

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение высшего
образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

Выполнила: st_95

Вебинар от 28.03.2025

Бизнес-кейс «Rocket»

Проектный практикум по разработке ETL-решений

Направление подготовки

38.03.05 Бизнес-информатика

Профиль подготовки

Аналитика данных и эффективное управление

Москва

2025

Вариант 8

Задание 1	Задание 2	Задание 3
Проанализировать структуру данных о запуске ракет для возможных улучшений	Создать DAG для скачивания данных по старым ракетам	Оценить возможные ошибки в структуре данных JSON и предложить их исправление

Постановка задачи:

Создать автоматизированный процесс сбора и обработки данных о предстоящих запусках ракет, а также прошедших запусков ракет с загрузкой связанных изображений.

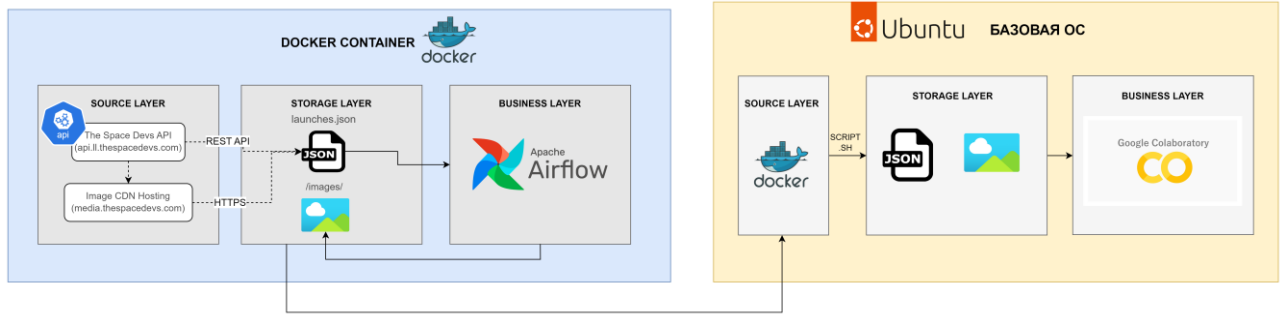
Функциональные требования:

- Ежедневное получение актуальных данных о предстоящих запусках ракет через API The Space Devs (v2.0.0) (а также о прошедших запусках по индивидуальному заданию)
 - Сохранение данных в формате JSON
 - Загрузка и сохранение изображений ракет
 - Уведомление о количестве загруженных изображений
 - Обеспечение отказоустойчивости при обработке данных

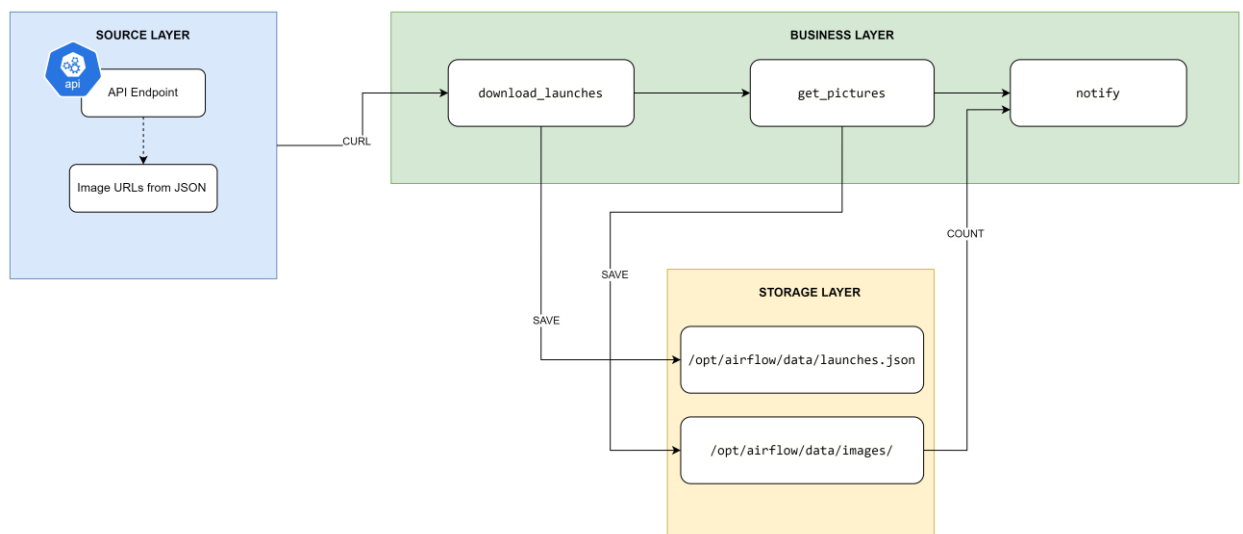
Технические требования:

- Использование Apache Airflow для оркестрации
- Хранение данных в директории /opt/airflow/data/
- Логирование процесса выполнения
- Обработка ошибок при загрузке изображений

Верхнеуровневая архитектура бизнес-кейса:



Архитектура DUG



Ход работы:

1. После того, как были запущены контейнеры, запускаем Dug (Рисунок 1). Все этапы были успешно выполнены

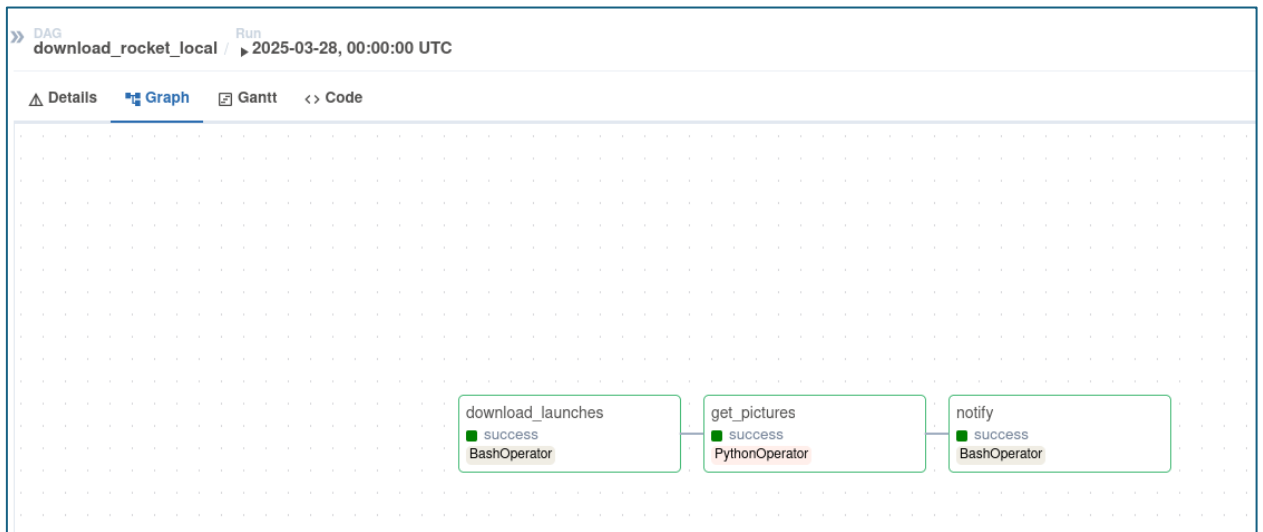


Рисунок 1

2. Изображения были успешно получены (Рисунок 2).

