ПЕРВОЕ ВЫСШЕЕ ТЕХНИЧЕСКОЕ УЧЕБНОЕ ЗАВЕДЕНИЕ РОССИ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОРНЫЙ УНИВЕРСИТЕТ

ОТЧЕТ ПО ПРАКТИКЕ

Обучающийся (ФИО):	Шифр
Плотникова Полина Сергеевна	200233

Вид практики:

Производственная практика

Место прохождения практики:

Учебно-научный центр цифровых технологий, Санкт-Петербург, В. О. 21 линия д. 2

Сроки прохождения практики:

22.06.2022 – 19.07.2022

Тема:

Разработка информационно-аналитической системы мониторинга публикаций на базе наиболее популярных международных баз цитирования

2. Содержание практики (вид работы, внедренные научные решения и инновации в производственной деятельности организации, сроки выполнения

Обучающийся ____19.07.2022 П. С. Плотникова
(подпись, дата) (инициалы, фамилия)

3.Заключение руководителя

_____

_____

Руководитель практики _____ _____
(подпись, дата) (инициалы, фамилия)

Санкт-Петербург, 2022 г.

Содержание

ПЕРВОЕ ВЫСШЕЕ ТЕХНИЧЕСКОЕ УЧЕБНОЕ ЗАВЕДЕНИЕ РОССИ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОРНЫЙ УНИВЕРСИТЕТ

ЗАДАНИЕ НА ПРАКТИКУ

Обучающийся (ФИО) Шифр группы

Плотникова Полина Сергеевна ИАС-20-1

Вид практики: Производственная практика

Срок проведения практики: 22.06.2022 – 19.07.2022

Руководитель практики от ВУЗа: Анкудинов И.Г.

Руководитель практики от организации: Жуковский В.Е.

Руководитель практики

План практики

№	Вид работы	Срок выполнения	Отметка о выполнении
1.	**Разработка и отладка программного кода** Формализация и алгоритмизация поставленных задач. Написание программного кода с использованием языков программирования. Определения и манипулирования данными. Оформление программного кода в соответствии с установленными требованиями. Работа с системой контроля версий. Проверка и отладка программного кода.	22.06 – 29.06	
2.	**Проверка работоспособности и рефакторинг кода ПО** Разработка процедур проверки работоспособности и измерения характеристик. программного обеспечения. Разработка тестовых наборов данных. Проверка работоспособности программного обеспечения. Рефакторинг и оптимизация программного кода	30.06 – 6.07	
3.	**Исправление дефектов, зафиксированных в БД дефектов.** Интеграция программных модулей и компонент и верификация выпусков программного продукта. Разработка процедур интеграции программных модулей. Осуществление интеграции программных модулей и компонент и верификации выпусков программного продукта.	7.06 – 19.07	

Руководитель практики от ВУЗа _____ _____
(подпись, дата) (инициалы, фамилия)

Руководитель практики от предприятия _____ _____
(подпись, дата) (инициалы, фамилия)

Обучающийся _____ _____
(подпись, дата) (инициалы, фамилия)

ПЕРВОЕ ВЫСШЕЕ ТЕХНИЧЕСКОЕ УЧЕБНОЕ ЗАВЕДЕНИЕ РОССИ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОРНЫЙ УНИВЕРСИТЕТ

ДНЕВНИК ПРАКТИКИ

Студент (ФИО) — Шифр группы

Плотникова Полина Сергеевна — ИАС-20-1

Вид практики

Производственная практика

Место прохождения практики

Учебно-научный центр цифровых технологий, Санкт-Петербург, В. О. 21 линия д. 2

Срок прохождения практики

22.06.2022 – 19.07.2022

Руководитель практики **от организации**

Дата	**Содержание работ**	**Основные результаты**	**Отметка руководителя**
22.06	Ознакомление с заданием	Составление поэтапного плана выполнения задания.	
23.06	Работа в приложении Postman	Изучены принципы работы приложения Postman, успешно запрошены данные по API с помощью приложения.	
24.06	Работа в приложении Postman	Проведен анализ полученных данных и составлен перечень статистик, которые можно высчитать.	
27.06	Работа в приложении Postman	Проведены тесты по поиску необходимых данных в Postman.	
28.06	Сбор данных с сервиса Google	Реализован сбор данных с различных страниц Google Scholar в формате json. Изучен модуль serpapi.	

29.06	Сбор имен профилей	Выполнен сбор имен профилей Санкт-Петербургского горного университета с первой страницы. Реализован перевод имени на английский или русский язык, если такового не имеется.	
30.06	Сбор информации об электронной почте	Написан код для сбора информации об электронной почте профилей.	
1.07	Сбор информации о статьях авторов с определенного года	Написан алгоритм, собирающий данные по каждому автору только с конкретного года по нынешний. Реализован переход на следующую страницу в профиле.	
4.07	Подсчет количества статей	Написан код, подсчитывающий количество статей автора.	
5.07	Подсчет количества цитирований	Написан код, подсчитывающий количество цитирований статей.	
6.07	Сбор статистики, включающий соавторов	Написан код, подсчитывающий количество соавторов.	
7.07	Сбор статистики, включающий соавторов	Написан код, подсчитывающий количество статей, где рассматриваемый автор является единственным автором.	
8.07	Сбор статистики, включающий соавторов	Написан код, подсчитывающий количество статей, где рассматриваемый автор является первым в списке авторов (не считая, где он единственный).	
11.07	Сбор статистики, включающий соавторов	Написан код, подсчитывающий количество статей, где рассматриваемого автора нет в списке авторов.	
12.07	Сбор информации с определенного количества профилей	Реализован алгоритм, применяющий поиск информации по заданному количеству профилей.	

		Написан код для перехода на профиль следующей страницы.	
13.07	Сокращение кода	Реализована оптимизация алгоритмов (совмещение нескольких алгоритмов в один).	
14.07	Вывод статистики	Написан код для вывода собранной информации на страницу в виде таблицы.	
15.07	Проверка работы	Проведена проверка множества возможных запросов с разным количеством лет и профилей.	
18.07	Визуальное оформление	Создан файл css со стилями для оформления данных.	
19.07	Написание отчета	Написан отчет о прохождении производственной практики.	

Введение

Производственная практика является важнейшим звеном в системе подготовки высококвалифицированных специалистов, также, как и учебный процесс.

Цель практики: научиться разрабатывать информационно-аналитическую систему мониторинга публикаций на базе популярных международных баз цитирования Scopus, Web of Science, Google Scholar, ORCID, Publons, ResearchGate, Elibrary.

Задачи практики:

1. Научиться работать с API;
2. Уметь использовать приложение Postman;
3. Реализовать пользовательский интерфейс для визуализации информации и аналитики.

