**Министерство науки и высшего образования российской федерации**

**ФГАОУ ВО «Пермский государственный национальный исследовательский университет»**

Физический факультет

Кафедра радиоэлектроники и защиты информации

Технологии Java

Лабораторная работа №2 на тему

**«****Технологии Java для создания серверных приложений»**

Выполнил:

студент 2 курса физического факультета

группа ПМИ-2-2021

направление «01.03.02 Прикладная математика и информатика»

профиль «Инженерия программного обеспечения»

Макарова Полина Фёдоровна

Пермь, 2022 г.

**Оглавление**

[1. Введение 3](#_Toc130163292)

[2. Постановка задачи 4](#_Toc130163293)

[3. Практическая реализация 5](#_Toc130163294)

[3.1. Используемый инструментарий 5](#_Toc130163295)

[3.2. Модель данных 6](#_Toc130163296)

[3.3. Программный интерфейс 7](#_Toc130163297)

[3.4. Хранение данных 18](#_Toc130163298)

[3.5. Логика работы программы 23](#_Toc130163299)

[3.6. Тестирование программы 24](#_Toc130163300)

[4. Заключение 28](#_Toc130163301)

[5. Список литературы 29](#_Toc130163302)

# Введение

**Цель работы.** Ознакомление с существующими технологиями Java для создания кроссплатформенных микросервисных приложений, обладающих программным интерфейсом API, работающим по протоколу HTTP REST.

**Краткая теория:**

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД). Существует в реализациях для множества UNIX-подобных платформ. Реляционные базы данных, такие как PostgreSQL, помогают пользователям упорядочить данные и составить представление о связях между ними. Это программное обеспечение совместимо со всеми основными операционными системами, включая Linux, Windows и Macintosh. Оно поддерживает текст, изображения, звуки и видео. Благодаря этому база данных пользуется популярностью у пользователей и компаний с различными потребностями. PostgreSQL считается самой востребованной технологией баз данных среди разработчиков, уступая только MySQL.[1]

Postman - это платформа API для создания и использования API. Postman упрощает каждый этап жизненного цикла API и упрощает совместную работу. [2]

JUnit5 — это мощное и гибкое обновление фреймворка JUnit, которое предоставляет множество улучшений и новых функций для написания тестов. JUnit 5 — наиболее широко используемая среда тестирования для приложений Java. [3]

По сути Spring Framework представляет собой просто контейнер внедрения зависимостей, с несколькими удобными слоями (доступ к базе данных, прокси, аспектно-ориентированное программирование, RPC, веб-инфраструктура MVC). Это все позволяет вам быстрее и удобнее создавать Java-приложения. [4]

# Постановка задачи

В рамках работы необходимо разработать микросервис, позволяющий работать с данными дневника еды (вариант № 18), с использованием технологий Java, обладающий программным интерфейсом (API) и функционалом сохранения и загрузки данных в СУБД.

Написать кроссплатформенное микросервисное приложение, позволяющее работать с данными дневника еды (вариант № 18).

Приложение должно хранить данные в СУБД Postgre SQL.

Приложение должно позволять загружать данные из электронной таблицы (в формате «ДневникЕды.xlsx») и сохранять их в БД.

Приложение должно позволять строить аналитический отчёт по алгоритму «выбрать такой день, у которого было больше всего еды с калориями меньше 400, сформировать карточку дня в виде документа docx» (вариант № 18). Подготовка данных для аналитического отчёта должна быть осуществлена средствами БД на сколько это возможно.

Приложение должно позволять через REST API получать(запрашивать), добавлять, удалять, редактировать данные, которые хранятся в БД и которые требуются для построения аналитического отчёта.

Программа должна использовать инструменты Java, Java SDK, коллекции, java.time.\*, Apache POI, Hibernate, Spring.

В программе должен быть реализован как минимум 1 автоматический тест.

# Практическая реализация

## Используемый инструментарий

Таблица 1

Инструментарий, использованный при решении поставленной задачи

|  |  |
| --- | --- |
| Язык программирования | Kotlin |
| Версия JDK | 8 |
| Среда разработки | IntelliJ IDEA Ultimate 2022.2.4 |
| Система автоматизированной сборки | Gradle |
| Инструменты JDK | java.util.Collections  java.time  java.util.UUID |
| Библиотеки | Apache POI, Hibernate, Spring, Junit 5 |

## Модель данных

Модель данных представляет собой набор классов с полями:

1. CDate - дата:
2. идентификатор даты, UUID id;
3. дата, LocalDate date.
4. CTimeInterval – временной промежуток:
5. идентификатор временного промежутка, UUID id;
6. название, String name;
7. временной промежуток, String vr.
8. CFood – еда:
9. идентификатор еды, UUID id;
10. название, String name;
11. калории, Double kal;
12. идентификатор даты, когда была съедена еда, UUID id;
13. идентификатор временного промежутка, когда была съедена еда, UUID id.

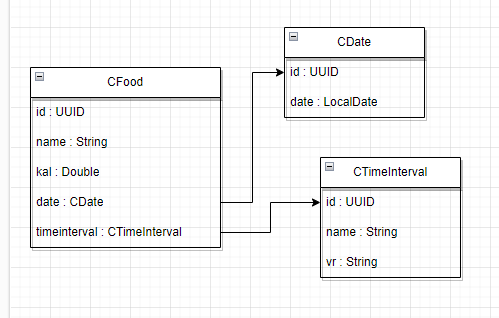


Рисунок - Модель данных

## Программный интерфейс

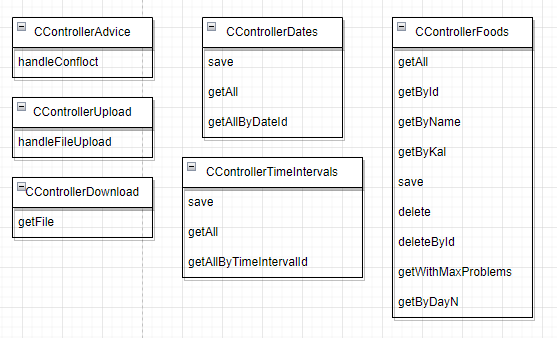


Рисунок - Диаграмма классов REST контроллеров

**CControllerUpload:**

handleFileUpload– загрузка БД:

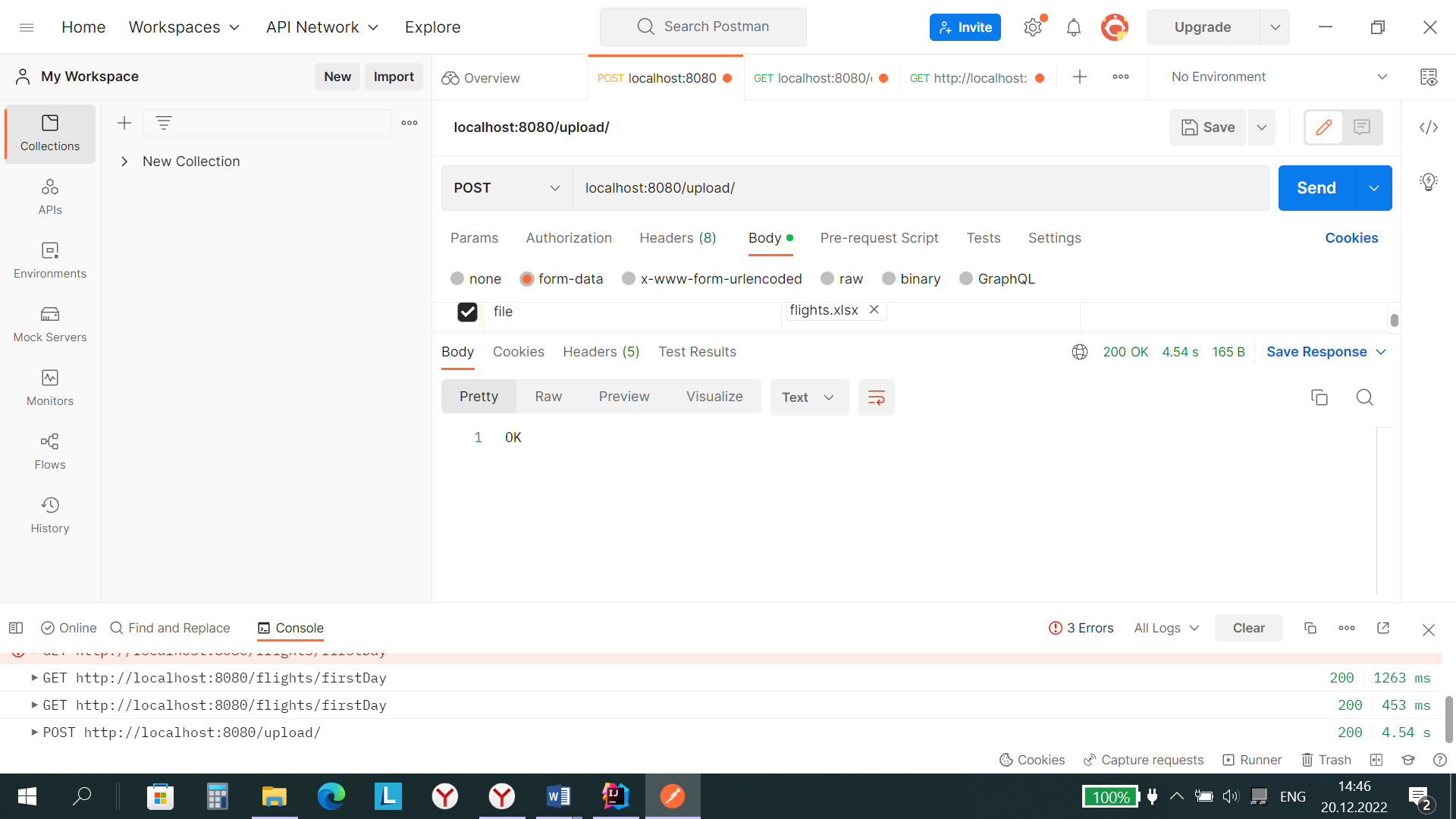


Рисунок - Запрос через Postman на загрузку БД данными из Excel файла

**CControllerFoods:**

getAll – получение всей еды из таблицы

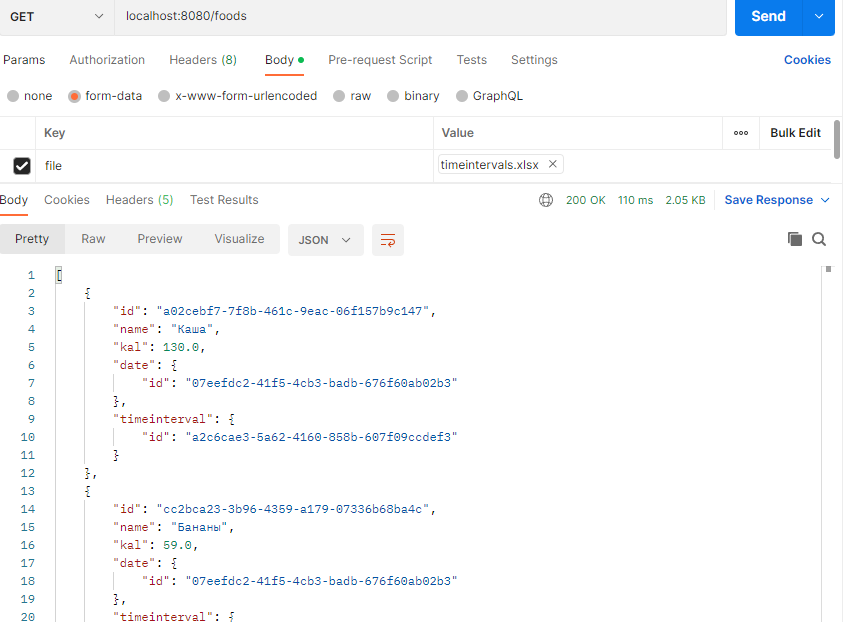


Рисунок - Запрос через Postman на получение всей еды

getById – получение еды по id

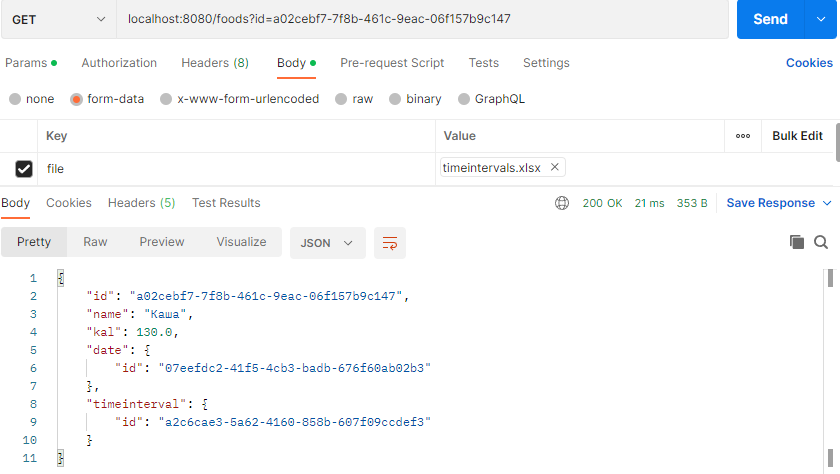


Рисунок - Запрос на получение еды с выбранным id

getByName – получение еды по названию

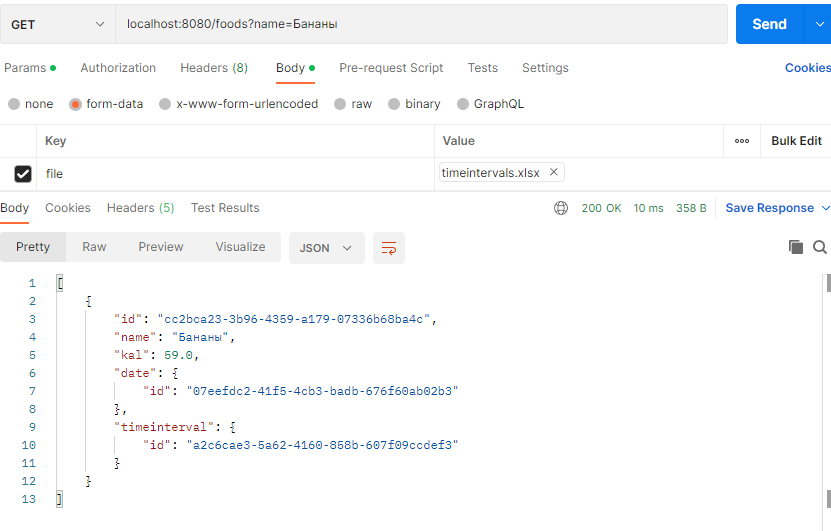


Рисунок - Запрос на получение еды с выбранным названием

getByDayN – список, включающий в себя еду "Бананы"

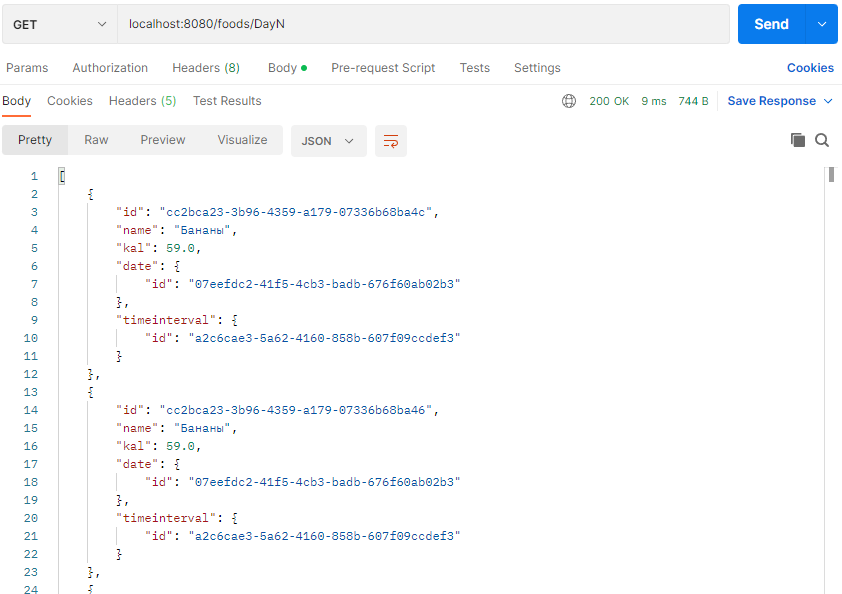


Рисунок - Запрос на список с названием "Бананы"

