Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-ВОСТОЧНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИМЕНИ М.К. АММОСОВА»

Институт математики и информатики

Кафедра «Информационные технологии»

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_/ Н.В. Николаева/

Протокол №\_\_\_\_ от «\_\_\_\_» \_\_\_\_\_\_\_\_ 2023 г.

АВТОМАТИЗАЦИЯ ПРОЦЕДУРЫ ПУБЛИКАЦИИ РАСПИСАНИЯ ИМИ СВФУ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Выполнил: студент IV курса

группы БА-ФИИТ-19 ИМИ

Собянин Сергей Павлович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Научный руководитель: старший преподаватель кафедры «Информационные технологии» ИМИ Эверстов В.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Якутск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc137555824)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ АВТОМАТИЗАЦИИ ПРОЦЕССА ПУБЛИКАЦИЯ РАСПИСАНИЯ В ИМИ СВФУ 6](#_Toc137555825)

[1.1 Постановка задачи 6](#_Toc137555826)

[1.2 Описание API системы публикации расписания СВФУ 20](#_Toc137555827)

[1.3 Обзор аналогов 27](#_Toc137555828)

[1.4 Обзор инструментов разработки 30](#_Toc137555829)

[Выводы по главе 1 36](#_Toc137555830)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕДУРЫ ПУБЛИКАЦИИ РАСПИСАНИЯ СВФУ 38](#_Toc137555831)

[2.1 Требования к ПО 38](#_Toc137555832)

[2.2 Модель классов расписания 40](#_Toc137555833)

[2.3 Модель взаимодействия 41](#_Toc137555834)

[2.4 Архитектура ПО 43](#_Toc137555835)

[2.5 Описание тестового сервера 44](#_Toc137555836)

[2.6 Описание приложения 45](#_Toc137555837)

[Выводы по главе 2 49](#_Toc137555838)

[ЗАКЛЮЧЕНИЕ 50](#_Toc137555839)

[СПИСОК ЛИТЕРАТУРЫ И ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 51](#_Toc137555840)

[ПРИЛОЖЕНИЯ 54](#_Toc137555841)

# ВВЕДЕНИЕ

Составление расписания учебных занятий – очень трудоемкий и кропотливый процесс, требующий большого количества времени и сил. При работе над учебным расписанием необходимо учитывать различные критерии и ограничения. Например, такие, как сложность дисциплины и количество часов, утвержденных учебным планом и требуемых для ее изучения, аудиторный фонд корпуса, в котором проводятся занятия, в частности и учебного учреждения в целом, наличие и количество необходимого для обучения оборудования, равномерное распределение нагрузки на студентов и преподавателей и их численность [2, с. 51].

Эти и многие другие требования необходимо учитывать для успешного составления расписания.

Существуют следующие основные способы составления расписания [2, с. 51]:

1. Ручной. Большой плюс этого метода заключается в визуализации и материализации, однако главный минус заключается в том, что становится трудно вносить изменения уже после публикации расписания.
2. Автоматизированный. Как становится понятно из названия, при применении этого метода пользуются специализированным программным обеспечением (ПО). Такой способ позволяет значительно ускорить работу, помогает учитывать все критерии и требования, необходимые для составления расписания, упрощает процесс последующего изменения и исправления.

После того, как расписание составлено и откорректировано, его необходимо опубликовать. Используя для этого ручной метод, придется потратить огромное количество бумажных носителей, а в случае появления необходимости внесения каких-либо изменений, процесс скорее всего придется начинать с самого начала. На такой случай существуют различные электронные визуализаторы. С их помощью можно отказаться от бумажных носителей, а составление, публикация и последующая корректировка расписания займет совсем небольшое количество времени, если сравнивать с ручным методом, что делает автоматизированный способ является наиболее эффективным и удобным.

На данный момент расписание института математики и информатики СВФУ хоть и составляется с применением некоторых инструментов автоматизации, но в основном все данные заносятся вручную. Учитывая количество курсов и групп, этот процесс становится весьма ресурсозатратным, следовательно, требует улучшений.

Из всего вышесказанного становится понятно, что в настоящее время существует проблема слабой автоматизации или и вовсе ее отсутствия у имеющейся системы публикации расписания в нашем институте.

Таким образом, цель моей выпускной квалификационной работы заключается в следующем: автоматизировать рутинный процесс публикации учебного расписания института математики и информатики Северо-Восточного федерального университета им. М.К. Аммосова.

Для достижения поставленной цели были определены следующие задачи:

1. анализ расписания ИМИ и системы публикации расписания СВФУ;
2. обзор библиотек и инструментов разработки;
3. проектирование собственного решения по автоматизации процедуры публикации расписания.

Объект исследования: процесс публикации и обновления расписания занятий института математики и информатики.

Предмет исследования: методы и технологии автоматизации процедуры публикации расписания ИМИ СВФУ.

Гипотеза исследования: автоматизация процедуры публикации расписания ИМИ СВФУ упростит и ускорит работу ответственных за это лиц и работу института в целом, а также облегчит процесс редактирования уже опубликованного расписания и убыстрит поставку актуального расписания студентам.

Методы исследования: анализ предметной области и литературы; аналогия и сравнение с альтернативными решениями; сбор требований; моделирование тестового сервера для тестирования и эксплуатации ПО.

Данная работа состоит из введения, двух глав, выводов по каждой главе, заключения и списка использованной литературы из 18 наименований, трех приложений, содержит 69 страниц, а также 26 рисунков.

В первой главе проведен обзор предметной области: анализ расписания и системы и процесса его публикации в ИМИ СВФУ.

Во второй главе описывается архитектура приложения и его работа. В заключении подводятся итоги проделанной работы, приведены выводы из результатов проделанной работы.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ АВТОМАТИЗАЦИИ ПРОЦЕССА ПУБЛИКАЦИЯ РАСПИСАНИЯ В ИМИ СВФУ

# 1.1 Постановка задачи

Обзор существующих форматов расписания занятий ИМИ

На сегодняшний день основным источником учебного расписания для студентов ИМИ является Excel-файл (также имеется его менее удобная PDF-версия). Данный файл представляет собой документ, состоящий из нескольких страниц, каждая из которых посвящена отдельному курсу. Каждая страница файла оформлена в виде таблиц, отведенных каждой учебной группе соответствующего курса, состоящих из таких столбцов, как наименование группы, общее количество обучающихся. Также указывается время, день недели, ФИО преподавателя, вид учебной деятельности и аудитория.

Само расписание составляется и формируется в Excel-файл следующим образом:

На основании учебных планов и выписок из карточек учебных поручений преподавателей заполняется номенклатура дисциплин на текущий семестр учебного года по учебным группам: какие предметы будут у студентов в этом семестре, и какой преподаватель их ведёт.

«Процесс составления расписания обычно начинают с потоковых лекций, занятий по физической культуре, иностранных языков. Учитывается наличие свободных аудиторий; распорядок преподавателей. У студентов не может быть больше 5 пар. Если дисциплина предусматривает и лекции, и лабораторные/практические, то сначала стараемся поставить лекцию. «Окна» между парами крайне нежелательны», – слова ответственной за составление расписания в ИМИ СВФУ, Романовой Натальи Анатольевны, доцента кафедры дифференциальных уравнений и зам. директора по УР ИМИ.

Каждый студент может найти нужное ему расписание учебных занятий на официальном сайте Северо-Восточного федерального университета им. М.К. Аммосова.

Для этого ему необходимо нажать на раздел «Студент» в навигационном меню, из раскрывшегося списка выбрать пункт «Расписание занятий».

На открывшейся странице (рисунок 1) он сразу может выбрать свой институт, после чего будет перенаправлен на другую страницу (рисунок 2), которая содержит файлы с расписанием учебным занятий для различных уровней обучения (например, бакалавриат, аспирантура), для разных форм обучения (очная, заочная) и файлы с расписанием экзаменационных сессий для разных временных промежутков. Все файлы, представленные на странице, в Excel-формате.

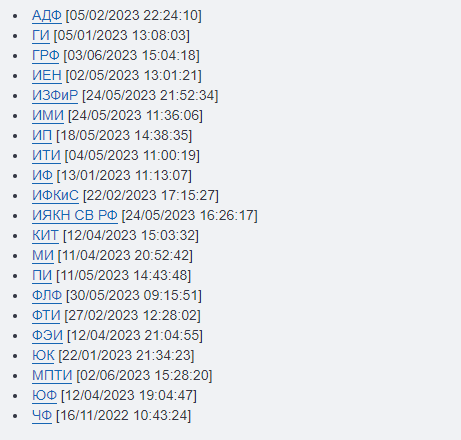


Рисунок 1 – Выбор УЧП



Рисунок 2 – Excel-файлы с расписанием

Есть и другой способ просмотра учебного расписания. Для этого все на той же странице, которая открывается при выборе пункта «Расписание занятий», в верхней части окна необходимо выбрать из выпадающего списка свое учебное подразделение, а также дату, расписание на которую вас интересует в соответствии с рисунком 3 и рисунком 4.

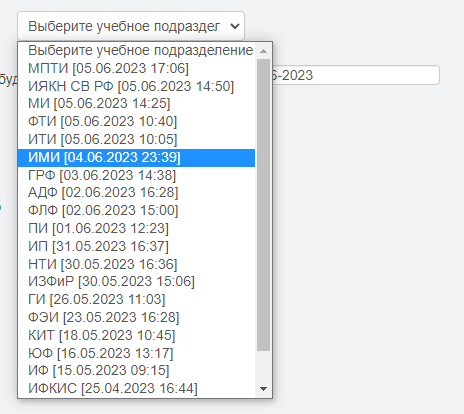


Рисунок 3 – Выбор УЧП

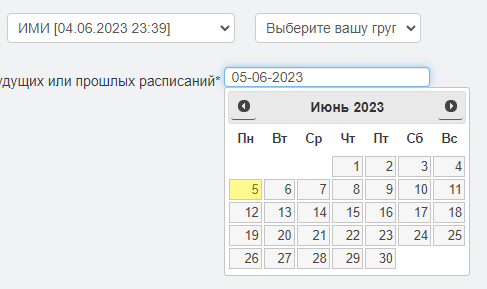


Рисунок 4 – Выбор даты

После того, как выбор сделан, появляется поле с выпадающим списком для выбора учебной группы, чье расписание пользователь хочет узнать (рисунок 5).

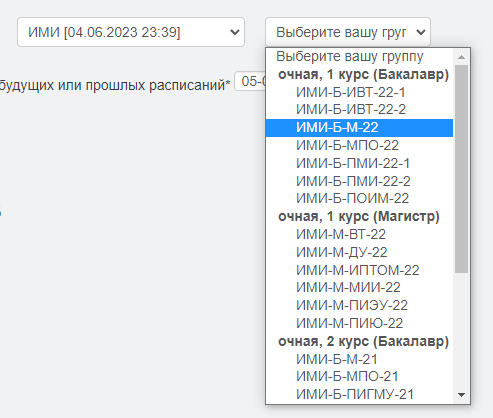


Рисунок 5 – Выбор учебной группы

А на странице выводится связанное с выбранной датой предупреждение о том, что интересующей пользователя группы в выпадающем списке может не быть в связи с тем, что в выбранную дату у такой группы попросту нет занятий (рисунок 6):

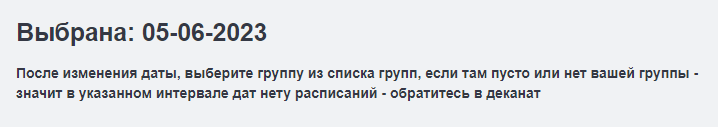


Рисунок 6 – Предупреждение

После выбора учебной группы, на странице отображается расписание на всю неделю, включающую в себя выбранную ранее дату, в соответствии с рисунком 7. Расписание, отображенное на данной странице, представляет собой несколько таблиц, включающих в себя следующие столбцы:

* Время
* Название предмета
* ФИО преподавателя
* Номер аудитории
* Дополнительная информация

Помимо этого, указываются также день недели, число, месяц и год, четность выбранной недели, дата и время последнего изменения расписания.

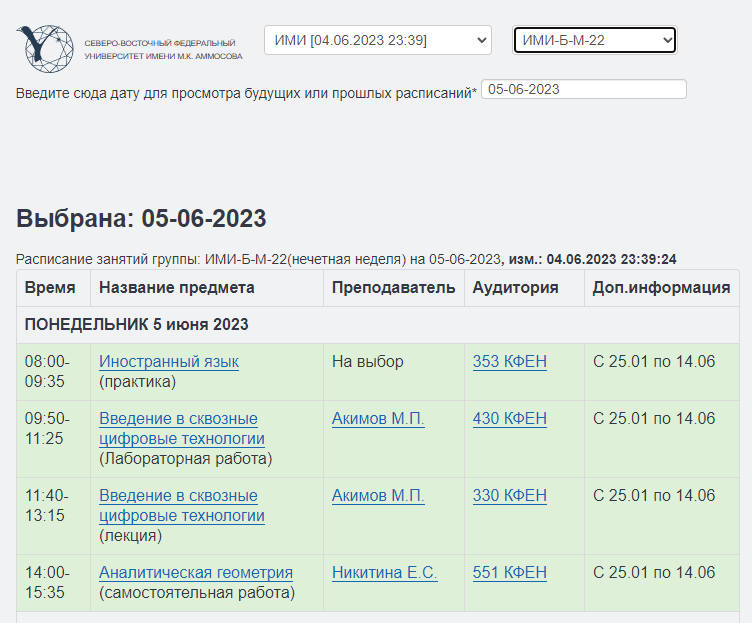


Рисунок 7 – Онлайн-расписание

Онлайн-расписание обладает некоторыми преимуществами в сравнении с тем, что хранится в excel-документе.

Во-первых, при нажатии на дисциплину, можно увидеть вообще все занятия по ней (рисунок 8).

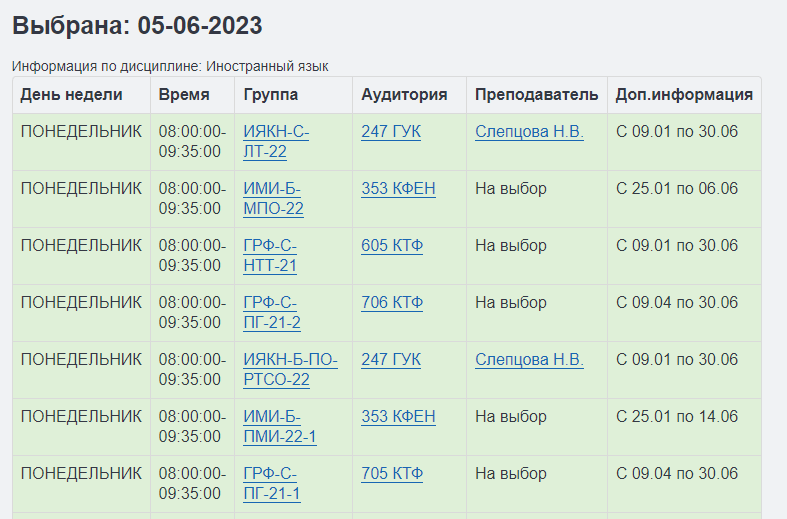


Рисунок 8 – Информация по дисциплине

На рисунке 9 можно увидеть, как при нажатии на преподавателя система отображает расписание с информацией обо всех его занятиях.



