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»

Курсовая работа по дисциплине  
«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

[Содержание 2](#_Toc169863034)

[Введение 4](#_Toc169863035)

[1 Постановка задачи 5](#_Toc169863036)

[1.1 Требования к разрабатываемой системе 5](#_Toc169863037)

[1.1.1 Функциональные требования 5](#_Toc169863038)

[1.1.2 Нефункциональные требования 6](#_Toc169863039)

[1.2 Требования к архитектуре 6](#_Toc169863040)

[1.3 Задачи, решаемые в процессе разработки 7](#_Toc169863041)

[2 Анализ предметной области 8](#_Toc169863042)

[2.1 Терминология (глоссарий) предметной области 8](#_Toc169863043)

[2.2 Обзор аналогов 10](#_Toc169863044)

[2.2.1 Додо Пицца 10](#_Toc169863045)

[2.2.2 Пицца Куба 10](#_Toc169863046)

[2.2.3 Неместные 11](#_Toc169863047)

[2.2.4 Сравнительная таблица аналогов 12](#_Toc169863048)

[2.3 Диаграммы, иллюстрирующие работу системы 12](#_Toc169863049)

[2.3.1 Диаграмма прецендентов 12](#_Toc169863050)

[2.3.2 Диаграмма последовательности 16](#_Toc169863051)

[2.3.3 Диаграмма состояний 17](#_Toc169863052)

[2.3.4 Диаграмма деятельности 18](#_Toc169863053)

[2.3.5 Диаграмма классов 19](#_Toc169863054)

[2.3.6 Диаграмма объектов 20](#_Toc169863055)

[2.3.7 Диаграмма сотрудничества 21](#_Toc169863056)

[2.3.8 Диаграмма развертывания 23](#_Toc169863057)

[2.3.9 ER-диаграмма 23](#_Toc169863058)

[3 Реализация 25](#_Toc169863059)

[3.1 Средства реализации 25](#_Toc169863060)

[3.1.1 Средства реализации серверной части сайта 25](#_Toc169863061)

[3.1.2 Средства реализации клиентской части сайта 25](#_Toc169863062)

[3.1.3 Средства реализации мобильной части сервиса 26](#_Toc169863063)

[3.2 Реализация серверной (backend) части сайта 26](#_Toc169863064)

[3.3 Реализация клиентской (frontend) части сайта 31](#_Toc169863065)

[3.4 Реализация мобильного (mobile) приложения 34](#_Toc169863066)

[3.5 Навигация по сайту 37](#_Toc169863067)

[3.5.1 Для неавторизованного пользователя 37](#_Toc169863068)

[3.5.2 Для авторизованного пользователя 39](#_Toc169863069)

[3.5.3 Для администратора 43](#_Toc169863070)

[3.6 Навигация по мобильному приложению 45](#_Toc169863071)

[4 Тестирование 50](#_Toc169863072)

[4.1 Дымовое тестирование 50](#_Toc169863073)

[4.2 Тестирование пользовательского интерфейса 52](#_Toc169863074)

[Заключение 54](#_Toc169863075)

[Список использованной литературы 55](#_Toc169863076)

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. Требования к разрабатываемой системе
     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. 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. Требования к архитектуре

Список требований к архитектуре:

* сайт должен быть построен с использованием протоколов HTTP;
* для хранения информации необходимо использовать реляционную базу данных;
* клиентская часть сайта 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 сайта должна быть написана с использованием технологий backend разработки, таких как Python и Django 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. Выбор Kivy обусловлен его многофункциональностью и гибкостью, что позволяет быстро создавать кроссплатформенные мобильные приложения с интерактивным пользовательским интерфейсом.
  1. Задачи, решаемые в процессе разработки

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

В процессе разработки сервиса доставки будут решаться следующие задачи:

* анализ предметной области: необходимо изучить особенности работы сервиса доставки;
* 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 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.

1. Анализ предметной области
   1. Терминология (глоссарий) предметной области

**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** — cхема 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.

**Django** — Фреймворк для разработки веб-сервисов на Python [1].

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

**Kivy** — Фреймворк Python с открытым исходным кодом для разработки мобильных приложений и другого программного обеспечения [3].

**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.

* 1. Обзор аналогов
     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.

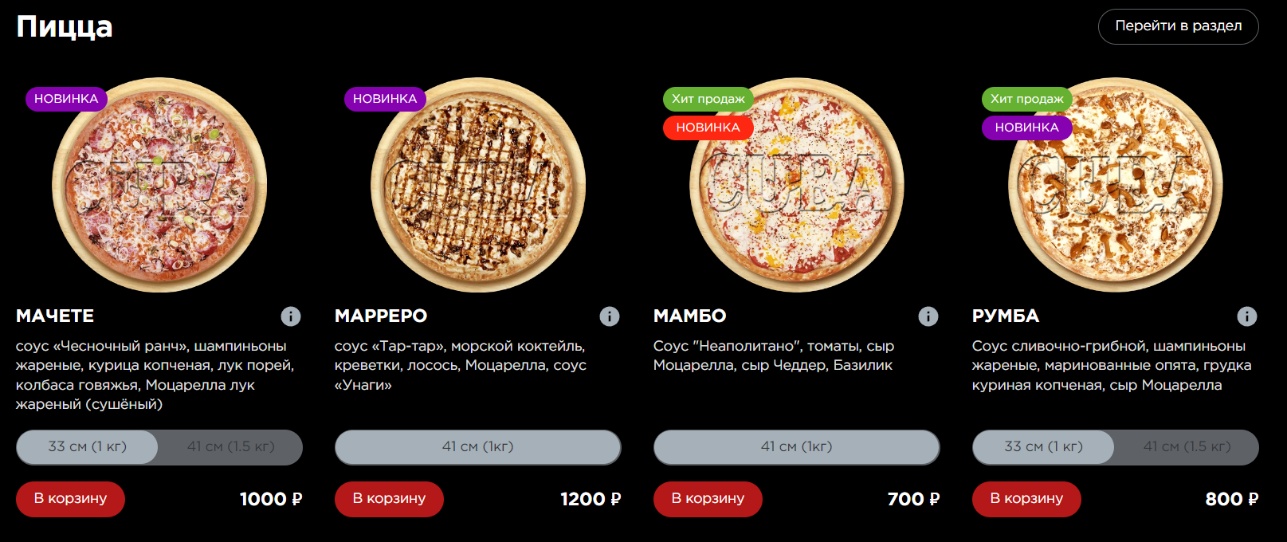


1. — Внешний вид каталога «Додо Пиццы»

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.

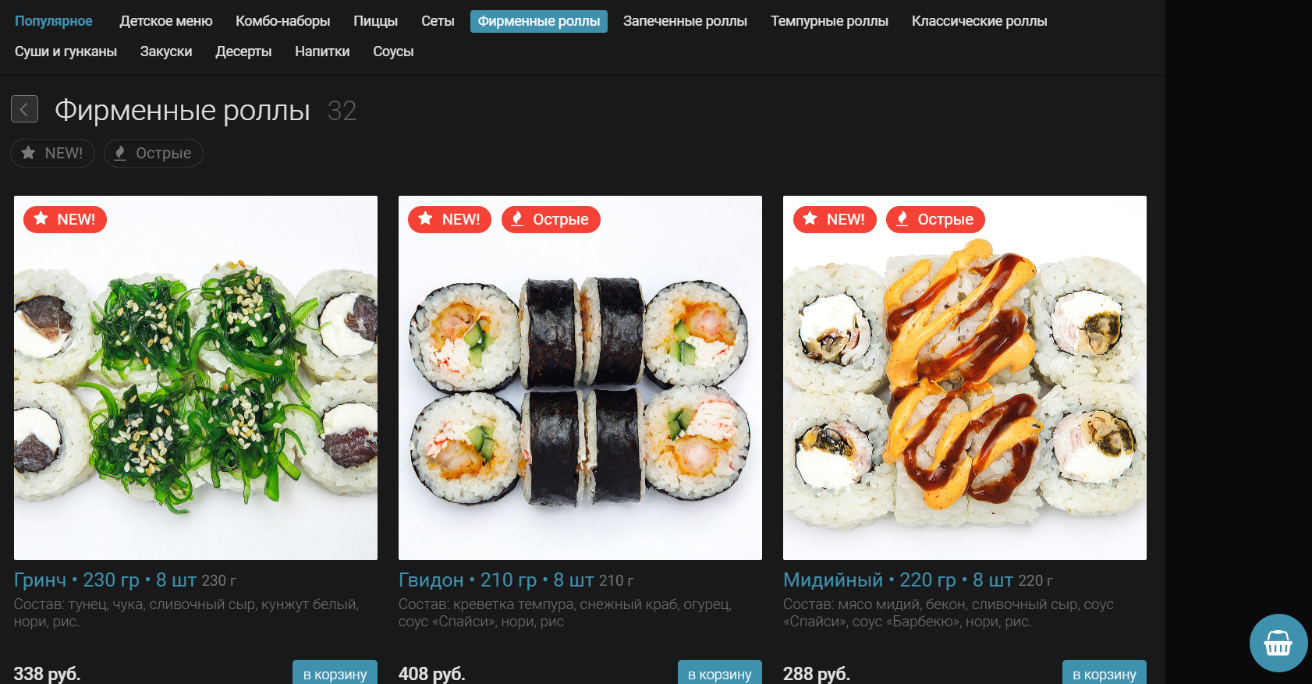
* + 1. Пицца Куба

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.



1. —Внешний вид каталога «Пицца Куба»
   * 1. 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.



1. — Внешний вид каталога «Неместные»
   * 1. Сравнительная таблица аналогов

Для удобства восприятия здесь представлена сравнительная таблица аналогов (Рисунок 4).

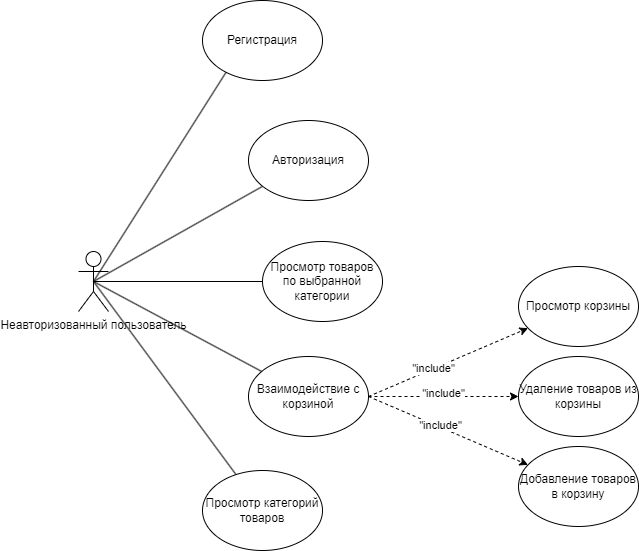


1. — Обзор аналогов
   1. Диаграммы, иллюстрирующие работу системы
      1. Диаграмма прецендентов

Диаграмма прецендентов 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:

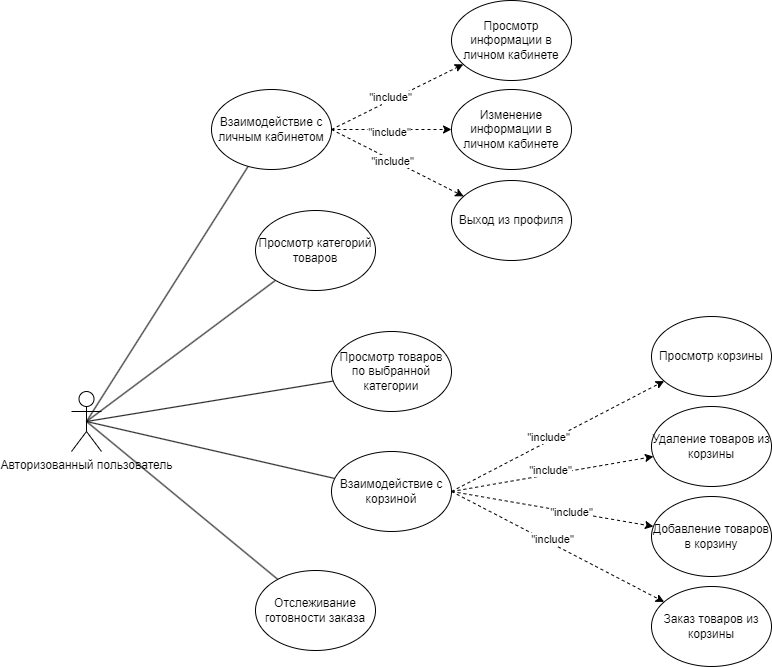
* зарегистрироваться в системе;
* авторизоваться в системе;
* просматривать категории товаров каталога;
* просматривать товары по выбранной категории;
* взаимодействовать с корзиной: просматривать, удалять и добавлять товары в корзину.



