МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

02.03.03 МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка технологии интеграции системы составления расписания учебных занятий в единое информационное пространство ТГУ

Студент Д. С. Корецкий _____

Руководитель А. А. Гальцев _____

Допустить к защите
Заведующий кафедрой к. тех. н, доцент, А.В. Очеповский _____

«_____»_____20_____г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав. кафедрой «Прикладная
математика и информатика»
_____ А.В. Очеповский

«____»_____2016 г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент Корецкий Дмитрий Сергеевич

1. Тема: Разработка технологии интеграции системы составления расписания учебных занятий в единое информационное пространство ТГУ
2. Срок сдачи студентом законченной выпускной квалификационной работы 20.05.2016
3. Исходные данные к выпускной квалификационной работе: технологический процесс публикации учебных занятий в единое информационное пространство ТГУ
4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов): анализ предметной области, проектирование новой технологии интеграции расписания учебных занятий в единое информационное пространство ТГУ, разработка технологии интеграции расписания учебных занятий в единое информационное пространство ТГУ
5. Ориентировочный перечень графического и иллюстративного материала: экранные формы, программное приложение, презентация по разработанной технологии интеграции
6. Дата выдачи задания « 11 » января 2016 г.

Руководитель выпускной
квалификационной работы _____ А.А. Гальцев

Задание принял к исполнению _____ Д.С. Корецкий

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав. кафедрой «Прикладная математика и информатика»
_____ А.В. Очеповский

«____»_____2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента Корецкого Дмитрия Сергеевича
по теме Разработка технологии интеграции системы составления расписания учебных занятий в единое информационное пространство ТГУ

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Анализ существующей технологии предоставления расписания учебных занятий в единое информационное пространство ТГУ	25.02.2016	25.02.2016	Выполнено	
Проектирование технологии интеграции системы составления расписания в единое информационное пространство ТГУ	25.03.2016	25.03.2016	Выполнено	
Кодирование технологии интеграции системы составления расписания в единое информационное пространство ТГУ	25.04.2016	25.04.2016	Выполнено	
Оформление пояснительной	10.05.2016	10.05.2016	Выполнено	

записки				
Подготовка презентации и доклада по ВКР	15.05.2016	15.05.2016	Выполнено	
Представление пояснительной записки к предзащите	18.05.2016	18.05.2016	Выполнено	

Руководитель выпускной квалификационной работы _____ А.А. Гальцев

Задание принял к исполнению _____ Д.С. Корецкий

Аннотация

Тема: Разработка технологии интеграции системы составления расписания в единое информационное пространство ТГУ

Ключевые слова: JAVA EE, SPRING, MYSQL, WEB-SERVICE, ИНТЕГРАЦИЯ, КЛИЕНТ-СЕРВЕР.

Актуальность темы бакалаврской работы заключается в том, что автоматизация процесса публикации расписания учебных занятий, значительно сокращает время, необходимое на ручную публикацию, и позволяет предоставлять информацию в требуемом формате на требуемый носитель информации (телефон, планшет, персональный компьютер).

Объектом работы является технология интеграции расписания учебных занятий в единое информационное пространство ТГУ.

Предметом работы является автоматизация процесса публикации расписания учебных занятий в единое информационное пространство ТГУ.

Целью данной выпускной квалификационной работы является разработка технологии интеграции расписания учебных занятий в единое информационное пространство ТГУ.

Для достижения поставленной цели в работе были решены следующие задачи:

- анализ технологии интеграции системы составления расписание в единое информационное пространство ТГУ;
- проектирование технологии интеграции;
- программная реализация технологии интеграции.

Работа состоит из введения, трех глав, заключения и списка использованной литературы. Разработанная, в ходе выпускной квалификационной работы, технология интеграции может быть использована для автоматизации процесса публикации расписания в Тольяттинском государственном университете.

В данной работе содержится 52 страницы, на которых 5 таблиц, 26 рисунок, 20 источников используемой литературы, одно приложение.

Оглавление

ВВЕДЕНИЕ

Использование информационных систем в различных сферах современного мира непрерывно развивается. Прогресс не обошел стороной и учебный процесс, в котором информационные системы занимают не самую последнюю позицию.

Электронные дневники, зачетки, ведомости, расписание – это информационные системы, помогающие достичь наивысших высот в современном образовании и сделать обучающий процесс максимально комфортным для обучающихся и преподавателей.

В настоящее время проблема эффективного и быстрого информирования о начале, изменении или переносе учебных занятий очень актуальна. Время, при современном темпе жизни человека, ценится очень высоко. Своевременное информирование об учебных занятиях позволяет сэкономить большое количество времени, которое человек сможет потратить более рационально, нежели чем на ожидание события, которого может и не произойти.

Поэтому **актуальность работы** заключается в том, что автоматизация публикации расписания учебных занятий значительно сокращает время, необходимое на ручную публикацию, и позволяет предоставлять информацию в требуемом формате на требуемый носитель информации (телефон, планшет, персональный компьютер).

Объектом работы является технология интеграции расписания учебных занятий в единое информационное пространство ТГУ.

Предметом работы является автоматизация процесса публикации расписания учебных занятий в единое информационное пространство ТГУ.

Целью выпускной квалификационной работы является разработка технологии интеграции расписания учебных занятий в единое информационное пространство ТГУ.

Для достижения цели необходимо выполнить следующие **задачи:**

- проанализировать технологию интеграции системы составления расписание в единое информационное пространство ТГУ;

- спроектировать технологию интеграции;

- реализовать технологию интеграции.

В ходе выполнения выпускной квалификационной работы будет создана система автоматической публикации расписания учебных занятий в виде серверного приложения, которое позволит:

- размещать информацию об учебных занятиях на портале Тольяттинского государственного университета;

- вносить изменения в расписание учебных занятий на портале Тольяттинского государственного университета.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка литературы и приложений.

Во введении описывается актуальность проводимого исследования, формируется цель и ставятся задачи, которые необходимо решить для достижения цели.

В первой главе описывается анализ технология интеграции системы составления расписания учебных занятий в единое информационное пространство ТГУ, подлежащей автоматизации.

Во второй главе проводится проектирование технологии интеграции системы составления расписания в единое информационное пространство ТГУ.

В третьей главе проводится кодирование технологии интеграции системы составления расписания в единое информационное пространство ТГУ.

В заключении приводятся основные выводы по работе, достигнутые в ходе выполнения выпускной квалификационной работы.

Глава 1 Анализ технологии интегрирования расписания в единое информационное пространство ТГУ

1.1 Описание структуры организации Тольяттинский государственный университет

Преобразование российского высшего образования началось в 1992 г. с принятия Федерального Закона «Об образовании» [8], который узаконил новые для нас понятия: бакалавриат, магистратура, многоуровневая система высшего образования. На сегодняшний день высшее образование Российской Федерации – это двухуровневая система подготовки. Первый уровень – бакалавриат (срок обучения 4 года), второй – магистратура (срок обучения 2 года). Наряду с этим существует образование по специальностям, так называемый «специалитет» (нормативный срок обучения 5 лет с получением квалификации «дипломированный специалист»).

29 мая 2001г. по инициативе первого мэра Тольятти Сергея Федоровича Жилкина и в соответствии с распоряжением Правительства Российской Федерации Тольяттинский политехнический институт, образованный на базе филиала Куйбышевского индустриального института в октябре 1966 года, был объединен с Тольяттинским филиалом Самарского государственного педагогического университета [10]. На базе этих двух высших учебных заведений был создан новый вуз - Тольяттинский государственный университет (ТГУ).

Тольяттинский государственный университет является высшим учебным заведением, которое не малозначимо для территории Поволжья и которое вносит огромный вклад в развитие автомобильной. Одна из главных целей ТГУ – подготовка высококвалифицированных и профессионально компетентных людей, которые смогут обеспечить устойчивое развитие своего города, региона и страны. Структура университета приведена на рисунке 1.1.

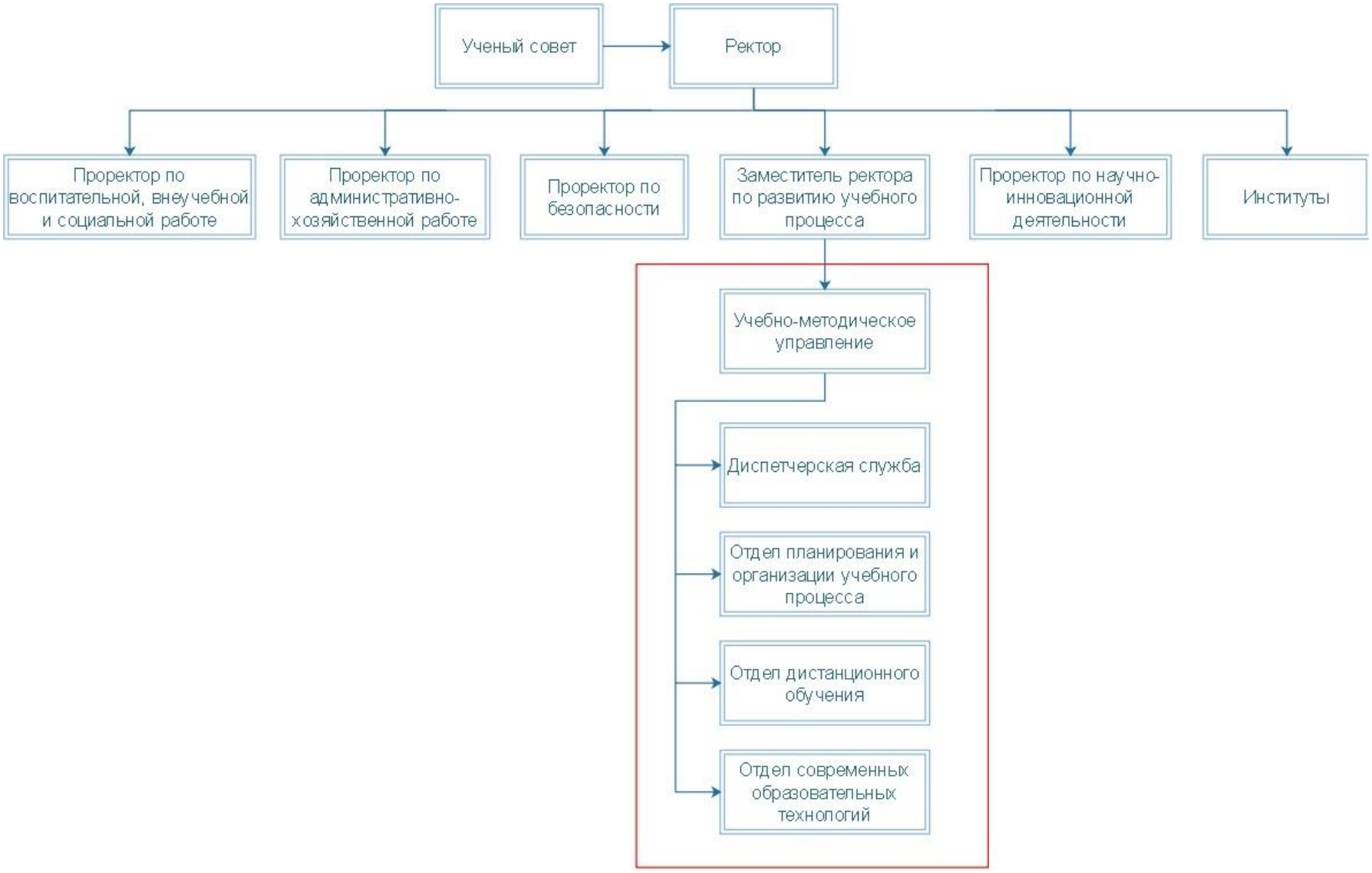

Рисунок 1.1 – Структура Тольяттинского государственного университета

Расписание учебных занятий является неотъемлемой частью учебного процесса. Оно позволяет выполнять в полном объеме учебный план через проведение учебных занятий, что обеспечивает выполнение главной цели Тольяттинского государственного университета, а именно подготовка высокообразованных специалистов по перечню направлений подготовки, специальностей, утвержденных в ТГУ.

Расписание учебных занятий Тольяттинского государственного университета разрабатывается и публикуется отделом учебно-методического управления – диспетчерская служба, и подтверждается заместителем ректора по развитию учебного процесса.

Последовательность действий, которые необходимо выполнить диспетчеру Диспетчерской службы, для публикации расписания учебных занятий на портале Тольяттинского государственного университета, изображена на рисунке 1.2.

Сотрудник диспетчерской службы выгружает из базы данных «Галактика Расписание учебных занятий» (РУЗ) готовое сформированное расписание для каждого института и при помощи HTML редактора FCKeditor (рис. 1.3) размещает на портале ТГУ в разделе «Расписание» (рис. 1.4) расписание учебных занятий, доступное для скачивания в формате документа электронного табличного процессора MS Excel 2010 [5].

