МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Специализация 1-40 01 01 10 «Программное обеспечение информационных технологий (программирование интернет-приложений)»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

Web-приложение «Подбор рецептов»

Выполнил студент Зинович Елизавета Игоревна

(Ф.И.О.)

Руководитель проекта асс. Нистюк О. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты асс. Нистюк О. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер асс. Нистюк О. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Содержание

[Введение 2](#_Toc1300)

[1 Постановка задачи 3](#_Toc26702)

[1.1 Аналитический обзор литературы 3](#_Toc32751)

[1.2 Анализ требований к программному средству и разработка функциональных требований 5](#_Toc7565)

[2 Проектирование приложения 7](#_Toc24031)

[2.1 Проектирование архитектуры программного средства 7](#_Toc15706)

[2.3 Проектирование логической структуры базы данных 8](#_Toc14788)

[3 Разработка приложения 10](#_Toc16226)

[3.1 Реализация серверной части программного средства 10](#_Toc7076)

[3.2 Реализация клиентской части программного средства 14](#_Toc26231)

[3.3 Реализация базы данных 16](#_Toc12900)

[3.4 Реализация WebSocket 18](#_Toc10207)

[4 Тестирование 20](#_Toc24665)

[5 Руководство пользователя 24](#_Toc15897)

[5.1 Руководство для неаутентифицированного пользователя 25](#_Toc22913)

[5.2 Руководство для аутентифицированного пользователя 28](#_Toc19132)

[5.3 Руководство для администратора 30](#_Toc18530)

[Заключение 32](#_Toc18526)

[Список используемых источников 33](#_Toc22227)

[ПРИЛОЖЕНИЕ А 34](#_Toc21669)

[ПРИЛОЖЕНИЕ Б 35](#_Toc8780)

# **Введение**

Кулинарные сайты — незаменимые помощники при приготовлении правильной, вкусной и здоровой пищи, как для начинающих, так и для опытных кулинаров. Они помогают придумать, что приготовить, на завтрак, обед или ужин.

На кулинарных ресурсах обычно есть различные разделы, например, закуски, первые и вторые блюда, десерты, напитки и т.п. Там присутствует подробный список ингредиентов, которые требуются для приготовления того или иного блюда.

Целью курсового проекта будет разработка Web-приложения «Подбор рецептов», предоставляющего возможности находить рецепты новых блюд.

В приложении должны присутствовать:

* регистрация, аутентификация и авторизация пользователя;
* добавление авторских рецептов пользователями;
* подбор рецептов по различным критериям;
* сохранение рецептов для пользователя;
* модерация, удаление рецептов администратором сайта.

Несмотря на то, что существует много кулинарных ресурсов, разработка подобного приложения со своими особенностями не теряет актуальности.

# **Постановка задачи**

## **Аналитический обзор литературы**

Существует большое количество сервисов по подбору рецептов. Одни из них – Povarenok и RussianFood.

Рассмотрим сервис RussianFood на рисунке 1.1. Можно отметить следующие особенности:

* представлена широкая база рецептов;
* интерфейс пользователя выглядит не современно;
* на странице присутствует довольно много визуально лишней информации.



Рисунок 1.1 – Сервис RussianFood, страница с рецептами

Сервис имеет преимущество: присутствует удобная форма фильтрации и поиска с возможностью включения и исключения ингредиентов, она приведена на рисунке 1.2.

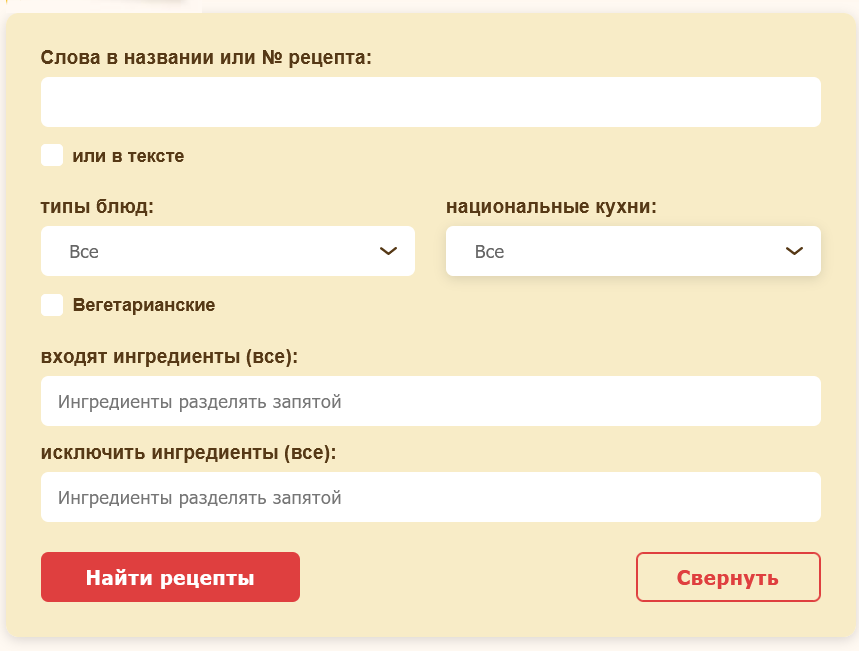


Рисунок 1.2 – Сервис RussianFood, форма фильтрации и поиска

Следующим рассматриваемым конкурентом станет Povarenok, приведенный на рисунке 1.3. Как и в случае с первым конкурентом, сервис имеет устаревший внешний вид, однако обладает достаточно обширным функционалом, включающим:

* регистрацию, вход в аккаунт;
* добавление рецептов;
* поиск, фильтрацию рецептов.

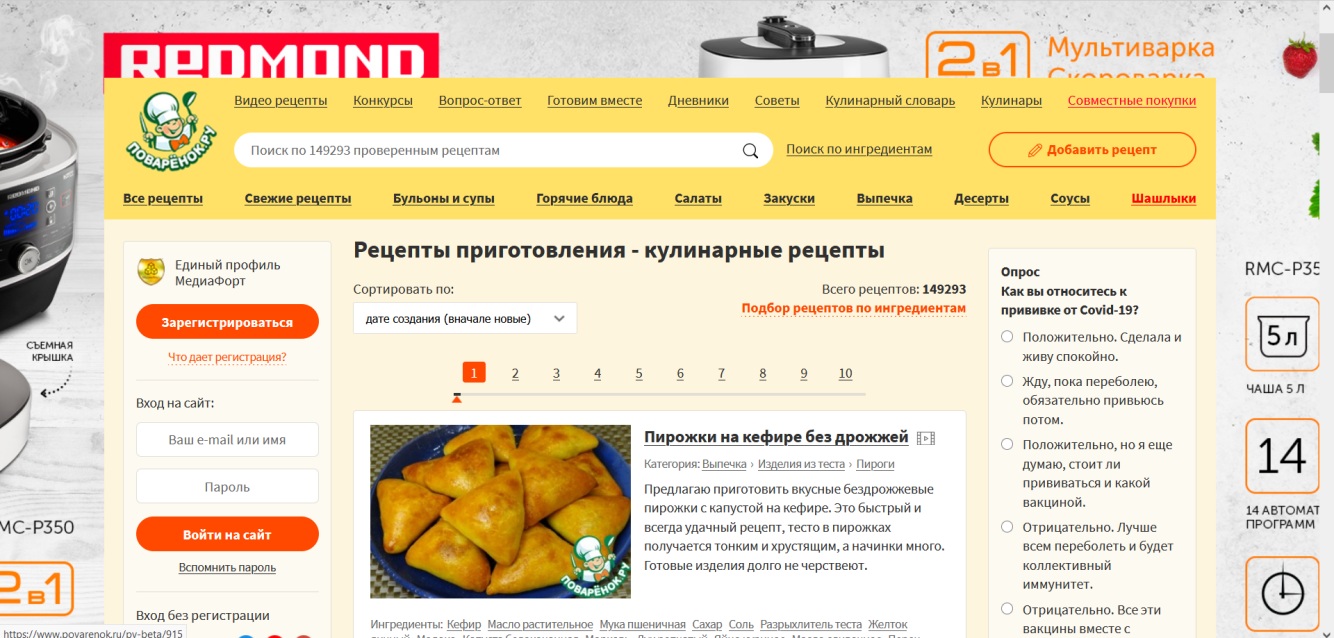


Рисунок 1.3 – Сервис Povarenok, главная страница

Исходя из анализа, можно определить, что необходимо реализовать в программном средстве, чтобы оно было конкурентноспособным:

* регистрацию, вход в аккаунт;
* добавление рецептов;
* поиск, фильтрацию рецептов.

