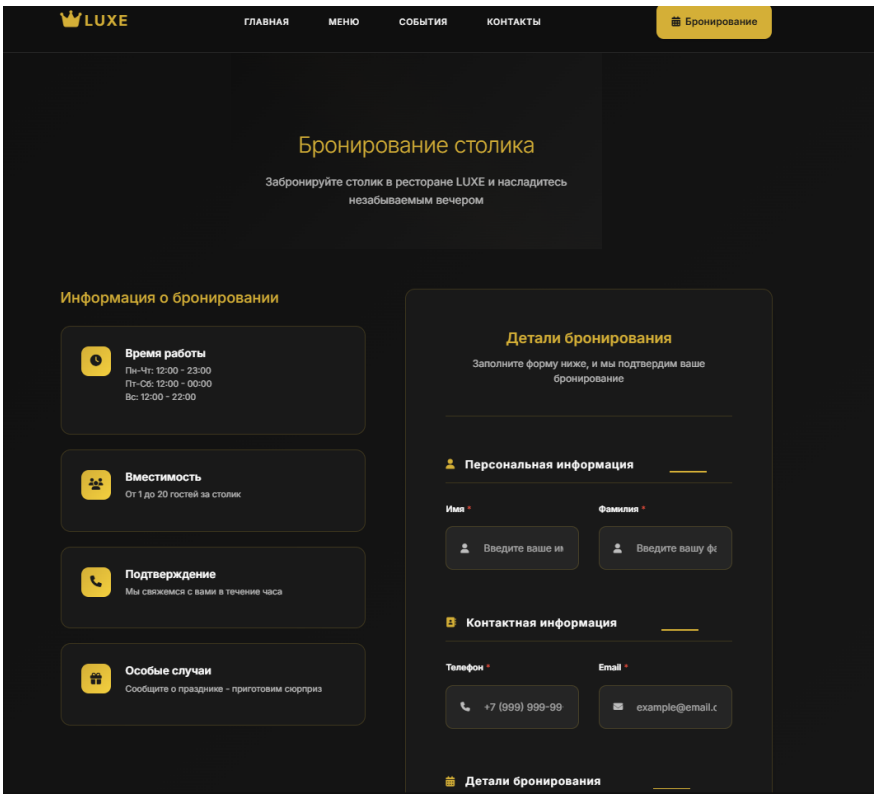

Рисунок 8 – Страница бронирования

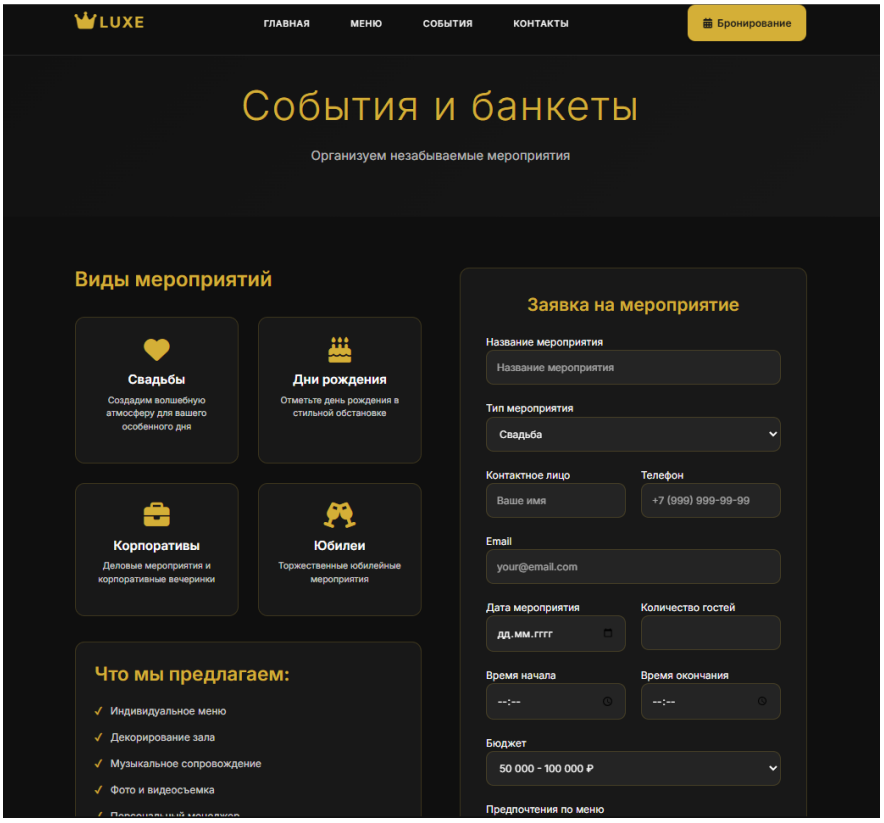

Рисунок 9 – Страница событий

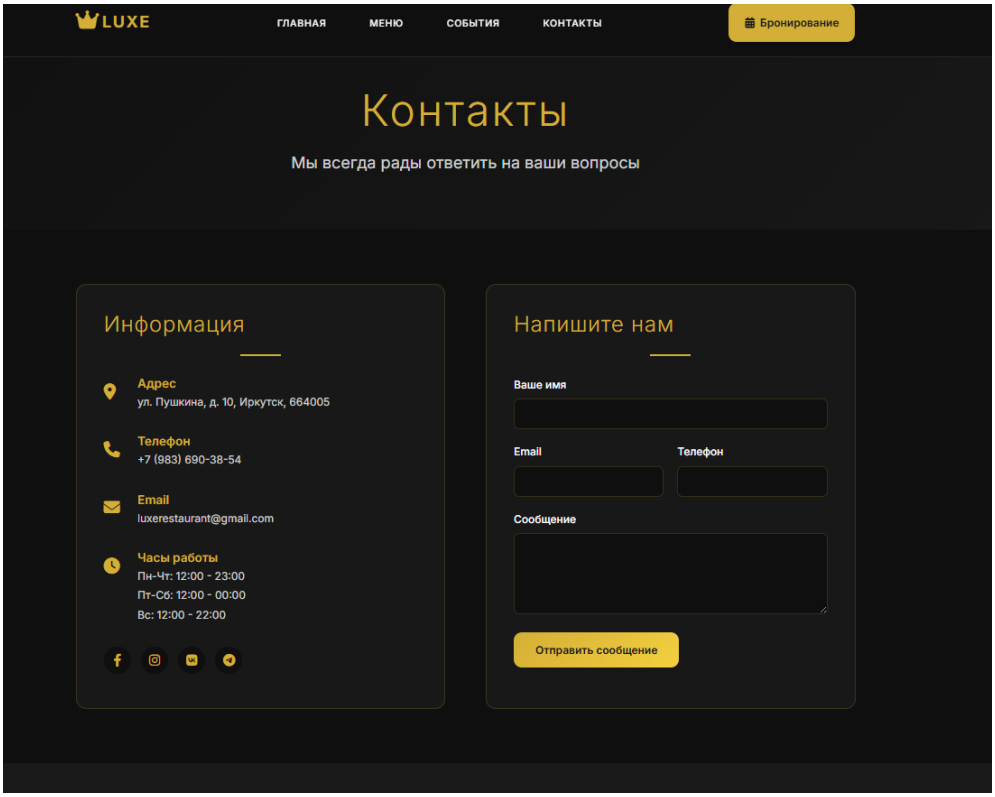

Рисунок 10 – Страница контактов

3.Разработка программного модуля

3.1Разработка интерфейса программного модуля

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

Все страницы были написаны на HTML в PyCharm.

На рисунке 11 изображены страницы главная страница. Эта страница будет самой первой для пользователя, который впервый раз зашёл на сайт.

```
<section class="hero-section">
    <div class="hero-content">
        <div class="container">
            <div class="hero-text">
                <h1 class="hero-title">Добро пожаловать в LUXE</h1>
                <p class="hero-subtitle">
                    Погрузитесь в мир изысканной кухни и безупречного сервиса в самом элегантном ресторане города
                </p>
                <div class="hero-actions">
                    <a href="{% url 'restaurant:reservation' %}" class="btn btn-primary">
                        <i class="fas fa-calendar-alt"></i>
                        Забронировать столик
                    </a>
                    <a href="{% url 'restaurant:menu' %}" class="btn btn-secondary">
                        <i class="fas fa-utensils"></i>
                        Посмотреть меню
                    </a>
                </div>
            </div>
        </div>
    </div>
```

Рисунок 11 – Главная старинца

На рисунке 12 изображена страница меню, где пользователь будет просматривать доступные категории блюд.

```
<section class="menu-page">
    <div class="container">
        <div class="page-header">
            <h1 class="page-title">Наше меню</h1>
            <p class="page-subtitle">Изысканные блюда от шеф-повара ресторана LUXE</p>
        </div>

        <div class="menu-categories-grid">
            {% for category in categories %}
            <div class="menu-category-card">
                <div class="category-header">
                    <div class="category-icon">
                        {% if category.name == 'Основные блюда' %}
                            <i class="fas fa-utensils"></i>
                        {% elif category.name == 'Напитки' %}
                            <i class="fas fa-glass-martini-alt"></i>
                        {% elif category.name == 'Вина' %}
                            <i class="fas fa-wine-glass-alt"></i>
                        {% elif category.name == 'Коктейли' %}
                            <i class="fas fa-cocktail"></i>
                        {% elif category.name == 'Десерты' %}
                            <i class="fas fa-birthday-cake"></i>
                        {% else %}
                            <i class="fas fa-utensils"></i>
                        {% endif %}
                    </div>