getByKal – получение еды по калориям

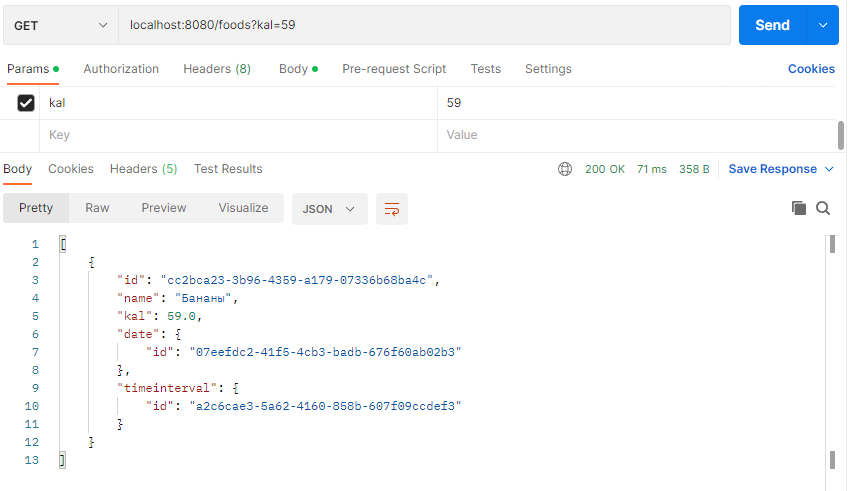


Рисунок - Запрос на получение еды по калориям

save – добавление еды

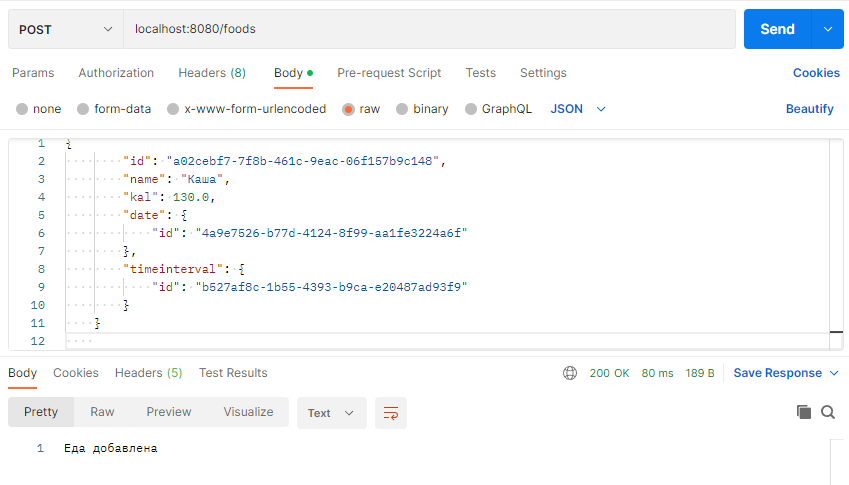


Рисунок - Добавление еды

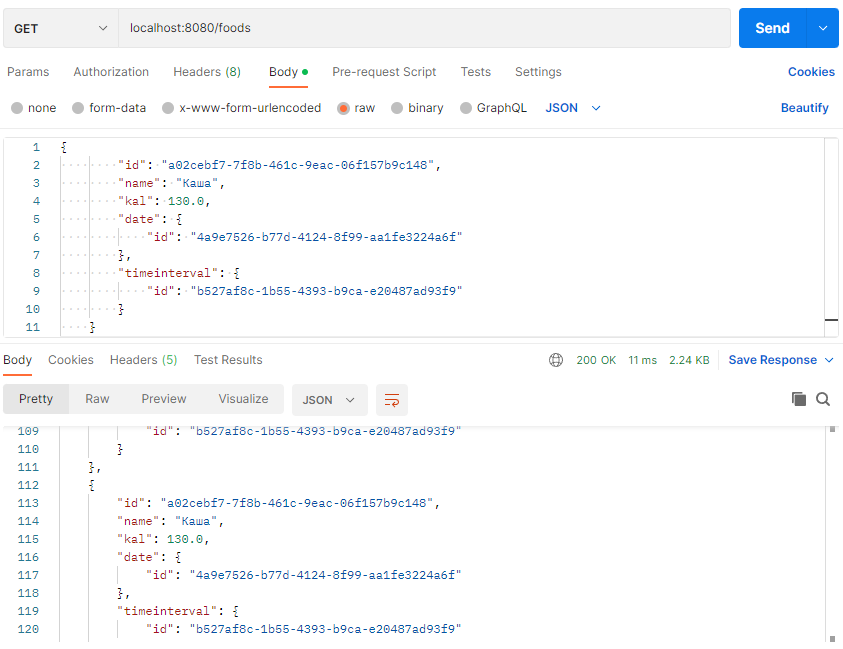


Рисунок - Проверка на добавление еды

delete – удаление еды

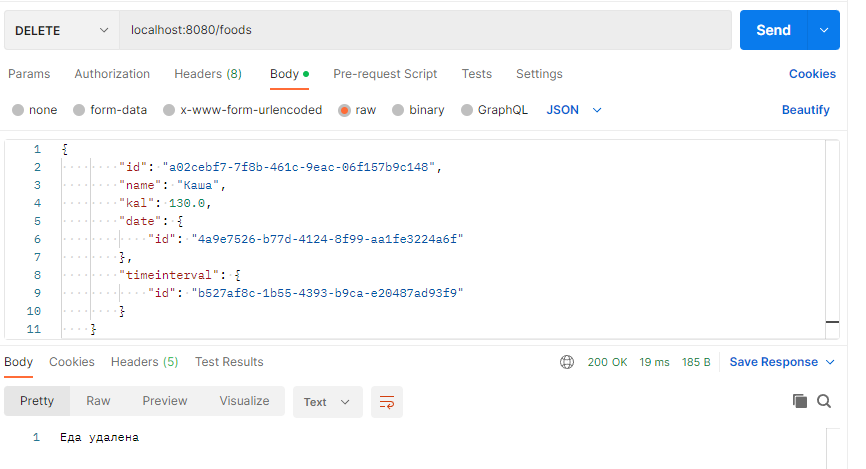


Рисунок - Удаление еды

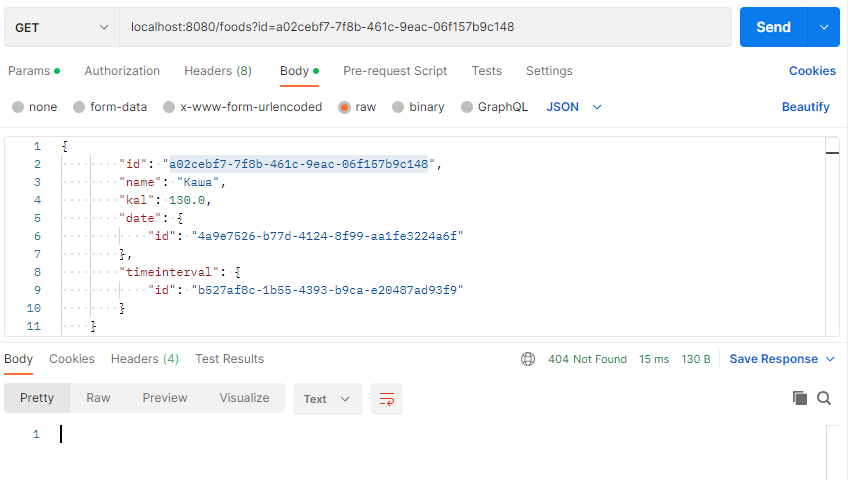


Рисунок - Проверка на удаление еды поиском по id

deleteById– удаление еды по id

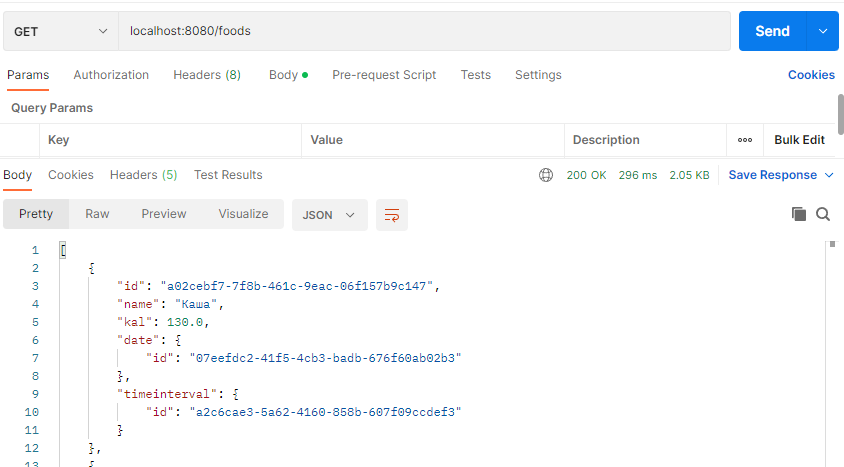


Рисунок - Список существующей в таблице еды

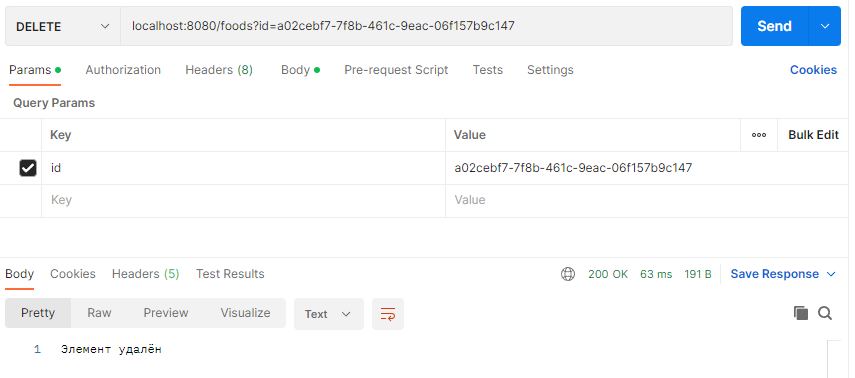


Рисунок - Удаляем еду из таблицы

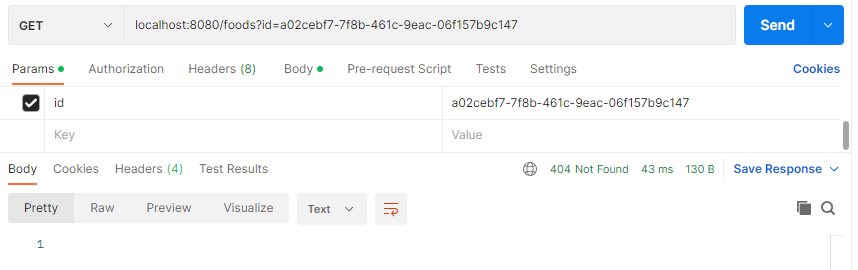


Рисунок - Проверяем удаление поиском по id

getWithMaxProblems – получение списка дат с максимальным количеством еды (калории меньше 400) и количество этой еды

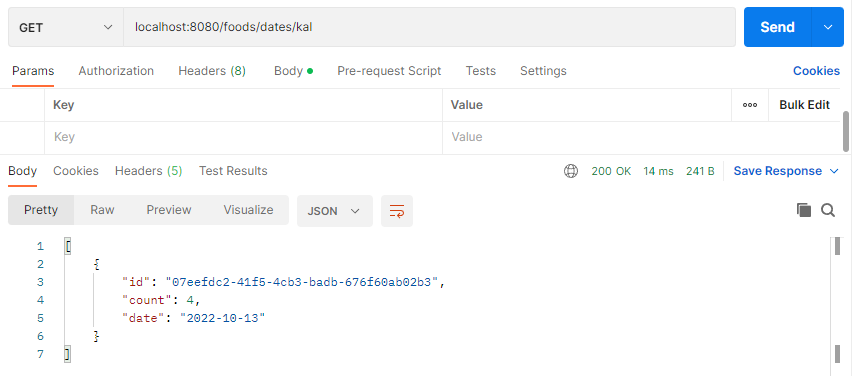


Рисунок - Получение списка дат с максимальным количеством еды меньше 400 калорий

**CControllerDates:**

save – добавление новой даты

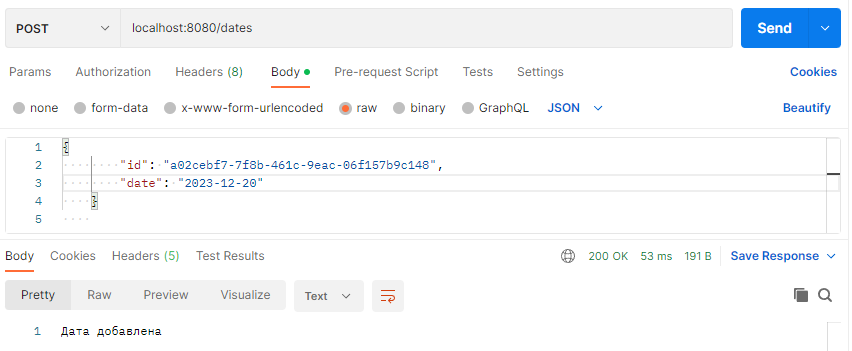


Рисунок - Добавление новой даты

getAllByDateId – поиск даты по id

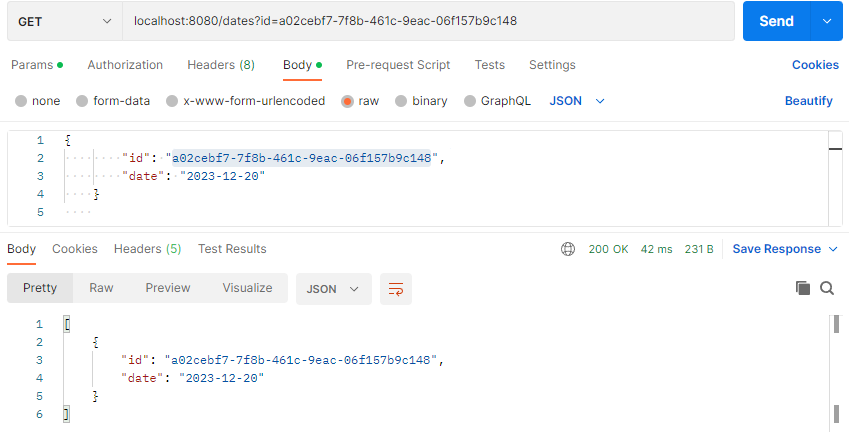


Рисунок - Поиск даты по id

getAll – вывести список всех дат

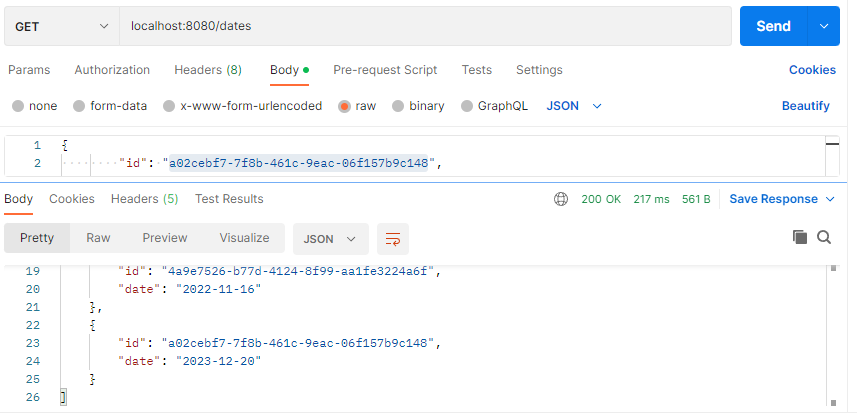


Рисунок 19 - Список всех дат

**CControllerTimeIntervals:**

save – добавление временного промежутка

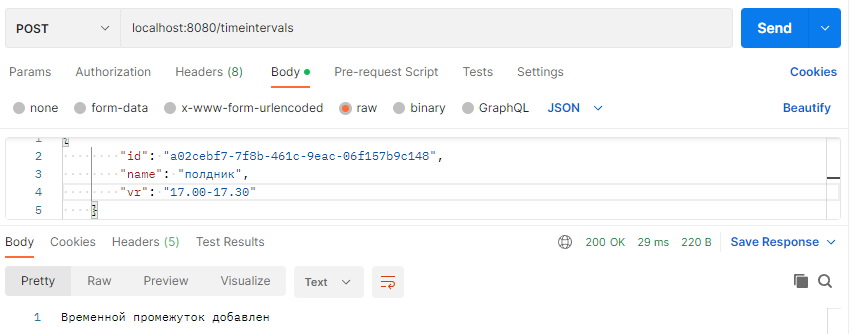


Рисунок - Добавление нового временного промежутка

getAllByTimeIntervalId – поиск временного промежутка по id

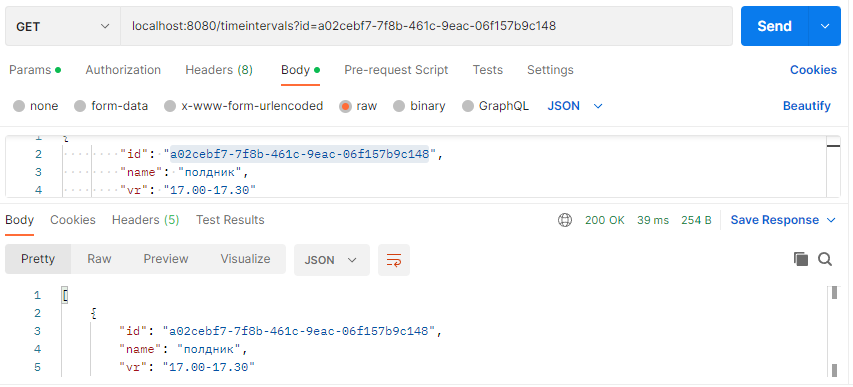


Рисунок - Поиск временного промежутка по id

getAllTimeIntervals – вывести список всех временных промежутков

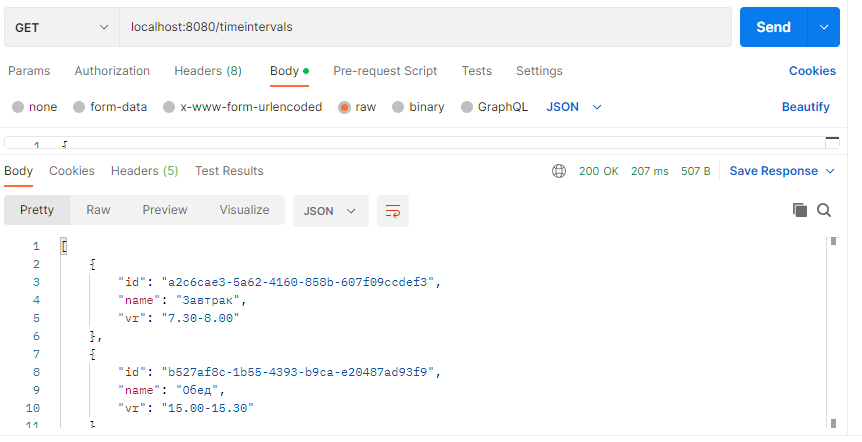


Рисунок - Список всех временных промежутков

**CControllerDownload:**

getFile – создание и возможность сохранения отчёта с запросом getByDayN и getWithMaxProblems

