ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

Программа подготовки бакалавров по направлению

38.03.05 Бизнес-информатика

Пилипенко Глеб Игоревич

Курсовая работа

по теме “Продуктизация и доработка навыка голосового помощника Алиса для работы с расписанием и новостями ВШЭ”

Выполнил:

Пилипенко Глеб Игоревич 22БИ-3

Принял: Дыдычкин Александр Алексеевич

**Нижний Новгород, 2024**

**Оглавление**

[Пояснительная записка 3](#_heading=h.gjdgxs)

[Техническое задание 7](#_heading=h.1fob9te)

[Анализ проблематики 10](#_heading=h.3znysh7)

[Анализ существующих решений, технологий и аналогов 16](#_heading=h.2et92p0)

[Полученные результаты, методика испытаний и итоги анализа 23](#_heading=h.tyjcwt)

[Список использованных источников 31](#_heading=h.3dy6vkm)

**Пояснительная записка**

**Введение**

Наименование: Написание модуля по внесению информации о студентах, преподавателях Высшей школы экономики, их расписании и работа с этими данными в Yandex Data Base, проектирования подпрограммы по взаимодействию с пользователем во время сессии навыка..

Разработка ведется на основании данный документов:

1. Официальная документация языка программирования Python.
2. Официальная документация по YDB.
3. Официальная документация YDB Python SDK.

Современные голосовые помощники становятся неотъемлемой частью повседневной жизни благодаря своей простоте и функциональности. Среди множества виртуальных ассистентов, представленных на российском рынке, особой популярностью пользуется «Алиса» от компании Яндекс. Данный голосовой помощник согласно [данным](https://sprut.ai/news/chto-proishodit-na-rynke-golosovyh-assistentov-v-rossii) доля, занимаемая данным продуктом на рынке выросла с 44% в октябре 2020 года до 83% в августе 2024.

Российский рынок умных колонок в 2024 году показал значительный рост. По данным группы «М.Видео-Эльдорадо», опубликованным 17 января 2025 года в TAdviser, объем продаж таких устройств увеличился на 20% в штучном выражении и на 26% в денежном по сравнению с предыдущим годом.

Цель данной работы – оптимизировать взаимодействие пользователя с голосовым помощником при получении информации о расписании, сделав этот процесс максимально быстрым и комфортным. Также необходимо пересмотреть выполненное ранее решение, оптимизировать его, сделать более универсальным и доступным для пользователей не только филиала НИУ ВШЭ в Нижнем Новгороде, но и в остальных кампусах университета.

Проблематика данной курсовой работы заключается в использовании баз данных компании Яндекс для удовлетворения потребности в универсальности и добавления возможности пользователю вносить данные расписания самостоятельно. Разрабатываемая подпрограмма в рамках командного проекта выполняется функцию по внесению, сбору, изменению и хранению информации о пользователях. Также выполнена работа по составлению прототипа модуля взаимодействия с пользователем.

**Назначение и область применения**

Цель реализуемого программного модуля заключается во внесении информации о пользователе, его расписании, поиску внесенного расписания в базе данных и приведении его в читаемый для пользователя вид.

Область применения включает в себя, в основном, использование студентами и преподавателями НИУ ВШЭ в независимости от кампуса при использовании голосового ассистента Алиса или применение в рамках других работ, где существует необходимость внесения расписания и информации в базу данных.

**Технические характеристики**

Главной задачей являлось анализ и изменение кода, который был выполнен в прошлой работе, а именно: изменение логики модуля взаимодействия с пользователем, добавление преподавательского режима, возможности вносить и получать пары майноров, английского и факультативов, расширение области применимости, которая бы включала все кампусы НИУ ВШЭ, написание подпрограммы, которая будет по полученным данным из модуля взаимодействия с пользователем осуществлять внесение информации о нем в базу данных, а также о предметах, которые необходимо добавить и осуществлять поиск по базе данных выбранных предметов и предоставлять массив, состоящих их объектов базы данных, которые содержат информацию о дате и времени занятий, наименовании предмета, номере аудитории, фамилии преподавателя и тип пар, т.е. лекционное, практическое занятие или научно-исследовательский семинар.

К основным ограничением данного способа хранения и заполнения расписания можно отнести: сложность в составлении и унификации списка предметов, т.е. заранее необходимо знать список дисциплин всех образовательных программ НИУ ВШЭ, для их обработки, необходимость постоянно добавлять информацию о дисциплинах вручную, что может быть не удобно пользователю.

Методы, которые применялись для разработки навыка включали использование официальных библиотек YDB и самого сервиса для создания заранее спроектированной базы данных, заполнения необходимой информации о пользователе: фамилия, имя, отчество и номер группы, если это студент, информации о группе, расписании. Используемый способ подразумевает в независимости от кампуса, вида предмета хранить расписание, что делает его более гибким, чем решение с парсингом гугл таблиц, а также в виду того, что все предметы хранятся в заранее установленным виде, универсальность данного решения также выше. Подпрограмма состоит из пяти модулей:

1. Модуль models, который содержит классы данных: базовый класс Lesson для занятий, содержащий атрибуты name, type\_l, building, auditorium, id\_lecturer, time, is\_weekly, is\_upper, lesson\_date. Два класса наследника PersonalLesson**,** GroupLesson, которые наследуют Lesson, добавляя соответственно атрибуты  id\_student или id\_group. Также присутствуют вспомогательные классы для таблиц: PersonalLessonStudent, GroupLessonGroup. Помимо этого в данном модуле содержатся классы Student, Lecturer для пользователей, содержащие данные о ФИО и номер группы, если это студент, объект группа, который хранит атрибуты name, edu\_year, edu\_program, faculty, edu\_format, edu\_level и вспомогательный класс LecturerGroup.
2. Модуль obj\_queries, содержащий функции для каждого из объектов: Group, Student, Lecturer по получению записей в базе данных объекта, идентификатора и проверке регистрации.
3. Модуль Registration\_ydb, хранящий в себе общую функцию регистрации registration\_user и методы, которые используются ей в зависимости от параметров для регистрации студента, лектора или группы.
4. Модуль schedule\_queries, осуществляющий внесение расписания в таблицы: insert\_lesson, insert\_lesson\_data insert\_help\_tables\_data, его поиск и изменение данных пользователя в БД. В силу особенностей хранения данных и выполнения запросов функции для поиска предмета по названию, лектору, дате разделены для студентов и преподавателей и также присутствуют отдельные методы для нахождения предмета по дню недели: find\_by\_week\_day\_lesson\_student, find\_by\_week\_day\_lesson\_lecturer.
5. Модуль utils содержит функцию make\_readable для вывода данных пользователю в заданном формате.