1. — Диаграмма прецедентов для неавторизованного пользователя

Авторизованный пользователь помимо функций, доступных неавторизованному пользователю, может:

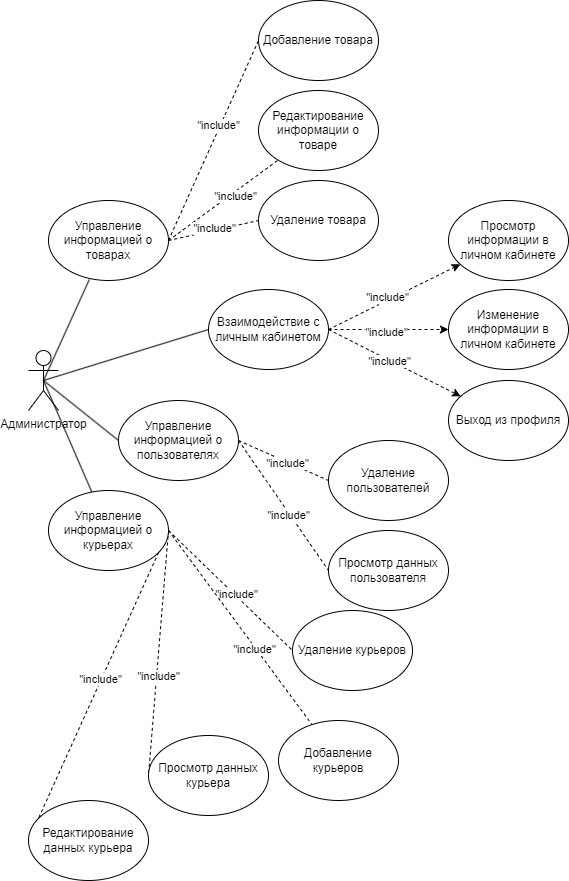
* взаимодействовать с личным кабинетом: просматривать и изменять информацию, выходить из профиля;
* взаимодействовать с корзиной, где помимо просмотра и добавления товаров становится доступна функция заказа товара из корзины;
* отслеживать готовность заказа.



1. —Диаграмма прецедентов для авторизованного пользователя

Существуют следующие функции, доступные администратору:

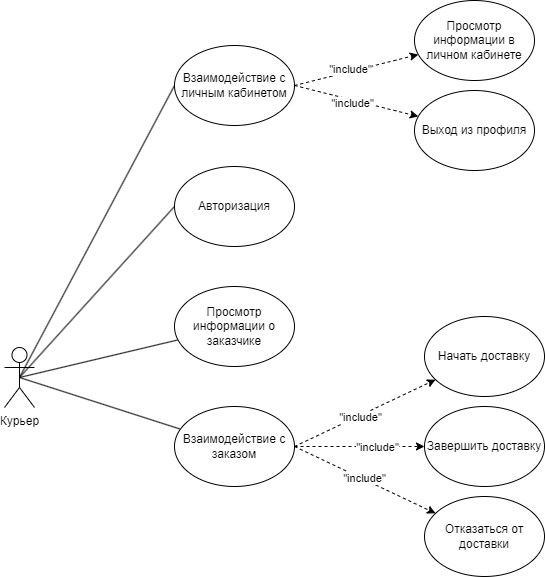
* управлять информацией о товарах: добавление, удаление и редактирование;
* взаимодействовать с личным кабинетом: просматривать и изменять информацию, выходить из профиля;
* управлять информацией о пользователях: удаление, просмотр данных;
* управлять информацией о курьерах: просмотр, добавление, удаление и редактирование.



1. — Диаграмма прецедентов для администратора

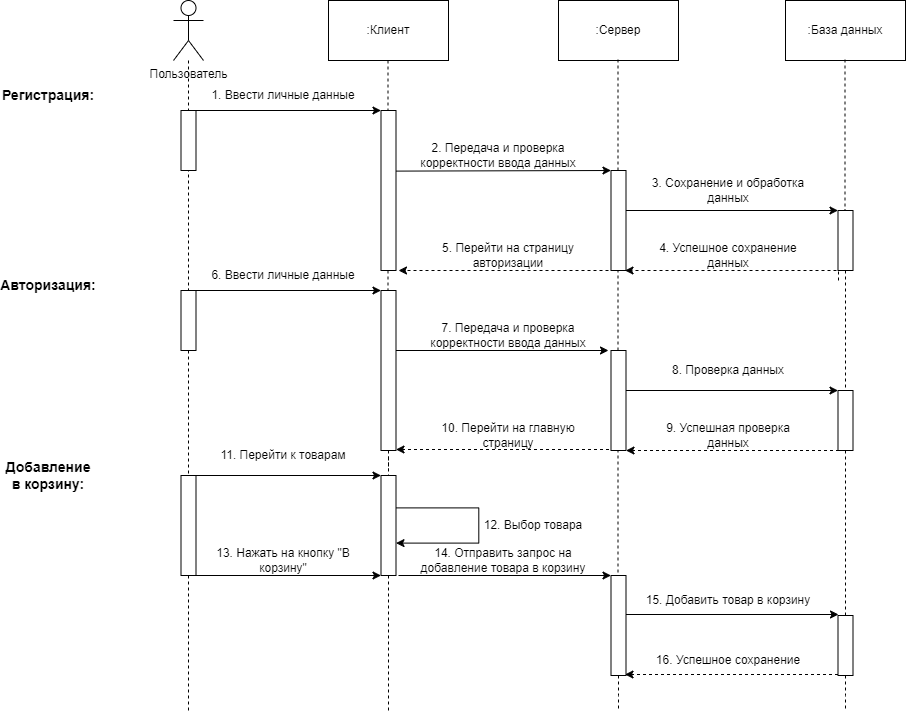
Также существуют следующие функции, доступные курьеру:

* 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.



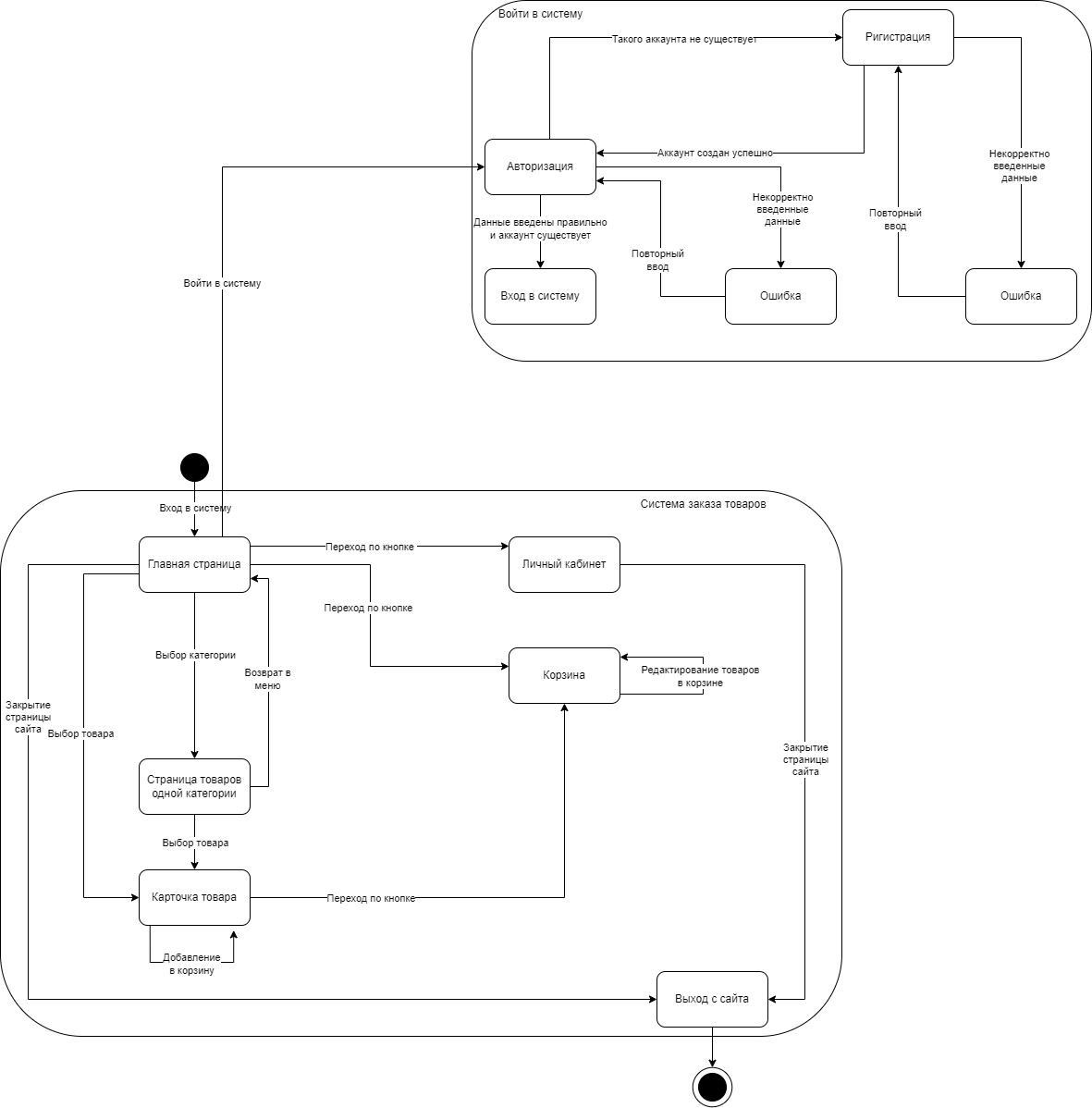
1. — Диаграмма прецедентов для курьера
   * 1. Диаграмма последовательности

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 [4]. Участником данной системы является пользователь, а объектами – клиент, сервер и база данных.



1. — Диаграмма последовательности
   * 1. Диаграмма состояний

Диаграмма состояний (Рисунок 10) отражает внутренние состояния объекта в течение его жизненного цикла от момента создания до разрушения [4]. На данной диаграмме рассмотрены состояния от момента входа в систему до полного выхода из нее.

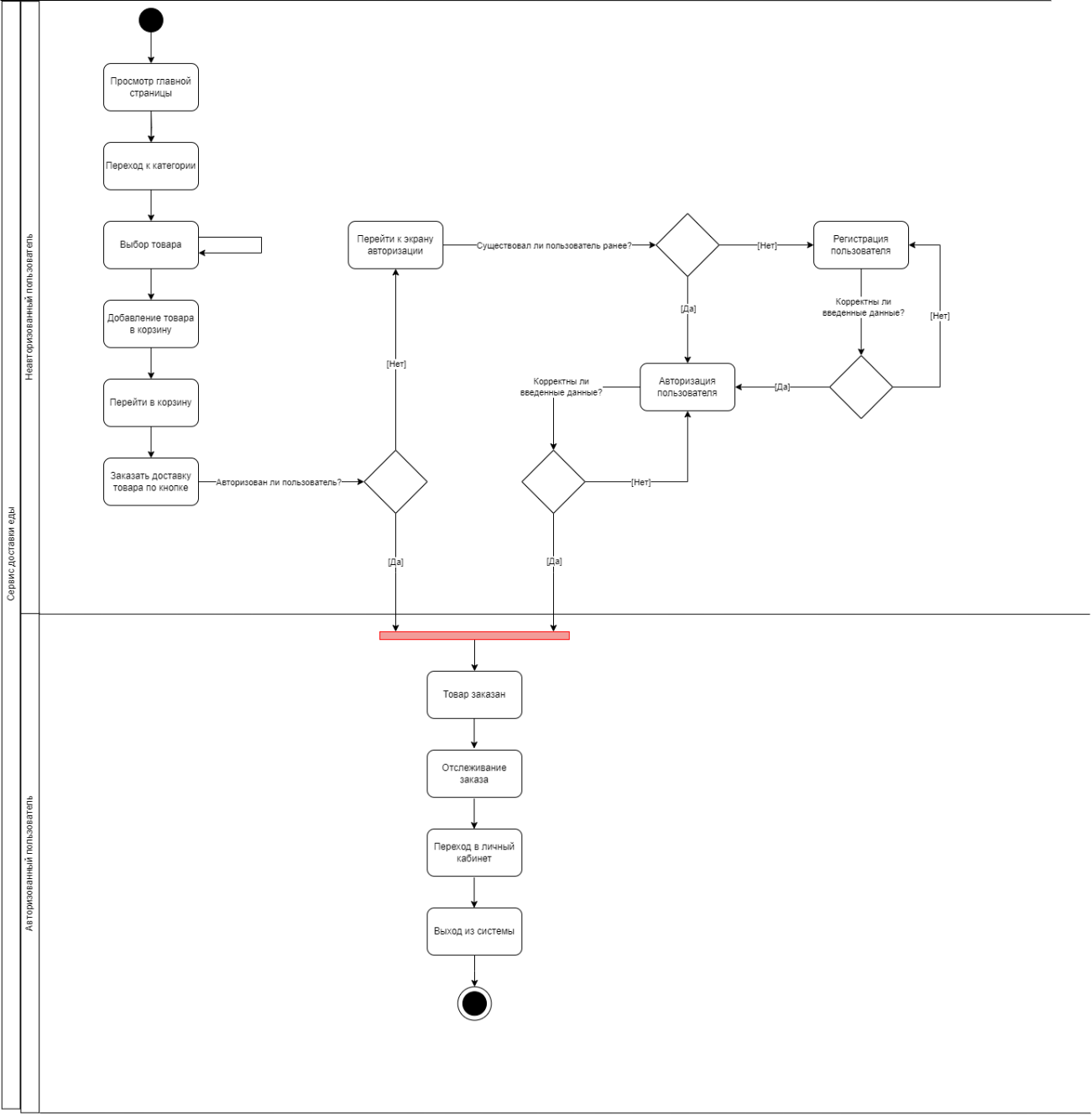


1. — Диаграмма состояний
   * 1. Диаграмма деятельности

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 [4].

