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

Макарова Виктория Сергеевна

Бизнес кейс «Rocket»

Проектный практикум по разработке ETL-решений

Направление подготовки

38.03.05 Бизнес-информатика

Профиль подготовки

Аналитика данных и эффективное управление

Курс обучения: 4

Форма обучения: очная

Преподаватель: кандидат технических наук,

доцент Босенко Тимур Муртазович

Москва

2025

Задачи

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.

1.1 Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес-кейса «Rocket» в draw.io. Необходимо использовать:

Source Layer - слой источников данных.

Storage Layer - слой хранения данных.

Business Layer - слой для доступа к данным пользователей.

1.2 Спроектировать архитектуру DAG Бизнес-кейса «Rocket» в draw.io. Необходимо использовать:

Source Layer - слой источников данных.

Storage Layer - слой хранения данных.

Business Layer - слой для доступа к данным пользователей.

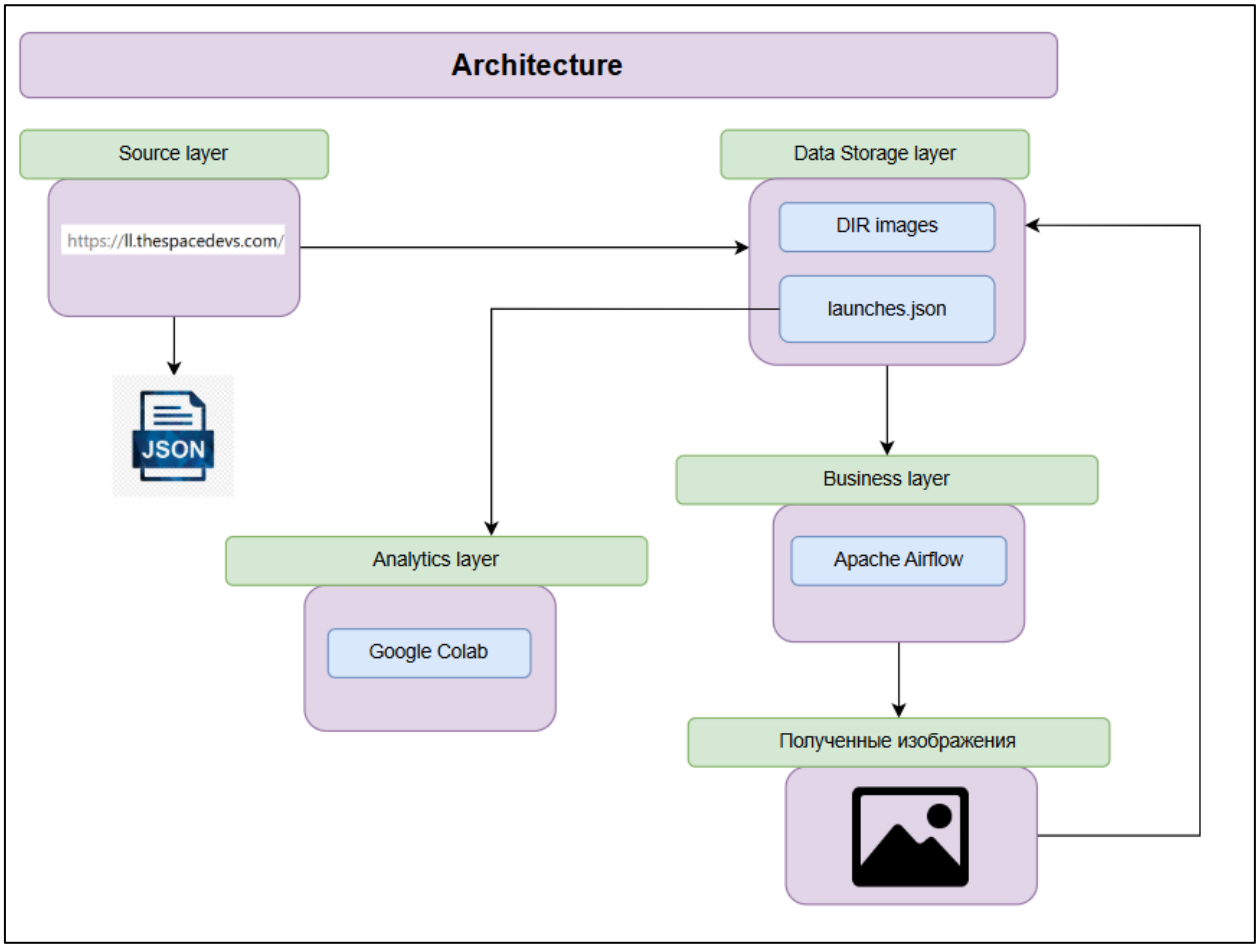
1.3 Построить диаграмму Ганта работы DAG в Apache Airflow.

