**Техническое описание проекта по курсу ООАД**

**FoodStorage**

**Студенты ФИТ НГУ**

**Софронова Дарья,**

**Кобелева Анастасия**

**группа 18208**

**Версия 1.0.1**

Содержание

[1. Введение](#_Toc59560251)

[1.1 Цель](#_Toc59560252)

[1.2 Область действия](#_Toc59560253)

[1.3 Определения и сокращения](#_Toc59560254)

[1.4 Ссылки](#_Toc59560255)

[1.5 Краткое описание](#_Toc59560256)

[2. Предметная область проекта](#_Toc59560257)

[2.1 Существующие проблемы](#_Toc59560258)

[2.2 Предполагаемое решение](#_Toc59560259)

[3. Требования к программному решению](#_Toc59560260)

[3.1 Роли](#_Toc59560261)

[3.2 Функциональные требования для роли Пользователь](#_Toc59560262)

[3.2.1 Посмотреть список рецептов](#_Toc59560263)

[3.2.2 Посмотреть рецепт](#_Toc59560264)

[3.2.3 Добавить рецепт](#_Toc59560265)

[3.2.4 Посмотреть продукты](#_Toc59560266)

[3.2.5 Добавить продукт](#_Toc59560267)

[3.2.6 Сделать продукт постоянно необходимым](#_Toc59560268)

[3.2.7 Посмотреть продукты, которые скоро испортятся](#_Toc59560269)

[3.2.8 Найти рецепт под портящиеся продукты](#_Toc59560270)

[3.2.9 Посмотреть список покупок](#_Toc59560271)

[3.2.10 Добавить продукт в список покупок](#_Toc59560272)

[3.2.11 Дополнить список покупок по рецепту](#_Toc59560273)

[3.3 Нефункциональные требования?](#_Toc59560274)

[4. Обзор архитектуры](#_Toc59560275)

[4.1.1 Компонентная модель системы](#_Toc59560276)

[4.1.1.1 Model](#_Toc59560277)

[4.1.1.2 GoodTypeDiagram](#_Toc59560278)

[4.1.1.3 GoodDiagram](#_Toc59560279)

[4.1.1.4 RecipeDiagram](#_Toc59560280)

[4.1.1.5 WishListDiagram](#_Toc59560281)

[5. Допущения и ограничения ?](#_Toc59560282)

[6. Известные проблемы](#_Toc59560283)

[6.1 Невысокая производительность приложения](#_Toc59560284)

Техническое описание проекта по курсу ООАД

# Введение

## Цель

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

## Область действия

Документ разработан в рамках проекта *FoodStorage* на основе стандартного шаблона *Inteks SEP* и предназначен для использования студентами ФИТ и преподавателями курса ООАД.

## Определения и сокращения

**Таблица 1: Определения и сокращения**

|  |  |
| --- | --- |
| **Термин** | **Описание** |
| КБЖУ | Количество калорий, белков, жиров и углеводов в продукте |
|  |  |

## Ссылки

В тексте содержатся ссылки на следующие документы:

1. <Имя файла документа>, v<версия> - <описание документа>

## Краткое описание

Содержание данного документа построено таким образом, чтобы дать ответ на следующие вопросы:

* Какие проблемы предметной области должен решать будущий программный продукт
* Посредством какой функциональности системы будут достигнуто решение проблем предметной области
* Какова архитектура программного решения

Описание предметной области и проблем, для решения которых предназначен будущий программный продукт, приведены в разделе 2.

Раздел 3 содержит описание требований к программному решению, раздел 4 – описание архитектуры выбранного решения.

# Предметная область проекта

 Часто людям в супермаркете тяжело вспомнить, чего не хватает из продуктов, а проверить наличие не представляется возможным. В спешке приходится решать, что нужно купить. Иногда можно забыть, что продуктов не осталось вовсе.