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.

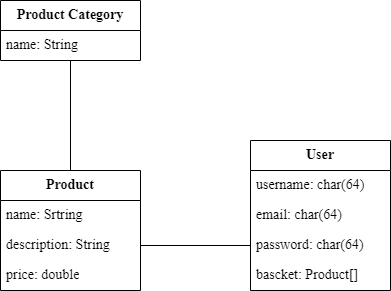


1. — Диаграмма деятельности (активности)
   * 1. Диаграмма классов

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

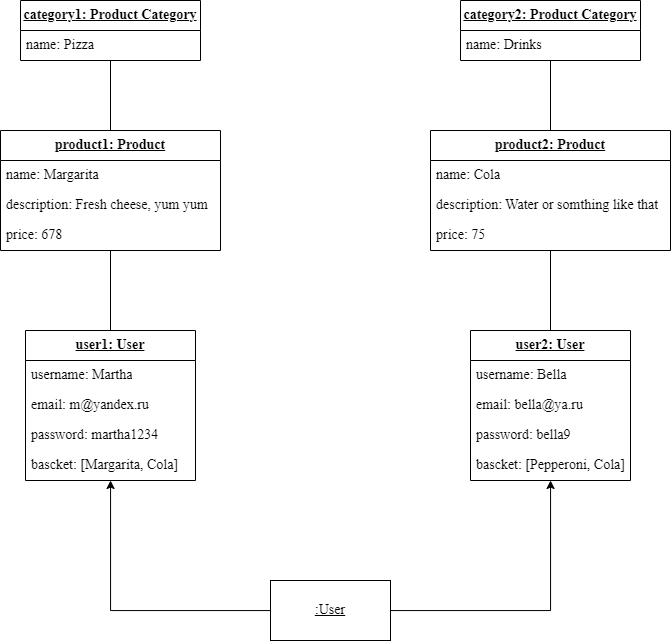
* класс «Пользователь»;
* класс «Продукт»;
* класс «Категории продуктов».

У каждого из классов существуют свои атрибуты.



1. — Диаграмма классов
   * 1. Диаграмма объектов

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

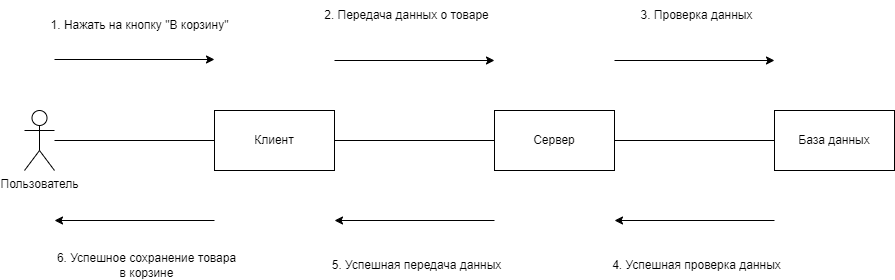


1. — Диаграмма объектов
   * 1. Диаграмма сотрудничества

Диаграмма сотрудничества (Рисунки 14-16) — это вид диаграммы взаимодействия, в котором основное внимание сосредоточено на структуре взаимосвязей объектов, принимающих и отправляющих сообщения [5].



1. — Диаграмма сотрудничества при авторизации

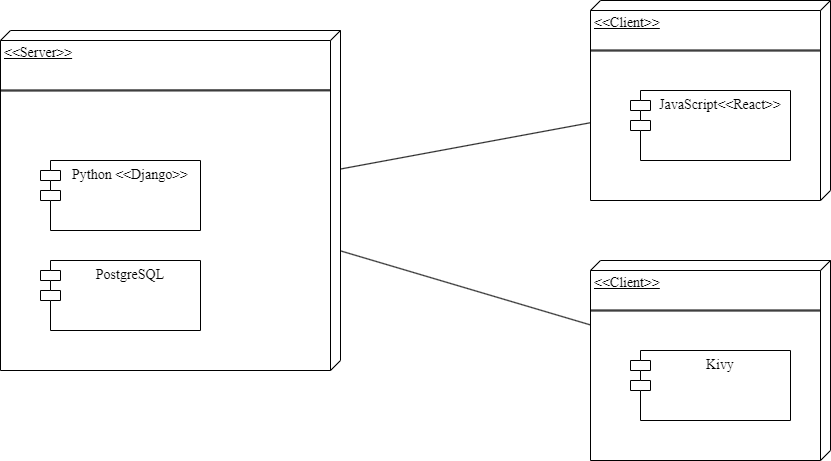


1. — Диаграмма сотрудничества при регистрации



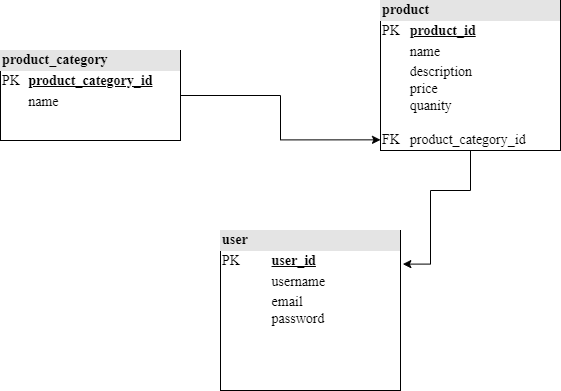
1. — Диаграмма сотрудничества при добавлении в корзину
   * 1. Диаграмма развертывания

Диаграмма развертывания (Рисунок 17) предназначена для представления общей конфигурации или топологии распределенной программной системы [4].



1. — Диаграмма развертывания
   * 1. 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).



1. — ER-диаграмма
2. Реализация
   1. Средства реализации
      1. Средства реализации серверной части сайта

Для разработки серверной (backend) части сайта был выбран следующий стэк технологий:

* Django — это высокоуровневый веб-фреймворк на языке Python;
* PostgreSQL это объектно-реляционная система управления базами данных (СУБД), которая предоставляет мощные средства для хранения, организации и манипулирования данными;
* DB Browser— это пользовательский интерфейс (GUI) для работы с базами данных;
* Swagger — инструмент для документирования и тестирования API.
  + 1. Средства реализации клиентской части сайта

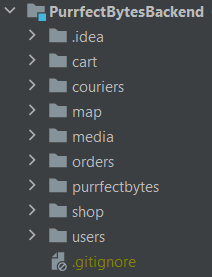
Для разработки клиентской (frontend) части сайта был выбран следующий стэк технологий:

* JavaScript — это один из наиболее популярных языков программирования, а также он является интерпретируемым языком скриптов;
* React — это open-source JavaScript библиотека для построения пользовательских интерфейсов;
* CSS (Cascading Style Sheets) — это язык стилей, используемый для задания внешнего вида веб-страниц;
* HTML (HyperText Markup Language) — это язык разметки, используемый для создания веб-страниц.
  + 1. Средства реализации мобильной части сервиса

Для разработки мобильной (mobile) части сервиса был выбран следующий стэк технологий:

* Kivy — это open-source фреймворк с открытым исходным кодом, написанный на Python, для разработки мультимедийных приложений.
  1. Реализация серверной (backend) части сайта

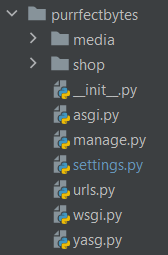
Серверная (backend) часть приложения была написана на языке Python с использованием фреймворка Django. Структура проекта представляет собой корневую папку purrfectbytes и дополнительные — cart, orders, shop и users (Рисунок 19).



1. — Структура проекта

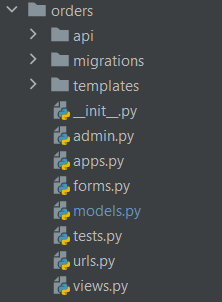
В корневой папке (Рисунок 20) особое внимание стоит обратить на файл settings.py, который содержит настройки проекта. Он содержит множество параметров для настройки работоспособности приложения.

* DEBUG - параметр, отвечающий за включение/выключение режима отладки;
* SECRET\_KEY - секретный ключ приложения, который применяется для генерации токенов, а также для шифрования паролей пользователей;
* DATABASES - настройки подключения к базе данных, в которой хранятся данные приложения;
* INSTALLED\_APPS - список приложений, установленных в проекте. Здесь указываются все приложения, которые будут использованы в проекте;
* MIDDLEWARE - список всех промежуточных компонентов, используемых в Django для обработки запросов и ответов между сервером и клиентом;
* TEMPLATES - настройки для генерации HTML-шаблонов, используемых для визуализации данных пользователя в БД;
* AUTH\_PASSWORD\_VALIDATORS - список компонентов, используемых для проверки безопасности паролей пользователей.



1. — Структура корневой папки проекта

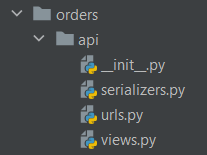
При этом фреймворк Django подразумевает разделение приложения на несколько основных компонентов (Рисунок 21).



1. — Основные компоненты Django на примере orders

* инициализация (\_\_init\_\_.py) — это специальный файл в Python, определяющий пакет. В Django он используется для определения пакета для приложения;
* административная панель (admin.py) — это административная панель Django, которая обеспечивает автоматическую генерацию форм и предоставляет удобный интерфейс для работы с данными;
* конфигурационный файл (apps.py) — это Python файл, который определяет основные настройки для приложения. Он содержит метаданные, относящиеся к данному приложению, и позволяет настроить приложение в соответствии с требованиями и лучше контролировать его работу;
* формы (forms.py) — это файл, который содержит классы форм, которые используются для обработки и валидации данных, отправленных пользователем через HTML-формы. Формы в Django позволяют вам определить, какие данные должны быть собраны с пользователя, и как эти данные должны быть проверены и обработаны. Классы форм могут содержать поля для ввода данных, а также методы для валидации и обработки этих данных;
* модели данных (models.py) — это классы Python, которые определяют структуру базы данных и могут быть использованы для создания или обновления схемы базы данных. Модели могут содержать поля для хранения данных (текстовые, числовые, даты, файлы и др.), а также методы для работы с этими данными;
* сериализация (serializers.py) — файл, обеспечивающий сериализацию и десериализацию данных, передаваемых через приложение. Он используется в Django для работы с данными, передаваемыми через HTTP в API-модулях. Файл содержит классы сериализаторов, которые определяют, какие поля модели должны быть преобразованы в JSON, XML и т.д. Они также могут обеспечивать валидацию данных во время десериализации;
* тесты (test.py) — это файл в приложении, который содержит модульные тесты для этого приложения. Эти тесты используются, чтобы убедиться, что функциональные возможности приложения работают должным образом, и выявить любые ошибки или ошибки до того, как приложение будет развернуто в рабочей среде;
* URL-адресация (urls.py) — это механизм маршрутизации запросов на определенные представления. Django использует файл urls.py для определения соответствующей представлению URL-адреса;
* представления (views.py) — это функции Python, которые обрабатывают запросы от клиента и возвращают HTTP-ответы. Представления могут включать в себя логику приложения, обработку данных из модели, взаимодействие с другими системами.

В верхней папке api (Рисунок 22) лежат компоненты необходимые для реализации drf.



1. — Компоненты для реализации drf

Из компонентов, которые не были описаны выше, здесь находится только компонент сериализации (serializers.py) — файл, обеспечивающий сериализацию и десериализацию данных, передаваемых через приложение. Он используется в Django для работы с данными, передаваемыми через HTTP в API-модулях. Файл содержит классы сериализаторов, которые определяют, какие поля модели должны быть преобразованы в JSON, XML и т.д. Они также могут обеспечивать валидацию данных во время десериализации.

Серверная часть приложения Django также может использовать множество дополнительных библиотек и компонентов.

База данных была развернута в облаке Yandex Cloud. Для настройки подключения к базе данных из приложения были указаны хост, порт, имя базы данных, имя пользователя и пароль.  
Сервер был развернут на виртуальной машине, которая также находится в облаке Yandex Cloud. с помощью системы контейнеризации Docker. Этот процесс был выполнен с использования системы контроля версий Git. После загрузки потребовалось настроить переменные окружения, представляющие собой конфиденциальные настройки, такие как данные аутентификации для базы данных. Клиентская часть была загружена в другом Docker-контейнере.

Для описания спецификации API использовался Swagger [6]. Для того чтобы интегрировать его в проект, нужно было внести изменения в код, включая добавление необходимой зависимости (drf\_yasg), добавить ее в INSTALLED\_APPS и описать спецификации всех методов.

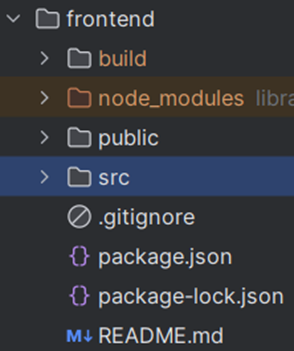
* 1. Реализация клиентской (frontend) части сайта

Клиентская часть приложения (frontend) разработана на языке JavaScript с использованием фреймворка React.js.

Одной из особенностей разработки приложения на React является декомпозиция приложения на независимые компоненты. Каждый компонент отвечает за определенный функциональный блок приложения и представляет собой функцию, которая возвращает HTML-код. Кроме того, компоненты могут содержать переменные и другие функции, что позволяет дополнительно упростить код и повысить его читабельность. Приложение загружается на страницу по мере необходимости, что уменьшает время загрузки страницы и повышает производительность приложения в целом.

