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 Реализация системы оповещения об изменении расписания в едином информационном пространстве ТГУ

Студент А. А. Скворцов _____

Руководитель А. В. Очеповский _____

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 Скворцов Артем Анатольевич
1. Тема: Реализация системы оповещения об изменении расписания в едином информационном пространстве ТГУ
2. Срок сдачи студентом законченной выпускной квалификационной работы 19.06.2016
3. Исходные данные к выпускной квалификационной работе:
— количество одновременно работающих пользователей, не менее 100;
— максимальное время формирования ответа за запрос пользователя, не более 10 сек.;
— режим работы 24/7/365;
— разработанное программное обеспечение должно удовлетворять спецификации Java EE 8;
4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов):
Введение
1. Анализ системы оповещения об изменениях расписания
2. Проектирование системы оповещения об изменении расписаний тольяттинского государственного университета
3. Кодирование модуля оповещения в едином информационном пространстве ТГУ
Заключение
5. Ориентировочный перечень графического и иллюстративного материала:

презентация, включающая блок-схемы работы приложения, графики, диаграммы, экранные формы, демонстрирующие работоспособность программного продукта.
6. Дата выдачи задания « 11 » января 2016 г.

Заказчик Начальник отдела разработки информационных систем, ЦНИТ ТГУ	_____	С.В. Баумгертнер
Руководитель выпускной квалификационной работы	_____	А.В. Очеповский
Задание принял к исполнению	_____	А.А. Скворцов

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 Скворцова Артема Анатольевича

по теме Реализация системы оповещения об изменении расписания в едином информационном пространстве ТГУ

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Выбор и утверждение темы ВКР	14.01.2016	14.01.2016	Выполнено	
Поиск и анализ литературы по проблеме реализации модулей оповещений	15.01.2016	15.01.2016	Выполнено	
Описание принципов разработки	16.01.2016	16.01.2016	Выполнено	
Анализ существующих сервисов оповещения	16.01.2016	16.01.2016	Выполнено	
Моделирование структуры модуля	17.01.2016	17.01.2016	Выполнено	
Обоснование выбора средств реализации модуля оповещения	17.01.2016	17.01.2016	Выполнено	
Описание алгоритмов работы основных	18.01.2016	18.01.2016	Выполнено	

модулей приложения				
Реализация модуля программными средствами	20.02.2016	20.02.2016	Выполнено	
Описание контрольного примера работы разработанного модуля	20.03.2016	20.03.2016	Выполнено	
Тестирование реализованного модуля	29.03.2016	29.03.2016	Выполнено	
Апробация модуля	30.04.2016	30.04.2016	Выполнено	
Оформление пояснительной записки ВКР	05.05.2016	05.05.2016	Выполнено	
Подготовка презентации к выступлению	10.05.2016	10.05.2016	Выполнено	
Предварительная защита ВКР	20.05.2016	20.05.2016	Выполнено	
Корректировка ВКР согласно сделанным замечаниям	01.06.2016	01.06.2016	Выполнено	
Проверка ВКР в системе «Антиплагиат.ВУЗ»	05.06.2016	05.06.2016	Выполнено	
Сдача пояснительной записки ВКР и реализованного программного приложения	19.06.2016	19.06.2016	Выполнено	

Руководитель выпускной квалификационной работы _____ А.В. Очеповский

Задание принял к исполнению _____ А.А. Скворцов

Аннотация

Темой данной выпускной квалификационной работы является «Реализация системы оповещения об изменении расписания в едином информационном пространстве ТГУ».

Работа выполнена студентом ТГУ, института математики физики и информационных технологий, группы МОб-1201, Скворцовым Артемом Анатольевичем.

Объект исследования – процесс составления и изменения расписания учебных занятий в ТГУ, **предмет** – система оповещения об изменении расписания.

Целью работы является реализация системы оповещения об изменениях расписания в едином информационном пространстве ТГУ

Задачами работы являются:

— анализ системы оповещения об изменении расписания ТГУ и составление требований к реализуемой системе;

— проектирование реализуемой системы оповещения;

— реализация и тестирование разработанной системы оповещения.

Работа состоит из введения, трех глав (аналитической, проектной и части реализации), заключения, списка использованной литературы и приложения.

В работе использованы современные системы и технологии проектирования информационных систем, Java, MySQL и Spring framework др.

Первая глава посвящена анализу системы оповещения об изменении расписания в ТГУ.

Во второй главе описано проектирование системы оповещения.

В третьей главе предоставлены реализация и тестирование системы оповещения.

В заключении сформулированы основные выводы, которые были сделаны в процессе исследования и описаны результаты практической реализации работы.

В приложении предоставлены фрагменты программного кода и другие дополнительные материалы.

В данной работе содержится 56 страниц, на которых 2 таблицы, 37 рисунков, 21 источник используемой литературы, два приложения.

Оглавление

Введение

Повседневно развиваются технологии разработки программного обеспечения, популярность информационных систем растет и вместе с этим увеличивается производительность программных продуктов. Соответственно с ростом производительности растет и результативность любой отрасли, использующей современное программное обеспечение.

В наше время самой прогрессивной областью информационного пространства являются информационные ресурсы, в которых хранится информация о компаниях и их направлении деятельности. Такие информационные ресурсы называются веб-сайтами и они доступны в любой точке мира благодаря сети интернет. Веб-сайт это набор веб-страниц, отображающих целевую информацию в удобном виде. Условный переход по страницам сайта может осуществить любой желающий, при помощи запроса через интернет браузер(специальную программу для отображения страниц). В этом случае браузер, используя протокол получения гипертекста(http), загружает на устройство клиента сформированную страницу в необходимом виде. Веб-сайты содержат данные общего доступа, распространяемые с определенными целями по всей сети интернет.

Тольяттинский государственный университет (ТГУ) является одним ведущих вузов России и обладает огромным количеством информации, для студентов, абитуриентов, сотрудников вуза. Как прогрессивный университет, ТГУ имеет свой веб-сайт, на котором существует вся необходимая информация для любого студента. Одной из основных целей является распространение расписания для студентов очной формы обучения, которые составляют большую часть обучающихся. Ежедневно расписания просматривают тысячи студентов. От точности расписания зависит успеваемость студентов, а значит и их будущее, как и будущее страны в целом, которое напрямую зависит от степени квалификации выпускников.

Исходя из вышесказанного, подчеркивается **актуальность** данной темы для развития технологий в России и по всему миру.

В данный момент существует проект по развитию распространения расписания в стенах ТГУ. Планируется переработка документов для вывода расписания и показ расписания в целом. Автоматизация вывода расписания ранее не рассматривалась, все документы с расписанием формировались в ручную. Наличие системы оповещения и рассылка уведомлений с измененными данными будет являться принципиально новой технологией. В существующих университетах крайне мала вероятность наличия автоматизированной системы оповещения., поэтому выполнение данной реализации отличается **новизной.**

Объектом исследования данной бакалаврской работы является процесс составления и изменения расписания учебных занятий в ТГУ.

Предметом исследования данной работы является система оповещения об изменениях расписания в едином информационном пространстве ТГУ.

Цель бакалаврской работы разработать систему оповещения в едином информационном пространстве ТГУ.

В **задачи** бакалаврской работы входят:

— анализ системы оповещения об изменении расписания ТГУ и составление требований к реализуемой системе;

— проектирование реализуемой системы оповещения;

— реализация и тестирование разработанной системы оповещения.

Данная выпускная квалификационная работа состоит из введения, трех глав, заключения и списка литературы.

Во введении описывается вводный курс в современные технологии, актуальность темы, объект исследования, ставятся цели и устанавливаются задачи работы, необходимые для выполнения поставленной цели.

В первой главе рассматривается существующая система оповещения и причины, по которым ее необходимо заменить. Также анализируются существующие модули оповещения, производится анализ разработанного алгоритма оповещения. В конце главы формируются требования к системе.

Во второй главе происходит выбор методологии разработки программного обеспечения, рассматриваются аналоги и анализируются их положительные и отрицательные стороны. Проектируется система в виде диаграмм классов, при использовании алгоритма рассылки. Производится постановка задачи на разработку.

В третьей главе осуществляется реализация системы оповещения, используются средства реализации, такие как язык Java, IDE Intellij Idea и СУБД MySQL. Осуществляется тестирование на локальной машине с отработкой подключения и проверкой оповещения через javascript клиент в браузере.

В заключении описываются достигнутые цели, выводы по разработке и планирование дополнительных версий продукта.

Глава 1 Анализ системы оповещения об изменениях расписания

1.1 Анализ системы оповещения в тольяттинском государственном университете

Высшее образование это совокупность систематизированных знаний и умений, получаемых в высшем учебном заведении, включающее в себя практические навыки решения теоретических и практических задач в рамках заданного профессионального профиля. Получение высшего образования требует определенных способностей и целеустремленности. На текущий момент в России, согласно конституции Российской Федерации, любой, имеющий среднее полное образование, может поступить в вуз на бесплатное обучение высшему образованию на конкурсной основе, либо пройти обучение в заочной форме, а также на платной основе. Очная форма, в отличии от заочной, предоставляет возможность изучать многие аспекты на практике и слушать лекции. При взаимодействии с преподавательским составом университета студент перенимает не только знания, но и опыт. Исключением является работа по специальности, где взаимодействие с коллегами и непосредственное изучение систем позволяет получить необходимый опыт.

Тольяттинский государственный университет (ТГУ), один из ведущих вузов России, в который ежегодно поступают более 2000 студентов. В ТГУ проходят обучение на бакалавриат по 39-ти направлениям подготовки. ТГУ является градостроительным университетом города Тольятти и от качества обучения студентов зависит будущее города и страны в целом.

Структура ТГУ состоит из множества отделов, за счет которых функционирует весь вуз. Во главе всей структуры ученый совет и ректор. Заместителями ректора ТГУ являются проректоры институтов, которые решают оперативные и тактические вопросы вуза. Также в ТГУ присутствует учебно-методическое управление, которое включает в себя такие подразделения как (рис. 1.1):

- диспетчерскую службу;

- отдел планирования и организации учебного процесса;
- отдел дистанционного обеспечения.

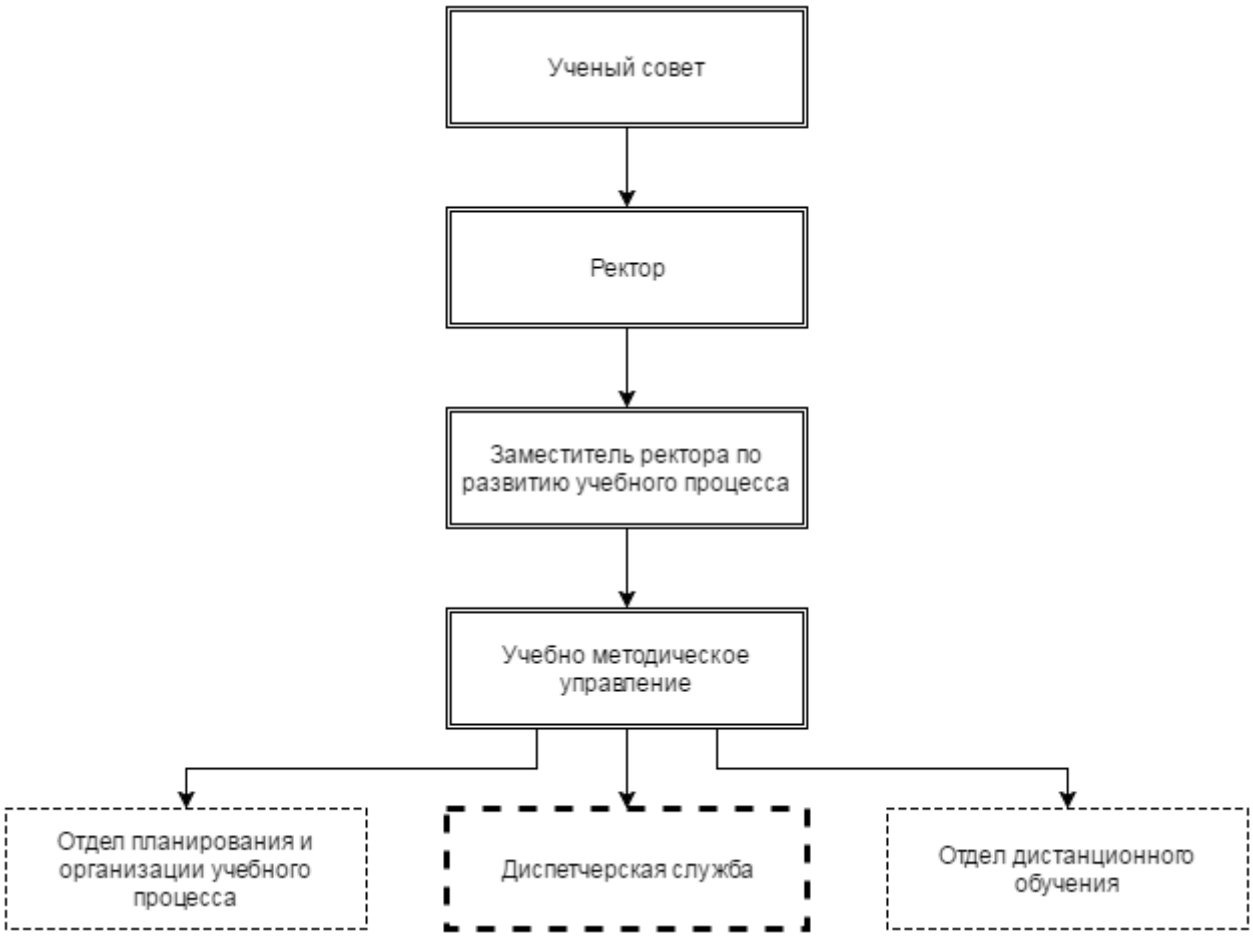

Рисунок 1.1 – Фрагмент структуры ТГУ.

