|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИУК "Информатика и управление"

КАФЕДРА ИУК5 "Системы обработки информации" \_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

Сервис приема заказов на приготовление и доставку пиццы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Студент ИУК5-82Б  \_\_Зейкан М.В.\_\_

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Карышев А.А.**\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2021 г.*

**АННОТАЦИЯ**

В данной выпускной квалификационной работе бакалавра поставлена задача разработать программную систему для работы пиццерии.

Работа состоит из трех частей: научно-исследовательской, проектно-конструкторской и проектно-технологической. Объем работы составляет страницы.

В научно-исследовательской части расписано техническое задание, рассмотрены аналоги системы и приведены аргументы необходимости разработки данной системы. Также в данном разделе приведен анализ, выбрана платформа на которой будет проходить разработка и отмечены требования для приложения.

В рамках проектно-конструкторской части приводится выбор основных

архитектурных решений, принятых в процессе проектирования приложения. Для этого определились большинство требований, предъявляемые к проекту, после чего были построены модели системы, отражающие ее работу с различных точек зрения. Также в данном разделе представлен процесс проектирования базы данных, для хранения данных, необходимых для работы приложения.

В проектно-технологической части приводится описание разработки

клиент-серверной системы в соответствии с принятыми архитектурными

решениями, были выбраны технологические решения поддерживающие жизненный путь программы. Также в данном разделе приведено описание работы с системой, отражающее возможности, которые система предоставляет разным ролям использования приложения.

**Содержание**

[**ВВЕДЕНИЕ** 5](#_Toc70069999)

[**1.Техническое задание** 6](#_Toc70070000)

[1.1. **Общие сведения** 6](#_Toc70070001)

[1.1.1. **Полное наименование программы** 6](#_Toc70070002)

[1.2. **Краткая характеристика области применения** 6](#_Toc70070003)

[1.3. **Заказчик** 6](#_Toc70070004)

[1.4. **Исполнитель** 6](#_Toc70070005)

[1.5. **Плановые сроки** 6](#_Toc70070006)

[1.6. **Порядок оформления и предъявления заказчику результатов работ по созданию системы** 6](#_Toc70070007)

[1.7. **Назначение и цели создания системы** 6](#_Toc70070008)

[1.7.2. **Цели создания системы** 7](#_Toc70070009)

[1.8 **Характеристика объекта автоматизации** 7](#_Toc70070010)

[1.8.1 **Краткие сведения об объекте автоматизации** 7](#_Toc70070011)

[1.8.2. **Сведения об условиях эксплуатации объекта автоматизации** 7](#_Toc70070012)

[1.9. **Требования к системе** 7](#_Toc70070013)

[1.9.1. **Требования к функциональным характеристикам** 7](#_Toc70070014)

[1.9.2 **Требования к надежности и безопасности** 8](#_Toc70070015)

[1.9.3. **Требования к программному обеспечению** 8](#_Toc70070016)

[1.9.4. **Требования к составу и параметрам технических средств** 8](#_Toc70070017)

[1.9.6. **Требования к методическому обеспечению** 8](#_Toc70070018)

[1.9.7. **Требования к дизайну** 8](#_Toc70070019)

[1.10. **Календарный план выполнения работы** 9](#_Toc70070020)

[1.11. **Порядок контроля и приёмки системы** 11](#_Toc70070021)

[1.11.1. **Общие требования к приёмке работы по стадиям** 11](#_Toc70070022)

[1.11.2 **Статус приемочной комиссии** 12](#_Toc70070023)

[1.11.3. **Требования к программной документации** 12](#_Toc70070024)

[1.11.4. **Источники разработки** 12](#_Toc70070025)

[**2.НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ** 14](#_Toc70070026)

[2.1 **Постановка задачи проектирования** 14](#_Toc70070027)

[2.2 **Описание предметной области** 14](#_Toc70070028)

[2.3. **Перечень задач, подлежащих решению в процессе разработки** 16](#_Toc70070029)

[2.4 **Сценарии использования** 17](#_Toc70070030)

[2.5 **Обоснование выбора инструментов и платформы разработки** 19](#_Toc70070031)

[2.6. **Обработка состояний** 21](#_Toc70070032)

[2.7 **Интерфейс взаимодействия пользователя с системой** 22](#_Toc70070033)

[2.8 **Анализ аналогов** 22](#_Toc70070034)

[2.9 **Архитектура системы** 25](#_Toc70070035)

[**3.1 Разработка структуры сайта** 26](#_Toc70070036)

[**3.2. Разработка архитектуры приложения.** 27](#_Toc70070037)

[2.5.3 **Запросы** 31](#_Toc70070038)

[2.6. **Логическая схема данных** 33](#_Toc70070039)

[2.7 **Алгоритмы и формулы** 34](#_Toc70070040)

[2.8. **Макет** 38](#_Toc70070041)

[**4.ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ** 39](#_Toc70070042)

[4.1. **Порядок развертывания системы.** 39](#_Toc70070043)

[4.2. **Интерфейс взаимодействия пользователя с системой.** 39](#_Toc70070044)

[4.3 **Руководство пользователя** 42](#_Toc70070045)

[4**.4 Экономическая часть** 44](#_Toc70070046)

[**4.4.1** **Цели и задачи, решаемые в экономической части** 44](#_Toc70070047)

[**4.4.2 Расчет амортизационных отчислений** 44](#_Toc70070048)

[**4.4.3 Расчет расходов на энергопотребление** 46](#_Toc70070049)

[**4.4.4 Расчет заработной платы программиста** 46](#_Toc70070050)

[**4.4.5 Расчет общих затрат на создание пакета программ** 47](#_Toc70070051)

[**4.4.6 Определение отпускной цены программы** 47](#_Toc70070052)

[**4.4.7 Расчет годовых затрат на эксплуатацию программы** 48](#_Toc70070053)

[**4.4.8 Расчет годовых затрат на выполнение работ ранее употреблявшимся способом** 49](#_Toc70070054)

[**4.4.9 Определение экономического эффекта программы** 50](#_Toc70070055)

[**4.4.9 Определение расчетного коэффициента экономической эффективности программы.** 50](#_Toc70070056)

[**4.4.10 Срок окупаемости разработанной программы** 50](#_Toc70070057)

[4.5 **Дальнейшее развитие** 51](#_Toc70070058)

[**ЗАКЛЮЧЕНИЕ** 52](#_Toc70070059)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 53](#_Toc70070060)

[**ПРИЛОЖЕНИЕ А** 63](#_Toc70070061)

# **ВВЕДЕНИЕ**

Целью данной выпускной квалификационной работы является создание приложения для управления работы пиццерии и оптимизации процесса приема заказов пиццы.

Для достижения поставленной цели были поставлены следующие задачи:

1. Обозначить и проанализировать задачи, которые требуется решить во время процесса разработки;

2. Решить, какие инструменты и платформа будут использоваться.

3. Спроектировать базу данных.

4. Определиться с архитектурой приложения.

5. Разработать клиент-серверное приложение

Актуальность данной темы раскрывается в том, что каждой пиццерии нужна система автоматизации для принятия заказов и реализации отчетности. Благодаря подобной системе ускорится работа всего персонала и клиенты за короткое время получат свою пиццу.

При разработке подобной системы важно понять, что кроме основной бизнес-логики в системе должен быть осуществлен механизм аутентификации пользователей, для ограничения доступа к системе, а также механизм разделения ролей в системе.

# 

# **1.Техническое задание**

## 1.1. **Общие сведения**

## 1.1.1. **Полное наименование программы**

“Сервис по доставке и продаже пиццы”

## 1.2. **Краткая характеристика области применения**

Данное приложение может быть использовано в любой пиццерии, готовой поддерживать сервер на постоянной основе, а также постоянно поддерживаться.

## 1.3. **Заказчик**

КФ МГТУ им. Н.Э. Баумана.

## 1.4. **Исполнитель**

Студент КФ МГТУ им. Н.Э. Баумана Зейкан М.В., гр. ИУК5-82Б

## 1.5. **Плановые сроки**

Плановые сроки по созданию системы:

Начало работ – 10 февраля 2021 года.

Окончание работ – 30 апреля 2021 года.

## 1.6. **Порядок оформления и предъявления заказчику результатов работ по созданию системы**

По мере разработки приложения результаты будут показываться научному руководителю на консультациях по дипломной работе.

## 1.7. **Назначение и цели создания системы**

1.7.1. **Назначение разработки системы**

Система создана для автоматизации процесса приема заказов и продажи пиццы и вычисления времени приготовления и доставки до пункта назначения.

## 1.7.2. **Цели создания системы**

Разработать веб-приложение для пиццерии “TastyPizza”, которое будет реализовано на платформе ASP.NET Core 5 совместно с фреймворком React.JS.

Получение дохода и автоматизация процесса продажи и доставки пиццы.

## 1.8 **Характеристика объекта автоматизации**

## 1.8.1 **Краткие сведения об объекте автоматизации**

Объектом автоматизации является сервис по продаже и доставке пиццы. Основным функциональным назначением приложения служит продажа пиццы, алгоритм подбора рекомендованной пиццы для клиента и алгоритм обработки заказов пользователей для получения времени ожидания готовки и доставки.

## 1.8.2. **Сведения об условиях эксплуатации объекта автоматизации**

Веб-приложение может использоваться любым пользователем интернета, имеющим мобильный телефон, ноутбук или любое устройство имеющее браузер для входа в интернет.

Функционирование системы должно происходить в любых условиях.

## 1.9. **Требования к** **системе**

## 1.9.1. **Требования к функциональным характеристикам**

-Отображение доступной для заказа пиццы;

-Функция выбора размера и типа теста пиццы;

-Ввод данных при оплате и выбор места доставки;

-Функциональная корзина с элементами удаления, добавления и оплаты товара.

-Вывод информации о сделанном заказе.

-Фильтрация пиццы по категориям, рейтингу и по популярности

-Алгоритм вычисления времени приготовления и доставки пиццы.

-Алгоритм подбора рекомендованной пиццы для пользователя по его вкусам.

## 1.9.2 **Требования к надежности и безопасности**

Особых требований не выдвигается.

## 1.9.3. **Требования к программному обеспечению**

Для разработки программного обеспечения потребуются net core 5, React.js, node.js 14.9, npm.

# 

## 1.9.4. **Требования к составу и параметрам технических средств**

В состав технических средств необходимо включить стационарный компьютер, либо выделенную машину на облачном хостинге со следующими характеристиками оборудования:

1. Windows servers, 8, 10, Mac OS, Linux
2. ОЗУ с объемом памяти от 4 ГБ
3. Процессор с частотой не менее 2.2 ГГц

## 1.9.6. **Требования к методическому обеспечению**

Разработку системы требуется сопровождать расчетно-пояснительной запиской и графической частью. Расчетно-пояснительная записка состоит из следующих частей:

1. Техническое задание
2. Исследовательская часть. (Научно-исследовательская часть.)
3. Проектно-конструкторская часть.
4. Проектно-технологическая часть.

## 1.9.7. **Требования к дизайну**

Дизайн сайта должен включать в себя “header” в котором будет находиться лого с названием пиццерии и кнопка корзины. Ниже будут расположены всевозможные категории пицц и фильтр по популярности, алфавиту и цене.

На центральной части будут расположены карточки с пиццами, на которых будут расположены картинка пиццы, кнопки выбора типа теста, размера пиццы и добавления в корзину.

Все страницы должны быть выполнены в одной стилистике.

## 1.10. **Календарный план выполнения работы**

Таблица 1 – календарный план работы

|  |  |  |
| --- | --- | --- |
| Стадии | Этапы работ | Дата |
| 1. Формирование требований к АС | 1.1. Обследование объекта и обоснование необходимости создания АС | 10.02.2021 |
| 1.2. Формирование требований пользователя к АС | 17.02.2021 |
| 1.3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания) | 20.0.2021 |
| 2. Разработка концепции АС | 2.1. Изучение объекта | 22.02.2021 |
| 2.2. Проведение необходимых научно-исследовательских работ | 22.02.2021 |
| 2.3. Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователя | 25.02.2021 |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
|  | 2.4. Оформление отчета о выполненной работе | 25.02.2021 |
| 3. Техническое задание | 3.1. Разработка и утверждение технического задания на создание АС | 29.02.2021 |
| 4. Эскизный проект | 4.1. Разработка предварительных проектных решений по системе и ее частям | 15.03.2021 |
| 4.2. Разработка документации на АС и ее части | 23.03.2021 |
| 5. Технический проект | 5.1. Разработка проектных решений по системе и ее частям | 3.04.2021 |
| 5.2. Разработка документации на АС и ее части | 30.04.2021 |
| 5.3. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку | 4.04.2021 |
| 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации | 17.04.2021 |
| 6. Рабочая документация | 6.1. Разработка рабочей документации на систему и ее части | 6.04.2021 |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
|  | 6.2. Разработка или адаптация программ | 26.03.2021 |
| 7. Ввод в действие | 7.1. Подготовка объекта автоматизации к вводу АС в действие | 1.04.2021 |
| 7.2. Подготовка персонала | - |
| 7.3. Комплектация АС поставляемая изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями) | 8.04.2021 |
| 7.6. Проведение предварительных испытаний | 12.04.2021 |
| 7.7. Проведение опытной эксплуатации | 12.04.2021 |

## 1.11. **Порядок контроля и приёмки системы**

## 1.11.1. **Общие требования к приёмке работы по стадиям**

Место проведения приемки работ КФ МГТУ им. Н.Э. Баумана.

Сроки проведения приёмки работ – консультации в соответствии с расписанием, а также другие даты, согласованные с заказчиком и исполнителем.

В окончательной приемке работ участвует комиссия.

## 1.11.2 **Статус приемочной комиссии**

Прием работы на всех этапах осуществляется непосредственным руководителем курсовой работы. По окончанию разработки работоспособное приложение демонстрируется на комиссии.

## 1.11.3. **Требования к программной документации**

В ходе выполнения выпускной квалификационной работы должны быть разработаны:

1) Расчетно-пояснительная записка объемом 67 страниц, состоящая из 3 частей:

* Научно-исследовательская часть, в т.ч. техническое задание
* Проектно-конструкторская часть
* Проектно-технологическая часть

1. 7 или более листов формата А1 графического материала.
2. Презентация.

## 1.11.4. **Источники разработки**

Настоящее Техническое Задание разработано на основе следующих документов и информационных материалов:

-ГОСТ 34.603-92 Информационная технология (ИТ). Виды испытаний автоматизированных систем.

-ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы

-ГОСТ 34.201-89 Комплекс стандартов на автоматизированные системы

-ГОСТ 34.320-96 Принцип концептуализации

-ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными

-ГОСТ 34.601-90 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы.

- ГОСТ 24.701-86 «Надежность автоматизированных систем управления».

-ГОСТ Р.50377-92 «Безопасность оборудования информационной технологии, включая электрическое конторское оборудование».

# **2.НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ**

## 2.1 **Постановка задачи проектирования**

Исследовать предметную область. Спроектировать серверную часть, в частности описать базу данных и логику приложения. Спроектировать интерфейс основной страницы, корзины, страницы заказа и страницы отчетности.

## 2.2 **Описание предметной области**

Пицца — это одно из тех блюд, которое пользуется популярностью и спросом в скромных закусочных и дорогих ресторанах, но также и у огромного количества желающих заказать пиццу на дом.

Доставка пиццы - услуга, при которой заказчику доставляют пиццу по указанному им адресу.  Заказ пиццы, как правило, предполагает звонок в пиццерию или оформление заказа на сайте.

Пиццерия имеет ограниченный ресурс печек и пицца-мейкеров. В одну печь можно вместить до двух пицц. Веб приложение использует формулы обработки времени доставки, которые указаны в проектно-конструкторской части.