Основная часть

Центр «Цифровых технологий» - опорный научно-исследовательский и образовательный кластер для интеграции сквозных цифровых технологий и решений. Его сотрудники занимаются реализацией и исследованием наиболее актуальных и перспективных на сегодняшний день технологий и концепций. Среди них – цифровые двойники, большие данные, дополненная реальность, робототехника и численное моделирование.

Основной вычислительный узел Центра - сервер FORSITE 2U RS2-2049-24HS со следующими характеристиками: 112 ядер CPU, терабайт оперативной памяти, 10000 графических ядер CUDA. Кластер позволяет производить вычисления любой сложности с применением программного обеспечения ведущих компаний-разработчиков: Ansys, Rocky DEM, Flownex SE, Dassault Systemes, AVEVA Group.

В структуру Центра входят аудитории и лаборатории, созданные в сотрудничестве с такими компаниями как Новатэк, Caterpillar и Schneider Electric. Благодаря этим коллаборациям, студенты и партнеры вуза имеют возможность с помощью 3D- и VR-технологий обучаться и повышать компетенции в области увеличения эффективности электропотребления предприятий, управления добычей нефти и газа на шельфе, руководства горнотранспортными процессами и техникой.

Задачи центра:

1. Создание ключевых условий, способствующих подготовке высококвалифицированных кадров для цифровой экономики.

2. Внедрение цифровых технологий для модернизации компаний минерально-сырьевого и топливно-энергетического комплекса.

3. Обучение сотрудников компаний-партнеров работе с новыми продуктами цифровизации.

4. Развитие сотрудничества в области высоких технологий с профильными компаниями и другими университетами.

5. Пополнение научной базы в сфере цифровых технологий, публикация результатов исследований в научных журналах мирового масштаба.

Должность: Практикант

Функциональные обязанности: Разработка информационно-аналитической системы мониторинга публикаций на базе наиболее популярных международных баз цитирования.

Характеристика программного обеспечения ЛВС организации: Python, C++, C#, Java script, Windows, Astra Linux.

Паспорт производственных компетенций

Паспорт производственных компетенций, сформированных при прохождении практической подготовки в рамках производственной (производственно-технологической) практики

по направлению подготовки 09.03.01

«*Информатика и вычислительная техника*»

профиль Автоматизированные системы обработки информации и управления

№ п/п	Наименование производственной компетенции	Оценка уровня освоения компетенций, сформирована/ не сформирована	Примечание
1.	*Способность выполнять работы и управлять работами по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы*		
2.	*Способность осуществлять концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности*		
3.	*Способность оценивать, выбирать и создавать варианты архитектуры программного средства*		
4.	*Способность разрабатывать модели компонентов информационных систем, включая модели баз данных и модели интерфейсов «человек - электронно-вычислительная машина»*		
5.	*Способность разрабатывать компоненты программно-аппаратных комплексов, Программное обеспечение, базы данных, используя современные инструментальные средства и технологии программирования, включая проектирование, отладку, проверку работоспособности и модификации ПО*		
6.	*Способность сопрягать программно-аппаратные средства в составе информационных и автоматизированных систем*		

Руководитель практики
от кафедры

_____
(подпись)

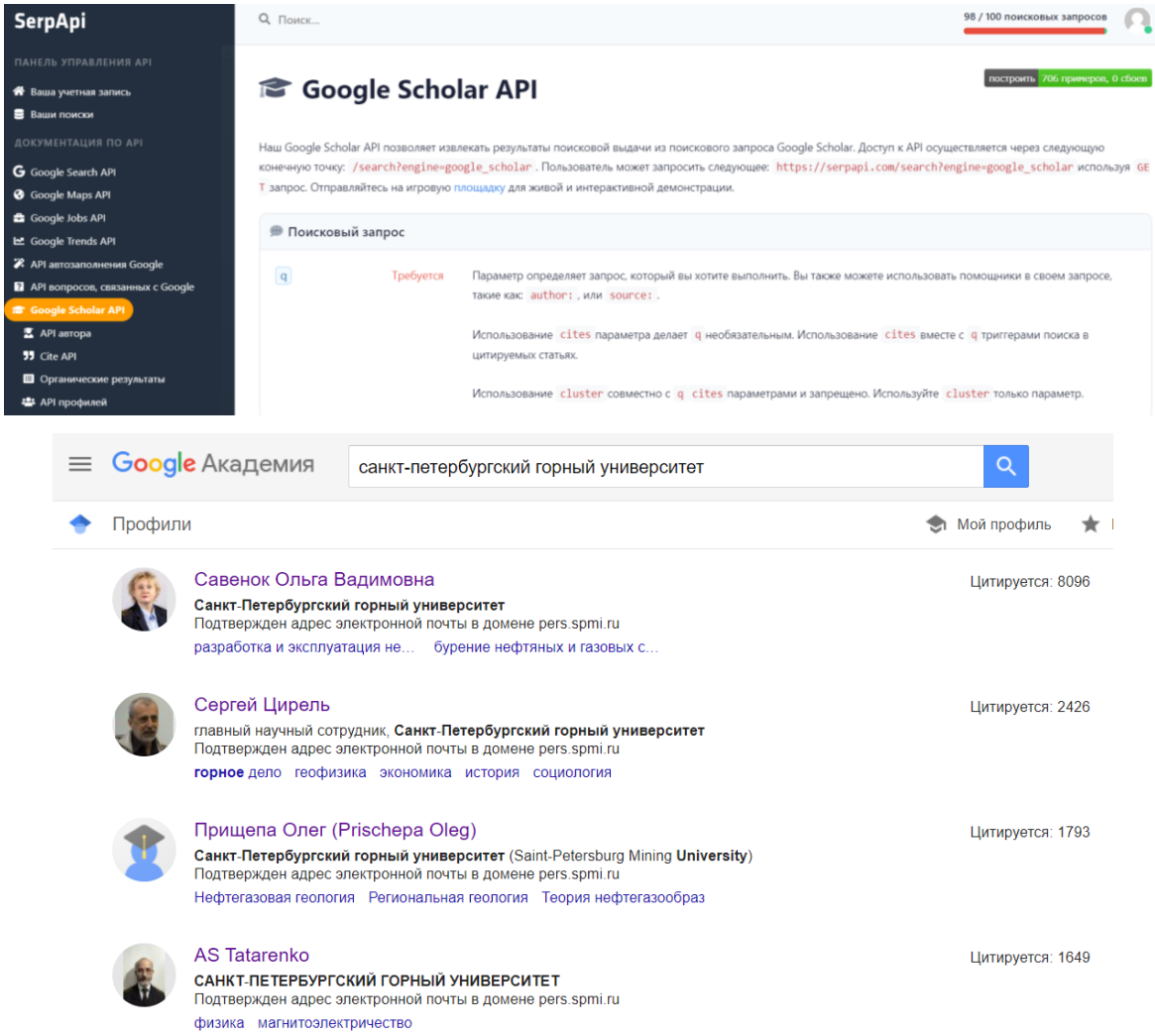
Перечень производственных компетенций сформирован в полном объеме и соответствует программе производственно-технологической практик

Руководитель практики
от организации, предприятия

_____(подпись)

1. Работа с API. Postman

В качестве базы цитирования был взят сайт Google Scholar. Извлекать данные позволяет Google Scholar API, ограничение – 100 поисковых запросов бесплатно или 30 000 платно в месяц.



