**Реферат**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е.А .

Реферат

Лит.

Листов

1

*74217076, 2024*

Пояснительная записка дипломного проекта содержит 70 страниц пояснительной записки, 10 таблиц, 13 листингов, 19 формул, 39 иллюстраций, 7 источников литературы, 12 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ANDROID, KOTLIN, JETPACK COMPOSE, SPRING FRAMEWORK, JAVA, MS SQL SERVER

Целью дипломного проекта являлось мобильное приложение для оценки физической активности. Пояснительная записка состоит из введения, семи разделов, заключения и источников литературы.

В первой главе проводится аналитический обзор аналогичных решений и постановка задач.

Вторая глава посвящена проектированию системы и содержит структурную схему базы данных, диаграмму вариантов использования, а также схему алгоритма, задействованного в проекте.

В третьей главе описаны процесс разработки, принципы функционирования и назначение созданных компонентов проекта.

В четвертой главе описано тестирования разработанного мобильного приложения.

В шестой главе приведено руководство программиста, позволяющее понять моменты развертывания частей проекта.

В седьмой главе производится расчет экономических параметров, срока окупаемости мобильного приложения.

Графическая часть включает в себя 4 диаграммы и 2 плаката, указанных в приложениях записки, суммарно имея 6 графических материалов.

**Abstract**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О.А.*

Утв.

Блинова Е. А.

Abstract

Лит.

Листов

1

*74217076, 2024*

The explanatory note of the diploma project contains 70 pages of an explanatory note, 10 tables, 13 listings, 19 formulas, 39 illustrations, 7 sources of literature, 12 appendices.

MOBILE APPLICATION, ANDROID, KOTLIN, JETPACK COMPOSE, SPRING FRAMEWORK, JAVA, MS SQL SERVER

The purpose of graduation project was a mobile application for accessing physical activity. The explanatory note consists of an introduction, seven sections, a conclusion and references.

In the first chapter, an analytical review of similar solutions and the formation of tasks is carried out.

The second chapter is devoted to the design of the system and contains a structural diagram of the database, a diagram of use cases, as well as a diagram of the algorithm involved in the project.

The third chapter describes the development process, the principles of operation and the purpose of the created components of the project.

The fourth chapter describes the testing of developed mobile application.

The sixth chapter contains a programmer’s guide that allow to understand the moments of deployment of parts of the project.

In the seventh chapter, the calculation of economic parameters, the payback period of the software tool is carried out.

The graphic part includes 4 diagrams and 2 posters specified in the appendices of the note, having a total of 6 graphic materials.

Содержание

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О.А.*

Утв.

Блинова Е. А.

Содержание

Лит.

Листов

2

*74217076, 2024*

[Введение 7](#_Toc2768)

[1 Постановка задачи и обзор аналогов 8](#_Toc29984)

[1.1 Актуальность исследуемой области 8](#_Toc12664)

[1.2 Обзор аналогов 8](#_Toc10946)

[1.2.1. Приложение «Bring!» 9](#_Toc4961)

[1.2.2. Веб-приложение «Listonic» 10](#_Toc32189)

[1.2.3. Веб-приложение «Russian food» 12](#_Toc11500)

[1.3 Постановка задачи 14](#_Toc969)

[1.4 Вывод по разделу 14](#_Toc1125)

[2 Проектирование приложения 15](#_Toc7083)

[2.1 Архитектура приложения 15](#_Toc20606)

[2.2 Проектирование базы данных 16](#_Toc12353)

[2.2.1. Описание таблиц 17](#_Toc2989)

[2.3 Диаграмма вариантов использования 21](#_Toc26878)

[2.4 Проектирование основных алгоритмов 23](#_Toc9024)

[2.5 Выбор средств реализации 23](#_Toc23305)

[2.5.1. Серверная часть 23](#_Toc10353)

[2.5.2. Клиентская часть 24](#_Toc10403)

[2.5.3. Централизованная база данных 24](#_Toc23231)

[2.6 Выводы по разделу 24](#_Toc10373)

[3 Разработка приложения 25](#_Toc24143)

[3.1 Реализация серверной части 25](#_Toc15694)

[3.2 Разработка клиентской части 29](#_Toc17963)

[3.3 Контейнеризация 32](#_Toc1337)

[3.4 Вывод по разделу 35](#_Toc6545)

[4 Тестирование приложения 36](#_Toc3252)

[4.1 Выводы по разделу 39](#_Toc29880)

[5 Руководство программиста 40](#_Toc17046)

[5.1 Руководство по развёртыванию приложения 40](#_Toc10415)

[5.2 Руководство для не авторизированного пользователя 41](#_Toc8955)

[5.3 Руководство для авторизированного пользователя 41](#_Toc6979)

[5.4 Руководство для администратора/модератора 44](#_Toc30227)

[5.5 Выводы по разделу 45](#_Toc28436)

[6 Технико-экономическое обоснование проекта 46](#_Toc10862)

[6.1 Общая характеристика разрабатываемого программного средства 46](#_Toc32057)

[6.2 Исходные данные для проведения расчётов 47](#_Toc20096)

[6.3 Методика обоснования цены 47](#_Toc13605)

[6.3.1 Объем программного средства 48](#_Toc2934)

[6.3.2 Основная заработная плата 49](#_Toc23733)

[6.3.3 Дополнительная заработная плата 50](#_Toc21399)

[6.3.4 Отчисления в Фонд социальной защиты населения 50](#_Toc14720)

[6.3.5 Расходы на материалы 51](#_Toc18629)

[6.3.6 Расходы на оплату машинного времени 51](#_Toc13800)

[6.3.7 Прочие прямые затраты 52](#_Toc17365)

[6.3.8 Накладные расходы 52](#_Toc26497)

[6.3.9 Сумма расходов на разработку программного средства 52](#_Toc27605)

[6.3.10 Расходы на сопровождение и адаптацию 53](#_Toc18878)

[6.3.11 Полная себестоимость 53](#_Toc1160)

[6.3.12 Определение цены, оценка эффективности 53](#_Toc22283)

[6.4 Выводы по разделу 56](#_Toc5521)

[Заключение 57](#_Toc15294)

[Список использованных источников 58](#_Toc22568)

[ПРИЛОЖЕНИЕ А Диаграмма развёртывания 59](#_Toc31109)

[ПРИЛОЖЕНИЕ Б Логическая схема базы данных 60](#_Toc1874)

[ПРИЛОЖЕНИЕ В Блок-схема алгоритма добавления продукта в хранилище 61](#_Toc18227)

[ПРИЛОЖЕНИЕ Г Блок-схема алгоритма формирования списка покупок 62](#_Toc20202)

Введение

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е. А.

Введение

Лит.

Листов

1

*74217076, 2024*

Схемы автоматизации и использования информационных систем также могут быть успешно применены к составлению и управлению списками покупок. В современном мире, когда ритм жизни ускоряется, а выбор товаров в магазинах огромен, эффективное управление списком покупок становится важным аспектом организации покупок. Пользователи могут создавать списки покупок в специальных приложениях или веб-сервисах, где они могут добавлять необходимые продукты, следить за количеством и планировать покупки на основе своих потребностей.

Информационные системы могут предоставлять подробную информацию о продуктах, включая цены, сроки годности, пищевую ценность и другие характеристики. Это помогает покупателям принимать более информированные решения. Системы могут предоставлять рекомендации по покупкам, исходя из предыдущих покупок, предпочтений и даже диетических ограничений.

Более технологичные системы могут автоматически обновлять список покупок на основе имеющихся запасов дома, учитывая, что нужно пополнить. Информационные системы могут оптимизировать маршрут по магазину, чтобы сэкономить время и энергию покупателей. Могут предоставлять возможность сканировать товары и оплачивать их через мобильное приложение, что упрощает процесс покупок.

Основной аудиторией данного приложения будут семьи, активные люди, студенты и все, кто ценит своё время и стремится к оптимизации процесса покупок и приготовления пищи. С помощью приложения они смогут легко составлять списки покупок на основе рецептов, управлять своими запасами и экономить как время, так и энергию, необходимые для планирования и совершения покупок.

Целью данного дипломного проекта является создание приложения для просмотра списка имеющихся продуктов, просмотра идей для рецептов, а так же составление списка покупок на основании составов рецептов.

Задачи дипломного проекта:

* изучить аналоги разрабатываемого приложения на рынке;
* изучить и выбрать подходящие технологии для реализации проекта,
* спроектировать структуру базы данных,
* разработать сервер;
* разработать клиент;
* протестировать разработанное приложение;
* разработать руководство программиста по развёртыванию приложения;
* выполнить экономическое обоснование проекта.

# Постановка задачи и обзор аналогов

ФИО

Подпись

Дата

Лист

1

БГТУ 01.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е. А.

1 Постановка задачи и обзор аналогов

Лит.

Листов

8

*74217076, 2024*

* 1. Актуальность исследуемой области

В современном мире, где ритм жизни постоянно нарастает, эффективное планирование питания и управление покупками становятся все более актуальными задачами. Существует постоянная потребность в инструментах, способных помочь людям сократить время на походы по магазинам и упростить процесс составления списка покупок. Приложения, основанные на рецептах, представляют собой потенциальное решение этой проблемы, позволяя пользователям создавать списки покупок на основе ингредиентов, необходимых для приготовления конкретных блюд.

С разнообразием продуктов и услуг на рынке, выбор подходящих продуктов для покупки становится все более сложным. Приложения, предлагающие функциональность создания списков покупок на основе рецептов, могут значительно облегчить этот процесс, предоставляя пользователям точные списки ингредиентов, необходимых для приготовления конкретных блюд. Такие приложения также могут предложить альтернативные продукты или варианты блюд, учитывая диетические предпочтения или аллергии, что делает их еще более ценными в современном мире.

С развитием технологий и повышением доступности смартфонов и интернета, приложения для управления покупками становятся неотъемлемой частью повседневной жизни. Приложения, основанные на рецептах, представляют собой логичное дополнение к этому тренду, объединяя функциональность планирования питания с возможностью удобного составления списков покупок. Исследование и разработка таких приложений имеет большое значение в контексте современного образа жизни, подчеркивая важность инноваций в области управления ресурсами и оптимизации повседневных процессов.

* 1. Обзор аналогов

Для создания web-приложения с привлекательным пользовательским интерфейсом и удобной функциональностью необходимо проанализировать аналоги. Анализ конкурентов способствует не только выявлению достоинств и недостатков, но также помогает понять, к чему стоит стремиться при разработке собственного web-приложения и выявить основные потребности пользователей. В результате поиска аналогичных решений с похожей тематикой было выбрано два web-приложения. Анализ будет осуществляться путем оценки приложений по различным критериям, таким как: интерфейс, функциональность, наличие нестандартных подходов и т.д.

* + 1. Приложение «Bring!»

В качестве одного из аналогов было рассмотрено приложение «Bring!» [1]. Главная страница приложения представлена на рисунке 1.1.

«Bring» [1] – это web-приложение, которое предназначено конкретно для создания списков покупок, — оно делает подготовку и сам поход в супермаркет максимально простыми и удобными. Не в последнюю очередь это связано с контекстным указателем в поисковой строке, а также большими пиктограммами записей.

Отдельного внимания также заслуживает отдельный раздел приложения с идеями блюд на завтрак, обед и ужин. Они включают как рецепты, так и списки покупок. Разработчики программы не забыли про возможность создания нескольких каталогов, а также совместного доступа к ним.



Рисунок 1.1 – Главная страница «Bring!»

Программа предоставляет множество действий, например, добавление и удаление продуктов из списка «Дом», который отражает имеющиеся продукты; На рисунке 1.2 изображена вкладка «Идеи». Вкладка предлагает на выбор множество рецептов для приготовления.



Рисунок 1.2 – Вкладка «Идеи»

Приложение имеет богатый функционал, начиная от выбора рецепта и завершая выбором иконки для продукта. Однако, в приложении отсутствует возможность указывать единицы измерения продуктов (штуки, килограммы и т.д.), что следует учесть при разработке своего web-приложения. Так же отсутствует возможность суммировать цену продуктов.

* + 1. Веб-приложение «Listonic»

«Listonic» [2] – web-приложение с минимальным интерфейсом, но богатым функционалом. В отличие от предыдущего аналога, обладает возможностью указания стоимости продуктов (но только в долларах, что ограничивает круг пользователей приложения), добавлять свои продукты в список, указывать им иконку, сортировать список по алфавиту. Так же есть возможность подсчитать итоговую стоимость продуктов.

Глядя на интерфейс веб-приложения «Listonic» ощущается философия минимализма. У пользователя есть минимальный набор кнопок и, соответственно действий, что исключает возможность запутаться в них. В связи с тем, что интерфейс не перегружен, пользователь быстро справляется со своей задачей и не проявляет негативных эмоций в сторону приложения. Однако, такое по вкусу не каждому.

Главная страница показана на рисунке 1.3.

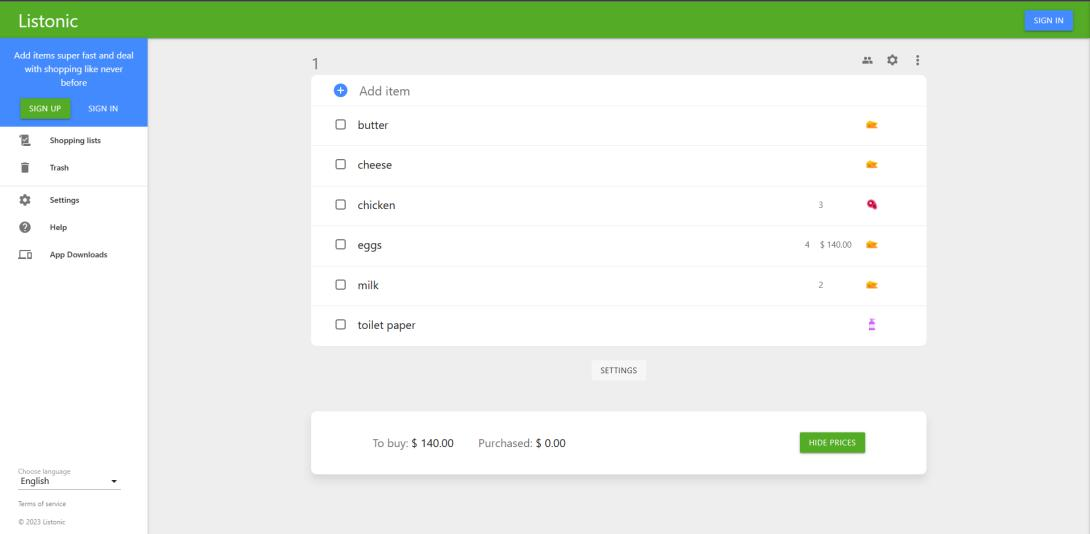


Рисунок 1.3 – Главная страница «Listonic»

Дополнительно в приложении присутствует функция «Корзины». Список после удаления попадает в этот раздел. Корзина изображена на рисунке 1.4.