Основной HTML файл называется index.html, именно на него загружаются компоненты, которые будут выведены в браузере. В нём вызывается index.js, в котором находится точка входа React-приложения, и вызывается основной компонент App.js, который будет меняться в зависимости от действий пользователя и которому прилагается App.css – файл стилей. Все остальные компоненты вызываются по мере необходимости.

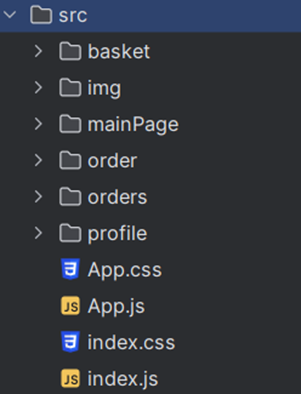
Структура проекта на React включает в себя определенные основные элементы (Рисунок 23).



1. — Основные элементы React-проекта

* основная папка проекта src/, содержащая исходный код;
* папка public/, содержащая файлы, которые будут доступны публично;
* файл package.json, содержащий информацию о проекте, а также список зависимостей для установки;
* файл package-lock.json с информацией о текущей версии зависимостей.

В свою очередь в папке src/ (Рисунок 24) хранятся следующие разделы приложения.



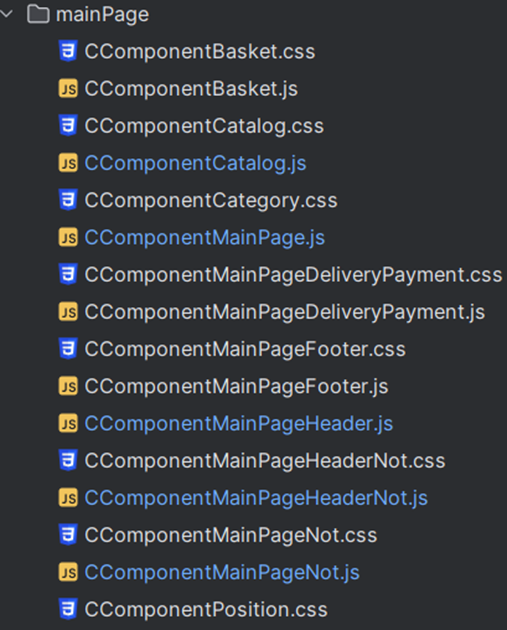
1. — Структура папки src/

* папка basket/ содержит компоненты страницы с корзиной заказа для авторизованного и неавторизованного пользователей (header и основная часть с информацией о выбранных позициях), к файлам типа .js прилагается .css стили (Рисунок 25);



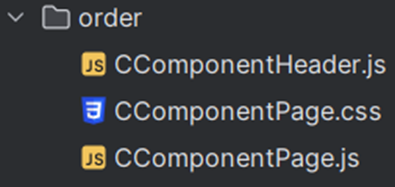
1. — Структура папки basket/ проекта

* папка img/ содержит изображения необходимые для дизайна;
* папка mainPage/ содержит компоненты header, footer и основная части страницы (каталог товаров). Они реализованы в двух вариантах – для авторизованного пользователя и для неавторизованного пользователя. Эта папка содержит общий footer для всех страниц. К файлам типа .js прилагается .css стили (Рисунок 26);



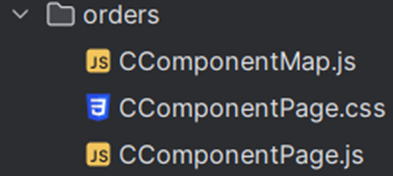
1. — Структура папки mainPage/ проекта

* папка order/ содержит компоненты страницы с оформлением заказа (header и основная часть с информацией о закзазе), к файлам типа .js прилагается .css стили (Рисунок 27);



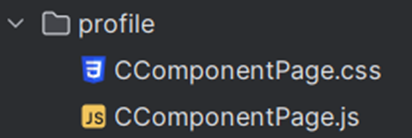
1. — Структура папки order/ проекта

* папка orders/ содержит компоненты страницы с заказами пользователя заказа (header, footer и основная часть с информацией о заказах), к файлам типа .js прилагается .css стили (Рисунок 28);



1. — Структура папки orders/ проекта

* папка profile/ order содержит компоненты страницы с информацией о пользователе (header, footer и основная часть с информацией о пользователе и его адресах), к файлам типа .js прилагается .css стили (Рисунок 29).



1. — Структура папки profile/ проекта
   1. Реализация мобильного (mobile) приложения

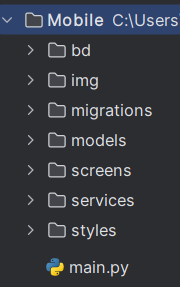
Мобильное приложение (mobile) разработано на языке Python с использованием фреймворка Kivy.

В Kivy каждый экран или виджет представлен отдельным компонентом, который может быть создан в виде класса Python и определен в файле KV для визуального описания. Эти компоненты могут включать в себя различные элементы пользовательского интерфейса, такие как кнопки, текстовые поля, изображения и многое другое.

Основной точкой входа в приложение Kivy является файл Python, который содержит код для создания и управления приложением. Этот файл может загружать различные компоненты и устанавливать их в качестве основных экранов или виджетов.

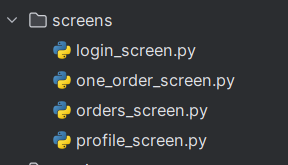
Таким образом, подход к декомпозиции приложения на независимые компоненты в Kivy основан на создании классов Python для логики и виджетов, определенных в файлах KV для визуального представления. Это позволяет создавать модульные и легко поддерживаемые приложения.

Структура проекта на Kivu включает в себя определенные основные элементы (Рисунок 30).



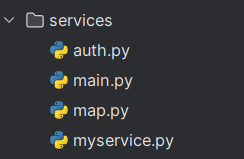
1. — Структура проекта

* папка bd/, содержащая файл database.py, необходимый для подключения к базе данных и создания сессий для взаимодействия с ней;
* папка img/, содержащая изображения, которые используются в приложении;
* папка migrations/, содержащая скрипты миграций для управления изменениями в структуре базы данных ;
* папка models/, содержащая файлы, которые определяют модели данных для приложения. Эти модели определяют структуру данных, которые будут храниться в базе данных;
* папка screens/, содержащая файлы, описывающие экраны пользовательского интерфейса приложения. Каждый экран представляет собой отдельный компонент приложения и отвечает за определенный функциональный блок, такой как авторизация, список заказов, конкретный заказ и профиль пользователя (Рисунок 31);



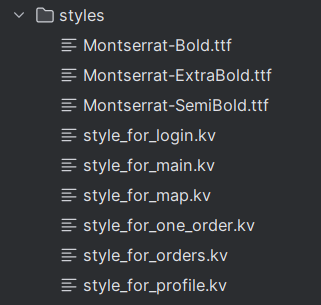
1. — Структура папки screens/ проекта

* папка services/, содержащая файлы, отвечающие за реализацию вспомогательных функций и сервисов, необходимых для работы приложения. Эти модули предоставляют функционал для аутентификации пользователей, работы с картами и фоновой работы приложения (Рисунок 32);



1. — Структура папки services/ проекта

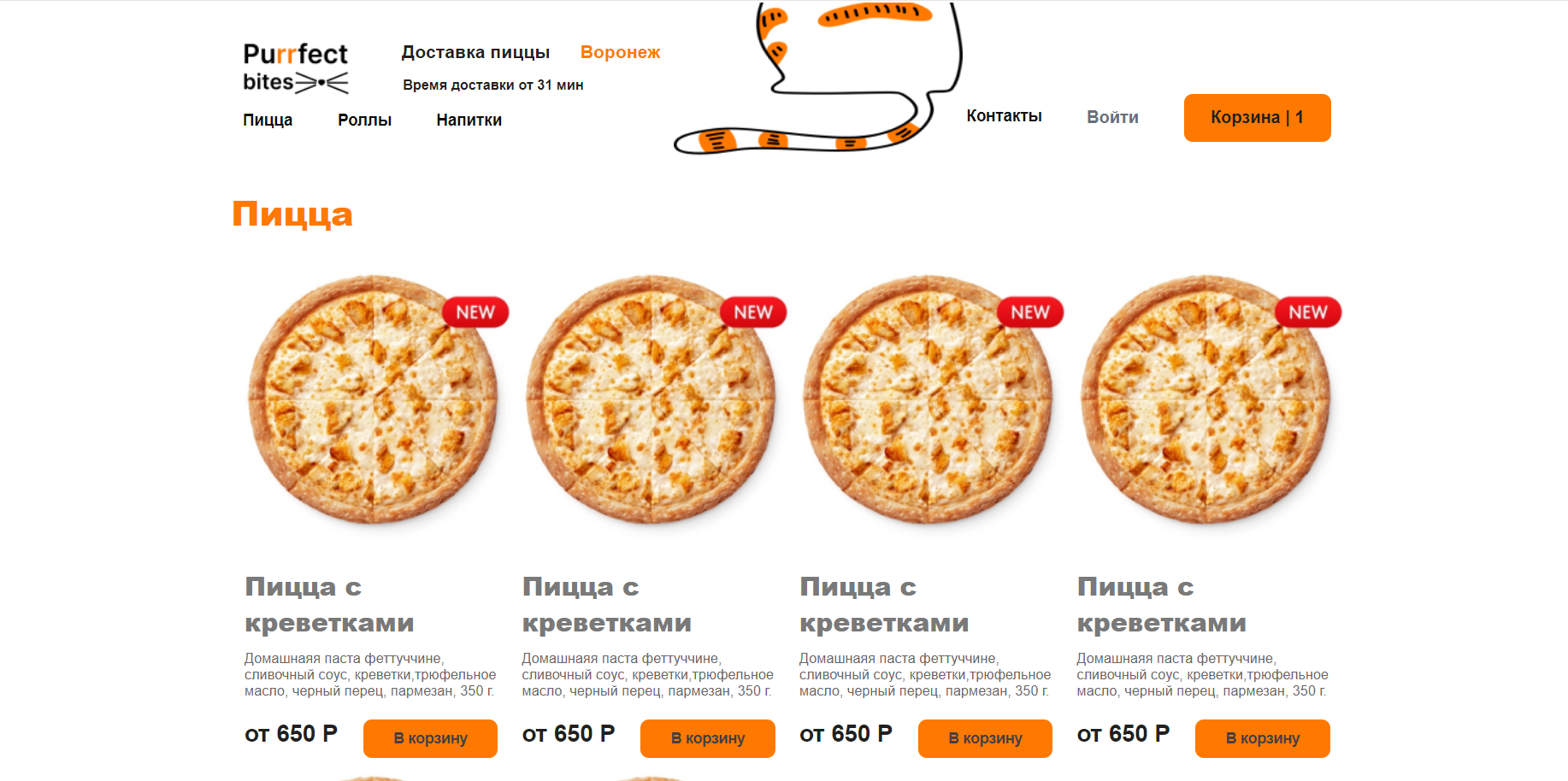
* папка styles/, содержащая файлы, которые определяют внешний вид элементов интерфейса приложения. В этой папке хранятся файлы KV, описывающие стили для различных экранов и виджетов, а также шрифты, используемые для оформления пользовательского интерфейса (Рисунок 33);



1. — Структура папки styles/ проекта

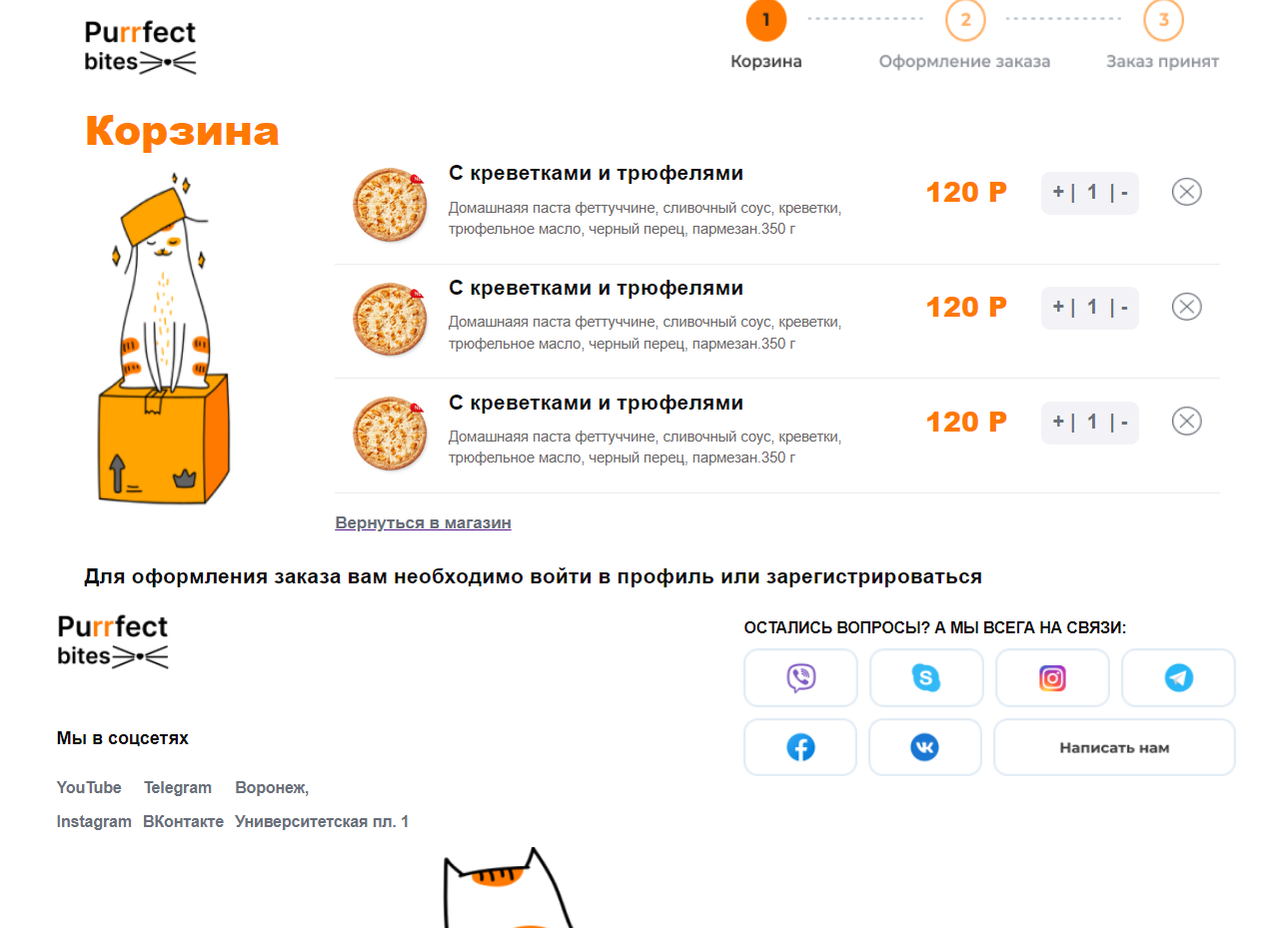
* файл main.py, содержащий точку входа в приложение
  1. Навигация по сайту
     1. Для неавторизованного пользователя

Первое, что видит пользователь при открытии сайта – главная страница. (Рисунок 34).



