

Спринт 2

Тема: Аналитические веб-сервисы

В рамках данного спринта ведется отработка следующих навыков:

- Подключение к произвольному внешнему веб-сервису по REST API / SOAP и парсинг ответа из него с целью последующего использования в обработке данных.
- Создание и публикация собственного аналитического веб-сервиса, в том числе с вызовом внутри ML-модели.

Теоретическая база

Понятие, типы и архитектура веб-сервисов. Веб-стандарты. Характеристики веб-сервисов. Преимущества и недостатки. Этапы развития веб-сервисов. Первые средства создания веб-сервисов. Современное распространение технологии. XML. Структура XML-документа. Правила задания имен. JSON. Структура JSON-документа. Правила задания ключей и значений. Сравнение XML и JSON. SOAP. Структура SOAP-сообщения. WSDL. Структурная схема WSDL. REST. REST-запросы. Описание REST-сервиса. Сравнение SOAP/WSDL и REST.

Программное обеспечение

1. Python, VS Code (или другая IDE) с плагинами:
 - a. GigaCode
 - b. Continue (опционально)
2. Облачная среда Google Colab, плагин Jupiter.
3. Ollama для запуска локальных LLM-моделей (опционально).

Все практические задачи выкладывать в свой проект на [gitlab/github/gitverse](#).

Важные замечания.

1. При выполнении всех работ допускается использовать AI-ассистента.
2. В комментариях к коду (блоки markdown) необходимо отразить, какой AI-ассистент использовался.

Практическая работа №2. Подключение к внешнему веб-сервису

Материалы для выполнения работы и задание расположено здесь:

https://github.com/amalrik1/develop_ml_course/tree/main/practice1

Порядок выполнения: подключаемся к внешнему веб-сервису из Python, вызываем методы, парсим json, при необходимости проводим постобработку.

ВАРИАНТЫ ЗАДАНИЙ

Задания для DaData API (требует бесплатной регистрации – до 10 000 запросов/день)

Документация: <https://dadata.ru/api/>

Регистрация: <https://dadata.ru/profile/>

1. Подсказки по адресам

Метод: https://suggestions.dadata.ru/suggestions/api/4_1/rs/suggest/address

Документация: <https://dadata.ru/api/suggest/address/>

Задание: по запросу «Москва Тверская» вывести первые 3 полных адреса.

2. Подсказки по ФИО

Метод: https://suggestions.dadata.ru/suggestions/api/4_1/rs/suggest/name

Документация метода: <https://dadata.ru/api/suggest/name/>

Задание: написать функцию, которая по введенному имени «Алекс» найдет все варианты полных ФИО и определит пол для каждого варианта.

3. Поиск организаций по названию

Метод: https://suggestions.dadata.ru/suggestions/api/4_1/rs/suggest/party

Документация метода: <https://dadata.ru/api/suggest/party/>

Задание: найти все организации со словом «Сбербанк» в названии. Для каждой найденной организации вывести: полное название, ИНН, адрес, статус (действующая/ликвидированная), ФИО руководителя.

4. Детальная информация об организации по ИНН

Сайт: https://suggestions.dadata.ru/suggestions/api/4_1/rs/findById/party

Документация метода: <https://dadata.ru/api/find-party/>

Задание: по ИНН «7707083893» получить максимально полную информацию об организации: уставный капитал, коды ОКВЭД, количество сотрудников, финансовые показатели, учредители. Создать структурированный отчет.

Задания для Яндекс Геокодер API (требуется бесплатной регистрации – 1000 запросов/день)

Документация: <https://yandex.ru/dev/geocode/doc/ru/>

Регистрация: <https://developer.tech.yandex.ru/>

Получение ключа: <https://developer.tech.yandex.ru/services/>

5. Прямое геокодирование адресов

Описание: Получение координат по адресу с учетом опечаток

Метод: <https://geocode->

maps.yandex.ru/1.x/?apikey=KEY&geocode=ADDRESS&format=json

Документация метода: <https://yandex.ru/dev/geocode/doc/ru/request>

Примеры запросов: <https://yandex.ru/dev/geocode/doc/ru/examples>

Задание: создать геокодер для адреса «Москва, Красная площадь, 1». Получить точные координаты, проверить обработку опечаток («Красная площадь»). Вывести широту, долготу и точность определения.