                    <h2 class="category-title">{{ category.name }}</h2>
                </div>
```

Рисунок 12 - Страница категорий меню

На рисунке 13 изображена страница блюд, где пользователь будет просматривать доступные блюда. На эту страницу пользователь попадает после категорий меню.

```
<section class="menu-item-detail section">
    <div class="container">
        <div class="menu-item-content grid grid-cols-2">
            <div class="menu-item-image">
                {% if menu_item.image %}
                    <img src="{{ menu_item.image.url }}" alt="{{ menu_item.name }}" class="item-image">
                {% else %}
                    <div class="placeholder-image">
                        <i class="fas fa-utensils"></i>
                    </div>
                {% endif %}
            </div>

            <div class="menu-item-info">
                <div class="breadcrumb">
                    <a href="{% url 'restaurant:menu' %}">Меню</a>
                    <span>/</span>
                    <a href="{% url 'restaurant:category_detail' menu_item.category.slug %}">{{ menu_item.category.
                    <span>/</span>
                    <span>{{ menu_item.name }}</span>
                </div>
```

Рисунок 13 - Страница блюд

На рисунке 14 и 15 изображены страницы бронирования и оформление событий, где пользователь может заказать себе столик или провести свой праздник в ресторане.

```
<section class="reservation-page">
    <div class="container">
        <div class="page-header">
            <h1 class="page-title">Бронирование столика</h1>
            <p class="page-subtitle">Забронируйте столик в ресторане LUXE и насладитесь незабываемым вечером</p>
        </div>
        <div class="reservation-content">
            <div class="reservation-info">
                <h2 class="info-title">Информация о бронировании</h2>
                <div class="info-grid">
                    <div class="info-card">
                        <div class="info-icon">
                            <i class="fas fa-clock"></i>
                        </div>
                        <div class="info-content">
                            <h4>Время работы</h4>
                            <p>Пн-Чт: 12:00 - 23:00<br>
                               Пт-Сб: 12:00 - 00:00<br>
                               Вс: 12:00 - 22:00</p>
                        </div>
                    </div>
```

Рисунок 14 - Страница бронирования

```
<section class="events-hero">
    <div class="container">
        <h1 class="page-title">События и банкеты</h1>
        <p class="page-subtitle">Организуем незабываемые мероприятия</p>
    </div>
</section>

<section class="events-content">
    <div class="container">
        <div class="events-grid">
            <div class="events-info">
                <h2>Виды мероприятий</h2>
```

