МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему «Разработка Web API для предоставления данных по расписанию учебных занятий (на примере фгбоу во ТГУ)»

Студент	А. О. Кузовенков	
	(И.О. Фамилия)	*(личная подпись)*
Руководитель	А. Б. Кузьмичев	
	(И.О. Фамилия)	*(личная подпись)*
Консультанты	К. А. Селиверстова	
	(И.О. Фамилия)	*(личная подпись)*

Допустить к защите

Заведующий кафедрой к.т.н., доцент кафедры ПМИ, А.В. Очеповский
(ученая степень, звание, И.О. Фамилия)

_____
(личная подпись)

«_____» _____ 20 ___ г.

Тольятти 2019

АННОТАЦИЯ

Тема: Разработка Web API для предоставления данных по расписанию учебных занятий (на примере фгбоу во ТГУ).

Ключевые слова: расписание, функция, модуль, публикация, подготовка.

Актуальность темы бакалаврской работы заключается в том, что создание Web API для предоставления данных по расписанию учебных занятий улучшит процесс получения актуальной информации об учебных занятиях, заменит ручной труд и сделает расписание доступным практически с любого устройства.

Объектом работы является процесс публикации расписания учебных занятий в ТГУ.

Предметом работы является подготовка данных для публикации расписания учебных занятий в ТГУ.

Целью выпускной квалификационной работы является разработка Web API для предоставления данных по расписанию учебных занятий в ТГУ.

Для достижения поставленной цели в работе были решены следующие задачи:

– проанализировать процесс публикации расписания учебных занятий в ТГУ;

– выработать требования к системе подготовки данных для публикации расписания занятий;

– спроектировать систему подготовки данных для публикации расписания занятий;

– реализовать систему подготовки данных для публикации расписания занятий;

– проанализировать статистику работы системы подготовки данных для публикации расписания занятий.

Работа состоит из введения, трех глав, заключения и списка использованной литературы. В ходе выпускной квалификационной работы был проведён анализ публикации расписания учебных занятий в ТГУ. По

результатам анализа было принято решение о создании системы подготовки данных для публикации расписания учебных занятий. Практическая ценность выпускной работы заключается в том, что разработанная система готова к использованию в ТГУ.

В данной работе содержится 1 приложение, 52 страницы, на которых 15 таблиц, 13 рисунков, 6 блок-схем, 20 источников используемой литературы.

ABSTRACT

The topic of the given graduation work is The development of Web API for providing timetable data at Togliatti State University.

This graduation work consists of 52 pages, including 13 figures, 15 tables and the list of 20 references.

The aim of the work is to develop a Web API that provides timetable data for the further publication, improves process of receiving fresh information about timetable, replaces manual labor and makes the timetable available on any device.

The object of the graduation work is the timetable publication process at Togliatti State University.

The subject of the graduation work is timetable data pre-processing.

The key issue of the graduation work is improvement of the timetable publication process.

During the work, an analysis of timetable publication process at Togliatti State University was carried out. The current timetable publication process is not effective. The dispatch service employee does all work. Since the publication of the timetable is done manually, human errors are possible. According to the analysis, it was decided to create a data pre-processing system for timetable publication. This system represent a web-application that runs on server of Togliatti State University.

In addition, an analysis of the work statistics of this system was conducted. It was found out that using of cache subsystem reduces timetable receiving time by 55,8%.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6
1 АНАЛИЗ ПРОЦЕССА ПУБЛИКАЦИИ РАСПИСАНИЯ УЧЕБНЫХ ЗАНЯТИЙ В ТОЛЬЯТТИНСКОМ ГОСУДАРСТВЕННОМ УНИВЕРСИТЕТЕ .. 8
1.1 Описание структуры Тольяттинского государственного университета 8
1.2 Анализ существующего процесса публикации расписания учебных занятий в Тольяттинском государственном университете 9
1.3 Выработка требований к системе подготовки данных для публикации расписания занятий 10
1.4 Выводы и результаты по разделу 1 12
2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ ПОДГОТОВКИ ДАННЫХ ДЛЯ ПУБЛИКАЦИИ РАСПИСАНИЯ ЗАНЯТИЙ 14
2.1 Проектирование функций системы подготовки данных для публикации расписания занятий 14
2.2 Разработка архитектуры системы подготовки данных для публикации расписания занятий 27
2.3 Выводы и результаты по разделу 2 33
3 КОДИРОВАНИЕ СИСТЕМЫ ПОДГОТОВКИ ДАННЫХ ДЛЯ ПУБЛИКАЦИИ РАСПИСАНИЯ ЗАНЯТИЙ 35
3.1 Выбор инструментов для реализации системы подготовки данных по расписанию учебных занятий 35
3.2 Реализация функций системы подготовки данных для публикации расписания занятий 39
3.3 Анализ статистики работы системы подготовки данных для публикации расписания занятий 46
3.4 Выводы и результаты по разделу 3 48
ЗАКЛЮЧЕНИЕ 49
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 50
ПРИЛОЖЕНИЕ А 52

ВВЕДЕНИЕ

На сегодняшний день во многих сферах человеческой деятельности очень часто используются информационные системы и учебный процесс не является исключением. В образовательных учреждениях сегодня используются различные электронные дневники, ведомости, расписание и т. д. Всё это позволяет сделать процесс обучения максимально комфортным и эффективным.

Учитывая современный темп жизни человека, время является крайне ценным ресурсом. Поэтому в настоящее время очень важно эффективно и быстро предоставлять актуальную информацию об учебных занятиях. Это позволит сэкономить большое количество времени и сосредоточится на более важных делах.

Таким образом актуальность работы заключается в том, что создание Web API для предоставления данных по расписанию учебных занятий улучшит процесс получения актуальной информации об учебных занятиях, заменит ручной труд и сделает расписание доступным практически с любого устройства.

Объектом работы является процесс публикации расписания учебных занятий в ТГУ.

Предметом работы является подготовка данных для публикации расписания учебных занятий в ТГУ.

Целью выпускной квалификационной работы является разработка Web API для предоставления данных по расписанию учебных занятий в ТГУ.

Для достижения поставленной цели необходимо выполнить следующие задачи:

– проанализировать процесс публикации расписания учебных занятий в ТГУ;

– выработать требования к системе подготовки данных для публикации расписания занятий;

– спроектировать систему подготовки данных для публикации расписания занятий;

– реализовать систему подготовки данных для публикации расписания занятий;

– проанализировать статистику работы системы подготовки данных для публикации расписания занятий.

Введение раскрывает актуальность и значимость работы, определяет объект и предмет исследования, цель и задачи.

В первой главе анализируется процесс публикации расписания учебных занятий в ТГУ, вырабатываются требования к системе подготовки данных для публикации расписания занятий.

Во второй главе разрабатываются функции, описывается архитектура системы подготовки данных для публикации расписания занятий.

В третьей главе описывается реализация системы подготовки данных для публикации расписания занятий и проводится анализ статистики работы.

В заключении делаются выводы по проделанной работе, рассматриваются перспективы развития системы подготовки данных для публикации расписания занятий.

1 АНАЛИЗ ПРОЦЕССА ПУБЛИКАЦИИ РАСПИСАНИЯ УЧЕБНЫХ ЗАНЯТИЙ В ТОЛЬЯТТИНСКОМ ГОСУДАРСТВЕННОМ УНИВЕРСИТЕТЕ

1.1 Описание структуры Тольяттинского государственного университета

Тольяттинский государственный университет был создан 29 мая 2001 года путём объединения Тольяттинского политехнического института с Тольяттинским филиалом Самарского государственного педагогического университета.

Основной целью Тольяттинского государственного университета является подготовка высококвалифицированных кадров.

Структура Тольяттинского государственного университета представлена на рисунке 1 [15].

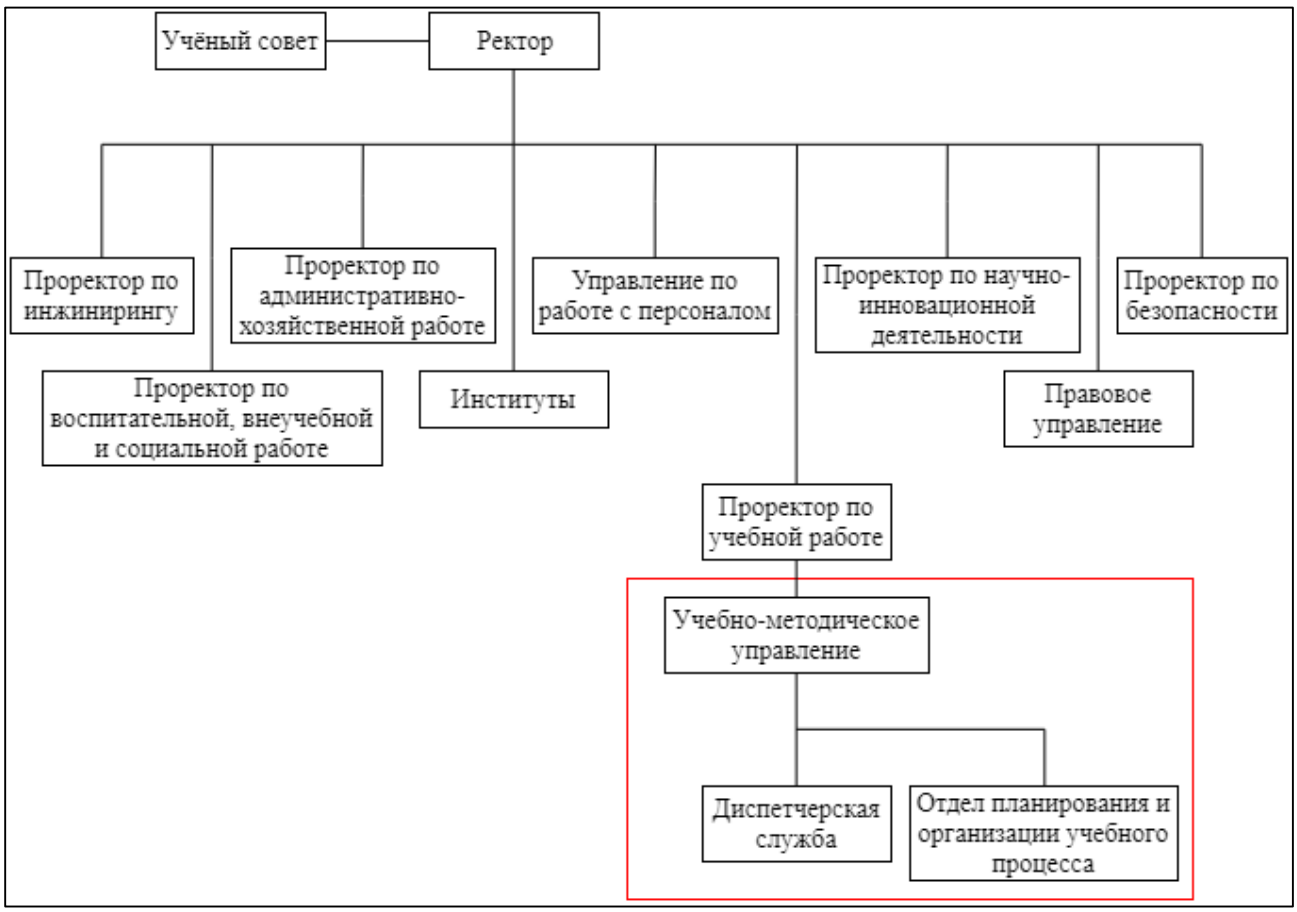


Рисунок 1 – Структура ТГУ

На рисунке 1 прямоугольником выделена та часть структуры ТГУ, которая рассматривается в рамках данной работы. Именно учебно-методическое управление занимается разработкой и публикацией расписания учебных занятий в ТГУ.

1.2 Анализ существующего процесса публикации расписания учебных занятий в Тольяттинском государственном университете

Для составления расписания учебных занятий в ТГУ используется технология построения, встроенная в систему «Галактика: Расписание учебных занятий». В настоящее время процесс публикации расписания учебных занятий в ТГУ выглядит следующим образом (смотреть рисунок 2).

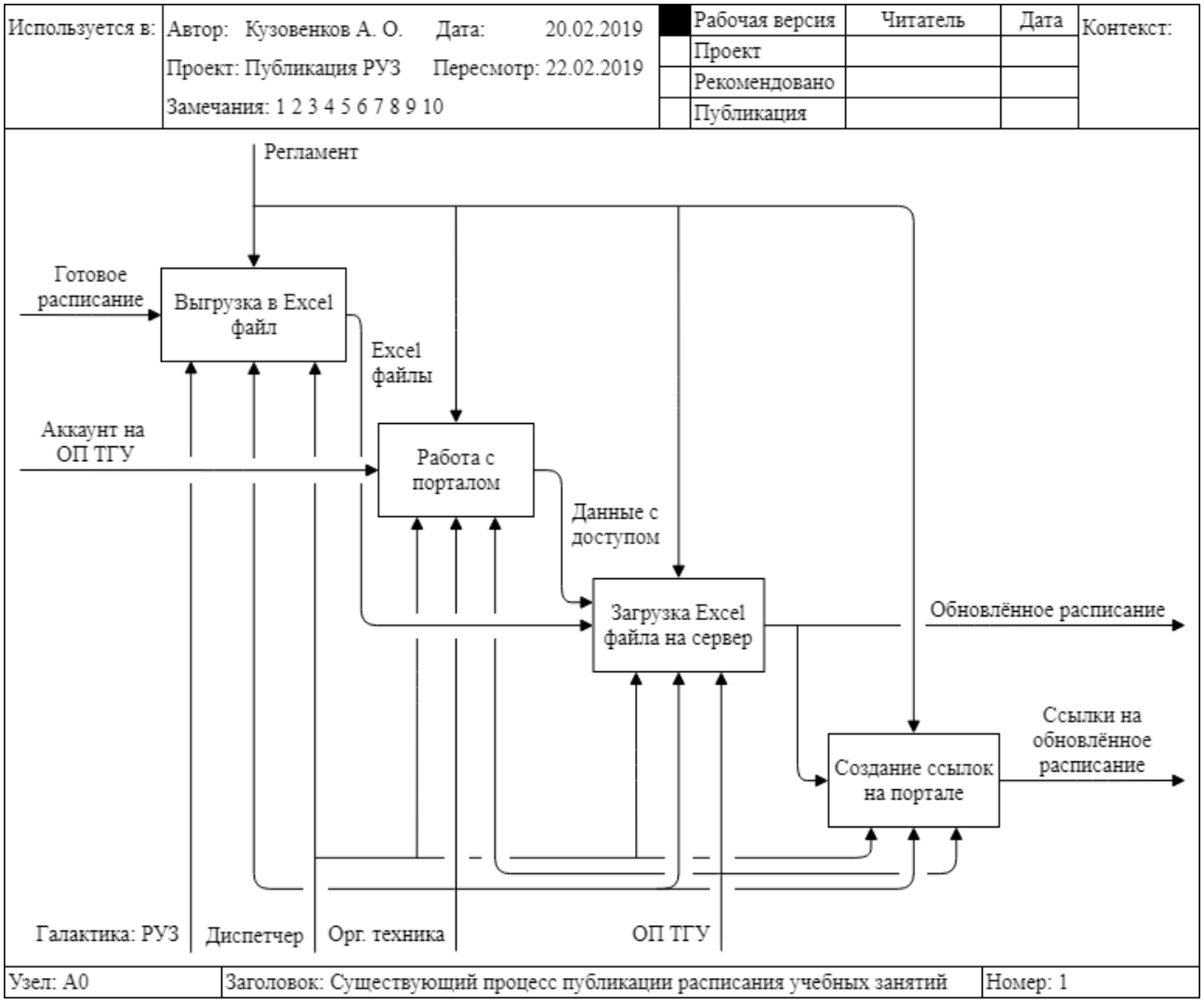


Рисунок 2 – Модель IDEF0 «Как есть»

Публикация расписания осуществляется сотрудником диспетчерской службы. Вначале диспетчер осуществляет экспорт расписания учебных занятий, полученного в результате работы системы «Галактика: Расписание учебных занятий», в Excel файлы. Затем он авторизуется на образовательном портале ТГУ и при помощи HTML редактора FCKeditor загружает полученные Excel файлы на сервер института. В конце диспетчер формирует ссылки на загруженные документы.

