Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Северо-Восточный федеральный университет им. М.К. Аммосова»

Институт математики и информатики

Кафедра «Информационные технологии»

РАЗРАБОТКА ПРОГРАММЫ АВТОМАТИЗАЦИИ ПРОЦЕССА ПУБЛИКАЦИИ УЧЕБНОГО РАСПИСАНИЯ

КУРСОВАЯ РАБОТА

Направление подготовки: 02.03.02 «Фундаментальная информатика и информационные технологии»

Выполнил: студент III курса  
группы Б-ФИИТ-19 ИМИ СВФУ  
Собянин Сергей Павлович

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

(подпись)

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

(подпись)

Якутск 2022

Содержание

[Введение 3](#_Toc106141556)

[1. РАСПИСАНИЕ ЗАНЯТИЙ В ИМИ 5](#_Toc106141557)

[1.1 Составления расписания в ИМИ 5](#_Toc106141558)

[1.2 Описание системы публикации расписания УСП СВФУ им. М. К. Аммосова 6](#_Toc106141559)

[1.3 Обзор инструментов разработки 12](#_Toc106141560)

[2. ПРОЕКТИРОВАНИЕ ПРОГРАММЫ АВТОМАТИЗАЦИИ 20](#_Toc106141561)

[2.1 Требования к ПО 20](#_Toc106141562)

[2.2 Модель классов расписания 20](#_Toc106141563)

[2.4 Архитектура ПО 23](#_Toc106141564)

[2.5 Описание прототипа 24](#_Toc106141565)

[Заключение 28](#_Toc106141566)

[Список использованной литературы 29](#_Toc106141567)

[Приложения 31](#_Toc106141568)

[Приложение 1. Модель классов расписания 31](#_Toc106141569)

[Приложение 2. Реализация модели классов на Python 32](#_Toc106141570)

[Приложение 3. Структура проекта 34](#_Toc106141571)

[Приложение 4. Исходный код прототипа 35](#_Toc106141572)

# **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 использовании 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.

Из всего вышесказанного становится понятно, что автоматизированный способ является наиболее эффективным и удобным.

После того, как расписание составлено и откорректировано, его необходимо опубликовать. Если опять-таки использовать ручной метод, то придется распечатать огромное количество листов А4, а если еще и понадобится внести какие-либо изменения или исправления, то это будет весьма неудобно. На такой случай существуют различные электронные визуализаторы. С их помощью можно отказаться от бумажных носителей, а корректировка расписания займет совсем небольшое количество времени.

Таким образом, целью моей курсовой работы является разработка прототипа программы автоматизации процесса публикации учебного расписания института математики и информатика Северо-Восточного федерального университета им. М. К. Аммосова. Для достижения поставленной цели были определены 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.

Структурно, курсовая работа состоит из введения, двух глав, заключения, списка использованной литературы и приложения.

# 1. РАСПИСАНИЕ ЗАНЯТИЙ В ИМИ

# **1.1 Составления расписания в ИМИ**

На сегодняшний день основным источником учебного расписания для студентов ИМИ является Excel-файл. 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.

# **1.2 Описание системы публикации расписания УСП СВФУ им. М. К. Аммосова**

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 он сразу может выбрать свой институт, после чего будет перенаправлен на другую страницу, 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.

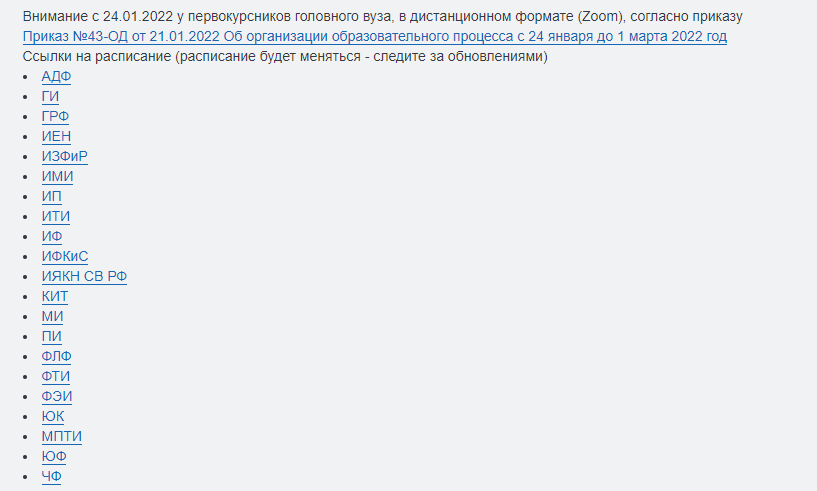


Рисунок 1 – Список с УЧП в виде ссылок

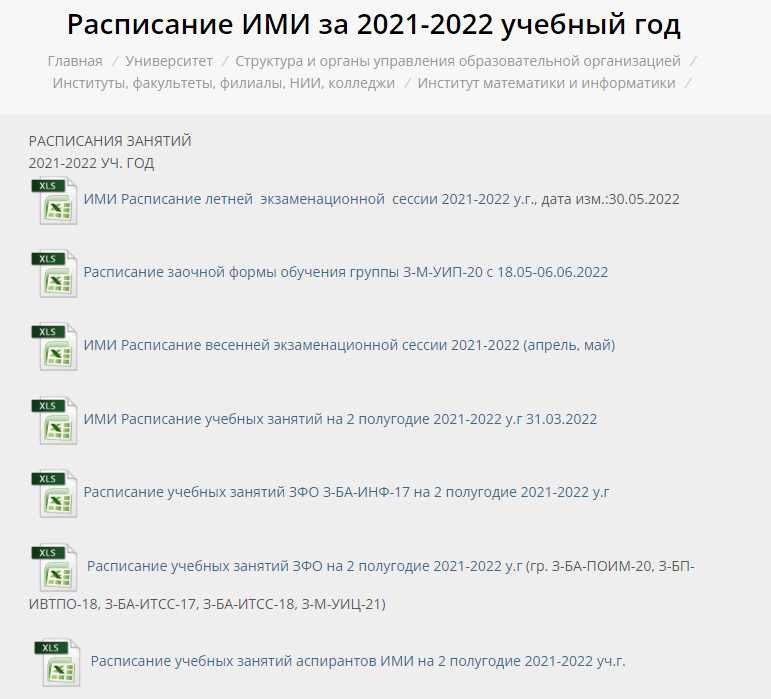


Рисунок 2 – Веб-страница с Excel-файлами расписания

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.