Пример готовки указан на рисунке 1. Алгоритмы формул реализованы также в конструкторской части.

Процесс начинается с заказа клиентом пиццы (заказ на сайте, либо по телефону), приняв заказ, пицца-мейкер начинает процесс приготовления. Если ингредиентов для пиццы не хватает, то пицца будет не доступна для заказа и ее тип будет не активным. Когда пицца будет готова ее передают курьеру, он отвозит пиццу по указанному адресу. Клиент получает свой заказ и начинает (по своему усмотрению) наслаждается аппетитной пиццей.

#### У клиента есть предпочтения как по вкусовым качествам, так и по цене. На начальном этапе могут быть приготовлены пиццы вида: “пепперони”, “сырная цыпленок барбекю”, “кисло-сладкий цыпленок”, “чизбургер-пицца”, “маргарита”, “четыре сезона”, “овощи и грибы”. Также должна быть возможность добавления новинок. Цена должна быть в диапазоне от 245 до 800 рублей, что позволит сервису доставки пиццы найти своего покупателя.

Критерии продавца – если пицца доступна на сайте значит все ресурсы для ее приготовления имеются. Основное условие продажи пиццы, это:

1. Выбрана минимум одна пицца:
2. Произведена оплата.

Если курьер занят, то время доставки увеличивается и корректное время указывается при покупке пиццы.

Время на доставку определяется во время выбора клиентом адреса доставки. Получив маршрут до места доставки, приложение выдает примерное время ожидания заказа.

Сервис будет предоставлять пиццу с видом теста двух типов – тонкое и традиционное.

Приложение позволяет формировать список самых популярных пицц, по алфавиту и по цене, а также сортирует по категориям (мясные, вегетарианские, острые и гриль).

Администратор может добавлять данные о новом блюде или удалять данные о нем, а также редактировать информацию о пицце.

Клиент может просмотреть основную информацию, меню, узнать цену интересующих блюд или просмотреть список пицц по разным критериям. Так же клиент может оформить заказ, включив в него понравившиеся ему блюда.



Рисунок 1 – Пример готовки.

## 2.3. **Перечень задач, подлежащих решению в процессе разработки**

-Разработать интерфейс.

-Разработать алгоритм вычисления времени, которое будет потрачено на дорогу до точки доставки и на приготовление пиццы.

-Разработать алгоритм подбора рекомендации пиццы для клиента.

-Реализовать сервер, который будет отвечать на запросы клиента для передачи данных о пицце, обрабатывать информацию для получения времени доставки и приготовления пиццы, и выводить данные о заказах в страницу отчетности.

-Реализовать функцию покупки требуемого количества пиццы и ввода данных пользователя.

-Спроектировать возможность указания точки доставки на карте.

-Развертывание приложения.

## 2.4 **Сценарии использования**

**Actor** –человек или другая система, который(е) взаимодействуют с разрабатываемой системой. Актером может быть покупатель, контент менеджер, продукт менеджер, но также может быть и другая система, как пример telegram-bot.

Самыми наиболее распространенными актерами для сайта, разработанного нами будут: клиент и менеджер. Есть еще и посетитель, но исключая важные случаи посетитель – это тот же клиент, который не совершает покупок. Т.е. в случае, когда функциональность одного актера вся включает в себя функциональность другого, то нужно оставлять только 1-го актера. [8]

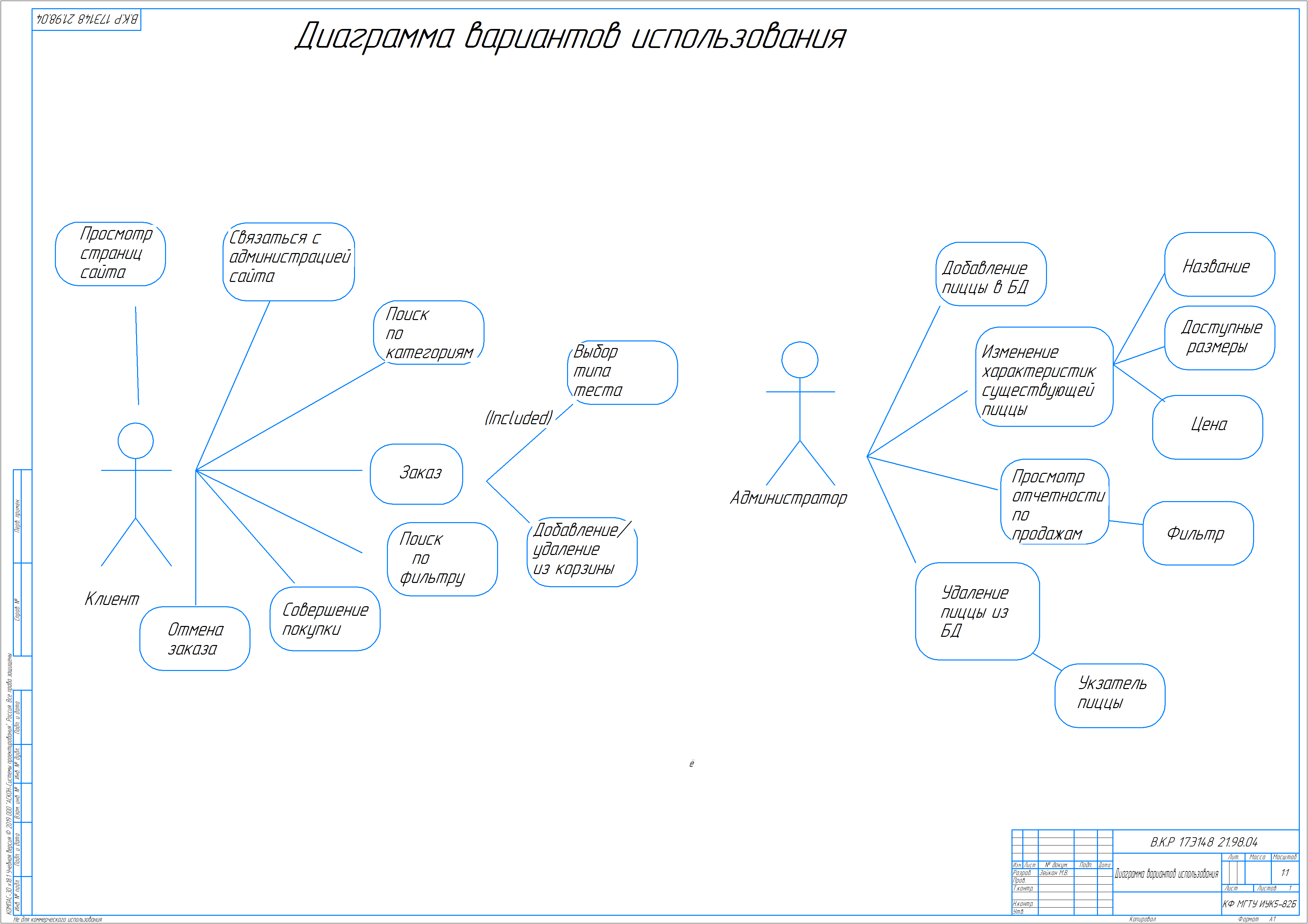


Рисунок 3 –Сценарий использования клиентом.

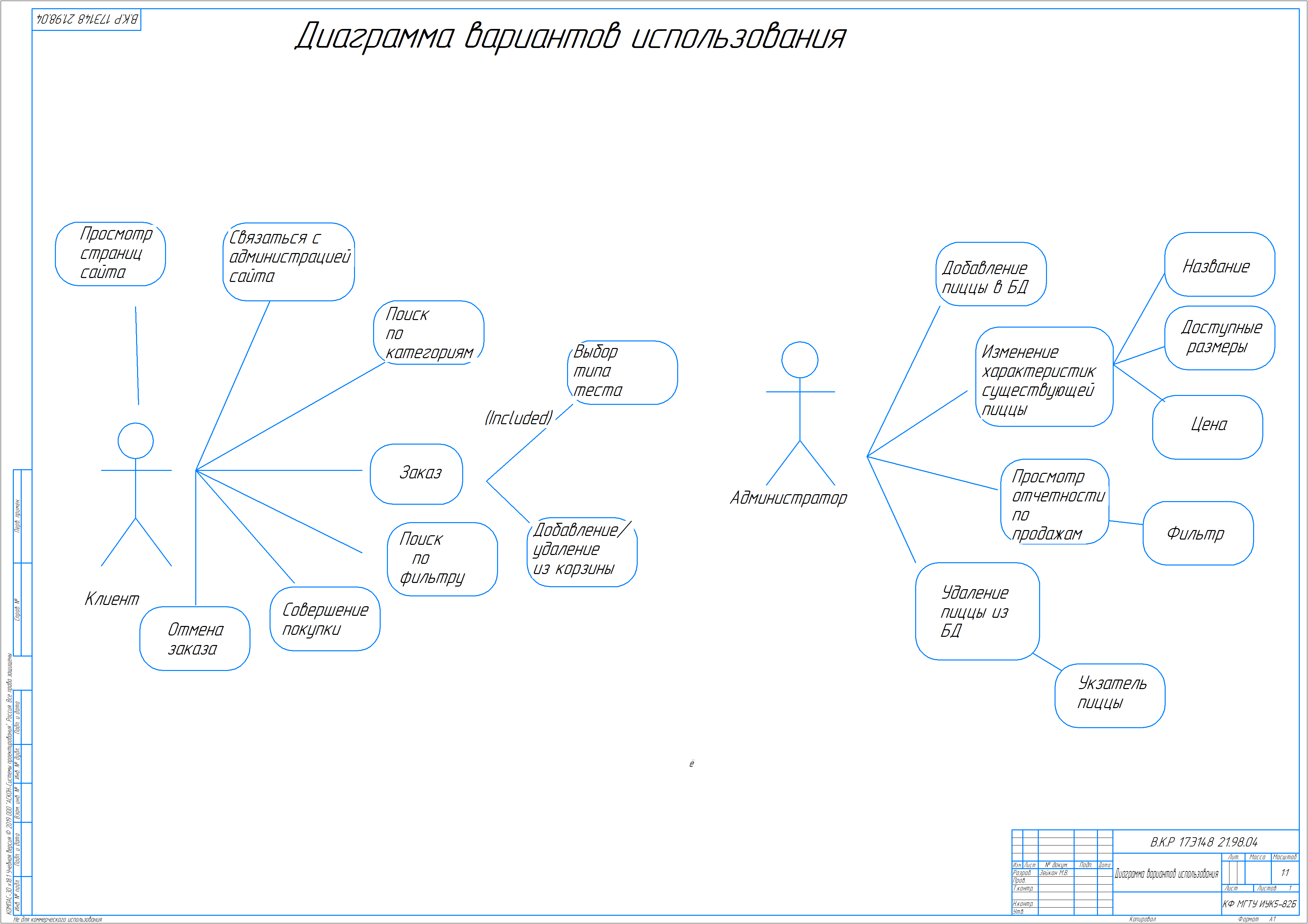


Рисунок 4 –Сценарий использования администратором.

## 2.5 **Обоснование выбора инструментов и платформы разработки**

**Серверная часть(Backend)**

Основной платформой является ASP.NET CORE 5, которая спроектирована для создания различных по масштабам веб-приложений. На ней будет держаться наш сервер с базой данных и обрабатывать данные вычисления времени приготовления и доставки пиццы. Вся информация о клиентах и приготовляемой пицце также будет находиться на сервере в базе данных. Из плюсов net core можно отметить свою мультиплатформенность, благодаря которой можно сэкономить ресурсы и взять сервер на Linux. Последнее обновления фреймворка до версии 5.0 добавило новую версию языка c# 9.0, а также ускорило работу самого фреймворка. Постоянная поддержка со стороны разработчиков Microsoft стабильно улучшает core и устраняет его недостатки.

Языком программирования для сервера выбираем c#, так как это один из основных языков данной платформы, и он очень хорошо подходит для реализации сервера в совокупности с другими технологиями.

Entity Framework Core – одна из самых популярных технологий в среде .Net Core, позволяющая обращаться к данным, менять и манипулировать ими, она очень подходит для данного типа приложений.

SQLSMS(SSMS) - предоставит средства для настройки и обслуживания баз данных. С помощью SSMS можно будет развертывать, отслеживать и обновлять компоненты уровня данных, реализованные данным веб-приложениями.

**Клиентская часть(Frontend)**

Клиентская часть будет написана на React.js. Оптимизированное одностраничное приложение полностью подходит для наших задач. Клиент будет выводить все имеющиеся на сервере пиццы.

React.

Выбираем react для сервера по следующим причинам:

1. Взглянув на код, всегда можно сказать, как ваш компонент будет представлен в готовом решении.

2. Связывание JavaScript и HTML в JSX делает компоненты простыми для понимания.

3. Можно представлять React на сервере без перезагрузки страницы.

Мы могли бы использовать angular, фреймворк от Google, но он больше подходит для огромных Enerprise проектов. Следующий вариант - Vue, у него достаточно легкий порог вхождения, немного легче чем у React, но у Vue будет меньше так называемая помощь комьюнити, по сравнению с популярным React. Для нашего фреймворка есть попросту больше написанных библиотек и больше заданных вопросов и ответов на них в stackoverflow.

Языком программирования фронтенда является javascript т.к. это основной язык для React и с помощью его будут реализованы основные функции передачи данных от клиента на сервер.

## 2.6. **Обработка состояний**

**React hooks**

Приложение фильтрует и сортирует контент состоящий из пицц. Реализуется это при помощи хуков react. Хуки – конструкции, которые позволяют использовать состояние и другие возможности React без написания классов.

Хуки – это есть простые функции, но для правильного их применения есть несколько правил:

-Использовать их только на верхнем уровне.

-Вызывать хуки только из компонентов React.

-Вызывать из пользовательского хука[9].

Благодаря этим правилам, все состояния будут понятны читающим ваш код. Для соблюдения правил есть хороший плагин для ESLint под названием “eslint-plugin-react-hooks”, принуждающий работать по правилам. Он по умолчанию включен в Create React App.

Хуки решают множество проблем, казалось бы, несвязанных между собой.

Задействуется TypeScript - язык который был создан для объявления строгой типизации и отлавливания большей части багов во время стадии разработки. С его помощью указываем какие типы обязаны получать наши компоненты.

## 2.7 **Интерфейс взаимодействия пользователя с системой**

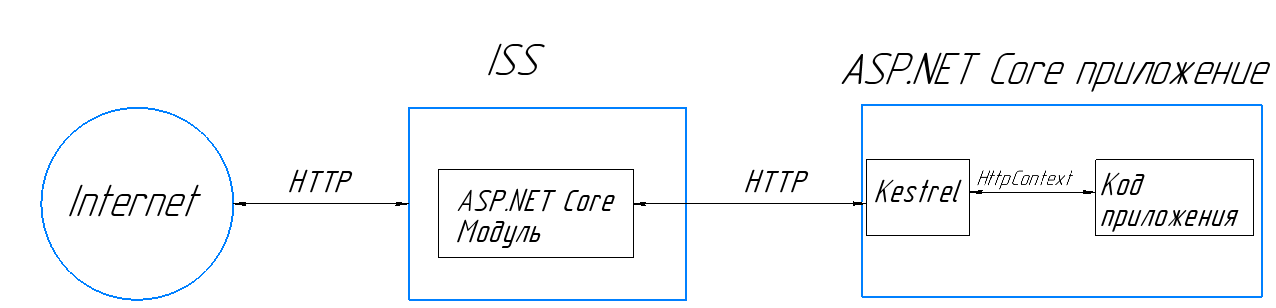


Рисунок 4–Архитектура веб приложения.

Система является Web приложением. Это тип приложений, написанных на архитектуре “клиент-сервер”. Особенность заключается в том, что Web приложение находится и выполняется на удаленном сервере, клиент это получает только результаты вычислительной работы. Работа данного приложения основывается на получение запросов от пользователя, их обработки и выдачи результата. Передача запросов и результатов обработки происходит через интернет.

