

Новосибирский Государственный Университет  
Факультет Информационных Технологий

# **Техническое описание проекта по курсу ООАД**

**FridgeMate**

**Студенты ФИТ НГУ  
Лукиша А.Р.  
группа 22204**

**Версия 1.0.1**

# Содержание

<b>1. Введение</b>	<b>4</b>
1.1 Цель	4
1.2 Область действия	4
1.3 Определения и сокращения	4
1.4 Краткое описание	5
<b>2. Предметная область проекта</b>	<b>6</b>
2.1 Существующие проблемы	6
2.2 Предполагаемое решение	7
<b>3. Требования к программному решению</b>	<b>7</b>
3.1 Роли	7
3.2 Функциональные требования для роли Неавторизованный пользователь	7
3.2.1 Просмотр главной страницы	7
3.2.2 Регистрация	8
3.2.3 Вход в аккаунт	8
3.3 Функциональные требования для роли Авторизованный пользователь	9
3.3.1 Просмотр главной страницы	9
3.3.2 Просмотр каталога рецептов	10
3.3.3 Просмотр конкретного рецепта	10
3.3.4 Добавление продуктов в холодильник	10
3.3.5 Уведомление о скором истечении срока годности продукта	11
3.3.6 Удаление продуктов из холодильника	11
3.3.7 Формирование списка покупок	12
3.3.8 Управление бюджетом на продукты	12
3.3.9 Планирование рациона	13
3.3.10 Просмотр холодильника	13
3.4 Нефункциональные требования	13
<b>4. Обзор архитектуры</b>	<b>14</b>
4.1 Компонентная модель системы	14
4.2 Основные микросервисы	15
4.2.1 Компонент 1: Сервис управления продуктами	15
4.2.2 Компонент 2: Сервис рекомендаций	16
4.2.3 Компонент 3: Сервис рецептов	16
4.2.4 Компонент 4: Сервис уведомлений	16
4.2.5 Компонент 5: Сервис управление бюджетом	17
4.2.6 Компонент 6: Сервис управление пользователями	17

4.3 Компоненты сторонних производителей	18
4.4 Схема развертывания приложения	18
<b>5. Допущения и ограничения</b>	<b>18</b>
5.1 Допущения	18
5.1.1 Доступ к интернету	18
5.1.2 Интуитивный интерфейс	19
5.1.3 Базовые компьютерные навыки	19
5.2 Ограничения	20
5.2.1 Ограниченная мобильная поддержка	20
5.2.2 Безопасность данных	20
5.2.3 Ограничения по времени	20
5.2.4 Потеря интереса пользователей	20
<b>6. Известные проблемы</b>	<b>20</b>
6.1 Невысокая производительность приложения	21
Уровень	21
6.2 Уведомления	21
Уровень	21
6.3 Консистентность данных	21
Уровень	21
Лист регистрации изменений	22
<b>Приложение</b>	<b>24</b>

# Техническое описание проекта по курсу ООАиД

## 1. Введение

### 1.1 Цель

Данный документ представляет собой техническое описание проекта *FridgeMate* и содержит основные требования к разрабатываемой в рамках проекта программной системе и описание архитектуры программного решения.

### 1.2 Область действия

Документ разработан в рамках проекта *FridgeMate* на основе стандартного шаблона и предназначен для использования студентами НГУ.

### 1.3 Определения и сокращения

Таблица 1: Определения и сокращения

Термин	Описание
<b>FridgeMate</b>	Веб-приложение для управления продуктами питания, отслеживания сроков годности и получения рецептов на основе имеющихся продуктов.
<b>RESTful API</b>	Интерфейс для взаимодействия с ресурсами через веб-протоколы, используя принципы REST для обмена данными между клиентом и сервером.
<b>Авторизация</b>	Процесс проверки прав доступа пользователя к определенным ресурсам или функциям системы после его идентификации.
<b>Микросервисная архитектура</b>	Архитектурный стиль, который структурирует приложение как набор небольших независимых сервисов, каждый из которых выполняет определенную функцию.
<b>PostgreSQL</b>	Объектно-реляционная система управления базами данных, используемая для хранения и управления данными приложения, включая продукты и рецепты.
<b>Spring Boot</b>	Фреймворк для создания микросервисов на Java, который упрощает настройку и развертывание приложений.
<b>Docker</b>	Платформа для контейнеризации приложений, позволяющая упрощать их развертывание, управление зависимостями и изоляцию окружений.
<b>Redis</b>	Резидентная система управления базами данных класса NoSQL, работающая со структурами данных типа «ключ — значение». Используется как для баз данных, так и для реализации кэшей, брокеров сообщений.
<b>Нейронные сети (AI)</b>	Алгоритмы, которые используются для анализа данных и генерации рекомендаций на основе имеющихся продуктов и предпочтений пользователей.

