**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 №3 на тему**

**«Технологии Java для создания настольных 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](#_Toc130933641)

[2. Постановка задачи 5](#_Toc130933642)

[3. Используемый инструментарий 6](#_Toc130933643)

[4. Модель данных 7](#_Toc130933644)

[5. Интерфейс программы 9](#_Toc130933645)

[6. Обмен данных с серверной частью 17](#_Toc130933646)

[7. Тестирование программы 20](#_Toc130933647)

[8. Заключение 21](#_Toc130933648)

[9. Список литературы 22](#_Toc130933649)

[Приложение 23](#_Toc130933650)

# Введение

Цель работы:

Ознакомление с существующими технологиями Java для создания кроссплатформенных приложений для персональных компьютеров, обладающих графическим интерфейсом пользователя.

Краткое описание выбранной предметной области: дневник еды.

Хотелось бы иметь доступ к списку имеющихся еды, дат и временных промежутков, возможность добавлять новые, удалять и редактировать. При данном подходе будет совершенно меньше ошибок при работе с дневником.

Краткая теорию по новому материалу, технологиям и библиотекам, которые будет использованы при выполнении данной работы:

JavaFX расширяет возможности Java, позволяя разработчикам использовать любые библиотеки Java в приложениях JavaFX. Таким образом разработчики получают доступ к расширенным возможностям Java и могут использовать технологию презентаций JavaFX, предназначенную для создания привлекательных визуальных образов.

• Позволяет разработчикам интегрировать веб-ресурсы векторной графики, анимации, аудио и видео в многофункциональные, интерактивные и реалистичные приложения.

• Расширяет возможности технологии Java, позволяя использовать любую библиотеку Java в приложении JavaFX.

• Позволяет организовать эффективный рабочий процесс на всех этапах от проектирования до разработки. При этом проектировщики могут самостоятельно выбирать инструменты для взаимодействия с разработчиками. [1]

TornadoFX является фреймворком JavaFX для Kotlin. Он разработан, чтобы раскрыть возможности Kotlin для различных целей, таких как определение представлений, внедрение зависимостей, делегированные свойства, функции расширения управления и другие практические функции. [2]

TornadoFX - это фреймворк MVC. Представление содержит корневой узел пользовательского интерфейса, а также логику представления. Контроллеры содержат бизнес-логику. Элегантная поддержка асинхронности гарантирует, что длительные задачи выполняются в фоновом потоке и что результат применяется в потоке пользовательского интерфейса.

Она поставляется с клиентом REST и встроенной поддержкой JSON. Другие функции включают внедрение зависимостей, автоматическую загрузку представления FXML, безопасные для типов конструкторы и различные усовершенствования стандартного JavaFX API, ставшие возможными благодаря функциям расширения Kotlin. [3]

# Постановка задачи

Написать кроссплатформенное приложение для настольных персональных компьютеров, позволяющее работать с данными дневника еды (вариант № 18).

Приложение должно иметь графический интерфейс пользователя.

Приложение должно позволять просматривать, добавлять, удалять, редактировать данные, которые хранятся на сервере и которые требуются для построения аналитического отчёта.

Программа должна использовать инструменты Java, Java SDK, коллекции, java.time.\*, TornadoFX.

# Используемый инструментарий

Таблица 1

|  |  |
| --- | --- |
| Язык программирования | Kotlin |
| Версия JDK | 16 |
| Среда разработки | IntelliJ IDEA Ultimate 2022.2.4 |
| Система автоматизированной сборки | Gradle |
| Инструменты JDK | java.util.Collections  java.time  java.util.UUID |
| Библиотеки | TornadoFX  SpringFramework  PostgreSQL-коннектор |

# Модель данных

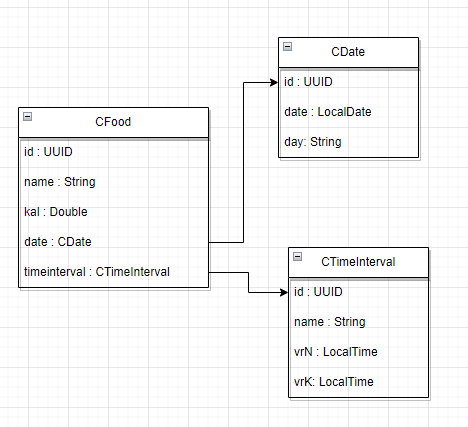


Рисунок 1 – Диаграмма классов данных предметной области

Модель данных представляет собой набор классов с полями:

CDate - дата:

* 1. идентификатор даты, UUID id;
  2. дата, LocalDate date;
  3. день недели, String day.

CTimeInterval – временной промежуток:

1. идентификатор временного промежутка, UUID id;
2. название, String name;
3. начало временного промежутка, LocalTime vrN;
4. конец временного промежутка, LocalTime vrK.

CFood – блюдо(еда):

1. идентификатор еды, UUID id;
2. название, String name;
3. калории, Double kal;
4. идентификатор даты, UUID id;
5. идентификатор временного промежутка, UUID id.

# Интерфейс программы

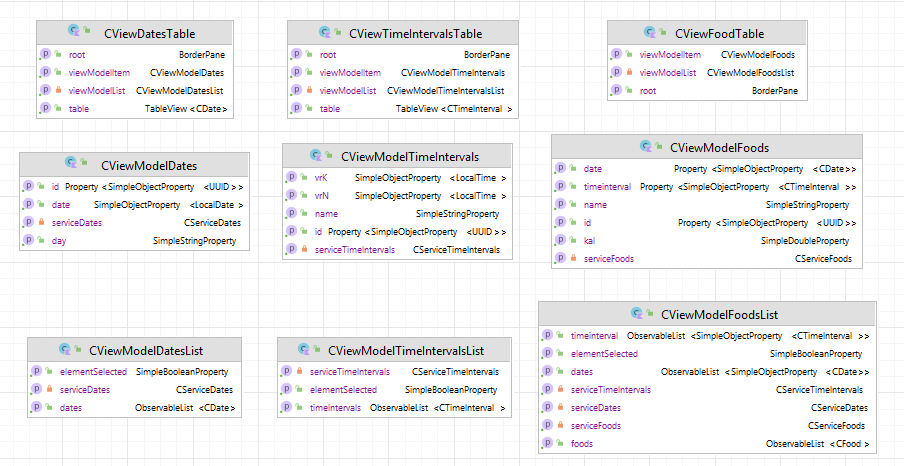


Рисунок 2 – Диаграмма классов окон программы и моделей представления

Интерфейс программы состоит из 3 окон. В каждом из них есть таблица со списком из модели представления, форма для редактирования и различные кнопки.

**Окно таблицы с блюдами:**

Окно блюд позволяет нам изменять собственные поля блюда и выбирать дату и временной интервал. Удалить блюдо или создать новое, сохранить изменения в базу данных.

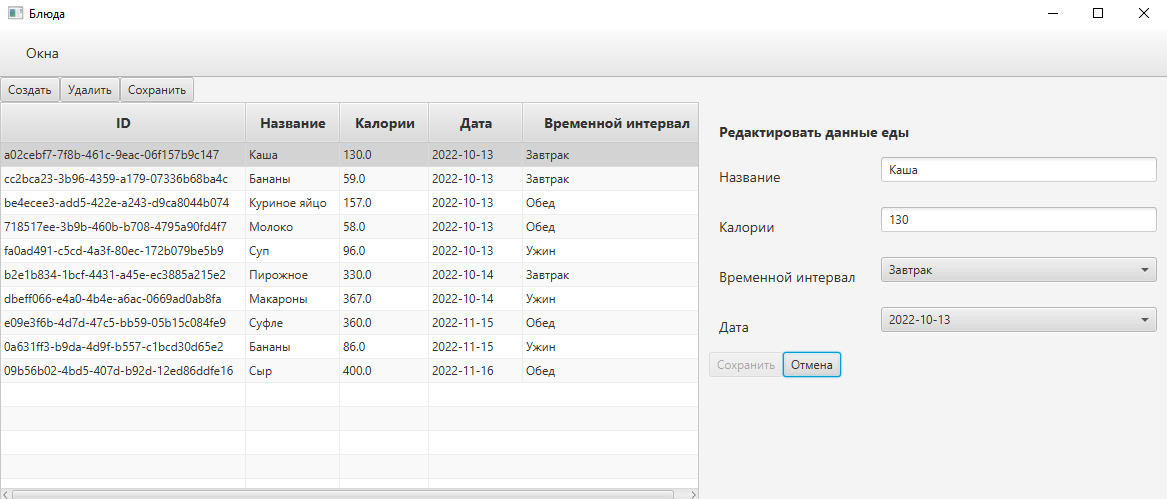


Рисунок – Окно блюд

Пока не внесены изменения кнопка «Сохранить» горит серым.

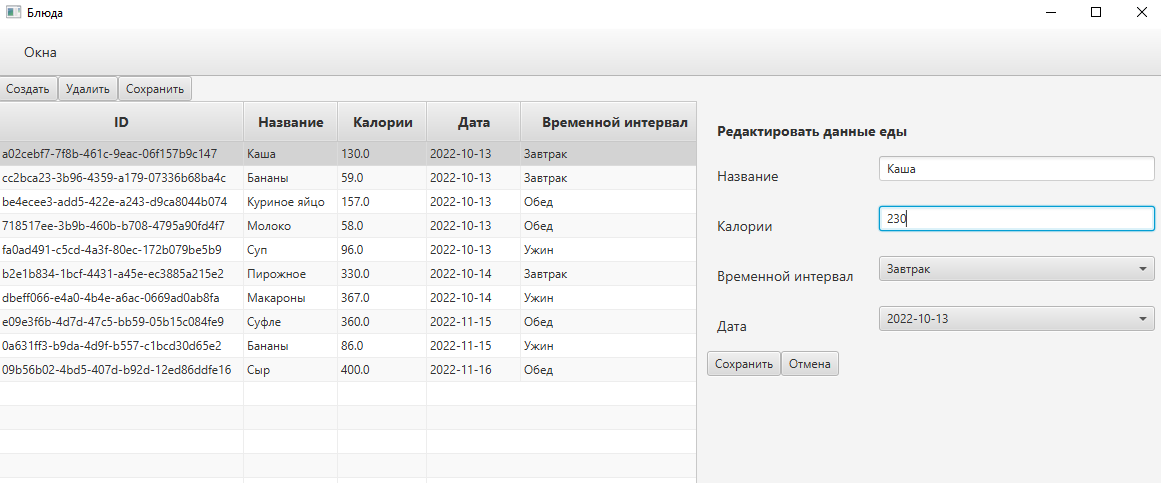


Рисунок – Изменение калорий

Временной интервал и дату можно выбирать из соответствующих списков временного интервала и даты в редактировании данных еды.