6. Обратное геокодирование координат

Описание: Определение адреса по географическим координатам

Метод: <https://geocode->

maps.yandex.ru/1.x/?apikey=KEY&geocode=LON,LAT&format=json

Документация метода: <https://yandex.ru/dev/geocode/doc/ru/request>

Параметры запроса: <https://yandex.ru/dev/geocode/doc/ru/parameters>

Задание: по координатам (37.617635, 55.755814) определить точный адрес. Получить информацию о ближайших объектах: улице, доме, районе, метро. Создать функцию для массовой обработки координат.

7. Поиск объектов по типу

Метод: <https://geocode->

maps.yandex.ru/1.x/?apikey=KEY&geocode=COORDS&kind=metro&format=json

Документация метода: <https://yandex.ru/dev/geocode/doc/ru/parameters>

Типы объектов: <https://yandex.ru/dev/geocode/doc/ru/kind>

Задание: найти ближайшие 3 станции метро к координатам центра Москвы (37.6175, 55.7558). Для каждой станции вывести название, расстояние от исходной точки, линию метро (если доступно).

8. Геокодирование с ограничением области поиска

Метод: <https://geocode->

maps.yandex.ru/1.x/?apikey=KEY&geocode=QUERY&ll=LON,LAT&spn=WIDHT,HEIGHT

Документация метода: <https://yandex.ru/dev/geocode/doc/ru/parameters>

Ограничение области: <https://yandex.ru/dev/geocode/doc/ru/ll-spn>

Задание: найти все объекты «Парк» в пределах МКАД (центр: 37.6175, 55.7558, область поиска: 0.5°×0.5°). Получить список парков с координатами и описанием каждого.

Задания для VK API (публичные методы без токена или с простой регистрацией приложения)

Документация: <https://dev.vk.com/ru>

Справочник методов: <https://dev.vk.com/ru/method>

Регистрация приложения: <https://vk.com/apps?act=manage>

Права доступа: <https://dev.vk.com/ru/reference/access-rights>

9. Получение информации о пользователе

Метод: https://api.vk.com/method/users.get?user_ids=USER_ID&fields=city,bdate,sex,photo_200&v=5.131

Документация метода: <https://dev.vk.com/ru/method/users.get>

Поля пользователя: <https://dev.vk.com/ru/reference/objects/user>

Задание: по ID любого пользователя (например, собственного ID) получить: имя, фамилию, город, дату рождения, пол, фото. Создать функцию для красивого вывода анкеты пользователя с обработкой приватных данных.

10. Анализ друзей пользователя (если профиль открыт)

Метод: https://api.vk.com/method/friends.get?user_id=USER_ID&fields=city,bdate,sex&v=5.131

Документация метода: <https://dev.vk.com/ru/method/friends.get>

Поля друзей: <https://dev.vk.com/ru/reference/objects/user>

Задание: получить список друзей пользователя и провести анализ: соотношение по полу, возрастные группы, топ-5 самых популярных городов среди друзей. Требуется открытый список друзей.

11. Информация о группах пользователя

Метод: https://api.vk.com/method/groups.get?user_id=USER_ID&extended=1&fields=members_count,activity&v=5.131

Документация метода: <https://dev.vk.com/ru/method/groups.get>

Поля группы: <https://dev.vk.com/ru/reference/objects/group>

Задание: Получить список групп пользователя (если доступен). Для каждой группы вывести: название, количество участников, тип активности, дату создания. Найти самые популярные группы по количеству участников.

12. Описание: Поиск пользователей ВКонтакте по имени и городу

Метод: <https://api.vk.com/method/users.search?q=Александр&city=1&count=10&fields=city,bdate,sex&v=5.131>

Документация метода: <https://dev.vk.com/ru/method/users.search>

Задание:

1. Выполнить запрос, чтобы получить до 10 пользователей с именем «Александр» из Москвы (city=1).
2. Для каждого пользователя вывести: имя, фамилию, пол и дату рождения.
3. Подсчитать, сколько из найденных пользователей младше 30 лет.