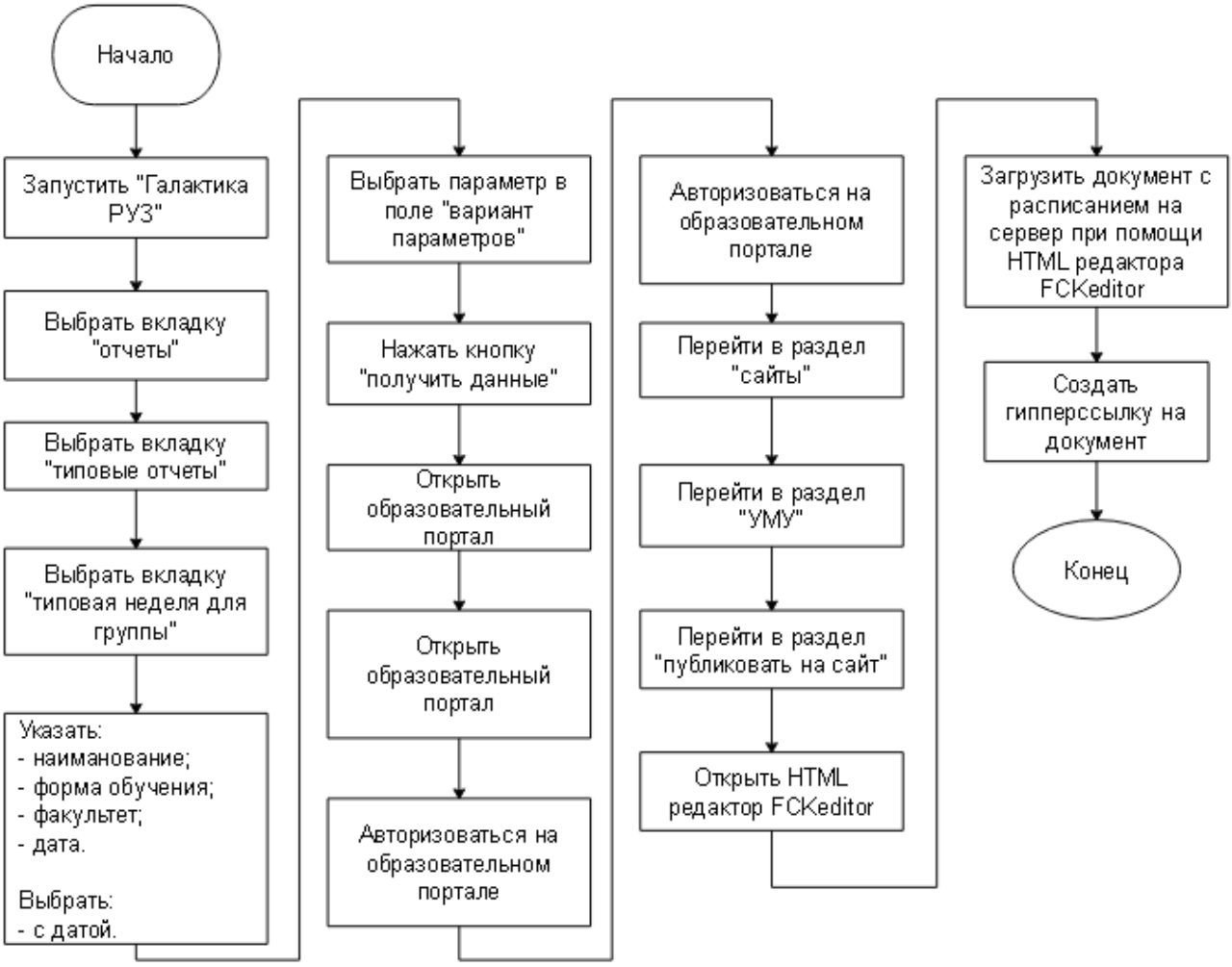

Рисунок 1.2 – Последовательность действий диспетчера для публикации расписания

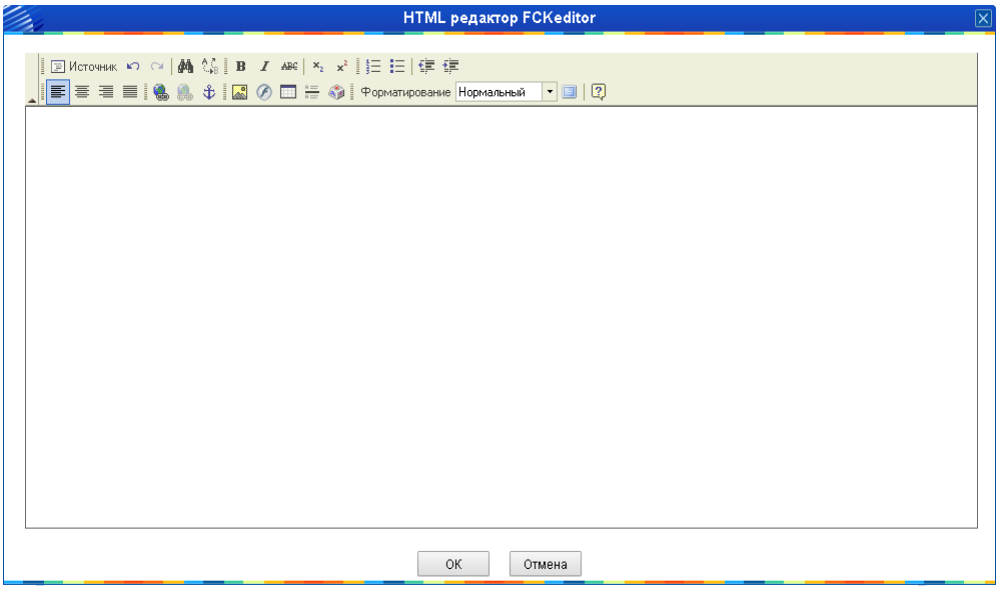

Рисунок 1.3 – HTML редактор FCKeditor

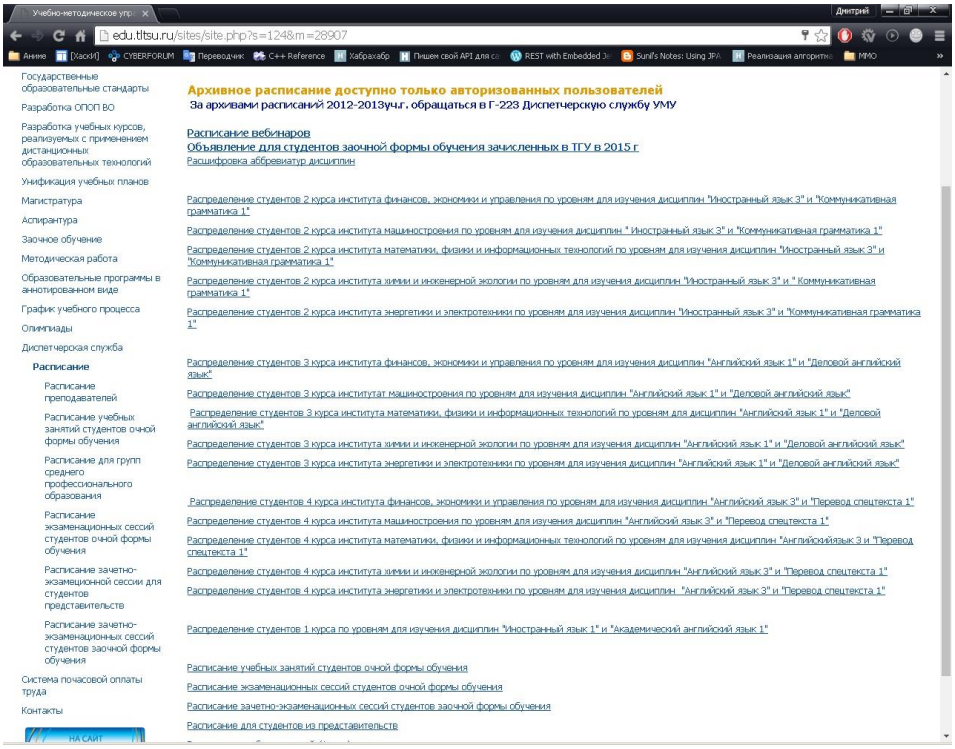

Рисунок 1.4 – Раздел «Расписание» на образовательном портале ТГУ

Публикация расписания длительный процесс, который отнимает у диспетчера целый рабочий день, при этом, так как вся работа проводится в ручном режиме, не исключены ошибки при публикации документов с расписанием на портале ТГУ.

1.2 Составление модели публикации расписания учебных занятий в единое информационное пространство ТГУ «как есть»

Перед непосредственным составлением модели интеграции системы составления расписания учебных занятий в единое информационное пространство ТГУ, необходимо описать существующую систему публикации расписания учебных занятий в едином информационном пространстве ТГУ с позиции «как есть». Для этого составим функциональную модель системы.

Для моделирования системы необходимо определить цель и точку зрения на модель.

Модель разрабатываемой системы призвана ответить на вопросы:

- как происходит публикация расписания учебных занятий в единое информационное пространство ТГУ?

- какие результирующие документы мы получаем?

Проанализировав данный список вопросов, мы можем определить цель модели: Определить действия, необходимые для публикации расписания учебных занятий в едином информационном пространстве ТГУ.

Выберем точку зрения модели, которая необходима для моделирования системы, из списка позиций:

- заместитель директора по учебно-методической работе;
- заведующий кафедры;
- диспетчер;
- начальник учебной части;
- методист учебной части;
- методист кафедры;
- преподаватель;
- студент.

Для анализа системы была выбрана точка зрения «диспетчер», так как она наиболее полно охватывает процесс интеграции системы составления расписания в единое информационное пространство ТГУ.

Составим контекстную диаграмму А-0, которая будет определять основные процессы или подсистемы ИС, внешние входы и выходы, механизмы и управляющие элементы. Модель IDEF0 «как есть» служит для выявления основных действий, совершаемых диспетчером при публикации расписания учебных занятий в единое информационное пространство (ЕИП) ТГУ.

Полученное обновленное расписание и ссылки на обновленное расписание используются сотрудниками ТГУ, а также студентами, при планировании учебного процесса и распределении своего личного времени.

На рисунке 1.5 представлена контекстная диаграмма публикация РУЗ в ЕИП ТГУ.

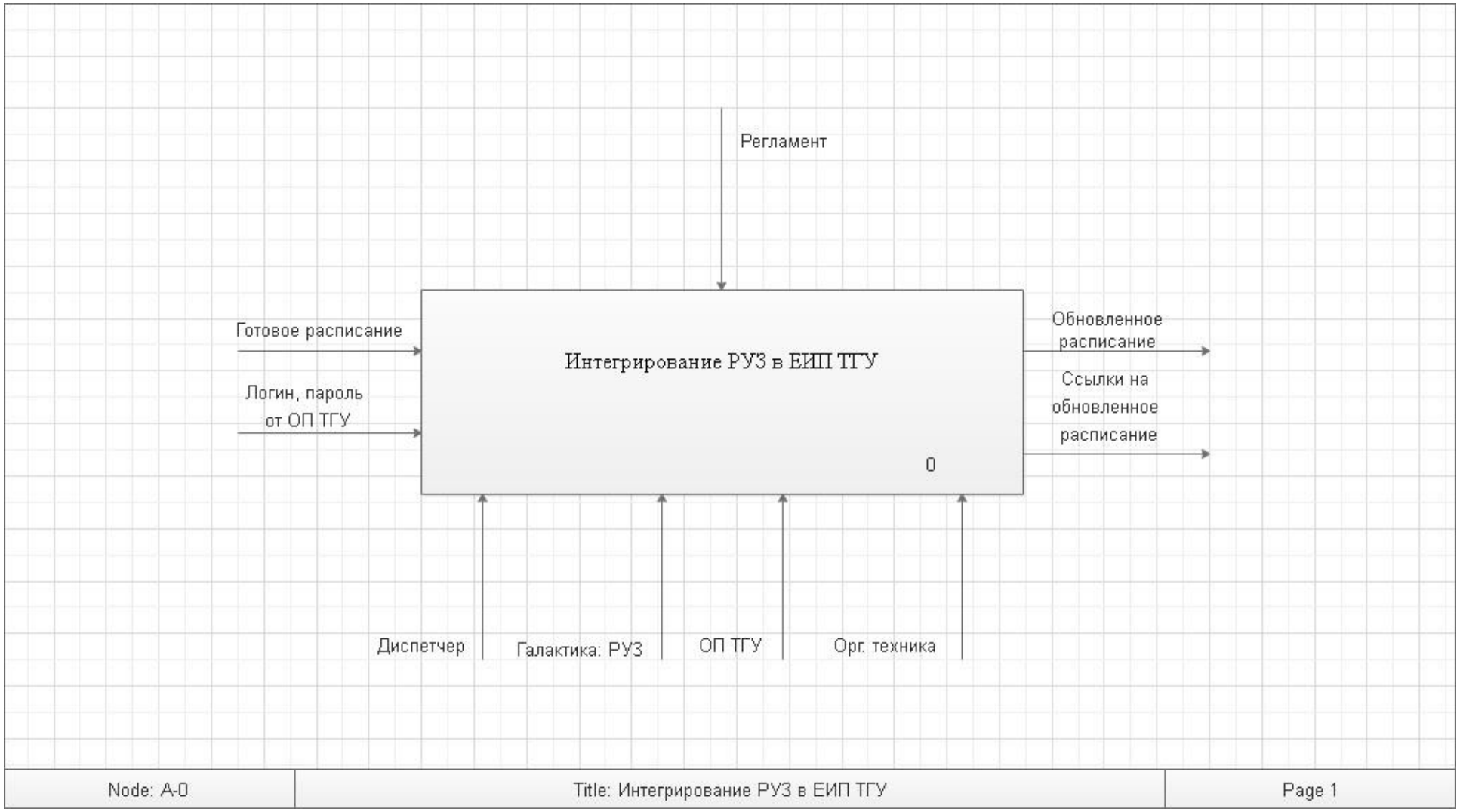

Рисунок 1.5 – Контекстная диаграмма процесса публикации РУЗ в ЕИП ТГУ. Модель IDEF0 «как есть»

Детализирование контекстной диаграммы осуществляется при помощи диаграмм нижнего уровня. Этот процесс декомпозиции продолжается до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процессы становятся элементарными и детализировать их далее невозможно.

Произведя разбиение работы «Интегрирование РУЗ в ЕИП ТГУ» на более мелкие составляющие, получаем декомпозицию контекстной диаграммы (рис. 1.6).

Расписание учебных занятий Тольяттинского государственного университета разрабатывается отделом учебно-методического управления, диспетчерской службой и утверждается заместителем ректора по развитию учебного процесса. При составлении расписания учебных занятий используется технология построения, встроенная в систему «Галактика Расписание учебных занятий», которая позволяет исключать ошибки, связанные с человеческим фактором, при составлении расписания учебных занятий для ТГУ. Данная часть процесса составления расписания учебных занятий может являться темой отдельного проекта и, в виду своей сложности, в рамках данного проектирования рассматриваться не будет. Поэтому данные получаемые вследствие работы системы «Галактика Расписание учебных занятий» выступают в качестве входных данных.

Для публикации готового расписания учебных занятий, в едином информационном пространстве ТГУ, полученном при работе системы «Галактика Расписание учебных занятий», экспортируются в MS Excel 2010 сотрудником диспетчерской службы. Пример расписания учебных занятий (РУЗ) приведён на рисунке 1.7.

Процесс «Публикации РУЗ в ЕИП ТГУ» состоит в последовательном совершении определенных действий сотрудником диспетчерской службы, которые должен выполнить диспетчер для публикации нового или обновленного расписания учебных занятий в единое информационное пространство ТГУ для дальнейшего активного использования его в учебном процессе.

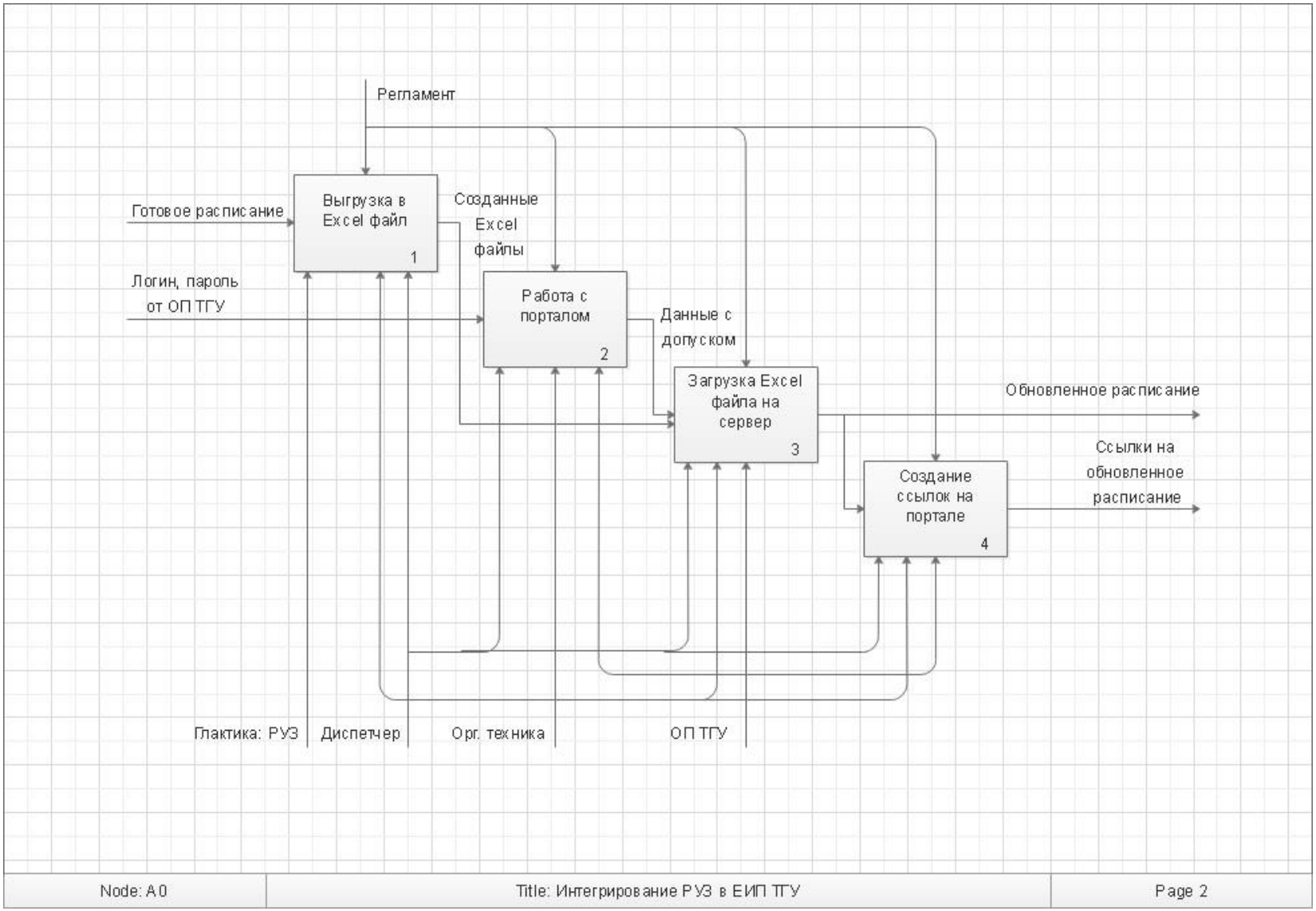

Рисунок 1.6 – Декомпозиция работы «Публикация РУЗ в ЕИП ТГУ». Модель IDEF0 «как есть»