Индивидуальное задание Вариант 7

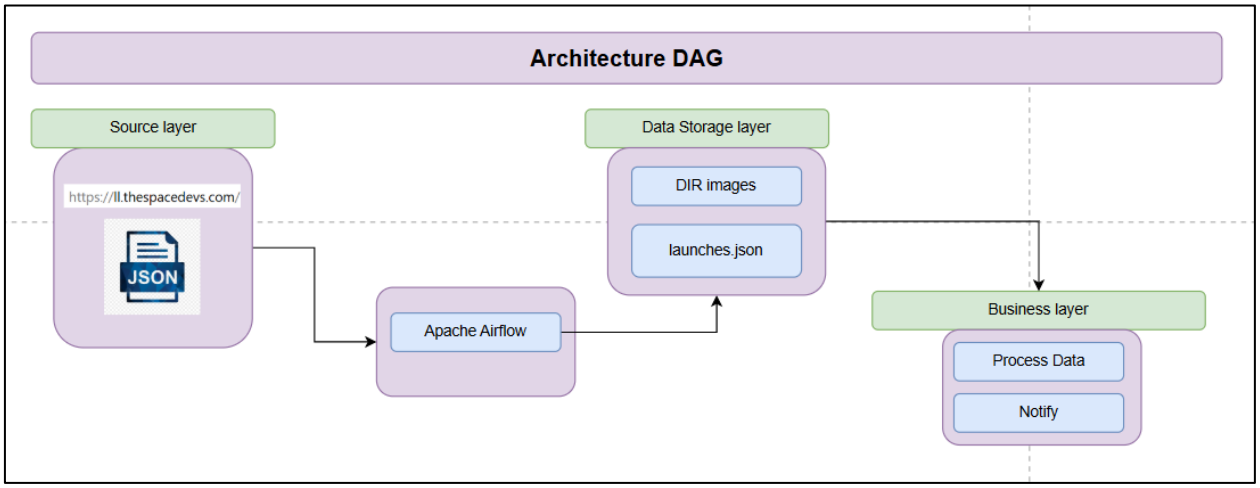
1. Создать отчет по скачиванию изображений на основе данных JSON.
2. Оценить необходимость использования BashOperator и PythonOperator для таких задач.
3. Изучить альтернативы для получения данных о стартах ракет с использованием API.