Для получения приемущества над конкурентами в реализуемом приложении можно улучшить функционал фильтрации и поиска: поиск по названию рецепта дополнится поиском по тексту инструкции рецепта, а фильтрация будет происходить по категории блюда, приему пищи, а также сможет включать и исключать ингредиенты. Также будет важно разработать более современный пользовательский интерфейс.

## **1.2 Анализ требований к программному средству и разработка функциональных требований**

Приложение должно иметь разделение ролей. Для каждой роли были разработаны UML-диаграммы вариантов использования.

Также приложение должно предоставлять пользователям возможность входа с актуальными логином и паролем, производить валидацию вводимых пользователем данных, обрабатывать возникающие исключительные ситуации.

Чтобы соответствовать ранее выработанным требованиям, программное средство должно предоставлять пользователям, в зависимости от их роли, следующие функциональные возможности.

Функции неавторизованного пользователя:

* выполнение регистрации, входа в приложение под своими учетными данными, сброс пароля;
* просмотр главной страницы со списком рецептов;
* поиск рецептов и фильтрация;
* получение браузерных уведомлений о публикации новых рецептов;
* просмотр подробной информации о некотором рецепте.

Функциональные возможности отображены на UML-диаграмме вариантов использования на рисунке 1.4.



Рисунок 1.4 – UML диаграмма вариантов использования приложения для неавторизованного пользователя

Функции авторизованного пользователя:

* смена пароля;
* просмотр главной страницы со списком рецептов;
* поиск рецептов и фильтрация;
* просмотр подробной информации о некотором рецепте;
* сохранение рецепта в свою книгу рецептов, просмотр ее содержимого и удаление рецептов из нее;
* получение браузерных уведомлений о публикации новых рецептов;
* просмотр, создание, редактирование и удаление собственных рецептов.

Функциональные возможности отображены на UML-диаграмме вариантов использования на рисунке 1.5.



Рисунок 1.5 – UML диаграмма вариантов использования приложения для авторизованного пользователя

Функции администратора сервиса:

* смена пароля;
* просмотр главной страницы со списком рецептов;
* поиск рецептов и фильтрация;
* просмотр подробной информации о некотором рецепте;
* сохранение рецепта в свою книгу рецептов, просмотр ее содержимого и удаление рецептов из нее;
* просмотр, создание, редактирование и удаление собственных рецептов;
* модерация новых рецептов пользователей (одобрение, отклонение);
* удаление любого рецепта;
* просмотр, создание, редактирование и удаление приемов пищи, ингредиентов, категорий рецептов.

Функциональные возможности отображены на UML-диаграмме вариантов использования на рисунке 1.6.

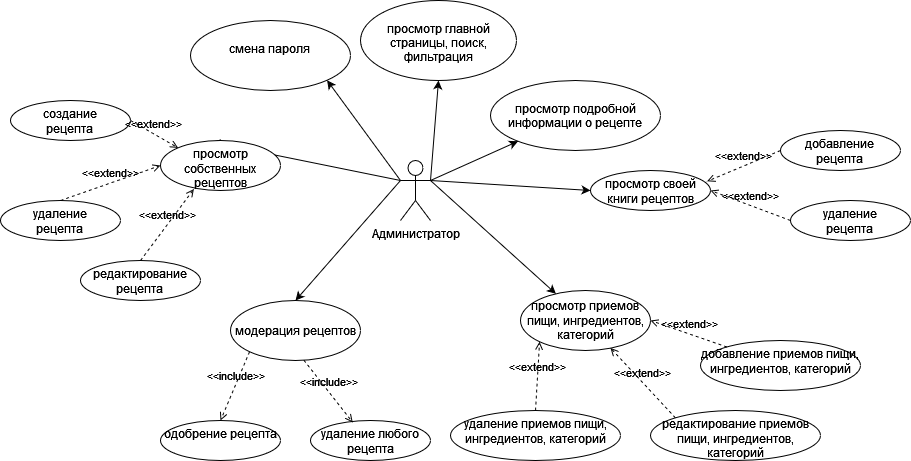


Рисунок 1.6 – UML диаграмма вариантов использования приложения для администратора

# **Проектирование приложения**

## **2.1 Проектирование архитектуры программного средства**

В предыдущей главе были разработаны технические требования к программному средству.

Для их реализации необходимо разработать два компонента — сервер и клиент.

Серверная часть должна содержать:

* HTTPS-сервер, реализующий конечные точки для методов GET, POST, PUT, DELETE для получения, добавления, обновления и удаления данных соответственно;
* WebSocket-сервер, реализующий возможность обмена данными между браузером и сервером через постоянное соединение, засчет чего получение клиентом мгновенных браузерных уведомлений о публикации новых рецептов.

Основными компонентами программного средства должны стать:

* модели, которые определят структуру и логику используемых данных;
* представления, которые определят клиентскую часть приложения;
* контроллеры, что будут обрабатывать входящие запросы, взаимодействовать с сервисами, и отправлять в ответ клиенту некоторый результат обработки;
* маршрутизаторы, которые сопоставят запросы с маршрутами и выберут для обработки запроса определенный контроллер;
* сервисы, что выступят как промежуточный слой между моделями и контроллерами.

Клиентская часть должна быть представлена совокупностью страниц, использующих HTML5, CSS3, а также JavaScript.

## **2.3 Проектирование логической структуры базы данных**

Для хранения данных программного средства «Подбор рецептов» требуется разработка базы данных.

Она состоит из 8 таблиц, схема приведена на рисунке в приложении А. Ниже представлено описание реализации таблиц базы данных.

Таблица users представляет собой описание пользователя приложения, с данными для входа. Содержит следующие поля:

* id, целочисленного типа, автоинкрементируемое, первичный ключ — уникальный номер записи в таблице;
* email, набор символов, не допускает null значений — адрес электронной почты пользователя;
* login, набор символов, не допускает null значений — логин пользователя;
* password, набор символов, не допускает null значений — пароль для входа в аккаунт;
* isGranted, логический тип, не допускает null значений, значение по умолчанию 0 (или false) — метка пользователя с повышенными привилегиями (администратор);
* resetPasswordLink, набор символов, допускает null значения — временная ссылка для сброса пароля пользователя;
* temporaryPassword, набор символов, допускает null значения — временный пароль пользователя.

Таблица meals хранит возможные приемы пищи. Содержит следующие поля:

* id, целочисленного типа, автоинкрементируемое, первичный ключ — уникальный номер записи в таблице;
* meal, набор символов, не допускает null значений — название приема пищи.

Таблица categories хранит категории, к которым можно отнести рецепты. Содержит следующие поля:

* id, целочисленного типа, автоинкрементируемое, первичный ключ — уникальный номер записи в таблице;
* category, набор символов, не допускает null значений — название категории.

Таблица ingredients хранит список всех возможных ингредиентов для рецепта. Содержит следующие поля:

* id, целочисленного типа, автоинкрементируемое, первичный ключ — уникальный номер записи в таблице;
* name, набор символов, не допускает null значений — название ингредиента;
* measurement, набор символов, допускает null значения — единица измерения количества ингредиента.

Таблица images хранит изображения, связанные с рецептом. Содержит следующие поля:

* id, целочисленного типа, автоинкрементируемое, первичный ключ — уникальный номер записи в таблице;
* uri, набор символов, не допускает null значений — адрес изображения;
* description, набор символов, допускает null значения — краткое описание изображения;
* recipeId, целочисленного типа, не допускает null значений, используется для связи с полем id таблицы recipes — идентификатор рецепта.

Таблица cookbooks хранит список сохраненных пользователем рецептов. Содержит следующие поля:

* recipeId, целочисленного типа, не допускает null значений, используется для связи с полем id таблицы recipes — идентификатор рецепта;
* userId, целочисленного типа, не допускает null значений, используется для связи с полем id таблицы users — идентификатор пользователя.

Таблица recipe\_ingredients хранит список ингредиентов отдельного рецепта.

Содержит следующие поля:

* id, целочисленного типа, автоинкрементируемое, первичный ключ — уникальный номер записи в таблице;
* recipeId, целочисленного типа, не допускает null значений, используется для связи с полем id таблицы recipes — идентификатор рецепта;
* ingredientId, целочисленного типа, допускает null значения, используется для связи с полем id таблицы ingredients — идентификатор ингредиента;
* quantity, целочисленного типа, не допускает null значений— количество ингредиентов этого вида в рецепте.