<b>Swagger</b>	Инструмент для подготовки документации к API и проведения тестов API. Он позволяет программистам, техническим писателям и тестирующим быстрее создавать, описывать и проверять программный интерфейс.
<b>Рекомендательная система</b>	Механизм, который анализирует данные о пользователях и их предпочтениях для предложения блюд и оптимальных решений на основе доступных продуктов.
<b>API Gateway</b>	Компонент, который управляет запросами к микро-сервисам, обеспечивая маршрутизацию, агрегацию и аутентификацию.
<b>React.js и React Native</b>	JavaScript-библиотека для создания пользовательских интерфейсов, позволяющая разрабатывать интерактивные и отзывчивые веб-приложения и фреймворк для разработки кроссплатформенных мобильных приложений с использованием React.
<b>Hibernate</b>	Фреймворк для работы с объектно-реляционным отображением (ORM) в Java, который упрощает взаимодействие с базами данных через JPA (Java Persistence API).
<b>JPA (Java Persistence API)</b>	Спецификация для управления постоянством данных в Java, позволяющая разработчикам работать с базами данных, используя объектно-ориентированный подход.
<b>PyTorch</b>	Фреймворк для глубокого обучения с открытым исходным кодом, используемый для разработки и обучения нейронных сетей, предоставляющий динамическое вычислительное график.
<b>HTTP (Hypertext Transfer Protocol)</b>	Протокол, используемый для передачи данных по сети, который обеспечивает обмен информацией между клиентом и сервером в веб-приложениях.
<b>HTTPS (Hypertext Transfer Protocol Secure)</b>	Безопасная версия HTTP, обеспечивающая защищенную передачу данных между клиентом и сервером с использованием шифрования, что предотвращает перехват информации.

## 1.4 Краткое описание

FridgeMate — это веб-приложение с интеграцией в мобильные устройства, созданное для управления продуктами питания и оптимизации их использования, ориентированное на людей с ограниченным бюджетом. Основной целевой аудиторией продукта предполагаются студенты вузов. Оно включает функционал отслеживания покупок, установления сроков годности, предложения рецептов на основе имеющихся ингредиентов, а также предоставляет рекомендации по сбалансированному питанию и управлению бюджетом на продукты.

## 2. Предметная область проекта

Приложения управления продуктами питания и рационализацией их использования с акцентом на планирование бюджета и здорового питания. Проект направлен на решение проблем, связанных с отслеживанием покупок, сроков годности продуктов и оптимизацией использования имеющихся ресурсов для составления рационов, что особенно актуально для студентов и людей с ограниченным бюджетом.

### 2.1 Существующие проблемы

1. Сложности в планировании питания на ограниченный бюджет
  - a. **Проблема:** Студенты и люди с ограниченными средствами часто не могут эффективно планировать свое питание в условиях жесткого бюджета, что приводит к ненужным расходам.
  - b. **Источник проблемы:** Ограниченные знания о том, как оптимально использовать продукты и составлять рацион на основе имеющихся ингредиентов.
  - c. **Решение:** FridgeMate предлагает персонализированные рецепты и советы по управлению бюджетом, которые учитывают имеющиеся продукты и финансовые ограничения пользователя.
2. Недостаток инструментов для поддержания сбалансированного питания в условиях ограниченного бюджета и времени
  - a. **Проблема:** В условиях ограниченного времени и бюджета люди часто выбирают менее полезные продукты, что негативно сказывается на их здоровье.
  - b. **Источник проблемы:** Мало инструментов, которые помогают быстро и удобно планировать сбалансированное питание.
  - c. **Решение:** FridgeMate предоставляет рекомендации по сбалансированному питанию (рецепты блюд), исходя из имеющихся ингредиентов и бюджета, что помогает пользователям придерживаться здорового рациона без дополнительных расходов.
3. Отсутствие платформ для управления продуктами с интеграцией мобильных устройств
  - a. **Проблема:** Многие приложения либо ограничиваются функциями списка покупок, либо не предоставляют комплексного подхода к управлению продуктами.
  - b. **Источник проблемы:** Люди часто не видят необходимости или не обладают инструментами для отслеживания сроков годности и рационального использования продуктов.
  - c. **Решение:** FridgeMate предлагает интеграцию с мобильными устройствами для легкого отслеживания покупок и сроков годности, а также рекомендации по использованию продуктов, чтобы минимизировать расходы.

## 2.2 Предполагаемое решение

Мы разрабатываем веб-приложение FridgeMate, которое поможет пользователям оптимизировать использование продуктов питания. Оно будет включать в себя функции для отслеживания покупок, управления сроками годности, предоставления рецептов на основе имеющихся ингредиентов и рекомендаций по питанию, а также инструменты для управления бюджетом. FridgeMate обеспечит удобный, современный интерфейс и интеграцию с мобильными устройствами, что делает его незаменимым помощником для студентов и людей с ограниченным бюджетом.

## 3. Требования к программному решению

Данный раздел описывает требования к программной системе, разрабатываемой в рамках проекта FridgeMate.

### 3.1 Роли

Роль - это что-то (например: другая система) или кто-то (например: человек) вне системы, которые взаимодействуют с ней. В предлагаемой к разработке системе идентифицированы следующие роли:

- **Неавторизованный пользователь** - не имеет доступа к основному контенту.
- **Авторизованный пользователь** – может использовать основной функционал приложения имеет полный доступ к контенту.