Рисунок 9 – Расписание занятий преподавателя

При нажатии на аудитории можно лицезреть все занятия, которые в ней проводятся (рисунок 10).

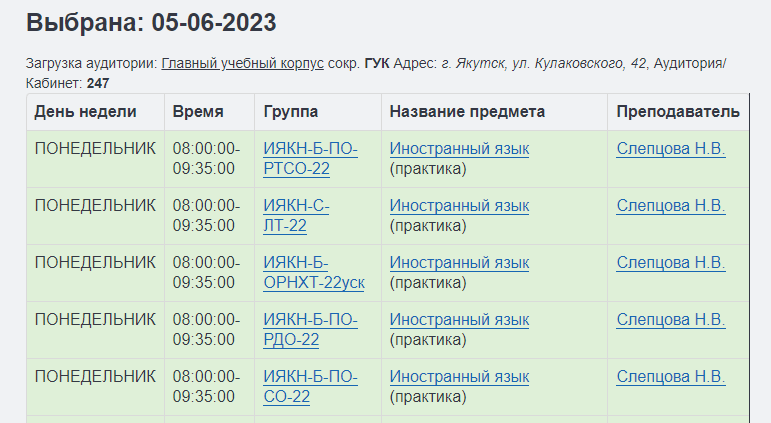


Рисунок 10 – Загруженность аудитории

Описание процедуры публикации расписания ИМИ СВФУ

На данный момент расписание в ИМИ, как и в других институтах и факультетах, составляется и публикуется через систему публикации расписания УСП СВФУ. Оно основывается на уже составленном расписании, хранящемся в файлах Excel-формата.

Мною проанализирована веб-страница, на которой формируется и публикуется расписание, а также отправляемые запросы.

Формируется и публикуется расписание в данной системе следующим образом:

1. Для начала необходимо авторизоваться в личном кабинете на сайте СВФУ, потому что не у всех зарегистрированных пользователей есть доступ к этой системе;
2. После авторизации, если у пользователя есть права, у него в личном кабинете в меню отображается пункт «Публикация расписания» (рисунок 11);

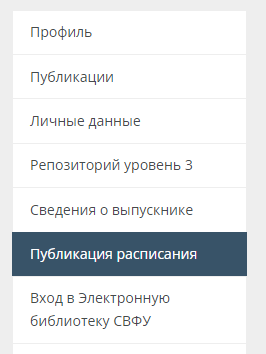


Рисунок 11 – Система публикации расписания

1. При нажатии на пункт «Публикация расписания» происходит перенаправление на веб-страницу, по умолчанию имеющую 5 полей с выпадающими списками и кнопку для выбора группы и учебного плана (рисунок 12):
   1. Первое поле выбора: учебный год. По умолчанию указан текущий учебный год;
   2. Второе поле выбора: курс. По умолчанию указан первый курс;
   3. Третье поле выбора: форма обучения. По умолчанию указана очная;
   4. Четвертое поле выбора: уровень обучения. По умолчанию указан бакалавриат;
   5. Пятое поле выбора: семестр. По умолчанию указан текущий учебный семестр. Данное поле является обязательным.

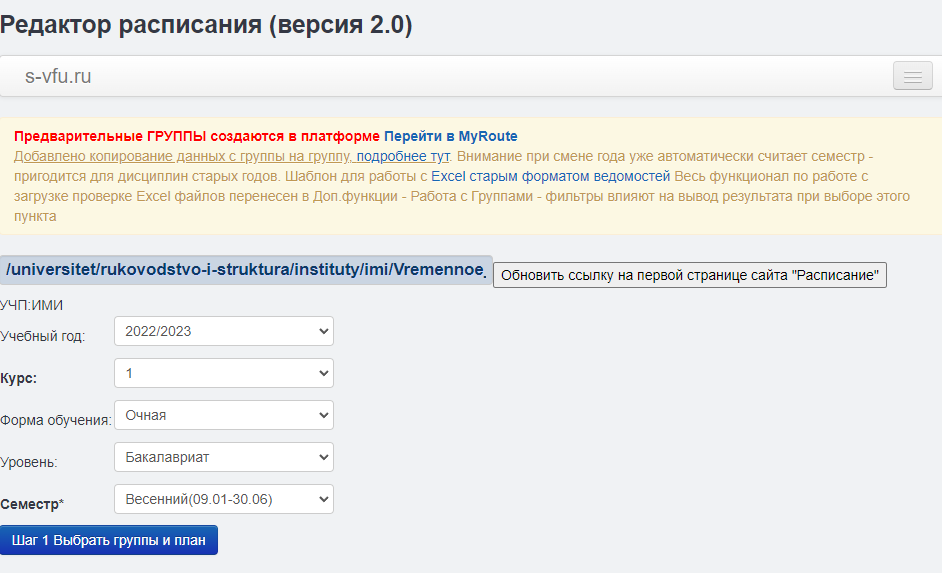


Рисунок 12 – Заполнение стартовой формы

1. После заполнения всех полей и нажатия на кнопку «Шаг 1 Выбрать группы и план», в соответствии с рисунком 13 появляется новое поле с выпадающим списком, разделенным на две категории: «Без расписания» и «Есть расписание». Список состоит из полей, содержащих наименование учебной группы, код и наименование направления подготовки, а также количество лет обучения и число обучающихся студентов.

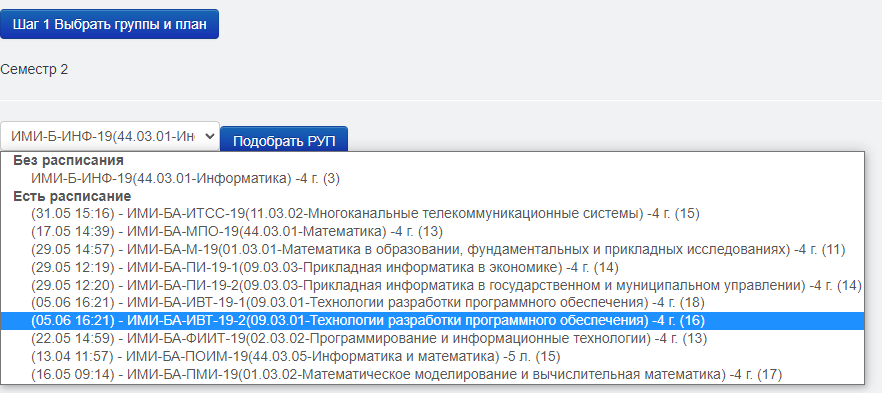


Рисунок 13 – Выбор группы и плана

1. При выборе группы и плана и нажатии на кнопку «Подобрать РУП» появляется модальное окно, содержащее чекбокс для дополнительного отображения предыдущего и следующего семестров (рисунок 14). Также присутствует кнопка-переключатель, предлагающая применить уже существующий план или выбрать новый. Напоследок присутствуют поля с выбором даты начала и окончания семестра. Если их не заполнять, то будут применены даты по умолчанию. В самом конце есть две стандартные кнопки: «Применить» и «Закрыть».

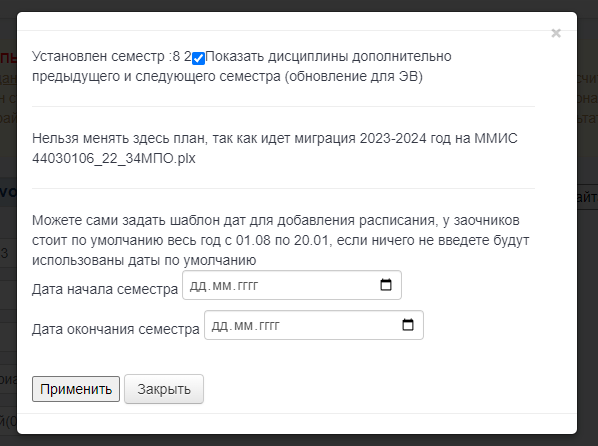


Рисунок 14 – Выбор РУП

1. После нажатия на «Применить» появляется таблица с расписанием, которое можно редактировать. Еще отображается кнопка добавления новой строки и сохранения всех введенных изменений с последующей публикацией (рисунок 15).

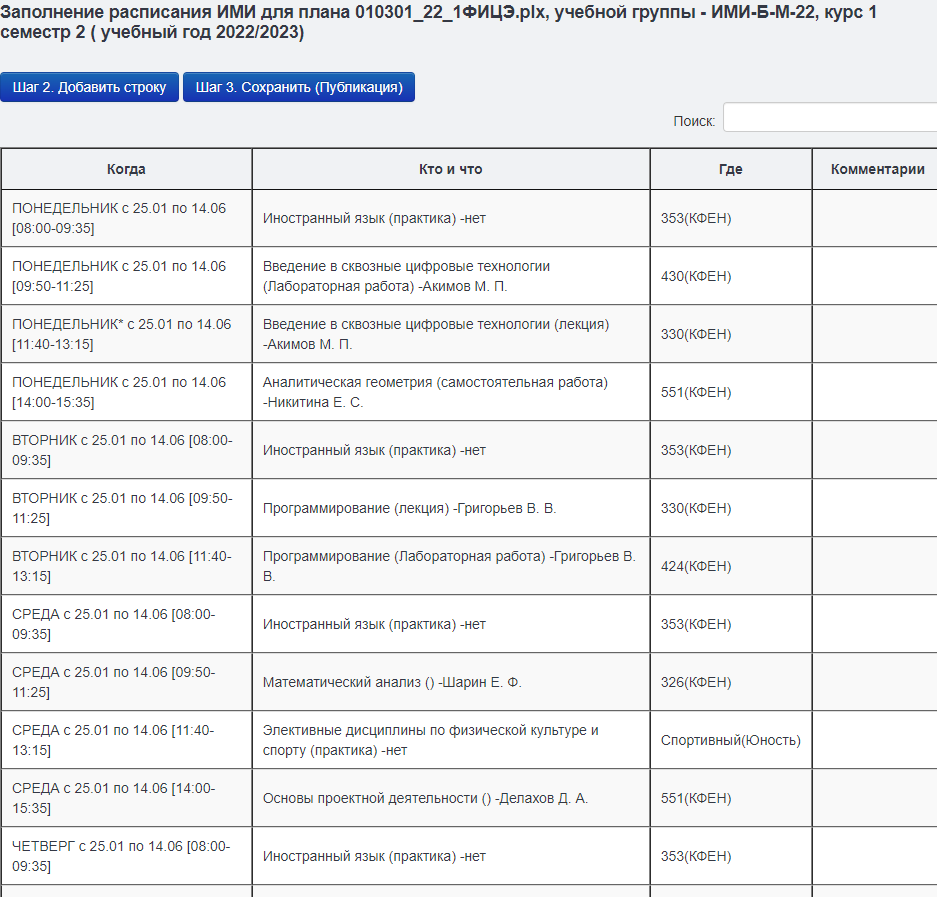


Рисунок 15 – Таблица с расписанием

1. При изменение существующих строк таблицы или добавлении новых отображается модальное окно.

Оно содержит в себе следующие поля:

* Выбор дисциплины. Выпадающий список, поделенный на категории, в качестве разделителя выступает номер семестра. У каждого семестра свои дисциплины, с указанием отводимых на их изучение часов и типа контроля (зачет, зачет с оценкой и так далее);
* ППС. Поле с автозаполнением. Проанализировав веб-страницу с добавлением расписания и запросы, отправляющиеся с нее, мною было установлено, что при изменении текста в данном поле отправляется GET-запрос по адресу «https://www.s-vfu.ru/stud/searchadddata.php?tablename=svfudbnew.forexcel&term=але», где в качестве параметров передается наименование таблицы и текст, введенный пользователем;
* Почасовики. Выпадающий список. Анализ показал, что поля с данными сразу присутствуют в HTML-коде страницы;
* Подгруппа. Выпадающий список из 9 подгрупп;
* День недели. Выпадающий список из 7 дней недели.
* Пара. Выпадающий список из временных промежутков 6 пар;
* Чётность. Выпадающий список с полями «чет/нечет». Если оставить поле пустым, то это будет значить, что дисциплина не чередуется по неделям;
* Дата начала. По умолчанию указаны даты, выбранные ранее;
* Дата окончания. По умолчанию указаны даты, выбранные ранее;
* Тип учебного занятия. Выпадающий список. Анализ показал, что поля с данными сразу присутствуют в HTML-коде страницы;
* Корпус. Анализ данных также показал, что поля с данными изначально есть в HTML-коде страницы, однако поле с автозаполнением, возможное благодаря подключенному скрипту, написанному на jQuery;
* Аудитория. При выборе корпуса появляется поле с выбором аудитории. Анализ показал, что при выборе корпуса и аудитории отправляется POST-запрос по адресу «https://www.s-vfu.ru/user/rasp/new/ajax.php».

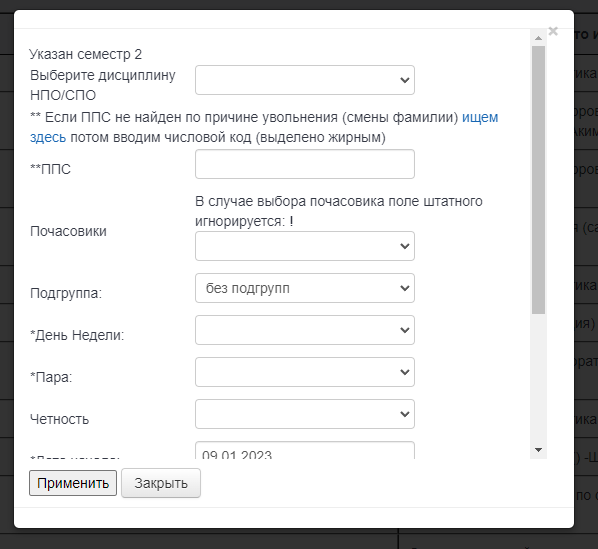


Рисунок 16 – Добавление строки с занятием в таблицу

Помимо добавления новых строк, в системе есть множество других функций по работе с расписанием занятий группы. При нажатии на соответствующую строку, открывается контекстное меню со всеми возможностями системы:

* исправить ячейку;
* удалить строку;
* создать копию строки;
* удалить расписание; очистить расписание;
* добавить студентов;
* поиск свободных аудиторий;
* создать копию расписания группы по курсу;
* создать копию расписания группы по направлению;
* создать копию расписания группы по со сдвигом года.