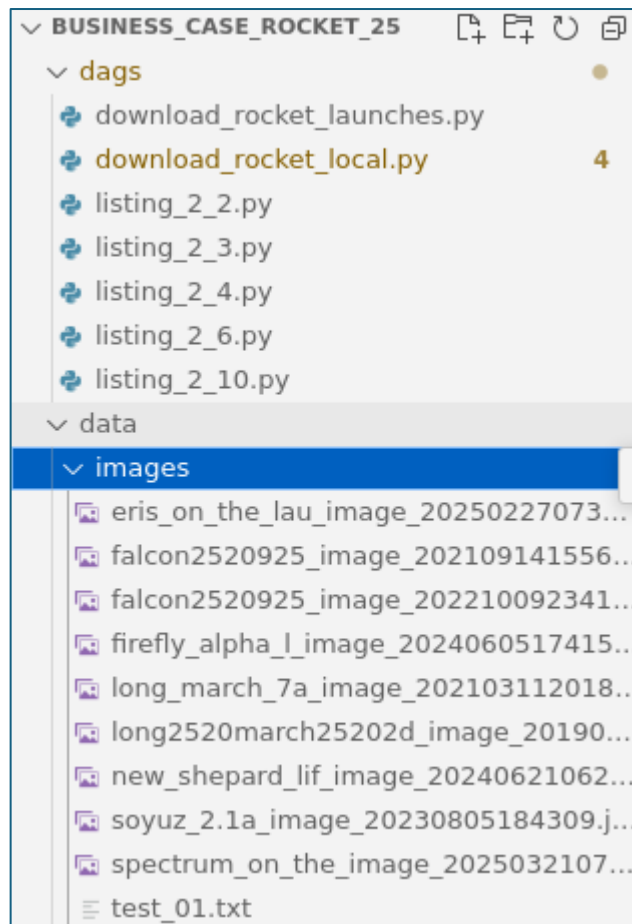


Рисунок 2

Общее задание. Создать исполняемый файл с расширением .sh, который автоматизирует выгрузку данных из контейнера в основную ОС данных, полученные в результате работы DAG в Apache Airflow:

1. Листинг кода для автоматизации выгрузки данных из контейнера в основную ОС (Рисунок 3).

```
download_rocket_local.py 4  export_airflow_data.sh x
$ export_airflow_data.sh
1  #!/bin/bash
2
3  # Скрипт для экспорта данных из контейнера Airflow в /home/mgpu/airflow_export
4
5  # Параметры
6  CONTAINER_NAME="business_case_rocket_25-scheduler-1"
7  AIRFLOW_DATA_DIR="/opt/airflow/data"
8  HOST_EXPORT_DIR="/home/mgpu/airflow_export"
9  TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
10 EXPORT_PATH="$HOST_EXPORT_DIR/$TIMESTAMP"
11
12 # Проверяем существование директории /home/mgpu
13 if [ ! -d "/home/mgpu" ]; then
14     echo "Ошибка: Директория /home/mgpu не существует!"
15     exit 1
16 fi
17
18 # Создаем директорию для экспорта
19 mkdir -p "$EXPORT_PATH/images"
20
21 # Устанавливаем правильные права
22 chown -R mgpu:mgpu "$HOST_EXPORT_DIR"
23 chmod -R 755 "$HOST_EXPORT_DIR"
24
25 # 1. Копируем файл launches.json
26 echo "Копируем launches.json..."
27 docker cp "$CONTAINER_NAME:$AIRFLOW_DATA_DIR/launches.json" "$EXPORT_PATH/" && \
28     chown mgpu:mgpu "$EXPORT_PATH/launches.json" || \
29     echo "Не удалось скопировать launches.json"
30
31 # 2. Копируем изображения
32 echo "Копируем изображения..."
33 if docker exec "$CONTAINER_NAME" test -d "$AIRFLOW_DATA_DIR/images"; then
34     docker cp "$CONTAINER_NAME:$AIRFLOW_DATA_DIR/images/" "$EXPORT_PATH/images/" && \
35         chown -R mgpu:mgpu "$EXPORT_PATH/images" || \
36         echo "Не удалось скопировать изображения"
37 else
38     echo "Директория с изображениями не найдена в контейнере"
39 fi
40
41 # 3. Создаем файл с информацией
42 echo "Создаем README..."
43 echo "Данные экспортированы из контейнера $CONTAINER_NAME" > "$EXPORT_PATH/README.txt"
44 echo "Дата экспорта: $(date)" >> "$EXPORT_PATH/README.txt"
45 echo "Количество изображений: $(ls -l "$EXPORT_PATH/images/" 2>/dev/null | wc -l)" >> "$EXPORT_PATH/README.txt"
46 chown mgpu:mgpu "$EXPORT_PATH/README.txt"
47
48 echo "Данные успешно экспортированы в $EXPORT_PATH"
49 echo "Владелец файлов: $(stat -c '%U:%G' "$EXPORT_PATH")"
```

Рисунок 3

2. Делаем файл исполняемым (Рисунок 4)

```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
mgpu@mgpu-VirtualBox:~/Documents/workshop-on-ETL/business_case_rocket_25$ chmod +x export_airflow_data.sh
mgpu@mgpu-VirtualBox:~/Documents/workshop-on-ETL/business_case_rocket_25$
```

Рисунок 4

3. Запускаем скрипт (Рисунок 5).

```
mgpu@mgpu-VirtualBox:~/Documents/workshop-on-ETL/business_case_rocket_25$ sudo ./export_airflow_data.sh
[sudo] password for mgpu:
Копируем launches.json...
Successfully copied 27.6kB to /home/mgpu/airflow_export/20250328_103551/
Копируем изображения...
Successfully copied 1.64MB to /home/mgpu/airflow_export/20250328_103551/images/
Создаем README...
Данные успешно экспортированы в /home/mgpu/airflow_export/20250328_103551
Владелец файлов: mgpu:mgpu
```

Рисунок 5

4. Проверяем, что файлы были выгружены в директорию (Рисунок 6).

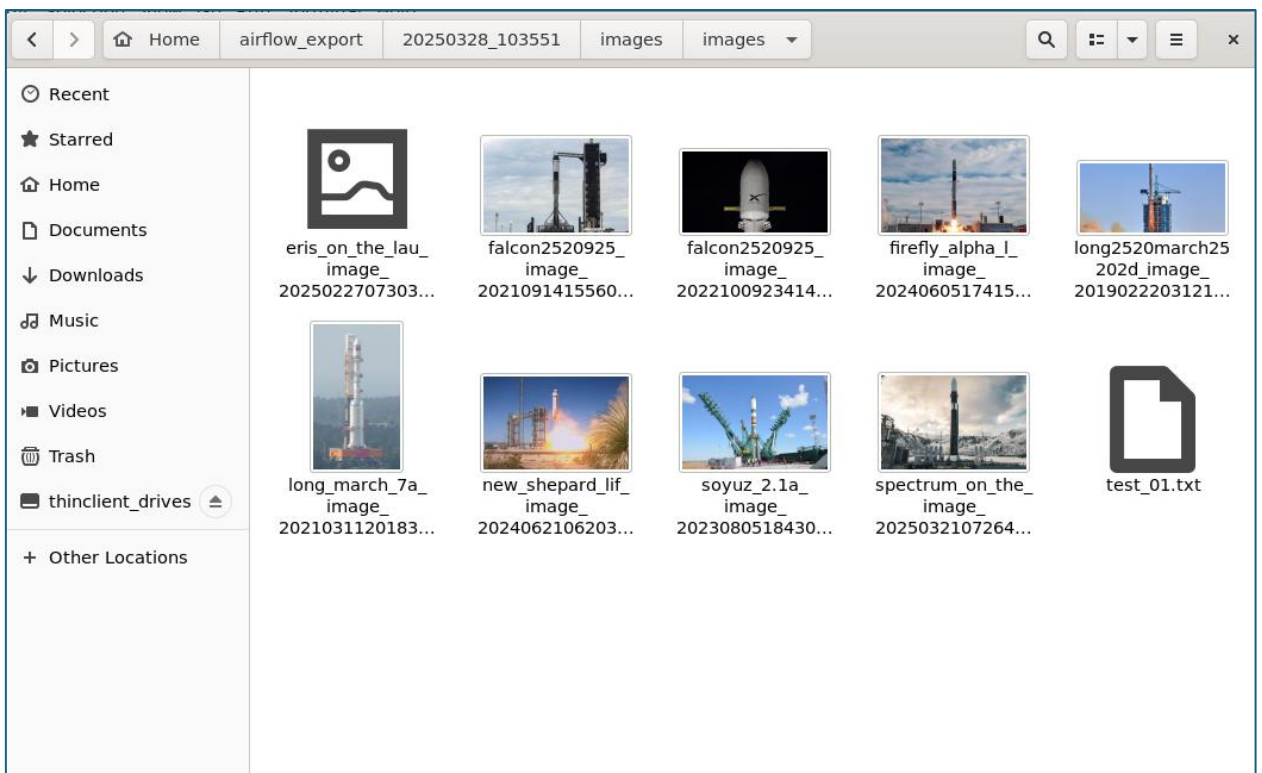


Рисунок 6

Диаграмма Ганта работы DAG в Apache Airflow (Рисунок 7).