### 3.2 Функциональные требования для роли Неавторизованный пользователь

#### 3.2.1 Просмотр главной страницы

**Акторы:** Неавторизованный пользователь / Авторизованный пользователь

**Цель:** Просмотреть основное содержимое приложения, включая ключевые разделы

**Предусловие:** Приложение открыто, пользователь находится на главной странице

**Триггер:** Открытие приложения

**Сценарии:**

**Основной сценарий:**

- Пользователь открывает приложение и видит главную страницу.
- На странице отображаются основные функции приложения «Регистрация» и «Вход».
- Пользователь может выбрать регистрацию или вход в систему для получения доступа к полному функционалу.

**Альтернативный сценарий:**

- Пользователь открывает приложение.
- Пользователь закрывает приложение.

### 3.2.2 Регистрация

**Акторы:** Неавторизованный пользователь

**Цель:** создание учетной записи, чтобы стать авторизованным пользователем

**Предусловие:** открыто главное страница (Пользователь находится на странице, которая доступна для неавторизованного пользователя)

**Триггер:** нажатие кнопки «Регистрация», которая находится в левом верхнем углу

**Сценарии:**

**Основной сценарий:**

- Пользователь находится на главной странице.
- Пользователь нажимает кнопку «Регистрация».
- Открывается страница создания учетной записи.
- Перед пользователем появляется форма для регистрации с полями для ввода имени пользователя, электронной почты, пароля и подтверждения пароля.
- Пользователь заполняет все поля и нажимает кнопку для завершения регистрации.
- Если регистрация прошла успешно, пользователю выводится сообщение с просьбой подтвердить почту.
- Пользователь подтверждает электронную почту через ссылку, отправленную на указанную почту.
- Пользователь входит в созданный аккаунт и попадает на главную страницу приложения.

**Альтернативный сценарий:**

- Пользователь находится на главной странице.
- Пользователь нажимает на кнопку «Регистрация».
- Открывается страница создания учетной записи.
- Перед пользователем отображаются кнопки для регистрации через социальные сети (например, Gmail), а также поля для ввода стандартных данных.
- Пользователь выбирает регистрацию через соцсеть.
- Если у пользователя есть учетная запись в выбранной соцсети, и она не связана с уже существующей записью в системе FridgeMate, регистрация происходит автоматически.
- Пользователь входит в зарегистрированный аккаунт и переходит на главную страницу приложения.

### 3.2.3 Вход в аккаунт

**Акторы:** Неавторизованный пользователь

**Цель:** Войти в учетную запись для управления продуктами или получения рекомендаций

**Предусловие:** Открыта главная страница или доступная для неавторизованного пользователя страница

**Триггер:** Нажатие на кнопку «Вход» в левом верхнем углу

**Сценарии:**

#### Основной сценарий:

- Пользователь находится на главной странице.
- Пользователь нажимает кнопку «Вход».
- Открывается страница для входа в учетную запись.
- Перед пользователем появляются поля для ввода электронной почты или имени пользователя и пароля.
- Пользователь вводит данные и нажимает кнопку для входа.
- Если данные введены верно, пользователь входит в учетную запись и попадает на главную страницу приложения.

#### Альтернативный сценарий:

- Пользователь находится на главной странице.
- Пользователь нажимает кнопку «Вход».
- Открывается страница для входа.
- Пользователь выбирает вход через социальную сеть (например, Gmail).
- Если учетная запись в соцсети связана с учетной записью в FridgeMate, пользователь автоматически входит в систему.

### 3.3 Функциональные требования для роли Авторизованный пользователь

#### 3.3.1 Просмотр главной страницы

**Акторы:** Неавторизованный пользователь / Авторизованный пользователь

**Цель:** Просмотреть основное содержимое приложения, включая ключевые разделы

**Предусловие:** Приложение открыто, пользователь находится на главной странице

**Триггер:** Открытие приложения

**Сценарии:**

#### Основной сценарий:

- Авторизованный пользователь открывает приложение и видит главную страницу.
- На странице отображаются основные разделы: «Холодильник», «Список покупок», «Рецепты», «Управления бюджетом», а также уведомления (например, о продуктах с истекающим сроком годности)..
- Пользователь может перейти в любой из разделов через основную навигацию.
- Пользователю также могут быть предложены рецепты, основанные на имеющихся продуктах.

#### Альтернативный сценарий:

- Пользователь открывает приложение.
- Пользователь закрывает приложение.

### 3.3.2 Просмотр каталога рецептов

**Акторы:** Авторизованный пользователь

**Цель:** Изучение доступных рецептов для приготовления блюд на основе имеющихся ингредиентов

**Предусловие:** Открыта главная страница приложения

**Триггер:** Открытие раздела «Каталог рецептов»

**Сценарии:**

**Основной сценарий:**

- Пользователь с главной страницы заходит в раздел «Каталог рецептов».
- Перед пользователем открывается список рецептов, отсортированных по доступным ингредиентам или категориям (например, "Завтраки", "Обеды").
- Пользователь может просматривать рецепты, сортировать их по времени приготовления или предпочтениям.

### 3.3.3 Просмотр конкретного рецепта

**Акторы:** Авторизованный пользователь

**Цель:** Ознакомление с рецептом для приготовления блюд

**Предусловие:** Пользователь открыл каталог рецептов или ввел название блюда в строку поиска

**Триггер:** Пользователь нажимает на название рецепта

**Сценарии:**

**Основной сценарий:**

- Пользователь находится в каталоге рецептов или ввел название блюда в поиск.
- Пользователь выбирает рецепт, нажав на его название.
- Открывается страница с подробной информацией о рецепте, включающая шаги приготовления, время готовки, список ингредиентов.