1. После всех преобразований над расписанием можно опубликовать расписание на сайте СВФУ нажатием кнопки «Шаг 3. Сохранить (Публикация)» (рисунок 15). Все внесенные изменения отразятся в расписании на сайте, представленном на рисунке 7.

Ниже на рисунке 17 представлена схема процесса публикации расписания с применение BPMN 2.0.

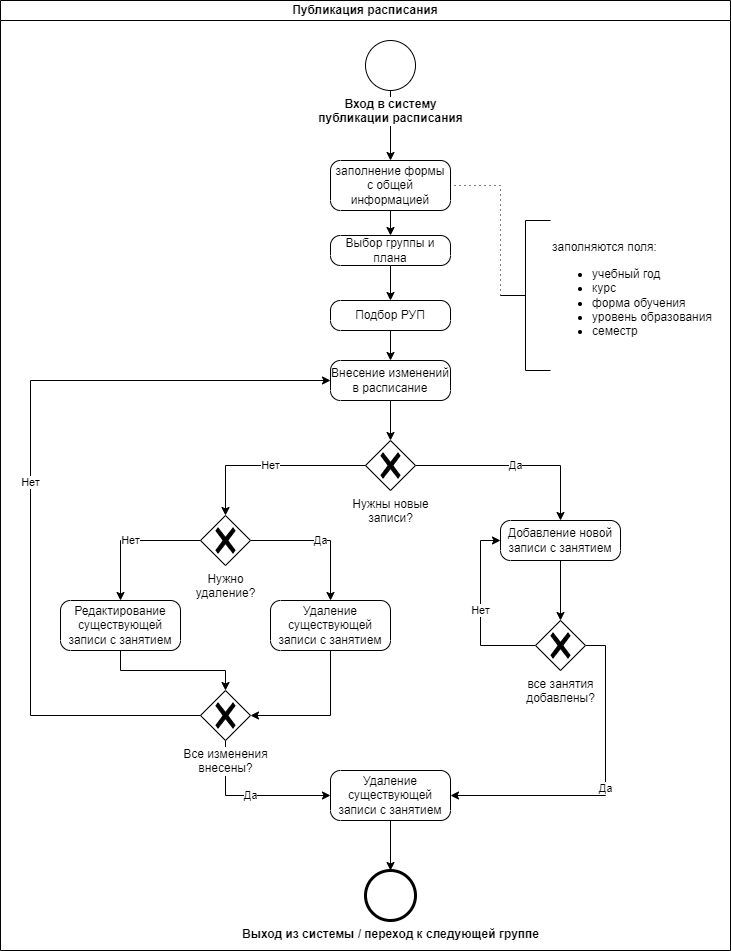


Рисунок 17 – Процесс публикации расписания

# 1.2 Описание API системы публикации расписания СВФУ

API (Application Programming Interface) – это программный интерфейс приложения. Представляет собой специальный протокол для взаимодействия программного обеспечения, позволяющий использовать функции одного приложения внутри другого. Механизмы, определяющие взаимодействие двух приложений, используя запросы и ответы (клиент-серверную архитектуру) [16, с. 52].

Существует несколько видов API:

• SOAP (Simple Object Access Protocol) API. Простой протокол доступа к объектам. Клиент и сервер обмениваются сообщениями посредством XML и HTML, что позволяет разработчикам не волноваться о языках и платформе [16, с. 52].

• RPC (Remote Procedure Call) API. Удаленный вызов процедур. Клиент выполняет функцию (или процедуру) на сервере, и сервер отправляет результат обратно клиенту в JSON- или XML-формате [16, с. 52].

• REST (Representational State Transfer) API. Передача состояния представления. Самый популярный подход на данный момент. Представляет данные как ресурсы и использует стандартные HTTP-методы для представления транзакций создания, чтения, обновления и удаления этих ресурсов, то есть стандартные CRUD операции [16, с. 52].

В ходе описания процесса публикации расписания мною также был осуществлен тщательный анализ API production-сервера СВФУ, в частности http-запросов, отправляющихся на него, и откликов в формате html-кода и установил, что на сервере используется REST API.

Взаимодействие с production-сервером проходит следующим образом.

Есть две конечные точки доступа к ресурсу:

1. <https://www.s-vfu.ru/user/rasp/new/> на этот маршрут отправляются запросы, предназначенные для записи данных в Базу;
2. <https://www.s-vfu.ru/user/rasp/new/ajax.php> на этот маршрут отправляются запросы для получения данных с сервера.

Данные, передаваемые в телах запросов, могут изменяться, однако во всех запросах неизменным остается присутствие параметра action: действие, которое надо выполнить. На сервере, куда генерируются запросы для изменения и публикации расписания, есть скрипт с функциями, которые обрабатывают запросы в зависимости от полученного action. Всего таких action на сервере 23 штуки, но самые основные это:

* loadgroup – получение списка групп, соответствующих заданным критериям;  
  Параметры запроса для такого действия:
  + id – идентификатор пользователя в системе BITRIX;
  + action – действие, которое необходимо совершить;
  + fac – факультет;
  + code – код уровня образования. Бакалавриат – 3, Магистратура – 4, Специалитет – 5, Аспирантура – 6;
  + course – курс;
  + form – форма обучения (очная, заочная, очно-заочная);
  + semestr – семестр;
  + year – год обучения.
* choicerup – получение РУПов по группам;

Параметры запроса для такого действия:

* + id – идентификатор пользователя в системе BITRIX;
  + action – действие, которое необходимо совершить;
  + fac – факультет;
  + course – курс;
  + form – форма обучения (1|очная, 2|заочная, 99|очно-заочная);
  + semestr – семестр;
  + year – год обучения;
  + groupname – наименование учебной группы, пример: (02.03.02|7471|ИМИ-Б-ФИИТ-21|5998).
* show – отображение списка занятий для выбранной группы и РУПа (если они ранее были добавлены);

Параметры запроса для такого действия:

* + action – действие, которое необходимо совершить;
  + fac – факультет;
  + code – код уровня образования. Бакалавриат – 3, Магистратура – 4, Специалитет – 5, Аспирантура – 6;
  + course – курс;
  + formshort – код формы обучения (1 – очная, 2 – заочная, 99 – очно-заочная);
  + formname – наименование формы обучения (очная, заочная, очно-заочная);
  + semestr – семестр;
  + year – год обучения;
  + id\_group – идентификатор группы, обрезанная часть наименования группы, полученного от сервера в предыдущем запросе;
  + filename – название файла с рабочим учебным планом;
  + startdate – дата начала обучения, по умолчанию 01.09 для осеннего семетра и 09.01 для весенного;
  + enddate – дата окончания обучения, по умолчанию 31.12 для осеннего семестра и 30.06 для весеннего.
* addrow – вызов формы для заполнения с полями занятия;

Параметры запроса для такого действия:

* + id – идентификатор действия, для добавления строки по умолчанию равняется единице;
  + full – строка, представляющая с собой конкатенацию нескольких параметров запроса. Состоит из факультета, названия файла с РУП, наименования группы, семестра, курса, года, глобального семестра (осенний, весенний), кода уровня обучения и формы обучения;
* insertrow – заполненная форма с занятием отправляется на сервер, данные записываются в Базу Данных, в таблице на главной странице появляется новая запись;

Параметры запроса для такого действия:

* + action – действие, которое необходимо совершить;
  + fac – факультет;
  + code – код уровня образования. Бакалавриат – 3, Магистратура – 4, Специалитет – 5, Аспирантура – 6;
  + course – курс;
  + formshort – код формы обучения (1 – очная, 2 – заочная, 99 – очно-заочная);
  + formname – наименование формы обучения (очная, заочная, очно-заочная);
  + semestr – семестр;
  + year – год обучения;
  + id\_group – идентификатор группы, обрезанная часть наименования группы, полученного от сервера в предыдущем запросе;
  + filename – название файла с рабочим учебным планом;
  + c – дата начала обучения по дисциплине, по умолчанию 01.09 для осеннего семетра и 09.01 для весенного;
  + d – дата окончания обучения по дисциплине, по умолчанию 31.12 для осеннего семестра и 30.06 для весеннего;
  + global\_semestr – глобальный семестр (осенний, весенний);
  + I – наименование дисциплины;
  + J – штатный преподаватель;
  + hours – почасовик;
  + podgruppa – количество подгрупп, на которое разбито занятие;
  + В – день недели;
  + F – время занятия;
  + chet – четность (0 – каждую неделю, 1 – четная, 2 – нечетная);
  + H – вид учебной деятельности (СРС, практика, лекция и т.д.)
  + L – корпус, где будет занятие;
  + K – аудитория, где будет занятие.
* edit – отправляется на второй конечный ресурс, появляется форма с полями, которые можно отредактировать и кнопкой подтверждения/отмены внесения изменений для выбранной строки.

Параметры для этого запроса:

* + id – идентификатор действия, по умолчанию 1;
  + data – идентификатор занятия в базе данных с сервера;
  + full – общая строка со всеми параметрами;
  + fac – факультет.
  + action – действие, которое необходимо совершить
* updaterasp – при отправке на первый маршрут к ресурсу запрос идет на сервер и в соответствующую строку вносятся изменения в Базе Данных.

Параметры для этого запроса:

* + data – общая строка, объединяющая все параметры;
  + id\_group – идентификатор группы, обрезанная часть наименования группы, полученного от сервера в предыдущем запросе;
  + filename – название файла с рабочим учебным планом;
  + global\_semestr – глобальный семестр (осенний, весенний);
  + semestr – семестр;
  + course – курс;
  + fac – факультет;
  + year – год обучения;
  + formshort – код формы обучения (1 – очная, 2 – заочная, 99 – очно-заочная);
  + action – действие, которое необходимо совершить;
  + block – столбец таблицы с расписанием;
  + id – идентификатор строчки с занятием;
  + I – наименование дисциплины;
  + H – вид учебной деятельности (СРС, практика, лекция и т.д.)
  + J – штатный преподаватель;
  + hours – почасовик.
* delete – если отправлять на второй конечный ресурс, то появляется форма с подтверждением удаления выбранной строки; при отправке на первый маршрут к ресурсу запрос идет на сервер и соответствующая строка удаляется из Базы Данных; запись также пропадает из таблицы на главной странице;

Параметры для этого запроса при отправке на первый конечный ресурс:

* + id – идентификатор действия, по умолчанию 1;
  + data – идентификатор занятия в базе данных с сервера;
  + full – общая строка со всеми параметрами;
  + fac – факультет.
  + action – действие, которое необходимо совершить;

Параметры для этого запроса при отправке на второй конечный ресурс:

* + id – идентификатор строки в таблице;
  + full – общая строка со всеми параметрами;
  + fac – факультет.
  + action – действие, которое необходимо совершить;
* public – если отправлять на второй ресурс, запускаются процесс валидации данных и открывается модальное окно с подтверждением публикации расписания; при отправке на первый маршрут к ресурсу данные с формы отправляются на сервер, расписание публикуется на сайте СВФУ.

Параметры для такого запроса при отправке на первый конечный ресурс:

* + data – общая строка, объединяющая все параметры;
  + id\_group – идентификатор группы, обрезанная часть наименования группы, полученного от сервера в предыдущем запросе;
  + filename – название файла с рабочим учебным планом;
  + global\_semestr – глобальный семестр (осенний, весенний);
  + formshort – код формы обучения (1 – очная, 2 – заочная, 99 – очно-заочная);
  + semestr – семестр;
  + course – курс;
  + fac – факультет;
  + year – год обучения;
  + action – действие, которое необходимо совершить.

Параметры для такого запроса при отправке на первый конечный ресурс:

* + id – идентификатор действия, по умолчанию 1;
  + action – действие, которое необходимо совершить;
  + full – общая строка со всеми параметрами;
  + fac – факультет.

Процесс публикации расписания с добавлением, редактированием и удалением строк с занятиями изображен на диаграмме последовательности в приложении А.

# 1.3 Обзор аналогов

В данном разделе будет осуществлен анализ и обзор имеющихся аналогов предлагаемого мною ПО.

Так как разрабатываемое мною приложение будет иметь уникальный, специфичный функционал, как таковых эквивалентных решений у него не имеется. Поэтому в качестве аналогов были выбраны немного отличающиеся, но все еще являющиеся программами для автоматизации составления расписания, продукты.

Все приведенные ниже программы являются полной заменой уже использующейся в СВФУ системе публикации расписания, в то время как мое решение призвано безболезненно интегрироваться в эту систему.

1С: Автоматизированное составление расписания. Университет

1С: Автоматизированное составление расписания – это программный продукт, разработанный компанией 1С, который предназначен для автоматизации процесса составления расписания в университетах и других образовательных учреждениях. Этот продукт позволяет упростить и оптимизировать работу планировщиков, преподавателей и студентов, связанную с созданием и использованием расписания занятий.

