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 УЧЕБНОГО РАСПИСАНИЯ**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Выполнил: студент IV курса

группы БА-ФИИТ-19 ИМИ СВФУ

Собянин Сергей Павлович

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

(подпись)

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

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

(подпись)

Якутск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc136894527)

[ГЛАВА 1. ПУБЛИКАЦИЯ РАСПИСАНИЯ В ИМИ СВФУ 6](#_Toc136894528)

[1.1 Постановка задачи 6](#_Toc136894529)

[1.2 Описание API системы публикации расписания СВФУ 13](#_Toc136894530)

[1.3 Обзор аналогов 14](#_Toc136894531)

[1.4 Обзор инструментов разработки 18](#_Toc136894532)

[Выводы по главе 1 24](#_Toc136894533)

[2.1 О приложении 25](#_Toc136894534)

[2.2 Архитектура ПО 29](#_Toc136894535)

[2.3 Описание тестового сервера 30](#_Toc136894536)

[2.4 Описание приложения 31](#_Toc136894537)

[2.5 Апробация 34](#_Toc136894538)

[Выводы по главе 2 34](#_Toc136894539)

[ЗАКЛЮЧЕНИЕ 35](#_Toc136894540)

[СПИСОК ЛИТЕРАТУРЫ И ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc136894541)

[ПРИЛОЖЕНИЯ 38](#_Toc136894542)

# 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 [14].

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

Существуют следующие основные способы составления расписания [14]:

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 курсовой 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 предметной области;
2. изучение и анализ расписания ИМИ;
3. анализ системы публикации расписания, анализ запросов;
4. анализ и формирование требований;
5. обзор библиотек и инструментов разработки;
6. проектирование, тестирование и реализация собственного решения по автоматизации системы публикации расписания.

Объект исследования: процедура публикации расписания ИМИ СВФУ.

Предмет исследования: автоматизация 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 боевого сервера для тестирования и эксплуатации ПО.

Данная работа состоит из введения, двух глав, выводов по каждой главе, заключения и списка использованной литературы из 16 наименования, четырех приложений, содержит 50 страниц, а также 15 рисунков. В первой главе проведен обзор предметной области: проведен анализ расписания и системы его публикации в СВФУ. Во второй главе описывается архитектура приложения и его работа. В заключении подводятся итоги проделанной работы, приведены выводы из результатов проделанной работы.

# ГЛАВА 1. 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.

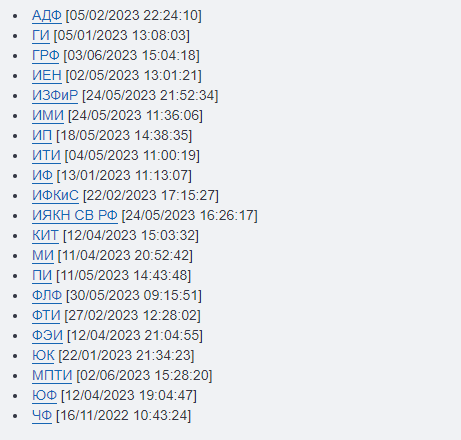


Рисунок 1 – Выбор УЧП



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.

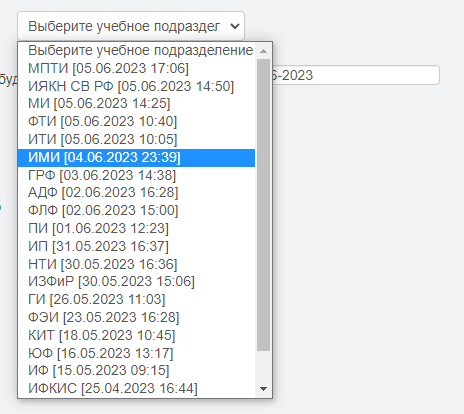


Рисунок 3 – Выбор УЧП

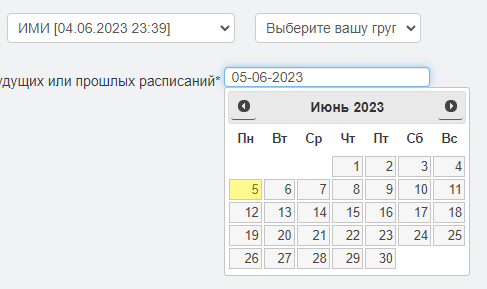


Рисунок 4 – Выбор даты

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

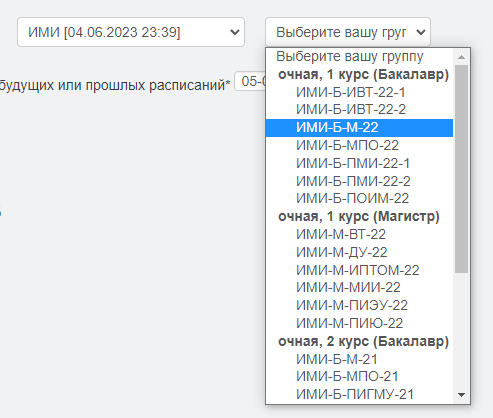


Рисунок 5 – Выбор учебной группы

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):

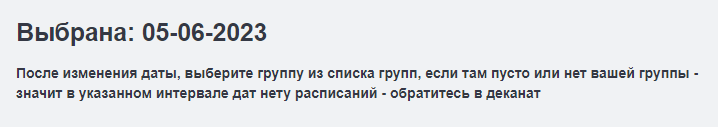


Рисунок 6 – Предупреждение

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.

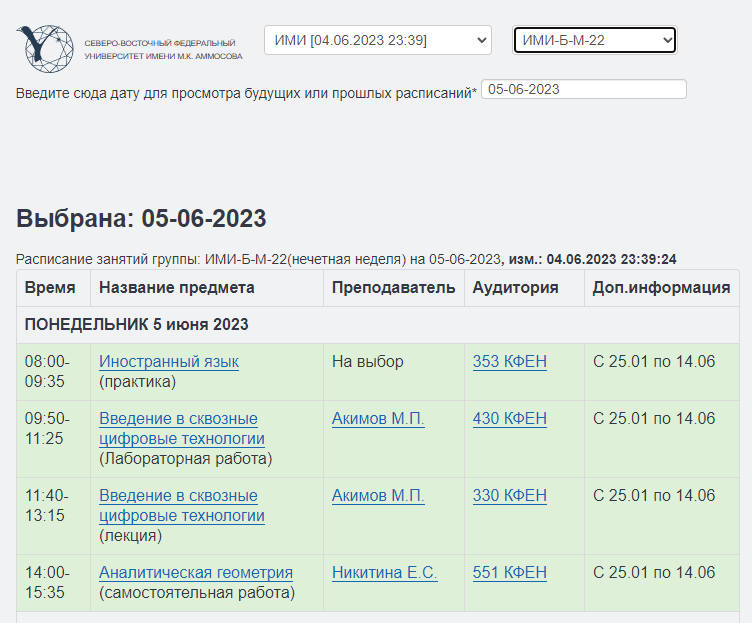


Рисунок 7 – Онлайн-расписание

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).

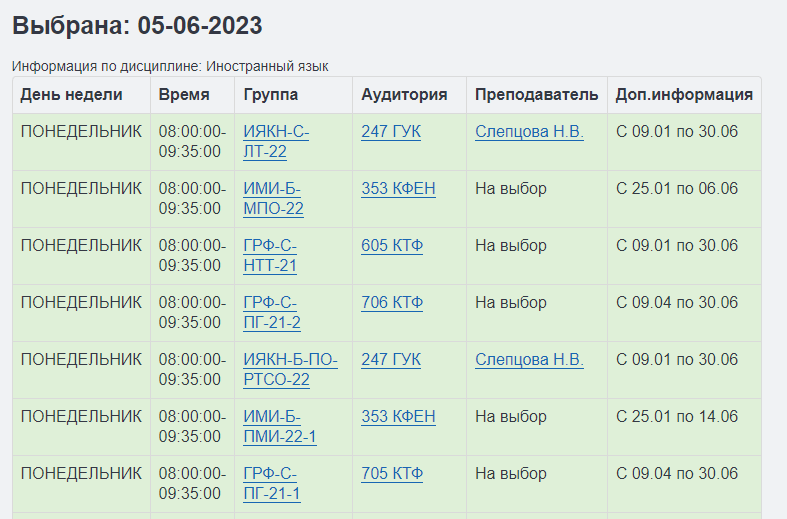


Рисунок 8 – Информация по дисциплине

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).

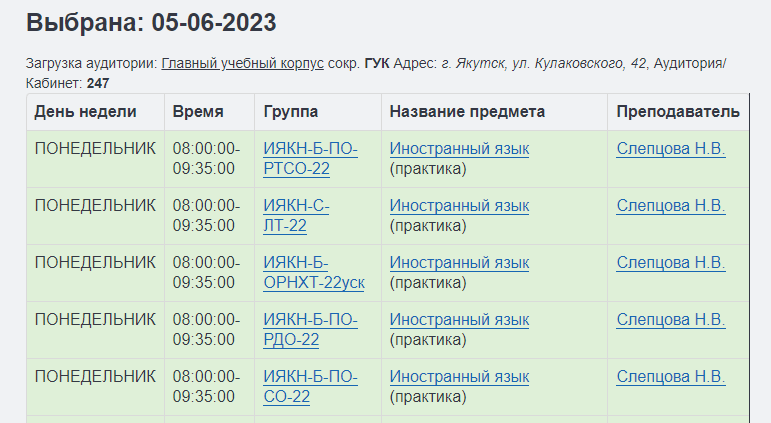


Рисунок 10 – Загруженность аудитории

**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.

Я проанализировал веб-страницу, 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);

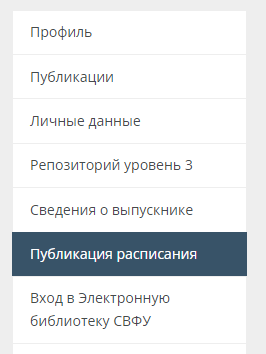


Рисунок 11 – Система публикации расписания

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.

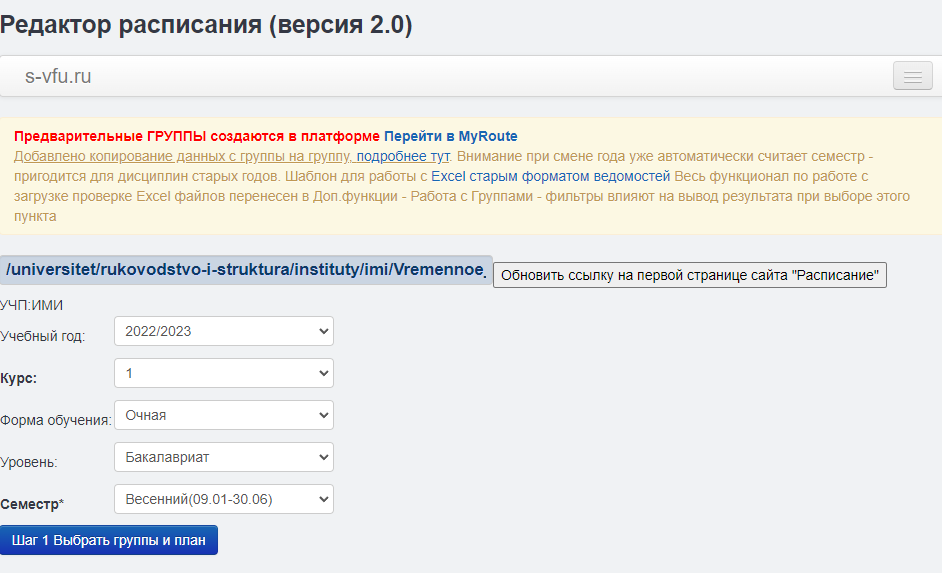


Рисунок 12 – Заполнение стартовой формы

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.

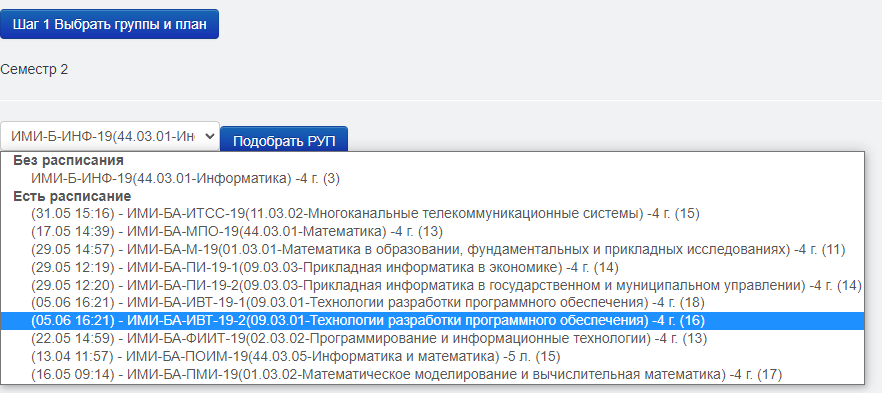


Рисунок 13 – Выбор группы и плана

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».

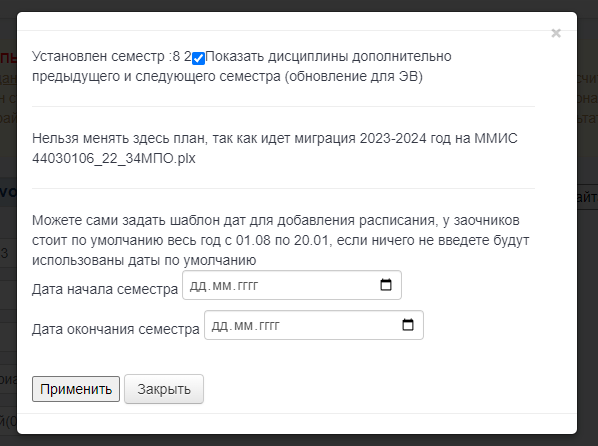


Рисунок 14 – Выбор РУП

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).

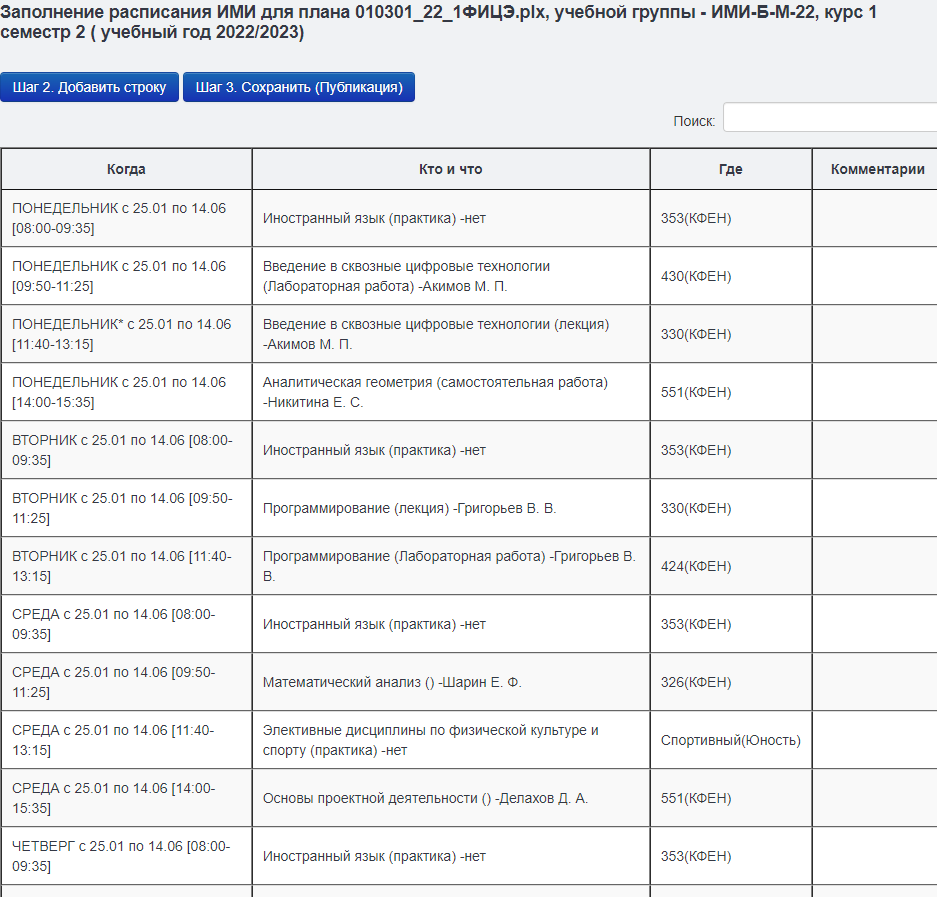


Рисунок 15 – Таблица с расписанием

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 (профессорско-преподавательский состав)**. Поле с автозаполнением. Проанализировав веб-страницу с добавлением расписания и запросы, отправляющиеся с нее, я выяснил, 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».

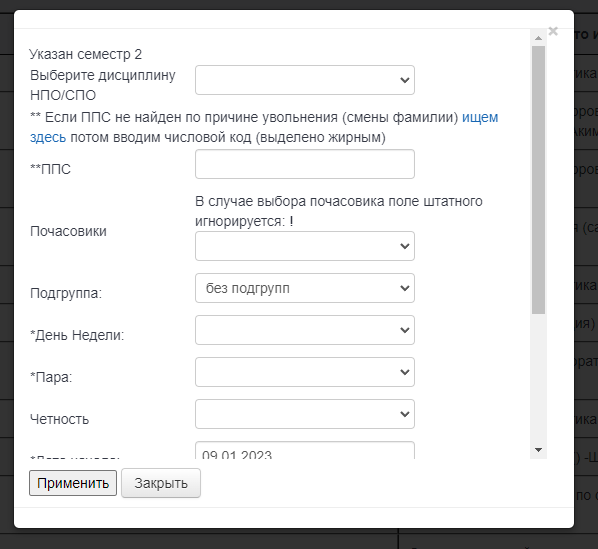


Рисунок 16 – Добавление строки с занятием в таблицу

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.

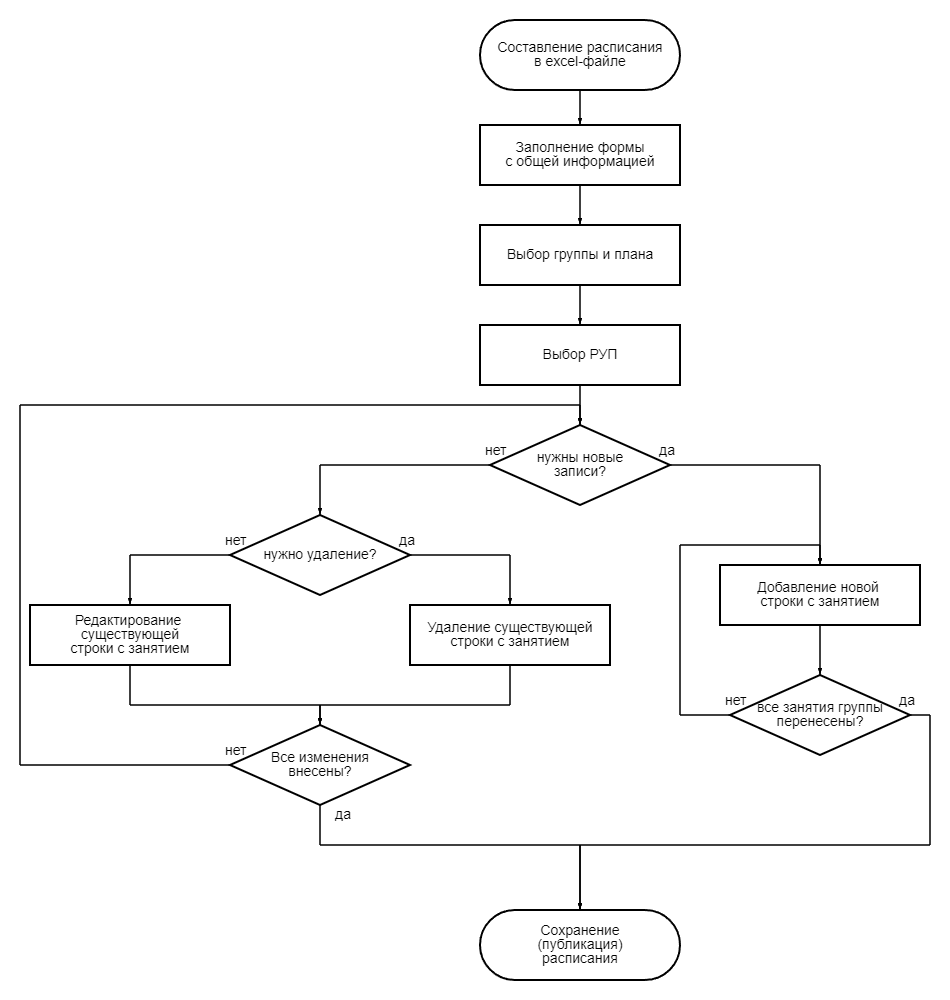


Рисунок 17 – Процесс публикации расписания

# 1.2 Описание API системы публикации расписания СВФУ

В ходе описания процесса публикации расписания мною также был осуществлен тщательный анализ http-запросов, отправляющихся на production-сервер СВФУ, и откликов в формате html-кода.

Взаимодействие с production-сервером проходит следующим образом.

Есть две конечных 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:

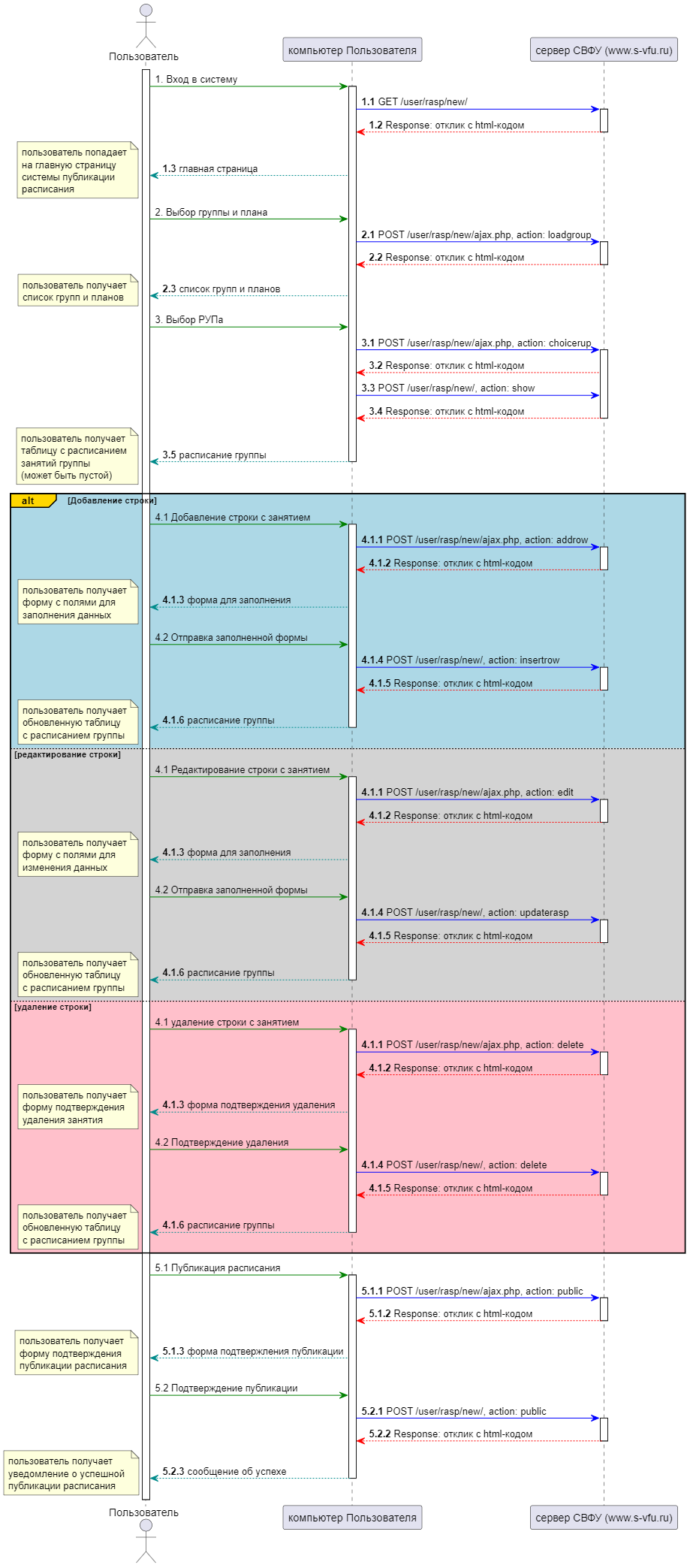
* + action – действие, которое необходимо совершить;
  + fac – факультет;
  + code – код уровня образования. Бакалавриат – 3, Магистратура – 4, Специалитет – 5, Аспирантура – 6;
  + course – курс;
  + formshort – код формы обучения (1 – очная, 2 – заочная, 99 – очно-заочная);
  + formname – наименование 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.
* 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:

* + 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;
* public – если отправлять на второй ресурс, запускаются проверки валидности 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:

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



# 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

Программа «1С: Автоматизированное составление расписания. Университет» предназначена для решения задач автоматизированного составления учебных расписаний и оперативного управления помещениями в высших учебных заведениях.  С его помощью составлять расписание можно в автоматическом, ручном и смешанном режимах с учетом многих ограничений и условий. При этом можно построить как допустимое расписание, так и оптимизированное, в котором сокращено количество окон или количество используемых помещений.

Реализован следующий функционал:

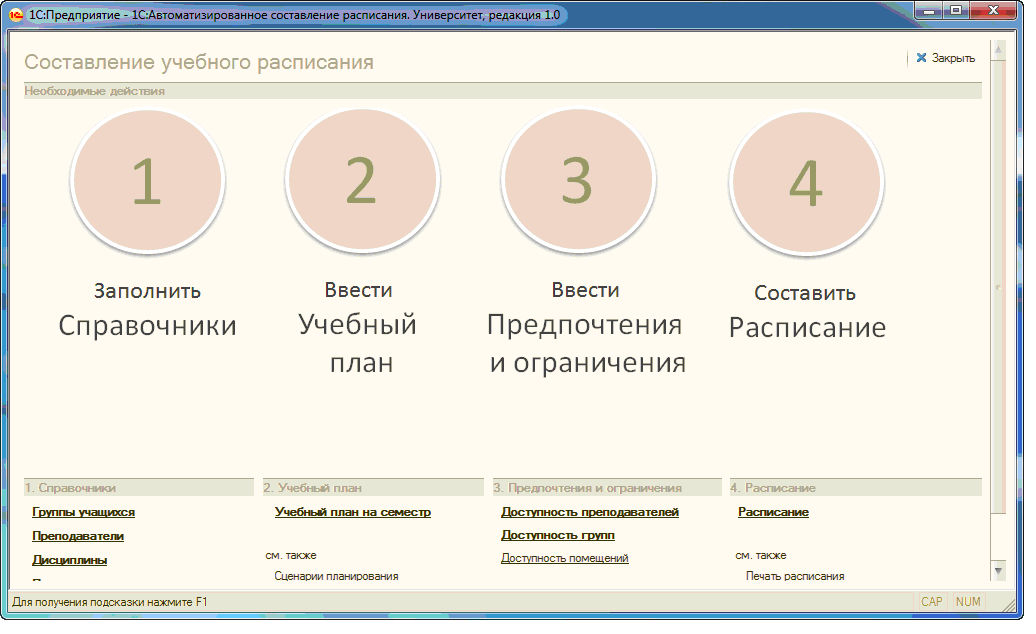
* составление расписаний в режимах:
  + ручной,
  + автоматический,
  + смешанный,

а также в режимах:

* + по помещениям,
  + по группам,
  + по преподавателям.
* удобная форма "шахматка" для быстрой ручной модификации расписания перетаскиванием "drag&drop"
* составление нескольких расписаний и выбор лучшего;
* составление расписаний в разрезе:
  + семестров;
  + сценариев;
  + кафедр;
  + типа расписания;
  + периода, на который составлено расписание;
* учет пожеланий и возможностей преподавателей, групп студентов, помещений;
* оптимизация учебных расписаний по одному из критериев:
  + количество используемых помещений,
  + минимизировать количество окон и т.д.;
* сравнение на допустимость при составлении расписания в любом режиме:
  + тип помещения / тип занятия,
  + вместимость помещения / количество студентов в группе;
* выбор произвольной периодичности расписания (неделя, две недели, семестр, фиксированный период и т.д.);
* составление расписания сессии;
* учет параллельных занятий, разбиения на подгруппы и потоковых лекций при составлении расписания;
* учет максимального допустимого количества занятий в день для группы студентов или преподавателя при составлении расписания;
* построение расписания для 2-х и более смен;
* оперативное резервирование помещений (дополнительно можно указывать мероприятие и/или причину резервирования);
* просмотр расписаний и ввод предпочтений по web-интерфейсу;
* уведомление об изменении расписаний по e-mail для студентов и преподавателей (отсылаемый документ формируется автоматически в форме отчета "Печать расписания");
* быстрая форма замены занятий;
* учет графика учебно-производственного процесса;
* отчеты: "Проведенные занятия", "Использование помещений по периодам", "Нагрузка";
* загрузка справочников (в том числе списки преподавателей, дисциплин, групп) и учебных планов в форматах Excel и XML.
* оперативное изменение расписаний (путем перетаскивания);
* разграничение доступа по документам.

Процесс составления расписаний в системе делится на следующие этапы:

* ввод первичной информации: курсы, группы, дисциплины, преподаватели, помещения;
* ввод учебного плана на семестр, в котором указывается, кто, для кого, какое занятие и в каком объеме должен провести;
* ввод ограничений и предпочтений на преподавателей, студентов, помещения;
* составление учебного расписания.



Стоимость данного решения составляет от 95 до 99 тысяч рублей.

AscTimeTables

Популярное программное обеспечение, используемое для составления расписаний занятий в учебных заведениях по всему миру.

Основной функционал данного ПО:

* автоматическое создание расписания;
* возможность изменять автоматически созданное расписание вручную;
* проверка расписания на несостыковки и несоответствия условиям и ограничениям;
* упрощенный ввод данных и импорт из электронных документов;
* возможность публикации расписания онлайн;
* собственный веб-сервер EduPage;
* адаптация под нужды учебного заведения;
* электронный дневник.

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.

# 1.4 Обзор инструментов разработки

В современном мире существует огромное количество различных языков программирования (ЯП), заточенных под определенные парадигмы разработки. Однако мой выбор пал на ЯП Python в силу его особенностей и широких возможностей, например, таких, как [4]:

• Поддержка множества парадигм программирования (ООП, функциональное, императивное структурное, процедурное и т.д.)

• Разработка на Python позволяет разбивать программы на составные части – модули, которые, в свою очередь, можно объединять в пакеты.

• Поддержка полной интроспекции – возможности запросить тип и структуру объекта данных во время выполнения программы

• Обширная стандартная библиотека имеет набор модулей для работы с операционной системой (ОС), различными сетевыми протоколами, архивами, мультимедийными форматами, текстовыми кодировками, регулярными выражениями, криптографическими протоколами и т.д. Поддерживает юнит-тестирование.

• Большое разнообразие сторонних дополнительных поддерживаемых библиотек.

Среди основных преимуществ выбранного мной языка программирования можно выделить [4]:

• Скорость разработки. Для написания программ на Питоне требуется гораздо меньший объем кода, чем в случае с другими популярными языками (Java, C). Это заметно ускоряет разработку, позволяя создавать сложное ПО быстрее, чем на других ЯП.

• Логичный понятный синтаксис. Благодаря этому достоинству чтение и понимание кода значительно упрощаются, что делает его более легким в освоении. Так Python считается языком программирования с относительно низким порогом вхождения.

• Разнообразие библиотек. Как уже было упомянуто ранее, помимо стандартной библиотеки, предоставляющей широкий спектр возможностей, для Python доступен большой выбор сторонних библиотек, значительно расширяющий его функционал. Среди самых популярных следует отметить SQLAlchemy (для работы с базами данных), Pygame (для разработки мультимедийных приложений и игр), Flask и Django (для разработки серверной части ПО), NumPy (для разработки в сфере машинного обучения и искусственного интеллекта).

• Масштабируемость. Написанные на Питоне программы и приложения легко расширяются и масштабируются, благодаря возможности адаптации их высокоуровневой логики. Масштабируемость – это способность программного обеспечения корректно работать на малых и на больших системах с производительностью, которая увеличивается пропорционально вычислительной мощности системы.

• Универсальность. Python – это интерпретируемый язык, который используется для кодинга практически на все современных платформах. Он не нуждается в компиляции и его код можно писать в обычном текстовом документе.

• Глобальное сообщество. Одним из важных факторов бурной популяризации Питона считается многочисленное сообщество разработчиков и энтузиастов данного языка. Его развитие ведется на базе регулярно обновляемой и четко регламентированной документации PEP (предложений по развитию Python).

Однако, справедливости ради необходимо указать и на минусы Python’a. Таковыми можно считать недостаточное быстродействие, вызванное устаревшим GIL (Global Interpreter Lock), и динамическую типизацию, из-за которой ПО оказываются довольно ресурсоемкими, потребляя немалые объемы памяти.

Так как для разработки программы автоматизирования системы публикации расписания необходимо отправлять и обрабатывать HTTP-запросы, необходимо выбрать соответствующую библиотеку, позволяющую Python работать с такими запросами.

По умолчанию в Python доступна стандартная библиотека urllib, позволяющая считывать содержимое веб-сайта, выполнять HTTP-запросы, в частности GET и POST (благодаря модулю urllib.request), получать данные о файлах cookie, отправлять заголовки запроса и получать заголовки ответа [1].

Однако несмотря на доступность стандартной библиотеки urllib, я выбрал для себя в качестве инструмента обработки HTTP-запросов стороннюю библиотеку Requests, которая де-факто является на данный момент общепринятым стандартом.

Основные преимущества и возможности библиотеки Requests [6][8][10] [13]:

• код, написанный с применением Requests, не является громоздким, легко читается, а функции и методы наглядно настраиваются под специфические нужды;

• поддержка постоянного HTTP-соединения и его повторное использование;

• применение международных и национальных доменов;

• сессии с сохранением Cookie: передача и получение значений в формате ключ: значение;

• автоматическое декодирование контента;

• SSL верификация;

• аутентификация пользователей на большинстве ресурсов с сохранением;

• поддержка proxy при необходимости;

• загрузка и выгрузка файлов;

• стриминговые загрузки и фрагментированные запросы;

• задержки соединений;

• передача требуемых заголовков на web-ресурсы;

• встроенный json-декодер и многое другое.

Все вышеупомянутые преимущества модуля Requests делают ее более предпочтительной для разработки моей системы автоматизирования, нежели стандартная библиотека urllib.

В качестве среды для разработки я выбрал PyCharm.

PyCharm – это интегрированная среда разработки для языка программирования Python. Предоставляет широкий спектр различных средств разработки, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA. Является кроссплатформенной, совместима с Windows, macOS, Linux. PyCharm Community Edition (бесплатная версия) находится под лицензией Apache License, а PyCharm Professional Edition (платная версия) является проприетарным ПО [2].

Эта среда разработки очень полезна в первую очередь за счёт своих ключевых возможностей [12]:

• Мощный и функциональный редактор кода с подсветкой синтаксиса, авто-форматированием и авто-отступами для поддерживаемых языков.

• Простая и мощная навигация в коде.

• Помощь при написании кода, включающая в себя автодополнение, авто-импорт, шаблоны кода, проверка на совместимость версии интерпретатора языка, и многое другое.

• Быстрый просмотр документации для любого элемента прямо в окне редактора, просмотр внешней документации через браузер, поддержка docstring – генерация, подсветка, автодополнение и многое другое.

• Большое количество инспекций кода.

• Мощный рефакторинг кода, который предоставляет широкие возможности по выполнению быстрых глобальных изменений в проекте.

• Полная поддержка свежих версий Django фреймворка.

• Поддержка Google App Engine.

• Поддержка IronPython, Jython, Cython, PyPy wxPython, PyQt, PyGTK и др.

• Поддержка Flask фреймворка и языков Mako и Jinja2.

• Редактор Javascript, Coffescript, HTML/CSS, SASS, LESS, HAML.

• Интеграция с системами контроля версий (VCS).

• UML диаграммы классов, диаграммы моделей Django и Google App Engine.

• Интегрированное Unit-тестирование.

• Интерактивные консоли для Python, Django, SSH, отладчика и баз данных.

• Полнофункциональный графический отладчик (Debugger).

• Поддержка схем наиболее популярных IDE/редакторов. таких как Netbeans, Eclipse, Emacs, эмуляция VIM редактора.

• Поддерживаемые языки: Python (Versions: 2.x, 3.x), Jython, Cython, IronPython, PyPy, Javascript, CoffeScript, HTML/CSS, Django/Jinja2 templates, Gql, LESS/SASS/SCSS/HAML, Mako, Puppet, RegExp, Rest, SQL, XML, YAML.

• PyCharm имеет несколько цветовых схем, а также настраиваемую подсветку синтаксиса кода.

• Интеграция с баг/issue-треккерами, такими как JIRA, Youtrack, Lighthouse, Pivotal Tracker, GitHub, Redmine, Trac и так далее.

• Огромная, постоянно пополняемая коллекция плагинов.

• Кроссплатформенность (Windows, Mac OS X, Linux).

Помимо всего этого, PyCharm помогает писать красивый код, который легко поддерживать. IDE контролирует качество кода с помощью проверок соответствия требованиям PEP8.

PyCharm создается программистами для программистов, именно поэтому эта IDE подходит лучше всего для разработки на языке программирования Python.

Помимо вышеупомянутых инструментов, я буду использовать Flask – фреймворк для создания веб-приложений на Python, использующий набор инструментов Werkzeug (инструментарий для WSGI – стандартного интерфейса Python между веб-приложениями и различными серверами, предназначен как для разработки, так и развёртывания), а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков – минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности. Он обеспечивает гибкость и является более доступным фреймворком для новых разработчиков, так как позволяет создать веб-приложение быстро, используя только один файл Python. Flask – это расширяемая система, которая не обязывает использовать конкретную структуру директорий и не требует сложного шаблонного кода перед началом использования [3].

Также буду использовать язык гипертекстовой разметки веб-страниц HTML и CSS (формальный язык описания внешнего вида документа, написанного с использованием языка разметки) для их оформления. Выбранная мною интегрированная среда разработки поддерживает данные инструменты.

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].

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.

• 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.

• 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.

# Выводы по главе 1

ГЛАВА 2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ АВТОМАТИЗАЦИИ ПРОЦЕДУРЫ ПУБЛИКАЦИИ РАСПИСАНИЯ СВФУ

# 2.1 О приложении

Требования к ПО

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

ПО будет основано на клиент-серверной архитектуре, то есть состоять из клиентской и серверной частей.

Клиентская сторона, в данном случае, будет представлять собой веб-приложение, которое, отправляя запросы на сервер и получая ответы, будет предоставлять пользователю интерфейс, позволяющий публиковать расписание.

Серверная часть будет принимать запросы от клиентской части, формировать и отправлять ответы на них [5].

Основные функциональные требования к ПО:

* Вход в систему под учетной записью СВФУ;
* Форма для выбора и отправки расписания в формате excel-документа;
* Парсинг excel-документа;
* Парсинг откликов сервера в виде html-кода;
* Логгирование ошибок и исключений;
* Информирование пользователя о результатах публикации расписания.

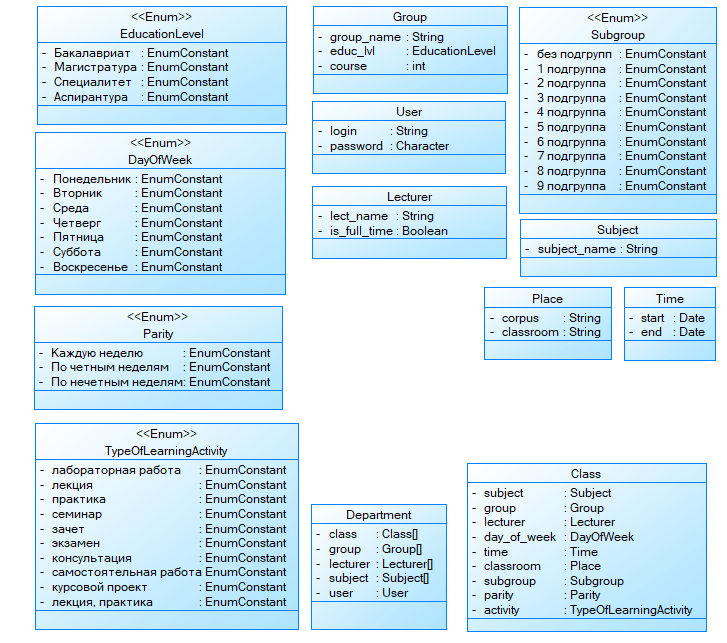
Модель классов расписания

Проанализировав расписание ИМИ СВФУ во всех существующих форматах, я разработал 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.



**Диаграмма прецедентов**

Use Case Diagram (диаграмма прецедентов, она же диаграмма вариантов использования) – диаграмма, отражающая отношения между акторами (участниками) и прецедентами (вариантами) [9].

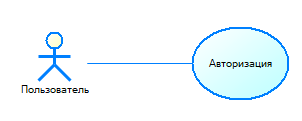
Участник – это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями.

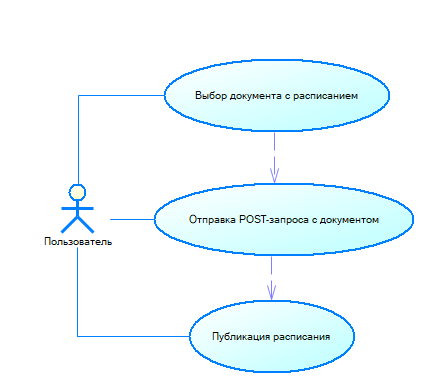
Прецедент (use case) – описание множества последовательных событий, приводящих к наблюдаемому участником результату.

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, отправляется POST-запрос с документом.
4. Отображение расписания. Результат загрузки расписания можно увидеть на сайте СВФУ.





Требования к excel-файлу с расписанием

Для своей работы я взял 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:

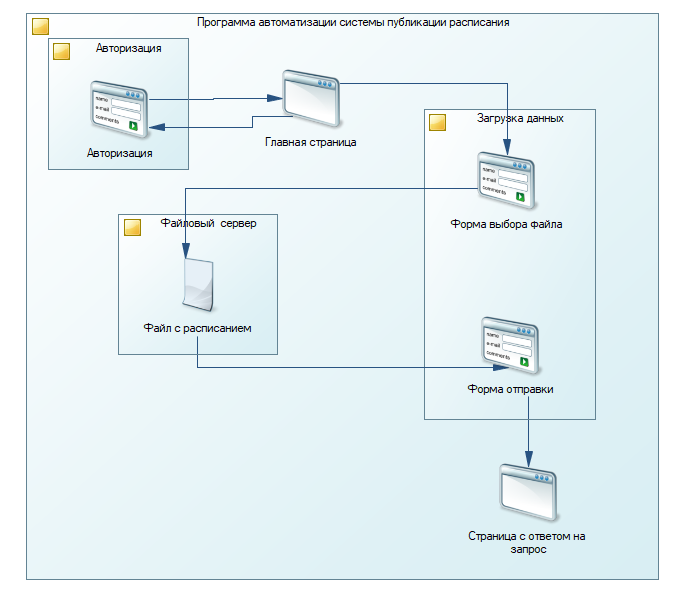
* **Главное требование:** т.к. за основу был взят excel-файла с расписанием занятий БАК и МАГ (за основу был взят файл «ИМИ Расписание учебных занятий на 2 полугодие 2022-2023., от 28.02.2023), то необходимо строго придерживаться структуры данного файал;
* **День недели:** объединенная ячейка на 6 строк (6 пар);
* **Курс и полугодие** должны быть арабскими цифрами;
* **В качестве разделителя** должен выступать “\” или “/”;

# 2.2 Архитектура ПО

Клиент-серверная архитектура

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 [6].