Многие, в особенности девушки, следят за фигурой и считают *КБЖУ* для сверки со своей нормой. Это приходится делать для каждого блюда отдельно, взвешивая все продукты и рассчитывая *КБЖУ* для каждого, суммируя позднее (обычно в каком-то приложении). Было бы удобно получать расчет *КБЖУ*, просто выбрав рецепт.

Некоторые люди хранят рецепты в разных местах: в записной книжке, в отдельных файлах или на сайтах/в приложениях. Сложно выбрать рецепт, если в наличии имеется только ограниченный набор продуктов.

Людям, имеющим трудности с  хранением продуктов и отслеживанием их срока годности, подсчетом *КБЖУ* для каждого блюда в целях похудения или контроля питания было бы удобно использовать мобильное приложение для решения всех вышеперечисленных проблем, т.к. мобильное устройство всегда под рукой.

## Существующие проблемы

* Легко забыть, когда заканчивается срок годности у каждого купленного продукта;
* Проблемно составлять список покупок на бумаге или в заметках и отслеживать наличие продуктов;
* Сложно планировать питание;
* Неудобно вручную подсчитывать *КБЖУ*;
* Неудобное создание, хранение, выбор и использование рецепта;
* Приходится тратить много времени на выбор продуктов из-за отсутствия списка покупок.

## Предполагаемое решение

Решением описанных выше проблем может стать мобильное приложение, позволяющее пользователю:

* получать уведомление о конце срока годности продуктов;
* проверять наличие продуктов дома, находясь в супермаркете;
* выбрать постоянно необходимые продукты и получать напоминания об их отсутствии;
* спланировать меню, опираясь на продукты в наличии;
* автоматически подсчитывать *КБЖУ* на основе количества ингредиентов и данных продуктов, используемых в рецепте и существующих в действительности

# Требования к программному решению

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

## Роли

1. Пользователь – человек, использующий приложение для эффективного хранения продуктов, планирования меню, подсчета *КБЖУ*, подбора рецептов.

## Функциональные требования для роли Пользователь

### Посмотреть список рецептов

Открыть страницу рецептов (нижняя левая иконка).

### Посмотреть рецепт

Нажать на поле с выбранным рецептом на странице рецептов.

### Добавить рецепт

Нажать кнопку «+» в верхнем правом углу на странице рецептов и в открывшуюся форму внести данные.

### Посмотреть продукты

Открыть страницу продуктов (нижняя средняя иконка).

### Добавить продукт

Нажать кнопку «+» на странице продуктов, в открывшуюся форму ввести данные, либо отсканировать штрих код (если такой функционал будет реализован)

### Сделать продукт постоянно необходимым

Нажать на кнопку «!» рядом с продуктом на странице продуктов.

### Посмотреть продукты, которые скоро испортятся

Нажать на кнопку «\*» на главном экране, откроется список.

### Найти рецепт под портящиеся продукты

Нажать на кнопку «Подобрать рецепты» на странице продуктов, которые скоро испортятся.

### Посмотреть список покупок

Открыть страницу покупок (нижняя правая иконка).

### Добавить продукт в список покупок

Нажать на кнопку «Корзина покупок» рядом с продуктом на странице продуктов.

### Дополнить список покупок по рецепту

Нажать на кнопку «Корзина покупок» рядом с продуктом в списке продуктов для рецепта на странице рецепта.