За счёт наличия исполняемой части, Web-приложения способны выполнять практически те же операции, что и обычные Windows-приложения, с тем лишь ограничением, что код исполняется на сервере, в качестве интерфейса системы выступает браузер, а в качестве среды, посредством которой происходит обмен данными, - интернет [9].

## 2.8 **Анализ аналогов**

В данном разделе произведем юзабилити тестирование. Все три описанные сети пиццерий, это российские организации, которые широко известны в своей стране.

**Ташир пицца:** Не везде есть интуитивно понятный дизайн. На сайте продаются салаты, пиццы и напитки. Имеется возможность оплатить как наличными, так и картой. Имеется постоянное отслеживание геолокации курьера. Есть возможность ввода промокодов по акциям. Ниже дизайн, указанный на рисунке 1. Высокая цена пиццы по сравнению с конкурентами. Интересная задумка с комбинацией разных пицц по низкой цене.

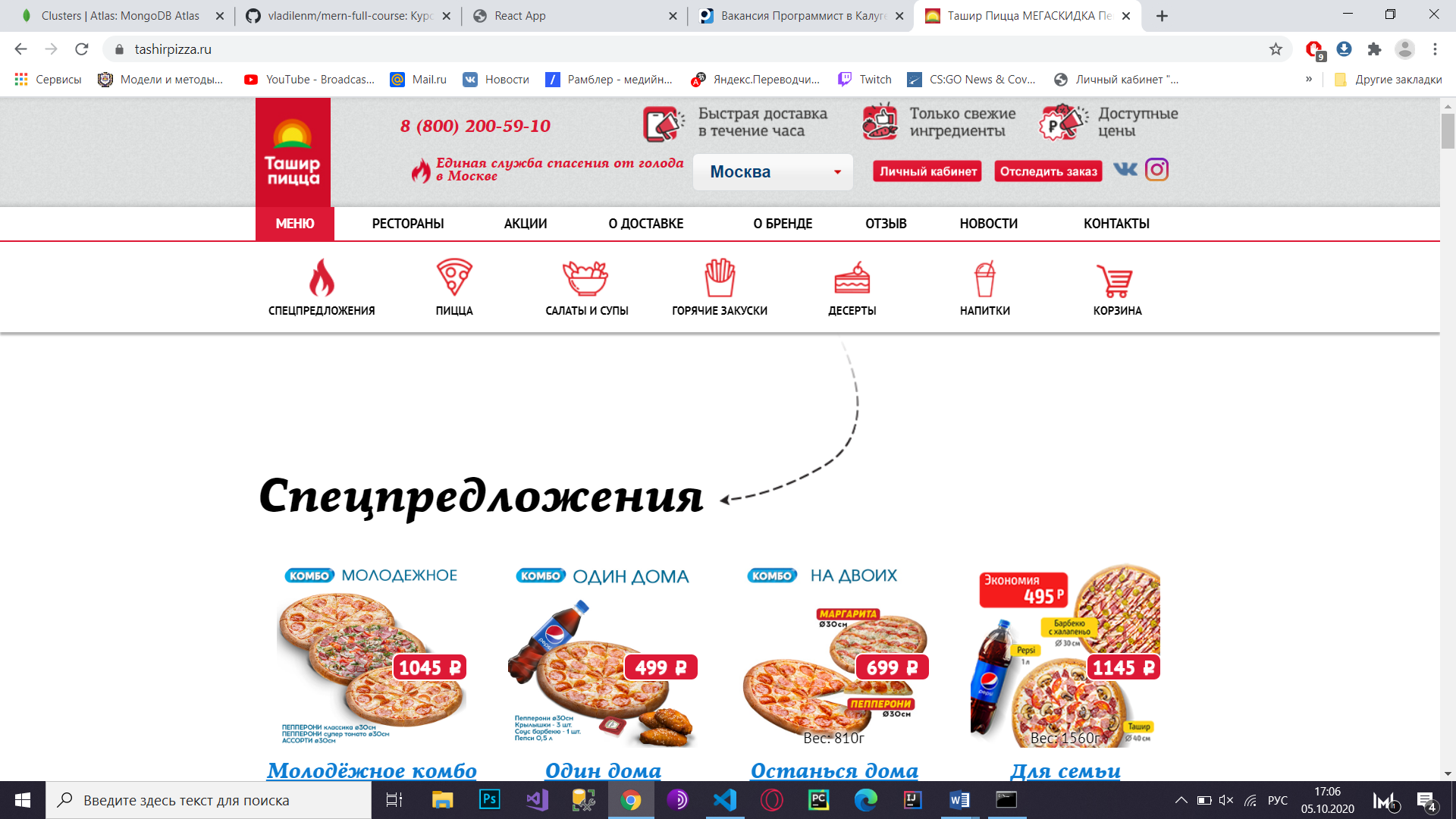


Рисунок 5 – Ташир пицца

**Додо Пицца**- интуитивно понятный дизайн веб-сайта и имеющееся приложение для телефонов. Есть возможность оплатить как наличными, так и картой. Есть промокоды для акций. Имеется функция, позволяющая собрать из разных пицц одну целую. Удобная имитация загрузки пицц на рисунке 2, ее позаимствуем для нашего сайта.

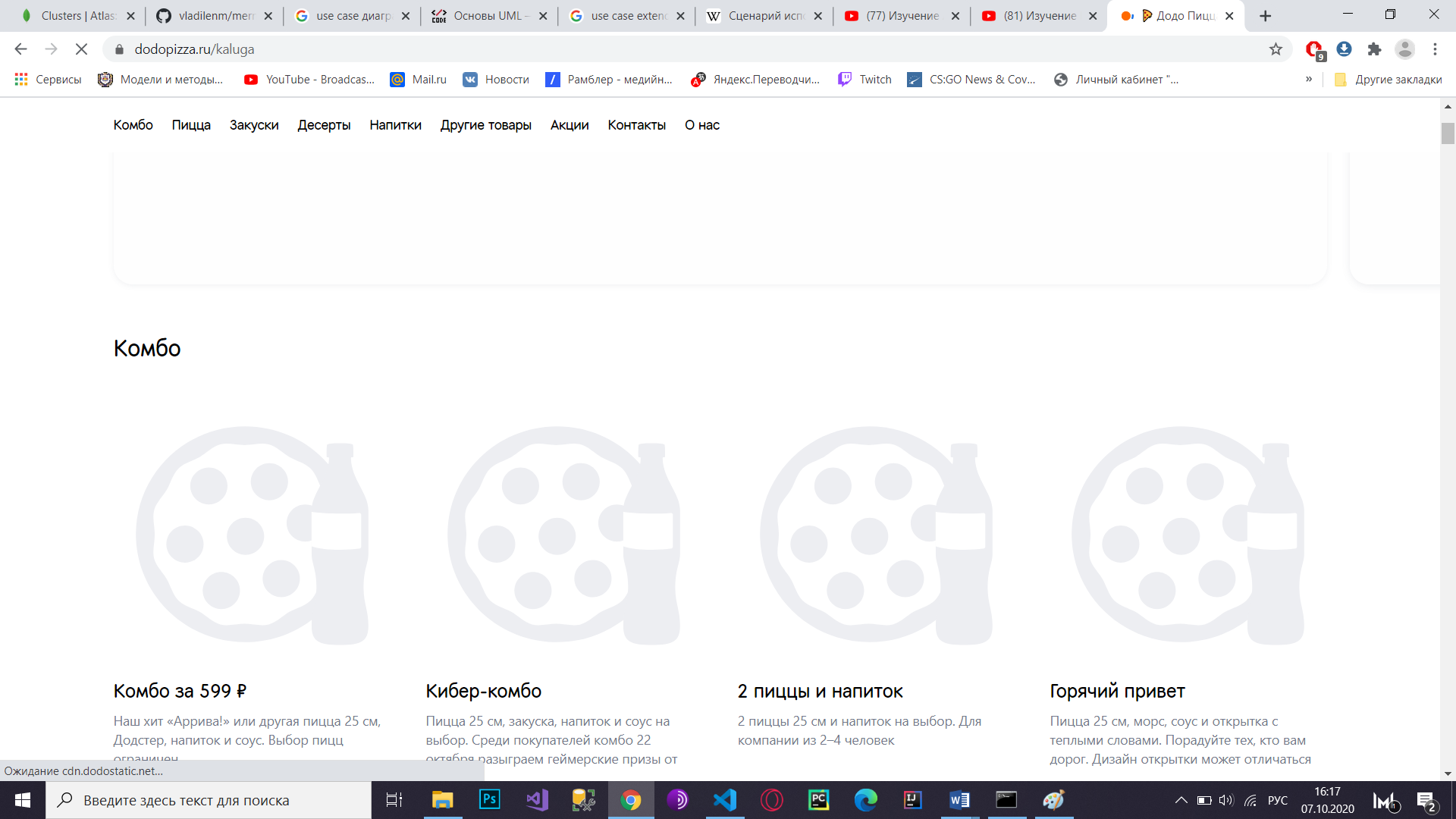
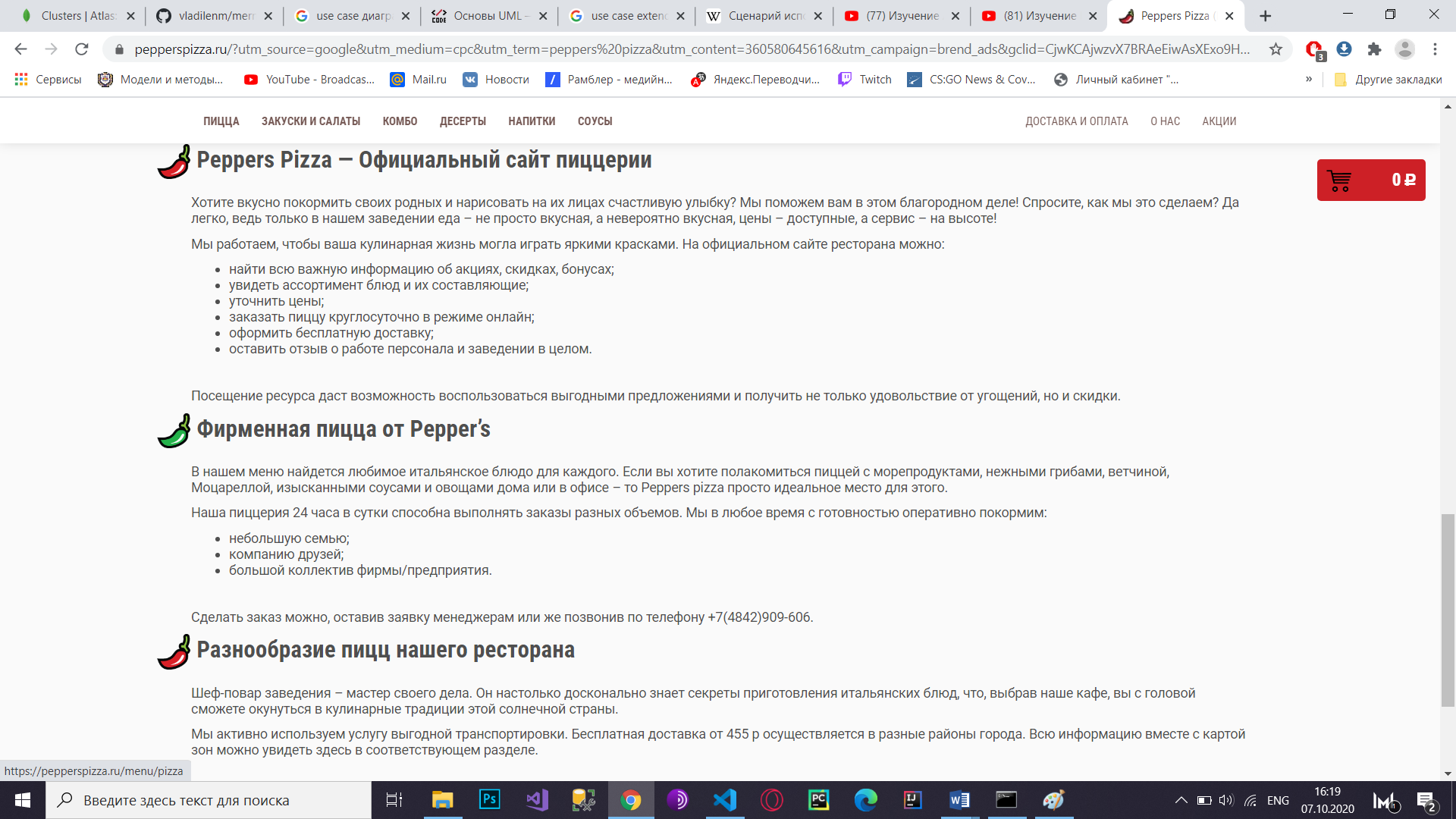


Рисунок 6- Имитация загрузки пицц.

**Peppers pizza** - большое разнообразие пицц. Имеется возможность оплатить как наличными, так и картой. Много лишней информации на главной странице сайта, указанно на рисунке 3. При самовывозе заказа пиццерия дарит 1 пиццу в подарок, что позволяет сэкономить на курьере. Заказ от 455 рублей-хорошая особенность для экономии сил и ресурсов курьеров. Также, как и у конкурентов есть промокоды для скидок и акций.

  
 Рисунок 7 – Peppers pizza.

Вывод: сайт может реализовать большую часть перечисленного функционала. Компания будет получать доход с продажи пиццы. У сайта будет интуитивно понятный для клиента дизайн, вариативный выбор видов пиццы на любой вкус. Основным плюсом будет качественно оптимизированный сайт. При опоздании курьера, возможна выдача одной бесплатной пиццы самого маленького диаметра (26 см). А также при самовывозе будет скидка 10 процентов.

## 2.9 **Архитектура системы**

Основой архитектуры серверной части будет Onion-Archtitecture.

Onion-архитектура представляет собой разделение веб-приложения на уровни(levels). Есть 1 независимый уровень, который существует в центре архитектуры. От этого уровня зависит 2й уровень, от 2го – 3й и так далее. То есть получается, что вокруг 1й независимого уровня наслаивается 2й-зависимый [10].

Первый уровень вокруг модели домена образуют интерфейсы, которые управляют работой с моделью домена. Обычно это интерфейсы репозиториев, через которые мы взаимодействуем с базой данных.

Внешний уровень составляет такие компоненты, которые очень часто изменяются. Обычно внешний уровень образуют пользовательский интерфейс, тесты, какие-то вспомогательные классы инфраструктуры приложения. К этому уровню также относятся конкретные реализации интерфейсов, объявленных на нижележащих уровнях. Например, реализация интерфейса репозитория, который объявлен на уровне Domain Services. Все внутренние уровни, которые можно соединить в Application Core, описывают интерфейсы, а определенная реализация этих интерфейсов располагается на самом внешнем уровне.