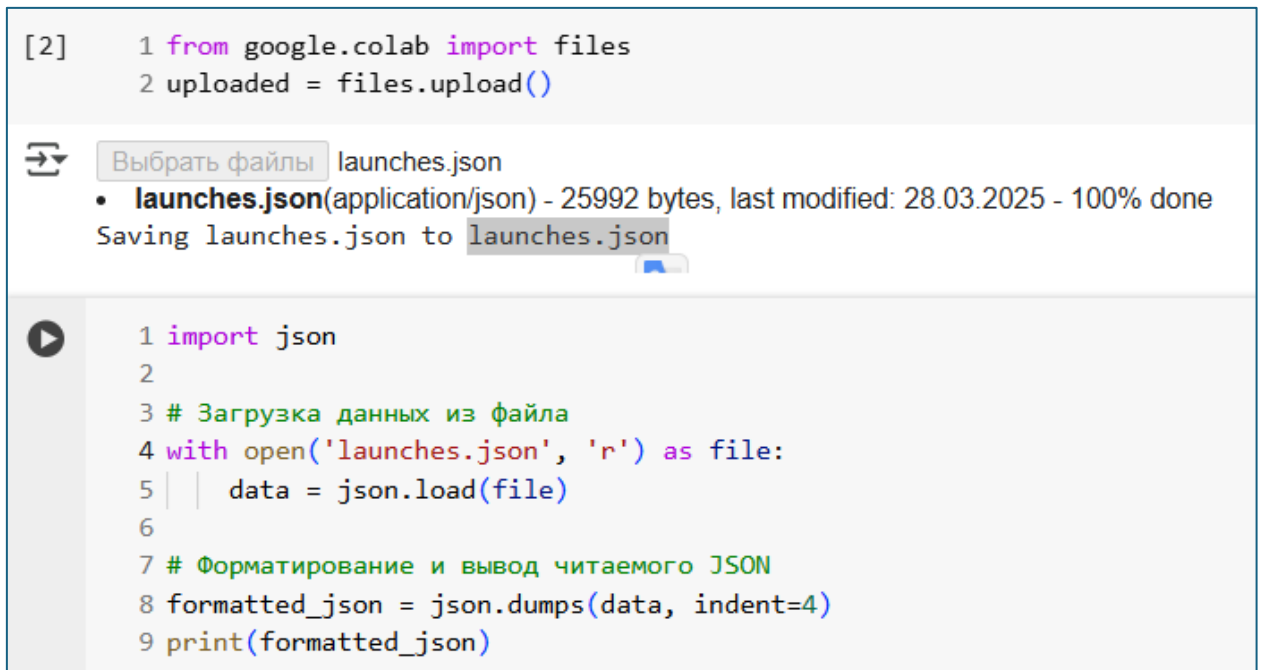
Рисунок 7

Верхнеуровневая архитектура аналитического решения задания Бизнес-кейса «Rocket» в draw.io

Выполнение индивидуального задания:

1. Проанализировать структуру данных о запуске ракет для возможных улучшений

Для анализа структуры данных перейдём в Google Colab импортируем файл, полученных с помощью DAG, и отобразим данные в читаемом виде (Рисунок 8).



The screenshot shows a Google Colab notebook. At the top, there is a code cell with the following Python code:

```
[2] 1 from google.colab import files
     2 uploaded = files.upload()
```

Below the code cell, a file upload dialog is visible. It shows a button labeled "Выбрать файлы" (Select files) and a list of files. The file "launches.json" is selected, and its details are shown: "launches.json(application/json) - 25992 bytes, last modified: 28.03.2025 - 100% done". Below this, it says "Saving launches.json to launches.json".

Below the upload dialog, there is a code cell with a play button icon on the left. The code in this cell is:

```
1 import json
2
3 # Загрузка данных из файла
4 with open('launches.json', 'r') as file:
5     data = json.load(file)
6
7 # Форматирование и вывод читаемого JSON
8 formatted_json = json.dumps(data, indent=4)
9 print(formatted_json)
```

Рисунок 8

Результат (Рисунок 9).

```
{
  "count": 331,
  "next": "https://11.thespacedevs.com/2.0.0/launch/upcoming/?limit=10&offset=10",
  "previous": null,
  "results": [
    {
      "id": "e652a538-6d40-4b55-97a6-7c757ec4e1e9",
      "url": "https://11.thespacedevs.com/2.0.0/launch/e652a538-6d40-4b55-97a6-7c757ec4e1e9/",
      "launch_library_id": null,
      "slug": "spectrum-maiden-flight",
      "name": "Spectrum | Maiden Flight",
      "status": {
        "id": 2,
        "name": "TBD"
      },
      "net": "2025-03-29T11:30:00Z",
      "window_end": "2025-03-29T14:30:00Z",
      "window_start": "2025-03-29T11:30:00Z",
      "inhold": false,
      "tbddtime": false,
      "tbddate": false,
      "probability": null,
      "holdreason": "",
      "failreason": "",
      "hashtag": null,
      "launch_service_provider": {
        "id": 1046,
        "url": "https://11.thespacedevs.com/2.0.0/agencies/1046/",
        "name": "Isar Aerospace",
        "type": "Private"
      }
    }
  ]
}
```

Рисунок 9

Общая структура:

- count: общее количество записей.
- next и previous: ссылки для пагинации.
- results: массив объектов, каждый из которых представляет отдельный запуск.

Рассмотрим данные по одному запуску на листинге 1.