Рисунок 15 - Страница мероприятий

На рисунке 16 изображена страница контактных данных ресторана и место расположение.

```
<section class="contact-hero">
    <div class="container">
        <h1 class="page-title">Контакты</h1>
        <p class="page-subtitle">Мы всегда рады ответить на ваши вопросы</p>
    </div>
</section>

<section class="contact-content">
    <div class="container">
        <div class="contact-grid">
            <div class="contact-info-card">
                <h2 class="section-title">Информация</h2>

                <div class="contact-item">
                    <div class="contact-icon">
                        <i class="fas fa-map-marker-alt"></i>
                    </div>
                    <div class="contact-text">
                        <h4>Адрес</h4>
                        <p>ул. Пушкина, д. 10, Иркутск, 664005</p>
                    </div>
                </div>

                <div class="contact-item">
                    <div class="contact-icon">
                        <i class="fas fa-phone"></i>
                    </div>
                    <div class="contact-text">
                        <h4>Телефон</h4>
                        <p>+7 (983) 690-38-54</p>
                    </div>
                </div>
```

Рисунок 16 - Страница контактов

3.2Создание базы данных

Процесс разработки базы данных для веб-приложения «Ресторана LUXE» включал следующие этапы:

Проектирование схемы:

Определены основные сущности: категории меню, позиции меню, бронирования, отзывы клиентов, события и банкеты.

Разработана структура таблиц и связи между ними.

Реализация структуры производилась с помощью создания таблиц: category, menu_item, review, event, reservations. Определены поля для каждой таблицы, включая первичные и внешние ключи.

После реализации программного продукта было произведено тестирование: проверка корректности создания таблиц и связей и тестирование основных операций с данными.

Этот процесс обеспечивает создание эффективной базы данных для хранения всей необходимой информации ресторана, учитывая сложные взаимосвязи между различными типами данных. База данных веб-приложения «Ресторана LUXE» состоит из 5 таблиц.

Структура таблиц соответствует схеме базы данных из пункта 2.2.

Таблица «categories» представлена на рисунке 17.

```
CREATE TABLE categories (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  type VARCHAR(100) NOT NULL,
  description TEXT,
  is_active CHAR(20),
  created_at DATETIME
```

Рисунок 17 – Таблица «categories»

Таблица «menu_item» представлена на рисунке 18.

```
CREATE TABLE menu_items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(200) NOT NULL,
  description TEXT,
  price DECIMAL(10,2),
  ingredients TEXT
```

Рисунок 18 – Таблица «menuitem»

Таблица «reservations» представлена на рисунке 19.

```
CREATE TABLE reservations (
  id INT AUTO_INCREMENT PRIMARY KEY,
  first_name CHAR(50) NOT NULL,
  last_name CHAR(50) NOT NULL,
  phone CHAR(20),
  email CHAR(50),
  date DATE NOT NULL,
  time CHAR(5) NOT NULL,
  guests_count INT,
  special_requests TEXT
```

Рисунок 19 – Таблица «reservations»

Таблица «reviews» представлена на рисунке 20.

```
CREATE TABLE reviews (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name CHAR(100),
  email CHAR(100),
  title CHAR(200),
  overall_rating INT,
  food_rating INT,
  service_rating INT,
  atmosphere_rating INT
```

Рисунок 20 – Таблица «reviews»

Таблица «events» представлена на рисунке 21.

```
CREATE TABLE events (
  id INT AUTO_INCREMENT PRIMARY KEY,
  title CHAR(200),
  event_type CHAR(20),
  guests_count INT,
  final_cost DECIMAL(10,2),
  event_date DATE,
  start_time TIME
```

Рисунок 21 – Таблица «events»

3.3Разработка программного модуля

В соответствии с заданием, необходимо реализовать следующие функции:

Просмотр блюд.

Бронирование столика.

Организация мероприятий.

Разработка функционала веб-приложения «Ресторна LUXE» была реализована на Python, база данных и работа с ней также реализована на Python. Взаимодействие с базой данных протекает через PostgreSQL. Заполнение начальных данных происходит через кастомную команду populate_db и populate_menu_db, коды этих команд изображены на рисунках 22 и 23.

```
def create_categories(self):
    categories_data = [
        {
            'name': 'Кухня',
            'slug': 'kitchen',
            'type': 'kitchen',
            'description': 'Авторские блюда',
            'sort_order': 1
        },
        {
            'name': 'Бар',
            'slug': 'bar',
            'type': 'bar',
            'description': 'Крепкие напитки',
            'sort_order': 2
        },
        {
            'name': 'Винная карта',
            'slug': 'wine',
            'type': 'wine',
            'description': 'Изысканные вина',
            'sort_order': 3
        },
        {
            'name': 'Коктейли',
            'slug': 'cocktails',
            'type': 'cocktails',
            'description': 'Авторские коктейли',
            'sort_order': 4
        }
    ]

    for cat_data in categories_data:
        category, created = Category.objects.get_or_create(
            slug=cat_data['slug'],
            defaults=cat_data
        )
        if created:
            self.stdout.write(f'Created category: {category.name}')

def create_menu_items(self):
    categories = Category.objects.all()
```

Рисунок 22 - Заполнение БД начальными данными

```
def create_kitchen_menu(self):
        category = Category.objects.get(slug='kitchen')

        kitchen_items = [
            {
                'name': 'Тартар из тунца',
                'description': 'Свежий тунец с авокадо, лаймом и кунжутным маслом',
                'price': 1200.00,
                'ingredients': 'Тунец, авокадо, лайм, кунжутное масло, микрозелень',
                'calories': 280,
                'cooking_time': 15,
                'is_popular': True,
                'is_gluten_free': True,
                'sort_order': 1
            },
            {
                'name': 'Фуа-гра с карамелизированными грушами',
                'description': 'Деликатесная фуа-гра с грушами в портвейне',
                'price': 2800.00,
                'ingredients': 'Фуа-гра, груши, портвейн, мед, тимьян',
                'calories': 420,
                'cooking_time': 20,
                'is_popular': True,
                'sort_order': 2
            },
            {
                'name': 'Устрицы Белон',
                'description': 'Свежие устрицы с мигнонетом и лимоном (6 шт)',
                'price': 3200.00,
                'ingredients': 'Устрицы Белон, лук-шалот, уксус, лимон',
                'calories': 180,
                'cooking_time': 5,
                'is_gluten_free': True,
                'sort_order': 3
            },

            {
                'name': 'Стейк Рибай Black Angus',
                'description': 'Премиальный стейк из мраморной говядины с трюфельным маслом',
                'price': 4500.00,
                'ingredients': 'Говядина Black Angus, трюфельное масло, розмарин, чеснок',
                'calories': 680,
                'cooking_time': 25,
                'is_popular': True,
                'is_gluten_free': True,
                'sort_order': 10
            },
```