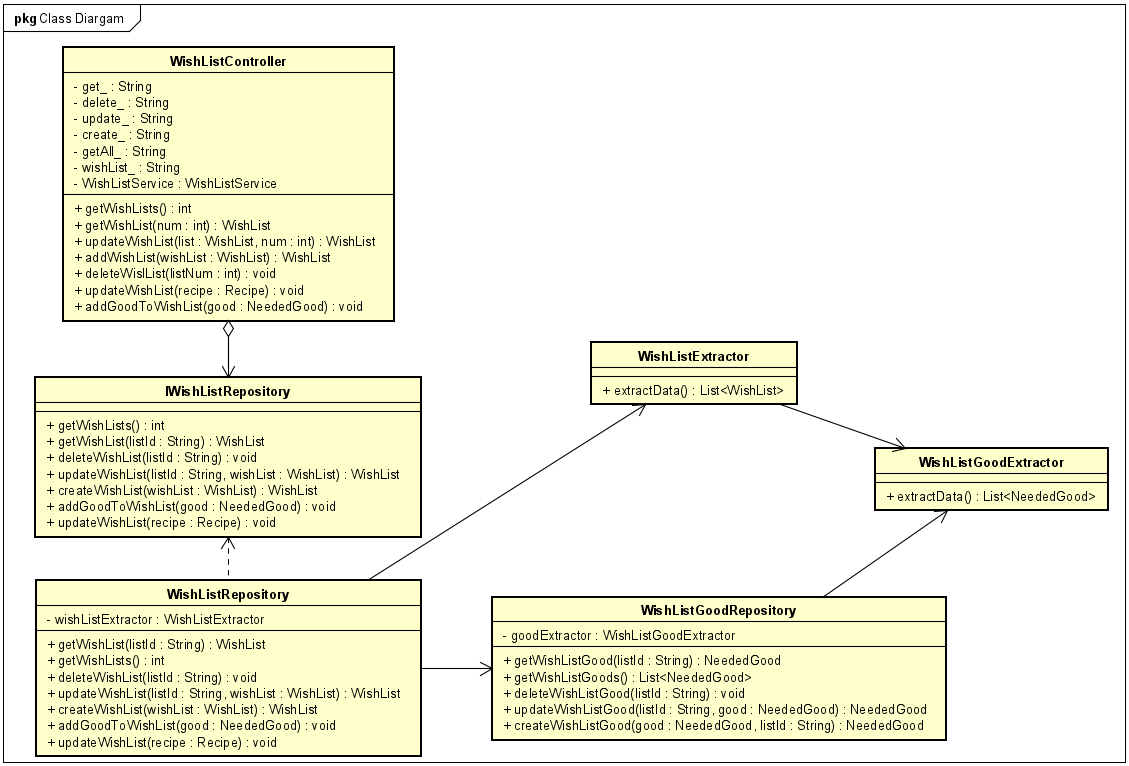
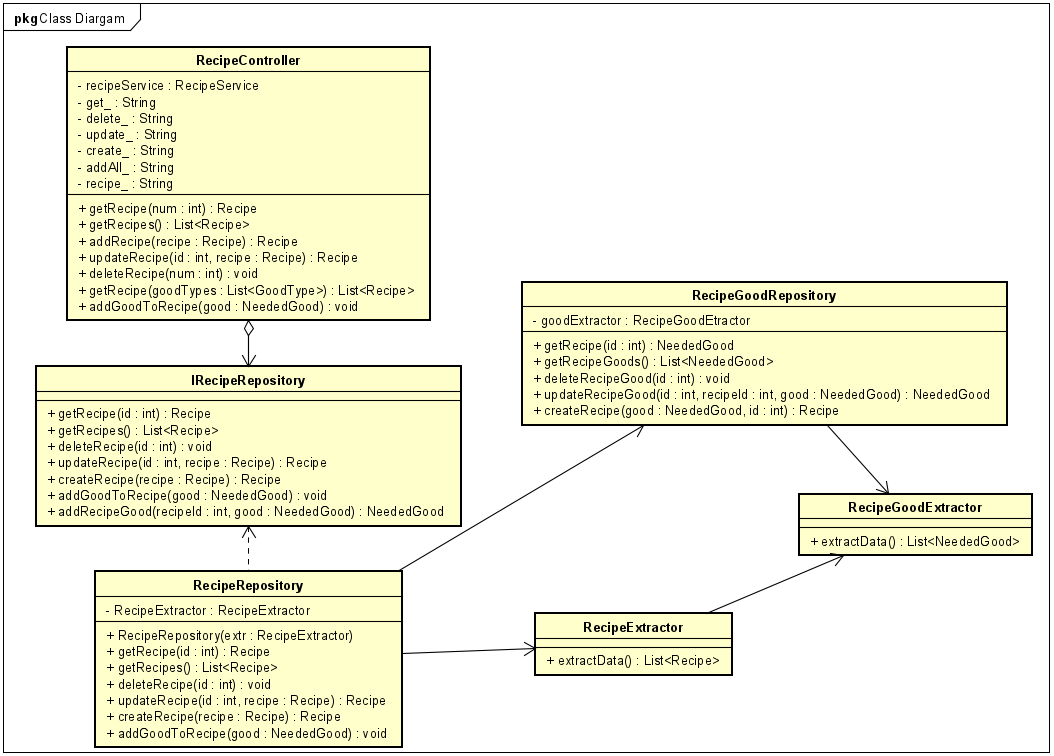