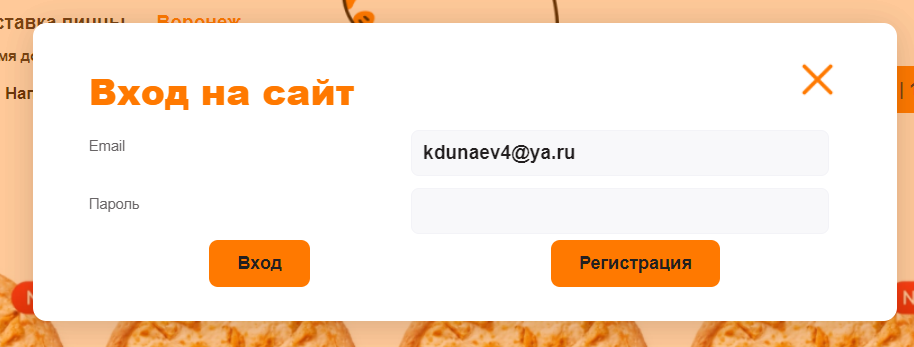
1. — Главная страница неавторизованного пользователя

Пользователь может перейти в корзину. (Рисунок 35)



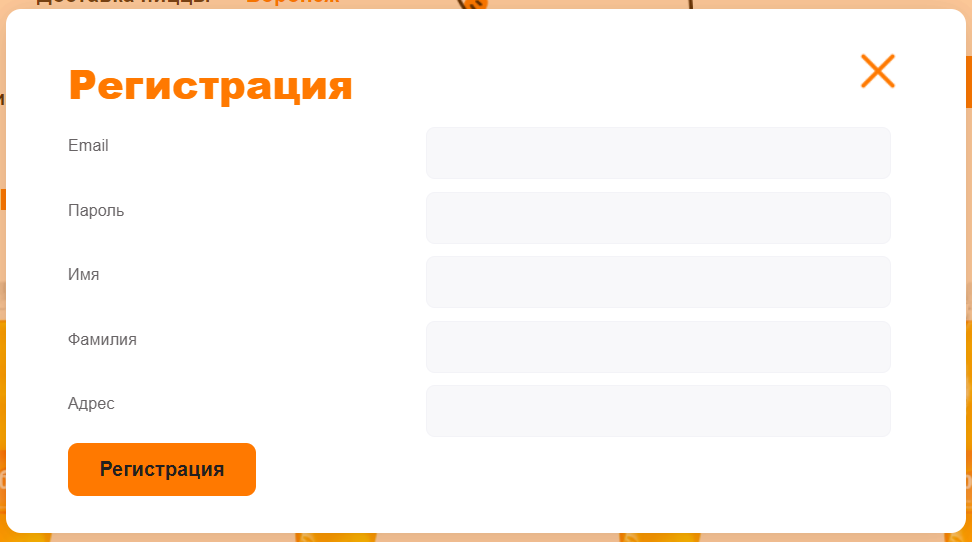
1. — Корзина неавторизованного пользователя

На главной странице при нажатии на кнопку «Войти», появляются формы авторизации. Зарегистрированный пользователь может ввести email и пароль, а затем нажать «Вход». Незарегистрированный пользователь нажимает на кнопку «Регистрация». (Рисунок 36)



1. — Экран авторизации

При нажатии на кнопку «Регистрация» пользователь перенаправляется на регистрацию. У него есть возможность ввести данные о себе и зарегистрироваться. (Рисунок 37)

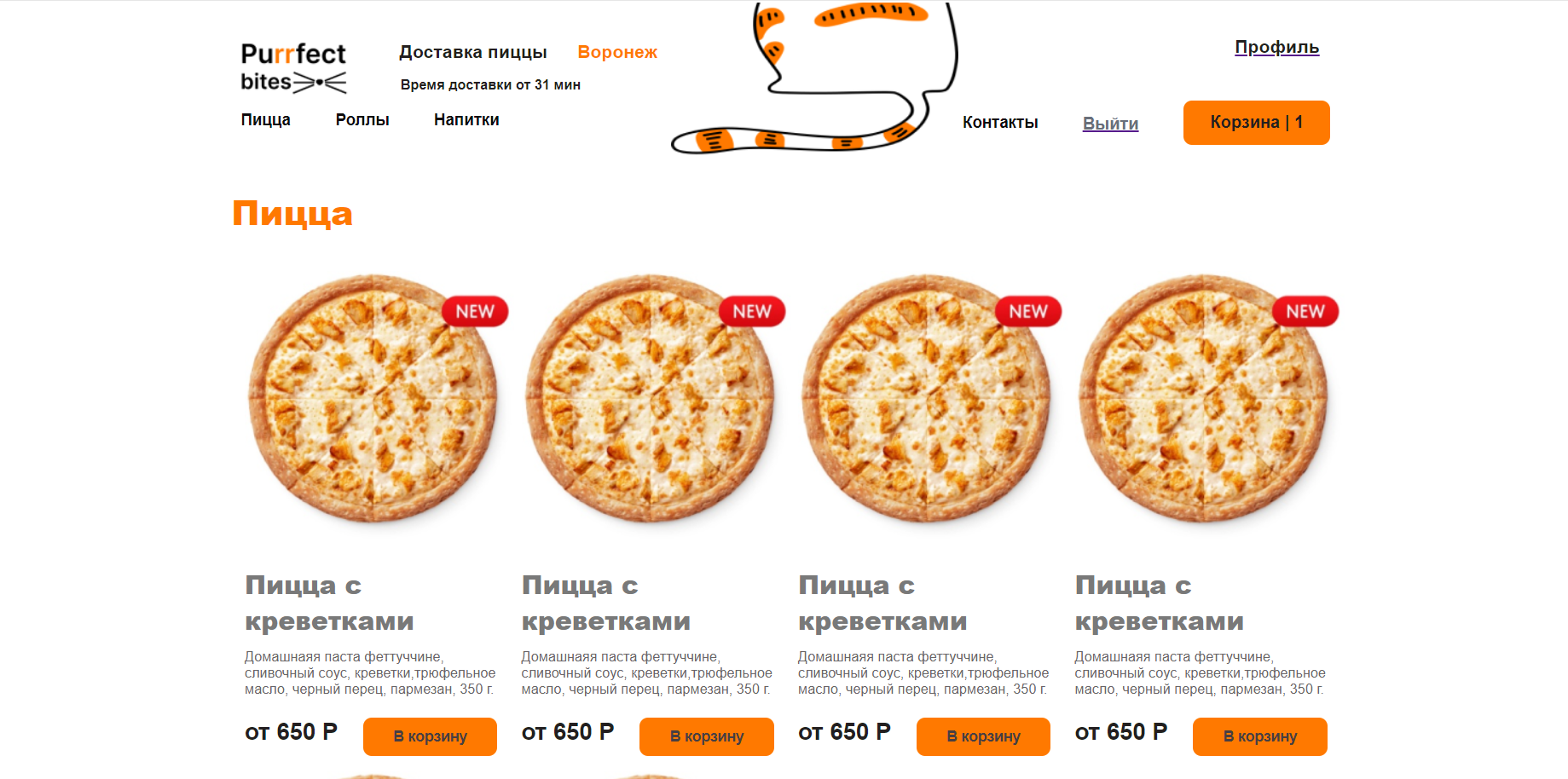


1. — Регистрация

При нажатии на кнопку регистрация пользователь перенаправляется на главную страницу зарегистрированного пользователя.

* + 1. Для авторизованного пользователя

После авторизации пользователь перенаправляется на главную страницу авторизированного пользователя. (Рисунок 38)



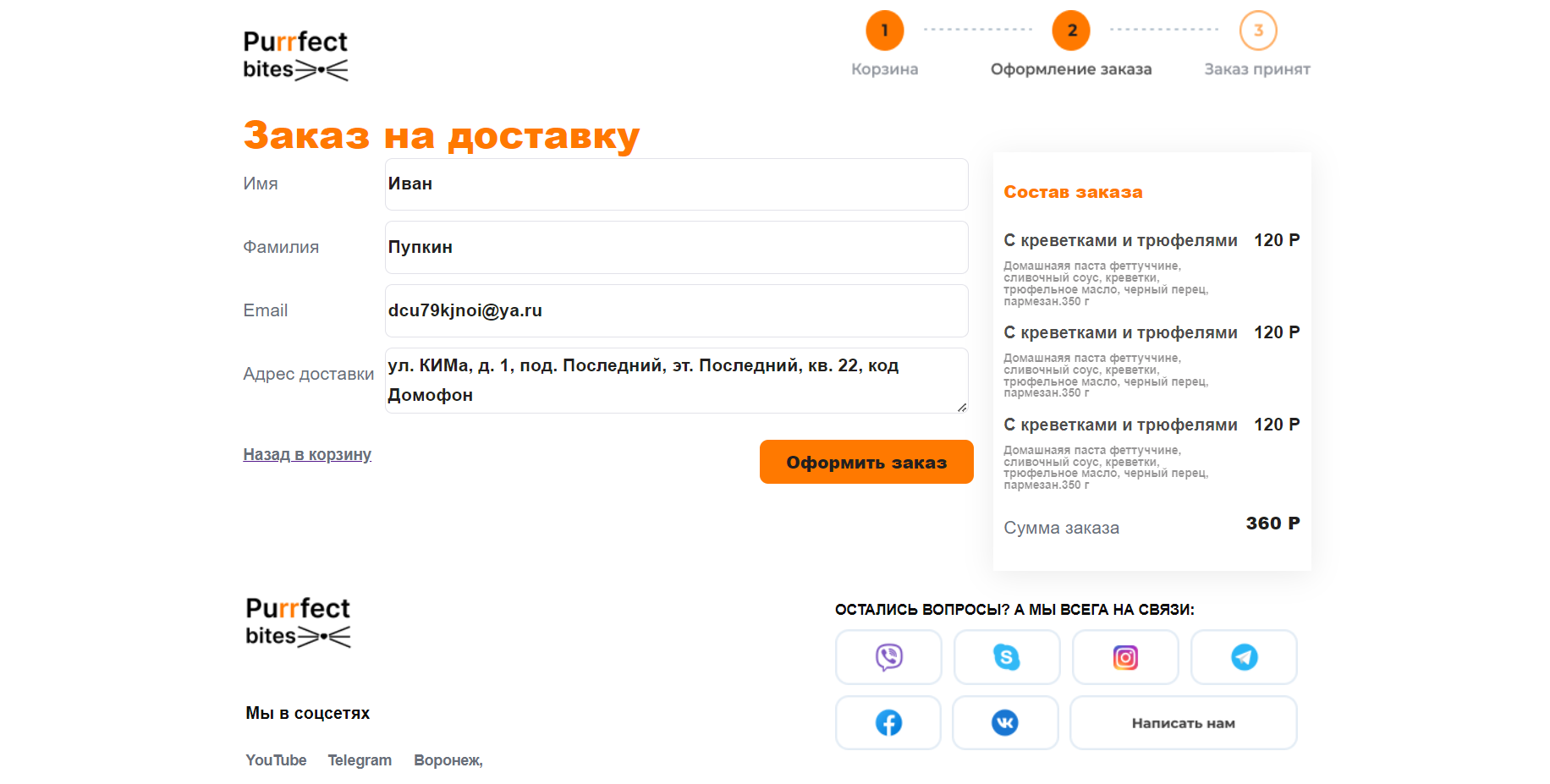
1. — Главная страница авторизованного пользователя

Пользователь может перейти в корзину. (Рисунок 39)