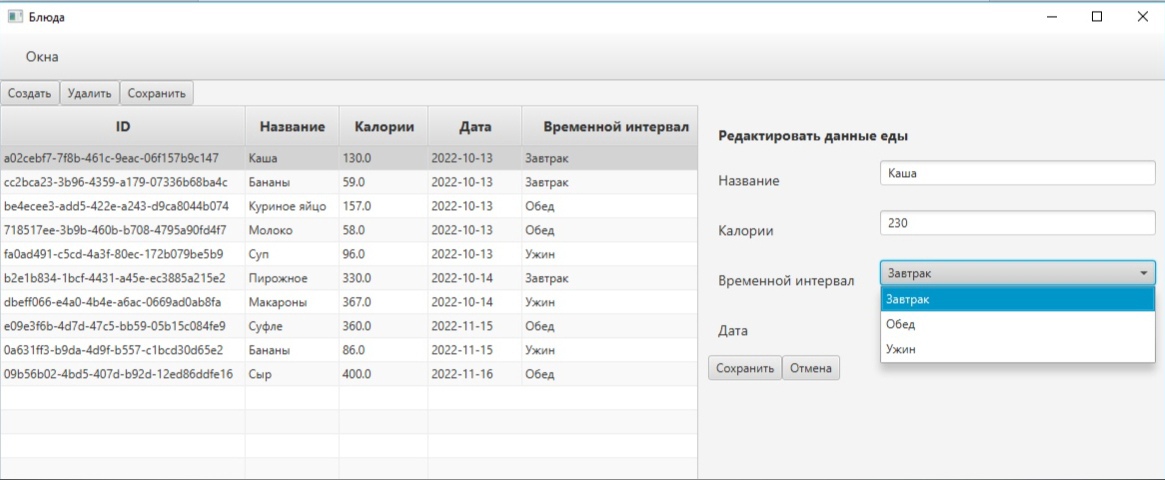


Рисунок – Выбор временного промежутка

При сохранении изменённых данных кнопка «Сохранить» снова гаснет.

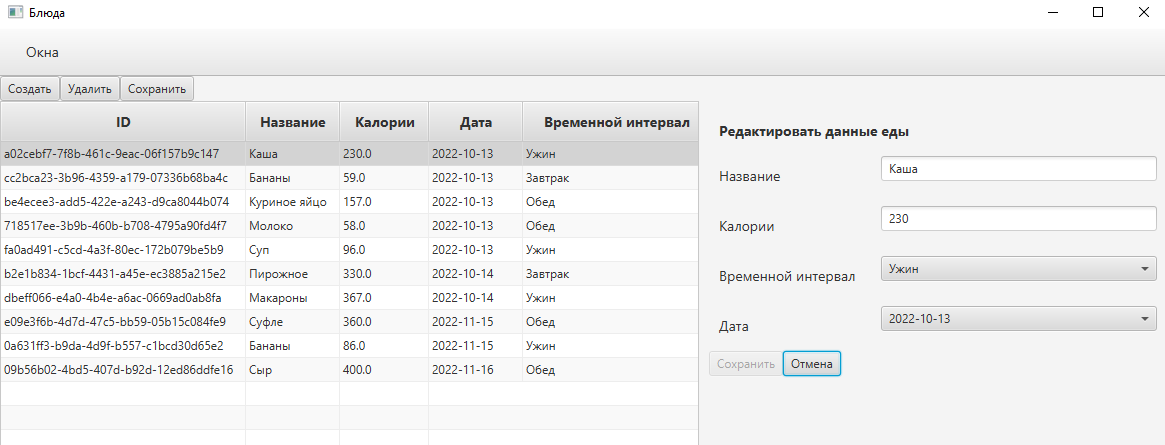


Рисунок - – Изменённые данные

Для создания нового блюда, нажимается кнопка «Создать» вверху таблицы. Новое блюдо создаётся со случайным id и нулевым значением калорий.

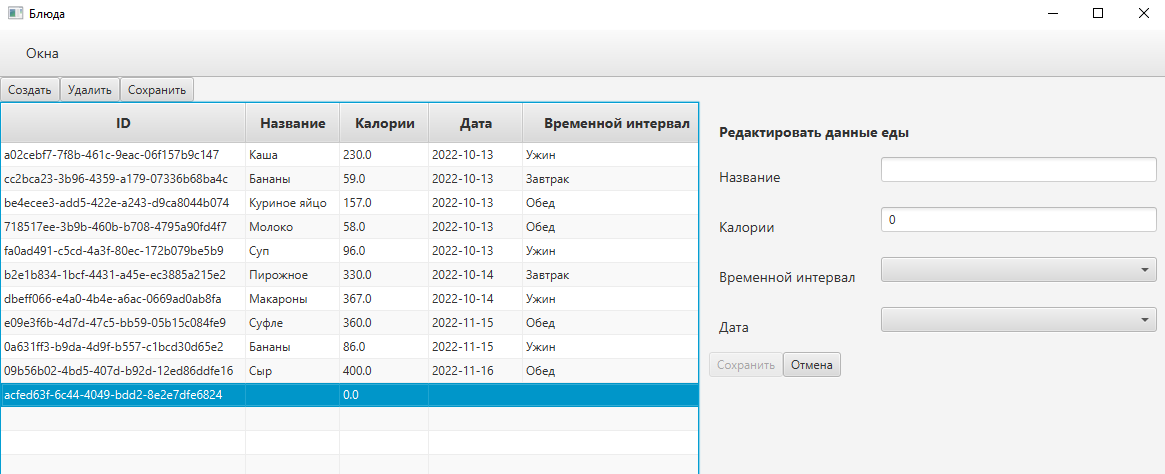


Рисунок – Создано новое блюдо

Данные нового блюда можно изменить и сохранить. Так же можно удалить блюдо, нажав на кнопку «Удалить» вверху таблицы.

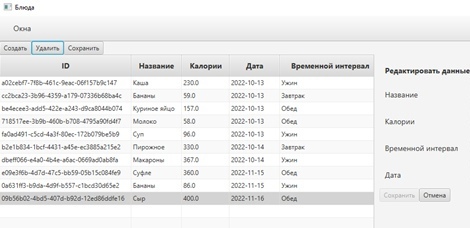


Рисунок – Удалено блюдо

Нажав на кнопку «Сохранить» вверху таблицы, данные будут сохраняться в базу данных.

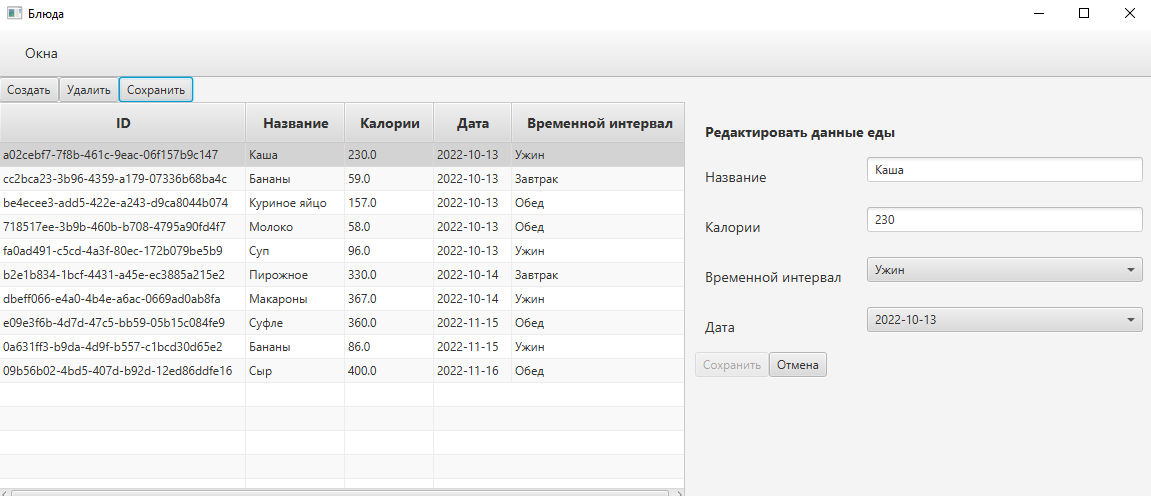


Рисунок – Сохранение данных в базу данных

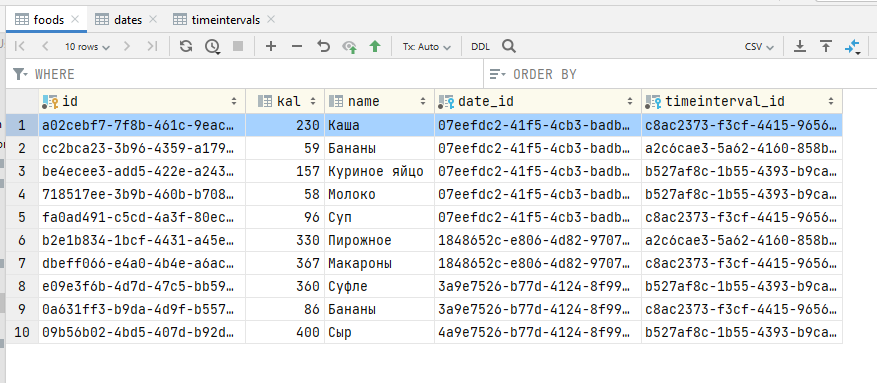


Рисунок – Изменённая таблица в базе данных

В базе данных в таблице блюда хранятся только id даты и временного промежутка. Если посмотреть на id ужина и id хранящееся в изменённом значении временного промежутка, то мы увидим, что они одинаковые, следовательно, данные сохранились правильно.