Таблица recipes хранит список рецептов. Содержит следующие поля:

* id, целочисленного типа, автоинкрементируемое, первичный ключ — уникальный номер записи в таблице;
* categoryId, целочисленного типа, не допускает null значений, используется для связи с полем id таблицы categories— идентификатор категории;
* authorId, целочисленного типа, не допускает null значений, используется для связи с полем id таблицы users — идентификатор автора рецепта;
* mealId, целочисленного типа, не допускает null значений, используется для связи с полем id таблицы meals — идентификатор приема пищи;
* datePublished, дата, не допускает null значений — дата публикации рецепта;
* timeToCook, целочисленного типа, не допускает null значений — время, затрачиваемое на приготовление рецепта;
* instruction, набор символов, не допускает null значений — инструкция приготовления;
* title, набор символов, не допускает null значений — наименование рецепта;
* isApproved, логический тип, не допускает null значений, значение по умолчанию 0 (или false) — метка подтвержденного администратором рецепта.

# **Разработка приложения**

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

## **3.1 Реализация серверной части программного средства**

Для разработки серверной части приложения был использован Express.js — фреймворк для Node.js, который реализует слой функций, необходимых для создания эффективных приложений и API. Его использование значительно сокращает написание кода, а, значит, уменьшается затрачиваемое на разработку время.

На рисунке 3.1 отображена структура проекта разрабатываемого программного средства.

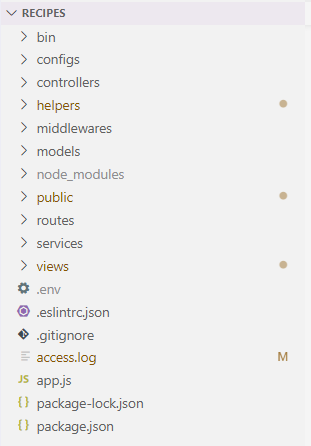


Рисунок 3.1 – Структура проекта сервера

В директории «configs» находятся файлы, содержащие настройки подключения к базе данных, а также параметры для создания cookies.

Директория «controllers» хранит функции, выступающие обработчиками маршрутов. Пример одного из методов контроллера, а именно одобрение администратором рецепта при модерации, приведен в листинге 3.1.

async function approveRecipe(req, res, next) {

    try {

        let {id} = req.params;

        let recipe = await recipesService.approve(id);

        return res.json({recipe});

    } catch (e) {

        next(e);

    }

}

Листинг 3.1 – Контроллер admin, одобрение администратором рецепта

В «services» находятся функции, отвечающие за поставку данных контроллерам, например, с их помощью происходит обращение к базе данных.

Пример одного из методов сервисов, а именно создание нового ингредиента для рецепта, приведен в листинге 3.2.

exports.create = async ({ recipeId, ingredientId, quantity }) => {

    const recipe = await Recipes.findOne({ where: { id: recipeId } });

    if (!recipe) throw new NotExistsError('recipe');

    const ingredient = await Ingredients.findOne({ where: { id: ingredientId } });

    if (!ingredient) throw new NotExistsError('ingredient');

    const result = await RecipeIngredients.create({

        ingredientId: ingredient.id,

        recipeId: recipe.id,

        quantity,

    });

    return result;

};

Листинг 3.2 – Сервис recipeIngredients, создание нового ингредиента для рецепта

Директория «middlewares» содержит промежуточные обработчики которые нужны для проверки JWT токена и доступности действия исходя из роли запрашивающего.

Директория «models» содержит файлы с описаниями сущностей базы данных. Приложение использует ORM-библиотеку «Sequelize» для взаимодействия с базой данных. Конфигурацию подключения модель импортирует из файла sequelizeconfig.js (расположен в директории «configs»).

Пример реализации одной из моделей, представляющей собой категорию, приведен в листинге 3.3.

module.exports = function DefineCategories(sequelize, DataTypes) {

    return sequelize.define('categories', {

        id: {

            autoIncrement: true,

            type: DataTypes.INTEGER,

            allowNull: false,

            primaryKey: true,

        },

        category: {

            type: DataTypes.TEXT,

            allowNull: false,

            unique: 'Category constraint',

        },

    }, {

        sequelize,

        tableName: 'categories',

        schema: 'public',

        timestamps: false,

        indexes: [

            {

                name: 'Category constraint',

                unique: true,

                fields: [

                    { name: 'category' },

                ],

            },

            {

                name: 'categories\_pkey',

                unique: true,

                fields: [

                    { name: 'id' },

                ],

            },

        ],

    });

};

Листинг 3.3 – Модель «categories»

Директория «routers» содержит описания всех маршрутов, причем любая взаимосвязанная совокупность маршрутов выносится в отдельный файл. Ниже, в листинге 3.4, представлено добавление маршрутизаторов с учётом их расположения в директориях проекта. Также мы настраиваем приложение на их использование по определённому пути.

app.use('/login', login\_router);  
app.use('/', require('./routes/index'));

app.use('/', require('./routes/auth'));

app.use('/meals', require('./routes/meals'));

app.use('/categories', require('./routes/categories'));

app.use('/recipes', require('./routes/recipes'));

app.use('/ingredients', require('./routes/ingredients'));

app.use('/cookbooks', require('./routes/cookbooks'));

app.use('/admin', require('./routes/admin'));

app.use('/userrecipes', require('./routes/userRecipes'));

Листинг 3.4 – Регистрация маршрутизаторов

Пример реализации одного из маршрутизаторов, отвечающего за книгу рецептов (cookbooks), приведен в листинге 3.5.

const express = require('express');

const { validate } = require('express-validation');

const router = express.Router();

const controller = require('../controllers/cookbooks');

const authMiddleware = require('../middlewares/auth');

const validation = require('../helpers/validators/cookbooksValidation');

router.get('/', controller.renderCookbook);

router.post('/', authMiddleware, controller.findAllForUser);

router.post('/add/:recipeId', authMiddleware, validate(validation.create), controller.create);

router.delete('/delete/:recipeId', authMiddleware, validate(validation.remove), controller.delete);

module.exports = router;

Листинг 3.5 – Реализация маршрутизатора для книги рецептов

Директория «public» содержит статические файлы: изображения, стили, файлы с расширением .js, содержащие функции, использующиеся на стороне клиента, например, для отправки запросов на сервер.

В директории «helpers» содержатся остальные вспомогательные файлы, используемые сервером, такие, как собственные классы ошибок, правила валидации. Пример файла cookbooksValidation из этой директории приведен в листинге 3.6.

const { Joi } = require('express-validation');

const create = {

    params: Joi.object({

        recipeId: Joi.number().required(),

    }),

};

const remove = {

    params: Joi.object({

        recipeId: Joi.number().required(),

    })

};

module.exports = {

    create,

    remove,

};

Листинг 3.6 – Правила валидации для книги рецептов

Для реализации https соединения были определены опции ключевой информации, приведенные в листинге 3.7.

const httpsOptions = {

  key: fs.readFileSync(\_\_dirname + '/../helpers/cert/cert.key'),

  cert: fs.readFileSync(\_\_dirname + '/../helpers/cert/cert.crt'),

};

Листинг 3.7 – Параметры для создания https соединения

Сертификаты и ключи хранятся в директории «cert» на сервере.

## **3.2 Реализация клиентской части программного средства**

Клиентская часть проекта работает на чистом JavaScript, без использования каких-либо фреймворков. Для компоновки страниц использован шаблонизатор Handlebars.

Для работы с движками представлений в Express.js определен ряд глобальных настроек, которые мы можем установить. Прежде всего это настройка view engine, которая устанавливает используемый шаблонизатор, и views, которая устанавливает путь к папке с представлениями внутри проекта (если этот параметр не установлен, то по умолчанию используется папка с именем views).

Подключение шаблонизатора к серверной части представлено в листинге 3.8.

app.set('views', path.join(\_\_dirname, 'views'));

app.engine("hbs", expressHbs.engine({

  layoutsDir: "views/layouts",

  defaultLayout: "layout",

  extname: "hbs",

  partialsDir: "views/partials/",

}))

app.set('view engine', 'hbs');

Листинг 3.8 – Подключение шаблонизатора Handlebars

На Рисунке 3.2 отображена структура файлов клиентской части. Директория «views» содержит представления страниц, а также вложенные директории «partials» и «layouts».

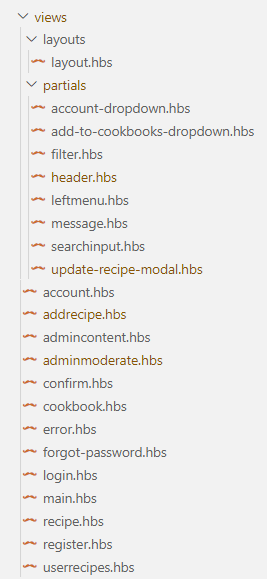


Рисунок 3.2 – Структура клиентской части приложения

В директории «partials» находятся все частичные представления приложения, которые встраиваются страницы. Они могут содержать как html, так и фрагменты скриптов на языке JavaScript. Пример частичного представления, реализующего поисковую строку, приведен в листинге 3.9.