Такая система публикации расписания имеет ряд недостатков:

– так как публикация расписания выполняется вручную, возможны человеческие ошибки;

– для того чтобы открыть файлы формата xls (xlsx), требуется установка дополнительного программного обеспечения;

– так как публикация расписания учебных занятий осуществляется на неделю, требуется еженедельно создавать новые файлы формата xls (xlsx).

Перечисленные выше недостатки делают процесс публикации расписания учебных занятий менее эффективным. Чтобы устранить данные недостатки, необходимо автоматизировать этот процесс.

1.3 Выработка требований к системе подготовки данных для публикации расписания занятий

Для того чтобы исправить недостатки, процесс публикации расписания учебных занятий в ТГУ планируется организовать следующим образом (смотреть рисунок 3).

В отличие от существующей системы публикации учебных занятий, вся работа будет выполняться автоматизированной системой (разрабатываемым приложением). Она, используя данные аккаунта на образовательном портале, будет извлекать расписание учебных занятий из базы данных «Галактика: Расписание учебных занятий». В конце система будет преобразовывать полученные данные в пригодный для дальнейшего отображения вид.

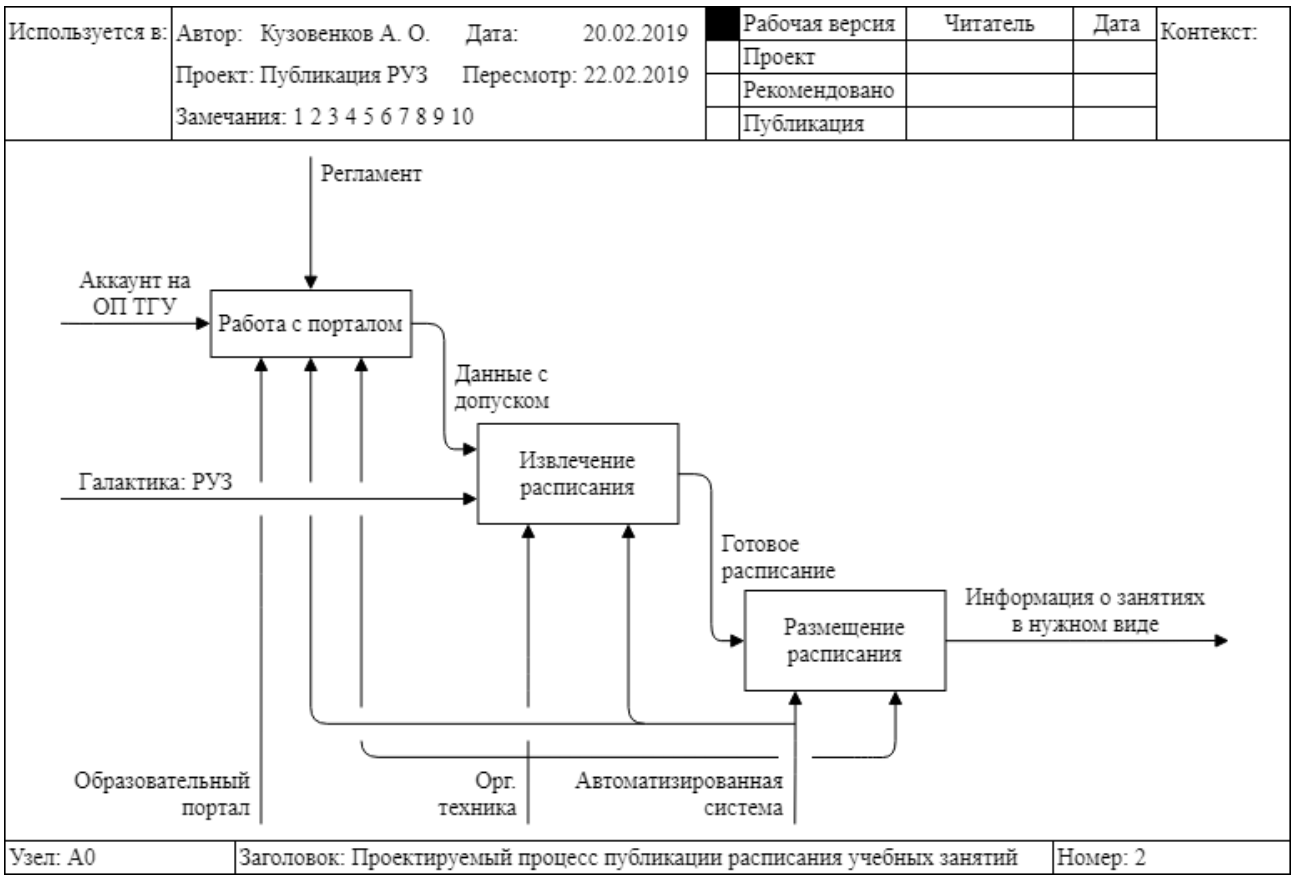


Рисунок 3 – Модель IDEF0 «Как будет»

Таким образом, создание системы подготовки данных для публикации расписания учебных занятий позволит добиться следующих результатов:

– пропадёт необходимость в использовании человеческих ресурсов и как следствие исчезнут ошибки ручного труда;

– процесс публикации станет стабильнее;

– любые изменения в расписании будут обрабатываться значительно быстрее;

– можно будет осуществлять просмотр расписания учебных занятий с любого устройства.

Разрабатываемая система будет представлять из себя серверное приложение. Выработаем требования к ней. Для этого будем использовать систему классификации требований FURPS+. Выработанные требования к системе подготовки данных для публикации расписания учебных занятий представлены в таблице 1.

Таблица 1 – Требования к системе

№	Требование	Полезность
Functionality – Функциональные требования		
1	Предоставлять список институтов (для групп студентов, преподавателей и аудиторий)	Критичное
2	Показывать кафедры каждого института (для аудиторий и преподавателей)	Критичное
3	Возвращать списки преподавателей и аудиторий каждой кафедры	Критичное
4	Показывать список групп студентов каждого института	Критичное
5	Предоставлять расписание для преподавателя, аудитории и группы студентов на выбранный промежуток времени	Критичное
6	Позволять осуществлять поиск преподавателей, аудиторий и групп студентов	Важное
7	Позволять подготавливать расписание на печать	Полезное
Usability – Требования к удобству использования		
8	Должно иметься описание всех функций	Важное
Reliability – Требования к надежности		
9	Должна корректно реагировать на неверные данные и ошибки	Важное
10	Должна быть доступна 24 часа в сутки 7 дней в неделю	Важное
Performance – Требования к производительности		
11	Время выполнения одного запроса не должно превышать 500 миллисекунд	Критичное
Supportability – Требование к поддержке		
12	Должна быть разделена на модули	Важное
13	Должно вестись логирование событий	Критичное
Implementation requirements – Требования к реализации		
14	Должна быть кроссплатформенной	Критичное
Interface requirements – Требование к интерфейсу		
15	Должна выводить расписание в виде таблицы	Важное

Таким образом было выявлено, что необходимо реализовать 15 требований.

1.4 Выводы и результаты по разделу 1

В первом подразделе рассмотрена структура Тольяттинского государственного университета.

Во втором подразделе был проведен анализ существующего процесса публикации расписания учебных занятий в Тольяттинском государственном университете. В результате анализа было выяснено, что публикация расписания учебных занятий выполняется вручную сотрудником диспетчерской службы. На выходе получаются файлы формата xls (xlsx). А для того чтобы открыть такие файлы, требуется установка дополнительного программного обеспечения. Таким

образом, был сделан вывод о том, что данные недостатки делают процесс публикации расписания учебных занятий менее эффективным. Чтобы их исправить, было принято решение о разработке системы подготовки данных для публикации расписания учебных занятий.

В третьем подразделе были выработаны требования к системе подготовки данных для публикации расписания учебных занятий. Было выявлено, что необходимо реализовать 15 требований.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ ПОДГОТОВКИ ДАННЫХ ДЛЯ ПУБЛИКАЦИИ РАСПИСАНИЯ ЗАНЯТИЙ

2.1 Проектирование функций системы подготовки данных для публикации расписания занятий

Для того чтобы понять, как именно будет осуществляться извлечение данных и их дальнейшая публикация, представим каждый из этих процессов в виде конечного автомата:

$$M = (S, A, \delta, q_0, F) \quad (1)$$

где S – множество состояний,

A – алфавит,

δ – функция переходов,

q_0 – начальное состояние,

F – множество конечных состояний.

Рассмотрим сначала процесс извлечения данных. В данном случае под множеством состояний *S* подразумевается набор, представленный в таблице 2.

Таблица 2 – Состояния (извлечение данных)

Состояние	Описание
H	Начальное состояние. В данном состоянии система просто ожидает, когда произойдёт какое-либо событие.
S_1	Обновляет компоненты. В данном состоянии система обновляет информацию о преподавателях, группах, аудиториях, институтов, кафедр и дисциплин в локальной базе данных.
S_2	Обновляет занятия. В данном состоянии система обновляет в локальной базе данных информацию о занятиях.
S_3	Обновление активности. В данном состоянии система проверяет на присутствие преподавателей, групп, аудиторий, институтов и кафедр за определенное время в системе.
F	Конечное состояние. Означает что запрос выполнен успешно.

Алфавитом *A* для процесса извлечения данных является набор событий, представленный в таблице 3.

Таблица 3 – События (извлечение данных)

Событие	Описание
a_1	Время таймера «обновление компонент» истекло. Данное событие означает, что с момента последнего срабатывания таймера «обновление компонент» прошло заданное (или более) количество времени.
a_2	Время таймера «обновление занятий» истекло. Данное событие означает, что с момента последнего срабатывания таймера «обновление занятий» прошло заданное (или более) количество времени.
a_3	Время таймера «обновление активности» истекло. Данное событие означает, что с момента последнего срабатывания таймера «обновление активности» прошло заданное (или более) количество времени.
a_4	Обновление завершено. Данное событие означает, что система закончила обновлять данные.

Функцию переходов δ можно представить в виде таблицы переходов:

$$T = (|S| \times |A|) \quad (2)$$

где S – множество состояний,

A – алфавит.

Далее таблица 4 даёт табличное представление функции δ.

Таблица 4 – Функция переходов (извлечение данных)

	a_1	a_2	a_3	a_4
H	S_1	S_2	S_3	
S_1				F
S_2				F
S_3				F

Более наглядно полученный конечный автомат представлен на рисунке 4.

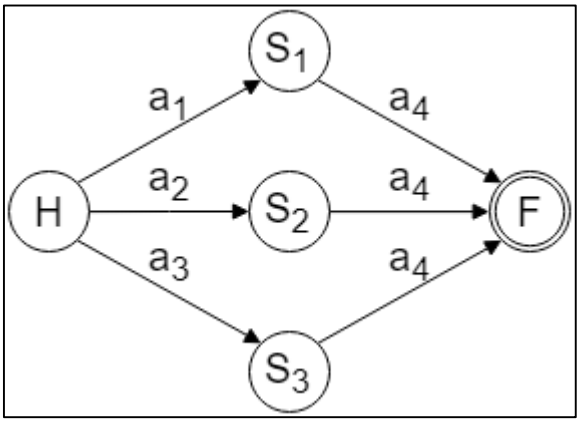


Рисунок 4 – Конечный автомат (процесс извлечения данных)

В случае с публикацией данных, когда пользователь ещё не авторизован, под множеством состояний *S* подразумевается набор, представленный в таблице 5.