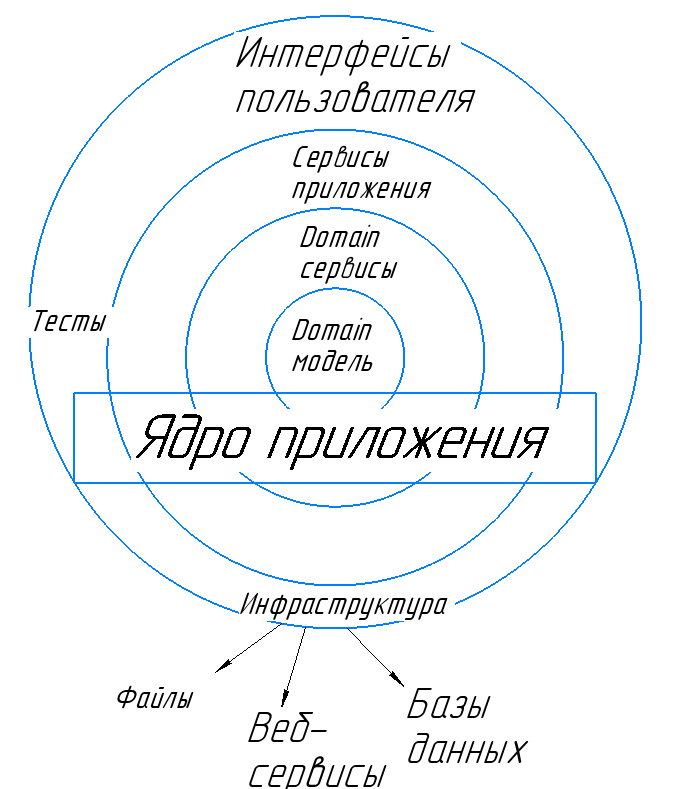


Рисунок 8 – Визуальный эскиз архитектуры.

Клиент в свою очередь будет спроектирован на архитектуре MVC отправлять запросы на сервер и получать ответ при помощи технологии axios. Axios — это JavaScript-библиотека. Она представляет собой HTTP-клиент, основанный на технологии promise и предназначенный для браузеров и для Node.js .

**3.ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ**

## **3.1 Разработка структуры сайта**

Приложение является клиент-серверным, что подразумевает под собой четыре уровня: клиент, который представляет пользовательский интерфейс, web-сервер, который осуществляет обработку запросов, сервер приложения, на котором осуществляется имитация данных

Последовательность разработки заключается в последовательном создании:

1. алгоритмов обработки данных;
2. интерфейса;
3. алгоритмов отображения данных из сервера на клиентском уровне.

## **3.2. Разработка архитектуры приложения.**

**3.2.1 Интерфейс**

Приложение состоит из модулей:

-Home

-Cart

-FinishPage

-InfoAboutSales

-Edit

Модуль Home- страница клиентской части, которая доступна любому пользователю системы. В этом модуле пользователь может просматривать различную информацию о пиццах.

Основной модуль содержит в себе следующие компоненты:

1. Категории пиццы – имеются данные категории на выбор: мясная,

2. Фильтр пиццы – имеется 3 варианта данного фильтра: по популярности, по алфавиту и по цене.

3. PizzaBlock – компонент содержащий информации об одной конкретной пицце.

Модуль Cart – модуль, который выводит данные о выбранных товарах и переходит в модуль заказа после окончательного выбора.

Модуль Cart содержит в себе:

1. Добавление пиццы – при нажатии на кнопку “добавить” в корзину будет добавляться выбранная пицца.

2. Удаление выбранной пиццы - при нажатии на кнопку “удалить” из корзины будет убираться выбранная пицца.;

3.Order – часть, которая доступна всем пользователям, отвечает за функцию заказа пиццы. Данный модуль позволяет пользователю оформить заказ с выбранными блюдами.

Модуль заказа содержит в себе следующие компоненты:

- “Оплата пиццы”;

- “Карта для определения места доставки”;

- “Данные о заказе”;

Модуль FinishPage выводит информацию о времени ожидания заказа и благодарит за заказ, доступен всем.

InfoAboutSales содержит все данные о совершенных заказах, включает номер заказа, имя курьера, дату, адрес доставки и время доставки. Очень важная страница для подведения отчетности.

Компонент Edit создается для реализации добавления, удаления и модификации данных о пиццах админом.

Ниже приведена иерархия компонентов.

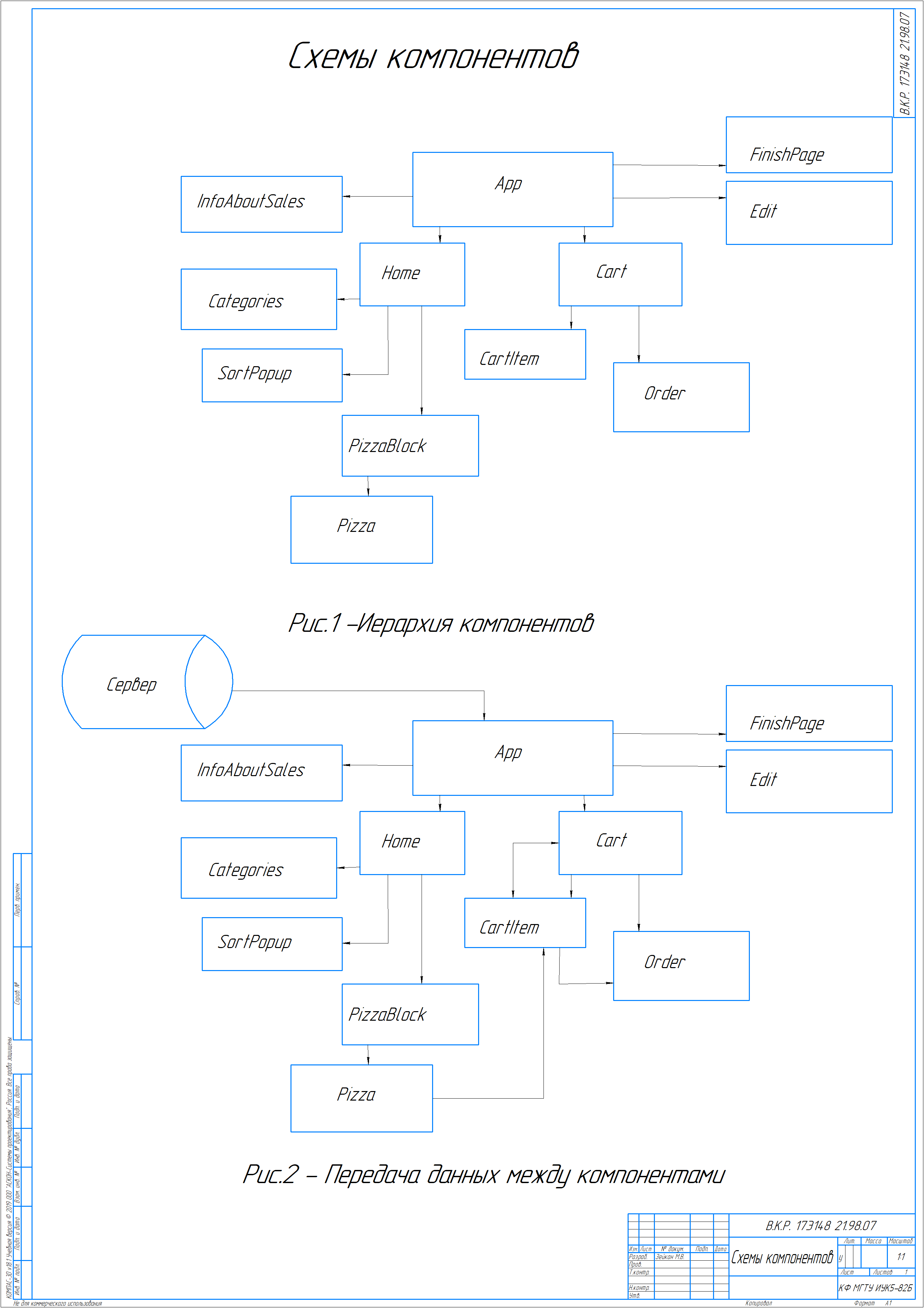


Рисунок 8 –Иерархия компонентов.

**Redux.**

Для добавления товаров в корзину нам потребуются знания redux. Redux – библиотека с простым API предназначенная для управления состояниями приложения в разных компонентах. Redux можно использовать как в Redux, Angular, Vue, так и в других фреймворках, конструируется в основном в клиенте. Включает в себя инструменты, которые упрощают передачу данных хранилища через контекст.

Ниже приведен рисунок передачи данных по компонентам с помощью Redux.

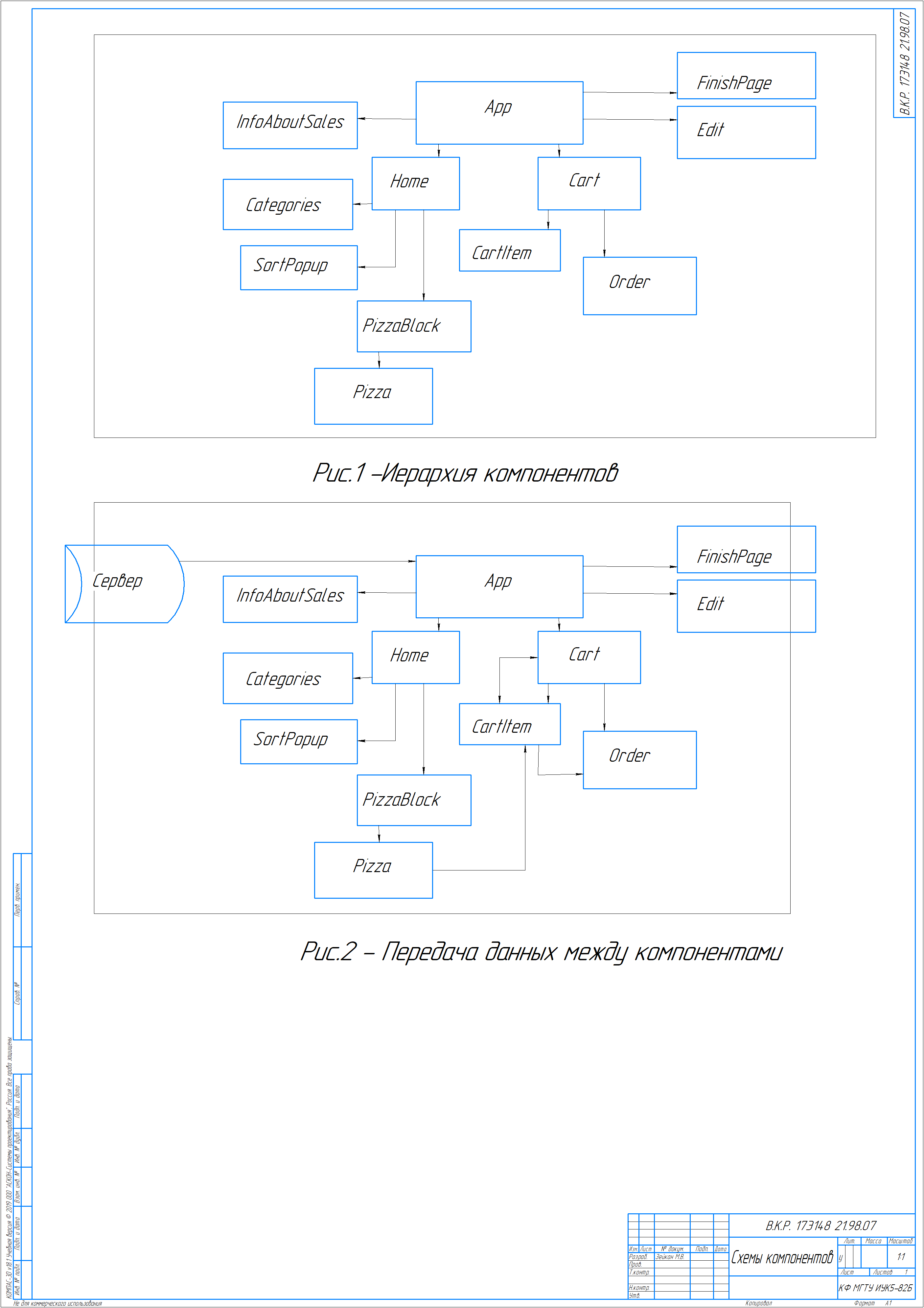
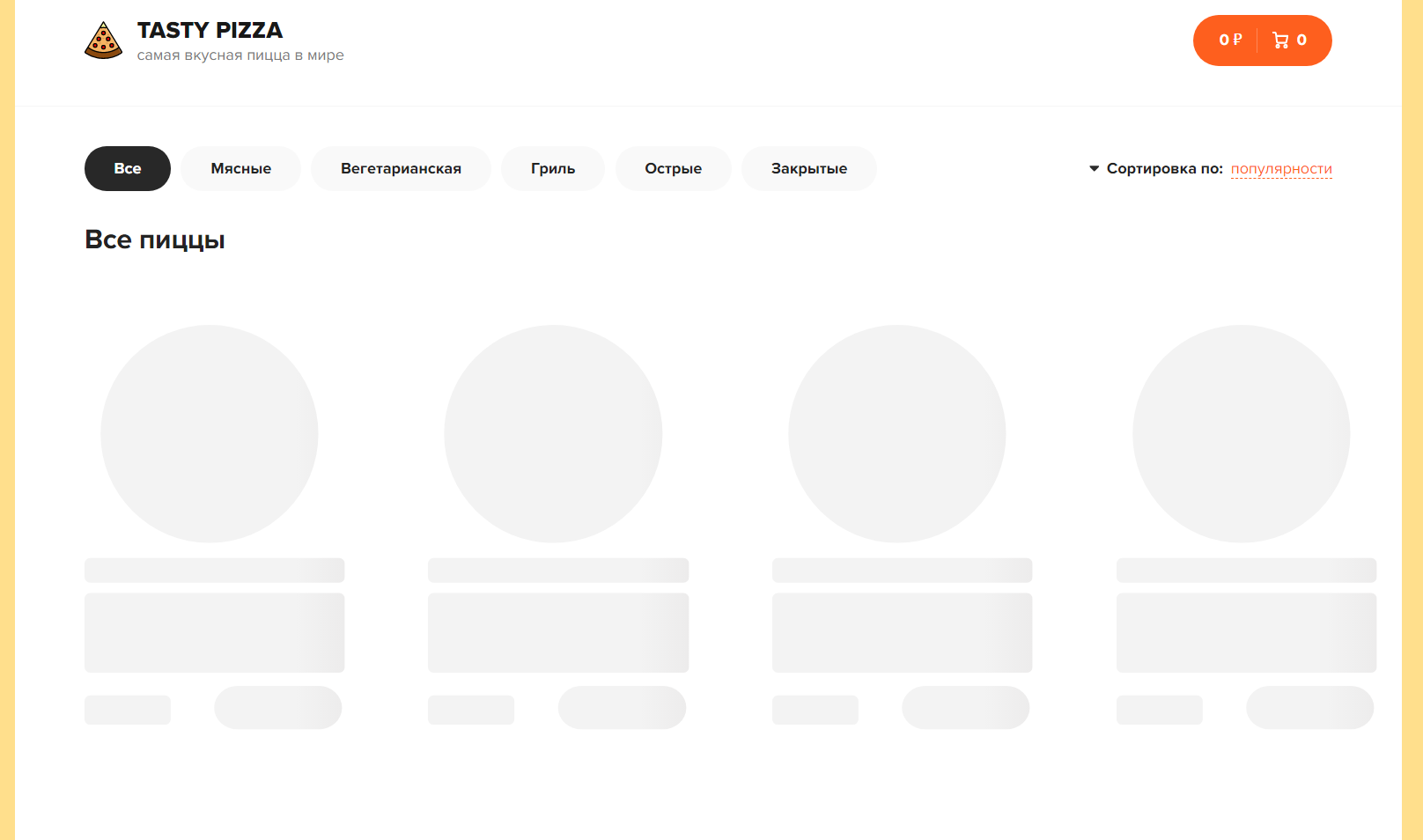


Рисунок 9 – Передача данных по компонентам.

**Клиентская часть(Frontend)**

При долгих запросах на сервер нам нужно что-то выводить на экран, когда сервер долго отвечает и клиенту нечего грузить, нам поможет библиотека “react-content-loader”, которая будет создавать имитацию загрузки контента как у Додо-пицца, такой же особенностью пользуется twitter, YouTube и многие другие, пример имитации указан на рисунке 10. Для реализации данной функции импортируем библиотеку “react-content-loader” и в наш полученный компонент записываем и настраиваемые атрибуты скорости анимации-speed, ширину-width, высоту-height карточки, а также цвет заднего фона-backgroundColor, количество и формы имитированных кнопок, пример кода приведен в приложении.