Запрашивать данные можно только при наличии индивидуального ключа (‘key’), который дается пользователю после регистрации на сайте.

Для получения данных были использованы следующие запросы:

1. https://serpapi.com/search.json?engine=google_scholar_profiles&mauthors= Санкт-Петербургский горный университет&api_key='key'&after_author= ‘id’

Параметр	Значение
engine=google_scholar_profiles	Устанавливается обязательное значение google_scholar_profiles, чтобы использовать механизм API Google Scholar.
mauthors=Санкт-Петербургский горный университет	Параметр определяет профили Санкт-Петербургского горного университета

after_author=‘id’	Параметр по специальному токену, находящемуся в конце страницы, позволяет переходить на следующую страницу профилей

2. https://serpapi.com/search.json?engine=google_scholar_author&author_id=’author’&api_key='key'&sort=pubdate&num=100&start=n

Параметр	Значение
engine=google_scholar_author	Позволяет получать данные со страницы пользователя Google Scholar.
author_id =’author’	Обязательный параметр, позволяющий определять автора по его id
sort=pubdate	Сортирует статьи по дате публикации
num=100	Вывод 100 статей (максимум) в одном запросе
start=n	Позволяет получать следующую страницу в профиле, где n – количество пропускаемых статей

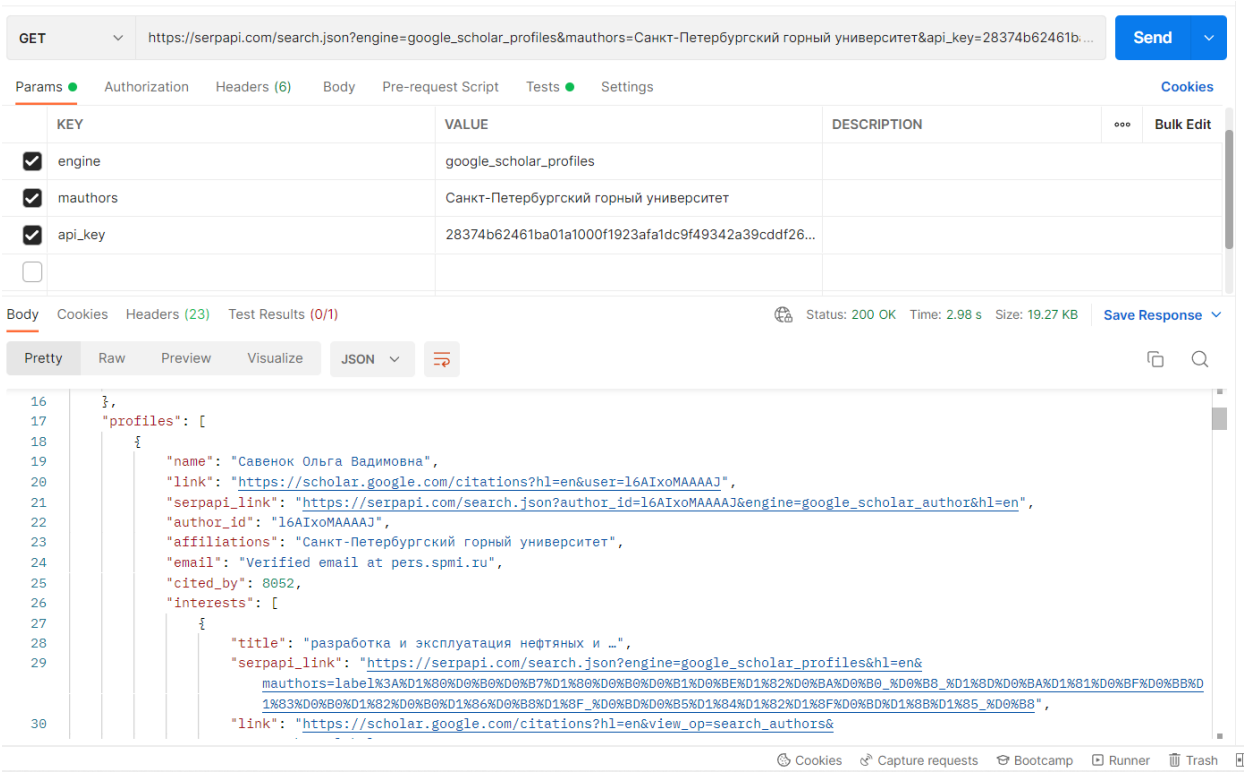
3. https://serpapi.com/search.json?engine=google_scholar_author&view_op=view_citation&citation_id=’citation’&api_key=’key’

Параметр	Значение
engine=google_scholar_author	Позволяет получать данные со страницы пользователя Google Scholar.
view_op=view_citation	Параметр используется для просмотра определенных частей страницы, в данном случае для просмотра статей
citation_id=’citation’	Параметр используется для получения полной информации об отдельной статье