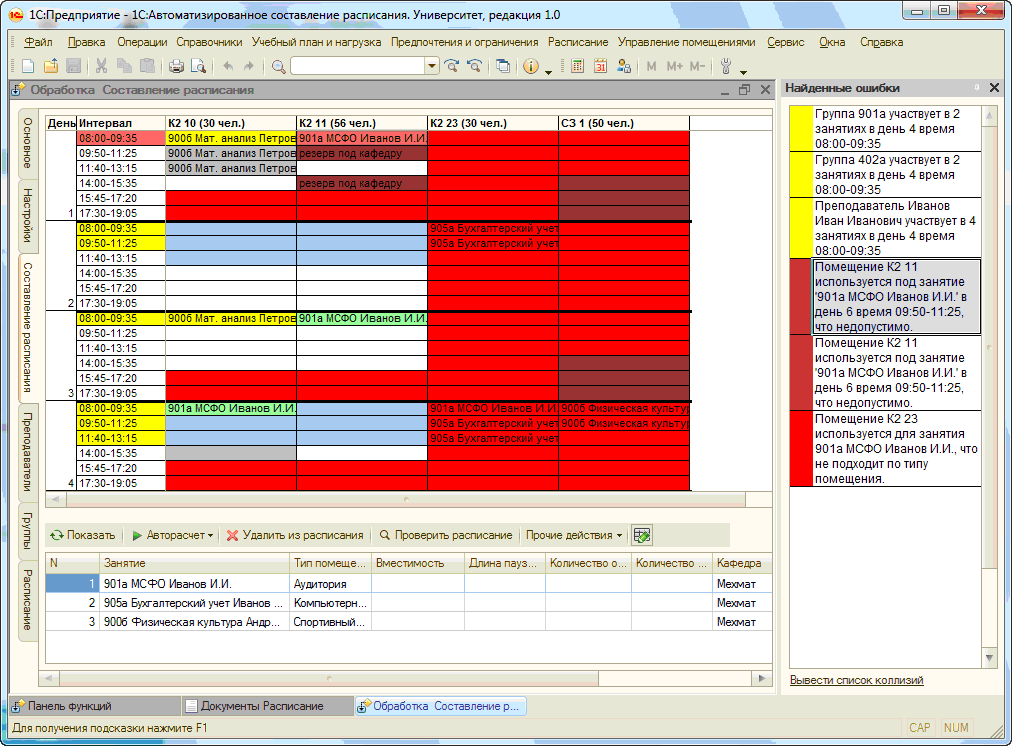


Рисунок 18 – Интерфейс ПО 1С: Автоматизированное составления расписания

Основные возможности продукта включают:

* Гибкое планирование расписания: Программа позволяет оптимизировать распределение учебных занятий, учитывая различные ограничения, такие как доступность преподавателей и аудиторий, предпочтения студентов и другие параметры. Планирование может быть выполнено автоматически или с помощью интерактивного режима.
* Учет учебных программ и нагрузки: Продукт позволяет вести учет учебных программ и нагрузки на преподавателей, а также учитывать специфические требования к расписанию для разных курсов и групп студентов. Это помогает предотвратить конфликты и обеспечить балансировку нагрузки между преподавателями.
* Интеграция с другими системами: 1С: Автоматизированное составление расписания может быть интегрировано с другими информационными системами в университете, такими как системы учета студентов и преподавателей. Это позволяет автоматически получать необходимую информацию для составления расписания и обмениваться данными с другими системами.
* Веб-интерфейс и мобильное приложение: Пользователи могут получить доступ к расписанию через веб-интерфейс или мобильное приложение. Это позволяет просматривать расписание, получать уведомления о изменениях, записываться на занятия и выполнять другие операции, связанные с расписанием.

Продукт 1С: Автоматизированное составление расписания значительно упрощает и оптимизирует процесс планирования расписания в университетах, позволяя сократить время, затрачиваемое на его создание и обеспечивая более эффективное использование ресурсов.

Стоимость данного решения составляет от 105 до 110 тысяч рублей в зависимости от версии продукта.

AscTimeTables

ASC Timetables – это программное обеспечение, разработанное для автоматизации составления расписания в учебных заведениях, таких как школы, колледжи и университеты. Это мощный инструмент, который помогает упростить и оптимизировать процесс планирования занятий.

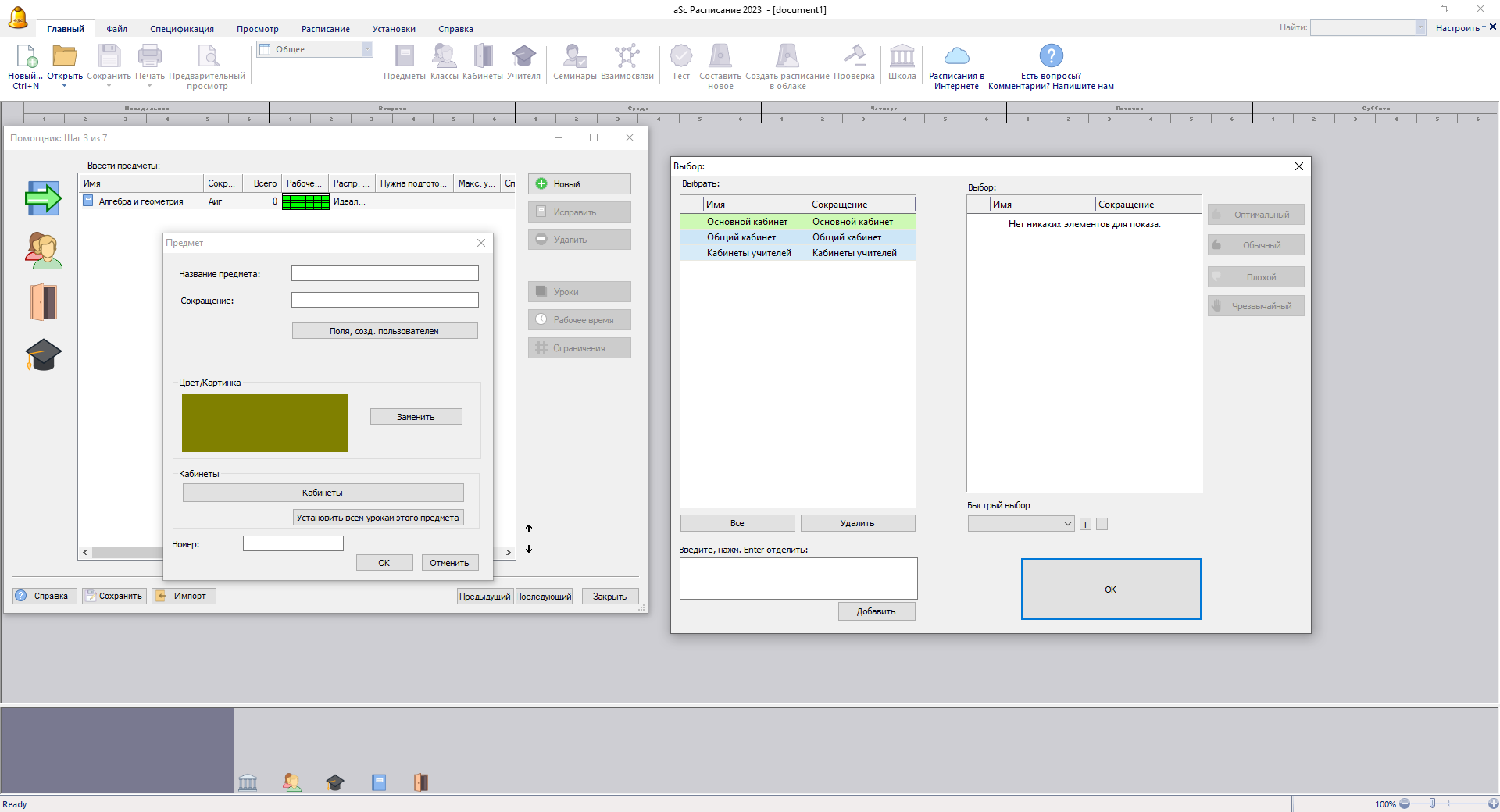


Рисунок 19 – Интерфейс ПО ASC Timetables

Программа предлагает следующие возможности:

1. Интуитивно понятный интерфейс: Программа имеет простой и понятный интерфейс, что облегчает ее использование. Пользователи могут легко создавать и редактировать расписание занятий с помощью графического интерфейса.

2. Гибкое планирование: ASC Timetables позволяет учитывать различные ограничения и предпочтения при составлении расписания. Это включает доступность учителей и аудиторий, предпочтения по времени занятий, требования к нагрузке и другие факторы. Программа автоматически учитывает эти ограничения и генерирует оптимальное расписание.

3. Автоматическое создание расписания: ASC Timetables может автоматически составить расписание занятий на основе заданных параметров. Она использует алгоритмы оптимизации для максимально эффективного использования ресурсов и предотвращения конфликтов.

4. Предпросмотр и редактирование: Пользователи могут предварительно просматривать и редактировать созданное расписание. Это позволяет вносить корректировки, если требуется, и убедиться в его правильности перед окончательным утверждением.

5. Генерация отчетов и экспорт: ASC Timetables предлагает возможность генерировать различные отчеты, такие как расписание для учителей и студентов, сводные таблицы, графики и другие. Также можно экспортировать расписание в различные форматы, такие как Excel, PDF и HTML.

ASC Timetables является полезным инструментом для учебных заведений, позволяющим сэкономить время и упростить процесс составления расписания занятий. Он обеспечивает более эффективное использование ресурсов, предотвращает перекрестные конфликты и помогает всем участникам образовательного процесса легко ориентироваться в расписании.

Стандартная, премиум и профессиональная версии данной программы стоят 149, 499, 1995 евро соответственно.

Основными недостатками приведенных выше готовых решений, на мой взгляд, являются отсутствие возможности интеграции с уже имеющейся в СВФУ системой публикации расписания и высокая стоимость.

Нам необходимо решение, которое будет внедряться в процесс публикации расписания, автоматизируя его и при это не нарушая работу остальных уже настроенных в системе процессов.

# 1.4 Обзор инструментов разработки

В современном мире существует огромное количество различных языков программирования (ЯП), заточенных под определенные парадигмы разработки.

Множество из них подходит для разработки веб-приложения. В качестве основных мной были выбраны C# и Python. Для окончательного выбора мною был проведен сравнительный анализ этих двух языков программирования, где критериями выступили функциональные требования к моему ПО автоматизации расписания.

1. Синтаксис и простота использования:
   1. Python известен своим простым и понятным синтаксисом, который позволяет разработчикам писать код лаконично и читабельно. Он обладает большим количеством встроенных функций и библиотек, что упрощает разработку.
   2. C# имеет более формальный синтаксис, но он все равно достаточно легок для изучения. Он обладает мощной системой типов и широкими возможностями для разработки веб-приложений.
2. Обработка данных и парсинг:
   1. Python имеет богатый набор инструментов для обработки данных и парсинга файлов различных форматов, включая Excel и HTML. Библиотеки, такие как openpyxl, pandas, Beautiful Soup и lxml, делают процесс парсинга относительно простым и эффективным.
   2. C# также обладает мощными инструментами для обработки данных и парсинга файлов. Для работы с Excel-файлами можно использовать библиотеку EPPlus, а для парсинга HTML-кода - HtmlAgilityPack.
3. Работа с сессиями, авторизацией и генерацией запросов:
   1. В Python существует множество библиотек, таких как Requests, Flask и Django, которые обеспечивают поддержку сессий, авторизации и генерации запросов. Они предоставляют простые и интуитивно понятные API для работы с веб-серверами и клиентами.
   2. C# также имеет солидные фреймворки, такие как ASP.NET и HttpClient, которые обеспечивают поддержку сессий, авторизации и генерации запросов. Они предоставляют широкий набор инструментов для разработки веб-приложений с поддержкой веб-серверов и клиентов.
4. Экосистема и поддержка:
   1. Python имеет активное сообщество разработчиков и богатую экосистему сторонних библиотек и фреймворков. Это обеспечивает доступ к большому количеству ресурсов, документации и примеров кода, а также поддержку со стороны сообщества.
   2. C# также имеет большую и активную экосистему, особенно в контексте платформы .NET. Есть множество библиотек, фреймворков и инструментов, которые упрощают разработку веб-приложений. Существует также широкая поддержка со стороны Microsoft и сообщества разработчиков.

В конечном итоге мой выбор в пользу Python обусловлен его простотой использования, мощными инструментами для обработки данных и парсинга excel-файлов и html-разметки, наличием большого разнообразия различных библиотек, покрывающих практически все нужны программиста, и активного сообщества разработчиков, что способствует быстрому решению возникающих проблем и ошибок.

После выбора языка программирования, мне необходимо было определиться с библиотеками для разработки функционала, отвечающего требованиям.

Программа по автоматизации процедуры публикации расписания должна работать с HTTP-запросами, поддерживать сессии и авторизацию, парсить html-код откликов на запросы.

Для разработки подобного функционала в Python есть три библиотеки: сторонняя Requests и модули, входящие в стандартную библиотеку языка, urllib и http\_client.

Для окончательного выбора был осуществлен сравнительный анализ этих трех библиотек по следующим критериям:

1. Уровень абстракции:
   1. Requests предоставляет более высокий уровень абстракции и обладает простым и удобным API. Он позволяет выполнять HTTP-запросы и работать с сессиями, авторизацией и заголовками с минимальным количеством кода.
   2. urllib и http.client предоставляют более низкий уровень абстракции и требуют немного больше кода для выполнения HTTP-запросов. Они предоставляют более гибкий контроль над запросами и позволяют более точно настраивать различные параметры.
2. Простота использования:
   1. Requests известен своим простым и интуитивно понятным API, что делает его легким для изучения и использования даже для новичков в программировании.
   2. urllib и http.client требуют более детального понимания работы с HTTP-протоколом и имеют немного более сложный синтаксис. Они могут потребовать больше усилий для использования в сравнении с Requests.
3. Наличие функций и поддержка:
   1. Requests предоставляет обширный набор функций, включая поддержку сессий, авторизации, заголовков, параметров запросов и т. д. Он также имеет большое активное сообщество разработчиков и обновляется регулярно, обеспечивая надежную поддержку и решение проблем.
   2. urllib и http.client также предоставляют базовые функции для выполнения HTTP-запросов и обработки ответов. Однако они могут потребовать больше усилий для настройки и реализации дополнительных функций, таких как сессии и авторизация.
4. Зависимости:
   1. Requests является отдельной сторонней библиотекой, и для ее использования потребуется установка отдельного пакета. Однако это стандартная и широко используемая библиотека, и ее установка и использование обычно не вызывают проблем.
   2. urllib и http.client являются частью стандартной библиотеки Python, что означает, что они уже встроены в язык и не требуют дополнительной установки.

В итоге, было решено использовать Requests как наиболее популярный и широко используемый вариантом среди разработчиков Python для работы с HTTP-запросами за счет высокого уровня абстракции, поддержки авторизации и удобной работы с сессиями и запросами.

Помимо вышеупомянутых инструментов, мне также понадобится фреймворк для написания каркаса моего веб-приложения.

Flask и Django – два популярных фреймворка для разработки веб-приложений на языке Python. Мною был выбран Flask по следующим причинам:

1. Гибкость: Flask является микрофреймворком, что означает, что он предоставляет только основные функции и минимальные зависимости. Это делает его очень гибким и позволяет разработчику выбирать и интегрировать дополнительные библиотеки и инструменты по своему усмотрению. Вы можете создавать приложения с минимальным набором функций или масштабировать их до сложных проектов.
2. Простота и прозрачность: Flask имеет простую и интуитивно понятную структуру, что делает его легким для изучения и использования. Он предоставляет минималистичный подход к разработке, который помогает сосредоточиться на самом приложении, а не на сложных фреймворковых концепциях. Flask также обладает небольшим объемом кода, что упрощает его понимание и настройку.
3. Масштабируемость и производительность: Благодаря своей гибкости, Flask позволяет эффективно создавать и масштабировать приложения. Он не навязывает строгую структуру проекта и позволяет выбирать необходимые компоненты для каждой задачи. Это позволяет достичь лучшей производительности и эффективности в случаях, когда требуется особая оптимизация и настройка.
4. Низкий порог вхождения: Flask предоставляет простую и легкую в использовании документацию, а также обладает большим коммьюнити и активным сообществом разработчиков. Это значительно упрощает процесс изучения и позволяет быстро находить ответы на вопросы и проблемы.
5. Расширяемость: Flask предлагает широкий спектр расширений и пакетов, которые облегчают разработку различных функций и интеграцию с другими инструментами. Вы можете выбирать и настраивать только необходимые расширения, что помогает сделать приложение более легковесным и оптимизированным.

