МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных наук

Кафедра программирования и информационных технологий

Разработка веб-приложения для переноса расписания занятий факультета компьютерных наук в сервис «Google Calendar»

Курсовая работа

09.03.03 Информационные системы и технологии

Программирование и информационные технологии

Допущено к защите в ГЭК

Зав. кафедрой _____С.Д. Махортов, д. т. н., профессор __.__.2023

Обучающийся _____Д.А. Змаев, 3 курс, д/о

Руководитель _____П.С. Лысачёв, ст. преподаватель

Воронеж 2023

Содержание

Введение

В настоящее время, когда все больше процессов становятся автоматизированными и современные технологии позволяют оптимизировать многие аспекты жизни человека, важно не оставаться в стороне от прогресса. В области образования и учебного процесса также есть много возможностей для автоматизации. Одним из примеров является использование сервиса «Google Calendar» для отслеживания расписания занятий в университете.

В рамках курсовой работы будет проведен анализ возможностей «Google Calendar» API. Будет разработана архитектура приложения, которая будет учитывать специфику расписания занятий факультета компьютерных наук. Далее, будет разработано приложение, обеспечивающее автоматический перенос расписания занятий в сервис «Google Calendar». Наконец, приложение будет протестировано и документировано.

В результате выполнения данной курсовой работы студенты факультета компьютерных наук смогут получать доступ к своему расписанию занятий в календаре «Google Calendar», что может облегчить организацию своего времени.

1 Постановка задачи

Цель курсовой работы - Разработка веб-приложения для переноса расписания занятий факультета компьютерных наук в сервис «Google Calendar».

1.1 Цель создания системы

Целью данной курсовой работы является разработка веб-приложения, которое будет автоматически переносить расписание занятий факультета компьютерных наук в сервис «Google Calendar».

1.2 Требования к создаваемой системе

Разрабатываемое веб-приложение должно иметь следующие возможности:

- предоставление пользователю возможности выбора расписания в зависимости от его курса, группы и подгруппы;
- реализация аккаунта администратора для выгрузки расписания в случае его изменения;
- предоставление пользователю возможности выбора срока заполнения расписания;

1.3 Задачи, решаемые в процессе разработки

В ходе разработки системы определены следующие задачи:

- спроектировать систему с учетом полученной в ходе анализа информации;
- изучить и реализовать взаимодействие с внешним ресурсом «Google Calendar API»;
- разработать пользовательский веб-интерфейс;
- описать процесс разработки и результат.

2 Анализ предметной области

2.1 Терминология (глоссарий) предметной области

Веб-приложение – это программа, которая запускается в браузере и использует веб-технологии для выполнения своих функций.

Парсинг — это процесс извлечения данных из структурированного или источника, такого как веб-страница или файл.

Валидация — это процесс проверки правильности и соответствия данных определенным критериям или правилам. В веб-разработке, валидация часто используется для проверки данных, введенных пользователем в форму на веб-странице, например, при регистрации аккаунта или заполнении заказа. Валидация может проверять формат, длину или тип данных, а также применять другие правила, определенные разработчиком. Цель валидации - обеспечить корректность и целостность данных, предотвратить ошибки ввода и улучшить пользовательский опыт.

Событие Google Calendar — это запись в календаре, которая представляет собой определенное событие, происходящее в определенное время.

Google токен — это уникальный код, который используется для аутентификации и авторизации пользователя в приложениях и сервисах Google. Токен представляет собой строку символов, которая выдается после успешной аутентификации пользователя в Google-аккаунте и может использоваться для доступа к определенным ресурсам или функциям API Google.

2.2 Анализ существующих решений

2.2.1 Электронный университет ВГУ

Электронный университет ВГУ (Воронежский государственный университет) — это образовательная платформа, которая была создана с

целью обеспечения доступа к образовательным ресурсам и услугам ВГУ через Интернет.

Электронный университет ВГУ предоставляет студентам и преподавателям ВГУ возможность получения доступа к онлайн-курсам, видеолекциям, учебным материалам, тестам и другим образовательным ресурсам. Это позволяет значительно упростить процесс обучения и повысить его эффективность.

У электронного университета ВГУ есть несколько основных направлений, которые включают в себя:

— дистанционное образование — это форма обучения, при которой студенты могут получать знания и навыки, не покидая свой дом. Это особенно удобно для тех, кто не может по каким-либо причинам посещать очные занятия;

— онлайн-курсы — это курсы, которые можно проходить полностью онлайн. Они могут быть как бесплатными, так и платными. Онлайн-курсы могут быть как для студентов ВГУ, так и для всех желающих;

— международные программы — это обучение на английском языке для иностранных студентов. ВГУ имеет партнерские отношения с многими университетами по всему миру, что позволяет студентам получить международный опыт и квалификацию;

— учебные материалы — это различные материалы, которые помогают студентам лучше усвоить учебный материал. Это могут быть видеолекции, презентации, тесты, задания и т.д. приложение, предоставляющее возможность создавать сайты для онлайн-обучения.

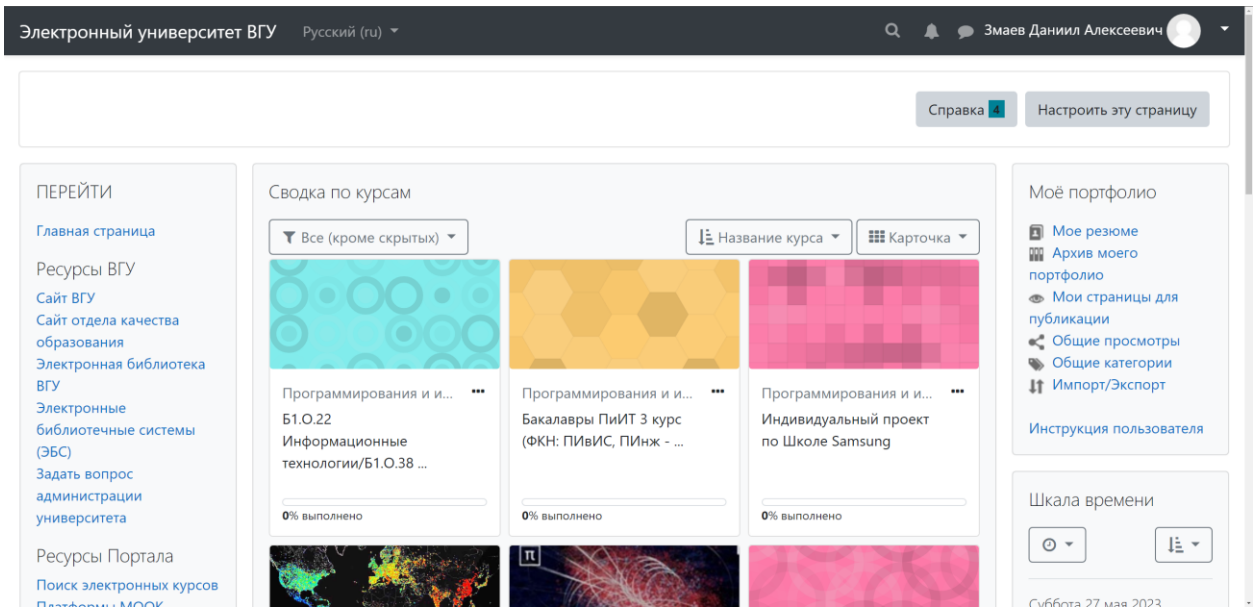

Рисунок 1 - Внешний вид интерфейса сайта Электронный университет ВГУ

Хотя Электронный университет ВГУ имеет значительную функциональность, в нем невозможно сохранить расписания обучающегося в какой-либо сервис, однако он имеет отношение к теме курсовой, потому что студенты ВГУ пользуются им в том числе как электронным расписанием занятием, так как оно частично занесено туда и в нем отображается, элемент курса в котором будет проводится занятие, время проведения занятия и ФИО преподавателя.