Рисунок 23 - Заполнение БД начальными данными

После заполнения БД начальными данными у нас появляется блюда в меню, так же создаётся учётная запись системного администратора, который может полностью взаимодействовать со всеми данными из БД. После этого мы переходим на сайт и перед нами главная страница. Код главной страницы изображён на рисунке 24.

```
class HomeView(TemplateView):
    template_name = 'restaurant/home.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['categories'] = Category.objects.filter(is_active=True).order_by('sort_order')
        context['featured_reviews'] = Review.objects.filter(
            is_approved=True, is_featured=True
        )[:3]
        context['popular_items'] = MenuItem.objects.filter(
            is_popular=True, is_available=True
        )[:6]
        return context
```

Рисунок 24 - Метод перехода страниц

Метод бронирования столика в ресторане пользователем. Код бронирования изображён на рисунке 25.

```
    first_name = forms.CharField(
        max_length=50,
        widget=forms.TextInput(attrs={
            'class': 'form-control',
            'placeholder': 'Введите ваше имя',
            'required': True
        }),
        label='Имя'
    )

    last_name = forms.CharField(
        max_length=50,
        widget=forms.TextInput(attrs={
            'class': 'form-control',
            'placeholder': 'Введите вашу фамилию',
            'required': True
        }),
        label='Фамилия'
    )

    email = forms.EmailField(
        widget=forms.EmailInput(attrs={
            'class': 'form-control',
            'placeholder': 'example@email.com',
            'required': True
        }),
        label='Email'
    )

    phone = forms.CharField(
        validators=[phone_regex],
        max_length=17,
        widget=forms.TextInput(attrs={
            'class': 'form-control',
            'placeholder': '+7 (999) 999-99-99',
            'required': True
        }),
        label='Телефон'
    )
```

Рисунок 25 - Метод бронирования столика

4.Тестирование программного модуля

Тестирование веб-приложения является критически важным этапом разработки, позволяющим выявить потенциальные ошибки и удостовериться в корректности работы всех функций системы. В данном документе представлены тестовые сценарии, включающие как успешные, так и неудачные кейсы, которые демонстрируют поведение системы при различных условиях. Тестирование охватывает ключевые функциональные модули приложения, включая авторизацию пользователей и управление контентом. Результаты тестирования помогут оценить стабильность работы приложения и выявить области, требующие доработки.

Сценарий 1

Таблица 9 – Сценарий тестирования на валидацию данных при бронирование

Поле	Описание
Дата теста	07.06.2025
Приоритет тестирования	Высокий
Заголовок/название теста	Проверка на валидацию при бронирование
Этапы теста	Пользователь заходит на сайт. Нажимает на кнопку бронирование. Заполняет свои данные: почта, пароль, имя, время и телефон.
Тестовые данные	qwe@mail.ru, qweqwe, qwe123, 12:00, 89836903854
Ожидаемый результат	Валидация и проверка действительно существующий почта
Фактический результат	Пользователь не зарегистрирован.

Таблица 10 – Чек-лист безуспешного бронирования столика

Чек-лист 1

Текстовый пример #	1
Приоритет тестирования	Высокий
Заголовок/название теста	Бронирование столика
Краткое изложение теста	Внесение данных пользователя
Этапы теста	Заполнение имени, фамилии, телефона и почты.
Тестовые данные	qwe@mail.ru, qweqwe, qwe123, 12:00, 89836903854
Ожидаемый результат	Валидация и проверка действительно существующий почты
Статус	Незачёт
Предварительное условие	Пользователю необходимо вводить действительные данные
Постусловие	После успешной бронирования, сайт отправляет запрос в ресторан.

Таблица 11 – Сценарий тестирования на действительность данных

Сценарий 2

Поле	Описание
Дата теста	07.06.2025
Приоритет тестирования	Средний
Заголовок/название теста	Проверка внесения данных пользователя в базу данных
Этапы теста	Пользователь заходит на сайт. Нажимает на кнопку бронирование. Заполняет свои данные: почта, пароль, имя, время и телефон.
Тестовые данные	Aleks@mail.ru, qwe123qwe, Александр, 12:00, 89836903854
Ожидаемый результат	Данные успешно внесутся.
Фактический результат	Успешно

Таблица 12 – Чек-лист успешного бронирования столика

Чек-лист 2

Текстовый пример #	2
Приоритет тестирования	Средний
Заголовок/название теста	Проверка внесения данных пользователя в базу данных
Краткое изложение теста	Внесение данных пользователя
Этапы теста	Заполнение имени, фамилии, телефона и почты.
Тестовые данные	Aleks@mail.ru, qwe123qwe, Александр, 12:00, 89836903854
Ожидаемый результат	Данные успешно внесутся.
Статус	Зачет
Предварительное условие	Пользователю необходимо вводить действительные данные
Постусловие	После успешного создания мероприятия система направляет запрос ресторану.

Таблица 13 – Сценарий тестирования создания мероприятия

Сценарий 3

Поле	Описание
Дата теста	07.06.2025
Приоритет тестирования	Средний
Заголовок/название теста	Проверка создания мероприятия на панели Django-администрирования
Этапы теста	Администратор заходит на сайт. Нажимает на кнопку создания мероприятия. Заполняет поля, краткое описание, заполнение данных пользователя и отправляет запрос ресторану.
Тестовые данные	Свадьба, Александр, 89836903854, Aleks@mail.ru, 9.06.2025, 15, 11:00, 18:00, 100000 рублей.
Ожидаемый результат	Данные успешно будут отображаться на панели Django.
Фактический результат	Успешно

Таблица 14 – Чек-лист успешного создания мероприятия

Чек-лист 3

Текстовый пример #	3
Приоритет тестирования	Средний
Заголовок/название теста	Проверка создания статьи на панели Django-администрирования
Краткое изложение теста	Во время проведения теста администратору необходимо забронировать столик.
Этапы теста	Администратор заходит на сайт. Нажимает на кнопку бронировать. Заполняет поля, вносит данные и указывает дату и время, отправляет запрос.
Тестовые данные	Александр, 89836903854, Aleks@mail.ru, 9.06.2025, 15, 11:00,
Ожидаемый результат	Данные успешно будут отображаться на панели Django.
Статус	Зачет
Предварительное условие	Администратору необходимо получить ответ от ресторана.
Постусловие	После успешного бронирования, можно будет просмотреть бронь.