Таблица 5 – Состояния (публикация, пользователь не авторизован)

Состояние	Описание
H	Начальное состояние. В данном состоянии система ожидает запросы от неавторизованного пользователя.
S_1	Проверяет логин и пароль. В данном состоянии система сопоставляет введённые пользователем логин (имя/идентификатор) и пароль с теми, что хранятся в базе данных аккаунтов.
S_2	Обновление данных пользователя. В данном состоянии система закрепляет за аккаунтом пользователя текущую сессию (и по желанию пользователя запоминает его компьютер).
S_3	Подготовка обычной главной страницы. В данном состоянии система просто извлекает обычную главную страницу, передаёт её пользователю. Данная страница представляет собой форму, на которой расположены названия всех институтов, кафедр, групп, аудиторий и т. д. в структурированном виде.
S_4	Извлечение расписания. В данном состоянии из локальной базы данных извлекается информация о занятиях.
S_5	Обработка расписания. В данном состоянии информация о занятиях преобразуется в обычное представление.
S_6	Публикация расписания. В данном состоянии обычное представление размещается на обычной странице просмотра расписания и передаётся пользователю. Данная страница представляет собой форму, на которой расположены элементы управления отображением представления.
S_7	Выполнение поиска. В данном состоянии система на основании текста, введённого пользователем, осуществляет поиск групп, преподавателей и аудиторий в базе данных.
S_8	Обработка результатов поиска. В данном состоянии полученные результаты поиска преобразуются в пригодный для человеческого восприятия вид и возвращаются пользователю.
F	Конечное состояние. Означает что запрос выполнен успешно.

Алфавитом *A* для процесса публикации в случае, когда пользователь ещё не авторизован, является набор событий, представленный в таблице 6.

Таблица 6 – События (публикация данных, пользователь не авторизован)

Событие	Описание
a_1	Запрос на авторизацию. Для возникновения данного события необходимо чтобы пользователь указал логин (имя/идентификатор) и пароль.
a_2	Логин и пароль верны. Данное событие означает, что логин и пароль совпали с теми, которые хранятся в базе данных аккаунтов.
a_3	Запрос главной страницы. Данное событие подразумевает, что пользователю нужна главная страница, с помощью которой он сможет выбрать нужное расписание.
a_4	Запрос показа расписания. Для возникновения данного события необходимо, чтобы пользователь указал идентификатор преподавателя, аудитории или группы студентов, а также временной промежуток.
a_5	Расписание извлечено. Данное событие возникает, когда запрашиваемое расписание извлечено из локальной базы данных.
a_6	Расписание обработано. Данное событие возникает, когда извлечённое расписание преобразовано в обычное представление.
a_7	Запрос на поиск. Для возникновения данного события необходимо чтобы пользователь указал текст поиска.
a_8	Поиск окончен. Данное событие возникает, когда система завершила поиск в базе данных.
a_9	Пользователь не прошёл авторизацию. Данное событие возникает в случае несовпадения логина и/или пароля с теми, что хранятся в базе данных аккаунтов. Пользователю было отправлено соответствующее сообщение.
a_{10}	Запрос выполнен. Данное событие означает, что запрос выполнен и был отправлен ответ пользователю.

Далее таблица 7 даёт табличное представление функции δ.

Таблица 7 – Функция переходов (публикация, пользователь не авторизован)

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
H	S_1		S_3	S_4			S_7			
S_1		S_2							F	
S_2										F
S_3										F
S_4					S_5					
S_5						S_6				
S_6										F
S_7								S_8		
S_8										F

Более наглядно полученный конечный автомат представлен на рисунке 5.

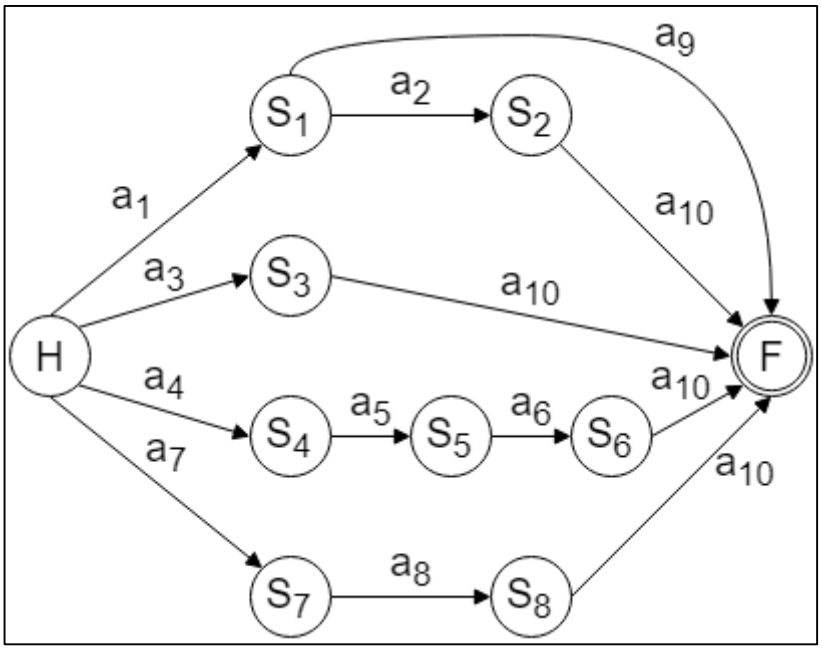


Рисунок 5 – Конечный автомат (публикация, пользователь не авторизован)

При публикации данных, когда пользователь уже авторизован, под множеством состояний *S* подразумевается набор, представленный в таблице 8.

Таблица 8 – Состояния (публикация, пользователь авторизован)

Состояние	Описание
H	Начальное состояние. В данном состоянии система ожидает запросы от авторизованного пользователя.
S_1	Извлечение расширенной главной страницы. В данном состоянии система просто извлекает расширенную главную страницу и передаёт её пользователю. Данная страница обладает большим функционалом в сравнении с обычной (предоставляет возможность печати расписания кафедры).
S_2	Извлечение расписания. В данном состоянии из локальной базы данных извлекается информация о занятиях.
S_3	Извлечение данных пользователя. В данном состоянии из базы данных извлекаются настройки пользователя, его личные данные и т. д.
S_4	Обработка расписания. В данном состоянии информация о занятиях преобразуется одно из нескольких представлений, которое выбрал пользователь (в том числе это может быть и обычное представление).
S_5	Публикация расписания. В данном состоянии представление размещается на расширенной странице просмотра расписания и передаётся пользователю. Данная страница обладает большим функционалом в сравнении с обычной (позволяет сменить само представление, может работать с данными пользователя).
S_6	Извлечение расписания кафедры. В данном состоянии из локальной базы данных извлекается информация о занятиях для каждого преподавателя указанной кафедры.

Продолжение таблицы 8

Состояние	Описание
S_7	Подготовка к печати. В данном состоянии расписание кафедры преобразуется в одно большое представление пригодное для печати и передаётся пользователю.
S_8	Выход из системы. В данном состоянии сбрасывается сессия пользователя и открепляется от аккаунта текущий компьютер (если он был запомнен).
S_9	Выполнение поиска. В данном состоянии система на основании текста, введённого пользователем, осуществляет поиск групп, преподавателей и аудиторий в базе данных.
S_{10}	Обработка результатов поиска. В данном состоянии полученные результаты поиска преобразуются в пригодный для человеческого восприятия вид и возвращаются пользователю.
F	Конечное состояние. Означает что запрос выполнен успешно.

Алфавитом *A* для процесса публикации в случае, когда пользователь уже авторизован, является набор событий, представленный в таблице 9.

Таблица 9 – События (публикация данных, пользователь авторизован)

Событие	Описание
a_1	Запрос главной страницы. Данное событие подразумевает, что пользователю нужна главная страница, с помощью которой он сможет выбрать нужное расписание.
a_2	Запрос показа расписания. Для возникновения данного события необходимо, чтобы пользователь указал идентификатор преподавателя, аудитории или группы студентов, а также временной промежуток.
a_3	Расписание извлечено. Данное событие возникает, когда запрашиваемое расписание извлечено из локальной базы данных.
a_4	Данные пользователя извлечены. Данное событие возникает, когда данные пользователя извлечены из базы данных и готовы к использованию.
a_5	Расписание обработано. Данное событие возникает, когда извлечённое расписание преобразовано в обычное представление.
a_6	Запрос печати расписания. Для возникновения данного события необходимо чтобы пользователь указал идентификатор кафедры и номер недели.
a_7	Расписание кафедры извлечено. Данное событие возникает, когда расписание кафедры извлечено из базы данных и готово к дальнейшим преобразованиям.
a_8	Запрос на выход из системы, Данное событие возникает, когда пользователь желает выйти из своего аккаунта.
a_9	Запрос на поиск. Для возникновения данного события необходимо чтобы пользователь указал текст поиска.
a_{10}	Поиск окончен. Данное событие возникает, когда система завершила поиск в базе данных.
a_{11}	Запрос выполнен. Данное событие означает, что запрос выполнен и был отправлен ответ пользователю.

Далее таблица 10 даёт табличное представление функции δ.

Таблица 10 – Функция переходов (публикация, пользователь не авторизован)

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
H	S_1	S_2				S_6		S_8	S_9		
S_1											F
S_2			S_3								
S_3				S_4							
S_4					S_5						
S_5											F
S_6							S_7				
S_7											F
S_8											F
S_9										S_{10}	
S_{10}											F

Более наглядно полученный конечный автомат представлен на рисунке 6.

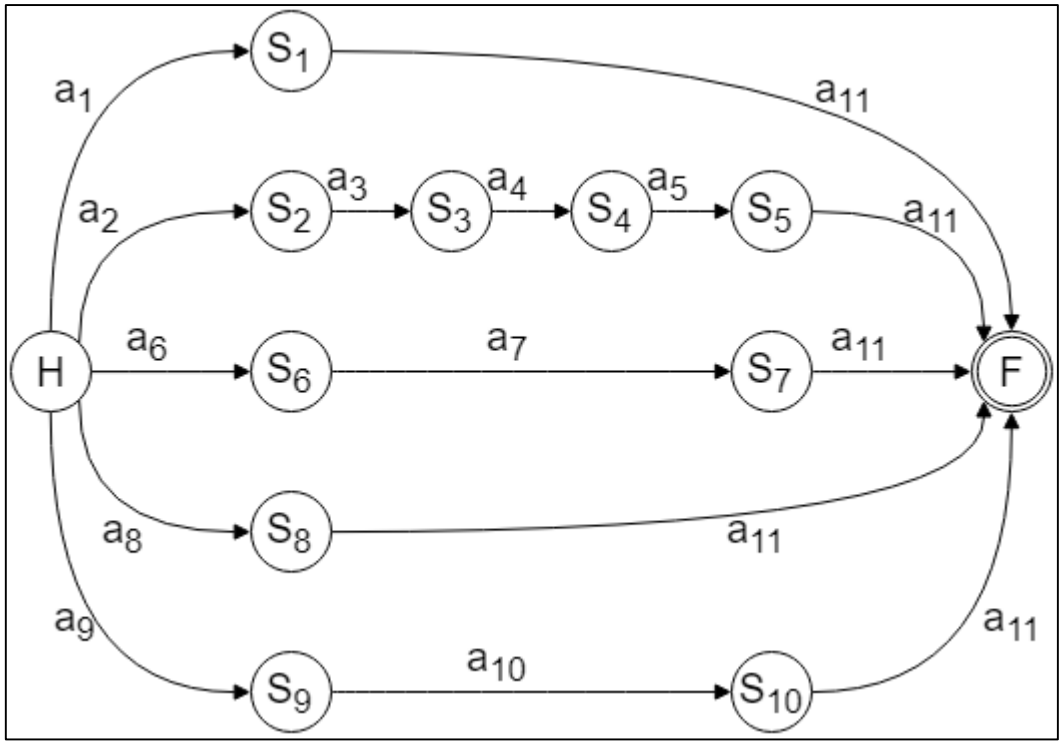


Рисунок 6 – Конечный автомат (публикация, пользователь авторизован)

На основе составленных конечных автоматов были выделены и сгруппированы функции системы (смотреть рисунок 7).

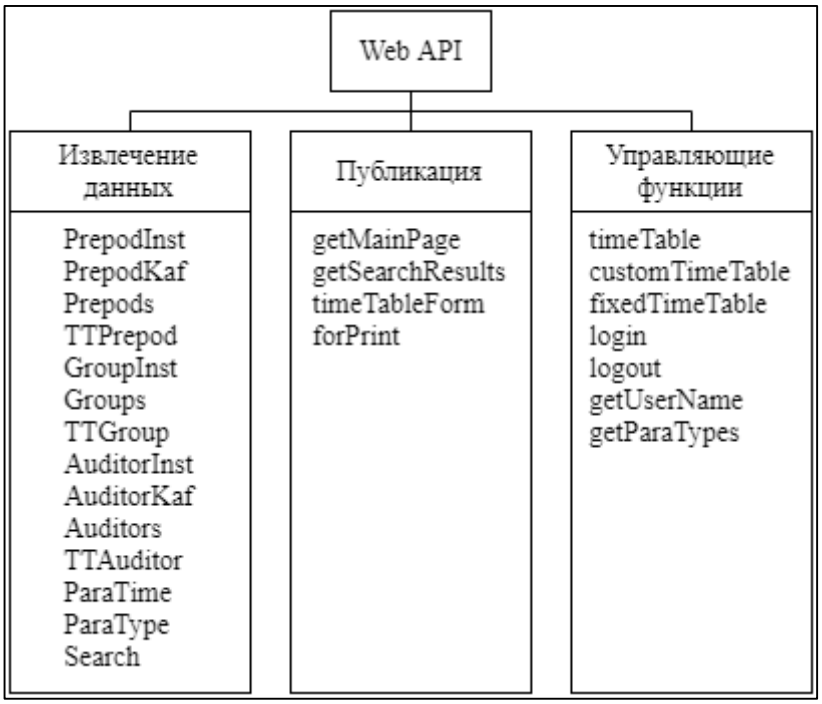


Рисунок 7 – Функции системы

Дерево взаимосвязей функций представлено на рисунке 8.