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



Описание API

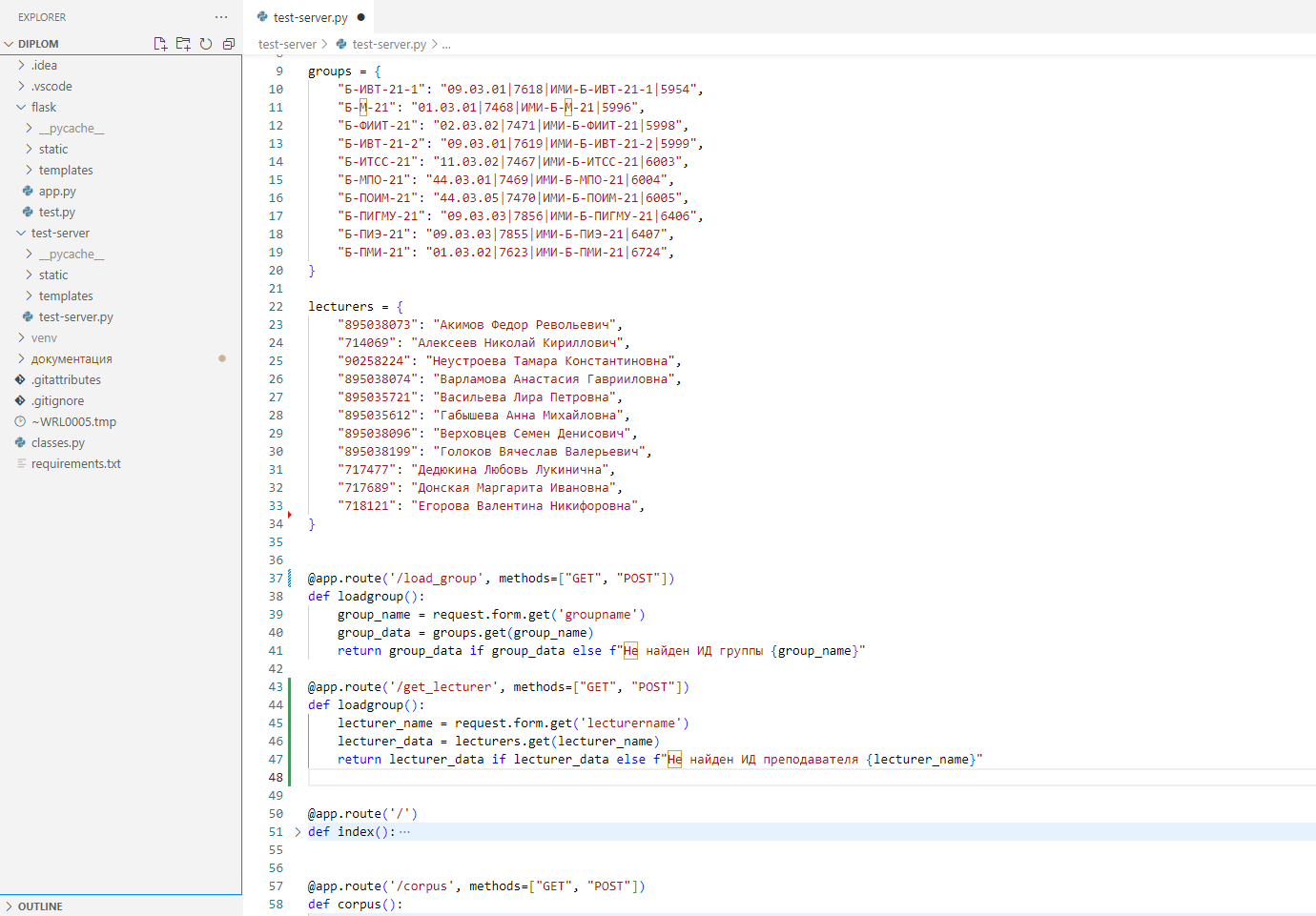
# 2.3 Описание тестового сервера

Прежде чем работать с production-сервером СВФУ, необходимо было провести тестирование приложения на тестовом контуре. Для этого я разработал тестовый сервер (заглушку), призванный имитировать работу «боевоего» сервера.

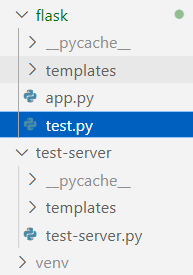
Он содержит в себе структуры данных, похожие на то, что в ответе на запросы возвращает сервер СВФУ.

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

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



# 2.4 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.

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 системе.  
   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.
2. @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.   
   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 авторизациии форма для прикрепления файла становится доступной для заполнения. Заполненная форма передается на следующий запрос.
3. @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.

# 2.5 Апробация

# Выводы по главе 2

# ЗАКЛЮЧЕНИЕ

Таким образом, в ходе курсовой 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.

Для достижения поставленных целей были поставлены следующие задачи:

1. изучение и анализ расписания ИМИ;

2. анализ предметной области;

3. анализ системы публикации расписания ИМИ;

4. анализ запросов;

5. обзор библиотек и инструментов разработки;

6. проектирование модели классов расписания;

7. реализация модели классов расписания на Python;

8. проектирование прототипа программы автоматизации.

Все поставленные задачи были выполнены.

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

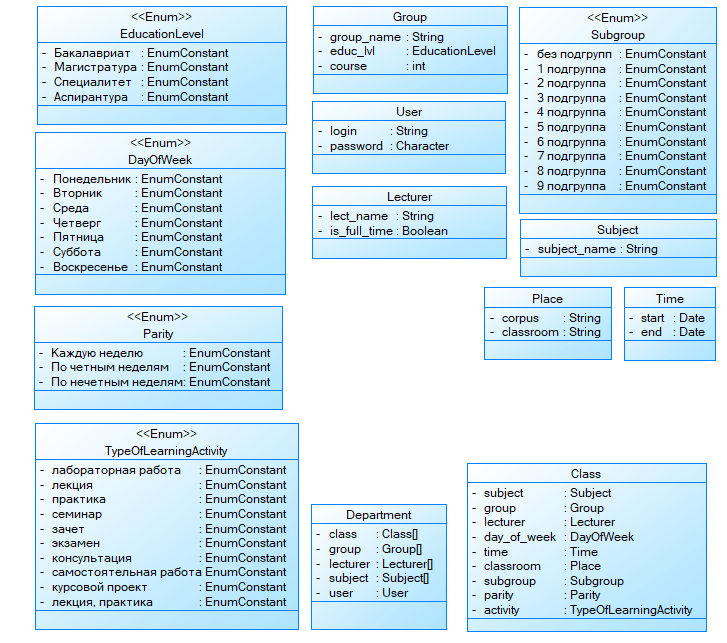
Помимо этого, я считаю, что разработанную мной систему автоматизации смогут применять и другие учебные подразделения, если будут строго придерживаться требований к формату документа с расписанием.

# СПИСОК ЛИТЕРАТУРЫ И ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Kumar, P. Python urllib [Электронный ресурс]. URL: <https://www.journaldev.com/20795/python-urllib-python-3-urllib> (дата обращения: 25.05.2022)
2. PyCharm – IDE для профессиональной разработки на Python [Электронный ресурс]. URL: <https://www.jetbrains.com/ru-ru/pycharm/> (дата обращения: 01.06.2022)
3. Welcome to Flask – Flask Documentation (2.1.x) [Электронный ресурс]. URL: <https://flask.palletsprojects.com/en/2.1.x/> (дата обращения: 25.05.2022)
4. 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: 24.05.2022)
5. 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: 10.06.2022)
6. 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: 09.06.2022)
7. Библиотека Requests в Python [Электронный ресурс]. URL: https://dvmn.org/encyclopedia/modules/requests/ (дата обращения: 24.05.2022)
8. Библиотека Requests: HTTP for Humans [Электронный ресурс]. URL: https://python.ru/post/97/ (дата обращения: 25.05.2022)
9. Диаграмма вариантов использования (Use Case Diagram) [Электронный ресурс]. URL: <https://flexberry.github.io/ru/fd_use-case-diagram.html> (дата обращения: 31.05.2022)
10. 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: 26.05.2022)
11. 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: 09.06.2022)
12. Ломцев, И. PyCharm: IDE для Python [Электронный ресурс]. URL: [https://timeweb.com/ru/community/articles/pycharm-ide-dlya-python-1#](https://timeweb.com/ru/community/articles/pycharm-ide-dlya-python-1) (дата обращения: 01.06.2022)
13. Макарик, М. Библиотека Requests: эффективные и простые HTTP-запросы в Python [Электронный ресурс]. URL: <https://smartiqa.ru/blog/python-requests> (дата обращения: 24.05.2022)
14. Методики составления расписания занятий [Электронный ресурс]. URL: [https://sensystem.ru/kompania/blog/tsifrovaya-shkola/metody-sostavleniya-raspisaniya-zanyatij](https://sensystem.ru/kompania/blog/tsifrovaya-shkola/metody-sostavleniya-raspisaniya-zanyatij%20) (дата обращения: 26.05.2022)
15. Руководство по работе с HTTP в Python [Электронный ресурс]. URL: [https://cloud.timeweb.com/blog/vvedenie-v-rabotu-s-bibliotekoj-requests-v-python](https://cloud.timeweb.com/blog/vvedenie-v-rabotu-s-bibliotekoj-requests-v-python%20) (дата обращения: 24.05.2022)
16. Что такое API? [Электронный ресурс]. URL: <https://aws.amazon.com/ru/what-is/api/> (дата обращения: 01.06.2022)

# ПРИЛОЖЕНИЯ

Приложение 1. Модель классов расписания



Приложение 2. 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 4. Исходный код прототипа