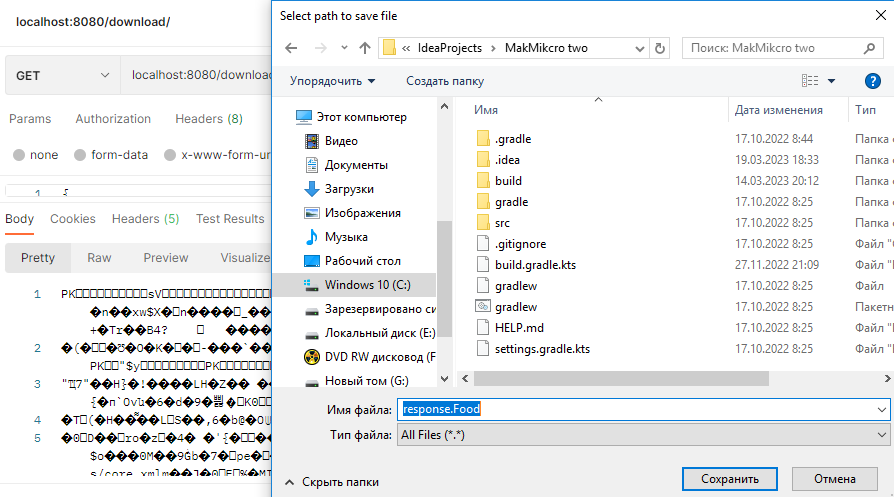


Рисунок - Запрос через Postman на создание и загрузку отчёта



Рисунок - Полученный файл

## Хранение данных

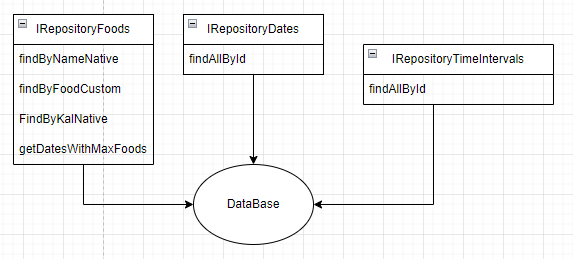


Рисунок - Диаграмма классов репозиториев

**Подключение к БД:**

В IntelliJ IDEA добавил новую базу данных, с указанием логина и пароля, хостом и портом.

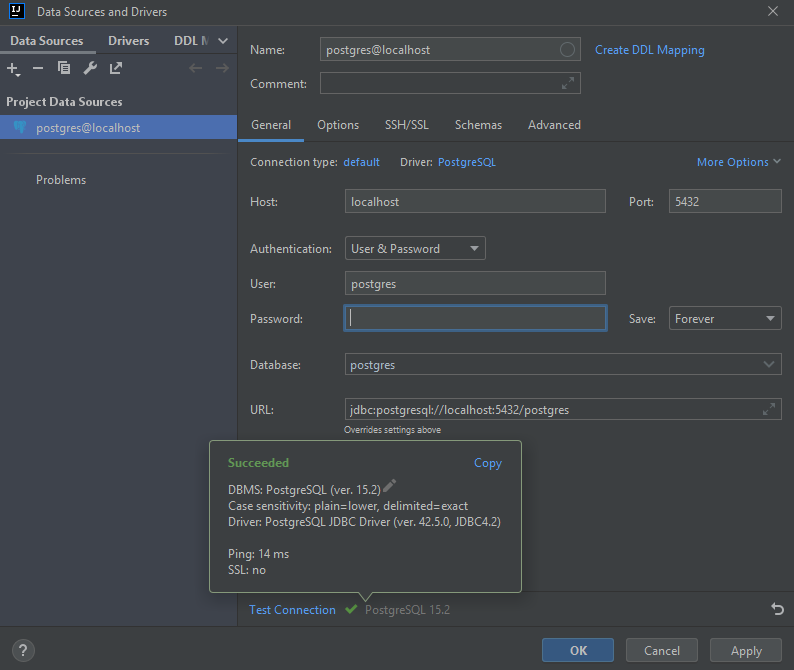


Рисунок - Подключение к БД в IntelliJ IDEA

Также в Resources создала файл Application.properties и добавил в него несколько свойств для правильного подключения к БД.

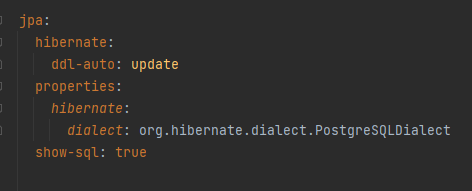


Рисунок - Настройка БД

**IRepositoryFoods:**

findByNameNative – получение списка еды по названию (через запрос на обычном SQL).

Алгоритм работы: делаем запрос к базе данных, выбираем необходимые колонки из таблицы еды, подходящие записи добавляем в список отфильтрованной еды.

|  |
| --- |
| *Листинг 1 – Запрос к БД для получения списка еды с названием name* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список еды с названием name (через запрос на обычном SQL). \*  \* @param name - параметр для фильтрации еды. \*  \* @return список отфильтрованной еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Query("SELECT \* FROM foods f WHERE f.name = ?1", nativeQuery= true) fun findByNameNative(  name : String ) : List<CFood> |

findByFoodCustom – получение списка еды по названию (через запрос JPQL).

Алгоритм работы: делаем запрос к базе данных, выбираем необходимые колонки из таблицы еды, подходящие записи добавляем в список отфильтрованной еды.

|  |
| --- |
| *Листинг 2 – Запрос к БД для получения списка еды с названием name* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список еды с названием name (через запрос JPQL). \*  \* @param name - параметр для фильтрации еды. \*  \* @return список отфильтрованной еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Query("SELECT f FROM CFood f WHERE f.name = ?1") fun findByFoodCustom(  name : String ) : List<CFood> |

findByKalNative – получение списка еды по калориям.

Алгоритм работы: делаем запрос к базе данных, выбираем необходимые колонки из таблицы еды, подходящие записи добавляем в список отфильтрованной еды.

|  |
| --- |
| *Листинг 3 – Запрос к БД для получения списка еды с калориями kal* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список еды с калориями kal (через запрос на обычном SQL). \*  \* @param kal - параметр для фильтрации еды. \*  \* @return список отфильтрованной еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Query("SELECT \* FROM foods f WHERE f.kal = ?1", nativeQuery= true) fun findByKalNative(  kal : Double ) : List<CFood> |

getDatesWithMaxFoods – получение списка дат с максимальным количеством еды меньше 400 калорий. Если у нескольких дат совпадает количество еды меньше 400 калорий, то выводиться список дат.

Алгоритм работы: делаем запрос к базе данных, выбираем необходимые колонки из таблицы еды, связываем с id даты имеющим больше всего еды меньше 400 калорий, и получаем нужную дату или список дат.

|  |
| --- |
| *Листинг 4 – Запрос к БД для получения списка дат с максимальным количеством еды меньше 400 калорий* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список дат с максимальным количеством еды меньше 400 калорий. \*  \* @return список отфильтрованных дат. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Query("""  select cast(t.id as varchar), t.date, t.count from (  select s.id, s.date, count(\*) as count  from dates s  left join foods m on s.id=m.date\_id  where m.kal<400  group by s.id, s.date  ) as t  where t.count in  (  select max(t1.cnt) from (  select s.id, count(\*) as cnt  from dates s  left join foods m on s.id=m.date\_id  where m.kal<400  group by s.id  ) as t1)  """,  nativeQuery = true ) fun getDatesWithMaxFoods() : List<IDateWithCounter> |

**IPilotWithCounter** – Специальный интерфейс, позволяющий получить результаты запроса из СУБД, которые не укладываются в сущность даты (здесь дополнительно есть поле – количество еды count).

|  |
| --- |
| *Листинг 5 – IPilotWithCounter* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Специальный интерфейс, позволяющий получить результаты запроса из СУБД, которые не укладываются \*  \* в сущность даты (здесь дополнительно есть поле - количество count). \*  \* @author Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* interface IDateWithCounter {  val id: UUID  val date: String  val count: Int  } |

**IRepositoryDates:**

findAllById – получение списка даты по id.

Алгоритм работы: делаем запрос к базе данных, выбираем необходимые колонки из таблицы дат, подходящие записи добавляем в список отфильтрованных дат (тут список, но на деле должен выводиться только одна дата).

|  |
| --- |
| *Листинг 6 –Запрос к БД для получения нужной даты по id* |
| @Repository interface IRepositoryDates : CrudRepository<CDate, UUID>{  fun findAllById(id: UUID): List<CDate>*//поиск по id* } |

**IRepositoryTimeIntervals:**

findAllById – получение списка даты по id.

Алгоритм работы: делаем запрос к базе данных, выбираем необходимые колонки из таблицы временных промежутков, подходящие записи добавляем в список отфильтрованных временных промежутков (тут список, но на деле должен выводиться только один временной промежуток).

|  |
| --- |
| *Листинг 7 – Запрос к БД для получения нужного временного промежутка по id* |
| @Repository interface IRepositoryTimeIntervals : CrudRepository<CTimeInterval, UUID>{  fun findAllById(id: UUID): List<CTimeInterval>*//поиск по id* } |

## Логика работы программы

**CServiceExcel:**

Методы класса осуществляют загрузку данных из страниц Excel и установку связей (за основу были взяты методы из 1 лабораторной работы). Затем эти данные загружаются в БД.

**CServiceFoods:**

Методы класса отвечают за работу с едой.

**CServiceDates:**

Методы класса отвечают за работу с датами.

**CServiceTimeIntervals**:

Методы класса отвечают за работу с временными промежутками.

**CServiceWord**:

Методы класса отвечают за формирования аналитического отчёта.

Создается файл с 2 таблицами:

* + Список дат с максимальным количеством еды (с калориями меньше 400) и количество этой еды;
  + Даты, в которых была записана еда “Бананы”.

