МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет Компьютерных наук

Кафедра информационных технологий управления

Доставка еды с отслеживанием заказа в реальном времени «Purrfect Bites»

«Технологии программирования»

09.03.02 Информационные системы и технологии
Информационные системы и технологии в управлении предприятием

Зав. кафедрой _____д.т.н., профессор А.О. Сирота __.__20__

Обучающийся _____К.В. Дунаева, 3 курс, д/о

Обучающийся _____А.С. Жданова, 3 курс, д/о

Обучающийся _____Е.Н. Лобова, 3 курс, д/о

Руководитель _____ В.С. Тарасов, ст. преподаватель

Руководитель _____ В.А. Ушаков, преподаватель

Воронеж 2024

Содержание

Введение

В современном мире, где скорость и удобство играют ключевую роль, услуги доставки еды становятся все более востребованными. Современный образ жизни требует эффективных решений для удовлетворения потребностей в питании, сохраняя при этом комфорт и уровень сервиса. В этом контексте технологии играют решающую роль, предоставляя новые возможности для оптимизации процесса доставки и повышения удовлетворенности клиентов.

Доставка еды с отслеживанием заказа в реальном времени - это инновационный подход к обеспечению качественного сервиса. Системы отслеживания заказов позволяют клиентам быть в курсе всех этапов выполнения заказа: от момента размещения заказа до его доставки. Это обеспечивает прозрачность и надежность процесса, позволяя клиентам быть уверенными в том, что их заказ обрабатывается эффективно и в срок.

Кроме того, отслеживание заказа в реальном времени предоставляет клиентам удобство и контроль над процессом доставки. Они могут отслеживать местоположение курьера и ожидать доставки в удобное для них время, что экономит их время и обеспечивает комфорт.

В данной курсовой работе был реализован сервис доставки, который будет предоставлять пользователям возможность отследить заказ в реальном времени.

1 Постановка задачи

Данный проект предназначен для обеспечения максимального удовлетворения потребностей клиентов и создания положительного опыта пользования услугами доставки еды. Взаимодействие с клиентами на протяжении всего процесса, начиная с создания заказа и заканчивая доставкой, позволяет улучшить качество обслуживания и повысить лояльность клиентов.

Помимо этого, целью данного проекта является создание сервиса доставки с отслеживанием в реальном времени. Этот сервис предоставит клиентам возможность выбирать и заказывать товары из определенного ассортимента, а также отслеживать их в реальном времени.

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

К разрабатываемому сервису выдвигаются следующие функциональные требования:

— получения информации о составе товаров;

— просмотра категории товаров;

— выбора товаров и оформления заказов с доставкой по указанному адресу авторизованными пользователями;

— связи с компанией, предоставляющей услуги;

— добавления, редактирования товаров и удаления их администратором;

— добавления, редактирования и удаления информации о сотрудниках администратором;

— выбора заказа для выполнения курьером;

— просмотра списка принятых заказов курьером.

1.1.2 Нефункциональные требования

К разрабатываемому сервису выдвигаются следующие нефункциональные требования:

— Сервис должен обладать интерфейсом, выполненном в едином стиле со всем необходимым набором функций.

— Сервис должен использовать современные технологии и инструменты разработки.

1.2 Требования к архитектуре

Список требований к архитектуре:

— Приложение должно быть построено с использованием протоколов HTTP.

— Для хранения информации необходимо использовать реляционную базу данных.

— Клиентская часть приложения должна быть написана с использованием технологий frontend разработки, таких как HTML, CSS, JavaScript с фреймворком React. Выбор этого фреймворка объясняется тем, что он обладает простым синтаксисом, позволяет обновлять только те элементы, которые требуют изменений и использовать повторно уже существующие элементы.

— Серверная часть приложения должна быть написана с использованием технологий backend разработки, таких как Python и Django [1] на основе архитектурного паттерна MVC. Выбор этого

фреймворка объясняется тем, что он включает в себя большое количество готового функционала. И, как правило, проекты, написанные на данном фреймворке, обладают быстрой загрузкой, могут хранить огромные данные на сервере и по умолчанию создают панель администратора для редактирования информации на сайте.

— Мобильное приложение будет разработано с использованием языка программирования Python и фреймворка Kivy на основе архитектурного паттерна MVC. Выбор Kivy обусловлен его многофункциональностью и гибкостью, что позволяет быстро создавать кроссплатформенные мобильные приложения с интерактивным пользовательским интерфейсом.

1.3 Задачи, решаемые в процессе разработки

Процесс организации данного веб-приложения построен на основе гибкой методологии Kanban.

В процессе разработки интернет-каталога кондитерских изделий будут решаться следующие задачи:

— Анализ предметной области: необходимо изучить особенности работы интернет-каталога кондитерских изделий.

— Проектирование базы данных: на основе полученных требований необходимо разработать структуру базы данных, которая будет использоваться в приложении.

— Разработка серверной части приложения: на этом этапе необходимо разработать серверную часть приложения, которая будет отвечать за обработку запросов клиента и взаимодействие с базой данных. Для этого используется фреймворк Django.

— Разработка клиентской части приложения: клиентская часть приложения должна быть написана с использованием современных технологий frontend разработки, таких как HTML, CSS, JavaScript.

— Разработка мобильного приложения для курьеров: на этом этапе необходимо создать мобильное приложение, которое будет использоваться курьерами для выполнения доставки заказов. Приложение должно обеспечивать удобный интерфейс для просмотра списка заказов, отслеживания маршрута, взаимодействия с клиентами и обновления статуса доставки.

— Тестирование и отладка: на этом этапе необходимо провести тестирование и отладку приложения, чтобы убедиться, что оно соответствует требованиям, определенным в начале проекта.

2 Анализ предметной области

2.1 Терминология (глоссарий) предметной области

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Клиент (клиентская сторона) – сайт, который предоставляет пользователю взаимодействовать со всей системой.

Сервер (серверная часть) – компьютер, обслуживающий другие устройства (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач.

Backend – логика работы сайта, внутренняя часть продукта, которая находится на сервере и скрыта от пользователя.

Frontend – презентационная часть информационной или программной системы, ее пользовательский интерфейс и связанные с ним компоненты.

GitHub – веб-сервис для хостинга IT-проектов и их совместной разработки.

Неавторизованный пользователь — пользователь, не прошедший авторизацию или не зарегистрированный в системе.

Авторизованный пользователь — пользователь, прошедший авторизацию в системе.

MVC — схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер - таким

образом, что модификация каждого компонента может осуществляться независимо.

CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки (HTML, XHTML, XML).

HTML – стандартизированный язык гипертекстовой разметки веб-страниц в браузере.

JavaScript – язык программирования высокого уровня, который используется для написания frontend- и backend-частей сайтов, а также мобильных приложений.

PostgreSQL – реляционная база данных с открытым кодом.

React – JavaScript-библиотека для создания пользовательских интерфейсов [2].

Сериализация — это процесс преобразования объектов Python в поток байтов, который может быть сохранен в файле или передан по сети.

Десериализация — это процесс получения потока байтов и преобразования его в объект Python.

Docker — это открытая платформа для разработки, доставки и работы с приложениями. С ее помощью можно создавать и развертывать приложения в легковесных, подключаемых контейнерах, которые могут быть запущены и масштабированы на любой платформе.