Выделенная на рисунке 1.1 «Диспетчерская служба» будет рассматриваться в дальнейшем, как служба, функция которой подлежит усовершенствованию. Рассмотрим диспетчерскую службу как функционирующий компонент учебно-методического управления.

Диспетчерская служба очень важный отдел, обеспечивающий доступность расписания ТГУ для студентов, сотрудников вуза и всех желающих узнать планирование занятий вуза. Диспетчерская служба состоит из нескольких человек, формирующих расписание по институтам и группам. Еженедельно, диспетчерская служба, используя базу данных, получает данные о занятиях в университете и формирует с их помощью файлы формата excel, где каждый файл соответствует определенному институту. При изменении

текущего расписания, диспетчерская служба редактирует excel файлы и обновляет их на сайте, что никак не могут отследить студенты и преподаватели.

Возникают ситуации, при которых необходимо изменить расписание по какой либо причине, если преподаватель по уважительной причине не может присутствовать на проводимом занятии или по какой-либо другой причине. Рассмотрим механизм оповещения, существующий в ТГУ на данный момент времени.

Как сказано в инструкции по организации исполнения учебных занятий ТГУ [2], в случае невозможности проведения занятия согласно расписанию, о чем известно более чем за сутки до проведения занятия, преподаватель обязан, в срок более чем за сутки до начала проведения учебного занятия, лично или телефонным звонком проинформировать заведующего кафедрой и предоставить подтверждающие документы о невозможности проведения занятия. Заведующий кафедрой должен принять решение о замене преподавателя или о замене занятия, приоритет ставится на замену преподавателя, т.к. замена занятия должна предусматривать изменение расписания, для чего необходимо согласовать замену с ДС учебно-методического управления (УМУ). В случае, если начальник ДС УМУ имеет возможность удовлетворить требованию заведующего кафедрой о переносе расписания, начальник ДС УМУ или в его отсутствие главный специалист по расписанию, вносят соответствующие изменения в расписание учебных занятий, размещенное на стендах, на сайте и на образовательном портале, изменения выделяются цветом. В соответствии с изменением расписания, заместитель директора института по учебной работе или соответствующий сотрудник управления сопровождения учебного процесса(УСУП) в течение часа с момента получения информации об изменении расписания, обязан оповестить старосту или в его отсутствие любого другого студента данной группы, о полученной информации. Староста или любой другой студент указанной группы обязан оповестить остальных студентов группы о внесенных изменениях.

При возникновении причин, возникших менее чем за сутки до проведения учебных занятий, преподаватель должен в максимально кратчайшие сроки информировать заведующего кафедрой (в его отсутствие начальника ДС УМУ) о невозможности проведения занятия. Впоследствии необходимо предоставить заведующему кафедрой подтверждающие документы. После, заведующий кафедрой принимает решение о замене занятия или замене преподавателя, уведомляет начальника ДС УМУ или главного специалиста по расписанию ДС УМУ. Информация об изменении расписания доносится до старосты группы или до любого другого студента данной группы.

При изменении места проведения занятия, заведующий кафедрой уведомляет начальника ДС УМУ о необходимости переноса учебного занятия в другую аудиторию, до начала занятий. При необходимости оперативного изменения аудитории информирует начальника ДС УМУ по телефону. Сотрудник кафедры готовит и размещает на двери аудитории, из которой переносится занятие, объявление с указанием аудитории, в которой будет проводиться занятие, группы, дисциплины, времени проведения занятия, фамилии и инициалов преподавателя.

Для выявления недостатков данной системы рассмотрим модель «как

есть» на рисунке 1.2.

Рисунок 1.2 – Модель процесса составления расписания учебных занятий ТГУ «как есть» в нотации IDEF0

Для того чтобы выявить недостатки модели процесса составления расписания учебных занятий в ТГУ, необходимо составить декомпозицию данного процесса, в полной мере отражающей структуру процессов формирования расписаний и вывода их для целевых пользователей. Составим декомпозицию как указано на рисунке 1.3.

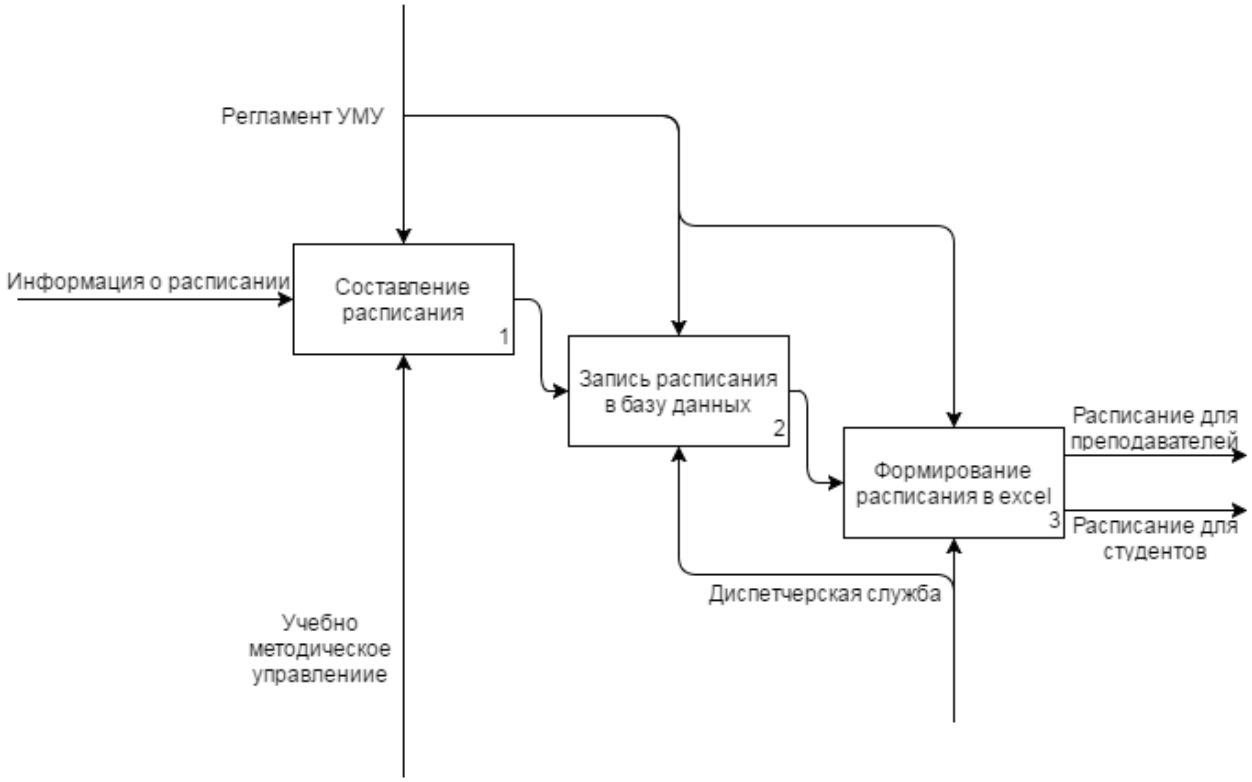

Рисунок 1.3 – Декомпозиция модели «как есть»

Недостатками данной системы оповещения является то, что она не предусматривает случаев, когда необходимо оповестить преподавателя об изменении расписания, например в случае освобождении от занятий всей группы или других непредусмотренных ситуаций. А в случае оповещения студентов, возможна недостаточная осведомленность студентов о предстоящих переносах учебных занятий, по причине недоступности связи со студентом.

В соответствии с перечисленными недостатками, было решено усовершенствовать диспетчерскую службу, реализовав **функцию** рассылки уведомлений об изменении расписания в едином информационном пространстве ТГУ. Данная функция будет реализована на стороннем сервере, в виде модуля. На сторонний сервер будет реплицироваться база данных, с необходимыми данными о расписании учебных занятий. Модуль будет

уведомлять всех подключенных пользователей об изменении расписания указанной группы, определяя время последнего уведомления каждого пользователя и его рабочей группы.

1.2 Архитектура модуля системы оповещения

Модуль системы оповещения это программа, позволяющая извещать пользователей о каких-либо изменениях по интересующим их данным посредством интернета. Любая система оповещения должна уметь хранить данные, анализировать их изменения, составлять сообщения для отправки данных и поддерживать один из протоколов передачи данных в виде постоянного обмена пакетами с пользователями и источником данных.

На текущий момент времени, нет информации о существующих аналогах модулей систем оповещения для расписания вузов и других учебных заведений. Поэтому было решено разработать собственный алгоритм рассылки оповещений.

Алгоритм рассылки это способ, которым будут поставляться данные пользователю. В начальной версии продукта, передача данных должна осуществляться при помощи http запросов, которые будут возвращать данные по определенной группе. В соответствии с понятием системы

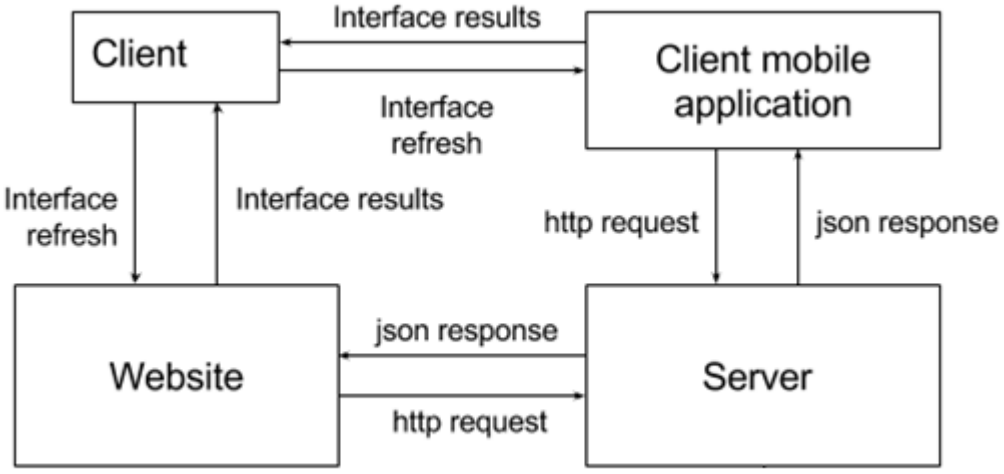

Рисунок 1.4– Схема взаимодействия клиента с сервером

оповещения можно создать схему взаимодействия между клиентом и сервером. Алгоритм первой версии продукта выглядит следующим образом, как представлено на рисунке 1.4:

Клиент (Client) используя любой из доступных ему способов общения с сервером, таких как клиентское мобильное приложение (Client mobile application) или вебсайт (Website), запрашивает данные об изменениях при помощи протокола http и получает json ответ, который выводится в виде интерфейса в зависимости от используемого приложения. Блок-схема алгоритма стандартного запроса о расписании изображена на рисунке 1.5.

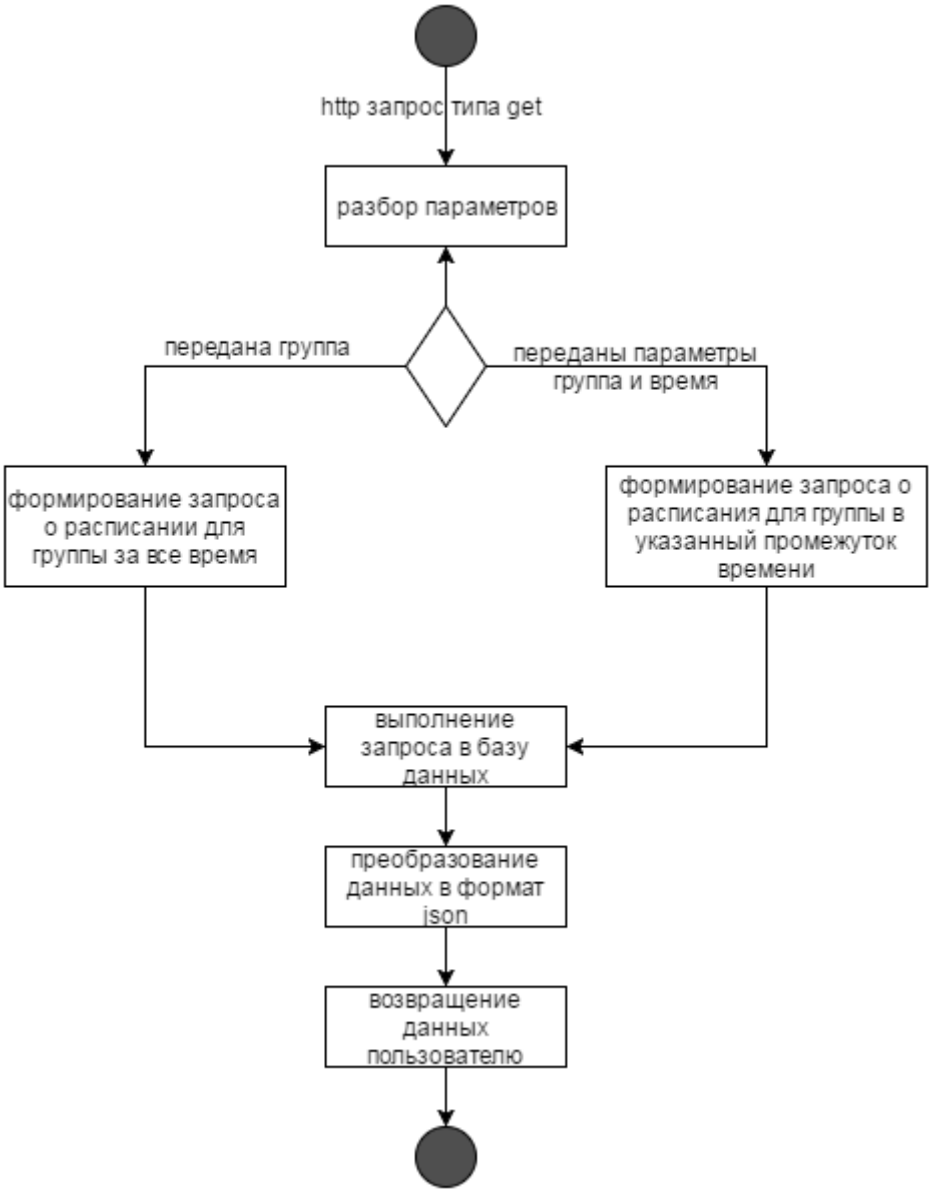

Рисунок 1.5 – Блок-схема алгоритма действий сервера при получении get запроса по протоколу http

В случае получения данных пользователем в виде json приложению пользователя останется лишь сделать выборку данных и выбрать нужные столбцы и строки. И далее вывести пользователю конечный результат при помощи предоставляемого интерфейса. Данная реализация системы оповещения эффективна при редких обновлениях и небольшом количестве пользователей, но по причине возможных частых обновлений была разработана система оповещения с использованием технологии websocket.