	1	2	3	4	5	6	7	8	9	1
9	2-я	Математический анализ 4 /Математический анализ/ (Лек) Дорофеев С.Н. МИб-1301 УЛК-302 [10:15-11:45]	Психология 1 /Психология/ (Пр) Бергис Т.А. МИб-1301 УЛК-506 [10:15-11:45]	Геометрия 2 /Геометрия/ (Лек) Дроздов Н.А. МИб-1301 УЛК-305 [10:15-11:45]	Английский язык 2 /Английский язык/ (Пр) Анисифорова Т.С. МИб-1301 (9чел) УЛК-618 [10:15-11:45] Английский язык 2 /Английский язык/ (Пр) Косс Е.В. МИб-1301 ПИб-1301 ПИб-1302 ПМИб-1301 ПМИб-1302 (№1001531) УЛК-610 [10:15-11:45] Английский язык 2 /Английский язык/ (Пр) Пушкарёв О.Ю. ПИб-1331 МИб-1301 ПИб-1301 ПИб-1302 (№1001513) УЛК-719 [10:15-11:45]	Безопасность жизнедеятельности (Пр) Гуляев В.А. МИб-1301 Г-330 [10:15-11:45]		2-я		
10	3-я	Математический анализ 4 /Математический анализ/ (Пр) Дорофеев С.Н. МИб-1301 УЛК-302 [12:45-14:15]	Физическая культура (Пр) Физкультура И. Физкультура 3 курс ИМФиИТ СПОРТ-28 [12:45-14:15]	Геометрия 2 /Геометрия/ (Пр) Дроздов Н.А. МИб-1301 УЛК-305 [12:45-14:15]	Алгебра 2 /Алгебра/ (Лек) Симонова Н.С. МИб-1301 УЛК-413 [12:45-14:15]	Физическая культура (Пр) Физкультура И. Физкультура 3 курс ИМФиИТ СПОРТ-28 [12:45-14:15]		3-я		
	4-я	Математический анализ 4 /Математический анализ/ (Пр) Дорофеев С.Н. МИб-1301 УЛК-418 [14:30-16:00]		Геометрия 2 /Геометрия/ (Пр) Дроздов Н.А. МИб-1301 УЛК-305 [14:30-16:00]	Алгебра 2 /Алгебра/ (Пр) Симонова Н.С. МИб-1301 УЛК-505 [14:30-16:00]	Деловой английский язык 2 (Пр) Москалюк А.В. МИб-1301 (9чел) УЛК-719 [14:30-16:00] Деловой английский язык 2 (Пр) Корнеева Н.А. МИб-1301 ПИб-1301 ПИб-1302 ПМИб-1301 ПМИб-1302 (№1001539) А-307 [14:30-16:00] Деловой английский язык 2		4-я		

1 курс | 2 курс | **3 курс** | 4 курс | 5 курс | Магистратура | Sheet

Готово

Рисунок 1.7 – Пример расписания учебных занятий в MS Excel

Сформированные документы системой «Галактика Расписание учебных занятий» с готовым расписанием, регламентирующие, когда у какой группы будет учебное занятие, необходимо загрузить на образовательный портал. Для этого сотрудник диспетчерской службы заходит на портал под своим логином и паролем (блок 2 на диаграмме А0), после чего ему открывается доступ для загрузки документов при помощи HTML редактора FCKeditor (блок 3 на диаграмме А0).

После загрузки всех документов, диспетчер формирует ссылки на загруженные документы в разделе «Расписание» на образовательном портале (блок 4 на диаграмме А0).

Процессы «Выгрузка в Excel файл», «Загрузка Excel файла на сервер», а также «Создание ссылок на портале» производятся вручную диспетчером. Процесс «Работа с порталом» подразумевает взаимодействие диспетчера (ввод логина и пароля) и образовательного портала (авторизация пользователя).

Таким образом, становятся ясны участники процесса публикации расписания учебных занятий: «Галактика Расписание учебных занятий», сотрудник диспетчерской службы, образовательный портал, орг. техника.

Стоит учитывать тот факт, что расписание на образовательном портале в настоящий момент хранится с использованием табличного процессора MS Excel, поэтому данные из документов могут быть доступны не со всех типов устройств, при этом высока вероятность возникновения ошибки при загрузке и формирование ссылок на документы. Кроме того, при изменении расписания необходимо загрузить новые документы с измененным расписанием и обновить ссылки на них, что делает процесс публикации расписания учебных занятий на образовательном портале довольно трудоёмким и критичным к человеческим ошибкам.

Таким образом, несмотря на существующие достоинства действующей системы публикации расписания на образовательном портале, данный процесс имеет и свои недостатки:

- формирование расписания учебных занятий с последующей

публикацией на образовательном портале занимает большое количество времени. Например, публикация расписания для всего университета на следующую неделю отнимает у сотрудника диспетчерской службы целый рабочий день;

- из-за отсутствия автоматизированной системы публикации возникает большое количество ошибок. Примером этому может послужить ситуация, когда ссылки на документы на образовательном портале перепутали местами;
- способ хранения документов с расписанием делает невозможным его просмотр на устройствах, не поддерживающих формат MS Excel;
- для просмотра расписания учебных занятий необходимо устанавливать дополнительное программное обеспечение (MS Excel).

Перечисленные недостатки существующей системы публикации расписания делают процесс публикации довольно неэффективным. Это является причиной нерационального распределения денежных средств вуза (время, потраченное диспетчером на публикацию может быть использовано более рационально). Учитывая всё выше сказанное, принято решение об постановке задачи на разработку технологии интеграции системы составления расписания в единое информационное пространство ТГУ.

1.3 Составление модели публикации расписания учебных занятий в единое информационное пространство ТГУ «как будет»

Для составления модели публикации расписания учебных занятий в единое информационное пространство ТГУ «как будет» составим функциональную модель системы. Данная модель призвана описать процессы, которые должны протекать в системе публикации расписания учебных занятий, чтобы устранить недостатки существующей системы.

Для моделирования системы необходимо определить цель и точку зрения на модель.

Модель разрабатываемой системы призвана ответить на вопросы:

- как должна происходит публикация расписания учебных занятий в единое информационное пространство ТГУ?

- какие результирующие документы мы получим?

Проанализировав данный список вопросов, мы можем определить цель модели: описать действия, механизмы и процессы, которые должны быть автоматизированы.

Выберем точку зрения модели, которая необходима для моделирования системы, из списка позиций:

- заместитель директора по учебно-методической работе;
- заведующий кафедры;
- диспетчер;
- начальник учебной части;
- методист учебной части;
- разработчик;
- преподаватель;
- студент.

Для анализа системы была выбрана точка зрения «разработчик», так как она наиболее полно охватывает процесс интеграции системы составления расписания в единое информационное пространство ТГУ.

Составим контекстную диаграмму А-0, которая будет определять основные процессы или подсистемы ИС, внешние входы и выходы, механизмы и управляющие элементы (рис. 1.8). Модель IDEF0 «как будет» служит для устранения недостатков существующей технологии интеграции системы составления расписания учебных занятий в единое информационное пространство ТГУ.

В отличие от модели «как есть», в модели «как будет» система «Галактика РУЗ» выступает не в роли механизма, а в роли источника данных, так как данные берутся непосредственно из хранилища данных системы, а не на выходе работы приложения.

Механизм автоматическая система публикации расписания (АСПР) – выступает в роли разрабатываемого приложения, которое будет автоматизировать процесс интеграции системы составления расписания в единое информационное пространство ТГУ.

Произведя разбиение работы «Интегрирование РУЗ в ЕИП ТГУ» на более мелкие составляющие, получаем декомпозицию данной модели (рис. 1.9).

Для того чтобы автоматизировать процесс публикации, процессы «Выгрузка в Excel файл», «Загрузка Excel файла на сервер» и «Создание ссылок на портале» из модели «как есть» были заменены на процесс «Обновление расписания» в модели «как будет» (блок 2 на диаграмме А0). Данный процесс занимается выгрузкой данных из хранилища данных системы «Галактика Расписание учебных занятий» и сохранением их в хранилище данных приложения. Процесс управляется приложением, поэтому механизм «Диспетчер» более не нужен для управления процессом публикации расписания учебных занятий. Данная замена позволила разгрузить диспетчерскую службу, уменьшив трудозатраты сотрудника диспетчерской службы на публикацию, обновление расписания учебных занятий. В свою очередь уменьшение влияния диспетчерской службы на процесс публикации, обновления расписания учебных занятий уменьшает влияние человеческого фактора на появление ошибок [14].

Для того чтобы понять, как именно происходит процесс «Обновление расписания» в модели «как будет» он был декомпозирован. Декомпозиция данного процесса представлена на диаграмме А2 (рис. 1.10).

Процесс обновления расписания выполняется в два этапа:

- запрос данных из системы «Галактика Расписания учебных занятий» (блок 1 на диаграмме А2);
- обновление данных в хранилище данных приложения (блок 2 на диаграмме А2).

Данные процессы выполняются автоматически после получения запроса на обновление расписания учебных занятий.

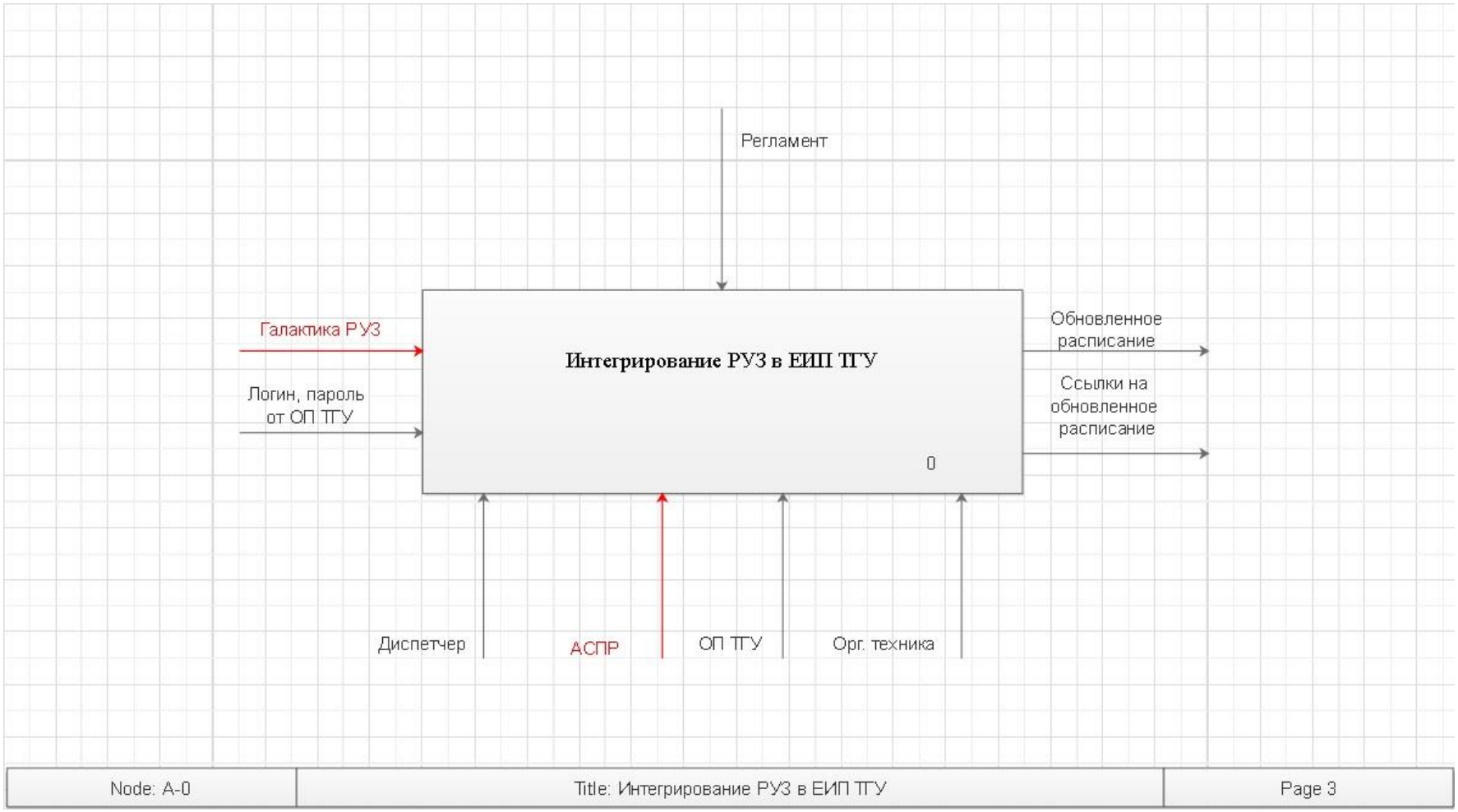

Рисунок 1.8 – Контекстная диаграмма процесса публикация РУЗ в ЕИП ТГУ. Модель IDEF0 «как будет»

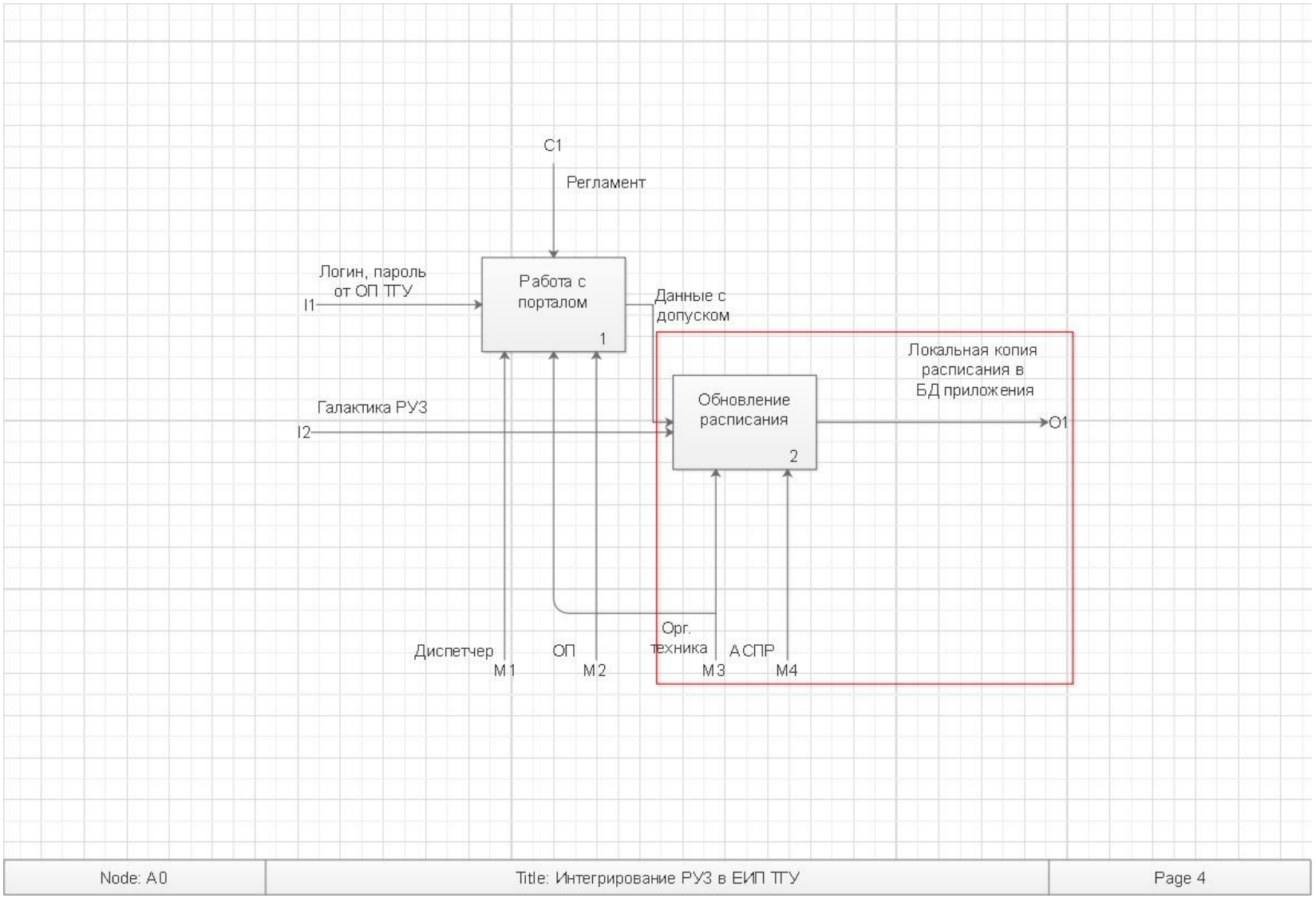

Рисунок 1.9 – Декомпозиция работы «Интегрирование РУЗ в ЕИП ТГУ». Модель IDEF0 «как будет»