Ход работы

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



Представлена архитектура DAG Бизнес-кейса «Rocket» в draw.io

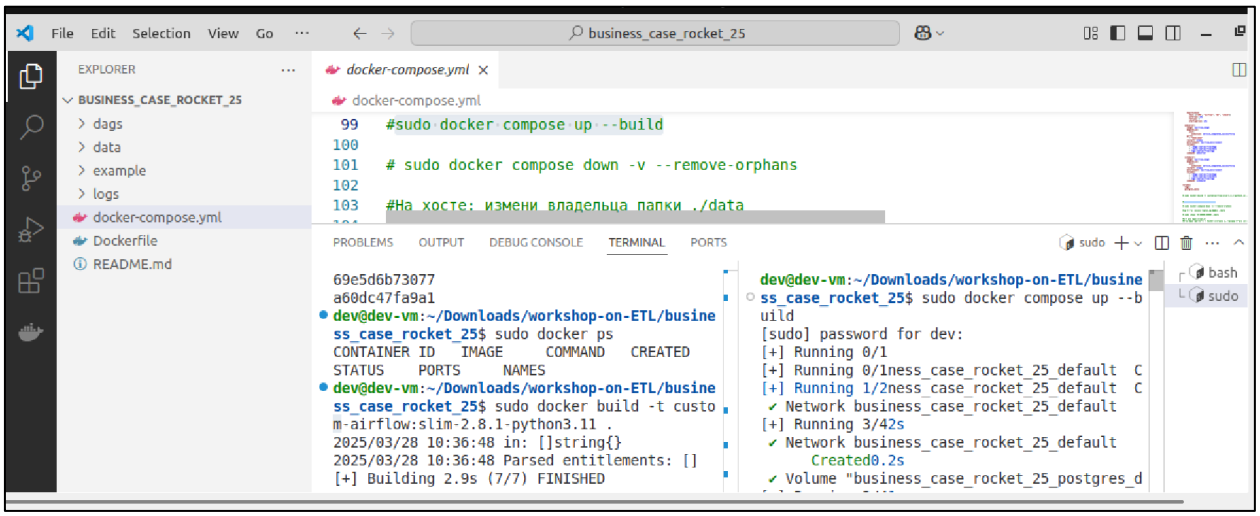



```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket_25$  sudo chown -R 50000:50000 ./data
[sudo] password for dev:
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket_25$
```

Проверка активных контейнеров и остановка их работы.