Таблица 15 – Сценарий тестирования бронирования

Сценарий 4

Поле	Описание
Дата теста	07.06.2025
Приоритет тестирования	Высокий
Заголовок/название теста	Оставления отзыва.
Этапы теста	Администратор заходит на сайт. Хочет создать отзыв. При нажатии на кнопку создания статьи система автоматически проверит его права.
Тестовые данные	-
Ожидаемый результат	Администратор не сможет оставил отзыв.
Фактический результат	Администратор не оставил отзыв.

Таблица 16 – Чек-лист безуспешной попытки оставления отзыва.

Чек-лист 4

Текстовый пример #	4
Приоритет тестирования	Высокий
Заголовок/название теста	Ограничение прав пользователей
Краткое изложение теста	Во время проведения теста администратор попытается оставить отзыв.
Этапы теста	Администратор заходит на сайт. Хочет оставить отзыв. При нажатии на кнопку создания статьи система автоматически проверит его права.
Тестовые данные	-
Ожидаемый результат	Администратор оставляет отзыв
Статус	Зачет
Предварительное условие	Администратор имеет права суперпользователя.
Постусловие	После попытки оставления отзыва администратором, система покажет успешность действия.

5.Документирование программного модуля

5.1Руководство пользователя

Веб-приложение «Ресторан LUXE» имеет простой и удобный интерфейс, что позволяет пользователю легко ориентироваться в его возможностях. При открытии сайта пользователя встречает главная страница, где, для просмотра меню необходимо перейти во вкладку меню. Главная страница представлена на рисунке 26.

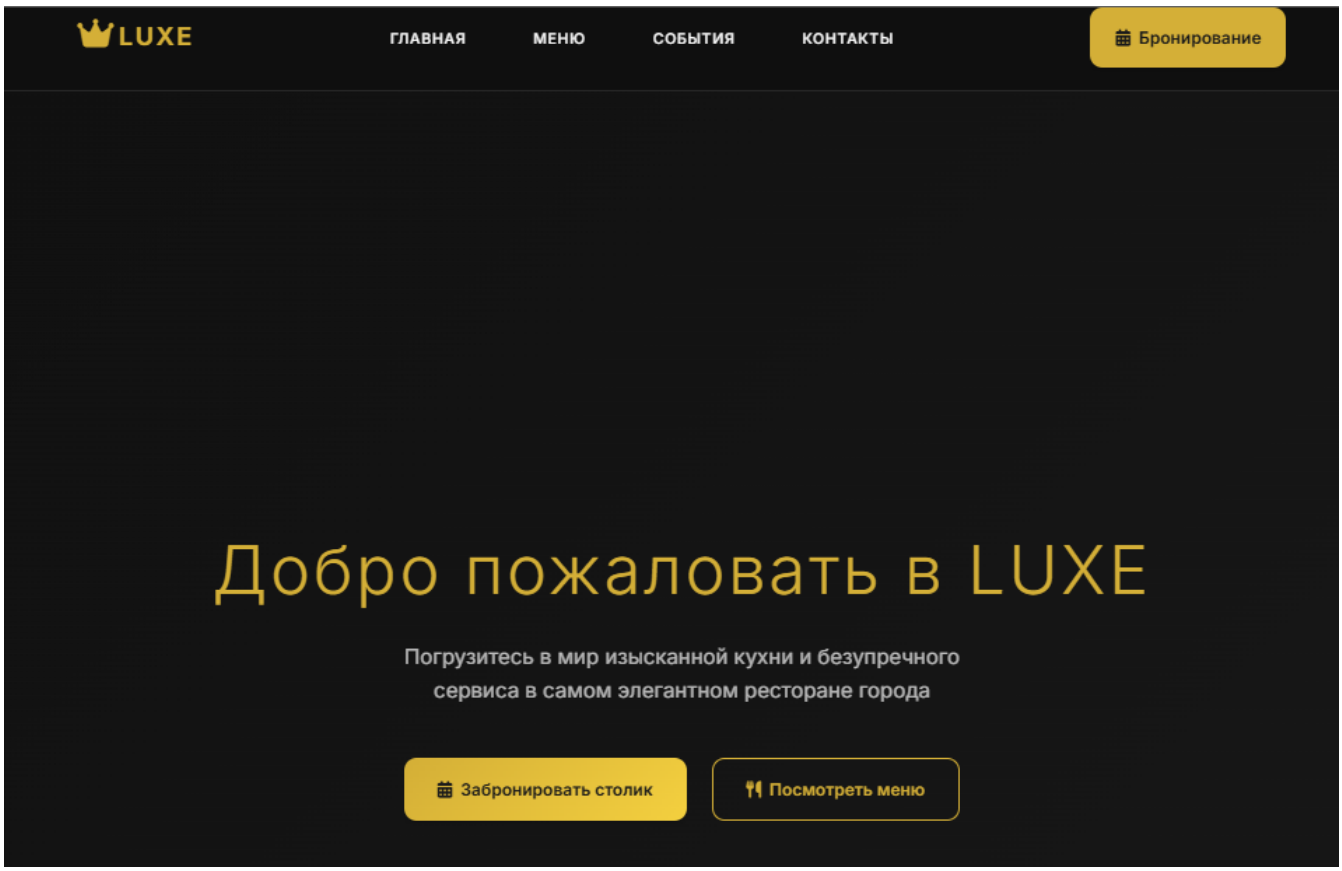

Рисунок 26 – Главная страница

После пользователь может забронировать столик на странице представленной на рисунке 27, для посещения ресторана.