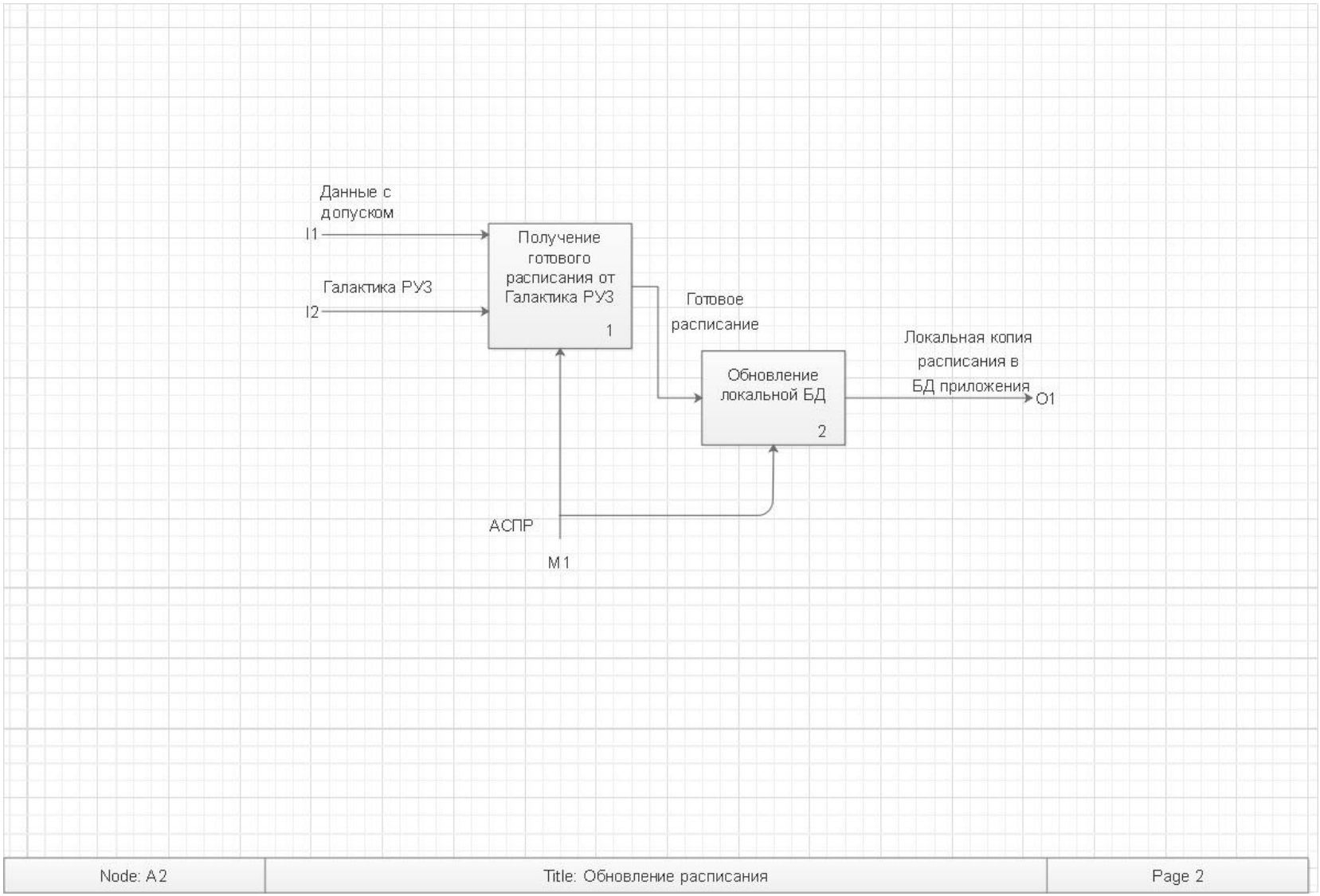

Рисунок 1.10 – Декомпозиция процесса «Обновление расписания»

Данная система позволяет публиковать расписание учебных занятий не только на образовательном портале, но и в других информационных системах, которые в дальнейшем могут быть интегрированы; автоматически публиковать расписание учебных занятий, тем самым сводя влияние человеческого фактора при публикации к минимуму; осуществлять просмотр расписания учебных занятий с любого устройства, с которого можно просматривать образовательный портал, так как старый способ хранения расписания в виде документов в формате MS Excel более не используется. Что в свою очередь локализует необходимость устанавливать дополнительное программное обеспечение для просмотра расписания учебных занятий необходимо устанавливать дополнительное программное обеспечение (MS Excel).

1.4 Выработка требований к разрабатываемой системе

Выработка требований необходима для выявления и классификации требований, которые предъявляются к разрабатываемой информационной системе. Требованиям можно дать определение, как «подробное описание того, что должно быть реализовано». Требования – это основа всех систем [1].

К целевой аудитории разрабатываемой технологии интеграции системы составления расписания в единое информационное пространство ТГУ можно отнести:

- сотрудники Тольяттинского государственного университета;
- студенты;
- пользователи образовательного портала ТГУ.

Разрабатываемая технология интеграции должна представлять собой серверное приложение, располагающееся на сервере образовательного портала ТГУ.

Информация, получаемая на выходе разрабатываемого приложения, должна быть доступна всем пользователям на портале ТГУ в разделе расписание. Непосредственный доступ к приложению должен ограничиваться доступом по локальной сети.

Пользователей разрабатываемого приложения можно разделить на две категории:

- сотрудники диспетчерской службы – занимаются непосредственной работой с приложением (обновление данных приложения);
- пользователи образовательного портала ТГУ.

Пользователи образовательного портала ТГУ должны иметь доступ только к общедоступной информации, получаемой в процессе работы приложения.

Приложение должно:

- автоматически и по требованию обновлять данные приложения из системы «Галактика РУЗ»;
- быть кроссплатформенным.

Данные, получаемые приложением из системы «Галактика РУЗ» должны храниться в системе управления базами данных в структурированном виде.

Для реализации программной части приложения предполагается использовать язык программирования Java [4; 7] и фреймворк для Java-платформы Spring.

В результате анализа предметной области были выделены недостатки существующей системы публикации расписания, составлена модель того, как должна будет выглядеть система, а так же составлены требования к будущей системе публикации расписания в единое информационное пространство ТГУ.

Глава 2 Проектирование технологии интеграции расписания в единое информационное пространство ТГУ

2.1 Архитектура системы публикации расписания

Первым, из самых важных преимуществ архитектуры клиент-сервер, является то, что при использовании данной архитектуры снижается сетевой трафик при выполнении запросов. Это достигается за счет того, что запрос, который посылает клиентское приложение серверу, компилируется и выполняется непосредственно самим сервером и результат передается обратно клиенту.

Вторым, и не менее важным по значимости, преимуществом систем с архитектурой клиент-сервер является возможность хранения бизнес-правил прямо на сервере, что избавляет разработчика от излишнего дублирования программного кода в приложениях, которые используют одну базу данных. Помимо этого, любое редактирование данных может быть произведено только в рамках правил, хранящихся на сервере.

Трехзвенная клиент-серверная архитектура информационной системы включает в себя три основных компонента:

- сервер баз данных, который управляет хранением, доступом, защитой, и резервным копированием данных, а также, в соответствии с бизнес-правилами, отслеживает целостность данных и выполняет запросы клиентских приложений;
- клиентское приложение, в котором содержится основная логика и интерфейс конечного пользователя, выполняющее проверку данных на корректность в соответствии с правилами, посылающее запросы к серверу и получающее, а также обрабатывающее, ответы от него;
- сеть и коммуникационное программное обеспечение, с помощью которого осуществляется взаимодействие между сервером и клиентом посредством сетевых протоколов.

Архитектура разрабатываемого приложения состоит из трех основных слоев:

- представление;

- бизнес-логика;

- управление данными.

Слой представления состоит из одного компонента – браузер, и он представляет собой клиентскую машину, у которой есть выход в интернет.

Слой бизнес-логики включает в себя JAVA EE сервер и разрабатываемое приложение.

Слой управления данными состоит из компонентов СУБД и БД. СУБД включает в себя две системы: Oracle Database, на которой функционирует база данных системы составления расписания «Галактика РУЗ», и СУБД, которая будет использоваться для хранения данных приложения.

Диаграмма компонентов, разрабатываемой технологии интеграции системы составления расписания в единое информационное пространство ТГУ, представлена на рисунке 2.1.

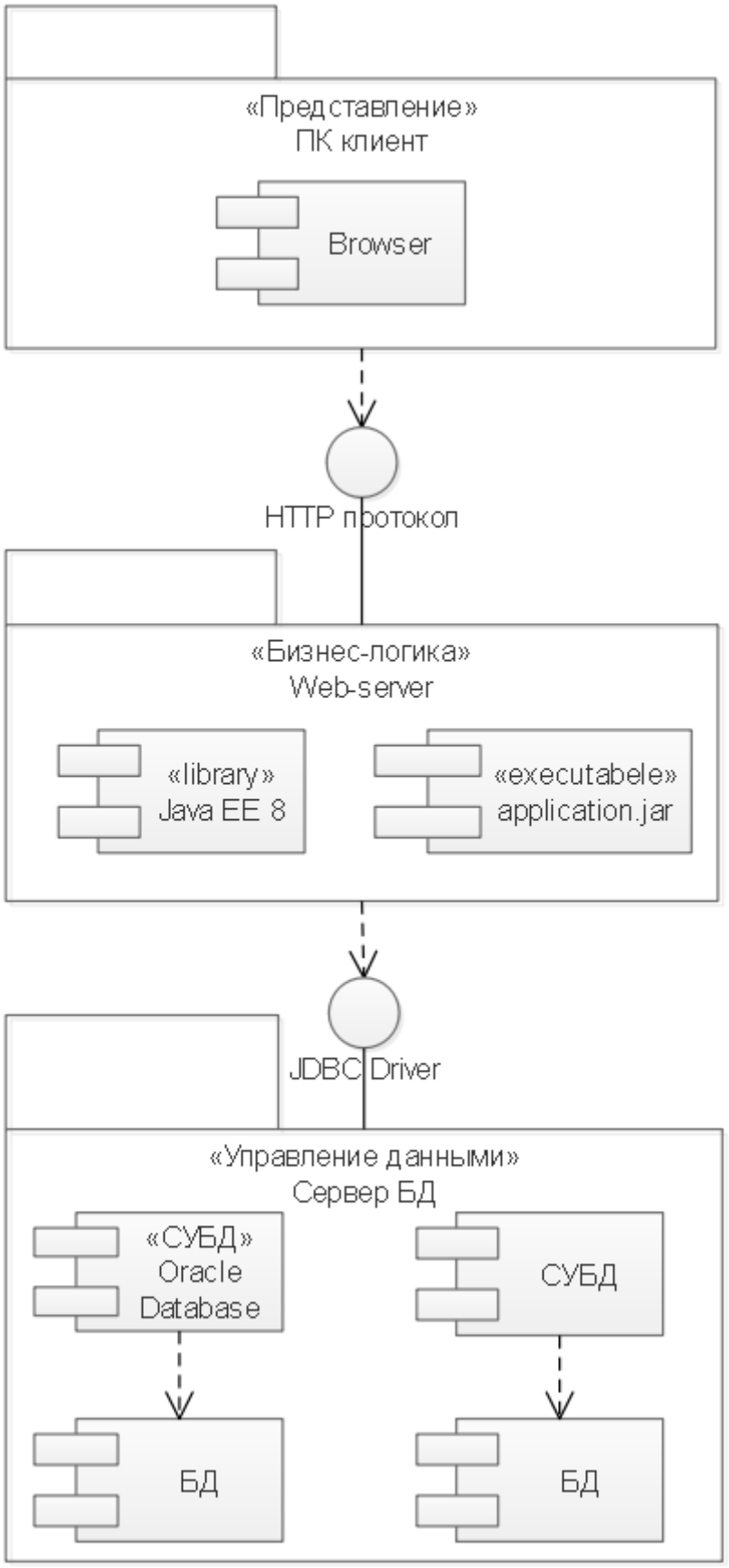

Рисунок 2.1 – Диаграмма компонентов проектируемой системы

В представленной на рисунке 2.1 диаграмме компонентов все функции по управлению данными и бизнес логика реализуются в компоненте «application.jar».

2.2 Разработка диаграммы вариантов использования

Диаграмма вариантов использования (ДВИ) используется в данной выпускной квалификационной работе для того, чтобы смоделировать функциональные требования к проектируемой системе [6].

Так как пользователями системы являются сотрудники диспетчерской службы, то варианты использования данной системы для них только один – обновление расписания.

Актеры. Диспетчер – это любой сотрудник диспетчерской службы, у которого будет доступ к проектируемой системе

Прецеденты. Обновление расписания – единственная функция, которую будет выполнять система. Данная функция интегрирует систему составления расписания в проектируемую систему.

Таблица 2.1 – Описание прецедента Обновление расписания

Прецедент: Обновление расписания
ID: 1
Краткое описание: Реализация обновления функции обновления расписания.
Главные актеры: 1. Сотрудник диспетчерской службы
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе диспетчера
Основной поток: 1. Диспетчер запускает процесс обновления данных 2. Система копирует данные из таблицы базы данных системы составления расписания 3. Система построчно обновляет данные в базе данных приложения 4. Если такая запись существует

Прецедент: Обновление расписания
4.1 Происходит обновление записи 5. Если такой записи не существует 5.1. Происходит добавление записи 6. Система выполняет действия 2-5 по всем имеющимся таблицам, участвующим в процессе публикации расписания
Постусловие: Данные в базе данных приложения обновлены
Альтернативные потоки: Нет

На рисунке 2.2 представлена диаграмма вариантов использования проектируемой системы.

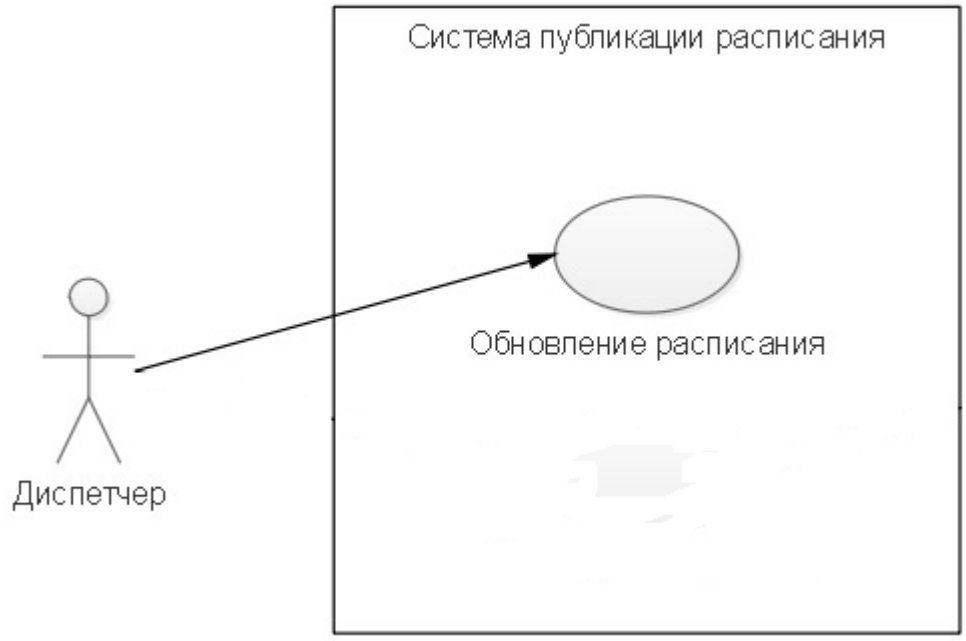

Рисунок 2.2 – Диаграмма вариантов использования проектируемой системы публикации расписания

Диаграмма вариантов использования отображает функциональные возможности проектируемой системы публикации расписания, которые должна выполнять система. Например, актер «диспетчер» инициирует вариант использования «обновление расписания», целью которого является обновление

данных из базы данных системы «Галактика Расписания учебных занятий» в базе данных приложения.

2.3 Разработка диаграммы классов

Разработка вариантов использования позволила выделить классы и определить отношения между ними в соответствии с особенностями предметной области, рассмотренными в п. 1.3. В таблице 2.2 представлены классы и их роли в проектируемой системе [2].