Рисунок 10 – Имитация загрузки

Для удобства ввода клиентом его данных используется стандартный input для ввода ФИО, номера и т.д., а для указания его местоположения воспользуемся библиотекой “react-google-maps”, к сожалению библиотека не обладает полной мощностью, как например через CDN. На карте клиент сможет указывать место доставки. С помощью js рассчитаем расстояние от пиццерии до точки заказа.

## 2.5.3 **Запросы**

Запросы на сервер производятся для получения данных о пицце и совершенных заказах.

Основная страница получает все доступные виды пицц и уже через кнопки категорий и фильтра алгоритм получает данные и по ним выводит отфильтрованные данные.

На странице InfoAboutSales выводится информация для отчетности.

Структура работы приложения с сервером посредством протокола http осуществляется за счёт принципа REST-Representational State Transfer. Он был придуман как простой и однозначный интерфейс для работы с данными, предполагающий всего несколько базовых операций с непосредственно сетевым хранилищем: для получения данных через ссылку - get, для получения данных заключенных в тело запроса- post, для изменения данных – put и delete- для удаления данных.

Большим плюсом использования данного варианта заключается в следующем:

-Независимость сервера от клиента – серверы и клиенты могут быть мгновенно заменены другими независимо друг от друга, так как интерфейс между ними не меняется. Северный хранить состояний клиента.

-Особенность адресов ресурсов - каждая единица данных имеет собственный уникальный url, который по сути целиком являются однозначными идентификатором ресурса.

-Независимость формата хранения данных от формата их передачи - сервер может поддерживать несколько различных форматов для передачи одних и тех же данных (JSON, XML и т.д.), но коллекционируют данные в своём внутреннем формате независимо от поддерживаем.

-Присутствие в ответе всех необходимых метаданных-помимо самих данных сервер должен возвращать детали обработки запроса, например, сообщения об ошибках, различные свойства ресурса, необходимые для дальнейшей работы с ним, например, общее число записей в коллекции для качественного отображения постраничной навигации.

## 2.6. **Логическая схема данных**



Рисунок 11 – Логическая схема

Логическая модель предметной области описывает сущности, а также их взаимоотношения между ними. Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках общей предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов.

Взаимоотношения между сущностями иллюстрируются с помощью связей, указанных на рисунках.

На рисунках 10 и 11 представлены логические модели данных для проектируемой системы с помощью концептуальной и логической схем. Имеется 7 сущностей: “Users”,” Orders details”,” Categories”,” Extra”, “Pizzas”, “Orders” и “TimeOfDelivered”.

“Users” содержит такие атрибуты: имя, телефон и адрес доставки. “Pizzas” содержит название, цену, id дополнительных параметров и id категории. “Orders” содержит дату заказа, стоимость, статус заказа, id пользователя, заказавшего пиццу и относится к таблице “Order details”, которая в свою очередь содержит количество пицц и id проданных пицц. “Extra” содержит доступные размеры и типы теста для определенной пиццы. “Categories” будет содержать виды пицц для сортировки на главном интерфейсе. “TimeOfDelivered” включает в себя время доставки и имя курьера.

В результате была спроектирована логическая схема данных, которая графически описывает данную структуру предметной области и облегчит дальнейший процесс создания модели базы данных в СУБД.

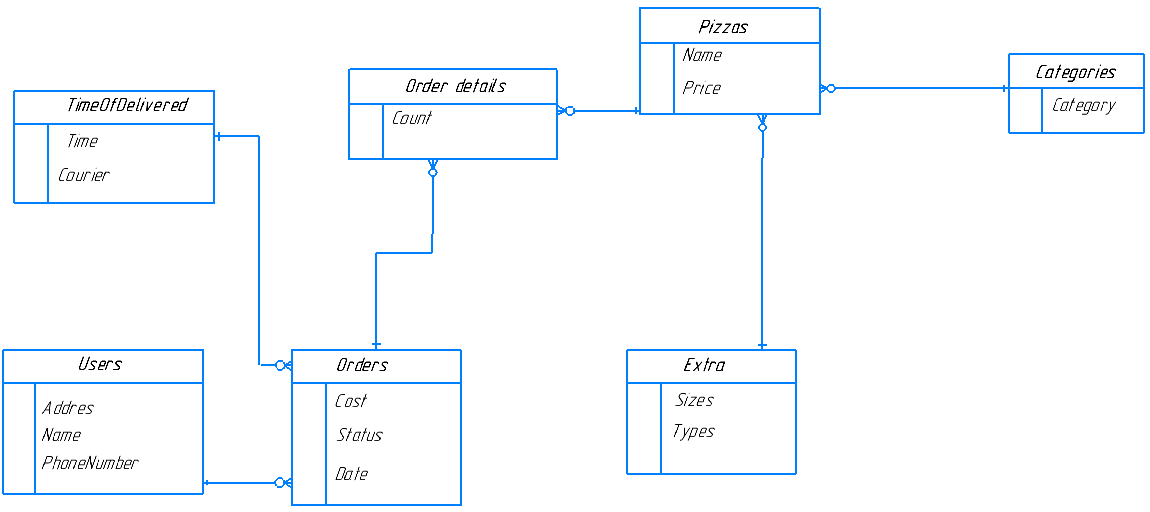


Рисунок 11 – Концептуальная схема

## 2.7 **Алгоритмы и формулы**

Веб приложение использует формулы обработки времени доставки:

, (1)

где T-полное время доставки;

D- время доставки, вычисленное на основании алгоритмов google map;

L- время на приготовление прошлых и новых заказов, которое прибавляется при различных условиях занятости печек.

Если в печке для пиццы осталось одно место, то сервер будет считать, что печка занята и ждать освобождения печки прибавляя время к заказу.

Формула для вычисления L выглядит так:

, (2)

где – сумма оставшегося времени действующих заказов;

P-время на приготовление текущего.

Формула вычисления

, (3)

где k- кол-во заказанных пицц;

t-среднее время приготовления пиццы;

q –кол-во свободных печей используемых для этого заказа, **которое будет увеличиваться от 1 до максимального количества печей в соотношении 1 печь на 2 и менее пиццы,** x коэффициент, который будет равен 1 при одной заказанной пицце, либо 2 при более чем одной заказанной пицце.

Ниже приведены алгоритмы спроектированные на данном этапе,

данный алгоритм производит вычисления времени, которое будет потрачено на приготовление заказа и доставку.

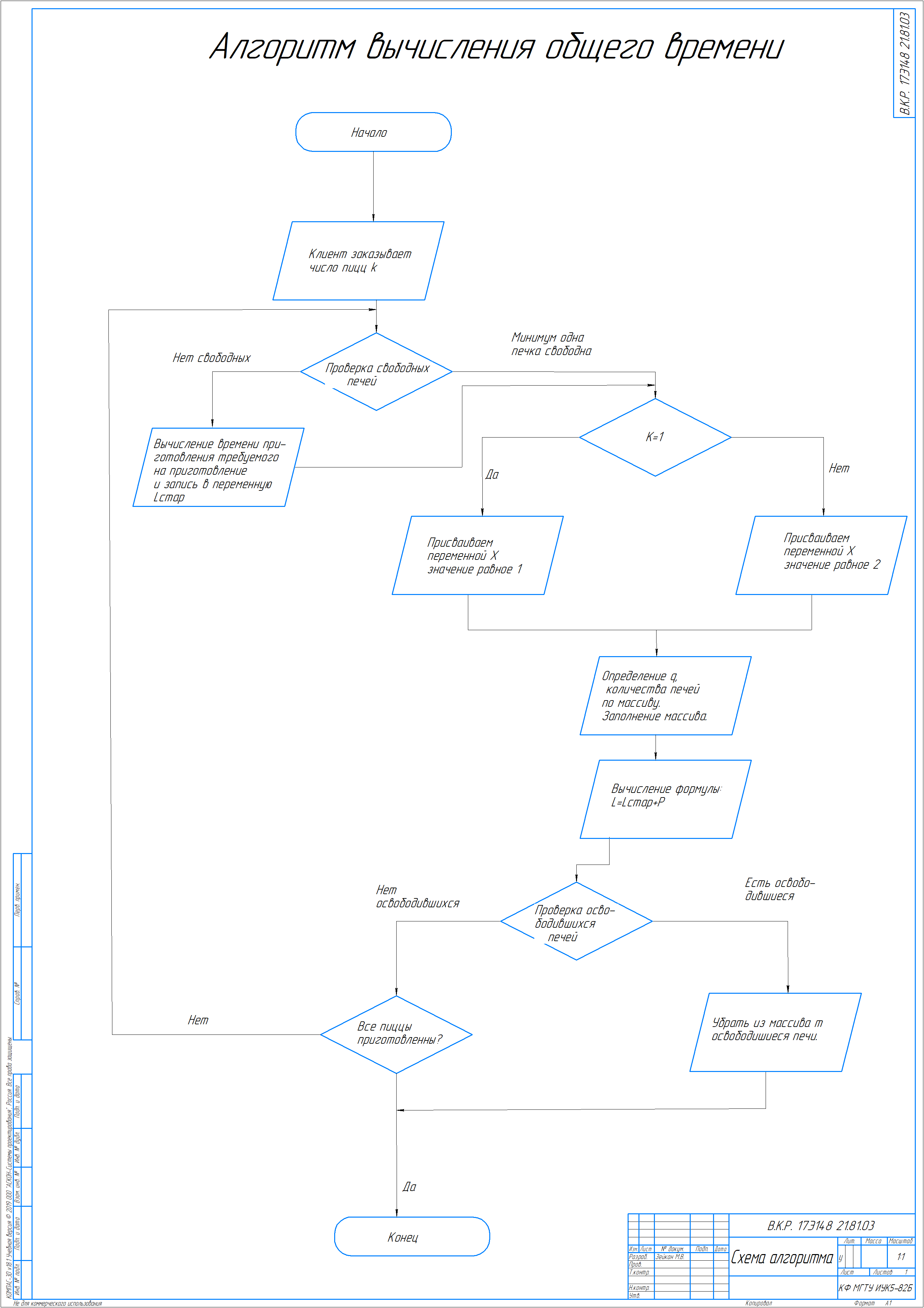


Рисунок 12 –Алгоритм, вычисляющий время приготовления и доставки.

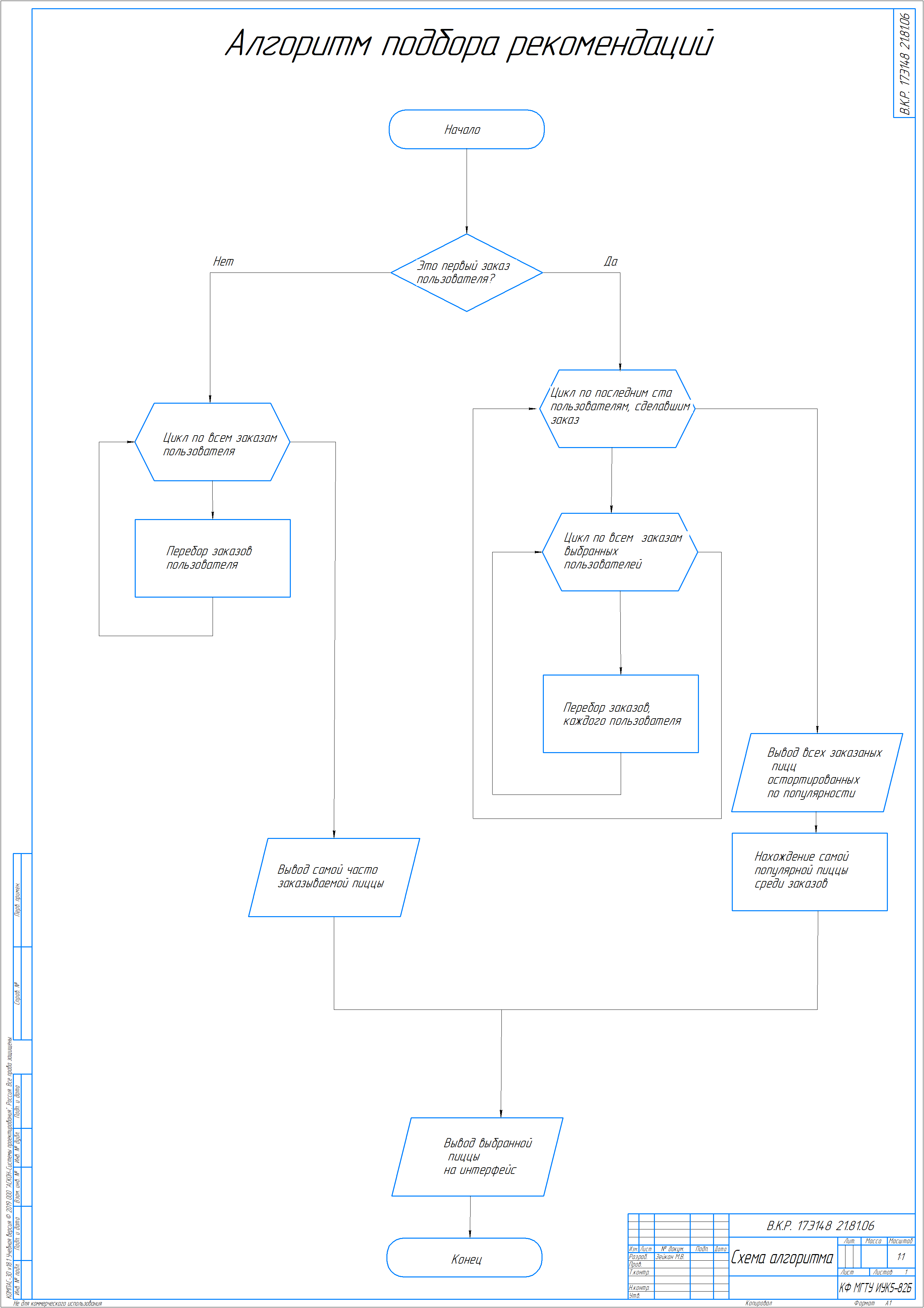


Рисунок 13-Алгоритм подбора рекомендуемой пиццы.

## 2.8. **Макет**

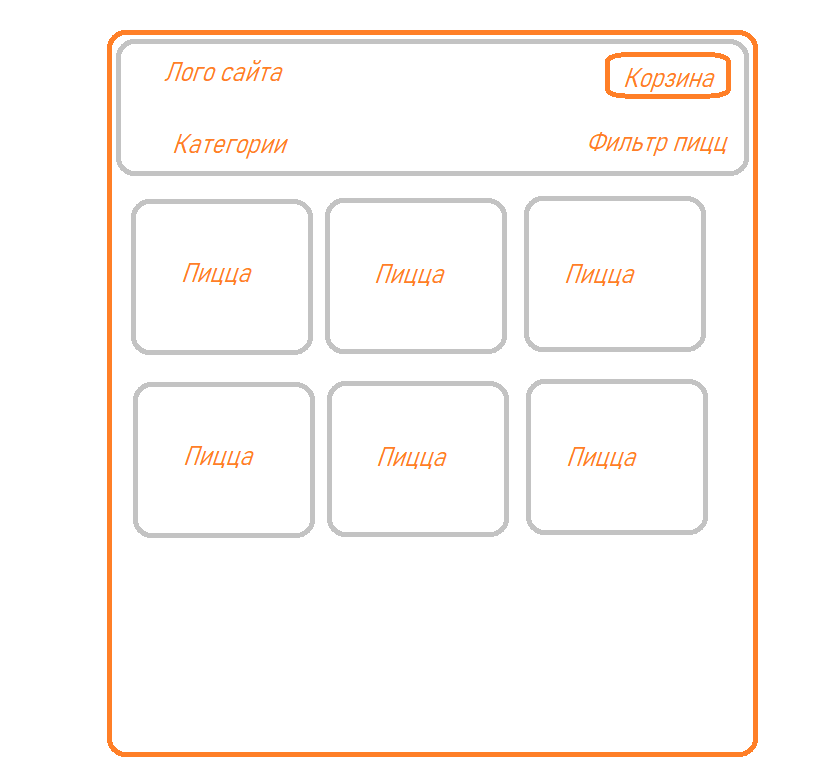


Рисунок 14 – Макет сайта.