В то время как Django предлагает более полнофункциональный и мощный набор инструментов для разработки веб-приложений, Flask отличается своей гибкостью, простотой использования и возможностью создания более легких и гибких приложений.

Также буду использовать язык гипертекстовой разметки веб-страниц HTML, CSS (формальный язык описания внешнего вида документа, написанного с использованием языка разметки) и шаблонизатор Jinja2 для их оформления.

Последний инструмент для начала работы – среда разработки, которая должна поддерживать работу со всеми вышеупомянутыми инструментами. Для сравнительного анализа были выбраны VS Code (Visual Studio Code) и PyCharm.

1. Легковесность и производительность: VS Code является легковесным редактором кода с меньшим потреблением ресурсов, что делает его быстрее в работе и более отзывчивым. Он запускается быстрее и обладает более низким потреблением памяти по сравнению с PyCharm.
2. Расширяемость: VS Code обладает огромным количеством расширений и плагинов, которые позволяют настроить его под конкретные потребности разработчика. Это означает, что вы можете добавить только необходимые инструменты и функциональность, что помогает сделать среду разработки более удобной и эффективной.
3. Поддержка других языков и платформ: VS Code является универсальным редактором кода и поддерживает большое количество языков программирования и платформ. Это означает, что вы можете использовать его не только для разработки на Python, но и для других языков и технологий, что делает его более гибким инструментом для разработчиков, работающих в разных областях.
4. Интеграция с Git и другими инструментами: VS Code обладает отличной интеграцией с системой контроля версий Git и другими инструментами разработки. Он предлагает удобные функции для коммитов, сравнения изменений, визуализации веток и других задач, связанных с управлением кодом.
5. Быстрый старт и простота использования: VS Code имеет интуитивный интерфейс, и его настройка и использование происходят быстро и легко. Он имеет простую установку и настраивается по мере необходимости, что особенно полезно для новичков в программировании или для разработчиков, которым нужна быстрая и простая среда разработки.

В то время как PyCharm является мощным и полнофункциональным IDE, VS Code предлагает легковесную и гибкую альтернативу с большим количеством расширений и простотой использования. Именно поэтому мой выбор пал на Visual Studio Code.

# Выводы по главе 1

В первой главе мной была проделана следующая работа%

* определены основные теоретические характеристики исследования:
  + выявлена проблема автоматизации процесса публикации расписания, заключающаяся в ее отсутствии;
  + определена актуальность;
  + поставлена цель по автоматизации процедуры публикации расписания;
  + сформированы задачи для достижения поставленной цели;
* произведен обзор имеющихся аналогов;
* проанализирована работа в системе публикации расписания и ее API;
* сформирован список инструментов разработки для решения поставленных задач.

В результате проведенного анализа было принято решение о необходимости разработки собственного решения по автоматизации процедуры публикации расписания.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕДУРЫ ПУБЛИКАЦИИ РАСПИСАНИЯ СВФУ

# 2.1 Требования к ПО

Разрабатываемая программа предназначена для автоматизации процесса публикации учебного расписания. Основная цель программы – упростить сложный трудоемкий процесс публикации расписания ИМИ СВФУ.

Для формирования потребностей к функционалу приложения мною был осуществлен сбор требований.

Во-первых, было проведено интервью – традиционный источник требований как для серийных продуктов, так и информационных систем в любых методиках разработки ПО [1, с. 51]. В качестве опрашиваемого источника требований была выбрана Заместитель директора по учебной работе Института математики и информатики Романова Наталья Анатольевна. Именно она занимается составлением и публикацией расписания. Основным требованием Натальи Анатольевны было избавление необходимости вводить каждое занятие по каждой группе каждого курса по отдельности, т.е. добавление возможности публиковать все новые занятия разом.

Во-вторых, был проведен анализ данных. Анализировались существующий процесс публикации расписания и запросы, которые генерировались при каждом действии в системе, на сервер. В конечном итоге благодаря анализу стало понятно, что система требует автоматизации, а также было определенно, каким именно образом этой автоматизации можно добиться [1, с. 51].

В результате проведенных интервью и анализа данных были сформированы основные функциональные требования к ПО:

* Вход в систему под учетной записью СВФУ: приложение должно использовать уже имеющиеся у пользователя логин и пароль от личного кабинета сайта СВФУ;
* Форма для выбора и отправки расписания в формате excel-документа: пользователю необходим интерфейс с возможностью выбора и отправки расписания в формате excel-файла;
* Парсинг excel-документа: приложение должно извлекать из полученного excel-файла всевозможные данные и формировать структуру данных, соответствующую спроектированной модели данных;
* Парсинг откликов сервера в виде html-кода: приложение должно уметь извлекать необходимые данные из полученных от production-сервера ответов;
* Логирование ошибок и исключений: в случае возникновения каких-либо ошибок, приложение не должно прекращать работу. Все ошибки должны попадать в исключения и записываться в текстовый файл для отладки;
* Информирование пользователя о результатах публикации расписания: в конечном итоге пользователь должен видеть результат своей работы.

Требования к формату excel-файла

Для своей работы нами был взят за основу файл с расписанием последней версии на сайте СВФУ.

Если строго придерживаться перечисленных ниже требований, то приложением сможет пользоваться не только Институт математики и информатики, но и остальные учебные подразделения.

Основные требования:

* Главное требование: т.к. за основу был взят excel-файла с расписанием занятий БАК и МАГ («ИМИ Расписание учебных занятий на 2 полугодие 2022-2023», от 28.02.2023), то необходимо строго придерживаться структуры данного файла:
  + названия листов – номер курса с указанием кафедры;
  + номер курса должен располагаться 2-ой строкой;
  + наименование группы должно начинаться с 4-ой строки;
  + день недели и время располагаются в 1 и 2 столбцах соответственно;
  + дисциплина, ФИО преподавателя, вид учебной деятельности и аудитория должны быть в каждом третьем, четвертом, пятом и шестом столбцах соответственно;
  + в каждом дне недели должно быть 6 пар (6 строк);
  + в каждом пятом и шестом столбцах четвертой строки должны храниться даты начала и окончания обучения для группы в формате «дд.мм»;
  + ФИО преподавателей пишется в формате «Фамилия И.О.»;
* День недели: объединенная ячейка на 6 строк (6 пар);
* Курс и полугодие должны быть арабскими цифрами;
* В качестве разделителя между несколькими дисциплинами в одной ячейке должен выступать перенос строки, то есть специальный символ “\n”. Перенос строки достигается комбинацией клавиш Alt + Enter;

# 2.2 Модель классов расписания

После анализа расписания ИМИ СВФУ во всех существующих форматах, мною была разработана модель классов расписания.

Основным классом здесь является Class, то есть занятие. Он является собирательным для всех остальных за исключением класса Department, который выступает в роли контейнера для всего. Он будет содержать в себе массивы Class, Group, Lecturer, Subject и пользователя User.

Класс Place содержит в себе поля корпуса и аудитории, класс Time – время начала и время окончания занятия (так сделано с целью предотвратить ошибки, если в дальнейшем СВФУ поменяет время пар), класс User – логин и пароль для авторизации в системе, класс Group – наименование, уровень образования и курс, класс Lecturer – имя преподавателя и поле для обозначения преподавателей, не входящих в штат, класс Subject – поле с одним только названия предмета.

Такие сущности, как «Уровень образования», «День недели», «Четность», «тип учебной активности» и «подгруппа» были представлены в виде перечисления – специального типа данных, позволяющих переменной объекту или переменной быть набором уже заранее определенных констант, то есть принимать только те значения, которые были объявлены в самом перечислении. Это является более рациональным и безопасным подходом, когда речь идет о фиксированном предопределенном наборе чего-либо.

На рисунке 20 представлена модель классов расписания.

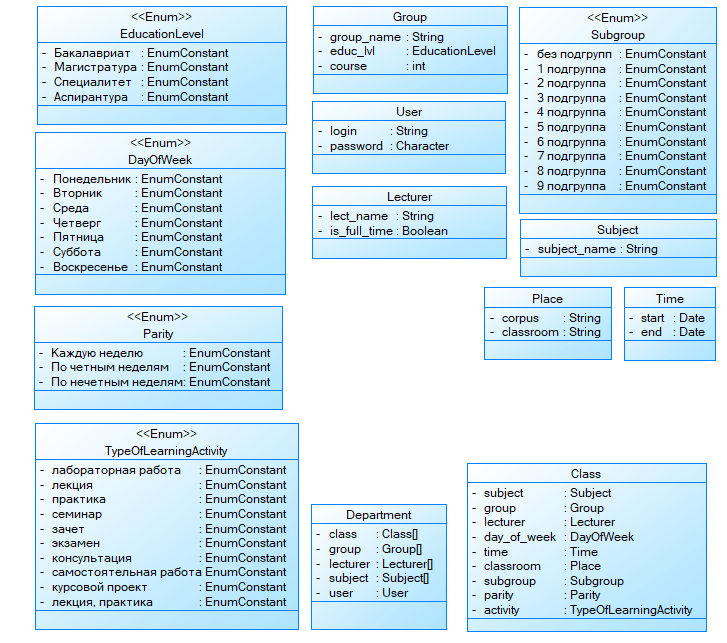


Рисунок 20 — Модель классов расписания

# 2.3 Модель взаимодействия

Use Case Diagram (диаграмма прецедентов, она же диаграмма вариантов использования) – диаграмма, отражающая отношения между акторами (участниками) и прецедентами (вариантами) [12, с. 52].

Участник – это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями.

Прецедент (use case) – описание множества последовательных событий, приводящих к наблюдаемому участником результату.

В качестве участника в моем случае выступает только Пользователь, больше действующих лиц нет.

Прецедентов же несколько:

1. Авторизация. Пользователь переходит к форме авторизации с главного окна, после успешной авторизации его перенаправляет к форме загрузки документа с расписанием и его отправки;
2. Выбор документа с расписанием. Пользователю предоставляется форма выбора и загрузки файлов.
3. Отправка POST-запроса с документом. После того, как пользователь прикрепил нужный ему файл, по нажатию соответствующей кнопки отправляется запускается парсинг документа и генерация запросов на сервер.
4. Отображение расписания. Результат загрузки расписания можно увидеть на сайте СВФУ.
5. Просмотр файлов с логами.

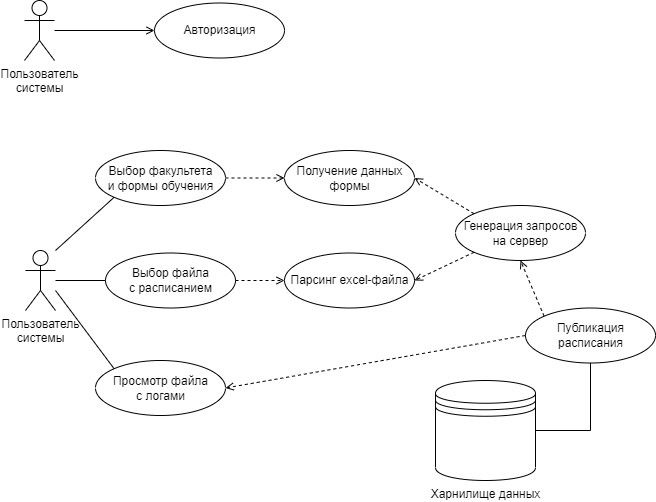


Рисунок 21 – Диаграмма прецедентов для авторизации

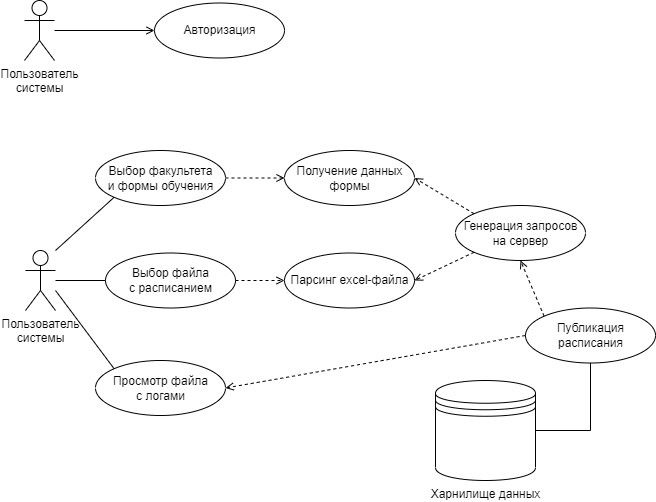


Рисунок 22 – Диаграмма прецедентов процесса публикации расписания

# 2.4 Архитектура ПО

Архитектура программного обеспечения – совокупность важнейших решений об организации программной системы. Включает в себя выбор структурных элементов, их объединение в более крупные системы и определенный архитектурный стиль. Иными словами, архитектура ПО отображает организацию и структуру системы, обосновывая ее поведение [9, с. 52].

От того, насколько хорошо прописана и разработана архитектура системы, напрямую зависит качество, производительность, простота и эффективность разработки и дальнейшего сопровождения ПО.

Мое веб-приложение по автоматизации процедур публикации расписания будет основано на клиент-серверной архитектуре [6, с. 51][8, с. 51], то есть состоять из клиентской и серверной частей. Клиентская сторона, в данном случае, будет представлять собой интерфейс веб-приложения, в котором будет возможность авторизоваться, указать учебный год и факультет, а также прикрепить excel-файл с расписанием.

Серверная часть будет обрабатывать документ с расписанием, генерировать на основе данных, полученных из файла, запросы на production-сервер СВФУ и обрабатывать полученные от него ответы.

Архитектурное решение для моего приложения, составленное с помощью языка моделирования архитектур Archimate 3.0 [17, с. 52], изображено на рисунке 23.

