Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

|  |  |
| --- | --- |
| Допускаю к защите |  |
| Руководитель | подпись |
|  | Л.С. Вахрушева |
|  | И.О. Фамилия |

|  |
| --- |
| Разработка личной информационной системы |
| для домашнего кондитера |

наименование темы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

|  |  |  |
| --- | --- | --- |
| Технологии разработки программных комплексов | | |
|  | 1.008.00.00 - ПЗ |  |

обозначение документа

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | АСУб-21-1 |  |  |  | Д.А. Екимов |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Нормоконтроль |  |  |  |  |  | Л.С. Вахрушева |
|  |  |  |  | подпись |  | И.О. Фамилия |

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Иркутск 2024 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

ЗАДАНИЕ

НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| По курсу | Технологии разработки программных комплексов | | | |
| Студенту | Екимову Д.А. | | | |
|  | (фамилия, инициалы) | | | |
| Тема проекта | | Разработка личной информационной системы для | | |
| домашнего кондитера | | | | |
| Исходные данные: | | |  | |
| Оптимизация процессов управления заказами, рецептурами, закупками и финансами в деятельности домашнего кондитера для повышения точности, эффективности и надежности бизнес-процессов в условиях высокой конкуренции и изменяющихся потребностей клиентов. (вариант №8) | | | | |
|  | | | | |
| |  |  |  | | --- | --- | --- | | Рекомендуемая литература: |  | | | 1. Гутгарц Р.Д Проектирование автоматизированных систем обработки информации и управления: учебное пособие для академического бакалавриата. – М.: Издательство Юрайт, 2019. | | | 1. Проектирование АСОИУ [Электронный ресурс] : методические указания по выполнению курсового проекта: 09.03.01 «Информатика и вычислительная техника» для бакалавров по специальности «Автоматизированные системы обработки информации и управления» / Иркут. нац. исслед. техн. ун-т ; сост. Р. Д. Гутгарц. - Электрон. дан. - Иркутск : ИРНИТУ, 2018. | | | 1. Проектирование информационных систем : учебник и практикум для академического бакалавриата / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук ; под общ. ред. Д. В. Чистова. – М. : Издательство Юрайт, 2016. | | | 1. Рудинский И.Д. Технология проектирования автоматизированных систем обработки информации и управления : учебное пособие для вузов / И. Д. Рудинский. - Москва : Горячая линия - Телеком, 2011. | | | | | |  |

Графическая часть на \_\_\_\_\_\_\_\_\_\_\_\_\_\_ листах.

Дата выдачи задания « 20 » сентября 2024 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Задание получил |  |  | Д.А. Екимов |
|  | подпись |  | И.О. Фамилия |

Дата представления проекта руководителю 2024 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель курсового проектирования |  |  | Л.С.Вахрушева |

**Содержание**

[Введение 5](#_Toc184417178)

[1. Анализ 6](#_Toc184417179)

[1.1. Развитие кондитерского дела 6](#_Toc184417180)

[1.2. Проблемы и вызовы домашнего кондитера 6](#_Toc184417181)

[1.3. Преимущества автоматизации 6](#_Toc184417182)

[1.4. Цель проекта 7](#_Toc184417183)

[1.5. Задачи для достижения цели 7](#_Toc184417184)

[1.6. Обзор существующих программных средств 8](#_Toc184417185)

[1.6.1. Kitchen Assistant 8](#_Toc184417186)

[1.6.2. MyCakes 8](#_Toc184417187)

[1.6.3. Cake Boss 9](#_Toc184417188)

[1.6.4. ChefTap 9](#_Toc184417189)

[1.6.5. Perfect Bake Pro 9](#_Toc184417190)

[1.7. Анализ существующих решений 10](#_Toc184417191)

[1.8. Вывод 10](#_Toc184417192)

[1.9. Процесс AS IS vs TO BE 11](#_Toc184417193)

[1.9.1. AS IS — Текущее состояние 11](#_Toc184417194)

[1.9.2. Диаграмма IDEF0 процесса AS IS 12](#_Toc184417195)

[1.9.3. TO BE — Будущее состояние 13](#_Toc184417196)

[1.9.4. Диаграмма IDEF0 процесса TO BE 14](#_Toc184417197)

[1.10. Описание вариантов использования 15](#_Toc184417198)

[1.10.1. Основные пользователи системы 15](#_Toc184417199)

[1.10.2. Основные варианты использования 15](#_Toc184417200)

[1.11. Выработка требований и постановка задачи 16](#_Toc184417201)

[1.11.1. Функциональные требования 16](#_Toc184417202)

[1.11.2 Нефункциональные требования 17](#_Toc184417203)

[1.11.3 Постановка задачи 18](#_Toc184417204)

[2. Проектирование 20](#_Toc184417205)

[2.1. Выбор и обоснование средств проектирования и реализации 20](#_Toc184417206)

[2.1.1. Средства проектирования 20](#_Toc184417207)

[2.1.2. Средства реализации 20](#_Toc184417208)

[2.2. Проектирование архитектуры приложения 21](#_Toc184417209)

[2.2.1. Схема архитектуры приложения 21](#_Toc184417210)

[2.2.2. Описание компонентов архитектуры 21](#_Toc184417211)

[2.2.3. Обоснование выбора архитектуры 22](#_Toc184417212)

[2.3. Проектирование хранилища данных 23](#_Toc184417213)

[2.3.1. Основные сущности 23](#_Toc184417214)

[2.3.2. Логическая модель данных 23](#_Toc184417215)

[2.4. Архитектура БД 24](#_Toc184417216)

[2.5. Проектирование пользовательского интерфейса (UI) 25](#_Toc184417217)

[2.5.1. Верхнеуровневое распределение экранов 25](#_Toc184417218)

[2.6. Таблица экранов и их описаний 26](#_Toc184417219)

[2.7. Отрисовка и описание макетов 29](#_Toc184417220)

[2.7.1. Формирование заказа 29](#_Toc184417221)

[2.7.2. Список ингредиентов 30](#_Toc184417222)

[2.7.3. Анализ продаж 32](#_Toc184417223)

[2.8. Карта экранов и состояний 34](#_Toc184417224)

[3. Разработка 35](#_Toc184417225)

[3.1. Математическое обеспечение 35](#_Toc184417226)

[3.2. Экраны приложения 36](#_Toc184417227)

[Заключение 43](#_Toc184417228)

[Список использованных источников 44](#_Toc184417229)

[Приложения 45](#_Toc184417230)

[Приложение А 45](#_Toc184417231)

[Приложение Б 49](#_Toc184417232)

# Введение

Современное развитие информационных технологий оказывает значительное влияние на все сферы человеческой деятельности, включая кулинарию и малый бизнес. Домашние кондитеры, которые совмещают процесс создания десертов с управлением заказами и финансовыми расчетами, сталкиваются с необходимостью оптимизации своих рабочих процессов. В условиях высокой конкуренции и растущих требований клиентов важно не только создавать качественную продукцию, но и эффективно управлять своим бизнесом.

Проблемы, с которыми сталкиваются домашние кондитеры, включают сложности в учете заказов, планировании закупок, ведении рецептов, расчетах себестоимости продукции и многом другом. Большинство из них пользуются такими инструментами, как блокноты, таблицы или мобильные приложения с ограниченным функционалом. Это приводит к дополнительным временным затратам, ошибкам и, как следствие, снижению доходности.

Цифровые решения, такие как информационные системы, способны автоматизировать необходимые задачи и предоставить пользователю комплексные инструменты для управления всеми аспектами бизнеса. Однако существующие программные продукты часто либо избыточны по функциональности для небольшого домашнего производства, либо не обеспечивают достаточного функционала.

Поэтому разработка специализированной информационной системы для домашних кондитеров является актуальной задачей. Такая система должна учитывать специфику их деятельности, предоставляя удобные инструменты для работы с заказами, рецептами, закупками и финансами. Она позволит минимизировать временные затраты на рутинные операции, сократить количество ошибок и обеспечить доступ к аналитическим данным для принятия взвешенных решений.

# Анализ

## Развитие кондитерского дела

Кондитерское дело является одной из самых сложных и творческих кулинарных дисциплин, требующих не только мастерства, но и внимательного учета множества факторов, начиная от правильного выбора ингредиентов и их соотношений, заканчивая температурными режимами, временем выпекания и особенностями хранения готовой продукции. Домашние кондитеры, несмотря на работу в условиях ограниченного пространства и ресурсов, стремятся к совершенству в своей деятельности, предлагая широкий ассортимент кондитерских изделий, таких как торты, пирожные, макароны, моти и многое другое.

Значительное число домашних кондитеров начинают свою карьеру как хобби, которое со временем превращается в бизнес. Этот процесс требует серьезного подхода к организации рабочего процесса, контроля над качеством продукции, а также грамотного взаимодействия с клиентами. Становится очевидным, что для успешного ведения такой деятельности необходимы специализированные инструменты, которые позволили бы автоматизировать и упрощать многие рутинные задачи.

## Проблемы и вызовы домашнего кондитера

Одной из основных проблем, с которой сталкиваются домашние кондитеры, является сложность управления процессом производства. Например, планирование закупок ингредиентов часто требует точного расчета необходимого количества продуктов для выполнения определенного числа заказов. Недостаток точных данных может привести либо к лишним расходам, либо к нехватке ресурсов в самый неподходящий момент. Это касается также учета остатков продуктов на складе и планирования сроков их использования.

Помимо этого, кондитеру необходимо иметь четкую систему управления своими рецептами. В процессе работы часто требуется быстро находить определенные рецепты, вносить изменения в пропорции ингредиентов в зависимости от размера порций или индивидуальных предпочтений клиентов. Для этого требуется гибкая и удобная система управления рецептами, которая была бы доступна в любое время и с любого устройства.

Финансовая составляющая также является немаловажным аспектом. Для домашнего кондитера необходимо отслеживать свои доходы и расходы, учитывать себестоимость продукции, чтобы грамотно планировать бизнес и оставаться конкурентоспособным. Это особенно актуально в условиях высокой конкуренции на рынке кондитерских изделий, когда необходимо не только предлагать качественный продукт, но и соблюдать приемлемый уровень цен.

## Преимущества автоматизации

Автоматизация работы кондитера приносит множество преимуществ. В первую очередь, это экономия времени, которое раньше тратилось на выполнение рутинных задач. Например, автоматический расчет необходимого количества ингредиентов для конкретного заказа позволяет избежать ошибок и сократить время на подготовку. Инвентаризация складских запасов, будучи автоматизированной, снижает риск забыть о необходимости пополнения продуктов.

Кроме того, система может напоминать кондитеру о скором истечении срока годности ингредиентов, что позволяет избежать использования некачественных продуктов и снижает риск потерь. Интеграция с бухгалтерскими системами помогает в точном расчете себестоимости продукции и прогнозировании доходов и расходов, что важно для ведения успешного бизнеса.

Также такие системы облегчают взаимодействие с клиентами. Кондитер может хранить в системе данные о заказах, предпочтениях клиентов, создавать шаблоны для повторных заказов и автоматически генерировать счета, что улучшает качество обслуживания.

## Цель проекта

Целью данного проекта является удовлетворение потребности домашних кондитеров в удобном и эффективном управлении всеми аспектами их деятельности. Проект направлен на автоматизацию ключевых процессов, включая планирование закупок, работу с рецептами, управление заказами и финансами, с целью сокращения временных затрат на рутинные операции и минимизации ошибок. Создание личной информационной системы выступает средством для достижения этой цели, обеспечивая интуитивно понятный и удобный инструмент, который позволит кондитерам сосредоточиться на творчестве и повышении качества своей продукции.

## Задачи для достижения цели

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Провести исследование потребностей домашних кондитеров и разработать функциональные требования к системе.
2. Разработать модуль для учета и планирования закупок ингредиентов с возможностью автоматического обновления информации по остаткам на складе.
3. Создать удобный интерфейс для хранения и управления рецептами, позволяющий быстро вносить изменения и адаптировать рецепты под индивидуальные требования.
4. Внедрить модуль для управления заказами, учета клиентских предпочтений и автоматической генерации счетов.
5. Реализовать систему учета финансов для анализа доходов, расходов и расчета себестоимости продукции.
6. Обеспечить интеграцию с календарями и другими приложениями для упрощения планирования рабочего времени.
7. Обеспечить безопасность данных пользователей и возможность их резервного копирования.

В результате выполнения проекта будет создано эффективное и удобное программное решение, которое упростит работу домашних кондитеров и повысит их продуктивность.

## Обзор существующих программных средств

Развитие информационных технологий привело к появлению множества программных решений, предназначенных для автоматизации деятельности в различных областях, включая домашнее кондитерское производство. На рынке уже существуют различные системы и приложения, которые помогают кондитерам управлять своими бизнес-процессами, отслеживать запасы, вести учет рецептов и заказывать ингредиенты. Ниже представлен обзор наиболее популярных программных решений, которые могут быть использованы для управления кондитерской деятельностью на домашнем уровне.

### ****Kitchen Assistant****

Одним из самых известных решений для управления кулинарными процессами является приложение **Kitchen Assistant**. Оно предоставляет пользователям следующие возможности:

* учет ингредиентов и их количества на складе;
* планирование меню на неделю или месяц вперед;
* хранение рецептов с возможностью изменения их параметров в зависимости от нужного объема продукции;
* автоматический расчет калорийности и пищевой ценности блюд;
* интеграция с онлайн-магазинами для заказа необходимых

ингредиентов.

Приложение подходит для домашнего использования, но ориентировано больше на планирование рационов и личное питание, нежели на массовое производство продукции для заказчиков.

### ****MyCakes****

**MyCakes** — специализированное приложение для домашних кондитеров, которое включает в себя расширенный функционал для управления заказами и учета расходов:

* ведение клиентской базы с сохранением истории заказов и

предпочтений;

* возможность создания счета для клиента и отслеживания платежей;
* интеграция с календарем для планирования заказов и учета времени на выполнение работ;
* автоматический расчет себестоимости продукции на основе введенных рецептов и стоимости ингредиентов;
* возможность добавления фотографий готовых изделий и их интеграция в каталог для демонстрации клиентам;

Приложение удобно для работы с клиентами и заказами, однако не обладает расширенным функционалом для управления запасами и не поддерживает сложные рецептурные операции.

### ****Cake Boss****

Программа **Cake Boss** также ориентирована на управление небольшим кондитерским бизнесом. Основные функции включают:

* учет финансов и возможность отслеживать прибыль и расходы;
* управление заказами и сохранение информации о клиентах;
* ведение инвентаризации и учет количества ингредиентов на складе;
* автоматический расчет стоимости продукции с учетом расхода

ингредиентов и времени работы;

* встроенные шаблоны договоров и коммерческих предложений для клиентов.

Программа удобна для управления бизнесом в целом, однако она имеет ограниченный функционал для ведения рецептур и не предлагает гибких настроек для работы с различными типами продукции.

### ****ChefTap****

Приложение **ChefTap** представляет собой кулинарный органайзер с возможностями для хранения и управления рецептами. Оно предоставляет следующие функции:

* хранение рецептов и возможность добавления собственных;
* поддержка работы с несколькими устройствами и синхронизация

данных;

* автоматическое создание списка покупок на основе выбранных

рецептов;

* возможность изменения количества порций и автоматический

пересчет ингредиентов;

* интеграция с веб-сайтами для загрузки рецептов и рекомендаций.