Описанная выше часть подпрограммы будет осуществлять взаимодействие с модулем по обработке пользовательских запросов во время работы навыка голосового ассистента, поэтому данный алгоритм преобразовывает данные, хранящиеся в YDB в массив атрибутов row класса ydb.ResultSet. Данный тип данных можно удобно использовать и вычленять всю необходимую информацию.

Входные данные содержат объект ydb.QuerySessionPool, который необходим для каждой функции по взаимодействию с базой данных и варьируются в зависимости от задачи. Так методы поиска требуют указания параметров, по которым будет осуществляться поиск, связан ли предмет с данной образовательной программой, идентификаторов студента, группы. В функции добавления предмета требуются данные пользователя в виде списка. Также некоторые другие функции требуют передачи объектов Lesson, Student или Lecturer в зависимости от задачи.

Написанная подпрограмма выполнена на языке программирования Python. Можно выделить следующие основные моменты, которые повлияли на данное решение:

Программное решение использует язык программирования Python. К критериям, которые сыграли ключевую роль в выборе данного языка программирования можно отнести:

* Использование остальными участниками данного языка программирования, поэтому для интеграции рассматриваемого решения необходимо применение единственного решения.
* Популярность используемого языка программирования.
* Наличие достаточного количества материалов и информации, а также библиотек для работы с Google Таблицами. Также данный язык предоставляет уникальный возможности по эффективности и скорости разработки за счет удобного синтаксиса, поддержки и развития сообщества программистов данного языка. Помимо этого, рассматриваемый язык программирования является одним из наиболее популярных для сбора и преобразования информации, как с Google Таблиц, так и в целом из других источников.

Программный модуль, в свою очередь, располагается в Yandex Cloud – платформе, предоставляющей удобные решения для хранения и запуска программного кода для Яндекс Диалогов. Данный сервис предоставляет бесплатную возможность тестирования кода для навыка голосового ассистента Яндекса в Yandex Cloud Functions, который позволяет запускать приложения в безопасном, отказоустойчивом и масштабируемом окружении без создания и обслуживания виртуальных машин с использованием рассматриваемого языка программирования. Также используемый сервис поддерживает интеграцию с Object Storage – другим сервисом, предоставляющим возможность бесплатного хранения необходимых файлов и их использование в рамках программного кода. Данное решение является жизненно необходимым в контексте рассматриваемого программного модуля, так как он использует файлы credentials.json и token.json, содержащие учетные данные разработчика, нужные для получения доступа к API Google таблиц и дальнейшей обработки информации. Возможность использования этих технических средств значительно повышает эффективность и позволяет ускорить процессы тестирования, разработки и внедрения рассматриваемого программного модуля по получению расписания студентов из Google Таблиц.

**Техническое задание**

Программа по парсингу Google таблиц – программа, которая позволяет получить доступ к информации, находящейся в таблицах, и в соответствии с полученным запросом и закономерностями обрабатывает имеющиеся неструктурированные или полуструктурированные данные, преобразовывает их в формат необходимый и удобный для анализа и дальнейшего использования.

Основной целью в рамках данного программного модуля является создание подпрограммы, предоставляющей данные о расписании студентов Высшей школы Экономики очных образовательных программ Нижнего Новгорода. Техническое решение должно извлекать информацию о расписании студентов из Google Таблиц и сохранять их в виде массива объектов.

Подпрограмма является частью системы, разрабатываемой как навык голосового ассистента Алиса. Данное решение позволит значительно улучшить пользовательский опыт, сделав процесс получения информации удобным и эффективным.

**Функциональные требования:**

1. Программа должна иметь возможность работать с Google Таблицами с расписанием студентов очных программ Нижнего Новгорода Высшей школы экономики.
2. После подключения к таблицам, алгоритм должен извлекать необходимые данные и обрабатывать их, а именно: название предмета, фамилию и инициалы преподавателя, день недели, номер группы, время занятий, аудиторию и тип пар.
3. Полученная информация должна быть сохранена для дальнейшей обработки в удобном формате.
4. Программа должна уметь работать со случаями, в которых в одной ячейке располагается несколько предметов, разделяя их и заполняя для каждого занятия необходимую информацию.

**Требования к надежности:**

1. Программный модуль должен уметь работать с некорректными данными, поступающими на вход, и возможными ошибками, связанными с потерей соединения или устаревания токена и обрабатывать их, возвращая соответствующую ошибку.
2. Должны применяться безопасные способы для аутентификации для защиты конфиденциальности данных разработчиков при получении доступа к Google Таблицам.
3. Модуль должен быть эффективен и показывать необходимую производительность при сборе информации о расписании студентов из Google Таблиц.

**Требования к информационной и программной совместимости:**

1. Подпрограмма должна быть реализована на языке программирования Python, начиная с версии 3.10.7, при этом платформа, находящаяся на устройстве, используемом для запуска навыка голосового ассистента не играет значения.
2. Для работы с Google Таблицами модуль должен корректно взаимодействовать с Google API для доступа к данным из таблиц, что включает в себя аутентификацию, авторизацию и обмен данными.
3. Итоговое программное решение должно быть совместимо с другими модулями, предусмотренными в проекте, а именно получать на вход ссылку и возвращать на выходе данные для модуля поиска необходимых элементов расписания.
4. Программа должна быть реализована с использованием принципов ООП

**Требования к маркировке и упаковке**

Реализованное программное решение должно располагаться и быть опубликовано в каталоге Яндекс навыки. В свою очередь, специальные требования к маркировке и упаковке разрабатываемого программного модуля по парсингу Google Таблиц не предъявляются.

**Требования к транспортированию и хранению.**

Рассматриваемый программный модуль должен располагаться на платформе Yandex Cloud Functions или другой, которая будет предоставлять возможность использования Webhook URL в рамках кода общего проекта до тех пор, пока разработчик или работник платформы не внесет изменения или не удалит продукт. Специальные требования к транспортировке не предъявляются.