Листинг 1 - Структура данных по одному запуску

```
{
  "id": "e652a538-6d40-4b55-97a6-7c757ec4e1e9",
  "url": "https://11.thespacedevs.com/2.0.0/launch/e652a538-6d40-4b55-97a6-7c757ec4e1e9/",
  "launch_library_id": null,
  "slug": "spectrum-maiden-flight",
  "name": "Spectrum | Maiden Flight",
  "status": {
    "id": 2,
    "name": "TBD"
  },
  "net": "2025-03-29T11:30:00Z",
  "window_end": "2025-03-29T14:30:00Z",
  "window_start": "2025-03-29T11:30:00Z",
  "inhold": false,
  "tbddtime": false,
  "tbddate": false,
  "probability": null,
  "holdreason": "",
  "failreason": "",
  "hashtag": null,
  "launch_service_provider": {
    "id": 1046,
    "url": "https://11.thespacedevs.com/2.0.0/agencies/1046/",
    "name": "Isar Aerospace",
    "type": "Private"
  }
}
```



```

    },
    "rocket": {
      "id": 8206,
      "configuration": {
        "id": 491,
        "launch_library_id": null,
        "url": "https://11.thespacedevs.com/2.0.0/config/launcher/491/",
        "name": "Spectrum",
        "family": "",
        "full_name": "Spectrum",
        "variant": ""
      }
    },
    "mission": {
      "id": 6778,
      "launch_library_id": null,
      "name": "Maiden Flight",
      "description": "First flight of the Isar Spectrum launch
vehicle.",
      "launch_designator": null,
      "type": "Test Flight",
      "orbit": {
        "id": 17,
        "name": "Sun-Synchronous Orbit",
        "abbrev": "SSO"
      }
    },
    "pad": {
      "id": 51,
      "url": "https://11.thespacedevs.com/2.0.0/pad/51/",
      "agency_id": null,
      "name": "Orbital Launch Pad",
      "info_url": null,
      "wiki_url": "https://en.wikipedia.org/wiki/And%C3%B8ya Space",
      "map_url": "https://www.google.com/maps?q=69.1084,15.5895",
      "latitude": "69.1084",
      "longitude": "15.5895",
      "location": {
        "id": 161,
        "url": "https://11.thespacedevs.com/2.0.0/location/161/",
        "name": "And\u00f8ya Spaceport",
        "country_code": "NOR",
        "map_image": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/map\_images/location\_161\_20240109072235.jpg",
        "total_launch_count": 0,
        "total_landing_count": 0
      },
      "map_image": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/map\_images/pad\_51\_20200803143605.jpg",
      "total_launch_count": 0
    },
    "webcast_live": false,
    "image": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/spectrum on the image\_20250321072643.jpeg",

```

```
"infographic": null,  
"program": []  
}
```

Идентификация:

- id: уникальный идентификатор запуска.
- slug: краткое название для удобства ссылок.

Основная информация:

- name: название миссии.
- status: статус запуска (например, "Go", "TBD").
- net, window_start, window_end: временные параметры запуска.

Ракета:

- Информация о ракете (rocket.configuration), включая название, семейство, полное имя и вариант.

Миссия:

- Тип миссии (type), описание и целевая орбита (orbit).
- Площадка запуска:
- Местоположение (pad.location), координаты, количество запусков с площадки.

Изображения и ссылки:

- Ссылки на изображения, карты и дополнительные ресурсы.

Проблемы структуры данных:

- 1) Данные были проверены на нулевые значения (Рисунок 9, Рисунок 10).

```
1 # Нормализация вложенных структур
2 df = pd.json_normalize(
3     data['results'],
4     meta=[
5         'id', 'name', 'slug',
6         ['status', 'name'],
7         'net', 'probability', 'hashtag'
8     ],
9     sep='_'
10 )
11
12 # Покажем структуру данных
13 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 60 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                    10 non-null     object
1   url                                  10 non-null     object
2   launch_library_id                    0 non-null      object
3   slug                                 10 non-null     object
4   name                                 10 non-null     object
5   net                                  10 non-null     object
6   window_end                           10 non-null     object
7   window_start                         10 non-null     object
8   inhold                               10 non-null     bool
9   tbdtime                              10 non-null     bool
```

```
1 # Функция для анализа пропусков
2 def analyze_nulls(df):
3     # Общее количество нулевых значений
4     total_nulls = df.isnull().sum().sum()
5
6     # Процент нулевых значений по колонкам
7     null_percent = df.isnull().mean() * 100
8
9     # Создаем DataFrame с результатами
10    null_analysis = pd.DataFrame({
11        'column': df.columns,
12        'null_count': df.isnull().sum(),
13        'null_percent': null_percent,
14        'dtype': df.dtypes
15    })
16
17    # Сортируем по убыванию процента пропусков
18    return null_analysis.sort_values('null_percent', ascending=False)
19
20 # Применяем анализ
21 null_report = analyze_nulls(df)
22 display(null_report)
```

Рисунок 10

Результат анализа (Рисунок 11).

	column	null_count	null_percent	dtype
launch_library_id	launch_library_id	10	100.0	object
probability	probability	10	100.0	object
infographic	infographic	10	100.0	object
hashtag	hashtag	10	100.0	object
mission_launch_designator	mission_launch_designator	10	100.0	object
mission_launch_library_id	mission_launch_library_id	10	100.0	object
pad_info_url	pad_info_url	9	90.0	object
pad_agency_id	pad_agency_id	6	60.0	float64
rocket_configuration_launch_library_id	rocket_configuration_launch_library_id	2	20.0	float64
pad_map_url	pad_map_url	1	10.0	object
window_end	window_end	0	0.0	object
slug	slug	0	0.0	object
tbddate	tbddate	0	0.0	bool
tbdtime	tbdtime	0	0.0	bool

Рисунок 11

Таким образом, выявлена **первая проблема** - Поля launch_library_id , Probability, Infographic, Hashtag, mission_launch_designator, mission_launch_library_id часто имеют значение null, что делает их бесполезными в текущем виде.

Решение:

Удалить поля с постоянным значением null, такие как launch_library_id.

2) Дублирование информации в объектах pad.location и pad (например, map_image присутствует на обоих уровнях).

Решение:

Перенести общие данные (например, изображения карты) на уровень объекта location.

3) Сложность временных данных:

Временные параметры (net, window_start, window_end) представлены отдельно, что может затруднить их обработку.

Решение:

Объединить параметры времени в один объект:

```
json
"time_window": {
  "net": "2025-03-29T11:30:00Z",
  "start": "2025-03-29T11:30:00Z",
  "end": "2025-03-29T14:30:00Z"
}
```

4) Отсутствие связи между объектами:

Нет четкой связи между программами (program) и миссиями, что затрудняет группировку данных по проектам.

Решение:

Добавить поле для связи между программами и миссиями:

```
json
"programs": [
  {
    "id": 25,
    "related_missions": ["7147", "7148"]
  }
]
```

2. Создать DAG для скачивания данных по старым ракетам

1) Листинг кода для нового DAGa, который парсит данные о старых запусках ракет (Рисунок 12).

```
download_past_rocket_local.py 4 x {} past_launches.json $ export_airflow_data.sh
dags > download_past_rocket_local.py > ...
1 import json
2 import pathlib
3
4 import airflow.utils.dates
5 import requests
6 import requests.exceptions as requests_exceptions
7 from airflow import DAG
8 from airflow.operators.bash import BashOperator
9 from airflow.operators.python import PythonOperator
10
11 dag = DAG(
12     dag_id="download_past_rocket_local",
13     description="Download rocket pictures of past launched rockets.",
14     start_date=airflow.utils.dates.days_ago(14),
15     schedule_interval="@daily",
16     catchup=False, # Добавляем чтобы не запускать старые пропущенные задачи
17 )
18
19 # Изменение пути для скачивания JSON-файла в папку data с прошлыми запусками
20 download_past_launches = BashOperator(
21     task_id="download_past_launches",
22     bash_command="curl -o /opt/airflow/data/past_launches.json -L 'https://ll.thespacedevs.com/2.0.0/launch/previous'", # noqa: E501
23     dag=dag,
24 )
25
26 def _get_past_pictures():
27     # Обеспечиваем существование директории для изображений в папке data
28     images_dir = "/opt/airflow/data/past_images"
29     pathlib.Path(images_dir).mkdir(parents=True, exist_ok=True)
30
31     # Загружаем все картинки из past_launches.json
32     with open("/opt/airflow/data/past_launches.json") as f:
33         launches = json.load(f)
34         image_urls = [launch["image"] for launch in launches["results"]]
35         for image_url in image_urls:
```

```
download_past_rocket_local.py 4 x {} past_launches.json $ export_airflow_data.sh
dags > download_past_rocket_local.py > ...
26 def _get_past_pictures():
27     images_dir = "/opt/airflow/data/past_images"
28     pathlib.Path(images_dir).mkdir(parents=True, exist_ok=True)
29
30     # Загружаем все картинки из past_launches.json
31     with open("/opt/airflow/data/past_launches.json") as f:
32         launches = json.load(f)
33         image_urls = [launch["image"] for launch in launches["results"]]
34         for image_url in image_urls:
35             try:
36                 response = requests.get(image_url)
37                 image_filename = image_url.split("/")[-1]
38                 target_file = f"{images_dir}/{image_filename}"
39                 with open(target_file, "wb") as f:
40                     f.write(response.content)
41                 print(f"Downloaded {image_url} to {target_file}")
42             except requests_exceptions.MissingSchema:
43                 print(f"{image_url} appears to be an invalid URL.")
44             except requests_exceptions.ConnectionError:
45                 print(f"Could not connect to {image_url}.")
46
47     get_past_pictures = PythonOperator(
48         task_id="get_past_pictures", python_callable=_get_past_pictures, dag=dag
49     )
50
51 # Обновляем команду уведомления, чтобы она считала количество изображений в папке data/past_images
52 notify_past = BashOperator(
53     task_id="notify_past",
54     bash_command='echo "There are now $(ls /opt/airflow/data/past_images/ | wc -l) past images."',
55     dag=dag,
56 )
57
58 download_past_launches >> get_past_pictures >> notify_past
```