2.2 Обзор аналогов

2.2.1 Додо Пицца

Додо – сеть кафе с возможностью доставки и самовывоза заказа. Обладает широким ассортиментом продуктов (пицца, напитки, паста, десерты и т.д.). Интерфейс приложения представлен на Рисунке 1.

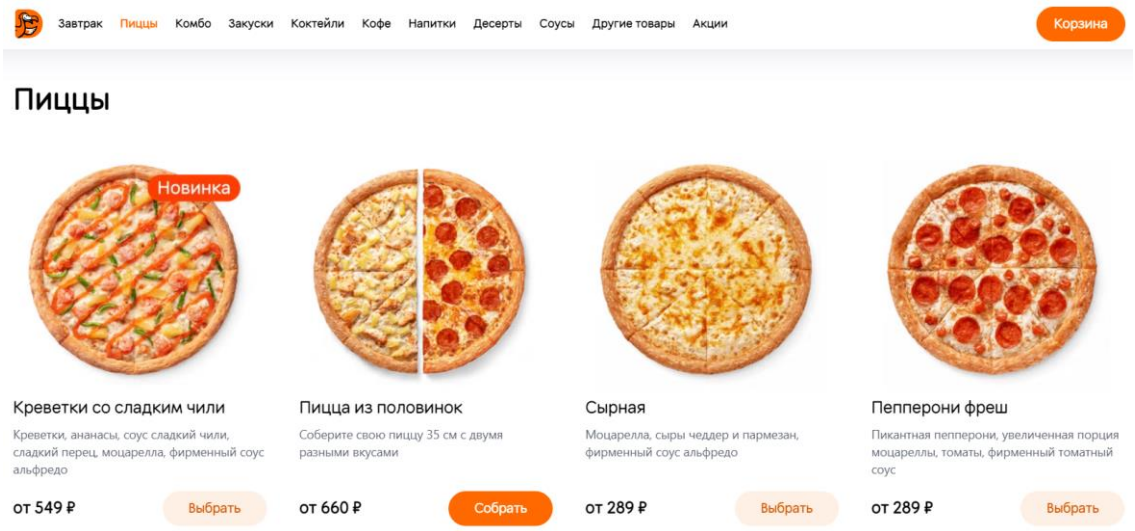

Рисунок 1 - Внешний вид каталога «Додо Пиццы»

Есть возможность посмотреть состав товаров, добавить нужные позиции в корзину. У приложения Додо весьма удобный интерфейс. Из минусов можно отметить отсутствие отслеживания заказа в реальном времени.

2.2.2 Пицца Куба

Куба - сервис по доставке пиццы. Ассортимент весьма скромен, в меню лишь несколько пицц, о других товарах речи не идет. Отслеживание в реальном времени отсутствует, интерфейс не интуитивно-понятный. Из плюсов Куба обладает лишь возможностями просмотра состава позиций и добавлением нужных товаров в корзину. Интерфейс приложения представлен на Рисунке 2.

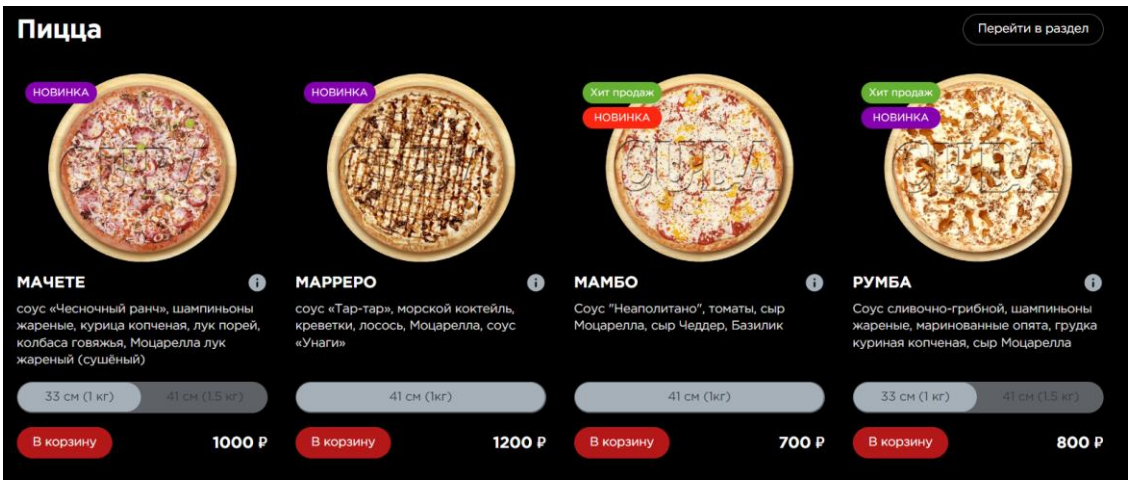

Рисунок 2 - Внешний вид каталога «Пицца Куба»

2.2.3 Неместные

Неместные - сервис по доставке блюд. Обладает высоким ассортиментом (суши и пицца), можно посмотреть состав позиций и добавить товары в корзину. Из минусов: отсутствие отслеживания заказа в реальном времени и удобного интерфейса. Интерфейс приложения представлен на Рисунке 3.

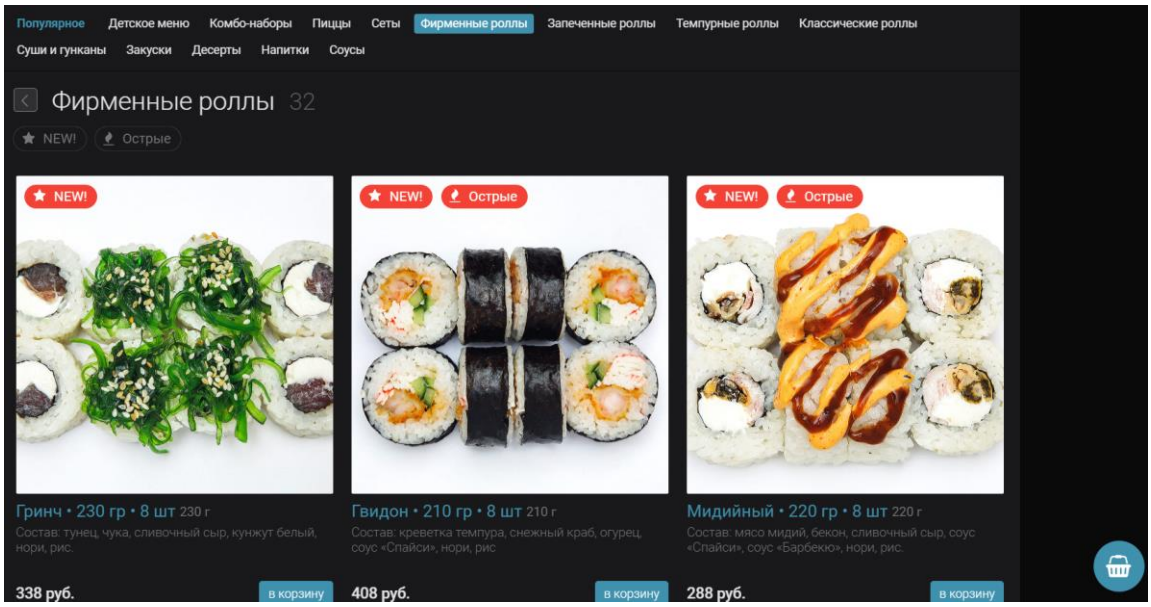

Рисунок 3 - Внешний вид каталога «Неместные»

2.2.4 Сравнительная таблица аналогов

Для удобства восприятия здесь представлена сравнительная таблица аналогов.