ChefTap удобен для управления рецептами и планирования закупок, однако не предоставляет функционала для работы с клиентами, ведения заказов или учета финансов, что ограничивает его использование в кондитерском бизнесе.

### ****Perfect Bake Pro****

**Perfect Bake Pro** — это умная система для приготовления кондитерских изделий, которая включает в себя кухонные весы и приложение. Основные функции:

* подключение к весам для точного расчета количества ингредиентов;
* предоставление рецептов с точными пропорциями и рекомендациями по времени выпечки;
* автоматическая корректировка рецептов на основе имеющихся

ингредиентов;

* ведение учета калорийности и пищевой ценности продуктов;
* возможность сохранения любимых рецептов и создания новых.

Несмотря на удобство в работе с рецептами и весами, система не предлагает инструментов для управления бизнесом, заказами или клиентской базой, что делает ее скорее инструментом для любителей, чем для профессиональных кондитеров.

Для удобства сравнения построим таблицу:

Таблица 1.1 – Сравнительная таблица существующих программных средств

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название программы | Учёт ингредиентов | Управление заказами | Учёт клиентов | Финансовый учёт | Рецептурная база |
| Kitchen Assistant | Да | Нет | Нет | Нет | Да |
| MyCakes | Да | Да | Да | Да | Частично |
| Cake Boss | Да | Да | Да | Да | Частично |
| ChefTap | Да | Нет | Нет | Нет | Да |
| Perfect Bake Pro | Частично | Нет | Нет | Нет | Да |

## Анализ существующих решений

Анализ существующих программных средств показывает, что на рынке уже присутствуют разнообразные инструменты для помощи домашним кондитерам, однако каждое из них решает лишь определенную часть задач. Например, **MyCakes** и **Cake Boss** предлагают хорошие возможности для ведения клиентской базы и управления заказами, но не всегда могут предложить продвинутую функциональность для работы с рецептами и финансовыми операциями. **Kitchen Assistant** и **ChefTap** удобны для управления рецептурными процессами, но не позволяют полноценно вести бизнес, отслеживать заказы и работать с клиентами.

## Вывод

На рынке существует несколько программных решений, которые частично закрывают потребности домашних кондитеров, но не предлагают комплексного подхода к управлению всеми аспектами деятельности. Основные недостатки существующих систем включают недостаточную интеграцию между рецептами, заказами и финансами, а также слабую поддержку инвентаризации и складского учета. Проект «Личная информационная система домашнего кондитера» предлагает решить эти проблемы, предоставив пользователям универсальную платформу, которая объединит управление рецептами, заказами, клиентами и финансами в одной системе. Это позволит домашним кондитерам работать более эффективно и сосредоточиться на творческой стороне своего бизнеса.

## Процесс AS IS vs TO BE

Для анализа текущего состояния процессов (AS IS) и их будущей оптимизации (TO BE) в рамках проекта «Личная информационная система домашнего кондитера» необходимо визуализировать существующие и планируемые бизнес-процессы. Для этого будет использована методология IDEF0 для построения функциональной структуры системы. Также можно применить BPMN для более детального описания шагов процессов.

### AS IS — Текущее состояние

На данный момент большинство домашних кондитеров выполняют процессы вручную или с использованием неинтегрированных инструментов, таких как Excel, бумажные записи или мобильные приложения для индивидуальных задач (например, калькуляция расходов, хранение рецептов). Эти процессы не автоматизированы и часто приводят к увеличению трудозатрат и ошибок в расчетах и планировании.

Основные процессы системы в текущем состоянии:

1. **Получение заказа:**

Заказ поступает от клиента через социальные сети или телефон. Домашний кондитер вручную записывает детали заказа (количество, дату, предпочтения клиента).

1. **Планирование закупок:**

На основе рецептов и количества заказов вручную рассчитываются необходимые ингредиенты. Контроль запасов также ведется вручную — кондитер должен сам проверять наличие и срок годности ингредиентов.

1. **Ведение рецептов:**

Рецепты могут храниться в блокнотах, текстовых файлах или специализированных приложениях, не связанных с другими аспектами бизнеса.

1. **Учет финансов:**

Финансовый учет часто ведется разрозненно, через Excel или вручную. Это затрудняет анализ доходов, расходов и расчет себестоимости продукции.

1. **Выполнение заказа:**

Процесс приготовления продукции осуществляется вручную, планирование времени приготовления основано на опыте.

1. **Доставка и оплата:**

Доставка часто осуществляется без четкой системы планирования, а оплата может производиться вручную при передаче заказа или через онлайн-переводы без автоматического учета.

### Диаграмма IDEF0 процесса AS IS

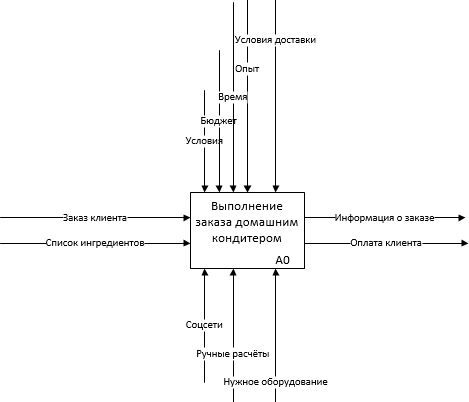


Рисунок 1.1 – Контекстная диаграмма процесса AS IS

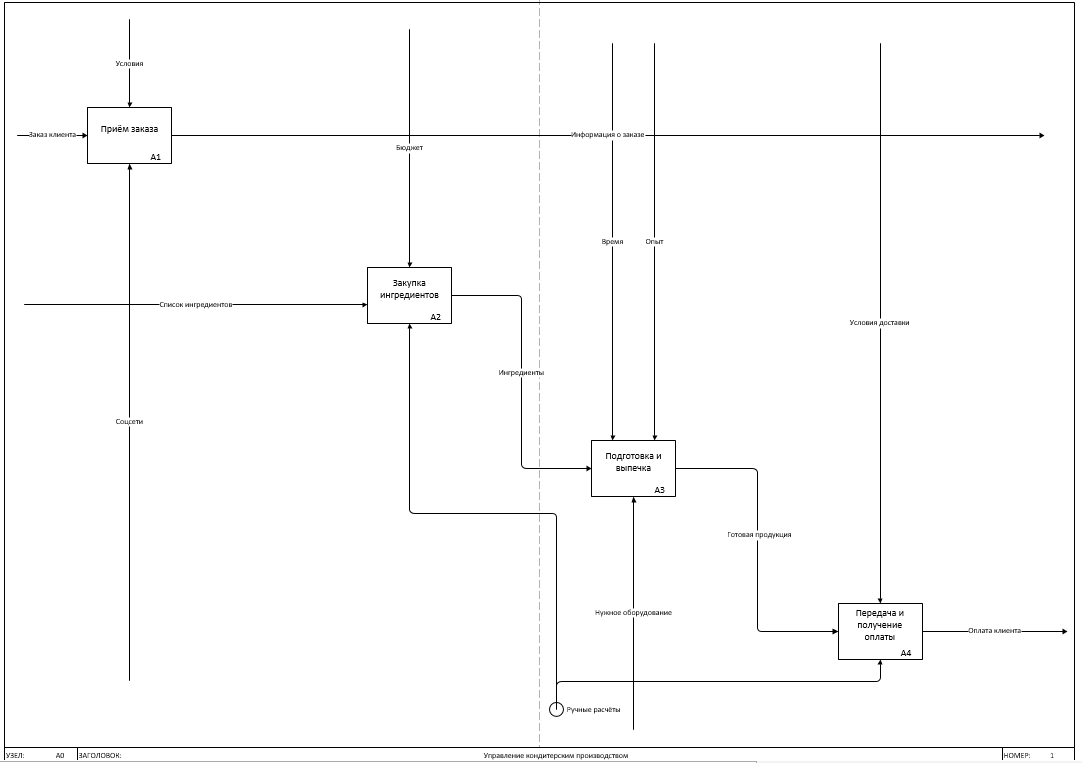


Рисунок 1.2 – Расширенная диаграмма AS IS

### TO BE — Будущее состояние

В будущей системе (TO BE) все процессы будут автоматизированы и интегрированы в единую информационную систему. Это позволит сократить время на выполнение рутинных операций, улучшить точность расчетов и упростить взаимодействие с клиентами.

Основные процессы системы в будущем состоянии:

1. **Автоматизированное получение заказа:**

Заказы поступают через систему (сайт, приложение), где клиенты могут выбрать продукцию, указать предпочтения и оплатить заказ. Все данные сохраняются автоматически.

1. **Интегрированное планирование закупок и инвентаризация:**

Система автоматически рассчитывает количество необходимых ингредиентов на основе введенных рецептов и текущих запасов. Она также напоминает о необходимости пополнения запасов и отслеживает срок годности ингредиентов.

1. **Централизованное управление рецептами:**

Рецепты хранятся в базе данных, и система автоматически подстраивает их под нужный объем заказов. Также система позволяет сохранять изменения и добавлять новые рецепты.

1. **Автоматизация учета финансов:**

Система автоматически ведет учет всех финансовых операций, рассчитывает себестоимость каждого изделия, генерирует отчеты по доходам и расходам.

1. **Оптимизация выполнения заказа:**

Планирование времени приготовления оптимизируется системой на основе количества заказов, доступных ресурсов и времени выполнения.

1. **Интеграция с сервисами доставки и оплаты:**

Система автоматически генерирует и отправляет счета клиентам, ведет учет оплат, интегрируется с курьерскими службами для отслеживания доставки.

### Диаграмма IDEF0 процесса TO BE

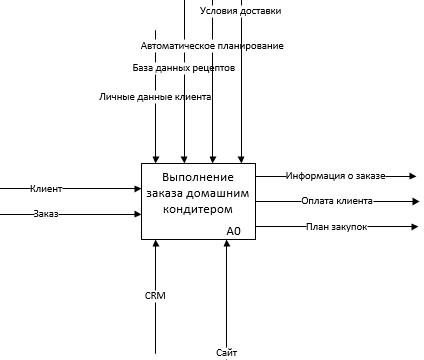


Рисунок 1.3 – Контекстная диаграмма процесса TO BE

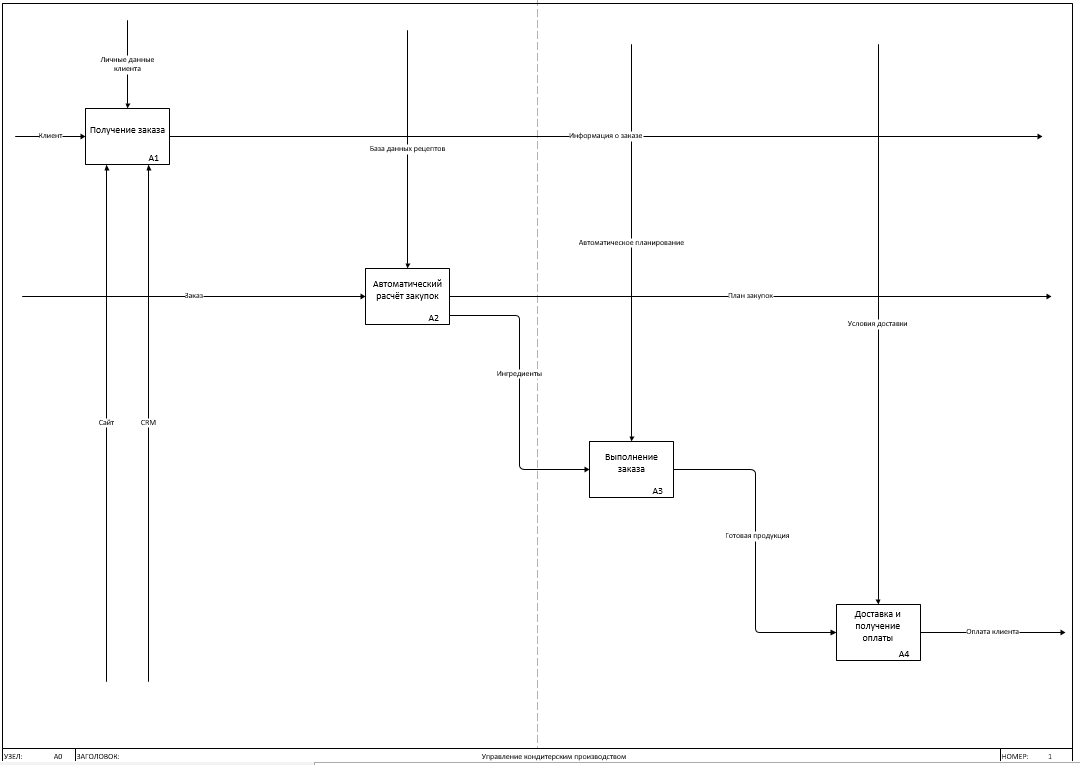


Рисунок 1.4 – Расширенная диаграмма TO BE

## Описание вариантов использования

Для описания функциональных требований к системе «Личная информационная система домашнего кондитера» с точки зрения пользователя необходимо построить диаграмму вариантов использования (Use Case Diagram), которая демонстрирует взаимодействие пользователей с системой и описывает основные функции, доступные каждому типу пользователя.

### Основные пользователи системы

1. **Кондитер (основной пользователь)** — главный пользователь системы, который занимается управлением бизнесом, ведет заказы, рецепты, контролирует финансовые операции и закупки ингредиентов.
2. **Клиент** — конечный пользователь, который оформляет заказы через систему, взаимодействует с кондитером и получает продукцию.

### Основные варианты использования

1. **Для кондитера:**
   1. **Управление заказами** — просмотр, создание и редактирование

заказов.

* 1. **Управление клиентами** — ведение базы данных клиентов, их предпочтений и истории заказов.
  2. **Управление рецептами** — создание, редактирование и хранение рецептов.
  3. **Планирование закупок** — автоматический расчет необходимых ингредиентов и учет текущих запасов.
  4. **Финансовый учет** — ведение учета расходов, доходов и себестоимости продукции.
  5. **Управление доставкой** — организация и планирование доставки готовой продукции клиентам.

1. **Для клиента:**
   1. **Оформление заказа** — возможность выбрать продукт, указать предпочтения и оформить заказ через систему.
   2. **Отслеживание заказа** — возможность отслеживать статус выполнения заказа (в процессе, готов, доставлен).
   3. **Оплата заказа** — выбор способа оплаты и её проведение через

интегрированные платежные сервисы.