Рисунок 12

Основные изменения:

- task_id изменен на "download_past_launches".

- `bash_command` изменена ссылка на API, чтобы получать данные о прошлых запусках: `"curl -o /opt/airflow/data/past_launches.json -L 'https://1l.thespacedevs.com/2.0.0/launch/previous'"`.
- `catchup` добавлен в DAG для избежания запуска пропущенных задач из прошлого
- `task_id` изменен на `"get_past_pictures"`.
- `python_callable` изменена на `_get_past_pictures`.
- Внутри функции `_get_past_pictures`:
- `images_dir` изменена на `"/opt/airflow/data/past_images"` для хранения изображений прошлых запусков.
- Путь к файлу с данными о запусках изменен на `"/opt/airflow/data/past_launches.json"`.
- `task_id` изменен на `"notify_past"`.
- `bash_command` изменена для подсчета файлов в директории `"/opt/airflow/data/past_images/"`.
- `dag`: Изменены зависимости задач, чтобы соответствовать новому пайплайну: `download_past_launches >> get_past_pictures >> notify_past`.

2) После пересборки и запуска контейнера запустим DAG (Рисунок 13), он был успешно выполнен.

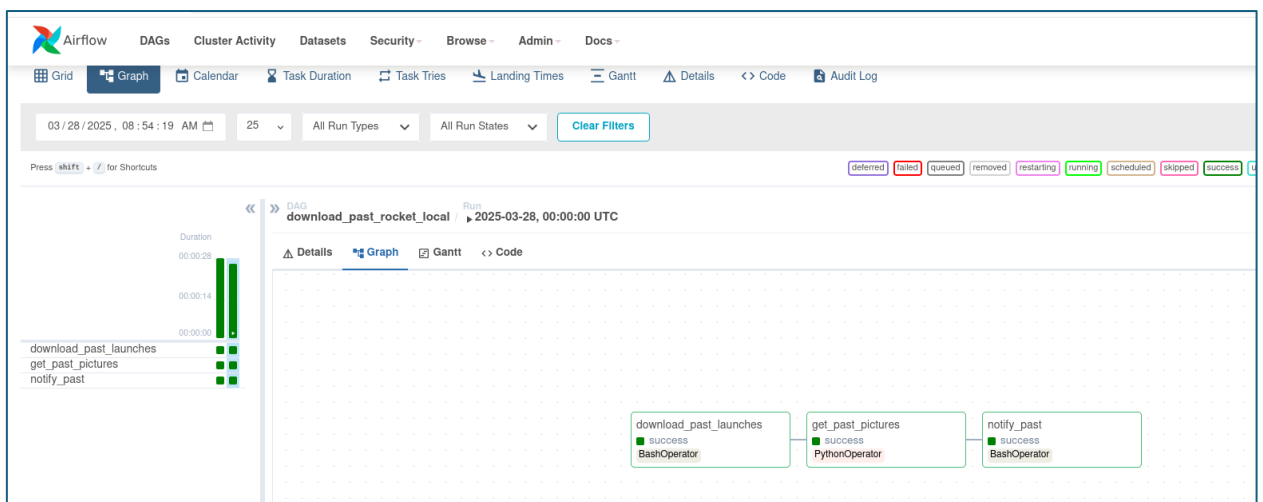


Рисунок 13

3) Диаграмма Ганта DAG для парсинга старых запусков (Рисунок 14).