2.2.2 ВГУ ФКН Балльно-рейтинговая система

ВГУ ФКН Балльно-рейтинговая система (далее БРС) — это система оценки студентов, используемая на факультете компьютерных наук ВГУ (Воронежский государственный университет).

БРС служит для отслеживания итоговых баллов за аттестации и семестр. Баллы начисляются за выполнение заданий, тестирование, защиту проектов, выполнение практических работ и т.д.

БРС позволяет студентам контролировать свой прогресс и улучшать свои результаты. Она также позволяет прозрачно и объективно оценивать

студентов, что является важным при подготовке карьеры и поступлении на магистратуру или аспирантуру.

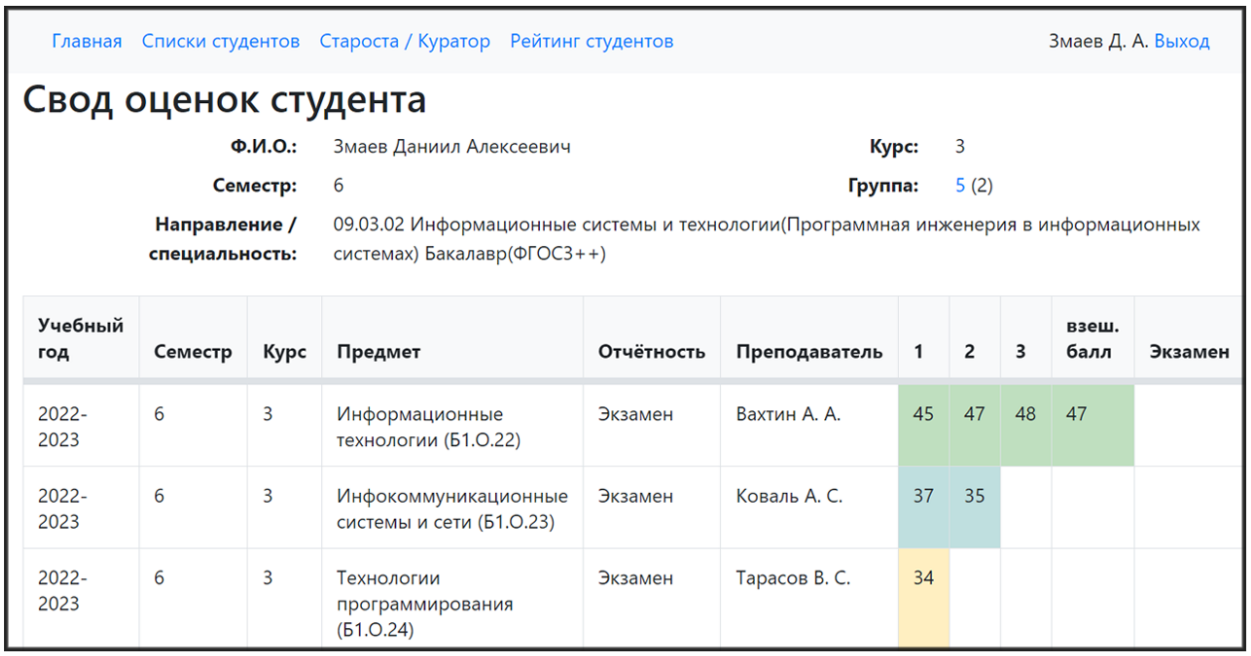

Рисунок 2 - Внешний вид интерфейса сайта ВГУ ФКН Балльно-рейтинговая система

БРС не имеет функциональной возможности отправки расписания студентов ФКН в их Google Calendar, однако, схожесть БРС и данной курсовой работы состоит в следующем: на данном ресурсе присутствуют баллы за аттестационные периоды, которые также связаны с расписанием занятий и могут быть включены в него.

2.3 Целевая аудитория

Целевая аудитория данного веб-приложения - студенты факультета компьютерных наук, которые используют сервис «Google Calendar» для организации своего расписания и планирования учебных занятий.

2.4 Жизненный цикл приложения

Рисунок 3 - Жизненный цикл приложения

Жизненный цикл приложения описывается следующими этапами:

2.4.3 Этап поиска расписания

На данном этапе администратору необходимо обратиться к сайту факультета компьютерных наук, для анализа расписание. Если расписание было обновлено, администратор переходит на следующий этап.

2.4.4 Этап скачивания расписания

На данном этапе администратору загружает расписание факультета компьютерных наук на свою локальную машину также, проделывает с расписанием необходимые манипуляции, например преобразование в формат, необходимый для загрузки.

2.4.5 Этап загрузки расписания в портал

На данном этапе администратор производит выгрузку файла с своей локальной машины на удаленный сервер с приложением через веб-интерфейс.

2.4.6 Этап распознавания расписания сервисом

На данном этапе происходит распознавание расписания для последующих обращений пользователей.

2.4.7 Этап ввода данных пользователем

На текущем этапе пользователь осуществляет ввод информации в форму, предназначенную для заполнения полей, необходимых для создания расписания. Данные поля включают в себя курс, номер группы, номер подгруппы и временной промежуток в неделях, на который требуется составить расписание.

2.4.8 Этап авторизации пользователя в аккаунт Google

На данном этапе пользователю необходимо выполнить авторизацию в свой Google аккаунт, для связи с Google Календарем, в который необходимо заполнить расписание.

2.4.9 Этап создания событий в Google Calendar

На данном этапе происходит заполнение Google Календаря пользователя событиями, исходя из данных, полученных из формы.

3 Реализация

3.1 Средства реализации

В качестве средств реализации были использованы:

- Python – это высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ;
- Flask – это фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2;
- SQLite – это компактная, встраиваемая реляционная база данных, которая работает на многих платформах, включая Windows, Mac OS X, Linux, Android и iOS. Она является самодостаточной и не требует отдельного сервера баз данных, поэтому ее можно легко интегрировать в различные приложения и использовать для хранения и управления структурированными данными, такими как текст, числа, изображения и т.д;
- Google Calendar Api – это интерфейс программирования приложений, который позволяет разработчикам создавать приложения, которые могут получать доступ к календарю Google, создавать, изменять и удалять события в календаре, а также выполнять другие операции;
- Google OAuth — это протокол авторизации, который позволяет пользователям авторизовываться на сторонних сайтах и приложениях с помощью учетной записи Google;

— xlrd — это библиотека для работы с файлами Excel в Python. Она позволяет читать данные из файлов Excel в форматах .xls и .xlsx и использовать их в своих скриптах на Python.

3.2 Разработка архитектуры

При проведении работ по проектированию архитектуры программного продукта можно выделить следующие этапы:

— анализ требований и выбор технологий: на этом этапе будет проведен анализ требований к приложению, определены функциональные и нефункциональные требования, а также выбраны технологии, которые будут использоваться для разработки приложения;

— проектирование базы данных: на этом этапе будет спроектирована база данных для приложения, определены таблицы, поля, а также выбрана СУБД для хранения данных. В данном случае, используется SQLite, которая является легковесной и встраиваемой СУБД;

— разработка серверной части приложения: на этом этапе будет разработана серверная часть приложения, которая будет отвечать за обработку запросов от клиентской части, взаимодействие с базой данных и работу с API Google Calendar. Для разработки серверной части можно использовать фреймворк Flask на языке Python;

— разработка клиентской части приложения: на этом этапе будет разработана клиентская часть приложения, которая будет отвечать за отображение данных пользователю и взаимодействие с серверной частью приложения. В данном случае в качестве клиентской части используется веб-страница, формируемая статическим шаблонизатором Jinja.