Таблица 2.2 - Описание ролей классов в системе

Название класса	**Роль класса в системе**
Entity	
Auditorium	Содержит список аудиторий со всех корпусов университета. Позволяет выбирать и обновлять их.
Building	Содержит список всех корпусов университета. Позволяет выбирать и обновлять их.
Chair	Содержит список всех кафедр. Позволяет выбирать и обновлять их.
ContentOfSchedule	Содержит список всех учебных занятий. Позволяет выбирать и обновлять их.
Discipline	Содержит список всех дисциплин по всем направлениям обучения университета. Позволяет выбирать и обновлять их.
Group	Содержит список всех учебных групп со всех направлений обучения университета. Позволяет выбирать и обновлять их.
KindOfWork	Содержит список всех типов учебных занятий. Позволяет выбирать и обновлять их.
Lecturer	Содержит список всех преподавателей университета. Позволяет выбирать и обновлять их.
Stream	Содержит список всех учебных потоков по всем учебным направлениям университета. Позволяет выбирать и обновлять их.
SubGroup	Содержит список всех подгрупп со всех направлений университета. Позволяет выбирать и обновлять их.
TypeOfAuditorium	Содержит всех возможных типов аудиторий. Позволяет выбирать и обновлять их.

Продолжение таблицы 2.2

Service	
AuditoriumDAO	Организует доступ к данным, получаемым классом Auditorium из баз данных.
BuildingDAO	Организует доступ к данным, получаемым классом Building из баз данных.
ChairDAO	Организует доступ к данным, получаемым классом Chair из баз данных.
ContentOfScheduleDAO	Организует доступ к данным, получаемым классом ContentOfSchedule из баз данных.
DisciplineDAO	Организует доступ к данным, получаемым классом Discipline из баз данных.
GroupDAO	Организует доступ к данным, получаемым классом Group из баз данных.
KindOfWorkDAO	Организует доступ к данным, получаемым классом KindOfWork из баз данных.
LecturerDAO	Организует доступ к данным, получаемым классом Lecturer из баз данных.
StreamDAO	Организует доступ к данным, получаемым классом Stream из баз данных.
SubGroupDAO	Организует доступ к данным, получаемым классом SubGroup из баз данных.
TypeOfAuditoriumDAO	Организует доступ к данным, получаемым классом TypeOfAuditorium из баз данных.
Mapper	
AuditoriumMapper	Преобразует исходные данные, полученные классом Auditorium.
BuildingMapper	Преобразует исходные данные, полученные классом Building.
ChairMapper	Преобразует исходные данные, полученные классом Chair.
ContentOfScheduleMapper	Преобразует исходные данные, полученные классом ContentOfSchedule.
DisciplineMapper	Преобразует исходные данные, полученные классом Discipline.
GroupMapper	Преобразует исходные данные, полученные классом Group.
KindOfWorkMapper	Преобразует исходные данные, полученные классом KindOfWork.

Продолжение таблицы 2.2

LecturerMapper	Преобразует исходные данные, полученные классом Lecturer.
StreamMapper	Преобразует исходные данные, полученные классом Stream.
SubGroupMapper	Преобразует исходные данные, полученные классом SubGroup.
TypeOfAuditoriumMapper	Преобразует исходные данные, полученные классом TypeOfAuditorium.
Database connection	
MainTemplateJDBC	Осуществляет подключение к базам данных и содержит все Data Access Object.

Как можно увидеть из таблицы 2.2, количество классов в системе слишком велико для того, чтобы представить их всех на диаграмме классов. Поэтому далее будет описываться диаграмма классов только для сущности расписание. Остальные классы системы реализуются по аналогии. На рисунке 2.3 представлен фрагмент UML диаграммы классов проектируемой системы и связи между ними.

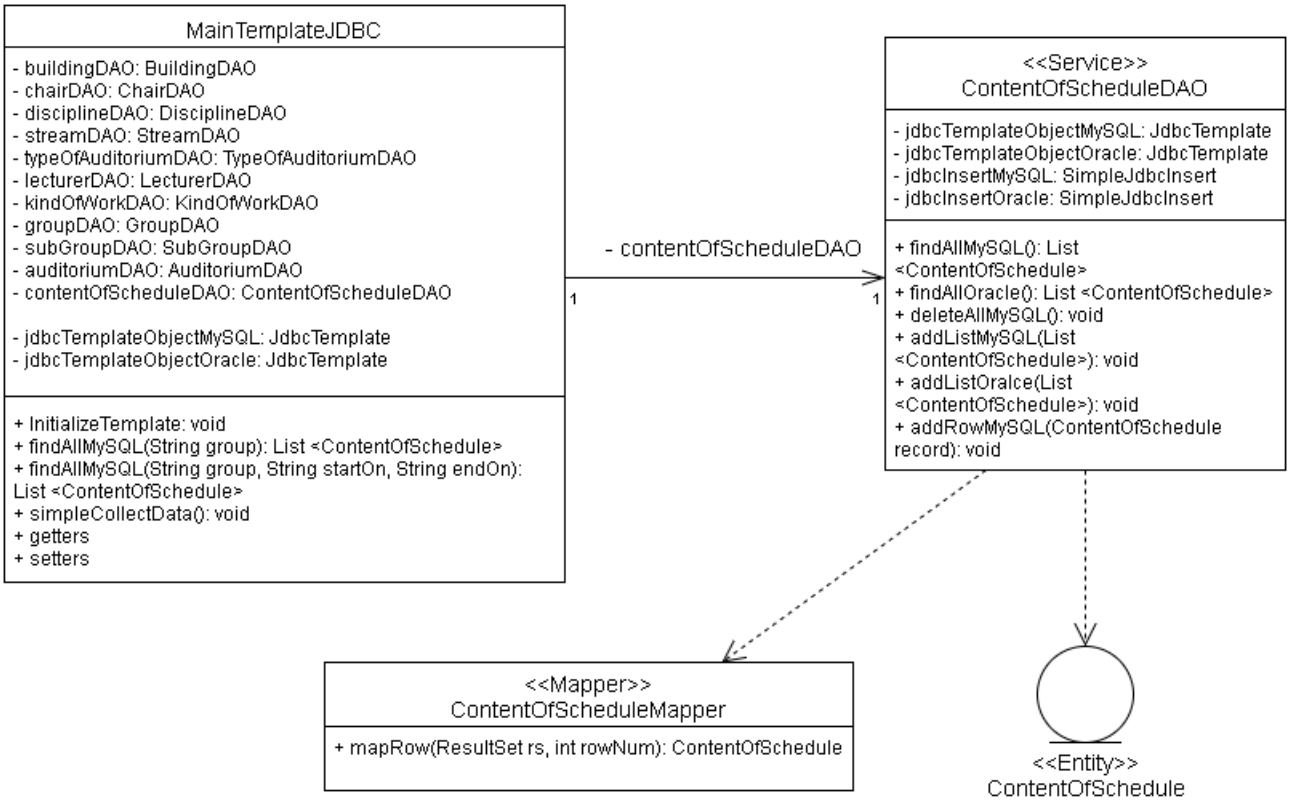

Рисунок 2.3 – Фрагмент UML диаграммы классов проектируемой системы

Прежде всего, необходимо описать класс ContentOfSchedule, который будет имитировать таблицу базы данных. Данный класс является Entity классом и используется классом ContentOfSchedule для хранения данных полученных из базы данных. Спецификация класса ContentOfSchedule представлена в таблице 2.3.

Таблица 2.3 – Спецификация класса ContentOfSchedule

Модификатор доступа	Атрибут	Тип	Назначение
Private	contentOfScheduleID	int	Хранит идентификатор (первичный ключ) таблицы базы данных ContentOfSchedule
Private	StartOn	Timestamp	Хранит значение даты и времени начала занятия
Private	EndOn	Timestamp	Хранит значение даты и времени окончания занятия
Private	ModifiedTime	Timestamp	Хранит дату и время последнего изменения записи таблицы
Private	Discipline	Int	Хранит идентификатор таблицы Discipline (внешний ключ)
Private	KindOfWork	Int	Хранит идентификатор таблицы KindOfWork (внешний ключ)
Private	Lecturer	Int	Хранит идентификатор таблицы Lecturer (внешний ключ)
Private	Auditorium	Int	Хранит идентификатор таблицы Auditorium (внешний ключ)
Private	Stream	Int	Хранит идентификатор таблицы Stream (внешний ключ)
Private	Group	Int	Хранит идентификатор таблицы Group (внешний ключ)
Private	SubGroup	Int	Хранит идентификатор таблицы SubGroup (внешний ключ)

Класс ContentOfScheduleDAO – сервисный класс, который предоставляет доступ к данным базы данных. Данный класс абстрагирует сущности системы (Entity) и делает их отображение на БД, определяет общие методы использования соединения. Спецификация данного класса представлена в таблице 2.4.

Таблица 2.4 – Спецификация класса ContentOfScheduleDAO

Модификатор доступа	Атрибут	Тип	Назначение
Private	jdbcTemplateObjectMySQL	JdbcTemplate	Предоставляет возможность для извлечения данных из базы данных, выполнение вставки, удаления / обновления, а также для извлечения результатов. Скрывает все обработки исключений низкого уровня и упрощает доступ к базе данных
Private	jdbcTemplateObjectOracle	JdbcTemplate	
Private	jdbcInsertMySQL	SimpleJdbcInsert	Позволяет вставлять данные в базу данных MySQL
Private	jdbcInsertOracle	SimpleJdbcInsert	Позволяет вставлять данные в базу данных Oracle

Класс ContentOfScheduleMapper, содержит всего один метод mapRow, который позволят преобразовывать результирующие таблицы в коллекцию List типа ContentOfSchedule. При выполнении запроса в классах типа DAO, класс ContentOfScheduleMapper используется как параметр. Тип, который будет возвращать класс, будет зависеть от типа класса, используемого при включении интерфейса RowMapper.

После того как были реализованы классы доступа и работы с данными, необходимо создать класс подключения к базам данных - MainTemplateJDBC. Данный класс можно разделить на две составные части:

- подключение к базе данных;
- использование классов доступа к данным (DAO).

Классы в проектируемой системе реализуют CRUD технологию, поэтому реализация оставшихся классов делается по аналогии с данной.

2.4 Разработка взаимодействия между элементами системы посредством диаграммы последовательности

В соответствии с функциями системы необходимо реализовать диаграмму последовательности функции обновления расписания.

Для того чтобы сотрудник диспетчерской службы обновил расписание из системы составления расписания учебных занятий «Галактика Расписание учебных занятий», необходимо вызвать функцию update() из html формы на образовательном портале, после чего система выполнит обновление данных из Oracle Database в MySQL.

После того как система получает запрос на обновление расписание она выполняет следующую последовательность действий:

1. Вызывает функцию simpleUpdate(), которая запускает процесс обновления.
2. Запрашивает данные по таблице из Oracle Database.
3. Сравнивает полученные данные с данными хранящимися в MySQL.
4. Обновляет записи, которые не совпали (были изменены) или добавлять запись в таблицу, если такой записи не существует.
5. Повторяет пункты 3-4 для всех таблиц базы данных.

Диаграмма последовательности функции обновления расписания представлена на рисунке 2.4.

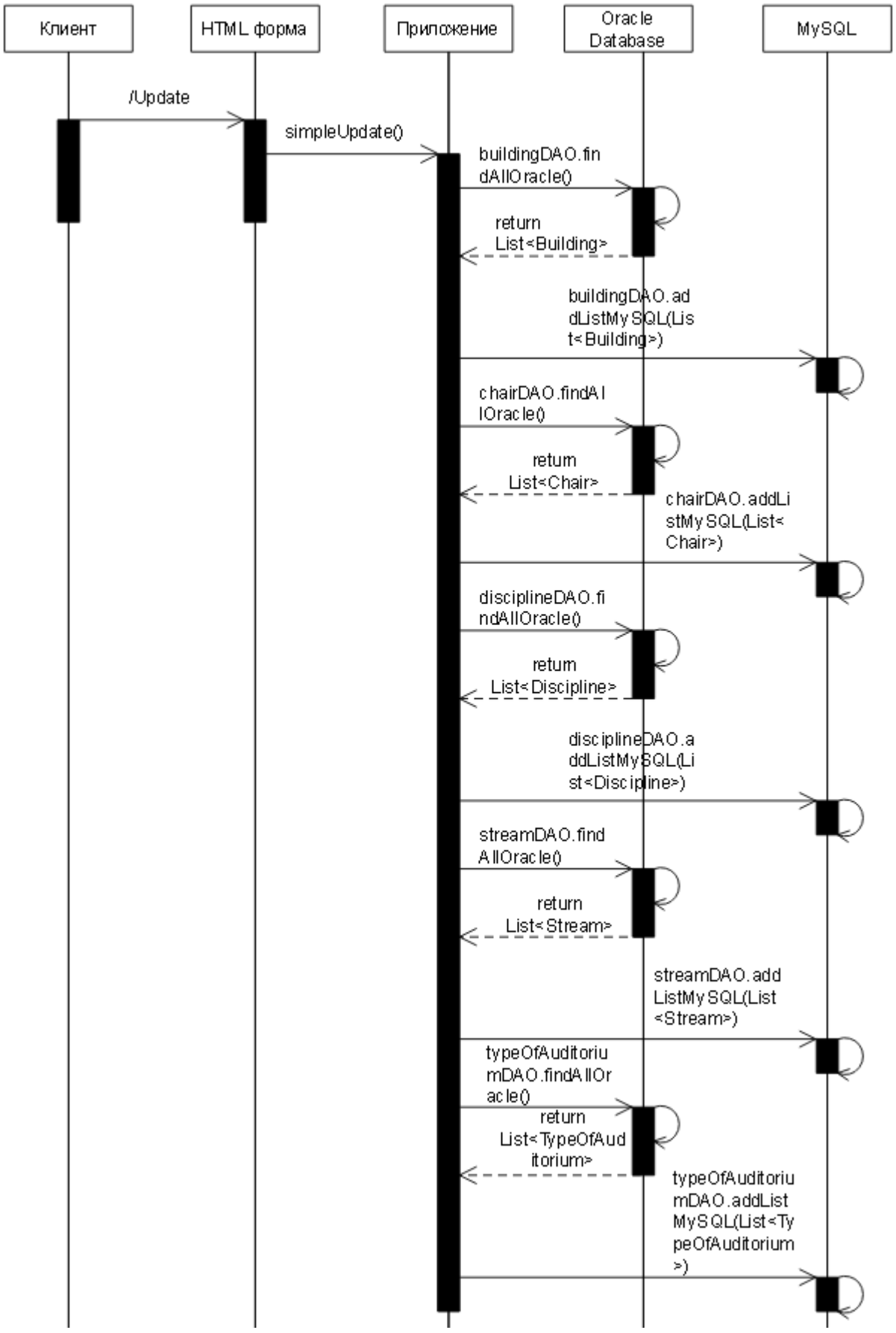

Рисунок 2.4 – Фрагмент диаграммы последовательности функции обновления расписания

Построенная диаграмма последовательности отображает динамический аспект объекта автоматизации и позволяет определить порядок реализации функций, что в свою очередь позволяет разработчику закодировать систему так, как она была задумана.

2.5 Разработка концептуальной модели данных

Описание занятия в расписании учебных занятий содержит 7 основных полей:

- дата и время начала занятия;
- дата и время окончания занятия;
- название дисциплины;
- тип занятия (лекция, лабораторная работа и т.д.);
- фамилия имя отчество преподавателя;
- номер и аббревиатура аудитории;
- поток, если занятие для всего потока;
- группа, если занятие для учебной группы;
- подгруппа, если занятие для подгруппы.

Все поля, кроме даты и времени начала, и окончания занятия, являются списочными, т.е. значение для этих полей выбирается из определенных списков. Поля «Аудитория» и «Подгруппа» являются составными.

Поле «Аудитория» состоит из сущностей «Тип аудитории» и «Строение» (аббревиатура).

Поле «Подгруппа» состоит из сущностей «Группа». Несмотря на то, что в расписания учебных занятий уже содержится сущность «Группа», наличие сущности «Подгруппа» позволяет избежать дублирования данных и тем самым уменьшить объем базы данных.

Помимо перечисленных выше сущностей, есть сущность «Кафедра», в которой содержится сущность «Преподаватель». Данная сущность позволит в будущем просматривать расписание занятий преподавателей на конкретной кафедре. Например, если подгруппа состоит из четырех групп, то для хранения

этой информации в таблице с расписанием необходима только одна запись, а не четыре.

На рисунке 2.5 представлена ER-модель данных проектируемой системы публикации расписания.