**Тестирование**

1. Необходимо протестировать программу на различных наборах входных данных, а именно, расписание различных образовательных программ, которое зафиксировано в таблицах, имеющих схожий формат.
2. Необходима проверка, что подпрограмма корректно обрабатывает различные форматы данных в таблице. Например, когда формат записи фамилии преподавателя, аудитории или других элементов расписания отличается в различных Google Таблицах.

**Стадии и этапы разработки**

1. Выявление и определение функциональных и иных требований к реализуемой подпрограмме.
2. Изучение формата данных в таблицах и необходимых операций с ними.
3. Разработка общей структуры модуля и определение основных компонентов, из которых он должен состоять.
4. Написание программного кода для работы с Google API, извлечения и обработки различных данных.
5. Проведение тестирования модуля для проверки корректности работы на различных образовательных программах и небольших изменениях в формате данных.
6. Исправление ошибок, возникших при тестировании.
7. Обеспечение совместимости с другими программными модулями рассматриваемого проекта и функционирования на выбранной платформе.
8. Интегрирование программного модуля в итоговый проект, тестирование его в рамках общего кода и исправление ошибок.
9. Написание технической документации по модулю.

**Анализ проблематики**

Основными задачами в рамках анализа проблематики являются: изучение источников, из которых доступно получение необходимой информации, в том числе их потенциальные возможности, формат хранения и структура данных о расписании, в рамках который они организованы. Кроме того, будут рассмотрены различные проблемы, связанные с разработкой и внедрением данного программного решения в общий проект. Следующая цель для изучения - преимущества и недостатки, которые демонстрирует рассматриваемый способ организации и получения информации в контексте разрабатываемого навыка. Также будут рассмотрены рамки и условия, в которых программный модуль разрабатывается, реализованные до текущего момента методы и подходы для решения задач схожей или такой же проблематики, в том числе реализованные навыки по обработке расписания, исходный код которых находится в открытом доступе.

В процессе изучения и анализа литературы, а также возможных источников расписания студентов НИУ ВШЭ было выявлено несколько потенциальных вариантов, к которым относятся: парсинг входящих файлов, поступающих на корпоративную почту и содержащих расписание, извлечение данных о расписании из Единого личного кабинета студента, использование сервиса РУЗ, а также итогового наиболее оптимального варианта - парсинга Google Таблиц с сайта с расписанием студентов НИУ ВШЭ Нижнего Новгорода.

Одним из решений, направленных на решение задачи о получении расписания, является использование и парсинг электронной почты. Данный метод рассматривался как наиболее универсальный, так как в независимости от кампуса позволяет получать информацию о расписании. Для его реализации требуется программный модуль, который бы смог находить файл, содержащий расписание и обрабатывать его. Во время изучения аспекта, касающегося формата, в котором на почту присылается расписание было выявлено, что оформление и даже тип данных отличаются не только у разных кампусов, но и образовательных программ. Так, например, расписание представляется, как в формате excel, так и pdf, и у некоторых образовательных программ оформление таблиц отличается. Таким образом, реализация данного способа в силу того, что невозможно создать универсальный программный модуль, который бы обрабатывал различные форматы расписания и типы файлов, была отменена.

Следующим возможным решением было выбрано использование подключения к РУЗ и получение информации из данного источника. Этот метод представлялся одним из наиболее универсальных и полноценных способов создания подпрограммы для скрапинга данных расписания. Информация из данного источника связана с конкретным студентом, то есть учитываются индивидуальные аспекты в расписании, такие как занятия по английскому языку, майнору и иные курсы, о которых не содержатся сведения в Google Таблицах или структурах, предоставляющих данные о групповом расписании. Также информация носит наиболее актуальный характер, так как постоянно обновляется. Использование данного способа получения необходимых сведений позволило бы сделать режим преподавателя, в силу того, что он содержит необходимую информацию. Существуют уже реализованные проекты в рамках исследуемой предметной области, что позволило бы ускорить разработку и учесть ошибки других разработчиков. Однако, в нынешней ситуации доступ к РУЗ организован только через внутренние сети НИУ ВШЭ или через VPN подключение к сетям университета и только для сотрудников, в рамках данных ограничений не было найдено способа, позволяющего реализовать подпрограмму с использованием рассматриваемого метода.

Помимо вышеперечисленных, Высшая школа экономики предлагает такой сервис, как Единый личный кабинет. Данная цифровая экосистема предлагает возможность просмотра расписания студентов, которое также является актуальным и обладает всеми преимуществами использования РУЗ, возможно, сервис и использует это решение. Расписание хранится в удобном формате, содержащем все необходимые сведения, была также найдена возможность скрапинга его в виде json-формате, но в рамках браузера при получении сURL xhr запроса, содержащего расписание. Основным ограничением, которое делает невозможным реализацию данного метода является необходимость авторизации каждого пользователя при входе на страницу ЕЛК и введение им конфиденциальных данных, включающих корпоративный логин и пароль, что является недопустимым при разработке навыка голосового ассистента Алиса.

В силу невозможности реализации остальных вариантов, программный модуль по извлечению информации о расписании студентов НИУ ВШЭ использует Google Таблицы, размещенные на сайте[2] в открытом доступе. На данный момент рассматриваемый способ парсинга является наиболее релевантным по причине того, что источник информации является практически единственным, где существует необходимые сведения и в схожем формате. Другие способы получения информации не полностью удовлетворяют заданным требованиям к реализуемой подпрограмме.

Google Таблицы – это таблицы в онлайн формате, которые представляют собой удобный инструмент для создания, хранения и обработки информации в виде таблиц. Электронная таблица – основной объект в Google Таблицах, который может содержать несколько листов, каждый из которых содержит структурированную информацию, содержащуюся в ячейках. Ресурс Spreadsheet представляет каждую электронную таблицу и имеет уникальное значение spreadsheetId, содержащее буквы, цифры, дефисы или символы подчеркивания. Идентификатор таблицы можно найти в URL-адресе Google Таблиц: https://docs.google.com/spreadsheets/d/ *spreadsheetId* /edit#gid=0.