**Альтернативный сценарий:**

- Пользователь переходит по прямой ссылке на страницу рецепта, не используя каталог.

### 3.3.4 Добавление продуктов в холодильник

**Акторы:** Авторизованный пользователь

**Цель:** Добавить продукт в личный список продуктов в холодильнике

**Предусловие:** Пользователь авторизован в системе и находится на главной странице приложения

**Триггер:** Нажатие на кнопку «Добавить продукт» или «Приобрёл» в разделе списка покупок.

**Сценарии:**

**Основной сценарий:**

- Пользователь нажимает кнопку «Добавить продукт».

- Открывается форма с полями для ввода информации о продукте (название, количество, срок годности, категория продукта).
- Пользователь заполняет поля и нажимает кнопку «Сохранить».
- Продукт добавляется в список «Холодильник», и пользователь видит обновленный список с добавленным продуктом.

**Альтернативный сценарий:**

- Пользователь сканирует штрих-код продукта с помощью мобильного устройства.
- Приложение автоматически заполняет информацию о продукте на основе базы данных.
- Пользователь подтверждает данные или вносит изменения и добавляет продукт в холодильник.

### **3.3.5 Уведомление о скором истечении срока годности продукта**

**Акторы:** Авторизованный пользователь

**Цель:** Получить уведомление о продуктах в разделе «Уведомления», срок годности которых скоро истекает

**Предусловие:** Продукты были ранее добавлены в список «Холодильник»

**Триггер:** Истечение определенного времени до окончания срока годности продукта

**Сценарии:**

**Основной сценарий:**

- Пользователь добавил продукт с указанием срока годности.
- Приложение отслеживает срок годности продукта.
- За несколько дней до истечения срока годности пользователь получает уведомление (push-уведомление) с напоминанием об истечении срока.
- Пользователь может увидеть список продуктов с истекающим сроком годности в разделе «Мои продукты».

### **3.3.6 Удаление продуктов из холодильника**

**Акторы:** Авторизованный пользователь

**Цель:** Удалить продукт из личного списка продуктов в холодильнике

**Предусловие:** Продукты были ранее добавлены в список «Холодильник»

**Триггер:** Нажатие на кнопку «Удалить продукт»

**Сценарии:**

**Основной сценарий:**

- Пользователь нажимает кнопку «Удалить продукт».
- Продукт удаляется из списка «Холодильник», и пользователь видит обновленный список без данного продукта.

**Альтернативный сценарий:**

- Срок годности продукта истекает.

- Продукт удаляется из списка «Холодильник», и пользователь видит обновленный список без данного продукта.

### 3.3.7 Формирование списка покупок

**Акторы:** Авторизованный пользователь

**Цель:** Создать список продуктов для покупки на основе текущего запаса и рецептов

**Предусловие:** Пользователь авторизован

**Триггер:** Пользователь открывает раздел «Список покупок»

**Сценарии:**

**Основной сценарий:**

- Пользователь из главной страницы переходит в раздел «Список покупок».
- Приложение предлагает рекомендации по продуктам для покупки на основе имеющихся запасов и рецептов, которые пользователь хотел бы приготовить.
- Пользователь может вручную добавить продукты в список покупок или выбрать из предложенного списка.
- Список сохраняется, и пользователь может его использовать при посещении магазина (например, с помощью мобильного приложения).
- При нажатии кнопки «Приобрёл» пользователь осуществляет переход к сценарию «Добавление продуктов в холодильник».

**Альтернативный сценарий:**

- Приложение автоматически создает список покупок на основе продуктов, срок годности которых истекает, и доступных рецептов, которые можно приготовить.
- Пользователь может редактировать список перед покупкой.

### 3.3.8 Управление бюджетом на продукты

**Акторы:** Авторизованный пользователь

**Цель:** Управлять и отслеживать затраты на продукты

**Предусловие:** Пользователь авторизован и регулярно добавляет продукты в приложение

**Триггер:** Пользователь открывает раздел «Управления бюджетом»

**Сценарии:**

**Основной сценарий:**

- Пользователь из главной страницы переходит в раздел «Управления бюджетом».
- Пользователь вводит информацию о покупках продуктов, включая цену и количество.
- Приложение предлагает рекомендации по продуктам для покупки на основе имеющихся запасов и рецептов, которые пользователь хотел бы приготовить.
- Пользователь получает рекомендации по оптимизации покупок на основе проанализированных данных.

**Альтернативный сценарий:**

- Пользователь может настроить ежемесячный бюджет на продукты.

- Приложение отслеживает, как пользователь укладывается в бюджет, и отправляет push-уведомления о том, когда расходы приближаются к установленному лимиту.