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket_25$ docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED       STATUS
 PORTS                                       NAMES
69e5d6b73077   custom-airflow:slim-2.8.1-python3.11   "/usr/bin/dumb-init …"   6 days ago    Up About a minute
 0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp   business_case_umbrella_25-webserver-1
a60dc47fa9a1   custom-airflow:slim-2.8.1-python3.11   "/usr/bin/dumb-init …"   6 days ago    Up About a minute
 8080/tcp                                    business_case_umbrella_25-scheduler-1
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket_25$ sudo docker stop $(sudo docker ps -q)
69e5d6b73077
a60dc47fa9a1
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket_25$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket_25$
```

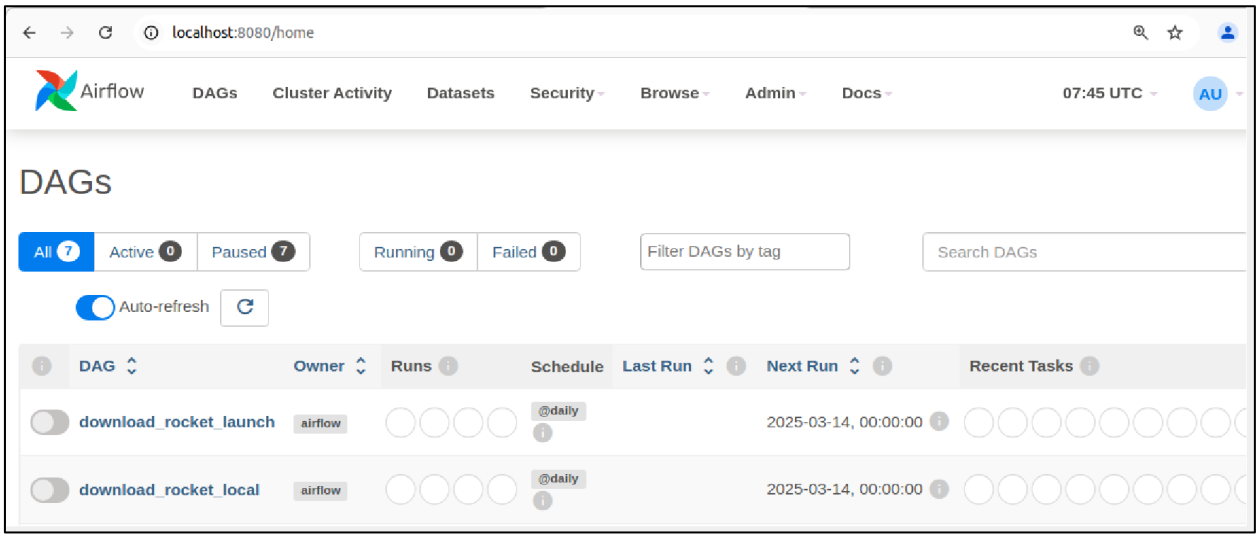
Билдим и запускаем контейнеры.



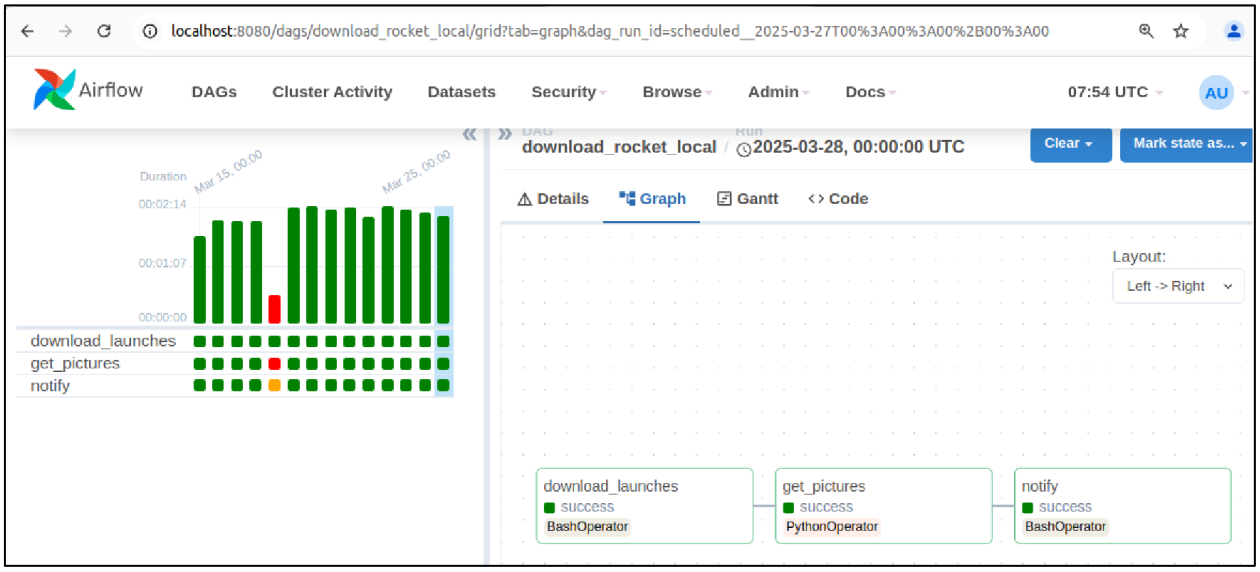
Код используемого дага

```python
import json
import pathlib

import airflow.utils.dates
import requests
import requests.exceptions as requests_exceptions
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator

dag = DAG(
    dag_id="download_rocket_local",
    description="Download rocket pictures of recently launched rockets.",
    start_date=airflow.utils.dates.days_ago(14),
    schedule_interval="@daily"
)

# Изменение пути для скачивания JSON-файла в папку data
download_launches = BashOperator(
    task_id="download_launches",
    bash_command="curl -o /opt/airflow/data/launches.json -L 'https://ll.thespacedevs.com/2.0.0/launch/upcoming'",  # noqa: E501
    dag=dag,
)

def _get_pictures():
    # Обеспечиваем существование директории для изображений в папке data
    images_dir = "/opt/airflow/data/images"
    pathlib.Path(images_dir).mkdir(parents=True, exist_ok=True)