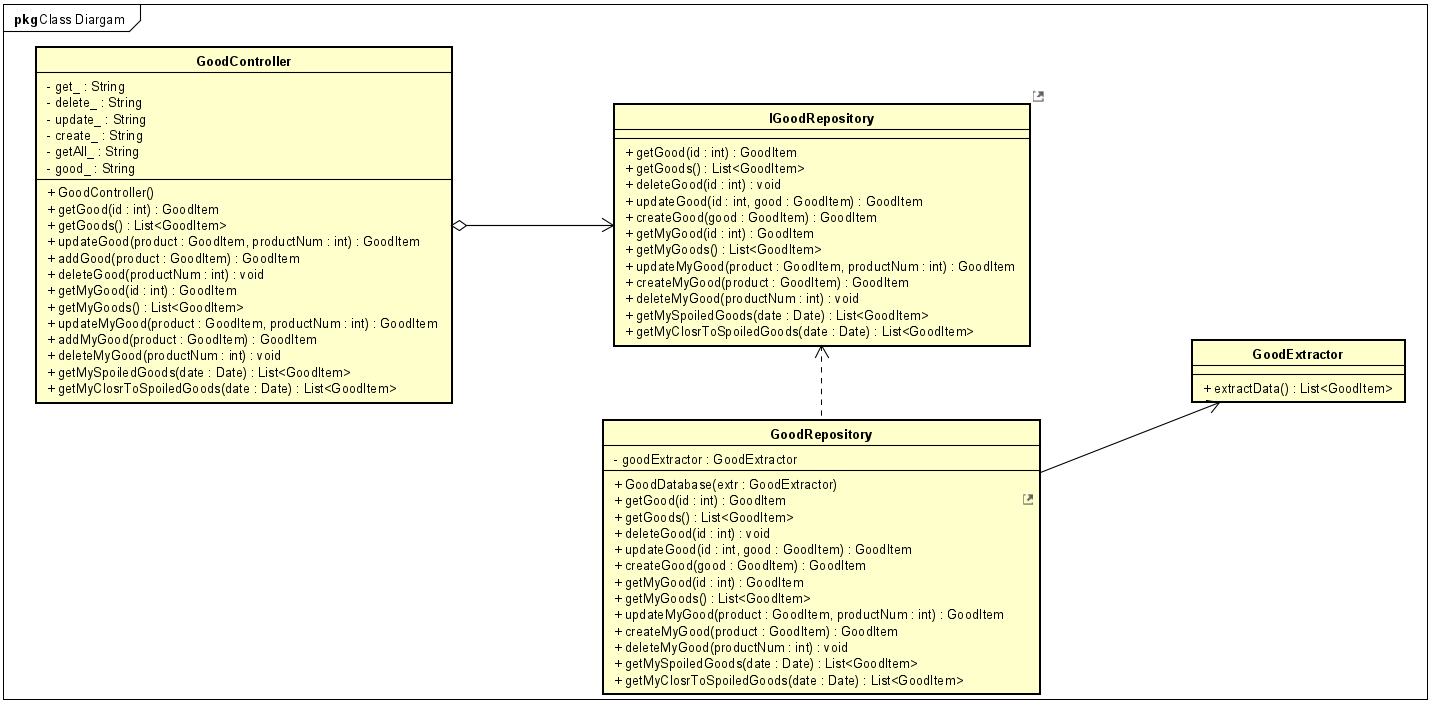
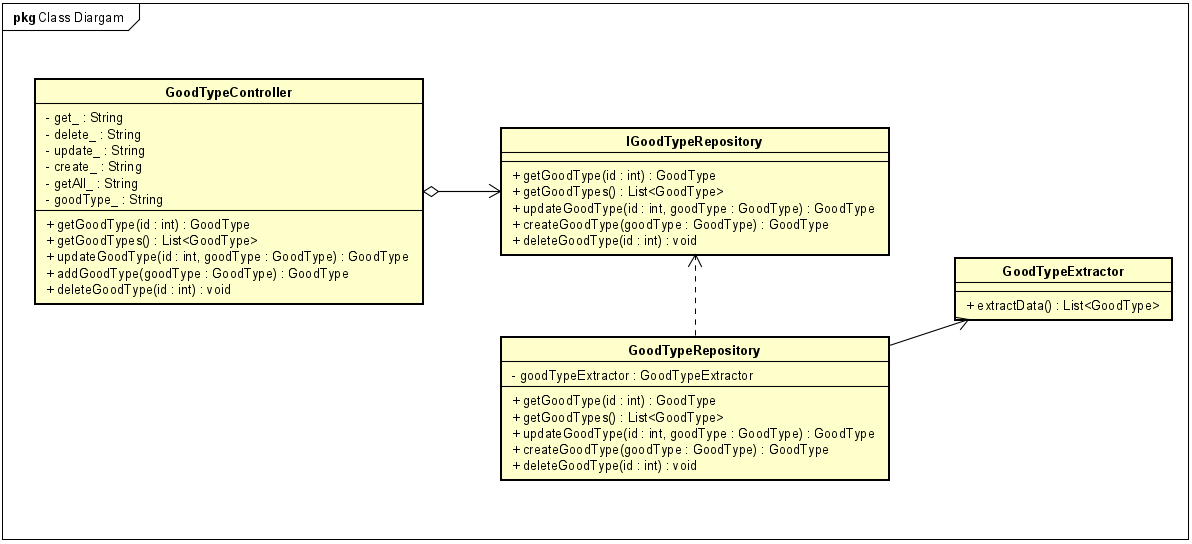