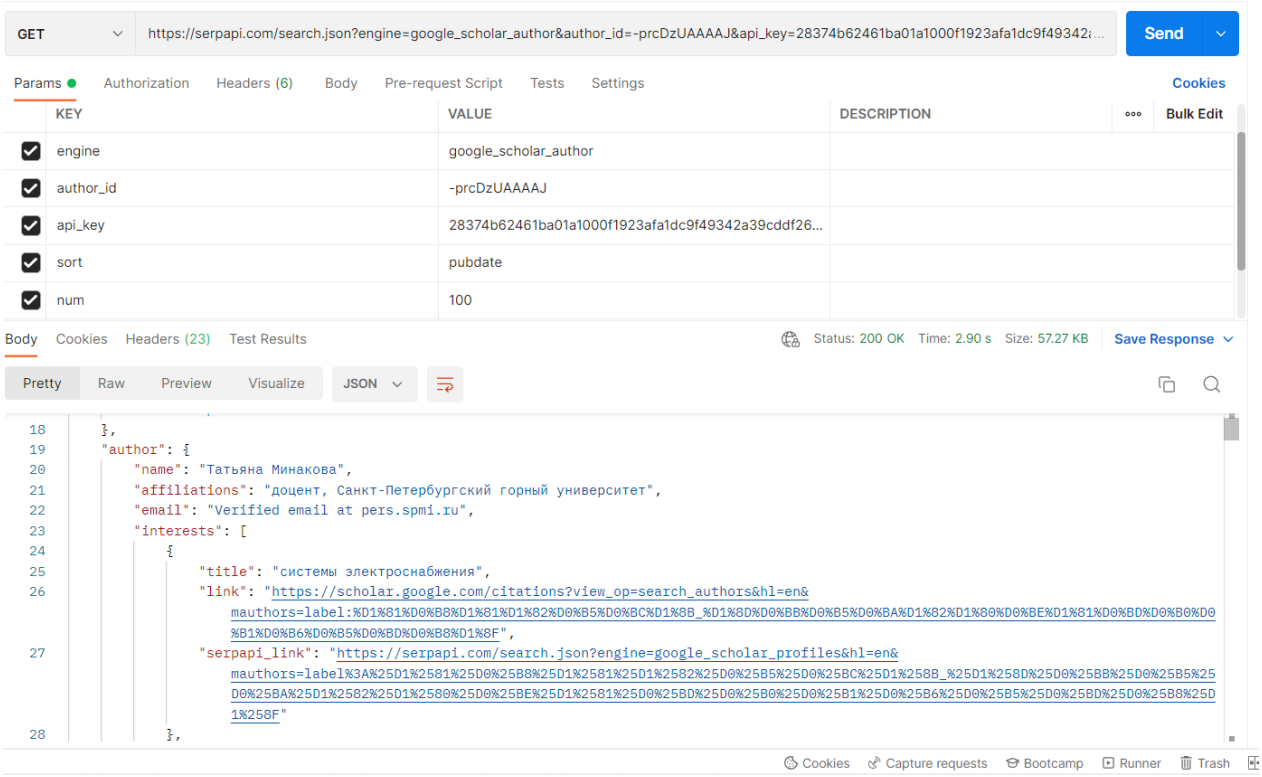
Postman - это классическое приложение, способное выполнять запросы API к любому API HTTP. Он обычно используется для тестирования и изучения API-интерфейсов.

Вывод данных при запросах в Postman:

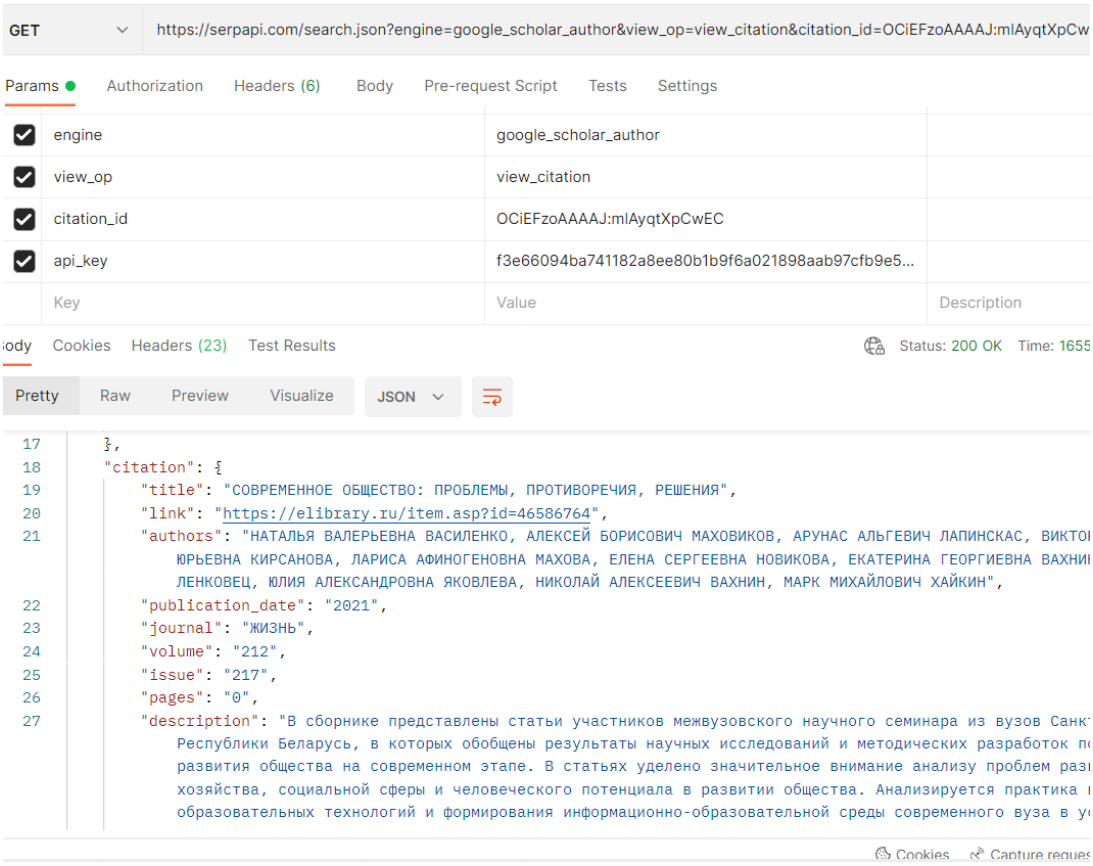
1. Данные json страницы с профилями Санкт-Петербургского горного университета:



2. Данные json одного из профилей:

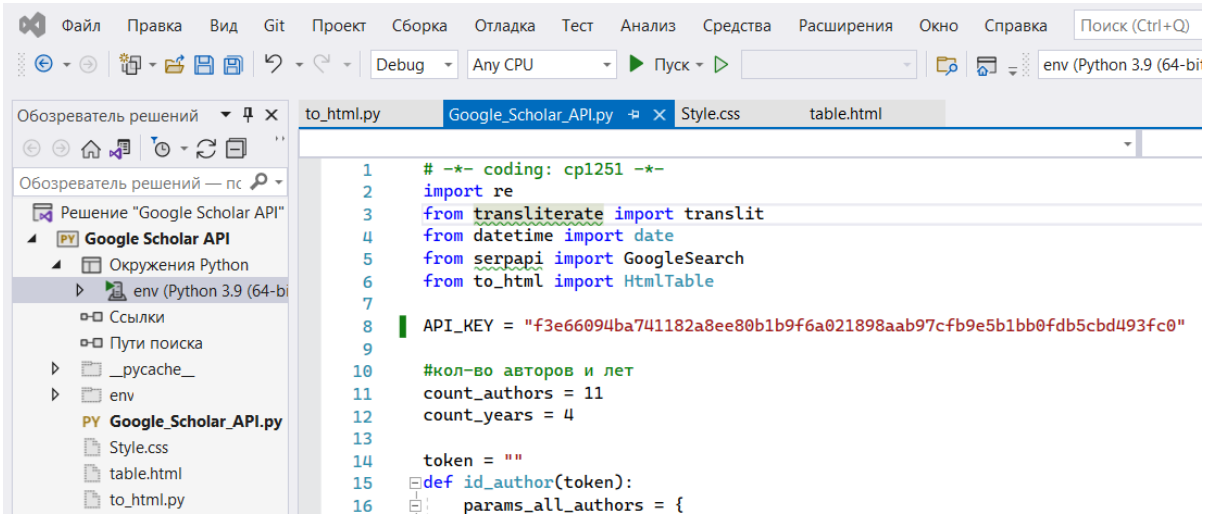


3. Данные json одной статьи:



2. Серверная часть

Код написан в файле Google_Search_API.py на ЯП Python 3.9. Для проекта было создано виртуальное окружение и импортированы модули re, transliterate, datetime, serpapi и файл to_html.py, который создает html файл.



Модуль Re для регулярных выражений. Regex, или регулярные выражения – язык для работы с текстом. Он позволяет производить поиск, замену и другие операции.

Модуль transliterate представляет собой двунаправленный транслитератор текста для Python. Транслитерирует (unicode) строки в соответствии с правилами, указанными в

языковых пакетах, то есть заменяет все русские буквы на английские и наоборот по правилам транслитерации.

Модуль datetime предоставляет классы для обработки времени и даты разными способами.

Модуль serpapi – это API, позволяющий запрашивать данные у сервисов Google.

Написанный код собирает и анализирует данные разного количества профилей и разных временных промежутков (от определенного года по текущий). Количество профилей и последних лет можно настроить в коде в переменных count_authors и count_years соответственно.

Все данные берутся у профилей Санкт-Петербургского горного университета. Собирается и обрабатывается следующий список данных:

1. Имя на русском;
2. Имя на английском;
3. Email;
4. Количество статей;
5. Количество цитирований;
6. Количество соавторов;
7. Количество статей, где рассматриваемый автор является единственным автором;
8. Количество статей, где рассматриваемый автор является первым в списке авторов (не считая, где он единственный);
9. Количество статей, где рассматриваемого автора нет в списке авторов.

Пункты 4-9 являются информацией за конкретное количество лет.

Описание кода:

1. Функция записывает несколько страниц с профилями

```
token = ""
def id_author(token):
    params_all_authors = {
        "engine": "google_scholar_profiles",
        "mauthors": "Санкт-Петербургский горный университет",
        "api_key": API_KEY,
        "after_author": token
        }
    search = GoogleSearch(params_all_authors)
    result_all_authors = search.get_dict()
    return result_all_authors
```

Используется в:

```
#запись всех id
all_id = []

while len(all_id) < count_authors:
    result_all_authors = id_author(token)
    for a_id in result_all_authors["profiles"]:
        if len(all_id) < count_authors:
            all_id.append(a_id["author_id"])
    token = result_all_authors["pagination"]["next_page_token"]
```

2. Функция передает id профиля и собирает все данные и статистику:

```
def info_author(author_id):
    params_author = {
        "engine": "google_scholar_author",
        "author_id": author_id,
        "api_key": API_KEY,
        "sort": "pubdate",
        "num": 100
        }
    search = GoogleSearch(params_author)
    result_author = search.get_dict()

    #ИМЯ
    name = result_author["author"]["name"]
    name_ru = ' '.join(re.findall(r"\b([а-яА-ЯёЁ]+)\b", name))
```

Подробнее о функции info_author(author_id):

Записывает имя автора и данные о том, на каком сайте подтверждена его эл. почта.

```
#ИМЯ
name = result_author["author"]["name"]
name_ru = ' '.join(re.findall(r"\b([а-яА-ЯёЁ]+)\b", name))
name_en = ' '.join(re.findall(r"\b([a-zA-Z]+)\b", name))

if len(name_ru) == 0:
    name_ru = translit(name_en, "ru")
if len(name_en) == 0:
    name_en = translit(name_ru, language_code = "ru", reversed = True)
    name_en = re.sub(r'[^\w\s]','', name_en)

all_name = name_ru.split(" ") + name_en.split(" ")
#EMAIL
email = result_author["author"]["email"]
```

Определяется текущий год и временной промежуток для поиска и во все возвращающиеся данные изначально записываются нули.

```
#начальные значения
today = date.today()
today_year = today.year

articles = {}
cit = {}
articles["Всего"] = 0
cit["Всего"] = 0
for i in range(count_years):
    articles[int(today_year - i)] = 0
    cit[int(today_year - i)] = 0

co_authors = 0
author_1 = 0
author_first = 0
without_author = 0
```

Проверяется наличие статей за установленный временной промежуток. Если их нет, то функция заканчивает работу выводом «нулевых» данных. Если есть, то собираются все данные профиля за определенные года и считается количество статей и цитирований за каждый год отдельно и в сумме.

```
if int(result_author["articles"][0]["year"]) <= int(today_year - count_years):
    return(name_ru, name_en, email, articles, cit, co_authors, author_1, author_first, without_author)
#КОЛ-ВО СТАТЕЙ И ЦИТИРОВАНИЙ
all_year = []
start = 0
y = today_year

while True:
    for year in result_author["articles"]:
        if year["year"] != "":
            y = int(year["year"])
            if y > (today_year - count_years):
                all_year.append(year) #вся инф за определенные года
                articles[y] += 1      #кол-во статей
                if year["cited_by"]["value"]:
                    cit[y] += int(year["cited_by"]["value"])   #кол-во цитирований
    if y > (today_year - count_years):
        start += 1
        result_author = Next_Page(start, params_author)  #переход на след страницу
    else:
        break

for i in articles:
    articles["Всего"] += int(articles[i])
    cit["Всего"] += int(cit[i])
```

Далее идет работа с данными об авторах статей. Записываются все авторы. Если список авторов статьи неполный, то запрашивается страница статьи с полным списком ее авторов.

```
#АВТОР И СОАВТОРЫ
all_authors = []

for author in all_year:
    if author["year"] != "":
        string = author["authors"]
        if ", ..." in string:
            citation_id = author["citation_id"]
            result_article = article(citation_id)
            all_authors.append(result_article["citation"]["authors"])
        else:
            all_authors.append(author["authors"])  #все авторы
```