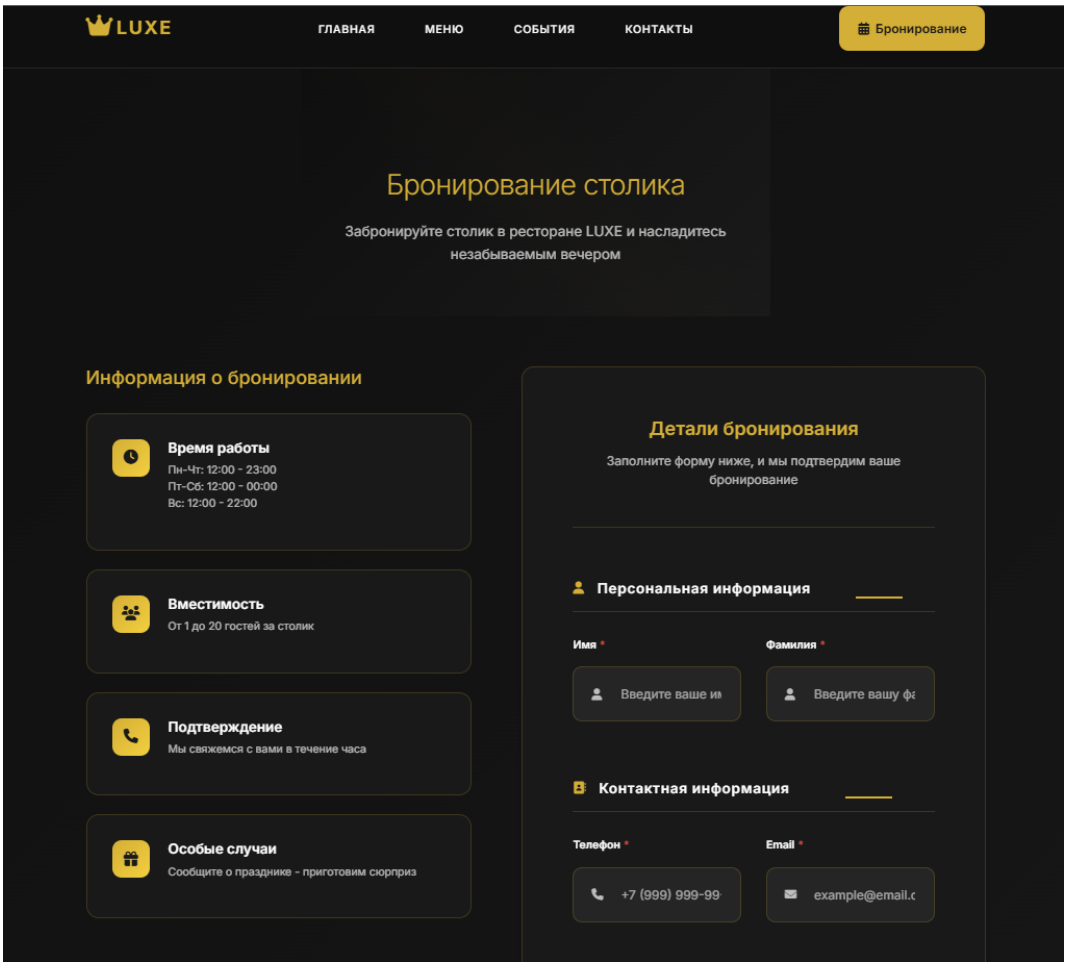

Рисунок 27 – Страница бронирования

На главной странице пользователь может нажать на кнопку контакты для получения более подробной информации о связи с рестораном, страница поиска представлена на рисунке 28.