3.2.1 Диаграмма вариантов использования

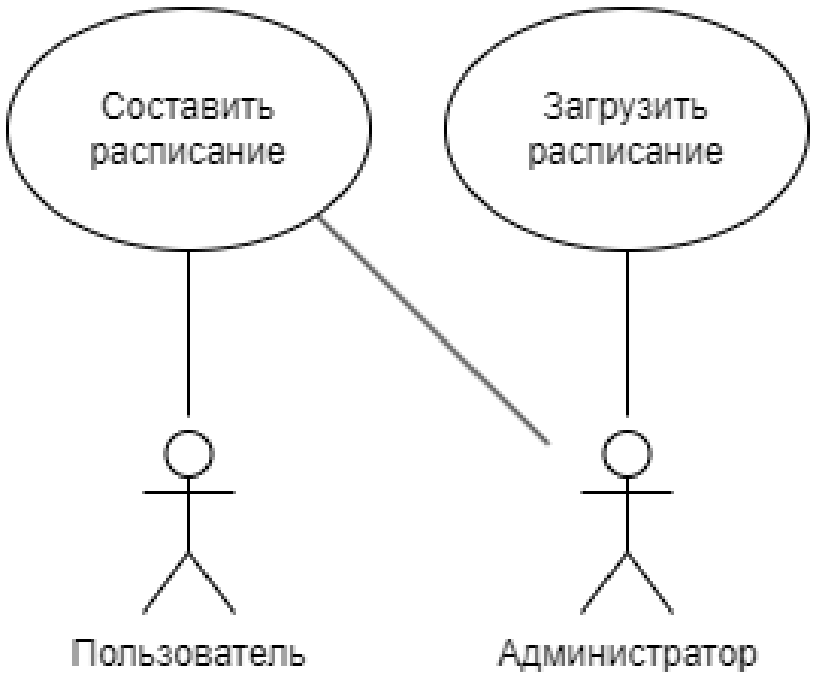

Рисунок 4 - Диаграмма вариантов использования

Диаграмма вариантов использования — это тип диаграммы, который используется для описания функциональности системы с точки зрения пользователей, которые будут использовать систему.

Диаграмма вариантов использования, представленная на рисунке 4, демонстрирует следующие функциональные возможности акторов. Пользователь имеет функциональную возможность составить расписание, администратор также может составить расписание и загрузить расписание в портал.

3.2.2 Диаграмма состояний

Рисунок 5 - Диаграмма состояний

Диаграмма состояний – это тип диаграммы, который используется для описания состояний объекта, его переходов между состояниями и действий, которые происходят при переходе между ними.

Диаграмма состояний, представленная на рисунке 5, демонстрирует следующие этапы использования веб-сервиса: начальное состояние – это главная страница веб-сервиса, в зависимости от типа пользователя, осуществляется переход на страницу расписания иди страницу входа в аккаунт администратора. В первом случае прежде, чем перейти в следующее состояние, составление расписания, происходит проверка заполненных полей. После составления расписания происходит переход в состояния завершения.

В случае перехода в состояние страницы входа в аккаунт администратора, прежде чем переместиться в состояние страницы загрузки расписания, происходит проверка заполненных полей. После перехода в состояние загрузки файла, в случае если формат файла был верный, происходит переход в состояние завершения.

3.2.3 Диаграмма последовательности пользователя

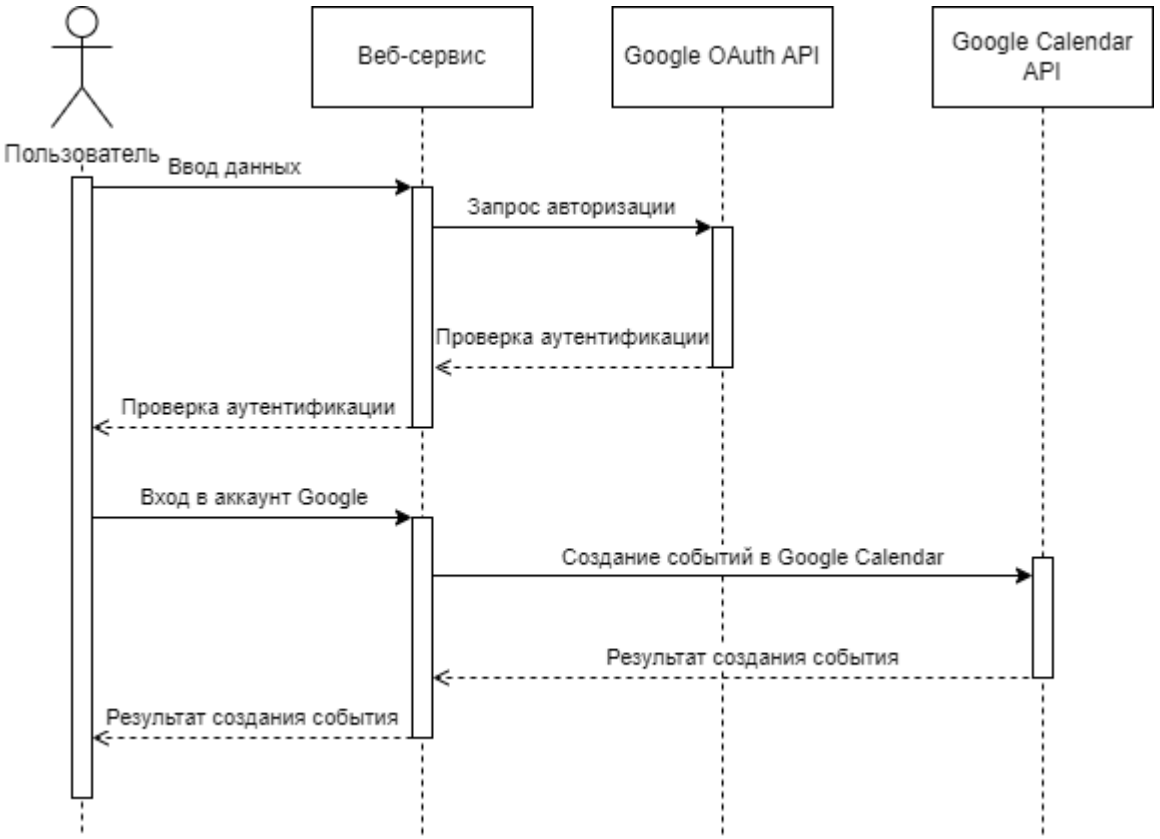

Рисунок 6 - Диаграмма последовательности пользователя

Диаграмма последовательности — это тип диаграммы, который показывает, как объекты взаимодействуют друг с другом в рамках определенного сценария.

На рисунке 6 предоставлена диаграмма последовательности, которая в данном контексте, показывает, как клиентский объект взаимодействует с веб-сервисом для создания расписания.

3.2.4 Диаграмма последовательности администратора

Рисунок 7 - Диаграмма последовательности администратора

На рисунке 7 предоставлена диаграмма последовательности, которая в данном контексте, показывает, как администратор взаимодействует с веб-сервисом для загрузки расписания в веб-сервис.

3.3 Этапы разработки

3.3.5 Получение данных

На данном этапе поступают данные, которые используются для разбора ячеек файла с расписанием занятий. В данном случае, данные могут быть получены из Excel-файла формата xls.

3.3.6 Авторизация пользователя при помощи сервиса «Google OAuth»

На данном этапе происходит аутентификации пользователей на веб-сервисе при помощи сервиса «Google OAuth». Данный сервис позволяет получить токен пользователя Google для заполнения его «Google Calendar».