Websocket – это специальный протокол, обеспечивающий взаимодействие типа клиент-сервер. Протокол websocket был разработан специально для серверов, для постоянной связи с клиентом. Способ установки соединения по протоколу websocket показан на рисунке 1.6.

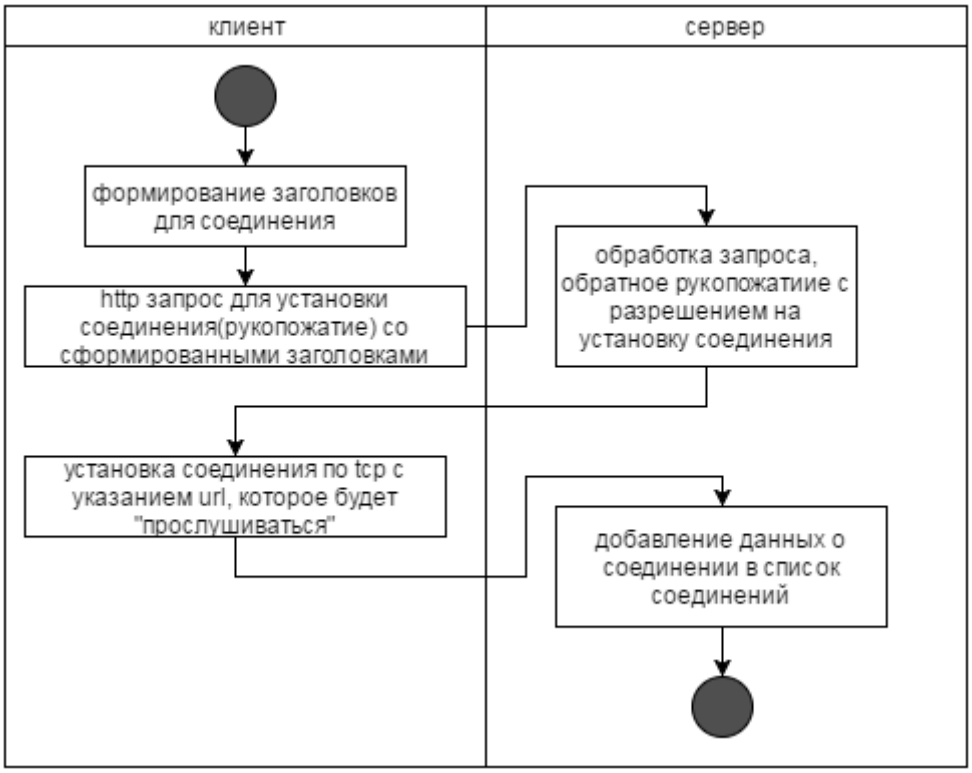

Рисунок 1.6 – Установка соединения через протокол websocket

При необходимости отсылки сообщения, указывается лишь адрес, по которому необходимо совершить рассылку, после чего механизм автоматически рассылает информацию по всем клиентам.

Использование websocket-ов не нагружает сервер, так как каждое соединение находится в состоянии ожидания и почти не требует процессорного времени на обработку, активируясь только в случае отправки данных или их

получения. В результате анализа был разработан алгоритм подключения клиента к серверу, при использовании протокола websocket (рисунок 1.7).

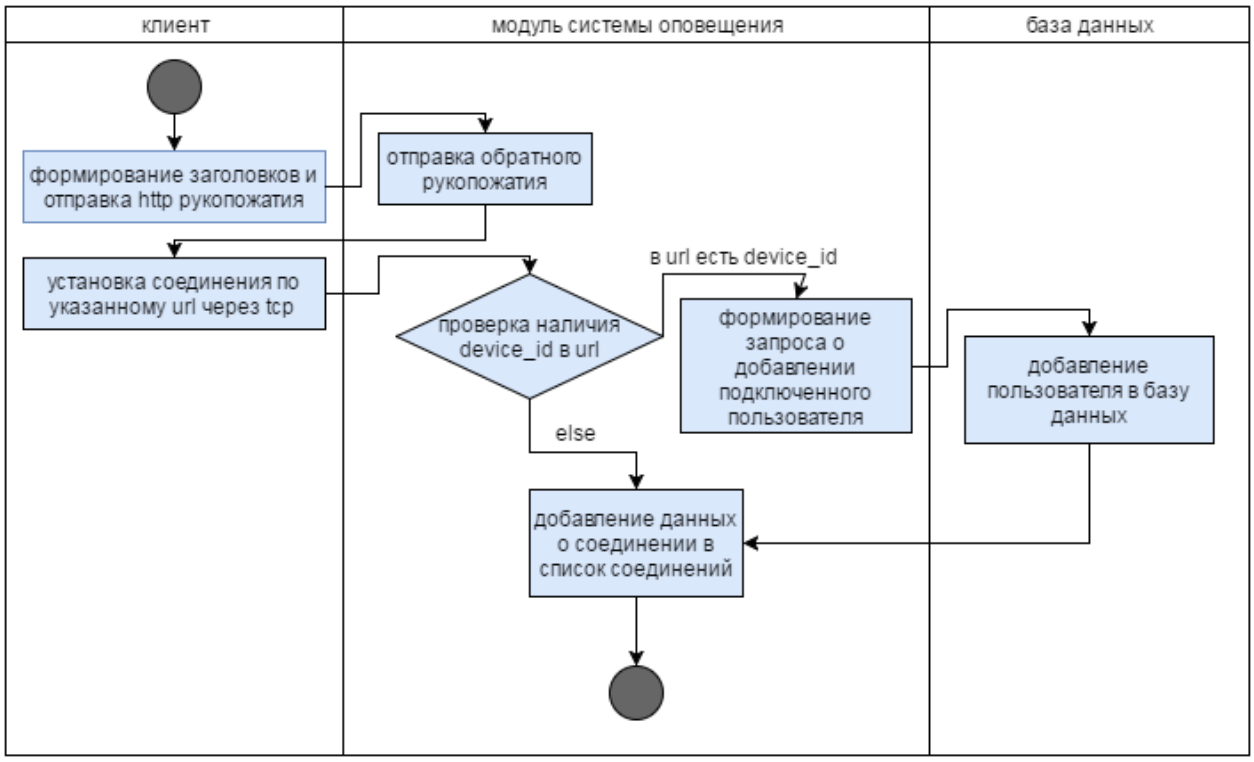

Рисунок 1.7 – Установка соединения пользовательским приложением с сервером

Подключение стабилизируется при решении приложения открыть соединение, для чего необходимо отправить http приветствие на сервер и в дальнейшем «подписаться» на рассылку оповещений для определенной группы. Ожидается, что приложение будет отсылать идентификатор при подключении к сокету. Что позволит добавить пользователя в базу и запомнить настройки его оповещений.

Обновление базы, как и оповещение пользователей, срабатывает по расписанию, определенному на сервере. Каждые 10 минут происходит сверка данных в таблицах с расписанием учебных занятий, при помощи алгоритма проверки максимального (последнего) времени корректировки данных для каждой группы в таблице расписаний занятий. Сравнение этих данных, позволит выявить различия в расписании, которое существует в двух базах данных. Группы, у которых были найдены различия, записываются в отдельный список. Затем происходит обновление внутренней базы данных и

последующая рассылка данных, которая действует только по «подпискам» с меткой этой группы или пользователям с интересами в данной группе. Отобразим на схеме механизм оповещения пользователей, при использовании таймера и списка подключений по websocket (рис 1.8).

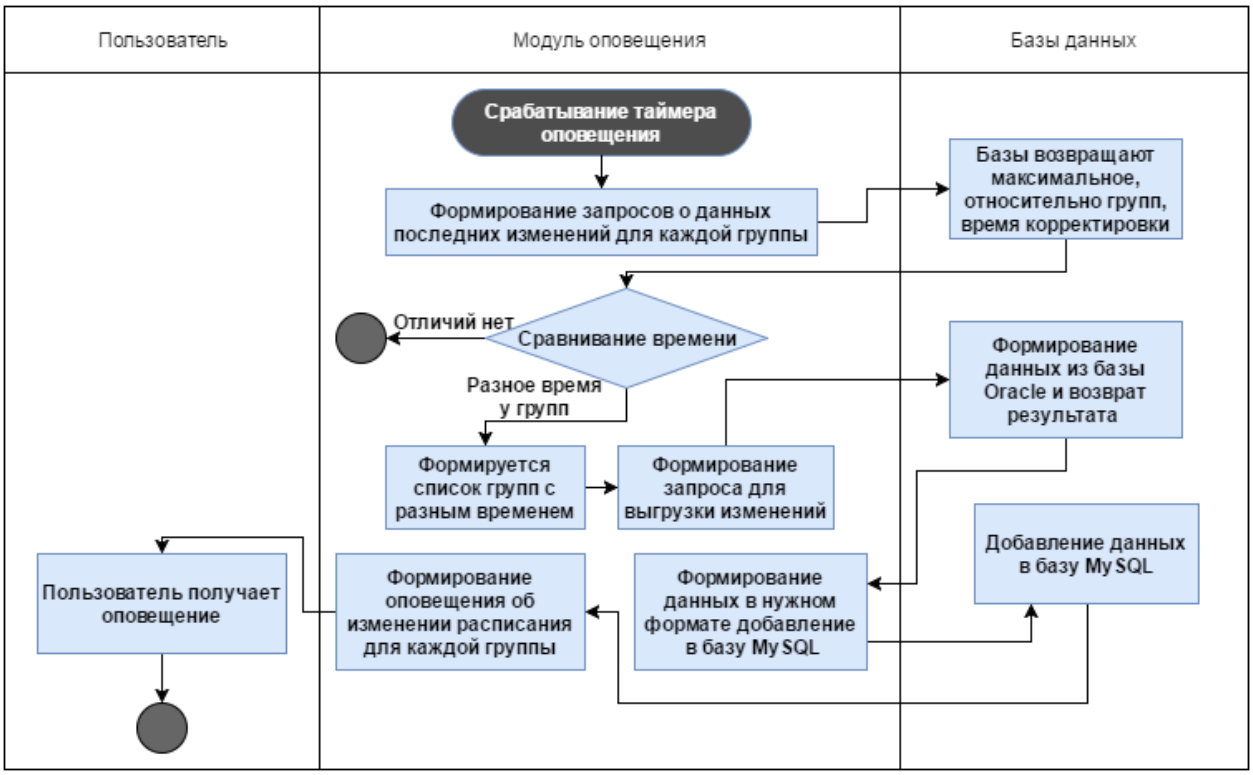

Рисунок 1.8 – Схема механизма оповещения пользователей при срабатывании таймера

В результате составления механизмов рассылки и установки соединения был разработан алгоритм работы системы оповещения ANM, которая в противоположность недостаткам существующей системы оповещения в едином информационном пространстве ТГУ, обладает автоматизированным алгоритмом оповещения.

Для выявления улучшаемых функций, проведем сравнительный анализ существующей системы оповещения, предусмотренной инструкцией по организации исполнения расписания учебных занятий[2], и разработанной системой оповещения ANM. Каждая из систем будет оцениваться по выбранным критериям. Оценка критерию будет присваиваться по 10-ти бальной шкале, где 10 баллов характеризуют систему как соответствующую ожиданиям критерия, 5 баллов соответствуют наполовину и 0 баллов не

удовлетворяют критерию. Для этого составим таблицу, с указанием баллов по критериям и полному отчету в каждой ячейке по выставленной оценке (таблица 1.1).

Таблица 1.1 – Сравнительный анализ систем оповещения.

Критерий оценки системы	Планируемая система оповещения ANM.	Действующая система оповещения
Скорость оповещения студентов	9/10 Автоматическая система мгновенно оповещает всех пользователей.	2/10 Многочисленные звонки студентам, бумажка на двери аудитории со сведениями переноса.
Количество кадров необходимых для полного оповещения	8/10 Необходимо внести лишь изменения в БД, оповещение студентов будет автоматизировано.	3/10 Необходимость оповещения студентов, задействование дополнительных кадров.
Уточнение информации, объем передаваемой информации	10/10 При оповещении, можно указать любую информацию, что никак не повлияет на скорость.	2/10 Объем информации напрямую повлияет на длительность звонков.
Автоматизация	10/10 Система полностью автоматизирована и сама определяет изменения в расписании.	0/10 Стандартная система не предусматривает никакой автоматизации.

Проведение сравнительного анализа показало, что по поставленным критериям планируемая система ANM лидирует, что сильно сократит использование человеческих ресурсов и автоматизирует процесс рассылки оповещений об изменениях в учебных занятиях.

1.3 Формирование требований к разрабатываемой системе оповещения

Был проведен анализ недостатков существующей системы оповещения в едином информационном пространстве ТГУ, в результате которого было решено информатизировать систему оповещения. В результате анализа

разработки алгоритма системы ANM и анализа различий между существующей системой и планируемой, было решено разработать автоматизированный модуль системы оповещения ТГУ. В предполагаемый функционал планируемого модуля системы оповещения входит:

— функционал обмена данными с базами данных, доступ к данным;

— функционал установки соединения по websocket;

— функционал планировщика задач, для периодического обновления данных;

— функционал рассылки данных по существующим соединениям;

— функционал хранения пользовательских групп.