Предусмотрен переход на другие страницы, нажав на кнопку «Окно» в верху таблицы. При нажатии всплывает окно с выбором перехода на две другие таблицы.

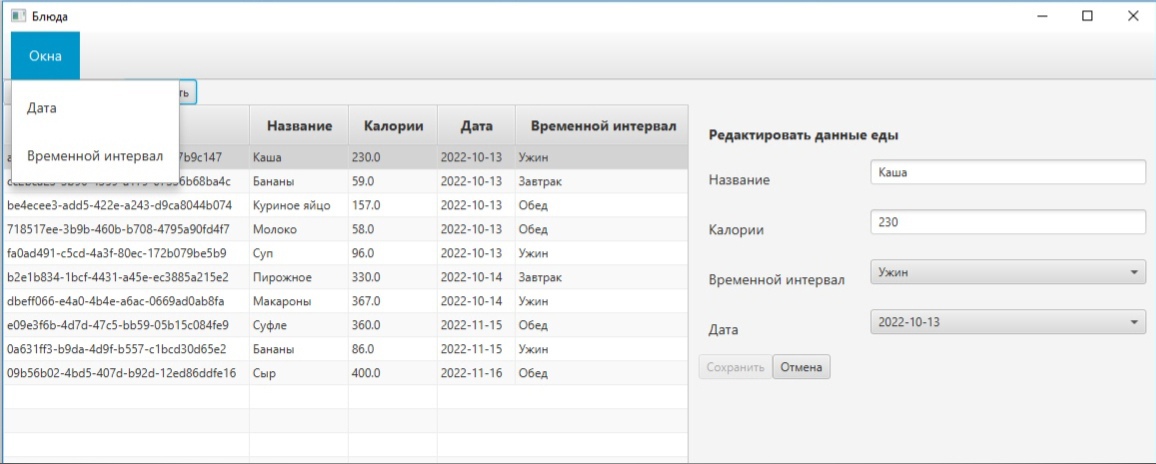


Рисунок – Выбор окон

**Окно таблицы с датами:**

Окно дат представляет собой таблицу с данными дат и форму для редактирования этих данных. В окне дат можно также создать новую дату, удалить, сохранить в базу данных и изменить данные.

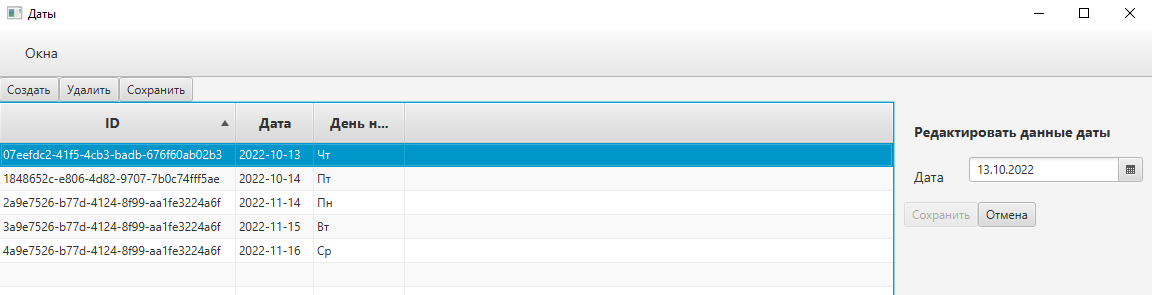


Рисунок – Окно дат

Дату рождения можно выбрать через календарь или написать самому. День недели меняется автоматически.

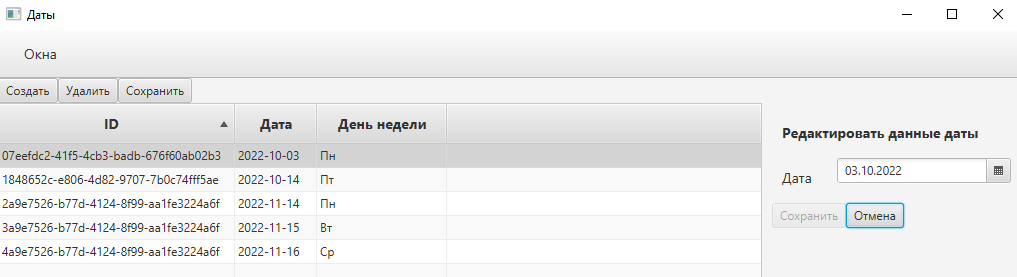


Рисунок – Изменение даты рождения

Сохраняем данные в базу данных. Проверив на соответствие id в таблице и id в базе данных делаем вывод, что данные изменились.

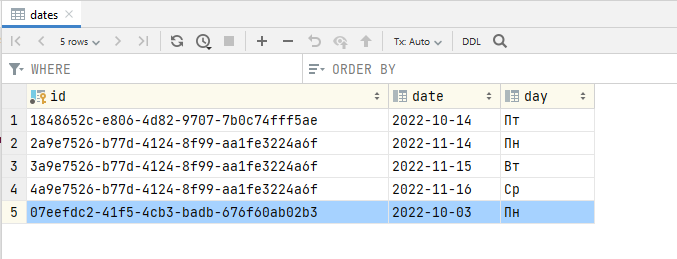


Рисунок – Изменённая таблица в базе данных

При создании новой даты вставляются начальные значения 1972.09.27 (1000 дней с начала эпохи Unix).

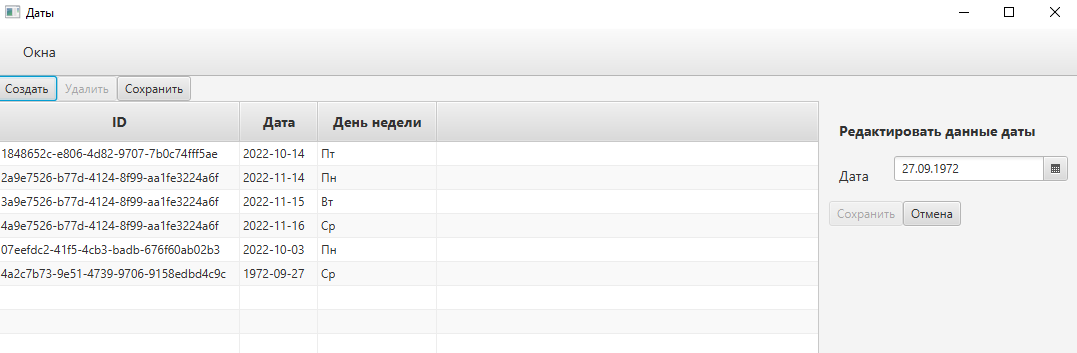


Рисунок – Создание новой даты

Предусмотрен переход на другие две таблицы.

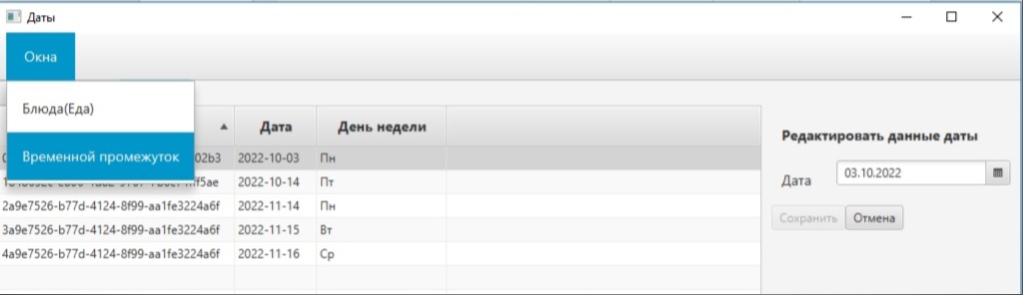


Рисунок – Выбор окон

**Окно таблицы с временными промежутками:**

Окно временных промежутков представляет собой таблицу с данными временных промежутков и форму для редактирования этих данных. В окне временных промежутков можно также создать новый временной промежуток, удалить, сохранить в базу данных и изменить данные.

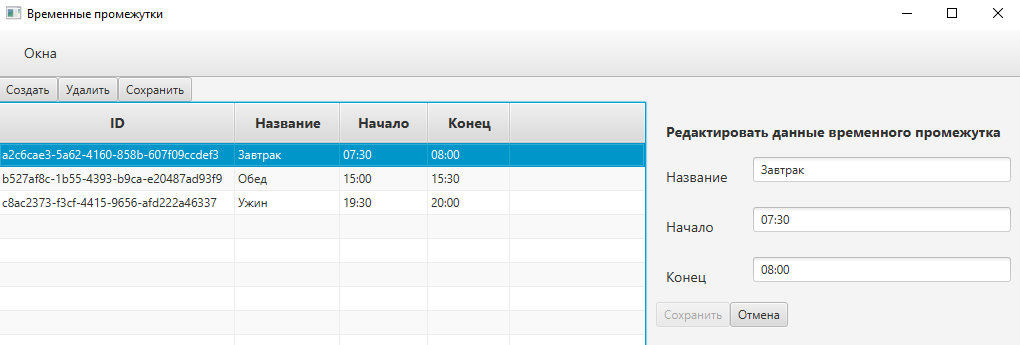


Рисунок – Окно временных промежутков

Сохранение данных в базу данных.