3.3.7 Создание событий в сервисе «Google Calendar»

На данном этапе создаются события в «Google Calendar» на основе данных о занятиях, полученных на предыдущем шаге. Для создания событий, используется «Google Calendar API» и библиотека «google-api-python-client».

3.3.8 Обработка ошибок

На данном этапе происходит обработка возможных ошибок, которые могут возникнуть в процессе разбора ячеек файла расписания или создания событий в сервисе «Google Calendar».

3.4 Реализация логики

Логику приложения можно разделить на несколько этапов:

3.4.9 Описание класса Parser

На этом этапе создается класс Parser, принимающий в конструктор путь к файлу с расписанием. Класс имеет несколько методов, основные из них представлены в списке ниже:

— метод get_merged_cell_value(self, row, col) служит для получения значений объединённых ячеек. Он необходим, т.к. по умолчанию в ячейках такого типа значение хранится в первой ячейке, а остальные ссылаются на нее;

— метод __get_time(val) служит для разделения времени в ячейках и последующей записи их в массив, необходимый для указания времени начала пары и ее конца в событии «Google Calendar»;

— метод parse_nominator_schedule(self, course_num, group_num, subgroup_num) служит для заполнения списка занятий для четных недель, который содержит в себе день недели, время начала и конца занятия название предмета и ФИО преподавателя;

— метод parse_denominator_schedule(self, course_num, group_num, subgroup_num) служит для заполнения списка занятий для нечетных недель. Имеет схожую функциональность с методом parse_nominator_schedule(), однако необходим, так как при распознавании расписания для знаменателя используются ячейки другого вида и имеются особые отступы.

3.4.10 Описание класса Schedule

На этом этапе создается класс Schedule, ответственный за создание событий в веб-сервисе «Google Calendar». Класс имеет несколько методов, основные из них представлены в списке ниже:

— Метод process() необходим для авторизации пользователя через аккаунт «Google», при успешной авторизации программа получает и сохраняет токен пользователя в папку «resources», необходимый для распознавания принадлежности аккаунта и календаря. После успешного получения токена запускается метод create_shedule(). После заполнения расписания токен пользователя удаляется.

— Метод create_events(service, start_time, end_time, day_index, summary, week_amount, weektype) служит для заполнения расписания в веб-сервис «Google Calendar» путем создания событий (рис3). Так как в расписание имеются занятия,

проводимые один раз в две недели, то создать события необходимо два раза для четных и нечетных недель, поэтому используется INTERVAL=2.

— метод create_schedule(service, course, group, subgroup, week_amount). Данный метод создает экземпляр класса Parser и вызывает два метода: parse_denominator_schedule и parse_nominator_schedule, запоминая списки, переданные из них. После успешного выполнения методов, вызывается метод create_event(), записывающий элементы списков в веб-сервис «Google Calendar».

```
event = {
    'summary': f'{summary}',
    'location': 'VSU',
    'description': '',
    'start': {
        'dateTime': datetime.combine(start_date, start_time).isoformat(),
        'timeZone': timezone.zone,
    },
    'end': {
        'dateTime': datetime.combine(start_date, end_time).isoformat(),
        'timeZone': timezone.zone,
    },
    'recurrence': [
        f'RRULE:FREQ=WEEKLY;INTERVAL=2;COUNT={week_amount};BYDAY={day}'
    ],
    'reminders': {
        'useDefault': False,
    },
}
```

Рисунок 8 - Пример создания события

3.4.11 Описание механизма маршрутизации приложения

В качестве механизма маршрутизации данного приложения, используется код на языке программирования Python, описанный в файле app.py, являющийся частью архитектуры фреймворка Flask.

3.5 Реализация интерфейса

При открытии веб-приложения пользователь попадает на главную страницу, на которой содержится информация о приложении, а также навигационная панель.

3.5.1 Интерфейс пользователя

Пользовательский интерфейс данного приложения был спроектирован с целью обеспечения возможности передачи данных, необходимых для выполнения функциональных требований разрабатываемого веб-сервиса.

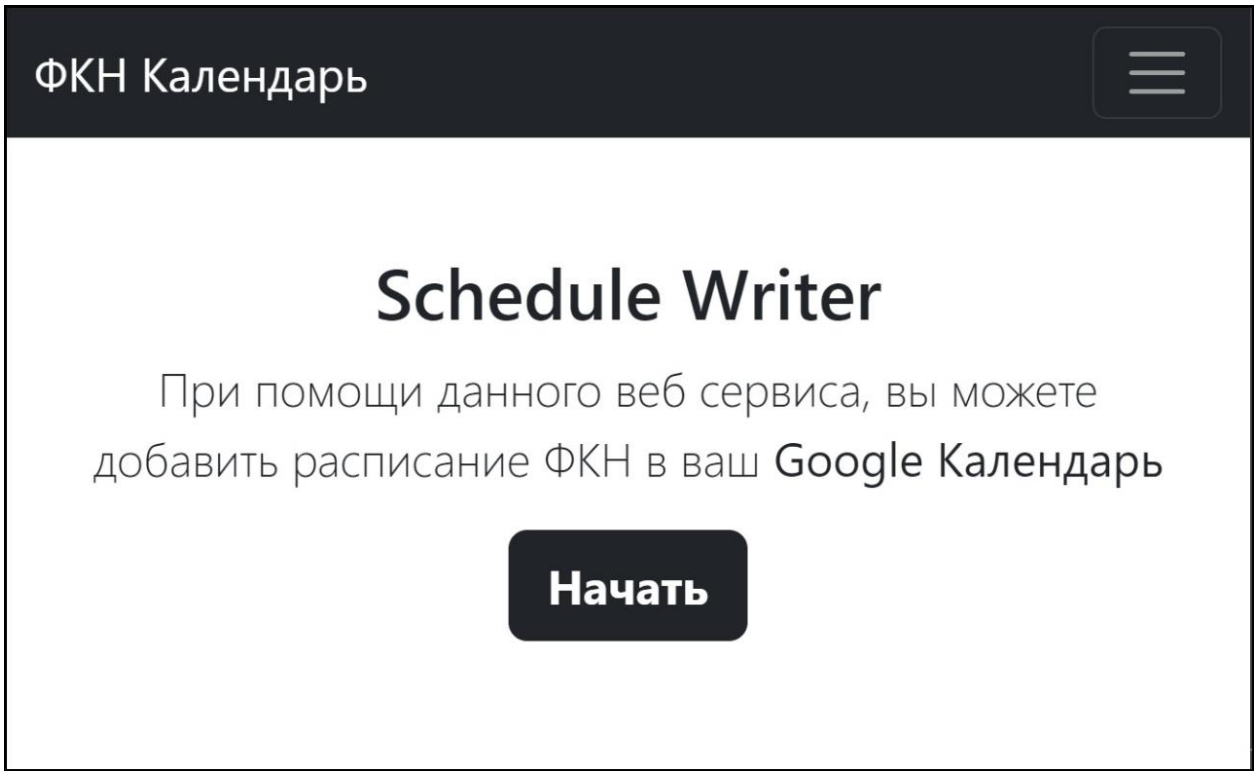

Рисунок 9 - Главная страница веб-приложения

При нажатии на кнопку «Начать» или выборе раздела «Расписание» в меню навигации пользователь попадает на страницу «Заполнение расписания».