    # Загружаем все картинки из launches.json
    with open("/opt/airflow/data/launches.json") as f:
        launches = json.load(f)
        image_urls = [launch["image"] for launch in launches["results"]]
        for image_url in image_urls:
            try:
                response = requests.get(image_url)
                image_filename = image_url.split("/")[-1]
                target_file = f"{images_dir}/{image_filename}"
                with open(target_file, "wb") as f:
                    f.write(response.content)
                print(f"Downloaded {image_url} to {target_file}")
            except requests_exceptions.MissingSchema:
```

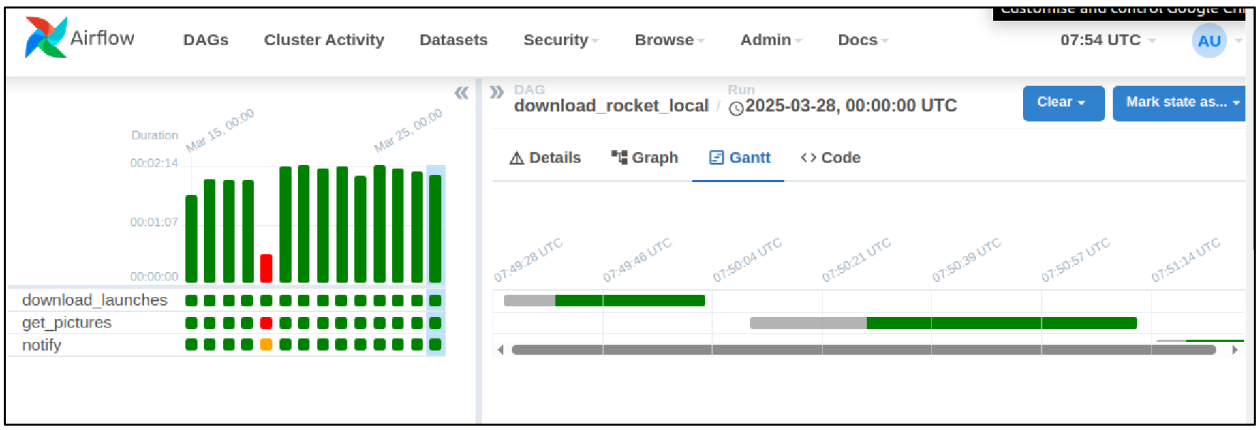
Проверяем доступность AirFlow.



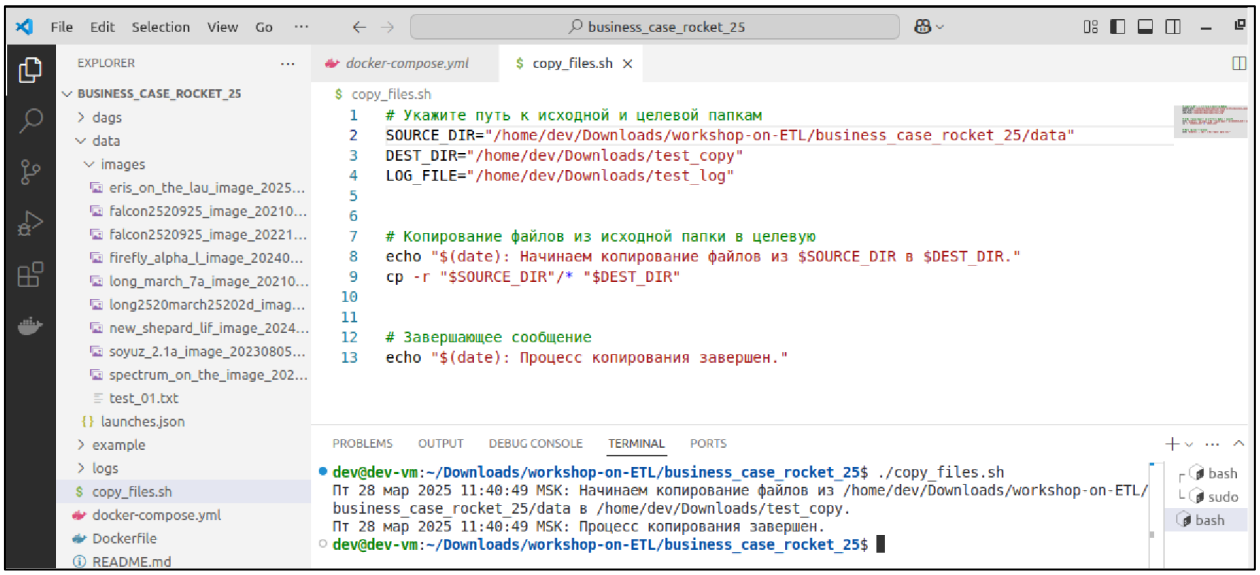
Запускаем даг download_rocket_local. Даг отработал успешно.



На рисунке представлена диаграмма Ганта выполнения дага.



Далее создаем исполняемый файл copy_files.sh, который автоматизирует выгрузку данных из контейнера в основную ОС данных, полученные в результате работы DAG в Apache Airflow.



Задание 1

Переходим к аналитике и созданию отчета в GoogleColab

Импортируем файл launches.json и выведем данные этого файла

