**Введение**

Кафе быстрого питания "Бистро" с роботами-официантами и экранами для заказов предлагает новый сервис, который обеспечивает клиентов удобным способом совершать заказы и наслаждаться вкусной едой. Клиенты могут легко выбирать блюда, оформлять заказы и получать информацию о меню через удобные экраны. Новый сервис "Бистро" решает проблемы, связанные с организацией заказов и обслуживанием клиентов.

Сервис "Бистро" ставит перед собой следующие цели:

1. Увеличение числа клиентов;
2. Увеличение числа оформляемых заказов;
3. Увеличение прибыли компании;

Методы достижения целей сервиса "Бистро":

1. Обеспечить клиентам удобный способ выбора блюд и оформления заказов;

2. Обеспечить эффективное управление заказами и оптимизацию процесса обслуживания.

Новый сервис "Бистро" создан для того, чтобы удовлетворить потребности клиентов и обеспечить им приятный опыт посещения кафе быстрого питания.

**1. Проектирование сервиса «*Бистро*»**

В рамках данной главы был проведен анализ предметной области «Бистро». Были определены сценарии использования, разработан проект интерфейса и спроектирована база данных.

**1.1. Описание предметной области предприятие общественного питания «Бистро»**

Инновационное предприятие общественного питания "Бистро" ежедневно занимается приготовлением различных блюд по заказам клиентов. Каждое блюдо имеет название, вид (бургер, салат, суп, пицца), состав продуктов, калорийность, вес порции, изображение блюда, цена. Блюда могут быть изменены клиентом. Каждый продукт имеет название, калорийность (ккал на 100г продукта), цена (за 1 кг, руб). Каждый продукт может входить в состав нескольких блюд и не входить ни в один состав блюд.

“Бистро” называется инновационным, потому что полным обслуживанием клиентов занимаются Роботы модели "БистроБот". В их обязанности входят оформление и выдача заказа. Когда покупатель приходит в “Бистро”, он садится за стол и к нему подъезжает "БистроБот" с сенсорным экраном, взаимодействуя с которым клиент оформляет заказ. Каждый заказ имеет одно или несколько блюд, каждое блюдо можно заказать в нескольких экземплярах, также заказ имеет номер стола клиента. После оформления заказ проходит ряд последовательных этапов (операций): обрабатывается, готовится, и наконец доставляется клиенту.

**1.2. Исследования: метод персонажа**

1. Клиент

**Клиент, который хочет кушать**

|  |  |
| --- | --- |
| Характеристика | Среднестатистический гражданин РФ от 12 до 55 лет |
| Что делает | 1. Просматривает меню с целью выбора желаемого блюда, 2. выбирает блюдо; 3. убирает продукты, которые он не хочет видеть в своём блюде; 4. просматривает свой заказа и по желанию удаляет некоторые позиции; 5. оформляет заказ и наблюдает за статусом заказа. |
| Что для него важно | 1. Быстрое обслуживание и минимальное время ожидания заказа. 2. качественная и вкусная еда, приготовленная из свежих ингредиентов. 3. чистота и уютная обстановка в кафе. 4. широкий выбор блюд и напитков; 5. возможность оформления заказа через удобное приложение. |

**1.3. Сценарии использования сервиса «*Бистро*»**

Исходя из потребностей клиента можно выделить следующий сценарий использования:

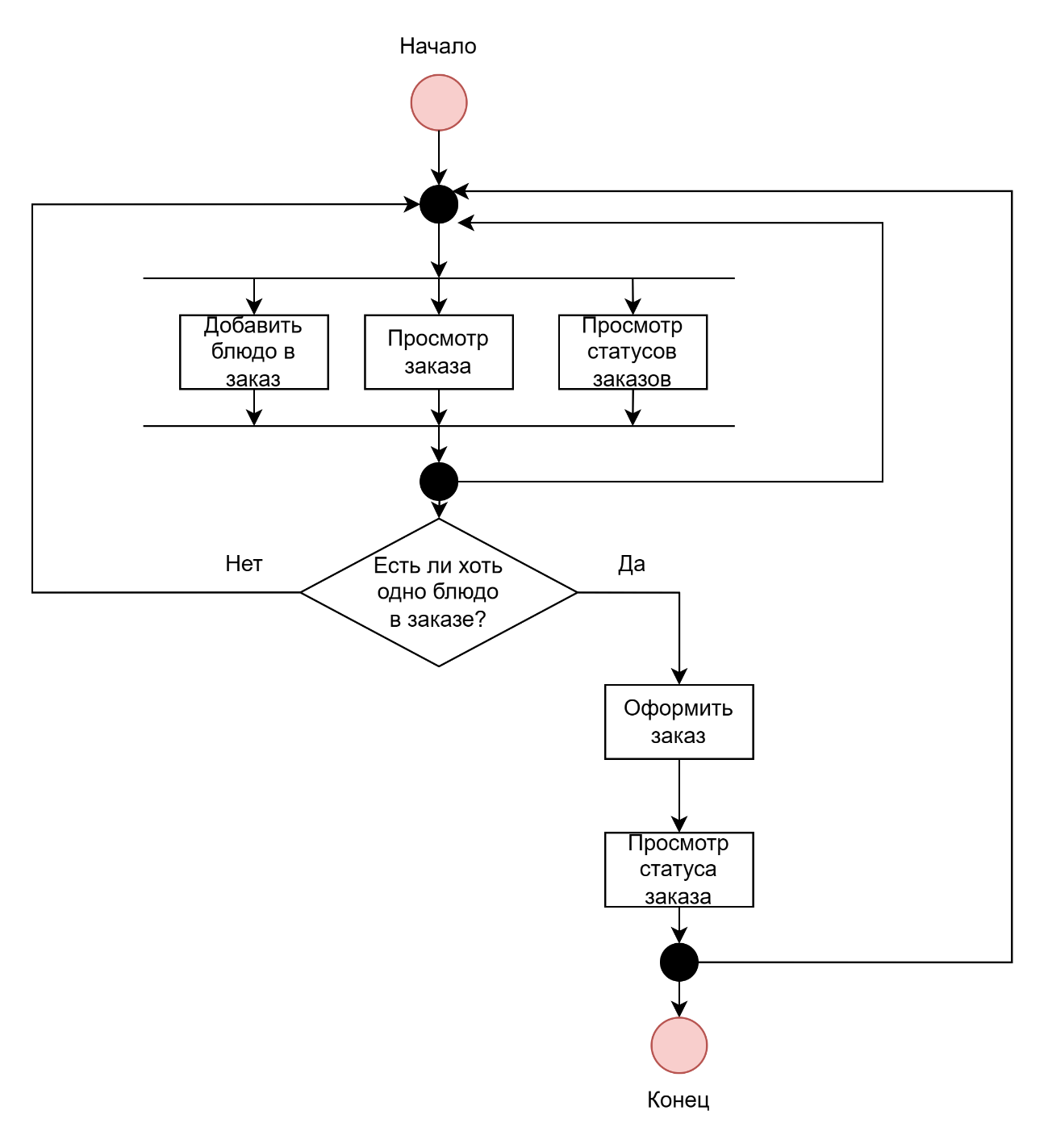


Рисунок 1.3.1 – Сценарий использования пользователя «Клиент»

В свою очередь, под добавить блюдо в заказ понимается:

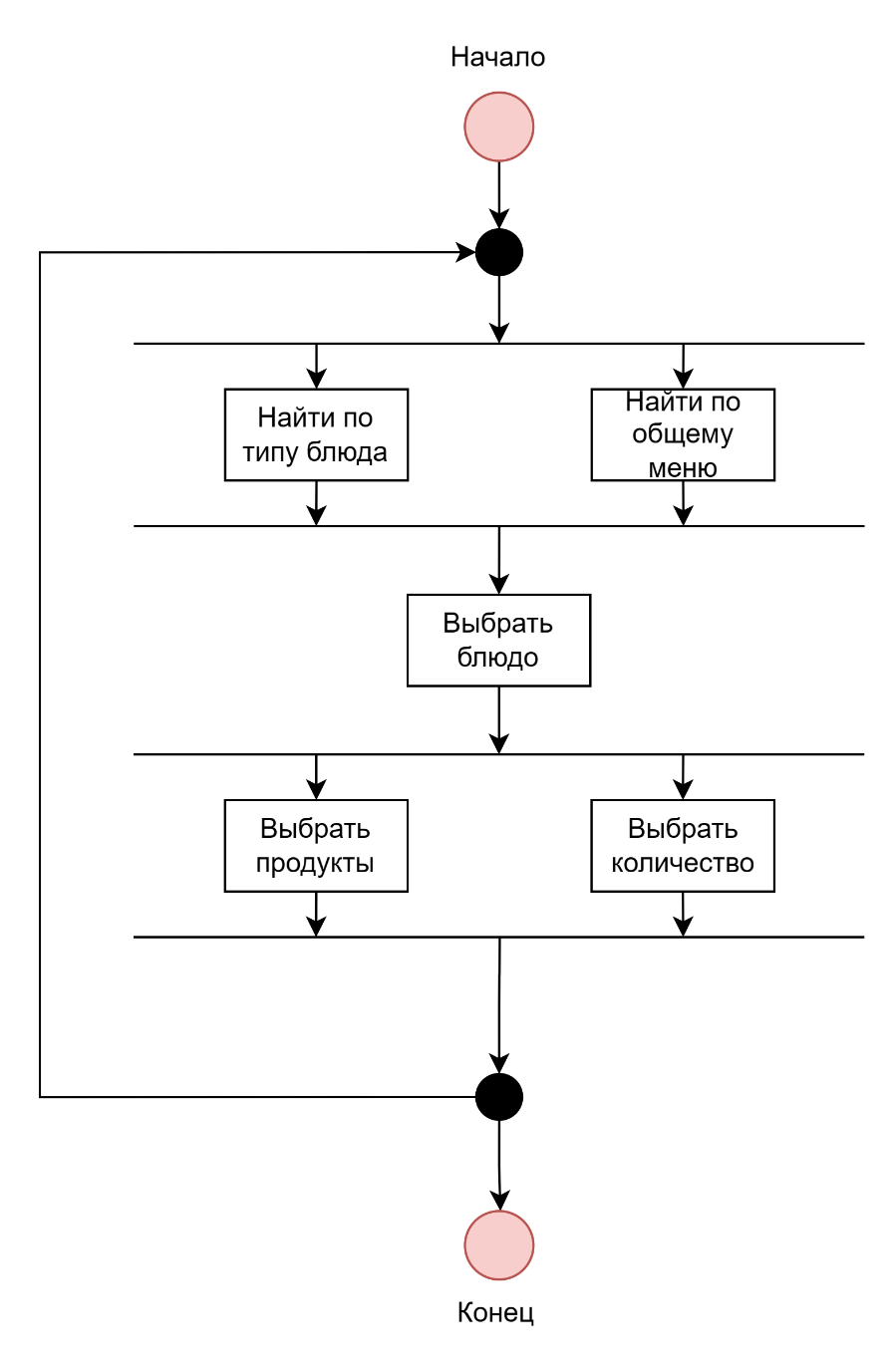


Рисунок 1.3.2 – Сценарий использования при добавлении блюда в заказ

В свою очередь, под просмотреть заказ понимается:

Изображение выглядит как луна

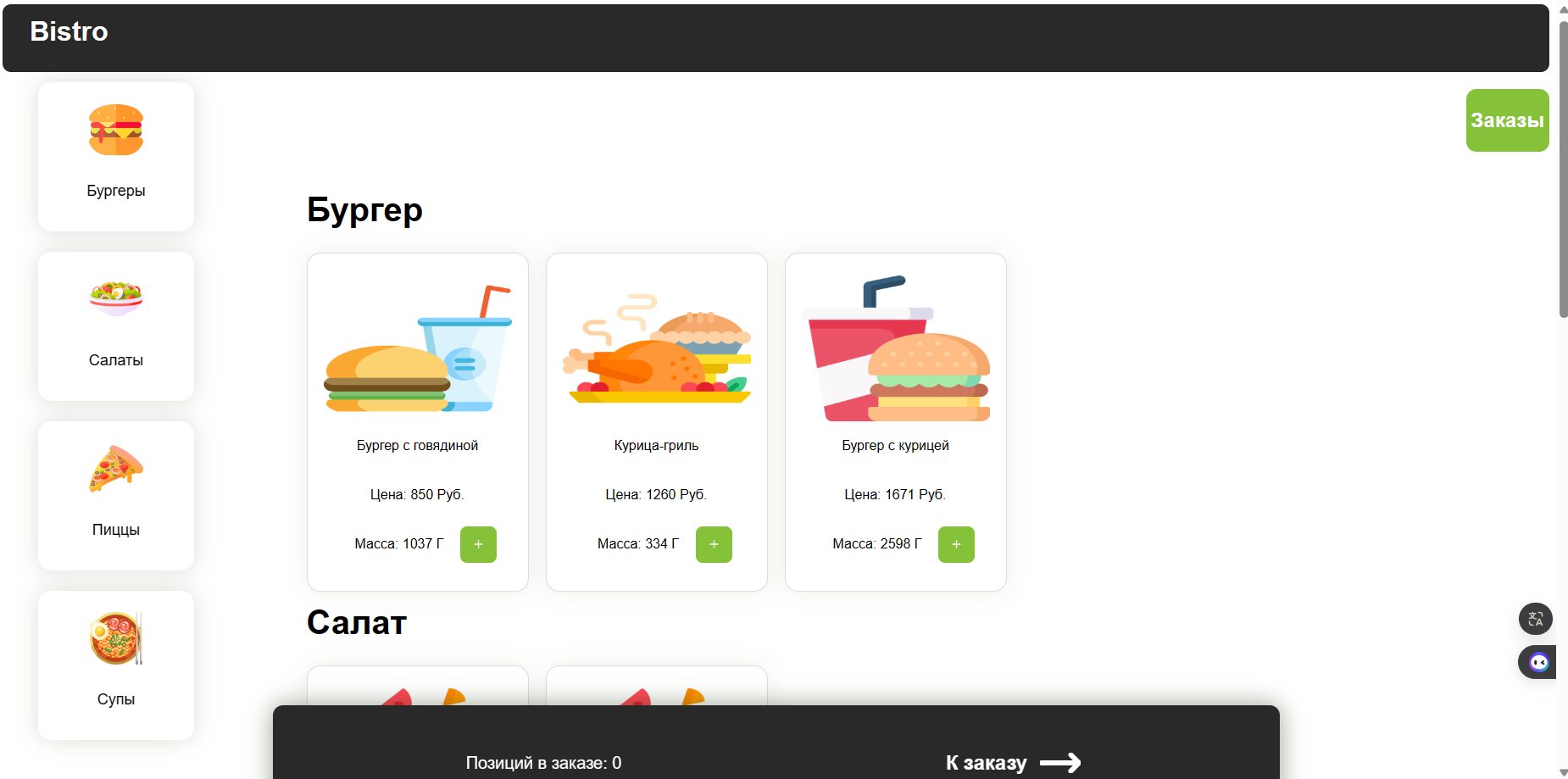
Автоматически созданное описание

Рисунок 1.3.3 - Сценарий использования при просмотре заказа

**1.4 Прототипы интерфейса**

Для сервиса «*Бистро*» был разработан прототип интерфейса, включающий следующие страницы:

1. Страница «Меню»;
2. Страница «Выбор блюда»;
3. Страница «Просмотр заказа»;
4. Страница «Состояние заказа».

а) 

б) Изображение выглядит как текст, снимок экрана, диаграмма, Операционная система

Автоматически созданное описание

Рисунок 1.4.1 – Страница «Меню». Страница при просмотре данных о бургерах (а), Страница при просмотре данных о салатах (б)

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 1.4.2 – Страница «Выбор блюда».

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Операционная система

Автоматически созданное описание

Рисунок 1.4.3 – Страница «Просмотр заказа»

а)Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

б) Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

в) Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 1.4.4 – Страница «Состояние заказа». Страница с состоянием «Обработка» (а), Страница с состоянием «Уже готовим» (б), Страница с состоянием «Заказ готов» (в)

Исходя из приведенных выше окон, можно определить следующую визуальную карту:

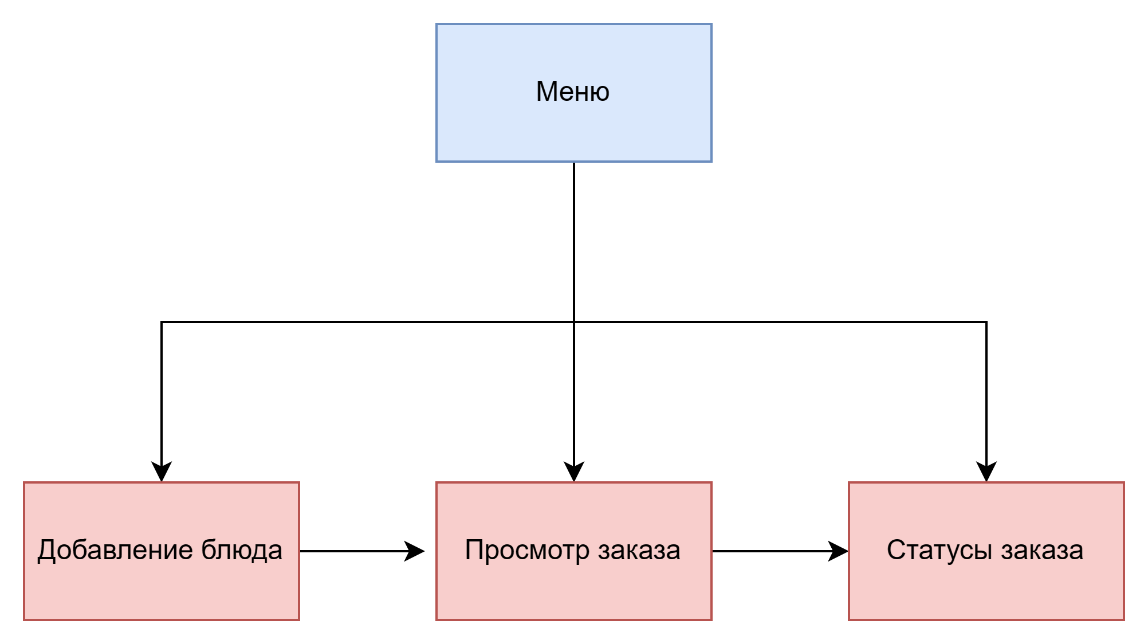


Рисунок 1.4.5 – Визуальная карта «Бистро»