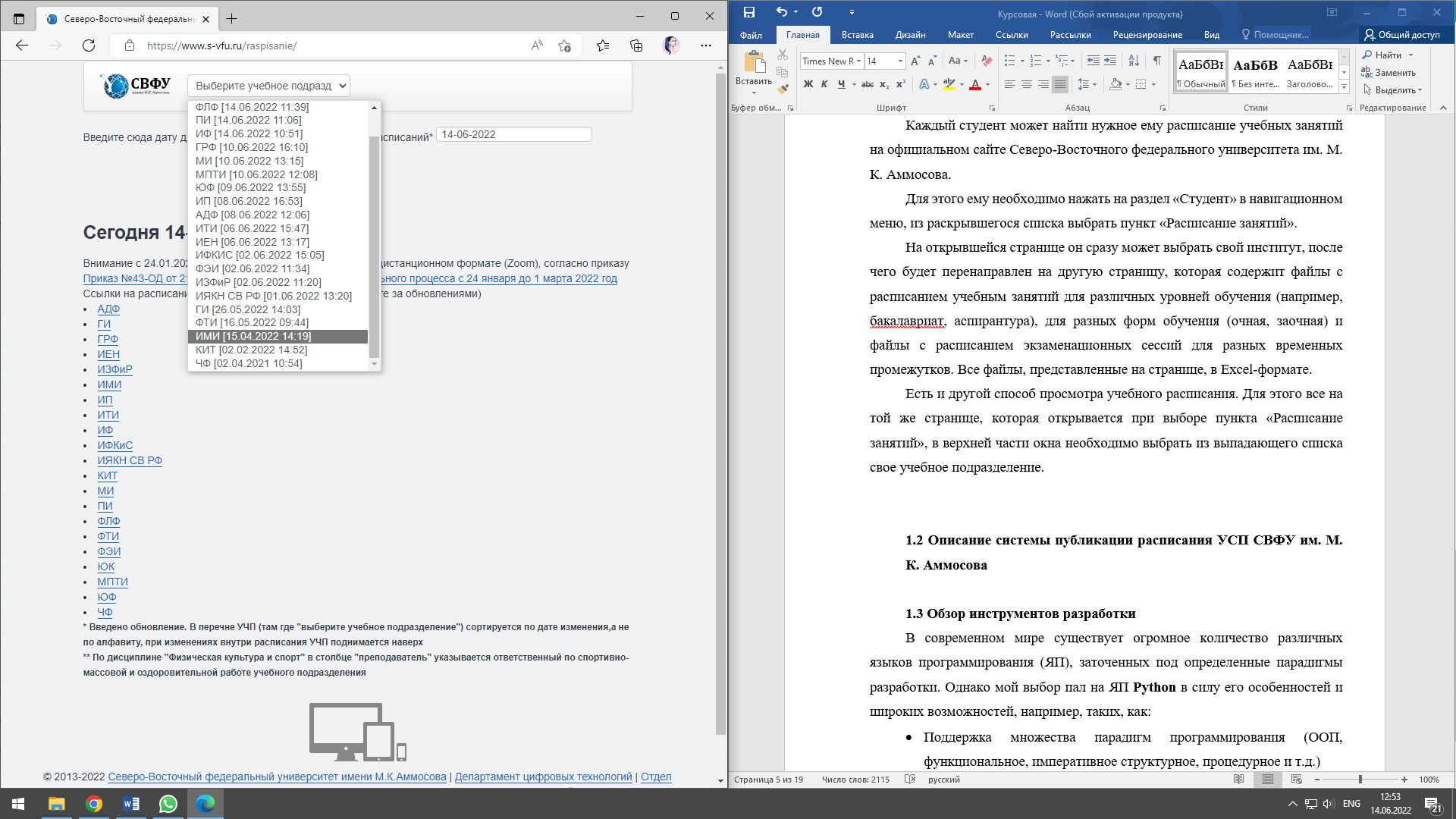


Рисунок 3.1 – Выбор УЧП

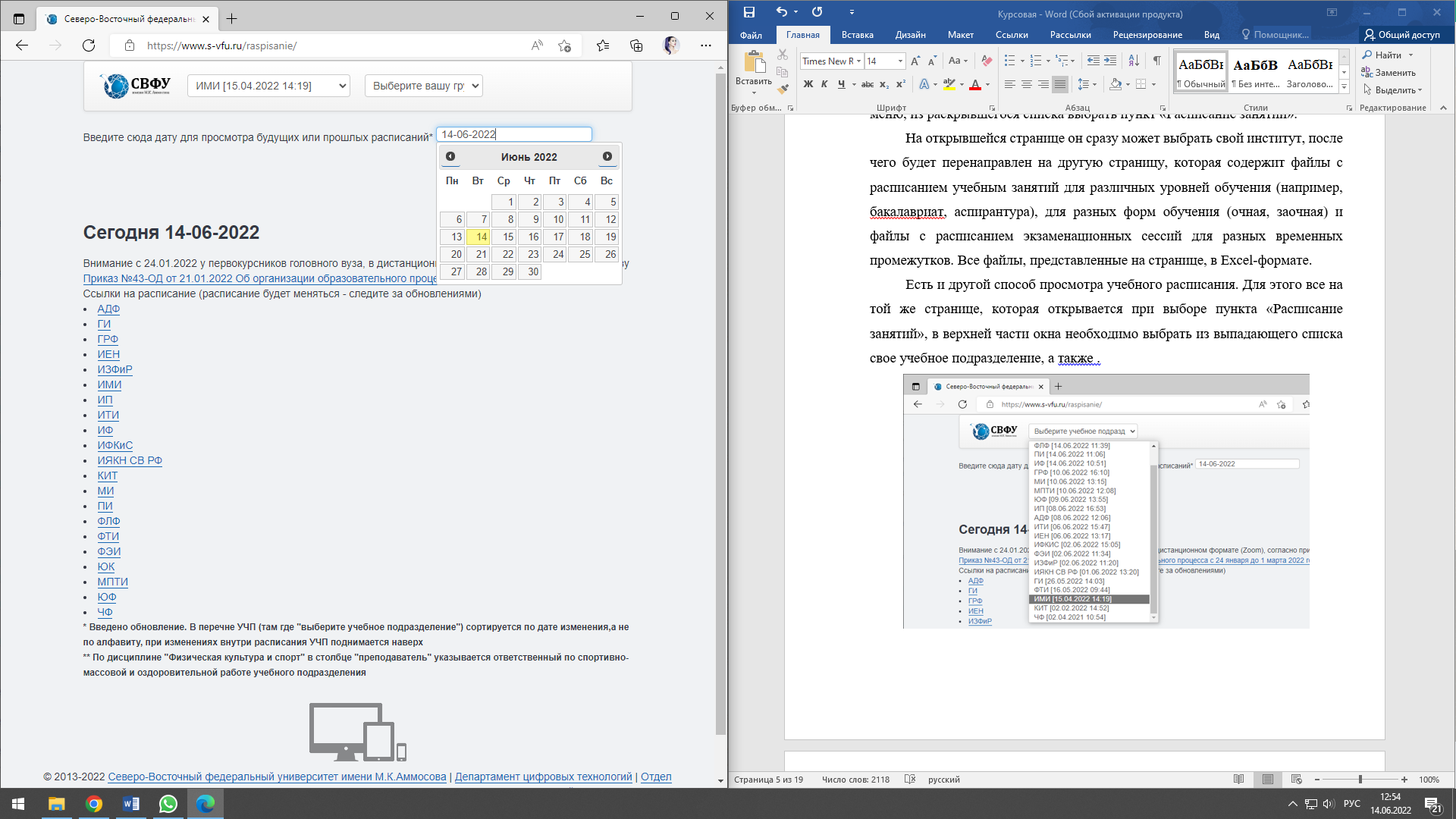


Рисунок 3.2 – Выбор даты

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

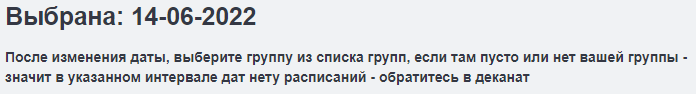


Рисунок 3.3 – Предупреждение с веб-страницы расписания