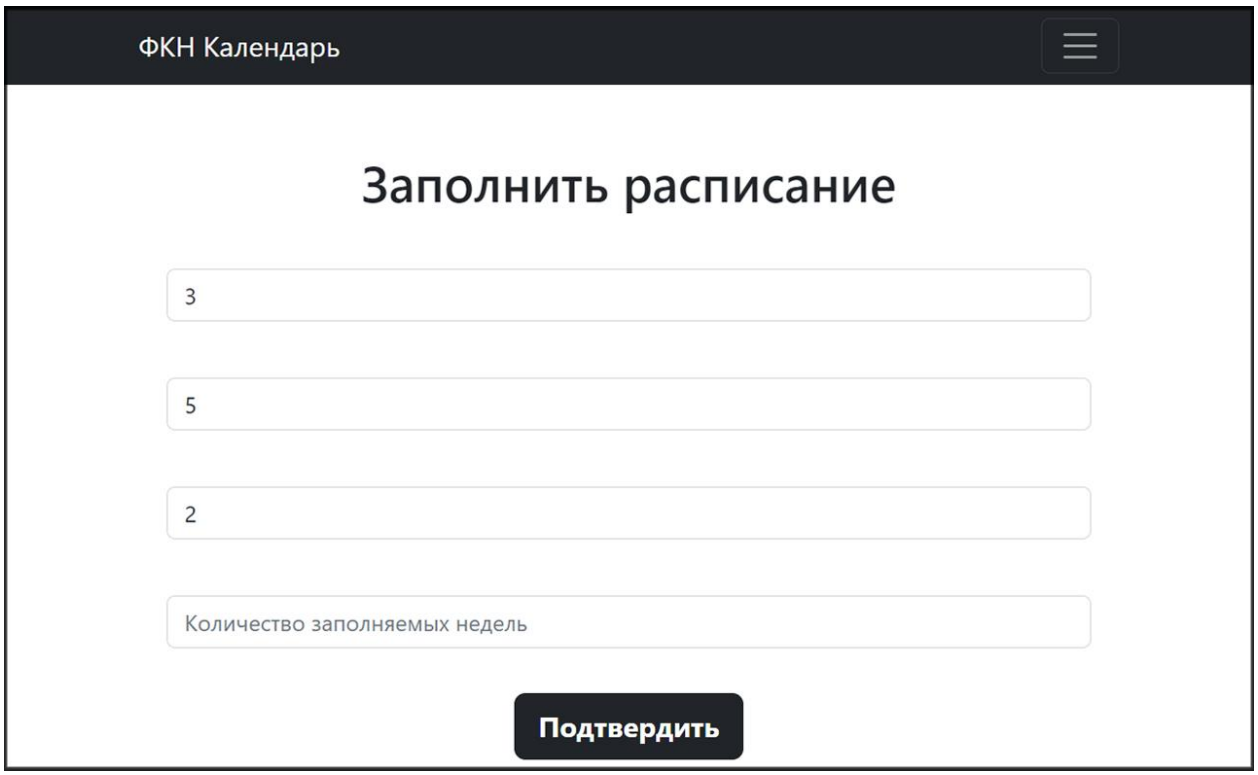

Рисунок 10 - Страница «Заполнение расписания»

При корректном вводе всех полей пользователем, его перенаправит на страницу авторизации в Google аккаунт.

Если поля содержат неправильно заполненную информацию или не заполнены вовсе, пользователь получит сообщение с просьбой исправить некорректные данные или заполнить недостающие поля.

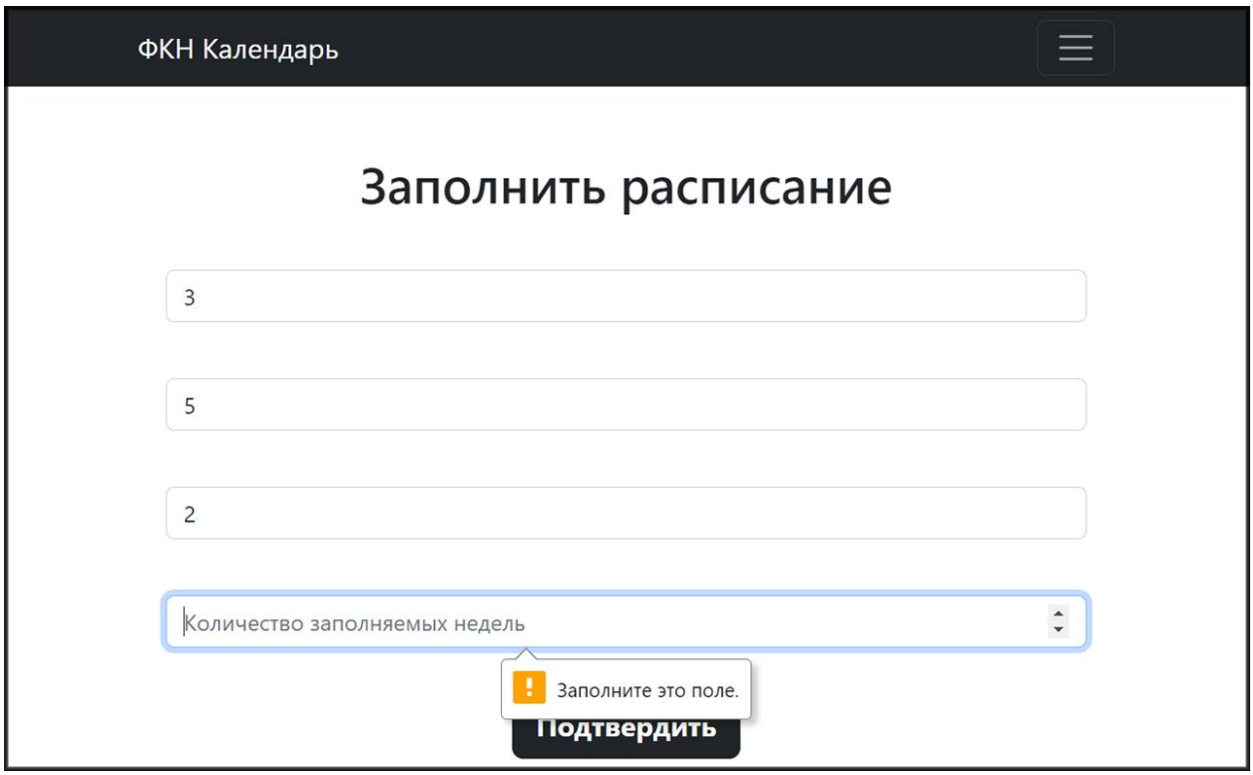

Рисунок 11 - Ошибка валидации

По завершении процесса загрузки, пользователь может осуществить просмотр полученного результата в «Google Calendar», связанном с аккаунтом, на который происходил вход.

3.5.2 Интерфейс администратора

После выбора опции «Админ» на навигационной панели, происходит открытие страницы входа в учетную запись администратора.