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

    <input class="search-input" type="text" placeholder="Search"/>

    <img src="/images/search.svg" width="20px"/>

</div>

<script>

    $(document).ready(async function() {

       let input = document.getElementsByClassName('search-input')[0];

        input.oninput = async function() {

            await getRecipes(`search=${input.value}`);

        };

    });

</script>

Листинг 3.9 – Частичное представление, реализующее поисковую строку

В директории «layouts» хранится мастер-страница – каркас для всех страниц приложения, где описаны основные общие стили и html разметка.

Клиентская часть приложения представляет собой страницу, в которой запросы и обновления происходят асинхронно, без перезагрузки всей страницы.

Запросы к серверной части реализованы в с помощью Fetch API. Функции, отправляющие их, размещены как во встроенных в тело страниц скриптах, так и в отдельных JavaScript файлах. Пример одной из таких функций представлен в листинге 3.10.

async function postAddCookbookRecord(id) {

    return await fetch(`/cookbooks/add/${id}`, { method: 'POST', headers: requestHeaders, });

}

Листинг 3.10 – Функция postAddCookbookRecord, отправляющая запрос на сервер

## **3.3 Реализация базы данных**

Для хранения данных программного средства «Подбор рецептов» была создана реляционная база данных в СУБД PostgreSQL.

Ниже представлено описание реализации описанных ранее таблиц в терминах используемой СУБД.

Таблица users представляет собой описание пользователя приложения, с данными для входа. Содержит следующие поля:

* id, integer, автоинкрементируемое, primary key — уникальный номер записи в таблице;
* email, text, not null — адрес электронной почты пользователя;
* login, text, not null — логин пользователя;
* password, text, not null — пароль для входа в аккаунт;
* isGranted, boolean, not null, значение по умолчанию 0 (или false) — метка пользователя с повышенными привилегиями (администратор);
* resetPasswordLink, text, null — временная ссылка для сброса пароля пользователя;
* temporaryPassword, text, null — временный пароль пользователя.

Таблица meals хранит возможные приемы пищи. Содержит следующие поля:

* id, integer, автоинкрементируемое, primary key — уникальный номер записи в таблице;
* meal, text, not null — название приема пищи.

Таблица categories хранит категории, к которым можно отнести рецепты. Содержит следующие поля:

* id, integer, автоинкрементируемое, primary key — уникальный номер записи в таблице;
* category, text, not null — название категории.

Таблица ingredients хранит список всех возможных ингредиентов для рецепта. Содержит следующие поля:

* id, integer, автоинкрементируемое, primary key — уникальный номер записи в таблице;
* name, text, not null — название ингредиента;
* measurement, text, null — единица измерения количества ингредиента.

Таблица images хранит изображения, связанные с рецептом. Содержит следующие поля:

* id, integer, автоинкрементируемое, primary key — уникальный номер записи в таблице;
* uri, text, not null — адрес изображения;
* description, text, null — краткое описание изображения;
* recipeId, integer, not null, foreign key references recipes(id) — идентификатор рецепта.

Таблица cookbooks хранит список сохраненных пользователем рецептов. Содержит следующие поля:

* recipeId, integer, not null, foreign key references recipes(id) — идентификатор рецепта;
* userId, integer, not null, foreign key references users(id) — идентификатор пользователя.

Таблица recipe\_ingredients хранит список ингредиентов отдельного рецепта.

Содержит следующие поля:

* id, integer, автоинкрементируемое, primary key — уникальный номер записи в таблице;
* recipeId, integer, not null, foreign key references recipes(id) — идентификатор рецепта;
* ingredientId, integer, null, foreign key references ingredients(id) — идентификатор ингредиента;
* quantity, integer, not null — количество ингредиентов этого вида в рецепте.

Таблица recipes хранит список рецептов. Содержит следующие поля:

* id, integer, автоинкрементируемое, primary key — уникальный номер записи в таблице;
* categoryId, integer, not null, foreign key references categories(id) — идентификатор категории;
* authorId, integer, not null, foreign key references users(id) — идентификатор автора рецепта;
* mealId, integer, not null, foreign key references meals(id) — идентификатор приема пищи;
* datePublished, date, not null — дата публикации рецепта;
* timeToCook, integer, not null — время, затрачиваемое на приготовление рецепта;
* instruction, text, not null — инструкция приготовления;
* title, text, not null — наименование рецепта;
* isApproved, boolean, not null, значение по умолчанию 0 (или false) — метка подтвержденного администратором рецепта.

## **3.4 Реализация WebSocket**

Веб-сокеты в приложении реализованы с помощью библиотеки Socket.io, и используются для оповещения клиентов, подключенных к серверу, о публикации новых рецептов (после успешной модерации администратором).

Сервер веб-сокета присоединяется к https-серверу, как показано в листинге 3.11.

var server = https.createServer(httpsOptions, app);

socketapi.io.attach(server);

Листинг 3.11 – Реализация и обрабатываемые события на сокете клиента

Первым делом осуществляется подписка на событие «connection», чтобы клиенты смогли подключиться. В обработчике сервер принимает данные и преобразует их в объект JavaScript. Реализация серверной части веб-сокета представлен в листинге 3.12.

io.on('connection', function(client) {

    client.on('join', function(data) {

        client.on('recipe added', function(data) {

            client.broadcast.emit('broad', data);

        });

    });

});

Листинг 3.12 – Серверный веб-сокет и обрабатываемое событие

Код веб-сокета клиента представлен в листинге 3.13. Первым делом создается интерфейс конфигурации сервиса и модуль, который обеспечивает возможность конфигурирования при подключении. Далее описывается сам интерфейс (модель) сообщения веб-сокета. Далее создаются методы для подключения. Создается обработчик «on» для подписки на приходящие от сервера сообщения, а так же вызывается метод «emit» для их отправки.

let socket = io();

socket.on('connect', function(data) {

    socket.emit('join', 'client connected');

});

socket.on('broad', function(data) {

    notifyMe(data.user, data.title);

});

Листинг 3.13 – Реализация и обрабатываемые события на сокете клиента

Для отображения уведомления браузера на стороне клиента разработана функция notifyMe, код которой приведен в листинге 3.14.

function notifyMe(user, message) {

    if (!("Notification" in window)) {

        alert("This browser does not support desktop notification");

    }

    else if (Notification.permission === "granted") {

        var options = {

                body: message,

                dir : "ltr",

                icon: "../images/logo.png",

            };

        var notification = new Notification(user + " published a new recipe",options);

    }

    else if (Notification.permission !== 'denied') {

        Notification.requestPermission(function (permission) {

            if (!('permission' in Notification)) {

                Notification.permission = permission;

            }

            if (permission === "granted") {

                var options = {

                        body: message,

                        dir : "ltr",

                };

                var notification = new Notification(user + " published a new recipe", options);

            }

        });

    }

}

Листинг 3.14 – Реализация и обрабатываемые события на сокете клиента

Благодаря представленным выше событиям реализовано оповещение пользователей о публикации новых рецептов.

# **4 Тестирование**

В разработанном программном средстве предусматриваются многие негативные сценарии.

Прежде всего были проведены тесты аутентификации, такие как проверка на неверный ввод данных.

Создавая новую учетную запись, предназначенную для помещения в базу данных, при незаполненных обязательных полях на форме появляется сообщение, продемонстрированное на рисунке 4.1. Сообщения появляются по одному для каждого из необходимых полей.

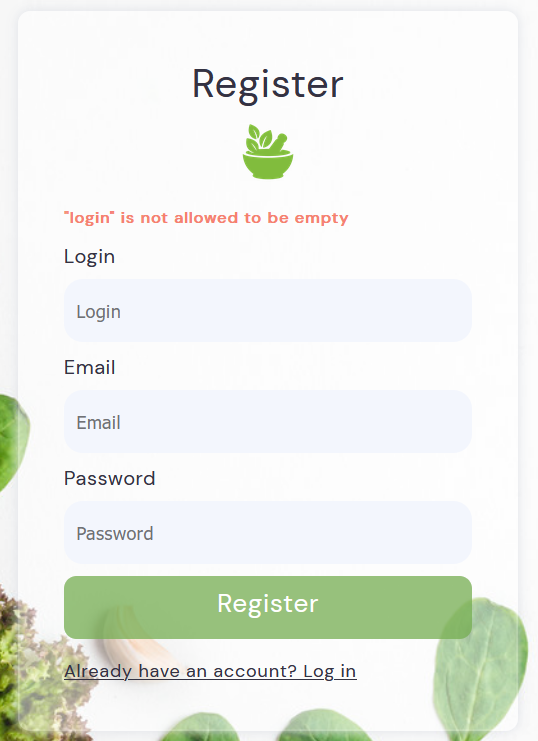


Рисунок 4.1 – Сообщение, возникающее при пустых обязательных полях

При попытке зарегистрировать пользователя с логином, который уже имеется в базе данных, выводится сообщение, отображенное на рисунке 4.2. Аналогичное сообщение отображается и при попытке зарегистрировать пользователя с существующим адресом электронной почты.