Нефункциональные требования системы описывают атрибуты качества, характеристики программного обеспечения. Данные требования включают в себя:

К нефункциональным требованиям относится:

— программа должна быть разработана на java при использовании фреймворка spring;

— при разработке допускается использование ide как intellij idea;

— прием данных должен осуществляться через протокол http, tcp передаваемые пакеты должны содержать данные в формате json;

— поддержание постоянного соединения должно быть при помощи механизма socket’s;

— тестирование программы может проводиться без использования сторонних модулей или серверов.

Составленные требования позволяют начать проектирование системы.

В результате анализа предметной области были выделены недостатки существующего алгоритма оповещения, проанализированы существующие модули рассылки и разработан собственный алгоритм для модуля рассылки оповещений в едином информационном пространстве ТГУ. Были составлены функциональные и нефункциональные требования к будущей системе оповещения.

Глава 2 Проектирование системы оповещения об изменении расписаний тольяттинского государственного университета

2.1 Общая архитектура системы оповещения

Для того чтобы составить представление об общей архитектуре модуля системы оповещения необходимо представить модель «как должно быть», опираясь на составленную ранее модель «как есть». Составим данную модель на рисунке 2.1.

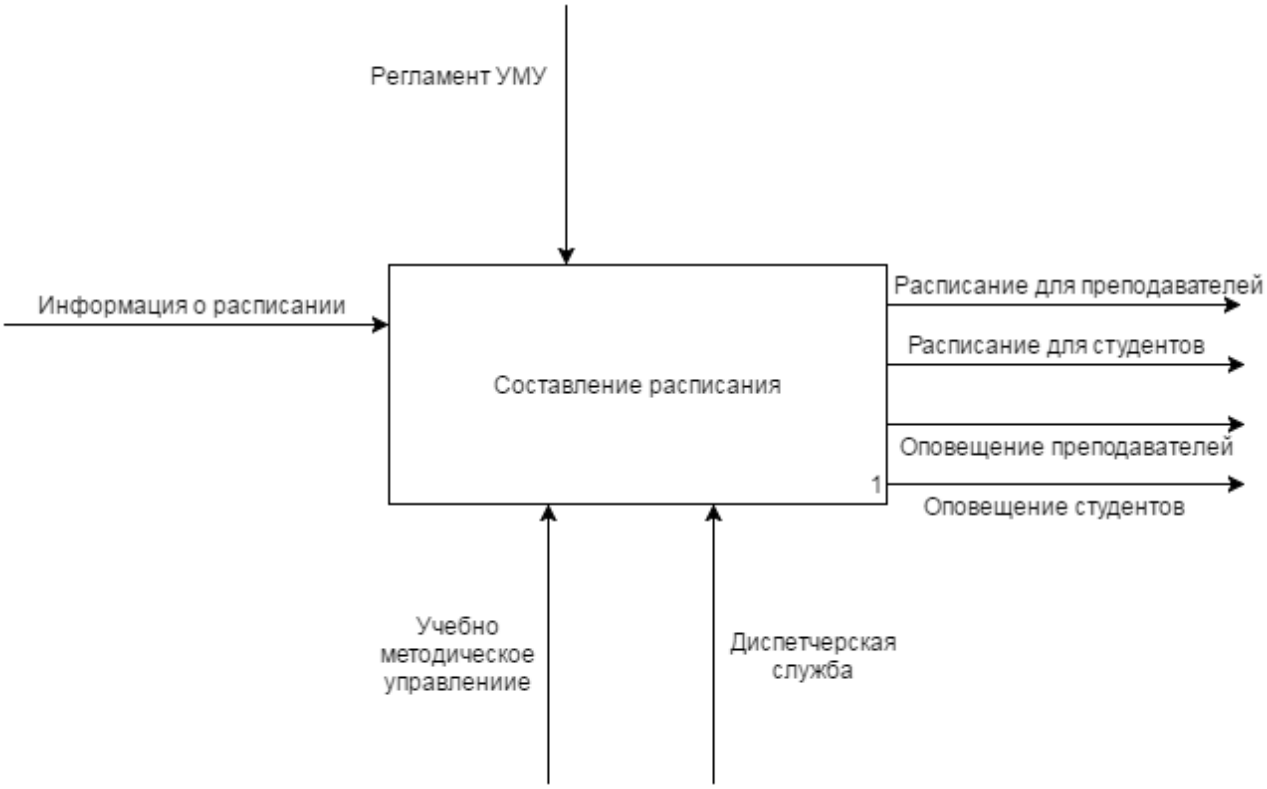

Рисунок 2.1 – Модель процесса составления расписания учебных занятий ТГУ «как должно быть» в нотации IDEF0

Как видно из рисунка 2.1, в отличии от модели «как есть» модель «как должно быть» имеет дополнительный вывод данных в виде оповещений для студентов и преподавателей вуза.

Для более подробного изучения модели составим ее декомпозицию, которая имеет усовершенствованный механизм и формирования расписания и,

не существующий до этого, механизм оповещения. Отразим данные изменения на рисунке 2.2.

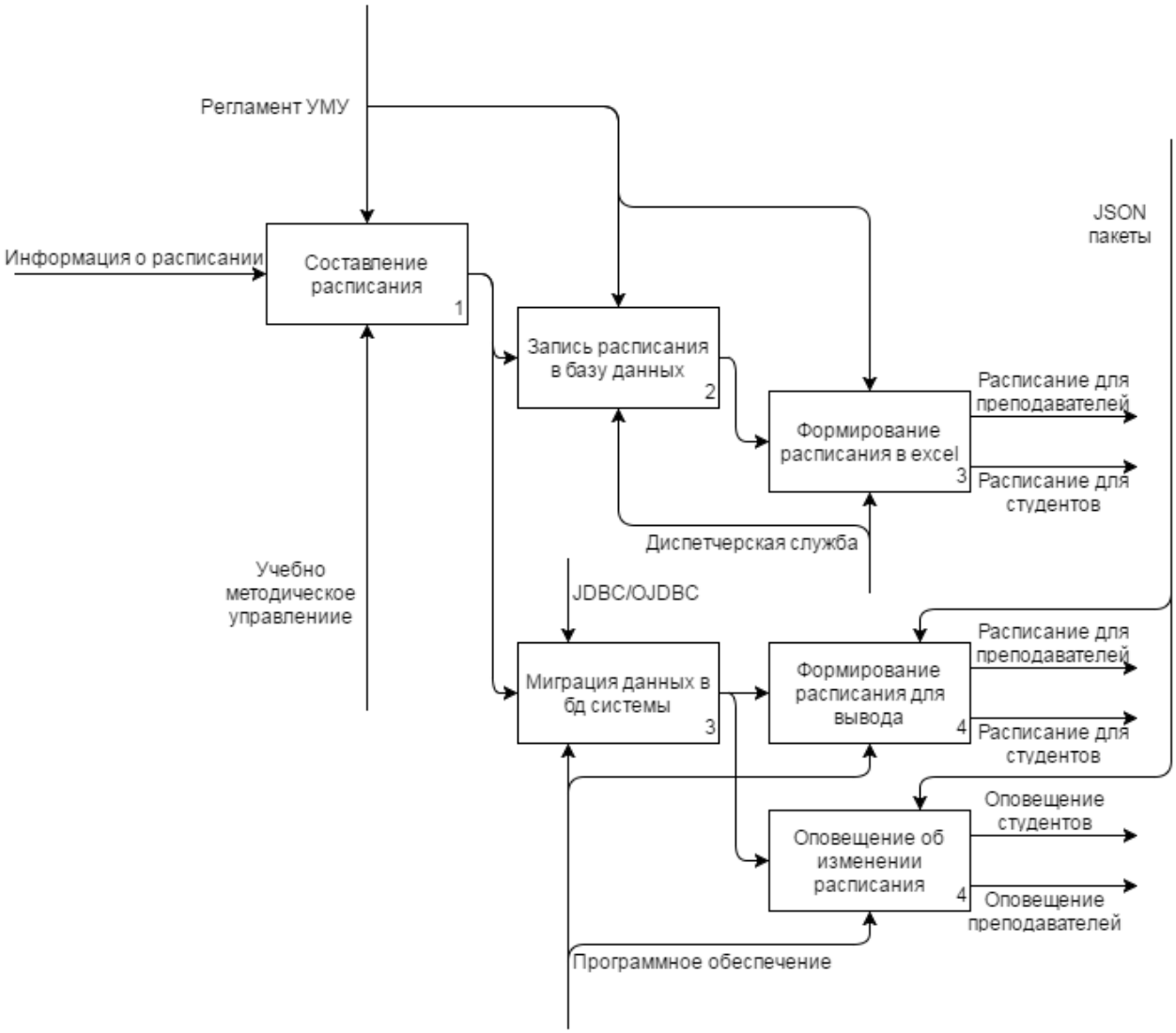

Рисунок 2.2 – Декомпозиция модели «как должно быть»

На данной декомпозиции показан реализуемый модуль оповещения об изменении расписания, с учетом зависимости от стороннего сервера и использования отдельной базы данных. После формирования расписания происходит выгрузка данных в отдельную базу данных и автоматическое реагирование на изменение данных о расписании занятий.

2.2 Выбор метода разработки программного обеспечения

Метод проектирования является основным этапом разработки, так как от него зависят все этапы разработки. Эффективность разработки программного продукта зависит от выбранной методологии и самого метода. От выбора

метода проектирования системы зависят такие важные параметры разработки как:

— общая длительность разработки продукта;

— время выхода первой рабочей версии продукта;

— качество промежуточного и конечного программного продукта.

В зависимости от преследуемой при разработке цели различают спиральную, каскадную и итерационную модели. Выбор модели определяет то, с какой скоростью будет разрабатываться проект, на стоимость разработки, на качество первой версии и количество версий в целом.

Данный проект имеет неизменяемые цели разработки и несколько версий алгоритма оповещения и внутренних функций. При разработке данного продукта, заказчиком является УМУ ТГУ, что означает постоянный контакт с клиентом. Под такой тип технологий разработки программного обеспечения можно выбрать серию подходов называемой Agile (Гибкой). Гибкая серия подходов к разработке программного обеспечения ориентирована на использование итеративной разработки. В приоритетах задач данного проекта является реализация и ранний выпуск с дальнейшим улучшением версий продукта. Суть гибкой серии подходов заключается не только в постоянном контакте с заказчиком, она включает в себя также динамическое формирование требований, это и является ключевым пунктом в выборе методологии. Требования данного проекта будут разрастаться в соответствии с появлением программ, использующих данную программу для получения данных.

2.2.1 Анализ доступных альтернатив методов разработки программного обеспечения

Многие методы разработки программного обеспечения используют ценности и принципы гибкой итеративной технологии программирования, одними из самых популярных из них являются Scrum, Lean, FDD. Рассмотрим их по отдельности.

Scrum – метод разработки программного обеспечения, который прежде всего ставит акцент на более качественном контроле процесса разработки. Данный метод включает в себя тщательный подбор кадров, контроль разработки в виде ежедневных отчетов каждого члена команды. Итерации выпуска версий называются спринтами, согласно распространенному мнению, один спринт длится в среднем 4 недели. Каждому члену команды выдается задание, которое называется историей, при выполнении плана истории ее необходимо вписать в бэклист, описывающий все выполненные истории команды за прошедшее время от начала спринта. В случае если один член группы разработчиков закончил со своими задачами, он должен быть готов помочь остальным членам группы, чтобы ускорить достижение цели спринта.

Вывод: метод scrum подходит для масштабных проектов с огромным штабом людей и упор в ней ставится на качество программного обеспечения при помощи коммуникации и взаимопомощи всех участников разработки друг другу.

Lean – или бережливая разработка, является методом разработки программного обеспечения, включает в себя принципы гибкой разработки. Исходя из названия, методология Lean ставит акцент на тщательном тестировании каждой ветки, чтобы не пришлось возвращаться и переписывать код, а также проводится рефакторинг каждого участка кода. В данной методологии также присутствует поощрение членов группы при качественном и быстром выполнении плана. Lean поддерживает идею постоянного совершенствования и обучение всего персонала новым технологиям. В начальных этапах, ставится огромный акцент на удовлетворение заказчика и проводится полное утверждение требований к продукту. При постоянном спросе Lean определяет какие стороны проекта имеют наибольший спрос и совершенствуется в этих направлениях.

Вывод: метод Lean обеспечивает стремление к перспективным направлениям разработки и рассчитана на постоянное обучение сотрудников.

Требовательна к представлению конечного программного продукта, со стороны заказчика.

FDD – метод разработки, который прежде всего опирается на осязаемый результат работы, как для членов команды так и для заказчика. Планирование и анализ занимают наименьшее время, при разработке методом FDD. Каждый разработчик работает над своим блоком или участком кода, функцией, которые называют свойствами. Функции разбивают на свойства, где свойство занимает по времени разработки не более 2-ух недель, в противном случае, его разбивают на отдельные свойства. Иногда, в соответствии с необходимостью, над свойством работают несколько разработчиков. Регулярные сборки свойств позволяют постоянно обновлять продукт и выпускать новые версии.

Вывод: FDD является методом разработки с быстрым стартом, минимумом проектирования и большим количеством времени на разработку и отладку программы. Данный метод подходит для небольших программных продуктов, рассчитывающих на быстрый выход на рынок, с постоянными обновлениями и добавлением функционала.

2.2.2 Выбор метода разработки программного обеспечения

Проведем сравнительный анализ методологий, выбранных как доступные аналоги. При сравнивании будет использоваться 10-ти бальная шкала, где 10 баллов являются максимально соответствующими по отношению к критерию выбора, 5 баллов удовлетворяют критерий лишь вполовину и 0 баллов вовсе не соответствуют критерию. Результаты сравнивания представлены на таблице 2.1.

Таблица 2.1 – Сравнительный анализ методологий.