Для обработки данных по авторам статей следует привести имена авторов к единому виду и использовать только фамилии. В этом же цикле можно посчитать количество статей, в которых рассматриваемый автор является единственным автором.

```
all_au = []
all_sn = []
all_surname = ""

for string in all_authors:
    all_au.append(string.lower().title())

for string in all_au:
    if "," in string:
        st = string.split(",")
        all_surname = ""
        for s in st:
            surname = ' '.join(s.split()[-1:])
            all_surname += surname + " "
        all_sn.append(all_surname)     #только фамилии
    else:
        surname = ' '.join(string.split()[-1:])
        all_sn.append(surname)
        for n in all_name:
            if n == surname:
                author_1 += 1   #этот автор является единственным автором статьи
```

Перебрав список авторов статей, можно найти количество статей, где рассматриваемый автор является первым в списке авторов (не считая, где он единственный) и количество статей, где рассматриваемого автора нет в списке авторов.

```
str_surname = []
author_f = 0
au = 0

for name in all_sn:
    n = name.split()
    for s in n:
        str_surname.append(s)
        if s in all_name:
            au += 1
            if n[0] == s:
                author_f += 1

if articles["Всего"] >= au:             #автора нет в списке авторов статьи
    without_author = articles["Всего"] - au

author_first = author_f - author_1      #автор записан первый среди соавторов
```

Вывод всех соавторов должен осуществляться с помощью запроса с параметром view_op=list_colleagues. Он работает, если сам автор добавил другого автора к себе в соавторы. Однако в нашем случае авторы не добавляли соавторов, поэтому вместо готового запроса можно написать код для сбора соавторов и их подсчета.

```
list_not_author = []
ln_author = []

for word in str_surname:
    count_name = 0
    count = 0
    for fragment in all_name:
        count_name += 1
        if fragment != word:
            count += 1
    if count == count_name:
        list_not_author.append(word)

for word in list_not_author:
    word_en = translit(word, language_code='ru', reversed=True)
    if word_en not in ln_author:
        ln_author.append(word_en)
co_authors = len(ln_author)    #кол-во соавторов

return(name_ru, name_en, email, articles, cit, co_authors, author_1, author_first, without_author)
```

После подсчета соавторов функция возвращает все статистические данные по профилю.

3. Функция переходит на следующую страницу в профиле и используется при сборе всех данных профиля за определенные года:

```
def Next_Page(st, params):
    params["start"] = st*100
    search = GoogleSearch(params)
    result_page = search.get_dict()
    return result_page
```

4. Функция возвращает все данные определенной статьи и используется в сборе всех авторов статьи:

```
def article(citation_id):
    params_articles = {
    "engine": "google_scholar_author",
    "view_op": "view_citation",
    "citation_id": citation_id,
    "api_key": API_KEY
        }
    search = GoogleSearch(params_articles)
    result_article = search.get_dict()
    return result_article
```

3. Клиентская часть

Все возвращенные данные функции info_author(author_id) передаются в цикл, который передает их в файл to_html.py.

```
for a_id in all_id:
    author_id = [a_id]
    info = info_author(author_id)
    table.to_table(info)
```

В файле to_html.py код выполняет функцию сбора таблицы тегами html с принятыми в него данными.

to_html.py* | Google_Scholar_API.py* | Style.css | table.html

```
4   def __init__(self, headers):
5       # Создаём класс, изначально прописав заголовки полей
6       self.html = None
7       self.clean_table(headers)
8
9   def to_table(self, line):
10      # Дописываем строку (1 автора) в таблицу
11      self.html += '<tr>'
12      for i in line:
13          data = i
14          if isinstance(i, dict):
15              data = ''.join([f'{j}: {i[j]}<br>' for j in i])
16          self.html += f'<td>{data}</td>'
17      self.html += '</tr>'
18
19  def clean_table(self, headers):
20      self.html = '<table class="table"><thead><tr>'
21      for i in headers:
22          self.html += f'<th>{i}</th>'
23      self.html += '</tr></thead><tbody>'
24
25  def get_table(self):
26      return self.html + '</tbody></table>'
27
```

В файле Google_Search_API.py собирается вся структура сайта и передается в файл table.html.

```
headers = ["Имя на русском", "Имя на английском", "Email", "Кол-во статей",
table = HtmlTable(headers)


h1 = "<h1>Статистика по данным авторов Санкт-Петербургского горного университета</h1>"
h2 = "<h2>Источник данных: <a href='https://scholar.google.ru/schhp?hl=ru'>Google Академия</a></h2>"
out = "<link rel='stylesheet' href='style.css'>" + h1 + h2 + table.get_table()
f = open('table.html', 'w')
f.write(out)
f.close()
```

Информация отображается в виде таблицы без рамок

table.html | table.html

Файл | D:/учеба/Практика%202022/Google%20Scholar%20API/Google%20Scholar%20API/table.html

Яндекс | Группы | МЕТЕОНОВА - пог... | Яндекс.Почта

Статистика по данным авторов Санкт-Петербургского горного университета

Источник данных: Google Академия

Имя на русском	Имя на английском	Email	Кол-во статей	Кол-во цитирований	Кол-во соавторов	Кол-во статей, где автор единственный	Кол-во статей, где автор первый в списке	Кол-во статей, где нет автора
Савенок Ольга Вадимовна	Savenok Olga Vadimovna	Verified email at pers.spmi.ru	Всего: 229 2022: 11 2021: 53 2020: 54 2019: 41 2018: 70	Всего: 1356 2022: 299 2021: 160 2020: 103 2019: 324 2018: 470	142	4	58	30
Сергей Цирель	Sergej Tsirel	Verified email at pers.spmi.ru	Всего: 30 2022: 0 2021: 1 2020: 5 2019: 16 2018: 8	Всего: 129 2022: 0 2021: 41 2020: 18 2019: 23 2018: 47	130	6	7	2
Прищепа Олег	Prischepa Oleg	Verified email at pers.spmi.ru	Всего: 42 2022: 2 2021: 10 2020: 11 2019: 11 2018: 8	Всего: 145 2022: 2 2021: 13 2020: 43 2019: 51 2018: 36	44	2	28	6
А С Татаренко	A S Tatarenko	Verified email at pers.spmi.ru	Всего: 15 2022: 0 2021: 1 2020: 3 2019: 11 2018: 0	Всего: 72 2022: 0 2021: 18 2020: 14 2019: 40 2018: 0	21	3	1	0