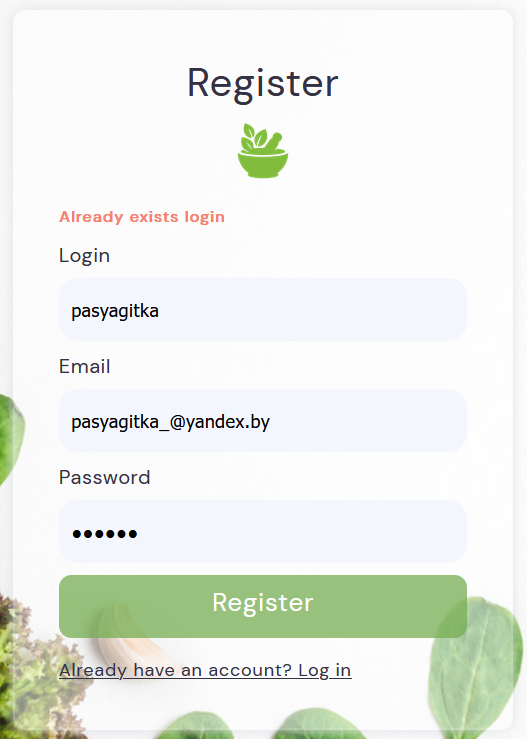


Рисунок 4.2 – Сообщение о существующем пользователе с введенным логином

В приложении присутствует аутентификация пользователя. В базе данных хранятся допустимые комбинации значений логина и пароля. Если при входе в систему указываются неверные данные, выводится сообщение об ошибке. Одно из таких сообщений продемонстрировано на рисунке 4.3.

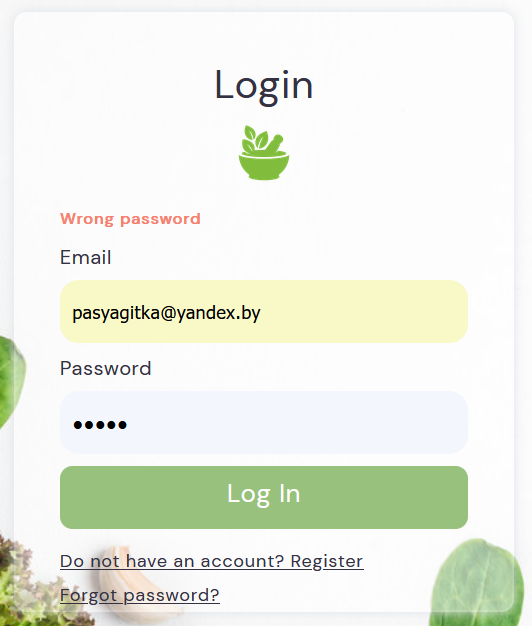


Рисунок 4.3 – Сообщение о неверных данных для входа

Если пользователь попробует создать рецепт, пропустив ввод обязательных полей, приложение вернет сообщение с ошибкой. Пример такого сообщения показан на рисунке 4.4.

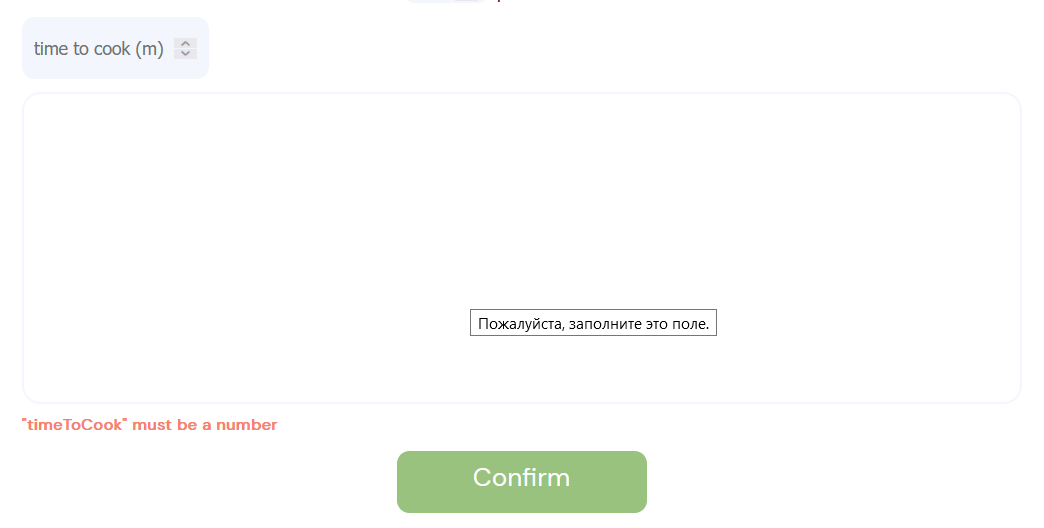


Рисунок 4.4 – Валидация создания рецепта

При попытке входа неавторизированного пользователя на недоступные ему страницы, возникнет окно с сообщением об ошибке, а затем пользователя перенаправит на страницу для входа в приложение. Сообщение об ошибке показано на рисунке 4.5.

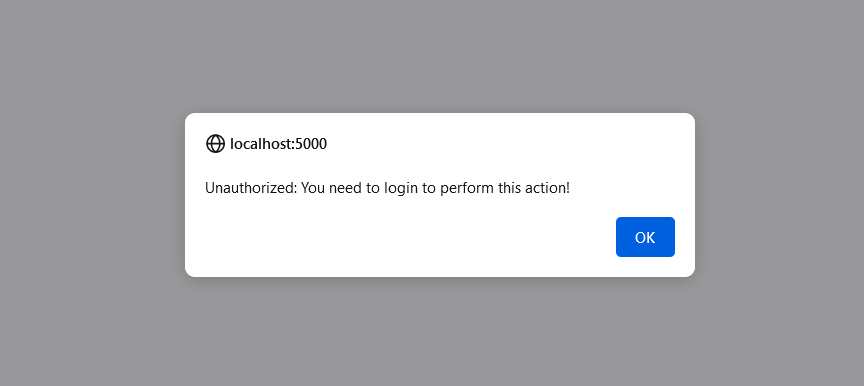


Рисунок 4.5 – Сообщение об ошибке для неавторизированного пользователя

При попытке доступа к ресурсу пользователя с недостаточными привилегиями, появится сообщение об ошибке, приведенное на рисунке 4.6, а затем пользователя перенаправит обратно на страницу, откуда он перешел.

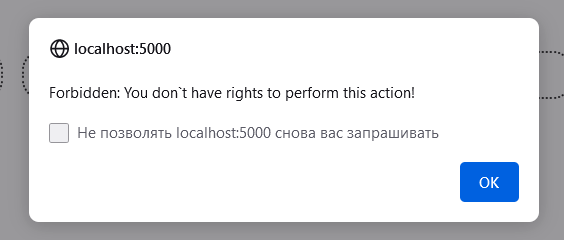


Рисунок 4.6 – Сообщение об ошибке для пользователя с недостаточными привилегиями

# **5 Руководство пользователя**

Для корректной работы приложения необходимо наличие браузера, а также подключение к сети Internet.

Фактически, как таковая установка для разработанного приложения не требуется. Для запуска приложения в локальной сети нужно установить конфигурацию, в частности скачать все используемые пакеты из npm (командой npm install). Так как база данных находится в облаке, ее конфигурация уже произведена и не требует каких либо действий при разворачивании сервера.

Чтобы иметь возможность пользоваться основным функционалом приложения, необходимо войти в аккаунт.

Для создания аккаунта необходимо ввести логин, адрес электронной почты и пароль, при этом логин и адрес электронной почты должны быть уникальными.

При входе в аккаунт необходимо ввести данные, указанные при регистрации.

## **5.1 Руководство для неавторизированного пользователя**

В разработанном веб-приложении пользователь является неавторизированным, если токен, хранящийся в LocalStorage браузера клиента, не является действительным, или же если этого токена вовсе нет.

Функционал неаутентифицированного пользователя представляет собой:

* выполнение регистрации, входа в приложение под своими учетными данными, сброс пароля;
* просмотр главной страницы со списком рецептов;
* поиск рецептов и фильтрация;
* получение браузерных уведомлений о публикации новых рецептов;
* просмотр подробной информации о некотором рецепте.

Форма регистрации приведена на рисунке 5.1. Все поля форме регистрации являются обязательными и валидируются. Регистрация осуществляется после нажатия на кнопку «Register».