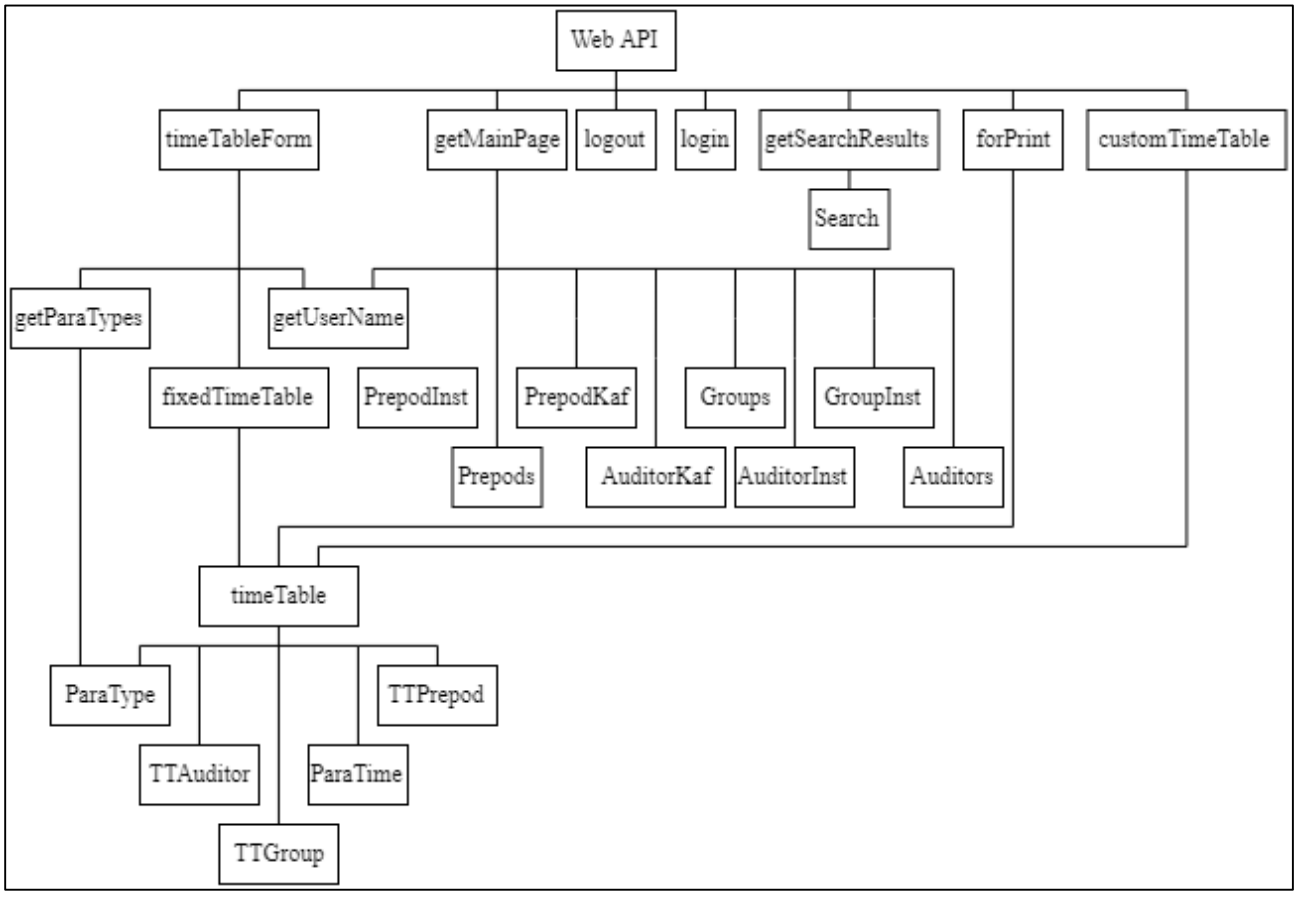


Рисунок 8 – Взаимосвязь функций системы

Подробное описание данных функций приведено в таблице 11.

Таблица 11 – Описание функций системы

№	Название	Параметры	Описание
1	PrepodInst	1. token.	Возвращает информацию о всех институтах преподавателей.
2	PrepodKaf	1. token. 2. prepodInstId – это переменная, которая представляет собой уникальный целочисленный идентификатор института преподавателей. В зависимости от этого идентификатора извлекается информация о кафедрах соответствующего института.	Возвращает информацию о всех кафедрах преподавателей заданного института.
3	Prepods	1. token. 2. prepodKafId – это переменная, которая представляет собой уникальный целочисленный идентификатор кафедры преподавателей. В зависимости от этого идентификатора извлекается информация о преподавателях соответствующей кафедры.	Возвращает информацию о всех преподавателях заданной кафедры.
4	TTPrepod	1. token. 2. prepodId – целочисленная переменная, представляющая собой уникальный идентификатор преподавателя. 3. from – текстовая переменная, которая содержит дату начала временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-02-20»). 4. to – текстовая переменная, хранящая дату конца временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-03-12»).	Возвращает информацию о занятиях, которые ведёт заданный преподаватель на выбранном промежутке времени.
5	GroupInst	1. token.	Возвращает информацию о всех институтах групп.
6	Groups	1. token. 2. groupInstId – это переменная, которая представляет собой уникальный целочисленный идентификатор института групп студентов. В зависимости от этого идентификатора извлекается информация о группах студентов соответствующего института.	Возвращает информацию о всех группах студентов заданного института.

Продолжение таблицы 11

№	Название	Параметры	Описание
7	TTGroup	1. token. 2. groupId – целочисленная переменная, представляющая собой уникальный идентификатор группы студентов. 3. from – текстовая переменная, которая содержит дату начала временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-02-20»). 4. to – текстовая переменная, хранящая дату конца временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-03-12»).	Возвращает информацию о занятиях, которые проходят у заданной группы студентов на выбранном промежутке времени.
8	AuditorInst	1. token.	Возвращает информацию о всех институтах аудиторий.
9	AuditorKaf	1. token. 2. auditorInstId – это переменная, которая представляет собой уникальный целочисленный идентификатор института аудиторий. В зависимости от этого идентификатора извлекается информация о кафедрах соответствующего института.	Возвращает информацию о всех кафедрах аудиторий заданного института.
10	Auditors	1. token. 2. auditorKafId – это переменная, которая представляет собой уникальный целочисленный идентификатор кафедры аудиторий. В зависимости от этого идентификатора извлекается информация об аудиториях соответствующей кафедры.	Возвращает информацию о всех аудиториях заданной кафедры.
11	TTAuditor	1. token. 2. auditorId – целочисленная переменная, представляющая собой уникальный идентификатор аудитории. 3. from – текстовая переменная, которая содержит дату начала временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-02-20»). 4. to – текстовая переменная, хранящая дату конца временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-03-12»).	Возвращает информацию о занятиях, которые проходят в заданной аудитории на выбранном промежутке времени.

Продолжение таблицы 11

№	Название	Параметры	Описание
12	ParaTime	1. token.	Возвращает информацию о всех возможных промежутках времени проведения какого-либо занятия.
13	ParaType	1. token.	Возвращает информацию о всех возможных типах занятий.
14	Search	1. token. 2. text – это строковая переменная, которая содержит текст поиска.	Позволяет осуществить поиск.
15	getParaTypes	1. token.	Возвращает типы занятий (лекция, практика, экзамен и т. д.).
16	getMainPage	1. token. 2. mode – строковая переменная, определяющая режим главной страницы. Может принимать одно из следующих значений: advanced (расширенная), normal (обычная).	Возвращает главную страницу (обычную или расширенную).
17	getSearchResults	1. token. 2. text – это строковая переменная, которая содержит текст поиска.	Возвращает результаты поиска в виде графических элементов.
18	timeTable	1. token. 2. mode – строковая переменная, которая определяет вид таблицы. Может принимать одно из следующих значений: normal (обычное представление), advanced (расширенное представление), courseInfo (информация о курсах), subject (по дисциплинам). 3. category – символьная переменная, которая определяет категорию расписания. Может принимать одно из следующих значений: c (для аудитории), l (для преподавателя), g (для группы студентов). 4. id – целочисленная переменная, представляющая собой уникальный идентификатор. Зависит от выбранной категории. 5. dayFrom – текстовая переменная, начало временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-02-20»). 6. dayTo – текстовая переменная, конец временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-03-12»).	Возвращает представление расписания в виде таблицы.

Продолжение таблицы 11

№	Название	Параметры	Описание
19	forPrint	1. token. 2. chairId – целочисленная переменная, представляющая собой уникальный идентификатор кафедры преподавателей. 3. week – строковая переменная, которая содержит номер недели в формате «ГГГГ-WНН», где ГГГГ – год, а НН – сам номер недели (например, «2019-W24»).	Возвращает представление расписания кафедры, готовое к печати.
20	customTimeTable	1. token. 2. category – символьная переменная, которая определяет категорию расписания. Может принимать одно из следующих значений: c (для аудитории), l (для преподавателя), g (для группы студентов). 3. id – целочисленная переменная, представляющая собой уникальный идентификатор. Зависит от выбранной категории. 4. dayFrom – текстовая переменная, которая содержит дату начала временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-02-20»). 5. dayTo – текстовая переменная, хранящая дату конца временного промежутка в формате «ГГГГ-ММ-ДД» (например, «2018-03-12»).	Возвращает обычное представление расписания (произвольный промежуток времени, не более чем на год).
21	timeTableForm	1. token. 2. mode – строковая переменная, которая определяет вид таблицы. Может принимать одно из следующих значений: normal (обычное представление), advanced (расширенное представление), courseInfo (информация о курсах), subject (по дисциплинам). 3. category – символьная переменная, которая определяет категорию расписания. Может принимать одно из следующих значений: c (для аудитории), l (для преподавателя), g (для группы студентов). 4. id – целочисленная переменная, представляющая собой уникальный идентификатор. Зависит от выбранной категории.	Возвращает форму для работы с представлениями.

Продолжение таблицы 11

№	Название	Параметры	Описание
22	fixedTimeTable	1. token. 2. mode – строковая переменная, которая определяет вид таблицы. Может принимать одно из следующих значений: day (обычное представление на один день), week (обычное представление на неделю), advanced (расширенное представление на семестр), courseInfo (информация о курсах на семестр), subject (по дисциплинам на семестр). 3. category – символьная переменная, которая определяет категорию расписания. Может принимать одно из следующих значений: c (для аудитории), l (для преподавателя), g (для группы студентов). 4. id – целочисленная переменная, представляющая собой уникальный идентификатор. Зависит от выбранной категории. 5. value – строковая переменная, которая представляет собой смещение от текущей даты.	Возвращает представление расписания (временные промежутки фиксированы).
23	login	1. name – строковая переменная, которая представляет собой имя пользователя. 2. password – строковая переменная, которая представляет собой пароль от учётной записи пользователя. 3. remember – символьная переменная, которая отвечает за запоминание компьютера пользователя. Может принимать одно из следующих значений: t (запомнить компьютер), f (не запоминать компьютер).	Осуществляет авторизацию пользователя.
24	logout	-	Осуществляет выход из системы для текущего пользователя.
25	getUserName	-	Возвращает имя текущего пользователя.

Как можно заметить, для вызова почти любой функции необходимо указывать обязательный аргумент token. token – строковая переменная, которая представляет собой секретный ключ, необходимый для аутентификации того, кто эти функции хочет использовать. Без правильно указанного ключа какую-либо функцию вызвать не получится.

Таким образом, используя разработанные ранее конечные автоматы, были получены данные функции.

2.2 Разработка архитектуры системы подготовки данных для публикации расписания занятий

Архитектура системы подготовки данных для публикации расписания занятий представлена на рисунке 9.

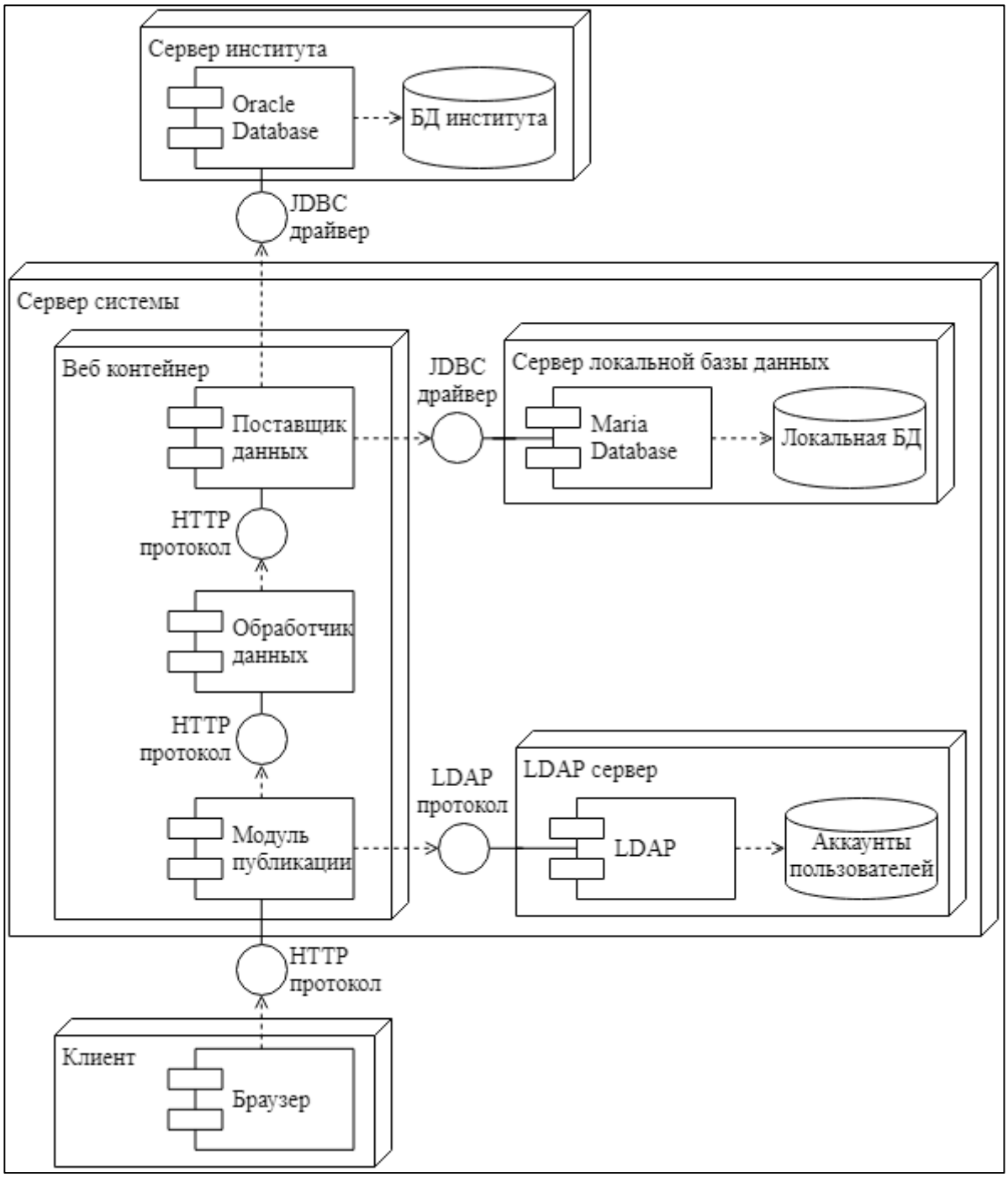


Рисунок 9 – Диаграмма компонентов

Из диаграммы компонентов видно, что система состоит из трёх модулей. Помимо этого, в работе системы также участвуют:

– база данных института (источник всех входных данных);

– промежуточная база данных (для удобной работы с информацией из базы данных института);

– база данных аккаунтов пользователей.