```
import json
from collections import defaultdict

# Загрузка данных
with open('launches.json', 'r') as f:
    data = json.load(f)
data
```


```
{'count': 331,
 'next': 'https://ll.thespacedevs.com/2.0.0/launch/upcoming/?limit=10&offset=10',
 'previous': None,
 'results': [{'id': 'e652a538-6d40-4b55-97a6-7c757ec4e1e9',
   'url': 'https://ll.thespacedevs.com/2.0.0/launch/e652a538-6d40-4b55-97a6-7c757ec4e1e9/',
   'launch_library_id': None,
   '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,
   'tbdtime': False,
   'tbddate': False,
   'probability': None,
   'holdreason': '',
   'failreason': '',
   'hashtag': None,
   'launch_service_provider': {'id': 1046,
    'url': 'https://ll.thespacedevs.com/2.0.0/agencies/1046/',
    'name': 'Isar Aerospace',
    'type': 'Private'},
   'rocket': {'id': 8206,
    'configuration': {'id': 491,
     'launch_library_id': None,
     'url': 'https://ll.thespacedevs.com/2.0.0/config/launcher/491/',
```


Создадим отчет о предстоящих запусках

```python
import json
from datetime import datetime


# Анализ данных
total_launches = data['count']
upcoming_launches = len(data['results'])

# Обработка каждого запуска
launch_details = []
for launch in data['results']:
    # Парсинг даты
    net_time = datetime.strptime(launch['net'], '%Y-%m-%dT%H:%M:%SZ')

    details = {
        'name': launch['name'],
        'status': launch['status']['name'],
        'date': net_time.strftime('%Y-%m-%d'),
        'time': net_time.strftime('%H:%M:%S'),
        'provider': launch['launch_service_provider']['name'],
        'rocket': launch['rocket']['configuration']['full_name'],
        'window': f"{launch['window_start']} to {launch['window_end']}"
    }
    launch_details.append(details)

# Генерация отчета
report = f"""
=== ОТЧЕТ О ПРЕДСТОЯЩИХ ЗАПУСКАХ ===
Всего предстоящих запусков в системе: {total_launches}
Анализируемых в этом отчете: {upcoming_launches}

Детали ближайших запусков:
"""

for idx, launch in enumerate(launch_details, 1):
    report += f"""
{idx}. {launch['name']}
```

```
=== ОТЧЕТ О ПРЕДСТОЯЩИХ ЗАПУСКАХ ===
Всего предстоящих запусков в системе: 331
Анализируемых в этом отчете: 10

Детали ближайших запусков:

1. Spectrum | Maiden Flight
   - Ракета: Spectrum
   - Провайдер: Isar Aerospace
   - Дата: 2025-03-29
   - Время: 11:30:00 UTC
   - Статус: TBD
   - Окно запуска: 2025-03-29T11:30:00Z to 2025-03-29T14:30:00Z

2. Long March 7A | Unknown Payload
   - Ракета: Long March 7A
   - Провайдер: China Aerospace Science and Technology Corporation
   - Дата: 2025-03-29
   - Время: 16:05:00 UTC
   - Статус: Go
   - Окно запуска: 2025-03-29T15:57:00Z to 2025-03-29T17:15:00Z

3. Firefly Alpha | FLTA006 (Message in a Booster)
   - Ракета: Firefly Alpha
   - Провайдер: Firefly Aerospace
   - Дата: 2025-03-30
   - Время: 13:37:00 UTC
   - Статус: TBD
   - Окно запуска: 2025-03-30T13:37:00Z to 2025-03-30T15:16:00Z

4. Falcon 9 Block 5 | Starlink Group 6-80
   - Ракета: Falcon 9 Block 5
   - Провайдер: SpaceX
   - Дата: 2025-03-30
   - Время: 19:20:20 UTC
   - Статус: TBD
   - Окно запуска: 2025-03-30T19:16:00Z to 2025-03-30T23:45:00Z
```

Выведем статистику по используемым провайдерам.

Статистика по провайдерам

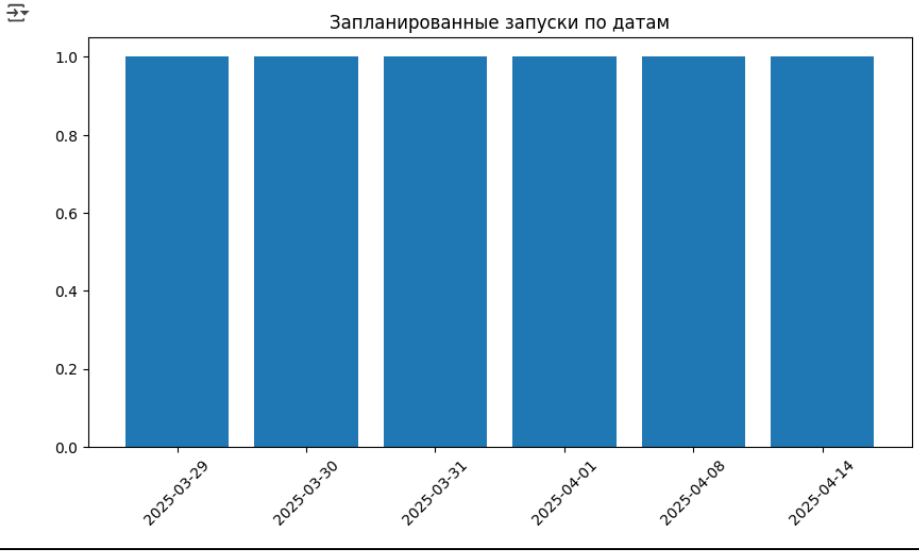
```python
from collections import Counter
providers = Counter([l['provider'] for l in launch_details])
providers
```

```
Counter({'Isar Aerospace': 1,
         'China Aerospace Science and Technology Corporation': 2,
         'Firefly Aerospace': 1,
         'SpaceX': 3,
         'Gilmour Space Technologies': 1,
         'Russian Federal Space Agency (ROSCOSMOS)': 1,
         'Blue Origin': 1})
```

Таким образом больше всего провайдеров SpaceX : 3

Построим график по ближайшим запланированным запускам ракет.