	ДОДО ПИЦЦА	cuba	
Широкий ассортимент	+	-	+
Состав позиции	+	+	+
Корзина	+	+	+
Отслеживание заказа в реальном времени	-	-	-
Удобный интерфейс	+	-	-

Рисунок 4 - Обзор аналогов

2.3 Диаграммы, иллюстрирующие работу системы

2.3.1 Диаграмма прецендентов (Use case)

Диаграмма прецендентов (Use case) представлена для четырех типов акторов: неавторизованного пользователя, авторизованного пользователя, администратора и курьера. У каждого из них своя модель поведения, которую можно проследить на Рисунках 5-8.

Неавторизованный пользователь может:

— Зарегистрироваться в системе.

— Авторизоваться в системе.

— Просматривать категории товаров каталога.

— Просматривать товары по выбранной категории.

— Взаимодействовать с корзиной: просматривать, удалять и добавлять товары в корзину.

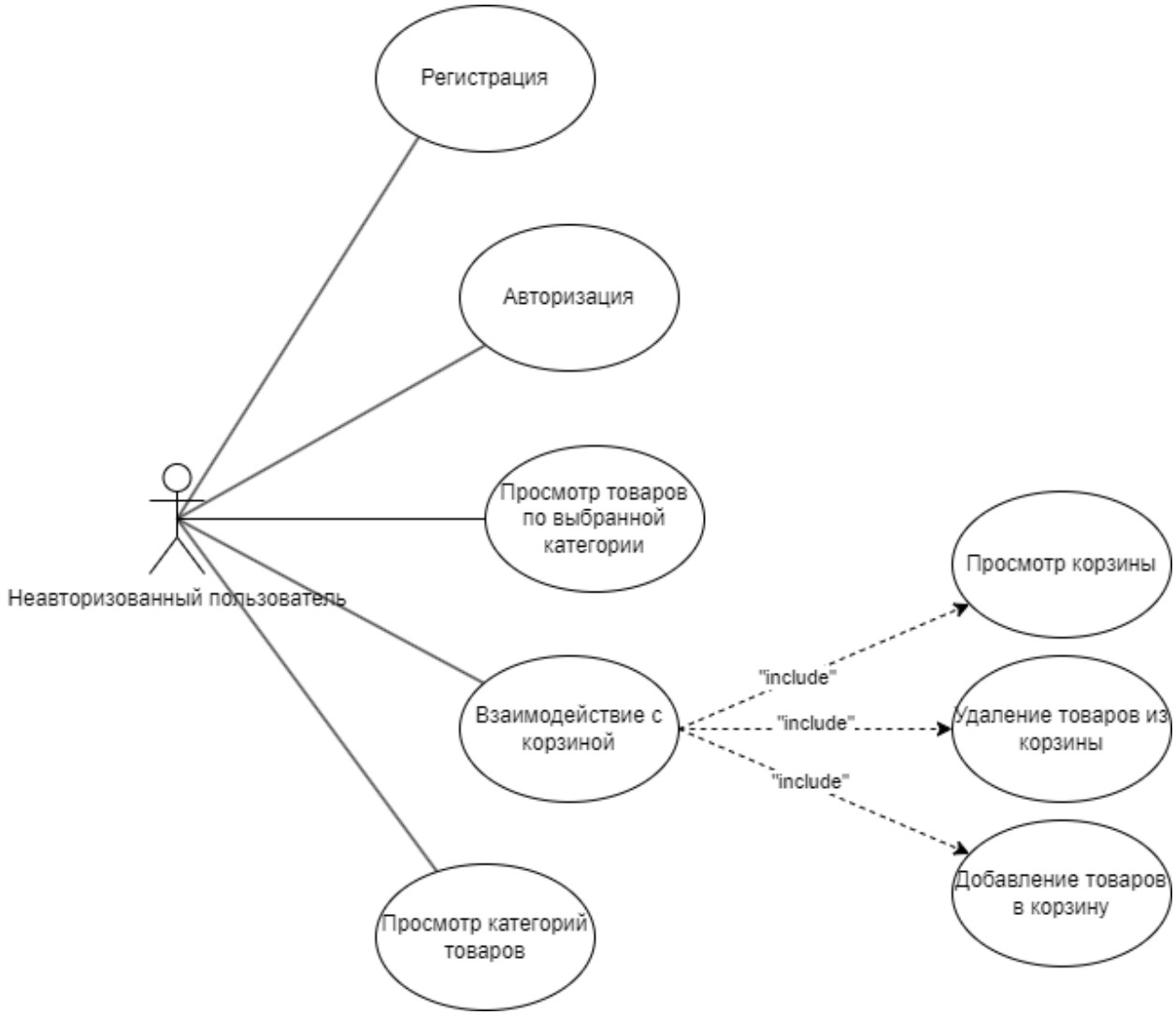

Рисунок 5 - Диаграмма прецедентов (Use case) для неавторизованного пользователя

Авторизованный пользователь помимо функций, доступных неавторизованному пользователю, может:

— Взаимодействовать с личным кабинетом: просматривать и изменять информацию, выходить из профиля.

— Взаимодействовать с корзиной, где помимо просмотра и добавления товаров становится доступна функция заказа товара из корзины.

— Отслеживать готовность заказа.