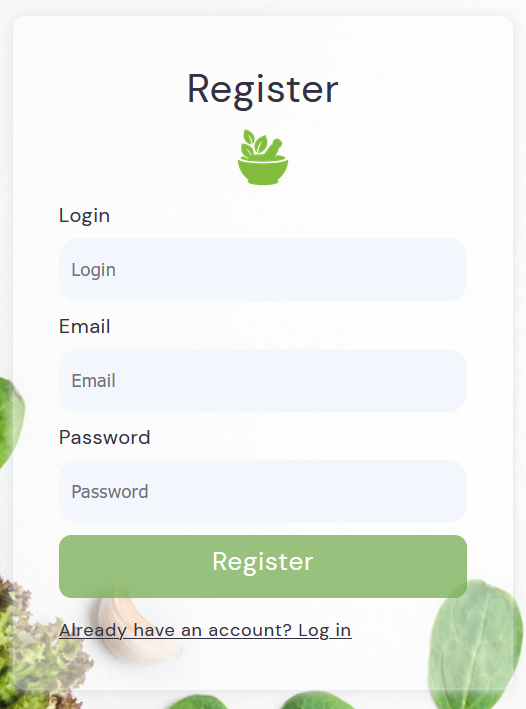


Рисунок 5.1 – Форма регистрации

Для успешной регистрации пользователю необходимо ввести валидный электронной почты, логин и пароль, где логин не может быть меньше 3 символов, а пароль меньше 6.

Форма входа в аккаунт приведена на рисунке 5.2. Для входа пользователю необходимо ввести использованный при регистрации адрес электронной почты, и соответствующий ему пароль.

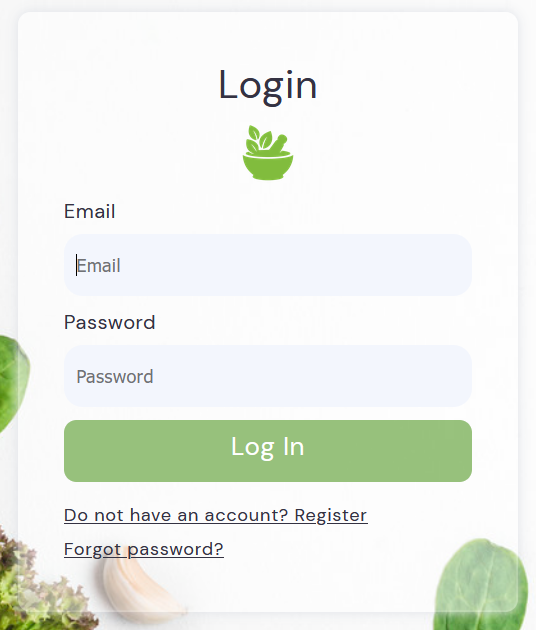


Рисунок 5.2 – Форма входа в аккаунт

Пользователь, забывший пароль, может восстановить его, перейдя по ссылке «Forgot password?» на форму восстановления, которая имеет вид, представленный на рисунке 5.3. В неё необходимо ввести адрес электронной почты от аккаунта, к которому был утерян доступ.

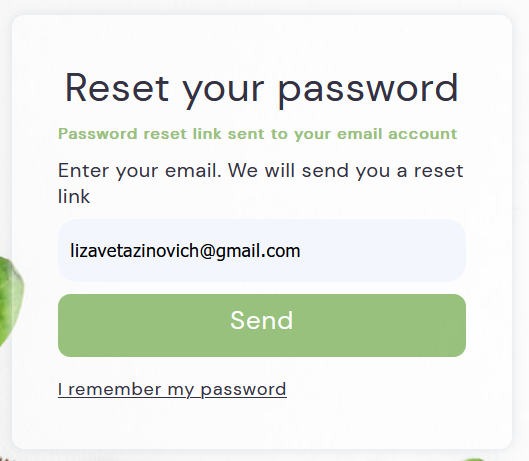


Рисунок 5.3 – Форма запроса восстановления пароля

После нажатия на кнопку «Send» на указанный электронный адрес придет письмо, отображенное на рисунке 5.4. Оно содержит временный пароль и ссылку для подтверждения сброса. Ссылка действует лишь один раз. После перехода по ней пользователь сможет войти в приложение с новым паролем.

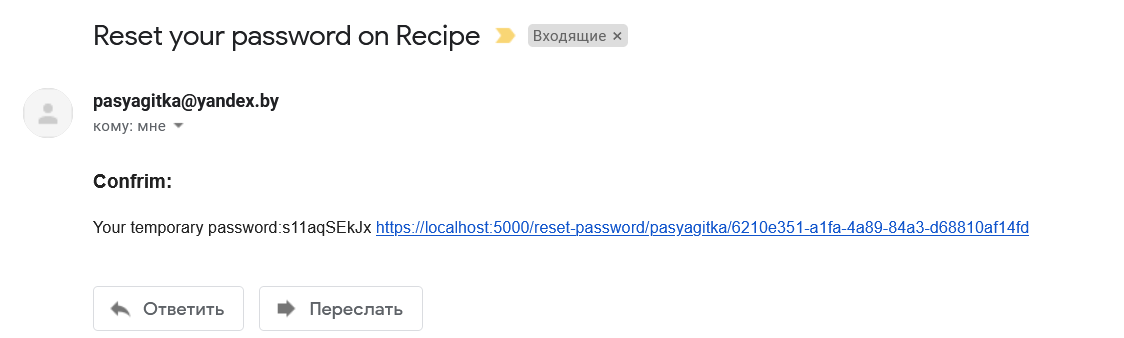


Рисунок 5.4 – Письмо с подтверждением сброса пароля

После подтверждения сброса пароля на электронный адрес пользователя придет письмо об успешном сбросе пароля, представленное на рисунке 5.5.

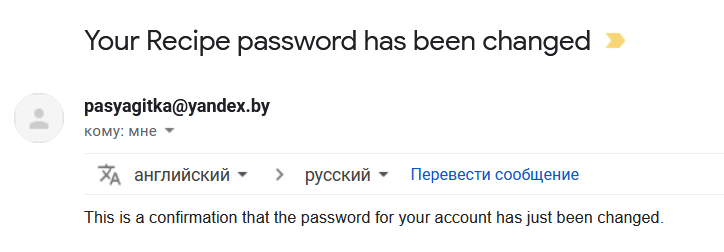


Рисунок 5.5 – Письмо с уведомлением об успешном сброса пароля

Пользователь, даже не входя в аккаунт, может попасть на главную страницу приложения, изображенную на рисунке 5.6. На главной странице он может выполнять поиск по рецептам, просматривать информацию о конкретном рецепте, а также применять фильтрацию по нажатию кнопки «Apply».

Критерии для фильтрации:

* категория блюда (например, салат, суп);
* прием пищи (например, завтрак, обед);
* включение ингредиентов (показывать рецепты, где обязательно содержатся отмеченные ингредиенты);
* исключение ингредиентов (не показывать рецепты, где содержатся отмеченные игредиенты).

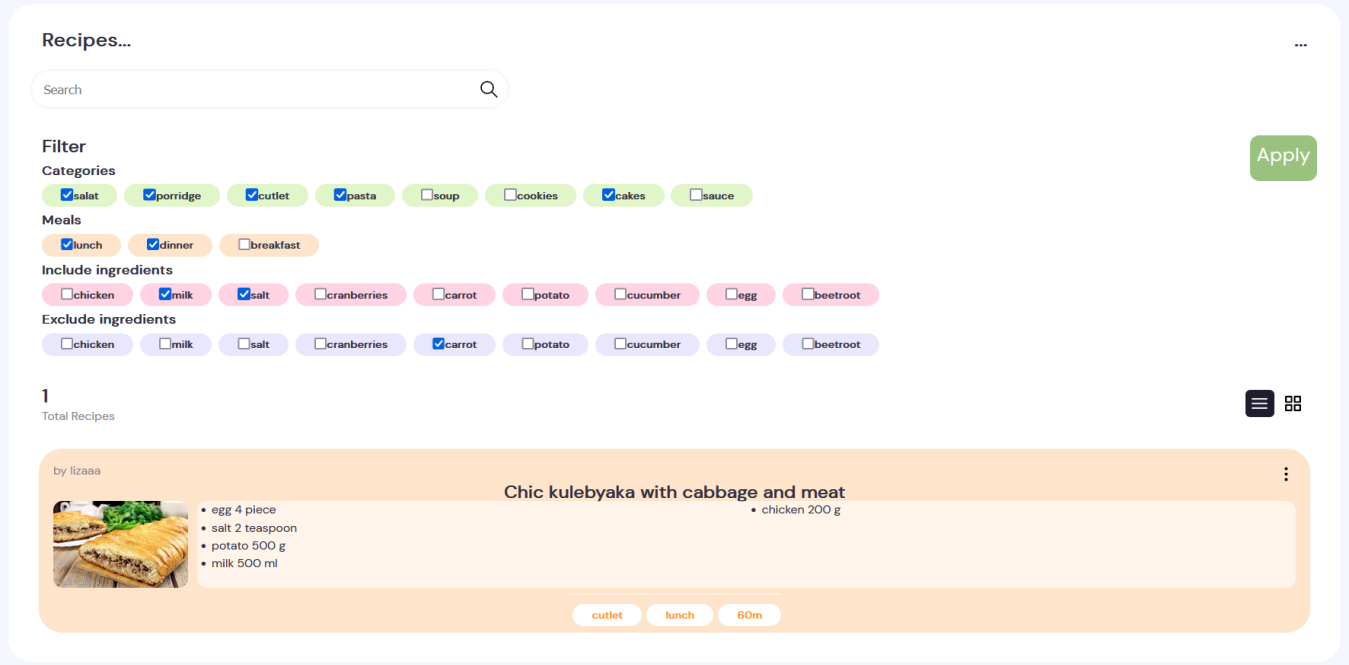


Рисунок 5.6 – Главная страница с применением фильтров рецептов

Если неавторизированный пользователь совершит попытку выполнить действия или открыть страницы, которые доступны только авторизованным пользователям, запрос будет перенаправлен на страницу входа в аккаунт.