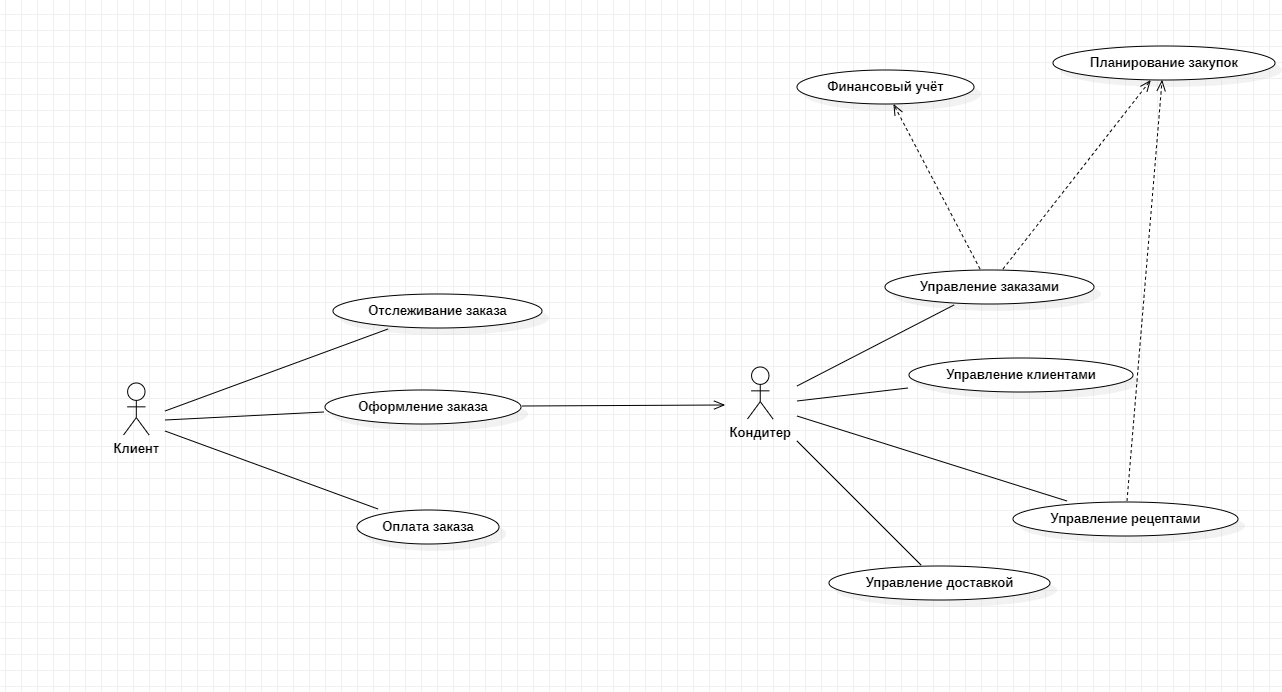


Рисунок 1.5 – Диаграмма прецедентов

## Выработка требований и постановка задачи

На данном этапе проекта проводится детальная разработка требований к системе «Личная информационная система домашнего кондитера». Эти требования формируют основу для дальнейшего проектирования системы и описывают её функциональные и нефункциональные аспекты.

### Функциональные требования

Функциональные требования описывают основные действия, которые система должна выполнять для удовлетворения потребностей пользователей. Эти требования охватывают различные модули системы и взаимодействие между ними.

1. **Управление заказами:**
   * система должна позволять пользователю (кондитеру) создавать новые заказы, редактировать существующие и отслеживать их статус;
   * необходимо обеспечить возможность ввода данных о клиенте, включая контактную информацию, предпочтения и детали заказа (дата выполнения, количество, тип продукции);
   * система должна автоматически уведомлять кондитера о новых поступивших заказах и изменениях статуса текущих заказов.
2. **Управление клиентами:**
   * система должна хранить информацию о клиентах, включая их имя, контактные данные и историю заказов;
   * система должна предоставлять возможность просматривать и редактировать данные клиентов;
   * необходимо реализовать поиск и фильтрацию клиентов по различным критериям (например, по имени или количеству заказов).
3. **Управление рецептами:**
   * система должна позволять кондитеру сохранять рецепты с возможностью редактирования ингредиентов и методов приготовления;
   * рецепты должны автоматически подстраиваться под количество заказов, с учетом пропорций ингредиентов;
   * система должна сохранять возможность создания новых рецептов и их добавления в базу данных.
4. **Планирование закупок:**
   * система должна автоматически рассчитывать количество необходимых ингредиентов на основе текущих заказов и доступных рецептов;
   * необходимо предусмотреть возможность контроля наличия ингредиентов на складе и управления запасами (учет сроков годности, количество);
   * система должна отправлять уведомления о необходимости закупок при достижении минимального уровня запасов.
5. **Финансовый учет:**
   * система должна вести учет расходов и доходов, а также рассчитывать себестоимость продукции на основе ингредиентов и других затрат;
   * необходимо реализовать возможность формирования отчетов о доходах, расходах и прибыли за выбранные периоды времени;
   * система должна позволять управлять платежами и интегрироваться с онлайн-сервисами для оплаты.
6. **Управление доставкой:**
   * система должна позволять кондитеру планировать доставку готовой продукции, управлять курьерами и отслеживать доставленные заказы;
   * необходимо реализовать возможность выбора службы доставки и интеграцию с внешними логистическими сервисами;
   * клиенты должны иметь возможность отслеживать статус доставки через личный кабинет или уведомления.
7. **Интеграция с клиентом (Личный кабинет клиента):**
   * клиенты должны иметь доступ к системе через личный кабинет, где они смогут просматривать свои заказы, отслеживать их статус и оплачивать;
   * система должна предоставлять клиентам возможность выбирать продукцию, указывать свои предпочтения и оформлять заказы;
   * система должна поддерживать отправку уведомлений клиентам о статусе заказа и возможности получения продукции.

### Нефункциональные требования

Нефункциональные требования описывают качество работы системы, её производительность, безопасность и удобство использования.

1. **Удобство использования:**
   * система должна обладать простым и интуитивно понятным интерфейсом для пользователей с различными уровнями компьютерной грамотности (как для кондитера, так и для клиентов);
   * важна возможность быстрого доступа к ключевым функциям системы без лишних шагов.
2. **Производительность:**
   * система должна обрабатывать запросы пользователей в режиме реального времени. Время отклика не должно превышать 2 секунд для основных операций (создание заказа, расчет закупок);
   * система должна справляться с нагрузкой до 1000 активных пользователей одновременно без снижения производительности.
3. **Безопасность:**
   * система должна обеспечивать защиту данных клиентов и кондитера, включая личную информацию, платежные данные и историю заказов;
   * необходимо реализовать защиту данных при передаче по сети через SSL/TLS шифрование;
   * доступ к различным модулям системы должен быть ограничен в зависимости от роли пользователя (администратор, кондитер, клиент).
4. **Масштабируемость:**
   * система должна поддерживать возможность расширения функционала и увеличение количества пользователей без необходимости полного перепроектирования;
   * инфраструктура должна быть адаптируема для увеличения объема данных и нагрузки на серверную часть.
5. **Надежность:**
   * система должна работать стабильно без простоев. В случае сбоев необходимо предусмотреть механизмы аварийного восстановления;
   * важна возможность регулярного резервного копирования данных для предотвращения потерь информации.
6. **Совместимость:**
   * система должна поддерживать работу на различных устройствах (настольные ПК, планшеты, мобильные телефоны);
   * важно обеспечить совместимость с основными операционными системами (Windows, macOS, Android, iOS).

### Постановка задачи

Цель данного проекта заключается в создании **информационной системы для домашнего кондитера**, которая автоматизирует и оптимизирует процессы управления заказами, рецептами, закупками, финансовым учетом и доставкой. Система должна обеспечивать удобный интерфейс для работы кондитера, а также интеграцию с клиентами через личный кабинет, позволяя им оформлять заказы и отслеживать их выполнение.

Основные задачи проекта:

1. Разработка и внедрение системы управления заказами и клиентами.
2. Создание модуля управления рецептами и автоматического расчета ингредиентов.
3. Внедрение системы планирования закупок и управления запасами.
4. Реализация системы финансового учета и интеграция с онлайн-сервисами оплаты.
5. Организация управления доставкой и интеграция с логистическими сервисами.
6. Обеспечение надежной и безопасной работы системы с возможностью расширения функционала в будущем.

# Проектирование

## Выбор и обоснование средств проектирования и реализации

### Средства проектирования

**Инструменты для проектирования интерфейса пользователя (UI/UX)**

Выбор: Figma

Обоснование: Figma является современным инструментом для проектирования пользовательского интерфейса, который предоставляет обширный набор инструментов для создания макетов интерфейсов и имеет возможности прототипирования. Figma, в частности, позволяет вести совместную работу над проектом в реальном времени.

**Средства проектирования структур и процессов**

Выбор: Microsoft Visio

Обоснование: Этот инструмент позволяет разрабатывать диаграммы (например, диаграммы потоков данных, IDEF0, BPMN), которые описывают архитектуру системы, её процессы и связи между компонентами. Он удобен для визуализации сложных процессов и позволяет представлять схемы в наглядной и структурированной форме.

### Средства реализации

**Выбор платформы разработки**

Выбор: Веб-платформа на основе JavaScript / Node.js + React

Обоснование: Использование платформы JavaScript с Node.js на стороне сервера и React на стороне клиента обеспечивает высокую производительность, модульность и кроссплатформенность. Эти технологии поддерживают асинхронную обработку данных, что важно для повышения скорости работы системы. Выбор React для фронтенда обусловлен популярностью, большим сообществом и широким набором готовых компонентов.

**База данных**

Выбор: PostgreSQL

Обоснование: PostgreSQL подходит для реляционных данных и обеспечения надежности транзакций, что важно для финансового учета.

**Средства для серверной логики и API**

Выбор: Express.js (для Node.js)

Обоснование: Express.js — это минималистичный фреймворк, который поддерживает создание RESTful API, обеспечивает высокую скорость разработки и гибкость в настройке маршрутов, а также легко интегрируется с различными базами данных.

**Средства обеспечения безопасности**

Выбор: JWT (JSON Web Token) для аутентификации и авторизации

Обоснование: JWT является популярным стандартом для защиты данных пользователя и обеспечения безопасного доступа к ресурсам системы. Он позволяет реализовать авторизацию через токены и использовать защищенные каналы связи.

## Проектирование архитектуры приложения

Проектирование архитектуры приложения «Личная информационная система домашнего кондитера» заключается в создании структурного подхода к организации компонентов системы и их взаимодействию. Это необходимо для обеспечения удобства разработки, масштабируемости, надежности и производительности. В данном разделе представлена схема архитектуры, описание её компонентов и обоснование выбора архитектурного подхода.

### Схема архитектуры приложения

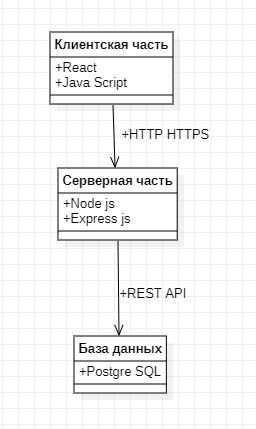


Рисунок 2.1 – Схема архитектуры приложения

### Описание компонентов архитектуры

**Клиентская часть (Frontend-приложение)**

Используемые технологии: React.

Описание:

Клиентская часть отвечает за взаимодействие с пользователем, предоставление удобного интерфейса и динамическое обновление данных. React обеспечивает создание компонентов, поддержку состояния приложения и взаимодействие с сервером через API.

Функции клиента:

* отображение интерфейса для работы с заказами, клиентами, рецептами и другими функциями;
* валидация данных перед их отправкой на сервер;
* управление пользовательскими сессиями (например, с использованием токенов).

**Серверная часть (Backend-приложение)**

Используемые технологии: Node.js с фреймворком Express.js.

Описание:

Серверная часть отвечает за обработку запросов от клиентской части, взаимодействие с базой данных, бизнес-логику и управление API. Express.js обеспечивает легкость в построении маршрутов, обработку HTTP-запросов и интеграцию с различными внешними сервисами.

Основные функции сервера:

* обработка CRUD операций для заказов, клиентов, рецептов и других данных;
* управление сессиями, авторизация и аутентификация пользователей (с помощью JWT);
* взаимодействие с базой данных, выполнение запросов, обеспечение логики бизнес-процессов.

**База данных (PostgreSQL)**

Описание:

Хранилище данных, обеспечивающее доступ к информации о клиентах, заказах, рецептах, финансах и других сущностях.

Роли базы данных:

Сохранение и предоставление доступа к данным с поддержкой транзакционности.

### Обоснование выбора архитектуры

* 1. Архитектура клиент-сервер:

Выбор клиент-серверной архитектуры обусловлен необходимостью обеспечения разделения интерфейса пользователя (Frontend) и бизнес-логики (Backend). Это дает возможность масштабировать клиентскую и серверную части независимо друг от друга, улучшает читаемость и поддержку кода, а также обеспечивает гибкость при добавлении новых функций.

* 1. REST API:

Использование RESTful API позволяет клиенту и серверу обмениваться данными через четко определенные маршруты и методы. Этот подход обеспечивает легкую интеграцию с внешними сервисами и упрощает расширение функциональности системы.

* 1. Модульность:

Выбор React/Vue.js, Node.js и Express.js обусловлен их модульной структурой и обширной экосистемой. Это позволяет разработчикам быстро добавлять новые модули и функции, а также облегчает повторное использование кода.

* 1. Гибкость работы с данными:

Поддержка как реляционных, так и NoSQL баз данных позволяет выбрать оптимальный способ хранения данных в зависимости от их структуры и объема. PostgreSQL обеспечивает строгую структурированность и транзакции, а MongoDB подходит для гибких данных.

* 1. Масштабируемость и производительность:

Node.js и выбранные инструменты обеспечивают высокую производительность за счет асинхронной обработки запросов, что особенно важно для приложения, работающего с большим количеством пользователей и запросов в реальном времени.

## Проектирование хранилища данных

### Основные сущности

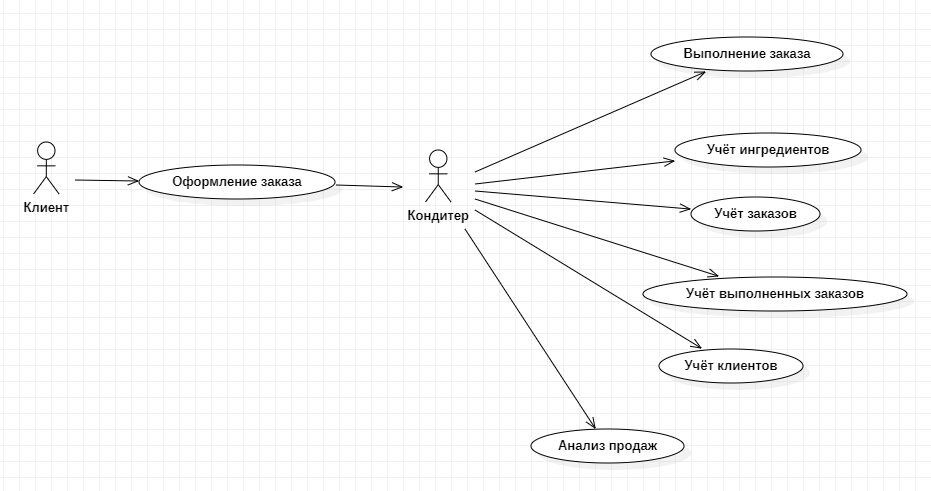


Рисунок 2.2 – Основные сущности

### Логическая модель данных

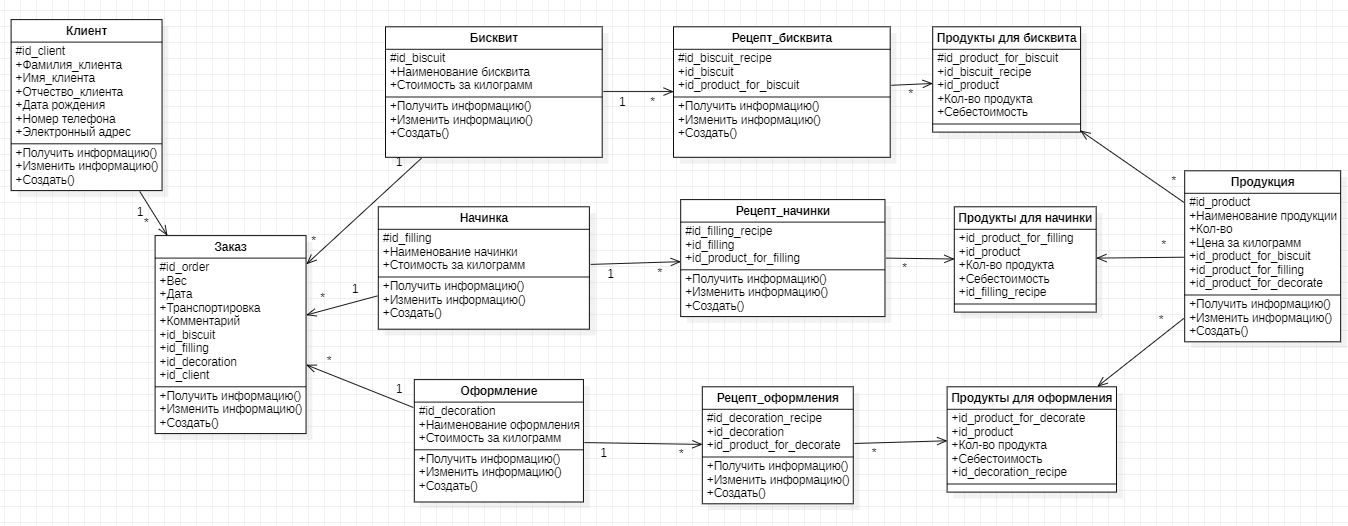


Рисунок 2.3 – Диаграмма классов

## Архитектура БД

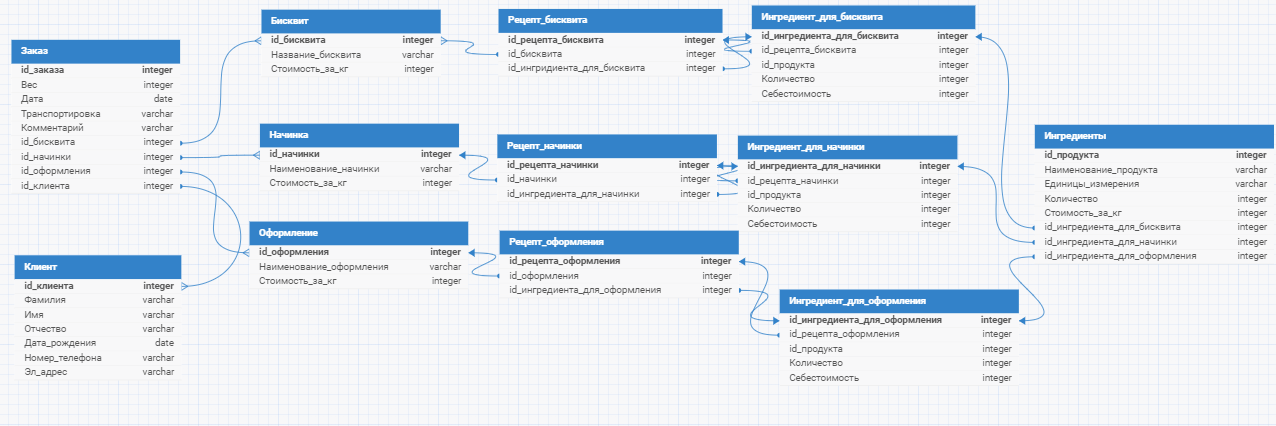


Рисунок 2.4 – Архитектура БД

## Проектирование пользовательского интерфейса (UI)

### Верхнеуровневое распределение экранов

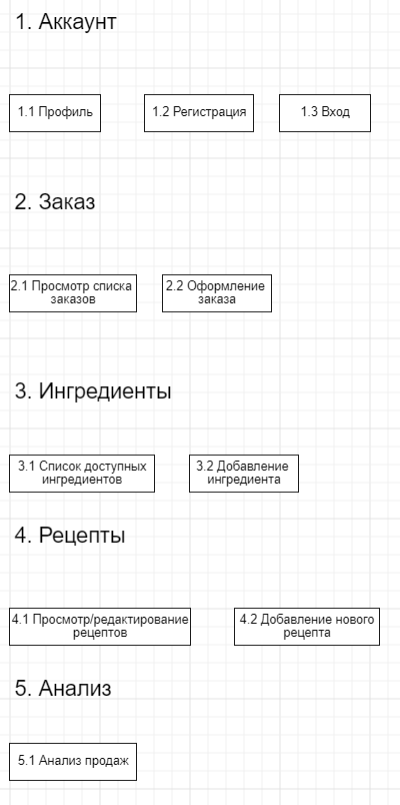


Рисунок 2.5 – Распределение экранов

## Таблица экранов и их описаний

Таблица 2.1 – Таблица экранов и их описаний

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер экрана** | **Название** | **Поля ввода для валидации** | **Описание экрана и его поведения** | **Список возможных состояний** | **Всплывающие уведомления** | **Идентификаторы UI-контролов** | **Используемые стили** |
| **1** | Профиль пользователя | Имя, Email, пароль | Экран профиля пользователя с возможностью редактирования персональных данных, смены пароля и настроек. | Состояние редактирования, состояние просмотра | Успешное сохранение изменений | SaveProfileButton, NameInput, EmailInput | Основной стиль приложения |
| **2** | Вход | Email, пароль | Экран входа в систему. Пользователь вводит свои учетные данные для аутентификации. При успешном входе перенаправляется на главную страницу. При ошибке отображается сообщение об ошибке. | Состояние входа, ошибка входа | Сообщение об ошибке (неверный логин/пароль) | LoginButton, EmailInput, PasswordInput | Основной стиль приложения |
| **3** | Регистрация | Имя, Email, пароль, подтверждение пароля | Экран регистрации нового пользователя. После заполнения и подтверждения данных создается учетная запись. Валидация полей на корректность (например, email-формат, длина пароля). | Состояние регистрации, ошибка регистрации | Успешная регистрация, сообщение об ошибке | SignUpButton, NameInput, EmailInput, PasswordInput | Основной стиль приложения |
| **4** | Список заказов | Поиск по названию заказа | Экран для просмотра всех заказов. Доступен поиск и фильтрация. Выбор заказа перенаправляет на экран деталей. | Состояние загрузки, состояние пустого списка | Сообщение об отсутствии заказов | SearchInput, OrderItem | Основной стиль приложения |

Таблица 2.1 – Продолжение

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер экрана** | **Название** | **Поля ввода для валидации** | **Описание экрана и его поведения** | **Список возможных состояний** | **Всплывающие уведомления** | **Идентификаторы UI-контролов** | **Используемые стили** |
| **5** | Оформление заказа | Комментарий, статус заказа | Экран отображает детальную информацию о конкретном заказе. Пользователь может изменить выбрать бисквит, начинку, декор, добавить комментарии. | Состояние редактирования, состояние просмотра | Успешное оформление заказа | SaveButton, StatusDropdown | Основной стиль приложения |
| **6** | Список ингредиентов | Поиск по названию ингредиента | Экран для просмотра всех ингредиентов. Доступен поиск Выбор ингредиента перенаправляет на страницу изменения ингредиента | Состояние загрузки, состояние пустого списка | Сообщение об отсутствии ингредиентов | SearchInput, OrderItem | Основной стиль приложения |
| **7** | Добавление ингредиента | Наименование ингредиента, ед. изм., цена за ед. изм., кол-во | Экран отображает поля для ввода информации о новом ингредиенте | Состояние редактирования, состояние просмотра | Сообщение о успешном добавлении ингредиента | SaveButton, StatusDropdown | Основной стиль приложения |
| **8** | Рецепты | Поиск по названию рецепта | Экран со списком рецептов, доступных для просмотра и редактирования. Поддержка создания новых рецептов. | Состояние загрузки, состояние пустого списка | Всплывающее сообщение об ошибке | RecipeSearch, RecipeCard | Основной стиль приложения |
| **9** | Добавление/редактирование рецепта | Название, ингредиенты, описание, изображения | Экран создания нового рецепта или редактирования существующего. Валидация полей (например, обязательные поля, проверка формата). | Состояние редактирования, состояние ошибки | Успешное сохранение рецепта | SaveRecipeButton, IngredientInput | Основной стиль приложения |

Таблица 2.1 – Продолжение

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер экрана** | **Название** | **Поля ввода для валидации** | **Описание экрана и его поведения** | **Список возможных состояний** | **Всплывающие уведомления** | **Идентификаторы UI-контролов** | **Используемые стили** |
| **10** | Анализ продаж | Период (даты) | Экран анализа финансовых данных (доходы, расходы, прибыль). Возможен выбор периода и генерация отчетов. | Состояние фильтрации, состояние пустых данных | Всплывающее сообщение об ошибке | DateFilter, GenerateReportButton | Основной стиль приложения |

## Отрисовка и описание макетов

### Формирование заказа

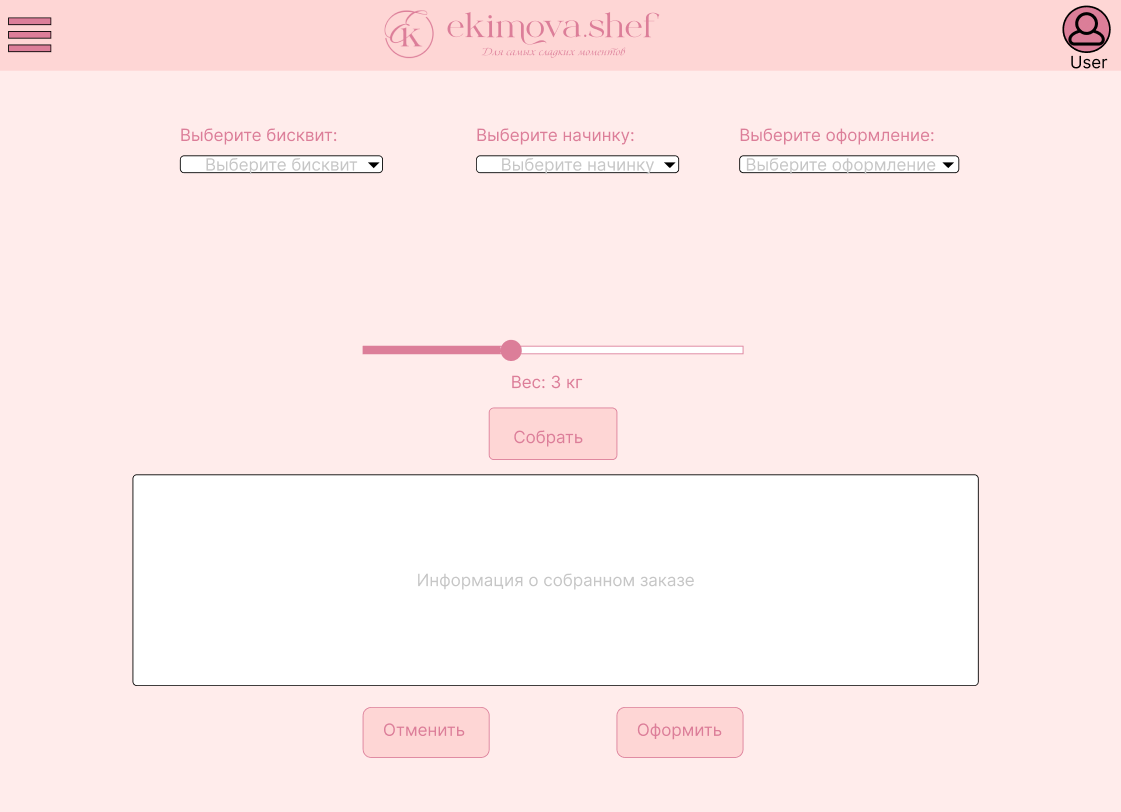


Рисунок 2.6 – Макет экрана для формирования заказа

Таблица 2.2 – Описание входной информации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название поля** | **Тип**  **значения поля** | **Формат**  **записи значения поля** | **Диапазон**  **изменения**  **значения поля** | **Примечание** |
| Бисквит | Строка | - | - | Данные выбираются из предоставленного справочника |
| Начинка | Строка | - | - | Данные выбираются из предоставленного справочника |
| Оформление | Строка | - | - | Данные выбираются из предоставленного справочника |
| Вес | Число | - | Определяется ползунком (мин – 1,5кг. Макс – 7кг.) | Данные выбираются перемещением ползунка |

### Список ингредиентов



Рисунок 2.7 – Макет экрана для просмотра списка ингредиентов

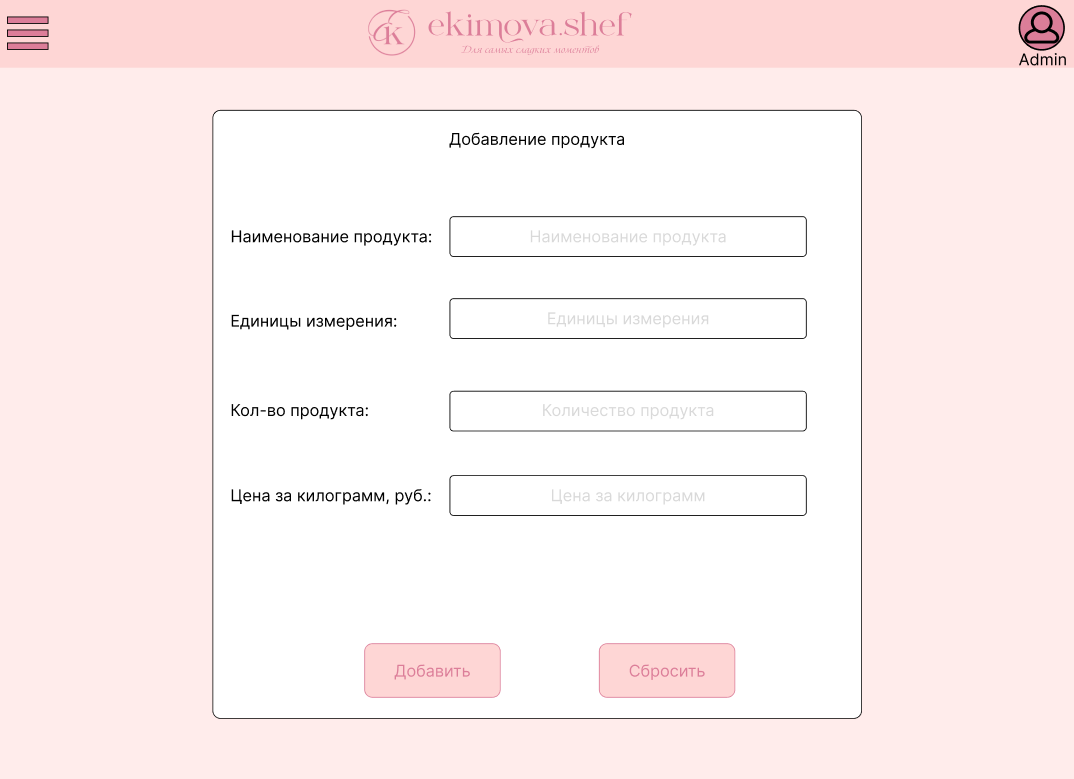


Рисунок 2.8 – Макет экрана добавления ингредиента

Таблица 2.3 – Описание входной информации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название поля** | **Тип значения поля** | **Формат записи значения поля** | **Диапазон изменения значения поля** | **Примечание** |
| Наименование продукта | Строка | - | - | Данные вводятся пользователем |
| Единицы измерения | Строка | - | - | Данные выбираются из предоставленного справочника |
| Кол-во продукта | Число | - | (0, 100000] | Данные вводятся пользователем |
| Цена за килограмм, руб. | Число | - | (0, 100000] | Данные вводятся пользователем |

### Анализ продаж

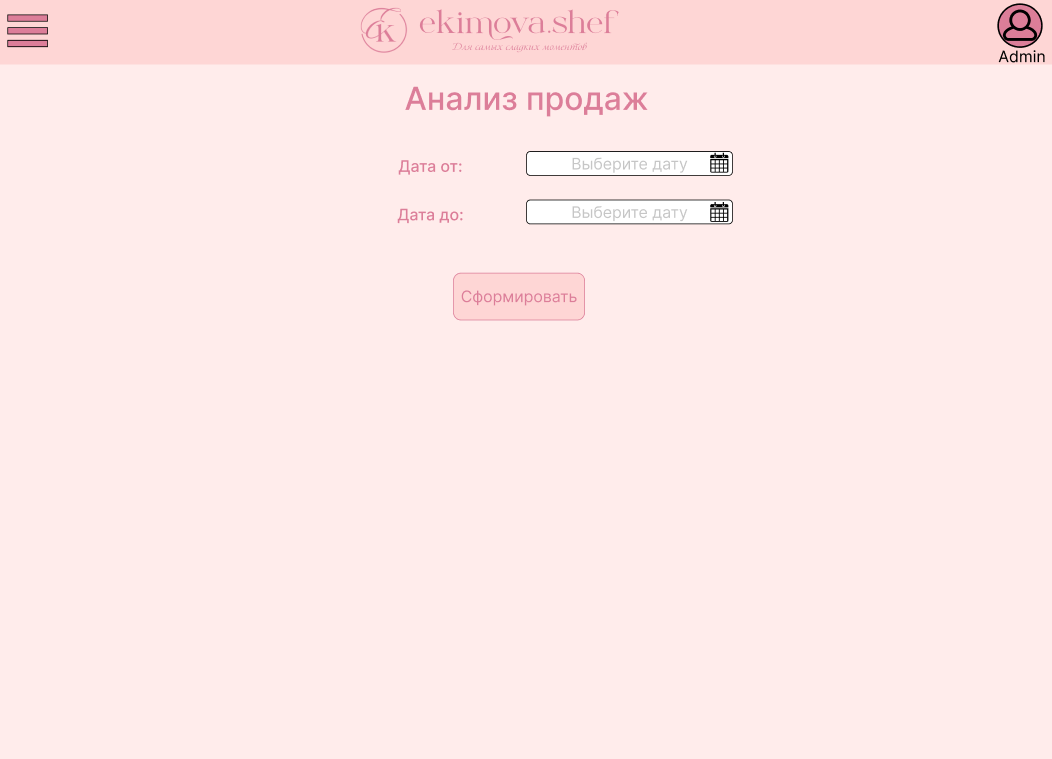


Рисунок 2.9 – Макет экрана анализа продаж

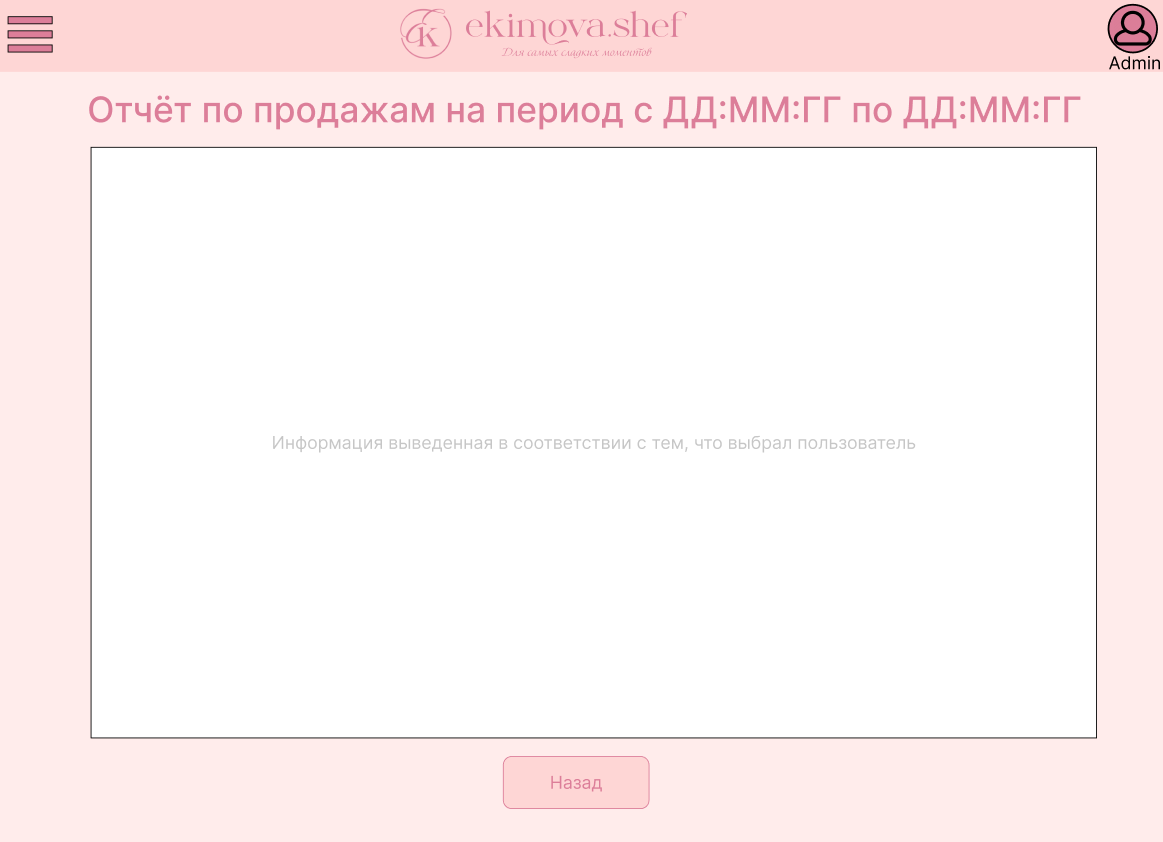


Рисунок 2.10 – Макет экрана отчёта по продажам

Таблица 2.4 – Описание входной информации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название поля** | **Тип**  **значения поля** | **Формат**  **записи**  **значения поля** | **Диапазон**  **изменения**  **значения поля** | **Примечание** |
| Дата от | Дата | ДД:ММ:ГГ | - | Данные вводятся пользователем |
| Дата до | Дата | ДД:ММ:ГГ | - | Данные вводятся пользователем |

## Карта экранов и состояний

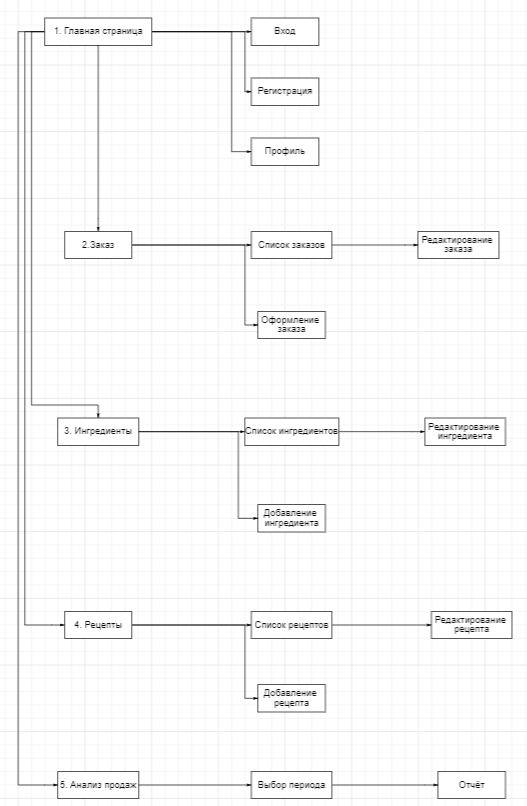


Рисунок 2.11 – Карта экранов

# Разработка

В данной работе будет представлена часть разработки, связанная с учётом ингредиентов и анализе закупок.

## Математическое обеспечение

Математическое обеспечение для обеих задач представлено на рисунках 17 и 18.

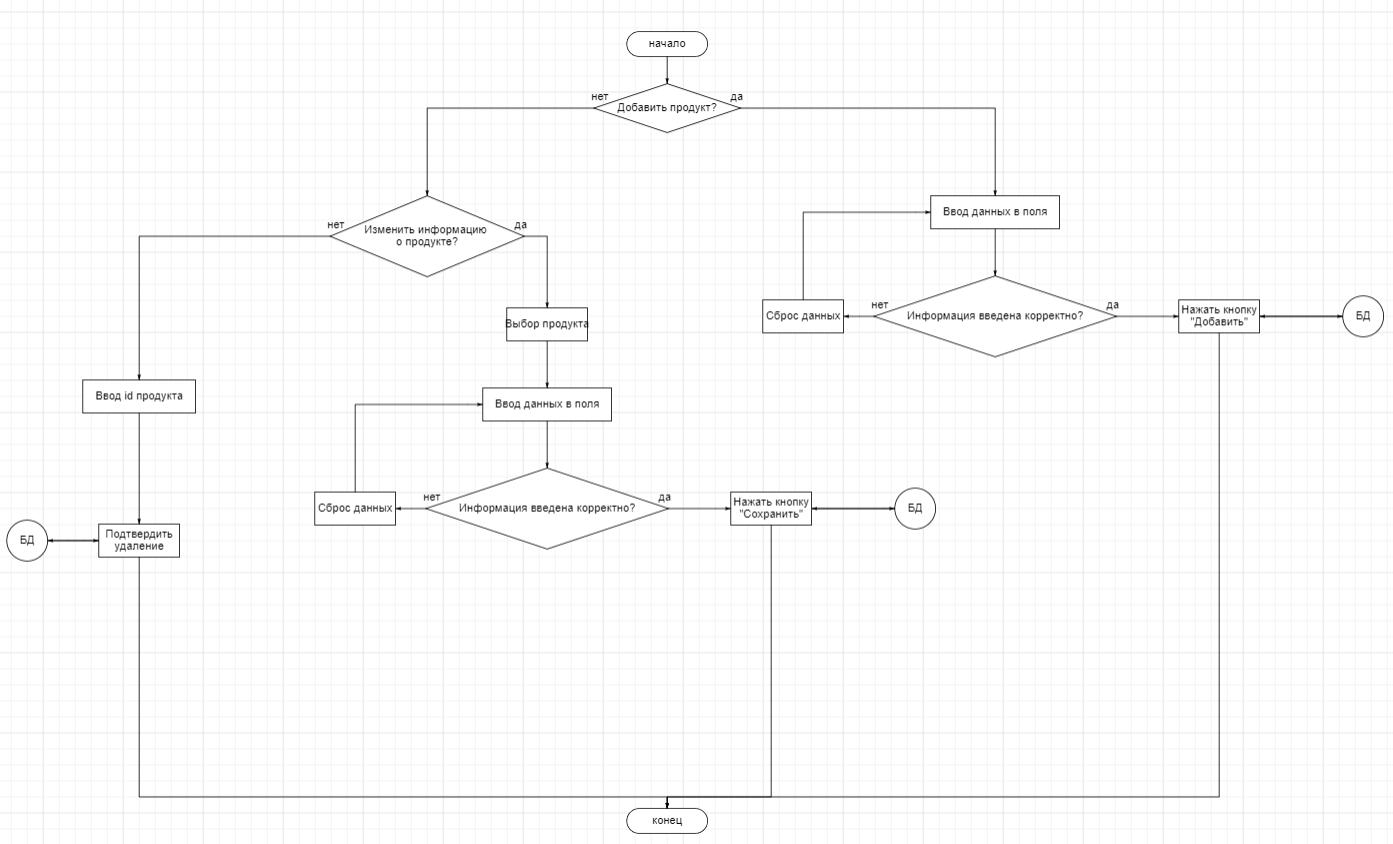


Рисунок 3.1 – Алгоритм работы задачи учёта ингредиентов

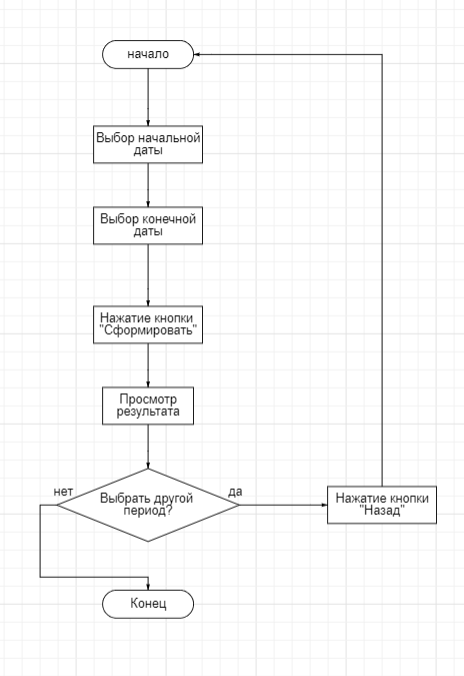


Рисунок 3.2 – Алгоритм работы задачи анализа закупок

## Экраны приложения

Главная страница приложения представлена на рисунке 3.3.