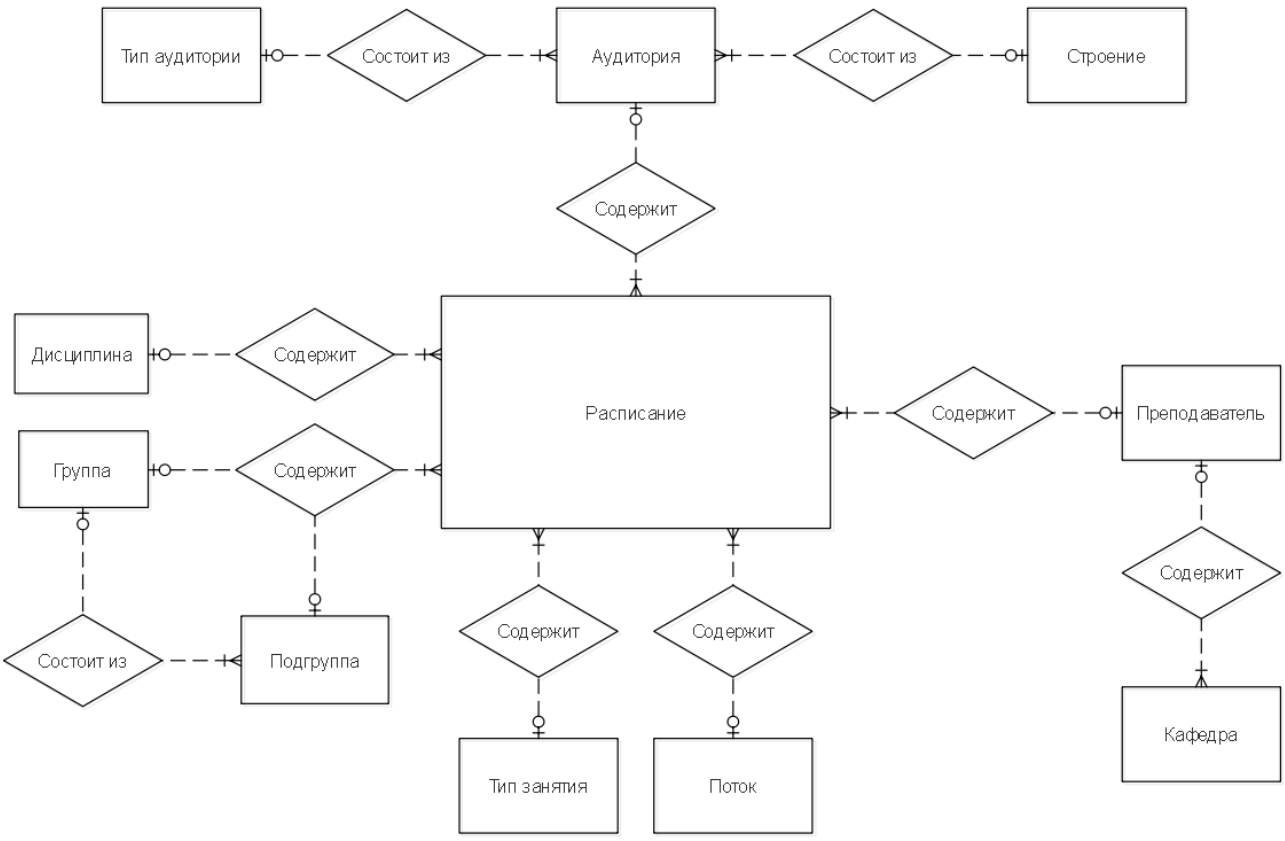

Рисунок 2.5 – ER-модель данных

Данная ER-модель позволяет выделить важнейшие сущности в проектируемой системе и связи между ними.

2.6 Моделирование логического представления базы данных

Логическая модель не имеет привязки к какой-то конкретной системе управления базами данных. Более того, выражение логической модели данных средствами реляционной модели не является обязательным. Одним из основных средств разработки логической модели данных, на текущий момент, являются всевозможные варианты UML-диаграмм [12].

Концептуальная схема переведена в логическую модель БД и представлена на рисунке 2.6 в виде диаграммы сущностных классов в методологии UML.

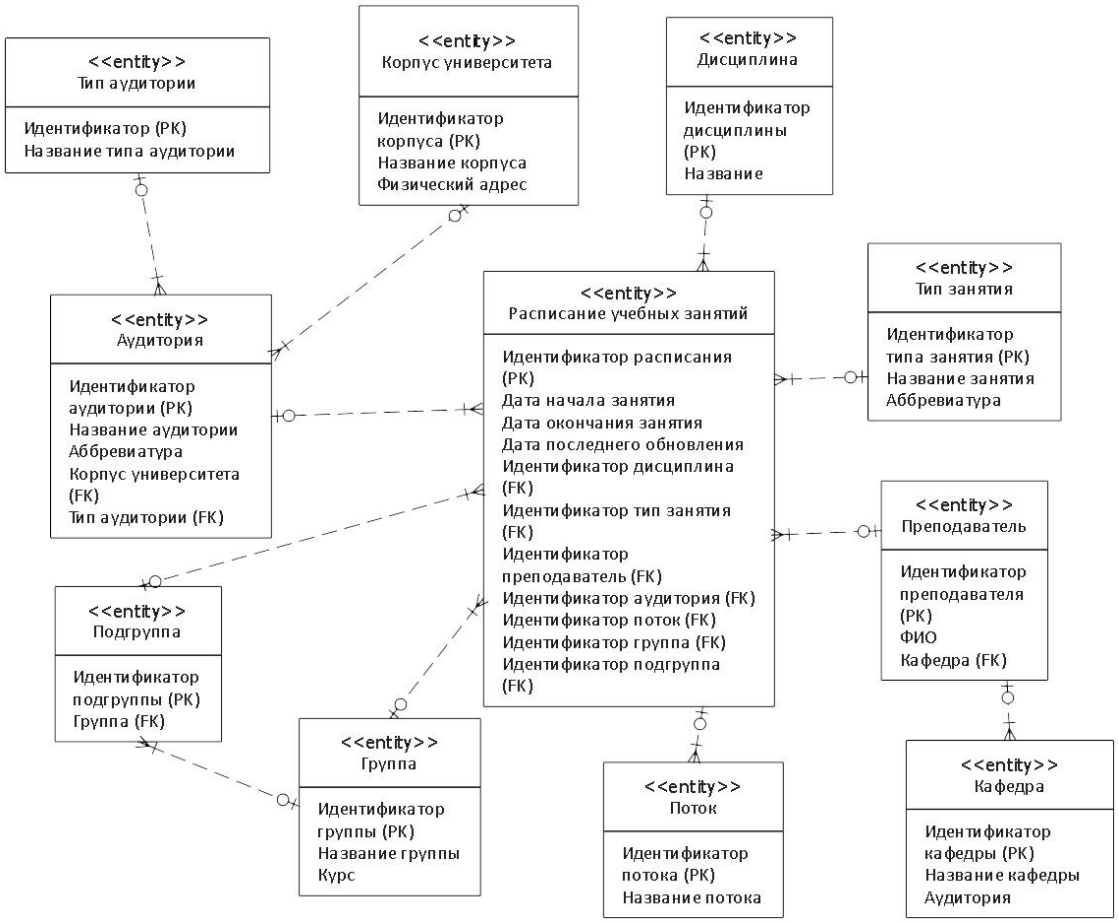

Рисунок 2.6 – Логическая модель данных

Логическая модель данных, представленная на рисунке 2.2, является начальным прототипом базы данных, которая в дальнейшем будет создана.

2.7 Выбор системы управления базами данных

При выборе СУБД можно выделить основные критерии, с помощью которых будет производиться выбор в зависимости от целей, которые будет выполнять данная система. Условно, разделим их на несколько групп:

- моделирование данных;
- производительность;
- надежность;

- степень владения разработчиком.

Моделирование данных. В моделирование данных входят такие параметры, как: используемая модель данных (иерархическая, сетевая, реляционная, объектно-реляционная и объектная), триггеры и хранимые процедуры, типы данных предусмотренные СУБД, реализация языка запросов, масштабируемость.

Надежность. К надежности можно отнести следующие особенности СУБД: восстановление системы после сбоев, резервное копирование [13].

Степень владения разработчиком. Не маловажным фактором при выборе СУБД является степень владения той или иной СУБД разработчиком, что позволит использовать все особенности СУБД по максимуму.

Производительность. К производительности СУБД можно отнести то, как она позволяет оптимизировать запросы и какое максимальное количество запросов в секунду она может обработать.

Чтобы определиться с выбором СУБД рассмотрим и сравним их. Оцениваться критерии выбора СУБД будут по пятибалльной шкале.

Таблица 2.5 – Выбор системы управления базами данных

Критерии	Oracle	MySQL	PostgreSQL	SQL Server 2016
Моделирование данных	5	4	4	4
Производительность	5	4	5	4
Надежность	4	4	4	4
Степень владения разработчиком	2	5	3	1
Итог	16	17	16	13

Исходя из сравнения, приведённого в таблице 2.5, для реализации базы данных была выбрана MYSQL, так как она является легкодоступной,

быстродействующей, простой в эксплуатации СУБД. При этом поддерживает большое количество языков программирования и может переноситься на платформы других фирм, что в свою очередь является большим плюсом, потому что в дальнейшем планируется развитие проекта для нужд сторонних организаций [9].

2.8 Физическое проектирование базы данных

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты [13].

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например, реляционной, сетевой или иерархической. Однако, приступая к физическому проектированию базы данных, прежде всего, необходимо выбрать конкретную целевую СУБД [3]. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных [16; 17].

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

В случае реляционной модели данных под этим подразумевается следующее:

- создание набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной логической модели данных;
- определение конкретных структур хранения данных и методов доступа к ним, обеспечивающих оптимальную производительность СУБД;
- разработка средств защиты создаваемой системы.

Проектирование базы данных — это итерационный процесс, который имеет свое начало, но не имеет конца и состоит из бесконечного ряда уточнений. Его следует рассматривать, прежде всего, как процесс познания. Как только проектировщик приходит к пониманию работы предприятия и смысла обрабатываемых данных, а также выражает это понимание средствами выбранной модели данных, приобретенные знания могут показать, что требуется уточнение и в других частях проекта. Особо важную роль в общем процессе успешного создания системы играет концептуальное и логическое проектирование базы данных. Если на этих этапах не удастся получить полное представление о деятельности предприятия, то задача определения всех необходимых пользовательских представлений или обеспечения защиты базы данных становится чрезмерно сложной или даже неосуществимой. К тому же может оказаться затруднительным определение способов физической реализации или достижения приемлемой производительности системы. С другой стороны, способность адаптироваться к изменениям является одним из признаков удачного проекта базы данных [11].

Для проектирования физической модели проектируемой системы конвертируем сущности, полученные на этапе концептуального и логического проектирования, в таблицы базы данных.

Для конвертации из логической модели в физическую использовалось вспомогательное case-средство Navicat и его функция «Model». Данная программа позволяет автоматически сгенерировать базу данных на основе модели данных введенной в «Model».

На рисунке 2.7 представлена физическая модель базы данных проектируемой системы публикации расписания, полученная после конвертации логической модели.

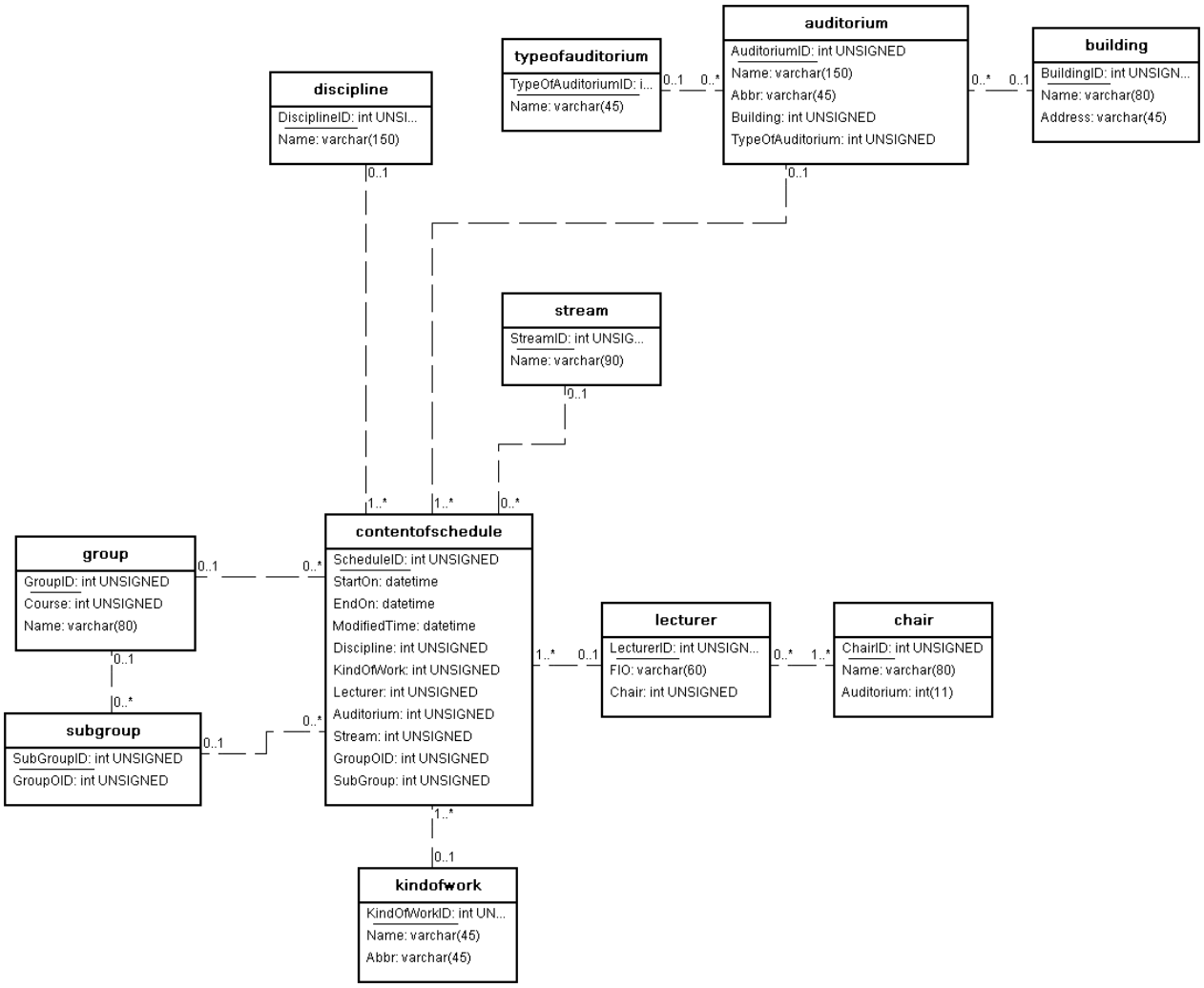

Рисунок 2.7 – Физическая модель базы данных

Данная физическая модель данных описывает реализацию объектов логической модели на уровне объектов базы данных MySQL.

В этой главе была выбрана архитектура проектируемой системы. Так же были составлены диаграммы вариантов использования и последовательности будущей системы публикации расписания.

Глава 3 Кодирование технологии интеграции системы составления расписания в единое информационное пространство ТГУ

3.1 Реализация функций приложения

На этапах анализа и проектирования системы публикации расписания была выделена основная функция системы:

- обновление расписания из системы «Галактика Расписание учебных занятий».

Реализация функции обновления расписания учебных занятий. Реализуем классы, которые будут выбирать данные из базы данных (Auditorium, Building, Chair, ContentOfSchedule, Discipline, Group, KindOfWork, Lecturer, Stream, SubGroup, TypeOfAuditorium). Исходный код данных классов приведен в приложении А. Далее необходимо реализовать mapper-класс для каждого из реализованных классов. На рисунке 3.1 приведена реализация класса ContentOfScheduleMapper.

```
public class ContentOfScheduleMapper implements RowMapper<ContentOfSchedule> {
    public ContentOfSchedule mapRow(ResultSet rs, int rowNum) throws SQLException {
        ContentOfSchedule contentOfSchedule = new ContentOfSchedule();
        contentOfSchedule.setOID(rs.getInt("OID"));
        contentOfSchedule.setDiscipline(rs.getInt("Discipline"));
        contentOfSchedule.setGroup(rs.getInt("GroupOID"));
        contentOfSchedule.setKindOfWork(rs.getInt("KindOfWork"));
        contentOfSchedule.setLecturer(rs.getInt("Lecturer"));
        contentOfSchedule.setStream(rs.getInt("Stream"));
        contentOfSchedule.setSubGroup(rs.getInt("SubGroup"));
        contentOfSchedule.setAuditorium(rs.getInt("Auditorium"));
        contentOfSchedule.setEndOn(rs.getTimestamp("EndOn"));
        contentOfSchedule.setStartOn(rs.getTimestamp("StartOn"));
        contentOfSchedule.setModifiedTime(rs.getTimestamp("ModifiedTime"));

        return contentOfSchedule;
    }
}
```

Рисунок 3.1 – Реализация класса ContentOfScheduleMapper

Данный класс будет использоваться в момент получения данных из таблиц базы данных и возвращать экземпляры класса ContentOfSchedule.