### 3.3.9 Планирование рациона

**Акторы:** Авторизованный пользователь

**Цель:** Запланировать рацион на неделю или месяц

**Предусловие:** Пользователь авторизован и ввел свои предпочтения

**Триггер:** Пользователь открывает раздел «Планирование рациона»

**Сценарии:**

**Основной сценарий:**

- Пользователь из главной страницы переходит в раздел «Планирование рациона».
- Приложение предлагает пользователю выбрать рецепты для каждого дня недели на основе доступных ингредиентов и предпочтений.
- Пользователь может настроить план, добавив или удалив блюда, и утвердить план на неделю.
- Приложение автоматически создает список покупок на основе запланированных рецептов и недостающих ингредиентов.

### 3.3.10 Просмотр холодильника

**Акторы:** Авторизованный пользователь

**Цель:** Посмотреть текущий список продуктов, находящихся в холодильнике

**Предусловие:** Пользователь авторизован

**Триггер:** Пользователь открывает раздел «Холодильник»

**Сценарии:**

**Основной сценарий:**

- Пользователь авторизован и находится на главной странице приложения.
- Пользователь переходит в раздел «Холодильник».
- Приложение отображает список продуктов, находящихся в холодильнике.
- Пользователь видит информацию о каждом продукте, включая название, количество, категорию и срок годности.
- Пользователь может отсортировать продукты по категориям или сроку годности.
- Пользователь может добавить и удалить продукты находящиеся в холодильнике.

## 3.4 Нефункциональные требования

### 1. Производительность

- a. **Время отклика системы:** Приложение должно обеспечивать быструю загрузку страниц, функций управления продуктами и предложении рецептов для удобного и плавного пользовательского опыта.

b. **Обработка задач:** Все процессы, включая расчет рецептов и уведомления о сроках годности, должны обрабатываться быстро, чтобы минимизировать задержки и не вызывать дискомфорта у пользователей.

## 2. Масштабируемость

a. **Гибкость при росте:** Система должна быть способна обрабатывать увеличение количества пользователей, особенно при масштабировании до нескольких тысяч студентов и поддерживать их активность без снижения производительности.

## 3. Ограничения по используемым компонентам

a. **Ограничения по объему данных:** Приложение должно эффективно работать с ограниченными объемами данных (например, поддержка до 1000 пользователей на начальных этапах и до определенного количества продуктов, рецептов и уведомлений).

b. **Минимальные серверные требования:** Приложение должно работать на сервере с минимальными требованиями к ресурсам для снижения стоимости поддержки и обеспечения надежной работы.

## 4. Удобство использования

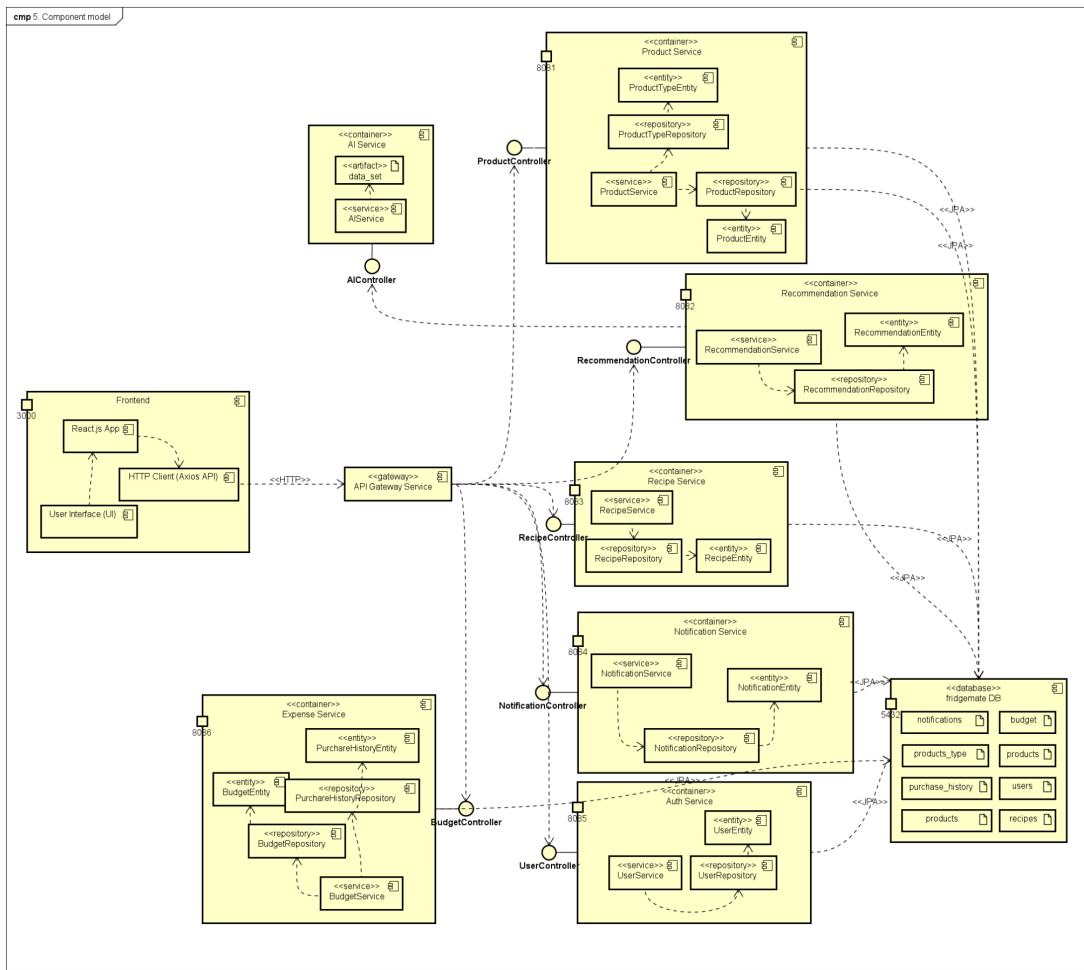
a. **Простота и доступность интерфейса:** Интерфейс должен быть интуитивно понятным и удобным для пользователей всех уровней, включая студентов, которые могут иметь минимальный опыт работы с подобными приложениями.