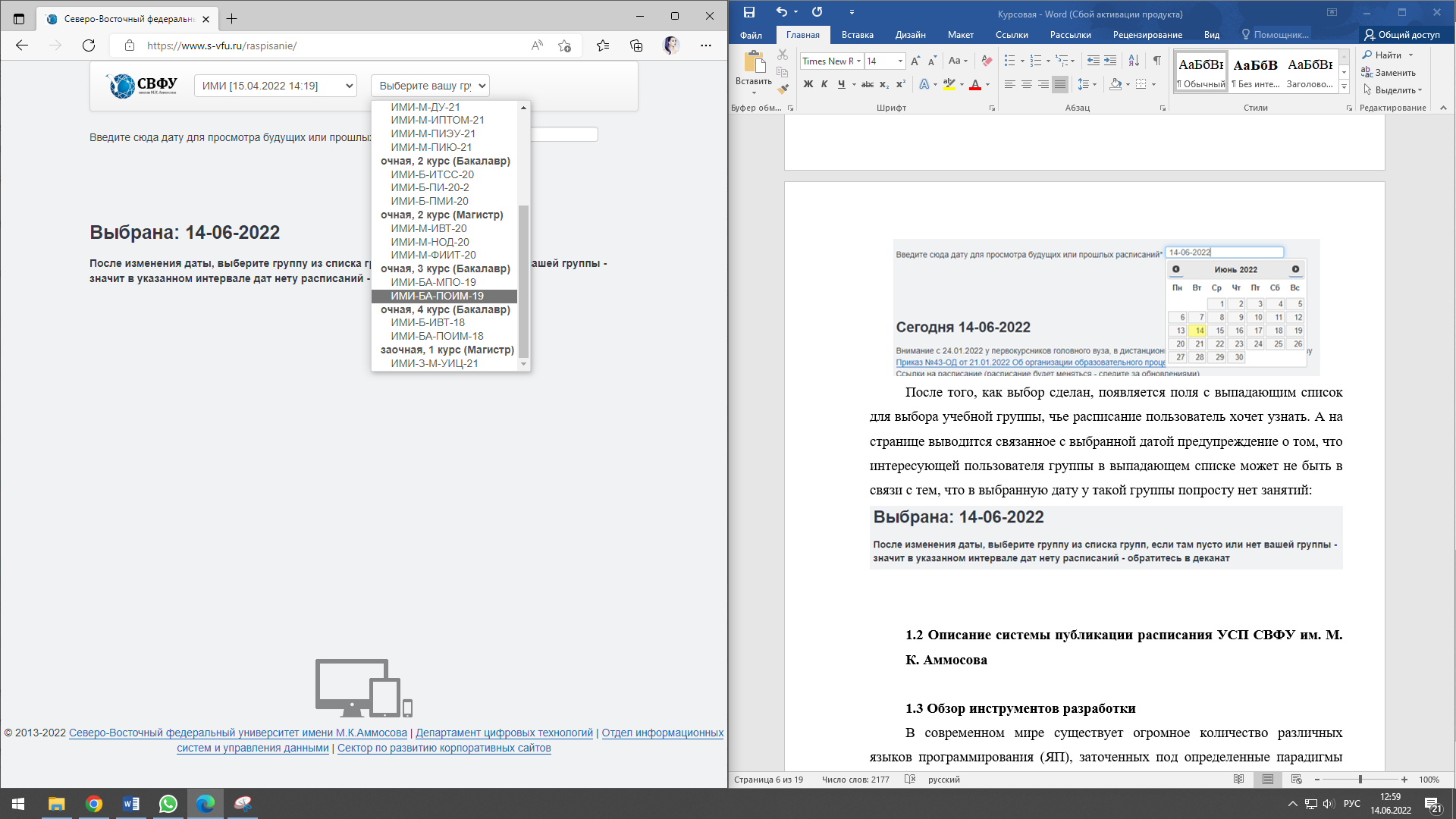


Рисунок 3.4 – Выбор учебной группы

После выбора учебной группы, на странице отображается расписание на всю неделю, включающую в себя выбранную ранее дате. 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 и дата и время последнего изменения расписания.