ContentOfScheduleMapper использует setter’ы класса ContentOfSchedule для установки значения полям класса.

Далее реализуем класс доступа к данным. Все классы, относящиеся к данному виду, оканчиваются на постфикс DAO. Приведем пример класса этого вида на примере ContentOfScheduleDAO – класс доступа к данным таблицы contentofschedule. Так как реализация данных классов очень большая, то в пример будет приведена только небольшая часть кода осуществления работы с базой данных (рис. 3.2).

```
public List<CustomContentOfSchedule> findAllMySQL(String GroupName, String StartOn, String EndOn) {
    String sql = "select " +
            "cos.OID," +
            "cos.StartOn," +
            "cos.EndOn," +
            "cos.ModifiedTime," +
            "dis.`Name` as Discipline," +
            "kow.`Name` as KindOfWork," +
            "lect.FIO as Lecturer," +
            "aud.`Abbr` as Auditorium " +
            "from contentofschedule cos " +
            "inner join discipline dis on cos.`Discipline` = dis.`OID` " +
            "inner join kindofwork kow on cos.`KindOfWork` = kow.`OID` " +
            "inner join lecturer lect on cos.`Lecturer` = lect.`OID` " +
            "inner join auditorium aud  on cos.`Auditorium` = aud.`OID` " +
            "where (GroupOID = (select OID from `group` where `name` like '%" + GroupName + "%' limit 1) " +
            "or subgroup in (select OID from `subgroup` where `GroupOID` in (select OID from `group` where `name` like '%" + GroupName +
            "or stream in (select OID from `stream` where `Name` like '%" + GroupName + "%')) " +
            "and DATE(StartOn) >= '" + StartOn+ "' " +
            "and DATE(EndOn) <= '" + EndOn + "' " +
            ";";
    return this.jdbcTemplateObjectMySQL.query(sql, new CustomContentOfScheduleMapper());
}

public List<ContentOfSchedule> findAllOracle() {
    return this.jdbcTemplateObjectOracle.query("select OID," +
            "\"StartOn\"," +
            "\"EndOn\"," +
            "\"ModifiedTime\"," +
            "\"Discipline\"," +
            "\"KindOfWork\"," +
            "\"Lecturer\"," +
            "\"Auditorium\"," +
            "\"Stream\"," +
            "\"Group\" as GroupOID," +
            "\"SubGroup\" from \"ContentOfSchedule\"", new ContentOfScheduleMapper());
}

public void deleteAllMySQL() {
    this.jdbcTemplateObjectMySQL.execute("DELETE FROM contentofschedule WHERE OID > 0");
}

public void addListMySQL(List<ContentOfSchedule> list) {
    for (ContentOfSchedule entry : list) {
        this.addRowMySQL(entry);
    }
}
```

Рисунок 3.2 – Пример реализации классов DAO

Так же, как и классы-mapper, классы DAO создаются для каждого класса, осуществляющего выборку данных из базы данных.

За соединение с базами данных отвечает класс MainTemplateJDBC. Данный класс отвечает за функционирование функции системы публикации расписания – обновление расписания учебных занятий из системы «Галактика Расписание учебных занятий». На рисунке 3.3 приведен метод класса MainTemplateJDBC, который осуществляет обновление данных.

```
public void simpleUpdate(){

    //Building Update
    List<Building> building = buildingDAO.findAllOracle();

    for(int i = 0; i < building.size(); i++){
        List<Building> check = buildingDAO.checkUpdate(building.get(i));
        if(check.isEmpty()){
            buildingDAO.addRowMySQL(building.get(i));
        }else if(!building.get(i).equals(check.get(0))){
            buildingDAO.updateRowMySQL(building.get(i));
        }
    }

    //Chair Update
    List<Chair> chair = chairDAO.findAllOracle();

    for(int i = 0; i < chair.size(); i++){
        List<Chair> check = chairDAO.checkUpdate(chair.get(i));
        if(check.isEmpty()){
            chairDAO.addRowMySQL(chair.get(i));
        }else if(!chair.get(i).equals(check.get(0))){
            chairDAO.updateRowMySQL(chair.get(i));
        }
    }

    //Discipline Update
    List<Discipline> discipline = disciplineDAO.findAllOracle();

    for(int i = 0; i < discipline.size(); i++){
        List<Discipline> check = disciplineDAO.checkUpdate(discipline.get(i));
        if(check.isEmpty()){
            disciplineDAO.addRowMySQL(discipline.get(i));
        }else if(!discipline.get(i).equals(check.get(0))){
            disciplineDAO.updateRowMySQL(discipline.get(i));
```

Рисунок 3.3 – Фрагмент реализации метода обновления данных

Данный метод запрашивает данные из базы данных Oracle Database системы «Галактика Расписание учебных занятий», получает коллекцию данных каждой таблицы и при помощи метода checkUpdate () проверяет была

ли изменена запись. Если запись была изменена, тогда происходит ее обновление при помощи вызова updateRowMySQL. Если же такая запись не была найдена, тогда запись добавляется в базу при помощи вызова метода addRowMySQL.

Для реализации данной функции почти все готово. Осталось реализовать контроллер, который при помощи аннотации @RequestMapping позволяет задать методам контроллера адреса, по которым они будут доступны для клиента. На рисунке 3.4 представлена реализация контроллера DataResponseController.

```
@RestController
@RequestMapping("/Data")
public class DataResponseController {

    private ApplicationContext context = new ClassPathXmlApplicationContext("DatabaseBeans.xml");
    MainTemplateJDBC mainTemplateJDBC = (MainTemplateJDBC)context.getBean("mainTemplateJDBC");

    @RequestMapping("/UpdateDatabase")
    public String UpdateDatabase(){

        mainTemplateJDBC.simpleUpdate();
        return "База обновлена";
    }

    @RequestMapping("/GetAllGroupsOracle")
    public List<String> GetAllGroupsOracle() { return mainTemplateJDBC.getGroupDAO().findAllGroupsOracle(); }

    @RequestMapping("/GetAllGroupsMySQL")
    public List<String> GetAllGroupsMySQL() { return mainTemplateJDBC.getGroupDAO().findAllGroupsMySQL(); }

    @RequestMapping("/SelectOracle")
    public List<ContentOfSchedule> dataSelectOracle() { return mainTemplateJDBC.findAllOracle(); }

    @RequestMapping("/SelectMySQL")
    public List<CustomContentOfSchedule> dataSelectMySQL(@RequestParam(value="group", defaultValue="") String group,
                                        @RequestParam(value="starton", defaultValue="") String starton,
                                        @RequestParam(value="endon", defaultValue="") String endon) {...}
}
```

Рисунок 3.4 – Реализация контроллера DataResponseController

Теперь при обращении к приложению по адресу «Data/GetAllGroupsOracle», «Data/GetAllGroupsMySQL», «Data/GetAllLecturerMySQL», «Data/SelectOracle», «/SelectMySQL» клиенту будут возвращаться данные в формате JSON и по адресу «Data/UpdateDatabase» будет запускаться процесс обновления расписания.

3.2 Тестирование разработанного приложения

Тестирование разработанного приложения будет состоять из одновременной проверки всех свойств с использованием сервера, на котором расположено приложение. Для начала необходимо собрать рабочий проект в .jar файл и затем запустить его на локальной машине, на которой уже установлены необходимые базы данных. На момент тестирования использовался Spring framework версии 1.3.5 [18; 19].

Для тестирования приложения необходимо разработать демоверсию клиента, который будет запрашивать данные у сервера и получать ответ. Для этого был разработан файл index.html, который при помощи javascript [15] осуществляет запросы к разработанному приложению и выводит результат этих запросов на экран. На рисунке 3.5 представлен фрагмент кода на javascript, который при помощи ajax выполняет запрос на сервер после нажатия на кнопку с id равным «show».

```
$('#show').click(function () {
    $.ajax({
        url: "http://localhost:8090/Data/SelectMySQL?group="+$('#group').val()+"&starton="+ $('#StartOn').val()+"
        type: "GET",
        data: JSON,
        success: function(data){
            $('#tbody').empty();
            if(data.length == 0){
                console.log('Расписание не было найдено'); return;
            }
            $(data).each(function(e){
                $('#tbody').append("<tr><td>"+["Вс", "Пн", "Вт", "Ср", "Чт", "Пт", "Сб"][new Date(data[e]['startO
            });
        }
    });
});
```

Рисунок 3.5 – Фрагмент кода, выполняющий запрос на сервер

При отправке сообщений формируется строка запроса с введенными данными пользователя.

Вывод полученного сообщения производится на страницу при помощи события success, которое срабатывает после каждого успешного запроса.

Внешний вид интерфейса имеет два поля для ввода начала и окончания занятий, одно поля для выбора группы из списка и таблицу, в которую будут выводиться результаты запросов (рис 3.6).

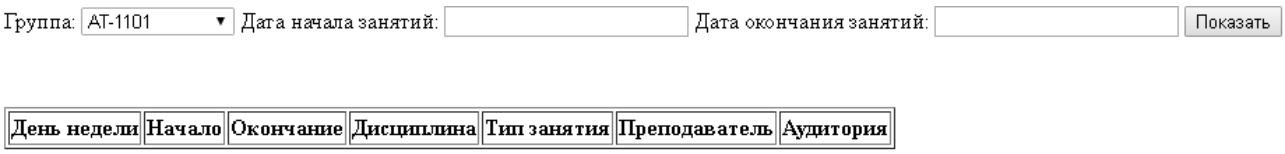

Рисунок 3.6 – Внешний вид интерфейса клиента

После нажатия на кнопку «Показать» на интерфейсе, вызывается функция отправки данных на сервер и при получении сервером сообщения json с данными из формы для ввода данных, происходит поиск расписания для указанной группы. На рисунке 3.7 приведен пример просмотра расписания всех учебных занятий, начинающихся с 2016-09-15 для группы АТ-1101.

Группа: АТ-1101 Дата начала занятий: 2016-09-15 Дата окончания занятий: Показать

День недели	Начало	Окончание	Дисциплина	Тип занятия	Преподаватель	Аудитория
Вт	2015-09-15 16:15:00.0	2015-09-15 17:45:00.0	Испытания автомобиля	Лекция	Соломатин Николай Сергеевич	Д-201
Вт	2015-09-15 14:30:00.0	2015-09-15 16:00:00.0	Испытания автомобиля	Лекция	Соломатин Николай Сергеевич	Д-201
Вт	2015-09-22 14:30:00.0	2015-09-22 16:00:00.0	Испытания автомобиля	Лекция	Соломатин Николай Сергеевич	Д-201
Вт	2015-09-22 16:15:00.0	2015-09-22 17:45:00.0	Испытания автомобиля	Лекция	Соломатин Николай Сергеевич	Д-201
Ср	2015-09-16 10:15:00.0	2015-09-16 11:45:00.0	Организация и планирование производства	Лекция	Капрова Вера Григорьевна	Г-320
Ср	2015-09-23 10:15:00.0	2015-09-23 11:45:00.0	Организация и планирование производства	Лекция	Капрова Вера Григорьевна	Г-324
Ср	2015-09-16 12:45:00.0	2015-09-16 14:15:00.0	Автоматические системы автомобиля	Лекция	Черепанов Леонид Ананьевич	Д-302
Ср	2015-09-23 12:45:00.0	2015-09-23 14:15:00.0	Автоматические системы автомобиля	Лекция	Черепанов Леонид Ананьевич	Д-201
Ср	2015-09-16 16:15:00.0	2015-09-16 17:45:00.0	Основы автотехнической экспертизы	Лекция	Черепанов Леонид Ананьевич	Д-201
Вт	2015-09-15 12:45:00.0	2015-09-15 14:15:00.0	Сертификация продукции автомобилестроения	Лекция	Соломатин Николай Сергеевич	Д-201
Вт	2015-09-22 12:45:00.0	2015-09-22 14:15:00.0	Основы автотехнической экспертизы	Лекция	Черепанов Леонид Ананьевич	Д-201
Пн	2015-09-14 14:30:00.0	2015-09-14 16:00:00.0	Специализированное программное обеспечение	Лекция	Зотов Алексей Викторович	Д-203
Пн	2015-09-14 16:15:00.0	2015-09-14 17:45:00.0	Основы качества и надежности автомобиля	Лекция	Зотов Алексей Викторович	Д-203
Пт	2015-09-18 10:15:00.0	2015-09-18 11:45:00.0	Основы активной и пассивной безопасности автомобиля	Лекция	Зотов Алексей Викторович	Д-201
Пт	2015-09-04 14:30:00.0	2015-09-04 16:00:00.0	Проектирование автомобиля 2 /Проектирование автомобиля/	Лекция	Еремина Ирина Васильевна	Д-101
Пт	2015-09-04 16:15:00.0	2015-09-04 17:45:00.0	Проектирование автомобиля 2 /Проектирование автомобиля/	Лекция	Еремина Ирина Васильевна	Д-101
Вт	2015-09-01 14:30:00.0	2015-09-01 16:00:00.0	Испытания автомобиля	Лекция	Соломатин Николай Сергеевич	Д-201

Рисунок 3.7 – Пример просмотра расписания учебных занятий

Для тестирования приложения необходим сервер. Собираем версию приложения при помощи команды mvn package. Запускаем сервер при помощи команды java –jar и собранного .jar файла через командную строку windows (рис 3.8) [20].

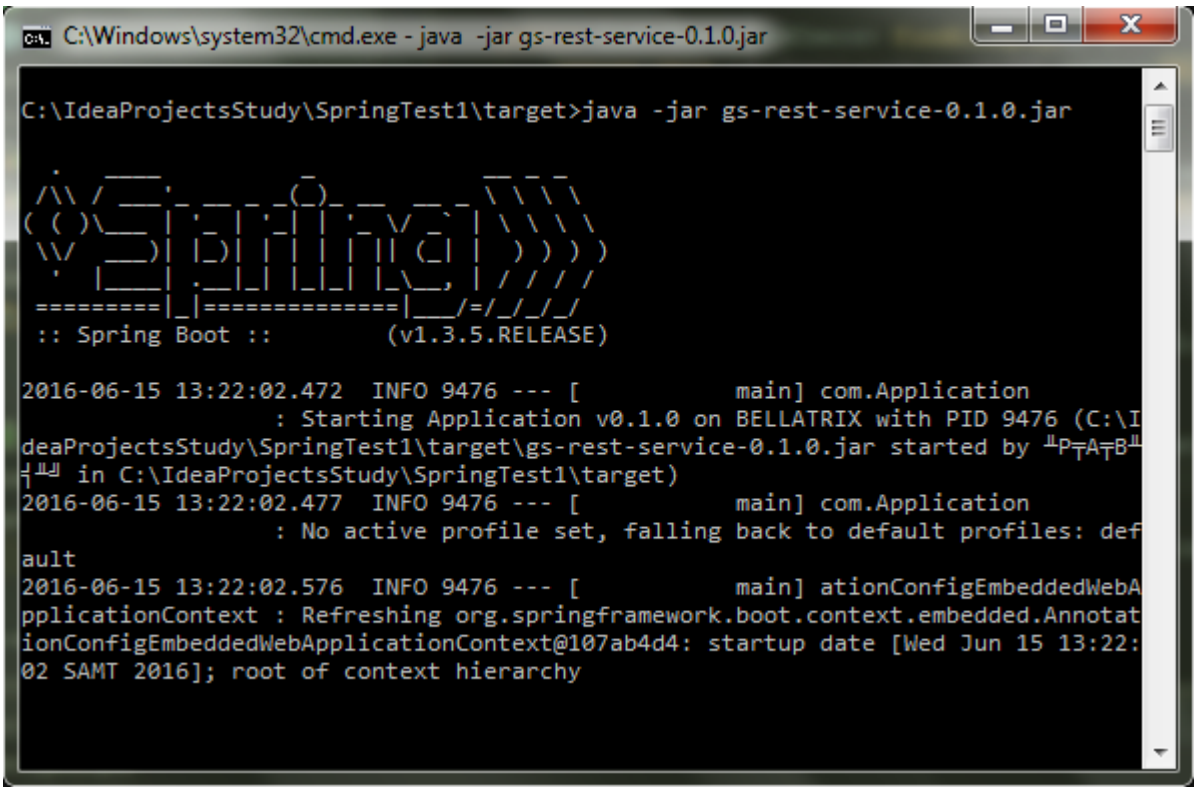

Рисунок 3.8 – Запуск сервера из командной строки

После того как приложение было запущено оно готово к работе. Обращаясь по адресам, приведенным в п. 3.1, мы можем обновлять расписание в базе данных приложения из системы составления расписания, а так же получать информацию о расписании.