Первый модуль занимается тем, что обеспечивает удобный интерфейс доступа к промежуточной базе данных, в которую он периодически помещает информацию (о списках групп, преподавателей, аудиторий, институтов, о занятиях и т. д.), взятую из базы данных института и преобразованную в пригодный для дальнейшего извлечения и анализа вид. В результате посредством этого интерфейса можно легко осуществить поиск групп, преподавателей или аудиторий, получить расписание и т. д. Другими словами, первый модуль реализует основной функционал системы.

Второй модуль на основании данных, полученных от первого модуля, генерирует представления, подготавливает результаты поиска, и т. д. Также помимо общения с первым модулем данный модуль выполняет кэширование представлений, к которым чаще всего обращаются. Далее в таблице 12 представлено описание компонент второго модуля.

Таблица 12 – Описание компонент второго модуля

Название компоненты	Роль компоненты в системе
Cache	Структура, которая хранит в себе только время последнего обращения к ней.
TimeTable	Структура, которая является расширением Cache и хранит в себе таблицу с расписанием и её метаданные: название представления, категория (группа, преподаватель, аудитория), идентификатор группы, преподавателя или аудитории, временной промежуток, последнее обращение к таблице.
CourseInfo	Структура, которая содержит в себе информацию, представляющую собой краткую сводку о практиках или лекциях конкретной дисциплины: имена преподавателей или названия групп, количество прошедших занятий, дата и время следующего занятия.
Entity	Структура, состоящая из двух полей: имя и идентификатор. Необходима для работы с группами, преподавателями и аудиториями.

Продолжение таблицы 12

Название компоненты	Роль компоненты в системе
Para	Структура, которая представляет собой информацию о занятии: полное название предмета, краткое название предмета, тип занятия (лекция, практика и т. д.), присутствующие группы, отметка о том менялась ли информация о данном занятии, фамилия и инициалы преподавателей, которые ведут данное занятие, название аудиторий, в которых проходит занятие, время его проведения (начало и конец).
ParaTime	Структура, представляющая собой описание временного промежутка одного занятия: идентификатор, короткое название промежутка (только номер занятия), полное название промежутка (номер занятия и окончание), время начала занятия, время окончания занятия.
SearchResult	Структура, состоящая из списков Entity. Несёт в себе информацию о найденных по поисковому запросу группах, преподавателях и аудиториях.
Line	Структура, представляющая собой строку таблицы (список ячеек)
Table	Структура, представляющая собой таблицу (список строк Line)
SmallTag	Структура, представляющая собой небольшое описание графического элемента представления.
Tag	Структура, представляющая собой расширенное описание графического элемента представления.
Address	Структура, представляющая собой сетевой запрос. Упрощает работу с запросом путём его разделения на составляющее компоненты: сетевой адрес, параметры запроса (пары key-value).
DateUtils	Набор функций для работы с датами.
NetUtils	Набор функций для работы с сетью.
Stopwatch	Структура, представляющая собой секундомер. Позволяет замерять продолжительность выполнения тех или иных действий.
TextUtils	Набор функций для работы со строками.
AnalysisManager	Набор функций для создания краткой сводки о практиках или лекциях конкретной дисциплины (коллекции CourseInfo).
DataManager	Набор функций для получения информации о занятиях, группах, преподавателях, аудиториях, результатах поиска, о существующих временных промежутках занятий и т. д. Данные функции преобразуют информацию, поступающую в виде строк, в коллекции соответствующих структур (Entity, Para, ParaTime и т. д.).
GraphicManager	Набор функций для создания графических элементов (представления, кнопки и т. д.). Выполняется преобразование коллекций структур (Entity, Para и т. д.) в HTML код.
StringManager	Набор функций для преобразования различных данных в строки нужного формата.
Logger	Набор функций для ведения логирования (отчёта о работе модуля).
Main	Основной класс, который занимается обработкой запросов, отправкой необходимого ответа, кэшированием часто просматриваемого расписания, обновлением локальных данных (описаний временных промежутков занятий, списков институтов, типов занятий и т. д.).

Третий модуль уже непосредственно работает с пользователями: размещает представления, сгенерированные вторым модулем, осуществляет авторизацию пользователей, подготовку представления к печати, размещает результаты поиска группы/преподавателя/аудитории, выполняет манипуляции с данными пользователей. Далее в таблице 13 представлено описание компонент третьего модуля.

Таблица 13 – Описание компонент третьего модуля

Название компоненты	Роль компоненты в системе
User	Структура, которая содержит основные сведения о пользователе: логин (имя/идентификатор), идентификаторы компьютеров пользователя, идентификатор текущей сессии и ссылку на дополнительную информацию (UserData) об этом пользователе.
UserData	Структура, которая содержит дополнительную информацию о пользователе: его настройки, личные данные и ссылку на логин (имя/идентификатор).
Address	Структура, представляющая собой сетевой запрос. Упрощает работу с запросом путём его разделения на составляющее компоненты: сетевой адрес, параметры запроса (пары key-value).
DateUtils	Набор функций для работы с датами.
LDAPUtils	Набор функций для работы с базой аккаунтов и паролей пользователей.
NetUtils	Набор функций для работы с сетью.
Stopwatch	Структура, представляющая собой секундомер. Позволяет замерять продолжительность выполнения тех или иных действий.
Logger	Набор функций для ведения логирования (отчёта о работе модуля).
Main	Основной класс, которой занимается обработкой запросов пользователей, разграничением доступа, авторизацией пользователей, манипулированием данными пользователей, выполнением запросов к первому модулю.
TableManager	Набор функций для взаимодействия с первым модулем.
UserService	Набор функций для работы с аккаунтами пользователей и их данными.

Как можно заметить, модули системы в большей степени представляют собой наборы функций, преобразующие данные из одного вида в другой. Таким образом, как таковых отношений меду классами нет. Следовательно, отсутствует смысл в представлении диаграммы классов.

Для того чтобы понять, как именно работают три вышеописанных модуля, рассмотрим диаграммы последовательности. Далее на рисунке 10 представлена диаграмма последовательности, описывающая процесс поиска групп, преподавателей и аудиторий.

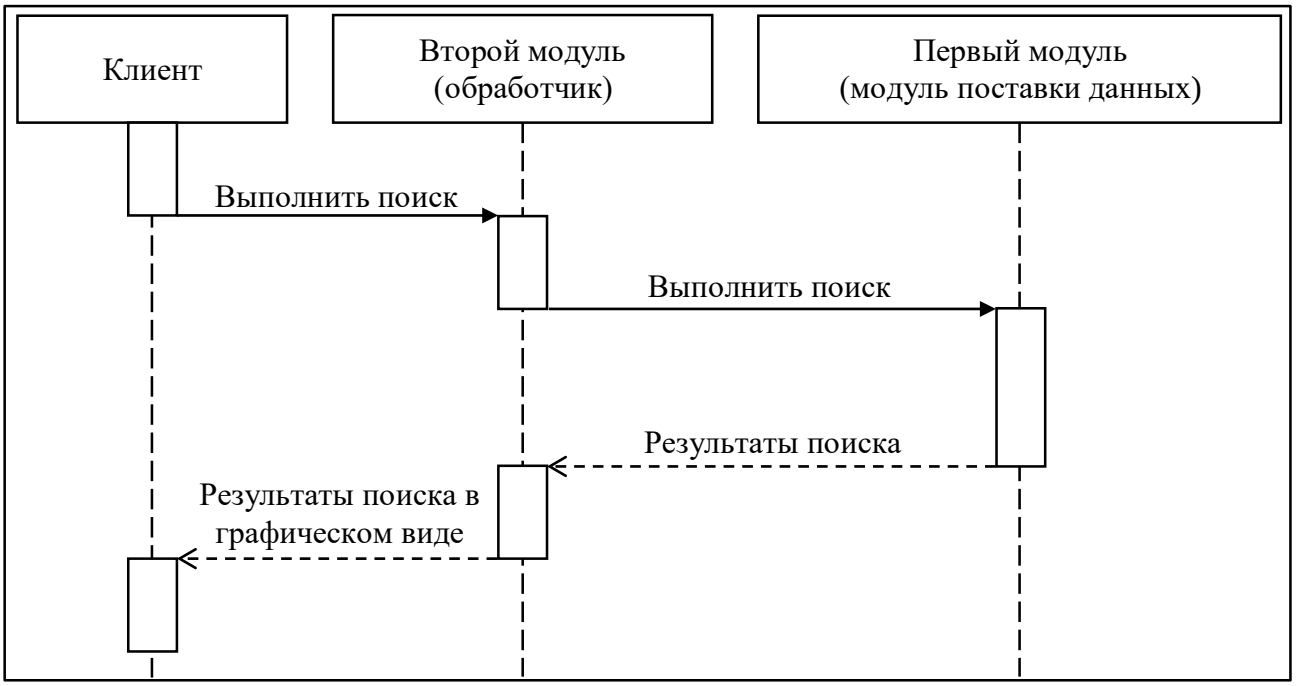


Рисунок 10 – Диаграмма последовательности процесса поиска

Как видно из диаграммы последовательности процесса поиска данный запрос довольно простой. Вначале пользователь пишет текст поиска и отправляет запрос. Затем второй модуль разбирает текст поиска и отправляет запрос на поиск первому модулю. Первый модуль посредством взаимодействия с локальной (промежуточной) базой данных получает результаты поиска и преобразует их в строку нужного формата. Второй модуль с помощью функций DataManager преобразует полученную строку в сущность SearchResult. Далее эта сущность с использованием функций GraphicManager преобразуется в графическое представление, которое в результате возвращается пользователю. Сами результаты поиска в общем виде представляют собой три списка: групп, преподавателей и аудиторий. Элементами этих списков являются пары идентификатор-имя.

Теперь рассмотрим основной процесс, создание представлений расписания (смотреть рисунок 11).

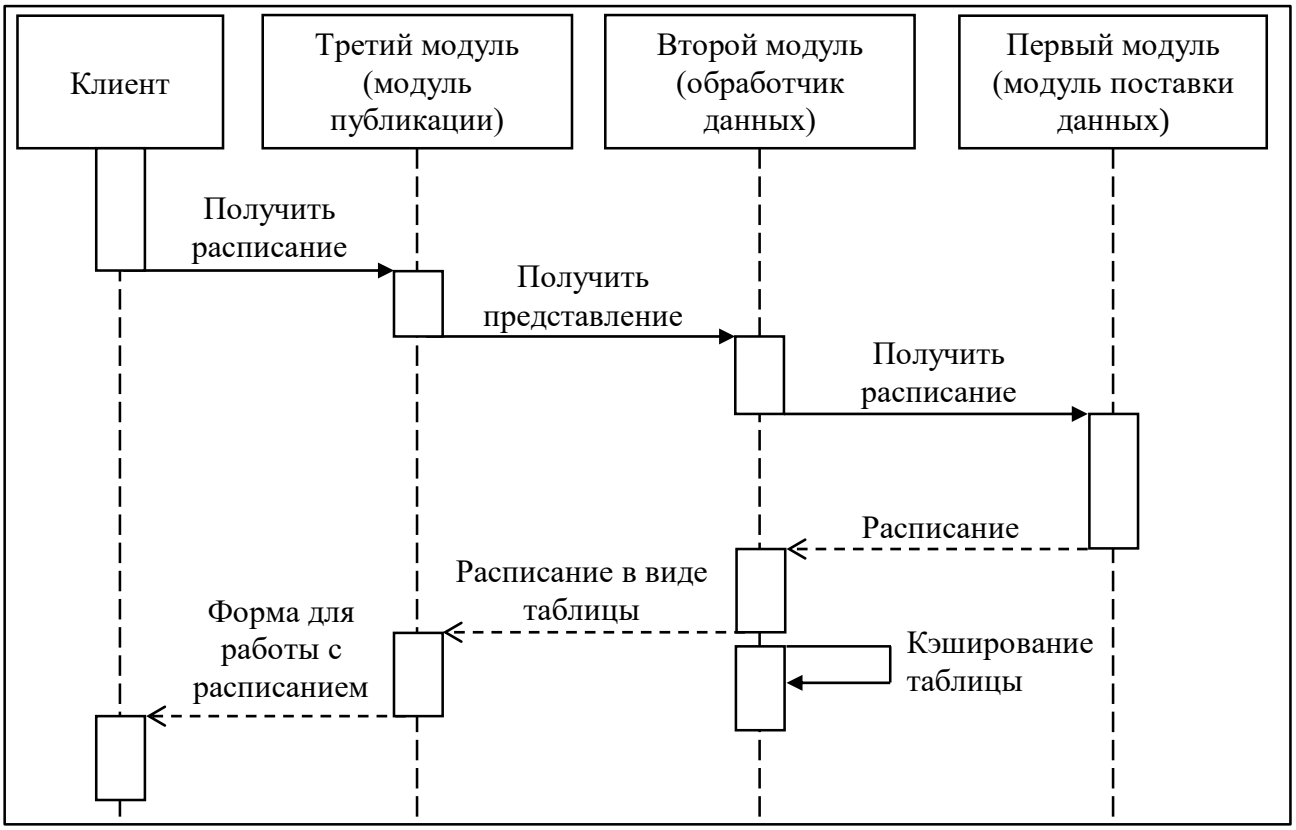


Рисунок 11 – Диаграмма последовательности процесса создания представления

Из диаграммы последовательности процесса создания представления видно, что вначале клиент осуществляет запрос к третьему модулю. В его запросе обязательно содержится:

– идентификатор;

– кому этот идентификатор принадлежит (группе, аудитории или преподавателю);

– вид представления (обычное, расширенное и т. д.).