## **5.2 Руководство для авторизированного пользователя**

Функции авторизированного пользователя содержат функции неавторизированного (не включая регистрацию, авторизацию и сброс пароля), а также следующие:

* смена пароля;
* сохранение рецепта в свою книгу рецептов, просмотр ее содержимого и удаление рецептов из нее;
* просмотр, создание, редактирование и удаление собственных рецептов.

Добавить рецепт в свою книгу рецептов можно нажав на кнопку «Save» в выпадающем меню на рецепте, удалить, нажав на «Remove», как представлено на рисунке 5.7.

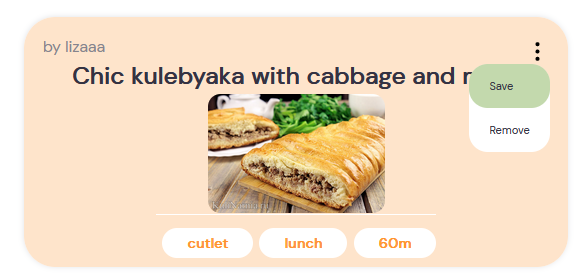


Рисунок 5.7 – Сохранение рецепта в книгу рецептов

Просмотреть книгу рецептов можно выбрав соответствующий пункт в выпадающем меню аккаунта.

Сменить пароль позволяет форма, представленная на рисунке 5.8.

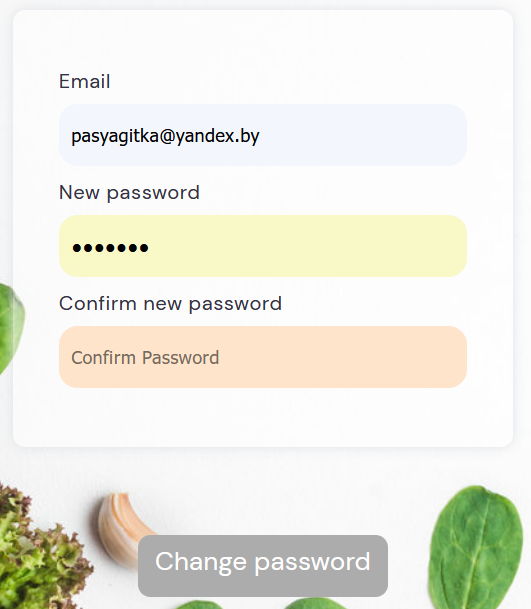


Рисунок 5.8 – Форма смены пароля

Форма для создания рецепта представлена в приложении Б. Практически идентична ей форма для редактирования рецепта, которую можно вызвать по нажатию кнопки «е» на странице рецептов пользователя, как показано на рисунке 5.9.

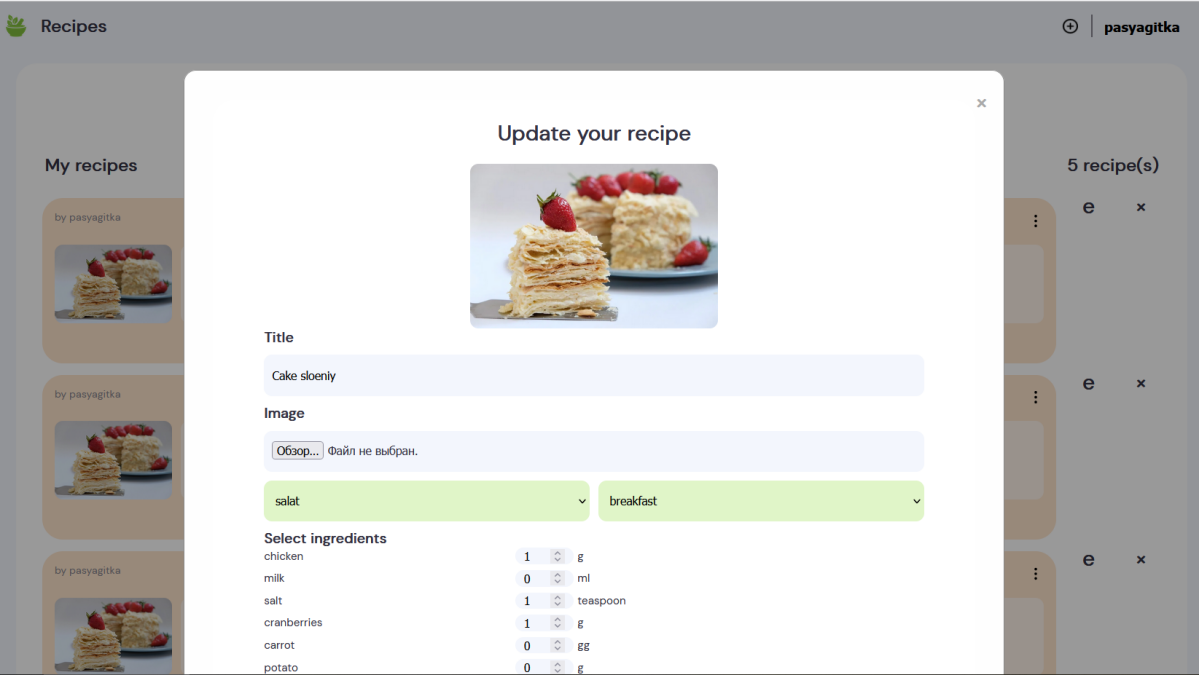


Рисунок 5.9 – Форма обновления рецепта

## **5.3 Руководство для администратора**

Помимо функций пользователя, администратор имеет право на:

* модерацию новых рецептов пользователей (одобрение, отклонение);
* удаление любого рецепта;
* просмотр, создание, редактирование и удаление приемов пищи, ингредиентов, категорий рецептов.

При попытке доступа обычного пользователя к функциям пользователя-администратора, первый получит сообщение об ошибке и будет перенаправлен на предыдущую страницу.

На рисунке 5.10 отображена панель рецептов для администратора, на которой можно одобрить или отклонить новый рецепт, а также удалить любой рецепт.

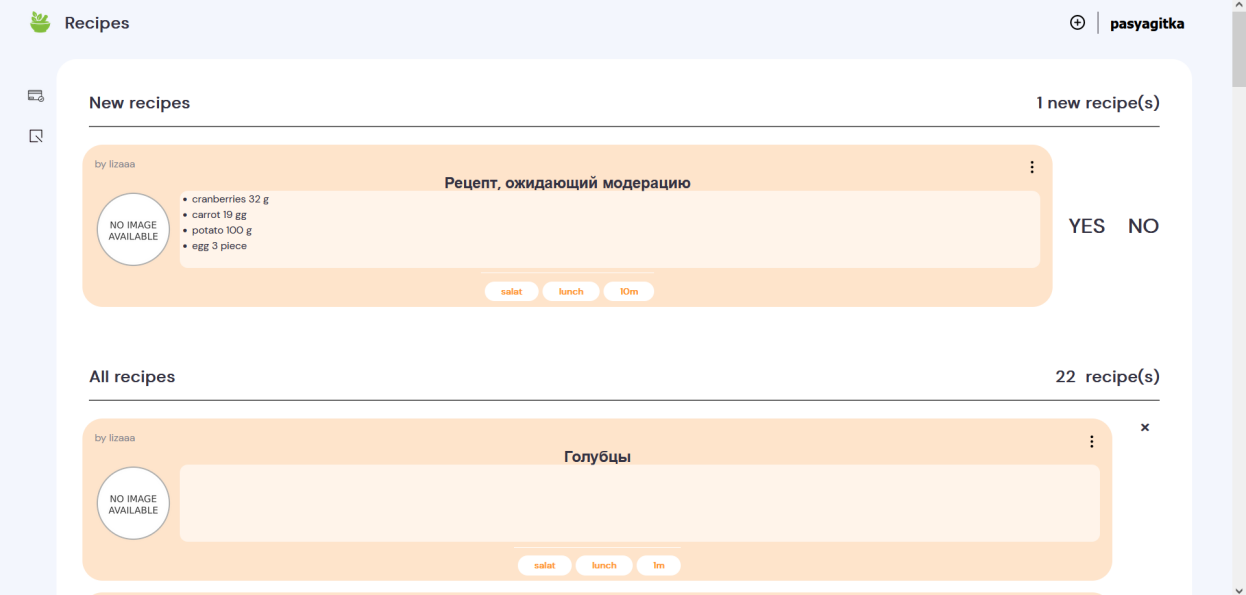


Рисунок 5.10 – Панель рецептов для администратора

При подтверждении администратором рецепта у всех подключенных к серверу клиентов возникает браузерное уведомление, показанное на рисунке 5.11.