ЗАКЛЮЧЕНИЕ

В процессе выполнения бакалаврской работы была выделена актуальность исследуемой темы, определены объект, предмет исследования, цели и задачи работы. Был проведен анализ действующей технологии интеграции расписания учебных занятий («Галактика Расписание учебных занятий»), в результате которого было принято решение об автоматизации процесса публикации расписания учебных занятий в Тольяттинском государственном университете.

По итогам проведенного анализа были сформулированы функциональные требования разрабатываемой технологии интеграции и публикации расписания учебных занятий ТГУ.

В ходе проектировании информационной системы были составлены логическая и физическая модель данных проектируемой системы, а также диаграммы последовательности действий, классов и вариантов использования. Была выбрана система управления данных, при помощи которой будет реализованная данная система.

В результате разработки реализована первичная версия приложения по автоматизации публикации расписания учебных занятий Тольяттинского государственного университета при помощи среды разработки Intellij idea, языка программирования Java и универсального фреймворка Spring для Java-платформ.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Учебники и учебные пособия

1. Вигерс К. Разработка требований к программному обеспечению // К. Вигерс, Д. Битти. -СПб:BHV,2014.-736с.
2. Гома, Х. UML-проектирование систем реального времени параллельных и распределенных приложений / Х. Гома. – 2-е изд. – ДМК Пресс, 2011. – 704с.
3. Гущин А. Н. Базы Данных. 2-е изд., испр. и доп.: учебно-методическое пособие / А. Н. Гущин – М. Берлин: Директ – Медиа, 2015. – 311с.
4. МакГрат, Майк. Программирование на Java для начинающих / Майк Мак - Грат; [пер. с англ. М.А. Райтмана]. – Москва: Издательство 8. «Э», 2016. – 192с.
5. Уокенбах Дж. Microsoft Excel 2010. Библия пользователя. : Пер. с англ. – М.: Вильямс, 2011. –912 с.
6. Фаулер М. Рефакторинг. Улучшение существующего кода // М. Фаулер. – СПб : Символ Плюс, 2015. – 432с.
7. Хорстманн К. Java SE 8. Базовый курс // К. Хорстманн. – Москва:Вильямс, 2016. – 464 с.
8. Прохоренок, Н.А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок, В.А. Дронов. – 4-е изд. – СПб.: Изд-во «БХВ-Петербург», 2015. – 766с.

Электронные ресурсы

9. 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)
10. Справочное руководство по MySQL [Электронный ресурс]: MySQL Manual. – URL: http://www.mysql.ru/docs/man/

11. Тольяттинский государственный университет [Электронный ресурс] // Тольяттинский государственный университет: История Тольяттинского государственного университета. – URL: http://www.tltsu.ru (1.06.2016)

Литература на иностранном языке

12. Baesens, B. Beginning Java Programming: The Object-Oriented Approach / B. Baesens, A. Backiel, S. Vanden Broucke. – 1st edition, Wrox, 2015.
13. Bill Burke, RESTful Java with JAX-RS 2.0, Second Edition / Bill Burke - O'Reilly Media, Inc. – 392 p.
14. David Skla. Learning PHP: A Gentle Introduction to the Web's Most Popular Language / David Skla. - O'Reilly, 2016. – 416c.
15. Deitel, H. Java How to Program / H. Deitel, P. Deitel. – 9th edition, Prentice Hall, 2015.
16. Hudson O. Getting started with IntelliJ IDEA // O. Hudson, Birmingham: Packt Publishing, 2013. – 114p.
17. Krochmalski J. IntelliJ IDEA Essentials // J. Krochmalski. – Birmingham: Packt Publishing, 2014.-263p.
18. Masoud Kalali, Developing RESTful Services with JAX-RS 2.0, WebSockets, and JSON / Masoud Kalali – United Kingdom, Birmingham,: Packt Publishing, LTD, 2013. – 107 p.
19. Pro Spring MVC: With Web Flow (Expert's Voice in Spring) / M. Deinum, K. Serneels, C. Yates and other, – New York : Apress. – 2012. – 596 p.
20. Spring Data / M. Pollack, O. Gierke, T. Risberg and other, - California: O'Reilly Media. - 1st edition. - 316.

Приложение А

Листинг кода реализации классов

Класс Auditorium.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class Auditorium implements Table {
    private int OID;
    private String Name;
    private String Abbr;
    private int Building;
    private int TypeOfAuditorium;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Auditorium)) return false;

        Auditorium that = (Auditorium) o;

        if (getOID() != that.getOID()) return false;
        if (getBuilding() != that.getBuilding()) return false;
        if (getTypeOfAuditorium() != that.getTypeOfAuditorium()) return false;
        if (getName() != null ? !getName().equals(that.getName()) : that.getName() != null) return
false;
        return !(getAbbr() != null ? !getAbbr().equals(that.getAbbr()) : that.getAbbr() != null);

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getName() != null ? getName().hashCode() : 0);
        result = 31 * result + (getAbbr() != null ? getAbbr().hashCode() : 0);
```

```
    result = 31 * result + getBuilding();
    result = 31 * result + getTypeOfAuditorium();
    return result;
}

public void setAbbr(String abbr) {
    Abbr = abbr;
}

public void setBuilding(int building) {
    Building = building;
}

public void setName(String name) {
    Name = name;
}

public void setOID(int OID) {
    this.OID = OID;
}

public void setTypeOfAuditorium(int typeOfAuditorium) {
    TypeOfAuditorium = typeOfAuditorium;
}

public int getBuilding() {
    return Building;
}

public int getOID() {
    return OID;
}

public int getTypeOfAuditorium() {
    return TypeOfAuditorium;
```

```
    }

    public String getAbbr() {
        return Abbr;
    }

    public String getName() {
        return Name;
    }
}
```

Класс Building.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class Building implements Table {
    private int OID;
    private String Name;
    private String Address;

    public void setAddress(String address) {
        Address = address;
    }

    public void setName(String name) {
        Name = name;
    }

    public void setOID(int OID) {
        this.OID = OID;
    }

    public int getOID() {
        return OID;
    }
```

```
    public String getAddress() {
        return Address;
    }

    public String getName() {
        return Name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Building)) return false;

        Building building = (Building) o;

        if (getOID() != building.getOID()) return false;
        if (getName() != null ? !getName().equals(building.getName()) : building.getName() != null)
return false;
        return !(getAddress() != null ? !getAddress().equals(building.getAddress()) :
building.getAddress() != null);

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getName() != null ? getName().hashCode() : 0);
        result = 31 * result + (getAddress() != null ? getAddress().hashCode() : 0);
        return result;
    }
}
```

Класс Chair.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class Chair implements Table {
    private int OID;
    private String Name;
    private int Auditorium;

    public void setAuditorium(int auditorium) {
        Auditorium = auditorium;
    }

    public void setName(String name) {
        Name = name;
    }

    public void setOID(int OID) {
        this.OID = OID;
    }

    public int getAuditorium() {
        return Auditorium;
    }

    public int getOID() {
        return OID;
    }

    public String getName() {
        return Name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
```

```
        if (!(o instanceof Chair)) return false;

        Chair chair = (Chair) o;

        if (getOID() != chair.getOID()) return false;
        if (getAuditorium() != chair.getAuditorium()) return false;
        return !(getName() != null ? !getName().equals(chair.getName()) : chair.getName() != null);

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getName() != null ? getName().hashCode() : 0);
        result = 31 * result + getAuditorium();
        return result;
    }
}
```

Класс ContentOfSchedule.

```
package CRUD.tables.standard;
import CRUD.tables.Table;
import java.sql.Timestamp;

public class ContentOfSchedule implements Table {
    private int OID;
    private Timestamp StartOn;
    private Timestamp EndOn;
    private Timestamp ModifiedTime;
    private int Discipline;
    private int KindOfWork;
    private int Lecturer;
    private int Auditorium;
    private int Stream;
    private int Group;
```

```
    private int SubGroup;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof ContentOfSchedule)) return false;

        ContentOfSchedule that = (ContentOfSchedule) o;

        if (getOID() != that.getOID()) return false;
        if (getDiscipline() != that.getDiscipline()) return false;
        if (getKindOfWork() != that.getKindOfWork()) return false;
        if (getLecturer() != that.getLecturer()) return false;
        if (getAuditorium() != that.getAuditorium()) return false;
        if (getStream() != that.getStream()) return false;
        if (getGroup() != that.getGroup()) return false;
        if (getSubGroup() != that.getSubGroup()) return false;
        if (getStartOn() != null ? !getStartOn().equals(that.getStartOn()) : that.getStartOn() != null)
return false;
        if (getEndOn() != null ? !getEndOn().equals(that.getEndOn()) : that.getEndOn() != null) return
false;
        return !(getModifiedTime() != null ? !getModifiedTime().equals(that.getModifiedTime()) :
that.getModifiedTime() != null);

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getStartOn() != null ? getStartOn().hashCode() : 0);
        result = 31 * result + (getEndOn() != null ? getEndOn().hashCode() : 0);
        result = 31 * result + (getModifiedTime() != null ? getModifiedTime().hashCode() : 0);
        result = 31 * result + getDiscipline();
        result = 31 * result + getKindOfWork();
        result = 31 * result + getLecturer();
```

```
    result = 31 * result + getAuditorium();
    result = 31 * result + getStream();
    result = 31 * result + getGroup();
    result = 31 * result + getSubGroup();
    return result;
}

public void setAuditorium(int auditorium) {
    Auditorium = auditorium;
}

public void setDiscipline(int discipline) {
    Discipline = discipline;
}

public void setEndOn(Timestamp endOn) {
    EndOn = endOn;
}

public void setGroup(int group) {
    Group = group;
}

public void setKindOfWork(int kindOfWork) {
    KindOfWork = kindOfWork;
}

public void setLecturer(int lecturer) {
    Lecturer = lecturer;
}

public void setModifiedTime(Timestamp modifiedTime) {
    ModifiedTime = modifiedTime;
}
```

```
public void setOID(int OID) {
   this.OID = OID;
}

public void setStartOn(Timestamp startOn) {
   StartOn = startOn;
}

public void setStream(int stream) {
   Stream = stream;
}

public void setSubGroup(int subGroup) {
   SubGroup = subGroup;
}

public int getAuditorium() {
   return Auditorium;
}

public int getDiscipline() {
   return Discipline;
}

public int getGroup() {
   return Group;
}

public int getKindOfWork() {
   return KindOfWork;
}

public int getLecturer() {
   return Lecturer;
}
```

```
    public int getOID() {
        return OID;
    }

    public int getStream() {
        return Stream;
    }

    public int getSubGroup() {
        return SubGroup;
    }

    public Timestamp getEndOn() {
        return EndOn;
    }

    public Timestamp getModifiedTime() {
        return ModifiedTime;
    }

    public Timestamp getStartOn() {
        return StartOn;
    }
}
```

Класс Discipline.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class Discipline implements Table {
    private int OID;
    private String Name;

    @Override
```

```
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Discipline)) return false;

        Discipline that = (Discipline) o;

        if (getOID() != that.getOID()) return false;
        return !(getName() != null ? !getName().equals(that.getName()) : that.getName() != null);

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getName() != null ? getName().hashCode() : 0);
        return result;
    }

    public void setName(String name) {
        Name = name;
    }

    public void setOID(int OID) {
        this.OID = OID;
    }

    public int getOID() {
        return OID;
    }

    public String getName() {
        return Name;
    }
}
```

Класс Group.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class Group implements Table {
    private int OID;
    private int Course;
    private String Name;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Group)) return false;

        Group group = (Group) o;

        if (getOID() != group.getOID()) return false;
        if (getCourse() != group.getCourse()) return false;
        return !(getName() != null ? !getName().equals(group.getName()) : group.getName() != null);
    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + getCourse();
        result = 31 * result + (getName() != null ? getName().hashCode() : 0);
        return result;
    }

    public void setCourse(int course) {
        Course = course;
    }

    public void setName(String name) {
        Name = name;
    }
```

```
    public void setOID(int OID) {
        this.OID = OID;
    }

    public int getCourse() {
        return Course;
    }

    public int getOID() {
        return OID;
    }

    public String getName() {
        return Name;
    }
}
```

Класс KindOfWork.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class KindOfWork implements Table {
    private int OID;
    private String Name;
    private String Abbr;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof KindOfWork)) return false;

        KindOfWork that = (KindOfWork) o;

        if (getOID() != that.getOID()) return false;
```

```
        if (getName() != null ? !getName().equals(that.getName()) : that.getName() != null) return
false;
        return !(getAbbr() != null ? !getAbbr().equals(that.getAbbr()) : that.getAbbr() != null);

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getName() != null ? getName().hashCode() : 0);
        result = 31 * result + (getAbbr() != null ? getAbbr().hashCode() : 0);
        return result;
    }

    public void setAbbr(String abbr) {
        Abbr = abbr;
    }

    public void setName(String name) {
        Name = name;
    }

    public void setOID(int OID) {
        this.OID = OID;
    }

    public int getOID() {
        return OID;
    }

    public String getAbbr() {
        return Abbr;
    }

    public String getName() {
```

```
        return Name;
    }
}
```

Класс Lecturer.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class Lecturer implements Table {
    private int OID;
    private String FIO;
    private int Chair;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Lecturer)) return false;

        Lecturer lecturer = (Lecturer) o;

        if (getOID() != lecturer.getOID()) return false;
        if (getChair() != lecturer.getChair()) return false;
        return !(getFIO() != null ? !getFIO().equals(lecturer.getFIO()) : lecturer.getFIO() != null);

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getFIO() != null ? getFIO().hashCode() : 0);
        result = 31 * result + getChair();
        return result;
    }

    public void setChair(int chair) {
```

```
        Chair = chair;
    }

    public void setFIO(String FIO) {
        this.FIO = FIO;
    }

    public void setOID(int OID) {
        this.OID = OID;
    }

    public int getChair() {
        return Chair;
    }

    public int getOID() {
        return OID;
    }

    public String getFIO() {
        return FIO;
    }
}
```

Класс Stream.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class Stream implements Table {
    private int OID;
    private String Name;

    public void setName(String name) {
        Name = name;
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Stream)) return false;

        Stream stream = (Stream) o;

        if (getOID() != stream.getOID()) return false;
        return !(getName() != null ? !getName().equals(stream.getName()) : stream.getName() !=
null);

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getName() != null ? getName().hashCode() : 0);
        return result;
    }

    public void setOID(int OID) {
        this.OID = OID;
    }

    public int getOID() {
        return OID;
    }

    public String getName() {
        return Name;
    }
}
```

Класс SubGroup.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class SubGroup implements Table {
    private int OID;
    private int Group;

    public void setGroup(int group) {
        Group = group;
    }

    public void setOID(int OID) {
        this.OID = OID;
    }

    public int getGroup() {
        return Group;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof SubGroup)) return false;

        SubGroup subGroup = (SubGroup) o;

        if (getOID() != subGroup.getOID()) return false;
        return getGroup() == subGroup.getGroup();

    }

    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + getGroup();
```

```
        return result;
    }

    public int getOID() {
        return OID;
    }
}
```

Класс TypeOfAuditorium.

```
package CRUD.tables.standard;
import CRUD.tables.Table;

public class TypeOfAuditorium implements Table {
    private int OID;
    private String Name;

    public void setName(String name) {
        Name = name;
    }

    public void setOID(int OID) {
        this.OID = OID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof TypeOfAuditorium)) return false;

        TypeOfAuditorium that = (TypeOfAuditorium) o;

        if (getOID() != that.getOID()) return false;
        return !(getName() != null ? !getName().equals(that.getName()) : that.getName() != null);

    }
```

```
    @Override
    public int hashCode() {
        int result = getOID();
        result = 31 * result + (getName() != null ? getName().hashCode() : 0);
        return result;
    }

    public int getOID() {
        return OID;
    }

    public String getName() {
        return Name;
    }
}
```