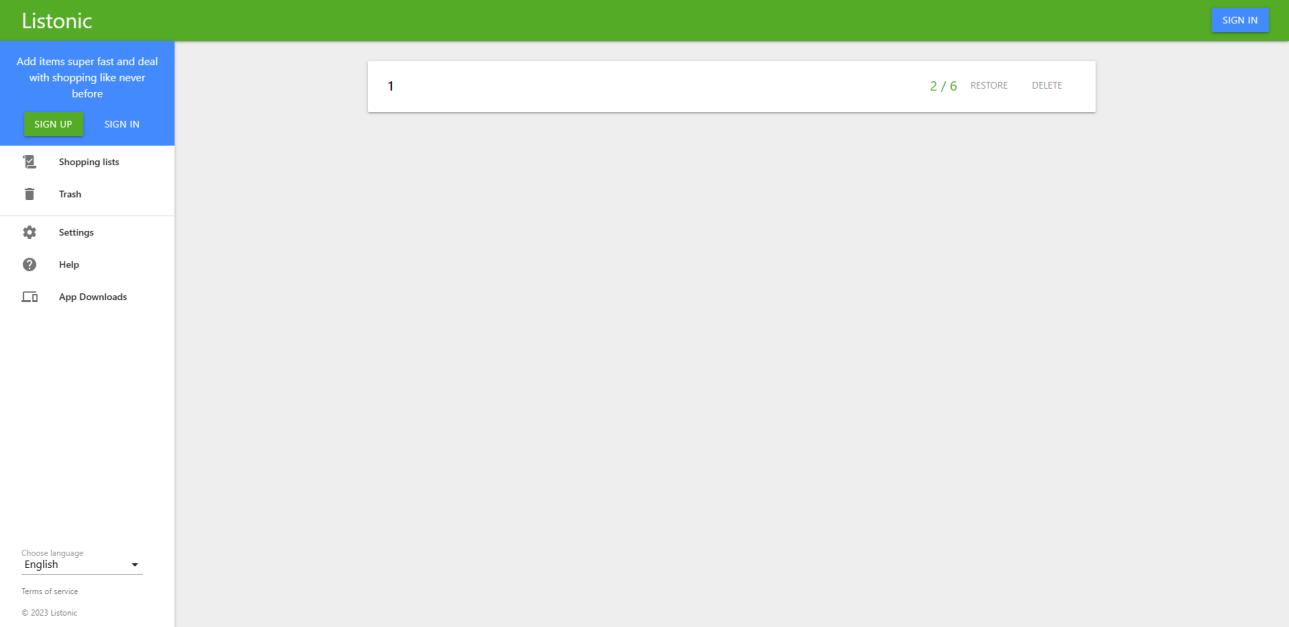


Рисунок 1.4 – Страница корзины приложения «Listonic»

Удалённые списки покупок можно восстановить при необходимости, или же удалить окончательно. Очень удобно, если пользователь передумает удалять список. Из минусов можно отметить, что списки покупок не содержат суммарную стоимость продуктов, из которых они состоят, а та же отсутствует дата подтверждения покупки, что исключает возможность сортировки по датам. Это стоит учесть при создании собственного веб-приложения.

* + 1. Веб-приложение «Russian food»

«Russian food» [3] – приложение-склад рецептов. Здесь любой пользователь может посмотреть неисчисляемое количество различных рецептов на любую тематику, будь то вегетарианские блюда, закуски, горячие блюда, супы и т.д. Несмотря на название, рецепты здесь от различных народов и времён. Часть главного экрана продемонстрирована на рисунке 1.5.

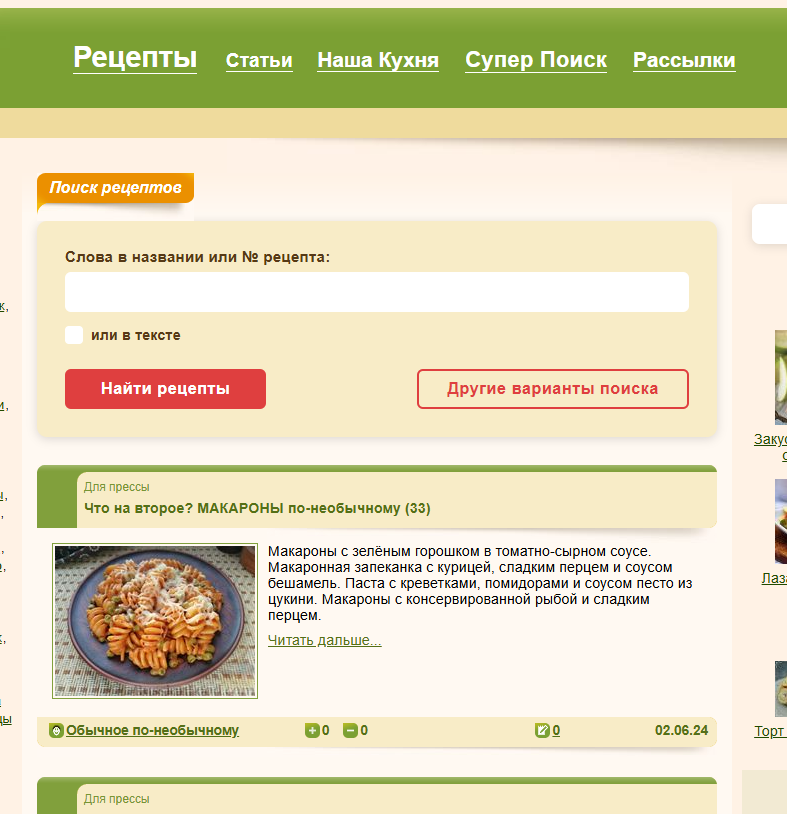


Рисунок 1.5 – Главная страница «Russian food»

На главной странице нас приветствуют сразу несколько областей. Первая – полоса с пунктами меню. В ней выделяется пункт «Супер поиск» (рисунок 1.6). В отличие от поиска на главное странице, этот может искать рецепты по большему числу параметров, таких как ингредиенты, типы блюд, национальность и т.д. Это может стать очень удобным критерием для интернационализации приложения. Чем больше народов привлечено в приложение – тем больше различных блюд и способов их приготовления. Так же поддерживается опция выбора вегетарианских блюд, что очень удобно для людей, которые не могут или не хотят употреблять животное мясо. Так же поддерживается исключение продуктов, если пользователь испытывает непереносимость одного или нескольких компонентов.

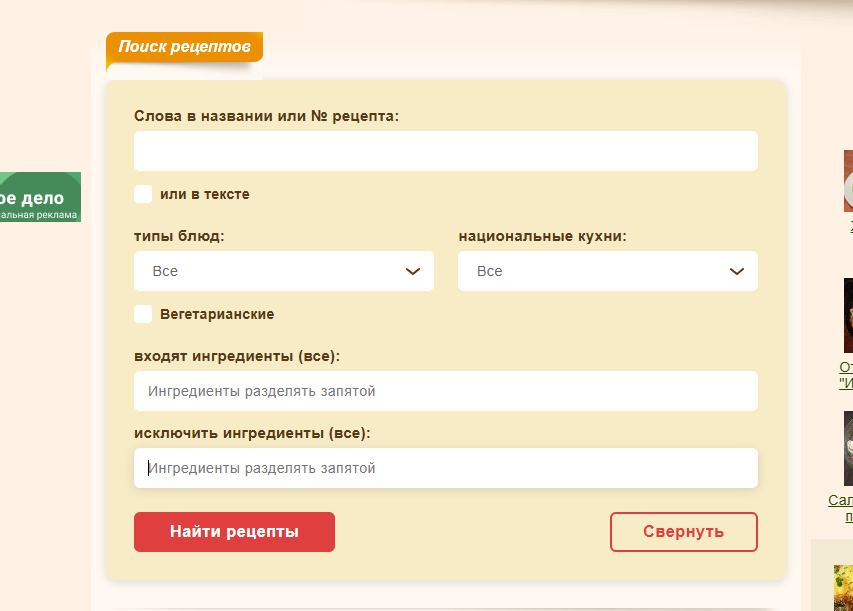


Рисунок 1.6 – «Супер поиск»

Однако, самая интересная область приложения – это область добавления собственных рецептов. Здесь авторизованный пользователь волен создавать собственные рецепты. Часть интерфейса создания показана на рисунке 1.7.

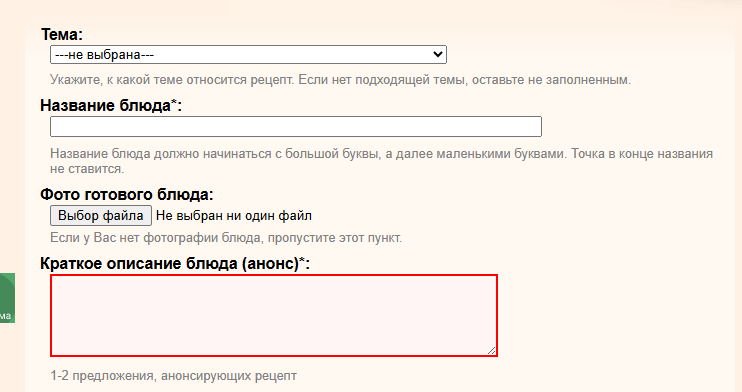


Рисунок 1.7 – Создание собственного рецепта

Здесь очень много удобных опций, таких как выбор темы, в которой это блюдо появится при поиске, загрузка фотографий блюда (в т. ч. и для пошагового приготовления) и прочее.

В целом, приложение «Russian food» очень удобное и практичное, но не имеет возможности отслеживания продуктов, имеющихся у пользователя.

* 1. Постановка задачи

Целью данного дипломного проекта является создание web-приложения для составления списка покупок. Для реализации проекта необходимо установить список задач, выполнение которых будет отображать прогресс:

 предоставлять возможность регистрации и авторизации по логину и

паролю;

 предоставлять роль администратора, модератора и стандартного пользователя;

 предоставлять возможность пользователю управлять содержимым

своего списка продуктов: добавлять, изменять, удалять продукты;

 предоставлять пользователю возможность работы с динамическим чек-

листом покупок: автоматическое добавление и удаление выбранных из

рецептов продуктов с учётом имеющихся в личном списке продуктов;

 предоставлять возможность пользователю добавлять и удалять

продукты в чек-листе;

 предоставлять пользователю возможность подтверждения покупки для

автоматического пополнения списка имеющихся продуктов;

 предоставлять возможность администратору управлять базой рецептов

и продуктов;

 предоставлять возможность администратору управлять аккаунтами

пользователей: добавлять, удалять, восстанавливать.

1.4 Вывод по разделу

Перед тем, как приступить к реализации самого проекта, необходимо определится с целями, задачами и актуальностью самого проекта. Данные разделы позволяют здраво оценить на какие элементы необходимо обратить внимание при реализации той или иной функциональности, удостоверится в современности и конкурентоспособности среди других представителей данного сегмента.

В ходе анализа существующих аналогов, была выявлена целесообразность разработки и определён основной спектр функциональных возможностей, которые необходимо реализовать при разработке данного программного продукта. Также, на основании использования приложений-аналогов были выработаны основные критерии для построения дизайна приложения, позволяющего осуществлять наиболее простое и интуитивное его использование.

В разделах с постановкой и актуальностью задачи были вынесены основные цели, которые необходимо реализовать в приложении, определены основные роли пользователей, использующих приложение, а также обозначены пункты, подтверждающие актуальность разработки приложения.

1. Проектирование приложения

ФИО

Подпись

Дата

Лист

1

БГТУ 02.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е. А.

2 Проектирование приложения

Лит.

Листов

8

*74217076, 2024*

* 1. **Архитектура приложения**

Основная задача дипломного проекта – создание клиент-серверного приложения, предоставляющего возможности для формирования списка покупок на основе рецептов.

Web-приложение представляется клиентской стороной, делегирующей серверу основное выполнение функционала. Таким образом, делая систему разбитой на логический блоки, тем самым упрощая формирование точек связи. Сервером является приложение, расположенное на хостинге для круглосуточного доступа пользователей к данным. Сами данные хранятся в СУБД, с которым связывается сервер для работы с информацией о пользователях. Отдельно хранятся загруженные картинки. Диаграмма развёртывания приложения представлена на рисунке 2.1 и в приложении А.

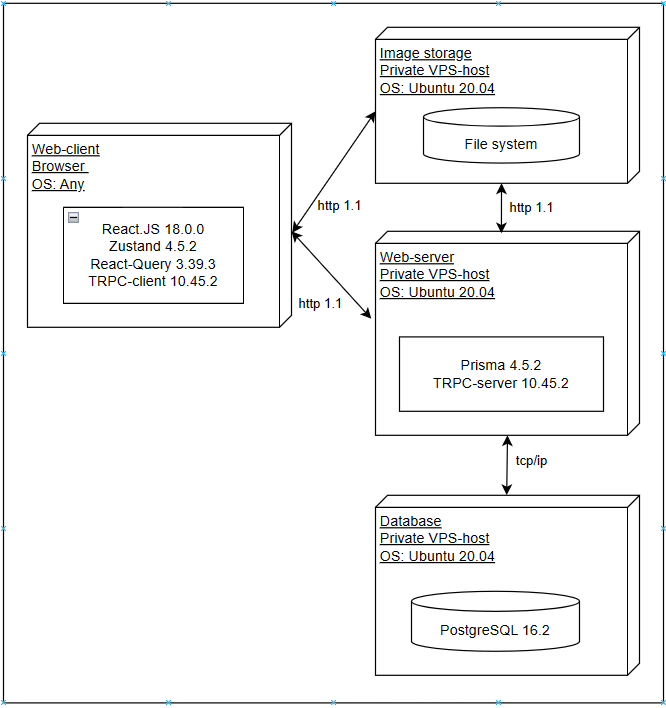


Рисунок 2.1 – Диаграмма развёртывания приложения

* 1. Проектирование базы данных

Перед началом разработки приложения была спроектирована база данных для сервера с 15 таблицами. Список таблиц с их кратким описанием приведён в таблице 2.1. Логическая схема базы данных приведена ниже на рисунке 2.2.

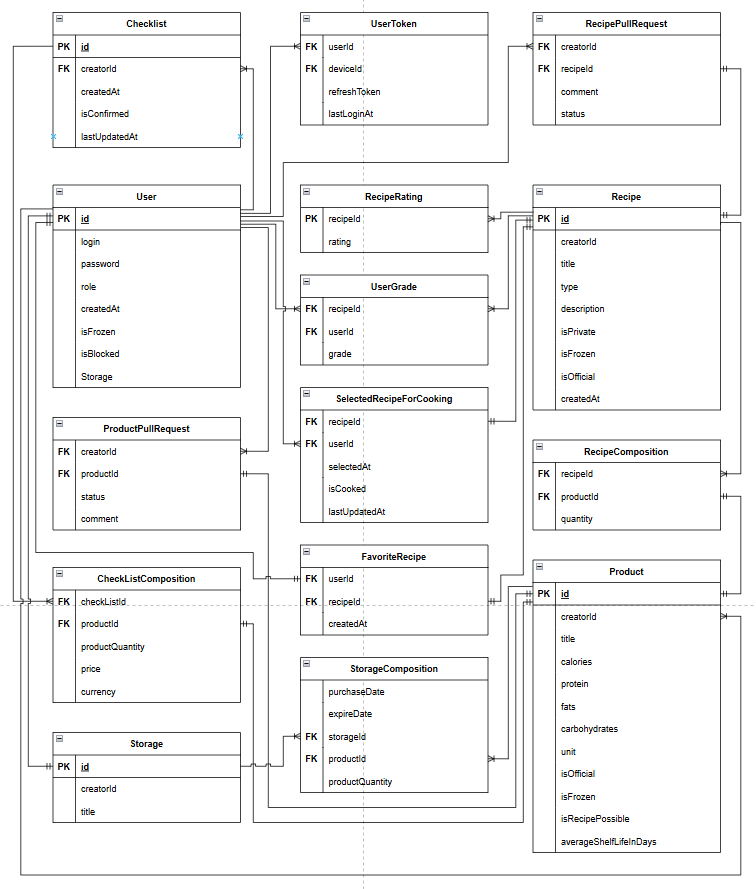


Рисунок 2.2 – Логическая схема базы данных

Обобщённое описание направленности таблиц даёт первичное представление о созданной системе связи данных и является мини-презентацией особенностей структуры. Также данная структура приведена в приложении Б.

Структура представлена пятнадцатью таблицами, представленными в таблице 2.1.