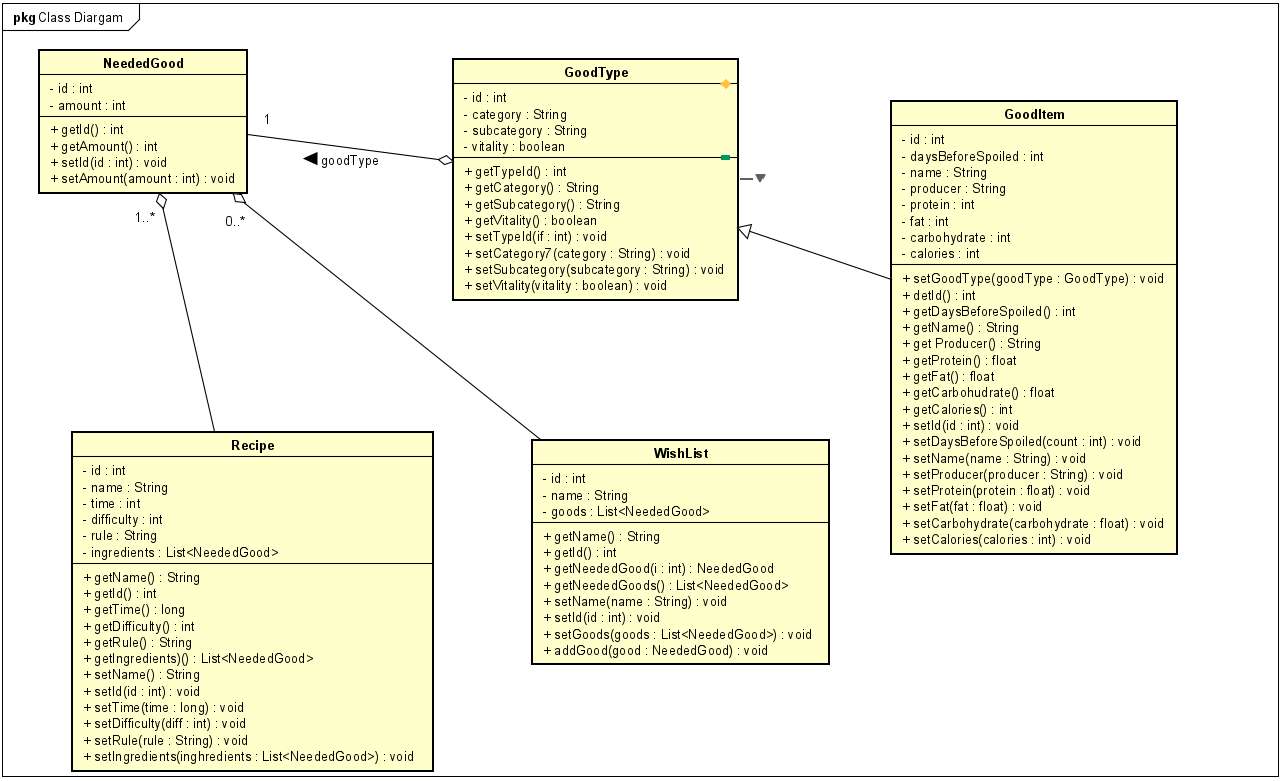
## Нефункциональные требования?