## Тестирование программы

Загружаемый файл:

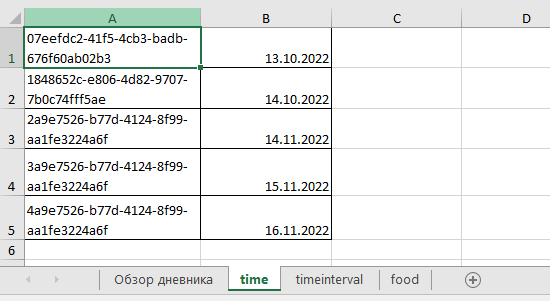


Рисунок - Таблица даты

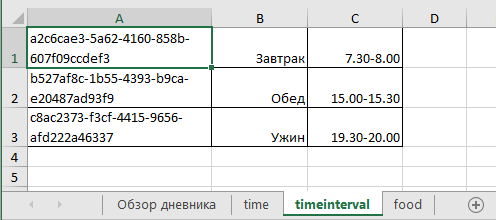


Рисунок - Таблица временного промежутка

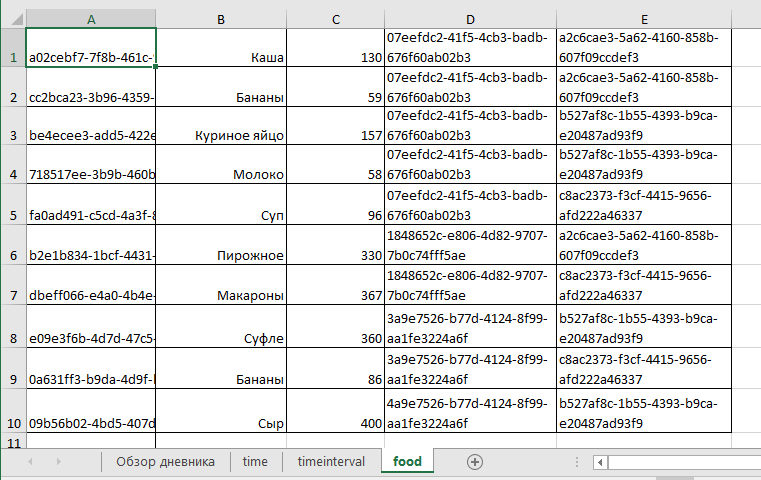


Рисунок - Таблица еды

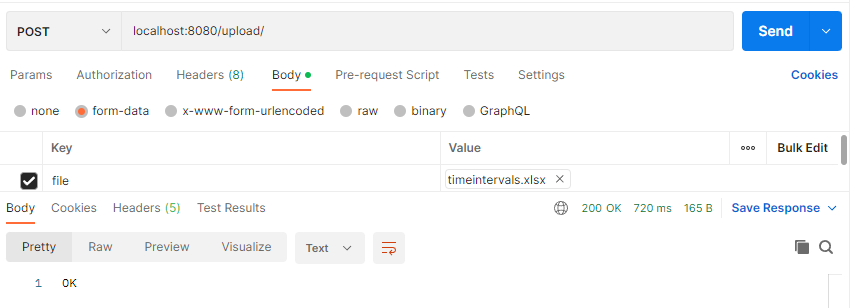


Рисунок - Запрос на загрузку файла

Разработаны автоматические тесты для проверки функционала контроллера и сервиса еды:

СApplicationTests– класс с автоматическими тестами для проверки полноценного функционирования методов контроллера еды. Имитируем необходимый репозиторий еды с помощью библиотеки Spring Mock.

Создаём продуктивный контроллер еды и передаём в конструктор имитацию репозитория еды.

|  |
| --- |
| *Листинг 8 – СApplicationTests* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Класс с автоматическими тестами для проверки функционала контроллера еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @SpringBootTest class СApplicationTests {   *//Продуктивный сервис использовать не будем,  //его функционал тестируется в отдельном классе,  //здесь его имитируем с помощью метода mockk()* private val serviceFoods : CServiceFoods  = *mockk*()   *//А вот контроллер еды продуктивный,  //но будет вызывать методы имитируемого сервиса.* private val controllerFoods = CControllerFoods(serviceFoods)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Проверка обработки запроса на удаление еды по идентификатору. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Test  fun testDeleteFoodsById()  {  val id = UUID.randomUUID()  *every* **{**serviceFoods.deleteById(id)**}** returns "Объект удалён"   val result = controllerFoods.deleteById(id)  Assertions.assertEquals("Объект удалён", result)  } } |

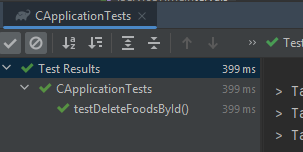


Рисунок - Проверка функционала контроллера еды

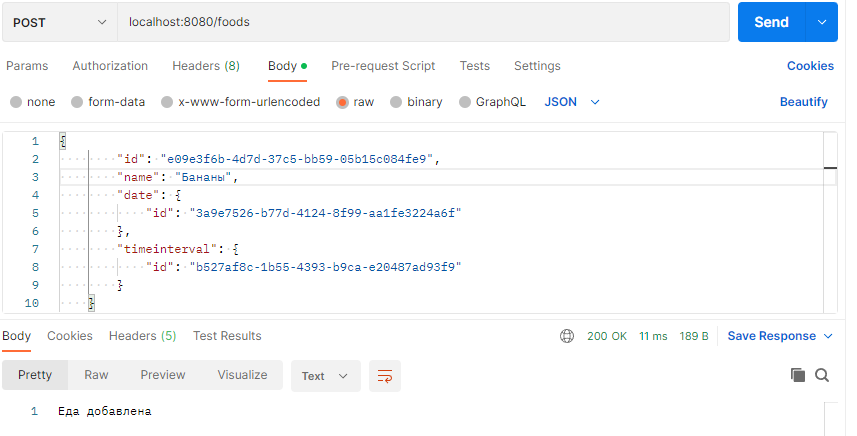


Рисунок - Создание еды без калорий

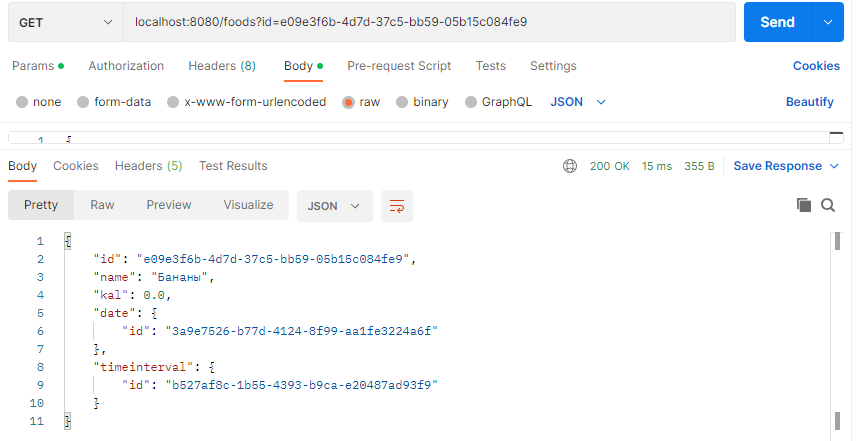


Рисунок - Проверка созданной еды

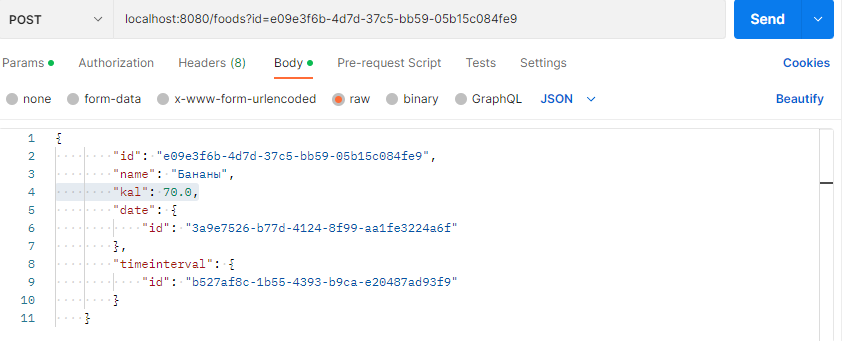


Рисунок - Изменение данных еды

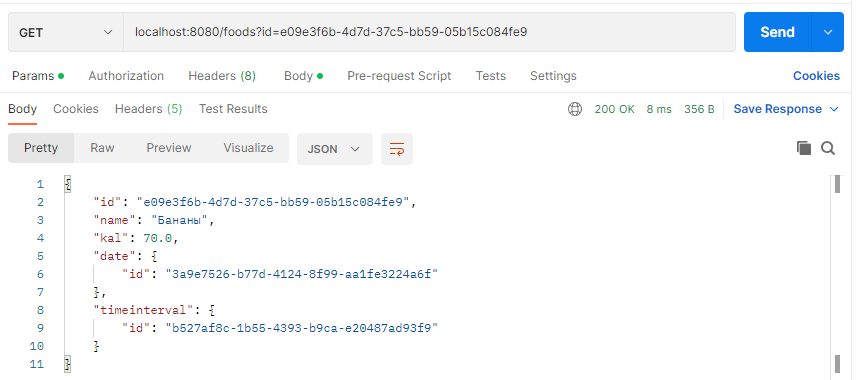


Рисунок - Проверка изменённых данных

# Заключение

Разработано 3 основных класса-сущности и один дополнительный специальный интерфейс, позволяющий получить результаты запроса из СУБД, которые не укладываются в основную сущность. Для программного интерфейса разработано 6 классов-контроллеров, для хранения данных 3 класса-репозитория, для обработки логики 10 классов-сервисов.

Приложение написано, протестировано, решает поставленную задачу полностью. Получены знания о существующих технологиях Java для создания кроссплатформенных микросервисных приложений, обладающих программным интерфейсом API, работающим по протоколу HTTP REST, а также новые теоретические и практические знания о Postgre SQL, Postman, Junit 5, Spring Framework.

# Список литературы

# Azure: Что такое PostgreSQL? : [Электронный ресурс]. – URL: <https://azure.microsoft.com/ru-ru/resources/cloud-computing-dictionary/what-is-postgresql/> (дата обращения: 06.02.2023)

1. Postman API Platform [Электронный ресурс]. – URL: <https://www.postman.com/> (дата обращения: 06.02.2023)
2. Habr: Туториал по JUnit 5 - Введение [Электронный ресурс]. – URL: <https://habr.com/ru/post/590607/> (дата обращения: 06.02.2023)
3. Habr: Что такое Spring Framework? От внедрения зависимостей до Web MVC [Электронный ресурс]. – URL: <https://habr.com/ru/post/490586/> (дата обращения: 06.02.2023)