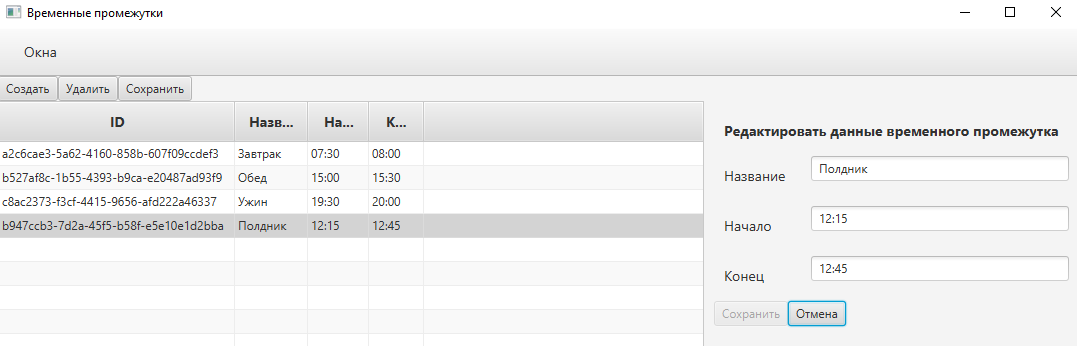


Рисунок – Создание нового временного промежутка

Сохраняем данные в базу данных. Проверив на соответствие id в таблице и id в базе данных делаем вывод, что данные сохранились.

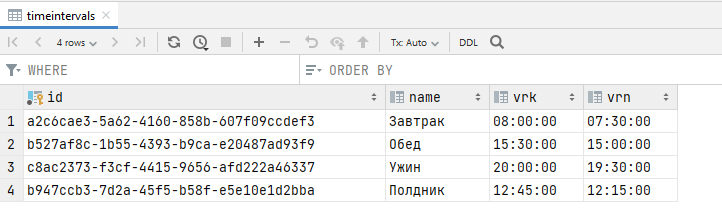


Рисунок – Изменённая таблица в базе данных

Предусмотрен переход на другие две таблицы.

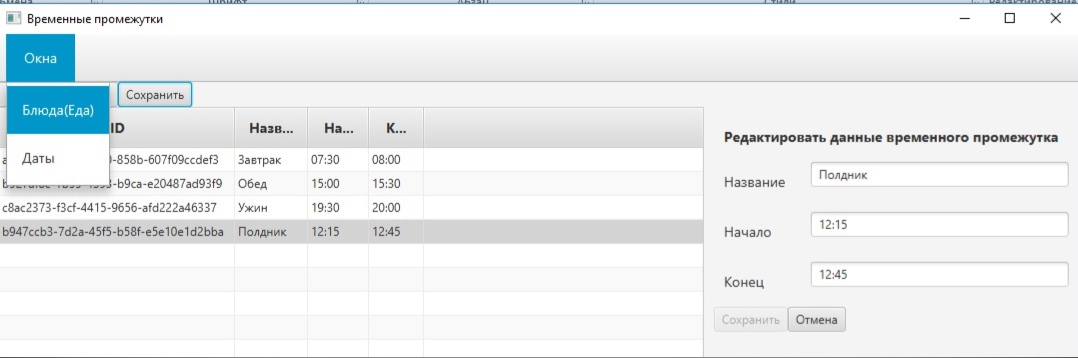


Рисунок – Выбор окон

# Обмен данных с серверной частью

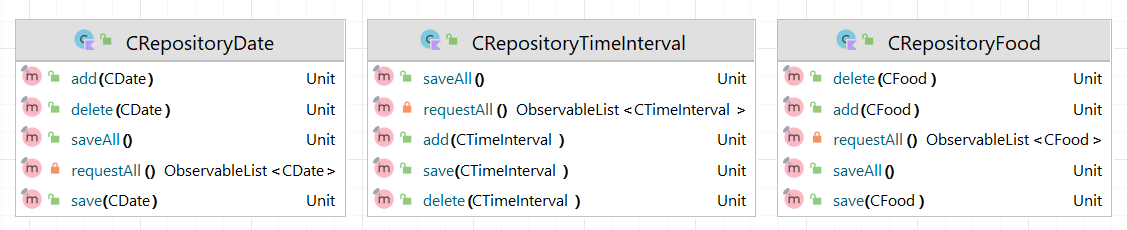


Рисунок – Диаграмма классов- репозиториев

У каждого из классов методы (API запросы) одинаковые. Разберём их на примере репозитория блюда CRepositoryFood.

Общие для всех методов api - Объект для отправки запросов к API сервера и foods - список блюд для локальной работы:

|  |
| --- |
| *Листинг 1 – CRepositoryFood* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Репозиторий с запросами к серверу в части списка еды.   \* @author Макарова П.Ф. 26.03.2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CRepositoryFood : Controller() {  *//Объект для отправки запросов к API сервера.* private val api: Rest by inject()   *//"Кэшированный" список еды для локальной работы.* private val foods: ObservableList<CFood> = FXCollections.observableArrayList() |

1. getAll - Запрос списка всех доступных объектов на сервере

|  |
| --- |
| *Листинг 2 – getAll* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка всех доступных объектов на сервере.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getAll(): ObservableList<CFood> {  if(foods.*isNotEmpty*())  return foods  *//Запрашиваем актуальные данные на сервере* val listFromServer = requestAll()  *//Пересохраняем в локальный список,  //в котором дополнительно будем кэшировать изменения в процессе  //редактирования таблицы.* foods.addAll(listFromServer)  *//Возвращаем ссылку на кэш.* return foods } |

1. requestAll - Запрос списка еды на сервере и обработка возможных проблем

|  |
| --- |
| *Листинг 3 – requestAll* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка еды на сервере и обработка возможных проблем. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun requestAll(): ObservableList<CFood> {  *//Собственно выполнение GET запроса к пути /foods* val response: Rest.Response = api.get("foods")  try {  if (response.ok()) {  *//Конвертация json в список объектов типа CFood* return response  .list()  .*toModel*()  } else if (response.statusCode == 404)  throw Exception("404")  else  throw Exception("getCustomer returned ${response.statusCode} ${response.reason}")  } catch (e: Exception) {  throw Exception("Отсутствует соединение с сервером.", e)  } finally {  response.consume()  } } |

1. save - Изменение блюда в локальном списке

|  |
| --- |
| *Листинг 4 – save* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Изменение еды в локальном списке.   \* @param food - объект для отправки.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save(food: CFood) {  *//Не стали делать отправку, т.к. приняли централизованно все изменения для списка отправлять вместе.* } |

1. saveAll - Отправка списка объектов на сервер

|  |
| --- |
| *Листинг 5 – saveAll* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка списка объектов на сервер. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll() {  *//Запрос данных с сервера.* val foodsFromServer = requestAll()   *//Перебираем всю еду с сервера,  //ищем ту, кто отсутствует в локальном списке,  //для них отправляем запрос на удаление.* foodsFromServer  .*forEach* **{** foodFromServer **->** *//Для того, чтобы метод contains сработал правильно,  //и чтобы не писать ручной перебор,  //переопределён метод Food.equals* if (!foods.contains(foodFromServer)) {  api.delete("foods", foodFromServer)  }  **}** *//Перебираем все записи из локального кэша* var temp: List<CFood>  foods  .*forEach* **{** foodLocal **->** *//Для каждой локальной записи находим записи из сервера с такими же id.* temp = foodsFromServer  .*filter* **{** foodFromServer **->** foodLocal.equals(foodFromServer)  **}** *//Записи с такими же id  // (должна быть максимум 1, но гарантий никто не даст)  // фильтруем по наличию изменений* temp  .*firstOrNull* **{** foodFromServer **->** foodLocal.hasChanges(foodFromServer)  **}** *//Если изменения в полях есть, публикуем текущую запись на сервер.* ?.*let* **{** api.post("foods", foodLocal)  **}** *//Если вообще на сервере записей с таким id нет  //публикуем текущую запись.* if (temp.isEmpty())  api.post("foods", foodLocal)   **}** } |

1. add - Добавление блюда в локальный список

|  |
| --- |
| *Листинг 6 – add* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Добавление еды в локальный список.   \* @param food - объект для отправки. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add(food: CFood) {  this.foods.add(food) } |

1. delete - Удаление блюда из локального списка

|  |
| --- |
| *Листинг 7 – delete* |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление еды из локального списка.   \* @param food - объект для отправки. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(food: CFood) {  foods.remove(food) } |

# Тестирование программы

Большинство показательных тестов было проведено на этапе разбора интерфейса.

Скаченный через запрос в Postman файл-отчёт после редактирования таблицы через интерфейс:



Рисунок – Полученный файл

# Заключение

Для интерфейса разработано 3 класса типа View и 6 классов типа ViewModel.

Поставленные перед данной работой цели были выполнены успешно.

Получены знания по работе с TornadoFX.