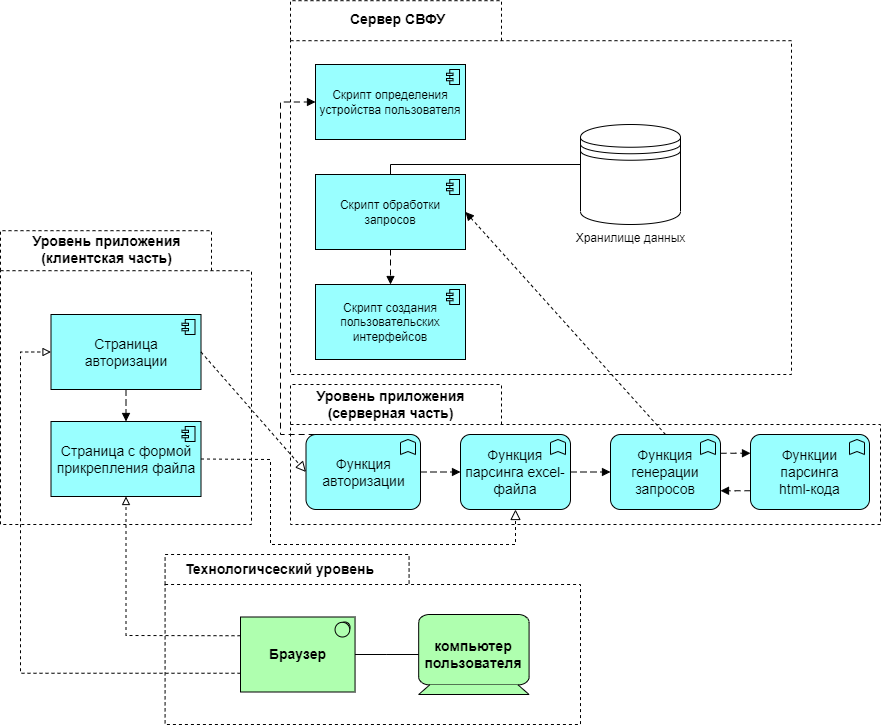


Рисунок 23 – Архитектура решения

# 2.5 Описание тестового сервера

Прежде чем работать с production-сервером СВФУ, необходимо было провести тестирование приложения на тестовом контуре. Для этого нами был разработан тестовый сервер (заглушка), призванный имитировать работу «боевого» сервера.

Он содержит в себе структуры данных, похожие на то, что в ответе на запросы возвращает сервер СВФУ.

Работает практически также, как и production-сервера: в зависимости от полученного action возвращает соответствующие данные.

Сервер также был разработан с помощью фреймворка Flask и библиотеки Requests.

Функций на сервере столько же, сколько и на production-сервере.

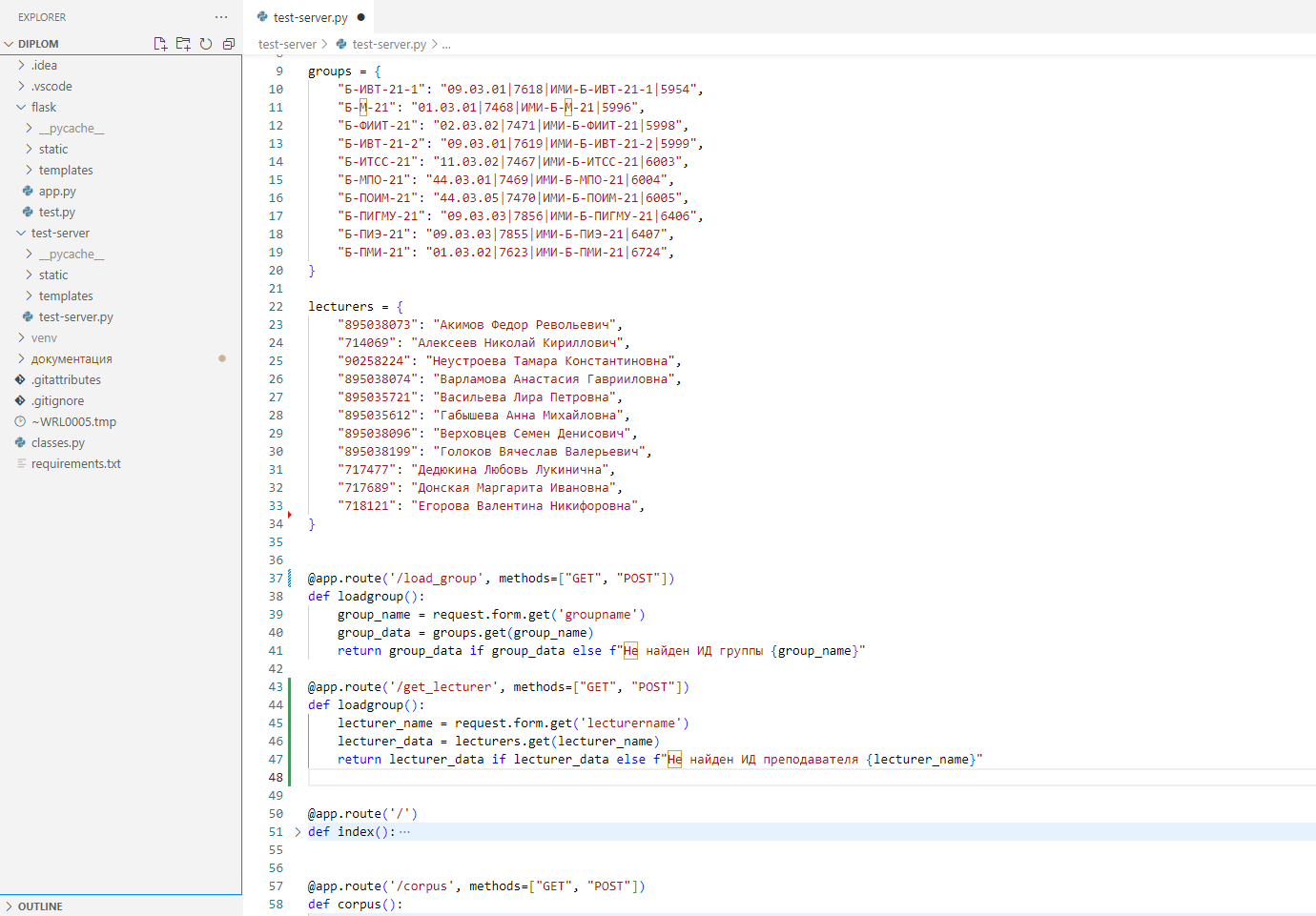


Рисунок 24 – Исходный код тестового сервера

# 2.6 Описание приложения

Разработанное мною веб-приложение написано на микрофреймворке Flask. Для парсинга Excel использую библиотеки OpenPyXL, т.к. она поддерживает разные форматы файлов, BeautifulSoup4 для парсинга html-кода, который приходит мне в отклике на запросы, библиотеку Requests для работы с http-запросами и сессиями.

Репозиторий проекта доступен по адресу: <https://github.com/abbadon321/diplom>.

Само приложение опубликовано по адресу: <https://contest.s-vfu.ru/rasp/>.

Структурно проект состоит из двух основных каталогов: в первом содержатся 2 папки с шаблонами и статическими файлами и само приложение. Второй каталог отвечает за тестовый веб-сервер.

Каркас веб-приложения состоит из трех маршрутов:

1. @app.route('/') / @app.route('/auth') – маршрут со стартовой страницей, содержащей форму с логином и паролем для входа в систему (рисунок 25).

Для данного маршрута написана функция def index(), которая в ответе возвращает шаблон с html-кодом страницы авторизации. Запрос с этой страницы идет на следующий маршрут. В случае ошибки, пользователь остается на текущей странице.

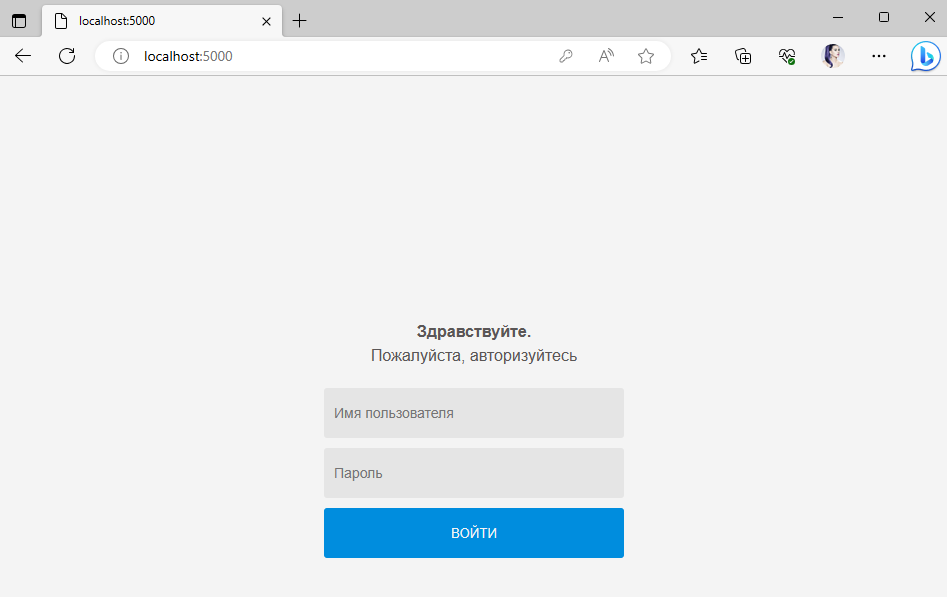


Рисунок 25 – Страница авторизации

1. @app.route('/main', methods=['post', 'get']) – маршрут с основной страницей, где расположена форма с выпадающими списками для выбора факультета, формы обучения и полем для прикрепления excel-файла (рисунок 26).

Для этого маршрута написана функция def authorize(), которая получает от прошлого маршрута данные формы, запускает сессию и отправляет запрос на сайт СВФУ для авторизации в системе. В случае успешной авторизации форма для прикрепления файла становится доступной для заполнения. Заполненная форма передается на следующий запрос.

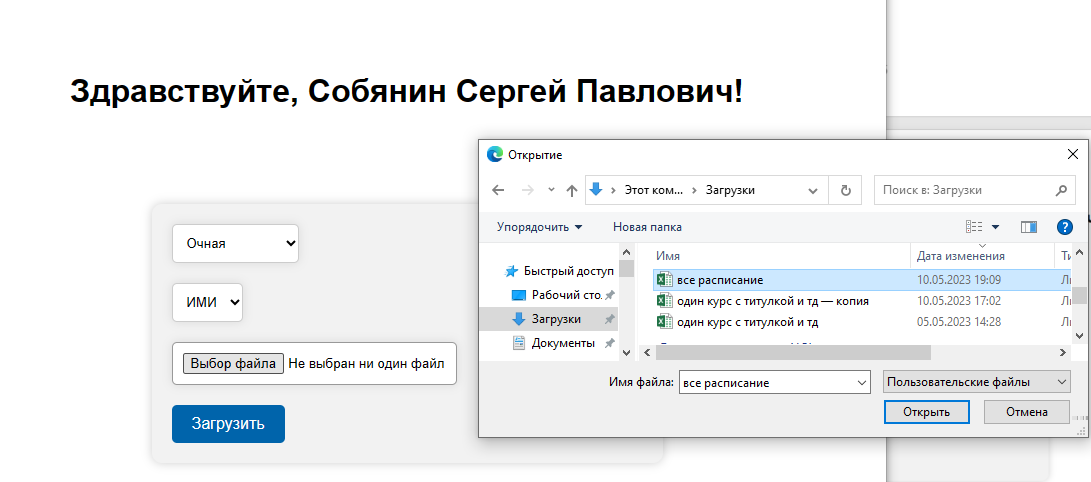


Рисунок 26 – Форма выбора и отправки файла

1. @app.route("/schedule", methods=['GET', 'POST']) – маршрут с освноной логикой приложения. Для этого маршрута написано большое количество функций.
   1. def schedule\_parse() – функция парсинга excel-файла. Основной алгоритм таков: программа получает файл, из него записывает все листы книги, получает информацию о семестре и текущем учебном годе, затем запускается цикл по этим листам. Сначала определяется курс и уровень обучения, далее идет поиск групп, как только группа найдена, сначала удаляются все ее старые занятия из Базы Данных, если они есть, затем начинается цикл по занятиям этой группы. В цикле вытаскиваются все пары, если они есть, далее идет добавление этого занятия в таблицу. Как только все занятия добавлены в таблицу, расписание для этой группы публикуется, цикл переходит на следующую группу.
   2. def get\_year\_and\_semestr(string) – группа для получения учебного года и семестра из файла с расписанием; на вход подается значение из определенной ячейки документа.
   3. def get\_parity(lesson) – получение четности занятия, на вход подается значение из ячейки с дисциплиной;
   4. def get\_code(group\_name) – получения кода формы обучения, на вход подается название группы;
   5. def get\_activity(act) – получение вид учебной активности, на вход подается значение из ячейки с учебной активностью;
   6. def extract\_corpus(string) – получение корпуса, на вход подается значение из ячейки с аудиторией;
   7. def query(full=None, id=None, action=None, fac=None, code=None, course=None, form="", semestr=None, year=None, filename=None, id\_group=None, groupname=None, chet="", weekday=None, activity=None, corpus=None, classroom=None, lesson=None, lecturer=None, time=None, full\_semestr=None, startdate="", enddate="") – основная функция. В зависимости от полученного action определяется конечный маршрут к ресурсу для отправки данных и сами данные. На вход подаются все данные, полученные из документа с расписанием.
   8. def parse\_loadgroup(html, groupname) – парсинг отклика на запрос с action – loadgroup, на вход подается html-код соответствующего отклик.
   9. def parse\_choicerup(html) – парсинг отклика на запрос с action – choicerup, на вход подается html-код соответствующего отклик.
   10. def parse\_addrow(html, lecturer) – парсинг отклика на запрос с action – addrow, на вход подается html-код соответствующего отклик.

Дополнительно было добавлено логирование с обработчиками исключений для авторизации, чтобы знать, какой пользователь вошел в систему и выполнял действия; для каждого вызова функций и получения данных из документа с расписанием, чтобы на production-сервер не отправлялись неправильные или незаполненные запросы.

Ход выполнения программы устроен таким образом, что в случае каких-либо ошибок, итерация прерывается и цикл переходит на следующую с записыванием информации о причинах в текстовый файл с логами.

# Выводы по главе 2

Во второй главе моей выпускной квалификационной работы были описаны проектирование и разработка приложения, а также его работа после реализации и результаты апробации.

В качестве результатов проектирования были предоставлены функциональные требования к ПО, требования к формату документа с расписанием, диаграмма классов, диаграмма прецедентов и схема архитектурного решения. Был также спроектирован тестовый сервер для тестирования разработанного программного обеспечения.

В конечном итоге мне удалось реализовать свое решение в виде приложения по автоматизации процедуры публикации расписания.