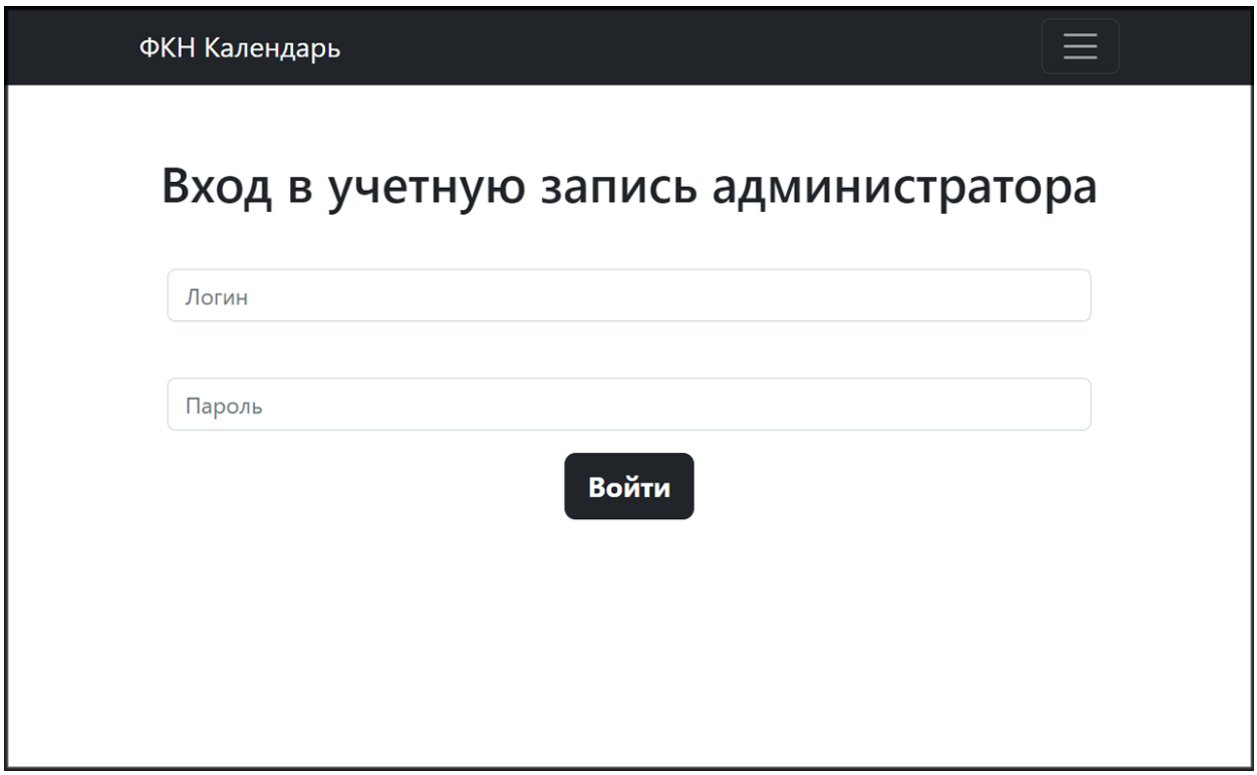

Рисунок 12 - Страница входа в учетную запись администратора

В случае ввода некорректных данных, система выведет сообщение об ошибке. вводе некорректных данных появится сообщение об ошибке:

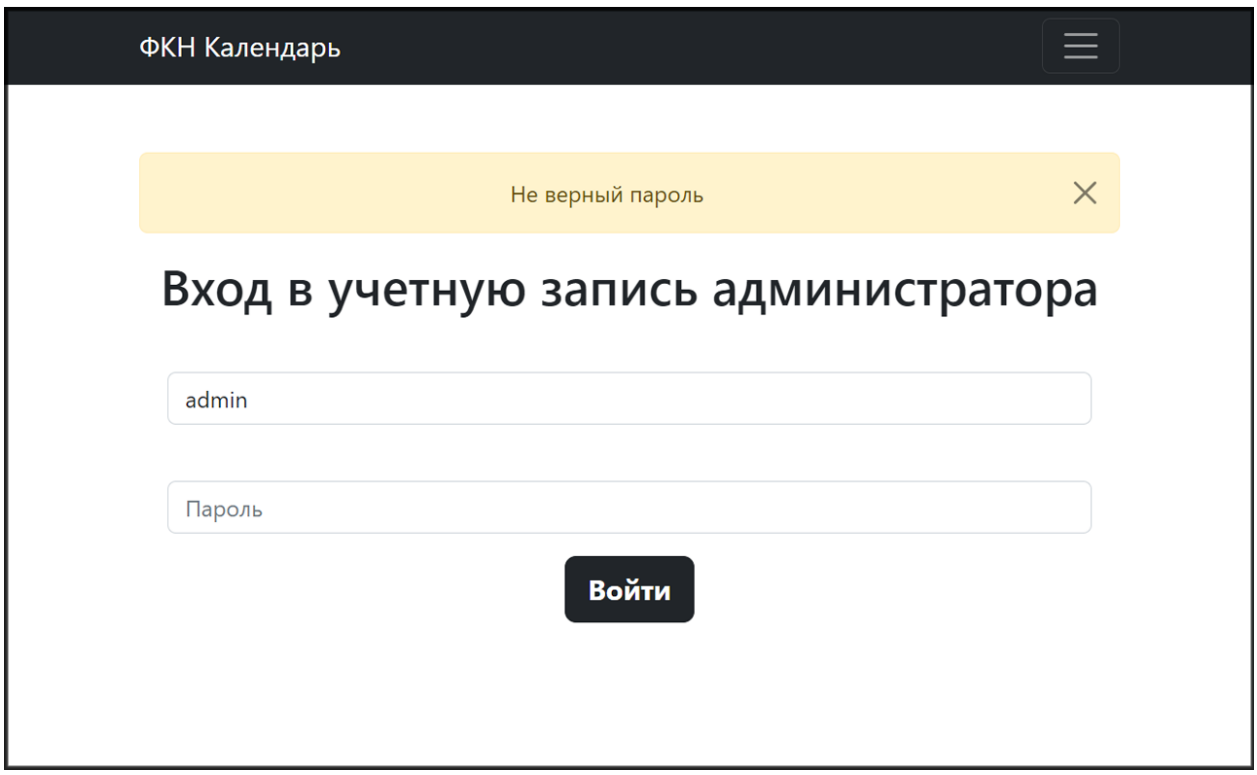

Рисунок 13 - Ошибка валидации администратора

После успешного выполнения процесса валидации, система перенаправит администратора на страницу выбора файла. На данной странице администратор может произвести замену расписания на новое в случае, если оно было изменено.

Важно отметить, что на странице также присутствует валидация: система принимает файлы только формата xls, и при попытке загрузки файлов иного формата, система выведет сообщение об ошибке.