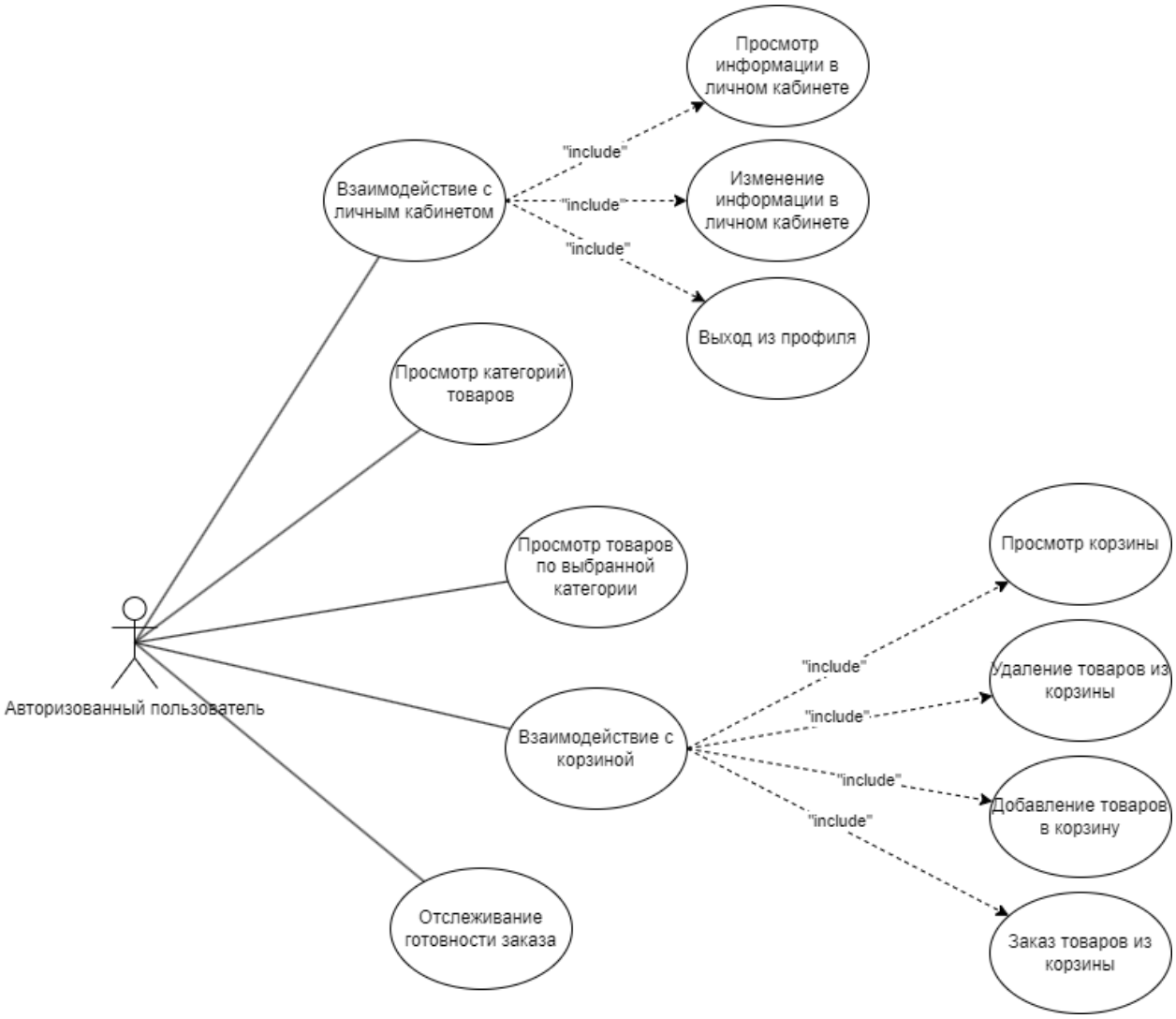

Рисунок 6 - Диаграмма прецедентов (Use case) для авторизованного пользователя

Существуют следующие функции, доступные администратору:

— Управлять информацией о товарах: добавление, удаление и редактирование.

— Взаимодействовать с личным кабинетом: просматривать и изменять информацию, выходить из профиля.

— Управлять информацией о пользователях: удаление, просмотр данных.

— Управлять информацией о курьерах: просмотр, добавление, удаление и редактирование.

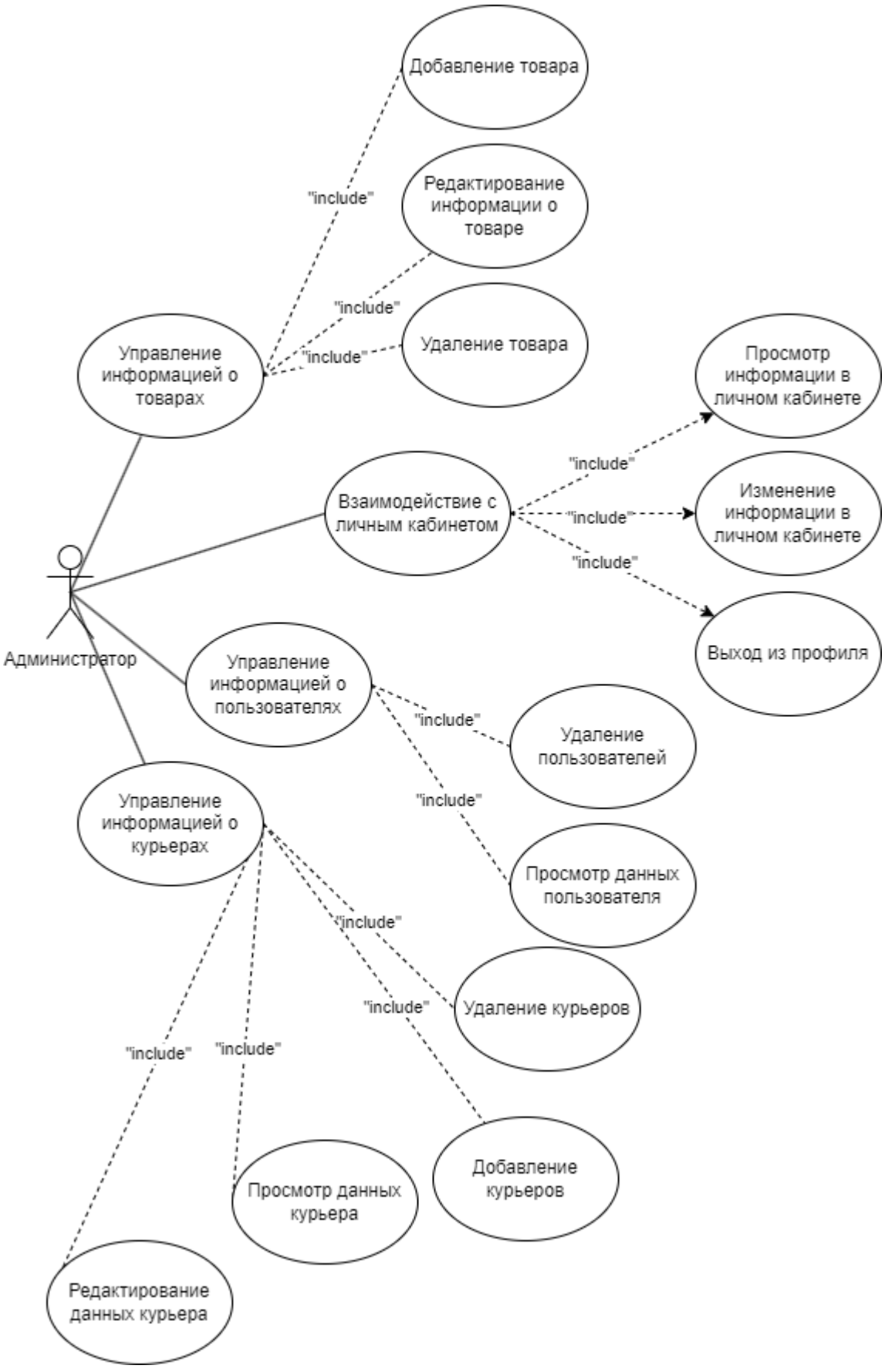

Рисунок 7 - Диаграмма прецедентов (Use case) для администратора

Также существуют следующие функции, доступные курьеру:

— Взаимодействовать с личным кабинетом: просматривать информацию, выходить из профиля.

— Авторизоваться в системе.

— Просматривать информацию о заказчике.

— Взаимодействовать с заказом: начать и завершить доставку, отказаться от доставки.