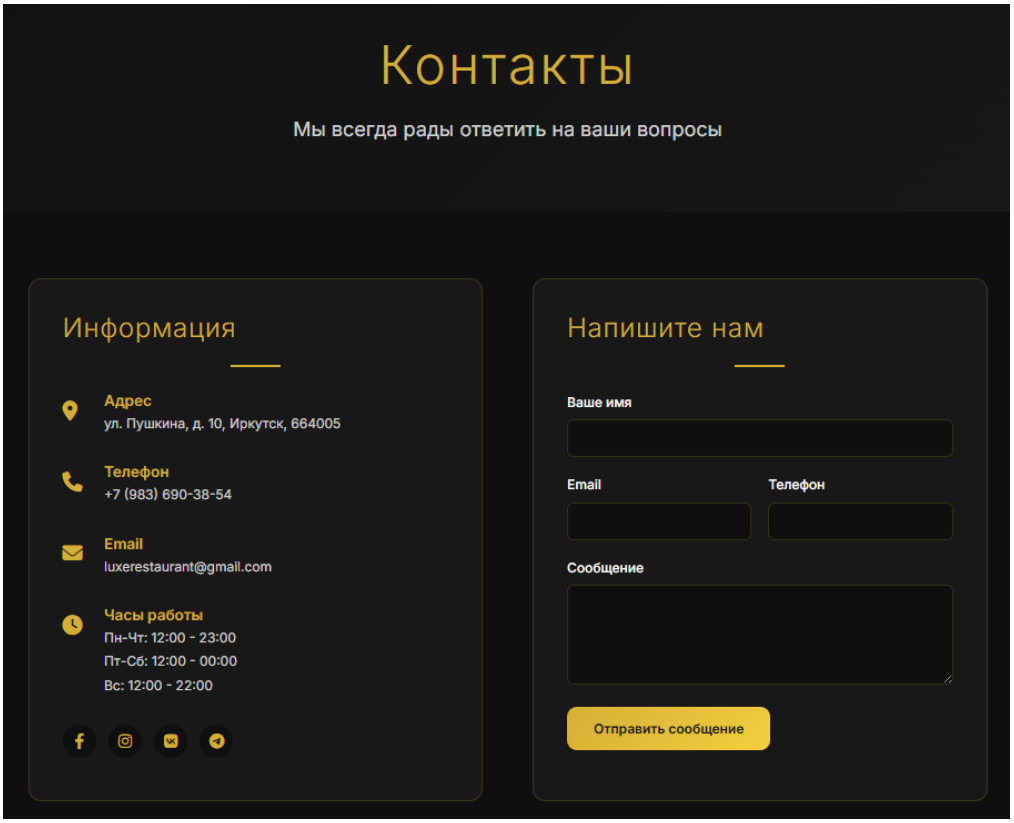

Рисунок 32 – Страница контактов

6.Оценка возможности модернизации программного модуля

Оценка возможности модернизации веб-приложения «ресторана LUXE» демонстрирует значительный потенциал для развития и расширения функциональности, поскольку существующая структура базы данных, соответствующая второй нормальной форме и включающая шесть основных таблиц (Category, MenuItem, Reservation, Review, Event), обеспечивает надежную основу для внедрения новых возможностей, таких как система персональных рекомендаций блюд на основе анализа предпочтений клиентов и истории их заказов, интеграция с системами онлайн-платежей для предварительной оплаты бронирований и мероприятий, добавление интерактивной схемы зала с возможностью выбора конкретного столика при бронировании, внедрение системы лояльности с накоплением бонусных баллов и автоматическим применением скидок, создание мобильного приложения с push-уведомлениями о статусе бронирования и специальных предложениях, интеграция с социальными сетями для автоматической публикации отзывов и фотографий блюд, добавление системы онлайн-заказа еды на вынос с отслеживанием готовности, внедрение чат-бота для быстрых ответов на часто задаваемые вопросы, создание административной панели с аналитикой посещаемости и популярности блюд, оптимизация поиска по меню с фильтрами по диетическим предпочтениям и аллергенам, реализация REST API для интеграции с внешними сервисами доставки и агрегаторами ресторанов, добавление системы управления запасами с автоматическим обновлением доступности блюд, внедрение CRM-системы для персонализированного обслуживания постоянных клиентов, что в совокупности сделает «ресторан LUXE» более технологичным, удобным для клиентов и эффективным в управлении.

7.Производственные задачи

ПК 01.01. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием

ПК 01.02. Разрабатывать программные модули в соответствии с техническим заданием

ПК 01.03. Выполнять отладку программных модулей с использованием специализированных программных средств

ПК 01.04. Выполнять тестирование программных модулей

ПК 01.05. Осуществлять рефакторинг и оптимизацию программного кода

ПК 01.06. Разрабатывать модули программного обеспечения для мобильных платформ

Этапы:

Изучить предметную область, для которой необходимо разработать программный модуль. Определить требования к функциональности, надежности, безопасности и другим аспектам системы.

Разработать концепцию программного модуля, включающую ее цели, функциональность, структуру и архитектуру. Определить технические требования к модулю, выбрать подходящие технологии и инструменты для его разработки.

Создать проектные документы, включающие описание предметной области, архитектуры и схемы баз данных, алгоритмы, прототипы интерфейсов и другие важные аспекты программного модуля.

Задание на 26.05.25

1. Исследовать предметную область будущего программного продукта.
2. Провести анализ инструментальных средств, обосновать свой выбор.

- Провести анализ и выбрать язык программирования, инструмент проектирования и среду разработки.
- Определить не менее 3 критериев сравнения.
- Для каждого выбранного инструмента сделать вывод.
- **Задание на 28.05.25**

3. Провести проектирование программного продукта.

- Построить диаграмму прецедентов, деятельности.
- **Задание на 29.05.25**
- Создание ER-диаграммы (сущности, связи, атрибуты).
- Нормализация БД (1NF →2NF → 3NF).
- Написание SQL-скриптов для создания таблиц.
- **Задание на 30.05.25**

1. Прототипирование интерфейса.

– Прототипы и макеты 5 основных окон программного продукта (не считая авторизации/регистрации)

– Описание выбора цвета и типографики

– Описание интерфейса

Задание на 31.05.-02.06.25

4. Разработать программный продукт.

- Разработка клиентской части на основе макетов
- Верстка интерфейса
- ПП должен соответствовать выбранной предметной области.
- ПП должен выполнять основные операции (просмотр, добавление, изменение и удаление данных).
- **Задание на 03.06.-06.06.25**

1. Разработка серверной части

– Работа с данными

2. Интеграция и отладка

– Связывание frontend и backend.

– Тестирование взаимодействия.

3. Проверить программный код на предмет соответствия стандартам кодирования:

– Определить стандарты кодирования.

– Структурировать проект и программный код.

– Наименовать переменные соответствующим стандартам.

– Наименовать функции соответствующим стандартам.

– Продемонстрировать комментарии, поясняющие смысл написанного кода.

4. Руководство пользователя (титульный лист, содержание (автособираемое), нумерация страниц, скриншоты функционала ПП, инструкции).

Заключение

В ходе производственной практики было разработано веб-приложение «Ресторан LUXE», в рамках которого поставленная цель создания комплексной системы управления рестораном и входящие в её состав задачи были успешно выполнены, в результате чего приложение предоставляет пользователям возможность просматривать меню ресторана с детальной информацией о блюдах, включая состав, калорийность и диетические характеристики, осуществлять бронирование столиков с выбором даты, времени и количества гостей, планировать специальные мероприятия и банкеты с указанием требований и предпочтений, оставлять отзывы с оценками качества еды, обслуживания и атмосферы, а администраторам системы обеспечивает функциональность управления категориями и позициями меню, обработки и подтверждения бронирований.

Цель работы предполагала решение следующих задач:

- Анализ предметной области.
- Выбор инструментов разработки.
- Проектирование базы данных.
- Разработка backend-части:
- Настройка моделей Django для хранения данных.
- Реализация CRUD-функционала для статей.
- Создание системы аутентификации и разграничения прав (например, модераторы/обычные пользователи).
- Разработка frontend-части:
- Верстка шаблонов с адаптивным дизайном.
- Добавление интерактивных элементов (поиск, фильтры, пагинация).
- Тестирование: проверка работоспособности всех функций, включая обработку пользовательского ввода и отображение контента.
- Документирование: описание архитектуры проекта, руководство для пользователей и разработчиков.