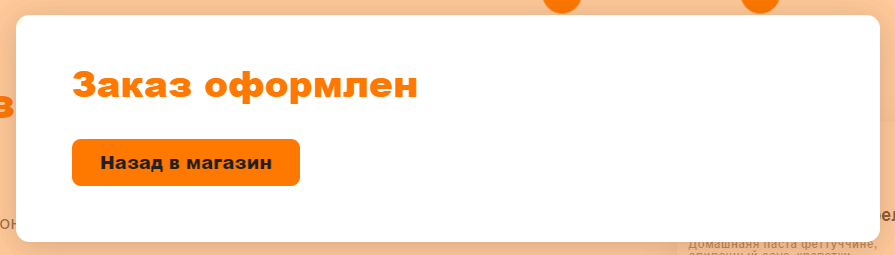
1. — Корзина авторизованного пользователя

В корзине пользователь может, нажав кнопку «Оформить заказ», перейти к оформлению заказа. На странице оформления заказа пользователь может, нажав кнопку «Оформить заказ», чтобы оформить заказ оформлению заказа. (Рисунок 40)



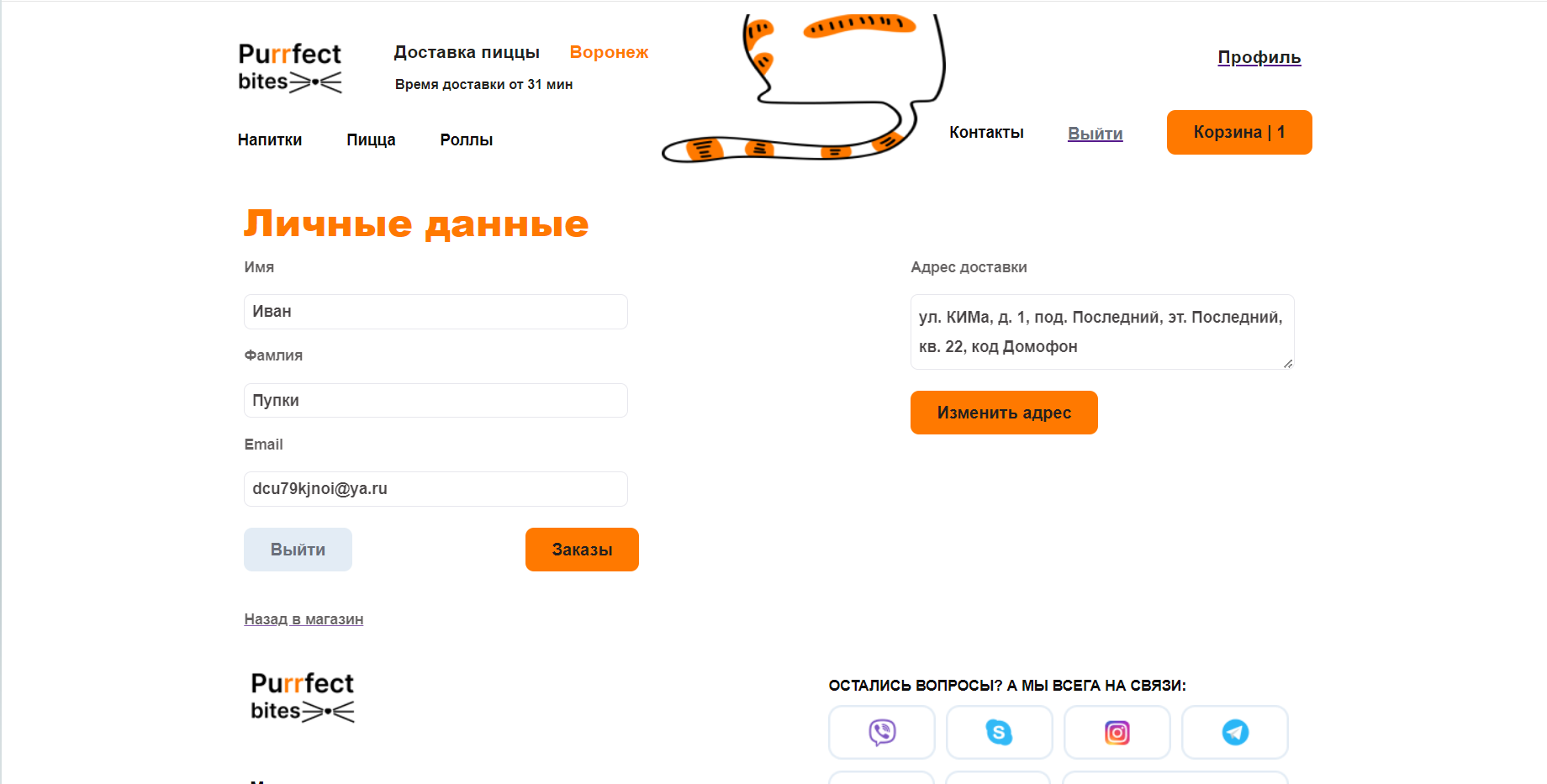
1. — Экран оформления заказа

При нажатии на кнопку «Оформить заказ» пользователь оформляет заказ. (Рисунок 41)

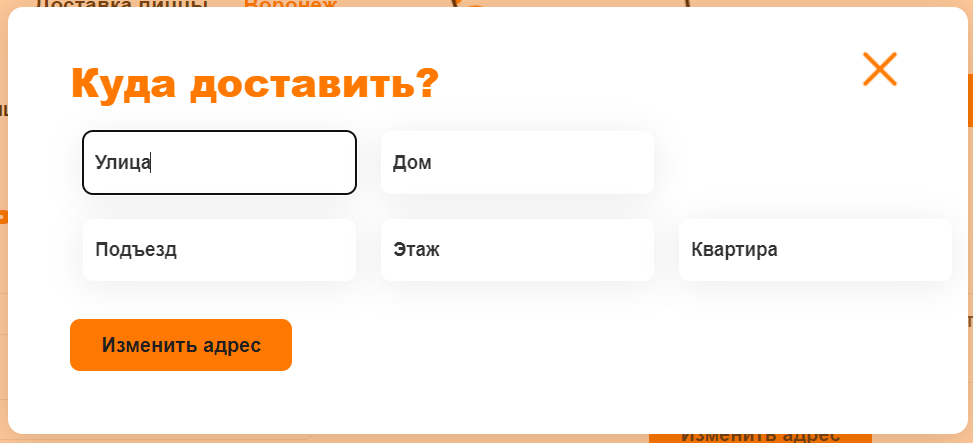


1. — Заказ оформлен

Пользователь может перейти в свой профиль. (Рисунок 42) У него есть возможность изменить адрес доставки. (Рисунок 43) Есть возможность перейти к просмотру заказов и выйти из профиля.

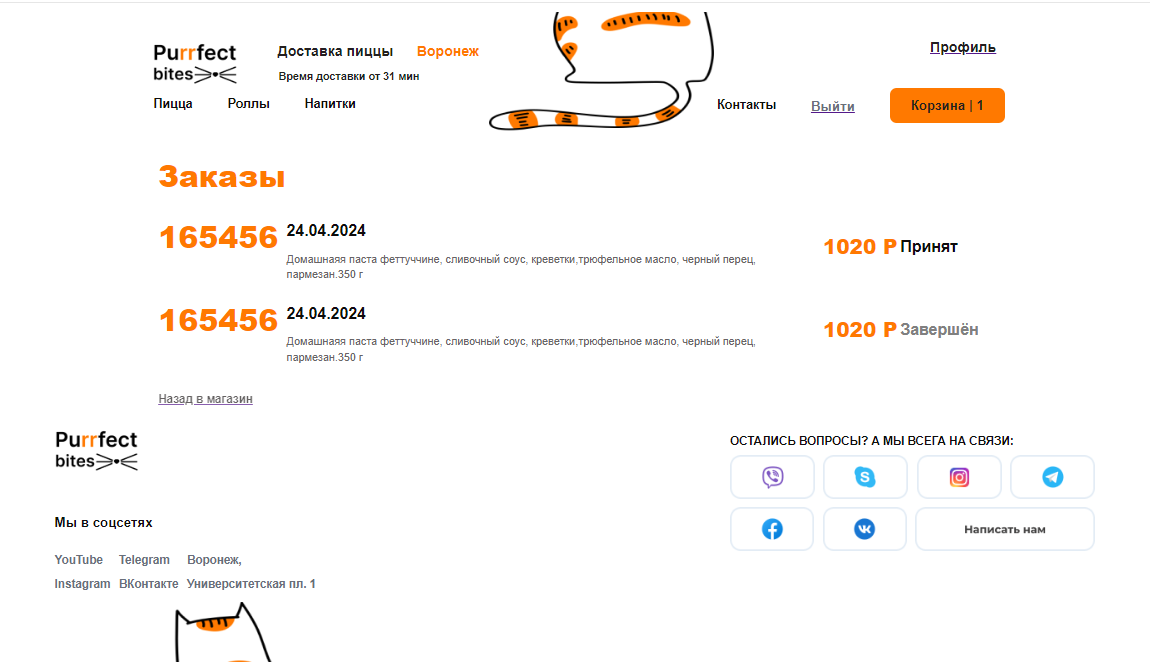


1. — Профиль пользователя



1. — Изменение адреса

. При нажатии на кнопку «Заказы», пользователя перенаправляет на страницу заказов. (Рисунок 44) При нажатии на номер заказа появляется карта с местоположением курьера. (Рисунок 45)