# **Список литературы**

1. Общая информация о JavaFX [Электронный ресурс]. - URL: <https://www.java.com/ru/download/help/javafx.html> (дата обращения: 28.03.2023)
2. Новая платформа приложений JavaFX для Kotlin — TornadoFX [Электронный ресурс]. - URL: <https://dzone.com/articles/a-new-javafx-app-framework-for-kotlin-tornadofx> (дата обращения: 28.03.2023)
3. Введение в TornadoFX[Электронный ресурс]. - URL: <https://translated.turbopages.org/proxy_u/en-ru.ru.4a0a315d-64231cc6-101a9359-74722d776562/https/www.baeldung.com/kotlin/tornadofx-intro> (дата обращения: 28.03.2023)

Приложение

|  |
| --- |
| *Листинг 8 – MyApp* |
| package com.example.demo.app  import com.example.demo.view.CViewFoodTable import tornadofx.\* import java.util.\*  class MyApp :App(CViewFoodTable::class, Styles::class) {  private val api : Rest by inject()  init {  *//Язык интерфейса приложения.* FX.locale = Locale("ru")  *//Базовая часть адреса для подключения к API сервера.* api.baseURI = "http://localhost:8080/"  } } |

|  |
| --- |
| *Листинг 9 – Styles* |
| package com.example.demo.app  import javafx.scene.text.FontWeight import tornadofx.Stylesheet import tornadofx.box import tornadofx.cssclass import tornadofx.*px* class Styles : Stylesheet() {  companion object {  val heading by *cssclass*()  val label by *cssclass*()  }   init {  heading **{** padding = *box*(10.*px*)  fontSize = 20.*px* fontWeight = FontWeight.*BOLD* **}** label **{** padding = *box*(10.*px*)  fontSize = 14.*px* **}** } } |

|  |
| --- |
| *Листинг 10 – CDate* |
| package com.example.demo.model  import javafx.beans.property.SimpleObjectProperty import javafx.beans.property.SimpleStringProperty import tornadofx.\* import java.time.DayOfWeek import java.time.LocalDate import java.util.\* import javax.json.JsonObject */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Класс дата \*  \* @autor Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CDate(  id: UUID?= null,  date: LocalDate= LocalDate.ofEpochDay(1000),  day: String= "" ): JsonModel {  val propertyId = SimpleObjectProperty(id)  var id by propertyId   val propertyDate = SimpleObjectProperty(date)  var date by propertyDate   val propertyDay = SimpleStringProperty(day)  var day by propertyDay   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Обновление полей текущего класса по данным из JSON. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun updateModel(json: JsonObject)  {  *with*(json) **{** id = UUID.fromString(*string*("id"))  if(*string*("date") != null)  date = LocalDate.parse(*string*("date"))  day = *string*("day")  **}** }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание JSON по данным из полей текущего класса. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun toJSON(json: JsonBuilder)  {  *with*(json) **{** add("id", id)  add("date", date?.toString())  add("day", day)  **}** }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Проверка на соответствие двух объектов друг другу по идентификтаорам. \*  \* @param other - другой обект для проверки. \*  \* @return true, если объект описывает туже сущность. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun equals(other: Any?)=(other is CDate) && id == other.id   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Проверка наличия изменений в объекте по сравнению с другим объектом. \*  \* @param other - другой обект для проверки. \*  \* @return true, если объект описывает ту же сущность, но имеет отличия в полях. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun hasChanges(other: CDate)=equals(other) && *//Совсем другой объект не считаем изменённым текущим.* ( *//Изменения в любом другом поле считаем изменениями в объекте.* date != other.date ||  day != other.day  )   override fun toString(): String {  return "Выберите значение"  } } |

|  |
| --- |
| *Листинг 11 –CFood* |
| package com.example.demo.model  import javafx.beans.property.SimpleDoubleProperty import javafx.beans.property.SimpleObjectProperty import javafx.beans.property.SimpleStringProperty import tornadofx.\* import java.util.\* import javax.json.JsonObject  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Класс блюдо(еда) \*  \* @autor Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CFood(  id: UUID? = null,  name: String = "",  kal: Double = 0.0,  date: CDate? = null,  timeinterval: CTimeInterval? = null ) : JsonModel {  val propertyId = SimpleObjectProperty(id)  var id by propertyId   val propertyName = SimpleStringProperty(name)  var name by propertyName  val propertyKal = SimpleDoubleProperty(kal)  var kal by propertyKal  val propertyDate = SimpleObjectProperty(date)  var date by propertyDate  val propertyTimeInterval = SimpleObjectProperty(timeinterval)  var timeinterval by propertyTimeInterval    */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Обновление полей текущего класса по данным из JSON. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun updateModel(json: JsonObject) {  *with*(json) **{** id = UUID.fromString(*string*("id"))  name = *string*("name")   kal = *double*("kal")!! *//string("kal")* var tmp\_date: JsonObject? = *jsonObject*("date")  date = CDate(UUID.fromString( tmp\_date?.getString("id")) )  var tmp\_timeInterval: JsonObject? = *jsonObject*("timeinterval")  timeinterval = CTimeInterval(UUID.fromString( tmp\_timeInterval?.getString("id")) )  **}** }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание JSON по данным из полей текущего класса. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun toJSON(  json: JsonBuilder  ) {  *with*(json) **{** add("id", id)  add("name", name)  add("kal", kal)  add("date", date?.id)  add("timeinterval", timeinterval?.id)  **}** }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Проверка на соответствие двух объектов друг другу по идентификтаорам. \*  \* @param other - другой объект для проверки. \*  \* @return true, если объект описывает туже сущность. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun equals(other: Any?) = (other is CFood) && id == other.id   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Проверка наличия изменений в объекте по сравнению с другим объектом. \*  \* @param other - другой объект для проверки. \*  \* @return true, если объект описывает ту же сущность, но имеет отличия в полях. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun hasChanges(other: CFood) = equals(other) && *//Совсем другой объект не считаем изменённым текущим.* ( *//Изменения в любом другом поле считаем изменениями в объекте.* name != other.name || kal != other.kal ||  date?.id != other.date?.id || timeinterval?.id != other.timeinterval?.id   )   override fun toString(): String {  return "food(id=$id, name='$name', kal='$kal', date=$date, timeinterval=$timeinterval)"  } } |

|  |
| --- |
| *Листинг 12 –CTimeInterval* |
| package com.example.demo.model  import javafx.beans.property.SimpleObjectProperty import javafx.beans.property.SimpleStringProperty import tornadofx.\* import java.time.LocalDate import java.util.\* import javax.json.JsonObject import java.time.LocalTime  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Класс временной интервал \*  \* @autor Макарова П.Ф. 12.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CTimeInterval(  id: UUID?= null,  name: String= "",  vrN: LocalTime = LocalTime.*MIN*,  vrK: LocalTime = LocalTime.*MIN* ): JsonModel {  val propertyId = SimpleObjectProperty(id)  var id by propertyId   val propertyName = SimpleStringProperty(name)  var name by propertyName   val propertyVrN = SimpleObjectProperty(vrN)  var vrN by propertyVrN   val propertyVrK = SimpleObjectProperty(vrK)  var vrK by propertyVrK    */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Обновление полей текущего класса по данным из JSON. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun updateModel(json:JsonObject)  {  *with*(json) **{** id = UUID.fromString(*string*("id"))  name = *string*("name")  vrN = LocalTime.parse(*string*("vrN"))  vrK = LocalTime.parse(*string*("vrK"))  **}** }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Создание JSON по данным из полей текущего класса. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun toJSON(json: JsonBuilder)  {  *with*(json) **{** add("id", id)  add("name", name)  add("vrN", vrN.toString())  add("vrK", vrK.toString())  **}** }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Проверка на соответствие двух объектов друг другу по идентификтаорам. \*  \* @param other - другой обект для проверки. \*  \* @return true, если объект описывает туже сущность. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* override fun equals(other:Any?)=(other is CTimeInterval) && id == other.id   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Проверка наличия изменений в объекте по сравнению с другим объектом. \*  \* @param other - другой обект для проверки. \*  \* @return true, если объект описывает ту же сущность, но имеет отличия в полях. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun hasChanges(other:CTimeInterval)=equals(other) && *//Совсем другой объект не считаем изменённым текущим.* ( *//Изменения в любом другом поле считаем изменениями в объекте.* name != other.name ||  vrN != other.vrN||vrK != other.vrK)   override fun toString(): String {  return "Выберите значение"  } } |

|  |
| --- |
| *Листинг 13 –CRepositoryDate* |
| package com.example.demo.repositories  import com.example.demo.model.CDate import javafx.collections.FXCollections import javafx.collections.ObservableList import tornadofx.Controller import tornadofx.Rest import tornadofx.toModel  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Репозиторий с запросами к серверу в части списка даты. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CRepositoryDate : Controller() {  *//Объект для отправки запросов к API сервера.* private val api: Rest by inject()   *//"Кэшированный" список дат для локальной работы.* private val dates = FXCollections.observableArrayList<CDate>()   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка всех доступных объектов на сервере. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getAll(): ObservableList<CDate> {  if(dates.*isNotEmpty*())  return dates  *//Запрашиваем актуальные данные на сервере* val listFromServer = requestAll()  *//Пересохраняем в локальный список,  //в котором дополнительно будем кэшировать изменения в процессе  //редактирования таблицы.* dates.addAll(listFromServer)  *//Возвращаем ссылку на кэш.* return dates  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка дат на сервере и обработка возможных проблем. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun requestAll(): ObservableList<CDate> {  *//Собственно выполнение GET запроса к пути /dates* val response: Rest.Response = api.get("dates")  try {  if (response.ok()) {  *//Конвертация json в список объектов типа CDate* return response  .list()  .*toModel*()  } else if (response.statusCode == 404)  throw Exception("404")  else  throw Exception("getCustomer returned ${response.statusCode} ${response.reason}")  } catch (e: Exception) {  throw Exception("Отсутствует соединение с сервером.", e)  } finally {  response.consume()  }  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Изменение даты в локальном списке. \*  \* @param date - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save(date: CDate) {  *//Не стали делать отправку, т.к. приняли централизованно все изменения для списка отправлять вмсте.* }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка списка объектов на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll() {  *//Запрос данных с сервера.* val datesFromServer = requestAll()   *//Перебираем всех дат с сервера,  //ищем тех, кто отсутствует в локальном списке,  //для них отправляем запрос на удаление.* datesFromServer  .*forEach* **{** dateFromServer **->** *//Для того, чтобы метод contains сработал правильно,  //и чтобы не писать ручной перебор,  //переопределён метод Date.equals* if (!dates.contains(dateFromServer)) {  api.delete("dates", dateFromServer)  }  **}** *//Перебираем все записи из локального кэша* var temp: List<CDate>  dates  .*forEach* **{** dateLocal **->** *//Для каждой локальной записи находим записи из сервера с такими же id.* temp = datesFromServer  .*filter* **{** dateFromServer **->** dateLocal.equals(dateFromServer)  **}** *//Записи с такими же id  // (должна быть максимум 1, но гарантий никто не даст)  // фильтруем по наличию изменений* temp  .*firstOrNull* **{** dateFromServer **->** dateLocal.hasChanges(dateFromServer)  **}** *//Если изменения в полях есть, публикуем текущую запись на сервер.* ?.*let* **{** api.post("dates", dateLocal)  **}** *//Если вообще на сервере записей с таким id нет  //публикуем текущую запись.* if (temp.isEmpty())  api.post("dates", dateLocal)   **}** }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Добавление даты в локальный список. \*  \* @param date - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add(date: CDate) {  dates.add(date)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление даты из локального списка. \*  \* @param date - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(date: CDate) {  dates.remove(date)  } } |

|  |
| --- |
| *Листинг 14 –CRepositoryFood* |
| package com.example.demo.repositories  import com.example.demo.model.CFood import javafx.collections.FXCollections import javafx.collections.ObservableList import tornadofx.Controller import tornadofx.Rest import tornadofx.toModel  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Репозиторий с запросами к серверу в части списка еды. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CRepositoryFood : Controller() {  *//Объект для отправки запросов к API сервера.* private val api: Rest by inject()   *//"Кэшированный" список еды для локальной работы.* private val foods: ObservableList<CFood> = FXCollections.observableArrayList()   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка всех доступных объектов на сервере. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getAll(): ObservableList<CFood> {  if(foods.*isNotEmpty*())  return foods  *//Запрашиваем актуальные данные на сервере* val listFromServer = requestAll()  *//Пересохраняем в локальный список,  //в котором дополнительно будем кэшировать изменения в процессе  //редактирования таблицы.* foods.addAll(listFromServer)  *//Возвращаем ссылку на кэш.* return foods  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка еды на сервере и обработка возможных проблем. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun requestAll(): ObservableList<CFood> {  *//Собственно выполнение GET запроса к пути /foods* val response: Rest.Response = api.get("foods")  try {  if (response.ok()) {  *//Конвертация json в список объектов типа CFood* return response  .list()  .*toModel*()  } else if (response.statusCode == 404)  throw Exception("404")  else  throw Exception("getCustomer returned ${response.statusCode} ${response.reason}")  } catch (e: Exception) {  throw Exception("Отсутствует соединение с сервером.", e)  } finally {  response.consume()  }  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Изменение еды в локальном списке. \*  \* @param food - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save(food: CFood) {  *//Не стали делать отправку, т.к. приняли централизованно все изменения для списка отправлять вместе.* }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка списка объектов на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll() {  *//Запрос данных с сервера.* val foodsFromServer = requestAll()   *//Перебираем всю еду с сервера,  //ищем ту, кто отсутствует в локальном списке,  //для них отправляем запрос на удаление.* foodsFromServer  .*forEach* **{** foodFromServer **->** *//Для того, чтобы метод contains сработал правильно,  //и чтобы не писать ручной перебор,  //переопределён метод Food.equals* if (!foods.contains(foodFromServer)) {  api.delete("foods", foodFromServer)  }  **}** *//Перебираем все записи из локального кэша* var temp: List<CFood>  foods  .*forEach* **{** foodLocal **->** *//Для каждой локальной записи находим записи из сервера с такими же id.* temp = foodsFromServer  .*filter* **{** foodFromServer **->** foodLocal.equals(foodFromServer)  **}** *//Записи с такими же id  // (должна быть максимум 1, но гарантий никто не даст)  // фильтруем по наличию изменений* temp  .*firstOrNull* **{** foodFromServer **->** foodLocal.hasChanges(foodFromServer)  **}** *//Если изменения в полях есть, публикуем текущую запись на сервер.* ?.*let* **{** api.post("foods", foodLocal)  **}** *//Если вообще на сервере записей с таким id нет  //публикуем текущую запись.* if (temp.isEmpty())  api.post("foods", foodLocal)   **}** }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Добавление еды в локальный список. \*  \* @param food - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add(food: CFood) {  this.foods.add(food)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление еды из локального списка. \*  \* @param food - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(food: CFood) {  foods.remove(food)  } } |

|  |
| --- |
| *Листинг 15 –CRepositoryTimeInterval* |
| package com.example.demo.repositories  import com.example.demo.model.CTimeInterval import javafx.collections.FXCollections import javafx.collections.ObservableList import tornadofx.Controller import tornadofx.Rest import tornadofx.toModel  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Репозиторий с запросами к серверу в части списка временного интервала. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CRepositoryTimeInterval : Controller() {  *//Объект для отправки запросов к API сервера.* private val api: Rest by inject()   *//"Кэшированный" список времнного интервала для локальной работы.* private val timeIntervals = FXCollections.observableArrayList<CTimeInterval>()   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка всех доступных объектов на сервере. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getAll(): ObservableList<CTimeInterval> {  if(timeIntervals.*isNotEmpty*())  return timeIntervals  *//Запрашиваем актуальные данные на сервере* val listFromServer = requestAll()  *//Пересохраняем в локальный список,  //в котором дополнительно будем кэшировать изменения в процессе  //редактирования таблицы.* timeIntervals.addAll(listFromServer)  *//Возвращаем ссылку на кэш.* return timeIntervals  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка временного интервала на сервере и обработка возможных проблем. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* private fun requestAll(): ObservableList<CTimeInterval> {  *//Собственно выполнение GET запроса к пути /timeintervals* val response: Rest.Response = api.get("timeintervals")  try {  if (response.ok()) {  *//Конвертация json в список объектов типа CTimeInterval* return response  .list()  .*toModel*()  } else if (response.statusCode == 404)  throw Exception("404")  else  throw Exception("getCustomer returned ${response.statusCode} ${response.reason}")  } catch (e: Exception) {  throw Exception("Отсутствует соединение с сервером.", e)  } finally {  response.consume()  }  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Изменение временного интервала в локальном списке. \*  \* @param timeInterval - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save(timeInterval: CTimeInterval) {  *//Не стали делать отправку, т.к. приняли централизованно все изменения для списка отправлять вмсте.* }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка списка объектов на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll() {  *//Запрос данных с сервера.* val timeIntervalsFromServer = requestAll()   *//Перебираем все временные интервалы с сервера,  //ищем тех, кто отсутствует в локальном списке,  //для них отправляем запрос на удаление.* timeIntervalsFromServer  .*forEach* **{** timeIntervalFromServer **->** *//Для того, чтобы метод contains сработал правильно,  //и чтобы не писать ручной перебор,  //переопределён метод CTimeInterval.equals* if (!timeIntervals.contains(timeIntervalFromServer)) {  api.delete("timeintervals", timeIntervalFromServer)  }  **}** *//Перебираем все записи из локального кэша* var temp: List<CTimeInterval>  timeIntervals  .*forEach* **{** timeIntervalLocal **->** *//Для каждой локальной записи находим записи из сервера с такими же id.* temp = timeIntervalsFromServer  .*filter* **{** timeIntervalFromServer **->** timeIntervalLocal.equals(timeIntervalFromServer)  **}** *// Записи с такими же id  // (должна быть максимум 1, но гарантий никто не даст)  // фильтруем по наличию изменений* temp  .*firstOrNull* **{** timeIntervalFromServer **->** timeIntervalLocal.hasChanges(timeIntervalFromServer)  **}** *//Если изменения в полях есть, публикуем текущую запись на сервер.* ?.*let* **{** api.post("timeintervals", timeIntervalLocal)  **}** *//Если вообще на сервере записей с таким id нет  //публикуем текущую запись.* if (temp.isEmpty())  api.post("timeintervals", timeIntervalLocal)   **}** }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Добавление временного промежутка в локальный список. \*  \* @param timeInterval - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add(timeInterval: CTimeInterval) {  timeIntervals.add(timeInterval)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление врпеменного промежутка из локального списка. \*  \* @param timeInterval - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(timeInterval: CTimeInterval) {  timeIntervals.remove(timeInterval)  } } |

|  |
| --- |
| *Листинг 16 –CServiceDates* |
| package com.example.demo.services  import com.example.demo.model.CDate import com.example.demo.repositories.CRepositoryDate import tornadofx.\* import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Сервис с логикой для работы со списком даты. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CServiceDates : Controller() {  private val repositoryDates : CRepositoryDate by inject()  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка всех доступных объектов на сервере. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getAll()=repositoryDates.getAll()    */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param date - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save(date: CDate)=repositoryDates.save(date)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка всех данных из списка на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll()=repositoryDates.saveAll()   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param date - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add(date:CDate) {  date.id = UUID.randomUUID()  repositoryDates.add(date)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param date - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(date:CDate)=repositoryDates.delete(date) } |

|  |
| --- |
| *Листинг 17 –CServiceFoods* |
| package com.example.demo.services  import com.example.demo.model.CFood import com.example.demo.repositories.CRepositoryFood import tornadofx.\* import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Сервис с логикой для работы со списком еды. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CServiceFoods : Controller() {  private val repositoryFood : CRepositoryFood by inject()  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка всех доступных объектов на сервере. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getAll()=repositoryFood.getAll()   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param food - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save(food:CFood)=repositoryFood.save(food)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка всех данных из списка на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll()=repositoryFood.saveAll()   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param food - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add(food:CFood) {  food.id= UUID.randomUUID()  repositoryFood.add(food)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param food - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(food:CFood)=repositoryFood.delete(food) } |

|  |
| --- |
| *Листинг 18 –CServiceTimeIntervals* |
| package com.example.demo.services  import com.example.demo.model.CTimeInterval import com.example.demo.repositories.CRepositoryTimeInterval import tornadofx.\* import java.util.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Сервис с логикой для работы со списком временного промежутка. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CServiceTimeIntervals : Controller() {  private val repositoryTimeInterval : CRepositoryTimeInterval by inject()  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Запрос списка всех доступных объектов на сервере. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun getAll() = repositoryTimeInterval.getAll()   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param timeInterval - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save(timeInterval: CTimeInterval)=repositoryTimeInterval.save(timeInterval)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка всех данных из списка на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll()=repositoryTimeInterval.saveAll()   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param timeInterval - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add(timeInterval:CTimeInterval) {  timeInterval.id = UUID.randomUUID()  repositoryTimeInterval.add(timeInterval)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка данных объекта на сервер. \*  \* @param timeInterval - объект для отправки. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(timeInterval: CTimeInterval)=repositoryTimeInterval.delete(timeInterval) } |

|  |
| --- |
| *Листинг 19 – CConverterDate* |
| package com.example.demo.util.convertes  import javafx.util.StringConverter import java.time.LocalDate import java.time.Year */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Контроллер вывода даты. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* object CConverterDate : StringConverter<LocalDate>() {  override fun toString(  id : LocalDate?  ) = id?.toString() ?: ""  override fun fromString(  string : String?  ) : LocalDate?  {  if (string == null) return null  return try {  LocalDate.parse(string)  }  catch (e : Exception)  {  null  }  } } |

|  |
| --- |
| *Листинг 20 – CConverterTime* |
| package com.example.demo.util.convertes  import javafx.util.StringConverter import java.time.LocalTime */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Контроллер вывода времени. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* object CConverterTime : StringConverter<LocalTime>() {  override fun toString(  id : LocalTime?  ) = id?.toString() ?: ""  override fun fromString(  string : String?  ) : LocalTime?  {  if (string == null) return null  return try {  LocalTime.parse(string)  }  catch (e : Exception)  {  null  }  } } |

|  |
| --- |
| *Листинг 21 – CViewDatesTable* |
| package com.example.demo.view  import com.example.demo.model.CDate import com.example.demo.util.convertes.CConverterDate import com.example.demo.viewmodels.CViewModelDates import com.example.demo.viewmodels.CViewModelDatesList import javafx.scene.layout.BorderPane import tornadofx.\* */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Страница со списком дат. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewDatesTable : View("Даты") {  *//Корневой элемент формы - элемент с 5ю областями (Верх, Низ, Левая, Правая, Центр)* override val root = BorderPane()  *//Модель представления для списка в целом.* private val viewModelList : CViewModelDatesList by inject()  *//Модель представления для одного элемента,  //отображаемого в правой боковой панели редактирвоания.* val viewModelItem = CViewModelDates(CDate())   *//Таблица дат.  //В качестве параметра передаём список дат из модели представления.* val table = *tableview*(viewModelList.dates)  **{** *isEditable* = true  *column*(messages["ID"],CDate::propertyId)  *column*(messages["Date"], CDate::propertyDate).*makeEditable*(CConverterDate)  *column*(messages["day"], CDate::propertyDay).*makeEditable*()   *//Изменение выделения предаём в модель представления правой формы.* viewModelItem.*rebindOnChange*(this) **{** selectedItem **->** item = selectedItem ?: CDate()  **}** *//Изменение выделения передаём в модель представления всего списка.  onSelectionChange* **{** viewModelList.onSelectionChange(this.*selectedItem*)  **}   }** init {  *//Верхняя часть окна* root.*top***{** *//Делится на 2 строки  vbox* **{** *//Самая верхняя строка - меню.  menubar* **{** *menu*("Окна") **{** *item*("Блюда(Еда)").*action***{** *//Пример замены содержимого окна на другую View* replaceWith<CViewFoodTable>()  **}** *item*("Временной промежуток").*action***{** *//Пример замены содержимого окна на другую View* replaceWith<CViewTimeIntervalsTable>()  **}  }  }** *//Вторая строчка сверху - кнопки для работы со списком объектов.  hbox* **{** *button*(messages["Add"]) **{** *action***{** viewModelList.add()  **}  }** *button*(messages["Delete"]) **{** *//Активность кнопки удалить зависит от поля в модели представления.  enableWhen*(viewModelList.elementSelected)  *action***{** viewModelList.delete(table.*selectedItem*)  **}  }** *button*(messages["Save"]) **{** *action***{** viewModelList.saveAll()  **}  }  }  }  }** root.*center***{** *//В центральной части располагается таблица.* this += table  **}** root.*right* **{** *//В правой части располагается форма для редактирования одного объекта.  form***{** *fieldset*("Редактировать данные даты") **{** *field*("Дата") **{** *datepicker*(viewModelItem.date)  **}** *//field("День недели") {  // textfield(viewModelItem.day)  //}  hbox***{** *button*(messages["Save"]) **{** *enableWhen*(viewModelItem.dirty)  *action***{** viewModelItem.save()  **}  }** *button*(messages["Cancel"]).*action* **{** viewModelItem.rollback()  **}  }  }  }  }** } } |

|  |
| --- |
| *Листинг 22 – CViewFoodTable* |
| package com.example.demo.view  import com.example.demo.model.CFood import com.example.demo.viewmodels.CViewModelFoods import com.example.demo.viewmodels.CViewModelFoodsList import javafx.scene.layout.BorderPane import tornadofx.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Страница со списком еды. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewFoodTable : View("Блюда") {  *//Корневой элемент формы - элемент с 5ю областями (Верх, Низ, Левая, Правая, Центр)* override val root = BorderPane()   *//Модель представления для списка в целом.* private val viewModelList: CViewModelFoodsList by inject()   *//Модель представления для одного элемента,  //отображаемого в правой боковой панели редактирования.* val viewModelItem = CViewModelFoods(CFood())   *//Таблица еды.  //В качестве параметра передаём список еды из модели представления.* private val table = *tableview*(viewModelList.foods)  **{** *isEditable* = true  *column*(messages["ID"], CFood::propertyId)  *column*(messages["Name"], CFood::propertyName)  *column*(messages["Kal"], CFood::propertyKal)   *column*(messages["Date"], CFood::propertyDate) **{** *cellFormat* **{** *text* = ""  if (**it** != null) {  var l = viewModelList.dates.*map* **{** it1 **->** it1.*value* **}**.*find* **{** i **->** i.id == it.id **}** if (l != null) {  *text* = l.date.toString()  }  }  **}  }** *column*(messages["TimeInterval"], CFood::propertyTimeInterval) **{** *cellFormat* **{** *text* = ""  if (**it** != null) {  var l = viewModelList.timeinterval.*map* **{** it1 **->** it1.*value* **}**.*find* **{** i **->** i.id == **it**.id **}** if (l != null) {  *text* = l.name  }  }  **}  }** *//Изменение выделения предаём в модель представления правой формы.* viewModelItem.*rebindOnChange*(this) **{** selectedItem **->** item = selectedItem ?: CFood()  **}** *//Изменение выделения передаём в модель представления всего списка.  onSelectionChange* **{** viewModelList.onSelectionChange(this.*selectedItem*)  **}  }** init {  *//Верхняя часть окна* root.*top* **{** *//Делится на 2 строки  vbox* **{** *//Самая верхняя строка - меню.  menubar* **{** *menu*("Окна") **{** *item*(messages["Date"]).*action* **{** *//Пример замены содержимого окна на другую View* replaceWith<CViewDatesTable>()  **}** *item*(messages["TimeInterval"]).*action* **{** *//Пример замены содержимого окна на другую View* replaceWith<CViewTimeIntervalsTable>()  **}  }  }** *//Вторая строчка сверху - кнопки для работы со списком объектов.  hbox* **{** *button*(messages["Add"]) **{** *action* **{** viewModelList.add()   **}  }** *button*(messages["Delete"]) **{** *//Активность кнопки удалить зависит от поля в модели представления.  enableWhen*(viewModelList.elementSelected)  *action* **{** viewModelList.delete(table.*selectedItem*)  **}  }** *button*(messages["Save"]) **{** *action* **{** viewModelList.saveAll()  **}  }  }  }  }** root.*center* **{** *//В центральной части располагается таблица.* this += table  **}** root.*right* **{** *//В правой части располагается форма для редактирования одного объекта.  form* **{** *fieldset*("Редактировать данные еды") **{** *field*(messages["Name"]) **{** *textfield*(viewModelItem.name)  **}** *field*(messages["Kal"]) **{** *textfield*(viewModelItem.kal) *//textfield(Double.toString(viewModelItem.kal))* **}** *field*(messages["TimeInterval"]) **{** *combobox*(viewModelItem.timeinterval) **{** *items* = viewModelList.timeinterval  setOnAction **{** viewModelItem.timeinterval.*value*.*value* = this.*selectedItem*?.*value* viewModelItem.markDirty(viewModelItem.timeinterval)  **}** *cellFormat* **{** *text* = **it**.*value*?.name **}  }  }** *field*(messages["Date"]) **{** *combobox*(viewModelItem.date) **{** *items* = viewModelList.dates  setOnAction **{** viewModelItem.date.*value*.*value* = this.*selectedItem*?.*value* viewModelItem.markDirty(viewModelItem.date)  **}** *cellFormat***{** *text* = **it**.*value*?.date.*toString*() **}  }  }** *hbox* **{** *button*(messages["Save"]) **{** *enableWhen*(viewModelItem.dirty)  *action* **{** table.*selectedItem*?.timeinterval = viewModelItem.timeinterval.*value*.*value* table.*selectedItem*?.date = viewModelItem.date.*value*.*value* viewModelItem.save()   **}  }** *button*(messages["Cancel"]).*action* **{** viewModelItem.rollback()  **}  }  }** setMaxSize(500.0, 50.0)  **}  }** } } |