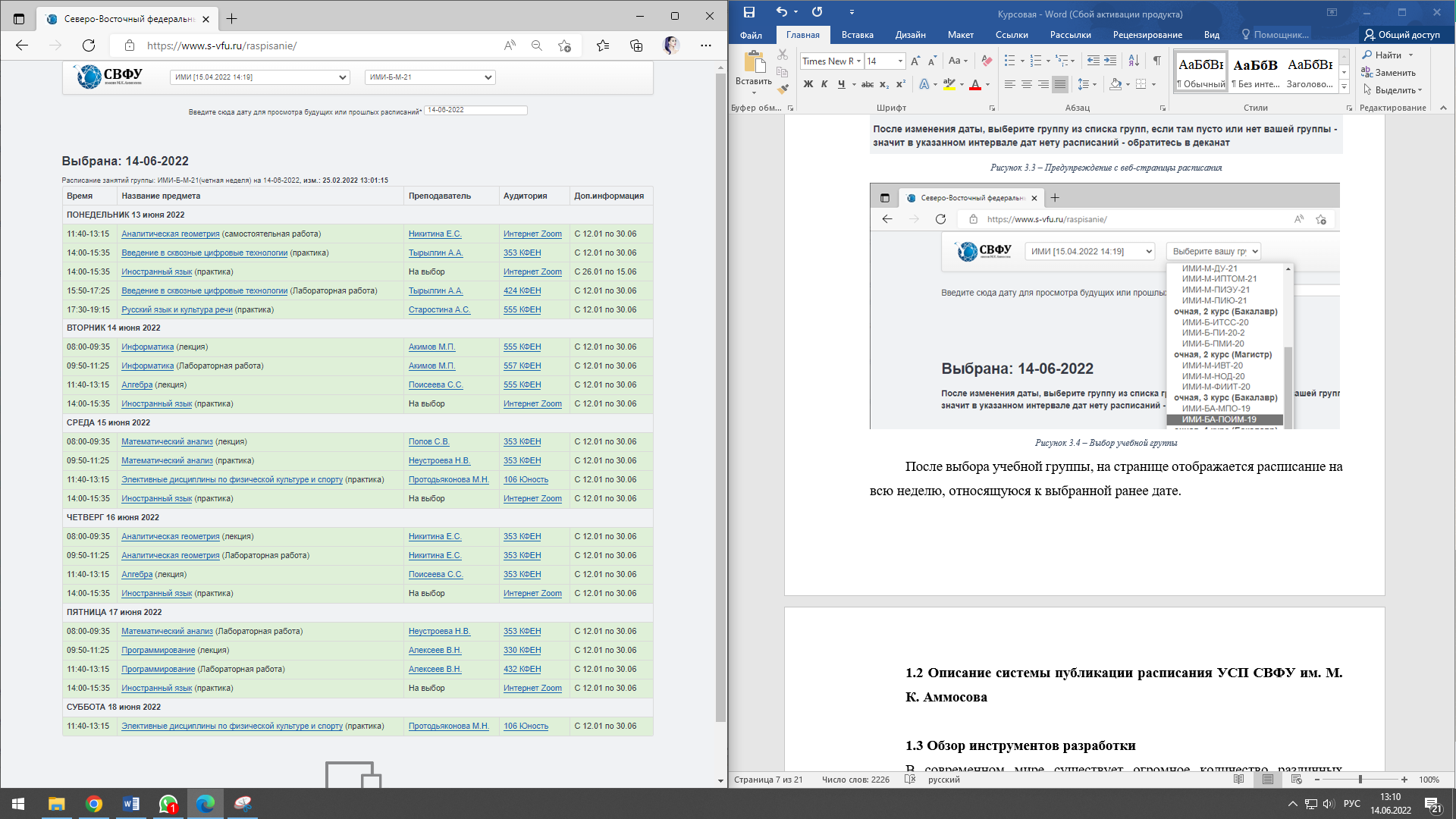


Рисунок 5 – Отображение расписания

На данный момент расписание в ИМИ 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, 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»;
3. 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;
   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.

После заполнения всех полей и нажатия на кнопку «Шаг1 Выбрать группы и план», 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.

При выборе группы и плана и нажатии на кнопку «Подобрать РУП» появляется модальное окно, содержащее чекбокс для дополнительного отображения предыдущего и следующего семестров. 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».

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.

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».

Публикация также осуществляется с помощью POST-запроса, отправленного по адресу «https://www.s-vfu.ru/user/rasp/new/».

# **1.3 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 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.

# 2. ПРОЕКТИРОВАНИЕ ПРОГРАММЫ АВТОМАТИЗАЦИИ

# **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.

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 требования к ПО:

* Форма для выбора и загрузки расписания в формате Excel-файла;
* Обработка Excel-файла программой;
* Формирование ответа;
* Отображение расписания;

# **2.2 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.

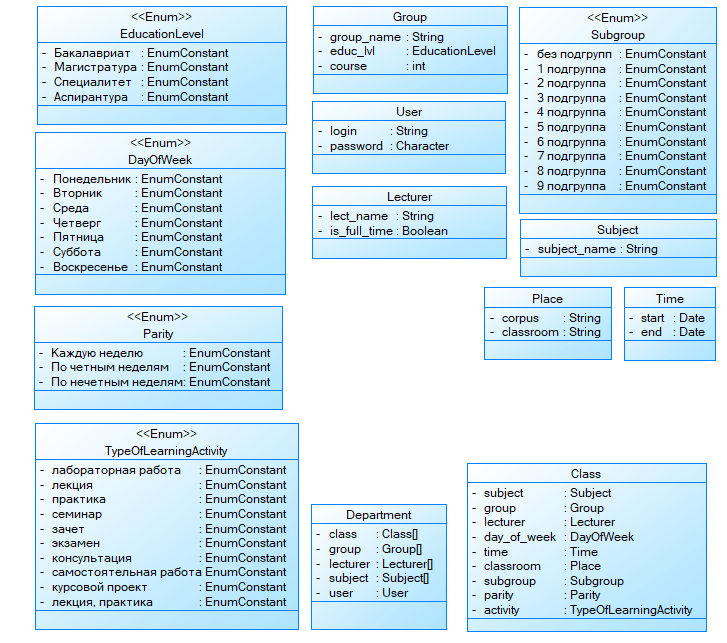


Рисунок 6 – Модель классов расписания

* 1. **Use Case Diagram**

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 выводится на странице отображения данных.