# 4. Обзор архитектуры

Этот раздел описывает архитектуру системы.

## 4.1 Компонентная модель системы

Архитектура системы FridgeMate построена на микросервисной архитектуре с использованием Spring Boot на стороне бэкенда и React.js на стороне фронтенда. Система включает компоненты для управления продуктами, рекомендаций, уведомлений и обработки данных. Все микросервисы взаимодействуют между собой через REST API и управляются через API Gateway. Контейнеризация осуществляется с помощью Docker для обеспечения масштабируемости и изоляции сервисов.



## 4.2 Основные микросервисы

### 4.2.1 Компонент 1: Сервис управления продуктами

**Предназначение:** Отвечает за обработку данных о продуктах, сроках годности, добавлении и удалении продуктов из базы данных.

#### Package Diagram:

- **ProductController:** отвечает за взаимодействие с пользователем.
- **ProductService:** бизнес-логика для работы с продуктами.
- **ProductRepository:** взаимодействие с базой данных PostgreSQL.
- **ProductEntity:** класс-сущность, хранящий информацию о продукте.
- **ProductDto:** класс-сущность, хранящий информацию о продукте для упрощённой передачи по json.

**Интерфейсы для взаимодействия:** REST API (для взаимодействия с фронтеном), Hibernate (взаимодействие с базой данных).

#### 4.2.2 Компонент 2: Сервис рекомендаций

**Предназначение:** Отвечает за нейронные сети, построенные на Python и PyTorch, которые анализируют паттерны покупок пользователей и предоставляют персонализированные рекомендации.

**Package Diagram:**

- RecommendationController: отвечает за получение запросов от других сервисов.
- RecommendationEngine: реализует AI на основе данных пользователей и продуктов.
- RecommendationRepository: взаимодействие с базой данных PostgreSQL для хранения и обновления данных о предпочтениях пользователей, результатах работы нейронной сети.
- RecommendationEntity: класс-сущность для хранения рекомендаций
- RecommendationDto: класс-сущность, хранящий информацию о рекомендации для упрощённой передачи по json.

**Интерфейсы для взаимодействия:** REST API (для связи с другими сервисами), Hibernate (взаимодействие с базой данных).

#### 4.2.3 Компонент 3: Сервис рецептов

**Предназначение:** Отвечает за обработку рецептов на основе имеющихся продуктов, предоставление рецептов пользователям.

**Package Diagram:**

- RecipeController: обеспечивает взаимодействие с фронтеном.
- RecipeService: бизнес-логика, предлагающая рецепты на основе данных о продуктах.
- RecipeRepository: взаимодействие с базой данных PostgreSQL для хранения рецептов.
- RecipeEntity: класс-сущность, представляющий рецепт блюда.
- RecipeDto: класс-сущность, хранящий информацию о рецепте для упрощённой передачи по json.

**Интерфейсы для взаимодействия:** REST API (для связи с другими сервисами), Hibernate (взаимодействие с базой данных), интеграция с сервисом управления продуктами для получения данных о доступных ингредиентах.

#### 4.2.4 Компонент 4: Сервис уведомлений

**Предназначение:** Отвечает за отправку пользователям уведомлений о скором истечении срока годности продуктов и других важных событиях.

**Package Diagram:**

- NotificationController: отвечает за взаимодействие с фронтеном.
- NotificationService: бизнес-логика для формирования уведомлений.

- NotificationRepository: взаимодействие с базой данных PostgreSQL для хранения информации об уведомлениях.
- NotificationEntity: класс-сущность, представляющий push-уведомление.
- NotificationDto: класс-сущность, хранящий информацию об уведомлении для упрощённой передачи по json.

**Интерфейсы для взаимодействия:** REST API (для передачи данных и взаимодействия с другими микросервисами), Hibernate (взаимодействие с базой данных), интеграция с системой сообщений Kafka для асинхронной доставки уведомлений.

#### 4.2.5 Компонент 5: Сервис управление бюджетом

**Предназначение:** Отвечает за управления бюджетом, предназначен для отслеживания и оптимизации расходов на продукты питания. Он предоставляет пользователям возможность вести учет своих покупок, задавать лимиты на расходы и получать рекомендации для более эффективного использования средств при покупке продуктов.