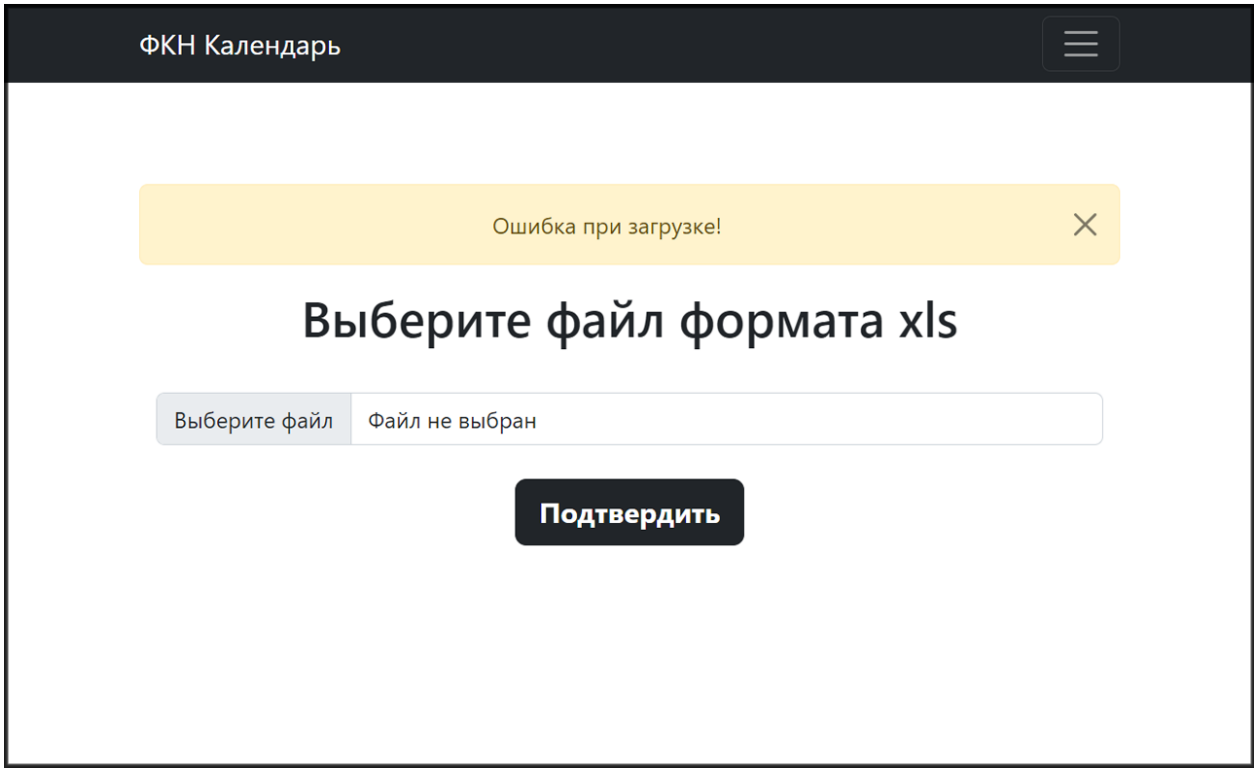

Рисунок 14 - Ошибка загрузки файла

При успешном выполнении операции, предыдущее расписание будет заменено на новое.

Следует отметить, что возможность изменения расписания доступна только администраторам, которые имеют доступ к данной странице после успешной авторизации в системе, использующей токены авторизации. Обычным пользователям доступ к данной странице невозможен.

4 Демонстрация работы

Для демонстрации функциональности загрузки файлов будет использован файл schedule.xml, который успешно загрузится в приложение.

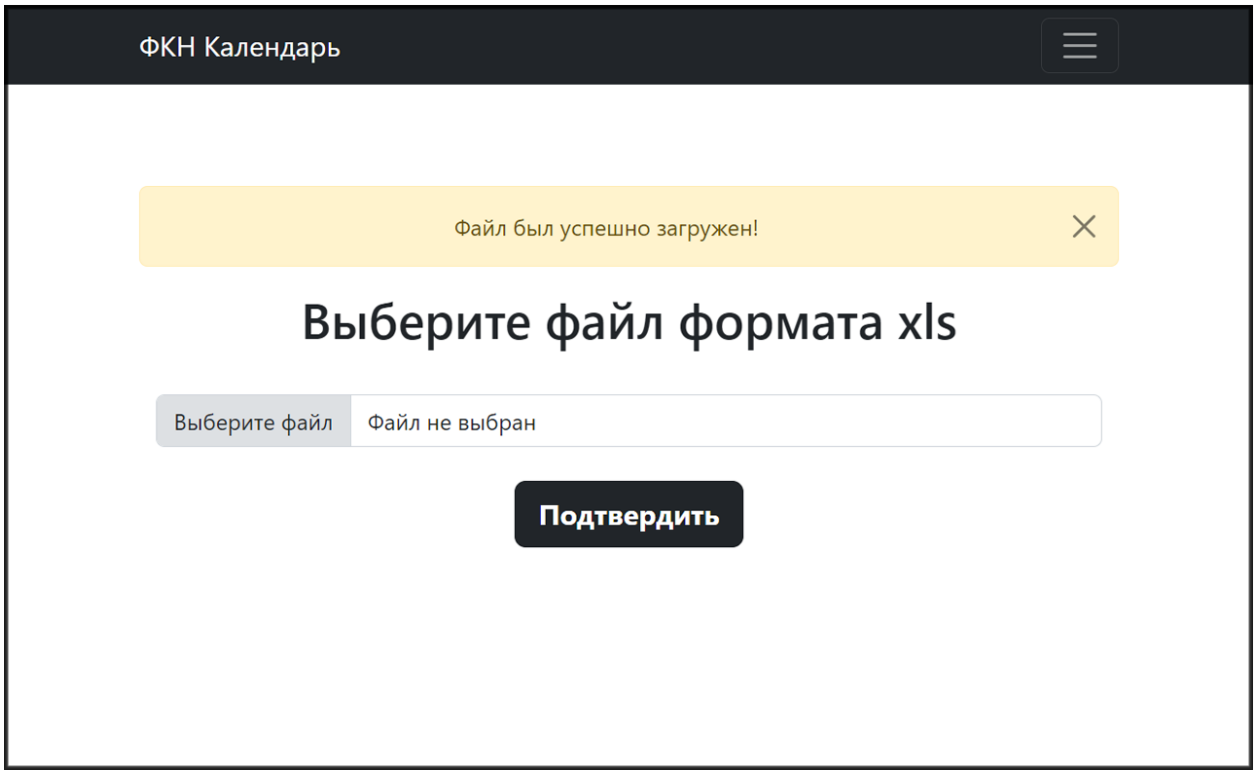

Рисунок 15 - Демонстрация успешной загрузки файла с расписанием

После успешной загрузки файла, необходимо перейти на вкладку «Расписание» на навигационной панели и произвести сохранение расписания в сервис «Google Calendar». В качестве примера выполнения работы программы, при вводе данных о третьем курсе, пятой группе и второй подгруппе был получен следующий результат.