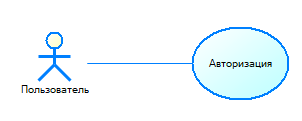


Рисунок 7 — Диаграмма прецедентов. Авторизация

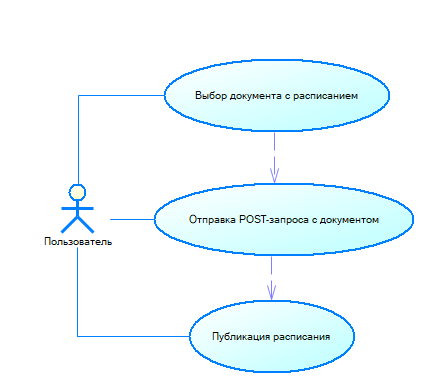


Рисунок 8 — Диаграмма прецедентов. Публикация расписания

# **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 [6].

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

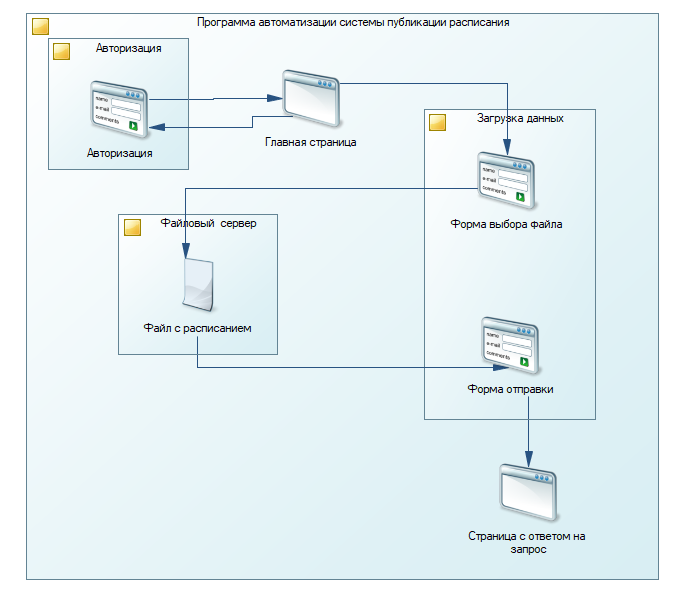


Рисунок 9 — Архитектура ПО автоматизации системы публикации расписания

# **2.5 Описание прототипа**

Так как на данный момент я не имею доступа к системе публикации расписания УСП СВФУ, разработанный мной прототип осуществляет лишь авторизацию в система университета путем POST HTTP-запроса и с использованием сессий.