**Package Diagram:**

- BudgetController: отвечает за получение запросов от других сервисов.
- BudgetService: бизнес-логика создания бюджета, расчета оставшихся средств, обновления записей о покупках и расчета статистики по расходам.
- BudgetRepository: взаимодействие с базой данных PostgreSQL для хранения информации из таблицы бюджета.
- BudgetEntity: класс-сущность, представляющий бюджет пользователя.
- PurchaseHistoryService: бизнес-логика работы с историей покупок.
- PurchaseHistoryRepository: взаимодействие с базой данных PostgreSQL для управления записями о покупках, с возможностью поиска и сортировки по дате, типу продуктов или сумме покупки.
- PurchaseHistoryEntity: класс для хранения истории покупок пользователя.
- BudgetDto: класс-сущность, хранящий информацию о бюджете для упрощённой передачи по json.

**Интерфейсы для взаимодействия:** REST API (для связи с другими сервисами), Hibernate (взаимодействие с базой данных), интеграция с сервисом рекомендаций который на основе этих данных предлагает оптимальные покупки или бюджетные рецепты, интеграция с сервисом уведомлений для отправки уведомления пользователю при превышении бюджета или достижении определенных пороговых значений.

#### 4.2.6 Компонент 6: Сервис управление пользователями

**Предназначение:** Этот компонент занимается администрированием пользователей в системе, предоставляя возможности для создания, редактирования и удаления учетных записей. Он отвечает за авторизацию, аутентификацию и управление ролями пользователей, обеспечивая безопасный доступ к функциям приложения в зависимости от прав.

#### Package Diagram:

- UsersController: отвечает за получение запросов от других сервисов.
- UserService: бизнес-логика управления учетными записями: регистрация, вход, сброс пароля, изменение данных и управление ролями.
- UsersEntity: класс-сущность, представляющий пользователя.
- UsersDto: класс-сущность, хранящий информацию о пользователе для упрощённой передачи по json.

**Интерфейсы для взаимодействия:** REST API (для связи с другими сервисами), Hibernate (взаимодействие с базой данных), интеграция с сервисом уведомлений для отправки уведомления пользователю при превышении бюджета или достижении определенных пороговых значений.

### 4.3 Компоненты сторонних производителей

**Список компонентов:** Java Spring Boot (используется для реализации микросервисов и обеспечения взаимодействия между ними), PostgreSQL (база данных для хранения информации), Hibernate (используется для удобного взаимодействия Java с базой данных), React.js + React Native (используется для создания интерактивного пользовательского интерфейса), Python + PyTorch (для работы нейронной сети и анализа данных), Docker (контейнеризация микросервисов для их изоляции и удобного развертывания), Redis (используется для кэширования внутри бэкенда), Swagger (используется для составления документации по API сервисов), EmailJS (сторонний веб сервис для отправки сообщений с подтверждением).

### 4.4 Схема развертывания приложения

Приложение развертывается в виде микросервисов, каждый из которых изолирован в отдельном Docker-контейнере. Сервисы взаимодействуют друг с другом через API Gateway, который маршрутизирует запросы между фронтендом и микросервисами.

Все компоненты взаимодействуют через HTTPS и REST API, обеспечивая надежную и гибкую архитектуру системы.

## 5. Допущения и ограничения

### 5.1 Допущения

#### 5.1.1 Доступ к интернету

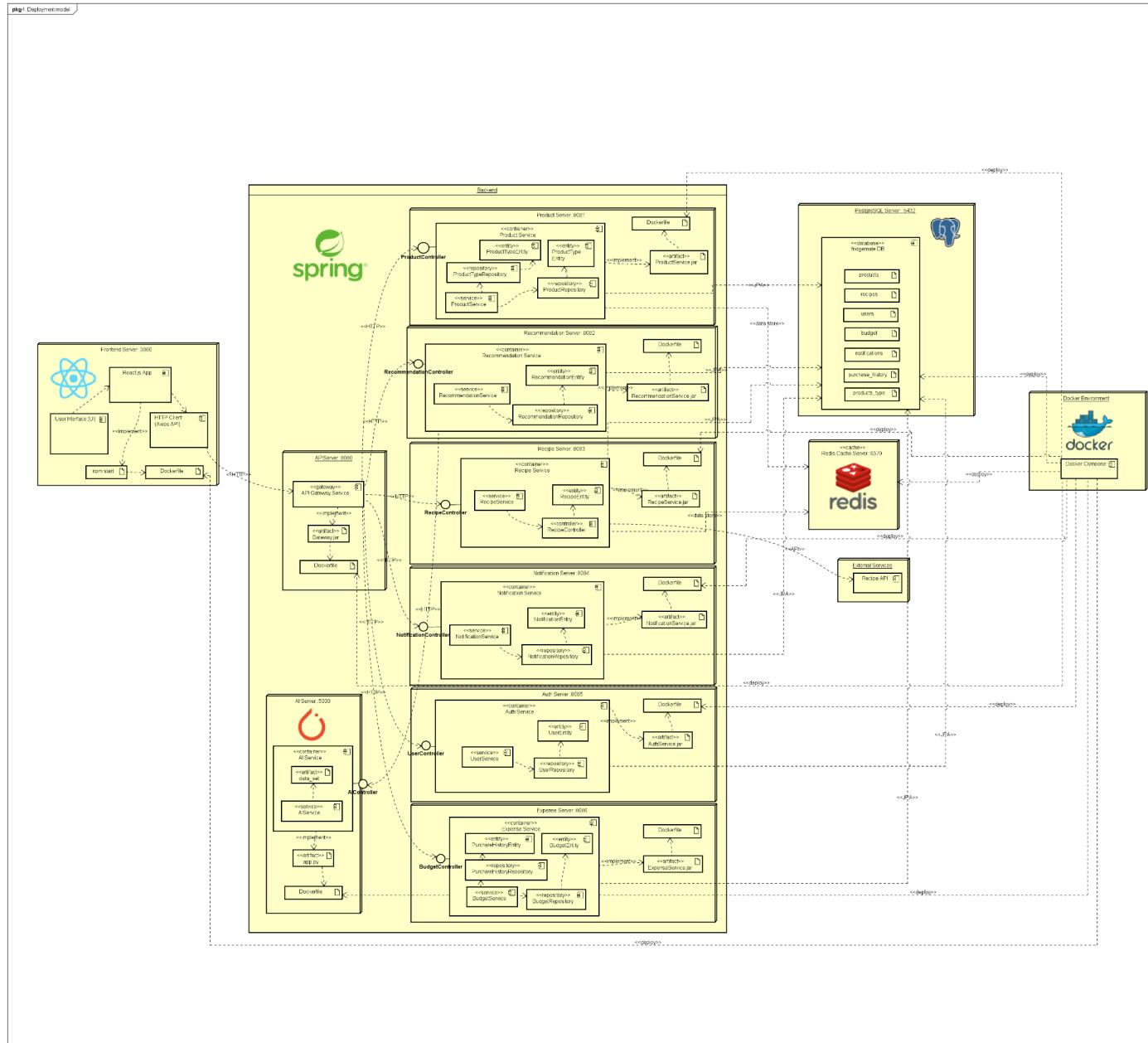
Предполагается, что все пользователи приложения FridgeMate имеют стабильное интернет-соединение для взаимодействия с платформой, включая доступ к веб-версии, мобильному приложению и сервисам рекомендаций.