Лист – страница или вкладка в электронной таблице. Ресурс Sheet представляет каждый лист и имеет уникальный заголовок и числовое значение sheetId. Данную информацию можно также обнаружить в URL-адресе Google Таблиц:https://docs.google.com/spreadsheets/d/ *spreadsheetId* /edit#gid=sheetId. В рамках исследуемой проблемы уникальный идентификатор листа не значительно важен, потому что расписание в основном располагается на первом листе.

Клетка – отдельное поле текста или данный на листе. Клетки или ячейки расположены в строках и столбцах и могут быть сгруппированы как диапазон ячеек. Ресурс CellData представляет каждую ячейку, но не имеет уникального значения идентификатора. Вместо этого координаты строки и столбца идентифицируют ячейки.

Обозначение А1 – синтаксис, используемый для определения ячейки или диапазона ячеек с помощью строки, содержащей имя листа, а также координаты начальной и конечной ячейки с использованием букв столбцов и номеров строк. Этот метод является наиболее распространённым и полезным при ссылке на абсолютный диапазон ячеек.

Обозначение R1C1 - Синтаксис, используемый для определения ячейки или диапазона ячеек с помощью строки, содержащей имя листа, а также координаты начальной и конечной ячейки с использованием номеров строк и номеров столбцов. Этот метод менее распространен, чем нотация A1, но может быть полезен при ссылке на диапазон ячеек относительно позиции данной ячейки.[3]

У Google таблиц существует свой API, который позволяет организовать взаимодействие с данными в электронной таблице. Работа с информацией включает в себя как чтение, так и изменение содержимого таблицы. Для организации взаимодействия с данным интерфейсом необходимо произвести несколько действий, позволяющий получить доступ к парсингу таблиц.

Информация, которая содержится в таблицах позволяет сделать наиболее универсальное решение, предоставляющее доступ к расписанию студентов групп различных специальностей. В таблицах содержатся сведения, которые включают в себя название, время, аудиторию, тип занятий и преподавателя. Доступ к данной информации предоставляется свободно без необходимости авторизации в сервисах Высшей школы экономики. Возникают трудности с определением расписания по английскому языку, занятиям по дисциплинам дополнительного профиля Minor и иным дополнительным курсам. Расписание предоставляется только для студентов Нижнего Новгорода. В рамках данного способа обработки расписания алгоритм реализации поиска расписания информации конкретного преподавателя представляется реализуемым только в формате просмотра всех таблиц и всех групп, что при наличии разницы в оформлении таблиц, включая, например, отсутствие инициалов при указании имени преподавателя, представляется сложным. Также рассматриваемый программный модуль является заложником формата самих таблиц, при изменении способа фиксации расписания или его оформления существующее решение потребует доработки или комплексного изменения в силу отсутствия алгоритма, позволяющего обрабатывать данные в независимости от формата. А именно при обработке каждой ячейки, содержащей название предмета отсутствует возможность найти сопряженные данные, так как их расположение представляется неизвестным и не было выявлено закономерностей, позволяющих при обработке конкретной ячейки находить время, аудиторию и другие атрибуты занятия. Сами данные организованы в виде совокупности ячеек, в верхней части таблицы располагаются названия корпусов. Каждая ячейка, содержащая название предмета также окрашена в определенный цвет, соответствующего корпуса. На 18-19 строках содержатся заголовки столбцов таблицы, а именно: день, время, номер группы и аудитория. Ниже в ячейках присутствует соответствующая информация. В ячейке под номером группы располагается название занятия, даты, во время которых оно проходит, если не подразумевается, что занятия в данный день недели и время постоянны, и фамилия преподавателя. Рассматриваемый формат и разница в оформлении у некоторых образовательных программ несут определенные технические ограничения, например, не всегда получается выделить фамилию преподавателя или тип предмета из-за либо их отсутствия, либо при указании только фамилии преподавателя. Однако, на данном этапе данный способ получения информации представляется наиболее полным.

Таким образом, Google таблицы являются самым оптимальным способом получения информации о расписании студентов, который соответствует требованиям конфиденциальности и возможностям, имеющимся у участников проекта.

В рассматриваемой предметной области существует несколько решений подобной задачи. Один из навыков требует ввода кода из бота, никакой дополнительной информации нет. Другое информационное решение под названием Помощник Вышки также не работоспособен, при вызове данного навыка выводится сообщение о том, что он не отвечает. У последнего навыка существует открытый код, расположенный на платформе GitHub. Навык также использует API РУЗ, к которому на данный момент доступа нет. При попытке запуска также выдается ошибка. По итогу, на данный момент рабочего решения, рассматриваемой в рамках проекта задачи в открытом доступе найти не удалось.

Что касается решений для других учебных заведений, то было найдена программа для Финансового университета, реализованная для собственного API расписания. Также для Томского Политехнического университета существует информационное решение, которое использует сайт, содержащий расписание. Остальные навыки, выполняющие подобные функции в рамках предметной области также применяют иные способы извлечения расписания, которые являются более корректными и универсальными.

**Анализ существующих решений, технологий и аналогов**

В рамках рассматриваемой предметной области существует несколько различных подходов, позволяющих реализовать получение необходимой информации из Google Таблиц. Так как информация о расписании студентов Высшей школы экономики располагается на сайте, то существует некоторое количество вариантов его извлечения, которые предоставляют возможность использовать различные языки программирования, таким образом общее число вариантов реализации технического решения вырастает многократно. Первый способ – использование существующих технологий и библиотек для получения данных с самого сайта, на котором располагается таблица с расписанием. Для данной задачи подходят различные языки программирования, такие как JavaScript, Python, PHP, C++ и другие, которые имеют разработанные библиотеки для веб-скрапинга. Второй способ – использование API Google Таблиц для работы уже с самими таблицами и обработки и извлечения данных из них. Рассматриваемое техническое решение также может быть также реализовано с применением широкого спектра языков программирования, таких как Python, Go, JavaScript, Java, Google Apps Script.

При получении информации с сайта используемый программный модуль отправляет HTTP-запросы к серверу, требуя то содержимое, которое расположено на самой странице. Существует два вида сайтов статические и динамические.

Первые состоят из постоянных страниц, которые не меняют содержимое в зависимости от пользователя. При выполнении запроса сервер сразу предоставит готовый HTML код в том формате, который был использован при его создании.