Список используемых источников

1 Habr / Работа с Django в Pycharm – URL: https://habr.com/ru/articles/336510/ (дата обращения: 01.06.2025)

2 Metanit / Основы Django в Pycharm – URL: https://metanit.com/py/pycharm/14.1.php(дата обращения: 02.06.2025)

3 Unetway / Python Синтаксис – URL: https://unetway.com/tutorial/sqlite–syntax?ysclid=la0uq3qfnr275255001 (дата обращения: 02.06.2025). – Текст: электронный.

4 Developer pycharm / pycharm – URL: https://developer.pycharm.com/studio (дата обращения: 03.06.2025). – Текст: электронный.

5 Pycharm Community Edition / pycharmcommunityedition – URL: https:// pycharmcommunityedition.org/dl/ (дата обращения: 03.06.2025). – Текст: электронный.

6 Tproger / Руководство по Pycharm– URL: https://tproger.ru/articles/android–studio–guide/ (дата обращения: 04.06.2025)

7 Proglib / Основы Python – URL: https://proglib.io/p/python-basics (дата обращения: 04.06.2025)

8 Blog.skillfactory / Как пользоваться Pycharm – URL: Pycharm: как пользоваться – начало работы и настройка (skillfactory.ru) (дата обращения: 05.06.2025). – Текст: электронный.

9 Pycharm.ru / Работа с базами данных в Pycharm – URL: https://pycharm.ru/courses/basics/19–rabota–s–bazami–dannyx–v–pycharm/ (дата обращения: 05.06.2025)

Practicum.yandex / Pycharm: как пользоваться – установка, настройка и установка – URL: https://practicum.yandex.ru/blog/kak–polzovatsya–pycharm/ (дата обращения: 06.07.2025). – Текст: электронный.

Приложение А Листинг – Работа с созданием статьи

```
@views
from django.shortcuts import render, get_object_or_404, redirect
from django.views.generic import ListView, DetailView, CreateView, TemplateView
from django.contrib import messages
from django.http import JsonResponse
from django.core.mail import send_mail
from django.conf import settings
from django.utils import timezone
from .models import Category, MenuItem, Reservation, Review, Event
from .forms import ReservationForm, ContactForm, EventRequestForm


class HomeView(TemplateView):
    template_name = 'restaurant/home.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['categories'] = Category.objects.filter(is_active=True).order_by('sort_order')
        context['featured_reviews'] = Review.objects.filter(
            is_approved=True, is_featured=True
        )[:3]
        context['popular_items'] = MenuItem.objects.filter(
            is_popular=True, is_available=True
        )[:6]
        return context
```

```
class ContactView(TemplateView):
    template_name = 'restaurant/contact.html'

    def post(self, request, *args, **kwargs):
        form = ContactForm(request.POST)
        if form.is_valid():
            # Здесь можно добавить отправку email
            name = form.cleaned_data['name']
            email = form.cleaned_data['email']
            phone = form.cleaned_data['phone']
            message = form.cleaned_data['message']

            messages.success(request,
                             f'Спасибо, {name}! Ваше сообщение отправлено. Мы свяжемся с вами в ближайшее время.')
            return redirect('restaurant:contact')
        else:
            messages.error(request, 'Пожалуйста, исправьте ошибки в форме.')
            return self.render_to_response(self.get_context_data(form=form))

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        if 'form' not in context:
            context['form'] = ContactForm()
        return context


class MenuCategoryListView(ListView):
    model = Category
```

```
    template_name = 'restaurant/menu_categories.html'
    context_object_name = 'categories'

    def get_queryset(self):
        return Category.objects.filter(is_active=True).order_by('sort_order')


class MenuCategoryDetailView(DetailView):
    model = Category
    template_name = 'restaurant/category_detail.html'
    context_object_name = 'category'
    slug_field = 'slug'
    slug_url_kwarg = 'slug'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['menu_items'] = self.object.menu_items.filter(
            is_available=True
        ).order_by('sort_order', 'name')
        return context


class MenuItemDetailView(DetailView):
    model = MenuItem
    template_name = 'restaurant/menu_item_detail.html'
    context_object_name = 'menu_item'


class ReservationCreateView(CreateView):
    model = Reservation
```

```
        form_class = ReservationForm
        template_name = 'restaurant/reservation.html'

        def get_context_data(self, **kwargs):
            context = super().get_context_data(**kwargs)
            if 'form' not in context:
                context['form'] = self.get_form()
            return context

        def form_valid(self, form):
            reservation = form.save()
            messages.success(
                self.request,
                f'Бронирование успешно создано! Код подтверждения: {reservation.confirmation_code}.'
            )
            return redirect('restaurant:reservation_success')

        def form_invalid(self, form):
            messages.error(self.request, 'Пожалуйста, исправьте ошибки в форме.')
            return super().form_invalid(form)


    class ReservationSuccessView(TemplateView):
        template_name = 'restaurant/reservation_success.html'


    class EventsView(TemplateView):
        template_name = 'restaurant/events.html'
```

```
        def post(self, request, *args, **kwargs):
            form = EventRequestForm(request.POST)
            if form.is_valid():
                event = form.save()
                messages.success(
                    request,
                    f'Заявка на событие "{event.title}" успешно отправлена! Наш менеджер свяжется с вами в ближайшее время.'
                )
                return redirect('restaurant:events')
            else:
                messages.error(request, 'Пожалуйста, исправьте ошибки в форме.')
                return self.render_to_response(self.get_context_data(form=form))

        def get_context_data(self, **kwargs):
            context = super().get_context_data(**kwargs)
            if 'form' not in context:
                context['form'] = EventRequestForm()
            return context


    def reservation_ajax(request):
        if request.method == 'POST':
            form = ReservationForm(request.POST)
            if form.is_valid():
                reservation = form.save()
                return JsonResponse({
                    'success': True,
                    'message': f'Бронирование успешно отправлено! Код подтверждения: {reservation.confirmation_code}',
```

```
                'confirmation_code': reservation.confirmation_code
            })
        else:
            return JsonResponse({
                'success': False,
                'errors': form.errors
            }, status=400)

    return JsonResponse({'success': False, 'message': 'Метод не поддерживается'}, status=405)
```