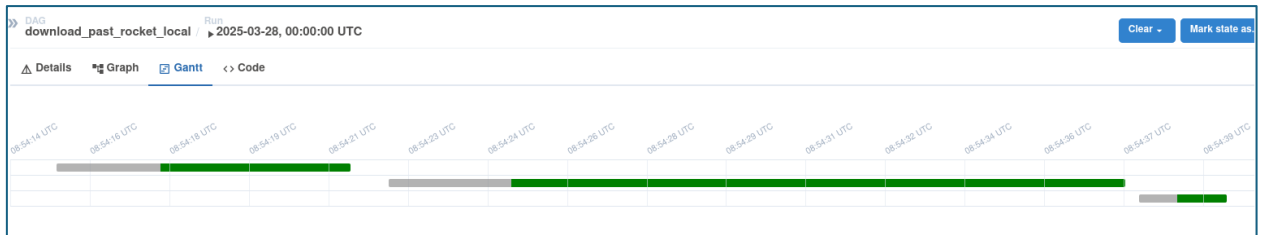


Рисунок 14

4) Проверка полученных данных в директории (Рисунок 15).

```
download_past_rocket_local.py 4 {} past_launches.json x $ export_airflow_data.sh
data > {} past_launches.json > ...
1 [{"count":7128,"next":"https://ll.thespacedevs.com/2.0.0/launch/previous/?limit=10&offset=10","previous":null,"results":
[{"id":"8bfaa532-18e6-4c40-9800-0714b5651f84","url":"https://ll.thespacedevs.com/2.0.0/launch/8bfaa532-18e6-4c40-9800-0714b5651f84","launch_library_id":null,
"slug":"falcon-9-block-5-starlink-group-11-7","name":"Falcon 9 Block 5 | Starlink Group 11-7","status":{"id":3,"name":"Success"},"net":{"id":3,"name":"Success"},
"window_end":"2025-03-27T01:56:10Z","window_start":"2025-03-26T22:11:40Z","inhold":false,"tbddtime":false,"tbddate":false,"probability":null,"holdreason":"","
"failreason":"","hashtag":null,"launch_service_provider":{"id":121,"url":"https://ll.thespacedevs.com/2.0.0/agencies/121/","name":"SpaceX","type":"Commercial"},
"rocket":{"id":8542,"configuration":{"id":164,"launch_library_id":188,"url":"https://ll.thespacedevs.com/2.0.0/config/launcher/164/","name":"Falcon 9",
"family":"Falcon","full name":"Falcon 9 Block 5","variant":"Block 5"},"mission":{"id":7133,"launch_library_id":null,"name":"Starlink Group 11-7",
"description":"A batch of 27 satellites for the Starlink mega-constellation - SpaceX's project for space-based Internet communication system.",
"launch_designator":null,"type":"Communications","orbit":{"id":8,"name":"Low Earth Orbit"},"abbrev":"LEO"},"pad":{"id":16,"url":"https://ll.thespacedevs.com/2.
0.0/pad/16/","agency_id":null,"name":"Space Launch Complex 4E","info_url":null,"wiki_url":"https://en.wikipedia.org/wiki/
Vandenberg_Space_Launch_Complex_4#SLC-4E","map_url":"https://www.google.com/maps?q=34.632,-120.611","latitude":"34.632","longitude":"-120.611","location":
{"id":11,"url":"https://ll.thespacedevs.com/2.0.0/location/11/","name":"Vandenberg SFB, CA, USA","country_code":"USA","map_image":"https://thespacedevs-prod.
nyc3.digitaloceanspaces.com/media/map_images/location_11_20200803142416.jpg","total_launch_count":797,"total_landing_count":26,"map_image":"https://
thespacedevs-prod.nyc3.digitaloceanspaces.com/media/map_images/pad_16_20200803143532.jpg","total_launch_count":185,"webcast_live":false,"image":"https://
thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png","infographic":null,"program":{"id":25,"url":"https://ll.
thespacedevs.com/2.0.0/program/25/","name":"Starlink","description":"Starlink is a satellite internet constellation operated by American aerospace company
SpaceX"},"agencies":[{"id":121,"url":"https://ll.thespacedevs.com/2.0.0/agencies/121/","name":"SpaceX","type":"Commercial"},"image_url":"https://
thespacedevs.com/2.0.0/program/25/","name":"Starlink","description":"Starlink is a satellite internet constellation operated by American aerospace company
SpaceX"},"agencies":[{"id":121,"url":"https://ll.thespacedevs.com/2.0.0/agencies/121/","name":"SpaceX","type":"Commercial"},"image_url":"https://
thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/starlink_program_20231228154508.jpeg","start_date":"2018-02-22T14:17:00Z","end_date":null,
"info_url":"https://starlink.com","wiki_url":"https://en.wikipedia.org/wiki/Starlink"}]},{"id":"60ad9119-12a0-4c07-98eb-2302b7c61010","url":"https://ll.
thespacedevs.com/2.0.0/launch/60ad9119-12a0-4c07-98eb-2302b7c61010/","launch_library_id":null,"slug":"long-march-3be-tianlian-2-04","name":"Long March 3B/E |
Tianlian 2-04","status":{"id":3,"name":"Success"},"net":{"id":3,"name":"Success"},"window_end":"2025-03-26T16:20:00Z","window_start":"2025-03-26T15:45:00Z",
"inhold":false,"tbddtime":false,"tbddate":false,"probability":null,"holdreason":"","failreason":"","hashtag":null,"launch_service_provider":{"id":88,
"url":"https://ll.thespacedevs.com/2.0.0/agencies/88/","name":"China Aerospace Science and Technology Corporation","type":"Government"},"rocket":{"id":8561,
"configuration":{"id":50,"launch_library_id":69,"url":"https://ll.thespacedevs.com/2.0.0/config/launcher/50/","name":"Long March 3","family":"Long March",
"full name":"Long March 3B/E","variant":"B/E"},"mission":{"id":7152,"launch_library_id":null,"name":"Tianlian 2-04","description":"Tianlian is a Chinese data
tracking and relay communications geostationary satellite series. The TL 2 (Tian Lian 2) satellites represent the second generation of this relay satellite
network, and is based on the DFH-4 Bus, a three-axis-stabilized telecommunications satellite platform.\r\n\r\nTL 2 will be used to support real-time
communications between orbiting satellites and ground control stations. This system will replace the current network of ground-based space tracking and
telemetry stations and space tracking ships.", "launch_designator":null,"type":"Communications","orbit":{"id":2,"name":"Geostationary Transfer Orbit",
"abbrev":"GTO"},"pad":{"id":45,"url":"https://ll.thespacedevs.com/2.0.0/pad/45/","agency_id":null,"name":"Launch Complex 2 (LC-2)","info_url":null,
"wiki_url":"https://en.wikipedia.org/wiki/Xichang_Satellite_Launch_Center","map_url":"https://www.google.com/maps?q=28.245564,102.026751","latitude":"28.
245564","longitude":"102.026751","location":{"id":16,"url":"https://ll.thespacedevs.com/2.0.0/location/16/","name":"Xichang Satellite Launch Center, People's
Republic of China","country_code":"CHN","map_image":"https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/map_images/location_16_20200803142513.jpg",
"total_launch_count":222,"total_landing_count":0,"map_image":"https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/map_images/pad_45_20200803143520.
jpg","total_launch_count":117,"webcast_live":false,"image":"https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/
```

Рисунок 15

Проверка выгрузки изображений (Рисунок 16).

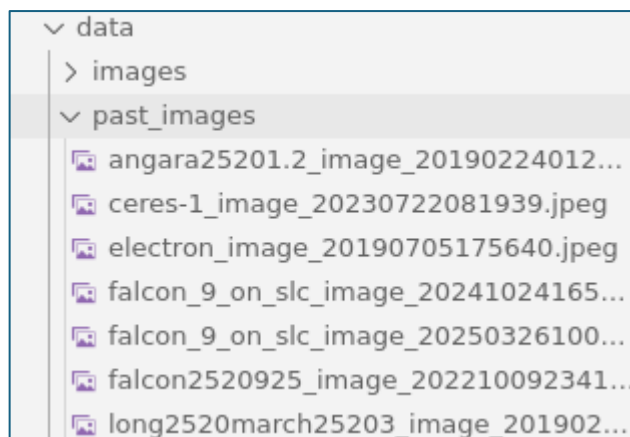


Рисунок 16

Видим, что изображения тоже были получены.

3. Оценить возможные ошибки в структуре данных JSON и предложить их исправление

1) Для того, чтобы проанализировать структуру полученных данных о старых запусках ракет перенесём данные с помощью скрипта на основную ОС.

Создаём файл `export_past_launches.sh` для автоматизации копирования (Рисунок 17).

```
$ export_past_launches.sh
1  #!/bin/bash
2
3  # Скрипт для экспорта данных о старых запусках из контейнера Airflow в /home/mgpu/airflow_export
4
5  # Параметры
6  CONTAINER_NAME="business_case_rocket_25-scheduler-1"
7  AIRFLOW_DATA_DIR="/opt/airflow/data"
8  HOST_EXPORT_DIR="/home/mgpu/airflow_export"
9  TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
10 EXPORT_PATH="$HOST_EXPORT_DIR/past_launches_$TIMESTAMP" # Добавляем префикс для старых запусков
11
12 # Проверяем существование директории /home/mgpu
13 if [ ! -d "/home/mgpu" ]; then
14     echo "Ошибка: Директория /home/mgpu не существует!"
15     exit 1
16 fi
17
18 # Создаем директорию для экспорта
19 mkdir -p "$EXPORT_PATH/images"
20
21 # Устанавливаем правильные права
22 chown -R mgpu:mgpu "$HOST_EXPORT_DIR"
23 chmod -R 755 "$HOST_EXPORT_DIR"
24
25 # 1. Копируем файл past_launches.json (или launches.json, если у вас старые данные там)
26 echo "Копируем past_launches.json..."
27 if docker exec "$CONTAINER_NAME" test -f "$AIRFLOW_DATA_DIR/past_launches.json"; then
28     docker cp "$CONTAINER_NAME:$AIRFLOW_DATA_DIR/past_launches.json" "$EXPORT_PATH/" && \
29         chown mgpu:mgpu "$EXPORT_PATH/past_launches.json" || \
```

Рисунок 17

Делаем файл исполняемым (Рисунок 18).

```
sudo: ./export_past_launches.sh: command not found
● mgpu@mgpu-VirtualBox:~/Documents/workshop-on-ETL/business_case_rocket_25$ chmod +x export_past_launches.sh
```

Рисунок 18

Запускаем скрипт (Рисунок 19).

```
● mgpu@mgpu-VirtualBox:~/Documents/workshop-on-ETL/business_case_rocket_25$ sudo ./export_past_launches.sh
[sudo] password for mgpu:
Копируем past_launches.json...
Successfully copied 26.1kB to /home/mgpu/airflow_export/past_launches_20250328_122435/
Копируем изображения...
Копируем изображения из /opt/airflow/data/past_images...
Successfully copied 1.47MB to /home/mgpu/airflow_export/past_launches_20250328_122435/images/
Создаем README...
Данные успешно экспортированы в /home/mgpu/airflow_export/past_launches_20250328_122435
Владелец файлов: mgpu:mgpu
```

Рисунок 19

Проверяем изображения (Рисунок 20).