Динамические сайты представляют собой страницы, которые в зависимости от пользователя будут иметь различную организацию данных. Данный тип страниц не существует в исходном виде на сервере, а при каждом новом запросе они формируются. Сначала сервер находит нужный документ и отправляет его интерпретатору, который выполняет код из HTML-документа и сверяется с файлами и базой данных. После этого документ возвращается на сервер и затем отображается в браузере. Для интерпретации страниц на серверной стороне используются языки программирования Java, PHP, ASP и другие. [4]

В зависимости от типа сайта необходимо использование библиотек и технических решений, способных выполнять отправку запроса на сервер и получение нужного кода HTML для его дальнейшей обработки. Так, например, в языке Python существуют библиотеки requests и selenium, которые предназначены для парсинга разного формата данных, первая подходит только для извлечения данных из статических сайтов, вторая, в свою очередь, годится для получения информации в независимости от типа сайта.

В случае рассматриваемой проблематики, информационное решение должно уметь отправлять запросы и получать информацию со статического сайта, так как таблица, которая располагается на нем не изменяется и одинакова для любого пользователя. Для решения поставленной задачи, как было упомянуто ранее возможно использование нескольких языков программирования.

JavaScript – является одним из наиболее распространенных решений, подходящих для выполнения широкого круга задач. Это обосновано прежде всего наличием значительного количества библиотек и инструментов, которые доступны для решения задач получения информации с сайта и поддержкой со стороны сообщества разработчиков и энтузиастов. Также в пользу данного языка говорит простота синтаксиса и наличие большого спектра ресурсов для обучения. К числу библиотек, которые подходят для решения существующей задачи подходят Unirest, Axios и Cheerio, которые не используют безголовые браузеры и драйверы и могут быть использованы в рамках Yandex Functoins. Все рассматриваемые решения поддерживают различные HTTP-методы, такие как GET, POST и другие и позволяют выполнять основную задачу, а именно: получение HTML кода страницы с расписанием.

С++ - язык программирования, который является наиболее высокоуровневым по сравнению с JavaScript и Python. Данный факт дает ему преимущество в плане гибкости и скорости, которое следует за сложностью синтаксиса и концепции языка, а также низкой скоростью разработки. Способность получать доступ к низкоуровневым системным ресурсам дает возможность создавать настраиваемые решения для парсинга. Также C++ является компилируемым языком, что предоставляет ему преимущество по сравнению с JavaScript и Python, которые требуют наличие интерпретатора, что влечет большое использование ресурсов и более низкую производительность. К основным библиотекам, которые используются для парсинга сайтов относятся libcurl, Boost.Asio, htmlcxx. Данные решения также позволяют получать HTML-содержимое необходимого сайта и обрабатывать его для извлечения нужного текста.

Python, в свою очередь, предлагает значительное количество библиотек для веб-скрапинга, позволяющие организовать технические решения, которые будут уникальны для каждого проекта. Также одними из основных преимуществ данного языка являются простой синтаксис и активная поддержка со стороны энтузиастов.

Учитывая контекст рассматриваемой проблемы, а именно: использование другими участниками Python в качестве языка программирования, что позволяет удобно объединять разрабатываемые подпрограммы в проект, отсутствие необходимости в анализе большого количества данных, где была бы нужна производительность языка C++, наличие опыта разработки на Python, для реализации программного модуля по извлечению расписания был выбран данный язык.

Для решения поставленной задачи выбранное техническое решение предполагает использование таких библиотек как Requests, Beautiful Soup, и Selenium. Последняя имеет возможность парсинга динамических веб-сайтов, что не имеет необходимости при парсинге рассматриваемого сайта с таблицей. Кроме того, использование Selenium влечет необходимость установки соответствующего драйвера браузера, который запускается и используется данной библиотекой. Для выполнения этих условий в среде запуска необходимо присутствие виртуальной машины, которая бы запускала драйвер, что усложняет сам процесс разработки, также не было найдено решения для этой проблемы в Yandex Cloud Functions. Библиотека Requests, в свою очередь, предоставляет возможность парсинга веб-страниц буквально в пару строк. Необходимо отправить GET-запрос на необходимый сайт с помощью функции get и с применением метода text получить HTML-код страницы. Также отсутствует необходимость использования других решений, кроме самой библиотеки. Анализ же самого кода выполняется с помощью библиотеки Beautiful Soup, которая позволяет получать тексты, атрибуты, ссылки и другие данные из документа на основе тегов, классов, идентификаторов, атрибутов и других характеристик элементов. Например, вы можете найти все ссылки или изображения на странице с определенным классом или атрибутом. Однако в существующем контексте, когда необходимо получение всей информации и при условии, что не было найдено возможных закономерностей по названию имен классов, тегов или других элементов текстов, которые бы позволили каким-либо образом структурировать информацию, было принято решение использования и поиска другого технического средства. Пример получаемой информации: </span> НИС &quot;Анализ литературы&quot; - Савченко Л.В.</td><td class="s8" dir="ltr">301<br><br>303<br><br>301</td><td.

Помимо вышеупомянутой технологии также существует другой способ взаимодействия с информацией, а именно: применение API Google Таблиц, который поддерживает различные языки программирования такие как Python, Go, JavaScript, Node.js, Java и другие. По ранее указанным причинам, для работы с изучаемым техническим решением был выбран Python. Касаемо языка программирования, необходима версия Python, начиная с 3.10.7 и новее, также нужен модуль, который организовывает управление пакетами pip, имеющийся проект в Google Cloud Platform.

При работе с API Google таблиц необходимо определенная настройка своего проекта в Google Cloud Platform, откуда будут загружены учетные данные разработчика для авторизации при работе с таблицами. Выбранный способ реализации подпрограммы требует наличия клиентских библиотек Google для работы с Python: Google-api-python-client, Google-auth-httplib2, Google-auth-oauthlib.

Для работы с Google Таблицами также возможно использование других библиотек, таких как Gspread или Pygsheets, однако в контексте рассматриваемой проблемы, когда необходимо только получение значений из таблицы, данные модули являются эквивалентными, так как получение информации происходит в рамках одного метода и основной задачей является уже работа с полученными сведениями.