Практическая работа №3. Разработка собственного веб-сервиса

Материалы для выполнения работы и пример веб-сервиса на FLASK расположены здесь:

https://github.com/amalrik1/develop_ml_course/tree/main/practice2

Вам предстоит разработать, опубликовать и вызвать *свой* собственный веб-сервис с ML-моделью, используя только Python, и сделать это двумя способами:

1. На локальном компьютере с использованием библиотеки FLASK.
2. На Google Colab с применением библиотеки (localtunnel, ngrok и другие, см. список [здесь](#)).

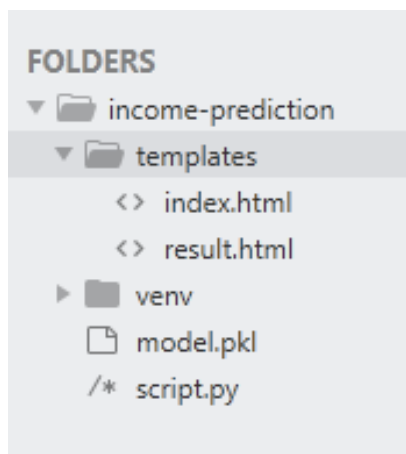
Внимание! Сервис ngrok не доступен из РФ.

Для понимания, как устроен веб-сервис на Python, нужно понимать следующие темы:

- Виртуальные окружения Python и работа с venv (пример [статьи](#))
- Проектирование RESTful API с помощью Python и Flask (пример [статьи 1](#), пример [статьи 2](#), пример [статьи 3](#))
- Сериализация и десериализация объектов в Python (пример [статьи](#)).

Создание и разворачивание веб-сервиса на локальной машине зависит от выбранного способа, от операционной системы и других нюансов. Ниже приведен рекомендуемый путь (для ОС Windows и Python 3.x, HE Anaconda), но вы можете идти своим путем.

Создайте на диске отдельную папку и поместите в нее файлы согласно структуре, указанной на рисунке.



Папка venv появится после установки виртуальной среды Python в эту папку. Сделать это проще всего так:

1. Установите библиотеку virtualenv для Python командой

```
pip install virtualenv
```

2. В командной строке перейдите в папку проекта и установите виртуальное окружение Python командой

```
virtualenv venv
```

Имя папки может быть любым.

3. Активируйте виртуальное окружение, запустив `activate.bat` из папки `Scripts`.
4. Установите все требуемые для работы сервиса библиотеки Python в виртуальное окружение.
5. Установите библиотеку `flask`.
6. Выполните команду `FLASK_APP = script.py` и запустите `flask`:

```
flask run
```

7. Откройте страницу <http://127.0.0.1:5000> – там отобразится веб-форма со страницы `index.html`.
8. Заполните форму и нажмите Submit. Сервис должен успешно отработать и выдать результат, открыв страницу `result.html`.

Если этого не происходит – ищите, в чем проблема (в консоли отображаются сообщения об ошибках). Причин может быть много – от неверных команд (для Linux они другие), до несовместимости версий библиотек.

Разрешается отправлять в веб-сервис готовый json, или использовать специальные программы для тестирования запросов к веб-сервисам, например, Postman.

Крайне желательно сделать авторизацию к вашему веб-сервису по логину и паролю.

Следующий этап – это перенос вашего веб-сервиса в среду выполнения Google Colab. Как это сделать, можно почитать [здесь](#) и [здесь](#). Остальное – на самостоятельную проработку и поиск источников.

Важно! Для выполнения работы у каждого студента должна быть своя ML-модель. Пока что можно взять любой учебный датасет: Ирисы Фишера, Титаник и другие с ресурса [Kaggle/datasets](https://www.kaggle.com/datasets), или вашу модель с практик на 1 курсе.