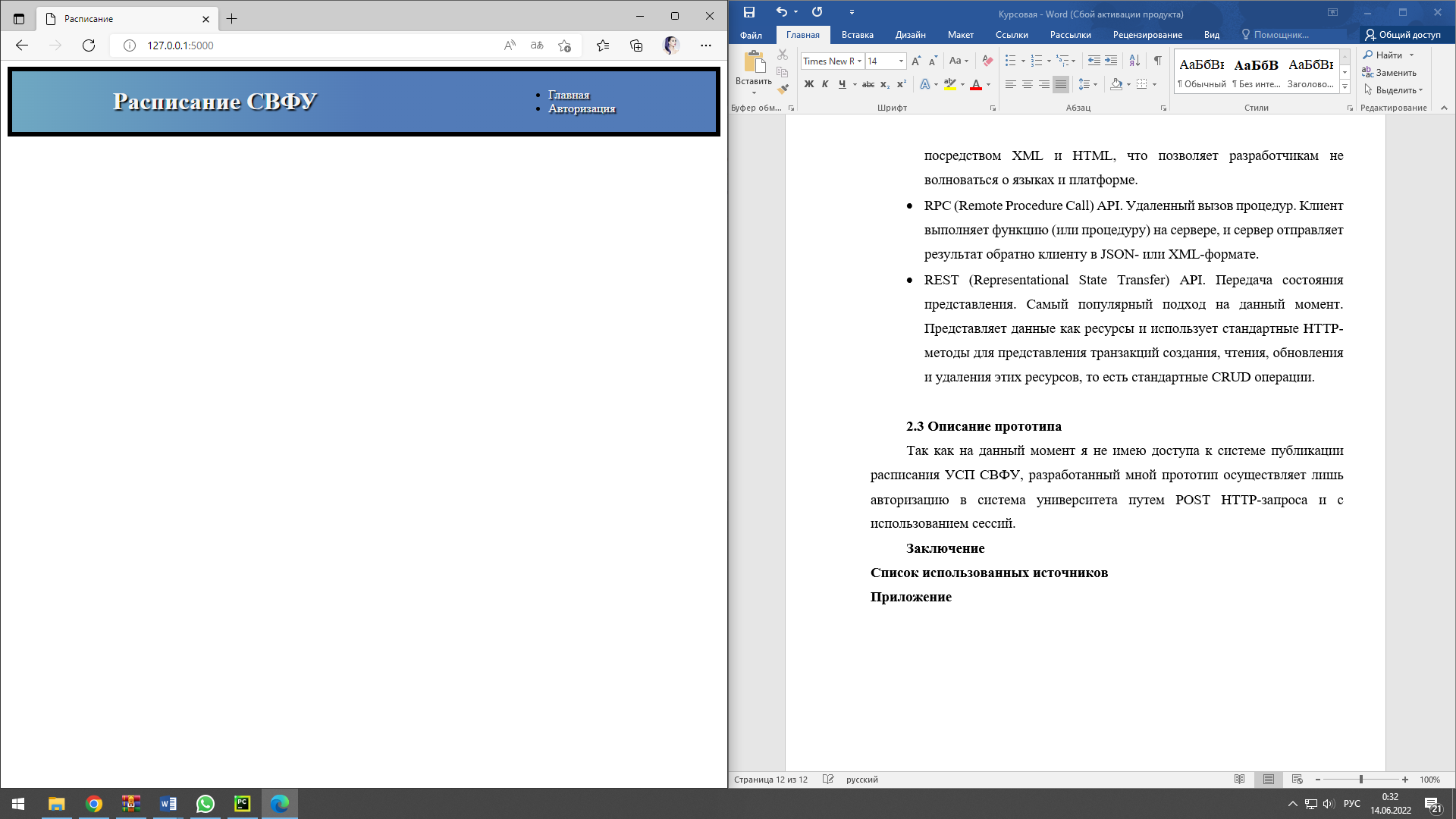


Рисунок 10. Главная страница прототипа веб-приложения

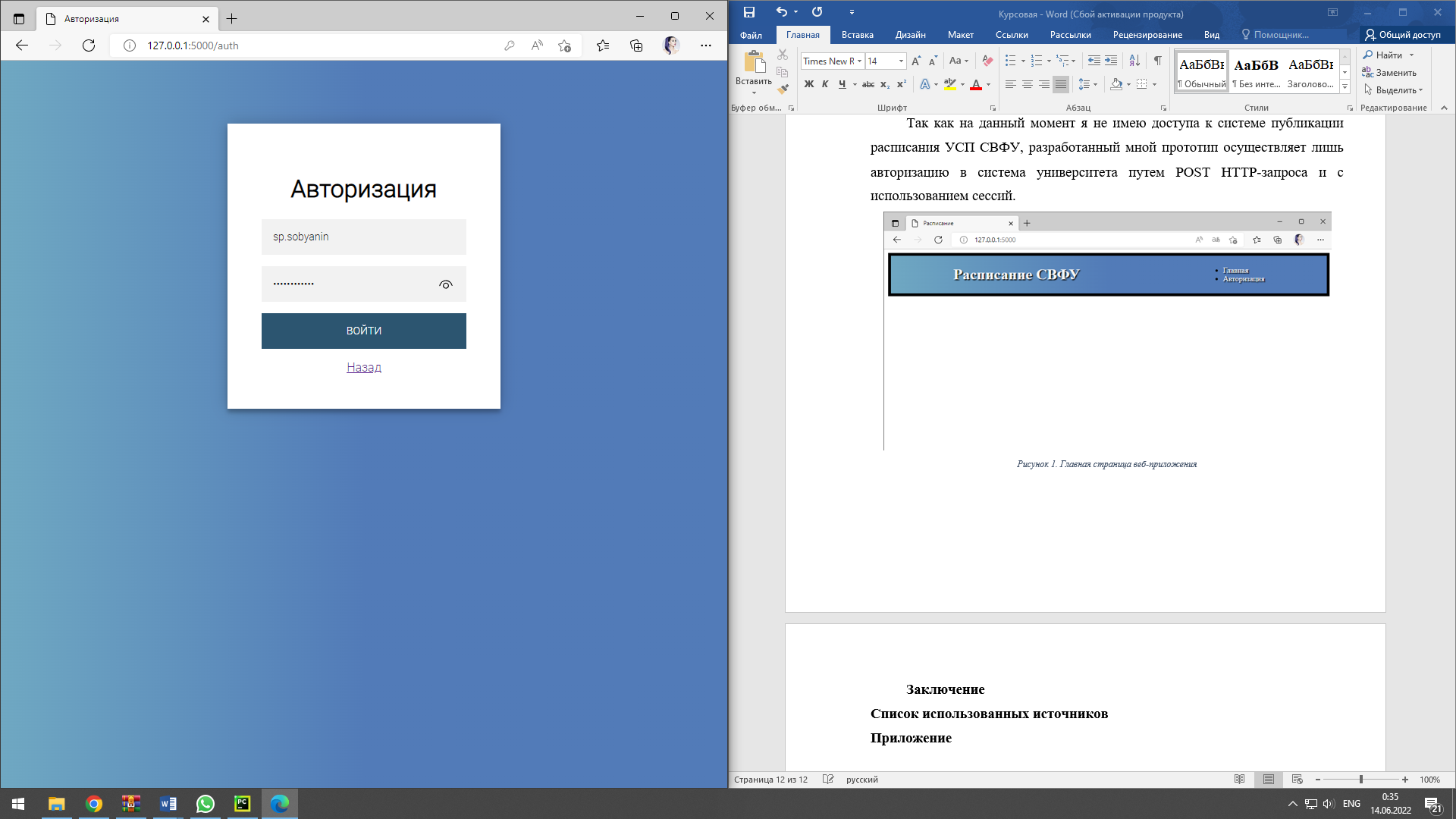


Рисунок 11. Форма авторизации в системе

При правильном вводе логина и пароля, приложение проводит успешную авторизацию при помощи запроса POST-методом и запущенной сессии. Программа получает ответ и записывает его в переменную, передавая на следующую страницу.