Для работы программы необходима настройка среды исполнения API Google Таблиц, которая подразумевает несколько шагов, указанный в Кратком руководстве по Python.[5] Прежде чем использовать API Google, необходимо включить их в проекте Google Cloud. Можно задействовать один или несколько API в одном проекте Google Cloud. Затем происходит настройка экрана согласия OAuth. Если используется новый проект Google Cloud для выполнения краткого руководства, то необходимо настроить экран согласия OAuth и добавить собственный аккаунт в качестве тестового пользователя. Для этого нужно:

1. В консоли Google Cloud выбрать >  **API и службы** > **Экран согласия OAuth**.
2. В качестве **типа пользователя** выбрать **«Внутренний»**, затем нажать **«Создать»** .
3. Заполнить форму регистрации приложения, затем нажать **«Сохранить и продолжить»**.
4. На данный момент можно пропустить добавление областей и нажать **«Сохранить и продолжить»**. В будущем, когда будет необходимость создания приложение для использования за пределами организации Google Workspace, необходимо изменить **тип пользователя** на **Внешний**, а затем добавить области авторизации, необходимые приложению.
5. Просмотреть сводку регистрации приложения. Чтобы внести изменения, нажать **«Изменить»**. Если регистрация приложения выглядит нормально, нажать **«Вернуться на панель управления»**.

Чтобы аутентифицировать конечных пользователей и получить доступ к пользовательским данным в приложении, необходимо создать один или несколько идентификаторов клиентов OAuth 2.0. Идентификатор клиента используется для идентификации одного приложения на серверах Google OAuth. Если приложение работает на нескольких платформах, необходимо создать отдельный идентификатор клиента для каждой платформы.

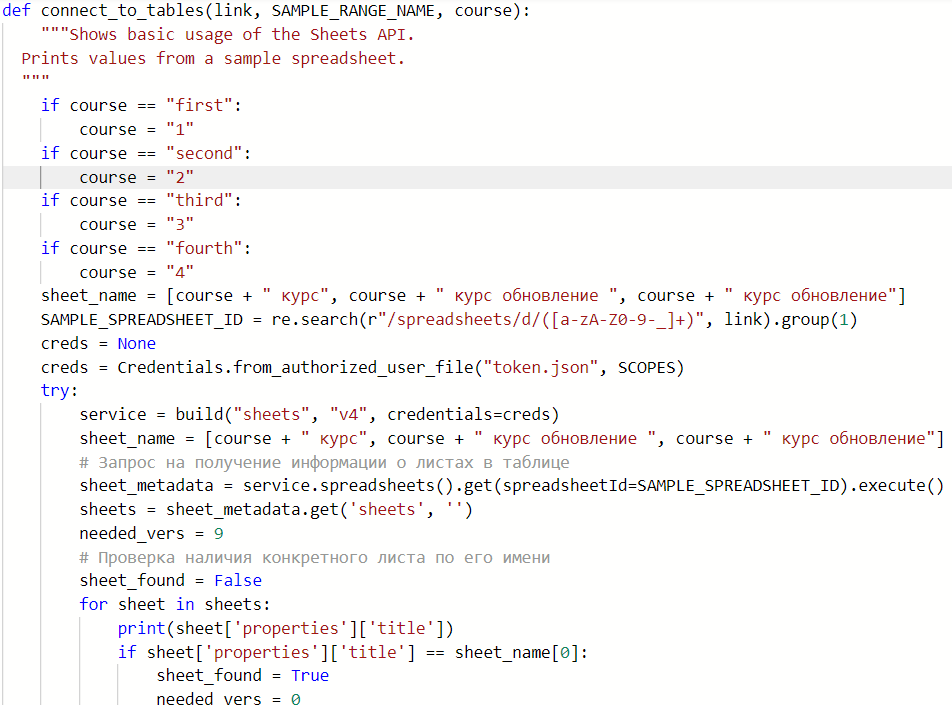
1. В консоли Google Cloud выбрать  > **API и службы** > **Учетные данные** .
2. Нажать **«Создать учетные данные»** > **«Идентификатор клиента OAuth** ».
3. Нажать **Тип приложения** > **Приложение для ПК** .
4. В поле **Имя** ввести имя учетных данных. Это имя отображается только в консоли Google Cloud.
5. Нажать **Создать** . Появится экран создания клиента OAuth, показывающий новый идентификатор клиента и секрет клиента.
6. Нажать **ОК** . Вновь созданные учетные данные появятся в разделе **«Идентификаторы клиентов OAuth 2.0».**
7. Сохранить загруженный файл JSON как credentials.json и переместить его в свой рабочий каталог.

Таким образом, для реализации программного модуля был выбран язык Python за счет отсутствия необходимости в высокой производительности, применение данного решения другими участниками проекта, что упрощает процесс интеграции программного модуля в совокупности с клиентскими библиотеками Google, так как при решении поставленной задачи существенной разницы между ней и аналогами не было обнаружено.

**Полученные результаты, методика испытаний и итоги анализа**

В процессе решения поставленной задачи был реализован программный модуль, состоящий из нескольких функций и обрабатывающий необходимую Google Таблицу с получением массива нужного расписания для всех групп образовательной программы. Для его работы требуется выполнение, помимо наличия указанных клиентских библиотек Google и файла с учетными данными пользователя, выполнение шагов по настройке среды, необходимо также импортирование библиотеки Re, которая будет использована для отделения из всех поступающих данных атрибутов занятия, а именно: времени, лектора, аудитории и типа занятия.

Первая функция, задействованная в программном модуле, называется connect\_to\_tables. На вход принимается переменная link, полученная, как результат функции take\_a\_link из подпрограммы другого участника проекта и представляющая собой ссылку на сайт с Google Таблицей расписания образовательной программы пользователя. Помимо этого, при запуске рассматриваемой функции необходимо передать в качестве аргумента переменную SAMPLE\_RANGE\_NAME - диапазон ячеек, который необходимо получить. Значения данной переменной отличается для очных и очно-заочных образовательных программ из-за того, что таблицы представлены для каждого вида обучения в различном формате и является фиксированным для всех образовательных программ каждого вида. Также на вход принимается переменная course, отвечающая за курс пользователя и необходимая для парсинга нужного листа, так как расписание всех курсов образовательной программы находятся в одной таблице.(рисунок 1)



Из полученной ссылки с помощью регулярного выражения делается срез строки с необходимым идентификационным номером таблицы, который требуется для получения всех данных. Для работы данной функции также необходимо наличие учетных данных пользователя в файле token.json. В изначальной версии функции, которая взята с сайта[2] в случае, если они отсутствуют, то запускаются методы, которые авторизуют пользователя через браузер и создают нужный файл. Однако среда Yandex Cloud Functions не поддерживает запись файлов, поэтому данная часть кода была удалена, но возникла необходимость периодически, в промежуток около 4-5 дней обновлять token.json вручную и загружать его на используемую платформу, так как учетные данные, записанные в данном файле, устаревают и возникает ошибка, не позволяющая продолжить работу функции.