Рисунок 8 - Диаграмма прецедентов (Use case) для курьера

2.3.2 Диаграмма последовательности (Sequence diagram)

Существует также диаграмма последовательностей (Рисунок 9), на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента. Участником данной системы является пользователь, а объектами – клиент, сервер и база данных.

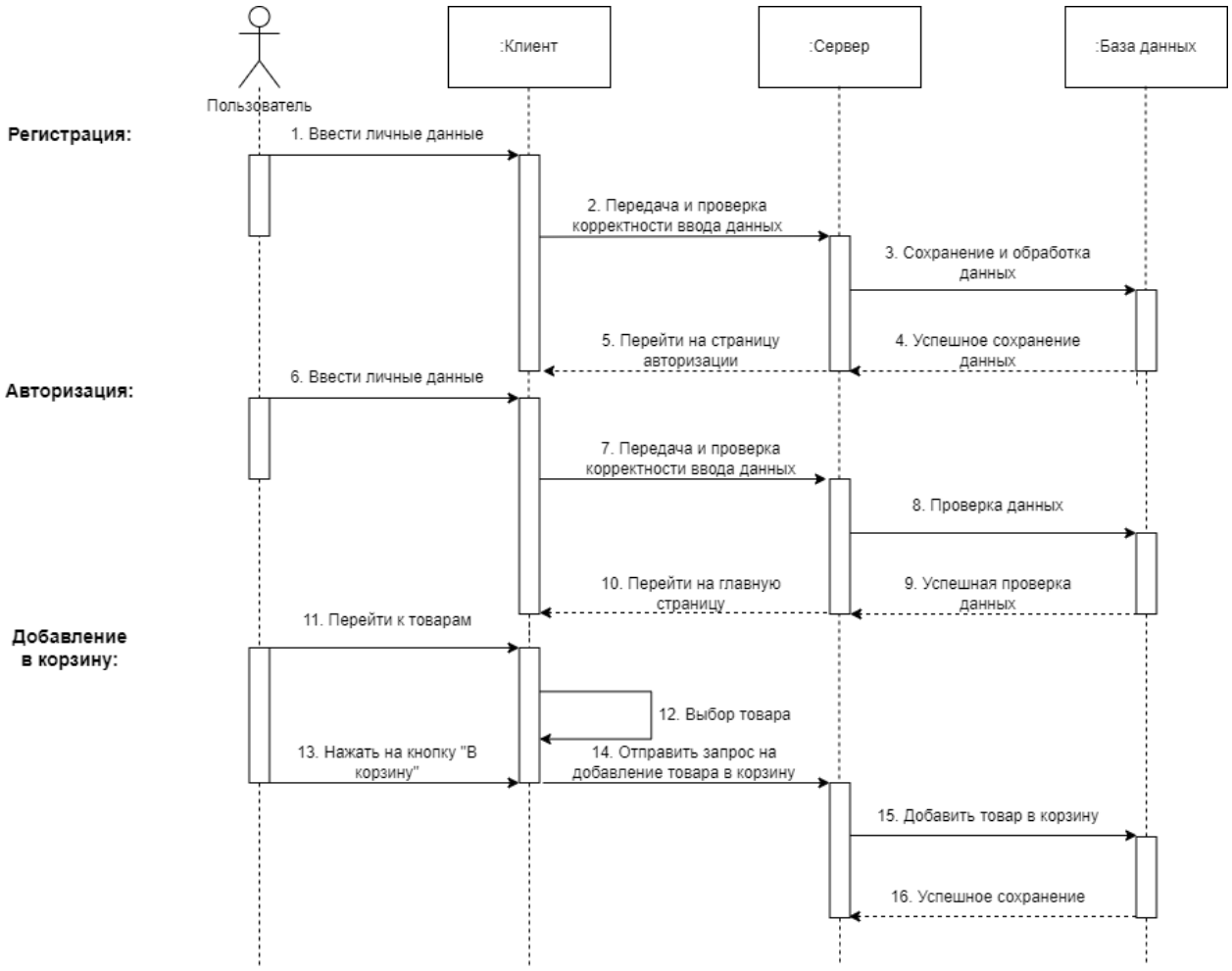

Рисунок 9 - Диаграмма последовательности

2.3.3 Диаграмма состояний (Statechart diagram)

Диаграмма состояний (Рисунок 10) отражает внутренние состояния объекта в течение его жизненного цикла от момента создания до разрушения. На данной диаграмме рассмотрены состояния от момента входа в систему до полного выхода из нее.