# ЗАКЛЮЧЕНИЕ

Таким образом, в ходе выпускной квалификационной работы были проанализированы составление и публикация расписания учебных занятий ИМИ СВФУ. На основе проведенного анализа стала понятно, что существующая система публикации расписания требует автоматизации для упрощения самого процесса публикации.

Был осуществлен анализ расписания ИМИ и системы публикации расписания СВФУ и проведен обзор библиотек и инструментов разработки в виде сравнительного анализа.

В результате проведенной работы было спроектировано и разработано веб-приложение по автоматизации процедуры публикации расписания.

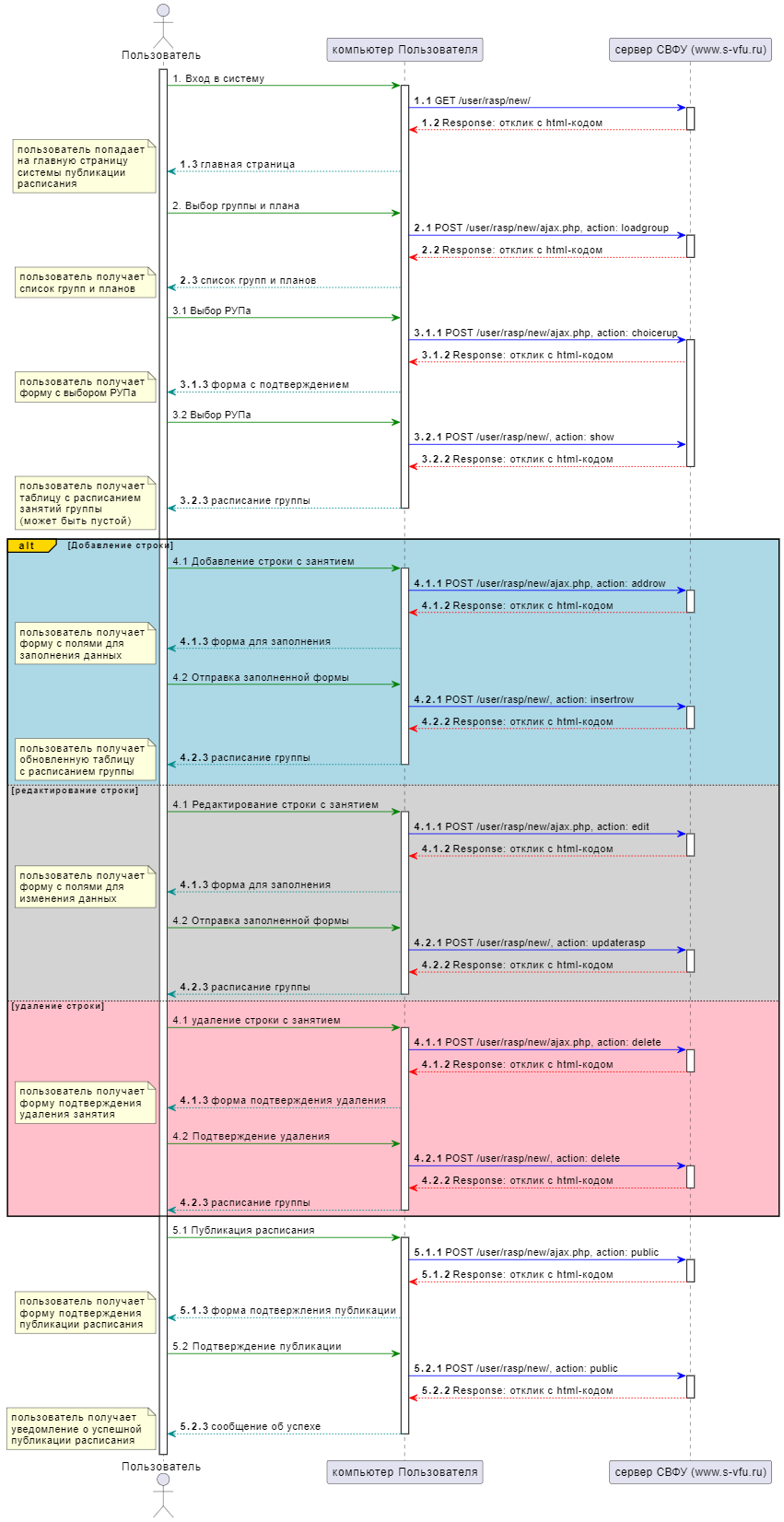
Помимо этого, можно считать, что разработанную мной систему автоматизации смогут применять и другие учебные подразделения, если будут строго придерживаться требований к формату документа с расписанием.

# СПИСОК ЛИТЕРАТУРЫ И ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вигерс, Карл Разработка требований к программному обеспечению. 3-е издание, дополненное / Карл Вигерс, Джой Битти. — Санкт-Петербург: БХВ-Петербург, 2019. — 736 с.
2. Гребенюк, О.С. (ред.) Системы управления учебным процессом в высшей школе: учебное пособие / О.С. Гребенюк. - Ростов н/Д: Феникс, 2011. - 336 с.
3. Герасимов, С.Г., Герасимова, В.И. Информационные технологии в управлении учебным процессом / С.Г. Герасимов, В.И. Герасимова. - Санкт-Петербург: БХВ-Петербург, 2014. - 368 с.
4. Корнеева, Д.А. Математическое моделирование и оптимизация процесса составления учебного расписания в вузе / Д.А. Корнеева // Вестник Московского государственного технического университета им. Н.Э. Баумана. Сер. Приборостроение. – 2014. – №3. – С. 129-143.
5. Семенов, С.П., Татаринцев, Я.Б. Сравнительный анализ подходов к автоматизации составления расписаний учебных занятий в образовательных учреждениях / С.П. Семенов, Я.Б. Татаринцев // Известия Алтайского государственного университета. – 2010. – С. 103-105.
6. Семиков, И.А. Клиент серверные архитектуры, сравнение уровневых архитектур // Актуальные проблемы современной науки и производства: Материалы VII Всероссийской научно-технической конференции, 21–23 нояб. 2022 г. – Рязань: РГРТУ, 2022. – С. 257-262.
7. Андреев, А. Язык программирования Python – причины популярности, преимущества и области применения [Электронный ресурс]. URL: <https://apix-drive.com/ru/blog/useful/jazyk-programmirovanija-python> (дата обращения: 24.04.2023)
8. Архитектура клиент-сервер // Курс лекций «Тестирование программного обеспечения» [Электронный ресурс]. URL: <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-6-ch1-arhitektura-klient-server.html> (дата обращения: 10.05.2023)
9. Архитектура программного обеспечения [Электронный ресурс]. URL: [https://datascience.eu/ru/компьютерное-зрение/архитектура-программного-обеспечения](https://datascience.eu/ru/компьютерное-зрение/архитектура-программного-обеспечения%20) (дата обращения: 20.10.2023)
10. Библиотека Requests в Python [Электронный ресурс]. URL: https://dvmn.org/encyclopedia/modules/requests/ (дата обращения: 14.03.2023)
11. Библиотека Requests: HTTP for Humans [Электронный ресурс]. URL: <https://python.ru/post/97/> (дата обращения: 25.02.2023)
12. Диаграмма вариантов использования (Use Case Diagram) [Электронный ресурс]. URL: <https://flexberry.github.io/ru/fd_use-case-diagram.html> (дата обращения: 31.05.2023)
13. Как работать с библиотекой Requests в Python [Электронный ресурс]. URL: [https://www.8host.com/blog/kak-rabotat-s-bibliotekoj-requests-v-python/](https://www.8host.com/blog/kak-rabotat-s-bibliotekoj-requests-v-python/%20) (дата обращения: 20.05.2023)
14. Клюковкин, Г. Выбор Request-Response парадигмы API: REST, RPC или GraphQL? [Электронный ресурс]. URL: <https://tproger.ru/articles/vybor-request-response-paradigmy-api-rest-rpc-ili-graphql> (дата обращения: 10.03.2023)
15. Макарик, М. Библиотека Requests: эффективные и простые HTTP-запросы в Python [Электронный ресурс]. URL: <https://smartiqa.ru/blog/python-requests> (дата обращения: 24.01.2023)
16. Что такое API? [Электронный ресурс]. URL: <https://aws.amazon.com/ru/what-is/api/> (дата обращения: 20.10.2023)
17. Welcome to the ArchiMate® 3.0 Specification [Электронный ресурс]. URL: <https://pubs.opengroup.org/architecture/archimate30-doc/> (дата обращения 10.05.2023)
18. Welcome to Flask – Flask Documentation (2.1.x) [Электронный ресурс]. URL: <https://flask.palletsprojects.com/en/2.1.x/> (дата обращения: 21.10.2023)

# ПРИЛОЖЕНИЯ

Приложение А. Диаграмма последовательности



Приложение Б. Реализация модели классов на Python

from enum import Enum

class Classroom:

    def \_\_init\_\_(self, aud\_number):

        self.aud\_number = aud\_number

class DayOfWeek(Enum):

    monday = 1

    tuesday = 2

    wednesday = 3

    thursday = 4

    friday = 5

    saturday = 6

class EducationLevel(Enum):

    bachelor = 1

    master = 2

    specialist = 3

    doctor = 4

class Group:

    def \_\_init\_\_(self, group\_name, educ\_lvl, course):

        self.group\_name = group\_name

        self.educ\_lvl = educ\_lvl

        self.course = course

class Lecturer:

    def \_\_init\_\_(self, name):

        self.name = name

class Parity(Enum):

    everyWeek = 1

    onEvenWeek = 2

    onOddWeek = 3

class Subject:

    def \_\_init\_\_(self, subject\_name):

        self.subject\_name = subject\_name

class Time:

    def \_\_init\_\_(self, start, end):

        self.start = start

        self.end = end

class TypeOfLearningActivity(Enum):

    labs = 1

    lecture = 2

    practice = 3

class User:

    def \_\_init\_\_(self, login, password):

        self.login = login

        self.password = password

class Lesson(Subject, Group, Lecturer, DayOfWeek, Time, Classroom, Parity,

            TypeOfLearningActivity):

    def \_\_init\_\_(self, subject, group, lecturer, day, time, classroom,

                 is\_sub\_divided, parity, activity, is\_online):

        self.subject = subject

        self.group = group

        self.lecturer = lecturer

        self.day = day

        self.time = time

        self.classroom = classroom

        self.is\_sub\_divided = is\_sub\_divided

        self.parity = parity

        self.activity = activity

        self.is\_online = is\_online

Приложение В. Исходный код программы

from flask import Flask, render\_template, request, redirect, url\_for, session, Blueprint

from openpyxl import load\_workbook

from bs4 import BeautifulSoup

import requests

import re

import jwt

import logging

from logging.handlers import RotatingFileHandler

bp = Blueprint('main', \_\_name\_\_)

logger = logging.getLogger(\_\_name\_\_)

logger.setLevel(logging.INFO)

log\_file = 'app.log'

file\_handler = RotatingFileHandler(log\_file, maxBytes=1024\*1024, backupCount=5, encoding="utf8")

file\_handler.setLevel(logging.INFO)

formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')

file\_handler.setFormatter(formatter)

logger.addHandler(file\_handler)

app = Flask(\_\_name\_\_)

app.secret\_key = 'mysecretkey'

my\_session = requests.Session()

system\_url = "https://www.s-vfu.ru/user/rasp/new/"

server\_url = "https://www.s-vfu.ru/user/rasp/new/ajax.php"

@bp.route('/')

@bp.route('/auth')

def index():

    return render\_template('index.html')

@bp.route('/main', methods=['post', 'get'])

def authorize():

    if request.method == 'POST':

        username = request.form.get('username')

        password = request.form.get('password')

    url = 'https://www.s-vfu.ru/?login=yes'

    data = {

        'AUTH\_FORM': 'Y',

        'TYPE': 'AUTH',

        'USER\_LOGIN': username,

        'USER\_PASSWORD': password,

        'Login': ''

    }

    cookies = {

        "entersite": "www.s-vfu.ru",

    }

    res = my\_session.post(url, data=data, cookies=cookies, verify=False)

    logger.info("Результат авторизации: " + str(res.status\_code))

    right\_index = res.text.find("<h1>") + 4

    left\_index = res.text.find("</h1>")

    if left\_index != -1:

        title = "Главная"

        name = res.text[right\_index:left\_index]

        logger.info(f'В систему вошел пользователь: {name}')

    else:

        title = "Login failed"

        right\_index = res.text.find("<strong>Ошибка!</strong>")

        left\_index = res.text.find("авторизация,") + 11

        name = "Ошибка авторизации! Неверные логин и/или пароль"

        logger.info(name)

        return redirect(url\_for('index'))

    res = my\_session.get(system\_url)

    soup = BeautifulSoup(res.text, 'html.parser')

    buid = str(soup.find('input', {'name': 'buid'}))

    if buid is None:

        error = f'Ошибка при попытке получения идентификатора BITRIX пользователя {name}!'

        logger.error(error)

        return render\_template('main.html', error=error)

    session['buid'] = buid

    return render\_template('main.html', name=name, title=title)

@bp.route("/schedule", methods=['GET', 'POST'])

def schedule\_parse():

    path = 'flask/static/'

    try:

        file = request.files['file']

    except KeyError as e:

        error = "Ошибка при получении файла: " + e

        logger.error(error)

        return render\_template('main.html', error=error)

    try:

        file.save(path + file.filename)

    except PermissionError as e:

        error = "Ошибка про попытке сохранить файл с расписанием: " + str(e)

        logger.error(error)

        return render\_template('main.html', error=error)

    except FileNotFoundError as e:

        error = "Ошибка про попытке сохранить файл с расписанием!" + str(e)

        logger.error(error)

        return render\_template('main.html', error=error)

    form = request.form.get('form')

    fac = request.form.get('fac')

    if fac and form:

        buid = session.get('buid', 'ошибка')

        try:

            wb = load\_workbook(filename=path + file.filename)

        except FileNotFoundError:

            error = "Указанный файл не найден!"

            logger.error(error)

            return render\_template('main.html', error=error)

        sheets\_names = wb.sheetnames

        # цикл по листам excel-файла

        for sh in sheets\_names:

            if sh.find("курс") == -1:

                logger.warning(f'Возможно, название листа "{sh}" выбранной книги не соответствует требованиям формата файла с расписанием!')

                continue

            wb.active = sheets\_names.index(sh)

            ws = wb.active

            course = str(ws.cell(row=2, column=1).value).strip()[0]

            year\_and\_semestr = str(ws.cell(row=1, column=1).value)

            year, semestr = get\_year\_and\_semestr(year\_and\_semestr)

            if course == "None" and year\_and\_semestr == "None":

                logger.warning("Не удалось извлечь значение курса из файла!")

                logger.warning("Не удалось извлечь значение года и семестра из файла!")