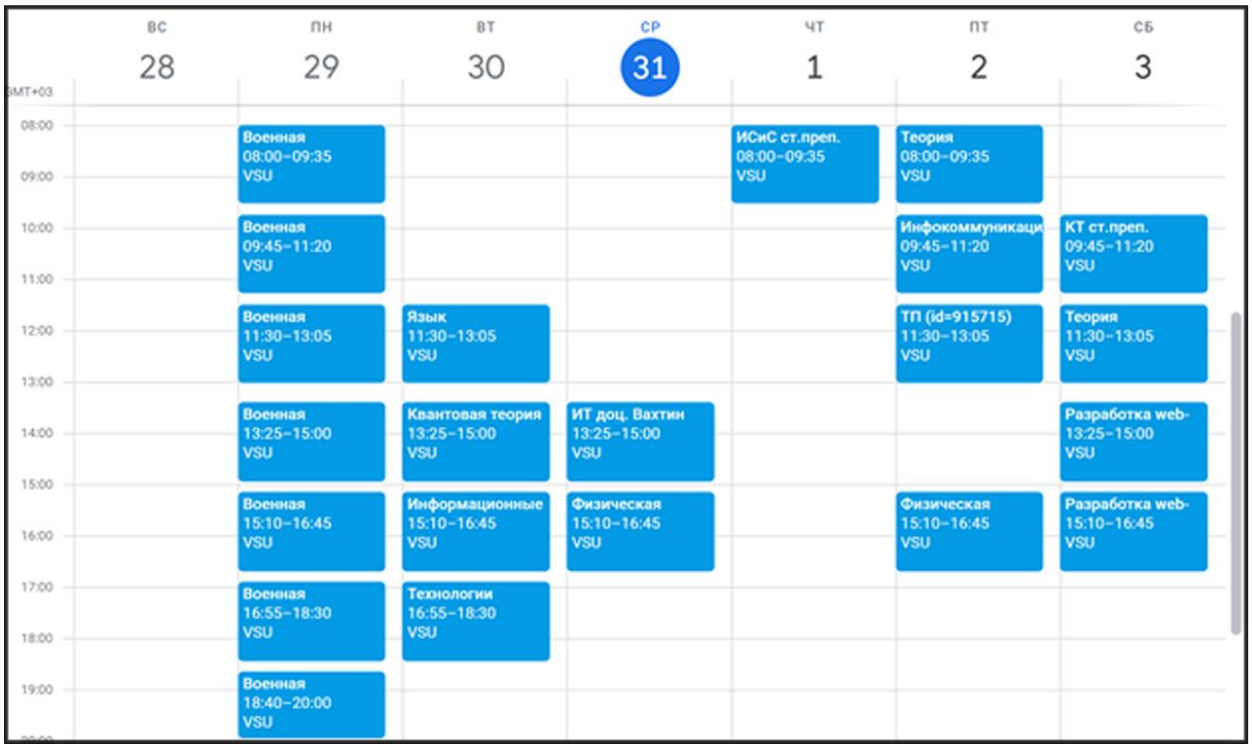

Рисунок 16 - Демонстрация результата выполнения программы

Заключение

Как результат выполнения данной курсовой работы, было разработано веб-приложение, которое позволяет автоматически переносить расписание занятий факультета компьютерных наук в сервис «Google Calendar». Были выполнены все поставленные перед проектом функциональные и нефункциональные требования, а именно: возможность автоматического добавления расписания занятий в Google Calendar на основе данных из источников, таких как Excel-файл, возможность настройки периода, на который будет добавлено расписание, возможность синхронизации с «Google Calendar» и уведомления студентов о предстоящих занятиях.

Проект был разработан с использованием Google Calendar API и SQLite, что позволило создать компактное, эффективное и удобное приложение. Веб-приложение может быть использовано студентами факультета компьютерных наук для доступа к их расписанию занятий.

В результате выполнения курсовой работы были получены знания и навыки, необходимые для разработки веб-приложений, а также опыт работы с «Google Calendar API» и «SQLite». Разработанное приложение может быть использовано в учебных заведениях для упрощения и оптимизации процесса обучения.

Список использованных источников

1. FLASK [электронный ресурс] – Режим доступа: https://flask.palletsprojects.com/ – Заглавие с экрана. – (дата обращения: 26.05.2023).

2. Использование OAuth 2.0 для доступа к API Google [электронный ресурс] – Режим доступа: https://developers.google.com/identity/protocols/oauth2?hl=ru – Заглавие с экрана. – (дата обращения: 26.05.2023).

3. Быстрый старт Python, Быстрый старт Python [электронный ресурс] – Режим доступа: https://developers.google.com/calendar/api/quickstart/python?hl=ru – Заглавие с экрана. – (дата обращения: 26.05.2023).

4. Календари и события [электронный ресурс] – Режим доступа: https://developers.google.com/calendar/api/concepts/events-calendars?hl=ru – Заглавие с экрана. – (дата обращения: 26.05.2023).

5. Создать события [электронный ресурс] – Режим доступа: https://developers.google.com/calendar/api/guides/create-events?hl=ru – Заглавие с экрана. – (дата обращения: 26.05.2023).

6. Использование OAuth 2.0 для приложений веб-сервера [электронный ресурс] – Режим доступа: https://developers.google.com/identity/protocols/oauth2/web-server?hl=ru – Заглавие с экрана. – (дата обращения: 26.05.2023).

7. WTForms Fields [электронный ресурс] – Режим доступа: https://wtforms.readthedocs.io/en/3.0.x/fields/#field-definitions – Заглавие с экрана. – (дата обращения: 26.05.2023).

8. WTForms Validators [электронный ресурс] – Режим доступа: https://wtforms.readthedocs.io/en/3.0.x/validators/#custom-validators – Заглавие с экрана. – (дата обращения: 26.05.2023).

9. WTForms CSRF Protection [электронный ресурс] – Режим доступа: https://wtforms.readthedocs.io/en/3.0.x/csrf/ – Заглавие с экрана. – (дата обращения: 26.05.2023).

10. SQLAlchemy Quick Start [электронный ресурс] – Режим доступа: https://flask-sqlalchemy.palletsprojects.com/en/3.0.x/quickstart/#create-the-tables – Заглавие с экрана. – (дата обращения: 26.05.2023).

11. Xlrd [электронный ресурс] – Режим доступа: https://flask-https://xlrd.readthedocs.io/en/latest/ – Заглавие с экрана. – (дата обращения: 26.05.2023).

12. Bootstrap Introduction [электронный ресурс] – Режим доступа: https://getbootstrap.com/docs/4.3/getting-started/introduction/ – Заглавие с экрана. – (дата обращения: 26.05.2023).

13. Flask-Login [электронный ресурс] – Режим доступа: https://flask-login.readthedocs.io/en/latest/#configuring-your-application – Заглавие с экрана. – (дата обращения: 26.05.2023).

14. Google API Client [электронный ресурс] – Режим доступа: https://pypi.org/project/google-api-python-client/ – Заглавие с экрана. – (дата обращения: 26.05.2023).

15. Google Calendar Simple API documentation! [электронный ресурс] – Режим доступа: https://google-calendar-simple-api.readthedocs.io/en/latest/– Заглавие с экрана. – (дата обращения: 26.05.2023).

Приложение А

```
def parse_nominator_schedule(self, course_num,
group_num, subgroup_num) -> list:

        group_index = self.__get_group_index(group_num,
course_num, subgroup_num)

        schedule_numerator = []

        col_start = 4

        for i in range(col_start,
self.worksheet.nrows):

            if self.worksheet.cell_value(i, 1) != '':

                val = self.get_merged_cell_value(i,
group_index)

                if val is None:

                    val = self.worksheet.cell_value(i,
group_index)

                if val != "":
schedule_numerator.append([self.__get_day_index(self.ge
t_merged_cell_value(i, 0)),
self.__get_time(self.get_merged_cell_value(i, 1)),
val])

        return schedule_numerator


    def parse_denominator_schedule(self, course_num,
group_num, subgroup_num) -> list:
```

```
        group_index = self.__get_group_index(group_num,
course_num, subgroup_num)
        schedule_denominator = []
        i = 5
        while i <= self.worksheet.nrows - 1:
            val = self.get_merged_cell_value(i,
group_index)
            if val is None:
                val = self.worksheet.cell_value(i,
group_index)
            if val != "":

schedule_denominator.append([self.__get_day_index(self.
get_merged_cell_value(i, 0)),

self.__get_time(self.get_merged_cell_value(i, 1)),
val])
            delimiter_list = [19, 36, 53, 70, 87]
            if i in delimiter_list:
                i += 1
            i += 2
        return schedule_denominator
```

Приложение Б

```
def create_events(service, start_time, end_time,
day_index, summary, week_amount, weektype):
    timezone = pytz.timezone('Europe/Moscow')
    today = datetime.today().weekday()
    day_offset = 7 - today  # Нужен для того, чтобы
расписание начиналось с пн
    if weektype == "nominator":
        start_date = timedelta(days=day_offset) +
datetime.now(tz=timezone).date()
    else:
        start_date = timedelta(days=day_offset + 7) +
datetime.now(tz=timezone).date()
    day = "MO"

    while start_date.weekday() != day_index:
        start_date += timedelta(days=1)
        day = get_day_name_by_index(day_index)

    event = {
        'summary': f'{summary}',
        'location': 'VSU',
        'description': '',
        'start': {
```

```
            'dateTime': datetime.combine(start_date,
start_time).isoformat(),
            'timeZone': timezone.zone,
        },
        'end': {
            'dateTime': datetime.combine(start_date,
end_time).isoformat(),
            'timeZone': timezone.zone,
        },
        'recurrence': [

f'RRULE:FREQ=WEEKLY;INTERVAL=2;COUNT={week_amount};BYDA
Y={day}'
        ],
        'reminders': {
            'useDefault': False,
        },
    }


    event =
service.events().insert(calendarId='primary',
body=event).execute()
    print('Event created: %s' %
(event.get('htmlLink')))
```