Далее третий модуль с помощью функций UserService проверяет: осуществил ли до этого пользователь авторизацию. Чтобы это проверить, третий модуль делает запрос к базе данных, в которой хранятся данные пользователей, и сопоставляет идентификатор текущей сессии пользователя и идентификатор его компьютера с теми, что хранятся в этой базе данных. Если произошло совпадение хотя бы одного из вышеперечисленного, то третий модуль извлекает данные того пользователя (UserData), которому принадлежит идентификатор текущей сессии и/или идентификатор компьютера. В случае отсутствия

совпадения пользователь считается неавторизованным. Затем на основании результатов проверки пользователя третий модуль с помощью функций TableManager отправляет запрос на получение таблицы с расписанием. Этот запрос, как и запрос пользователя, содержит идентификатор группы, преподавателя или аудитории, вид представления, а также включает в себя временной промежуток (с какой и по какой день).

Второй модуль, получив этот запрос, вначале пытается извлечь уже готовую таблицу из кэша. Если ему это не удаётся, то он отправляет практически идентичный по структуре запрос на получение расписания первому модулю. Первый модуль посредством взаимодействия с локальной (промежуточной) базой данных получает информацию о расписании за период, указанный в запросе, и преобразует её в строку нужного формата. Второй модуль, получив эту строку, с помощью функций DataManager преобразует его в коллекцию структур Para. Далее эта коллекция с помощь функций GraphicManager преобразуется в готовую таблицу и отправляется первому модулю. После этого второй модуль отправленную таблицу помещает в кэш.

Затем первый модуль, получив таблицу с расписанием, подготавливает необходимую форму (расширенную или обычную в зависимости от того, авторизовался ли пользователь или нет). Далее в соответствии с видом представления (таблицы) первый модуль размещает на форме элементы управления и саму таблицу, а также данные пользователя. В конце форма передаётся пользователю.

Процесс получения представления расписания для печати почти аналогичен. Отличается он тем, что запрос пользователя состоит только из идентификатора кафедры и номера недели.

2.3 Выводы и результаты по разделу 2

В первом подразделе были составлены конечные автоматы, а затем на их основе были выделены 25 функций.

Во втором подразделе была разработана архитектура системы подготовки данных для публикации расписания учебных занятий, путём создания диаграммы компонентов. Основу системы составляют три модуля: модуль поставки данных, модуль обработки данных и модуль публикации данных. Было показано, как эти модули взаимодействуют между собой, какие компоненты они имеют и через что проходят данные.

3 КОДИРОВАНИЕ СИСТЕМЫ ПОДГОТОВКИ ДАННЫХ ДЛЯ ПУБЛИКАЦИИ РАСПИСАНИЯ ЗАНЯТИЙ

3.1 Выбор инструментов для реализации системы подготовки данных по расписанию учебных занятий

Разрабатываемая система должна представлять собой серверное приложение. Следовательно, необходимо выбрать средство, с помощью которого оно будет реализовано [1]. На сегодняшний день такое приложение может быть разработано с использованием одного из следующих популярных средств:

– PHP. Скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений;

– Java. Сильно типизированный объектно-ориентированный язык программирования;

– Python. Высокоуровневый язык программирования общего назначения;

– C++. Статически типизированный язык программирования общего назначения.

Для того чтобы определиться со средством реализации системы, сначала необходимо выделить основные критерии:

– кроссплатформенность. Способность работать с двумя и более аппаратными платформами и (или) операционными системами;

– производительность. Способность как можно меньше зависеть от ресурсов оборудования: процессорного времени, пространства, занимаемого во внутренней и внешней памяти, пропускной способности каналов связи;

– надёжность. Способность безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью;

– наличие инструментов необходимых для разработки веб-приложений;

– поддерживаемость. Возможность поддержки, масштабируемость, ремонтопригодность, гибкость, модифицируемость, модульность, расширяемость, возможность локализации.

Теперь, используя пятибалльную шкалу, оценим средства по каждому критерию. Результаты оценивания представлены в таблице 14.

Таблица 14 – Оценивание средств реализации веб-приложения

Критерии	PHP	Java	Python	C++
Кроссплатформенность	5	5	5	4
Производительность	3	4	3	5
Надёжность	4	5	4	3
Наличие инструментов для веб-разработки	5	5	4	3
Поддерживаемость	4	5	4	3
Суммарный балл	21	24	20	18

Таким образом, на основании суммарных баллов, для разработки веб-приложения были выбраны средства платформы Java [16].

Также необходимо определить, как будет организован вызов каждой функции системы. Источником входных данных (информации об институтах, кафедрах, группах студентов, занятиях и т. д.) является база данных Oracle, расположенная на сервере института. Каждый раз обращаться напрямую к базе данных посредством SQL запросов довольно неудобно, так как SQL запросы довольно большие и их сложно конструировать. Поэтому, чтобы избежать эти проблемы, было принято решение реализовать функции системы в виде запросов протокола HTTP методом GET. В данном случае для реализации HTTP-запросов будет использоваться один из инструментов Java EE, который носит название Servlet API [14].

Теперь необходимо определиться с форматом выходных данных. На сегодняшний день основными форматами данных, которые используются для передачи информации по сети, являются:

– XML. Расширяемый язык разметки;

– JSON. Текстовый формат обмена данными, основанный на JavaScript;

– YAML. Формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования;

– INI.

Для осуществления выбора выделим основные критерии: читаемость кода, удобство редактирования, произвольная иерархия, простота реализации, скорость парсинга/сериализации, размер в сериализованном виде, распространённость, поддержка редакторами, поддержка языками программирования.

Используя снова пятибалльную шкалу, оценим форматы данных по каждому критерию. Результаты оценивания представлены в таблице 15.

Таблица 15 – Оценивание форматов данных

Критерии	XML	JSON	YAML	INI
Читаемость кода	1	4	4	5
Удобство редактирования	1	4	4	5
Произвольная иерархия	4	4	4	1
Простота реализации	2	4	1	5
Скорость парсинга/сериализации	2	4	2	5
Размер в сериализованном виде	1	4	4	5
Распространённость	5	5	2	2
Поддержка редакторами	5	5	3	5
Поддержка языками программирования	5	5	3	5
Суммарный балл	26	39	27	38

В результате, на основании суммарных баллов, в качестве формата выходных данных был выл выбран JSON. Чтобы было проще работать с данным форматом, была взята библиотека Gson.

Для более удобного использования функций системы было принято решение реализовать графический интерфейс, используя технологию JSP и инструменты HTML, JavaScript, CSS.

Для хранения аккаунтов пользователей был выбран LDAP. LDAP (Lightweight Directory Access Protocol) – это протокол прикладного уровня, который определяет методы для доступа к данным каталога, хранящимся на распределенных в сети серверах. Этот протокол также определяет и описывает

представление данных в службе каталогов (модель данных и информационную модель) и то, каким образом эти данные загружаются и выгружаются из службы каталогов.

В LDAP данные представляются в виде древовидной иерархической структуры, которая состоит из объектов, называемых записями. Верхнюю часть данного дерева принято называть корнем (root), а также базой (base) или суффиксом (suffix). У каждой записи кроме корневой всегда есть ровно одна родительская запись. Все записи могут иметь ноль или более дочерних записей. Каждая запись состоит из одного или более объектных классов. Сами объектные классы в свою очередь состоят из атрибутов, которые имеют имена и содержат собственно сами данные.

Так почему для хранения аккаунтов пользователей был выбран LDAP, а не наиболее популярный способ хранения данных, реляционная база данных? Рассмотрим преимущества LDAP:

– LDAP, в отличие от реляционных баз данных, при чтении и поиске данных работает гораздо эффективнее;

– в процессе функционирования изменить схемы LDAP гораздо проще, чем схемы реляционных баз данных;

– LDAP имеет более высокую степень распределенности;

– LDAP предоставляет единые методы доступа к данным. Иначе говоря, в независимости от того, какая конкретная реализация LDAP используется, внешний интерфейс доступа к данным не меняется. В реляционных базах данных удалённый метод доступа всегда зависит от конкретной реализации;

– в LDAP записи имеют более простые взаимосвязи в отличие от объектов реляционных баз данных;

– в базах данных транзакции сложнее, чем в LDAP;

– один каталог LDAP может эффективно использоваться сразу несколькими разными приложениями, в то время как одна реляционная база данных разрабатывается только для конкретного приложения.

Теперь рассмотрим основные недостатки использования LDAP:

– проигрывает при выполнении операции записи. Увеличение нагрузки при операциях записи вызвано тем, что происходит обновление индексов, которые необходимы для ускорения операции поиска;

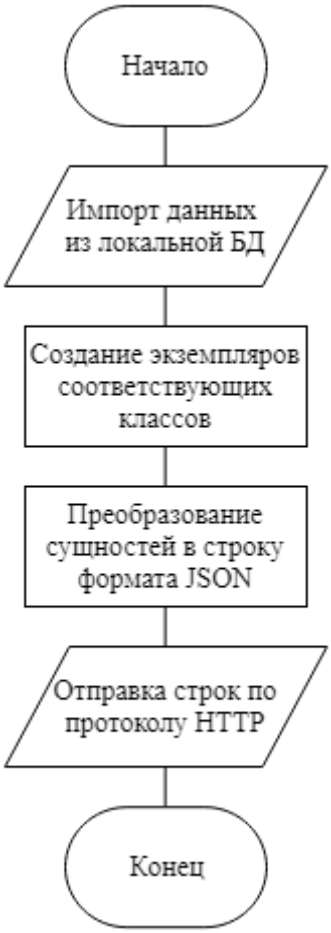
– реляционные базы данных для отката при неудачных операциях предоставляют более гибкие возможности. Отсюда следует, что LDAP для выполнения операции модификации подходит гораздо меньше.

В данном случае запись или модификация аккаунтов пользователей (добавление нового аккаунта, смена пароля у существующего аккаунта и т. д.) будут производиться довольно редко. В основном будет производится только операция чтения (для осуществления авторизации). Поэтому, исходя из описанных выше достоинств и недостатков, можно сделать вывод о том, что для работы с аккаунтами пользователей LDAP подходит как нельзя лучше.

Реализация системы подготовки данных для публикации расписания занятий выполнена на мультипарадигмальном языке программирования Scala и объектно-ориентированном языке программирования Java. Разработка велась в среде IntelliJ IDEA. В качестве сборщика проектов применялся SBT. Также для управления разработкой использовалась распределённая система управления версиями Git.

3.2 Реализация функций системы подготовки данных для публикации расписания занятий

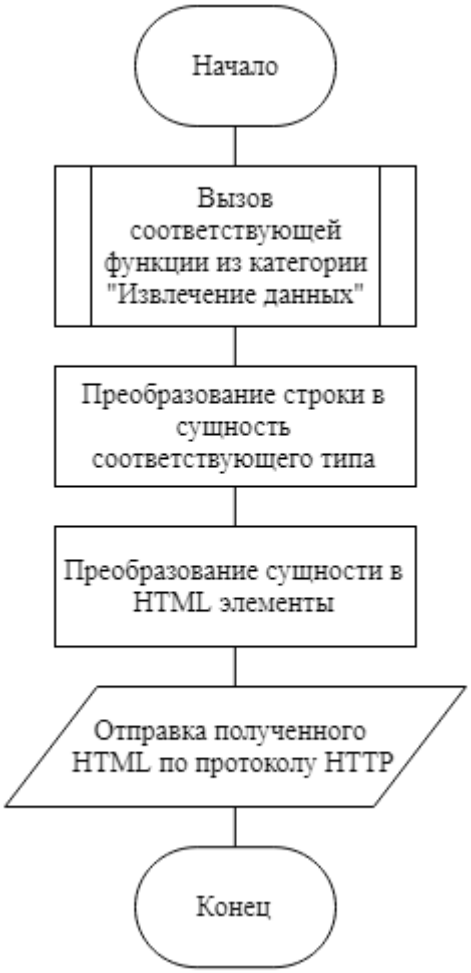
Функции из категории «Извлечение данных» были реализованы следующим образом (см. блок-схему 1). Реализация данных функций в виде программного кода представлена в приложении А, в файле Import.java. Вначале, посредством JDBC драйвера производится извлечение данных и локальной базы данных. Затем создаются сущности, соответствующие этим данным. Используя библиотеку Gson, данные сущности преобразуются в JSON объекты, которые впоследствии преобразуются этой же библиотекой в строки. Полученные строки отправляются по протоколу HTTP в качестве ответа.



Блок-схема 1 – Реализация функций «Извлечение данных»

Рассмотрим функцию getMainPage. Реализация данной функции в виде программного кода представлена в приложении А, в файле Main.scala. Главные страницы хранятся в виде файлов. При запросе необходимая главная страница сразу отправляется пользователю по протоколу HTTP.