**Приложение (Без тестов)**

|  |
| --- |
| *Листинг 9 – CControllerAdvice* |
| package ru.psu.java.laba2.makmikcro.controllers  import org.springframework.http.HttpHeaders import org.springframework.http.HttpStatus import org.springframework.http.ResponseEntity import org.springframework.web.bind.annotation.ControllerAdvice import org.springframework.web.bind.annotation.ExceptionHandler import org.springframework.web.context.request.WebRequest import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler  @ControllerAdvice*//обработка исключений* class CControllerAdvice : ResponseEntityExceptionHandler() {  @ExceptionHandler(  value = [  IllegalArgumentException::class,  IllegalStateException::class  ]  )  protected fun handleConfloct(  ex: RuntimeException?, request: WebRequest?  ): ResponseEntity<Any>{  val bodyOfResponse = ""  return handleExceptionInternal(  ex!!, bodyOfResponse,  HttpHeaders(), HttpStatus.*CONFLICT*, request!!  )  } } |

|  |
| --- |
| *Листинг 10 – CControllerDates* |
| package ru.psu.java.laba2.makmikcro.controllers  import org.springframework.web.bind.annotation.\* import ru.psu.java.laba2.makmikcro.model.CDate import ru.psu.java.laba2.makmikcro.model.CFood import ru.psu.java.laba2.makmikcro.services.IServiceDates import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Контроллер REST запросов к справочнику дат. \*  \* @author Макарова П.Ф. 12.03.2023. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @RestController @RequestMapping("/dates") *//Относительный путь, по которому должны поступать запросы.* class CControllerDates */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* Конструктор. \* \* @param serviceDates - сервис с логикой для работы со списком дат (устанавливается Spring-ом) \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* (  var serviceDates : IServiceDates ) {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание/изменение дат. \*  \* @param date - данные дат. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @PostMapping  fun save(@RequestBody date: CDate): String? *//создание* {  serviceDates.save(date)  return "Дата добавлена"  }   @GetMapping(params = ["id"])*//поиск по id* fun getAllByDateId(@RequestParam id: UUID): List<CDate>  {  return serviceDates.getAllByDateId(id)  }  @GetMapping  fun getAll() : List<CDate>  {  return serviceDates.getAllDates()  } } |

|  |
| --- |
| *Листинг 11 – CControllerDownload* |
| package ru.psu.java.laba2.makmikcro.controllers  import org.springframework.web.bind.annotation.\* import ru.psu.java.laba2.makmikcro.services.IServiceWord import javax.servlet.http.HttpServletResponse  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Контроллер REST запросов для формирования аналитического отчёта. \*  \* @author Макарова П.Ф. 12.03.2023. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @RestController @RequestMapping("/download") class CControllerDownload(  val serviceWord: IServiceWord ) {   @GetMapping("/")  fun getFile(  response: HttpServletResponse  ) {  serviceWord.downloadWord(response)  } } |

|  |
| --- |
| *Листинг 12 – CControllerFoods* |
| package ru.psu.java.laba2.makmikcro.controllers  import org.springframework.http.ResponseEntity import org.springframework.web.bind.annotation.\* import ru.psu.java.laba2.makmikcro.model.CFood import ru.psu.java.laba2.makmikcro.model.IDateWithCounter import ru.psu.java.laba2.makmikcro.services.IServiceFoods import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Контроллер REST запросов к справочнику еды. \*  \* @author Макарова П.Ф. 12.03.2023. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @RestController @RequestMapping("/foods") *//Относительный путь, по которому должны поступать запросы.* class CControllerFoods */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Конструктор. \*  \* @param serviceFoods - сервис с логикой для работы со списком еды (устанавливается Spring-ом)\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* (  val serviceFoods : IServiceFoods ) {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список всей еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @GetMapping  fun getAll() : Iterable<CFood>  {  return serviceFoods.getAll()  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по идентификатору. \*  \* @param id - идентификатор еды для поиска в формате UUID. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @GetMapping(  *//Этой строкой говорим, что метод вызывать только тогда,  //когда в параметрах запроса есть параметр id.* params = ["id"]  )  fun getById(  *//Значение перменной id заполняется содержимым из параметра запроса id* @RequestParam id: UUID) : ResponseEntity<CFood>  {  return serviceFoods.getById(id)  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по названию. \*  \* @param name - название еды для поиска в формате String. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @GetMapping(params = ["name"])  fun getByName(@RequestParam name: String) : Iterable<CFood>  {  return serviceFoods.getByName(name)  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по калориям. \*  \* @param kal - калории еды для поиска в формате Double. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @GetMapping(params = ["kal"])  fun getByKal(@RequestParam kal: Double) : Iterable<CFood>  {  return serviceFoods.getByKal(kal)  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание/изменение еды. \*  \* @param food - данные еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @PostMapping  fun save(  *//Данные даты должны быть в теле запроса.* @RequestBody food: CFood) : String?  {  serviceFoods.save(food)  return "Еда добавлена"  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление еды. \*  \* @param food - данные еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @DeleteMapping  fun delete(  *//Данные даты должны быть в теле запроса.* @RequestBody food: CFood) : String?  {  serviceFoods.delete(food)  return "Еда удалена"  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление еды по идентификатору. \*  \* @param id - идентификатор еды для удаления. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @DeleteMapping(params= ["id"])  fun deleteById(@RequestParam id: UUID): String?  {  return serviceFoods.deleteById(id)  }  *//Список дней с максимальным количеством еды (меньше 400 каллорий) и количество этих еды.* @GetMapping(path = ["/dates/kal"])  fun getWithMaxProblems() : List<IDateWithCounter>  {  return serviceFoods.getWithMaxFoods()  }  *// получить списки дат по названию("Бананы")* @GetMapping(path = ["/DayN"])  fun getByDayN() : Iterable<CFood>  {  return serviceFoods.getByDayN()  } } |

|  |
| --- |
| *Листинг 13 – CControllerTimeIntervals* |
| package ru.psu.java.laba2.makmikcro.controllers  import org.springframework.web.bind.annotation.\* import ru.psu.java.laba2.makmikcro.model.CDate import ru.psu.java.laba2.makmikcro.model.CTimeInterval import ru.psu.java.laba2.makmikcro.services.IServiceTimeIntervals import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Контроллер REST запросов к справочнику временного просежутка. \*  \* @author Макарова П.Ф. 12.03.2023. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @RestController @RequestMapping("/timeintervals") *//Относительный путь, по которому должны поступать запросы.* class CControllerTimeIntervals */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* Конструктор. \* \* @param serviceTimeIntervals - сервис с логикой для работы со списком временного промежутка (устанавливается Spring-ом) \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* (  var serviceTimeIntervals : IServiceTimeIntervals ) {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание/изменение временного промежутка. \*  \* @param timeinterval - данные временного промежутка. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @PostMapping  fun save(@RequestBody timeinterval: CTimeInterval) : String? *//создание* {  serviceTimeIntervals.save(timeinterval)  return "Временной промежуток добавлен"  }   @GetMapping(params = ["id"])*//поиск по id* fun getAllByTimeIntervalId(@RequestParam id: UUID): List<CTimeInterval>  {  return serviceTimeIntervals.getAllByTimeIntervalId(id)  }  @GetMapping  fun getAll() : List<CTimeInterval>  {  return serviceTimeIntervals.getAllTimeIntervals()  }  } |

|  |
| --- |
| *Листинг 14 – CControllerUpload* |
| package ru.psu.java.laba2.makmikcro.controllers  import org.springframework.web.bind.annotation.PostMapping import org.springframework.web.bind.annotation.RequestMapping import org.springframework.web.bind.annotation.RequestParam import org.springframework.web.bind.annotation.RestController import org.springframework.web.multipart.MultipartFile import ru.psu.java.laba2.makmikcro.services.IServiceExcel  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Контроллер REST запросов для для импорта данных из xlsx. \*  \* @author Макарова П.Ф. 12.03.2023. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @RestController @RequestMapping("/upload") class CControllerUpload (  val serviceExcel : IServiceExcel ) {  @PostMapping("/")  fun handleFileUpload(  @RequestParam("file") file : MultipartFile  ) : String?  {  serviceExcel.uploadExcel(file)  return "OK"  } } |

|  |
| --- |
| *Листинг 15 – CDate* |
| package ru.psu.java.laba2.makmikcro.model  import com.fasterxml.jackson.annotation.JsonIncludeProperties import java.time.LocalDate import java.util.\* import javax.persistence.Column import javax.persistence.Entity import javax.persistence.Id import javax.persistence.Table import javax.persistence.OneToMany  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Класс дата \*  \* @autor Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Entity @Table(name = "dates") class CDate {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Идендификатор  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Id  var id: UUID? = null   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Дата записи  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Column  var date: LocalDate? = null } |

|  |
| --- |
| *Листинг 16 – CFood* |
| package ru.psu.java.laba2.makmikcro.model  import com.fasterxml.jackson.annotation.JsonIncludeProperties import java.util.\* import javax.persistence.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Класс блюдо(еда) \*  \* @autor Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Entity @Table(name = "foods") class CFood() {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Идендификатор  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Id  var id: UUID? = null   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Название  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Column  var name: String? = null   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Каллории  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Column()  var kal : Double = 0.0   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Дата  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @ManyToOne  @JoinColumn(name="date\_id", nullable = false)  @JsonIncludeProperties(value = ["id"])  var date: CDate? = null   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Временной промежуток  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @ManyToOne  @JoinColumn(name="timeinterval\_id", nullable = false)  @JsonIncludeProperties(value = ["id"])  var timeinterval: CTimeInterval? = null   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Вывод  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun toString(): String {  return "Дата " + (if (date == null) "не указана" else date) + " Блюдо -" + name  } } |

|  |
| --- |
| *Листинг 17 – CTimeInterval* |
| package ru.psu.java.laba2.makmikcro.model  import com.fasterxml.jackson.annotation.JsonIncludeProperties import java.util.\* import javax.persistence.Column import javax.persistence.Entity import javax.persistence.OneToMany import javax.persistence.Table import javax.persistence.Id  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Класс временной интервал \*  \* @autor Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Entity @Table(name = "timeintervals") class CTimeInterval {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Идендификатор  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Id  var id: UUID? = null   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Название  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Column  var name: String? = null   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Временной промежуток  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Column  var vr: String? = null } |

|  |
| --- |
| *Листинг 18 – IDateWithCounter* |
| package ru.psu.java.laba2.makmikcro.model  import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Специальный интерфейс, позволяющий получить результаты запроса из СУБД, которые не укладываются \*  \* в сущность даты (здесь дополнительно есть поле - количество count). \*  \* @author Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* interface IDateWithCounter {  val id: UUID  val date: String  val count: Int } |

|  |
| --- |
| *Листинг 19 – IRepositoryDates* |
| package ru.psu.java.laba2.makmikcro.repositories  import org.springframework.data.repository.CrudRepository import org.springframework.stereotype.Repository import ru.psu.java.laba2.makmikcro.model.CDate import java.util.\*  @Repository interface IRepositoryDates : CrudRepository<CDate, UUID>{  fun findAllById(id: UUID): List<CDate>*//поиск по id* } |

|  |
| --- |
| *Листинг 20 – IRepositoryFoods* |
| package ru.psu.java.laba2.makmikcro.repositories  import org.springframework.data.jpa.repository.Query import org.springframework.data.repository.CrudRepository import org.springframework.stereotype.Repository import ru.psu.java.laba2.makmikcro.model.CFood import ru.psu.java.laba2.makmikcro.model.IDateWithCounter import java.time.LocalDate import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Интерфейс с заголовками методов для доступа к данным в СУБД. \*  \* @author Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Repository interface IRepositoryFoods : CrudRepository<CFood, UUID> {  *//Типовые методы наследуются из интерфйса CrudRepository  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список еды с названием name (через запрос на обычном SQL). \*  \* @param name - парметр для фильтрации еды. \*  \* @return список отфильтрованной еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Query("SELECT \* FROM foods f WHERE f.name = ?1", nativeQuery= true)  fun findByNameNative(  name : String  ) : List<CFood>   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список еды с названием name (через запрос JPQL). \*  \* @param name - парметр для фильтрации еды. \*  \* @return список отфильтрованной еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Query("SELECT f FROM CFood f WHERE f.name = ?1")  fun findByFoodCustom(  name : String  ) : List<CFood>   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список еды с калориями kal (через запрос на обычном SQL). \*  \* @param kal - парметр для фильтрации еды. \*  \* @return список отфильтрованной еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Query("SELECT \* FROM foods f WHERE f.kal = ?1", nativeQuery= true)  fun findByKalNative(  kal : Double  ) : List<CFood>   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список дат с максимальным количеством еды меньше 400 каллорий. \*  \* @return список отфильтрованных дат. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Query("""  select cast(t.id as varchar), t.date, t.count from (  select s.id, s.date, count(\*) as count  from dates s  left join foods m on s.id=m.date\_id  where m.kal<400  group by s.id, s.date  ) as t  where t.count in  (  select max(t1.cnt) from (  select s.id, count(\*) as cnt  from dates s  left join foods m on s.id=m.date\_id  where m.kal<400  group by s.id  ) as t1)  """,  nativeQuery = true  )  fun getDatesWithMaxFoods() : List<IDateWithCounter> } |