**1.5 Проектирование базы данных**

В результате анализа предметной области «Бистро» можно выделить следующие информационные объекты:

1. Продукт
2. Блюдо
3. Заказ
4. Шаг

На основе выделенных объектов можно получить следующую концептуальную модель, демонстрирующую связи между объектами:

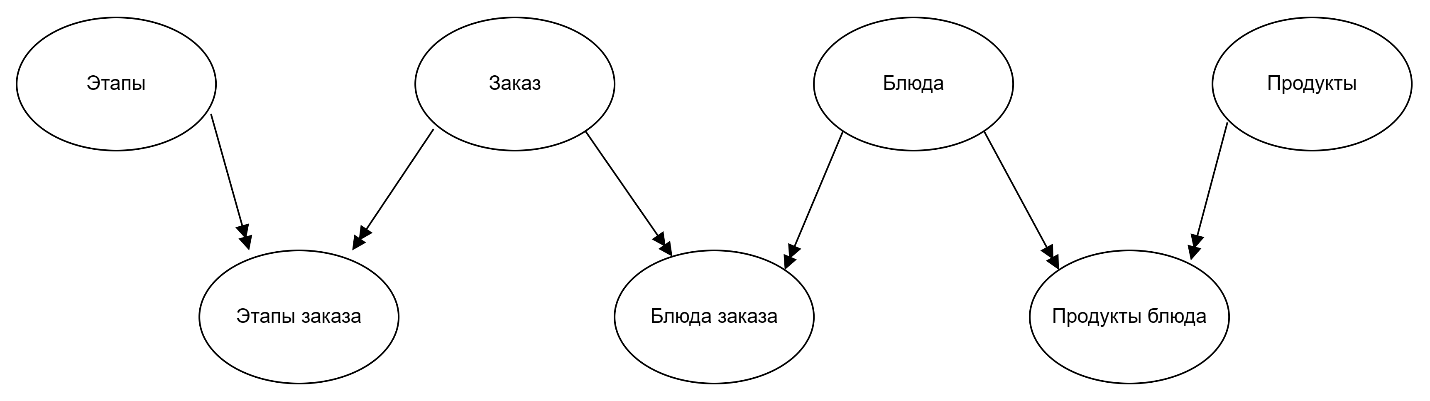


Рисунок 1.5.1 – Концептуальная модель предметной области «Бистро»

Для каждого информационного объекта необходимо сопоставить табличное представление, которое затем должно быть приведено к третьей нормальной форме.

Таким образом, была получена следующая логическая модель:

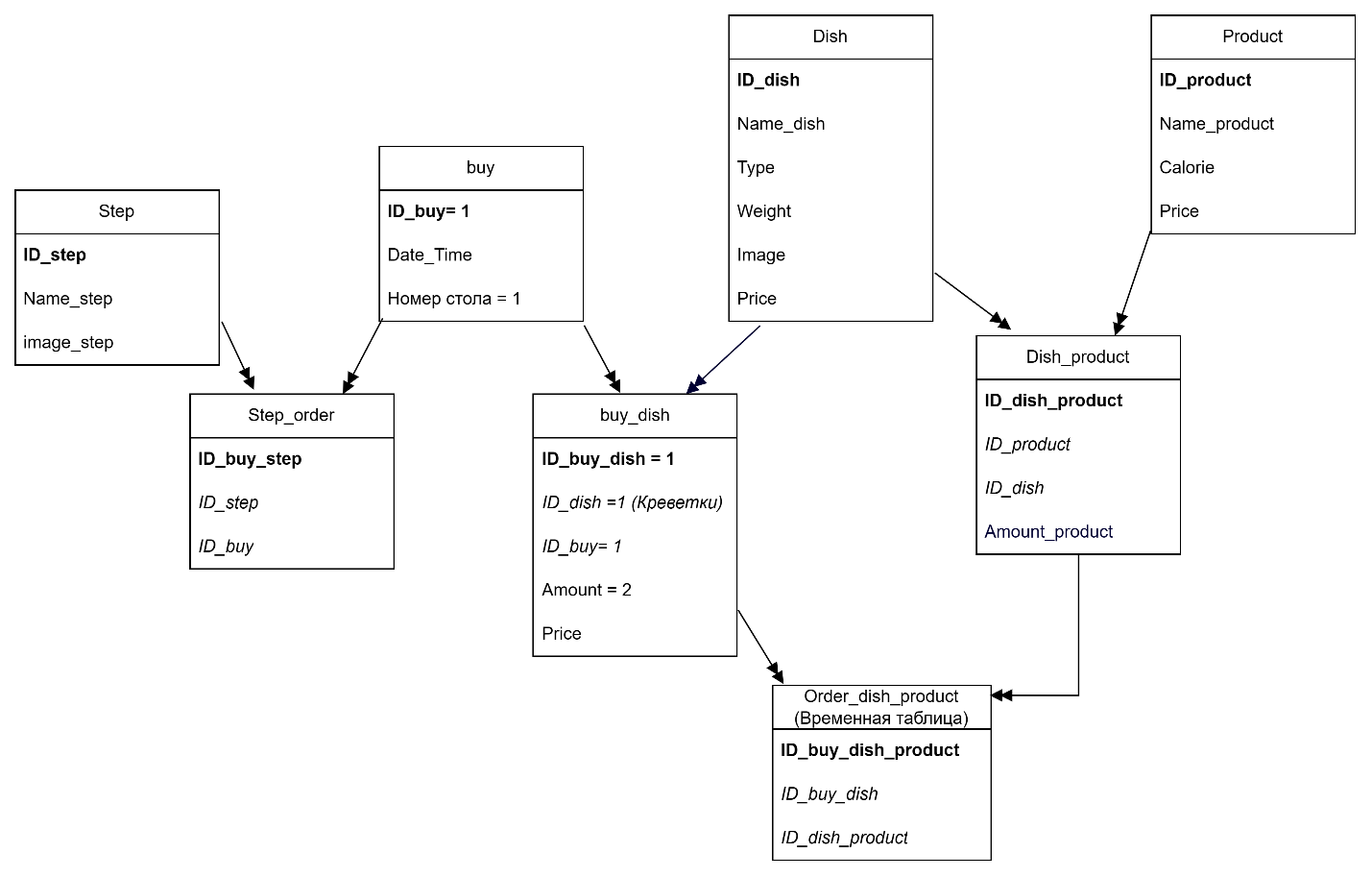


Рисунок 1.5.2 – Логическая модель предметной области

**2 Реализация сервиса «*Бистро*»**

В рамках данной главы была описана реализация проекта «*Бистро*»: определены функциональные возможности, выбраны средства реализации, сформирован граф сценария диалога.

**2.1 Функциональные особенности сервиса «*Бистро*»**

«*Бистро*» должен:

1. Хранить информацию о блюдах, продуктах, заказах, шагах;
2. Позволять редактировать заказы (добавление/удаление блюд);
3. Позволять редактировать блюда (добавлять/удалять продукты);
4. Позволять просматривать статус заказа.

**2.2 Выбор средств реализации**

В основе сервиса «*Бистро*» лежит шаблон *MVC*, позволяющий отделить бизнес-логику от пользовательского интерфейса.

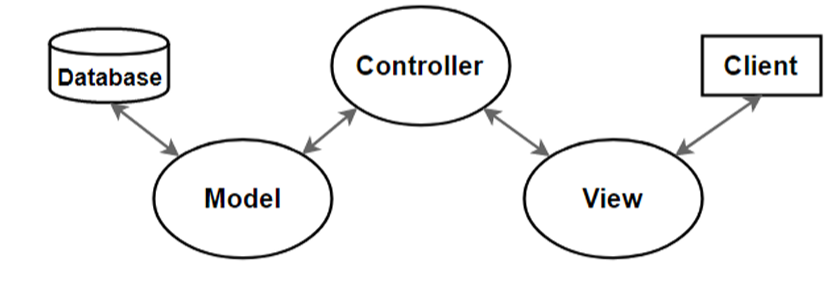


Рисунок 2.2.1 – Концептуальная схема шаблона MVC

Для реализации такого проекта был использован микрофреймфорк *Flask*, написанный на *Python*, отличается своей простотой и гибкостью, делая его доступным для начинающих, в то же время предлагая мощные возможности для создания сложных приложений.

*Flask* также помогает с версткой страниц, он использует систему шаблонов *Jinja2*, которая позволяет создавать *HTML*-страницы с динамическим содержимым*.*

Для формирования базы данных и работы с ней применяется *Python DB-API –* *SQLite*. *SQLite* – компактная, быстрая, универсальная СУБД. Хранит данные в локальном файле, не требует отдельного сервера для выполнения запросов или управления данными: вместо этого она использует библиотеку, которая работает внутри приложения.

**2.3 Алгоритм разработки приложения**

Рассмотрим процесс разработки в соответствии с технологией использования MVC - шаблона.

1. В app.py необходимо:

* импортировать модуль flask;
* создать объект-приложение;
* импортировать все контроллеры, используемые в приложении; настроить сессию.

2. В utils.py реализовать функцию для создания соединения с базой данных.

3. Для каждой страницы создать:

**Котроллер, в нем:**

* импортировать все необходимые функции модели
* установить соединение с базой данных (вызвать функцию модуля units.py);
* обработать отправленные пользователем данные (если они были переданы);
* вызвать необходимые модели и сохранить их результаты в переменные;
* вызвать генерацию шаблона страницы, указать параметры генерации;

**Модель, в ней:**

* разместить в отдельной функции запросы sql, выполняющие определенную последовательность действий (это может быть один запрос на выборку или несколько запросов корректировки данных), параметрами функции будут соединение с базой данных и значения для выбора данных, результат – отобранная из базы данных информация, в виде DataFrame.

Шаблон, в нем:

* создать структуру html-страницы, для генерации динамических частей использовать конструкции шаблонизатора.

Для реализации приложения используется структура модулей (рисунок 2.2.2). Также на схеме показаны связи между модулями.

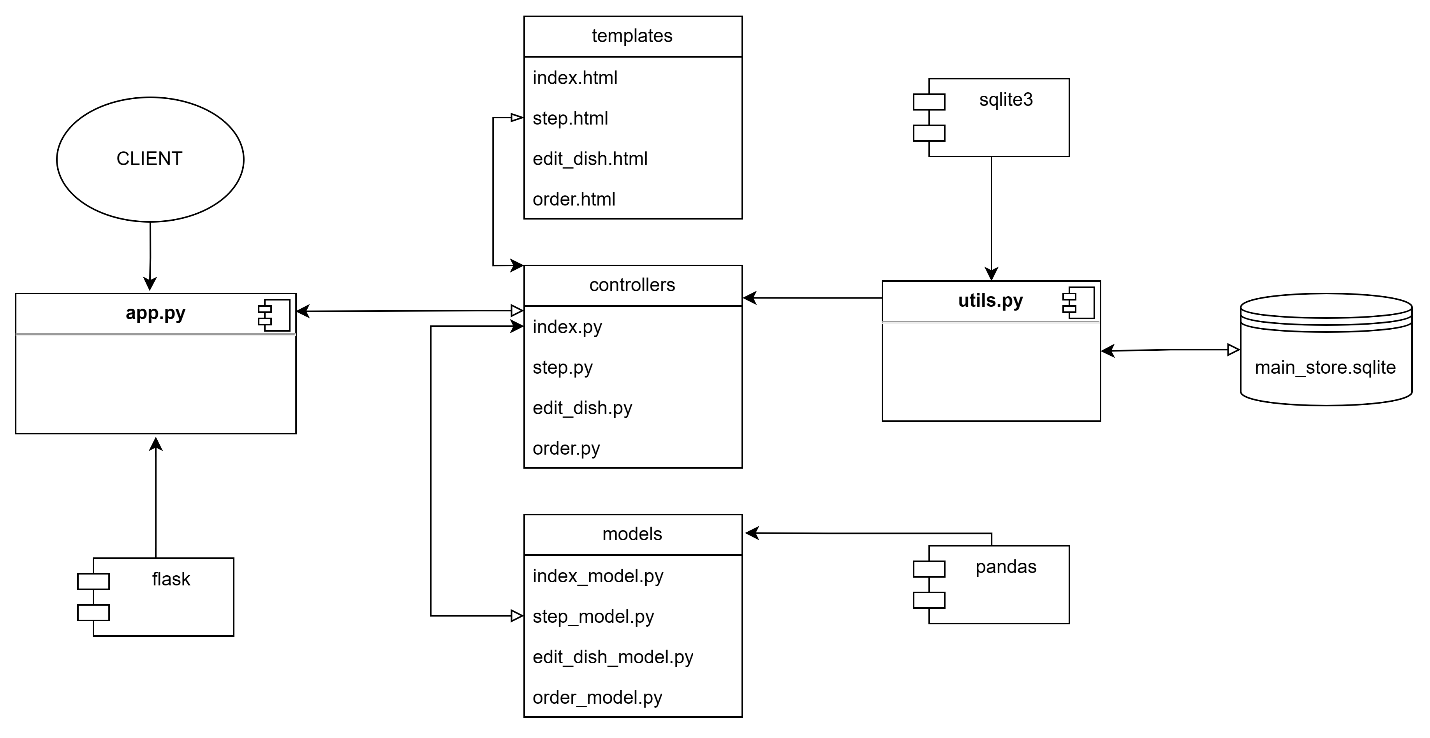


Рисунок 2.3.1 – Структура модулей приложения

**2.4 Отображение логической модели базы данных в среду *SQLite***

Таблице *buy* из логической модели базы данных сопоставим таблицу *buy* в среде *SQLite:*

|  |  |
| --- | --- |
| Таблица buy | |
| Атрибуты | CREATE TABLE buy (  id\_buy INTEGER PRIMARY KEY AUTOINCREMENT,  date\_time TEXT,  name\_guest TEXT  ) |

Таблице buy\_dishиз логической модели базы данных сопоставим таблицу buy\_dish в среде *SQLite:*

|  |  |
| --- | --- |
| Таблица *room* | |
| Атрибуты | CREATE TABLE buy\_dish (  id\_buy\_dish INTEGER PRIMARY KEY AUTOINCREMENT,  id\_dish INTEGER,  id\_buy INTEGER,  amount INTEGER,  price INTEGER,  FOREIGN KEY (id\_dish) REFERENCES dish (id\_dish) ON DELETE CASCADE,  FOREIGN KEY (id\_buy) REFERENCES buy (id\_buy) ON DELETE CASCADE  ) |

Таблице buy\_dish\_productиз логической модели базы данных сопоставим таблицу buy\_dish\_productв среде *SQLite*:

|  |  |
| --- | --- |
| Таблица buy\_dish\_product | |
| Атрибуты | CREATE TABLE buy\_dish\_product (  id\_buy\_dish\_product INTEGER PRIMARY KEY AUTOINCREMENT,  id\_buy\_dish INTEGER,  id\_dish\_product INTEGER,  FOREIGN KEY (id\_buy\_dish) REFERENCES buy\_dish (id\_buy\_dish) ON DELETE CASCADE,  FOREIGN KEY (id\_dish\_product) REFERENCES dish\_product (id\_dish\_product) ON DELETE CASCADE  ) |

Таблице dishиз логической модели базы данных сопоставим таблицу dishв среде *SQLite*:

|  |  |
| --- | --- |
| Таблица dish | |
| Атрибуты | CREATE TABLE dish (  id\_dish INTEGER PRIMARY KEY AUTOINCREMENT,  name\_dish TEXT,  type TEXT,  image TEXT  ) |

Таблице dish\_product из логической модели базы данных сопоставим таблицу dish\_product в среде *SQLite:*

|  |  |
| --- | --- |
| Таблица dish\_product | |
| Атрибуты | CREATE TABLE dish\_product (  id\_dish\_product INTEGER PRIMARY KEY AUTOINCREMENT,  id\_product INTEGER,  id\_dish INTEGER,  amount\_product INTEGER,  FOREIGN KEY (id\_product) REFERENCES product (id\_product) ON DELETE CASCADE,  FOREIGN KEY ("id\_dish") REFERENCES dish (id\_dish) ON DELETE CASCADE  ) |

Таблице productиз логической модели базы данных сопоставим таблицу productв среде *SQLite:*

|  |  |
| --- | --- |
| Таблица product | |
| Атрибуты | CREATE TABLE product (  id\_product INTEGER PRIMARY KEY AUTOINCREMENT,  name\_product TEXT,  calorie TEXT,  price INTEGER  ) |

Таблице step из логической модели базы данных сопоставим таблицу step в среде *SQLite:*

|  |  |
| --- | --- |
| Таблица step | |
| Атрибуты | CREATE TABLE step (  id\_step INTEGER PRIMARY KEY AUTOINCREMENT,  name\_step VARCHAR(50),  step\_image TEXT  ) |

Таблице step\_buy из логической модели базы данных сопоставим таблицу step\_buy в среде *SQLite:*

|  |  |
| --- | --- |
| Таблица step\_buy | |
| Атрибуты | CREATE TABLE step\_buy (  id\_step\_buy INTEGER PRIMARY KEY AUTOINCREMENT,  id\_step INTEGER,  id\_buy INTEGER,  FOREIGN KEY (id\_step) REFERENCES step (id\_step) ON DELETE CASCADE,  FOREIGN KEY (id\_buy) REFERENCES buy (id\_buy) ON DELETE CASCADE  ) |

**Заключение**

Цель достигнута. В рамках выполнения проекта был реализован сервис «*Бистро*», решающий проблемы клиентов, связанные с выбором блюд и оформлением заказа.

Для достижения цели были выполнены:

* Анализ предметной области;
* Исследование с помощью метода персонажа;
* Сценарии использования;
* Прототип интерфейса;
* Проект базы данных.