# 

# **4.ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ**

## 4.1. **Порядок развертывания системы.**

Предполагаемый порядок развертывания системы состоит в последовательном выполнении данных действий:

1. Зайти на сайт <https://nodejs.org/en/>, скачать подходящую версию node.js для вашей системы и установить.
2. Зайти на сайт <https://dotnet.microsoft.com/download/dotnet/5.0>, скачать версию для подходящей архитектуры и установить.

3)Распаковать исходники сервера проекта, зайти в директорию проекта /TastyPizza/TastyPizza.Web и запустить при помощи команды “dotnet run”.

4)Распаковать исходники клиента проекта, зайти в директорию проекта /ReactPizza/.Web, установить все зависимости с помощью npm i и запустить при помощи команды “npm run start”.

5)Проверка работоспособности приложения.

## 4.2. **Интерфейс взаимодействия пользователя с системой.**

В процессе проектирования разрабатываемой системы был реализован интерфейс приложения, рисунки указаны ниже. Основная страница реализована по требованиям и макету и имеет все необходимые для продажи пиццы функции.

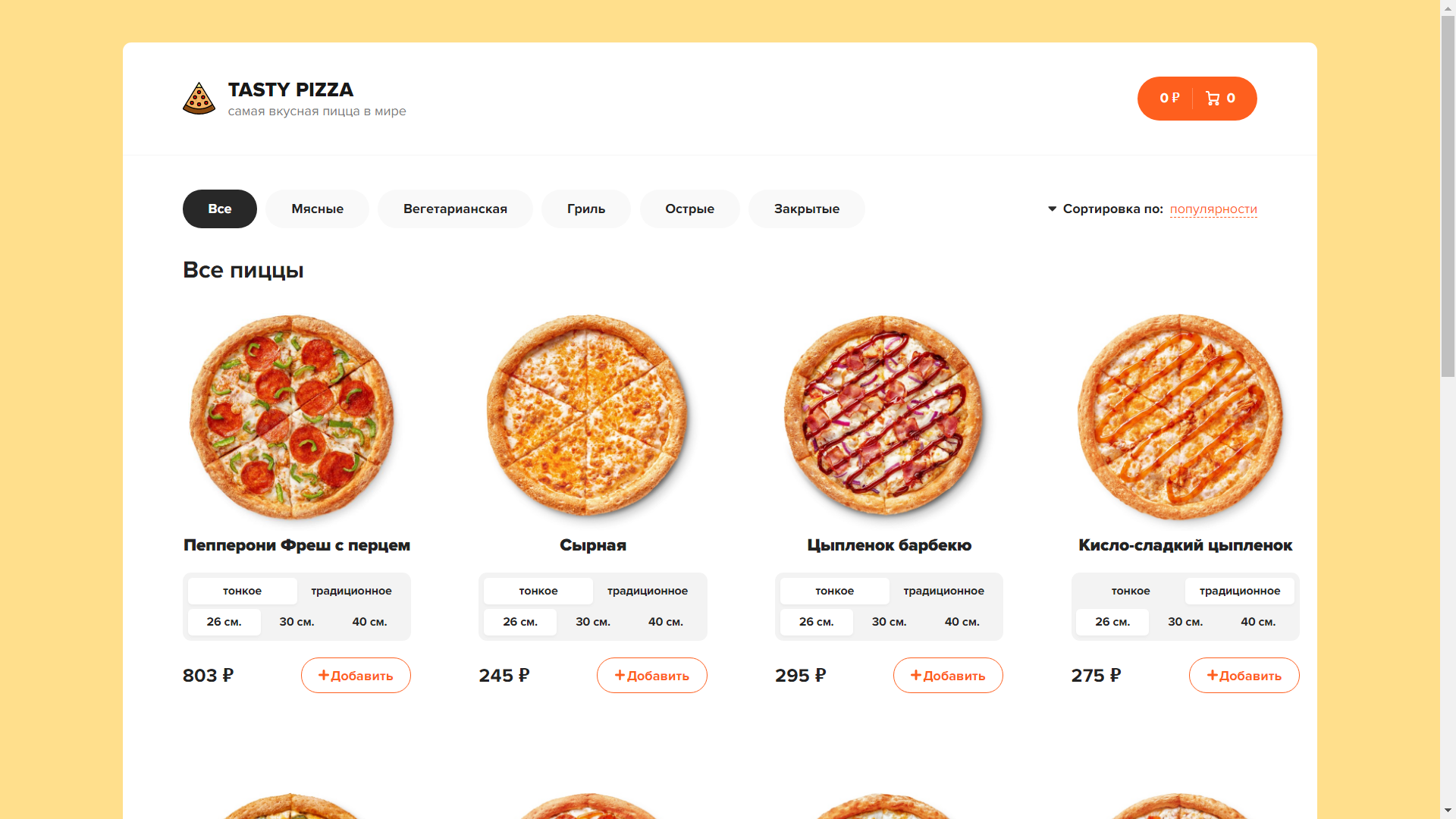


Рисунок 15 –Основная страница.

Если пицца не выбрана, то страница корзины будет выводить данную картинку, указанную на рисунке 16 и предлагать вернуться назад.

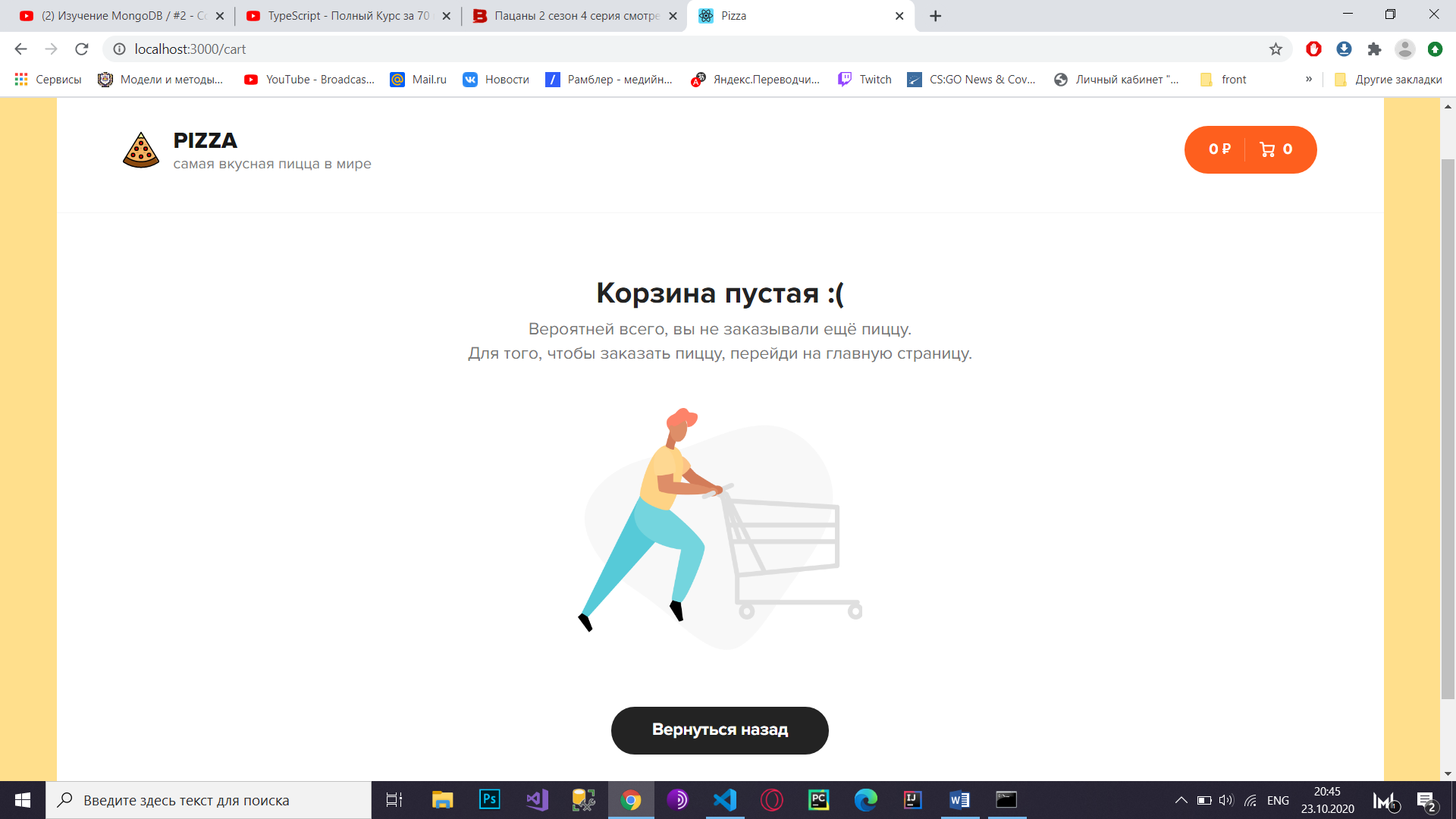
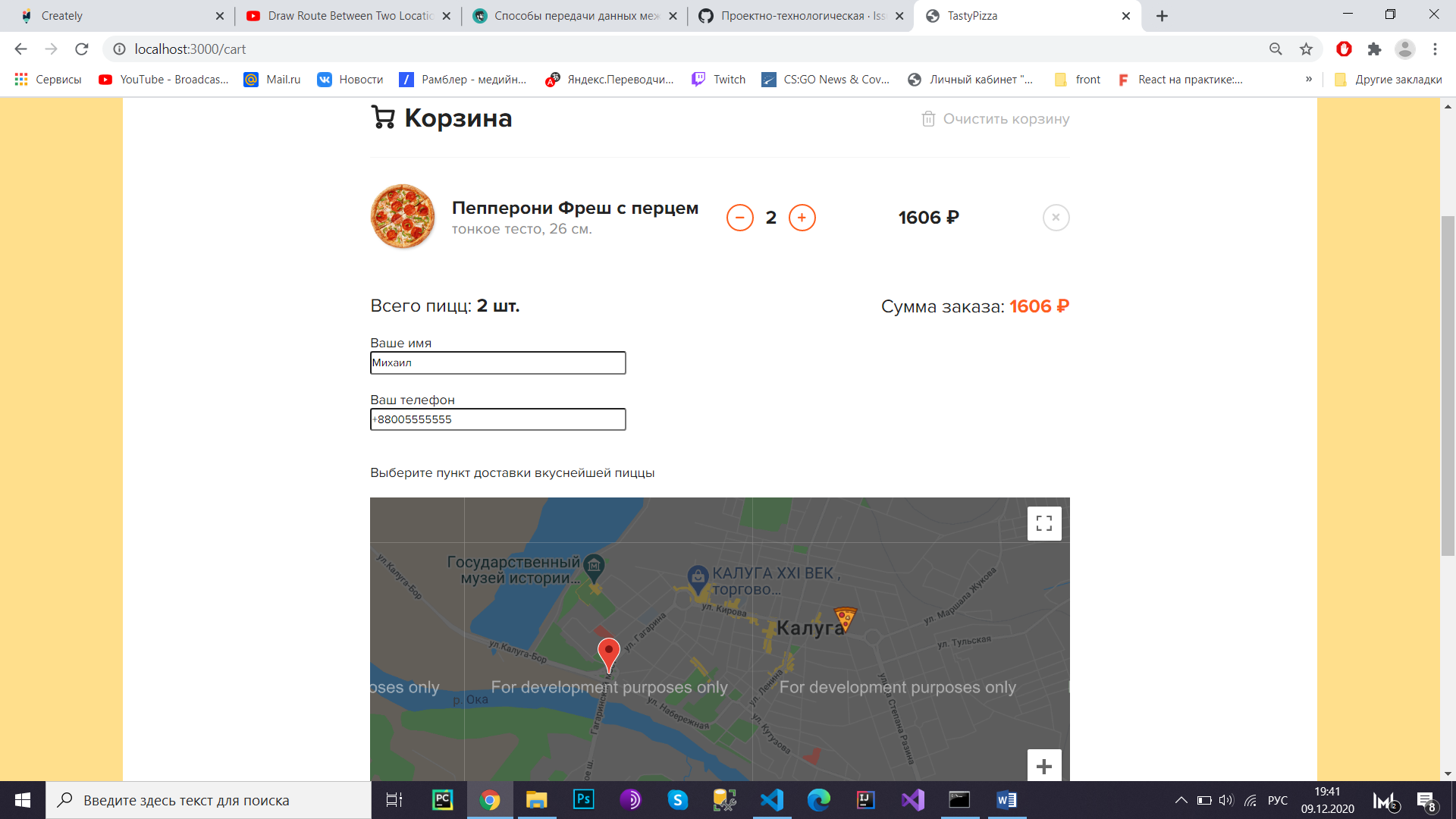
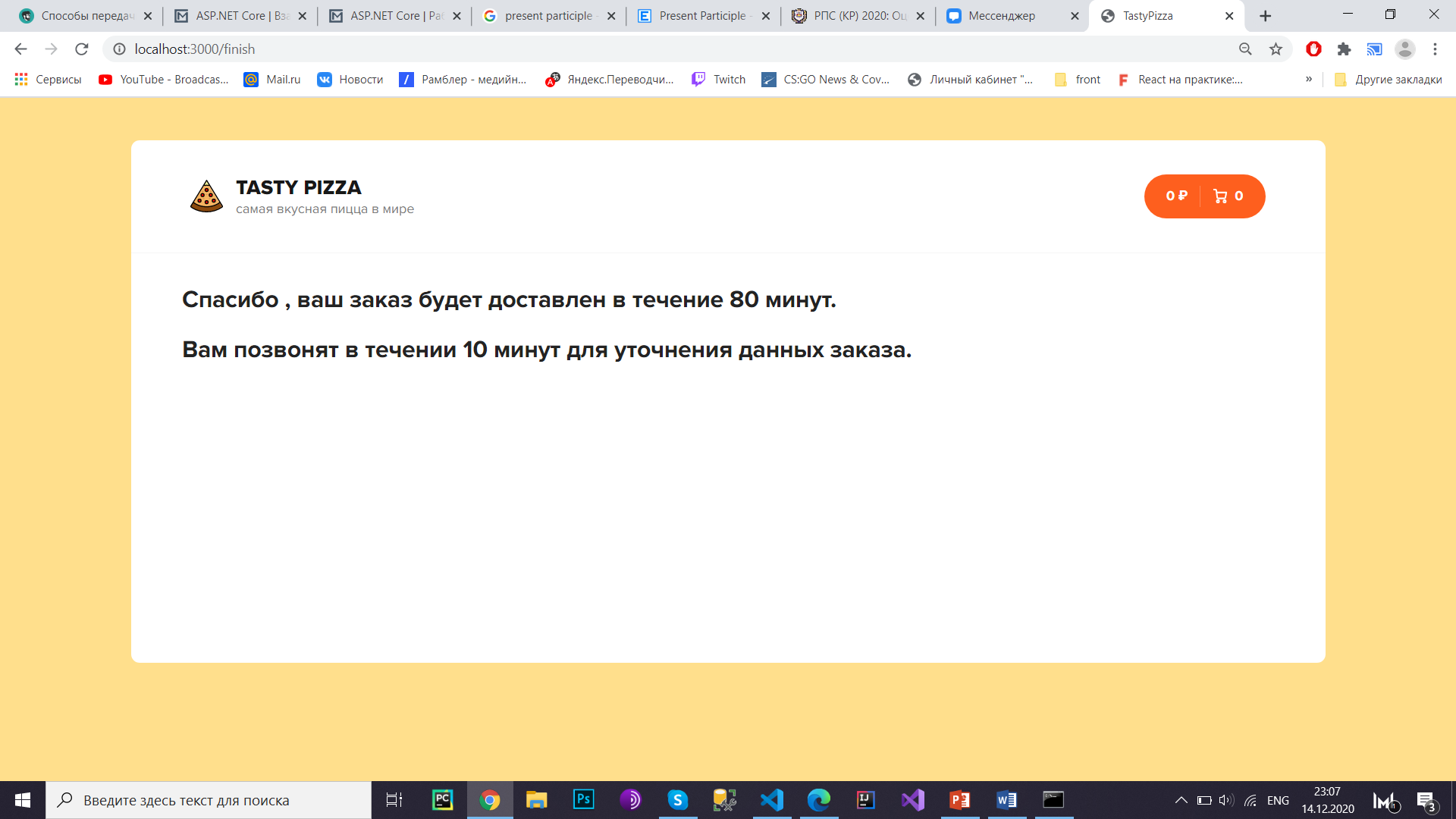
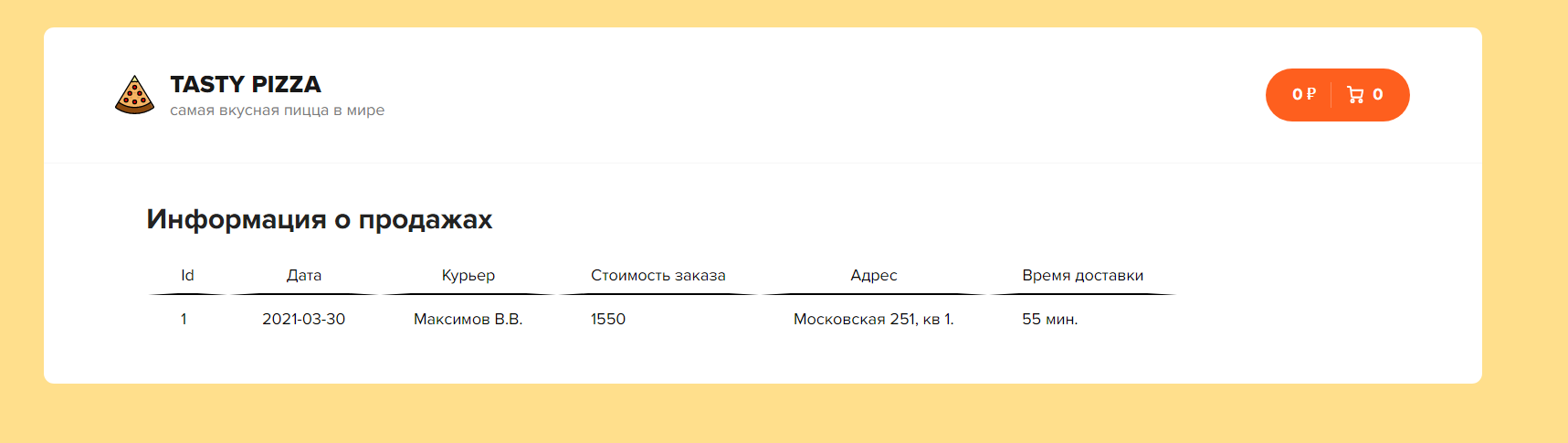


Рисунок 16–Пустая корзина

Рисунок 17– Подготовка заказа.

Рисунок 19 – Заказанный продукт при занятых печках.

На рисунке 19 можно увидеть страницу отчетности, админ будет проверять, как проходят дела у пиццерии в плане дохода.

Рисунок 20 – Страница отчетности.

## 4.3 **Руководство пользователя**

Руководство показано на рисунках ниже.

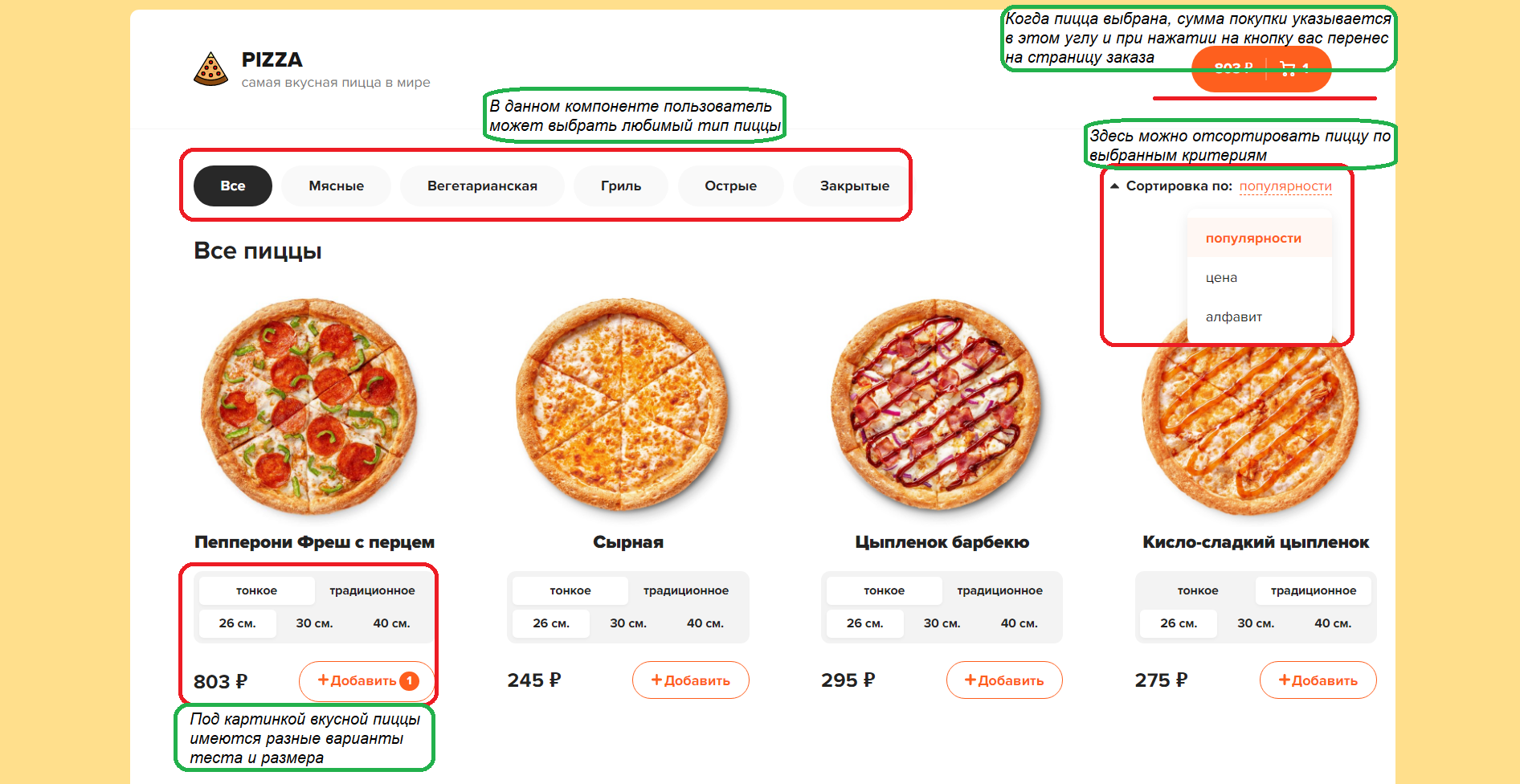


Рисунок 21 – Руководство 1

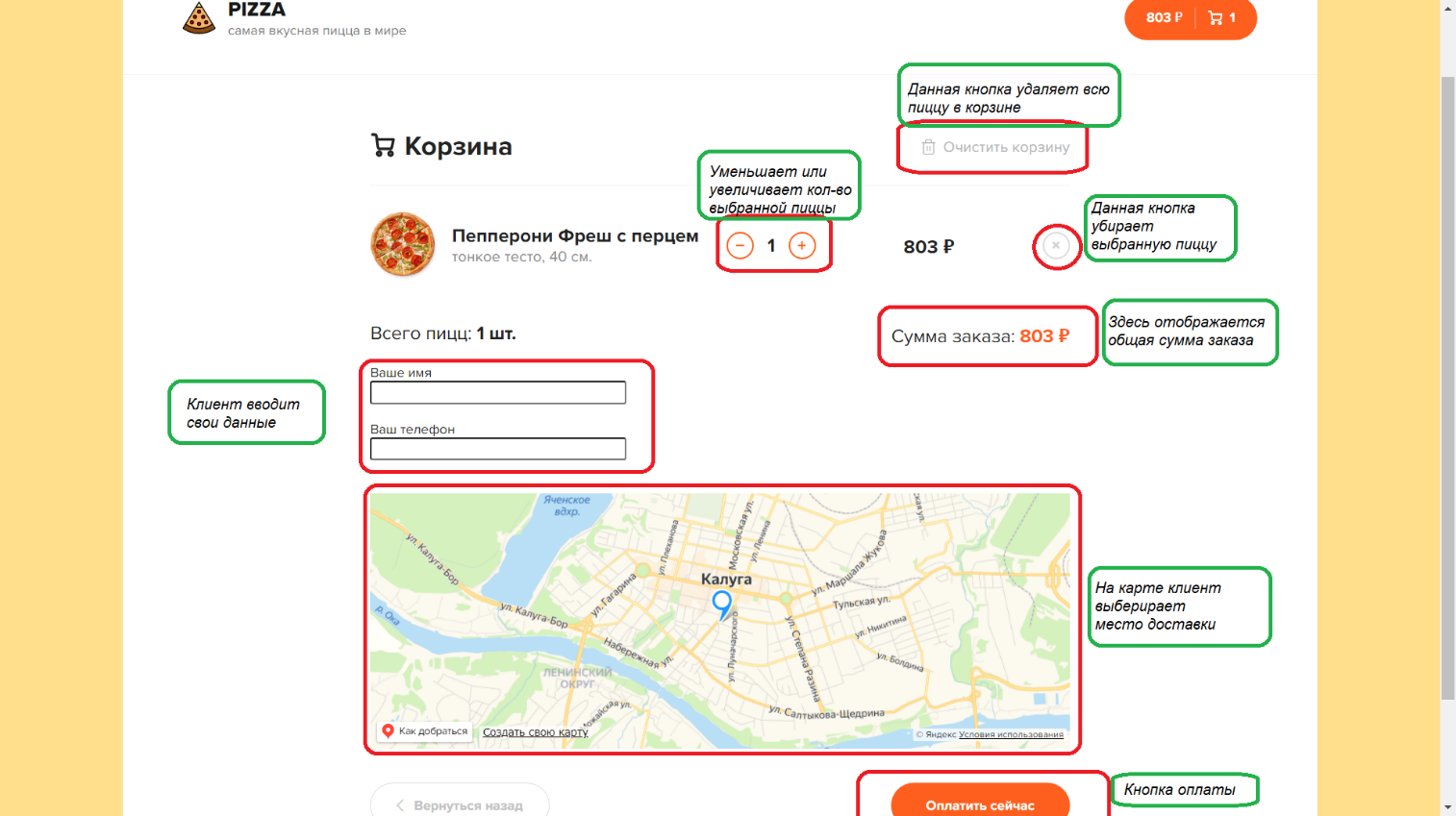


Рисунок 22 – Руководство 2



Рисунок 23 – Страница Edit.

## **4.4 Экономическая часть**

## **4.4.1 Цели и задачи, решаемые в экономической части**

В данном дипломном проекте было разработано веб-приложение “TastyPizza” для пиццерии. Результатом решения этой задачи будет являться программное обеспечение. Целью данного раздела является расчет затрат. В результате расчета находится себестоимость прикладной программы.

Для нахождения себестоимости необходимо учесть:

а) амортизационные отчисления на полное восстановление технических средств и программного обеспечения;

б) трудоемкость разработки программного продукта;

в) оплату труда программиста;

г) доплаты и надбавки к заработной плате;

д) затраты электроэнергии, расходуемой техническими средствами;

е) накладные расходы;

ж) единый социальный налог.

## **4.4.2 Расчет амортизационных отчислений**

В данной научно-исследовательской работе используются следующие технические средства:

1)      IBM совместимый компьютер;

2)      монитор LCD Acer 19”.

Закупочная цена компьютера – 50000 руб., монитора – 10000 руб. Примем норму амортизации на технические средства 20 %.

Общая стоимость технических средств, руб.:

(4)

где    – цена компьютера,

          – цена монитора.

Отсюда:

=50000+10000=60000.

Для создания пакета программ, являющегося конечным результатом исследований, применялось следующее программное обеспечение:

-       платформа Windows 10 – 14000 руб.;

Общая стоимость программного обеспечения составляет 14000 руб.

Общая стоимость технических средств и программного обеспечения, руб.:

                                                      (5)

=60000+14000=74000.

Годовые амортизационные отчисления на полное восстановление технических средств и программного обеспечения рассчитываются по формуле, руб.:

                                                              (6)

=74000·0,2=14800.

Амортизационные отчисления за период создания программного продукта, руб.:

 ,                                                           (7)

где   =85 дней – количество отработанных дней,

=280 дней – количество рабочих дней в году.

 =4492.85,

## **4.4.3 Расчет расходов на энергопотребление**

ПЭВМ, на которой была разработана программа, является потребителем электрической энергии сети переменного тока, напряжением 220 В. Согласно технической документации, суммарная мощность, потребляемая компьютером и монитором, составляет:

= 250 Вт·ч.

Расход денежных средств, связанный с энергопотреблением технических средств можно найти по формуле, руб.:

=Кдн·Враб·Мс·Цэн,                                                (8)

где       – период написания программы, дн., =85 дней;

– длительность рабочей смены, ч., =6 часов;

– мощность, потребляемая техническими средствами, кВт·ч;

– стоимость электроэнергии по действующим тарифам, р./кВт·ч; =1,5 рубля за кВт·ч.

Отсюда:

=85·6·0,25·1,5 = 191,25.

## **4.4.4 Расчет заработной платы программиста**

Исходя из фактически отработанного времени программиста, которое составило 55 рабочих шестичасовых дней, найдем количество фактически отработанного времени, ч.:

,                                                        (9)

где  – фактически отработанное время, ч.;

– количество отработанных дней, дн;

– продолжительность рабочего дня, ч.

=85·6=510 ч.

Принимая часовую заработную плату программиста в расчете 350 руб., получим основную заработную плату, руб.:

,                                                      (10)

где – часовая тарифная ставка программиста.

=510·350=178500.

Для определения общей суммы расходов на оплату труда необходимо учесть доплаты и надбавки. Принимаем удельный вид доплат и надбавок в размере 15 % от основной заработной платы, руб.:

=Зосн·0,15,                                                    (11)

=178500 ·0,15=26775.

Отсюда находим общие расходы на оплату труда, руб.:

=+,                                                               (12)

=178500+26775=205275.

Далее определим единый социальный налог; в соответствии со ставкой он составляет 26% от расходов на оплату труда, что составит, руб.:

                                                    (13)

=205275 ·0,26≈ 53371,5.

## **4.4.5 Расчет общих затрат на создание пакета программ**

Итого, затраты на создание пакета программ, составляют, руб.:

,                                           (14)

Т=4492,85 +191,25+205275+53371,5 ≈263330,6.

## **4.4.6 Определение отпускной цены программы**

Для нахождения отпускной цены необходимо учесть:

-        прибыль от реализации 20 %;

                                                                   (15)

где        – прибыль от реализации продукта, руб.