|  |
| --- |
| *Листинг 21 – IRepositoryTimeIntervals* |
| package ru.psu.java.laba2.makmikcro.repositories import org.springframework.data.repository.CrudRepository import org.springframework.stereotype.Repository import ru.psu.java.laba2.makmikcro.model.CTimeInterval import java.util.\*  @Repository interface IRepositoryTimeIntervals : CrudRepository<CTimeInterval, UUID>{  fun findAllById(id: UUID): List<CTimeInterval>*//поиск по id* } |

|  |
| --- |
| *Листинг 22 – CServiceDates* |
| package ru.psu.java.laba2.makmikcro.services  import org.springframework.beans.factory.annotation.Autowired import org.springframework.stereotype.Service import ru.psu.java.laba2.makmikcro.model.CDate import ru.psu.java.laba2.makmikcro.model.CFood import ru.psu.java.laba2.makmikcro.repositories.IRepositoryDates import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Сервис с бизнес-логикой для работы со списком дат. \*  \* @author Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Service class CServiceDates : IServiceDates {  @Autowired  lateinit var repositoryDates: IRepositoryDates   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание даты. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun save(date: CDate) {  try {  repositoryDates.save(date)  }  catch (e : Exception)  {  throw Exception("Не указаны обязательные данные для даты!", e)  }  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск дат по идентификатору. \*  \* @param id - идентификатор дат для поиска в формате UUID. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun getAllByDateId(id: UUID): List<CDate>  {  return repositoryDates.findAllById(id)  }   override fun getAllDates(): List<CDate> {  val list = ArrayList<CDate>()  list.*addAll*(repositoryDates.findAll())  return list  } } |

|  |
| --- |
| *Листинг 23 – CServiceExcel* |
| package ru.psu.java.laba2.makmikcro.services  import org.apache.poi.ss.usermodel.Cell import org.apache.poi.ss.usermodel.CellType import org.apache.poi.ss.usermodel.Row import org.apache.poi.ss.usermodel.Sheet import org.apache.poi.xssf.usermodel.XSSFWorkbook import org.springframework.stereotype.Service import org.springframework.web.multipart.MultipartFile import ru.psu.java.laba2.makmikcro.model.CDate import ru.psu.java.laba2.makmikcro.model.CFood import ru.psu.java.laba2.makmikcro.model.CTimeInterval import ru.psu.java.laba2.makmikcro.repositories.IRepositoryFoods import ru.psu.java.laba2.makmikcro.repositories.IRepositoryDates import ru.psu.java.laba2.makmikcro.repositories.IRepositoryTimeIntervals import java.time.LocalDate import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Загрузка из Excel.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Service class CServiceExcel (  val repositoryDates: IRepositoryDates,  val repositoryFoods: IRepositoryFoods,  val repositoryTimeIntervals: IRepositoryTimeIntervals ) : IServiceExcel {  override fun uploadExcel(file: MultipartFile) {  val wb = XSSFWorkbook(file.*inputStream*)  try {   val loadDate = loadDate(wb)  repositoryDates.saveAll(loadDate.values)  val loadTimeInterval = loadTimeInterval(wb)  repositoryTimeIntervals.saveAll(loadTimeInterval.values)  val loadFood = loadFood(wb)  loadFoodRelations(wb, loadFood, loadDate, loadTimeInterval)  repositoryFoods.saveAll(loadFood.values)  } catch (e: Exception) {  *println*("Формат файла не поддерживается!")  e.printStackTrace()  }   }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Загрузка информации о связях еды из электронной таблицы. \*  \* Результат в объектах карт foods, timeintervals, dates. \*  \* @param wb - рабочая книга с данными. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun loadFoodRelations(  wb: XSSFWorkbook,  foods: LinkedHashMap<UUID, CFood>,  dates: LinkedHashMap<UUID, CDate>,  timeintervals: LinkedHashMap<UUID, CTimeInterval>  ) {  val sheet: Sheet = wb.getSheetAt(3)  var row: Row?  var cell: Cell  val nRows = sheet.*lastRowNum* var sId: String  var sDateId: String?  var sTimeIntervalId: String?  var id: UUID?  var dateId: UUID?  var timeintervalId: UUID?  var date: CDate  var timeinterval: CTimeInterval  var food: CFood  var i = 0  while (i < nRows) {  row = sheet.getRow(i)  if (row == null) {  i++  continue  }  if (row.*lastCellNum* < 5) {  i++  continue  }  cell = row.getCell(0)  sId = cell.*stringCellValue* if (sId.*isEmpty*()) {  i++  continue  }  id = UUID.fromString(sId)  food = foods[id]!!  cell = row.getCell(3)  if (cell.*cellType* != CellType.*BLANK*) {  sDateId = cell.*stringCellValue* dateId = UUID.fromString(sDateId)  date = dates[dateId]!!  food.date = date  }  cell = row.getCell(4)  if (cell.*cellType* != CellType.*BLANK*) {  sTimeIntervalId = cell.*stringCellValue* timeintervalId = UUID.fromString(sTimeIntervalId)  timeinterval = timeintervals[timeintervalId]!!  food.timeinterval = timeinterval  }  i++  }  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Загрузка информации о временных промежутков из электронной таблицы. \*  \* Результат в карте timeintervals. \*  \* @param wb - рабочая книга с данными. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun loadTimeInterval(wb: XSSFWorkbook) : LinkedHashMap<UUID, CTimeInterval>  {  val timeintervals = LinkedHashMap<UUID, CTimeInterval>()  val sheet: Sheet = wb.getSheetAt(2)  var row: Row?  var cell: Cell  var i: Int  val nRows = sheet.*lastRowNum* var tiUUID: String  var name: String?  var vr: String?  var id: UUID?  var timeInterval: CTimeInterval  *// Перебираются строки 1 таблицы в файле .xlsx.* i = 0  while (i <= nRows) {  row = sheet.getRow(i)  *// Просматривается есть ли данные в ряду.* if (row == null) {  i++  continue  }  *// Просматривается все ли заполнены ячейки в ряду.* if (row.*lastCellNum* < 3) {  i++  continue  }  *// Достаётся UUID из 1 ячейки.* cell = row.getCell(0)  tiUUID = cell.*stringCellValue* if (tiUUID.length == 0) {  i++  continue  }   *// Создаётся новый объект класса.* timeInterval = CTimeInterval()  id = UUID.fromString(tiUUID)  *// Заполняется в объекте UUID из 1 ячейки в сторе.* timeInterval.id = id  cell = row.getCell(1)  name = cell.*stringCellValue* timeInterval.name = name  cell = row.getCell(2)  vr = cell.*stringCellValue* timeInterval.vr = vr  *// Объект помещается в карточку timeintervals.* timeintervals[id] = timeInterval  i++  }  return timeintervals;  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Загрузка информации о дате из электронной таблицы. \*  \* Результат в карте dates. \*  \* @param wb - рабочая книга с данными. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun loadDate(wb: XSSFWorkbook) : LinkedHashMap<UUID, CDate> {  val dates = LinkedHashMap<UUID, CDate>()  val sheet: Sheet = wb.getSheetAt(1)  var row: Row?  var cell: Cell  var i: Int  val nRows = sheet.*lastRowNum* var tUUID: String  var id: UUID?  var time: CDate  var date: LocalDate?  *// Перебираются строки 2 таблицы в файле .xlsx.* i = 0  while (i <= nRows) {  row = sheet.getRow(i)  *// Просматривается есть ли данные в ряду.* if (row == null) {  i++  continue  }  *// Просматривается все ли заполнены ячейки в ряду.* if (row.*lastCellNum* < 2) {  i++  continue  }  *// Достаётся UUID из 1 ячейки.* cell = row.getCell(0)  tUUID = cell.*stringCellValue* if (tUUID.length == 0) {  i++  continue  }   *// Создаётся новый объект класса.* time = CDate()  id = UUID.fromString(tUUID)  *// Заполняется в объекте UUID из 1 ячейки в сторе.* time.id = id  cell = row.getCell(1)  date = cell.*localDateTimeCellValue*.toLocalDate()  time.date = date  *// Объект помещается в карточку times.* dates[id] = time  i++  }  return dates;  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Загрузка информации об еде из электронной таблицы. \*  \* Результат в карте foods. \*  \* @param wb - рабочая книга с данными. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun loadFood(wb: XSSFWorkbook) : LinkedHashMap<UUID, CFood> {  val foods = LinkedHashMap<UUID, CFood>()  val sheet: Sheet = wb.getSheetAt(3)  var row: Row?  var cell: Cell  var i: Int  val nRows = sheet.*lastRowNum* var fUUID: String  var name: String?  var kalors: Double  var id: UUID?  var food: CFood  *// Перебираются строки 3 таблицы в файле .xlsx.* i = 0  while (i < nRows) {  row = sheet.getRow(i)  *// Просматривается есть ли данные в ряду.* if (row == null) {  i++  continue  }  *// Просматривается все ли заполнены ячейки в ряду.* if (row.*lastCellNum* < 5) {  i++  continue  }  *// Достаётся UUID из 1 ячейки.* cell = row.getCell(0)  fUUID = cell.*stringCellValue* if (fUUID.length == 0) {  i++  continue  }   *// Создаётся новый объект класса.* food = CFood()  id = UUID.fromString(fUUID)  *// Заполняется в объекте UUID из 1 ячейки в сторе.* food.id = id  *// Заполняется в объекте Название из 2 ячейки в строке.* cell = row.getCell(1)  name = cell.*stringCellValue* food.name = name  *// Заполняется в объекте Калории из 3 ячейки в строке.* cell = row.getCell(2)  kalors = cell.*numericCellValue* food.kal = kalors  *// Объект помещается в карточку foods.* foods[id] = food  i++  }  return foods;  }  } |

|  |
| --- |
| *Листинг 24 – CServiceFoods* |
| package ru.psu.java.laba2.makmikcro.services  import org.springframework.http.ResponseEntity import org.springframework.stereotype.Service import ru.psu.java.laba2.makmikcro.model.CDate import ru.psu.java.laba2.makmikcro.model.CFood import ru.psu.java.laba2.makmikcro.model.IDateWithCounter import ru.psu.java.laba2.makmikcro.repositories.IRepositoryFoods import java.time.LocalDate import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Сервис с бизнес-логикой для работы со списком еды. \*  \* @author Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Service class CServiceFoods  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Конструктор. \*  \* @param repositoryFoods - репозиторий с запросами к таблице еды в СУБД \*  \* (устанавливается Spring-ом). \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* (  val repositoryFoods : IRepositoryFoods  ) : IServiceFoods  {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список всей еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun getAll() :Iterable<CFood>  {  return repositoryFoods.findAll()  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по идентификатору. \*  \* @param id - идентификатор еды для поиска в формате UUID. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun getById(id: UUID) : ResponseEntity<CFood>  {  return repositoryFoods.findById(id)  .map **{** food **->** ResponseEntity.ok(food) **}** .orElse(ResponseEntity.notFound().build())  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по названию. \*  \* @param name - название еды для поиска в формате String. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun getByName(name: String) : List<CFood>  {  return repositoryFoods.findByNameNative(name)  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по калориям. \*  \* @param kal - калории еды для поиска в формате Double. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun getByKal(kal: Double) : List<CFood>  {  return repositoryFoods.findByKalNative(kal)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск дней с определённой едой(Бананы). \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun getByDayN() : List<CFood>  {  return repositoryFoods.findByNameNative("Бананы")  }   override fun getWithMaxFoods() : List<IDateWithCounter>  {  return repositoryFoods.getDatesWithMaxFoods()  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание/изменение еды. \*  \* @param food - данные еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun save(food: CFood)  {  repositoryFoods.save(food)  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление еды. \*  \* @param food - данные еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun delete(food: CFood)  {  repositoryFoods.delete(food)  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление еды по идентификатору. \*  \* @param id - идентификатор еды для удаления. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun deleteById(id: UUID): String  {  return if (repositoryFoods.existsById(id)) {  repositoryFoods.deleteById(id)  "Элемент удалён"  }  else  "Элемент не найден"  } } |