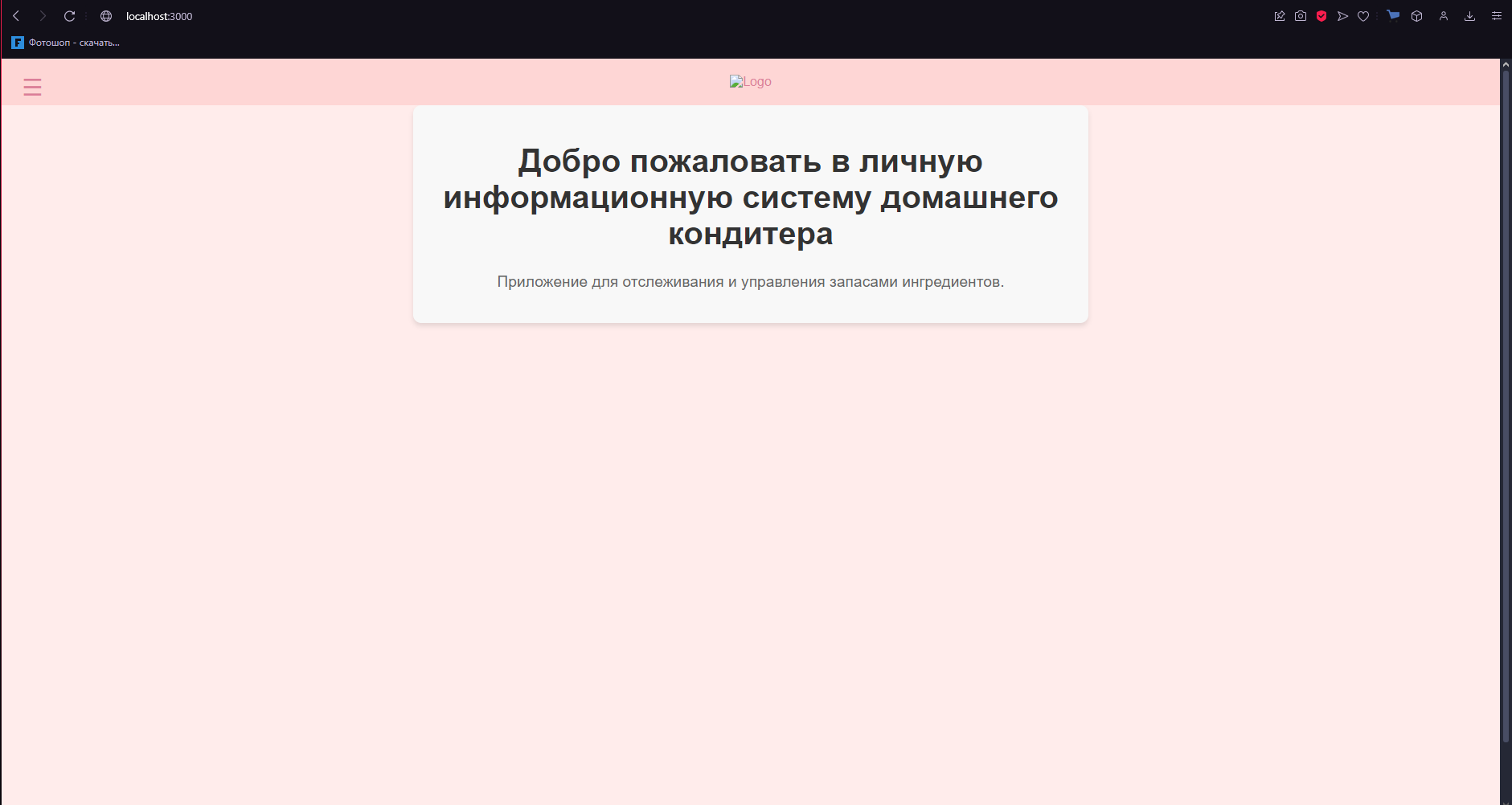


Рисунок 3.3 – Главная страница приложения

Для открытия навигационного меню необходимо нажать на соответствующий значок в левом верхнем углу страницы. Навигационное меню представлено на рисунке 3.4.

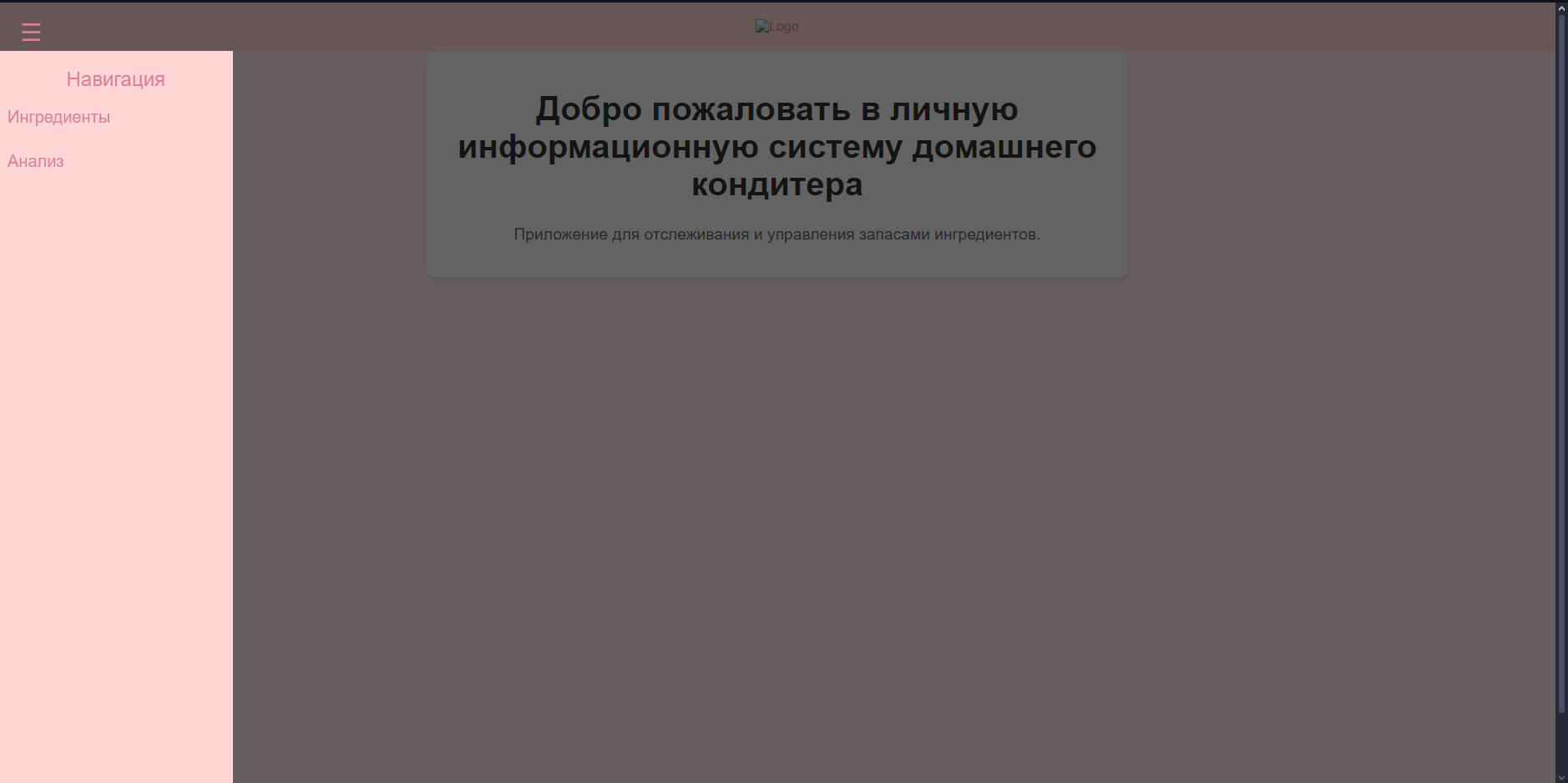


Рисунок 3.4 – Навигационное меню приложения

Назначения пунктов меню указаны в таблице 3.1.

Таблица 3.1 – Назначение пунктов меню

|  |  |
| --- | --- |
| **Название страницы** | **Назначение** |
| Ингредиенты | Страница “Ингредиенты” покажет список ингредиентов на складе, а также их количество, стоимость, дату закупки и пр. |
| Анализ | Страница “Анализ” покажет форму в которой необходимо задать период за который будет собираться информация для отчёта |

***Страница «Ингредиенты»:***

При переходе на страницу “Ингредиенты” пользователя встречает таблица с информацией об ингредиентах, которые используются домашним кондитером при приготовлении различных десертов (см. рисунок 3.5).

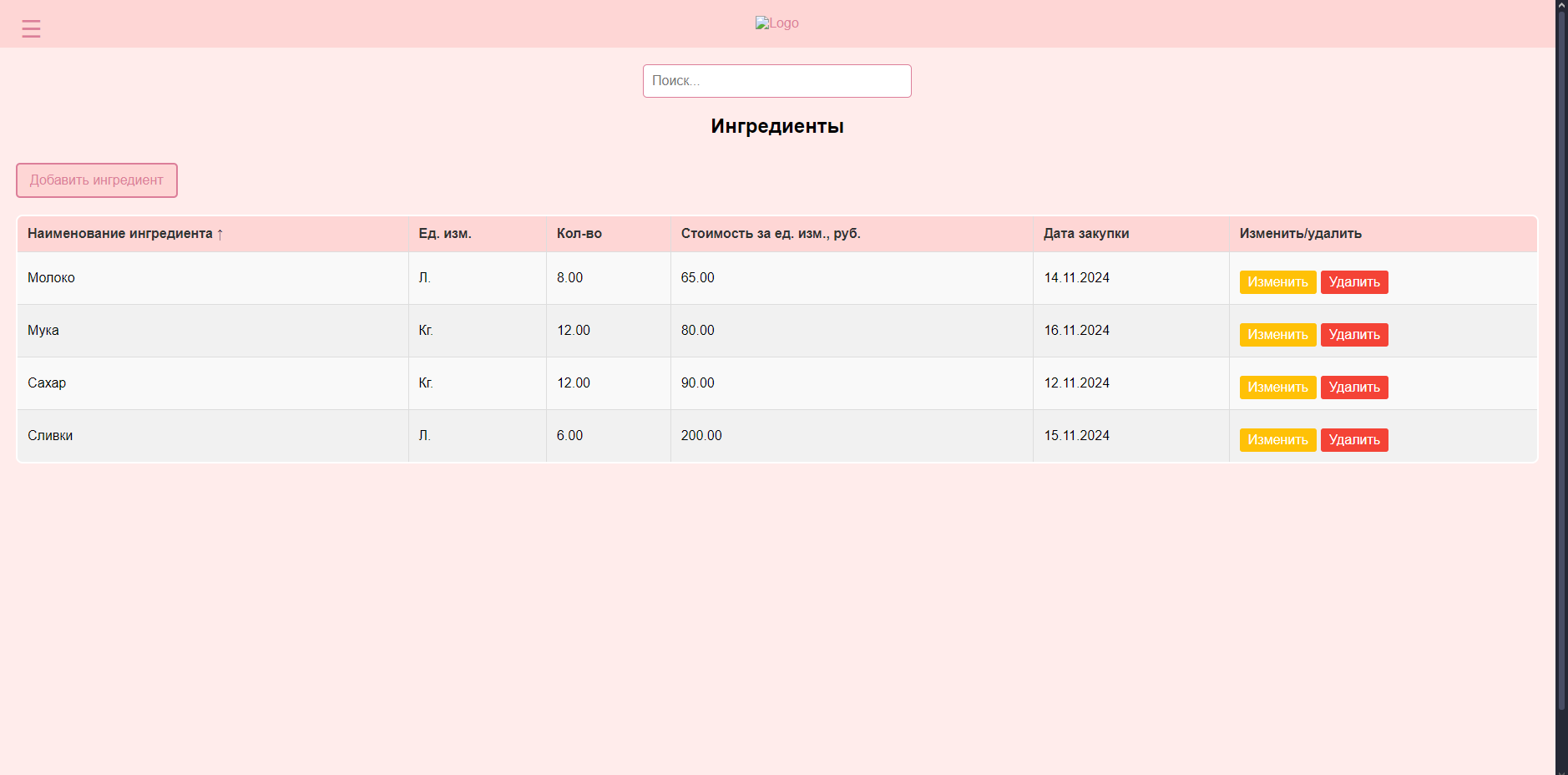


Рисунок 3.5 – Страница «Ингредиенты»

Под «шапкой» сайта расположена строка поиска, при вводе информации в которую происходит динамический поиск ингредиентов (см. рисунок 3.6).

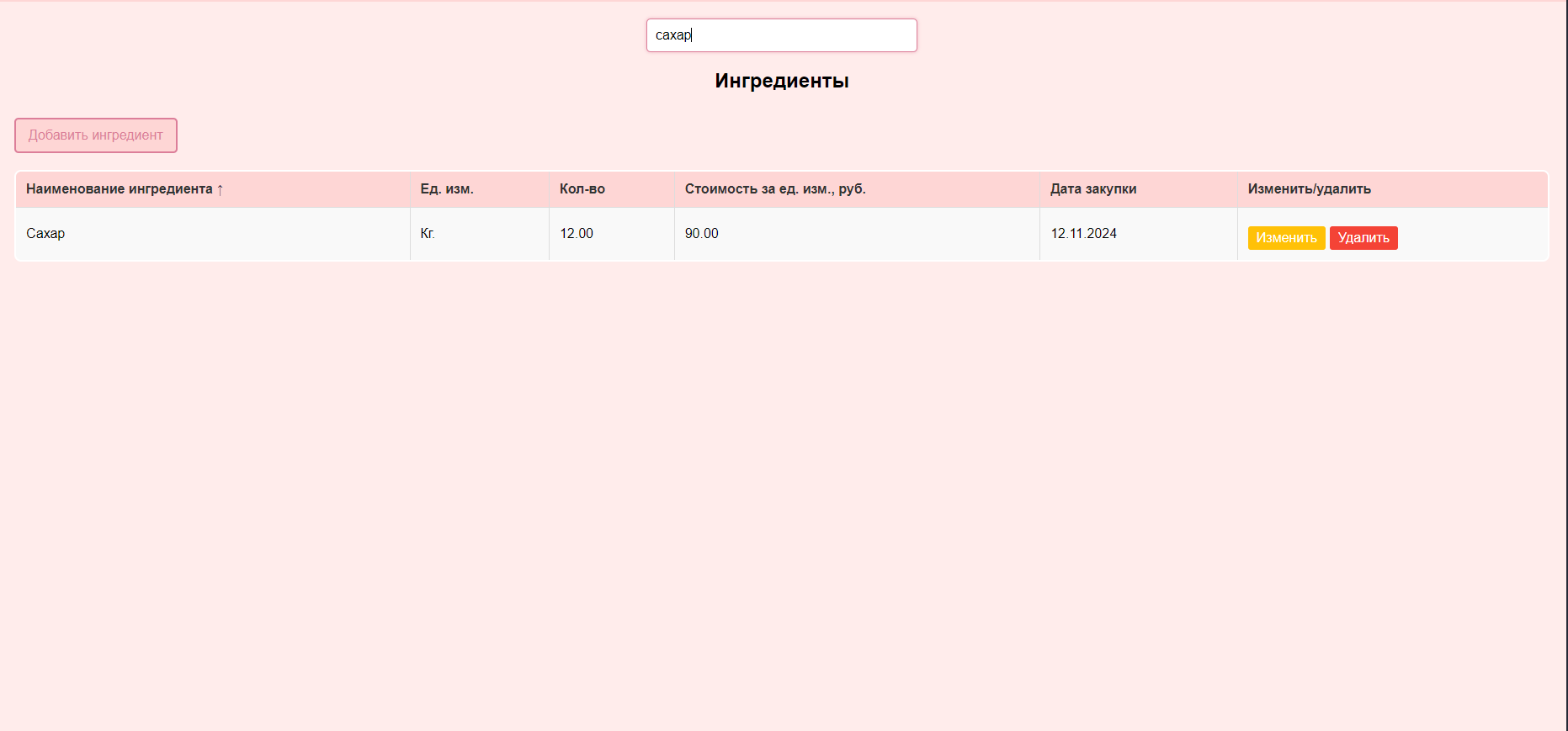


Рисунок 3.6 – Поиск ингредиента

В таблице у каждой записи присутствует поле с кнопками «Изменить» и «Удалить». При нажатии на кнопку «Изменить» осуществляется переход в режим редактирования, где можно будет изменить всю соответствующую информацию об ингредиенте (см. рисунок 3.7).

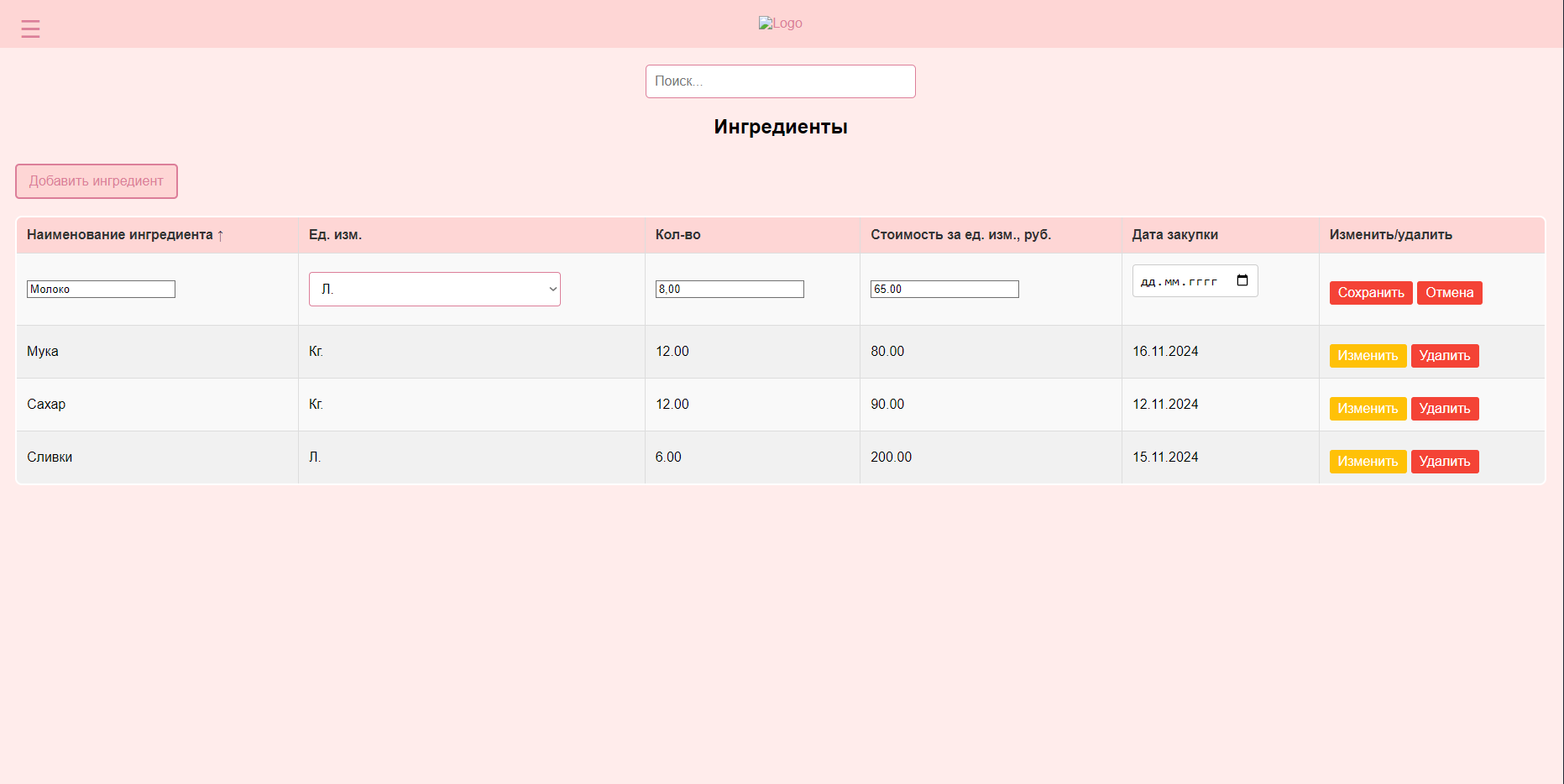


Рисунок 3.7 – Нажатие кнопки «Изменить»

Для сохранения изменений необходимо нажать на кнопку «Сохранить». Если изменения сохранять не нужно нажмите на кнопку «Отмена». При нажатии на кнопку «Удалить», выбранная запись будет удалена из базы данных и перестанет отображаться в системе. Над таблицей расположена кнопка «Добавить ингредиент» при нажатии на которую происходит переход к форме для заполнения информации об ингредиенте (см. рисунок 3.8).

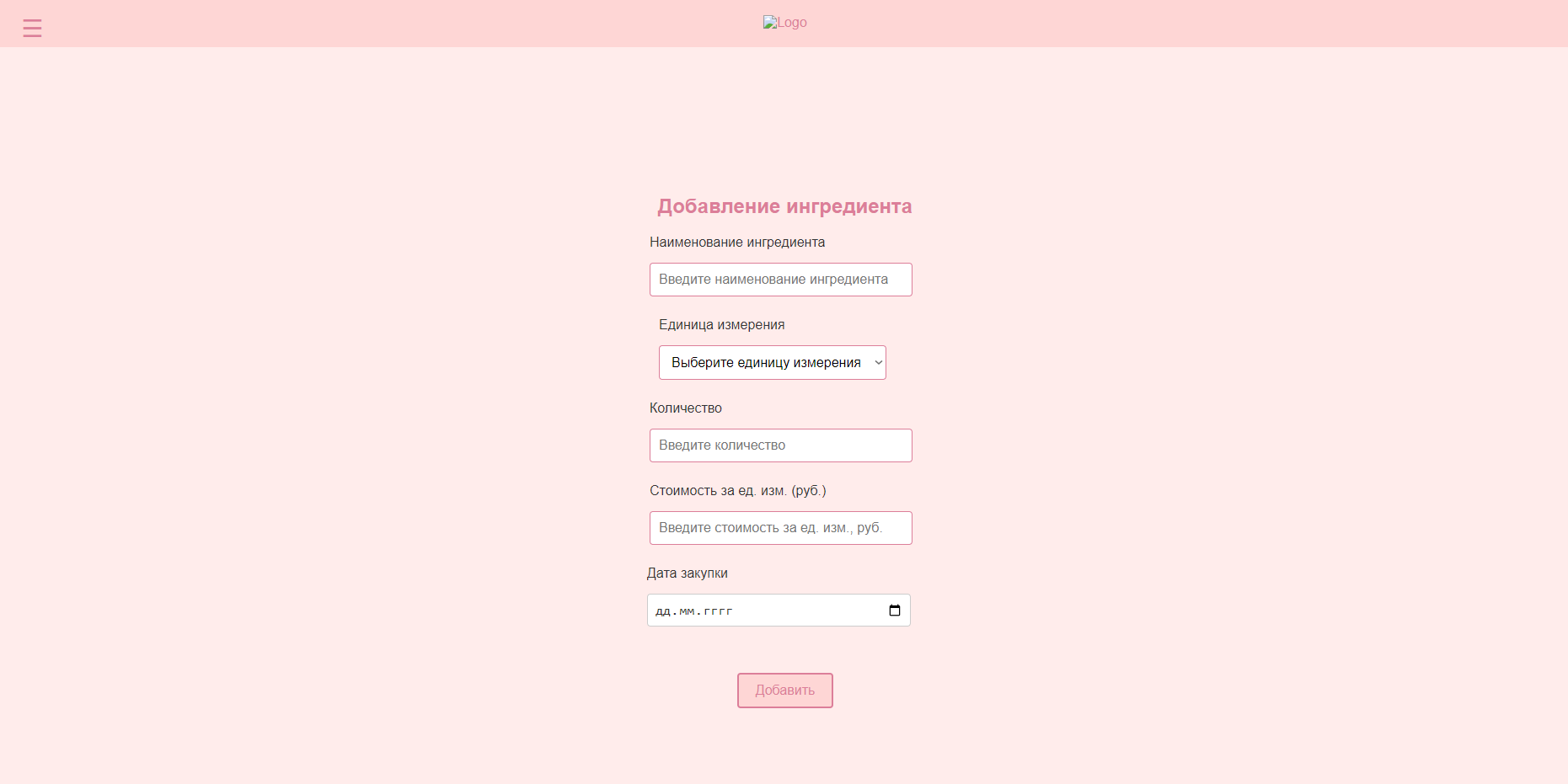


Рисунок 3.8 – Форма для заполнения информации об ингредиенте

Назначения полей в форме для заполнения указаны в таблице 3.2.

Таблица 3.2 – Назначение полей ввода

|  |  |  |
| --- | --- | --- |
| **Наименование поля** | **Тип данных** | **Назначение** |
| Наименование ингредиента | Строка | Информация о названии ингредиента |
| Единицы измерения | Строка. Данные выбираются из выпадающего списка | Информация о том в каких единицах измеряется ингредиент |
| Количество | Число | Информация о том, какое количество ингредиента доступно на “складе” |
| Стоимость за ед. изм. (руб.) | Число | Информация о стоимости ингредиента за единицу измерения |
| Дата закупки | Дата | Информация о дате последней закупки ингредиента. |

После заполнения формы необходимо нажать кнопку «Добавить» внизу страницы для того, чтобы информация была занесена в базу данных и отображалась на сайте.

***Страница «Анализ»:***

При переходе на страницу «Анализ» появляется форма с выбором

периода, за который необходимо сформировать отчёт (см. рисунок 3.9).

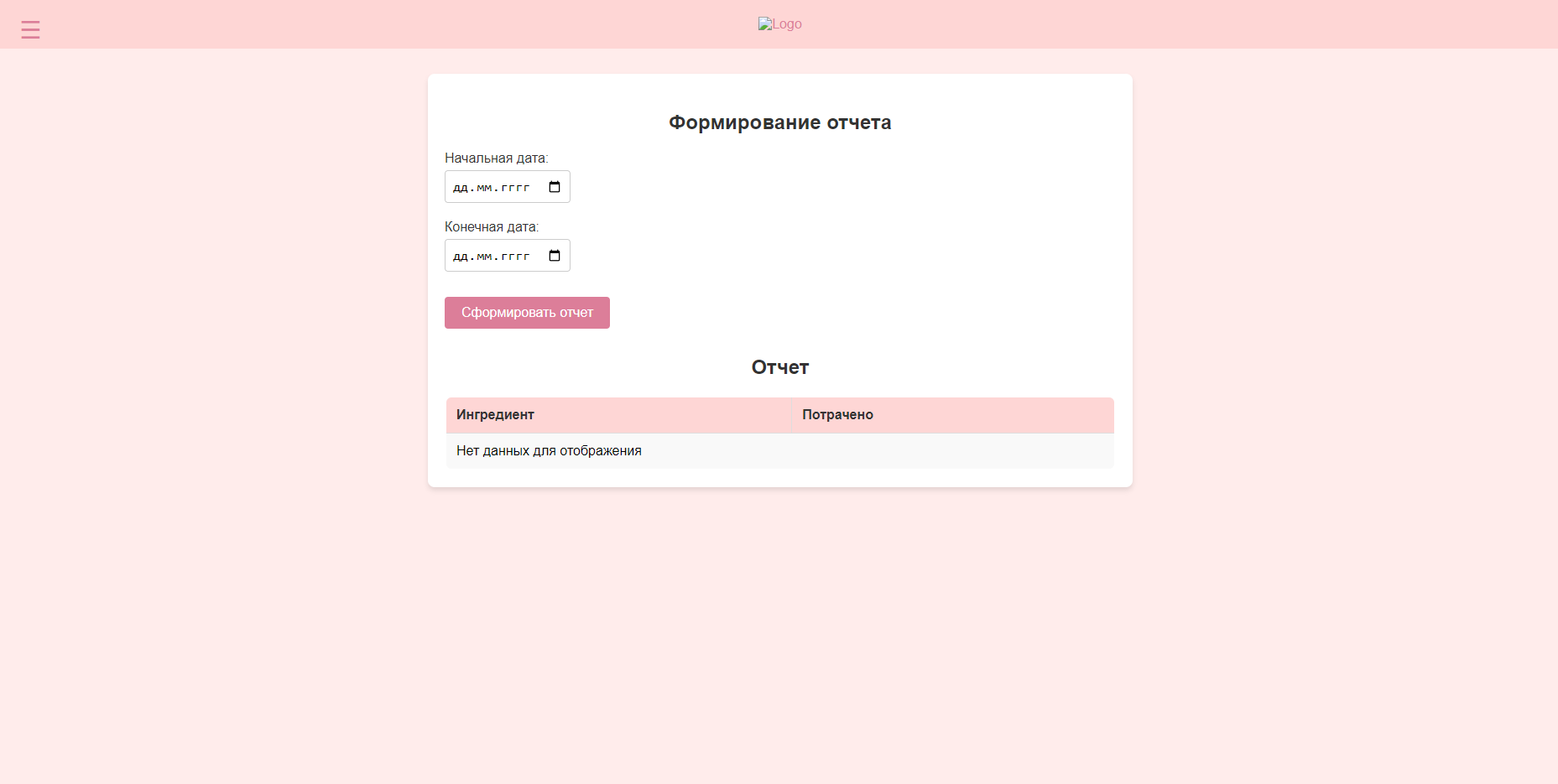


Рисунок 3.9 – Страница «Анализ»

Для формирования отчёта необходимо задать начальную и конечные даты в соответствующих полях. После этого нажать на кнопку «Сформировать отчёт». При выборе периода и нажатии кнопки будет выведена отчётная информация (см. рисунки 3.10-3.11).