                continue

            # цикл по всем группам

            for i in range(3, ws.max\_column, 4):

                if str(ws.cell(row=3, column=i).value).strip().lower() != "наименование группы":

                    continue

                group\_name = str(ws.cell(row=4, column=i).value).strip()

                if group\_name == "None":

                    logger.warning(f'Группа {group\_name} была пропущена, т.к. не удалось извлечь название группы из файла! ')

                    continue

                if semestr == "1":

                    startdate = str(ws.cell(row=4, column=i + 2).value).strip() + "." + year

                    enddate = str(ws.cell(row=4, column=i + 3).value).strip() + "." + year

                else:

                    startdate = str(ws.cell(row=4, column=i + 2).value).strip() + "." + str(int(year) + 1)

                    enddate = str(ws.cell(row=4, column=i + 3).value).strip() + "." + str(int(year) + 1)

                # получения кода для формы обучения

                code = get\_code(group\_name)

                # получение списка подходящих групп

                response = query(id=buid, action="loadgroup", fac=fac, code=code,

                                    course=course, form=form, semestr=semestr, year=year)

                # получение нужной группы из списка

                try:

                    group = parse\_loadgroup(response.text, group\_name)

                except AttributeError as e:

                    logger.error(f"Ошибка при попытке получить нужную группу из списка: {e}.\nГруппа пропущена")

                    continue

                if group is None:

                    logger.error(f"Ошибка при попытке получить группу {group\_name} из списка, полученного с сервера! Группа пропущена")

                    continue

                # получение id группы

                try:

                    group\_id = group[group.find("|") + 1:]

                except AttributeError:

                    logger.error("Ошибка при попытке получить идентификатор группы! Группа была пропущена")

                    continue

                # получение РУПа

                response = query(id=buid, action="choicerup",

                                    fac=fac, course=course, form=form, semestr=semestr, year=year, groupname=group)

                full\_semestr, filename = parse\_choicerup(response.text)

                if full\_semestr is None or filename is None:

                    logger.error("Ошибка при попытке получить значение семестра и РУПа группы! Группа была пропущена")

                    continue

                response = query(action="show", semestr=semestr, course=course, fac=fac,

                                    year=year, form=form, code=code, id\_group=group\_id, filename=filename)

                last\_index = str(group\_id).rfind("|") + 1

                full = f"{fac}|{filename}|{group\_id[:last\_index]}{full\_semestr}|{course}|{year}|{semestr}|{group\_id[last\_index:len(group\_id)]}|0{code}|{group\_id[last\_index:len(group\_id)]}|{form[0]}"

                # получение ИД всех существующих занятий

                try:

                    lessons = get\_lessons(response.text)

                except AttributeError as e:

                    logger.error(f"Ошибка при попытке получить идентификаторы существующих занятий: {e}.\nГруппа пропущена")

                    continue

                # удаление всех существующих занятий

                if len(lessons) > 0:

                    for k in range(0, len(lessons), 4):

                        query(action="delete", id=1,

                              cell\_id=lessons[k], full=full, fac=fac)

                        query(action="remove", cell\_id=lessons[k], full=full, id\_group=group\_id,

                              filename=filename, semestr=semestr, full\_semestr=full\_semestr, course=course,

                              fac=fac, year=year, form=form[0], code=code)

                weekday = str(

                                ws.cell(row=6, column=1).value).strip().upper()

                # цикл по занятиям одной группы

                for j in range(6, 42):

                    # проверка, что дисциплина есть (наличие пары)

                    if ws.cell(row=j, column=i).value is not None:

                        # получение дня недели

                        if j % 6 == 0:

                            weekday = str(

                                ws.cell(row=j, column=1).value).strip().upper()

                        time = str(ws.cell(row=j, column=2).value).replace(

                            ".", ":").replace(" -- ", "-")

                        lesson\_name = str(ws.cell(

                            row=j, column=i).value).split('\n')

                        for q in range(len(lesson\_name)):

                            lesson = lesson\_name[q].strip()

                            podgruppa, lesson = get\_podgruppa(lesson, q)

                            try:

                                lesson, chet = get\_parity(lesson)

                            except AttributeError as e:

                                logger.error(f"Ошибка при попытке получить четность: {e}.\nГруппа пропущена")

                                continue

                            lecturers = str(

                                ws.cell(row=j, column=i + 1).value).split("\n")

                            if len(lecturers) > 1:

                                lecturer\_name = lecturers[q].strip()

                            else:

                                lecturer\_name = lecturers[0].strip()

                            # добавление строки в таблицу

                            response = query(action="addrow", full=full)

                            # получение данных проподователя с сервера

                            if lecturer\_name != "None":

                                hours, lecturer = parse\_addrow(

                                    response.text, lecturer\_name)

                            else:

                                hours, lecturer = ("","")

                            activity = str(ws.cell(row=j, column=i + 2).value)

                            activity = get\_activity(activity)

                            classrooms = str(ws.cell(row=j, column=i + 3).value).split("\n")

                            if len(classrooms) > 1:

                                classroom = classrooms[q].strip()

                            else:

                                classroom = classrooms[0].strip()

                            corpus, classroom = extract\_corpus(classroom)

                            result = query(full=full, id\_group=group\_id, filename=filename, semestr=semestr, course=course,

                                        fac=fac, year=year, code=code, form=form[0], action="insertrow", full\_semestr=full\_semestr, lesson=lesson,

                                        lecturer=lecturer, weekday=weekday, time=time, chet=chet, startdate=startdate, enddate=enddate,

                                        activity=activity, corpus=corpus, classroom=classroom, hours=hours, podgruppa=podgruppa)

                    response = query(action="public\_check", full=full, fac=fac)

                    if response.text.find(f'После нажатия кнопки "Применить" расписание группы ИМИ-{group\_name} будет опубликовано') != -1:

                        print(response.text)

                        response = query(action="public", full=full, id\_group=group\_id, filename=filename, semestr=semestr,

                                    full\_semestr=full\_semestr, course=course, fac=fac, year=year, code=code, form=form[0])

        return render\_template('schedule.html', buid=buid, res=result.text)

        # return redirect(system\_url)

    else:

        error = "Ошибка при попытке получения факультета и формы обучения!"

        logger.error(error)

        return redirect(url\_for('main'))

def get\_lessons(html\_code):

    # Найти таблицу по ее идентификатору

    soup = BeautifulSoup(html\_code, 'html.parser')

    table = soup.find('table', id='mytable')

    # Получить все строки таблицы

    rows = table.find\_all('tr')

    lessons\_id = []

    # Пройтись по каждой строке и извлечь данные из ячеек

    for row in rows:

        cells = row.find\_all('td')

        for cell in cells:

            cell\_id = cell.get('id')

            if cell\_id:

                lessons\_id.append(cell\_id)

    return lessons\_id

def get\_year\_and\_semestr(string):

    string1 = re.findall(r"\b\d+\b(?=\s\*полугодие)", string)

    if string1:

        semestr = string1[0]

    else:

        semestr = None

    string2 = re.findall(r"полугодие\s+(\d+)\s\*-\s\*", string)

    if string2:

        year = string2[0][-4:]

    else:

        year = None

    return (year, semestr)

def get\_podgruppa(lesson, q):

    if lesson.find("1/2") == -1:

        podgruppa = 0

    else:

        podgruppa = q + 1

    return (podgruppa, (lesson[:len(lesson) - 5]).strip())

def get\_parity(lesson):

    if lesson.endswith("\*\*"):

        return ((lesson[:len(lesson) - 2]).strip(), "2")

    elif lesson.endswith("\*"):

        return ((lesson[:len(lesson) - 1]).strip(), "1")

    elif lesson.endswith("\*,\*\*") or lesson.endswith("\*, \*\*"):

        return ((lesson[:lesson.find("\*")]).strip(), "0")

    else:

        return (lesson, "0")

def get\_code(group\_name):

    string = group\_name.strip()[0].upper()

    if string == "Б":

        return "3"

    elif string == "М":

        return "4"

    elif string == "А":

        return "6"

    return "5"

def get\_activity(act):

    act = act.replace(" ", "").replace('\\', "").replace('/', "").replace(',', '').replace("\n",'')

    act = act.lower()

    if act == "лек":

        return "лекция"

    elif act == "пр":

        return "практика"

    elif act == "лекпр" or act == 'леклаб':

        return "лекция, практика"

    elif act == "лаб":

        return "Лабораторная работа"

    elif act == "срс":

        return "самостоятельная работа"

    else:

        return ""

def extract\_corpus(string):

    pattern1 = r'[а-яА-ЯёЁ]+'

    pattern2 = r'\d+'

    word = (re.search(pattern1, string))

    number = (re.search(pattern2, string))

    if word is not None and number is not None:

        return (word.group(), number.group())

    elif string == "Спортивный":

        return ("Юность", string)

    else:

        return ("КФЕН", number.group())

def query(full="", id="", action="", fac="",

          code="", course="", form="",

          semestr="", year="", filename="",

          id\_group="", groupname="",

          chet="", weekday="", activity="",

          corpus="", classroom="", lesson="",

          lecturer="", time="", full\_semestr="",

          startdate="", enddate="",

          cell\_id="", hours="", podgruppa=""):

    url = "https://www.s-vfu.ru/user/rasp/new/ajax.php"

    # выбрать группу

    if action == 'loadgroup':

        data = {

            'id': id,

            'action': action,

            "fac": fac,

            "code": code,

            "course": course,

            "form": form,

            "semestr": semestr,

            "year": year

        }

    # выбрать руп

    elif action == 'choicerup':

        data = {

            'id': id,

            'action': action,

            "fac": fac,

            "course": course,

            "form": form,

            "semestr": semestr,

            "year": year,

            "groupname": groupname,

        }

    # выбор группы и РУПа

    elif action == 'show':

        url = "https://www.s-vfu.ru/user/rasp/new/"

        data = {

            "global\_semestr": semestr,

            "semestr": full\_semestr,

            "course": course,

            "fac": fac,

            "year": year,

            "formshort": form[0],

            "formname": form[2:],

            "action": action,

            "allplany": "on",

            "code": "0" + code,

            "id\_group": id\_group,

            "plan": filename,

            "startdate": startdate,

            "enddate": enddate

        }

    # удаление строки

    elif action == 'delete':

        data = {

            "action": action,

            "id": id,

            "data": cell\_id,

            "full": full,

            "fac": fac

        }

    # подтверждение удаления строки

    elif action == 'remove':

        url = "https://www.s-vfu.ru/user/rasp/new/"

        data = {

            "data": full,

            "id\_group": id\_group,

            "filename": filename,

            "global\_semestr": semestr,

            "semestr": full\_semestr,

            "course": course,

            "fac": fac,

            "year": year,

            "form": "0" + code,

            "formshort": form,

            "action": "delete",

            "id": cell\_id[2:]

        }

    # добавление строки

    elif action == 'addrow':

        id = 1

        data = {

            "action": action,

            "id": id,

            "full": full

        }

    elif action == "choicecorpus":

        data = {

            "id": id,

            "action": action,

            "corpus": corpus,

            "fac": fac

        }

    # вставка строки

    elif action == 'insertrow':

        url = "https://www.s-vfu.ru/user/rasp/new/"

        data = {

            "data": full,

            'id\_group': id\_group,

            "filename": filename,

            "global\_semestr": semestr,

            "semestr": full\_semestr,

            "course": course,

            "fac": fac,

            "year": year,

            "form": "0" + code,

            "formshort": form,

            'id': 1,

            'action': action,

            'I': lesson,

            "J": lecturer,

            "hours": hours,

            'podgruppa': podgruppa,

            "B": weekday,

            "F": time,

            "chet": chet,

            # "c": "09.01.2023",

            # "d": "30.06.2023",

            'c': startdate,

            'd': enddate,

            "H": activity,

            "L": corpus,

            "K": classroom

        }

    # публикация расписания

    elif action == 'public\_check':

        data = {

            'id': 1,

            'action': "public",

            'full': full,

            "fac": fac

        }

    elif action == 'public':

        url = "https://www.s-vfu.ru/user/rasp/new/"

        data = {

            'data': full,

            'id\_group': id\_group,

            'filename': filename,

            'global\_semestr': semestr,

            'semestr': full\_semestr,

            'course': course,

            'fac': fac,

            'year': year,

            'form': "0" + code,

            'formshort': form,

            'action': action,

        }

    response = None

    try:

        response = my\_session.post(url=url, data=data)

    except requests.exceptions.Timeout as e:

        logger.error("Произошла ошибка: " + e)

    except requests.exceptions.TooManyRedirects as e:

        logger.error("Произошла ошибка: " + e)

    except requests.exceptions.ConnectionError as e:

        logger.error("Произошла ошибка: " + e)

    except requests.exceptions.RequestException as e:

        logger.error("Произошла ошибка: ")

    except Exception as e:

        logger.error("Произошла ошибка: "+ e)

    if response is None or response.text == 'error':

        logger.warning(f'При попытке выполнить действие "{action}" произошла непредвиденная ошибка!')

        logger.info(f'Параметры запроса для действия "{action}": {data}')

    return response

def parse\_loadgroup(html, groupname):

    soup = BeautifulSoup(html, 'html.parser')

    select = soup.find('select')

    if select:

        options = select.find\_all('option')

        for option in options:

            value = option.get('value')

            if value and groupname in value:

                return value

    return None

def parse\_choicerup(html):

    soup = BeautifulSoup(html, 'html.parser')

    plan = soup.find('input', {'name': 'plan'}).get('value')

    semestr = soup.find('input', {'name': 'semestr'}).get('value')

    if plan and semestr:

        return (semestr, plan)

    return (None, None)

def parse\_addrow(html, lecturer):

    soup = BeautifulSoup(html, 'html.parser')

    options = soup.find\_all('option')

    surname, initials = lecturer.split()

    initials = initials.replace(".", "")

    for option in options:

        text = option.text

        if text.startswith(surname):

            string = text.split()

            lecturer\_initials = string[1][0] + string[2][0]

            if initials == lecturer\_initials:

                return ("", text + "|" + option['value'])

    else:

            response = requests.get(

            url=f"https://www.s-vfu.ru/stud/searchadddata.php?tablename=svfudbnew.forexcel&term={surname} {initials[0]}")

        data = response.json()

        if len(data) > 0:

            logger.warning("Возможно, был указан неверный преподаватель, т.к. было найдено несколько совпадений по полученным фамилии и инициалам!")

        for d in data:

            string = str(d).split()

            if string[2].startswith(initials[1]):

                return (d[d.find("|") + 1:], "")

app.register\_blueprint(bp, url\_prefix='')

if \_\_name\_\_ == '\_\_main\_\_':

    app.debug = True

    app.run()