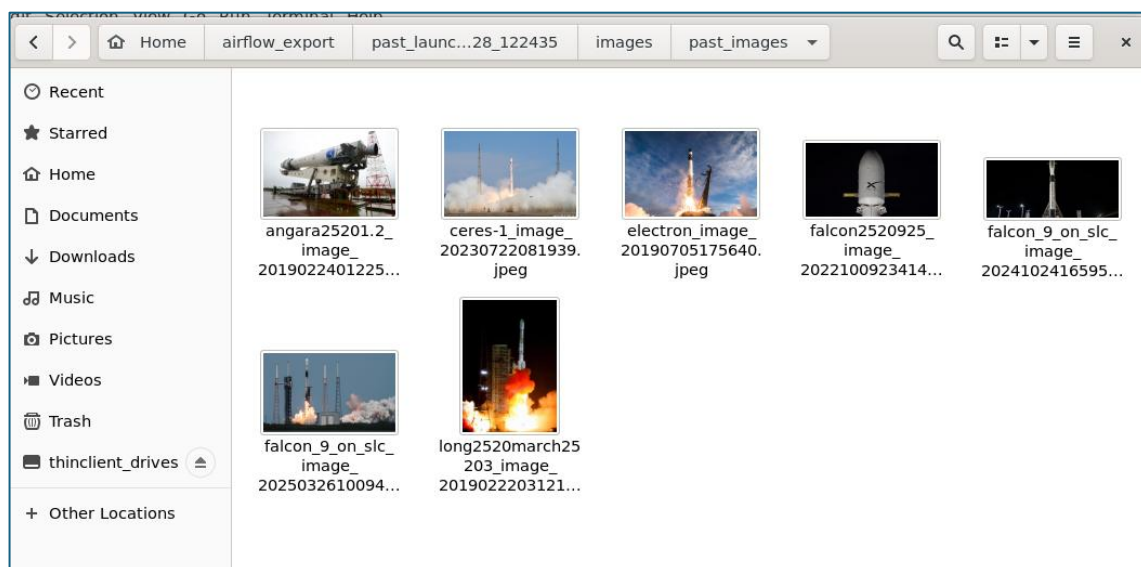


Рисунок 20

Также был скопирован файл JSON с данными (Рисунок 21).

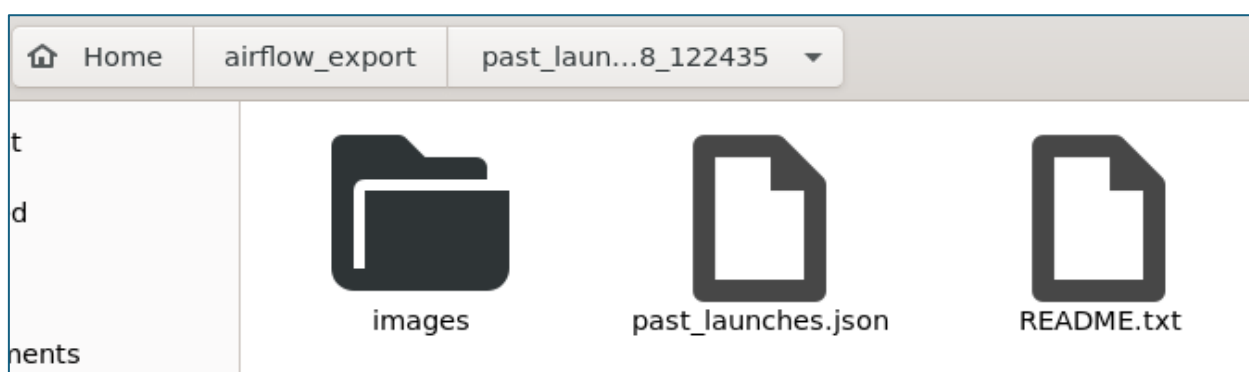


Рисунок 21

2) Переходим в Google Colab для анализа структуры полученных данных. Импортируем файл JSON и форматируем его для удобства чтения (Рисунок 22).

```
[9] 1 from google.colab import files
    2 uploaded = files.upload()

Выбрать файлы past_launches.json
• past_launches.json(application/json) - 24510 bytes, last modified: 28.03.2025 - 100% done
Saving past_launches.json to past_launches.json

1 import json
2
3 # Загрузка данных из файла
4 with open('past_launches.json', 'r') as file:
5     data = json.load(file)
6
7 # Форматирование и вывод читаемого JSON
8 formatted_json = json.dumps(data, indent=4)
9 print(formatted_json)

{
  "count": 7128,
  "next": "https://ll.thespacedevs.com/2.0.0/launch/previous/?limit=10&offset=10",
  "previous": null,
  "results": [
    {
      "id": "8bfaa532-18e6-4c40-9800-0714b5651f84",
      "url": "https://ll.thespacedevs.com/2.0.0/launch/8bfaa532-18e6-4c40-9800-0714b5651f84/",
      "launch_library_id": null,
      "slug": "falcon-9-block-5-starlink-group-11-7",
      "name": "Falcon 9 Block 5 | Starlink Group 11-7",
      "status": {
        "id": 3,
        "name": "Success"
      }
    }
  ]
}
```

Рисунок 22

Структура данных приведена в Листинге 2.