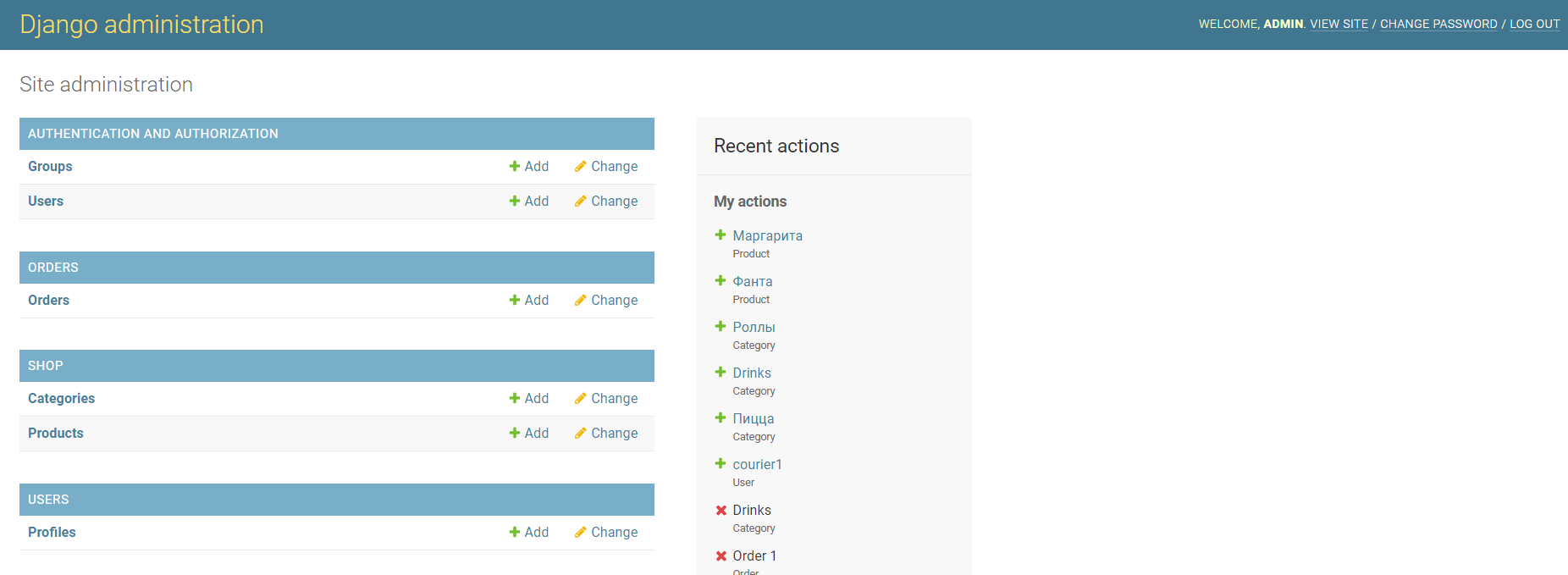


1. — Страница заказов

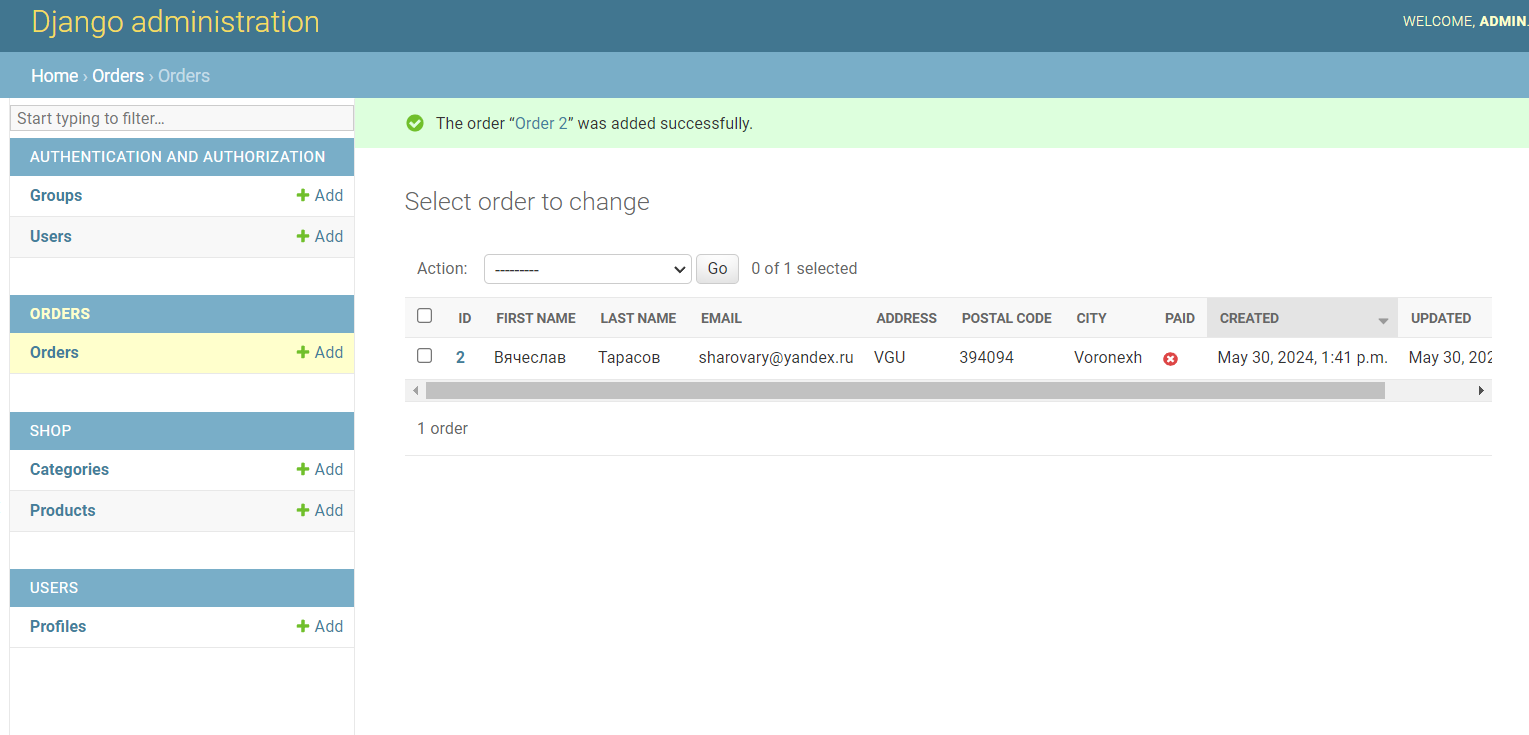


1. —Карта с местоположением курьера
   * 1. Для администратора

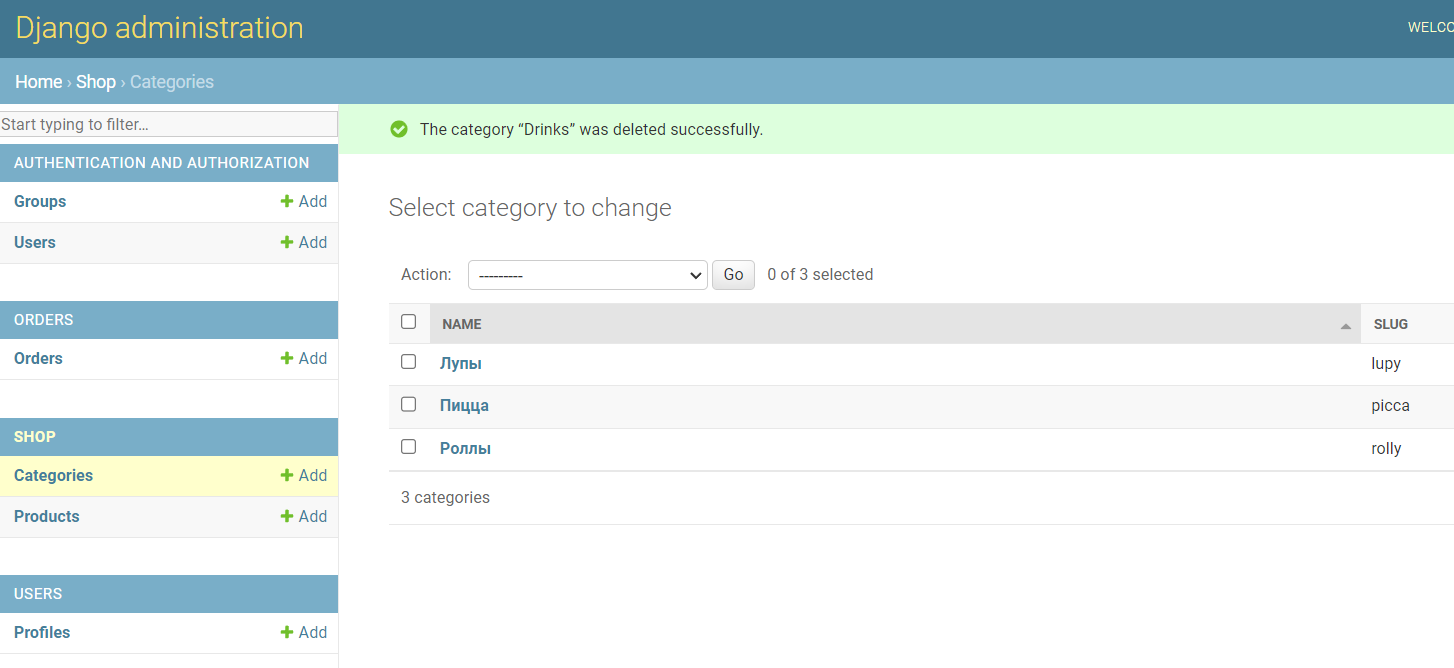
Для администратора у нас есть доступ к панели к панели администрирования (Рисунок 46), где при помощи встроенных инструментов Django возможно управление заказами (Рисунок 47), категориями (Рисунок 48), продуктами (Рисунок 49) и пользователями (Рисунок 50).



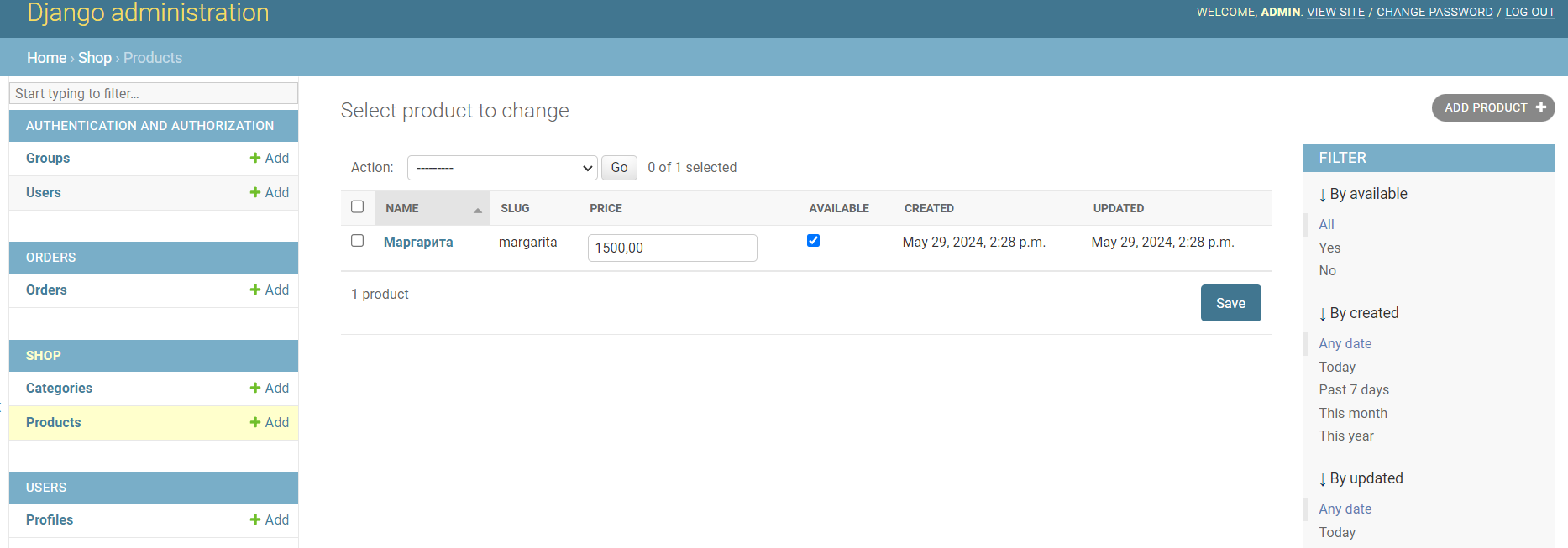
1. — Главная страница с панелью администратора



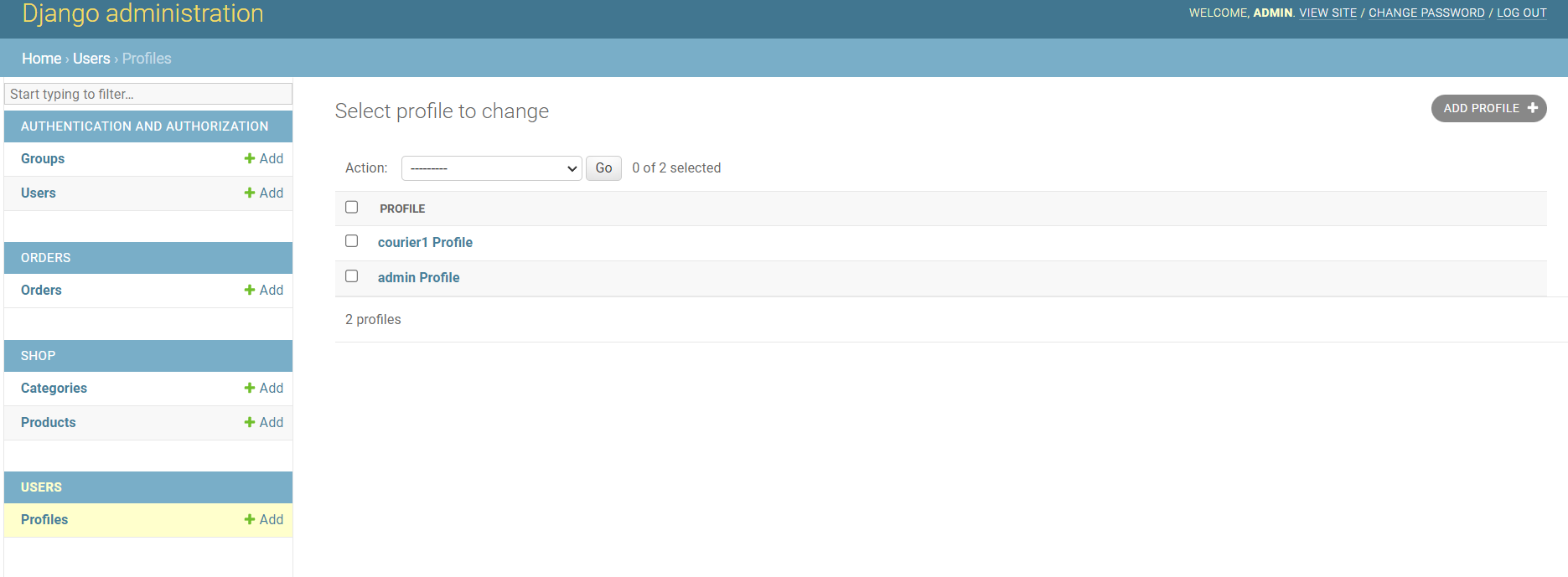
1. — Управление заказами



1. — Управление категориями

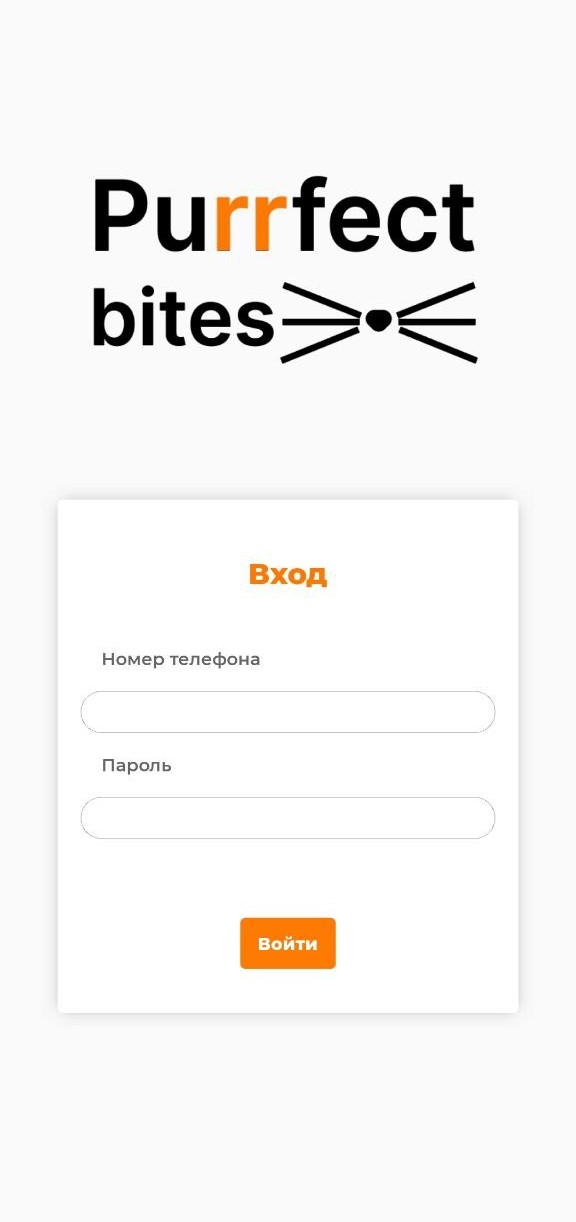


1. — Управление продуктами



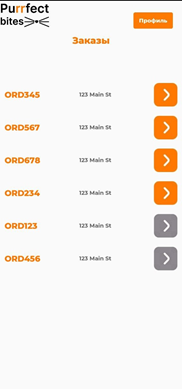
1. — Управление пользователями
   1. Навигация по мобильному приложению

Первое, что видит курьер при запуске мобильного приложения– экран авторизации. (Рисунок 51).