|  |
| --- |
| *Листинг 25 – CServiceTimeIntervals* |
| package ru.psu.java.laba2.makmikcro.services  import org.springframework.beans.factory.annotation.Autowired import org.springframework.stereotype.Service import ru.psu.java.laba2.makmikcro.model.CDate import ru.psu.java.laba2.makmikcro.model.CTimeInterval import ru.psu.java.laba2.makmikcro.repositories.IRepositoryTimeIntervals import java.util.\*  @Service class CServiceTimeIntervals : IServiceTimeIntervals{  @Autowired  lateinit var repositoryTimeIntervals: IRepositoryTimeIntervals   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание временного интервала. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun save(timeinterval: CTimeInterval) {  try {  repositoryTimeIntervals.save(timeinterval)  }  catch (e : Exception)  {  throw Exception("Не указаны обязательные данные для еды!", e)  }  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по идентификатору. \*  \* @param id - идентификатор еды для поиска в формате UUID. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun getAllByTimeIntervalId(id: UUID): List<CTimeInterval>  {  return repositoryTimeIntervals.findAllById(id)  }   override fun getAllTimeIntervals(): List<CTimeInterval> {  val list = ArrayList<CTimeInterval>()  list.*addAll*(repositoryTimeIntervals.findAll())  return list  } } |

|  |
| --- |
| *Листинг 26 – CServiceWord* |
| package ru.psu.java.laba2.makmikcro.services  import org.apache.poi.xwpf.usermodel.\* import org.springframework.stereotype.Service import ru.psu.java.laba2.makmikcro.model.CFood import ru.psu.java.laba2.makmikcro.model.IDateWithCounter import java.io.ByteArrayInputStream import java.io.ByteArrayOutputStream import java.io.IOException import java.io.InputStream import java.time.LocalDate import javax.servlet.http.HttpServletResponse  @Service class CServiceWord(val serviceFoods: IServiceFoods) : IServiceWord {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание заголовка для файла-отчёта. \*  \* @param document - заготовка файла-отчёта. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun createTitle(  document: XWPFDocument  ) {  *//Создание параграфа* val par = document.createParagraph()  *//Центрирование параграфа* par.*alignment* = ParagraphAlignment.*CENTER   //Создание куска текста* val run = par.createRun()  *//Установка содержимого текста* run.setText("Список дат с максимальным количеством еды меньше 400 калорий и количество этой еды")  *//Жирность* run.*isBold* = true  *//Шрифт* run.*fontFamily* = "!GOST\_A"  *//Размер шрифта* run.*fontSize* = 20  run.*color* = "2A52BE"  }   private fun createTitleDayN(  document: XWPFDocument  ) {  *//Создание параграфа* document.createParagraph().createRun().setText(" ")  val par = document.createParagraph()   *//Центрирование параграфа* par.*alignment* = ParagraphAlignment.*CENTER   //Создание куска текста* val run = par.createRun()   *//Установка содержимого текста* run.setText("Дни, когда были записаны Бананы")  *//Жирность* run.*isBold* = true  *//Шрифт* run.*fontFamily* = "!GOST\_A"  *//Размер шрифта* run.*fontSize* = 20  run.*color* = "4682B4"  }   *//Стиль текста шапок таблицы* private fun createHeaderCell(  row: XWPFTableRow,  pos: Int,  text: String  ) {  val par: XWPFParagraph  val run: XWPFRun  val cell: XWPFTableCell  cell = row.getCell(pos)  par = cell.addParagraph()  par.*alignment* = ParagraphAlignment.*CENTER* par.*verticalAlignment* = TextAlignment.*BOTTOM* run = par.createRun()  *//Установка содержимого текста* run.setText(text)  *//Жирность* run.*isBold* = true  *//Шрифт* run.*fontFamily* = "Times New Roman"  *//Размер шрифта* run.*fontSize* = 10  *//Цвет шрифта* run.*color* = "5171D1"  }   *//Стиль текста содержимого таблицы* private fun createCell(  row: XWPFTableRow,  pos: Int,  text: String  ) {  val par: XWPFParagraph  val run: XWPFRun  val cell: XWPFTableCell  cell = row.getCell(pos)  par = cell.addParagraph()  par.*alignment* = ParagraphAlignment.*CENTER* par.*verticalAlignment* = TextAlignment.*BOTTOM* run = par.createRun()  *//Установка содержимого текста* run.setText(text)  *//Жирность* run.*isBold* = true  *//Шрифт* run.*fontFamily* = "Times New Roman"  *//Размер шрифта* run.*fontSize* = 10  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание таблицы с едой в файле-отчёте. \*  \* @param document - заготовка файла-отчёта. \*  \* @param datesList - даты, данные которых выводятся в отчёт. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun createTableNight(  document: XWPFDocument,  datesList: Iterable<IDateWithCounter>  ) {   val table = document.createTable(1, 2)  table.*width* = 5 \* 1900  var row = table.getRow(0)  createHeaderCell(row, 0, "Дата")  createHeaderCell(row, 1, "Количество еды")   *//Создание строк с информацией по датам.* for (date in datesList) {  row = table.createRow()  date.date.*let* **{** createCell(row, 0, **it**) **}** date.count.*let* **{** createCell(row, 1, **it**.toString()) **}** }   *//Прокраска границ таблицы. Необходимость надо проверять в MS Word.* table.setBottomBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setTopBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setLeftBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setRightBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setInsideHBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setInsideVBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  }   private fun createTableDayN(  document: XWPFDocument,  foodList: List<CFood>  ) {  val table = document.createTable(1, 5)  table.*width* = 5 \* 1900  var row = table.getRow(0)  createHeaderCell(row, 0, "Дата")  createHeaderCell(row, 1, "Временной промежуток")  createHeaderCell(row, 2, "Время")  createHeaderCell(row, 3, "Еда")  createHeaderCell(row, 4, "Калории")   *//Создание строк с информацией по еде.* for (food in foodList) {  row = table.createRow()  if (food.date != null) {  food.date!!.date?.*let* **{** createCell(row, 0, **it**.toString()) **}** } else {  createCell(row, 0, "День не найден")  }  if (food.timeinterval != null) {  food.timeinterval!!.name?.*let* **{** createCell(row, 1, **it**) **}** } else {  createCell(row, 1, "Временной промежуток не найден")  }  if (food.timeinterval != null) {  food.timeinterval!!.vr?.*let* **{** createCell(row, 2, **it**) **}** } else {  createCell(row, 2, "Временной промежуток не найден")  }  food.name?.*let* **{** createCell(row, 3, **it**) **}** food.kal.*let* **{** createCell(row, 4, **it**.toString()) **}** }   *//Прокраска границ таблицы. Необходимость надо проверять в MS Word.* table.setBottomBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setTopBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setLeftBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setRightBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setInsideHBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  table.setInsideVBorder(XWPFTable.XWPFBorderType.*SINGLE*, 8, 0, "000000")  }    override fun downloadWord(response: HttpServletResponse)  {  try {   val b = ByteArrayOutputStream()  val doc = XWPFDocument()   *//Заголовок* createTitle(doc)  *//Таблица с датами.* createTableNight(doc, serviceFoods.getWithMaxFoods())  *//Заголовок 2 таблицы* createTitleDayN(doc)  *//Таблица датами еды Бананы.* createTableDayN(doc, serviceFoods.getByDayN())   doc.write(b) *// doc должен быть XWPFDocument* val inputStream: InputStream = ByteArrayInputStream(b.toByteArray())   response.*contentType* = "application/vnd.openxmlformats-officedocument.wordprocessingml.document"  org.apache.commons.io.IOUtils.copy(inputStream, response.*outputStream*);  response.flushBuffer();  } catch ( ex: IOException) {}  } } |

|  |
| --- |
| *Листинг 27 – IServiceDates* |
| package ru.psu.java.laba2.makmikcro.services  import ru.psu.java.laba2.makmikcro.model.CDate import ru.psu.java.laba2.makmikcro.model.CFood import java.util.UUID */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Интерфейс сервиса с бизнес-логикой для работы со списком полетов. \*  \* @author Макарова П.Ф. 12.03.2023. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* interface IServiceDates {  fun save(date : CDate)   fun getAllByDateId(id: UUID): List<CDate>   fun getAllDates(): List<CDate> } |

|  |
| --- |
| *Листинг 28 – IServiceExcel* |
| package ru.psu.java.laba2.makmikcro.services  import org.springframework.web.multipart.MultipartFile  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Интерфейс сервиса с бизнес-логикой для работы с входными данными. \*  \* @author Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* interface IServiceExcel {  fun uploadExcel(file : MultipartFile) *//загрузка данных* } |

|  |
| --- |
| *Листинг 29 – IServiceFoods* |
| package ru.psu.java.laba2.makmikcro.services  import org.springframework.http.ResponseEntity import ru.psu.java.laba2.makmikcro.model.CFood import ru.psu.java.laba2.makmikcro.model.IDateWithCounter import java.time.LocalDate import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Интерфейс сервиса с бизнес-логикой для работы со списком еды. \*  \* @author Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* interface IServiceFoods {  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Список всей еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getAll(): Iterable<CFood>  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по идентификатору. \*  \* @param id - идентификатор еды для поиска в формате UUID. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getById(id: UUID): ResponseEntity<CFood>  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по названию. \*  \* @param name - название еды для поиска в формате String. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getByName(name: String) : Iterable<CFood>  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск еды по калориям. \*  \* @param kal - калории еды для поиска в формате Double. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getByKal(kal: Double) : Iterable<CFood>  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Поиск дней с определённой едой(Бананы). \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getByDayN() : List<CFood>   fun getWithMaxFoods() : List<IDateWithCounter>   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание/изменение еды. \*  \* @param food - данные еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save(food: CFood)  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление еды. \*  \* @param food - данные еды. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(food: CFood)  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление еды по идентификатору. \*  \* @param id - идентификатор еды для удаления. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun deleteById(id: UUID): String } |

|  |
| --- |
| *Листинг 30 – IServiceTimeIntervals* |
| package ru.psu.java.laba2.makmikcro.services  import ru.psu.java.laba2.makmikcro.model.CTimeInterval import java.util.\*  interface IServiceTimeIntervals {  fun save(timeinterval : CTimeInterval)   fun getAllByTimeIntervalId(id: UUID): List<CTimeInterval>   fun getAllTimeIntervals(): List<CTimeInterval> } |

|  |
| --- |
| *Листинг 31 – IServiceWord* |
| package ru.psu.java.laba2.makmikcro.services  import javax.servlet.http.HttpServletResponse  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Интерфейс сервиса с бизнес-логикой для построения результатирующего файла. \*  \* @author Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* interface IServiceWord {  fun downloadWord(response : HttpServletResponse) *//получение отчёта* } |

|  |
| --- |
| *Листинг 32 – CApplication* |
| package ru.psu.java.laba2.makmikcro  import org.springframework.boot.autoconfigure.SpringBootApplication import org.springframework.boot.runApplication  @SpringBootApplication class CApplication  fun main(args: Array<String>) {  *runApplication*<CApplication>(\*args) } |