Таблица 2.1 – Описание таблиц

|  |  |
| --- | --- |
| Таблица | Описание |
| Checklist | Список предстоящих покупок |
| UserToken | Информация о сессии пользователя |
| RecipePullRequest | Запрос на получение статуса официального рецепта |
| Recipe | Информация о рецепте |
| RecipeRating | Рейтинг рецепта |
| User | Информация о пользователе |
| UserGrade | Оценки рецептов от разных пользователей |
| SelectedRecipeForCooking | Блюда, которые предстоит приготовить |
| ProductPullRequest | Запрос на получение статуса официального продукта |
| ChecklistComposition | Состав списка покупок |
| FavoriteRecipe | Избранные пользователем рецепты |
| Product | Информация о продукте |
| StorageComposition | Состав пользовательского списка имеющихся продуктов |
| Storage | Информация о пользовательском списке имеющихся продуктов |
| RecipeComposition | Состав рецепта |

Подробное описание содержимого некоторых основных таблиц приведено ниже.

* + 1. Описание таблиц

«Ядро» всей схемы базы данных – таблица User. Она содержит всю информацию об аккаунте пользователя. Описание полей в таблице 2.1.

Таблица 2.2 – Описание таблицы «User»

|  |  |
| --- | --- |
| Поле | Описание |
| login | Логин пользователя |
| password | Хэш пароля пользователя |
| role | Роль пользователя в системе (администратор, модератор, стандартный) |
| createdAt | Дата регистрации пользователя |
| isFrozen | Показывает удален ли аккаунт |
| isBlocked | Показывает запрещено ли пользователю создавать рецепты/продукты |
| id | Идентификатор пользователя |

Таблица Checklist характеризует список предстоящих покупок пользователя. Описание таблицы Checklist находится в таблице 2.3

Таблица 2.3 – Описание таблицы «Checklist»

|  |  |
| --- | --- |
| Поле | Описание |
| creatorId | Идентификатор создателя списка |
| createdAt | Дата создания |
| isConfirmed | Подтверждена ли покупка |
| lastUpdatedAt | Дата последнего обновления списка в базу |

Таблица ChecklistComposition Является продолжением таблицы Checklist. Она характеризует состав списка предстоящих покупок. Описание таблицы ChecklistComposition находится в таблице 2.4

Таблица 2.4 – Описание таблицы «ChecklistComposition»

|  |  |
| --- | --- |
| Поле | Описание |
| checklistId | Идентификатор списка покупок |
| productId | Идентификатор продукта |
| productQuantity | Количество продукта |
| price | Цена продукта |
| currency | Валюта, за которую приобретен продукт (BYN, USD, RUB) |

Таблица ProductPullRequest описывает попытку пользователя сделать созданный им продукт официальным, то есть признанным администраторами как корректный. «Официальный» значит, что пользователи могут не беспокоиться за качество продукта. Описание таблицы ProductPullRequest находится в таблице 2.5

Таблица 2.5 – Описание таблицы «ProductPullRequest»

|  |  |
| --- | --- |
| Поле | Описание |
| creatorId | Идентификатор создателя |
| productId | Идентификатор продукта |
| status | Статус запроса (Подтверждён, отклонён, ожидание) |
| comment | Комментарий от администраторов |

Таблица Storage описывает информацию о хранилище пользователя. У одного пользователя может быть только одно хранилище. Хранилище может быть только у обычного пользователя, администрация не может иметь хранилище в своем аккаунте. Описание таблицы Storage находится в таблице 2.6

Таблица 2.6 – Описание таблицы «Storage»

|  |  |
| --- | --- |
| Поле | Описание |
| id | Идентификатор хранилища |
| creatorId | Идентификатор пользователя, к которому привязано хранилище |
| title | Название хранилище (опционально) |

Таблица StorageComposition является продолжение Storage. Она описывает все продукты и их характеристики, имеющиеся на счету у пользователя. Описание таблицы StorageComposition находится в таблице 2.7

Таблица 2.7 – Описание таблицы «StorageComposition»

|  |  |
| --- | --- |
| Поле | Описание |
| purchaseDate | Дата приобретения продукта |
| expireDate | Срок годности продукта (дата до порчи) |
| storageId | Идентификатор хранилища |
| productId | Идентификатор продукта |
| productQuantity | Количество продукта (грамм, штук и т.д.) |

Таблица FavoriteRecipe описывает рецепты, которые пользователь посчитал достойными того, чтобы иметь к ним быстрый доступ и готовить чаще, чем другие рецепты. Описание таблицы FavoriteRecipe находится в таблице 2.8

Таблица 2.8 – Описание таблицы «FavoriteRecipe»

|  |  |
| --- | --- |
| Поле | Описание |
| userId | Идентификатор пользователя |
| recipeId | Идентификатор рецепта |
| createdAt | Дата создания |

Таблица SelectedRecipeForCooking описывает рецепты, которые пользователь выбрал для того, чтобы приготовить их в самое ближайшее время. Описание таблицы SelectedRecipeForCooking находится в таблице 2.9

Таблица 2.9 – Описание таблицы «SelectedRecipeForCooking»

|  |  |
| --- | --- |
| Поле | Описание |
| recipeId | Идентификатор рецепта |
| userId | Идентификатор пользователя |
| selectedAt | Дата выбора |
| isCooked | Приготовлен ли выбранный рецепт |
| lastUpdatedAt | Дата последнего обновления |

Таблица UserGrade хранит все пользовательские оценки рецептов. Описание таблицы UserGrade находится в таблице 2.10

Таблица 2.10 – Описание таблицы «UserGrade»

|  |  |
| --- | --- |
| Поле | Описание |
| recipeId | Идентификатор рецепта |
| userId | Идентификатор пользователя |
| grade | Оценка |

Таблица RecipeRating характеризует рейтинг каждого рецепта. Рейтинг высчитывается исходя из оценок пользователей. Описание таблицы RecipeRating находится в таблице 2.11

Таблица 2.11 – Описание таблицы «RecipeRating»

|  |  |
| --- | --- |
| Поле | Описание |
| recipeId | Идентификатор рецепта |
| rating | Рейтинг рецепта |

Таблица UserToken хранит информацию о сессии пользователя. Благодаря этой таблице поддерживается вход с различных пользовательских устройств. Описание таблицы UserToken находится в таблице 2.12

Таблица 2.12 – Описание таблицы «UserToken»

|  |  |
| --- | --- |
| Поле | Описание |
| userId | Идентификатор пользователя |
| deviceId | Идентификатор устройства |
| refreshToken | Токен обновления |
| lastLoginAt | Дата последнего обновления записи |

Таблица RecipePullRequest аналогична таблице ProductPullRequest, но характеризует запрос на добавление рецепта. Описание таблицы RecipePullRequest находится в таблице 2.13

Таблица 2.13 – Описание таблицы «RecipePullRequest»

|  |  |
| --- | --- |
| Поле | Описание |
| creatorId | Идентификатор создателя |
| recipeId | Идентификатор рецепта |
| status | Статус |
| comment | Комментарий администрации |

Таблица Recipe содержит общую информацию о рецепте, такую как название, тип, описание и прочее. Описание таблицы Recipe находится в таблице 2.14

Таблица 2.14 – Описание таблицы «Recipe»

|  |  |
| --- | --- |
| Поле | Описание |
| id | Идентификатор рецепта |
| creatorId | Идентификатор создателя |
| title | Название рецепта |
| type | Тип рецепта |
| description | Описание рецепта |
| isPrivate | Характеризует видимость рецепта |
| isFrozen | Характеризует удаление рецепта |
| isOfficial | Показывает, подтверждён ли рецепт администрацией |
| createdAt | Дата создания |

Таблица RecipeComposition характеризует состав рецепта. Описание таблицы RecipeComposition находится в таблице 2.15

Таблица 2.15 – Описание таблицы «RecipeComposition»

|  |  |
| --- | --- |
| Поле | Описание |
| recipeId | Идентификатор рецепта. |
| productId | Идентификатор продукта |
| quantity | Количество продукта |

Таблица Product содержит полную информацию о продукте. Описание таблицы Product находится в таблице 2.16

Таблица 2.16 – Описание таблицы «Product»

|  |  |
| --- | --- |
| Поле | Описание |
| id | Идентификатор продукта |
| creatorId | Идентификатор создателя |
| title | Название продукта |
| calories | Количество калорий |
| protein | Количество белка |
| fats | Количество жиров |
| carbohydrates | Количество углеводов |
| unit | Размерность продукта (граммы, штуки и т.д.) |
| isOfficial | Подтвержден ли продукт администрацией |
| isFrozen | Характеризует удаление продукта |
| isRecipePossible | Характеризует возможность воссоздания продукта (является ли продукт базовым) |
| averageShelfLifeInDays | Среднее время годности продукта |

* 1. Диаграмма вариантов использования

Диаграмма вариантов использования показывает все возможные варианты использования приложения и взаимодействия пользователей с различными ролями. Данная диаграмма для проекта представлена на рисунках 2.3 и 2.4.

В приложении присутствуют 3 основные роли:

* администратор;
* пользователь;
* модератор.

Жизнь приложения начинается с одного встроенного администратора, которые, впоследствии, может создать других администраторов, модераторов и обычных пользователей. Администратор вправе управлять пользователями (удалять, восстанавливать, создавать, назначать роль). Так же администратор имеет возможность создавать и удалять продукты и рецепты. Все изменения, внесённые администратором, будут отображаться у всех остальных пользователей.

Модератор имеет те же привилегии, что и администратор. Однако, модератор лишён способности создавать и удалять пользователей. Для манипуляции с пользователями модератор обязан связаться с одним из администраторов. Модератор способен изменять информацию о продуктах и рецептах.

У роли модератора и администратора своя экосистема в приложении. Две эти роли имеют больше привилегий, чем обычные пользователи, однако, в отличии от последних, не могу иметь и заполнять свои «хранилища» и помещать рецепты в избранное или в секцию «для приготовления». Выставление оценок рецептам этим ролям так же недоступна, для обеспечения честности и справедливости рейтинга рецептов.

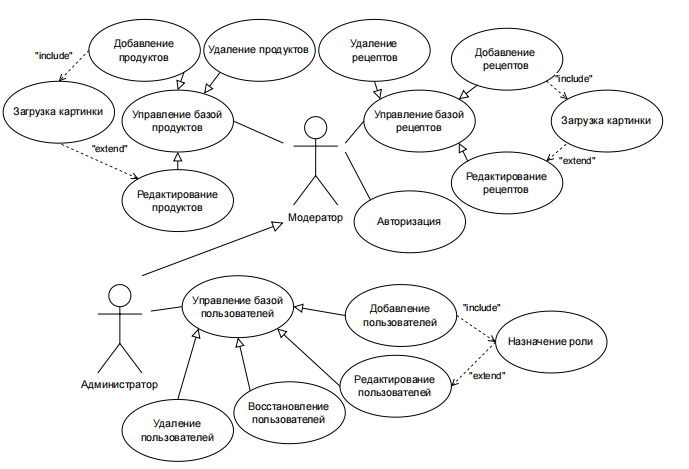


Рисунок 2.3 – Диаграмма вариантов использования роли администратор/модератор

Пользователь вправе зарегистрироваться в приложении или быть созданным администратором. Пользователь имеет доступ к рецептам и продуктам, а также к своему хранилищу продуктов. Пользователь имеет возможность выбирать рецепты, после чего продукты из выбранных рецептов попадут в чек-лист пользователя.

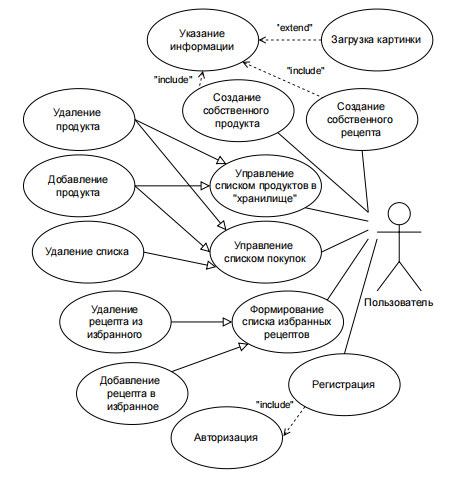


Рисунок 2.4 – Диаграмма вариантов использования обычного пользователя

Таким образом данная диаграмма демонстрирует прямую взаимосвязь всех трёх компонентов, где без проявления одного не может идти выполнение другого.

* 1. **Проектирование основных алгоритмов**

Два основных алгоритма в приложении – добавление продукта в хранилище и формирование списка покупок. Они находятся в приложениях В и Г соответственно.

Алгоритм добавления продукта в хранилище описывает то, как продукт попадает в хранилище пользователю.

Алгоритм формирования списка покупок учитывает те продукты, которые уже имеются в хранилище пользователя. То есть, если продукта нету в хранилище, – он добавляется в указанном количестве. Иначе, указанное количество вычитается из имеющегося, и разница между ними записывается в базу данных.

* 1. **Выбор средств реализации**

Для реализации проекта необходимо, первым делом, построить полную структуру проекта и выбрать платформу разработки. Для оптимизации временных затрат, в качестве экосистемы и языка программирования был выбран TypeScript (надмножество языка JavaScript), в силу строгой типизаций.

Структура проекта стоит из трёх компонентов:

* серверная часть;
* клиентская часть (UI);
* централизованная база данных.
  + 1. **Серверная часть**

Для реализации серверной части был выбран Bun.js с использованием фреймворка Elysia для упрощения разработки. Она отвечает за обработку запросов, работу с базой данных и логику приложения.

Bun.js (Bun) – это платформа с открытым исходным кодом для работы с языком [JavaScript](https://blog.skillfactory.ru/glossary/javascript/" \t "https://blog.skillfactory.ru/glossary/node-js/_blank). Она позволяет писать серверный код для веб-приложений и динамических веб-страниц, а также программ командной строки. В основе платформы – событийно-управляемая модель с неблокирующими операциями ввода-вывода, что делает ее эффективной и легкой.

Веб-фреймворк Elysia (Bun.js/TypeScript) представляет собой популярный веб-фреймворк, написанный на JavaScript и работающий внутри среды исполнения Bun.js. Этот модуль освещает некоторые ключевые преимущества этого быстрого фреймворка, установку среды разработки и выполнение основных задач веб-разработки и развёртывания.

Elysia – альтернатива популярному фреймворку ExpressJS. Так как это надстройка над API среды Bun, Elysia работает намного намного быстрее. Разработчики обещаю прирост в скорости ответа от сервера в десятки раз, по сравнению с конкурентом. Так же строгая типизация и отсутствие нужды описывать каждую конечную точку для клиента сильно облегчает работу разработчиков. Однако, стоит отметить минус, что для этого требуется так называемый монорепозиторий или отдельная генерация типов.

Во многи тестах среда выполнения Bun показывает себя в разы лучше, чем устаревший NodeJS. Это связано с тем, что Bun использует новый подход и инструменты, а так же некоторые системные API, которые ускоряют работу с вводом-выводом файлов. При этом, разработчикам удалось практически полную совместимость со средой NodeJS, что кратно увеличивает рейтинг Bun среди других сред исполнения.

Хостом для процесса сервера станет система на операционной системе GNU/Linux, так как среда Bun максимально оптимизирована под использование API именно систем на базе ядра Linux.

* + 1. **Клиентская часть**

Для реализации клиентской части выбрана библиотека React, отвечающая за создание пользовательского интерфейса.

React (React.js или ReactJS) – это бесплатная библиотека JavaScript с открытым исходным кодом.

К преимуществам этой библиотеки относится компонентный подход, Virtual DOM, мемоизация и ленивая загрузка компонентов. Несмотря на свой богатый функционал, React всё ещё остается библиотекой, что позволяет его использовать в качестве основы для различных фреймворков.

* + 1. Централизованная база данных

Хранение данных будет реализовываться с помощью PostgreSql. Так как она является одним из нескольких бесплатных популярных вариантов СУБД, часто используется для ведения баз данных веб-сайтов. Это была одна из первых разработанных систем управления базами данных, поэтому в настоящее время она хорошо развита, и позволяет пользователям управлять как структурированными, так и неструктурированными данными. Может быть использован на большинстве основных платформ, включая Linux. Прекрасно справляется с задачами импорта информации из других типов баз данных с помощью собственного инструментария.

* 1. Выводы по разделу

В данной главе была разработана диаграмма вариантов использования мобильного приложения, которая позволила составить более целостное понимание всех функциональных возможностей проекта.

Была спроектирована база данных, где определены необходимые таблицы, поля для каждой из них, типы данных и ограничения целостности. Также были установлены связи между таблицами, которые описаны на схеме базы данных.

Помимо этого, были представлены основные диаграммы, полно определяющие задумку проекта, что позволяет быстро и наглядно вникнуть в суть продукта.

Подобранный стек технологий полностью покрывает нужды для реализации проекта.

1. Разработка приложения

ФИО

Подпись

Дата

Лист

1

БГТУ 03.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е. А.

3 Разработка приложения

Лит.

Листов

15

*74217076, 2024*

* 1. Реализация серверной части

Построение правильной архитектуры проекта – важная часть разработки. Правильная структура ускоряет процесс разработки, что, в свою очередь, обеспечивает наименьшую энтропию функций и файлов (меньше мусора в проекте), наилучший уровень абстракции и максимальную производительность.

При разработке серверной части курсового проекта была создана структура, представленная на рисунке 3.1.

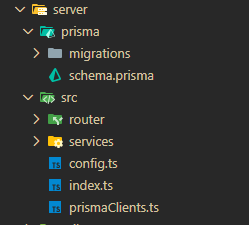


Рисунок 3.1 – Структура сервера

Данная структура достаточна для реализации всех нужд проекта. Вот её краткое описание:

Каждая из указанных папок играет ключевую роль в структуре и функционировании серверной части проекта, обеспечивая определённые функциональные возможности и обработку данных, необходимых для корректной работы приложения.

Логически, архитектура сервера распределена на две основные части: взаимодействие с базой данных и основная бизнес-логика сервера.

Так как в проекте решено использовать технологию ORM и её реализацию в виде Prisma ORM (ORM для JavaScript), для нее используется отдельная папка prisma. В ней находятся два основных компонента: файл схемы и папка с миграциями. Для описания таблиц в системе Prisma используется свой язык, слабо похожий на JavaScript.

Пример описания модели приведен на рисунке 3.2

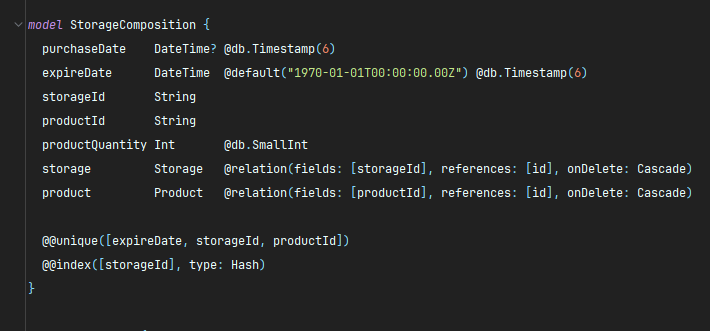


Рисунок 3.2 – Пример модели

Синтаксис удобен и достаточно понятен. Возможности движка Prisma позволяют задать все необходимые ограничения для базы данных и таблиц, такие как уникальные поля, отношения, поведения при удалении, индексация и прочее.

В папке с миграциями находятся датированные миграции, что позволяет читать историю и сохранять последовательность миграций. Так же, в случае ошибки, можно вернуться к предыдущей точке сохранения. Каждая миграция представляет собой файл типа sql, в котором описаны все изменения. Пример на рисунке 3.3

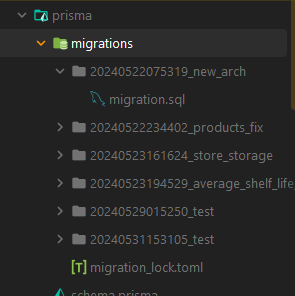


Рисунок 3.3 – История миграций проекта

Файл migration\_lock.toml хранит информацию о провайдере базы данных.

Основная логика сервера находится в папке src. Именно здесь находятся маршрутизаторы и сервисы. Для каждого сервиса существует свой файл, описывающий функционал и логику взаимодействия с бд. Структура на рисунке 3.4

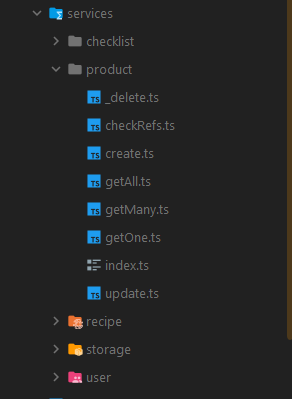


Рисунок 3.4 – Структура сервисов

Весь маршрутизатор реализован в папке router. Здесь определены все конечные точки приложения, а так же настраивается валидация параметров запроса. Структура маршрутизатора показана на рисунке 3.5.

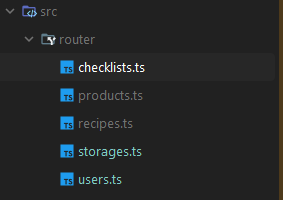


Рисунок 3.5 – Структура маршрутизатора

Под каждую сущность в проекте так же отведён свой файл с названием, соответствующим сущности.

Маршрутизатор распределён по разным файлам. Для объединения точек используется файл index. Он же является точкой входа в приложение (рисунок 3.6).

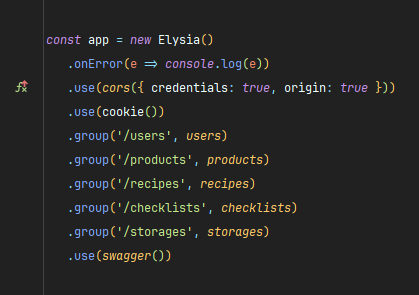


Рисунок 3.6 – Точка входа в приложение

Контроллеров в проекте нету как таковых, так как эту задачу берет на себя фреймворк.

Файл конфигурации сервера находится в файле config. Вид на рисунке 3.7

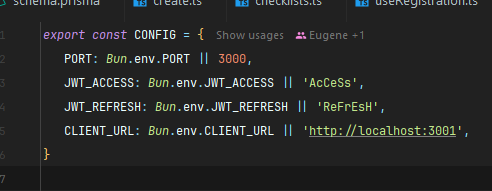


Рисунок 3.7 – Клиент базы данных и его опции

Настройка клиента базы данных находится в файле prismaClient. Клиент показан на рисунке 3.8

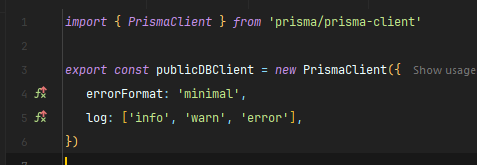


Рисунок 3.8 – Клиент базы данных и его опции

Так же в опциях указаны параметры вывода ошибок и логирование.

PrismaClient является мостом между базой данных и бизнес-логикой. JavaScript-код обращается к API Prisma через него, что гарантирует безопасность и оптимизацию запросов к базе данных. Благодаря типизации разработчик сразу видит ожидаемый результат, что снижает вероятность ошибки доступа к несуществующим данным к минимуму. Клиент автоматически конвертирует методы JavaScript в sql.

* 1. Разработка клиентской части

Как уже описывалось ранее, для реализации клиентской части была выбрана прогрессивная библиотека React.js. React – это JavaScript-библиотека для создания пользовательских интерфейсов. Проектирование с помощью React подразумевает создание компонентов, взаимодействующих друг с другом. Главным компонентом, с которого начинается работа приложения, является App.js. Структуру проекта можно увидеть на рисунке 3.9.

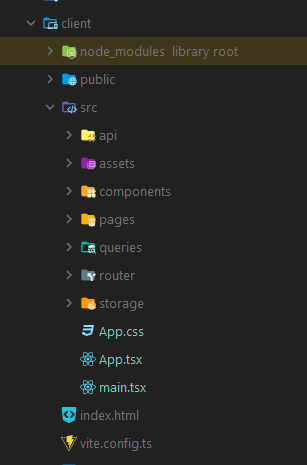


Рисунок 3.9 – Структура клиентской части

Папка components содержит внутри компоненты, которые представляют собой маленькие части пользовательского интерфейса приложения. Компоненты могут быть переиспользуемыми и содержать в себе логику и представление для отдельных элементов интерфейса. В компонентах имеется разделение на две папки, в которых содержатся модальные окна как для админа, так и для клиента.

Структура папки components изображена на рисунке 3.10

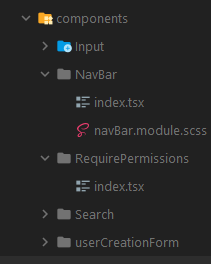


Рисунок 3.10 – Структура components

Пример компонента из папки components показан на рисунке 3.11.

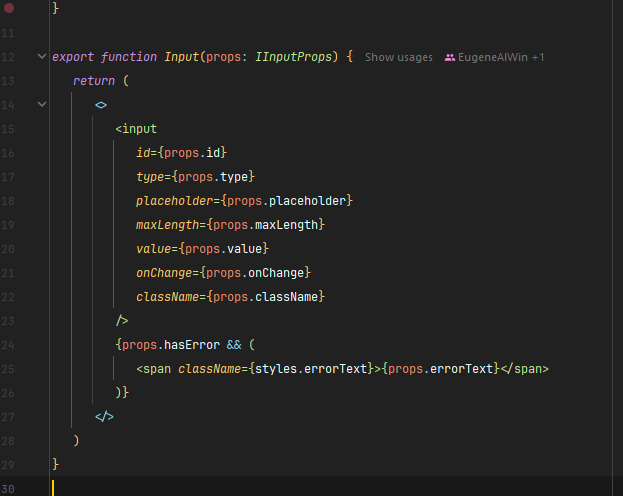


Рисунок 3.11 – Компонент Input

К каждому компоненту привязан свой файл-модуль со стилями. Концепция модульности стилей позволяет забыть про архитектуру БЭМ и подобные ей, так как сборщик автоматически генерирует уникальные стили для каждого файла компонентов, что исключает повторяемость и конфликт стилей.

Папка queries содержит объект treaty (аналог fetch) и дополнительные объекты для обращения к серверу. Структура папки на рисунке 3.12.

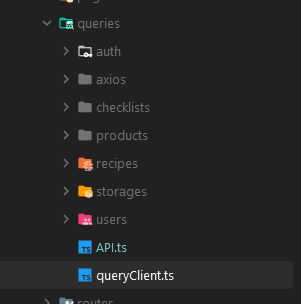


Рисунок 3.12 – Структура queries

Папка storage содержит всю логику по управлению глобальным состоянием приложения. Структура хранилища состояний показана на рисунке 3.13.

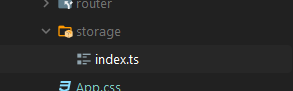


Рисунок 3.13 – Структура storage

Pages – папка, содержащая страницы приложения. Страницы собираются из компонентов. Структура pages показана на рисунке 3.14

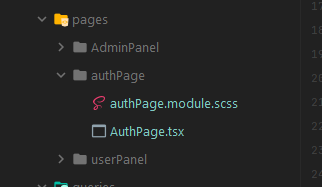


Рисунок 3.14 – Структура pages

Страницы разделены на интерфейс администратора, пользователя и страницу авторизации.

Assets хранит файлы, такие как картинки, которые динамически импортируются. В отличие assets, файлы в папке public являются общедоступными из любой точки приложения и при компиляции проекта не изменяют своей структуры и названия.

* 1. Контейнеризация

Самым удобным способом, благодаря которому можно осуществить развертывание приложения – это Docker. Docker позволяет создавать контейнеры, в которых приложении могут существовать независимо от других приложений.

Чтобы развернуть клиент-серверное приложение на React и Node.js в Docker, требуется создать Docker-контейнеры для каждой части приложения (клиентской и серверной) и настроить их работу вместе. Добавленый Dockerfile для клиентской части и серверной представлен на рисунке 3.15.

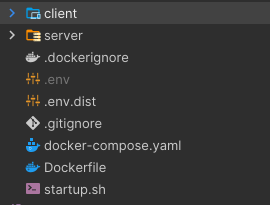


Рисунок 3.15 – Добавление Dockerfile для клиента и сервера

Код Dockerfile представлен в листинге 3.1.

|  |
| --- |
| FROM node:21-alpine3.18  WORKDIR /app  COPY . .  WORKDIR /app/server  RUN npm i  RUN npx prisma generate  RUN npm run compile  WORKDIR /app/client  RUN npm i  WORKDIR /app  RUN chmod +x startup.sh  CMD ["/bin/sh", "startup.sh" ] |

Листинг 3.1 – Dockerfile клиентской части проекта

Затем необходимо создать Docker-образ из Dockerfile в текущем рабочем каталоге и присваивания ему тег.

Создание образа представлено на рисунке 3.16.

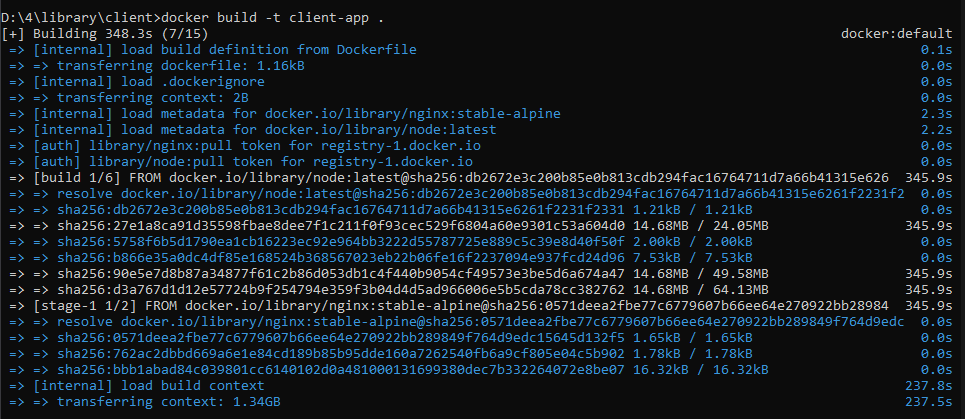


Рисунок 3.16 – Создание образа из Dockerfile

В листинге 3.2 приведен код startup.sh, выполняющийся после запуска контейнера.

|  |
| --- |
| #!/bin/sh  cd /app/server || exit  npm run apply\_migrations  npm start &  cd /app/client || exit  npm run build  npm run preview |

Листинг 3.2 – Листинг startup.sh

На данном этапе были созданы все необходимые docker-файлы, которые нужны для создания и запуска контейнеров Docker. Что бы не запускать все части проекта отдельно, был создан файл docker-compose.yaml. В нем будет происходить процесс создания сервисов, настройка их окружения, конфигурационных файлов и так далее. Как правило, к docker-compose файлу привязаны переменные среды, например, находящиеся в файле .env (environment). Указание этих переменных позволяет запускать экземляры контейнеров с различными параметрами. Правильно написанный docker-compose исключает его последующе редактирование, так как управление может производится лишь сменой переменных среды. Содержимое данного файла представлено в листинге 3.3.

|  |
| --- |
| services:  web:  restart: always  build:  context: ./  dockerfile: Dockerfile  ports:  - '3000:3000'  - '3001:3001'  networks:  - mynetwork  depends\_on:  - db  db:  image: postgres  restart: always  environment:  POSTGRES\_USER: '${USER}'  POSTGRES\_PASSWORD: '${PASSWORD}'  POSTGRES\_DB: '${DATABASE}'  ports:  - '5432:5432'  volumes:  - db\_data:/var/lib/postgresql/db\_data  networks:  - mynetwork  adminer:  image: adminer  restart: always  ports:  - '8080:8080'  networks:  - mynetwork  volumes:  db\_data:  networks:  mynetwork:  driver: bridge |

Листинг 3.3 – docker-compose.yaml

Затем Docker Compose читает конфигурацию из файла docker-compose.yaml, создает и запускает все контейнеры для сервисов, указанных в этом файле, и после этого отдает управление командной строке обратно, если указан соответствующий параметр. После создания контейнера доступна вся сводка о нем, журнал действий, данные о системе и прочее. Так же присутствует возможность интерактивного взаимодействия, если необходимого функционала недостаточно или требуется ручное взаимодействие.

Созданные контейнеры представлены на рисунке 3.17.

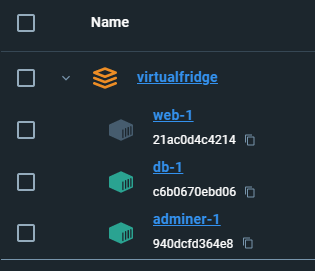


Рисунок 3.17 – Контейнеры

Использование Docker позволило эффективно изолировать приложение и его зависимости, обеспечивая повышенную масштабируемость и упрощая процесс разработки. Применение инструментов Bun.js и React в контейнерах дало возможность создать динамический пользовательский интерфейс.

* 1. Вывод по разделу

В данном разделе были подробно описаны этапы проектирования программного средства. Сперва были выбраны средства для реализации программного средства.

Для реализации серверной части приложения был использован Bun.js вместе с фреймворком Elysia.js для создания веб-приложения.

Для разработки клиентской части приложения была выбрана библиотека React. Этот инструмент позволяет строить динамичные интерфейсы, обладает высокой гибкостью и удобством разработки.

Для хранения данных была выбрана СУБД Postgresgl, так как известна своей надежностью и стабильностью. Она имеет долгую историю разработки, активное сообщество разработчиков и пользователей, а также широко используется в различных проектах, что говорит о ее надежности.

Также была сформулирована архитектура взаимодействия всех 3 частей программного средства на основе трехуровневой архитектуры.

После описания технологий была сформирована база данных и таблицы, из которых она состоит. Таблицы связаны в основном между собой связями один-ко-многим и один-к-одному.

Дополнительно были изучены варианты использования и построена соответствующая диаграмма, а также построены блок-схемы алгоритма вычета продуктов из списка покупок и составления общего списка покупок. Для удобства разработки и развёртывания приложения использовался Docker, который позволяет упаковать приложение со всеми зависимостями в контейнер, обеспечивая переносимость и удобство управления окружением приложения.

1. Тестирование приложения

ФИО

Подпись

Дата

Лист

1

БГТУ 04.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е. А.

4 Тестирование приложения

Лит.

Листов

4

*74217076, 2024*

Тестирование веб-приложений – это комплекс определённых услуг, который может включать в себя различные виды тестирования программного обеспечения.

Основная цель любого тестирования, в том числе и тестирования веб-приложений, – обнаружить все ошибки в программном обеспечении и разработать рекомендации по их предотвращению в будущем.

Одним из видов тестирования, которое затрагивает все возможные этапы при использовании ПО, является регрессионное тестирование. Данное тестирование предполагает тестирование любых способов взаимодействия с приложением любыми возможными методами.

Сперва необходимо протестировать форму регистрации и авторизации. Форма авторизации представлена на рисунке 4.1.

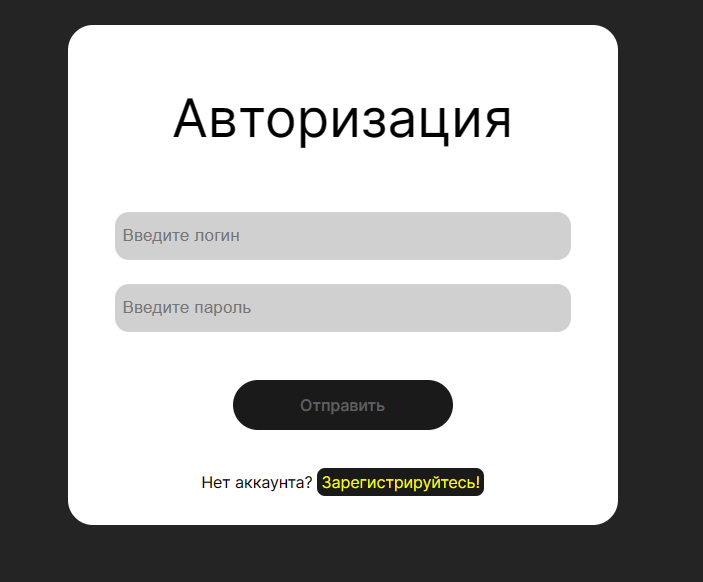


Рисунок 4.1 – Форма авторизации

В данном случае, форма регистрации находится в том же месте и на той же странице, что и форма авторизации. Для переключения режима необходимо нажать соответствующую кнопку.

Для каждой типа входа предусмотрена валидация полей как на клиентской части, так и на серверной. Даже при использовании стороннего клиента, сервер не пропусти невалидные данные.

Форма регистрации представлена на рисунке 4.2.

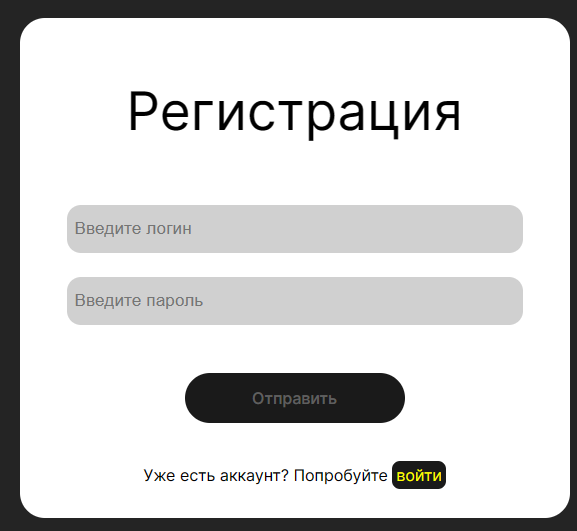


Рисунок 4.2 – Форма регистрации

Формы проверяют введённые в них данные при попытке отправки формы. По полям Login и Password стоит правило обязательного поля (рисунок 4.3).

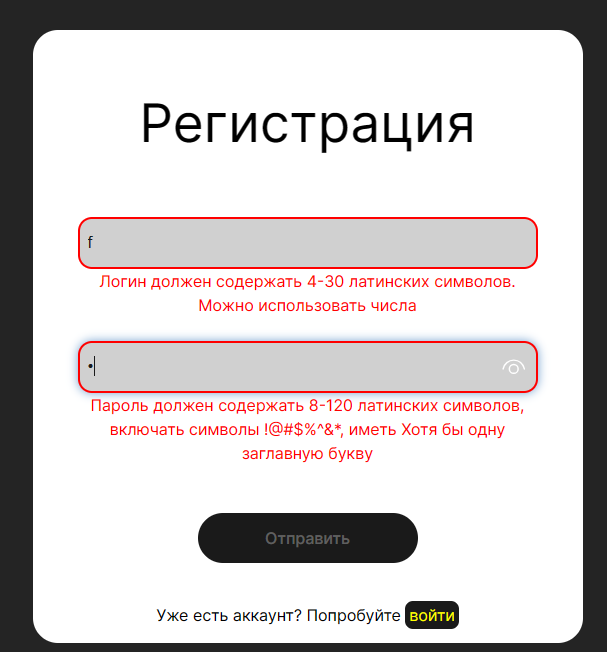


Рисунок 4.3 – Валидация обязательных полей

Для того, чтобы пользователь однозначно понимал, что не так с полями, предусмотрены подписи под каждым пользователем. Подпись содержит все необходимые требования к тексту в полях ввода. При попытке входа пользователя, который не был зарегистрирован, также появится предупреждающее сообщение (рисунок 4.4). Это однозначно указывает пользователю именно на его ошибку, а не ошибку сервера.

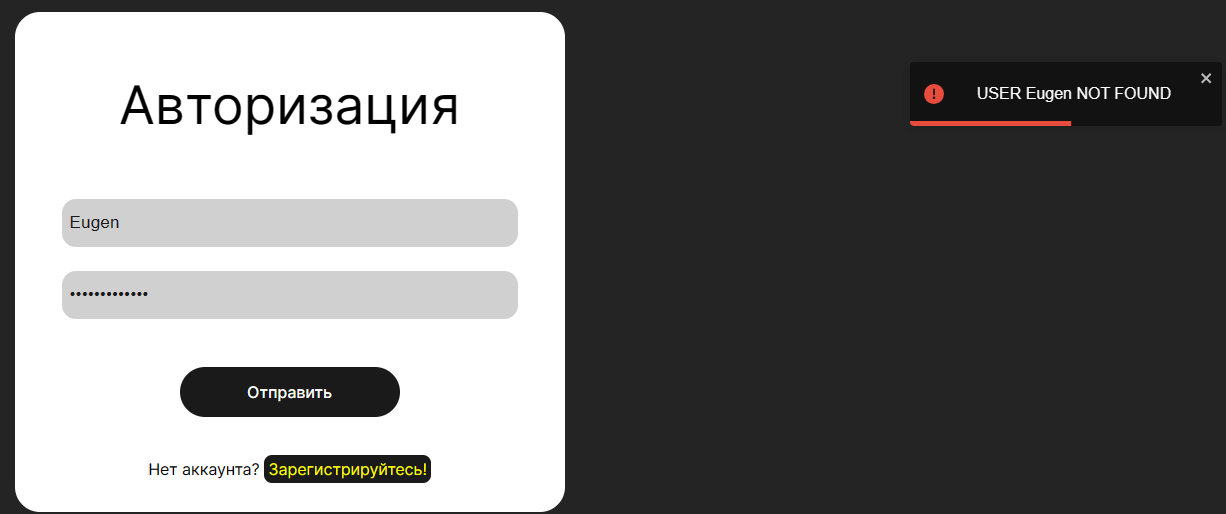


Рисунок 4.4 – Сообщение о незарегистрированном пользователе

Протестируем создание пользователя с учётной записи администратора. При попытке ввести некорректные данные поля горят красным цветом, а при наведении на них можно наблюдать всплывающее сообщение с подсказкой. Пример валидации данных изображён на рисунке 4.5. Валидация логина и пароля при создании пользователя позволяет избежать ситуации, когда пользователь создан администратором некорректно, а сам пользователь не имеет возможности войти в систему из-за ошибка в форме авторизации.

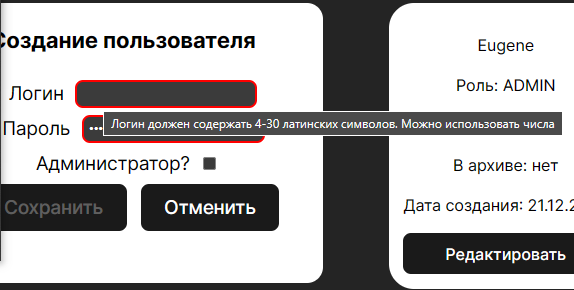


Рисунок 4.5 – Форма создания пользователя

При попытке создать пользователя, который существует, появится уведомление о том, что логин уже занят. Уникальность логина важна для однозначной идентификации пользователя. Ограничение прописано на уровне базы данных, так что даже обойдя валидацию на клиенте, ограничение целостности не будет нарушено.

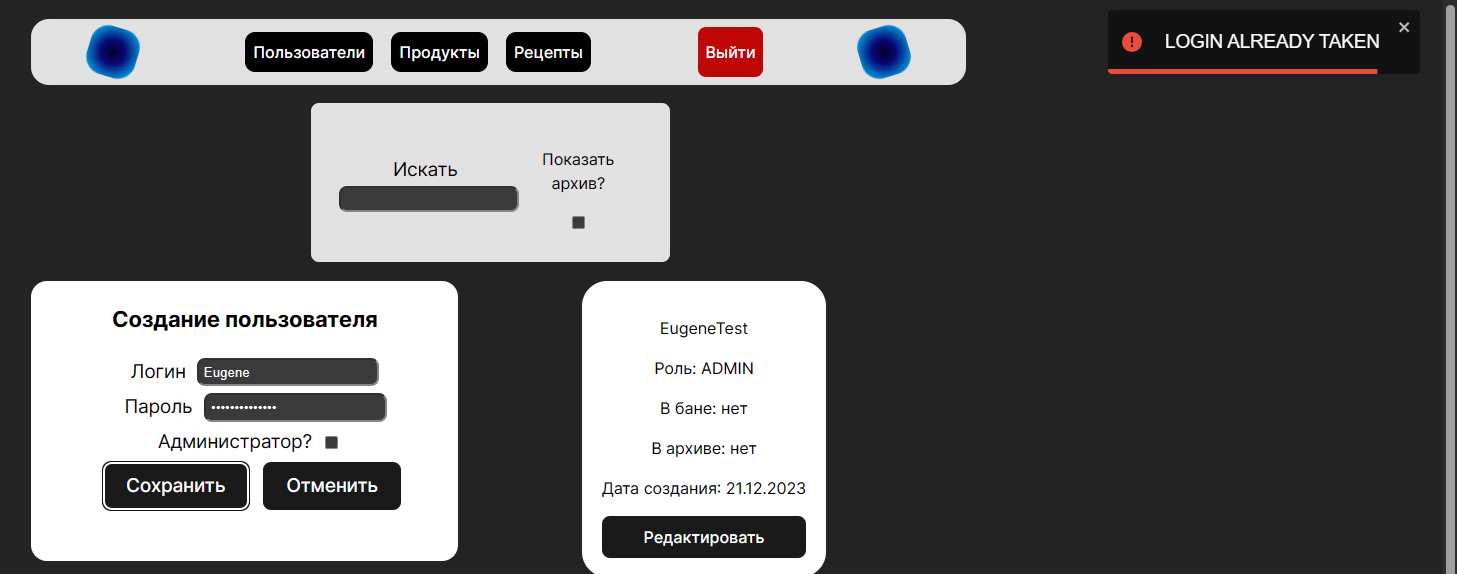


Рисунок 4.6 – Уведомление о занятом логине

Смоделируем ситуацию, что сервер перестал отвечать на запросы клиента по каким-либо техническим причинам. В такой ситуации пользователь получит уведомление о том, что сервер сломан. Ситуация, когда сервер отключен, изображена на рисунке 4.7.

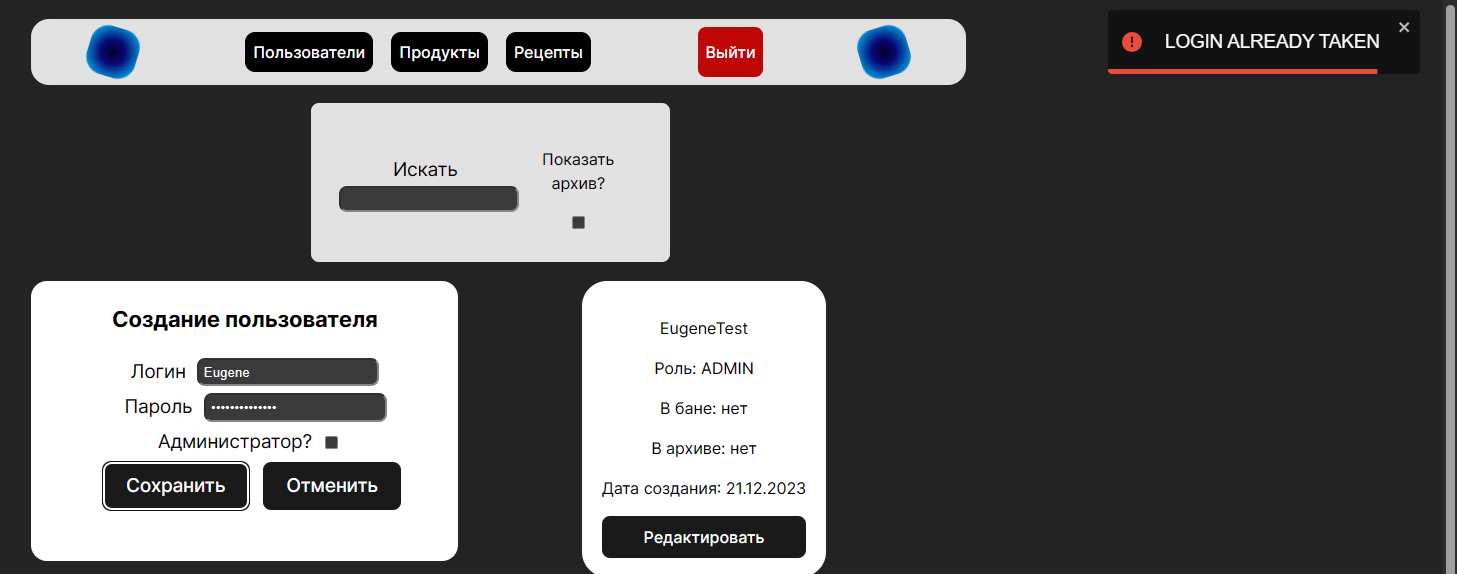


Рисунок 4.7 – Сервер перестал отвечать

В этом случае пользователь поймёт, что запрос на сервер не проходит не по его вине. А ответственный пользователь сможет связаться с администрацией и уведомить о проблеме, что положительно повлияет на скорость восстановления работоспособности приложения.

* 1. Выводы по разделу

В разделе, посвящённом тестированию, были рассмотрены основные процессы приложения, на которых пользователь может чаще всего совершать ошибки, и протестирована правильность работы приложения. Основными моментами, которые необходимо учесть при тестировании, так как это валидация основных форм, а также правильно отображения элементов в соответствии с необходимой бизнес-логикой.

1. Руководство программиста

ФИО

Подпись

Дата

Лист

1

БГТУ 05.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е. А.

5 Руководство программиста

Лит.

Листов

11

*74217076, 2024*

Пользователям, использующим приложение впервые, очень часто непонятны некоторые аспект использования приложения. Руководство предназначено для обеспечения подробной информации о функциональности и способах использования данного проекта. В нем находятся инструкции по развёртыванию и использованию приложения от разных ролей.

* 1. Руководство по развёртыванию приложения

Для успешного развёртывания приложения необходимо выполнить следующие шаги: необходимо зайти в папку с файлом «docker-compose.yaml», открыть терминал или PowerShell в этой папке и прописать команду «docker-compose up -d». При этом, если выполение идет в среде Windows, то требуется дополнительно установить WSL и Docker Desktop. Через некоторое время приложение развернётся в докере. Результат развёртывания можно увидеть на рисунке 5.1.

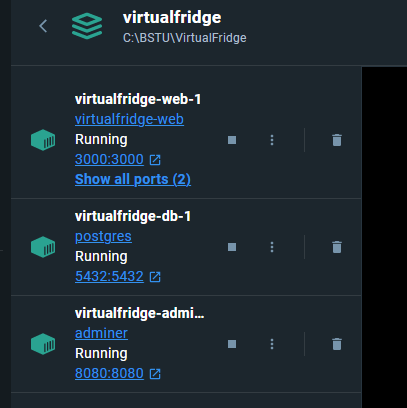


Рисунок 5.1 – Контейнер в Docker Desktop

Как видно, все 3 контейнера запущены, критических ошибок при запуске обнаружено не было, значит развёртывание прошло успешно.

* 1. Руководство для не авторизированного пользователя

У неавторизованного пользователя только две опции: зарегистрироваться или авторизоваться. При этом, для доступа к некоторым точкам сервера требуется обязательная авторизация. Рисунок 5.2 демонстрирует приватность методов на сервере.

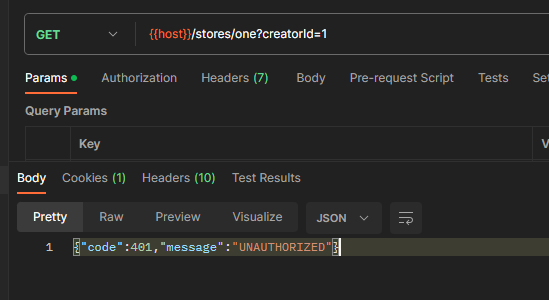


Рисунок 5.2 – Запрос, требующий авторизации

Как видно на рисунке 5.2, запрос на получение хранилища не прошёл, так как в заголовках не указан токен авторизации.

Не все методы являются приватными. Например, регистрация нового пользователя или просмотр рецептов. Ведь не все хотят быть активными участниками приложение. Многие хотят просто посмотреть что бы приготовить сегодня.

Приватными же методами являются создание/удаление рецептов/продуктов, или же создание хранилища и отслеживание продуктов в нем. Создание рецептов не может быть общедоступно, так как возможно столкновение с проблемой спама, когда база данных захламится безобразными записями. В случае авторизованного доступа злоумышленника можно быстро вычислить и заблокировать доступ к приложению. Это один из базовых способов обеспечения безопасности приложения.

* 1. Руководство для авторизированного пользователя

Авторизированному пользователю при входе доступно три вкладки: рецепты, чек-листы и хранилище.

На вкладке «Рецепты» пользователю доступны те рецепты, которые администраторы сделали видимыми. Пользователь способен с помощью поиска найти рецепт, который ему угоден. Гибкая система поиска по названию, типу блюда, калориям покрывает все нужды пользователя, предлагая быстрый поиск.

Сам рецепт представляет собой карточки с кратким описанием и его содержимым. Карточку рецепта можно развернуть и посмотреть подробнее. При раскрытии карточки пользователь может видеть пошаговое приготовление, нужные продукты, количество БЖУ продуктов, автора и знак того, что рецепт является подтверждённым. Так же у рецепта в любом виде присутствует рейтинг, что позволяет понять насколько вкусное то или иное блюдо.

Пользователь может нажать рецепт, после чего на том появится красная рамка. После этого станет активной кнопка «Сформировать список покупок», которая перенесёт все продукты из рецепта в чек-лист, за вычетом тех, что уже имеются у пользователя. Или появится уведомление о том, что у пользователя достаточно продуктов для приготовления рецепта, но при повторном нажатии продукты добавятся всё равно. Это исключает случайное добавление лишнего в хранилище.

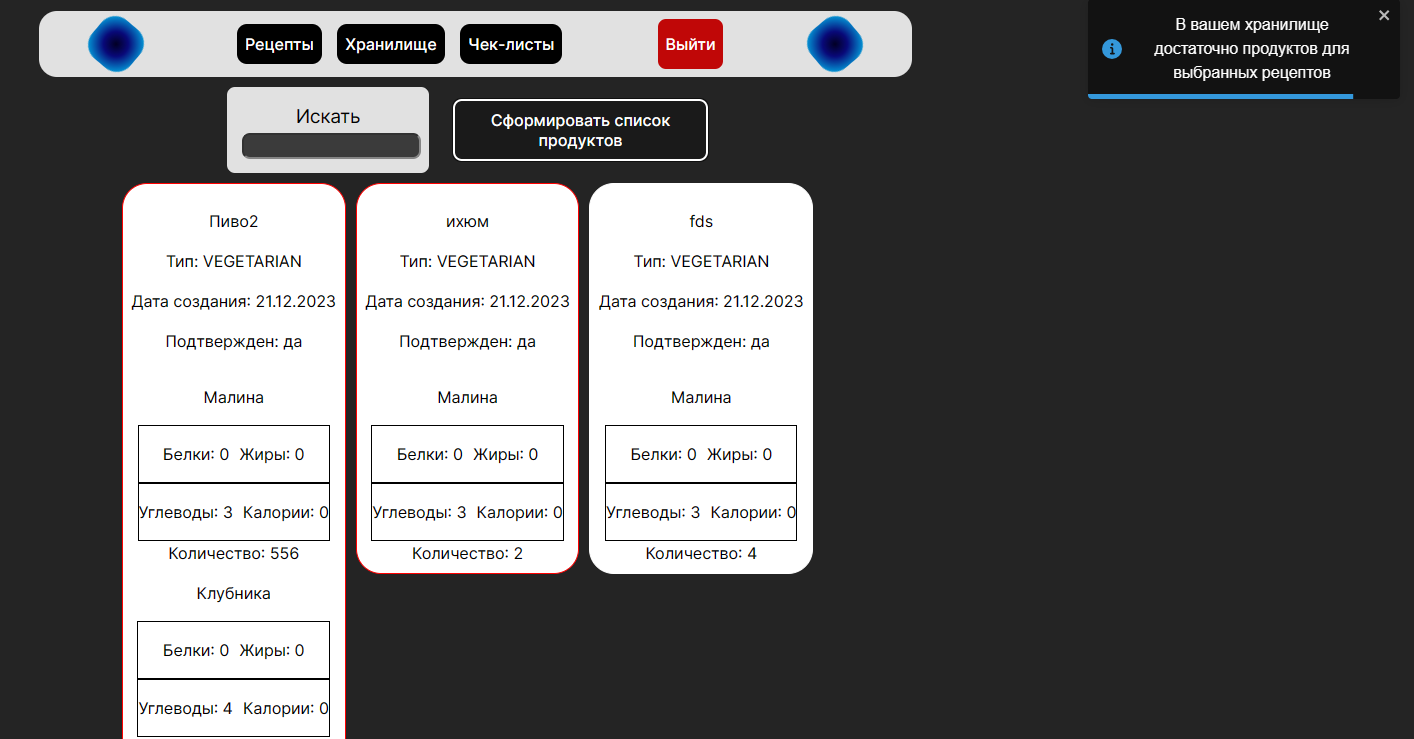


Рисунок 5.3 – Пример формирования списка продуктов

На вкладке «Хранилище» пользователю показываются все его продукты, которые он получил тем или иным образом. В каждой карточке с продуктом содержится название продукта, его количество, единицы измерения, дата, когда продукт испортится (если пользователь указал её или если указан средний срок хранения), а так же калории, белки, жиры, углеводы.

В самой карточке есть кнопки редактировать и удалить для соответствующих действий. В режиме редактирования есть возможность разделять один и тот же продукт на несколько частей и указывать, сколько может хранится та или иная часть. Это покрывает ситуацию, когда один тип проекта был приобретён в разное время и/или с разным сроком годности.

Помимо карточек c продуктами, по левой стороне есть всплывающая форма для добавления нового продукта. В форме присутствует поле для поиска продуктов по базе и после его выбора, последний появляется внизу формы, где можно отредактировать его количество, срок хранения и т.д.

Так же над карточками присутствует поле для поиска продуктов в хранилище по названию. Вид страницы «Хранилище» показан на рисунке 5.4.

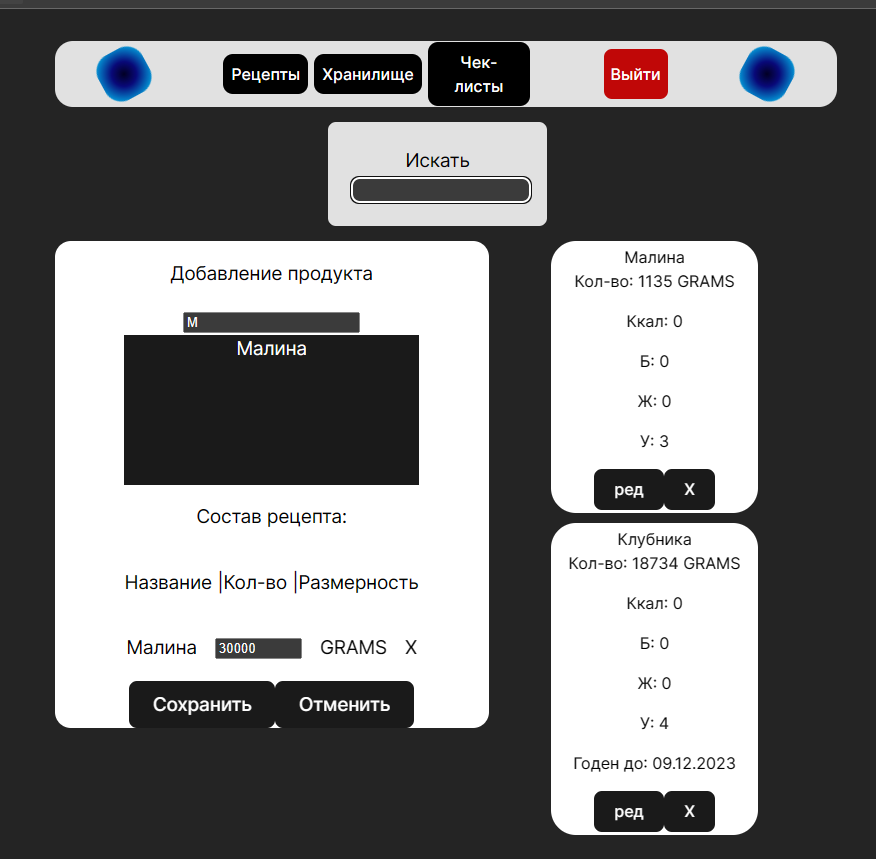


Рисунок 5.4 – Вид страницы «Хранилище»

На странице «Чек-листы» пользователь видит свои чек-листы. Каждый из них отображает дату создания и статус покупки. Так же есть две кнопки, которые дают посмотреть информацию о чек-листе и подтвердить покупку/возобновить чек-лист.

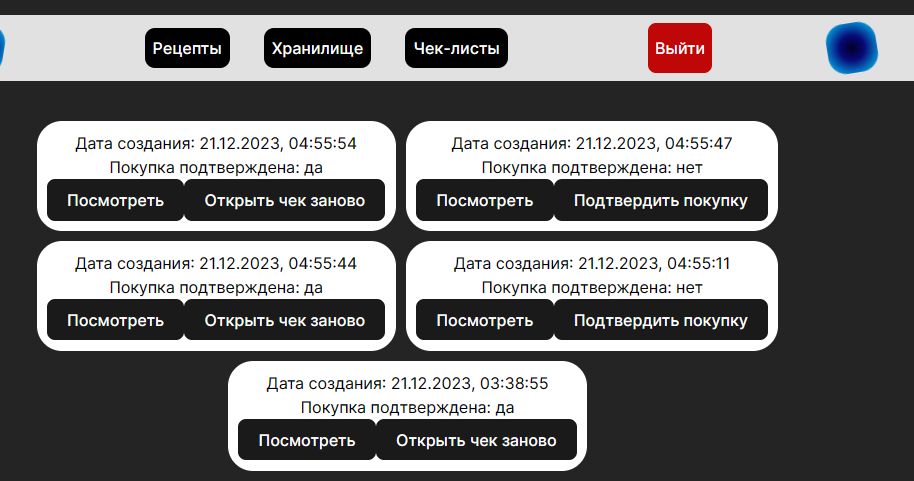


Рисунок 5.5 – Страница «Чек-листы»

После подтверждения покупки всё содержимое добавляется в хранилище.

* 1. Руководство для администратора/модератора

Администратору при входе в приложение доступны следующие вкладки: «Пользователи», «Продукты» и «Рецепты». Все они сводятся к одному принципу: показать весь список сущностей, соответствующих названию вкладки, которые можно редактировать или удалить. И показать форму для создания сущности. На рисунке 5.6 показана страница «Пользователи».

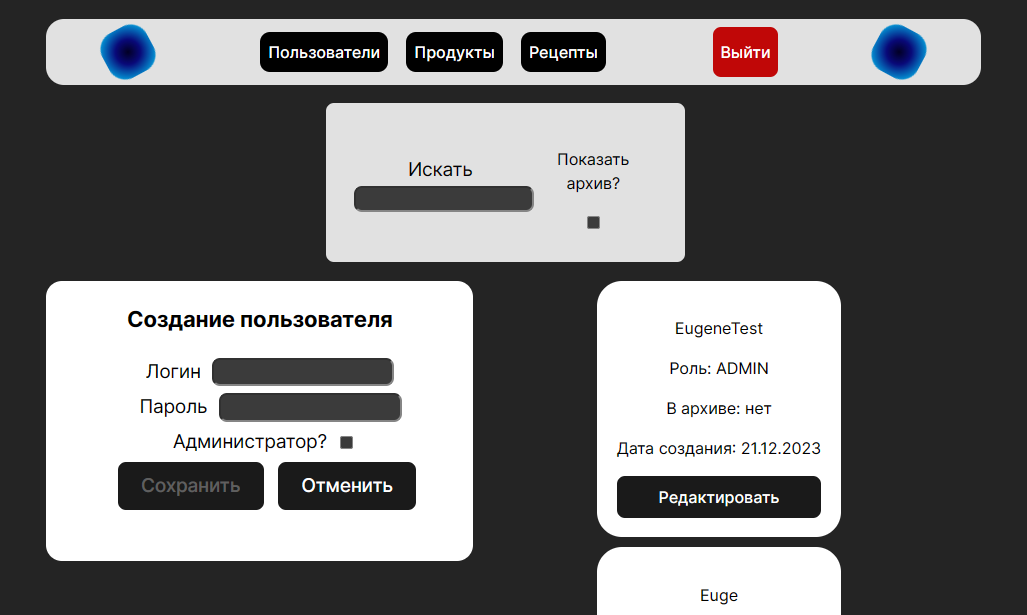


Рисунок 5.6 – Страница «Пользователи»

На этой странице присутствует специальный фильтр «Показать архив?». По нажатию на него появляется список архивированных пользователей.

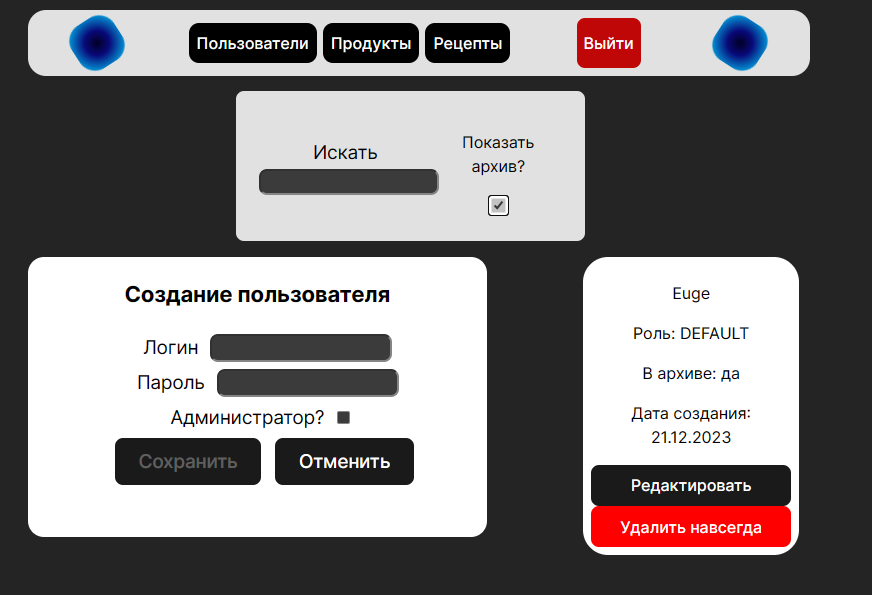


Рисунок 5.7 – Режим «Показать архив?»

В архиве можно как удалить пользователя безвозвратно, так и восстановить его.

На сервере не реализовано удаление записей, так как злоумышленник, получивший доступ, может удалить все данные и сломать всё приложение. Ущерб станет критическим.

Eсли замороженный пользователь попытается войти в аккаунт, то его ждёт разочарование в виде информации о том, что его профиль удалён. Демонстрация на рисунке 5.8.

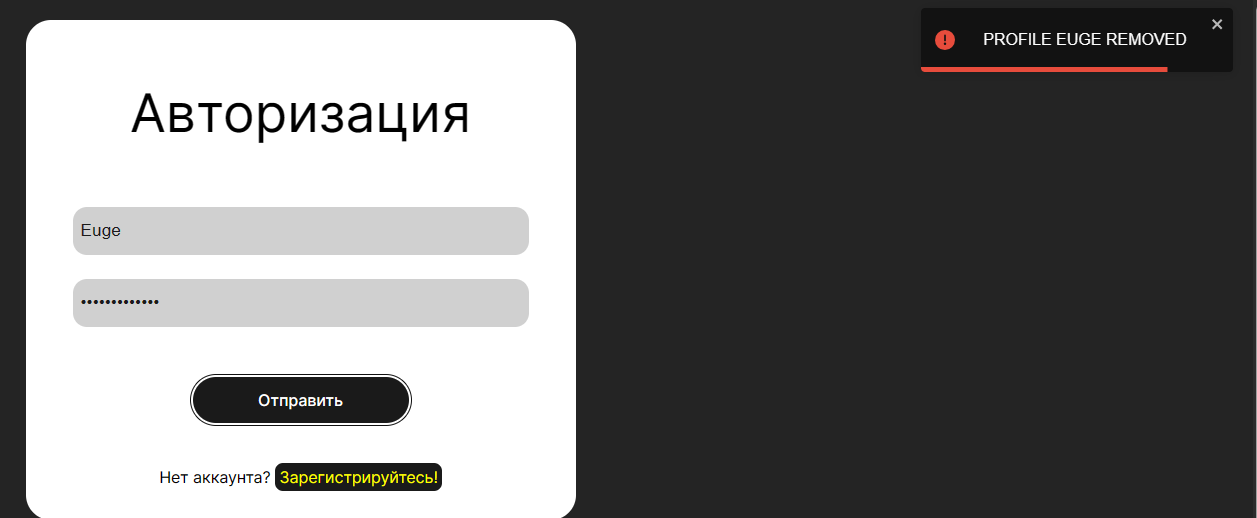


Рисунок 5.8 – Уведомление о том, что профиль удалён.

Как можно видеть, при попытке входа в удалённый аккаунт появляется сообщение с соответствующей ошибкой. В этом случае, если причина удаления не ясна, стоит связаться с администрацией для выяснений обстоятельств и причин удаления.

* 1. Выводы по разделу

В данном разделе были представлены руководства для всех пользователей приложений: неавторизованный пользователь, авторизированный пользователь, администратор/модератор. В самом начале описан способ развёртывания приложения на различных системах, таких как Linux и Windows.

В разделе с развёртыванием подробно описаны действия для запуска приложения в любой системе.

Для авторизированного пользователя были описаны возможности пополнения и редактирования своего хранилища, поиск продуктов и рецептов, а так же добавление собственных рецептов.

Для неавторизованного пользователя описано управление приложением с ограниченными функционалом, а именно регистрация и авторизация на начальном экране.

Для администратора/модератора были расписаны возможности по управлению приложением, такие как создание, редактирование и архивирование пользователей. Так же подробно описана работа с рецептами и продуктами, возможности создания и удаления. Подтверждение подлинности и качества пользовательских рецептов.

6 Технико-экономическое обоснование проекта

ФИО

Подпись

Дата

Лист

1

*43*

*БГТУ 06.00.ПЗ*

Разраб.

*Гончаревич Е. В.*

Провер.

*.*

*Сазонова Д. В.*

Консульт.

*Соболевский А.С. .*

Н. контр.

Нистюк О. А.

Утв.

*Блинова Е. А.*

6 Технико-экономическое обоснование проекта

Лит.

Листов

11

*74217076, 2024*

Основной целью экономического раздела является экономическое обоснование целесообразности разработки программного средства, представленного в дипломном проекте. В рамках данного раздела необходимо определить затраты, произведенные на всех стадиях разработки описанного программного модуля, а также полную стоимость программного средства.

6.1 Общая характеристика разрабатываемого программного средства

При выполнении дипломного проекта было разработано мобильное приложение, предназначенное для оценки уровня активности пользователя.

Цель приложения – предоставить пользователю возможность отслеживать свою физическую активность. Любой пользователь может войти в приложении, для этого надо лишь зайти в Google-аккаунт. Облегчить эту задачу сможет уже задействованные на мобильном устройстве аккаунты, которые можно будет выбрать в окне авторизации. После того, как пользователь зашел в свою учетную запись посредством Google- авторизации, он попадает на главную страницу и ему становятся доступны все имеющиеся функции приложения.

Мобильный продукт позволяет пользователю считать сожженные во время ходьбы калории, пройденные шаги, отслеживать уровень индекса массы тела при изменении физических параметров, добавлять количество выпитой жидкости для поддержания водного баланса в организме. Пользователю будут предложены упражнения, количество калорий которых может покрыть недожженные калории во время ходьбы. Так же имеется возможность просматривать статистику по своим шагам за определенные интервалы времени, получать достижения за количество пройденных шагов, дней посещения приложения и километров пути.

Во время разработки приложения использовались технологии Spring Framework, JPA, Compose, языки программирования Java, Kotlin, SQL, хостинг Heroku. Непосредственно в разработке мобильного продукта использовались библиотека Jetpack Compose для создания собственного пользовательского интерфейса, а также язык программирования Kotlin под разработку мобильного приложения на Android.

Стратегия монетизации предполагает платную месячную подписку на дополнительный функционал в приложении. В рамках данного раздела необходимо определить затраты на всех стадиях разработки программного средства.

Необходимо выполнить расчет экономии основных видов ресурсов в связи с использованием разработанного программного средства на основе успеха и монетизации приложений аналогов.

**6.2 Исходные данные для проведения расчётов**

Исходные данные для расчёта приведены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчёта

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Условные обозначения | Норматив |
| Численность разработчиков, чел | Чр | 1 |
| Коэффициент изменения скорости обработки информации, ед. | Кск | 0,6 |
| Норматив дополнительной заработной платы, % | Ндз | 14 |
| Норматив общепроизводственных и общехозяйственных расходов, % | Нобп, обх | 20 |
| Ставка отчислений в Фонд социальной защиты населения, % | Нфсзн | 34 |
| Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», % | Нбгс | 0,6 |
| Цена одного машино–часа, руб. | Смч | 0,06 |
| Норматив прочих прямых затрат, % | Нпз | 20 |
| Норматив расходов на сопровождение и адаптацию, % | Нрса | 17,0 |
| Ставка НДС, % | ННДС | 0 |
| Налог на прибыль, % | Нп | 0 |

Источниками исходных данных для данных расчетов выступают действующие законы и нормативно-правовые акты. Для ПВТ НДС и налог на прибыль 0%.

**6.3 Методика обоснования цены**

В современных рыночных экономических условиях программное средство (ПС) выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования программных средств. Выбор эффективных проектов программных средств связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у обычного пользователя системы.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации программных средств, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
* сокращения расходов на оплату машинного времени и других ресурсов на отладку программ;
* снижения расходов на материалы;
* ускорение ввода в эксплуатацию новых систем;
* улучшения показателей основной деятельности в результате использования передовых программных средств.

Стоимостная оценка программных средств у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на оплату машинного времени;
* прочие прямые затраты;
* накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена конечного программного средства.

**6.3.1 Объем программного средства**

В таблице 6.2 указаны в укрупнённом виде все работы, реально выполненные для создания, указанного в дипломной работе программного средства и количество рабочих дней, реально потраченных для выполнения этих работ.

Таблица 6.2 – Затраты рабочего времени на разработку ПС

|  |  |
| --- | --- |
| Содержание работ | Затраты рабочего времени, дней |
| Построение диаграмм для проектирования дипломного проекта | 6 |
| Создание базы данных | 3 |
| Разработка серверной части программного средства | 15 |
| Развертывание сервера | 1 |
| Разработка клиентской части программного средства | 21 |
| Тестирование работоспособности мобильного средства | 5 |
| Написание руководства пользователя | 2 |
| Всего | 53 |

Результат по данной таблице будет использовано далее для расчётов.

Для оценки объёма программного средства, все его функции классифицируются с использованием специального каталога функций, который определяет их объем. Общий объем программного средства *V*o, вычисляется как сумма объёмов *Vi* каждой из *n* его функций по формуле 6.1.

|  |  |  |
| --- | --- | --- |
|  | , | (6.1) |

где – объем *i*-ой функции ПС, условных машинных команд;

*n* – общее число функций.

В таблице 6.3 представлены функции, присутствующие в рассматриваемом программном средстве и соответствующий им объем в условных машино–командах.

Таблица 6.3 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| Номер функции | Содержание функции | Объем, условных  машино-команд |
| 101 | Организация ввода информации | 150 |
| 102 | Контроль, предварительная обработка | 450 |
| 111 | Управление вводом/выводом | 2400 |
| 204 | Обработка наборов и записей базы данных | 2670 |
| 207 | Манипулирование данными | 9550 |
| 504 | Обработка прерываний | 540 |
| 506 | Обработка ошибочных и сбойных ситуаций | 410 |
| 703 | Расчет показателей | 460 |
| 707 | Графический вывод результатов | 480 |

Опираясь на данные таблицы 6.3, можно определить объем программного средства, разработанного в ходе дипломного проектирования:

*Vo* = 150 + 450 + 2400 + 2670 + 9550 + 540 + 410 + 460 + 480 = 17110 (маш. команд).

Уточнённый объем программного средства *V*o*/* вычисляется по формуле 6.2.

|  |  |  |
| --- | --- | --- |
|  | , | (6.2) |

где *V*o – объем программного средства, усл. машино-команд;

Кск – коэффициент изменения скорости обработки информации.

Исходя из вычисленного объёма программного средства, можно определить уточненный объем программного средства:

*V*o*/* = 17110 ⋅ 0,6 = 10266 (условных машино–команд).

**6.3.2 Основная заработная плата**

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в программирования на Kotlin Android. В итоге было установлено, что средняя месячная заработная плата на позиции junior составляет 1600 рублей.

Согласно таблице 6.1, проект разрабатывался одним человеком на протяжении 53 дней, что соответствует 2,5 месяца, поскольку в среднем рабочих дней в месяце 21, отсюда суммарное количество дней разработки делим на усредненное количество рабочих дней в месяце. Таким образом, основная заработная плата будет рассчитываться по формуле 6.3

|  |  |  |
| --- | --- | --- |
|  | , | (6.3) |

где Соз– основная заработная плата, руб.;

Траз1 – время раработки , месяцев;

Сзп1 – средняя месячная заработная плата Kotlin Android разработчика.

|  |  |  |
| --- | --- | --- |
|  | руб. |  |

В дальнейшем для других расчётов используется основная заработная плата, рассчитанная по указанной выше методике.

**6.3.3 Дополнительная заработная плата**

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (6.4)

|  |  |  |
| --- | --- | --- |
|  | , | (6.4) |

где Соз– основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

руб.

**6.3.4 Отчисления в Фонд социальной защиты населения**

Отчисления в Фонд социальной защиты населения (ФСЗН) и по обязательному страхованию от несчастных случаем на производстве и профессиональных заболеваний определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей.

Отчисления в Фонд социальной защиты населения вычисляются по формуле 6.5

|  |  |  |
| --- | --- | --- |
|  |  | (6.5) |

где – основная заработная плата, руб.;

– дополнительная заработная плата на конкретное ПС, руб.;

– норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.6

|  |  |  |
| --- | --- | --- |
|  |  | (6.6) |

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили руб., а в фонд социальной защиты населения – руб.

**6.3.5 Расходы на материалы**

Сумма расходов на материалы СМ определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода НМ на уточненный объем программного средства *Vo/* , по формуле 6.7

|  |  |  |
| --- | --- | --- |
|  |  | (6.7) |

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равен 0,46 руб. (по данным, приведенным в приложении 2 таблице П 2.10 «Оценка значений среднего расхода материалов на разработку и отладку 100 строк кода применения ПС» методического пособия), можно определить сумму расходов на материалы.

СМ = 0,46 ⋅ 10266 / 100 = 47,22 руб.

Сумма расходов на материалы была вычислена на основе данных, приведенных в таблице 6.1 данного дипломного проектирования.

**6.3.6 Расходы на оплату машинного времени**

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем программного средства *Vo/* и на норматив расхода машинного времени на отладку ста строк исходного кода НМВ, по формуле 6.8

|  |  |  |
| --- | --- | --- |
|  | . | (6.8) |

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 12 (по данным, приведенным в приложении 2 таблице П 2.11 «Оценка значений среднего машинного времени на отладку 100 строк исходного кода без применения ПС» методического пособия), можно определить сумму расходов на оплату машинного времени.

Смв = 0,06 ⋅ 10266 ⋅ 12 / 100 = 73,92 руб.

Сумма расходов на оплату машинного времени была вычислена на основе данных, приведенных в таблице 6.1 данного дипломного проектирования.

**6.3.7 Прочие прямые затраты**

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз, и находится по формуле 6.9.

|  |  |  |
| --- | --- | --- |
|  |  | (6.9) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму прочих затрат.

Спз = ⋅ 20/ 100 = 800 руб.

**6.3.8 Накладные расходы**

Сумма накладныхрасходовСобп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх, по формуле 6.10.

|  |  |  |
| --- | --- | --- |
|  |  | (6.10) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов.

= ⋅ 20 / 100 = 800 руб.

**6.3.9 Сумма расходов на разработку программного средства**

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов, по формуле 6.11.

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + См + Смв + Спз + Собп,обх. | (6.11) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

Ср = +++ 27,36 + 47,22 + 73,92 + 800 + 800 = 7858,90 руб.

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе, и составила 7858,90 рублей.

**6.3.10** Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса, и находится по формуле 6.12.

|  |  |  |
| --- | --- | --- |
|  |  | (6.12) |

= 7858,90 ⋅ 17 / 100 = 1336,01 руб.

Сумма расходов на сопровождение и адаптацию была вычислена на основе данных, рассчитанных ранее в данном разделе.

Все проведенные выше расчеты необходимы для вычисления полной себестоимости проекта.

**6.3.11 Полная себестоимость**

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию программного средства Срса.

Полная себестоимость Сп вычисляется по формуле 6.13

|  |  |  |
| --- | --- | --- |
|  |  | (6.13) |

Сп = 7858,90 + 1336,01 = 9194,91 руб.

Полная себестоимость программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе.

**6.3.12 Определение цены, оценка эффективности**

Рассматриваемое программное средство разрабатывается для потребления другими лицами. После анализа способа монетизации продуктов-аналогов, был выбран способ монетизации «расширенная подписка на месяц».

Продукты-аналоги, наиболее близкие теме дипломного проекта:

* «Step Counter» – мобильное приложение на Android систему, в большинстве своем выполняет функцию шагомера со встроенными возможностями, как счет калорий, расстояния и отслеживанием принятого уровня жидкости за день. [2]
* «Stepz – шагомер и счет калорий» – мобильное приложение, перенявшее характеристики предыдущего, но с возможностью получения достижений. [3]

По балловому методу цены рассчитываются на основе оценок значимости различных параметров качества программного продукта для потребителей. Сначала каждому из тех параметров, по которым судят о качестве продукции, присваивается значение удельного веса, которое характеризует, насколько данный параметр важен для потребителя по сравнению с другими. Далее каждому параметру базового программного продукта присваивается определенное число баллов, суммирование которых с учетом весовых коэффициентов дает интегральный показатель качества продукции конкурента ИК:

|  |  |  |
| --- | --- | --- |
|  | ИК = Σ (Кi ПКi) / Σ Кi при Σ Кi = 1, | (6.14) |

где К*i* − весовой коэффициент, отражающий значимость *i-*го показателя;

ПК*i* − число баллов, присвоенное *i-*му показателю продукта конкурента.

Показатель качества рассматриваемого продукта ИР рассчитывается как:

ИР = Σ (Кi ПРi) / Σ Кi при Σ Кi = 1, (6.15)

где ПР*i* − число баллов, присвоенное *i-*му показателю качества рассматриваемого программного продукта.

Были выбраны следующие характеристики для показателей качества рассматриваемого программного продукта и программного продукта конкурента:

* Дизайн – то, как приложение выглядит, очень важно для любого современного приложения.
* Юзабилити – насколько приложение удобно в использовании.
* Функциональность – количество инноваций, внедряемых в приложение.
* Отсутствие багов – наличие несущественных ошибок в работе.

Расчет показателей качества базового и нового продуктов, согласно балловому методу, приводится в таблицах 6.4

Таблица 6.4 – Показатели качества рассматриваемого программного продукта и программного продукта конкурента

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Показатель качества | Весовой коэффициент | Моё приложение | Step Counter | Stepz – шагомер и счет калорий |
| Дизайн | 0,3 | 8 | 6 | 4 |
| Юзабилити | 0,4 | 7 | 7 | 6 |
| Функциональность | 0,2 | 7 | 7 | 5 |
| Отсутствие багов | 0,1 | 8 | 7 | 7 |
| Всего | 1 | 7,4 | 6,7 | 5,3 |

Расчёт прогнозного количества установок программного средства Кi при монетизации методом расширенной месячной подписки на дополнительный функционал приложения, рассчитывается по формуле 6.16:

|  |  |
| --- | --- |
|  | (6.16) |

где К0 − количество установок ПС конкурента;

T0 – количество лет существования приложения;

ИР – показатель рассматриваемого программного продукта;

ИК – показатель программного продукта конкурента.

К1 = (500 000 / 6 7,4) / 6,7 = 92 039,80 (установок в год),

К2 = (200 000 / 4 7,4) / 5,3 = 69 811,32 (установок в год),

К = (92 039,80 + 69 811,32) / 2 = 80 925,56 (установок в год).

Определение цены месячной расширенной подписки нового продукта Ц1 будет осуществляться по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (6.17) |

где Ц0 − цена подписки программного продукта конкурента,

ИР – показатель рассматриваемого программного продукта,

ИК – показатель программного продукта конкурента.

У приложения Step Counter ежемесячная подписка – 7$ (20,51),

У Stepz – шагомер и счет калорий – 5$ (14,65)

Ц1 = (20,51 7,4) / 6,7 = 22,65 рублей

Ц2 = ( 7,4) / 5,3 = 20,4 рублей

Ц = (22,65 + 20,45) / 2 = 21,55 рублей

Пользователи получают базовый функционал бесплатно, а за дополнительные опции платят. При таком раскладе обычно только 0,5 процент пользователей покупает расширенную подписку. В таком случае при среднем количестве установок в год, равном 80 925,56, денежные поступления от покупки подписки будут составлять Пост.в год = 8 719,73 рублей за год.

Количество покупателей продукта необходимых для окупаемости расширения Пп вычисляется по формуле 6.18:

|  |  |  |
| --- | --- | --- |
|  |  | (6.18) |

где – полная себестоимость, руб.;

Пост.в год – денежные поступления от продажи подписки за год, руб.;

Ткп = 9194,9 / 21,55 = 426 покупателя.

Срок окупаемости приложения Tок вычисляется по формуле 6.19

|  |  |  |
| --- | --- | --- |
|  |  | (6.19) |

где – полная себестоимость, руб.;

Пост.в год – денежные поступления от месячной подписки на дополнительный функционал приложения за год, руб.

Ток = 9194,9 / 8719,73 = 1,05 года.

**6.4 Выводы по разделу**

В таблице 6.5 представлены результаты расчётов для основных показателей данной главы в краткой форме. Данные по экономическому обоснованию приведены в приложении Е.

Таблица 6.5 – Таблица экономических показателей

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 2,5 |
| Количество программистов, чел. | 1 |
| Основная заработная плата, руб. | 4000 |
| Дополнительная заработная плата, руб. |  |
| Отчисления в Фонд социальной защиты населения и по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний, руб. |  |
| Расходы на материалы, руб. | 47,22 |
| Расходы на оплату машинного времени, руб. | 73,92 |
| Прочие прямые затраты, руб. | 800 |
| Накладные расходы, руб. | 800 |
| Себестоимость разработки программного средства, руб. | 7 858,90 |
| Расходы на сопровождение и адаптацию, руб. | 1 336,01 |
| Полная себестоимость, руб. | 9 194,91 |
| Годовые денежные поступления от продажи подписки, руб. | 8 719,73 |
| Срок окупаемости, лет | 1,05 |
| Количество покупателей для окупаемости, чел. | 426 |

Разработка программного средства, осуществляемая одним программистом в течение 2,5 месяцев, при заданных условиях обойдется в 9194,91 руб. Реализации данного программного средства будет приносить годовые денежные поступления в размере 8 719,73 рублей и окупиться при покупке 426 подписок через 1,05 года.

Заключение

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е. А.

Заключение

Лит.

Листов

1

*74217076, 2024*

В ходе создания программного продукта – списка покупок, был проведён анализ существующих платформ и сервисов, предоставляющих данные услуги. Изучены основные аспекты интерфейса, функциональности, и бизнес-моделей, выделены положительные аспекты и решения, которые могли бы быть успешно интегрированы в создаваемое программное обеспечение. Были сформулированы основные цели и актуальность разработки сервиса.

Проектирование программного продукта предполагало выбор основных технологий, фреймворков и библиотек, разработку архитектуры системы, включая клиентскую и серверную части, а также проектирование базы данных. На этапе проектирования были определены основные алгоритмы работы программы, их визуализация в виде блок-схем, а также создана диаграмма использования приложения.

Реализация программного продукта началась с создания серверной части. Была изучена структура проекта, аутентификации и авторизации пользователей. Архитектура была основана на использовании Bun.js, а также использовании базы данных PostgreSQL для хранения информации о рецептах, пользователях и продуктах.

Клиентская часть программного продукта также была разработана. Была определена структура проекта, осуществлена реализация HTTP клиентов для общения с серверной частью, а также выбран React для построения пользовательского интерфейса и управления глобальным состоянием приложения.

Тестирование программного продукта проводилось с целью выявления и устранения возможных ошибок. Все ключевые функции приложения были протестированы на различных уровнях доступа, чтобы гарантировать корректную работу системы для всех типов пользователей.

Было составлено руководство для пользователей программного продукта. Руководство содержит информацию о функциональности приложения для каждой из ролей пользователей.

В заключение была экономически обоснована стоимость проекта, а так же его окупаемость.

Список использованных источников

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Гончаревич Е. В.*

Пров.

*Сазонова Д. В.*

Н. контр.

*Нистюк О. А.*

Утв.

Блинова Е. А.

Список использованных источников

Лит.

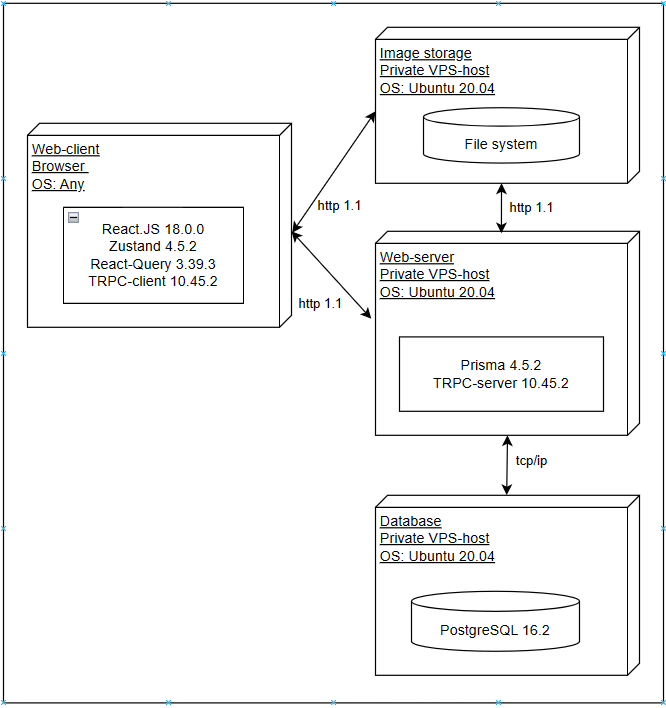
Листов

1

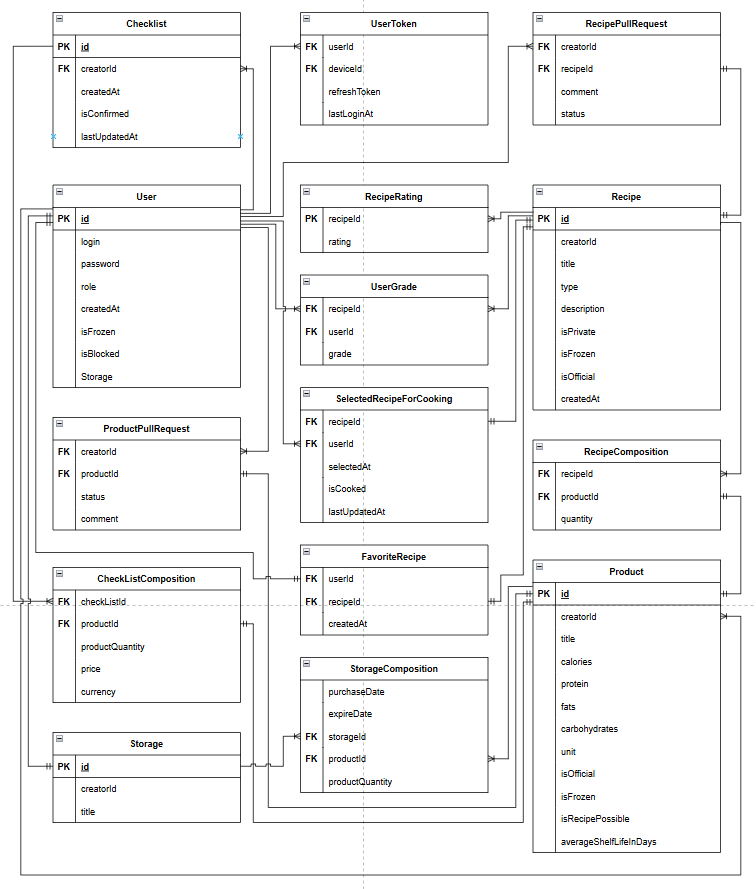
*74217076, 2024*

1. Bring [Электронный ресурс]/Читатель– Режим доступа: https://apps.apple.com/ru/app/bring/ – Дата доступа: 03.06.2024.
2. Listonic [Электронный ресурс]/Читатель – Режим доступа: https://new.app.listonic.com/en – Дата доступа: 03.06.2024.
3. Russian food [Электронный ресурс]/Читатель – Режим доступа: https://www.russianfood.com/ – Дата доступа: 03.06.2024.
4. PostgreSQL [Электронный ресурс] / PostgreSQL – Режим доступа: https://www.postgresql.org/ ­– Дата доступа: 03.06.2024
5. React – Getting Started [Электронный ресурс] – Режим доступа: https://reactjs.org/docs/getting-started.html – Дата доступа: 03.06.2024
6. Понимая Docker [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/253877/ – Дата доступа: 03.06.2024
7. Axios [Электронный ресурс] – Режим доступа: https://axios-http.com/ru/docs/intro – Дата доступа: 03.06.2024

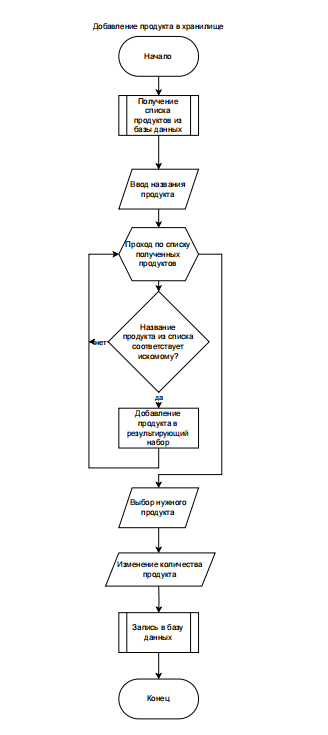
ПРИЛОЖЕНИЕ А Диаграмма развёртывания



1. ПРИЛОЖЕНИЕ Б Логическая схема базы данных



1. ПРИЛОЖЕНИЕ В Блок-схема алгоритма добавления продукта в хранилище



1. ПРИЛОЖЕНИЕ Г Блок-схема алгоритма формирования списка покупок