Метод	FDD	Scrum	Lean
Быстрый старт	9/10	5/10	3/10
Видимость процесса разработки со стороны заказчика	10/10	6/10	2/10
Общая длительность разработки	9/10	4/10	6/10
Гибкие требования для разработчиков при написании кода	10/10	8/10	6/10

Отсутствие ошибок и багов в программном коде	5/10	7/10	7/10
Легковесность	9/10	2/10	2/10

Как видно из таблицы 2.1 методология FDD является лидирующей по поставленным критериям, что делает ее наиболее подходящей для разработки данного проекта.

В итоге был выбран метод разработки, из гибкой серии подходов к разработке программного обеспечения, под названием Feature driven development, сокращенно FDD.

2.3 Разработка программного обеспечения методологией FDD

Аббревиатура FDD расшифровывается как Feature Driven Development, что в переводе означает «управляемая разработка функционала».

Метод разработки FDD включает в себя 5 этапов:

Этап 1 - Проектирование общей модели.

Этап 2 - Построение списка функций системы.

Этап 3 - Планирование создания функций.

Этап 4 - Проектирование списка функций.

Этап 5 - Реализация спроектированных функций.

Проектирование общей модели – проект, использующий FDD модель, начинает разработку с обозревания объема системы и его содержимого. Далее для каждой области разработки, разделенной на подгруппы, составляются подробные модели предметной области. Из предложенных моделей, для каждой из существующих областей, выбирается отдельная или комбинированная модель предметной области.

Построение списка функций системы – выводы, сделанные на этапе анализа, используются для определения списка функций системы, разложенных на предметные области. Функции системы проектируются как свойства, где свойство часть кода в виде метода, в понятии ООП, на разработку которого уйдет не больше двух недель. Такой подход позволяет максимизировать скорость разработки и выпуска версий. Каждый разработчик в команде

отвечает лишь за свои свойства, их работоспособность и удовлетворение конвенциям.

Необходимо заметить, что отладка программы происходит непосредственно во время ее разработки, с проверкой всех свойств. Проверку выполняет сам разработчик свойства, что позволяет ускорить тестирование во много раз.

Планирование создания функций – на этом этапе, происходит разбиение функционала на свойства и каждый разработчик выбирает свойства, которые он будет разрабатывать.

Проектирование списка функций (свойств) – этап проектирования функций определяет отношения между ними, каждый разработчик проектирует свой набор свойств и создает заготовки(тела) для методов этих свойств. Нередко при этом используются интерфейсы.

Реализация спроектированных функций (свойств) – данный этап включает в себя реализацию, то есть программирование свойств и объединение их в единые работоспособные функции. По сути эта часть включает в себя еще и тестирование реализованных функций.

Рассмотренные этапы будут выполнены не полностью, т.к. данный проект не представляет собой ничего масштабного и имеет состав из одного разработчика. Вследствие небольшого количества функционала, некоторые этапы будут реализовываться совместно.

2.4 Проектирование базы данных

Проектирование базы данных состоит из трех этапов построения схем баз данных и нормализации. Первым этапом является инфологическое проектирование, которое объективно отражает цели таблиц и столбцов, их связи. Вторым этапом проектирования базы данных является логическое проектирование, на котором выбираются названия столбцов и таблиц. Третьим этапом является физическое проектирование, определяющее типизацию столбцов и типы связей, оно будет описываться в главе реализации. Приступим к проектированию базы данных.

2.4.1 Инфологическое проектирование.

В таблице будут содержаться данные идентификаторов пользователей, массив, состоящий из названий групп интересующих пользователя, время подключения, для того чтобы определить степень осведомленности пользователя. И уникальный номер ячейки хранения данных, для безопасного обновления данных в таблице.

Построим инфологическую модель необходимой таблицы, рисунок 2.3.

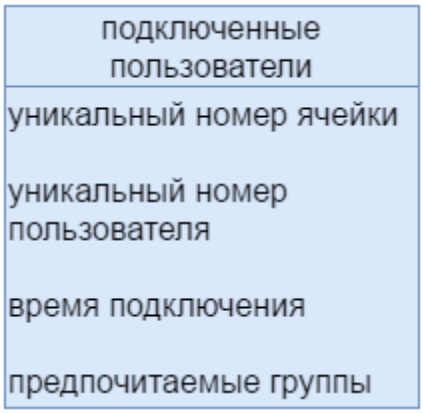

Рисунок 2.3 – Инфологическая модель проектируемой базы данных

2.4.2 Логическое проектирование.

Логическое проектирование очень важно, т.к. присваивание логичных, кратких и соответствующих цели названий во многом упрощает разработку. В соответствии с конвенцией о наименованиях таблицы и столбцов, их названия будет содержать прописные латинские буквы, без цифр, допускается содержание подчеркиваний. Построим логическую модель, рисунок 2.4.

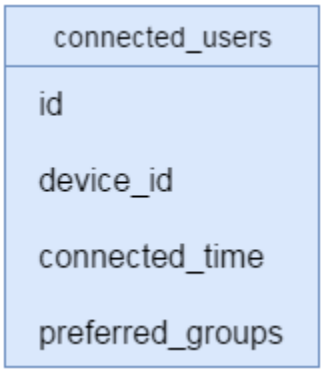

Рисунок 2.4 – Логическая модель проектируемой базы данных

2.5 Проектирование общей модели и построение функций системы оповещения

Проектирование общей модели заключается в выделении основных функций. Данное приложение содержит несколько основных функций:

— Операции доступа к базам данных.

— Операция нахождения групп с измененными записями из таблицы расписания учебных занятий.

— Операция обновления базы данных.

— Периодическое обновления базы данных (создание планировщика).

— Создание обратной связи при подключении через протокол websocket.

Операции доступа к базам данных – существует 2 базы данных, одна из них предоставляется на как поставщик данных, в данном случае расписания занятий и сопутствующей информации, такой как названия групп, распределение потоков, названия аудиторий и т.д. Вторая база данных создана реплецирования данных из первой базы, в упрощенном виде. И списка пользователей подключенных через websocket-ы. Для работы нашей системы нам необходимы обе базы данных. Доступ к ним предоставляется через сопутствующий программный интерфейс, расположенный на сервере. Для полной реализации данной функции, необходимо реализовать доступ к объекту connected_users второй базы данных.

Операция нахождения групп с измененными записями из таблицы расписания учебных занятий – данная функция должна автоматизировано, при помощи операций доступа к базам данных, находить различия в содержании таблиц расписания учебных занятий и возвращать набор групп, расписание которых было изменено.

Операция обновления базы данных – обновление базы данных должно происходить автоматически во время обнаружения несоответствий в базе, вследствие операции нахождения групп с измененными записями из таблицы расписания учебных занятий.

Периодическое обновление базы данных(создание планировщика) – для оповещения пользователей об изменениях в расписании, необходимо с определенной периодичностью вызывать функцию нахождения групп с измененными записями из таблицы расписания учебных занятий. Для этого необходимо реализовать таймер, периодически вызывающий указанную функцию.

Создание обратной связи при подключении к серверу через протокол websocket – данная функция должна конфигурировать все входящие соединения технологии websocket и определять адреса, по которым происходит рассылка данных, а также добавлять пользователей в базу данных, если они прислали свои уникальные идентификаторы.

С проектированием функционала окончено, далее переходим к планированию создания функций, разбиение функций на свойства.

2.6 Планирование создания функций и проектирования свойств

Основная цель **планирования созданий функций** это разбиение их на свойства в случае, если они являются труднореализуемыми. В данном случае все функции легко реализуемы. Логически удобно выделить 4 свойства по существующим функциям программы.

Свойство 1. Данное свойство будет состоять из функции операции доступа к базам данных, оно является легко реализуемым и тестируемым.\

Свойство 2. Данное свойство будет содержать 2 функции:

— операции нахождения групп с измененными записями из таблицы расписания учебных занятий;

— операция обновления базы данных.

Свойство 3. Данное свойство будет содержать функцию создания обратной связи при подключении к серверу через протокол websocket.

Свойство 4. Данное свойство будет содержать лишь функцию создания планировщика.

Все четыре свойства будут определены к одному разработчику и имеют наиболее удобный, рекомендуемый порядок выполнения по возрастанию, т.к. в таком порядке их будет удобнее тестировать, не возвращаясь к предыдущим участкам кода в случае возникновения ошибок.

При проектировании свойств методом разработки программного обеспечения FDD рекомендуется создать пустые тела функций и классов и определить связи между ними.

Данное свойство будет состоять из классов, необходимых для доступа к данным из базы данных. Для этого необходимо создать класс ConnectedUsers, имитирующий таблицу базы данных для удобства выборки и добавления данных. Рисунок 2.5 реализует заготовку под ConnectedUsers класс, его методы, параметры и конструкторы.

```
public class ConnectedUsers {
    private int id;
    private String device_id;
    private String connected_time;
    private List<Object> preferred_groups;
    public ConnectedUsers() {}
    public ConnectedUsers(String device_id, String preferred_groups) {}
    @Override
    public String toString() {}
    @Override
    public boolean equals(Object o) {}
    @Override
    public int hashCode() {}
    public List<Object> getPreferred_groups() {}
    public void setPreferred_groups(List<Object> preferred_groups) {}
    public String getConnected_time() {}
    public void setConnected_time(String connected_time) {}
    public String getDevice_id() {}
    public void setDevice_id(String device_id) {}
    public int getId() {}
    public void setId(int id) {}
}
```

Рисунок 2.5 – Заготовка класса ConnectedUsers

Также необходимо создать класс, позволяющий автоматически преобразовывать результирующие таблицы в список (List) из класса ConnectedUsers. Класс Connected UsersMapper будет использоваться как параметр, при выполнении запроса в DAO классе. Тип возвращаемого значения выборки будет зависеть от типа класса, используемого при включении интерфейса RowMapper. Рисунок 2.6 демонстрирует заготовку для переопределения класса RowMapper, позволяющий получать значения из результата запроса.

```
public class ConnectedUsersMapper implements RowMapper<ConnectedUsers> {
    @Override
    public ConnectedUsers mapRow(ResultSet resultSet, int i) throws SQLException {
        return new ConnectedUsers();
    }
}
```

Рисунок 2.6 – Заготовка для переопределения метода mapRow

Вышеперечисленные классы свойства 1 будут использоваться непосредственно в классе доступа к данным таблицы connected_users.

```
public class ConnectedUsersDAO {
    private JdbcTemplate jdbcTemplateObjectMySQL;
    private SimpleJdbcInsert jdbcInsertMySQL;

    public ConnectedUsersDAO(JdbcTemplate jdbcTemplateMySQL) {}

    public void setDataSourceMySQL(JdbcTemplate jdbcTemplate) {}

    public List<ConnectedUsers> findAllMySQL() {}

    public List<ConnectedUsers> findAllMySQL(String device_id) {}

    private int getId(String device_id) {}

    public void addGroupToUser(String device_id, String group) {}

    private List<Object> getGroupsFromUser(String device_id) {}

    public void deleteAllMySQL() {}

    public void addListMySQL(List<ConnectedUsers> list) {}

    public void addRowMySQL(ConnectedUsers record) {}
}
```

Рисунок 2.7– Заготовка для класса реализующего методы обмена данными с таблицей connected_users

Рисунок 2.7 показывает заготовку для класса Connected UsersDAO, содержащий все методы для обмена данными с базой данных и параметр, содержащий подключение к базе данных.

Проектирование свойства 2. Данное свойство состоит из методов класса MainJDBCTemplate. Это обширный класс, содержащий сформированное подключение к базам данных и методы использования data access object (DAO) классы, через которые и происходит непосредственный контакт с базами. На рисунке 2.8 отображены заготовки для функций свойства 2.

```
public List<String> updateDatabase() {...}

private void collectDataPrevCOS(List<ContentOfSchedule> prevDataCOS) {...}

public void updateAllExceptSchedule() {...}

private List<ContentOfSchedule> deleteAllMySQL() {...}
```

Рисунок 2.8– Заготовки методов класса MainJDBCTemplate

Проектирование свойства 3. Данное свойство состоит из конфигурационного класса WebSocketConfig, которые переопределяет стандартные методы класса Abstract Web Socket Message Broker Configurer. На рисунке 2.9 реализованы заготовки для конфигурирования websocket-ов.

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig extends AbstractWebSocketMessageBrokerConfigurer {
    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {}
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {}
    @Override
    public void configureClientInboundChannel(ChannelRegistration registration){}
}
```

Рисунок 2.9 – Реализация заготовок для конфигурирования подключений по протоколу websocket

Проектирование свойства 4. Данное свойство состоит из конфигурационного файла для планирования задач. Рисунок 2.10 показывает реализацию заготовки для планировщика задач.