Листинг 2 – структура данных о старых запусках ракет.

```
{
  "id": "8bfaa532-18e6-4c40-9800-0714b5651f84",
  "url": "https://ll.thespacedevs.com/2.0.0/launch/8bfaa532-18e6-4c40-9800-0714b5651f84/",
  "launch_library_id": null,
  "slug": "falcon-9-block-5-starlink-group-11-7",
  "name": "Falcon 9 Block 5 | Starlink Group 11-7",
  "status": {
    "id": 3,
    "name": "Success"
  },
  "net": "2025-03-26T22:11:40Z",
  "window_end": "2025-03-27T01:56:10Z",
  "window_start": "2025-03-26T22:11:40Z",
  "inhold": false,
  "tbddtime": false,
  "tbdddate": false,
  "probability": null,
}
```

```
"holdreason": "",
"failreason": "",
"hashtag": null,
"launch_service_provider": {
  "id": 121,
  "url": "https://ll.thespacedevs.com/2.0.0/agencies/121/",
  "name": "SpaceX",
  "type": "Commercial"
},
"rocket": {
  "id": 8542,
  "configuration": {
    "id": 164,
    "launch_library_id": 188,
    "url": "https://ll.thespacedevs.com/2.0.0/config/launcher/164/",
    "name": "Falcon 9",
    "family": "Falcon",
    "full_name": "Falcon 9 Block 5",
    "variant": "Block 5"
  }
},
"mission": {
  "id": 7133,
  "launch_library_id": null,
  "name": "Starlink Group 11-7",
  "description": "A batch of 27 satellites for the Starlink mega-constellation - SpaceX's
project for space-based Internet communication system.",
  "launch_designator": null,
  "type": "Communications",
  "orbit": {
    "id": 8,
    "name": "Low Earth Orbit",
    "abbrev": "LEO"
  }
},
"pad": {
  "id": 16,
  "url": "https://ll.thespacedevs.com/2.0.0/pad/16/",
```

```
    "agency_id": null,
    "name": "Space Launch Complex 4E",
    "info_url": null,
    "wiki_url":
"https://en.wikipedia.org/wiki/Vandenberg_Space_Launch_Complex_4#SLC-4E",
    "map_url": "https://www.google.com/maps?q=34.632,-120.611",
    "latitude": "34.632",
    "longitude": "-120.611",
    "location": {
      "id": 11,
      "url": "https://ll.thespacedevs.com/2.0.0/location/11/",
      "name": "Vandenberg SFB, CA, USA",
      "country_code": "USA",
      "map_image": "https://thespacedevs-
prod.nyc3.digitaloceanspaces.com/media/map_images/location_11_20200803142416.jpg",
      "total_launch_count": 797,
      "total_landing_count": 26
    },
    "map_image": "https://thespacedevs-
prod.nyc3.digitaloceanspaces.com/media/map_images/pad_16_20200803143532.jpg",
    "total_launch_count": 185
  },
  "webcast_live": false,
  "image": "https://thespacedevs-
prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png",
  "infographic": null,
  "program": [
    {
      "id": 25,
      "url": "https://ll.thespacedevs.com/2.0.0/program/25/",
      "name": "Starlink",
      "description": "Starlink is a satellite internet constellation operated by American
aerospace company SpaceX",
      "agencies": [
        {
          "id": 121,
          "url": "https://ll.thespacedevs.com/2.0.0/agencies/121/",
          "name": "SpaceX",
```

```

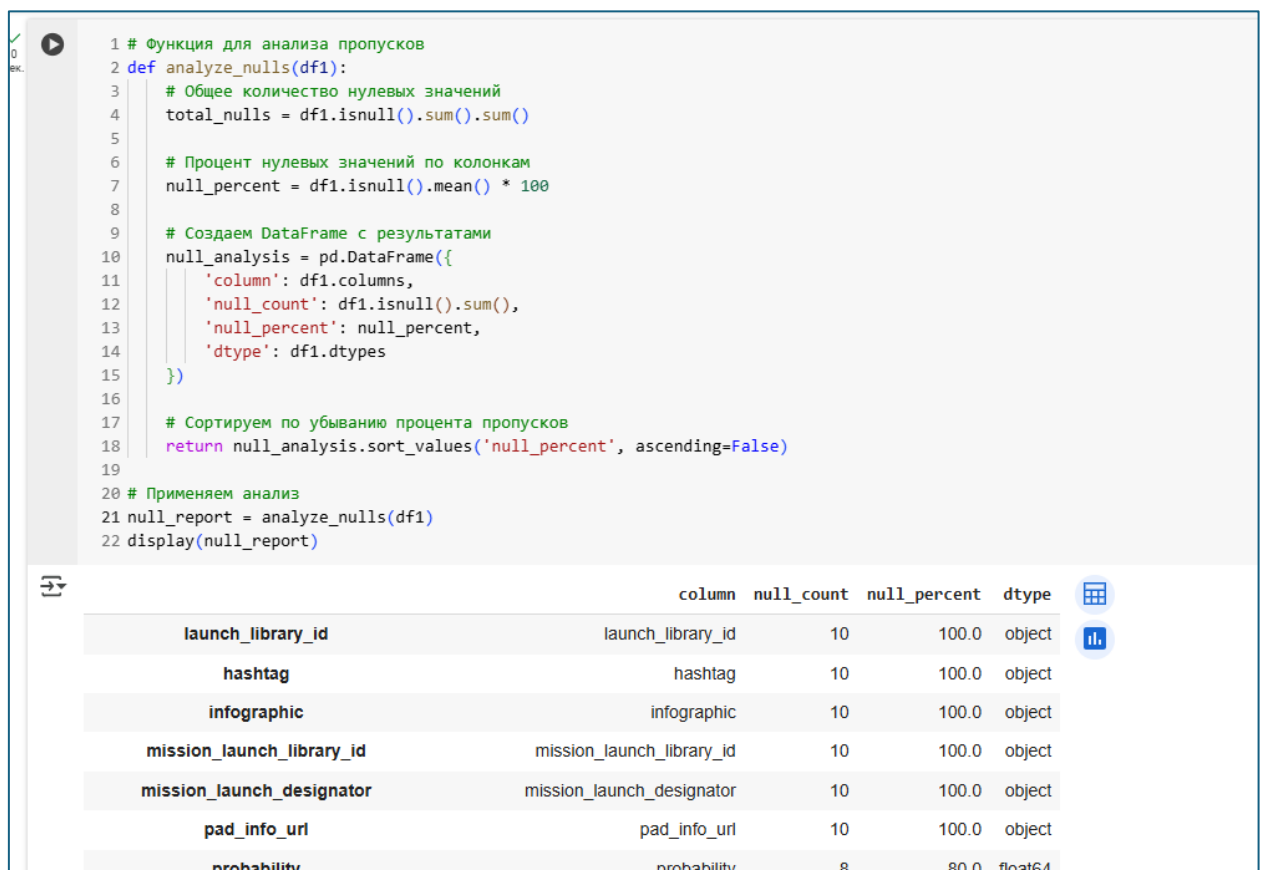
        "type": "Commercial"
    }
],
    "image_url": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/starlink_program_20231228154508.jpeg",
    "start_date": "2018-02-22T14:17:00Z",
    "end_date": null,
    "info_url": "https://starlink.com",
    "wiki_url": "https://en.wikipedia.org/wiki/Starlink"
}
]
}

```

Анализ структуры и возможных проблем:

Общая структура представлена в виде атрибутов верхнего уровня, таких как id, name, status, rocket, mission и pad, каждый из которых может содержать вложенные объекты или массивы.

Проанализируем пропущенные значения в данных (Рисунок 23).



```

1 # Функция для анализа пропусков
2 def analyze_nulls(df1):
3     # Общее количество нулевых значений
4     total_nulls = df1.isnull().sum().sum()
5
6     # Процент нулевых значений по колонкам
7     null_percent = df1.isnull().mean() * 100
8
9     # Создаем DataFrame с результатами
10    null_analysis = pd.DataFrame({
11        'column': df1.columns,
12        'null_count': df1.isnull().sum(),
13        'null_percent': null_percent,
14        'dtype': df1.dtypes
15    })
16
17    # Сортируем по убыванию процента пропусков
18    return null_analysis.sort_values('null_percent', ascending=False)
19
20 # Применяем анализ
21 null_report = analyze_nulls(df1)
22 display(null_report)

```

	column	null_count	null_percent	dtype
launch_library_id	launch_library_id	10	100.0	object
hashtag	hashtag	10	100.0	object
infographic	infographic	10	100.0	object
mission_launch_library_id	mission_launch_library_id	10	100.0	object
mission_launch_designator	mission_launch_designator	10	100.0	object
pad_info_url	pad_info_url	10	100.0	object
probability	probability	8	80.0	float64

Рисунок 23

1) Проблема: В структуре присутствуют поля, которые часто имеют значение null, например, `launch_library_id`, `info_url` и `hashtag`. Это занимает место и не добавляет полезной информации.

Решение: Рассмотреть возможность удаления этих полей, если они редко содержат значимые данные. Если эти поля необходимы, но не всегда доступны, можно оставить их, но учесть это при обработке данных.

2) Проблема: Данные о площадке запуска (`pad`) и ее местоположении (`pad.location`) сильно вложены, что усложняет доступ к отдельным атрибутам.

Решение: Рассмотреть возможность денормализации данных и вынесения наиболее часто используемых атрибутов на уровень выше. Например, `location_name` и `country_code` можно вынести непосредственно в объект `pad`.

Оценка эффективности

Использование Apache Airflow позволило автоматизировать парсинг данных из внешнего источника по API, а также выгрузку изображений. Использование DAG позволяет настроить периодичность выгрузки и это сокращает время выполнения задачи, если необходимо выполнять выгрузку на ежедневной основе.