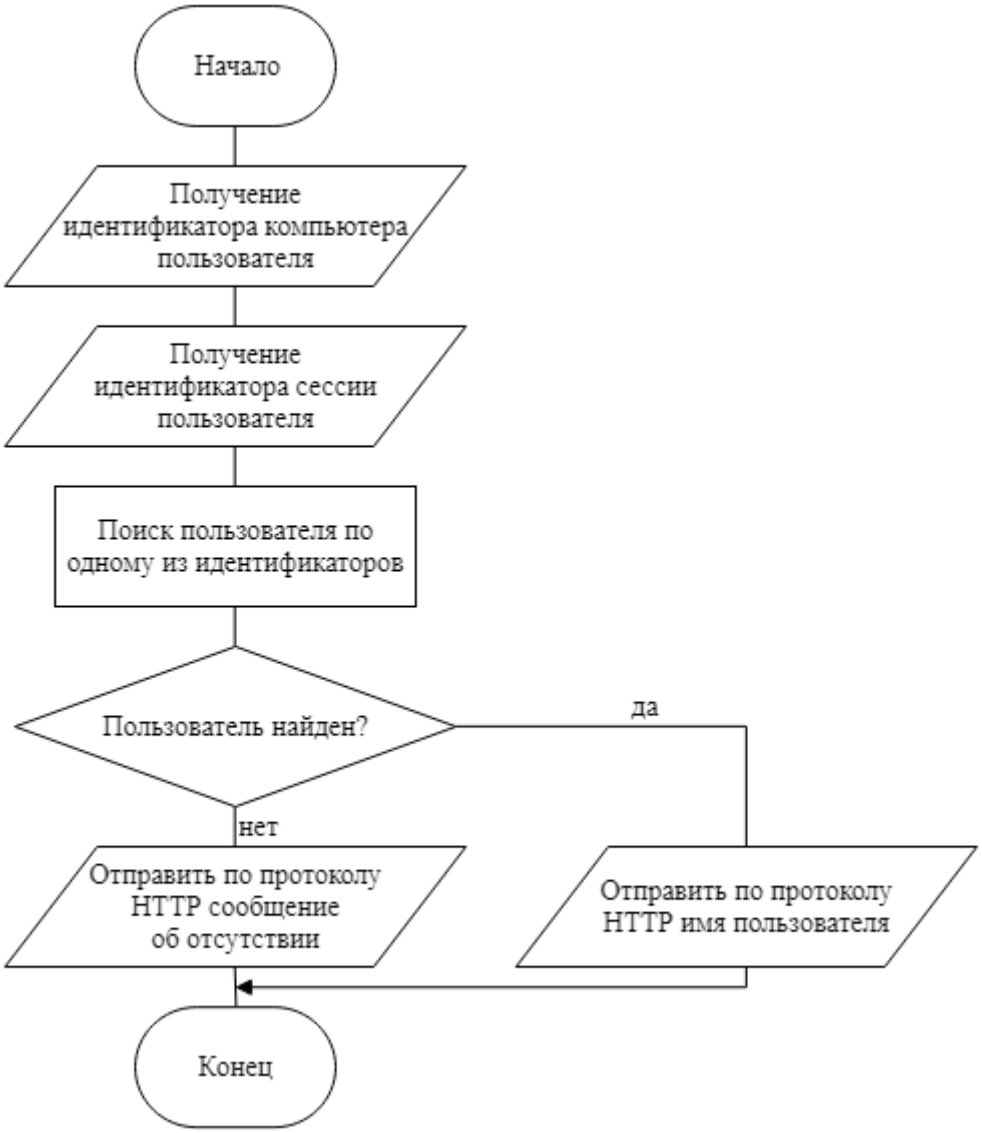
Функции getSearchResults, timeTable, customTimeTable, fixedTimeTable, forPrint имеет следующую реализацию (см. блок-схему 2). Реализация данных функций в виде программного кода представлена в приложении А, в файле Main.scala. Вначале, как и в случая с функциями из категории «Извлечение данных», происходит извлечение данных и преобразование их в строку формата JSON. Дальше эта строка отправляется по протоколу HTTP модулю-обработчику. Полученную строку модуль преобразует в HTML элементы и отправляет их по протоколу HTTP в качестве ответа.



Блок-схема 2 – Реализация функций поиска и создания таблиц с расписанием

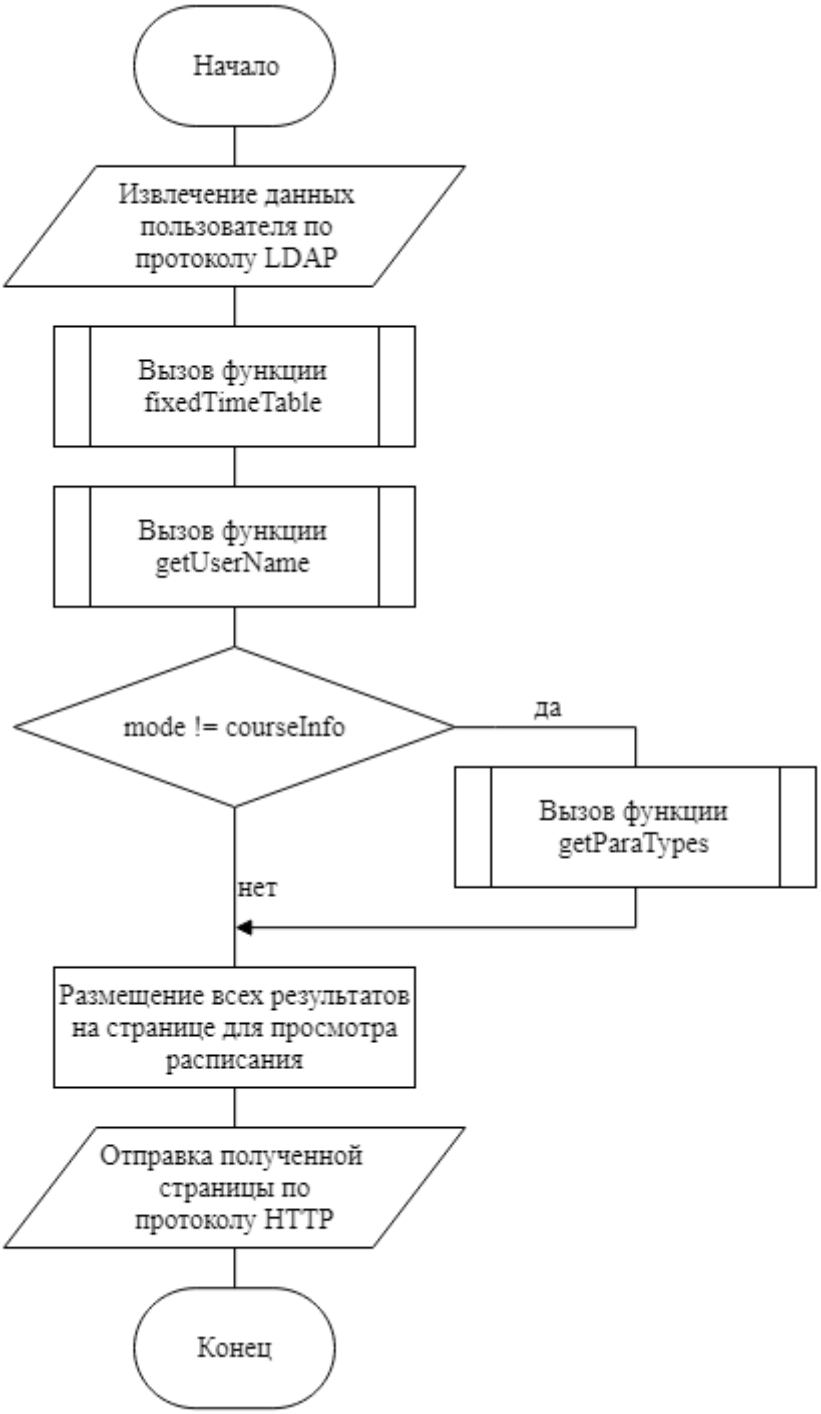
Рассмотрим функцию getParaTypes. Реализация данной функции в виде программного кода представлена в приложении А, в файле Main.scala. Данная функция вызывает обычную функцию ParaType. Из её результата она делает JSON массив, состоящий из названий типов занятий («лек», «пр» и т. д.). В конце полученный массив отправляется по протоколу HTTP.

Функция getUserName возвращает имя текущего пользователя. Рассмотрим её реализацию (см. блок-схему 3). Реализация данной функции в виде программного кода представлена в приложении А, в файле UserService.scala. Вначале система получает от браузера пользователя сохранённые идентификаторы сессии и его компьютера. Затем она осуществляет поиск пользователя по этим идентификаторам через протокол LDAP. Если пользователь был найден, то делается отправка его имени по протоколу HTTP, иначе отправляется сообщение об отсутствии пользователя.



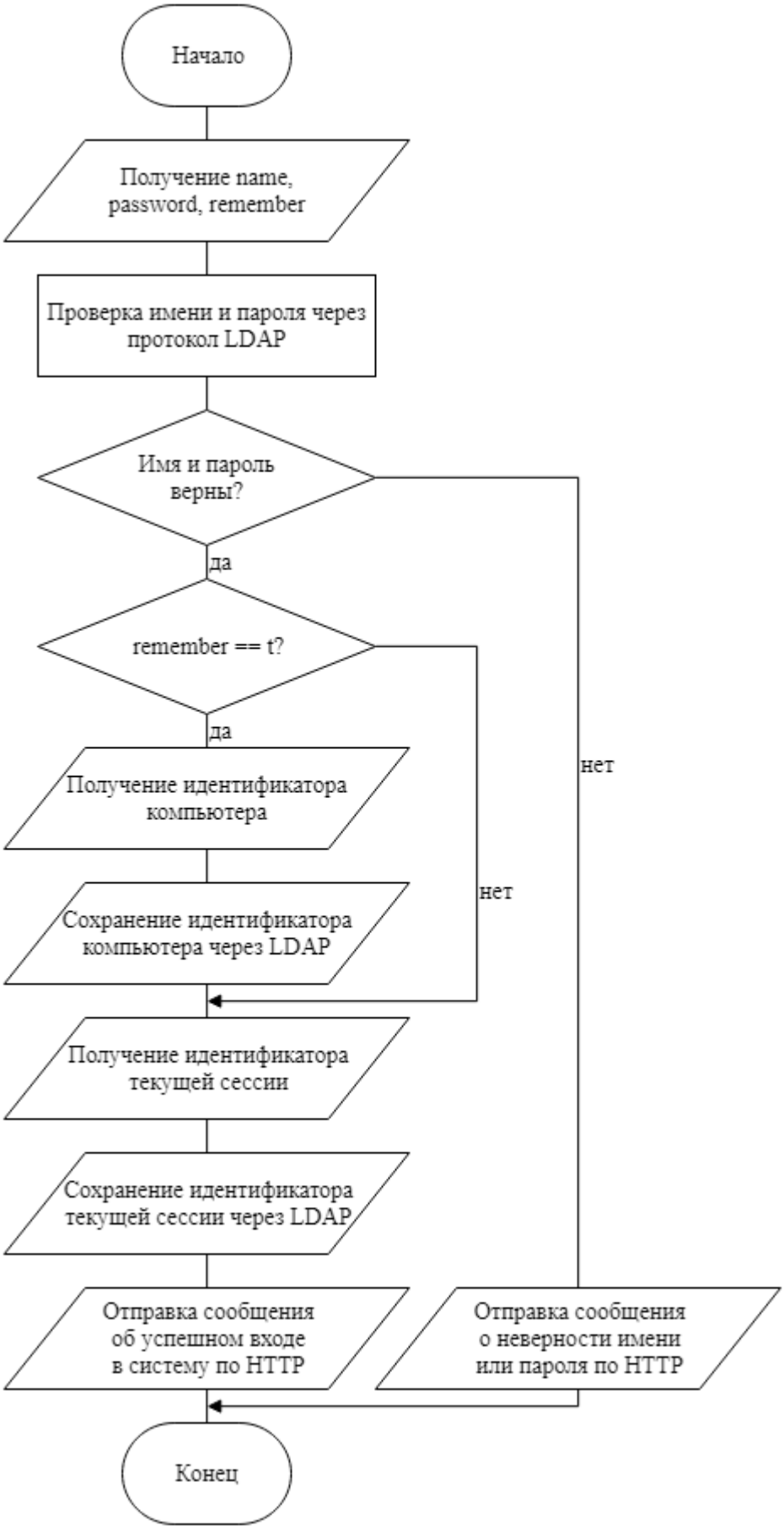
Блок-схема 3 – Реализация функции getUserName

Теперь рассмотрим функцию timeTableForm. Она имеет следующую реализацию (см. блок-схему 4). Реализация данной функции в виде программного кода представлена в приложении А, в файле Main.scala. Вначале извлекаются данные пользователя по протоколу LDAP. Дальше с помощью функции fixedTimeTable получаем таблицу с расписанием. С помощью функции getUserName получаем имя текущего пользователя. Если mode не равен courseInfo, то вызываем функцию getParaTypes. Затем все результаты размещаются на странице для просмотра расписания. В конце наполненная данными страница для просмотра расписания отправляется по протоколу HTTP пользователю.



Блок-схема 4 – Реализация функции timeTableForm

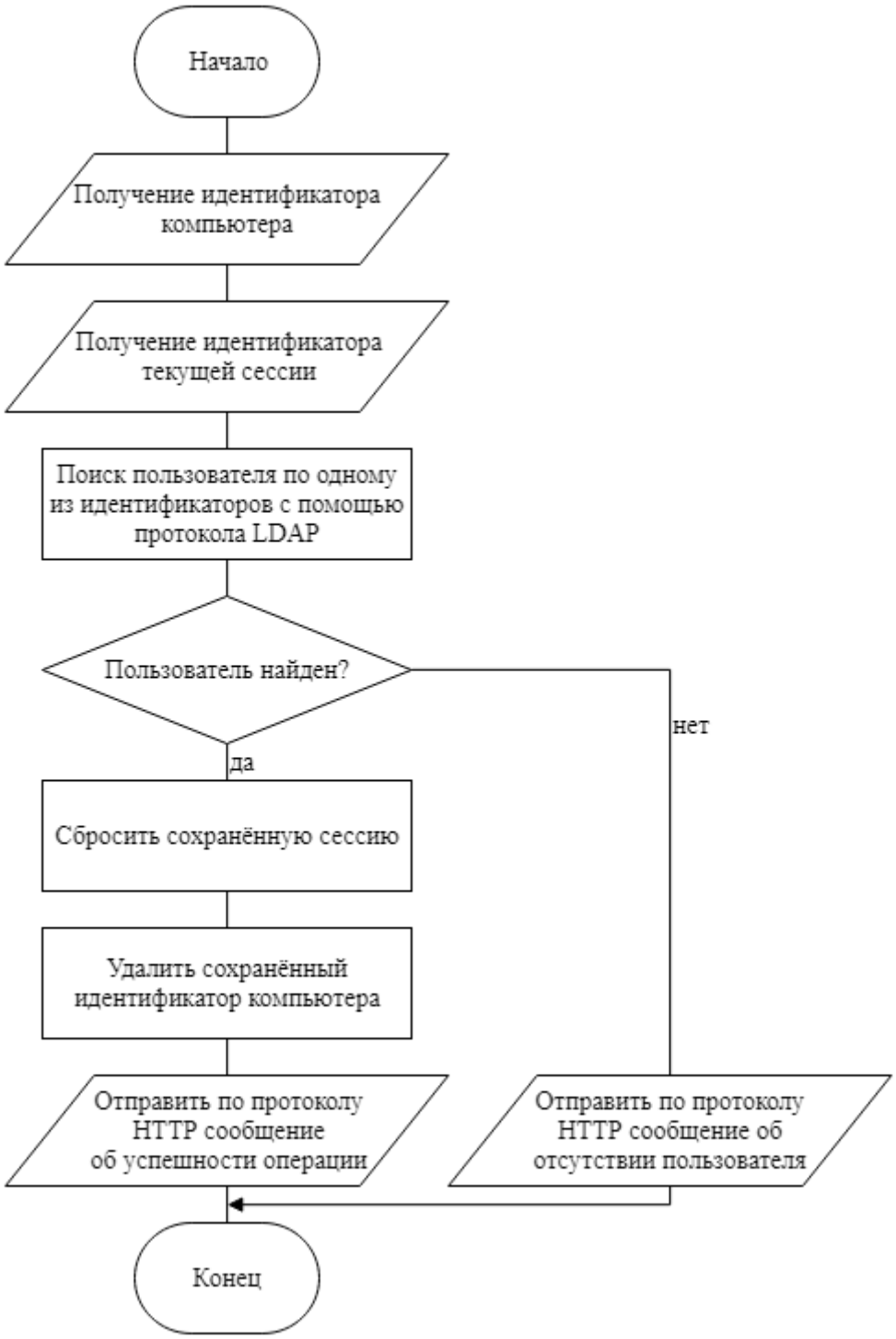
Функция login. Она имеет следующую реализацию (см. блок-схему 5). Вначале введённые пользователем имя и пароль сопоставляются с теми, которые хранятся в базе данных аккаутов. Если введённые данные верны, то по желанию пользователя сохраняется идентификатор его компьютера. Дальше сохраняется идентификатор текущей сессии. В конце отправляется сообщение об успешности операции или неудачи, если введённые данные неверны.