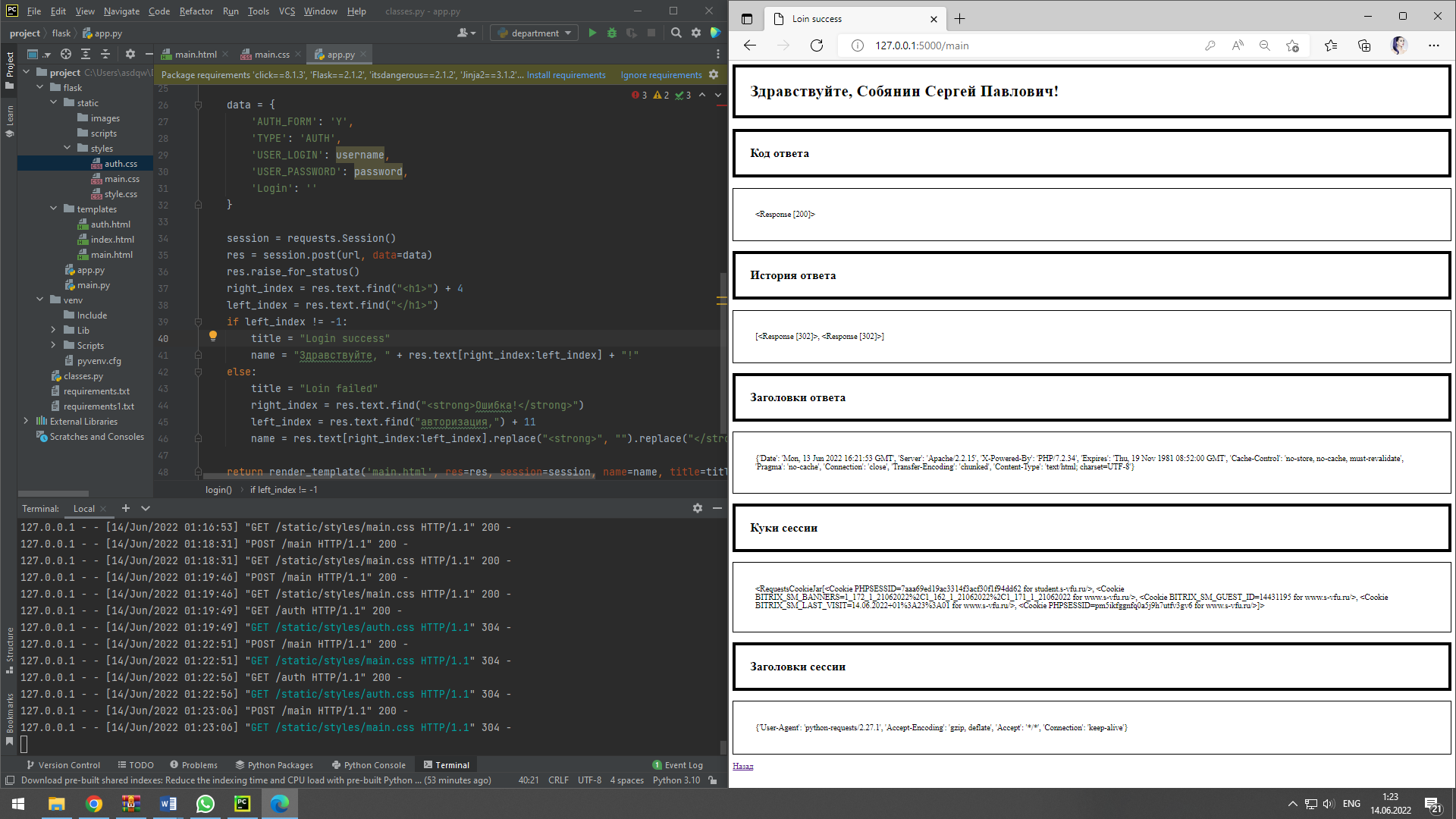


Рисунок 12 — Успешная авторизация

Ответ на запрос содержит большое количество различной информации, например, можно посмотреть код и историю ответов на запрос, заголовки ответа, куки и заголовки сессии и многое другое. Так, например, у меня есть доступ к HTML-код полученной в ответе страницы, с помощью которого я могу получить имя того, в чей личный кабинет я авторизовался, что продемонстрировано в самом верху рисунка 3.1.

При неправильном вводе логина и/или пароля мы имеем доступ к информации страницы, которая загружается при неудавшейся авторизации.

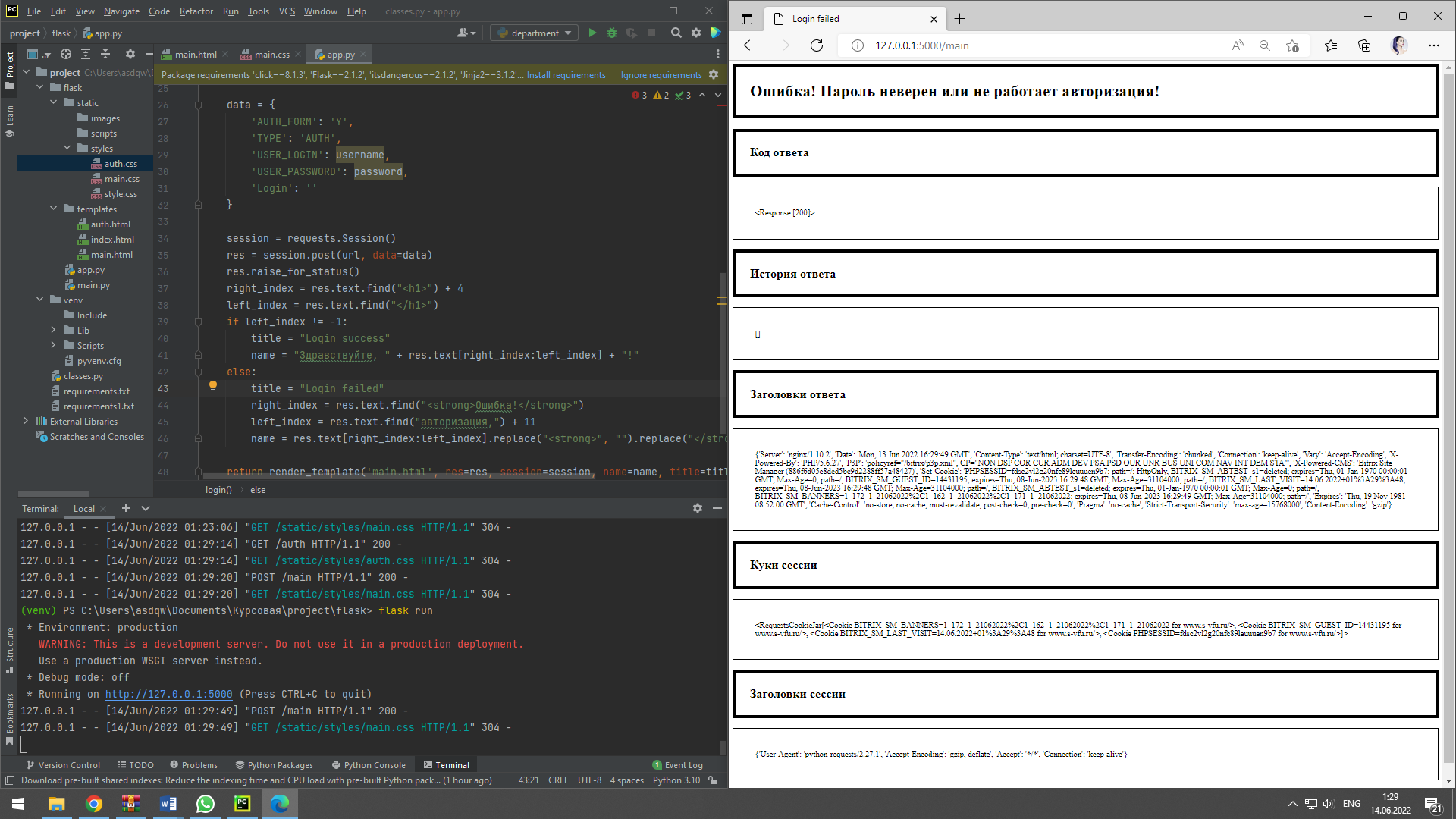


Рисунок 13 — Неудачная попытка авторизации

Несмотря на неверные данные, код ответа все равно 200, говорящий о том, что запрос успешный. Это потому, что хоть данные и были неправильные, сервер вернул в ответ существующую страницу, на которой просто выдается сообщение об ошибке (самый первый заголовок на рисунке 4.1., сообщение взято из HTML-кода ответа на запрос). К тому же, история запросов отсутствует, так как не происходит перенаправление с сайта www.s-vfu.ru на [www.student.s-vfu.ru](http://www.student.s-vfu.ru) из-за некорректных логина и/или пароля.

# **Заключение**

Таким образом, в ходе курсовой работы были проанализированы составление и публикация расписания учебных занятий ИМИ. 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 ходе написания курсового проекта были закреплены навыки программирования на Python и навыки работы с такими библиотеками и фреймворками этого языка программирования, как Requests и Flask.