Рисунок 5.11 – Уведомление о новом рецепте

На рисунке 5.12 отображена страница администратора, отвечающая за просмотр, создание, редактирование и удаление приемов пищи, ингредиентов, категорий рецептов.

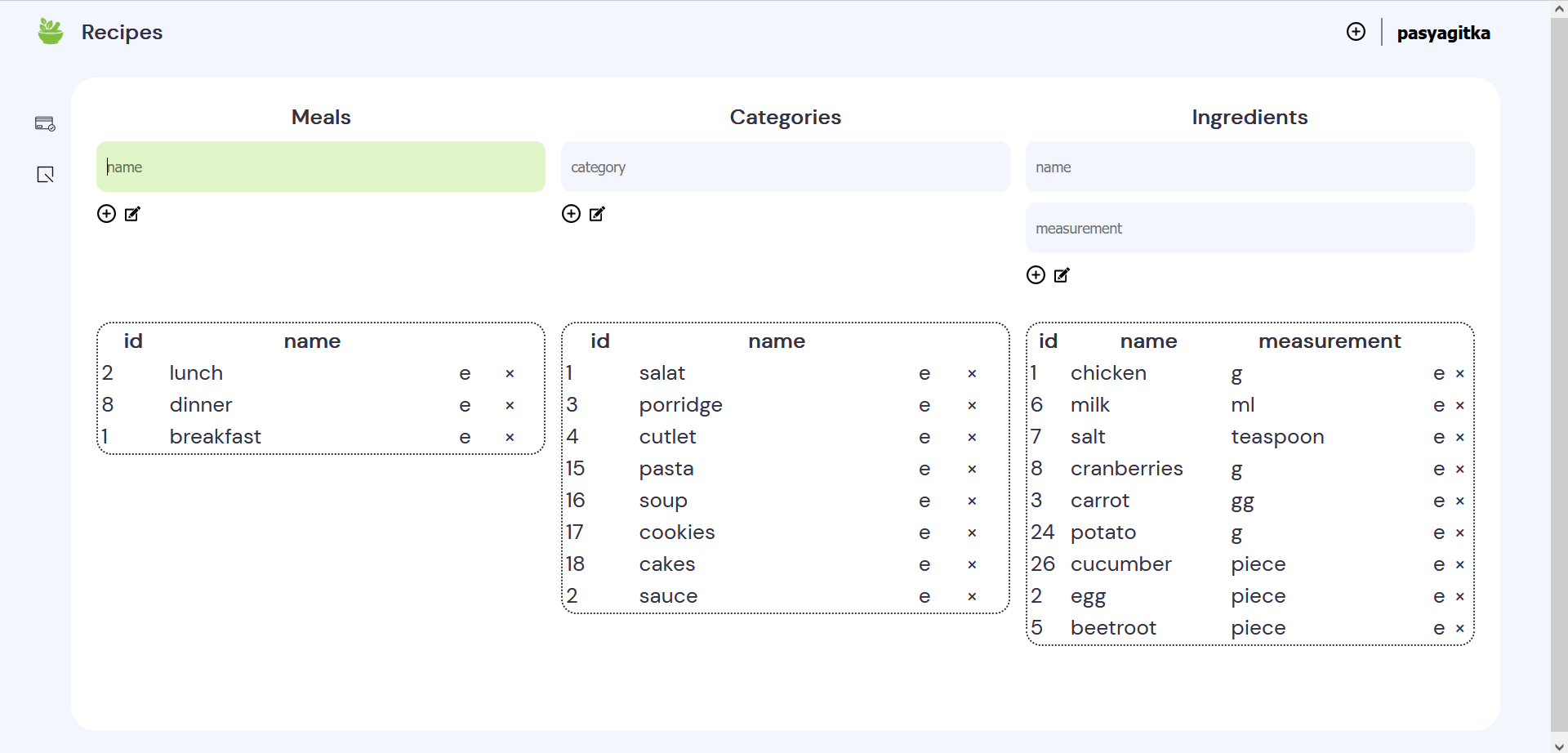


Рисунок 5.12 – Страница администратора, отвечающая за содержимое приложения

# **Заключение**

В результате курсовой работы было разработано Web-приложение «Подбор рецептов».

Оно основано на:

* платформе Node.js;
* фреймворке Express.js;
* ORM-библиотеке Sequelize;
* библиотеке для работы с WebSocket Socket.io;
* шаблонизаторе Handlebars.

Разработанное Web-приложение можно охарактеризовать следующим образом:

* функционирует по протоколам https и wss;
* имеет современный, асинхронный пользовательский интерфейс, интуитивно понятное и простое управление;
* обращается к базе данных PostgreSQL;
* предусматривает регистрацию, аутентификацию и авторизацию.

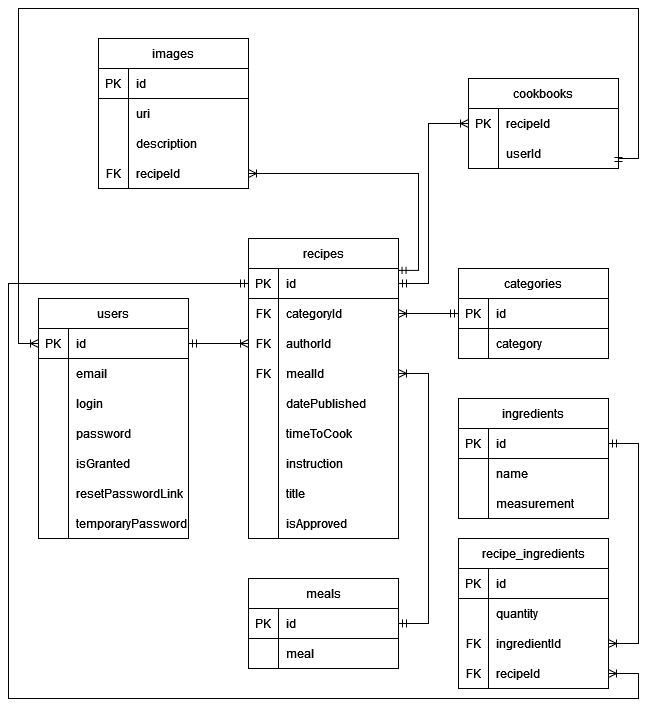
Программное средство удовлетворяет всем требованиям технического задания. Оно является логически завершённым и готово к использованию.

# **Список используемых источников**

1. About Node.js [Электронный ресурс] – Режим доступа: <https://nodejs.org/en/about/> – Дата доступа: 09.05.2022;
2. Socket.io [Электронный ресурс] – Режим доступа: <https://socket.io/docs/v4> – Дата доступа: 16.05.2022;
3. Express [Электронный ресурс] – Режим доступа: <https://expressjs.com/ru/> – Дата доступа: 10.05.2022.
4. Паттерн MVC [Электронный ресурс] – Режим доступа: https://metanit.com/web/nodejs/4.7.php – Дата доступа: 11.05.2022.
5. Sequelize: Model Basics [Электронный ресурс] – Режим доступа: https://sequelize.org/docs/v6/core-concepts/model-basics/ – Дата доступа: 05.04.2022.
6. NodeJsDev Express [Электронный ресурс] – Режим доступа: https://nodejsdev.ru/doc/express/ – Дата доступа: 17.04.2022.
7. Handlebars [Электронный ресурс] – Режим доступа: <https://handlebarsjs.com/> – Дата доступа: 01.05.2022.

# **ПРИЛОЖЕНИЕ А**

Рисунок 1 – Логическая схема базы данных



# **ПРИЛОЖЕНИЕ Б**

Рисунок 2 – Форма создания рецепта