                                                                  (16)

=263330,6 +263330,6 ·0,2 ≈ 315996,72.

## **4.4.7 Расчет годовых затрат на эксплуатацию программы**

Расчет годовых затрат необходимо провести для последующего анализа эффективности данного программного продукта.

Годовые затраты на эксплуатацию программы составляют:

    =n·С                                                    (17)

где – стоимость одного непосредственного решения на ЭВМ, руб.;

- нормативный коэффициент сложности (0.2–0.5);

С – себестоимость разработанной программы;

n – плотность потока заявок = 8000 з/год.

Стоимость одного непосредственного решения определяется, руб.:

                                     (18)

где   . – стоимость одного часа работы на ЭВМ;

 – время решения задачи на ЭВМ;

– трудоемкость программиста, затраченная на решение задачи на

ЭВМ (50 ч.);

 – районный коэффициент (1.3);

 – расходы косвенные.

. = 8 · 50 + 30 · 50 · 1,3 · 1.1 = 2545.

 = 8000 · 2545 + 0.3 · 14264 =20 364 279,2 р.

## **4.4.8 Расчет годовых затрат на выполнение работ ранее употреблявшимся способом**

Для определения затрат для ручной обработки данных необходимо иметь данные по квалификации специалиста и затраты времени при числовой обработке. При отсутствии данных затраты ручной обработки рассчитываются путем хронометража работы дипломником при выполнении расчетов по данному способу. Для нахождения годовых затрат на выполнение расчетов ранее употреблявшимся способом необходимо знать стоимость всех работ по выполнению одного расчета. Она составит, руб.:

     ,                                 (19)

где       . – стоимость выполненных расчетов ранее употреблявшимся способом;

 – зарплата специалиста, который проводил расчеты ранее

употреблявшимся способом;

 – затраты времени специалиста на один расчет (150 ч.);

– районный коэффициент (1.3);

– накладные расходы

=30 · 150 · 1.3 · 1,1 = 6435.

Зная плотность потока заявок по данному расчету определяем годовые затраты, руб.:

=n·,                                     (20)

= 8000 · 6435 =51 480 000.

## **4.4.9 Определение экономического эффекта программы**

При расчете экономического эффекта может быть получен отрицательный результат.

Это говорит о том, что выбранный метод с точки зрения экономического анализа менее эффективен.

Годовая экономия определяется по формуле, руб.:

                                          (21)

где        – годовые затраты на решение ранее применявшимся способом;

. – годовые затраты машинного решения задачи

 = 51480 000 – 31 115 720= 31 115 720.

**4.4.9 Определение расчетного коэффициента экономической эффективности программы.**

Расчет экономической эффективности проводится для определения наиболее экономически выгодного варианта обработки информации. Экономическая эффективность характеризует отношение общей величины эффекта к затратам, его вызвавшим, руб.

                                           (22)

где     С – себестоимость программы.

= 31 115 720 / (260 000 + 20 364 279) = 1,5

## **4.4.10 Срок окупаемости разработанной программы**

Срок окупаемости характеризуют период времени, в течении которого общие затраты на составление программы возмещаются за счет экономии текущих расходов. Срок окупаемости представляет собой отношение затрат на составление программы к экономическому эффекту от его внедрения, г.

        =1/                                      (23)

где   – срок окупаемости

  = 1 / 1,5 = 0,66.

Из расчетов вытекает, что программа окупится за 8 месяцев.

Выбранный способ решения считается экономически эффективным.

## 4.5 **Дальнейшее развитие**

В дальнейшем планируется доработка серверной части на платформе asp.net core 5, улучшение алгоритмов вычисления времени доставки, улучшение алгоритмов подбора рекомендации пиццы, авторизация/регистрация, работа над системой лояльности, а также возможность расширенной части администрирования, которая будет включать постоянный мониторинг статистики заказов и возможность добавления новых видов пицц. Также в планах добавление платежного модуля-онлайн и проектирование мобильного приложения под операционные системы Android и IOS.

# 

# **ЗАКЛЮЧЕНИЕ**

В результате выполнении выпускной квалификационной работы было спроектировано и написано приложение для работы пиццерии.

В разделе научно-исследовательской работы был обоснован выбор платформы и фреймворков, поставлены задачи, которые нужно решить. Был разработан алгоритм вычисления времени приготовления и доставки пиццы, описаны сценарии использования.

В проектно-конструкторской были спроектированы архитектура приложения, база данных, а также были разработаны и реализованы серверная часть приложения, веб-интерфейс системы и алгоритмы обработки данных.

В проектно-технологической был описан порядок развертывания веб-приложения, разработаны интерфейсы для взаимодействия пользователя с системой и написано руководство пользователя.

Получены навыки работы с платформами ASP Net Core 5, TypeScript, фреймворком React.js, в частности изучены основные хуки для обработки состояний в среде React, а также множество библиотек таких как: redux, google-map, bootstrap, express, config, content-loader, react-spring и так далее.

Цель данной работы, состоящая в проектировании веб-приложения для пиццерии, была достигнута и все задачи решены полностью.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Идиатуллина, К.С. Магистерская диссертация [Электронный ресурс]: учебное пособие / К.С. Идиатуллина, И.З. Гарафиев. - Казань : Издательство КНИТУ, 2012. - 88 с. : табл. - Библиогр.: с. 42-43. - ISBN 978-5-7882-1272-2 ;

Режим доступа: [http://biblioclub.ru/index.php?page=book&id=258812](http://biblioclub.ru/index.php?page=book&id=258812%20)

2. Вылегжанина, А.О. Разработка проекта [Электронный ресурс]: учебное пособие / А.О. Вылегжанина. - М. ; Берлин : Директ-Медиа, 2015. - 291 с. : ил., схем., табл. - URL: <http://biblioclub.ru/index.php?page=book&id=275277>

3. Ласковец, С.В. Методология научного творчества [Электронный ресурс]: учебное пособие / С.В. Ласковец. - М. : Евразийский открытый институт, 2010. - 32 с. - URL: <http://biblioclub.ru/index.php?page=book&id=90384>

4. Информационные ресурсы и поисковые системы: учебное пособие / Н.В. Максимов, О.Л. Голицына, Г.В. Тихомиров, П.Б. Храмцов. - М.: МИФИ, 2008. - 400 с. – URL: <http://biblioclub.ru/index.php?page=book&id=231125.>

5. Сергеев А.Н. Основы локальных компьютерных сетей [Электронный ресурс] : учеб. пособие — СПб.: Лань, 2016. — 184 с. — Режим доступа: [https://e.lanbook.com/book/87591](https://e.lanbook.com/book/87591%20)

6. Шнырёв С.Л. Базы данных: учебное пособие для вузов [Электронный ресурс] / С.Л.Шнырев. — Электрон.дан. — М. : НИЯУ МИФИ, 2011. — 224 с.

Режим доступа: <http://biblioclub.ru/index.php?page=book&id=231519>.

7. Сычев А.В. Перспективные технологии и языки веб-разработки [Электронный ресурс]: практикум/ Сычев А.В.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019.— 493 c.

Режим доступа: <http://www.iprbookshop.ru/79730.html>

8. Бурков А.В. Проектирование информационных систем в Microsoft SQL Server 2008 и Visual Studio 2008 [Электронный ресурс]/ Бурков А.В.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 310 c.— Режим доступа: <http://www.iprbookshop.ru/52166.html>

9. Грекул В.И. Проектирование информационных систем. Курс лекций [Электронный ресурс]: учебное пособиеnдля студентов вузов, обучающихся по специальностям в области информационных технологий/ Грекул В.И., Денищенко Г.Н., Коровкина Н.Л.— Электрон. текстовые данные.— Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017.— 303 c.— Режим доступа:

<http://www.iprbookshop.ru/67376.html>.

10. Аверченков, В.И. Основы научного творчества: учебное пособие / В.И. Аверченков, Ю.А. Малахов. - 2-е изд., стер. - М.: Флинта, 2011. - 156 с. – Режим доступа: <http://biblioclub.ru/index.php?page=book&id=93347>.

11. Таненбаум, Э., Уэзеролл, Д. Компьютерные сети. 5-е изд. – СПб.: Питер, 2012 – 960 с. 16. Чекмарев, Ю.В. Локальные вычислительные сети. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2010. — 200 с.

Режим доступа: <http://e.lanbook.com/book/1147> — Загл. с экрана.

12. Эрик, Р. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL [Электронный ресурс] / Р. Эрик, Р.У. Джим.. — Электрон.дан. — М. : ДМК Пресс, 2013. — 384 с. Режим доступа: <http://e.lanbook.com/book/58690>.

13. Алексеев В.А. Паттерны проектирования программных систем [Электронный ресурс]: методические указания к проведению лабораторных работ по курсу «Архитектура программных систем»/ Алексеев В.А.— Электрон. текстовые данные.— Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2016.— 33 c.—

Режим доступа: <http://www.iprbookshop.ru/74412.html>

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

14. Научная электронная библиотека <http://eLIBRARY.RU>.

15. Электронно-библиотечная система издательства «Лань» <http://e.lanbook.com>.

16. Электронно-библиотечная система «Университетская библиотека онлайн» <http://biblioclub.ru>.

17. Электронно-библиотечная система «IPRbooks» <http://www.iprbookshop.ru>.

18. Электронная библиотека диссертаций Российской государственной библиотеки. <http://diss.rsl.ru>

19. Реферативная база научно-технических журналов. <http://www.scopus.com>

20. Открытый архив номеров научного журнала «Информатика и системы управления» <http://ics.khstu.ru/journal/issues/>

21.Кирупа Чиннатамби. Изучаем React. Практическое руководство по созданию веб-приложений при помощи React и Redux. Второе издание. –Москва: Эксмо, 2019. — 368 c.

22. Волков, Ю.Г. Диссертация: Подготовка, защита, оформление: Практическое пособие / Ю.Г. Волков. - М.: Гардарики, 2005. - 185 с.

23. Панфилов, К. Создание веб-сайта от замысла до реализации / К. Панфилов. Дик Пресс, 2009, 440 с.

24. Таненбаум, Э. Компьютерные сети [Текст] / Э. Таненбаум.- 4-е изд.- СПб. :Питер, 2010.- 992 с. :ил. - ("Классика computer science")

25. Советов, Б.Я. Базы данных. Теория и практика: учебник для бакалавров. / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. − М: Юрайт, 2012. – 464 с.

26. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. Учебное пособие. 4-е издание – СПб.: Питер, 2010 – 945 с.

27. Иванова Г.С., Ничушкина Т.Н., Пугачев Е. Объектно-ориентированное программирование: учебник. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2007

28. Олифер В.Г., Олифер, Н.А. Компьютерные сети: Принципы, технологии, протоколы. Учебное пособие. – 4-е изд. – СПб.: Питер, 2014 – 944 с.

**Приложение:**

Код реализующий компонент вывода пиццы в компонент Main

function PizzaBlock({

  id,

  name,

  imageUrl,

  price,

  types,

  sizes,

  isLoading,

  onClickAddPizza,

  addedCount,

}) {

  const availableTypes = ["тонкое", "традиционное"];

  const availableSizes = [26, 30, 40];

  const [activeType, setActiveType] = React.useState(types[0]);

  const [activeSize, setActiveSize] = React.useState(0);

  if (isLoading) {

    return <LoadingBlock />;

  }

  const onSelectType = (index) => {

    setActiveType(index);

  };

  const onSelectSize = (index) => {

    setActiveSize(index);

  };

  const onAddPizza = () => {

    const obj = {

      id,

      name,

      imageUrl,

      price,

      size: availableSizes[activeSize],

      type: availableTypes[activeType],

    };

    onClickAddPizza(obj);

  };

  return (

    <div className="pizza-block">

      <img className="pizza-block\_\_image" src={imageUrl} alt="Pizza" />

      <h4 className="pizza-block\_\_title">{name}</h4>

      <div className="pizza-block\_\_selector">

        <ul>

          {availableTypes.map((type, index) => (

            <li

              key={type}

              onClick={() => onSelectType(index)}

              className={classNames({

                active: activeType === index,

                disabled: !types.includes(index),

              })}

            >

              {type}

            </li>

          ))}

        </ul>

        <ul>

          {availableSizes.map((size, index) => (

            <li

              key={size}

              onClick={() => onSelectSize(index)}

              className={classNames({

                active: activeSize === index,

                disabled: !sizes.includes(size),

              })}

            >

              {size} см.

            </li>

          ))}

        </ul>

      </div>

      <div className="pizza-block\_\_bottom">

        <div className="pizza-block\_\_price">{price} ₽</div>

        <Button

          onClick={onAddPizza}

          className="button button--outline button--add"

        >

          <svg

            width="12"

            height="12"

            viewBox="0 0 12 12"

            fill="none"

            xmlns="http://www.w3.org/2000/svg"

          >

            <path

              d="M10.8 4.8H7.2V1.2C7.2 0.5373 6.6627 0 6 0C5.3373 0 4.8 0.5373 4.8 1.2V4.8H1.2C0.5373 4.8 0 5.3373 0 6C0 6.6627 0.5373 7.2 1.2 7.2H4.8V10.8C4.8 11.4627 5.3373 12 6 12C6.6627 12 7.2 11.4627 7.2 10.8V7.2H10.8C11.4627 7.2 12 6.6627 12 6C12 5.3373 11.4627 4.8 10.8 4.8Z"

              fill="white"

            />

          </svg>

          <span>Добавить</span>

          {addedCount && <i>{addedCount}</i>}

        </Button>

      </div>

    </div>

  );

}

export default PizzaBlock;

Блок кода реализующий имитацию загрузки

function LoadingBlock() {

  return (

    <ContentLoader

      className="pizza-block"

      speed={2}

      width={280}

      height={460}

      viewBox="0 0 280 460"

      backgroundColor="#f3f3f3"

      foregroundColor="#ecebeb"

    >

      <circle cx="132" cy="142" r="115" />

      <rect x="0" y="273" rx="6" ry="6" width="275" height="26" />

      <rect x="0" y="310" rx="6" ry="6" width="275" height="84" />

      <rect x="0" y="418" rx="6" ry="6" width="91" height="31" />

      <rect x="137" y="408" rx="25" ry="25" width="135" height="46" />

    </ContentLoader>

  );

}

export default LoadingBlock;

Redux методы:

export const addPizzaToCart = (pizzaObj) => ({

  type: "ADD\_PIZZA\_CART",

  payload: pizzaObj,

});

export const clearCart = () => ({

  type: "CLEAR\_CART",

});

export const removeCartItem = (id) => ({

  type: "REMOVE\_CART\_ITEM",

  payload: id,

});

export const plusCartItem = (id) => ({

  type: "PLUS\_CART\_ITEM",

  payload: id,

});

export const minusCartItem = (id) => ({

  type: "MINUS\_CART\_ITEM",

  payload: id,

});

export const setSortBy = ({ type, order }) => ({

  type: "SET\_SORT\_BY",

  payload: { type, order },

});

export const setCategory = (catIndex) => ({

  type: "SET\_CATEGORY",

  payload: catIndex,

});

Получение пиццы методом get:

export const setLoaded = (payload) => ({

  type: "SET\_LOADED",

  payload,

});

export const fetchPizzas = (sortBy, category) => (dispatch) => {

  dispatch({

    type: "SET\_LOADED",

    payload: false,

  });

  axios

    .get(

      `/pizzas?${category !== null ? `category=${category}` : ""}&\_sort=${

        sortBy.type

      }&\_order=${sortBy.order}`

    )

    .then(({ data }) => {

      dispatch(setPizzas(data));

    });

};

export const setPizzas = (items) => ({

  type: "SET\_PIZZAS",

  payload: items,

});

# **ПРИЛОЖЕНИЕ А**