В дальнейшем планируется разработка полноценного программного обеспечения для автоматизации системы публикации расписания ИМИ СВФУ на основе уже разработанного прототипа, а также внедрение и последующая поддержка.

Помимо этого, я считаю, что разработанную мной в будущем систему автоматизации смогут применять и другие учебные подразделения при должной доработке.

# **Список использованной литературы**

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

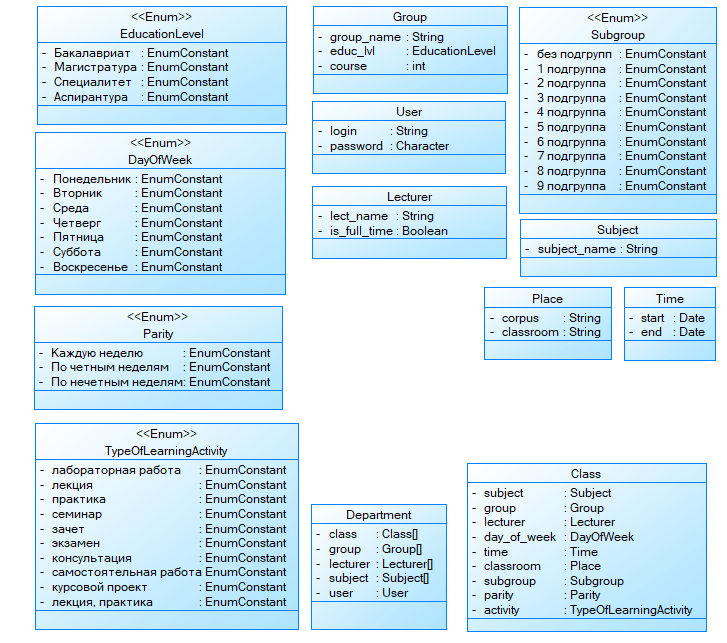


Рисунок 14 – Модель классов расписания

# Приложение 2. Реализация модели классов на Python

class Place:  
  
 def \_\_init\_\_(self, classroom, corpus):  
 self.classroom = classroom  
 self.corpus = corpus  
  
  
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, is\_full\_time):  
 self.name = name  
 self.is\_full\_time = is\_full\_time  
  
  
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 User:  
  
 def \_\_init\_\_(self, login, password):  
 self.login = login  
 self.password = password  
  
  
class Class:  
  
 def \_\_init\_\_(self, subject, group, lecturer, day, time, place,  
 subgroup, parity, activity):  
 self.subject = subject  
 self.group = group  
 self.lecturer = lecturer  
 self.day = day  
 self.time = time  
 self.place = place  
 self.subgroup = subgroup  
 self.parity = parity  
 self.activity = activity  
  
  
class User:  
  
 def \_\_init\_\_(self, login, password):  
 self.login = login  
 self.password = password  
  
  
class Department:  
  
 def \_\_init\_\_(self, user):  
 self.groups = []  
 self.classes = []  
 self.lectures = []  
 self.subjects = []  
 self.user = user  
  
  
EducationLevel = (  
 (1, 'бакалвриат'),  
 (2, 'специалитет'),  
 (3, 'магистратура'),  
 (4, 'аспирантура'),  
)  
  
TypeOfLearningActivity = (  
 (0, ''),  
 (1, 'практика'),  
 (2, 'лекция'),  
 (3, 'семинар'),  
 (4, 'лабораторная работа'),  
 (5, 'зачет'),  
 (6, 'экзамен'),  
 (7, 'консультация'),  
 (8, 'самостоятельная работа'),  
 (9, 'курсовой проект'),  
 (10, 'лекция, практика'),  
)  
  
Subgroup = (  
 (0, 'без подгрупп'),  
 (1, '1 подгруппа'),  
 (2, '2 подгруппа'),  
 (3, '3 подгруппа'),  
 (4, '4 подгруппа'),  
 (5, '5 подгруппа'),  
 (6, '6 подгруппа'),  
 (7, '7 подгруппа'),  
 (8, '8 подгруппа'),  
 (9, '9 подгруппа'),  
)  
  
DayOfWeek = (  
 (1, 'понедельник'),  
 (2, 'вторник'),  
 (3, 'среда'),  
 (4, 'четверг'),  
 (5, 'пятница'),  
 (6, 'суббота'),  
 (7, 'воскресенье')  
)  
  
Parity = (  
 (0, ''),  
 (1, 'нечетная'),  
 (2, 'четная'),  
)

# Приложение 3. Структура проекта

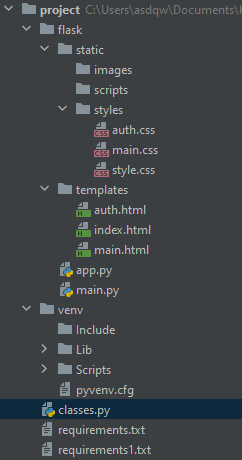


Рисунок 15 – Структура проекта

# Приложение 4. Исходный код прототипа

from flask import Flask, render\_template, request  
import requests  
  
  
app = Flask(\_\_name\_\_)  
  
  
@app.route('/')  
def index():  
 return render\_template('index.html')  
  
  
@app.route('/auth')  
def auth():  
 return render\_template('auth.html')  
  
  
@app.route('/main', methods=['post', 'get'])  
def login():  
 if request.method == 'POST':  
 username = request.form.get('username')  
 password = request.form.get('password')  
   
 url = 'https://www.s-vfu.ru/?login'  
  
 data = {  
 'AUTH\_FORM': 'Y',  
 'TYPE': 'AUTH',  
 'USER\_LOGIN': username,  
 'USER\_PASSWORD': password,  
 'Login': ''  
 }  
  
 session = requests.Session()  
 res = session.post(url, data=data)  
 res.raise\_for\_status()  
 right\_index = res.text.find("<h1>") + 4  
 left\_index = res.text.find("</h1>")  
 if left\_index != -1:  
 title = "Login success"  
 name = "Здравствуйте, " + res.text[right\_index:left\_index] + "!"  
 else:  
 title = "Login failed"  
 right\_index = res.text.find("<strong>Ошибка!</strong>")  
 left\_index = res.text.find("авторизация,") + 11  
 name = res.text[right\_index:left\_index].replace("<strong>", "").replace("</strong> <br>", "") + "!"  
  
 return render\_template('main.html', res=res, session=session, name=name, title=title)