```java
@Configuration
@EnableAsync
@EnableScheduling
public class ScheduleUpdatesConfig {

    @Autowired
    SimpMessagingTemplate simpMessagingTemplate;

    private ApplicationContext context;
    MainTemplateJDBC mainTemplateJDBC;

    @Scheduled(fixedRate = 600000, initialDelay = 60000)
    public void updateDatabase() {

    }
}
```

Рисунок 2.10 – Заготовка для реализации планировщика задач

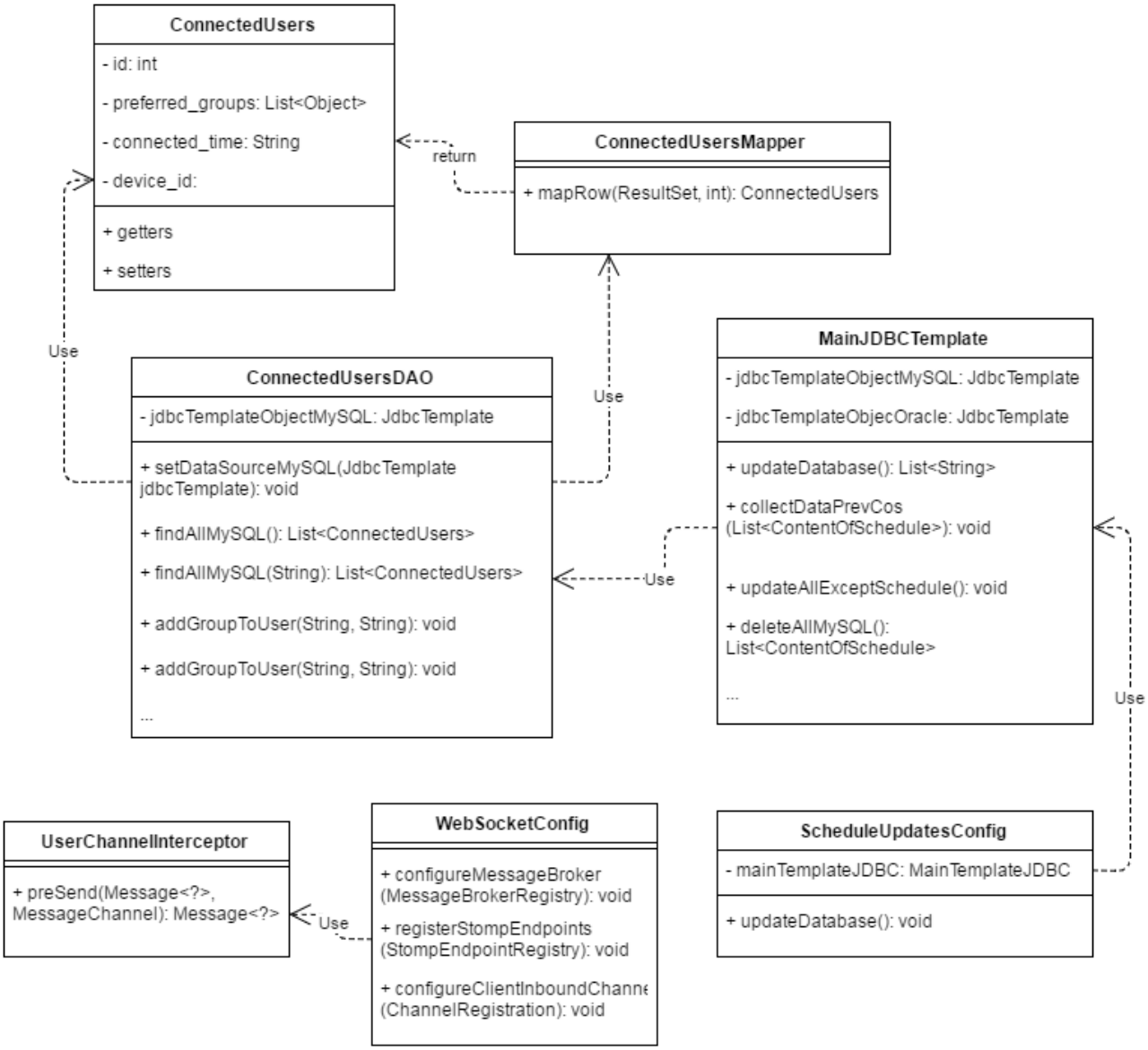

Рисунок 2.11 – UML диаграмма классов модуля системы оповещения

На рисунке 2.11 изображена финальная UML диаграмма всех существующих в программы классов и связей между ними. Используя данную диаграмму и описание всех методов и классов легко реализовать саму программу.

В этой главе была выбрана методология разработки программного обеспечения и выполнено планирование по выбранной методологии. Составлены и разбиты на свойства функции приложения, а так же, непосредственно в самом коде, сформированы заготовки в виде пустых тел классов и методов. Спланировано выполнение полученных функций и описано их представление в программе.

Глава 3 Кодирование модуля оповещения в едином информационном пространстве ТГУ

3.1 Реализация базы данных для модуля оповещения

Физическое проектирование состоит из типизации столбцов, выбора базы данных и реализации самой базы данных. При этом выбор базы данных пропущен, по причине использования готовой базы данных[1]. При помощи программы Workbench являющейся визуальным интерфейсом, упрощающей работу с СУБД MySQL, была создана таблица connected_users (рис 3.1).

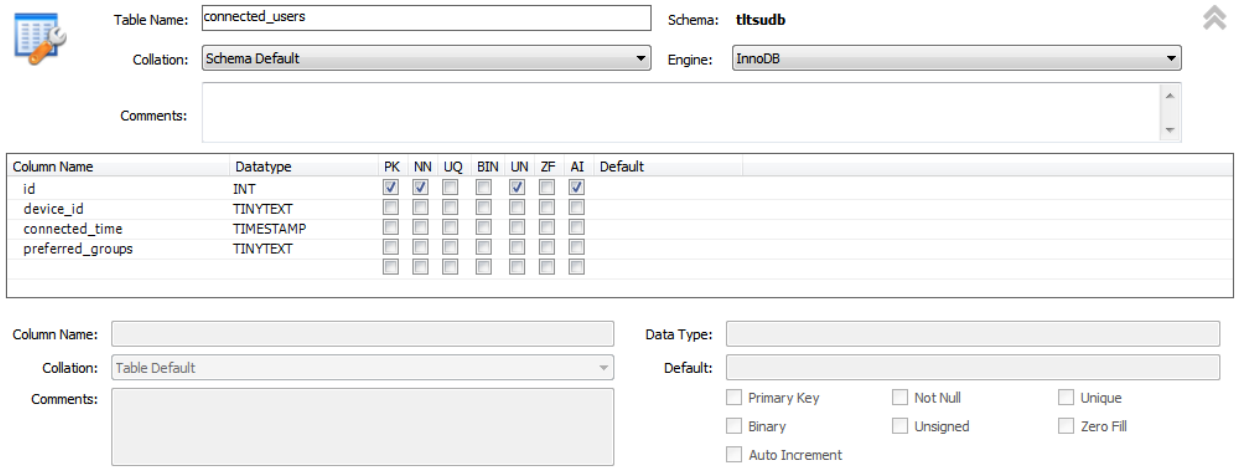

Рисунок 3.1 – Создание таблицы connected_users в MySQL Workbench

При генерации данных, значения будут указываться лишь для полей device_id и preferred_groups. .На этом создание базы данных окончено.

3.2 Реализация свойств модуля оповещения

Реализация, согласно проектированию, состоит из 4 свойств, которые будут реализовываться в том же порядке, в каком проектировались.

Реализация свойства 1. Реализуем класс ConnectedUsers. Листинг кода предоставлен в приложении А. Реализуем класс ConnectedUsersMapper, далее предоставлен листинг с его реализацией. Данный класс будет использоваться при выборке данных, и возвращать сформированные экземпляры класса ConnectedUsers. При реализации необходимо учесть имена таблицы, так как при помощи запрос названия ключей в ResultSet может варьироваться. Для

одного из полей зададим тернарный оператор, для проверки пустого значения, так как в случае отсутствия даты выборки не произойдет. Остальные поля, как правило, имеют автоматически устанавливаемые значения и отсутствие значения в них минимально. На рисунке 3.2 изображена итоговая реализация

```
public class ConnectedUsersMapper implements RowMapper<ConnectedUsers> {
    @Override
    public ConnectedUsers mapRow(ResultSet resultSet, int i) throws SQLException {
        ConnectedUsers connectedUsers = new ConnectedUsers();
        connectedUsers.setConnected_time(resultSet.getTimestamp("connected_time") == null ?
                "" :
                resultSet.getTimestamp("connected_time").toString());
        connectedUsers.setDevice_id(resultSet.getString("device_id"));
        connectedUsers.setId(resultSet.getInt("id"));
        connectedUsers.setPreferred_groups(new SimpleJsonParser().parseList
                (resultSet.getString("preferred_groups")));
        return connectedUsers;
    }
}
```

класса ConnectedUsersMapper.

Рисунок 3.2 – Реализация класса ConnectedUsersMapper

Далее реализуем класс ConnectedUsersDAO, в результате реализации, класс получился больше ожидаемого объема, были добавлены методы выборки по уникальному идентификатору и добавление группы для пользователя, которую он прослушивает. На рисунке 3.3 частично изображена реализация методов класса доступа к данным.

```
    private int getId(String device_id) {
        List<ConnectedUsers> list = this.findAllMySQL(device_id);
        if (list.size() == 0) return 0;
        return this.findAllMySQL(device_id).get(0).getId();
    }

    public void addGroupToUser(String device_id, String group) {
        int id = this.getId(device_id);
        if (id == 0) {
            System.out.println("no such user");
            return;
        }
        List<Object> groups = this.getGroupsFromUser(device_id);
        groups.add(group);
        this.jdbcTemplateObjectMySQL.execute("update connected_users set preferred_groups = '" +
                groups.toString() + "'" +
                "where `id` = " + id);
    }

    private List<Object> getGroupsFromUser(String device_id) {...}

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

    public void addListMySQL(List<ConnectedUsers> list) {...}

    public void addRowMySQL(ConnectedUsers record) {
        SqlParameterSource parameters = new MapSqlParameterSource()
                .addValue("device_id", record.getDevice_id())
                .addValue("preferred_groups", record.getPreferred_groups().toString());
        this.jdbcInsertMySQL.execute(parameters);
    }
}
```

Рисунок 3.3 – Реализация наиболее используемых методов класса

Реализация свойства 2. При реализации свойства 2 было закодировано 4 метода MainJDBCTemplate. В порядке использования методов, вначале были реализованы collectDataPrevCOS, обновляющий все данные, кроме самого расписания, для того чтобы узнать различия между прошлой и текущей версиями расписания учебных занятий и deleteAllMySQL, удаляющий все данные и отдающий запись устаревшей версии расписания. Данная реализация является временной и подлежит доработке. На рисунках 3.4 и 3.5 предоставлены реализации методов данного свойства.

```
private List<ContentOfSchedule> deleteAllMySQL() {
    /**
     * удаляет все данные из таблиц в правильном порядке
     * возвращает предыдущие записи
     */

    /** 2 lvl **/
    List<ContentOfSchedule> dataContentOfSchedule = contentOfScheduleDAO.findAllMySQL();
    contentOfScheduleDAO.deleteAllMySQL();

    /** 1 lvl **/
    auditoriumDAO.deleteAllMySQL();
    subGroupDAO.deleteAllMySQL();
    groupDAO.deleteAllMySQL();

    /** 0 lvl **/
    kindOfWorkDAO.deleteAllMySQL();
    lecturerDAO.deleteAllMySQL();
    typeOfAuditoriumDAO.deleteAllMySQL();
    streamDAO.deleteAllMySQL();
    disciplineDAO.deleteAllMySQL();
    chairDAO.deleteAllMySQL();
    buildingDAO.deleteAllMySQL();
    return dataContentOfSchedule;
}
```

Рисунок 3.4 – Реализация метода deleteAllMySQL

```
public void updateAllExceptSchedule() {
    collectDataPrevCOS(deleteAllMySQL());
}
```

Рисунок 3.5 – Реализация метода updateAllExceptSchedule

Представленная на рисунках реализация является начальным вариантом, в дальнейшем необходимо переписать содержимое данных методов и избежать удаления всей базы данных для простого обновления данных. Предполагается реализация поиска изменений в каждой из существующих таблиц, с

последующим обновлением и добавлением отсутствующих или обновленных столбцов.

```
private void collectDataPrevCOS(List<ContentOfSchedule> prevDataCOS) {
    /**
     * этот метод копирует все таблицы из oracle в mysql, кроме
     * contentOfSchedule, чтобы узнать какие были изменения
     * соблюден порядок формирования внешних ключей
     * первыми создаются таблицы, где не используются ключи и т.д.
     */

    /** 0 lvl **/
    buildingDAO.addListMySQL(buildingDAO.findAllOracle());
    chairDAO.addListMySQL(chairDAO.findAllOracle());
    disciplineDAO.addListMySQL(disciplineDAO.findAllOracle());
    streamDAO.addListMySQL(streamDAO.findAllOracle());
    typeOfAuditoriumDAO.addListMySQL(typeOfAuditoriumDAO.findAllOracle());
    lecturerDAO.addListMySQL(lecturerDAO.findAllOracle());
    kindOfWorkDAO.addListMySQL(kindOfWorkDAO.findAllOracle());

    /** 1 lvl **/
    groupDAO.addListMySQL(groupDAO.findAllOracle());
    subGroupDAO.addListMySQL(subGroupDAO.findAllOracle());
    auditoriumDAO.addListMySQL(auditoriumDAO.findAllOracle());

    /** 2 lvl **/
    contentOfScheduleDAO.addListMySQL(prevDataCOS);
}
```

Рисунок 3.6 – Реализация метода collectDataPrevCOS

Главный метод updateDatabase, обновляющий все данные возвращает список групп, расписание которых было обновлено или изменено. Выполнив специальный запрос на получение данных по последним внесенным в расписание изменениям для каждой группы в отдельности данный метод выполняет несколько этапов проверки на идентичность существующих групп в двух базах. В случае если были найдены отклонения от эталонной базы Oracle, проводится неполное обновление базы, с записью групп, у которых было изменено расписание учебных занятий, затем происходит вызов методом самого себя, в этом случае проверка изменений учебных занятий произойдет уже после добавление отсутствующих данных в базе данных MySQL, что позволит найти все отличия в расписании. Реализация метода updateDatabase предоставлена в приложении А.

Реализация свойства 3. Для реализации механизма общения клиента с сервером посредством технологии websocket были использованы стандартные средства Spring framework, при помощи конфигурационных классов, добавлен адрес подключения клиентов «/hello» и используемый префикс, при отправке любых сообщений «/app». Рассылка сообщений происходит через SimpleBroker по адресам начинающихся с «/topic». На рисунке 3.7 предоставлена реализация конфигурационного класа, для установки соединений по протоколу websocket.

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig extends AbstractWebSocketMessageBrokerConfigurer {

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker("/topic");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/hello").withSockJS();
    }

    @Override
    public void configureClientInboundChannel(ChannelRegistration registration) {
        registration.setInterceptors(new UserChannelInterceptor());
    }

}
```