После создания переменной с учетными данными вызываются методы библиотек Google, осуществляющие парсинг всей информации из таблицы нужного листа, которая возвращается функцией в виде переменной values.

В случае, если не будут найденные данные или не найден соответствующий лист вызываются исключения NoValuesExeption, No\_such\_sheet\_exeption, наследующиеся из класса Exception.

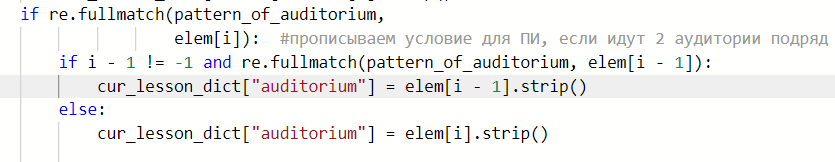
Данные, полученные в ходе скрапинга листа таблицы представляют собой двумерный вложенный список. Именно данный список обрабатывается функцией table\_parsing. Эта часть программы принимает в качестве аргументов link – ссылка на расписание образовательной программы, course – номер курса, которые необходимы для вызова функции connect\_to\_tables.

В рамках данной подпрограммы происходит обработка имеющейся информации о студентах. Сначала с помощью регулярный выражений выделяются названия групп, которые располагаются во втором вложенном списке переменной values. Затем создается список словарей schedule\_res, в которых ключом выступает название группы, а значением – вложенный словарь, где день недели – ключ, а значение еще один вложенный словарь, где уже идентификатором является строка “lessons”, по которой можно получить список словарей, в которых указаны необходимые атрибуты занятия: "name", "time", "auditorium", "lecturer", "lesson\_type". В данном виде идет организация всей информации, полученной в результате обработки переменной values.(рисунок2) 

После создания пустого макета происходит замена у каждого элемента вложенного списка values символов переносов строки и дефисов на пробелы, потому что, в зависимости от таблицы, между элементами переменной с ее содержанием может содержаться разное число данных знаков, которые для удобства изменяются на пробел.

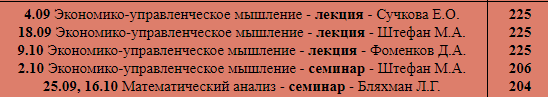
Далее происходит уже получение необходимых элементов расписания за счет выявленной в таблице закономерности, под которой подразумевается тот формат, в котором хранятся и передаются элементы вложенных списков переменной values. А именно: если ячейка хранит информацию о расписании, то место первого элемента списка зарезервировано под название дня недели, затем идет время, пустая строка, название предмета, аудитория, пустая строка, название предмета для следующей группы, аудитория и так далее. В случае, если занятий в данное время и день недели нет, то на месте имен предметов и аудиторий идут пустые строки. Также день недели обозначается только во вложенном списке с временем 8:00-9:20.

Используя выявленную закономерность, идет итерирование по всем элементам вложенных списков исходной переменной. Для определения, что элемент является временем или аудиторией используется регулярное выражение. Для каждой группы и дня недели заполняются словари cur\_lesson\_dict(рисунок 2). Инкрементация дня недели происходит, если был обнаружен элемент, который ему соответствует, а группы, если текущее число пустых строк превысило 3 или был найден элемент, который является аудиторией. Обнуление номера группы выполняется, в свою очередь, в случае перехода к новому вложенному списку. При тестировании было обнаружено, что в расписании 1 курса Программной инженерии в таблице встречается элемент, который вместо пустой строки представляет собой аудиторию, визуально не обозначенную в таблице. Для корректности работы было прописано условие(рисунок 3), во время возникновения которого в качестве аудитории берется предыдущий элемент.



Функция table\_parsing возвращает кортеж, состоящий из расписания в указанном формате с пустыми значениями для "lecturer", "lesson\_type", так как на данном этапе в связи с различным форматом таблиц их определить сложно, и список с названиями учебных групп.

С расписанием в описанном формате далее происходит работа с помощью функции lessons\_split, решающей проблемы, связанные с некоторыми особенностями записи информации в таблицах. Входными данными является та информация, которая возвращается предыдущей функцией. Первый шаг – работа с ячейками, в которых указано несколько предметов и аудиторий. Например, в расписании группы 23МББЭ1 (рисунок 4), данные приходят в формате, где в одной ячейке содержится информация о всех предметах.

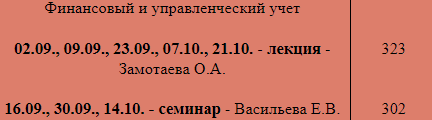


Первый вложенный цикл обрабатывает каждый элемент списка словаря по ключу “lessons”. В случае, если обнаруживается, что элемент расписания auditorium можно разбить на несколько элементов, то создаются списки, содержащие название, лекторов и аудитории каждых предметов. Производится итерирование по каждому элементу данных списков и заполняется переменная cur\_lesson\_dict, аналогичная той, что содержится в функции table\_parsing, затем она добавляется в исходный список schedule\_res, а элемент, содержащий информацию о нескольких занятиях удаляется.