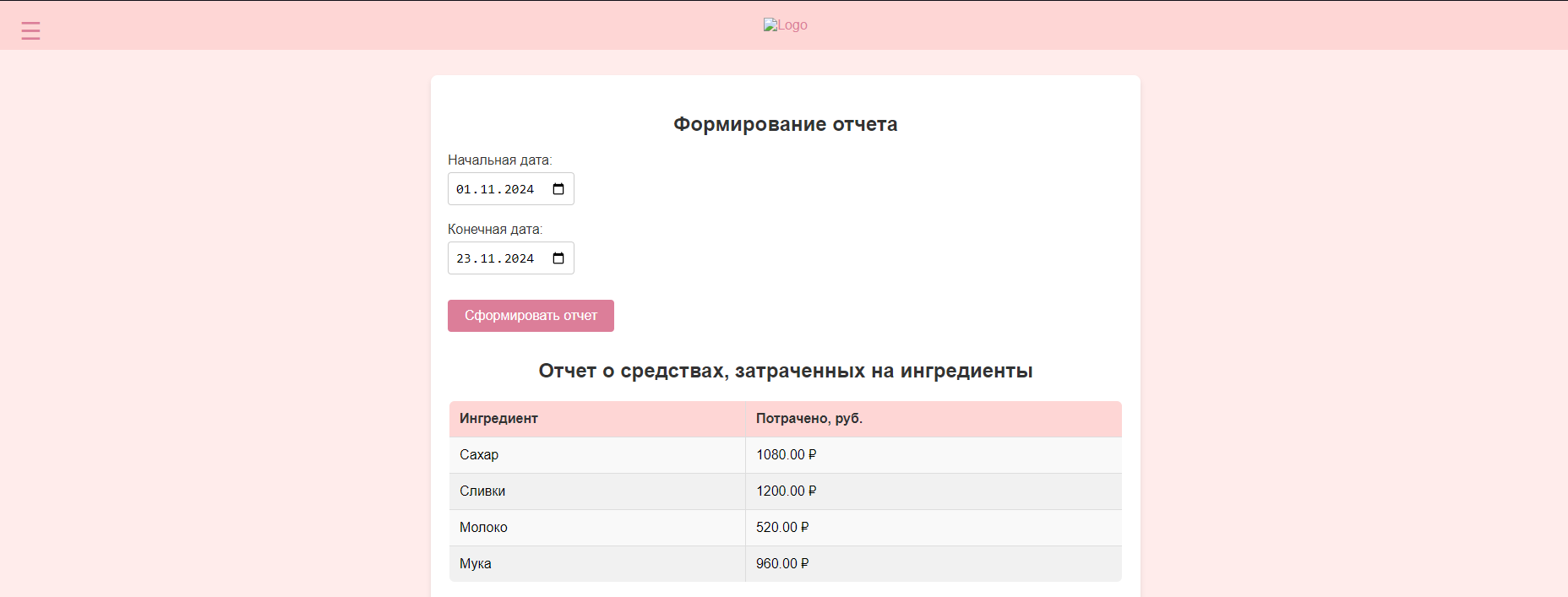


Рисунок 3.10 – Результаты анализа таблица

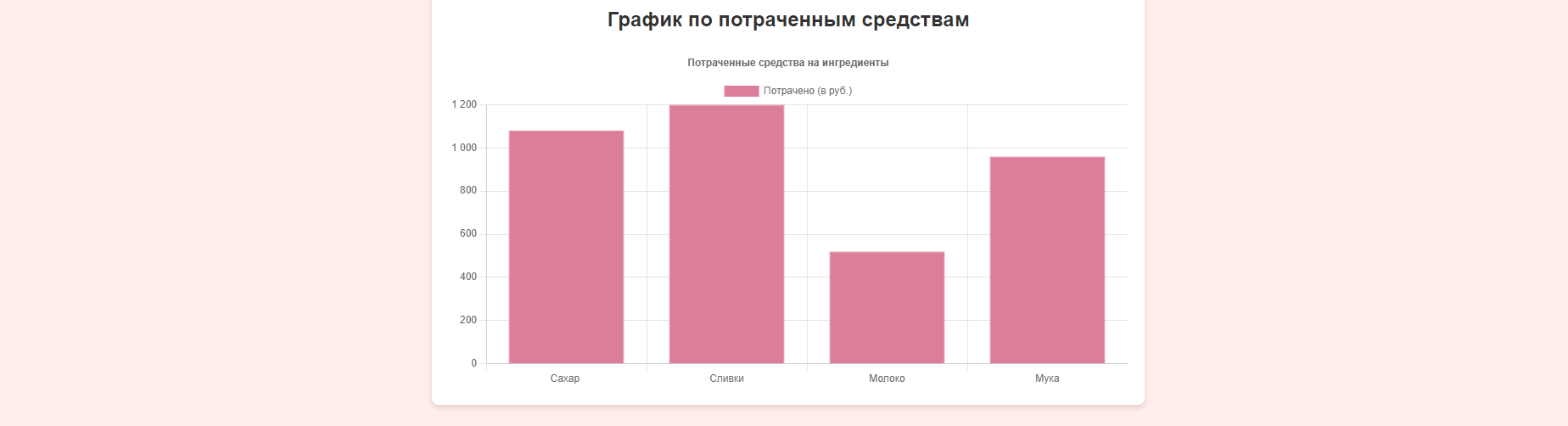


Рисунок 3.11 – Результаты анализа график

# Заключение

В результате выполнения курсовой работы был разработан проект личной информационной системы для домашнего кондитера. Основной целью системы является оптимизация управления ключевыми бизнес-процессами, включая учет заказов, управление рецептами, планирование закупок, финансовый учет и взаимодействие с клиентами.

В рамках работы был выполнен анализ текущего состояния процессов (AS IS) и сформулирована модель будущего состояния (TO BE). Были исследованы существующие программные средства, выявлены их сильные и слабые стороны. На основе анализа определены функциональные и нефункциональные требования, разработаны архитектура системы, пользовательский интерфейс и логическая структура базы данных. Прототип интерфейса и математическое обеспечение задач подтверждают реализуемость проекта.

Предложенные решения позволили объединить разрозненные задачи домашнего кондитера в единую платформу. Это значительно сократит временные затраты на управление, минимизирует ошибки и улучшит качество обслуживания клиентов.

Перспективы развития проекта:

* расширение функционала аналитики и отчетности.
* внедрение мобильного приложения для упрощения доступа пользователей.
* поддержка многопользовательского режима для работы небольших команд.
* автоматизация маркетинговых процессов, включая уведомления о скидках и акциях.

# Список использованных источников

1. Левин А.А. React и Redux: разработка web-приложений / А.А. Левин. – СПб: БХВ-Петербург, 2018. – 352 с.
2. Гутгарц Р.Д Проектирование автоматизированных систем обработки информации и управления: учебное пособие для академического бакалавриата. – М.: Издательство Юрайт, 2019.
3. Проектирование АСОИУ [Электронный ресурс] : методические указания по выполнению курсового проекта: 09.03.01 «Информатика и вычислительная техника» для бакалавров по специальности «Автоматизированные системы обработки информации и управления» / Иркут. нац. исслед. техн. ун-т ; сост. Р. Д. Гутгарц. - Электрон. дан. - Иркутск : ИРНИТУ, 2018 (дата обращения: 26.11.2024).
4. Рудинский И.Д. Технология проектирования автоматизированных систем обработки информации и управления : учебное пособие для вузов / И. Д. Рудинский. - Москва : Горячая линия - Телеком, 2011.
5. Дорф Р. Современные системы управления/ Р. Дорф, Р. Бишоп; пер. с англ. Б.И. Копылова. – М.: Лаборатория Базовых Знаний, 2006. – 832 с.

# Приложения

## Приложение А

Программный код страницы «Ингредиенты»

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import './ProductTable.css';

function ProductTable() {

const [products, setProducts] = useState([]);

const [editingId, setEditingId] = useState(null);

const [editData, setEditData] = useState({}); // Данные для редактирования

const [searchQuery, setSearchQuery] = useState('');

const [sortConfig, setSortConfig] = useState({

key: 'name', // Столбец по умолчанию для сортировки

direction: 'asc', // Направление сортировки (по возрастанию)

});

useEffect(() => {

fetchProducts();

}, []);

const fetchProducts = async () => {

try {

const response = await axios.get('http://localhost:5000/api/products');

setProducts(response.data);

} catch (error) {

console.error('Ошибка при получении данных', error);

}

};

const handleEdit = (product) => {

setEditingId(product.id\_product);

setEditData({ ...product });

};

const handleCancelEdit = () => {

setEditingId(null);

setEditData({});

};

const handleSaveEdit = async () => {

try {

await axios.put(`http://localhost:5000/api/products/${editingId}`, editData);

setEditingId(null);

setEditData({});

fetchProducts();

} catch (error) {

console.error('Ошибка при обновлении записи', error);

}

};

const handleChange = (e) => {

const { name, value } = e.target;

setEditData({ ...editData, [name]: value });

};

const handleDelete = async (id) => {

try {

await axios.delete(`http://localhost:5000/api/products/${id}`);

fetchProducts();

} catch (error) {

console.error('Ошибка при удалении записи', error);

}

};

const handleSearch = (e) => {

setSearchQuery(e.target.value.toLowerCase());

};

// Функция для форматирования даты

const formatDate = (date) => {

const d = new Date(date);

const day = String(d.getDate()).padStart(2, '0');

const month = String(d.getMonth() + 1).padStart(2, '0');

const year = d.getFullYear();

return `${day}.${month}.${year}`;

};

// Фильтрация продуктов по строке поиска

const filteredProducts = products.filter((product) =>

product.name.toLowerCase().includes(searchQuery)

);

// Функция для сортировки

const sortedProducts = [...filteredProducts].sort((a, b) => {

const { key, direction } = sortConfig;

if (a[key] < b[key]) {

return direction === 'asc' ? -1 : 1;

}

if (a[key] > b[key]) {

return direction === 'asc' ? 1 : -1;

}

return 0;

});

// Функция для изменения порядка сортировки

const handleSort = (key) => {

let direction = 'asc';

if (sortConfig.key === key && sortConfig.direction === 'asc') {

direction = 'desc';

}

setSortConfig({ key, direction });

};

// Функция для получения значка сортировки

const getSortIcon = (column) => {

if (sortConfig.key === column) {

return sortConfig.direction === 'asc' ? '↑' : '↓';

}

return '';

};

return (

<div className="product-table-container">

{/\* Строка поиска \*/}

<div className="search-input-container">

<input

type="text"

className="search-input"

placeholder="Поиск..."

value={searchQuery}

onChange={handleSearch}

/>

</div>

<h2>Ингредиенты</h2>

<button onClick={() => (window.location.href = '/add')} className="add-button">

Добавить ингредиент

</button>

<table>

<thead>

<tr>

<th onClick={() => handleSort('name')}>

Наименование ингредиента {getSortIcon('name')}

</th>

<th onClick={() => handleSort('unit')}>

Ед. изм. {getSortIcon('unit')}

</th>

<th onClick={() => handleSort('quantity')}>

Кол-во {getSortIcon('quantity')}

</th>

<th onClick={() => handleSort('unit\_price')}>

Стоимость за ед. изм., руб. {getSortIcon('unit\_price')}

</th>

<th onClick={() => handleSort('purchase\_date')}>

Дата закупки {getSortIcon('purchase\_date')}

</th>

<th>Изменить/удалить</th>

</tr>

</thead>

<tbody>

{sortedProducts.map((product) => (

<tr key={product.id\_product}>

{editingId === product.id\_product ? (

<>

<td>

<input

type="text"

name="name"

value={editData.name || ''}

onChange={handleChange}

/>

</td>

<td>

{/\* Выпадающий список для единицы измерения \*/}

<select

name="unit"

value={editData.unit || ''}

onChange={handleChange}

>

<option value="Кг.">Кг.</option>

<option value="Л.">Л.</option>

<option value="Шт.">Шт.</option>

</select>

</td>

<td>

<input

type="number"

name="quantity"

value={editData.quantity || ''}

onChange={handleChange}

/>

</td>

<td>

<input

type="text"

name="unit\_price"

value={editData.unit\_price || ''}

onChange={handleChange}

/>

</td>

<td>

<input

type="date"

name="purchase\_date"

value={editData.purchase\_date || ''}

onChange={handleChange}

/>

</td>

<td>

<button onClick={handleSaveEdit}>Сохранить</button>

<button onClick={handleCancelEdit}>Отмена</button>

</td>

</>

) : (

<>

<td>{product.name}</td>

<td>{product.unit}</td>

<td>{product.quantity}</td>

<td>{product.unit\_price}</td>

<td>{formatDate(product.purchase\_date)}</td>

<td>

<button onClick={() => handleEdit(product)} className="edit-button">

Изменить

</button>

<button onClick={() => handleDelete(product.id\_product)}>Удалить</button>

</td>

</>

)}

</tr>

))}

</tbody>

</table>

</div>

);

}

export default ProductTable;

## Приложение Б

Программный код страницы «Анализ»

import React, { useState } from 'react';

import axios from 'axios';

import { Bar } from 'react-chartjs-2'; // Импортируем диаграмму

import { Chart as ChartJS, CategoryScale, LinearScale, BarElement, Title, Tooltip, Legend } from 'chart.js';

import styles from './AnalysisPage.module.css'; // Импортируем стили

// Регистрируем необходимые компоненты для Chart.js

ChartJS.register(CategoryScale, LinearScale, BarElement, Title, Tooltip, Legend);

function AnalysisForm() {

const [startDate, setStartDate] = useState('');

const [endDate, setEndDate] = useState('');

const [report, setReport] = useState([]);

const [chartData, setChartData] = useState({});

const handleGenerateReport = async () => {

try {

// Отправляем запрос на сервер с диапазоном дат

const response = await axios.post('http://localhost:5000/api/analysis', {

startDate: startDate,

endDate: endDate,

});

// Логируем ответ для отладки

console.log('Report Data:', response.data);

// Преобразуем строку totalSpent в число и обновляем состояние с полученными данными

const formattedData = response.data.map(item => ({

...item,

totalSpent: parseFloat(item.totalspent).toFixed(2), // Преобразуем в число и округляем до 2 знаков

}));

// Обновляем данные отчета

setReport(formattedData);

// Формируем данные для диаграммы

const chartLabels = formattedData.map(item => item.name);

const chartValues = formattedData.map(item => parseFloat(item.totalSpent));

setChartData({

labels: chartLabels,

datasets: [

{

label: 'Потрачено (в руб.)',

data: chartValues,

backgroundColor: '#DC7E99',

borderColor: '#DC7E99',

borderWidth: 1,

},

],

});

} catch (error) {

console.error('Ошибка при формировании отчета:', error);

}

};

return (

<div className={styles.analysisContainer}>

<h3>Формирование отчета</h3>

<div>

<label>Начальная дата:</label>

<input

type="date"

value={startDate}

onChange={(e) => setStartDate(e.target.value)}

/>

</div>

<div>

<label>Конечная дата:</label>

<input

type="date"

value={endDate}

onChange={(e) => setEndDate(e.target.value)}

/>

</div>

<button onClick={handleGenerateReport}>Сформировать отчет</button>

<h4>Отчет о средствах, затраченных на ингредиенты</h4>

<table>

<thead>

<tr>

<th>Ингредиент</th>

<th>Потрачено, руб.</th>

</tr>

</thead>

<tbody>

{report.length > 0 ? (

report.map((item, index) => (

<tr key={index}>

<td>{item.name}</td>

<td>{item.totalSpent} ₽</td>

</tr>

))

) : (

<tr>

<td colSpan="2">Нет данных для отображения</td>

</tr>

)}

</tbody>

</table>

{/\* Отображение диаграммы, если есть данные \*/}

{report.length > 0 && (

<div className={styles.chartContainer}>

<h4>График по потраченным средствам</h4>

<Bar data={chartData} options={{ responsive: true, plugins: { title: { display: true, text: 'Потраченные средства на ингредиенты' } } }} />

</div>

)}

</div>

);

}

export default AnalysisForm;