* Мобильное приложение должно быть реализовано под платформу Android;
* Мобильное приложение должно поддерживаться устройствами с установленной операционной системой Android, начиная с версии 9.0;

# Обзор архитектуры

Этот раздел описывает архитектуру системы.

### Компонентная модель системы



#### Model

GoodItem, GoodType, Recipe, WishList, NeededGood – сущности базы данных, представляющие собой соответственно продукт, тип продукта, рецепт, список покупок, недостающий продукт.

#### GoodTypeDiagram

GoodTypeController – контроллер, отвечающий за работу с типами продуктов. IGoodTypeRepository – интерфейс для взаимодействия с базой данных. GoodTypeRepository – реализация интерфейса IGoodTypeRepository. GoodTypeExtractor отвечает за запросы к базе данных.

#### GoodDiagram

GoodController – контроллер, отвечающий за работу с продуктами. IGoodRepository – интерфейс для взаимодействия с базой данных. GoodRepository – реализация интерфейса IGoodRepository. GoodExtractor отвечает за запросы к базе данных.

#### RecipeDiagram

RecipeController – контроллер, отвечающий за работу с продуктами. IRecipeRepository – интерфейс для взаимодействия с базой данных. RecipeRepository – реализация интерфейса IRecipeRepository. RecipeExtractor отвечает за запросы к базе данных.

#### WishListDiagram

WishListController – контроллер, отвечающий за работу с продуктами. IWishListRepository – интерфейс для взаимодействия с базой данных. WishListRepository – реализация интерфейса IWishListRepository. WishListExtractor отвечает за запросы к базе данных.

# Допущения и ограничения

Пример: При разработке проекта принято допущение, что число транзакций в единицу времени значительно (более чем в 10 раз) снижается в ночное время, что позволяет в период с 01:00 до 6:00 производить автоматическое обновление программного обеспечения системы, требующее полной перезагрузки и остановки сервиса на период до 5 минут.

# Известные проблемы

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

## Невысокая производительность приложения

|  |  |
| --- | --- |
| **Проблема** | Производительность приложения экспоненциально деградирует при общем числе пользователей выше 10000 и числе одновременных сессий выше 100. |
| **Ранг** | 10 (высокий) |
| **Влияние на проект** | Невозможность использования системы при числе пользователей более 10000. |
| **Пути решения** | Кластеризация веб-сервера и сервера базы данных, а также применение load balancer в точке маршрутизации запроса к веб-серверу. |

Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Лист регистрации проверок

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |