МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий

(наименование института полностью)

Кафедра «Прикладная математика и информатика»

(наименование кафедры)

02.03.03 Математическое обеспечением и администрирование информационных систем

(код и наименование направления подготовки, специальности)

Технология программирование

(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Архитектура подсистемы импорта данных в систему публикации расписания учебных занятий ФГБОУ ВО "ТГУ"

Студент М.А. Михайлов _____
(И.О. Фамилия) (личная подпись)

Руководитель А.Б. Кузьмичев _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____2018г.

Тольятти 2018

АННОТАЦИЯ

Тема этой бакалаврской работы - разработка подсистемы импорта данных для публикации системы расписания.

Вопрос об организации учебного процесса всегда был одной из важнейших задач для высших учебных заведений. Одной из наиболее важных частей этой работы является информация о графике учебных занятий. Лучшим решением этой проблемы является система публикации расписания. Подсистема импорта является частью этой системы. Это будет модуль для получения данных из корпоративной базы данных учебного заведения.

Это приложение будет получать данные по расписанию благодаря JPQL.

Приложение должно хранить данные о группе студентов, собирать. формат и запись данных. Чтобы реализовать все функции проецируемого приложения, мы будем использовать язык объектно-ориентированного программирования Java. Платформа реализации приложения будет IDE NetBeans Studio. Исследование показало, что выбор языка Java и IDE NetBeans наиболее удобен для реализации такого приложения. Основной задачей, решаемой приложением, будет анализ данных из базы данных Oracle и представление данных в форме, необходимой для нашей базы данных. Благодаря всем этим условиям разработанное приложение удовлетворит все требования к нему.

В заключении обобщается проделанная работа и формируются выводы по

Работа содержит 59 страниц машинописного текста, 12 таблиц, 17 рисунков, 1 приложение. Для написания использованы 29 источников.

ABSTRACT

The theme of this bachelor's work is the development of the data import-export subsystem for the publication of the timetable system.
The question of organizing the educational process has always been one of the most important tasks for higher education institutions. One of the most important parts of this work is information on the schedule of training sessions. The best solution to this problem is the timetable publication system. The import-export subsystem is part of this system. This will be a module for obtaining data from the corporate database of the educational institution.
This application will receive data on a schedule thanks to JPQL.
The application should store data about a group of students, collect. format and data recording. To implement all the functions of the projected application, we will use the language of object-oriented Java programming. The platform for implementing the application will be the IDE NetBeans Studio. The study showed that the choice of the Java language and the NetBeans IDE is most convenient for implementing such an application. The main task solved by the application will be the analysis of data from the Oracle database and the presentation of data in the form required for our database. Thanks to all these conditions, the developed application will satisfy all requirements to it.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С течением времени прогресс и информационные технологии в частности стремительно развивается. Соответственно должны развиваться и корпоративные технологии, чтобы не потерять актуальность и востребованность. Особенно информационные системы публикации корпоративных данных, таких как расписание учебных занятий в ВУЗах.

Актуальность данной работы заключается в том, что автоматизации процесса публикации расписания учебных занятий значительно сокращает время, необходимое на публикацию данных, и позволяет предоставлять информацию в требуемой форме на практически любое устройство (телефон, планшет, персональный компьютер).

Новизна данной работы заключается в том, что ранее в ТГУ не существовало единой системы публикации расписания и, реализуя в этой дипломной работе модуль импорта данных, она будет спроектирована, разработана и внедрена в ФГБОУ ВО «ТГУ».

Объектом работы модуль импорта данных из базы данных ФГБОУ ВО «ТГУ».

Предметом работы является технология разработки модуля импорта данных из базы данных ФГБОУ ВО «ТГУ»

Целью выпускной квалификационной работы является разработка модуль импорта данных из базы данных ФГБОУ ВО «ТГУ». Для достижения цели необходимо выполнить следующие задачи:

- проанализировать текущую систему публикации расписания учебных занятий ФГБОУ ВО «ТГУ»;
- составить список требований к проектируемому приложению для модуль импорта данных из базы данных ФГБОУ ВО «ТГУ»;

– спроектировать модуль импорта данных из базы данных ФГБОУ ВО «ТГУ»;

– разработать модуль импорта данных из базы данных ВУЗа на основе полученных знаний и проектируемой модели серверного приложения.

– интегрировать модуль импорта данных из базы данных в единую систему публикации расписания ФГБОУ ВО «ТГУ»;

– поддерживать жизненный цикл разработанного и интегрированного модуля импорта данных из базы данных ВУЗа.

1 АНАЛИЗ СУЩЕСТВУЮЩЕЙ СИСТЕМЫ ПУБЛИКАЦИИ РАСПИСАНИЯ

1.1. Анализ ФГБОУ ВО «Тольяттинский государственный университет» с точки зрения публикации данных

Тольяттинский государственный университет – высшее учебное заведение, созданное в 2001 году решением правительства России на базе Тольяттинского филиала Самарского государственного педагогического университета и Тольяттинского политехнического института.

ТГУ – это старейший и крупнейший государственный вуз города Тольятти, который предоставляет подготовку по естественно-научному, техническому и гуманитарно-педагогическому направлениям, а также один из немногих российских вузов, которые имеют военный центр и учебную военную кафедру[10].

В интернете существует сайт университета, который находится по адресу: http://www.tltsu.ru. Главный блок-корпус ВУЗа расположен по адресу: 445020, г. Тольятти, ул. Белорусская, 14. Тольяттинский государственный университет объединяет институты:

1. ИВО - Институт военного обучения.
2. ИФЭиУ - Институт финансов, экономики и управления.
3. ИФКиС - Институт физической культуры и спорта.
4. ИхиИЭ - Институт химии и инженерной экологии.
5. ИП - Институт права.
6. АСИ - Архитектурно-строительный институт.
7. ГУМПИ - Гуманитарно-педагогический институт.

8. ИиИДПИ - Институт изобразительного и декоративно-прикладного искусства.
9. ИнМаш - Институт машиностроения.
10. ИМФиИТ - Институт математики, физики и информационных технологий.
11. ИэиЭ - Институт энергетики и электротехники.

В структуру ТГУ входят 47 кафедр и работают более 2200 преподавателей и сотрудников, среди которых 100 докторов и более 460 кандидатов наук. В среднем во ФГБОУ ВО «ТГУ» учится 12 000 студентов.

С точки зрения организации учебного процесса каждому ВУЗу необходимо публиковать расписание учебных занятий, и наиболее эффективно эту работу будет выполнять система публикации расписания, в состав которой войдет модуль импорта данных предприятия из базы данных ФГБОУ ВО «ТГУ». Роль этого модуля заключается в организации подключения к базе данных организации, отправке запроса данных, необходимых для публикации расписания, получение самих данных, их преобразования в вид структуры базы данных системы публикации и загрузка преобразованных данных в систему.

Расписание учебных занятий во ФГБОУ ВО «ТГУ» создается отделом учебно-методического управления и диспетчерской службой, а затем утверждается заместителем ректора по развитию учебного процесса. Во время составлении расписания занятий используется технология «Галактика Расписание учебных занятий», которая является дополнением ERP-системы «Галактика», позволяющая исключать ошибки, связанные с человеческим фактором, во время создания расписания. Эта часть процесса составления расписания слишком сложна и велика, что в рамках данного проектирования рассматриваться не будет. Таким образом, данные, получаемые вследствие работы системы «Галактика Расписание учебных занятий» выступают в роли

входных данных. Для публикации расписания учебных занятий, в едином информационном пространстве ФГБОУ ВО «ТГУ», экспортируются в MS Excel сотрудником диспетчерской службы. Расписание составляется и публикуется на 2 недели вперед на сайте университета.

Исходя из всего вышесказанного можно сделать вывод, что подобная система публикации расписания не является удобной и даёт возможность для появления альтернатив. В данной дипломной работе будет рассмотрена технология импорта данных, которая является частью организации учебного процесса ФГБОУ ВО «ТГУ», и на основании полученной технологии затем должна быть создана система публикации расписания занятий в ФГБОУ ВО «ТГУ».

1.2. Анализ технологии публикации учебных занятий в ФГБОУ ВО «ТГУ»

Чтобы проанализировать текущую систему публикации расписания, рассмотрим подробно процесс получения данных расписания.

Существует сайт университета, на котором публикуются табличный документы в формате xls с данным о расписании занятий студентов. Вид документа представляет собой большую сложную таблицу, в которой размещены данные о расписании занятий всего института по группам.

На рисунке 1.2.1 представлен алгоритм действий для просмотра расписания занятий. Например, для студента алгоритм будет:

1. В адресной строке браузера набрать URL-адрес сайта ФГБОУ ВО «ТГУ» https://www.tltsu.ru.
2. На главной странице сайта перейти по ссылке «Студентам».
3. На странице «Студентам» перейти по ссылке «Расписание занятий».
4. На странице «Расписание занятий» выбрать необходимую неделю и свой институт.

5. Скачать xls файл, в название которого входит название института и номер недели, и открыть его с помощью предустановленного приложения для просмотра табличных файлов.
6. Найти необходимый курс обучения и необходимую учебную группу.
7. Просмотреть расписание.

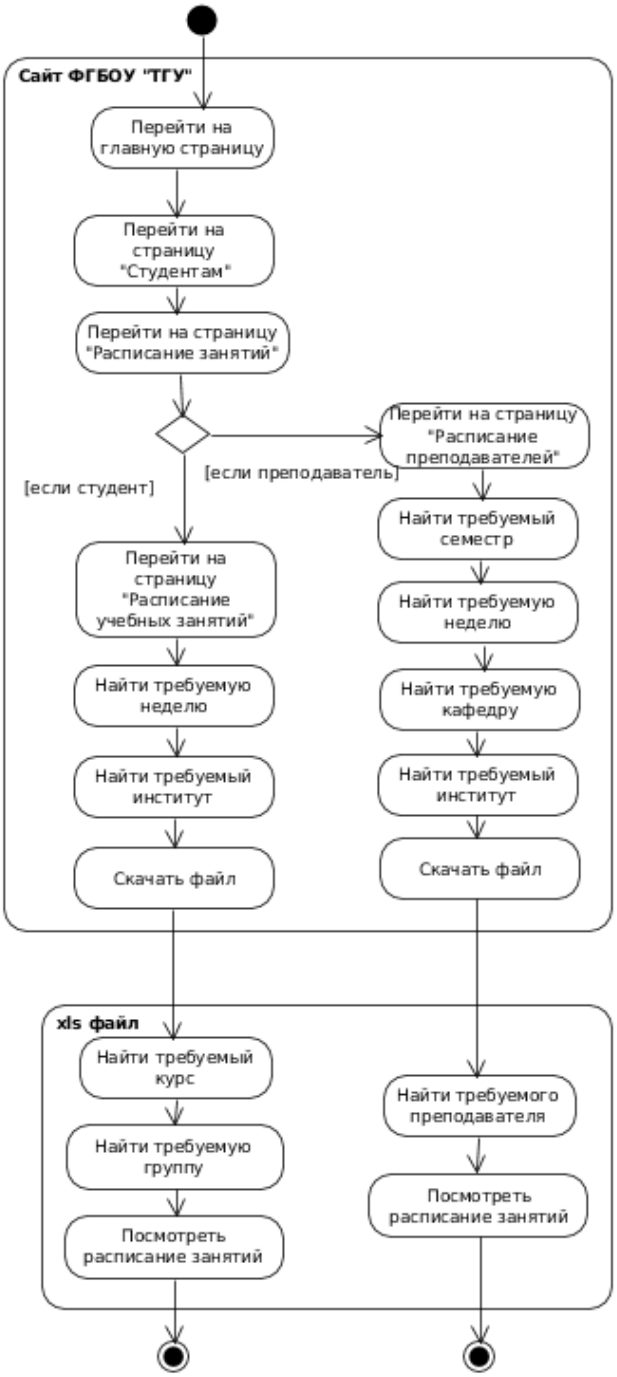


Рисунок 1.2.1 — Диаграммы деятельности получения расписания.

Как видно из рисунка 1.2.1 для получения расписания занятий, студенту необходимо совершить 10 действий, чтобы посмотреть своё расписание занятий.

Как видно из текущей модели получения расписания, в ней есть ряд минусов:

– для просмотра расписания нужно выкачивать xls файлы;

– чтобы найти требуемое расписание нужно выполнить большое количество шагов

– не на каждом устройстве есть программы для отображения формата xls;

– в файле с расписанием неудобно ориентироваться и быстро находить нужную информацию;

нет возможности посмотреть свободные аудитории.

Проведя 20 замеров времени с помощью 3 разных людях и на 3 устройствах, для того чтобы просмотреть расписание занятий своей группы в среднем у человека тратится 40 секунд. А время на загрузку самого файла занимает менее 1 секунды.

Таким образом, резюмируя всё вышесказанное можно сделать вывод, что такая рядовая задача, как просмотр расписания учебных занятий во ФГБОУ ВО «ТГУ» отнимает сравнительно много времени и является процессом сложным и неудобным для выполнения.

1.3 Выработка требований к проектируемой подсистеме

Проектируемая технология публикации расписания будет представлять из себя подсистему импорта данных, которая будет на основе входных данных конвертировать их в необходимый для отдельной базы данных вид и сохранять их в эту базу.

Для того чтобы разработать требования возьмем систему классификации требований FURPS+ [1].

Алгоритм получения данных расписания в подсистему импорта в разрабатываемой модели:

1. Подключение к базе данных ФГБОУ ВО «ТГУ»;

2. Загрузка данных на сервер подсистемы импорта;
3. Обработка загруженных данных в пригодный для подсистемы публикации расписания вид;
4. Сохранение данных обработанных;
5. Выдача обработанных данных по запросу.

Очевидные плюсы этой модели:

– возможность реализовать удобный для публикации вывод данных;

– независимость от неполадок и ошибок сервера на стороне ФГБОУ ВО «ТГУ».

Где в таблице 3.1.1 предоставленны все требования и их описание

Таблица 3.1.1 – Требования к системе

№	Требование	Статус	Полезность	Обоснование выбора
Funcrionality – Функциональные требования				
1	Добавление, отображение, изменение, удаление списка преподавателей, групп, аудиторий, кафедр, институтов, списка дисциплин, списка занятий и сведения по каждому из них.	Одобренные	Критичное	В подсистеме должно быть реализовано хранения информации о преподавателях, группах, аудиториях, кафедрах, списках дисциплин, списках занятий и институтах.

Продолжение таблицы 3.1.1

№	Требование	Статус	Полезность	Обоснование выбора

№	Требование	Статус	Полезность	Обоснование выбора
	Формирование и вывод API для модулей отображения данных	Одобренные	Критичное	В подсистеме должен быть реализован вывод API для модулей отображения данных
Usability – Требования к удобству использования				
16	Обеспечение разработчиков панелью администратора	Предложенные	Полезное	Подсистема должна иметь инструмент для помощи разработчикам.
Reliability – Требования к надежности				
17	Реализованная программа должна корректно реагировать на неверные данные и ошибки .	Предложенные	Важное	Подсистема должна уметь анализировать данные на наличие ошибок и обрабатывать их.
Performance – Требования к производительности				
20	Транзакция при работе с одной записью не дольше 1 секунды.	Одобренные	Критичное	Подсистема должна с минимальной по времени задержкой отвечать пользователю на его действие.
21	Транзакция (с условиями) с отображением не дольше 1 секунды	Одобренные	Критичное	
Supportability - Требование к поддержке				

Продолжение таблицы 3.1.1

№	Требование	Статус	Полезность	Обоснование выбора
22	При реализации подсистемы, использовать контроль версий	Одобренные	Критичное	Подсистема должна поддерживаться системой контроля версий для простой работой над проектом нескольким людям одновременно.
23	Оформлять документацию по одной из популярных систем автоматической генерации проектной документации	Одобренные	Критичное	Подсистема должна иметь документацию для быстрого ввода в курс дела новых людей в проекте.
Implementation requirements – Требования к реализации				
24	Подсистема должна быть легко переносимой	Предложенные	Важное	Подсистема должна быть реализованная в таком виде, чтобы ее можно было легко переносить.
25	В подсистеме должно быть реализовано аудит событий.	Предложенные	Важное	В подсистеме должна быть реализована потребность в журналировании действий в системе.
Interface requirements – Требование к интерфейсу				

Продолжение таблицы 3.1.1

№	Требование	Статус	Полезность	Обоснование выбора
26	Подсистема должна формировать вывод данных в json формате	Одобренные	Критичное	Сервисы, на которых ориентирована разрабатываемая подсистема, в основном используют json формат, т.к. он является самым легковесным, в качестве переноса данных.
Physical requirements – Физические требования				
27	Подсистема должна быть разработанна под компьютеры с amd64 или i386 архитектурой процессора	Одобренные	Критичное	Самыми распространенными архитектурами являются amd64 или i386, где i386 является распространенной, но не актуальной.

На основе анализа было выявлено, что необходимо реализовать 15 функциональных требований и 12 нефункциональных требований представленных в таблице. Из функциональных требований самыми важными являются обновление расписания. Из не функциональных самым приоритетным является — время отзыва приложения на действие пользователя не более 1 секунды.

Разрабатываемая модель будет представлять собой приложение с реализацией импорта данных из базы данных ВУЗа.

2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПОДСИСТЕМЫ ИМПОРТА РАСПИСАНИЯ

2.1. Разработка новой технологии публикации

Диаграмма вариантов использования (ДВИ) используется в данной выпускной квалификационной работе для того, чтобы смоделировать функциональные требования к проектируемой системе.

Так как пользователями являются разработчики слоя представления, то варианты использования данной подсистемы для них только один — получение сведения о расписании.

Актеры:

1. Разработчик — это любой человек, у которого будет доступ к получению информации из проектируемой подсистемы.
2. Таймер — это модуль в проектируемой системе, который будет начинать работать в определенное время.

Прецеденты новой технологии описаны в таблице 2.1.1, а описание основных прецедентов в таблицах 2.1.2-2.2.5.

Таблица 2.1.1 — Описание прецедентов

ID	Название прецедента	Описание
1	Получение списка институтов	функция, которая будет высылать на запрос разработчика список институтов преподавателей,аудиторий и групп
2	Получение списка кафедр	функция, которая будет высылать на запрос разработчика список кафедр аудиторий и преподавателей
3	Получение списка аудиторий	функция, которая будет высылать на запрос разработчика список аудиторий

Продолжение таблицы 2.1.1

ID	Название прецедента	Описание
4	Получение списка преподавателей	функция, которая будет высылать на запрос разработчика список преподавателей
5	Получение списка групп	функция, которая будет высылать на запрос разработчика список групп
6	Получение расписание	функция, которая будет высылать на запрос разработчика расписание аудиторий, преподавателей и групп
7	Обновление компонентов	функция, которая будет обновлять информацию о преподавателях, группах, аудиториях, институтов, кафедр и дисциплин
8	Обновление занятий	функция, которая будет обновлять информацию о занятиях
9	Обновление активности	функция, которая будет проверять на присутствие преподавателей, групп, аудиторий, институтов, и кафедр за последнее время в системе

Таблица 2.1.2 – Описание прецедента «Получение расписание групп»

Прецедент: Получение расписание
ID: 6
Краткое описание: Реализация функции получения расписание аудиторий, преподавателей и групп.
Главные актеры: 1. Разработчик
Второстепенные актеры:
Предусловие:

Прецедент: Получение расписание
Прецедент начинается по инициативе разработчика

Продолжение таблицы 2.1.2

Прецедент: Получение расписание
Основной поток: 1. Разработчик отправляет запрос на сервер 2. Система копирует данные из таблицы занятий базы данных 3. Система формирует json файл из собранной информации 4. Система отправляет разработчику json файл
Постусловие: Система отправила разработчику json файл
Альтернативные потоки: Нет

Таблица 2.1.3 – Описание прецедента «Скачивание и обновление занятий»

Прецедент: Скачивание и обновление занятий
ID: 7
Краткое описание: Реализация функции, которая будет обновлять информацию о занятиях.
Главные актеры: 1. Таймер
Второстепенные актеры:
Предусловие: Прецедент начинается по заданному времени в Таймере
Основной поток: 1. Система подключается к базе данных ТГУ 2. Система копирует данные из таблицы занятий базы данных ТГУ 3. Система обновляет построчно данные в аналогичной таблице(сущности) базе данных подсистемы
Постусловие: Система обновила таблицу занятий
Альтернативные потоки: Нет

Таблица 2.1.3 – Описание прецедента «Скачивание и обновление компонентов»

Прецедент: Скачивание и обновление компонентов
ID: 8
Краткое описание: Реализация функции, которая будет обновлять информацию о преподавателях, группах, аудиториях, институтов, кафедр и дисциплин.
Главные актеры: 1. Таймер
Второстепенные актеры:
Предусловие: Прецедент начинается по заданному времени в Таймере
Основной поток: 1. Система подключается к базе данных ТГУ 2. Система копирует данные из таблицы институтов базы данных ТГУ 3. Система обновляет построчно данные в аналогичной таблице(сущности) базе данных подсистемы 4. Система копирует данные из таблицы кафедр базы данных ТГУ 5. Система обновляет построчно данные в аналогичной таблице(сущности) базе данных подсистемы 6. Система копирует данные из таблицы дисциплин базы данных ТГУ 7. Система обновляет построчно данные в аналогичной таблице(сущности) базе данных подсистемы 8. Система копирует данные из таблицы аудиторий базы данных ТГУ 9. Система обновляет построчно данные в аналогичной таблице(сущности) базе данных подсистемы 10. Система копирует данные из таблицы групп базы данных ТГУ 11. Система обновляет построчно данные в аналогичной таблице(сущности) базе данных подсистемы 12. Система копирует данные из таблицы преподавателей базы данных ТГУ 13. Система обновляет построчно данные в аналогичной таблице(сущности) базе данных подсистемы
Постусловие: Подсистема обновила таблицы преподавателей, групп, аудиторий, институтов, кафедр и дисциплин.
Альтернативные потоки: Нет

Таблица 2.1.5 – Описание прецедента «Расчет активности компонентов»

Прецедент: Расчет активности компонентов
ID: 9
Краткое описание: Реализация функции, которая будет проверять на присутствие преподавателей, групп, аудиторий, институтов и кафедр за определенное время в системе.
Главные актеры: 1. Таймер
Второстепенные актеры:
Предусловие: Прецедент начинается по заданному времени в Таймере
Основной поток: 1. Система отправляет запрос на нахождении уникальных записей преподавателей в таблице занятий базы данных за заданный промежуток времени 2. Преподавателям, которые нашлись им назначается активность «1», остальным назначается активность «0» 3. Система отправляет запрос на нахождении уникальных записей аудиторий в таблице занятий базы данных за заданный промежуток времени 4. Аудитории, которые нашлись им назначается активность «1», остальным назначается активность «0» 5. Система отправляет запрос на нахождении уникальных записей групп в таблице занятий базы данных за заданный промежуток времени 6. Группы, которые нашлись им назначается активность «1», остальным назначается активность «0» 7. По каждой кафедре(преподавателей) производится запрос на нахождении записей преподавателей с активностью «1» в таблице преподавателей базы данных 8. Если такие записи существуют 8.1.Кафедре(преподавателей) назначается активность «1» 9. Если таких записей не существуют 9.1.Кафедре(преподавателей) назначается активность «0» 10. По каждой кафедре(аудиторий) производится запрос на нахождении записей аудиторий с активностью «1» в таблице аудиторий базы данных 11. Если такие записи существуют 11.1.Кафедре(аудиторий) назначается активность «1»

Прецедент: Расчет активности компонентов
12. Если таких записей не существуют 12.1. Кафедре(аудиторий) назначается активность «0»

Продолжение таблицы 2.1.5

Прецедент: Расчет активности компонентов
13. По каждому институту(преподавателей) производится запрос на нахождении записей кафедр(преподавателей) с активностью «1» в таблице кафедр базы данных 14. Если такие записи существуют 14.1. Институту(преподавателей) назначается активность «1» 15. Если таких записей не существуют 15.1. Институту(преподавателей) назначается активность «0» 16. По каждому институту(аудиторий) производится запрос на нахождении записей кафедр(аудиторий) с активностью «1» в таблице кафедр базы данных 17. Если такие записи существуют 17.1. Институту(аудиторий) назначается активность «1» 18. Если таких записей не существуют 18.1. Институту(аудиторий) назначается активность «0» 19. По каждому институту(групп) производится запрос на нахождении записей групп с активностью «1» в таблице групп базы данных 20. Если такие записи существуют 17.1. Институту(групп) назначается активность «1» 21. Если таких записей не существуют 18.1. Институту(групп) назначается активность «0»
Постусловие: Система обновила записи в таблицах параметр активности
Альтернативные потоки: Нет

На рисунке 2.1.6 показана диаграмма вариантов использования проектируемой подсистемы.

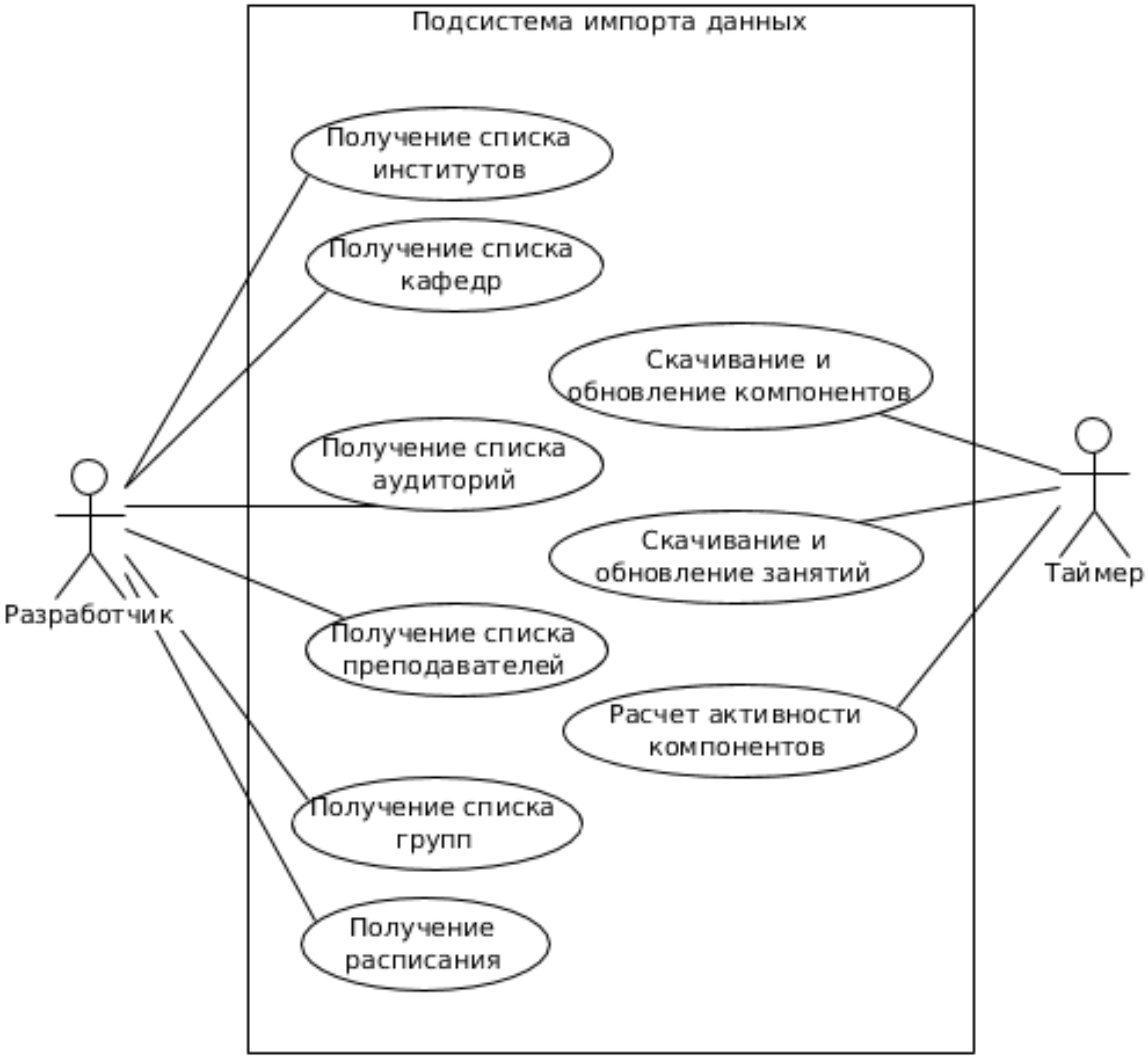


Рисунок 2.1.6 – Диаграмма вариантов использования проектируемой подсистемы импорта расписания

Диаграмма вариантов использования показывает функциональные возможности проектируемой подсистемы импорта расписания, которые должна выполнять система. Например, актер «разработчик» инициирует вариант использования «Получение списка институтов», целью которого является получить данных о том какие есть преподаватели, группы и аудитории в конкретном институте.

Таким образом, в данном пункте были составлены основные прецеденты, описаны самые важные из них, выделены 2 актера, и составлена диаграмма

прецедентов в которой было показано, какой актёр взаимодействует с каким прецедентом.

2.2 Разработка архитектуры подсистемы импорта данных

2.2.1 Разработка первой архитектуры на основе импорта данных из Oracle Database ФГБОУ ВО «ТГУ»

Первая архитектура будет работать на подключения к базе данных системы «Галактика».

Из явных преимуществ является, то что есть все необходимые данные для публикации расписания и возможность реализовать дополнительный функционал виде различного анализа расписания и вариативности его отображения. Но для работы с этими данными требуется реализовать свою базу данных ввиду того что данные содержащие там в первую очередь направленны на планирование расписания, из за чего будут замедления работы при поиске, и отображения данных в своих представлениях.

Сама подсистема из за этого будет поделена на сервер, где идет выполнения будущего приложения и сервера на котором будет хранится будущая база данных

На диаграмма компонентов представленной на рисунке 2.2.2.1, отображено взаимодействие подсистемы между компонентами.

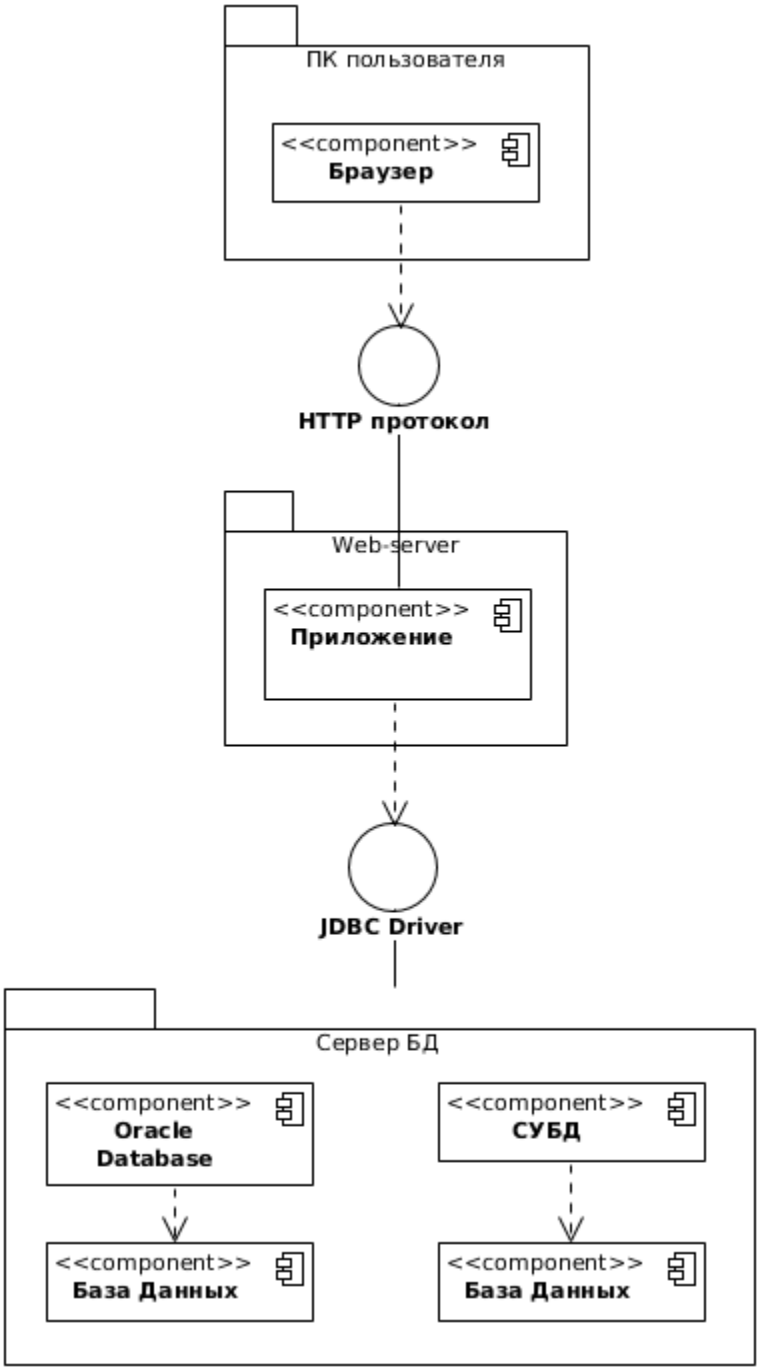


Рисунок 2.2.1.1 – Диаграмма компонентов проектируемой подсистемы

На рисунке 2.2.1.1 компонент «Приложение» выполняет главную работу, а именно анализ данных.

2.2.2 Разработка второй архитектуры на основе файлового хранения

Вторая архитектура будет работать на базе сформированных данных в электронных таблица, которые располагаются на сайте ФГБОУ ВО «ТГУ».

Из явных преимуществ является, то что уже имеются данные пригодные для публикации расписания, где требуется только произвести анализ документа до более простой модели хранения данных, например до хранения данных в JSON формате. На диаграмма компонентов представленной на рисунке 2.2.2.1, отображено взаимодействие подсистемы между компонентами.

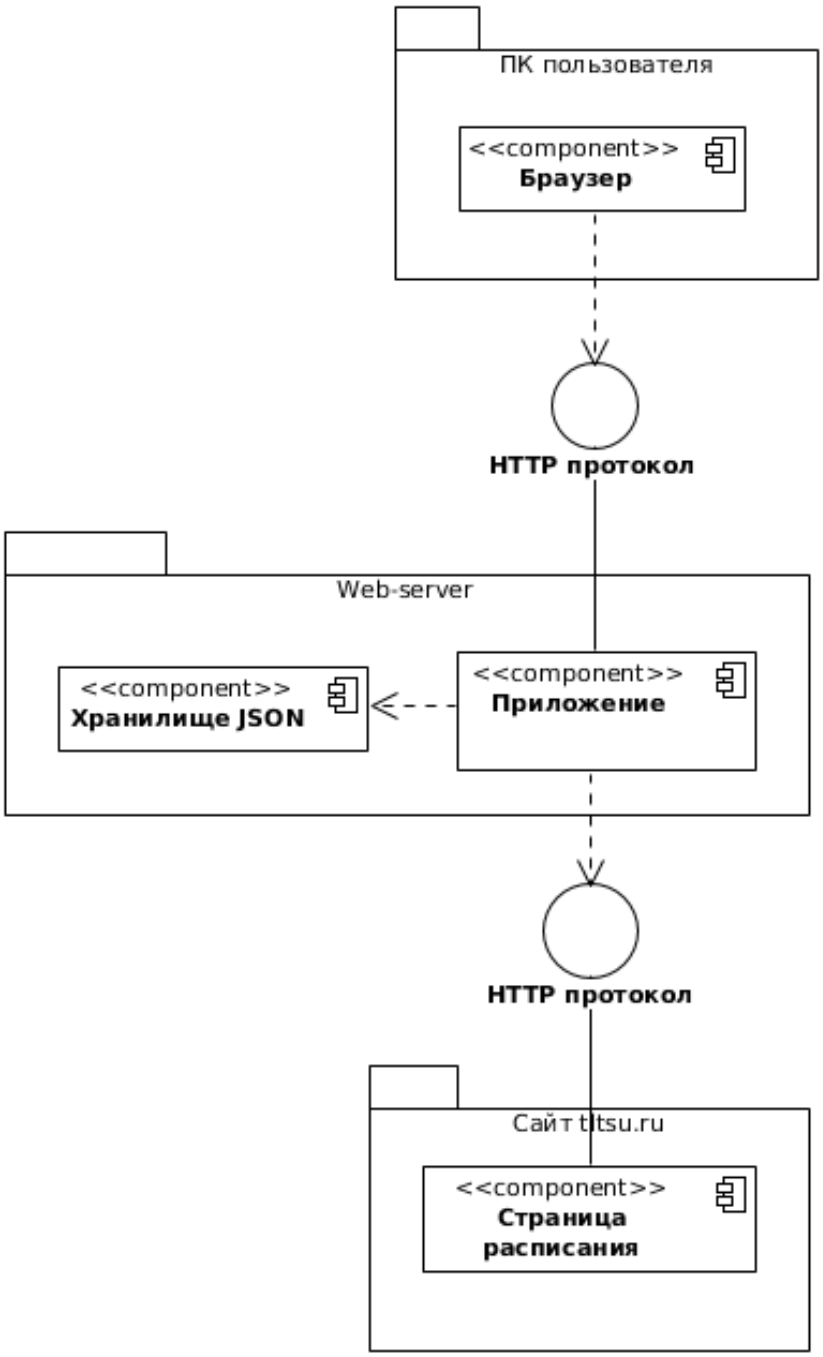


Рисунок 2.2.2.1 – Диаграмма компонентов проектируемой подсистемы

На рисунке 2.2.2.1 компонент «Приложение» выполняет главную работу, а именно анализ, скачивание и сохранение данных.

2.2.3 Выбор архитектуры подсистемы импорта данных

На основании разработанных архитектур, можно выделить несколько критериев для выбора:

- скорость работы подсистемы(отображение);
- скорость работы подсистемы(обновление);
- размер хранения данных;
- вариативность отображения данных.

Сравнение будет происходит из источников[4] [9] [15] [27] [28] [29], в таблице 2.2.3.1.

Таблица 2.2.3.1 - Выбор архитектуры

	База данных	Файлы
Скорость работы подсистемы(отображение)	4	5
Скорость работы подсистемы(обновление)	3	5
Размер хранения данных	5	2
Вариативность отображения данных	5	3
Итого:	17	15

Проведя сравнение можно сделать вывод что работа с базой данных позволяет лучше хранить и управлять данными, но с большим количеством времени, а в работе через хранение файлов все с точностью на оборот.

Подведя итог, будущая архитектура будет через использование систем управлениями баз данных, чтобы в будущем можно было не создавать отдельно новые решения под новые представления отображения, а только добавить соответствующий модуль.

2.3 Разработка взаимодействия между элементами подсистемы посредством диаграммы последовательности

В соответствии с функциями подсистемы надо реализовать диаграмму последовательности функции обновления компонентов, так как она взаимодействует с различными системами управлениями баз данных. Для того чтобы подсистема имела актуальный данные, необходимо реализовать алгоритм обновления данных из базы данных университета. Для этого, подсистеме нужно выполнить алгоритм, который указан ниже:

1. Вызывается функция обновления.
2. Запрашивает данные по таблицам из баз данных системы «Галактика».
3. Сравнивает данные, которые получили с данными из базы данных подсистемы.
4. Обновляет записи, которые не совпали или были изменены, а также добавляет записи в таблицу, если такой записи не существует.
5. Повторяет пункты 3-4 для таблиц студенты, преподаватели, группы, дисциплины, кафедры и институты. Диаграмма последовательности функции обновления расписания занятий представлена на рисунке 2.3.1

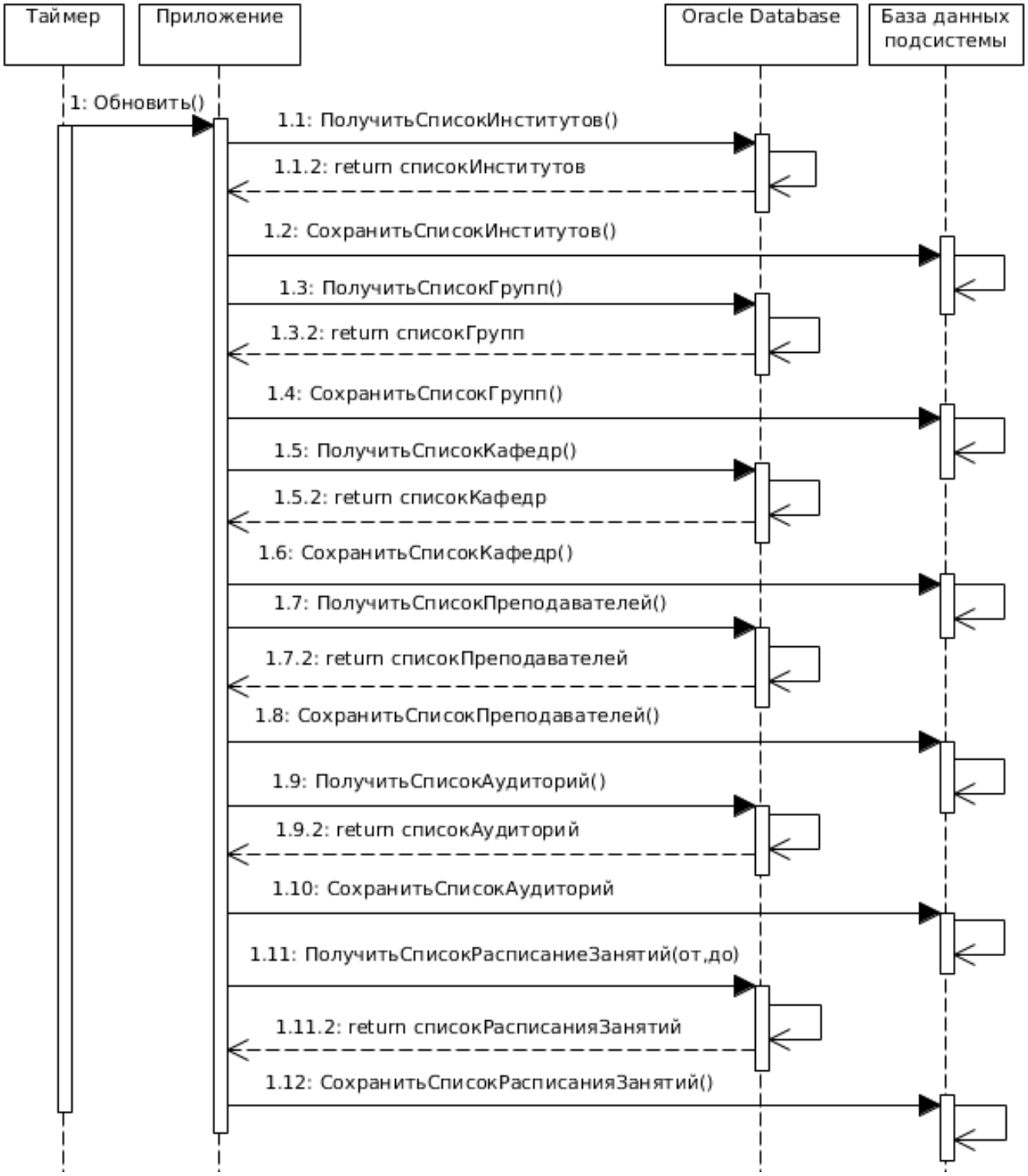


Рисунок 2.3.1 - Фрагмент диаграммы последовательности функции обновления расписания

Построенная диаграмма последовательности показывает порядок обновления и взаимодействия с системами управлениями баз данных подсистемы импорта и системы «Галактика»

2.4 Разработка диаграммы классов подсистемы импорта данных

С помощью разработанной диаграммы вариантов использования можно выделить классы и определить отношения между ними в соответствии с особенностями предметной области. В таблице 2.4.1 показаны классы и их роли в проектируемой подсистеме.

Таблица 2.4.1 - Описание ролей классов в системе

Название класса	**Роль класса в системе**
Entity1 (По таблицам ТГУ)	
PrAuditor	Содержит список аудиторий со всех корпусов университета. Позволяет их выбирать.
PrDiscipline	Содержит список дисциплин. Позволяет их выбирать.
PrFacultyStudent	Содержит список институтов. Позволяет их выбирать.
PrGroup	Содержит список групп со всех корпусов университета. Позволяет их выбирать.
PrKafedr	Содержит список кафедр со всех корпусов университета. Позволяет их выбирать.
PrLecturer	Содержит список преподавателей со всех корпусов университета. Позволяет их выбирать.
PrSchedule	Содержит список занятий. Позволяет их выбирать.
Dao1 (По таблицам ТГУ)	
PrAuditorDAO	Создает доступ к данным, которые используется классом PrAuditor.
PrDisciplineDAO	Создает доступ к данным, которые используется классом PrDiscipline.
PrFacultyStudentDAO	Создает доступ к данным, которые используется классом PrFacultyStudent.

Продолжение таблицы 2.4.1

Название класса	**Роль класса в системе**
PrGroupDAO	Создает доступ к данным, которые используется классом PrGroup.

Название класса	**Роль класса в системе**
PrKafedrDAO	Создает доступ к данным, которые используется классом PrKafedr.
PrLecturerDAO	Создает доступ к данным, которые используется классом PrLecturer.
PrScheduleDAO	Создает доступ к данным, которые используется классом PrSchedule.
Entity2 (По таблицам подсистемы)	
TAuditor	Содержит список аудиторий со всех корпусов университета. Позволяет выбирать и обновлять их.
TFaculty	Содержит список институтов. Может выбирать и обновлять их.
TGroup	Содержит список групп из всех корпусов университета. Может выбирать и обновлять их.
TKafedr	Содержит список кафедр из всех корпусов университета. Может выбирать и обновлять их.
TPredmet	Содержит список дисциплин. Может выбирать и обновлять их.
TPrepod	Содержит список преподавателей из всех корпусов университета. Может выбирать и обновлять их.
TRasp	Содержит список занятий. Может выбирать и обновлять их.
TParaType	Содержит список видов занятий. Может выбирать и обновлять их.
TParaTime	Содержит список время занятий. Может выбирать и обновлять их.
Dao2 (По таблицам подсистемы)	
TAuditorDAO	Создает доступ к данным, которые используется классом TAuditor.
TPredmetDAO	Создает доступ к данным, которые используется классом TPredmet.

Продолжение таблицы 2.4.1

Название класса	**Роль класса в системе**

Название класса	**Роль класса в системе**
TFacultyDAO	Создает доступ к данным, которые используется классом TFaculty.
TGroupDAO	Создает доступ к данным, которые используется классом TGroup.
TKafedrDAO	Создает доступ к данным, которые используется классом TKafedr.
TPrepodDAO	Создает доступ к данным, которые используется классом TPrepod.
TRaspDAO	Создает доступ к данным, которые используется классом TRasp.
TParaTypeDAO	Создает доступ к данным, которые используется классом TParaType.
TParaTimeDAO	Создает доступ к данным, которые используется классом TParaTime.
Controller	
ControllerApi	Организует вывод информации об доступном api через «api/»
ControllerApiForKafedr	Организует вывод информации об кафедрах через «api/PrepodKaf.json» и «api/AuditorKaf.json»
ControllerApiForUnit	Организует вывод информации об единицах(Преподаватели, группы и аудитории) через «api/Prepods.json», «api/Groups.json» и «api/Auditors.json»
ControllerApiInst	Организует вывод информации об институтах через «api/PrepodInst.json», «api/GroupInst.json» и «api/AuditorInst.json»
ControllerApiParaTime	Организует вывод информации о видах занятий через «api/ParaType.json»
ControllerApiParaType	Организует вывод информации о времени занятий через «api/ParaTime.json»
ControllerApiTT	Организует вывод информации об единицах(Преподаватели, группы и аудитории) через «api/TTPrepod.json», «api/TTGroup.json» и «api/TTAuditor.json»

Продолжение таблицы 2.4.1

Название класса	**Роль класса в системе**
Timer	
ScheduledUpdateRasp	Организует автоматический запуск классов и методов в них.
Update	
MainUpdate	Содержит все DAO и организует все виды обновлений в виде методов.
Config	
DataSource1Config	Содержит конфигурационные методы и переменные для подключения к базе данных ТГУ
DataSource2Config	Содержит конфигурационные методы и переменные для подключения к базе данных подсистемы

Из таблицы 2.4.1 видно, что количество классов в подсистеме много для того, чтобы показать их всех на диаграмме классов. Поэтому далее будет описываться диаграмма классов только для сущности обновления расписание аудитории и его вывода. Связи для студентов и преподавателей реализуются подобным образом. На рисунке 2.4.2 показан фрагмент UML диаграммы классов проектируемой подсистемы.

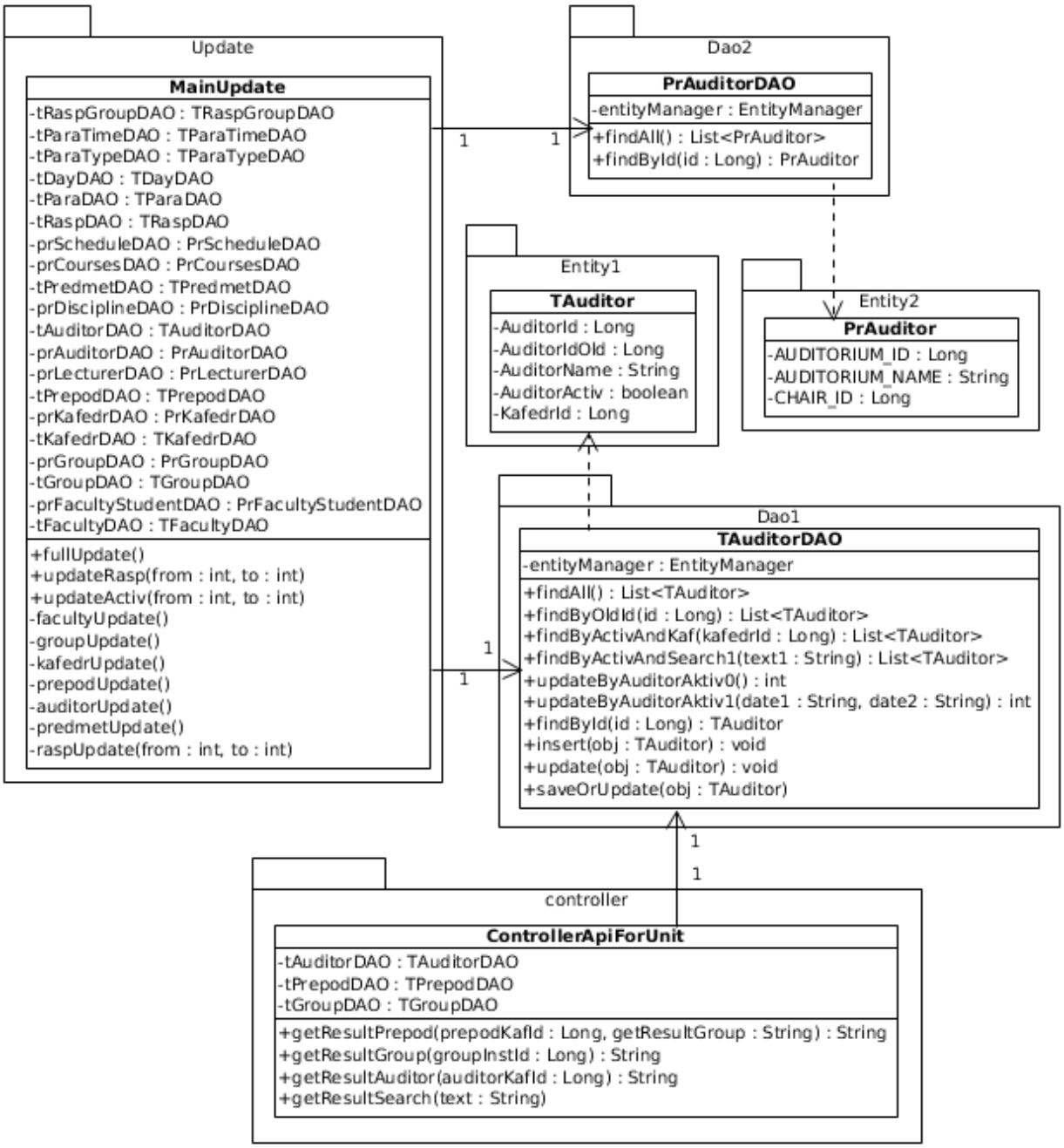


Рисунок 2.4.2 – Фрагмент диаграммы классов проектируемой подсистемы

Сначала необходимо описать класс MainUpdate , который занимается тем что:

1. Скачивает данные из базы данных ТГУ и преобразует в тот вид, который требуется таблицам подсистемы.
2. Проверяет активность кафедр, институтов, преподавателей, групп и аудиторий.

Спецификация данного класса представлена в таблице 2.4.3.

Таблица 2.4.3 — Методы класса MainUpdate

Модификатор доступа	Именование	Результат	Параметры	Назначение
public	fullUpdate	void		Запускает методы, которые отвечают за скачивание из таблиц базы данных ТГУ
private	facultyUpdate	void		Выкачивает данные из таблиц института базы данных ТГУ и переносит преобразуя в таблицы базы данных подсистемы
private	groupUpdate	void		Выкачивает данные из таблиц групп базы данных ТГУ и переносит преобразуя в таблицы базы данных подсистемы
private	kafedrUpdate	void		Выкачивает данные из таблиц кафедр базы данных ТГУ и переносит преобразуя в таблицы базы данных подсистемы
private	prepodUpdate	void		Выкачивает данные из таблиц преподавателей базы данных ТГУ и переносит преобразуя в таблицы базы данных подсистемы
private	auditorUpdate	void		Выкачивает данные из таблиц аудиторий базы данных ТГУ и переносит преобразуя в таблицы базы данных подсистемы

Продолжение таблицы 2.4.3

Модификатор доступа	Именование	Результат	Параметры	Назначение
private	predmetUpdate	void		Выкачивает данные из таблиц дисциплин базы данных ТГУ и переносит преобразуя в таблицы базы данных подсистемы
public	updateRasp	void	int, int	Анализирует входные параметры на корректность и запускает метод обновления расписания за определенный период.
private	raspUpdate	void	int, int	Выкачивает данные из таблиц занятий базы данных ТГУ и переносит преобразуя в таблицы базы данных подсистемы за определенный период
public	updateActiv	void	int, int	Назначает активность преподавателей, групп, аудиторий, институтов и кафедр за определенный промежуток времени

Класс MainUpdate, содержит методы по обновлению занятий и других компонентов подсистемы по средству подключения DAO классов к таблицам.

После того как были реализованы классы работы с данными таблиц и их обновления, необходимо создать классы вывода данных через сервлеты. Где они все будут подключаться через DAO классы для получения данных, после чего они преобразуют данные в требуемый вид и отправят их обратно пользователю.

3 РЕАЛИЗАЦИЯ ПОДСИСТЕМЫ ИМПОРТА ДАННЫХ

3.1 Выбор средств реализации подсистемы импорта

3.1.1 Выбор объектно-ориентированного языка программирования

Основываясь на собственных навыках создания приложений, а также следованию использования языков в мировой практике, было сделано решения, что в данном случае для сравнительного анализа будут выбранны языки программирования:

- PHP;
- C#;
- Java.

PHP – скриптовый язык общего назначения, где есть поддержка объектно-ориентированного программирования[15].

C# – сочетание объектно-ориентированной и контекстно-ориентированной концепции. Является основным языком разработки под приложения платформы Microsoft .NET[12].

Java – объектно-ориентированный язык программирования, где выполнение кода осуществляется с помощью виртуальной Java-машины[3] [6] [13] [14].

При выборе языка программирования можно отметить главные критерии, с помощью которых будет выполняться выбор. Это:

- опыт работы;
- библиотеки;
- производительность кода и требовательность к ресурсам;
- удобство сборки;
- язык и синтаксис;

– удобство отладки.

Сравнение будет происходит из источников, которые были указанны выше и опыта работы в таблице 3.1.1.1.

Таблица 3.1.1.1 - Выбор языков программирования

Функции и возможности	Языки программирования		
	Java	C#	PHP
Опыт работы	7	4	8
Библиотеки	5	5	4
Производительность кода и требовательность к ресурсам	4	6	5
Удобство сборки	6	6	8
Язык и синтаксис	5	5	4
Удобство отладки	6	5	3
Итого:	33	31	30

Таким образом, при проведении анализа был выбран язык программирования – Java. Java является очень удобным языком и простым в обучении. Существует огромное количество различных библиотек, которые можно применять для различных ситуаций. Также основным фактором будет это опыт работы с языком, что сильно увеличит скорость разработки приложения и отладка кода на ошибки.

3.1.2 Выбор системы управления базами данных

Основываясь на собственных навыках проектирования баз данных, а также следованию использования СУБД в мировой практике, было сделано решения, что в данном случае для сравнительного анализа будут выбранны следующие СУБД:

– PostgreSQL – объектно-реляционная система управления базами данных[29];

– Oracle Database – объектно-реляционная система управления базами данных[28];
– MariaDB – реляционная система управления базами данных[27].

При выборе системы управления баз даннызх можно отметить главные критерии, с помощью которых будет выполняться выбор. Это:

- надежность;
- моделирование данных;
- производительность;
- опыт работы.

Где под моделированием данных подразумевается сравнение представления таких параметров как:

- триггеры
- хранимые процедуры
- типы данных предусмотренные СУБД
- языки запросов

Сравнение будет происходит из источников, которые были указанны выше и опыта работы в таблице 3.1.2.1.

Таблица 3.1.2.1 - Выбор подсистемы управления базами данных

Критерии	PostgreSQL	Oracle Database	MariaDB
Надежность	6	4	4
Моделирование данных	5	7	4
Опыт работы	2	3	6
Производительность	4	4	5

Итог	17	18	22

Проведя сравнение систем управлений баз данных, был сделан выбор в пользу MariaDB. Один из положительных факторов будет такой как опыт работы с этой системой управлениями баз данных, что сильно ускорит работу при разработке баз данных.

3.2 Разработка основных функций подсистемы импорта данных

Основная задача реализуемого программного обеспечения - это создание подсистемы импорта данных расписания.

Выделим основные функции подсистемы:

- подключение к базам данных;
- обновление данных;
- вывод информации в виде json файлов.

3.2.1 Разработка функции подключение к базам данным

Создание функции обновления учебных занятий ФГБОУ ВО "ТГУ". Реализуем классы которые находятся в пакете Entity1 и Entity2. Эти классы отвечают за хранение содержимого таблиц баз данных.

Все они реализуют по похожему образу. Единственная разница состоит в том что они используют таблицы разных баз данных. В качестве примера на рисунке 3.2.1.1 показан реализация класса TGroup.

```java
@Entity
@Table(name = "tGroup")
public class TGroup implements Serializable {

    private static final long serialVersionUID = 7462371260880513l2L;


    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "GroupId")
    private Long GroupId;

    @Basic
    @Column(name = "GroupIdOld", length = 11, nullable = false,insertable = true, updatable = true)
    private Long GroupIdOld;

    @Basic
    @Column(name = "GroupName", length = 255, nullable = false,insertable = true, updatable = true)
    private String GroupName;

    @Basic
    @Column(name = "GroupActiv", length = 1, nullable = false,insertable = true, updatable = true)
    private boolean GroupActiv;

    @Basic
    @Column(name = "FacultyId", length = 1, nullable = false,insertable = true, updatable = true)
    private Integer FacultyId;
```

Рисунок 3.2.1.1 – Реализация класса TGroup

Экземпляры этого класса будут возвращаться при получения данных из таблицы TGroup.

Потом реализуем класс к доступу данным. Все эти классы, оканчиваются на постфикс DAO. Приведем пример класс получения доступа сущности TGroup - TGroupDAO. Из за объема класса будет приведен в примере только небольшая часть кода. На рисунке 3.2.1.2.

```
public List<TGroup> findByActiv() {
    String sql = "Select e from " + TGroup.class.getName() + " e WHERE e.GroupActiv=true";
    Query query = entityManager.createQuery(sql, TGroup.class);
    return query.getResultList();
}
public List<TGroup> findByActivAndInst(int groupInstId) {
    String sql = "Select e from " + TGroup.class.getName() + " e WHERE e.GroupActiv=true AND e.FacultyId="+groupInstId;
    Query query = entityManager.createQuery(sql, TGroup.class);
    return query.getResultList();
}
public List<TGroup> findByActivAndSearch1(String text1) {
    String sql = "Select e from " + TGroup.class.getName() + " e WHERE e.GroupActiv=true AND "
            + "lower(e.GroupName) LIKE lower('"+text1+"%') ";
    Query query = entityManager.createQuery(sql, TGroup.class);
    return query.getResultList();
}

public int updateByGroupAktiv0() {
    //System.out.println(new Date());
    String sql = "Update " +
            TGroup.class.getName() + " g " +
            "SET g.GroupActiv = false ";
    Query query = entityManager.createQuery(sql);
    return query.executeUpdate();
}

public int updateByGroupAktiv1(String date1,String date2) {
    //System.out.println(new Date());
    String sql = "Update " +
            TGroup.class.getName() + " g " +
            "SET g.GroupActiv = true " +
            "WHERE g.GroupId IN (Select rg.GroupId FROM " +
                TRaspGroupTree.class.getName() + " rg " +
                "JOIN rg.RaspId r " +
                "JOIN r.ParaId pr " +
                "JOIN pr.DayId d " +
                "WHERE (d.DayData >= '"+date1+"' AND d.DayData <= '"+date2+"') " +
                "AND r.RaspTimer=d.DataTimer " +
                "AND d.DataTimer=rg.RaspGroupTimer GROUP BY rg.GroupId " +
            ")";
    Query query = entityManager.createQuery(sql);
    return query.executeUpdate();
}
```

Рисунок 3.2.1.2 – Реализация класса TGroupDAO

На примере, который выше идет запрос к базе данных на JPQL, где потом идет обработка результата.

Для того чтобы связываться с базами данных необходимо создать конфигурацию под каждую базу данных. В которое входит описание:

- DriverManager;
- EntityManager;
- TransactionManager.

Для того чтобы можно было:

– подключаться к базе данных;

– чтение таблиц баз данных;

– изменение таблиц баз данных.

Ниже на рисунке 3.2.1.3 будет представлен пример подключения к базе данных ТГУ. По похожему принципу идет подключение к базе данных

```
public class DataSource1Config {
    @Autowired
    private Environment env;
    @Bean
    public DataSource ds1Datasource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName(env.getProperty("spring.datasource.driver-class-name.1"));
        dataSource.setUrl(env.getProperty("spring.datasource.url.1"));
        dataSource.setUsername(env.getProperty("spring.datasource.username.1"));
        dataSource.setPassword(env.getProperty("spring.datasource.password.1"));
        return dataSource;
    }
    @Bean
    public LocalContainerEntityManagerFactoryBean ds1EntityManager() {
        LocalContainerEntityManagerFactoryBean em = new LocalContainerEntityManagerFactoryBean();
        em.setDataSource(ds1Datasource());
        em.setPackagesToScan(new String[] { Constants.PACKAGE_ENTITIES_1 });
        em.setPersistenceUnitName(Constants.JPA_UNIT_NAME_1);
        HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
        em.setJpaVendorAdapter(vendorAdapter);
        HashMap<String, Object> properties = new HashMap<>();
        properties.put("hibernate.dialect", env.getProperty("spring.jpa.properties.hibernate.dialect.1"));
        properties.put("hibernate.show-sql", env.getProperty("spring.jpa.show-sql.1"));
        em.setJpaPropertyMap(properties);
        em.afterPropertiesSet();
        return em;
    }
    @Bean
    public PlatformTransactionManager ds1TransactionManager() {
        JpaTransactionManager transactionManager = new JpaTransactionManager();
        transactionManager.setEntityManagerFactory(ds1EntityManager().getObject());
        return transactionManager;
    }
```

подсистемы.

Рисунок 3.2.1.3 – Реализация класса DataSource1Config

После выполнения всех выше действий подсистема импорта данных готова к взаимодействию с данными из баз данных.

3.2.2 Разработка функции обновление данных

Для того чтобы обновлялись данные для начало надо описать класс актера «Таймер». Чтобы реализовать актера «Таймер» были применены библиотеки «Cron». На рисунке 3.2.2.1 можно увидеть реализацию ScheduledUpdateRasp.

```
@Scheduled(cron="0 0 23 * * *")
public void updateRaspActiv() {
    System.out.println("Start update updateRaspActiv"+ new Date().toString());
    System.out.println(getMainUpdate().updateActiv(-90,90));
    System.out.println("End update updateRaspActiv"+ new Date().toString());
}

@Scheduled(cron="0 0 8,14,17,21 * * *")
public void updateRaspWeek1() {
    System.out.println("Start update updateRaspWeek1 "+ new Date().toString());
    getMainUpdate().updateRasp(0,6);
    System.out.println("End update updateRaspWeek1 "+ new Date().toString());
}

@Scheduled(cron="0 0 20 * * *")
public void updateRaspWeek2to4() {
    System.out.println("Start update updateRaspWeek2 "+ new Date().toString());
    getMainUpdate().updateRasp(7,14);
    System.out.println("End update updateRaspWeek2 "+ new Date().toString());

    System.out.println("Start update updateRaspWeek3 "+ new Date().toString());
    getMainUpdate().updateRasp(15,21,true);
    System.out.println("End update updateRaspWeek3 "+ new Date().toString());

    System.out.println("Start update updateRaspWeek4 "+ new Date().toString());
    getMainUpdate().updateRasp(22,28,true);
    System.out.println("End update updateRaspWeek4 "+ new Date().toString());
}

@Scheduled(cron="0 0 0 * * FRI")
public void fullUpdate() {
    System.out.println("Start update fullUpdate "+ new Date().toString());
    getMainUpdate().fullUpdate();
    System.out.println("End update fullUpdate "+ new Date().toString());
}
```

Рисунок 3.2.2.1 – Реализация класса ScheduledUpdateRasp

На рисунке 3.2.2.1 отображены запуски для обновления расписания на 1,2,3,4 неделю вперед, обновление активности и обновление таблиц с информацией требуемых для занятий. Например таблиц преподавателей.

Основная логика обновления расписание занятий или других сущностей идет за счет взаимодействий различных баз данных и проверок на соответствия записей. Например на рисунке 3.2.2.2 отображена диаграмма функции обновления преподавателей, аудиторий, групп и остальных сущностей.

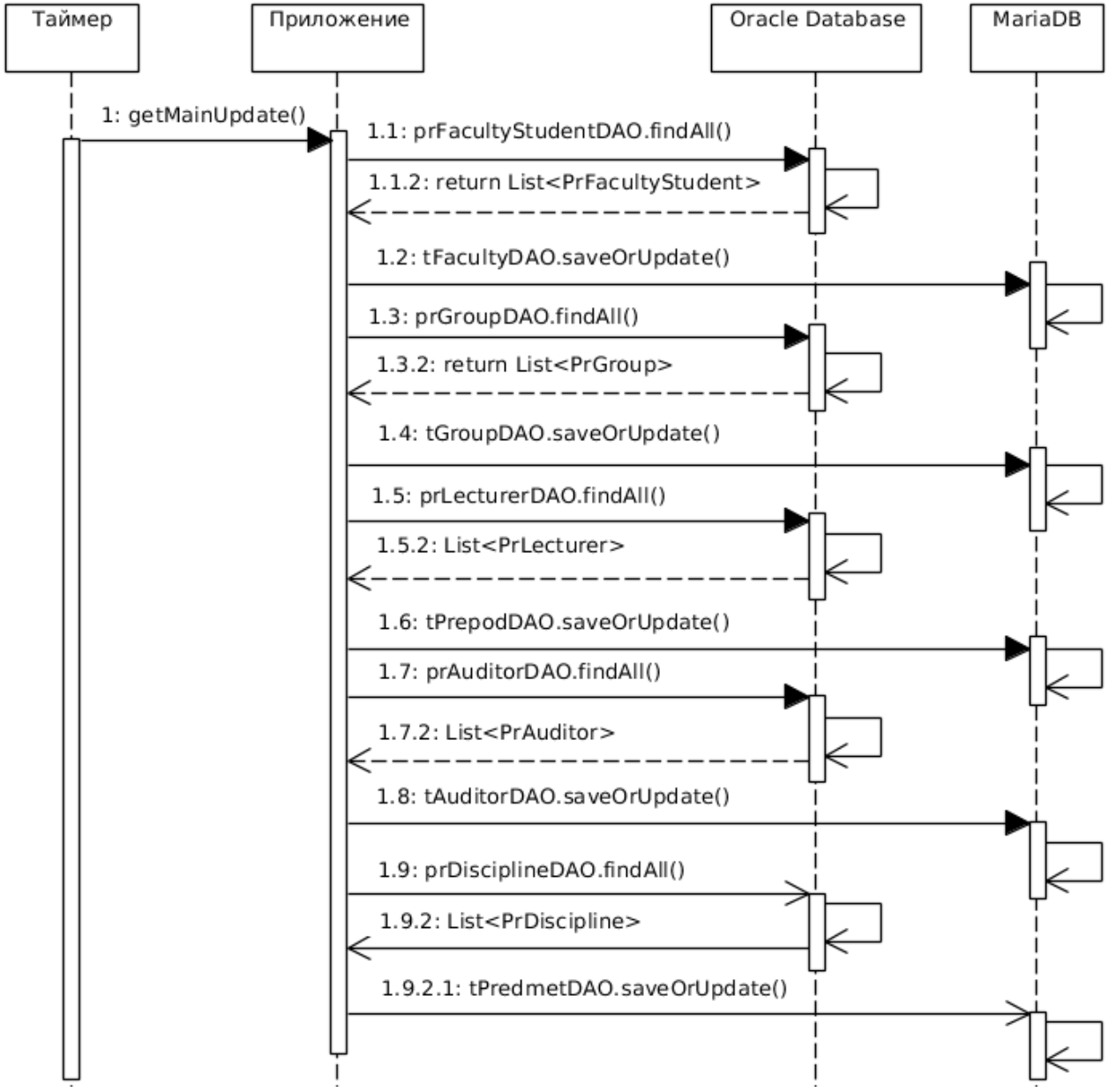


Рисунок 3.2.2.2 – Диаграмма последовательности метода getMainUpdate()

Основная логика проверки данных заключается в присваивание нового идентификатора и сохранения старого, для проверки с новыми данными. Но для сохранения данных о занятиях используется сравнение по хэш значению, который является результатом слияния идентификаторов:

- аудитории;
-
- преподавателям;
- дисциплины.

И сравнении номера обновления занятия. Где если идет не соответствие по номеру обновления, то занятие считается удаленным. Если идет несоответствие по хэш значению, то занятие считается измененное.

3.2.3 Разработка функции выдачи данных

Для того чтобы выводить данных нужно будет создать адреса, к которым можно будет обращаться. В качестве примера возьмем класс ControllerApiInst на рисунке 3.2.3.1. Он выводит институты для групп, преподавателей и аудиторий.

```
@RequestMapping(value = "/api/PrepodInst.json", method = GET, params = "token")
public String getResultPrepod(@RequestParam("token") String token) {
    if(tTokenDAO.isKey(tStatisticsDAO,token,new Long(1),"")){
        List<TFaculty> result = tFacultyDAO.findByActivPrepod();
        String count = "{\"prepodInsts\":[";
        int size = result.size();
        for(TFaculty p : result){
            count+=p.toJsonPrepod();
            size--;
            if(size!=0){count+=",";};
        }
        count += "]}";
        return count;
    }else{
         return "Token incorrect";
    }
}
```

Рисунок 3.2.3.1 – Реализация класса ControllerApiInst

На рисунке 3.2.3.1 идет реализация метода в котором идет маппинг по адресу «/api/PrepodInst.json». Где под запрос идет обработка и выдача данных. Работа с получения остальных данных идет подобным образом. В таблице 3.2.3.2 перечень всех api доступные подсистемой.

Таблица 3.2.3.2 - Перечень всех API

Api	Описание
PrepodInst.json	Институты преподавателей
PrepodKaf.json	Кафедры преподавателей
Prepods.json	Преподаватели в кафедре
TTPrepod.json	Расписание для преподавателей
GroupInst.json	Институты групп
Groups.json	Группы в институте
TTGroup.json	Расписание групп
AuditorInst.json	Институты аудиторий
AuditorKaf.json	Кафедры аудиторий
Auditors.json	Аудитории в кафедре
TTAuditor.json	Расписание аудиторий
ParaTime.json	Время занятий
ParaType.json	Виды занятий

Search.json	Поиск аудитор, групп, преподавателей

Где с помощью их можно получить любые данные по занятиям и всего, что с ним связано.

3.3 Описание разработанной подсистемы импорта данных

Разработанная подсистема импорта данных состоит из:

- приложения обновления данных;
- приложения выдачи данных;
- базы данных.

Также состоит из сторонних элементов, таких как:

- клиент, и его браузер в котором идет получения api через javascript;
- база данных системы «Галактика»

Где на рисунке 3.4.2 показано их взаимодействия.

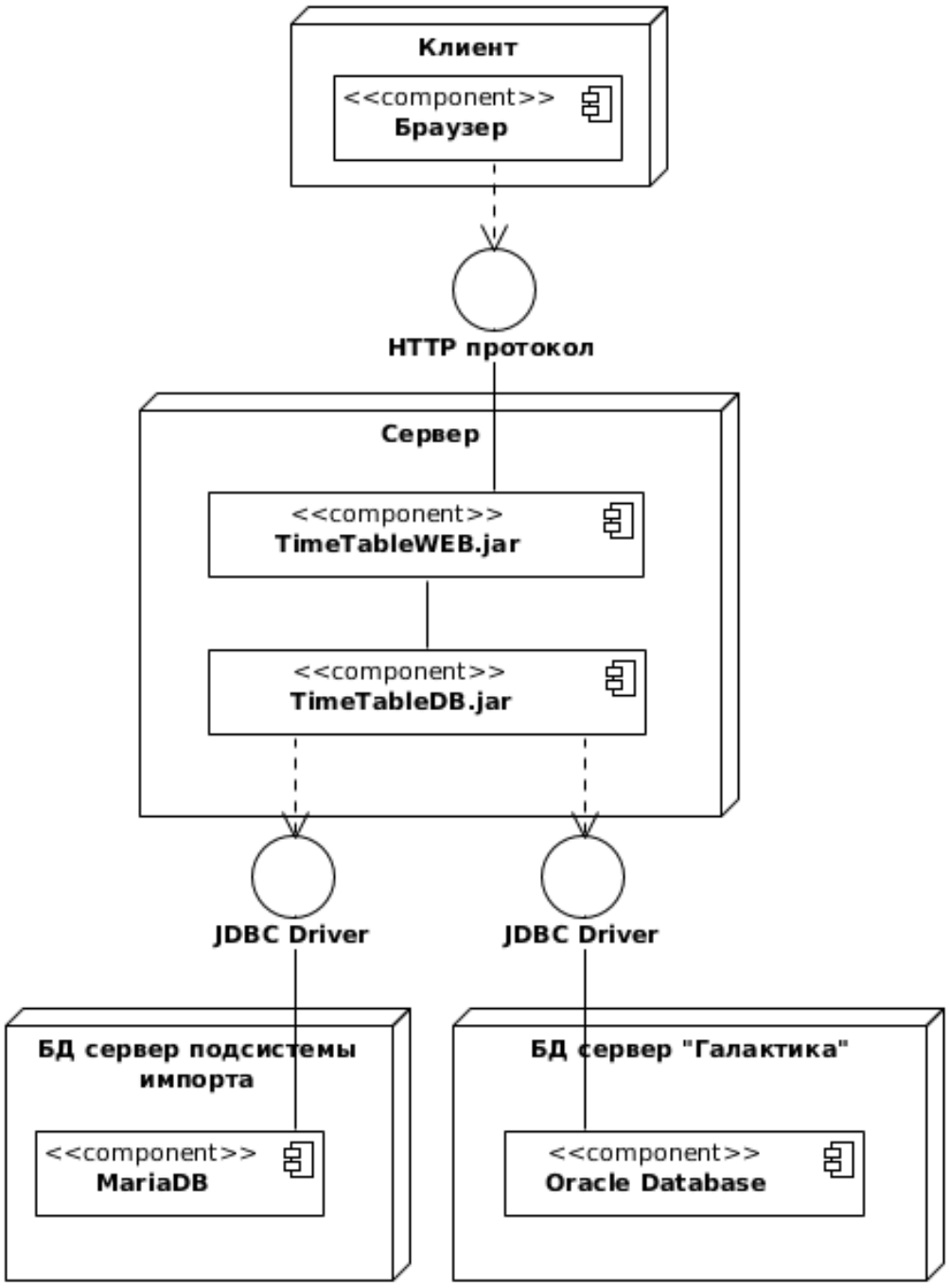


Рисунок 3.4.2 – Диаграмма развертывания подсистемы импорта данных

Таким образом, в данном пункте были показаны основные компоненты работы подсистемы импорта данных в систему публикации расписания учебных занятий ФГБОУ ВО "ТГУ"

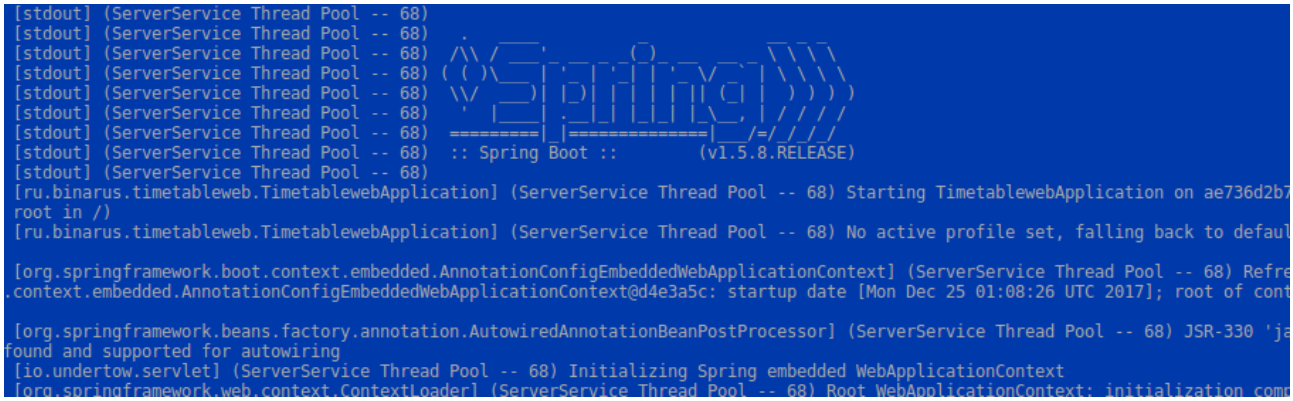
3.4 Тестирование разработанного приложения

Тестирование разработанной подсистемы будет будет состоять из:

– проверки корректности вывода данных;

– проверки корректности обновления данных.

Для тестирования подсистемы требуется сервер приложения. Для этого был установлен сервер приложений - wildfly. Где на рисунке 3.4.1 показан



запуск приложения на wildfly сервере приложений.

Рисунок 3.4.1 – Запуск сервера из командной строки

Приложение запускается и выводит информацию о завершения запуска. Для теста необходимо чтобы обновились таблицы баз данных, после этого можно увидеть работоспособность обновления данных. Где результат работы можно просмотреть в логах. На рисунке 3.4.2.

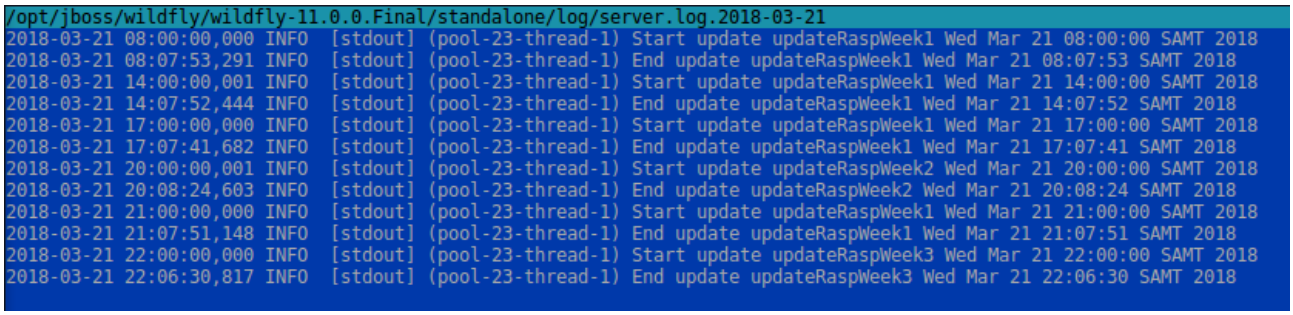


Рисунок 3.4.2 – Лог работы сервера

По логу рисунка 3.4.2 видно, что удачно обновилось:

– 4 раза обновилось расписание занятий на 1 неделю вперед;

– обновилось расписание занятий на следующую неделю;

– обновилось расписание занятий через неделю.

Теперь необходимо зайти на страницу информации о доступных данных,

Api	Параметры	Результат	Описание	Пример
PrepodInst.json	token	prepodInsts:[{ prepodInstId prepodInstName }]	Институты преподавателей	PrepodInst.json
PrepodKaf.json	token prepodInstId	prepodKafs:[{ prepodKafId prepodKafName }]	Кафедры преподавателей	PrepodKaf.json
Prepods.json	token prepodKafId	prepods:[{ prepodId prepodName }]	Преподаватели в кафедре	Prepods.json

TTPrepod.json	token
	prepodId
	from
	to
		tts:[{
		TT{
		paraTimeId
		paraDate
		paraType
		predmetName
		predmetSokr
		prepodId
		prepodFIO
		auditorId
		auditorName
		raspCorrect
		groups:[{
		groupId
		groupName
		}]
		}
		}]
			Расписание для преподавателей	TTPrepod.json

GroupInst.json	token	groupInsts:[{ groupInstId groupInstName }]	Институты групп	GroupInst.json
Groups.json	token groupInstId	groups:[{ groupId groupName }]	Группы в институте	Groups.json

		tts:[{
		TT{
		paraTimeId
		paraDate
		paraType		

на рисунке 3.4.3.

Рисунок 3.4.3 – страница информации о доступных данных.

Страница отображается корректно и все страницы по предоставлению данных работаю тоже корректно, в качестве примера приведу json+ преподавателей кафедры «Прикладная математика и информатика» на рисунке 3.4.4.

```
{"prepods":[{"prepodId":141,"prepodName":"Егорова Эльвира Валентиновна"},{"prepodId":142,"prepodName":"Никанов Евгений Борисович"},{"prepodId":150,"prepodName":"Панюкова Екатерина Владимировна"},{"prepodId":225,"prepodName":"Аникина Оксана Владимировна"},{"prepodId":226,"prepodName":"Ахмедханлы Дилара Микаил кызы"},{"prepodId":227,"prepodName":"Очеповский Андрей Викторович"},{"prepodId":228,"prepodName":"Лелонд Ольга Владимировна"},{"prepodId":248,"prepodName":"Тырыгина Галина Алексеевна"},{"prepodId":320,"prepodName":"Гущина Оксана Михайловна"},{"prepodId":333,"prepodName":"Глазова Вера Федоровна"},{"prepodId":382,"prepodName":"Туищев Алексей Иванович"},{"prepodId":383,"prepodName":"Казаченок Надежда Николаевна"},{"prepodId":387,"prepodName":"Тренина Марина Анатольевна"},{"prepodId":1804,"prepodName":"Султанов Тимур Геннадьевич"},{"prepodId":1988,"prepodName":"Ерофеева Елена Александровна"},{"prepodId":2042,"prepodName":"Митин Сергей Владимирович"},{"prepodId":2043,"prepodName":"Лисовская Мария Германовна"},{"prepodId":2105,"prepodName":"Сосина Наталья Алексеевна"},{"prepodId":4378,"prepodName":"Кузьмичев Алексей Борисович"},{"prepodId":4402,"prepodName":"Тонких Артем Петрович"},{"prepodId":4416,"prepodName":"Климов Виталий Сергеевич"},{"prepodId":4418,"prepodName":"Мкртычев Сергей Вазгенович"},{"prepodId":4424,"prepodName":"Баумгертнер Светлана Викторовна"},{"prepodId":4436,"prepodName":"Сафронов Александр Иванович"},{"prepodId":4438,"prepodName":"Рогова Наталья Николаевна"},{"prepodId":4505,"prepodName":"Шляпкин Андрей Владимирович"}]}
```


Рисунок 3.4.4 – содержимое json файла страницы преподавателей

Из рисунка 3.4.4 видны объекты, которые содержат:

- фамилию;
- имя;
- отчество;
- идентификатор преподавателя.

Из просмотра из базы данных видно, что присутствуют еще преподаватели кафедры, но из-за того что этот преподаватель больше не преподает, он не отображается в общем списке. Это показывает, что приложение корректно отсеивает неактивные элементы.

ЗАКЛЮЧЕНИЕ

Во время выполнение бакалаврской работы был проведен анализ уже существующей системы публикации ФГБОУ ВО "ТГУ". Была выделена актуальность исследуемой темы, определены объект, предмет исследования, цели и задачи работы. В результате чего было принято решение об улучшении процесса импорта расписания учебных занятий в ФГБОУ ВО "ТГУ", а так же по итогам анализа были сформулированы функциональные требования к разрабатываемой технологии. В ходе проектировании подсистемы были составлены две архитектуры, где был сделан выбор в сторону работы с базой данных. Также были сделаны диаграммы прецедентов, последовательности действий, развертывания, вариантов использования и классов. Была разработана модель приложения, которая умеет импортировать данные из базы данных ФГБОУ ВО "ТГУ", конкретно из базы данных системы «Галактика», а также выводить их для сервисов по отображению расписания учебных занятий ФГБОУ ВО "ТГУ", с мобильных устройств и отображения с интернет браузера. По результатам разработки были сделаны несколько версия приложения по импорту расписания учебных занятий ФГБОУ ВО "ТГУ", которые уже внедрены в учебный процесс университета, и которымм уже пользуются тысячи студентов и преподавателей.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2006. – 496 с.
2. Гущин А. Н. Базы Данных. 2-е изд., испр. и доп.: учебно-методическое пособие / А. Н. Гущин – М. Берлин: Директ – Медиа, 2015. – 311с.
3. Java 8. Полное руководство; 9-е изд.: Пер. с англ. - М. : ООО "И.Д. Вильяме",2015. - 1 376 с.
4. Microsoft Office Excel 2007. Анализ данных и бизнес-моделирование 14843K, 600 с.
5. Фаулер М. Рефакторинг. Улучшение существующего кода / М. Фаулер. – СПб : Символ Плюс, 2015. – 432с.
6. Программирование на Java для начинающих / Майк Мак-Грат ; [пер. с англ. М.А. Райтмана]. – Москва : Издательство «Э», 2016. – 192 с.
7. JavaScript. Подробное руководство. – Пер. с англ. – СПб: Символ#Плюс,2008. – 992 с.
8. Закон РФ от 10.07.1992 N 3266-1 (ред. от 12.11.2012) "Об образовании" [Электронный ресурс] // СПС КонсультантПлюс: Законодательство: Версия Проф. – URL: http://www.consultant.ru/document/cons_doc_LAW_1888 (1.06.2016)
9. MySQL. Библиотека профессионала.: Пер. с англ. — М.: Издательский дом"Вильяме", 2002. — 624 с.
10. Тольяттинский государственный университет [Электронный ресурс] // Тольяттинский государственный университет: История Тольяттинского государственного университета. – URL: http://www.tltsu.ru (1.06.2016)

11. Крис Шеффер, Кларенс Хо, Роб Харроп. Spring 4 для профессионалов = Pro Spring 4. — М.: «Вильямс», 2017. — 752 с.

12. Джон Скит. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод = C# in Depth, 3rd ed.. — М.: «Вильямс», 2014. — 608 с.

13. Кей С. Хорстманн. Java. Библиотека профессионала, том 1. Основы. 10-е издание = Core Java. Volume I - Fundamentals (Tenth Edition). — М.: «Вильямс», 2017. — 864 с.

14. Джошуа Блох. Java. Эффективное программирование = Effective Java. — М.: Лори, 2002. — 224 с.

15. PHP и MySQL. Исчерпывающее руководство. — СПб.: Питер, 2013. — 512 с.

16. Монахов Вадим. Язык программирования Java и среда NetBeans. — 3-е изд. — СПб.: БХВ-Петербург, 2011. — 704 с.

17. Изучаем C#. 3-е изд. — СПб.: Питер, 2014. — 816 с.

18. Lippert, Eric (March 19, 2009). "Representation and Identity". Fabulous Adventures In Coding. Blogs.msdn.com. Retrieved October 4, 2012.

19. Bill Burke, RESTful Java with JAX-RS 2.0, Second Edition / Bill Burke - O'Reilly Media, Inc. – 392 p. 1

20. Byrne, Joseph Patrick (2008). Encyclopedia of Pestilence, Pandemics, and Plagues: A-M. ABC-CLIO. p. 99.

21. David Mercer (1988). The Global IBM: Leadership in Multinational Management. Dodd, Mead. p. 374.

22. Henry Bakis (1987). "Telecommunications and the Global Firm". In F. E. Ian Hamilton. Industrial change in advanced economies. London: Croom Helm. pp. 130–160.

23. Mak, Gary (September 1, 2010). Spring Recipes: A Problem-Solution Approach (Second ed.). Apress. p. 1104.

24. Walls, Craig (November 28, 2010). Spring in Action (Third ed.). Manning. p. 700.

25. Johnson, Rod (October 2002). Expert One-on-one J2EE Design and Development (First ed.). Wrox Press. p. 750.

26. Sarin, Ashish (June 27, 2016). Getting started with Spring Framework (Third ed.). Self-published. p. 626.

27. Bartholomew, Daniel. MariaDB Cookbook. — 2014. p. 145.

28. Gupta, Saurabh (2012). Oracle Advanced PL/SQL Developer Professional Guide. Packt Publishing Ltd. p. 123.

29. Gilmore, W. Jason; Treat, Robert (February 27, 2006). Beginning PHP and PostgreSQL 8: From Novice to Professional. Apress. p. 896.

ПРИЛОЖЕНИЯ

Листинг кода реализации классов

Прикреплен на диске