Чтобы информация была более разборчивой, подключается файл со стилями style.css:

```
body {
    padding: 10px 7px 0 7px
}

h1 {
    font-family: 'Verdana', sans-serif;
    color: #454c5c;
    font-size: 22px;
    line-height: 1.5em;
    margin-bottom: 0;
}

h2 {
    font-family: 'Verdana', sans-serif;
    color: #454c5c;
    font-size: 18px;
    line-height: 1.0em;
    margin-top: 5px;
    margin-bottom: 20px;
}

.table {
    table-layout: fixed;
    width: 100%;
    margin-bottom: 20px;
    border: 15px solid #B0E0E6;
    border-top: 5px solid #B0E0E6;
    border-collapse: collapse;
}

.table th {
    width: 5%;
    font-weight: bold;
    font-family: 'Verdana', sans-serif;
    padding: 5px;
    background: #B0E0E6;
    border-bottom: 5px solid #B0E0E6;
    text-align: center;
    border-left: 1px solid #C0C0C0;
      border-right: 1px solid #C0C0C0;
    color: #454c5c;
}

.table td {
    width: 2%;
    font-family: 'Verdana', sans-serif;
    padding: 10px;
    line-height: 1.5em;
    border-bottom: 5px solid #B0E0E6;
    border-left: 1px solid #C0C0C0;
      border-right: 1px solid #C0C0C0;
    color: #454c5c;
}

.table thead tr th:first-child, .table tbody tr td:first-child {
      border-left: none;
}

.table thead tr th:last-child, .table tbody tr td:last-child {
      border-right: none;
}
```

```
.table tbody tr td:nth-child(-n+5) {
    text-align:left;
}

.table tbody tr td:nth-child(n+6) {
    text-align: center;
      }
```

Конечный вывод таблицы с данными:

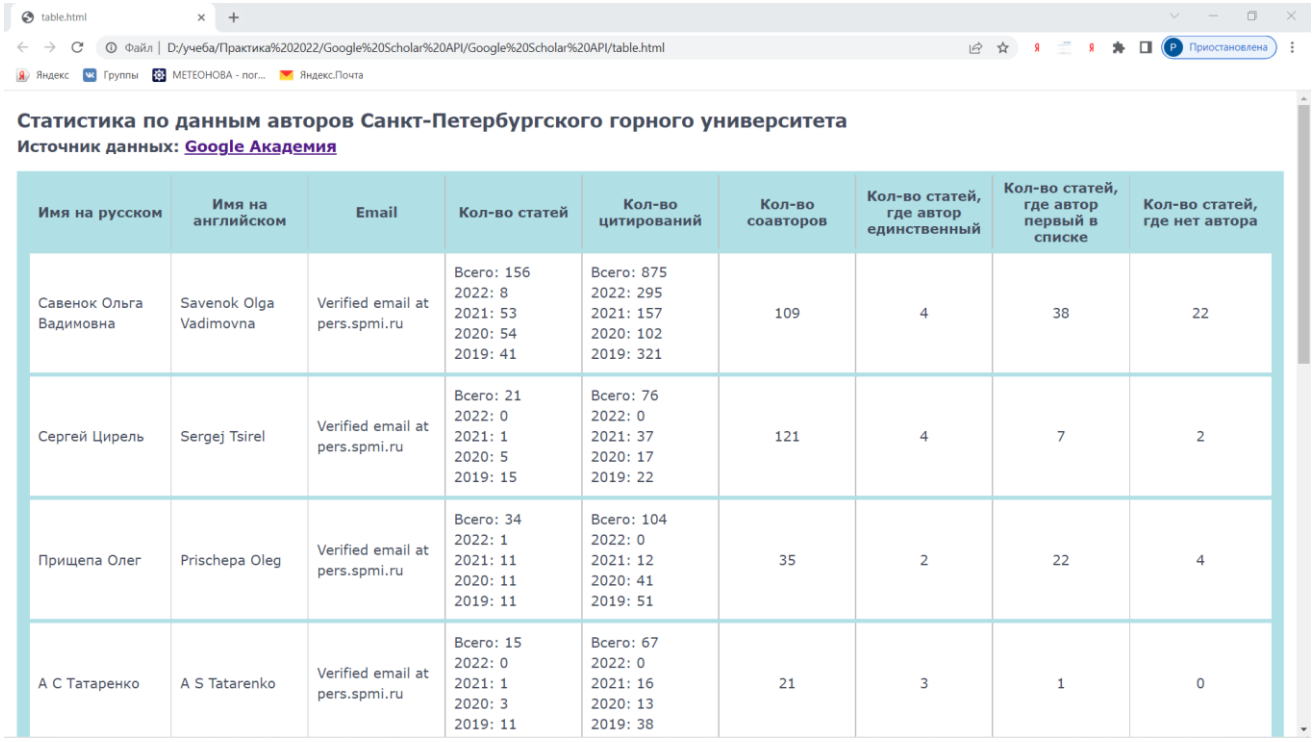


table.html

Файл | D:/учеба/Практика%202022/Google%20Scholar%20API/Google%20Scholar%20API/table.html

Яндекс | Группы | МЕТЕОНОВА - пог... | Яндекс.Почта

Приостановлена

Статистика по данным авторов Санкт-Петербургского горного университета

Источник данных: Google Академия

Имя на русском	Имя на английском	Email	Кол-во статей	Кол-во цитирований	Кол-во соавторов	Кол-во статей, где автор единственный	Кол-во статей, где автор первый в списке	Кол-во статей, где нет автора
Савенок Ольга Вадимовна	Savenok Olga Vadimovna	Verified email at pers.spmi.ru	Всего: 156 2022: 8 2021: 53 2020: 54 2019: 41	Всего: 875 2022: 295 2021: 157 2020: 102 2019: 321	109	4	38	22
Сергей Цирель	Sergej Tsirel	Verified email at pers.spmi.ru	Всего: 21 2022: 0 2021: 1 2020: 5 2019: 15	Всего: 76 2022: 0 2021: 37 2020: 17 2019: 22	121	4	7	2
Прищепа Олег	Prischepa Oleg	Verified email at pers.spmi.ru	Всего: 34 2022: 1 2021: 11 2020: 11 2019: 11	Всего: 104 2022: 0 2021: 12 2020: 41 2019: 51	35	2	22	4
А С Татаренко	A S Tatarenko	Verified email at pers.spmi.ru	Всего: 15 2022: 0 2021: 1 2020: 3 2019: 11	Всего: 67 2022: 0 2021: 16 2020: 13 2019: 38	21	3	1	0

На странице также имеется ссылка на поисковую строку Google Scholar.