Рисунок 3.7 – Реализация конфигурационного класса WebSocketConfig

Вторая часть реализации свойства состоит из обработки запросов от пользовательского приложения, с данными о предпочтительных группах и уникального идентификатора пользователя. Наличие уникального идентификатора необязательно для пользователя и служит для восстановления предыдущих данных о пользователе. Чтобы реализовать эту часть свойства необходимо переопределить метод preSend класса ChannelInterceptorAdapter. Метод preSend «перехватывает» данные о всех входящих сообщениях на адреса указанные в конфигурационном классе. Таким образом можно генерировать данные непосредственно при подключении пользователя или отправки им данных и многое другое. В переопределенном методе preSend реализована

проверка на тип сообщения и при подписке на адрес с уникальным значением длинной более 15-ти символов, создается запись в базе данных. В дальнейшем подключенный пользователь может отправлять данные в виде наименований групп для добавления их в список слушаемых. На рисунке 3.8 показана реализация переопределения метода preSend.

```
public class UserChannelInterceptor extends ChannelInterceptorAdapter {
    @Override
    public Message<?> preSend(Message<?> message, MessageChannel channel) {
        if (message.getHeaders().get("simpMessageType").toString().equals("SUBSCRIBE")) {
            String[] dest = ((Map)message.getHeaders().
                    get("nativeHeaders")).
                    get("destination").toString().split("/");
            if (dest.length == 3 && dest[2].length() > 15) {
                ApplicationContext context = new ClassPathXmlApplicationContext("DatabaseBeans.xml");
                MainTemplateJDBC mainTemplateJDBC = (MainTemplateJDBC)context.getBean("mainTemplateJDBC");
                mainTemplateJDBC.addConnUser(dest[2].substring(0, dest[2].length() - 1), "[]");
            }
        }
        StompHeaderAccessor accessor = StompHeaderAccessor.wrap(message);
        StompCommand command = accessor.getCommand();
        return message;
    }
}
```

Рисунок 3.8 – Реализация переопределения метода preSend

Реализация свойства 4. Последним реализуется планировщик, переодический вызывающий метод обновления данных базы данных. Для этого используем средства конфигурации Spring framework и при помощи аннотаций устанавливаем каждые 10 минут, при запуске сервера через 1 минуту, сверку данных и рассылку данных по группам, данные по которым произошли изменения. Рассылка данных происходит при помощи SimpMessagingTemplate, предоставляемой Spring framework. На рисунке 3.9 отображена реализация данного свойства.

```
@Configuration
@EnableAsync
@EnableScheduling
public class ScheduleUpdatesConfig {

    @Autowired
    SimpMessagingTemplate simpMessagingTemplate;

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

    @Scheduled(fixedRate = 600000, initialDelay = 60000)
    public void updateDatabase() {
        for(String group : mainTemplateJDBC.updateDatabase()) {
            simpMessagingTemplate.convertAndSend("/topic/" + group, mainTemplateJDBC.findAllMySQL(group));
        }
    }
}
```

Рисунок 3.9 – Реализация конфигурационного класса планировщика

Все свойства были разработаны, далее необходимо провести тестирование всей системы.

3.3 Тестирование модуля системы оповещения

Тестирование будет состоять из совместной проверки всех свойств с использованием сервера, на котором расположен модуль. Для этого при помощи maven необходимо построить рабочи й объект и запустить его на локальной машине, совместно с базами данных. При тестировании был использован Spring framework версии 1.3.5 RELEASE.

Чтобы успешно протестировать приложение, необходимо разработать клиента, подключающегося к серверу с использованием протокола websocket, получающего уведомления в определенные промежутки времени и выполняющего операции отправки данных на сервер, для добавления данных о пользователе в базу данных. Для этого был разработан файл index.html, который при помощи javascript создает заголовки и подключается к серверу, при этом пользователь может при помощи интерфейса отправлять данные о том, какие группы он прослушивает. Подключение по websocket к серверу происходит при помощи функции представленной на рисунке 3.10.

```
function connect() {
    console.log("started");
    var socket = new SockJS(document.getElementById('url').value);
    console.log(socket);
    stompClient = Stomp.over(socket);
    console.log(stompClient);
    stompClient.connect({}, function(frame) {
        setConnected(true);
        console.log('Connected: ' + frame);
        stompClient.subscribe('/topic/'+document.getElementById('msg').value, function(greeting){
            showGreeting(greeting.body.toString());
        });
    });
}
```

Рисунок 3.10 – Функция установки соединения по websocket

При отправке сообщений формируется json с данными о id пользователя и группе, которую он желает прослушивать, как показано на рисунке 3.11.

```
function sendGroup() {
    var group = document.getElementById('group').value;
    stompClient.send("/app/hello", {},
            JSON.stringify({
                'group': group,
                'id':document.getElementById('msg').value
            }));
    console.log(JSON.stringify({ 'group': group }));
}
```

Рисунок 3.11 – Функция формирования и отправки данных на канал соединения с сервером

Вывод получаемых сообщений производится непосредственно на страницу при помощи функции, указанной на рисунке 3.12.

```
function showGreeting(message) {
    var response = document.getElementById('response');
    var p = document.createElement('p');
    p.style.wordWrap = 'break-word';
    p.appendChild(document.createTextNode(message));
    response.appendChild(p);
}
```

Рисунок 3.12 – Вывод получаемых сообщений на html форму

Внешне интерфейс вывода изначально имеет 2 поля ввода, где верхнее поле это идентификатор пользователя при подключении, а нижнее поле адрес отправки сообщений на сервер(рис 3.13).

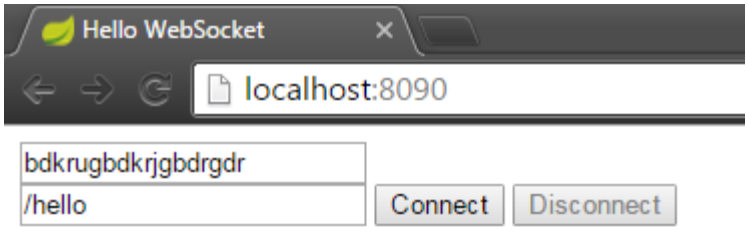

Рисунок 3.13 – Вид интерфейса в браузере

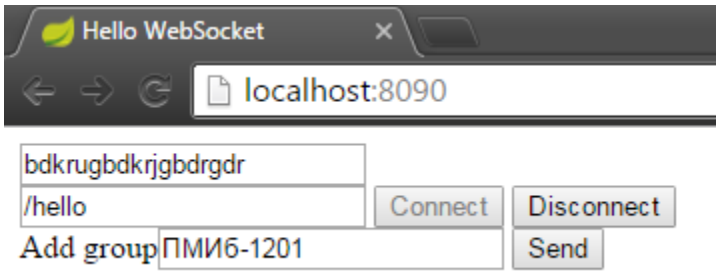

Рисунок 3.14 – Вид успешного подключения к серверу

При нажатии на кнопку Connect на интерфейсе, вызывается функция подключения к серверу, успешное подключение выглядит следующим образом, как изображено на рисунке 3.14.

Для отправки сообщений используется кнопка send, при этом формируется сообщение json, состоящее из идентификатора пользователя (верхнее поле ввода) и названия группы (нижнее поле ввода). Для проверки периодического вывода данных пользователю был реализован метод в созданном планировщике, который каждые 30 секунд делает выборку данных из базы данных и используя все идентификаторы оповещает пользователей о существующих групп у них. Данный подход осуществляет проверку всех разработанных свойств, кроме свойства 2, так как задействует их все по очереди. При получении сервером сообщения json с данными из форм для ввода данных, происходит добавление группы к указанному пользователю, уникальный адрес которого знает только он. Рисунок 3.15 демонстрирует периодический вывод данных, помещая в json ключ content со значением в виде списка групп.

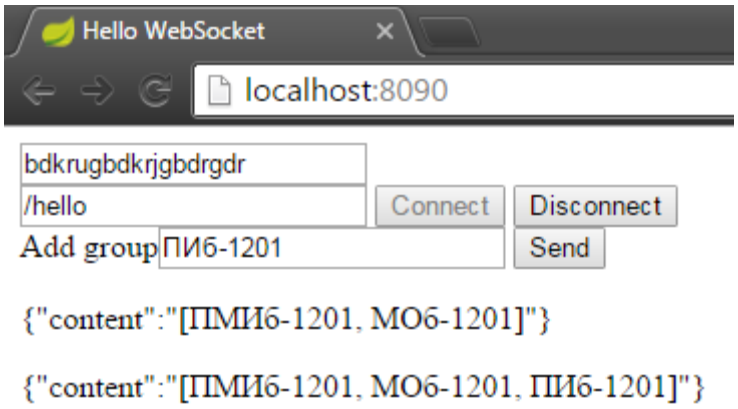

Рисунок 3.15 – Демонстрация оповещения пользователя примитивной выборкой данных из таблицы

Также необходимо проверить наличие записей непосредственно в самой таблице, так как все данные поступают оттуда. Для этого сделаем запрос, запрашивающий выборку всех данных из таблицы. Для выполнения проверки будем использовать Workbench, программу управления СУБД MySQL. Программа обладает динамическим формированием интерфейса для любой таблицы, что значительно упрощает проверку наличия данных при операциях изменения данных. На рисунке 3.16 изображена выборка данных и таблицы connected_users.

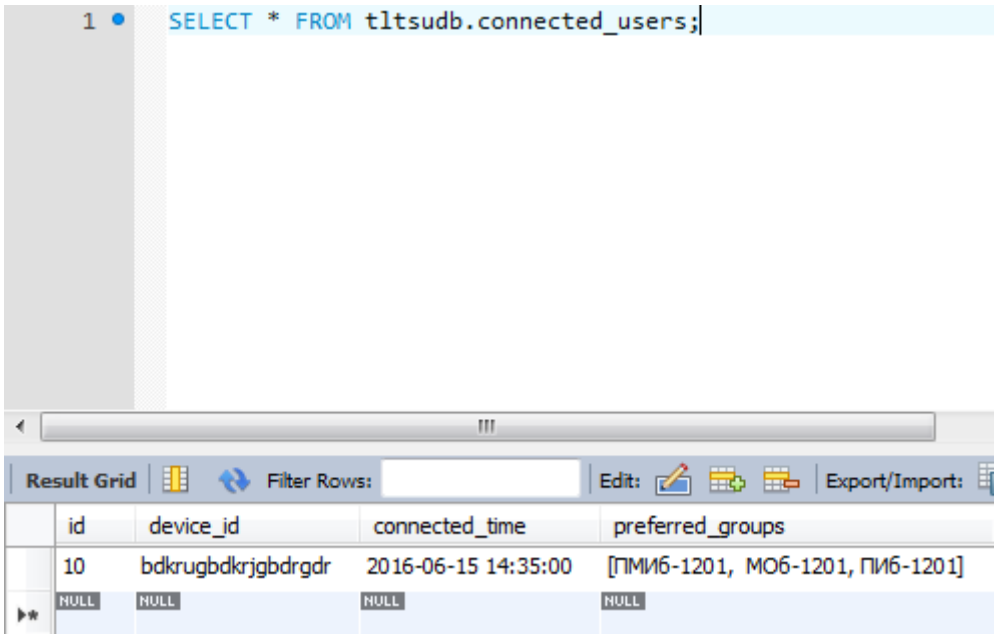

Рисунок 3.16 – Проверка наличия данных в задействованной таблице

Для тестирования необходим сервер. Строим версию приложения при помощи команды mvn package. Запускаем сервер при помощи сформированного .jar файла через командную строку windows(рис 3.17).

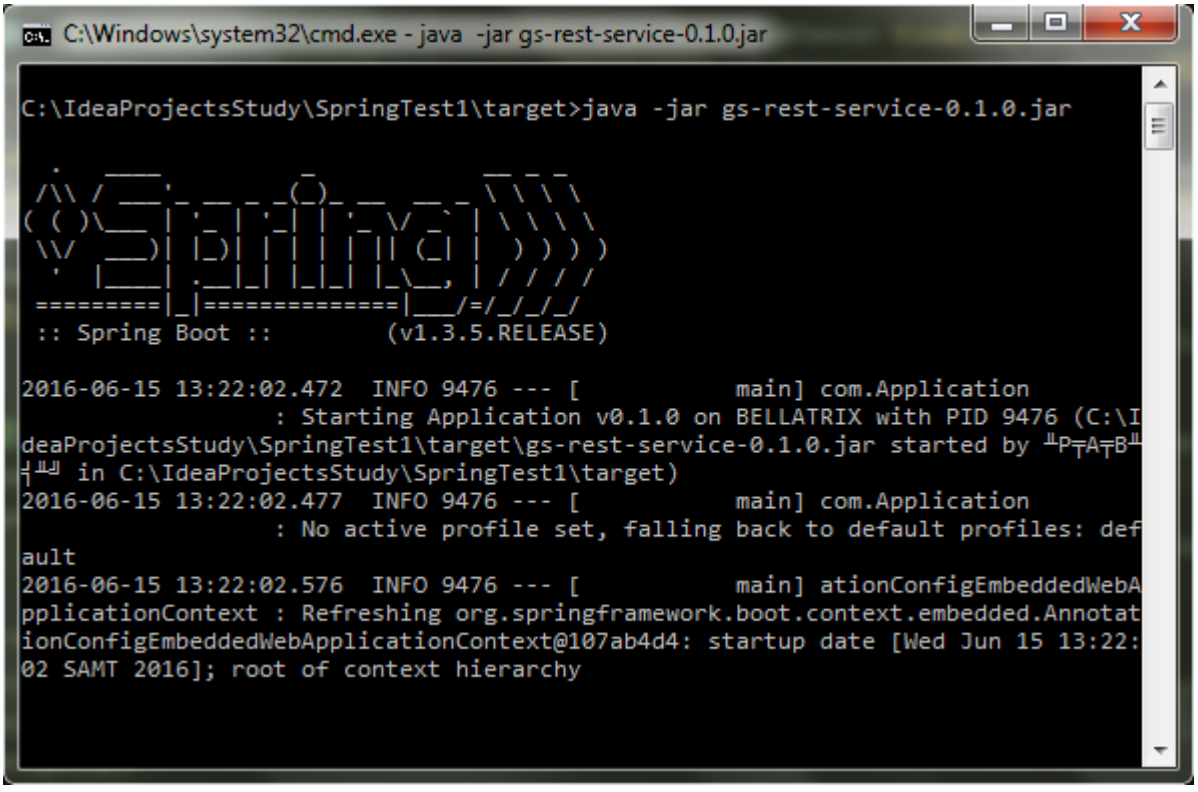

Рисунок 3.17 – Запуск сервера из командной строки

На момент написания данной выпускной квалификационной работы свойство 2 было не протестировано до конца, т.к. выдавало ошибки, остальные свойства работают верно.