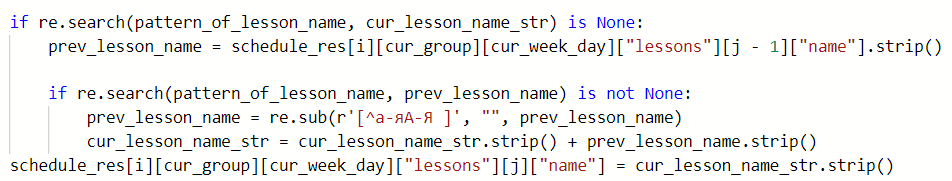
На данном этапе также происходит определение и заполнение параметра занятия lecturer. Для определения данного элемента пары используется регулярное выражение, которое учитывает варианты, когда преподаватель обозначается несколькими способами – комбинацией фамилии, инициалов и точек. Возникает проблема, когда лектор не обозначен или обозначен только фамилией, в этом случае подобрать необходимое регулярное выражение не удалось, так как оно соответствовало бы названию предмета.

Другой особенностью формата является наличие одной аудитории для двух занятий. Данный случай обрабатывается в следующем цикле. Идет работа также с элементом name каждого занятия и в случае, если при разбиении по паттерну преподавателя элемент становится списком длиной более 1, то идет итерация по полученному списку и занесение информации в исходный список всех групп. В исходной ячейке с несколькими предметами все значения приравниваются к пустой строке.

Затем идет обработка оставшихся предметов, которые не должны иметь никаких особенностей, связанных с наличием более одного предмета и аудитории в соответствующих значениях словаря. С применением регулярных выражений идет также итерация по каждому предмету и поиск паттернов, которые соответствуют лекторам и типам предметов. В данную часть подпрограммы также включена обработка ситуации. (рисунок 5)



В данном случае название предмета будет хранится только под соответствующим ключом предыдущего предмета. Если было обнаружено, что в name текущего занятия нет элемента, удовлетворяющего соответствующему паттерну, то выбирается часть названия предыдущего предмета и конкатенируется с соответствующей строкой текущего.(рисунок 6)



После данной обработки рассматриваемая функция возвращает список с полученными данными в указанном выше формате. Данный формат используется только в рамках тестирования и отладки подпрограммы. Для удобства работы с информацией он преобразуется в формат списка с объектами классов Group.

Последняя функция, которая присутствует в подпрограмме по парсингу расписания - transform\_to\_classes. На вход подается переменная schedule\_res. Происходит итерация по каждому элементу данного объекта с заполнением атрибутов объекта Group, которыми являются: название группы – поле класса string, день недели – list. В день недели, в свою очередь попадают объекты класса Lesson с заполненными атрибутами time, lecturer, name, lesson\_type и auditorium. Данный формат является итоговым выводом программного модуля, затем идет обработка полученной информации и нахождение необходимых элементов.

В случае возникновения ошибок в коде, которые в силу отличий формата заполнения таблиц могут появляться в различных местах программы, блок каждой функции помещен в конструкцию try – exception, при возникновении ошибки она печатается в консоли, а пользователю выводится сообщение: "Oшибка доступа к серверу, давайте я расскажу вам последние новости".

Для тестирования функции были выбраны расписания различных образовательных программ и курсов. Проверка работоспособности проводилась в рамках IDE PyCharm.

Сначала проверялось время работы подпрограммы. Для этого были использованы таблицы с расписанием во втором модуле. Время колебалось от 1 до примерно 2.6 секунд, что в контексте, когда ответ навыка ограничен 4.5 секундами представляется приемлемым.

Помимо этого, была протестирован факт запуска без ошибок программного модуля на всех образовательных программах Нижнего Новгорода в 1 и 2 модулях, было выявлено, что алгоритм выполняется на всех направлениях и курсах кроме Управления бизнесом в силу особенностей заполнения таблиц.

Также была проверена корректность отображения элементов расписания на выборке во втором модуле(таблица 1)

|  |  |  |
| --- | --- | --- |
| **Данные** | **Корректность отображения** | **Неучтенные элементы** |
| ОП - Юриспруденция Группы - 23Ю | Отделение нескольких предметов и одной аудитории корректно и все также | Не учтены случаи, когда лекция и семинар в одной клетке, некорректное отображение типа предмета |
| ОП - Математика Группы - 21ФМ | Абсолютно корректно | Нет |
| ОП - Прикладная математика и информатика  Группы - 22ПМИ | Абсолютно корректно | Нет |
| ОП - Бизнес-информатика  Группы - 22БИ | В целом корректно | Некорректное отделение 30.11 МБП как отдельного предмета, в параметр lecturer записан только один лектор |
| ОП - Компьютерные науки и технологии  Группы - 23КНТ | Абсолютно корректно | Нет |
| ОП - Цифровой маркетинг  Группы - 23ЦМ | Некорректно | Предметы во вторник в одной аудитории не отделяются, отделение слова отмена в отдельный предмет |
| ОП - Экономика | Некорректно | Отсутствие разделения предметов в одной аудитории, лишнее разделение, некоторые занятия не учтены |

**Список использованных источников**

1. Рынок разговорного AI в России 2020-2025[Электронный ресурс].: официальный сайт- <https://just-ai.com/blog/issledovanie-rynok-razgovornogo-ii-v-rossii-2020-2025> (дата обращения: 16.01.2024)
2. Сайт с расписанием Высшей школы Экономики Нижнего Новгорода[Электронный ресурс].: официальный сайт - <https://nnov.hse.ru/uch/schedule> (дата обращения: 19.01.2024)
3. Обзор API Google Таблиц: официальный сайт[Электронный ресурс]. - <https://developers.google.com/sheets/api/guides/concepts?hl=ru> (дата обращения: 20.02.2024)
4. Статические и динамические сайты сегодня: какие лучше и почему[Электронный ресурс]. — URL: https://o.jino.ru/journal/articles/staticheskie-dinamicheskie-sayty/ (дата обращения: 23.02.2024).
5. Краткое руководство по Python [Электронный ресурс]. — URL: <https://developers.google.com/sheets/api/quickstart/python?hl=ru> (дата обращения: 03.03.2024)
6. Документация языка Python: [Электронный ресурс]. официальный сайт- <https://www.python.org/doc/> (дата обращения: 04.03.2024)
7. Документация Yandex Cloud: [Электронный ресурс] официальный сайт.т - <https://yandex.cloud/ru/docs/functions/tutorials/alice-skill?utm_referrer=https%3A%2F%2Fyandex.ru%2F> (дата обращения: 05.03.2024)
8. Школа Алисы: официальный сайт[Электронный ресурс]. — URL: https://events.yandex.ru/events/webinars/alisa-school/index (дата обращения: 06.03.2024)