Полный код файла Google_Scholar_API.py:

```
# -*- coding: cp1251 -*-
import re
from transliterate import translit
from datetime import date
from serpapi import GoogleSearch
from to_html import HtmlTable

API_KEY = "e392888bb057d773516bec7c1da319c5eed354f1edb583ce9e0737b1ca6bf2c2"

#кол-во авторов и лет
count_authors = 4
count_years = 5

token = ""
def id_author(token):
    params_all_authors = {
        "engine": "google_scholar_profiles",
        "mauthors": "Санкт-Петербургский горный университет",
        "api_key": API_KEY,
        "after_author": token
        }
    search = GoogleSearch(params_all_authors)
    result_all_authors = search.get_dict()
    return result_all_authors

def Next_Page(st, params):
```

```
    params["start"] = st*100
    search = GoogleSearch(params)
    result_page = search.get_dict()
    return result_page

def article(citation_id):
    params_articles = {
    "engine": "google_scholar_author",
    "view_op": "view_citation",
    "citation_id": citation_id,
    "api_key": API_KEY
        }
    search = GoogleSearch(params_articles)
    result_article = search.get_dict()
    return result_article

def info_author(author_id):
    params_author = {
        "engine": "google_scholar_author",
        "author_id": author_id,
        "api_key": API_KEY,
        "sort": "pubdate",
        "num": 100
        }
    search = GoogleSearch(params_author)
    result_author = search.get_dict()

    #ИМЯ
    name = result_author["author"]["name"]
    name_ru = ' '.join(re.findall(r"\b([a-яА-ЯёЁ]+)\b", name))
    name_en = ' '.join(re.findall(r"\b([a-zA-Z]+)\b", name))

    if len(name_ru) == 0:
        name_ru = translit(name_en, "ru")
    if len(name_en) == 0:
        name_en = translit(name_ru, language_code = "ru", reversed = True)
        name_en = re.sub(r'[^\w\s]','', name_en)

    all_name = name_ru.split(" ") + name_en.split(" ")
    #EMAIL
    email = result_author["author"]["email"]

    #начальные значения
    today = date.today()
    today_year = today.year

    articles = {}
    cit = {}
    articles["Всего"] = 0
    cit["Всего"] = 0
    for i in range(count_years):
        articles[int(today_year - i)] = 0
        cit[int(today_year - i)] = 0

    co_authors = 0
    author_1 = 0
    author_first = 0
    without_author = 0

    if int(result_author["articles"][0]["year"]) <= int(today_year - count_years):
        return(name_ru, name_en, email, articles, cit, co_authors, author_1,
author_first, without_author)
    #КОЛ-ВО СТАТЕЙ И ЦИТИРОВАНИЙ
    all_year = []
    start = 0
```

```
    y = today_year

    while True:
        for year in result_author["articles"]:
            if year["year"] != "":
                y = int(year["year"])
                if y > (today_year - count_years):
                    all_year.append(year) #вся инф за определенные года
                    articles[y] += 1      #кол-во статей
                    if year["cited_by"]["value"]:
                        cit[y] += int(year["cited_by"]["value"])   #кол-во
цитирований
        if y > (today_year - count_years):
            start += 1
            result_author = Next_Page(start, params_author)  #переход на след
страницу
        else:
            break

    for i in articles:
        articles["Всего"] += int(articles[i])
        cit["Всего"] += int(cit[i])

    #АВТОР И СОАВТОРЫ
    all_authors = []

    for author in all_year:
        if author["year"] != "":
            string = author["authors"]
            if ", ..." in string:
                citation_id = author["citation_id"]
                result_article = article(citation_id)
                all_authors.append(result_article["citation"]["authors"])
            else:
                all_authors.append(author["authors"])  #все авторы

    all_au = []
    all_sn = []
    all_surname = ""

    for string in all_authors:
        all_au.append(string.lower().title())

    for string in all_au:
        if "," in string:
            st = string.split(",")
            all_surname = ""
            for s in st:
                surname = ' '.join(s.split()[-1:])
                all_surname += surname + " "
            all_sn.append(all_surname)     #только фамилии
        else:
            surname = ' '.join(string.split()[-1:])
            all_sn.append(surname)
            for n in all_name:
                if n == surname:
                    author_1 += 1   #этот автор является единственным автором статьи

    str_surname = []
    author_f = 0
    au = 0

    for name in all_sn:
        n = name.split()
        for s in n:
```

```
            str_surname.append(s)
            if s in all_name:
                au += 1
                if n[0] == s:
                    author_f += 1

    if articles["Всего"] >= au:             #автора нет в списке авторов статьи
        without_author = articles["Всего"] - au

    author_first = author_f - author_1      #автор записан первый среди соавторов

    list_not_author = []
    ln_author = []

    for word in str_surname:
        count_name = 0
        count = 0
        for fragment in all_name:
            count_name += 1
            if fragment != word:
                count += 1
        if count == count_name:
            list_not_author.append(word)

    for word in list_not_author:
        word_en = translit(word, language_code='ru', reversed=True)
        if word_en not in ln_author:
            ln_author.append(word_en)
    co_authors = len(ln_author)    #кол-во соавторов

    return(name_ru, name_en, email, articles, cit, co_authors, author_1,
author_first, without_author)

#запись всех id
all_id = []

while len(all_id) < count_authors:
    result_all_authors = id_author(token)
    for a_id in result_all_authors["profiles"]:
        if len(all_id) < count_authors:
            all_id.append(a_id["author_id"])
    token = result_all_authors["pagination"]["next_page_token"]

#html
headers = ["Имя на русском", "Имя на английском", "Email", "Кол-во статей", "Кол-во
цитирований", "Кол-во соавторов", "Кол-во статей, где автор единственный", "Кол-во
статей, где автор первый в списке", "Кол-во статей, где нет автора"]
table = HtmlTable(headers)

for a_id in all_id:
    author_id = [a_id]
    info = info_author(author_id)
    table.to_table(info)

h1 = "<h1>Статистика по данным авторов Санкт-Петербургского горного
университета</h1>"
h2 = "<h2>Источник данных: <a href='https://scholar.google.ru/schhp?hl=ru'>Google
Академия</a></h2>"
out = "<link rel='stylesheet' href='style.css'>" + h1 + h2 + table.get_table()
f = open('table.html', 'w')
f.write(out)
f.close()
```

Выводы о результатах прохождения практики

В ходе работы с практическим заданием было сделано следующее:

1. Изучение и работа в Postman;
2. Применение знаний программирования на Python;
3. Получение навыков работы с API;
4. Вывод статистических данных на сайт.

Список использованных источников

1. Профиль Горного университета в базе данных Goolge Scholar: https://scholar.google.ru/citations?view_op=search_authors&hl=ru&mauthors=Санкт-Петербургский+горный+университет&before_author=tJ0V_9cEAAAJ&astart=0
2. Учебно-научный центр цифровых технологий Горного Университета: https://nc-digital.spmi.ru/
3. Google Scholar API - https://serpapi.com/google-scholar-api
4. Postman: https://www.postman.com/
5. Самоучитель Python: https://pythonworld.ru/samouchitel-python