Блок-схема 5 – Реализация функции login

Реализация функции login в виде программного кода представлена в приложении А, в файле UserService.scala.

Функция logout. Она имеет следующую реализацию (см. блок-схему 6). Реализация данной функции в виде программного кода представлена в приложении А, в файле UserService.scala.



Блок-схема 6 – Реализация функции logout

Вначале система получает идентификаторы текущей сессии и компьютера пользователя. Затем по этим идентификаторам ищет аккаунт пользователя. Если аккаунт найден, то полученные идентификаторы удаляются и отправляется сообщение об успешности выхода. В случае отсутствия результатов поиска отправляется соответствующее сообщение.

Таким образом, используя выбранные ранее инструменты, были реализованы данные функции.

3.3 Анализ статистики работы системы подготовки данных для публикации расписания занятий

Для оценки работы системы подготовки данных для публикации расписания занятий было введено логирование. Его целью является запись информации о каждом событии, происходящем в системе, в файл. Информация о событии представляет собой JSON строку формата «event, date, time, details: {}», где event – это строковая переменная, которая содержит название события (например, «initBegin»), date – строковая переменная, хранящая дату этого события в формате «ГГГГ-ММ-ДД» (например, «2019-04-28»), time – строковая переменная, содержащая время события в формате «ЧЧ:ММ» (например, «08:47»), details – переменная, которая представляет собой структуру, хранящую дополнительную информацию о событии. Пример одной такой записи изображён на рисунке 12.

Так для оценки скорости подготовки представления вторым модулем с использованием кэширования и без него велась запись информации о каждом запросе представления. Структура этой информации представлена на рисунке 12. Если в переменной event содержится «timeTableFresh», то это означает, что представление создавалось заново, а не бралось кэша. Значение «timeTableCache» наоборот говорит о том, что представление было взято и кэша. В переменную dataReceiveTime (целочисленный тип) записывается время (в миллисекундах), которое прошло с момента получения вторым модулем запроса на выдачу представления до момента отправки ответа всё тем же вторым

модулем. На основании значения этой переменной и был проведён анализ скорости запросов на получение представления расписания.

Рисунок 12 – Информация о событии

Вначале были отобраны все события, у которых переменная event равна «timeTableFresh» или «timeTableCache», а значение переменной date находится на отрезке от 25 февраля 2019 года до 24 марта 2019 года. Затем по каждому дню было вычислено среднее арифметическое значений переменной dataReceiveTime (сумма всех значений, делённая на их количество). Далее был построен график, наглядно представляющий результаты (см. рисунок 13). Из графика видно, что, начиная с 4 марта 2019 года, время подготовки представления расписания значительно снизилась. Именно в этот день и было введено кэширование представлений. Подсчитаем среднее арифметическое отрезка от 25 февраля 2019 года до 3 марта 2019 года. Получим 63,95 миллисекунд. Аналогично подсчитаем среднее арифметическое отрезка от 4 марта 2019 года до 24 марта 2019 года. В результате получим 28,27 миллисекунд.

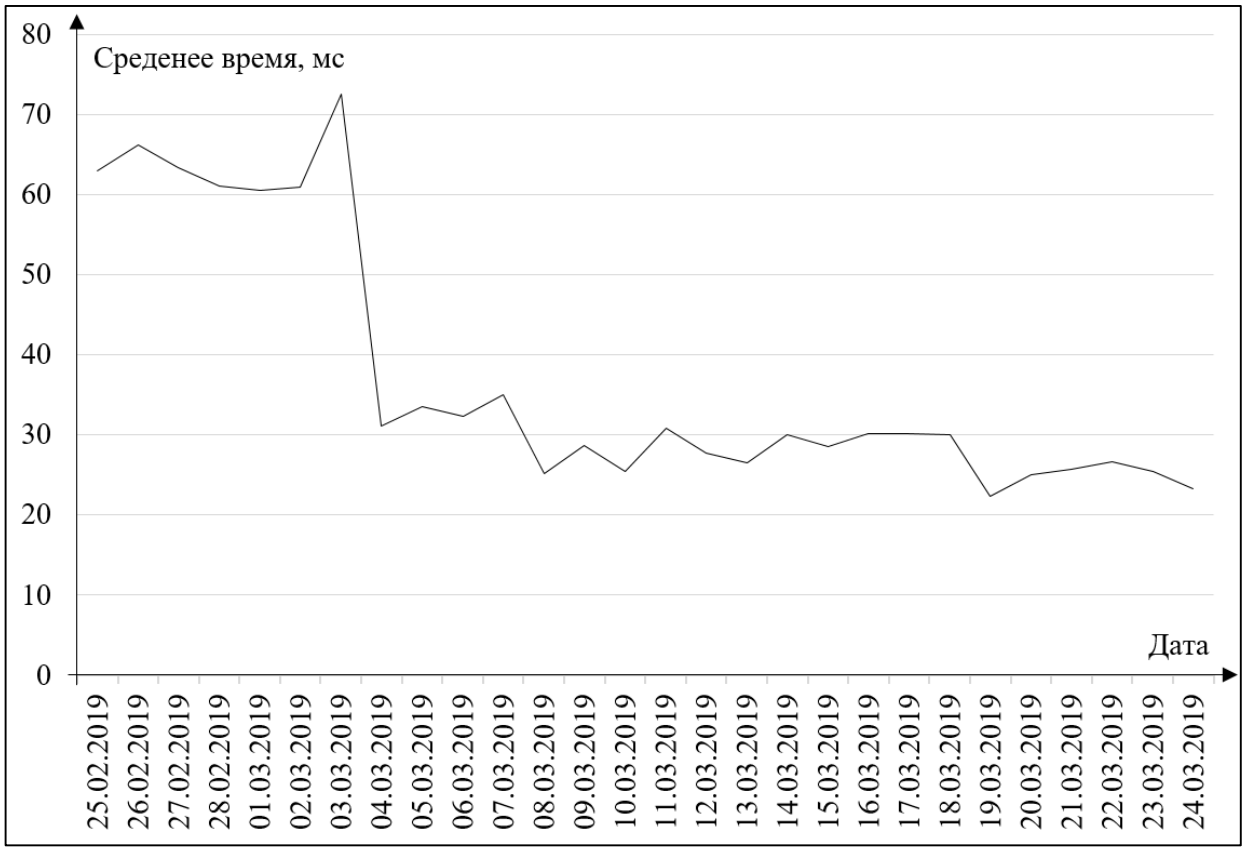


Рисунок 13 – Результаты анализа скорости

Таким образом можно сделать вывод о том, что введение кэширования позволило снизить время подготовки представления на 55,8%.

3.4 Выводы и результаты по разделу 3

В первом подразделе были определены инструменты для реализации системы подготовки данных по расписанию учебных занятий. Был сделан вывод о том, что самым лучшим инструментом для реализации системы является объектно-ориентированный язык программирования Java. Также выяснено что лучшим форматом передачи данных по сети является JSON.

Во втором подразделе с помощью выбранных инструментов были реализованы все выделенные функции системы.

В третьем подразделе был проведен анализ статистики работы системы, в результате которого был сделан вывод о том, что использование подсистемы кэширования снижает время подготовки расписания учебных занятий на 55,8%.

ЗАКЛЮЧЕНИЕ

В ходе данной работы был проведён анализ процесса публикации расписания учебных занятий в ТГУ. В результате было выяснено, что публикация расписания учебных занятий осуществляется вручную сотрудником диспетчерской службы и на выходе получаются файлы формата xls (xlsx). Всё это в совокупности делает данный процесс менее эффективным. Таким образом, чтобы устранить выявленные недостатки, было принято решение о разработке системы подготовки данных для публикации расписания занятий. Было выявлено, что необходимо реализовать 15 требований.

Опираясь на выработанные требования были разработаны конечные автоматы, которые описывают процессы, происходящие в системе. На основе полученных конечных автоматов, были выделены функции системы. Была разработана архитектура системы подготовки данных для публикации расписания занятий. В её основе лежат три модуля: модуль извлечения данных, модуль обработки данных, модуль публикации данных. Описано взаимодействие этих модулей.

После окончания проектирования системы, были выбраны необходимые инструменты для её реализации. Затем были реализованы все функции системы, выделенные ранее. В конце был проведен анализ статистики работы полученной системы. Было выяснено, что использование подсистемы кэширования снижает время подготовки расписания учебных занятий на 55,8%.

Разработанное Web API может использоваться высшими учебными заведениями для публикации расписания учебных занятий. Так данное приложение уже активно используется сотрудниками ТГУ.

В будущем планируется расширять функционал Web API. Так как была реализована авторизация пользователей, можно будет, например, добавить возможность создавать свои собственные события, которые впоследствии будут отображаться наравне с расписанием учебных занятий.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Введение в программные системы и их разработку [Электронный ресурс] : [учеб. пособие] / С. В. Назаров [и др.]. – 2-е изд., испр. – Москва : ИНТУИТ, 2016. – 649с. : ил.

2. Интерфейс. Основы проектирования взаимодействия [Текст] / А. Купер, Р. М. Рейманн, Д. Кронин, К. Носсел. – М. : Питер, 2017. – 720 с. – ISBN 978-5-496-01718-3.

3. Макконнелл, С. Совершенный код. Мастер-класс [Текст] / С. Макконнелл. – М. : БХВ-Петербург, 2017. – 896 с. – ISBN 978-5-9909805-1-8.

4. Абельсон, Х. Структура и Интерпретация Компьютерных Программ [Текст] / Х. Абельсон, Д. Д. Сассман. – М. : Добросвет, 2018. – 608 с. – ISBN 978-5-98227-708-4.

5. Седжвик, Р. Алгоритмы на Java [Текст] / Р. Седжвик, К. Уэйн. – М. : Вильямс, 2016. – 848 с. – ISBN 978-5-8459-2049-2.

6. Приемы объектно-ориентированного проектирования. Паттерны проектирования [Текст] / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – М. : Питер, 2016. – 366 с. – ISBN 978-5-459-01720-5.

7. Лиэнг, Ш. Интерфейс JNI. Руководство по программированию и спецификация [Текст] / Ш. Лиэнг. – М. : ДМК Пресс, 2018. – 280 с. – ISBN 978-5-97060-586-8.

8. Шилдт, Г. Java 8. Полное руководство [Текст] / Г. Шилдт. – М. : Вильямс, 2017. – 1376 с. – ISBN 978-5-8459-1918-2.

9. Кознов Д. В. Введение в программную инженерию [Электронный ресурс] : [учебное пособие] / Д. В. Кознов. – Москва : ИНТУИТ, 2016. – 306 с. : ил.

10. Программирование на Java для начинающих / Майк Мак-Грат ; [пер. с англ. М.А. Райтмана]. – Москва : Издательство «Э», 2016. – 192 с.

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

12. Одерски М., Спун Л., Веннерс Б. Scala. Профессиональное программирование = Programming in Scala: Updated for Scala 2.12. – Питер, 2018. – 688 с. – ISBN 978-5-496-02951-3.

13. Прокопец А. Конкурентное программирование на SCALA. – ДМК пресс, 2017. – 342 с. – ISBN 978-5-97060-572-1.

Электронные ресурсы

14. Создаём веб-приложение с Java Servlets [Электронный ресурс]. – 2018. – Режим доступа: https://tproger.ru/translations/building-a-web-app-with-java-servlets/.

15. Структура и органы управления образовательной организацией [Электронный ресурс]. – 2019. – Режим доступа: https://www.tltsu.ru/sveden/struct/.

Литература на иностранном языке

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

17. Bartholomew, Daniel. MariaDB Cookbook. – 2014. p. 145.

18. Scott Oaks. Java Performance: The Definitive Guide: Getting the Most Out of Your Code. – "O'Reilly Media, Inc.", 2014-04-10. – 425 с. – ISBN 9781449363543.

19. Baesens, B. Beginning Java Programming: The Object-Oriented Approach / B. Baesens, A. Backiel, S. Vanden Broucke. – 1st edition, Wrox, 2015.

20. Deitel, H. Java How to Program / H. Deitel, P. Deitel. – 9th edition, Prentice Hall, 2015.

ПРИЛОЖЕНИЕ А

Носитель с исходным текстом

К данной работе прикреплён компакт-диск, на котором находится следующее содержимое:

1. Исходный текст разработанной программы.
2. Текст пояснительной записки ВКР.