|  |
| --- |
| *Листинг 23 – CViewTimeIntervalsTable* |
| package com.example.demo.view  import com.example.demo.model.CTimeInterval import com.example.demo.util.convertes.CConverterTime import com.example.demo.viewmodels.CViewModelTimeIntervals import com.example.demo.viewmodels.CViewModelTimeIntervalsList import javafx.scene.layout.BorderPane import tornadofx.\* */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Страница со списком временного промежутка. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewTimeIntervalsTable : View("Временные промежутки") {  *//Корневой элемент формы - элемент с 5ю областями (Верх, Низ, Левая, Правая, Центр)* override val root = BorderPane()  *//Модель представления для списка в целом.* private val viewModelList : CViewModelTimeIntervalsList by inject()  *//Модель представления для одного элемента,  //отображаемого в правой боковой панели редактирвоания.* val viewModelItem = CViewModelTimeIntervals(CTimeInterval())   *//Таблица временных промежутков.  //В качестве параметра передаём список временных промежутков из модели представления.* val table = *tableview*(viewModelList.timeIntervals)  **{** *isEditable* = true  *column*(messages["ID"], CTimeInterval::propertyId)  *column*(messages["Name"],CTimeInterval::propertyName)  *column*(messages["VrN"],CTimeInterval::propertyVrN).*makeEditable*(CConverterTime)  *column*(messages["VrK"],CTimeInterval::propertyVrK).*makeEditable*(CConverterTime)   *//Изменение выделения предаём в модель представления правой формы.* viewModelItem.*rebindOnChange*(this) **{** selectedItem **->** item = selectedItem ?: CTimeInterval()  **}** *//Изменение выделения передаём в модель представления всего списка.  onSelectionChange* **{** viewModelList.onSelectionChange(this.*selectedItem*)  **}  }** init {  *//Верхняя часть окна* root.*top***{** *//Делится на 2 строки  vbox* **{** *//Самая верхняя строка - меню.  menubar* **{** *menu*("Окна") **{** *item*("Блюда(Еда)").*action***{** *//Пример замены содержимого окна на другую View* replaceWith<CViewFoodTable>()  **}** *item*("Даты").*action***{** *//Пример замены содержимого окна на другую View* replaceWith<CViewDatesTable>()  **}  }  }** *//Вторая строчка сверху - кнопки для работы со списком объектов.  hbox* **{** *button*(messages["Add"]) **{** *action***{** viewModelList.add()  **}  }** *button*(messages["Delete"]) **{** *//Активность кнопки удалить зависит от поля в модели представления.  enableWhen*(viewModelList.elementSelected)  *action***{** viewModelList.delete(table.*selectedItem*)  **}  }** *button*(messages["Save"]) **{** *action***{** viewModelList.saveAll()  **}  }  }  }  }** root.*center***{** *//В центральной части располагается таблица.* this += table  **}** root.*right* **{** *//В правой части располагается форма для редактирования одного объекта.  form***{** *fieldset*("Редактировать данные временного промежутка") **{** *field*(messages["Name"]) **{** *textfield*(viewModelItem.name)  **}** *field*(messages["VrN"]) **{** *textfield*(viewModelItem.VrN, CConverterTime)  **}** *field*(messages["VrK"]) **{** *textfield*(viewModelItem.VrK, CConverterTime)  **}** *hbox***{** *button*(messages["Save"]) **{** *enableWhen*(viewModelItem.dirty)  *action***{** viewModelItem.save()  **}  }** *button*(messages["Cancel"]).*action* **{** viewModelItem.rollback()  **}  }  }  }  }** } } |

|  |
| --- |
| *Листинг 24 – CViewModelDates* |
| package com.example.demo.viewmodels  import com.example.demo.model.CDate import com.example.demo.services.CServiceDates import tornadofx.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Модель представления для одной даты. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewModelDates(date:CDate): ItemViewModel<CDate>(date) {  private val serviceDates : CServiceDates by inject()   val id = bind(CDate::propertyId)  val date = bind(CDate::propertyDate)  val day = bind(CDate::propertyDay)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Сохранение данных из полей формы в объект в оперативной памяти. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save()  {  this.commit() *//Сохранение данных из полей на форме в сущность  //Передача данных на сервер (изменили на кэш только)* serviceDates.save(this.item)  } } |

|  |
| --- |
| *Листинг 25 – CViewModelDatesList* |
| package com.example.demo.viewmodels  import com.example.demo.model.CDate import com.example.demo.services.CServiceDates import javafx.beans.property.SimpleBooleanProperty import tornadofx.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Модель представления для списка дат. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewModelDatesList : Controller() {  *//Сервис с логикой для работы со списком дат.* private val serviceDates : CServiceDates by inject()   *//Список даты с возможность отслеживания изменений.* val dates = serviceDates.getAll()   *//Факт наличия выделения в таблице.  //От него зависит активность некоторых кнопок.* val elementSelected = SimpleBooleanProperty(false)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Добавление пустой записи в таблицу. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add()  {  serviceDates.add(CDate()) *//Передача данных на сервер* }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление записи из списка. \*  \* @param item - элемент для удаления. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(item:CDate?)  {  *//Если элемент не указан - ничего не делаем.* item ?: return  serviceDates.delete(item) *//Передача данных на сервер* }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Обработка изменения выделенных элементов в списке. \*  \* @param selectedItem - выделенный элемент. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun onSelectionChange(selectedItem:CDate?)  {  elementSelected.set(selectedItem != null)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка всех текущих данных из списка на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll()  {  serviceDates.saveAll() *//Передача команды на отправку данных на сервер* } } |

|  |
| --- |
| *Листинг 26 – CViewModelFoods* |
| package com.example.demo.viewmodels   import com.example.demo.model.CFood import com.example.demo.services.CServiceFoods import tornadofx.ItemViewModel  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Модель представления для одной еды. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewModelFoods(food:CFood) : ItemViewModel<CFood>(food) {  private val serviceFoods : CServiceFoods by inject()   val id = bind(CFood::propertyId)  val name = bind(CFood::propertyName)  val kal = bind(CFood::propertyKal)  val date = bind(CFood::propertyDate)  val timeinterval = bind(CFood::propertyTimeInterval)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Сохранение данных из полей формы в объект в оперативной памяти. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save()  {  this.commit() *//Сохранение данных из полей на форме в сущность   //Передача данных на сервер (изменили на кэш только)* serviceFoods.save(this.item)  } } |

|  |
| --- |
| *Листинг 27 –* *CViewModelFoodsList* |
| package com.example.demo.viewmodels  import com.example.demo.model.CFood import javafx.beans.property.SimpleBooleanProperty import javafx.beans.property.SimpleObjectProperty import com.example.demo.services.CServiceFoods import com.example.demo.services.CServiceDates import com.example.demo.services.CServiceTimeIntervals import tornadofx.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Модель представления для списка еды. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewModelFoodsList : Controller() {  *//Сервис с логикой для работы со списком рейсов.* private val serviceFoods : CServiceFoods by inject()  private val serviceDates : CServiceDates by inject()  private val serviceTimeIntervals : CServiceTimeIntervals by inject()   *//Список еды с возможность отслеживания изменений.* val foods = serviceFoods.getAll()  val dates = serviceDates.getAll().*map* **{** SimpleObjectProperty(**it**) **}**.*toObservable*()  val timeinterval = serviceTimeIntervals.getAll().*map* **{** SimpleObjectProperty(**it**) **}**.*toObservable*()   *//Факт наличия выделения в таблице.  //От него зависит активность некоторых кнопок.* val elementSelected = SimpleBooleanProperty(false)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Добавление пустой записи в таблицу. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add()  {  serviceFoods.add(CFood()) *//Передача данных на сервер* }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление записи из списка. \*  \* @param item - элемент для удаления. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(item:CFood?)  {  *//Если элемент не указан - ничего не делаем.* item ?: return  serviceFoods.delete(item) *//Передача данных на сервер* }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Обработка изменения выделенных элементов в списке. \*  \* @param selectedItem - выделенный элемент. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun onSelectionChange(selectedItem:CFood?)  {  elementSelected.set(selectedItem != null)  }  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка всех текущих данных из списка на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll()  {  serviceFoods.saveAll() *//Передача команды на отправку данных на сервер* } } |

|  |
| --- |
| *Листинг 28 – CViewModelTimeIntervals* |
| package com.example.demo.viewmodels  import com.example.demo.model.CTimeInterval import com.example.demo.services.CServiceTimeIntervals import tornadofx.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Модель представления для одного самолёта. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewModelTimeIntervals(timeInterval:CTimeInterval):ItemViewModel<CTimeInterval>(timeInterval) {  private val serviceTimeIntervals : CServiceTimeIntervals by inject()   val id = bind(CTimeInterval::propertyId)  val name = bind(CTimeInterval::propertyName)  val VrN = bind(CTimeInterval::propertyVrN)  val VrK = bind(CTimeInterval::propertyVrK)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Сохранение данных из полей формы в объект в оперативной памяти. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun save()  {  this.commit() *//Сохранение данных из полей на форме в сущность  //Передача данных на сервер (изменили на кэш только)* serviceTimeIntervals.save(this.item)  } } |

|  |
| --- |
| *Листинг 29 – CViewModelTimeIntervalsList* |
| package com.example.demo.viewmodels  import com.example.demo.model.CTimeInterval import com.example.demo.services.CServiceTimeIntervals import javafx.beans.property.SimpleBooleanProperty import tornadofx.\*  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Модель представления для списка самолётов. \*  \* @author Макарова П.Ф. 26.03.2023 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* class CViewModelTimeIntervalsList : Controller() {  *//Сервис с логикой для работы со списком временных промежутков.* private val serviceTimeIntervals : CServiceTimeIntervals by inject()   *//Список самолётов с возможность отслеживания изменений.* val timeIntervals = serviceTimeIntervals.getAll()   *//Факт наличия выделения в таблице.  //От него зависит активность некоторых кнопок.* val elementSelected = SimpleBooleanProperty(false)   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Добавление пустой записи в таблицу. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun add()  {  serviceTimeIntervals.add(CTimeInterval()) *//Передача данных на сервер* }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Удаление записи из списка. \*  \* @param item - элемент для удаления. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun delete(item:CTimeInterval?)  {  *//Если элемент не указан - ничего не делаем.* item ?: return  serviceTimeIntervals.delete(item) *//Передача данных на сервер* }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Обработка изменения выделенных элементов в списке. \*  \* @param selectedItem - выделенный элемент. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun onSelectionChange(selectedItem:CTimeInterval?)  {  elementSelected.set(selectedItem != null)  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Отправка всех текущих данных из списка на сервер. \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* fun saveAll()  {  serviceTimeIntervals.saveAll() *//Передача команды на отправку данных на сервер* } } |