3.4 Развертывание модуля оповещения

Развертывание модуля оповещения происходит вместе с включением разработанных конфигурационных классов в сервер созданный при использовании Spring framework версии 1.3.5 и выше. Так же необходимо в случае отсутствия базы данных, необходимо создать ее. При выполнении перечисленных действий, система автоматически использует модуль оповещения и будет работать по адресу сервера. На данном этапе развертывание проводилось в тестировании продукта, отклонений замечено не было.

Развертывание разработанного модуля ANM планируется на выделенный сервер, для возможности дальнейшей разработки клиентского программного интерфейса.

Заключение

При выполнении данной выпускной квалификационной работы была рассмотрена актуальность и новизна темы, выделены объект и предмет исследования, определены цели и задачи работы. Был проведен анализ действующей системы оповещения в ТГУ, рассмотрены приоритетные алгоритмы планируемой системы оповещений и проведено сравнивание систем с рассмотренным алгоритмом и существующим.

В результате анализа было решено информатизировать существующую систему, были сформированы требования по функциональным возможностям планируемой системы.

При проектировании были рассмотрены различные методы разработки программного обеспечения, проведены сравнения по критериям и выбран наилучший вариант. По выбранному методу разработки программного обеспечения были рассмотрены функции модуля системы оповещения и разбиты на свойства. Выбранные свойства были спроектированы в виде заготовок для кодирования.

В результате чего, при помощи интеграционной среды разработки и платформы для разработки серверных приложений Spring framework, была разработана и протестирована первая версия модуля системы оповещения в едином информационном пространстве ТГУ.

Конечным приложением является, готовый к развертыванию, модуль на Java, работающий на удаленном сервере, постоянно обменивающийся данными с конечными пользователями и оповещающий всех подключенных пользователей об изменениях в расписании учебных занятий ТГУ.

Список используемой литературы

Неопубликованные документы

1. Корецкий Д.С.. Разработка технологии интеграции системы составления расписания учебных занятий в единое информационное пространство ТГУ: бакалаврская работа / Д.С. Корецкий. – Тольятти: ТГУ, 2016.

Нормативно-правовые акты

2. Инструкция по организации исполнения расписания учебных занятий (в ред. от 15.01.2016) [Текст] // Тольятти: ТГУ. - 2016 - 11 с.

Периодические издания

3. Белл Ч. Обеспечение высокой доступности систем на основе MySQL [Текст] // Ч. Белл.- Москва : Русская Редакция, 2012. – 624с.

4. Вигерс К. Разработка требований к программному обеспечению [Текст] // К. Вигерс, Д. Битти. -СПб:BHV,2014.-736с.

5. Кон М. Scrum. Гибкая разработка ПО [Текст] // М. Кон. – Вильямс:Москва,2016.-576с.

6. МакГрат, М. Программирование на Java для начинающих [Текст] / М. МакГрат. – М.: Эксмо. – 2016. – 192 с.

7. Фаулер М. Рефакторинг. Улучшение существующего кода [Текст] // М. Фаулер. – СПб : Символ Плюс, 2015. – 432с.

8. Хорстманн К. Java SE 8. Базовый курс [Текст] // К. Хорстманн. – Москва:Вильямс, 2016. – 464с.

9. Шалыто А. А., Синхронное программирование [Текст] // Шалыто А. А., Шопырин Д. Г. Информационно-управляющие системы. – 2014 – 44с.

10. Эккель Б. Философия Java [Текст] // Б. Эккель. – Москва: Питер, 2016.-1168с.

Литература на иностранных языках

11. Andrew Lombardi, WebSocket: Lightweight Client-Server Communications [Text] // Andrew Lombardi – Sebastopol,: O.Reilly Media, Inc., 2015. – 125 p.

12. Dr. Danny Coward, Java WebSocket Programming (Oracle Press) [Text] // Dr. Danny Coward – New York,: McGraw-Hill Education, 2014. – 203 p.

13. Hudson O. Getting started with IntelliJ IDEA [Text] // O. Hudson, Birmingham: Packt Publishing, 2013. – 114p.

14. Ilya Grigorik, High Performance Browser Networking: What every web developer should know about networking and web performance [Text] / Ilya Grigorik – Sebastopol,: O.Reilly Media, Inc., 2013. – 361 p.

15. Krochmalski J. IntelliJ IDEA Essentials [Text] // J. Krochmalski. – Birmingham: Packt Publishing, 2014.-263p.

16. Masoud Kalali, Developing RESTful Services with JAX-RS 2.0, WebSockets, and JSON [Text] / Masoud Kalali – United Kingdom, Birmingham,: Packt Publishing, LTD, 2013. – 107 p.

17. Nicholas S. Williams, Professional Java for Web Applications [Text] / Nicholas S. Williams – Indianapolis, Indiana, : John Wiley & Sons, Inc., 2014. – 865 p.

18. O'Brien H. Agile Project Management: A Quick Start Beginner's Guide To Mastering Agile Project Management [Text] // H. O'Brien. – South Carolina:CreateSpace, 2015.-140p.

19. Olsen D. I The Lean Product Playbook: How to Innovate with Minimum Viable Products and Rapid Customer Feedback [Text] // D. Olsen, T.- New Jersey:Wiley,2015. – 336p.

20. Pro Spring MVC: With Web Flow (Expert's Voice in Spring) [Text] / M. Deinum, K. Serneels, C. Yates and other, – New York : Apress. – 2012. – 596 p.

21. Spring Data [Text] / M. Pollack, O. Gierke, T. Risberg and other, - California: O'Reilly Media. - 1st edition. - 316 p.

Приложение А

Листинг кода реализации классов

Класса ConnectedUsers.

```
public class ConnectedUsers {
    private int id;
    private String device_id;
    private String connected_time;
    private List<Object> preferred_groups;

    public ConnectedUsers() {}
    public ConnectedUsers(String device_id, String preferred_groups) {
        this.device_id = device_id;
        this.preferred_groups = new SimpleJsonParser().parseList(preferred_groups);
    }

    @Override
    public String toString() {
        return "ConnectedUsers{" +
                "id=" + id +
                ", device_id='" + device_id + '\'' +
                ", connected_time='" + connected_time + '\'' +
                ", preferred_groups=" + preferred_groups +
                '}';
    }

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

        ConnectedUsers that = (ConnectedUsers) o;

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

```
        if (getDevice_id() != null ? !getDevice_id().equals(that.getDevice_id()) : that.getDevice_id() != null)
            return false;
        if (getConnected_time() != null ? !getConnected_time().equals(that.getConnected_time()) : that.getConnected_time() != null)
            return false;
        return !(getPreferred_groups() != null ? !getPreferred_groups().equals(that.getPreferred_groups()) : that.getPreferred_groups() != null);
    }

    @Override
    public int hashCode() {
        int result = getId();
        result = 31 * result + (getDevice_id() != null ? getDevice_id().hashCode() : 0);
        result = 31 * result + (getConnected_time() != null ? getConnected_time().hashCode() : 0);
        result = 31 * result + (getPreferred_groups() != null ? getPreferred_groups().hashCode() : 0);
        return result;
    }
    public List<Object> getPreferred_groups() {return preferred_groups; }
    public void setPreferred_groups(List<Object> preferred_groups) {this.preferred_groups = preferred_groups; }
    public String getConnected_time() {return connected_time; }
    public void setConnected_time(String connected_time) {this.connected_time = connected_time; }
    public String getDevice_id() {return device_id; }
    public void setDevice_id(String device_id) {this.device_id = device_id; }
    public int getId() {return id;}
    public void setId(int id) {this.id = id; }
}
```

Метода updateDatabase класса MainJDBCTemplate.

```
public List<String> updateDatabase() {
        List<String> updatedGroups = new ArrayList<>();
        List<GroupMaxModTime> mysqlGroups = groupDAO.findAllGroupsMaxModTimeMySQL();
        List<GroupMaxModTime> oracleGroups = groupDAO.findAllGroupsMaxModTimeOracle();
```

```
    if (mysqlGroups.size() != oracleGroups.size()) {
        updateAllExceptSchedule();
        return this.updateDatabase();
    }
    for (int i = 0; i < mysqlGroups.size(); i++) {
        GroupMaxModTime groupMySQL = mysqlGroups.get(i);
        GroupMaxModTime groupOracle = oracleGroups.get(i);
        if (!groupMySQL.equals(groupOracle)) {
            if (!groupMySQL.getGroupName().equals(groupOracle.getGroupName())) {
                //updateAllExceptSchedule();
                //return this.updateDatabase();
                return updatedGroups;
            }
            if (!groupMySQL.getMaxModTime().equals(groupOracle.getMaxModTime())) {
                updatedGroups.add(groupMySQL.getGroupName());
            }
        }
        else {
            mysqlGroups.remove(i);
            oracleGroups.remove(i);
            i--;
        }
    }
    if (!updatedGroups.isEmpty()) {
        contentOfScheduleDAO.deleteAllMySQL();
        contentOfScheduleDAO.addListMySQL(contentOfScheduleDAO.findAllOracle());
    }
    return updatedGroups;
}
```

Приложение Б

Техническое задание

1. ВВЕДЕНИЕ

1.1. Наименование программы

Модуль асинхронного оповещения(Asynchronous notification module(ANM)).

1.2. Краткая характеристика области применения программы

Область применения ANM рассчитана для внедрения в сервис модернизации раздела «Расписание» в отделе УМУ ТГУ.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1. Основание для проведения разработки

Основанием для разработки является решение расширенного семинара ЦНИТ ТГУ.

2.2. Наименование и условное обозначение темы разработки

Реализация системы оповещения об изменении расписания в едином информационном пространстве ТГУ.

Условное обозначение темы разработки – «РСО-ТГУ»

3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

3.1. Функциональное назначение программы

Функциональным назначением программы является рассылка уведомлений подключенным пользователям об изменении расписания.

3.2. Эксплуатационное назначение программы

Программа должна эксплуатироваться в ФГБОУ ВО «Тольяттинский государственный университет».

Конечными пользователями являются: студенты, преподаватели и иные заинтересованные сотрудники вуза.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ

4.1. Требования к функциональным характеристикам

4.1.1. Требования к составу выполняемых функций

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- создание пула подключаемых пользователей;
- определение пользователей по группам;
- уведомление подключенных пользователей при наличии изменений в интересующем их потоке и группе;

4.1.2. Требования к организации входных данных

Входных данных не предусмотрено.

4.1.3. Требования к организации выходных данных

Выходные данные должны быть организованы в виде пакетов json, поставленные в очередь рассылки уведомлений по пулу подключенных пользователей, с принадлежности пользователей определенной группе и потоку.

4.1.4. Требования к временным характеристикам

Среднее время предоставления ответа на запрос должно оставлять не более 10 сек при условии функционирования надежного интернет соединения.

4.2. Требования к надежности

4.2.1. Требования к обеспечению надежного (устойчивого) функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением организационно-технических мероприятий в виде организации бесперебойного питания технических средств.

4.2.2. Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 5 минут, необходимых на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать 120 минут, достаточных для устранения неисправностей технических средств и переустановки программных средств.

4.2.3. Отказы из-за некорректных действий оператора

Надежность системы не должна зависеть от получаемых данных, система должна работать автоматически.

4.3. Условия эксплуатации

4.3.1. Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

Условия эксплуатации включают в себя условия ГОСТ 15150-69 «Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды».

4.4. Требования к информационной и программной совместимости

4.4.1. Требования к информационным структурам и методам решения

Требования к информационным структурам (файлов) на входе и выходе, а также к методам решения не предъявляются.

4.4.2. Требования к исходным кодам и языкам программирования

Исходные коды программы должны быть реализованы на языке Java.

4.4.3. Требования к защите информации и программ

В Системе должен быть обеспечен надлежащий уровень защиты информации в соответствии с законом о защите персональной информации и программного комплекса в целом от несанкционированного доступа – федеральный закон от 27.07.2006 N 149-ФЗ (ред. от 13.07.2015) "Об информации, информационных технологиях и о защите информации" (с изм. и доп., вступ. в силу с 10.01.2016).

4.5. Специальные требования

Специальные требования не предъявляются.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1. Предварительный состав программной документации

Состав программной документации должен включать в себя:

1. Техническое задание.
2. Текст программы.
3. Описание программы.
4. Программу и методики испытаний.
5. Описание применения.

6. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

6.1. Стадии разработки

Разработка должна быть проведена в три стадии:

- Разработка технического задания.
- Рабочее проектирование.
- Внедрение.

6.2. Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения требований и архитектуры разрабатываемого ПО.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

- Разработка программы.
- Разработка программной документации.
- Тестирование программы.

На стадии внедрения должен быть выполнен этап разработки - подготовка и передача программы.

6.3. Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

- постановка задачи;
- определение и уточнение требований к техническим средствам;
- определение требований к программе;
- определение стадий, этапов и сроков разработки модуля;
- выбор языков программирования;
- согласование и утверждение технического задания;
- составление инструкции для использования программного модуля.

6.4. Исполнители

Руководитель

Кандидат технических наук, доцент Очеповский А.В.

Исполнитель

Студент группы МОб-1201 Скворцов А.А.

7. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

7.1. Виды испытаний

Тестирование готового продукта осуществляется самим разработчиком в виде проверки всех функций и возможностей модуля.

7.2. Общие требования к приемке работы

Сдача работы производится с предоставлением документации с полным описанием функционала ANM и описанием способов его использования.