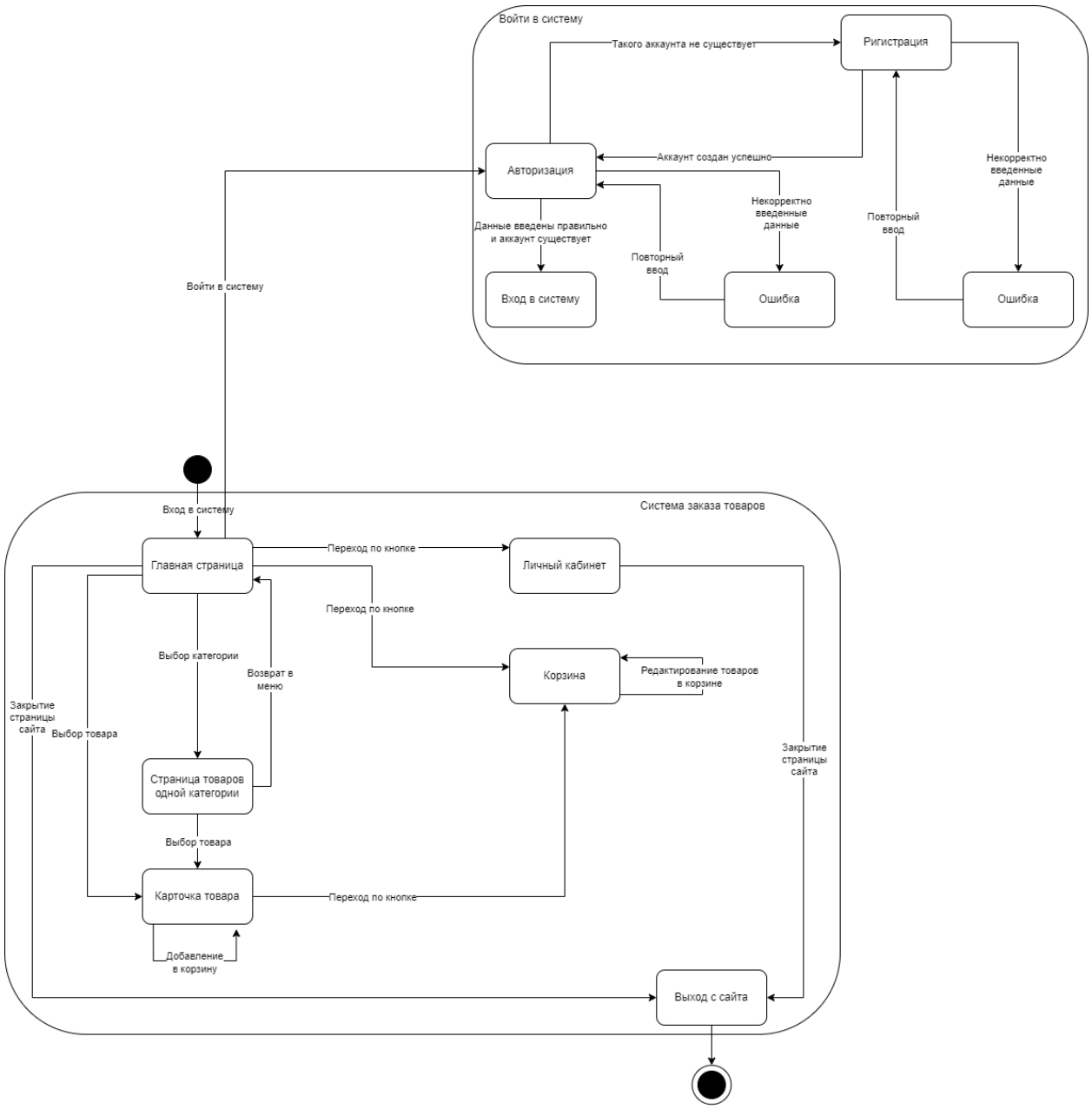

Рисунок 10 - Диаграмма состояний

2.3.4 Диаграмма деятельности (Activity diagram)

Диаграмма деятельности (Рисунок 11) представляет собой диаграмму, на которой показаны действия, состояния которых описаны на диаграмме состояний. Она описывает действия системы или людей, выполняющих действия, и последовательный поток этих действий.

В данном случае рассмотрен путь действий пользователя.

Диаграмма показывает, что пользователь, находясь в неавторизованной зоне системы не может заходить на свой профиль и заказывать товары.

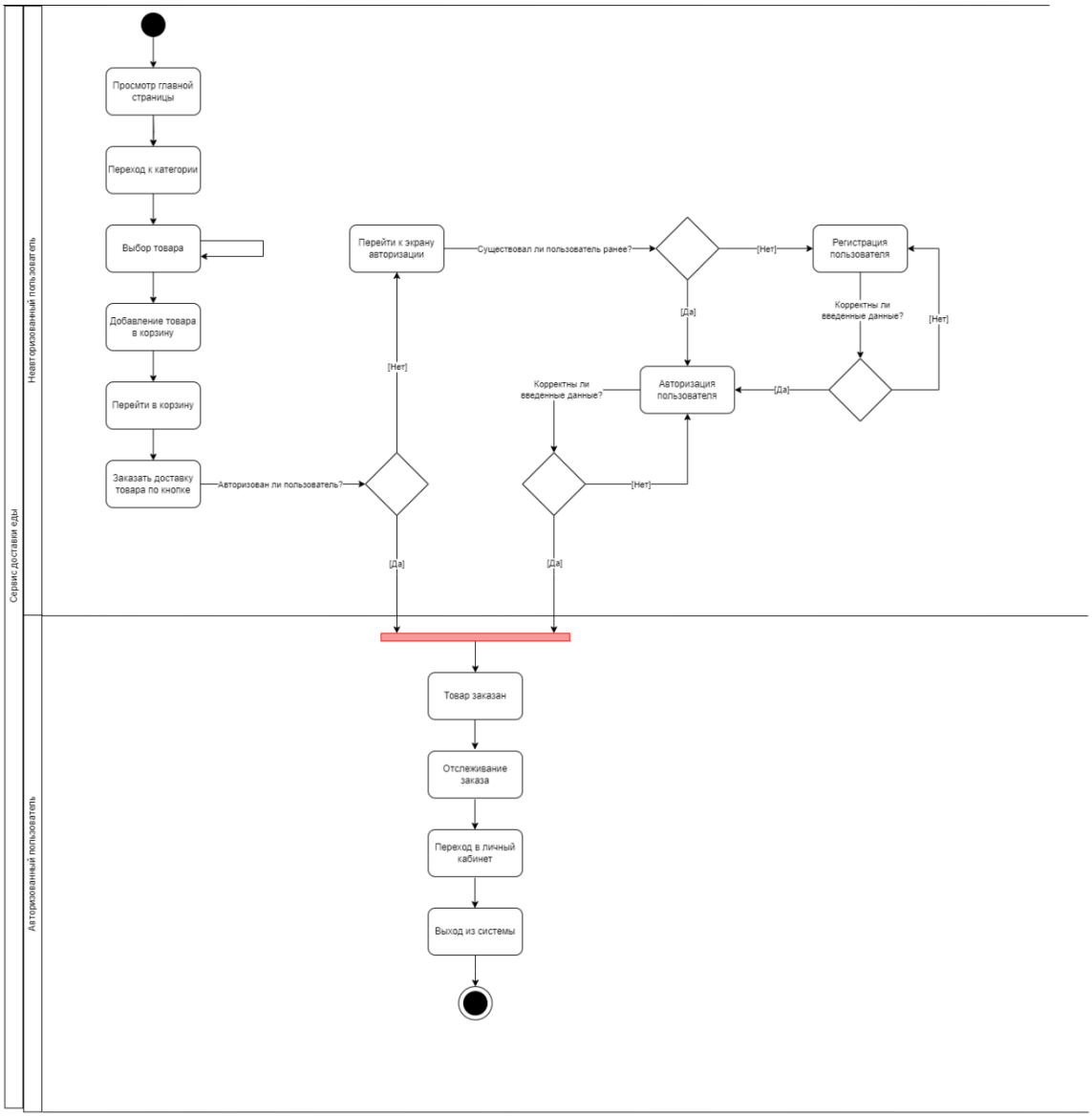

Рисунок 11 - Диаграмма деятельности (активности)

2.3.5 Диаграмма классов (Class diagram)

Диаграмма классов (Рисунок 12) демонстрирует общую структуру иерархии классов системы, их кооперацій, атрибутов, методов, интерфейсов и взаимосвязей между ними. В данной системе рассмотрены следующие классы:

— Класс «Пользователь».

— Класс «Продукт».

— Класс «Категории продуктов».

У каждого из классов существуют свои атрибуты.

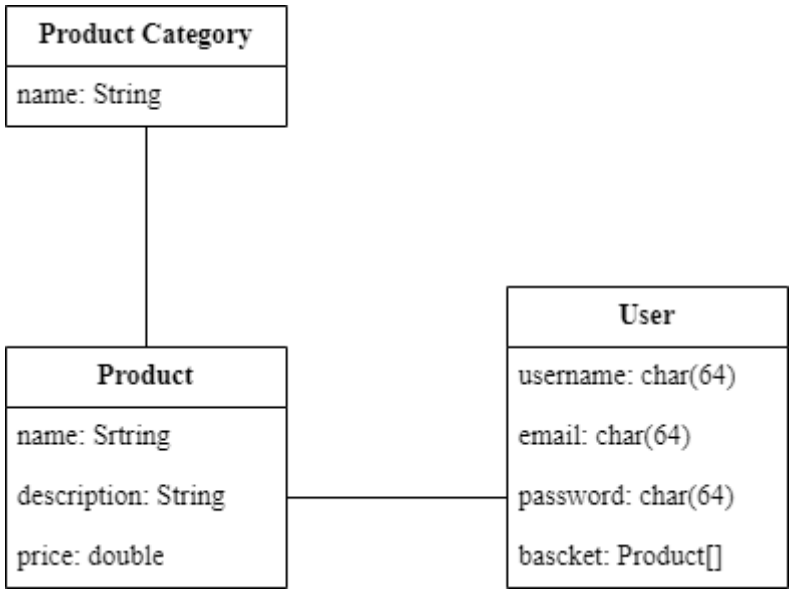

Рисунок 12 - Диаграмма классов

2.3.6 Диаграмма объектов (Object diagram)

По подобию диаграммы классов была выполнена диаграмма объектов. (Рисунок 13).

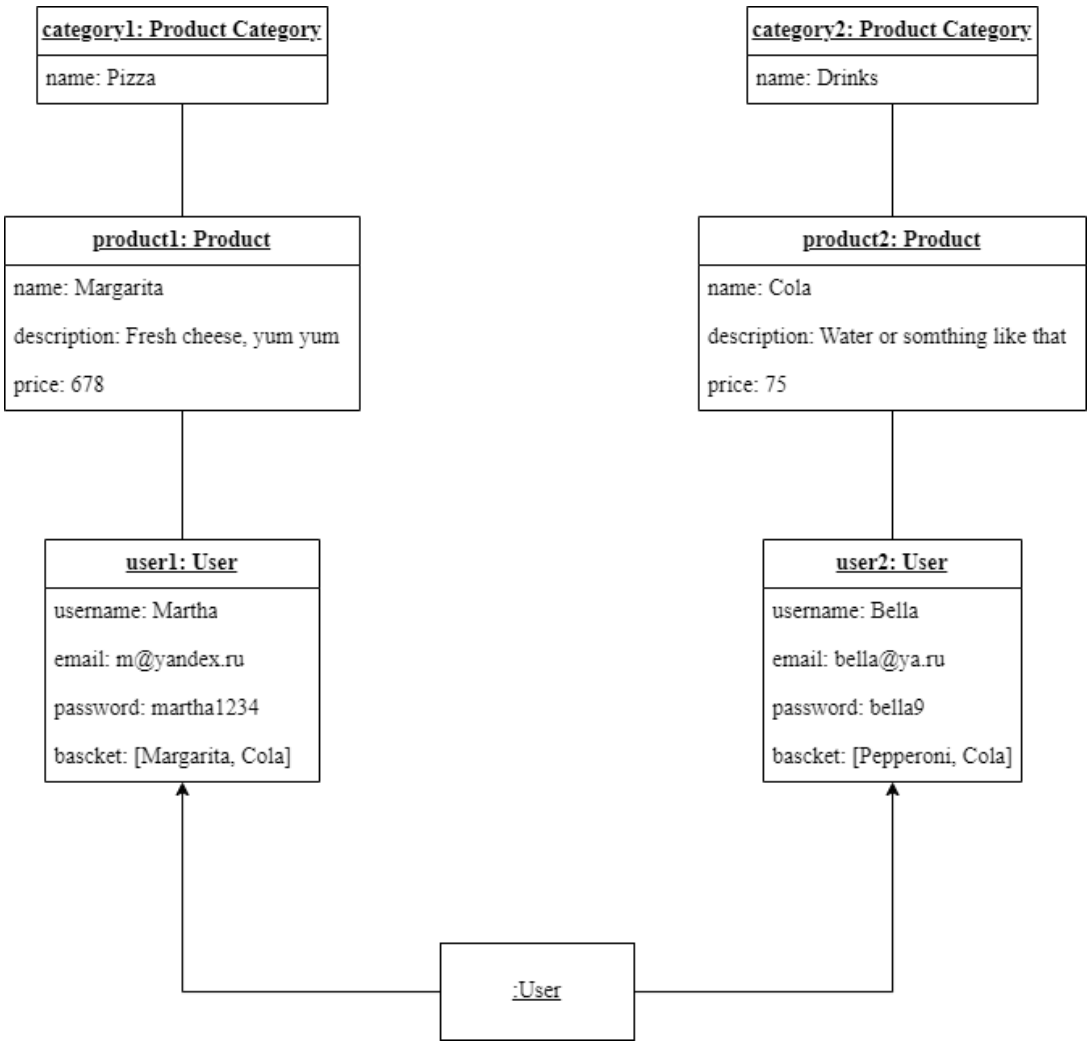

Рисунок 13 - Диаграмма объектов

2.3.7 Диаграмма сотрудничества (Collaboration diagram)

Диаграмма сотрудничества (Рисунки 14-16) — это вид диаграммы взаимодействия, в котором основное внимание сосредоточено на структуре взаимосвязей объектов, принимающих и отправляющих сообщения.

Рисунок 14 - Диаграмма сотрудничества при авторизации

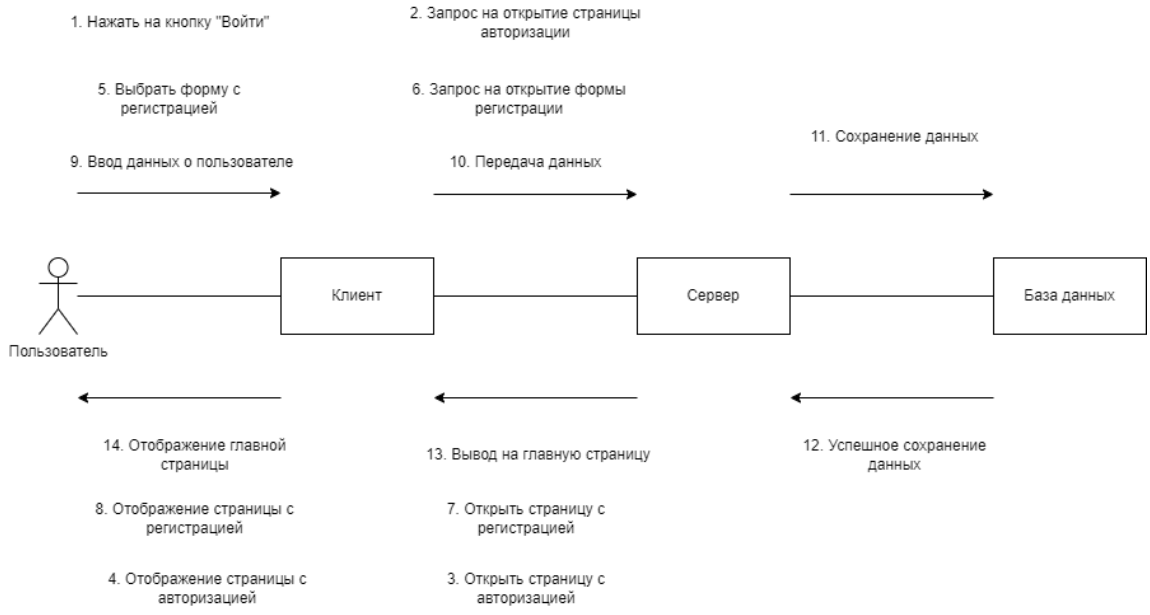

Рисунок 15 - Диаграмма сотрудничества при регистрации

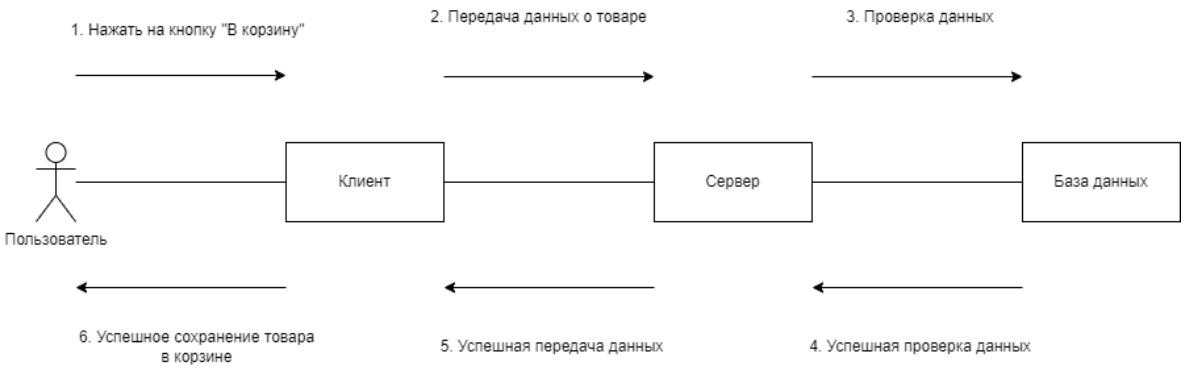

Рисунок 16 - Диаграмма сотрудничества при добавлении в корзину

2.3.8 Диаграмма развертывания (Deployment diagram)

Диаграмма развертывания (Рисунок 17) предназначена для представления общей конфигурации или топологии распределенной программной системы.

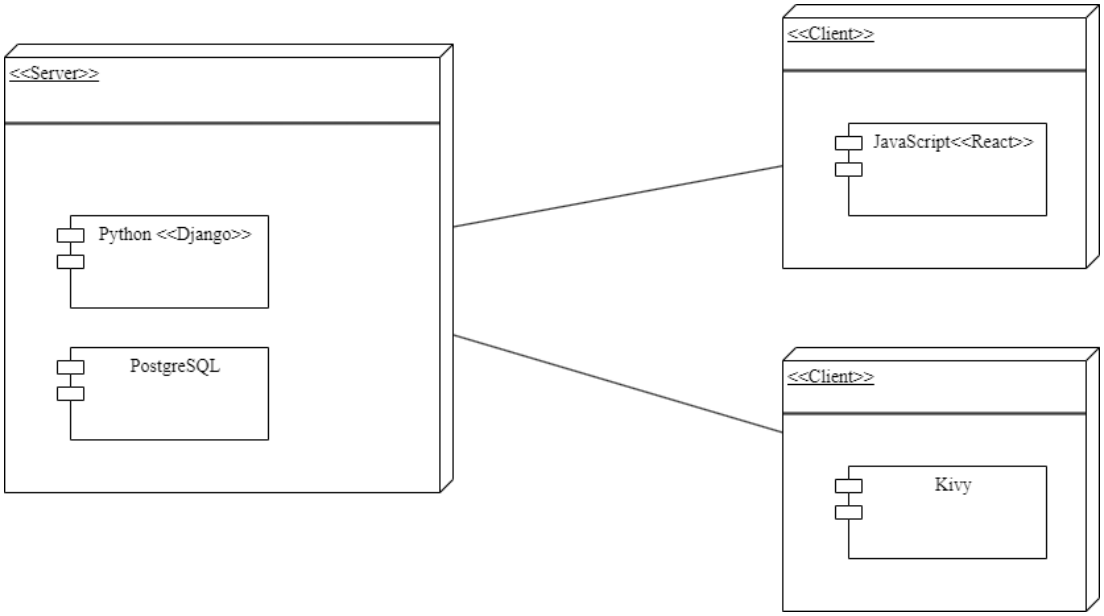

Рисунок 17 - Диаграмма развертывания

2.3.9 ER-диаграмма

ER-диаграмма — это графическое представление модели данных, которая используется для описания концептуальной структуры базы данных. В такой диаграмме есть сущности, которые представляют объекты, с которыми работает система, и связи между сущностями, описывающие их взаимодействия. ER-диаграмма помогает наглядно описать структуру базы данных и увидеть связи между ее элементами (Рисунок 18).

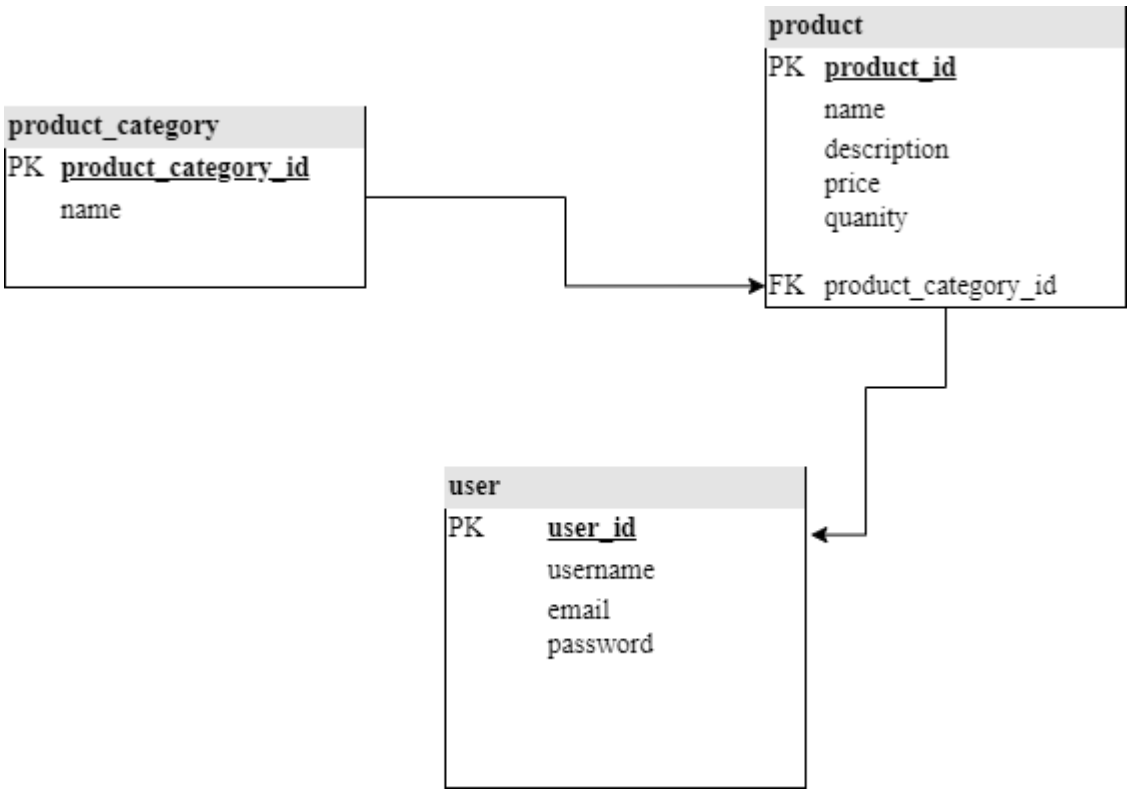

Рисунок 18 - ER-диаграмма