## 5.1.2 Интуитивный интерфейс

Ожидается, что пользователи смогут легко ориентироваться в интерфейсе приложения благодаря его продуманному дизайну.

## 5.1.3 Базовые компьютерные навыки

Допускается, что пользователи обладают базовыми навыками работы с компьютером и мобильными устройствами, что поможет им эффективно использовать приложение.



## 5.2 Ограничения

### 5.2.1 Ограниченная мобильная поддержка

На начальном этапе приложение будет иметь веб-версию и поддержку для мобильных устройств через кроссплатформенные решения (например, React Native), но полноценное мобильное приложение может быть ограничено в функциональности или доступности на разных платформах.

### 5.2.2 Безопасность данных

Хотя будут приняты меры по обеспечению безопасности пользовательских данных (шифрование, проверка подлинности), полностью гарантировать отсутствие уязвимостей невозможно, учитывая разнообразие атак и постоянно меняющиеся угрозы в киберпространстве.

### 5.2.3 Ограничения по времени

Временные рамки разработки могут ограничить объем тестирования всех компонентов системы, что может привести к оставшимся багам или не до конца оптимизированным функциям, влияющим на пользовательский опыт.

### 5.2.4 Потеря интереса пользователей

Существует вероятность, что пользователи могут потерять интерес к платформе, если функционал, связанный с предложением рецептов или оптимизацией использования продуктов, окажется недостаточно полезным, или если не будет интегрировано достаточно новых функций, стимулирующих их активность в приложении.

## 6. Известные проблемы

Ниже приводятся известные на данный момент проблемы и недоработки выработанного программного решения, а также возможные пути их устранения в последующих итерациях проекта.

### 6.1 Невысокая производительность приложения

<b>Проблема</b>	Высокая нагрузка на микросервисы при большом количестве пользователей может привести к снижению скорости ответа.
<b>Уровень</b>	10 (высокий)
<b>Влияние на проект</b>	Невозможность использования системы при числе пользователей более 10000.
<b>Пути решения</b>	Оптимизация кэша в Redis, масштабирование сервисов с использованием Kubernetes.

### 6.2 Уведомления

<b>Проблема</b>	Проблемы с отправкой уведомлений, связанные с Redis или задержками в обработке задач.
<b>Уровень</b>	2 (низкий)
<b>Влияние на проект</b>	Уведомления о скором истечении срока годности продуктов приходят с опозданием.
<b>Пути решения</b>	Настройка временных ограничений для задач, добавление fallback-логики.

### 6.3 Консистентность данных

<b>Проблема</b>	Несогласованность данных между сервисами из-за асинхронного взаимодействия.
<b>Уровень</b>	4 (средний)
<b>Влияние на проект</b>	Удалённый продукт продолжает отображаться в рекомендациях.

<b>Пути решения</b>	Реализация механизма событий EDA, настройка транзакций для критичных операций.
---------------------	--

## Лист регистрации изменений

Дата	Версия	Описание	Автор

[В качестве описания версии можно указывать какие изменения/дополнения были сделаны в этой версии по отношению к предыдущей.]

## Лист регистрации проверок

Дата	Версия	Описание	Автор

[Здесь описываются результаты проверки документа. Для каждой проверки указывается число, версия документа, описание результатов проверки и имя человека, который делал проверку.]

## Приложение