1. — Главный экран

После успешной авторизации, курьер перенаправляется на экран, отображающий список всех его заказов. (Рисунок 52)



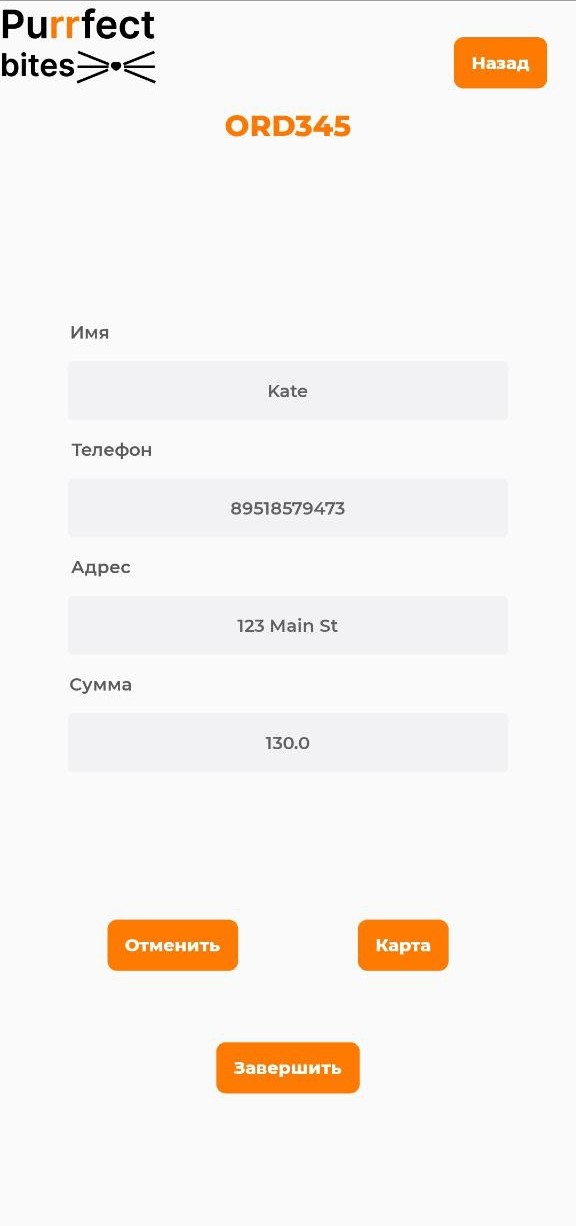
1. — Список заказов

Для просмотра персональных данных и выхода из своей учётной записи, курьер может перейти на экран «Профиль», доступный при нажатии на соответствующую кнопку. (Рисунок 53)



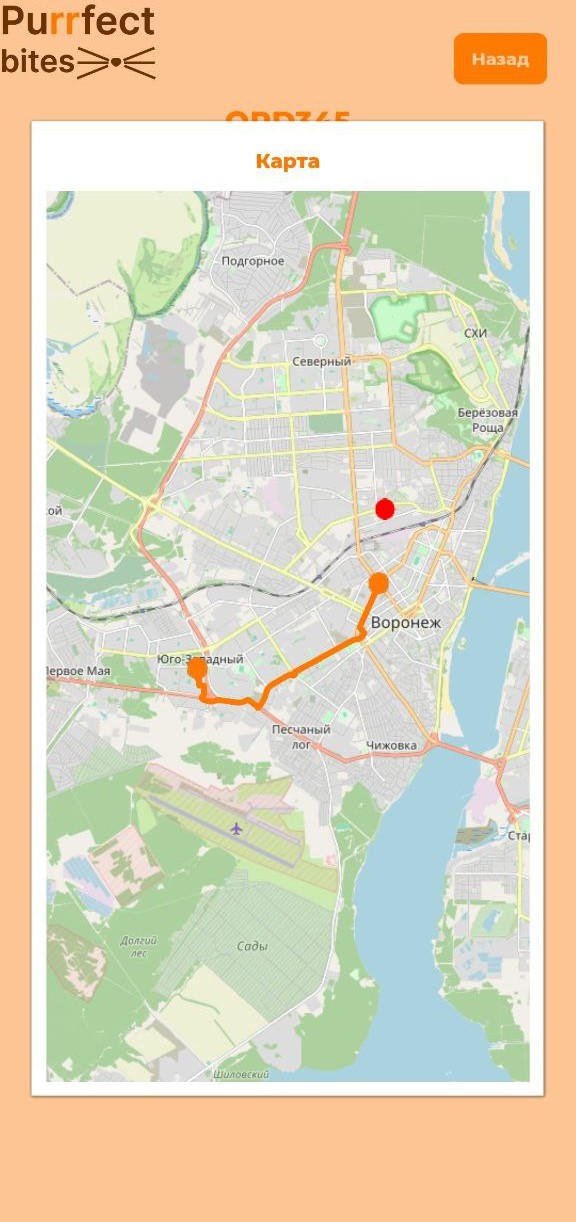
1. — Профиль курьера

Кроме того, при выборе конкретного заказа, курьеру предоставляется возможность ознакомиться с подробной информацией о заказе и изменить его статус. (Рисунок 54)



1. — Конкретный заказ

У курьера есть возможность открыть карту, чтобы увидеть своё текущее местоположение, стартовую точку маршрута, конечную точку маршрута и непосредственно маршрут. (Рисунок 55)



1. — Отображение карты
2. Тестирование
   1. Дымовое тестирование

Дымовое тестирование — это тип тестирования, при котором производится кратковременная проверка основной функциональности системы, чтобы убедиться в отсутствии серьезных проблем, ошибок или сбоев.

Такой тип тестирования полезен, чтобы обнаружить крупные проблемы в функциональности системы до того, как будут проведены более подробные и насыщенные тесты. Если в процессе дымового тестирования найдены ошибка или проблемы, то это может стать причиной для проведения дополнительных и более детальных тестов, чтобы устранить недочеты и проблемы в работе программы.

В ходе дымового тестирования выполняются базовые операции или сценарии, которые предполагаются как наиболее важные и часто используемые пользователем. Далее представлены результаты дымового тестирования для основных сценариев неавторизованного, авторизованного пользователей, администратора и курьера (Таблица 1-4).

Таблица 1 - Результаты дымового тестирования для неавторизованного пользователя

|  |  |
| --- | --- |
| Тестовый сценарий | Результат теста |
| Регистрация | Пройден |
| Авторизация | Пройден |
| Просмотр каталога товаров | Пройден |
| Просмотр позиции | Пройден |
| Взаимодействие с корзиной | Пройден |

Таблица 2 - Результаты дымового тестирования для авторизованного пользователя

|  |  |
| --- | --- |
| Тестовый сценарий | Результат теста |
| Просмотр каталога товаров | Пройден |
| Просмотр позиции | Пройден |
| Выход из профиля | Пройден |
| Взаимодействие с корзиной | Пройден |
| Оформление заказа | Пройден |
| Просмотр профиля | Пройден |
| Просмотр заказов | Пройден |
| Добавление адреса | Пройден |
| Просмотр карты с курьером | Пройден |

Таблица 3 - Результаты дымового тестирования для администратора

|  |  |
| --- | --- |
| Тестовый сценарий | Результат теста |
| Авторизация | Пройден |
| Взаимодействие с профилем | Пройден |
| Просмотр категорий каталога товаров | Пройден |
| Добавление категории каталога | Пройден |
| Изменение категории каталога | Пройден |
| Удаление категории каталога | Пройден |
| Просмотр списка позиций | Пройден |
| Добавление позиции | Пройден |
| Изменение данных о позиции | Пройден |
| Удаление позиции | Пройден |
| Просмотр списка пользователей  и их данных | Пройден |
| Добавление пользователя | Пройден |
| Удаление пользователя | Пройден |

Таблица 4 - Результаты дымового тестирования для курьера

|  |  |
| --- | --- |
| Тестовый сценарий | Результат теста |
| Авторизация | Пройден |
| Просмотр персональных данных | Пройден |
| Просмотр списка заказов | Пройден |
| Просмотр конкретного заказа | Пройден |
| Принятие всех заказов | Пройден |
| Отмена одного из заказов | Пройден |
| Завершение одного из заказов | Пройден |
| Просмотр карты одного из заказов | Пройден |

* 1. Тестирование пользовательского интерфейса

Тестирование пользовательского интерфейса (GUI-тестирование) — это процесс тестирования элементов управления в приложении, который помогает убедиться, что интерфейс соответствует ожидаемой функциональности, включая проверку различных функций и элементов, таких как окна, диалоговые окна, кнопки, переключатели, выпадающие списки, формы, меню и т.д.

Задача проведения GUI-тестов — убедиться, что в функциях пользовательского интерфейса отсутствуют дефекты. В Таблице 4 представлена часть результатов тестирования пользовательского интерфейса. В ней отражены некоторые тестовые сценарии для авторизованного пользователя.

Таблица 4 - Результаты GUI-тестирования

|  |  |  |
| --- | --- | --- |
| **Тестовый сценарий** | **Ожидаемый результат** | **Статус теста** |
| Нажатие на кнопку  «Выйти» | Выход из профиля. Переход на главную страницу для неавторизованного пользователя. | Пройден |
| Нажатие на кнопку «Корзина» | Переход на страницу корзины | Пройден |
| Нажатие на кнопку «Оформить заказ» в корзине | Переход на страницу оформления заказа | Пройден |
| Нажатие на кнопку «Оформить заказ» в оформлении заказа | Оформление заказа. Появляется окно с уведомлением. | Пройден |
| Нажатие на кнопку «Профиль»  в общей шапке нескольких страниц | Переход на страницу личного профиля пользователя | Пройден |
| Нажатие на кнопку  «Выйти» в профиле | Выход из профиля. Переход на главную страницу для неавторизованного пользователя. | Пройден |
| Нажатие на кнопку «Заказы» в профиле | Переход на страницу с заказами | Пройден |
| Нажатие на номер заказа в странице заказов | Появление карты с курьером. | Пройден |

Заключение

В ходе выполнения курсовой работы были выполнены все поставленные задачи. Был разработан сервис доставки еды с отслеживанием в реальном времени

В начале разработки был проведен анализ предметной области, определены основные требования к разрабатываемой системе, определены основные сценарии веб-приложения. По результатам разработки проводился ряд тестов с целью проверки работоспособности системы.

Благодаря проекту были решены следующие задачи у пользователя:

* просматривать категории и товары по выбранным категориям;
* взаимодействовать с корзиной;
* оформлять заказы и просматривать их историю;
* добавлять и сохранять адреса доставки;
* отслеживать свои заказы в реальном времени.

Следующие для администратора:

* добавлять, удалять, просматривать и редактировать товары, категории, заказы;
* редактировать права пользователей;
* обновлять статусы заказов.

И следующие для курьера:

* просматривать заказов;
* обновлять статусы заказов.

Таким образом, итоги разработки, проверенные в ходе тестирования, позволяют достигнуть поставленных заказчиком целей и решают сформулированные в начале разработки задачи.

Список использованной литературы

1. Django - что это за фреймворк на Python: возможности и преимущества [Электронный ресурс]. — Режим доступа: https://blog.skillfactory.ru/glossary/django/.— Заглавие с экрана. — (Дата обращения: 30.05.2024).

2. React – JavaScript-библиотека для создания пользовательских интерфейсов [Электронный ресурс]. — Режим доступа: https://ru.react.js.org/?ref=dtf.ru.— Заглавие с экрана. — (Дата обращения: 30.05.2024).

3. Kivy – Cross-platform Python Framework for GUI apps Development [Электронный ресурс]. — Режим доступа: https://kivy.org/. — Заглавие с экрана. — (Дата обращения: 30.05.2024).

4. Полное руководство по 14 типам диаграмм UML [Электронный ресурс]. — Режим доступа: https://www.cybermedian.com/ru/a-comprehensive-guide-to-14-types-of-uml-diagram/. — Заглавие с экрана. — (Дата обращения: 30.05.2024).

5. Диаграммы сотрудничества [Электронный ресурс]. — Режим доступа: https://helpiks.org/9-53448.html. — Заглавие с экрана. — (Дата обращения: 30.05.2024).

6. Тестирование API с помощью Swagger: особенности и преимущества [Электронный ресурс]. — Режим доступа: https://blog.ithillel.ua/ru/articles/api-testing-with-swagger. — Заглавие с экрана. — (Дата обращения: 30.05.2024).