```python
import matplotlib.pyplot as plt

dates = [l['date'] for l in launch_details]
plt.figure(figsize=(10,5))
plt.bar(dates, [1]*len(dates))
plt.xticks(rotation=45)
plt.title('Запланированные запуски по датам')
plt.show()
```



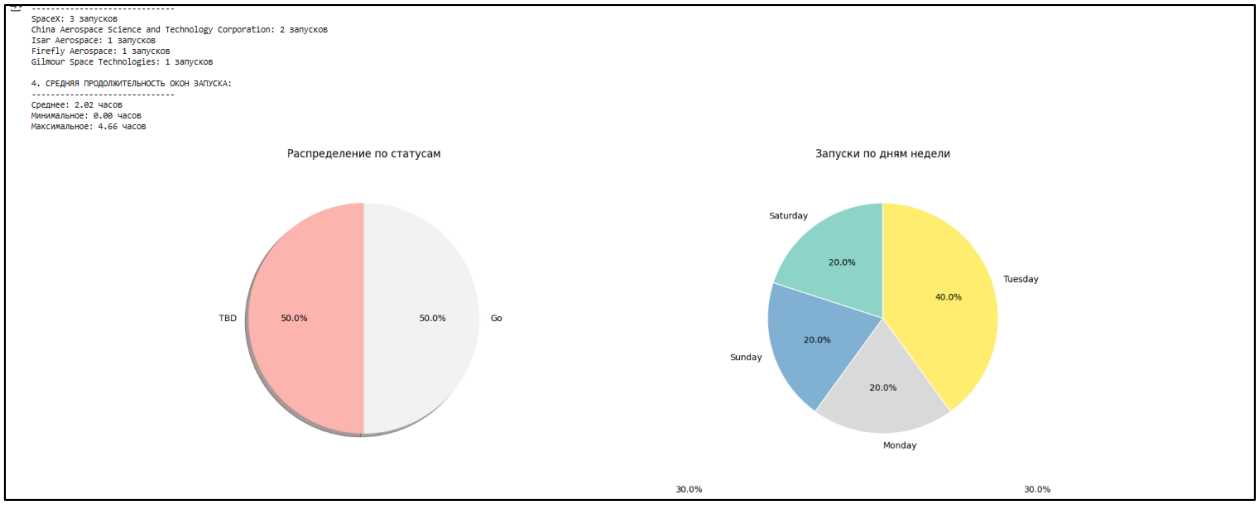
Выведем расширенных отчет о предстоящих запусках

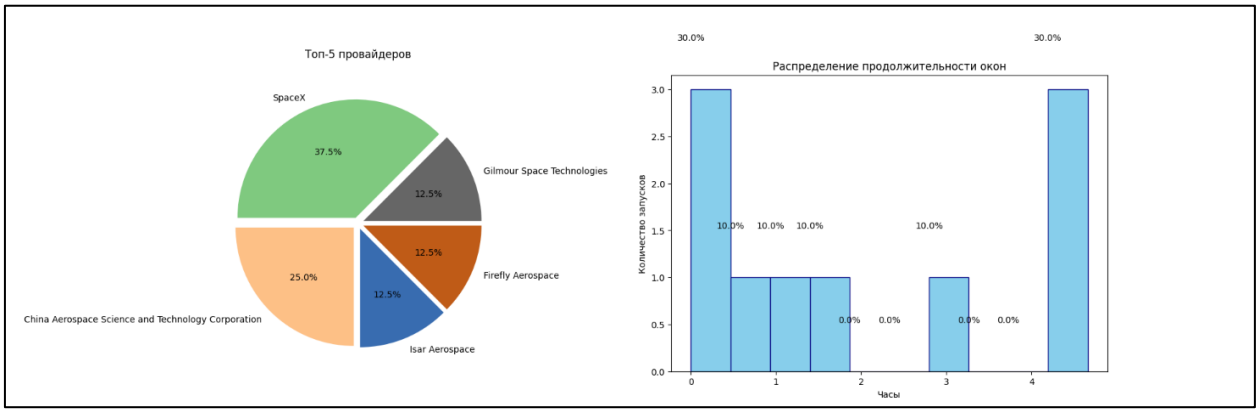
```python
import json
from datetime import datetime
from collections import Counter
import matplotlib.pyplot as plt
import calendar


# Основная обработка данных
launches = data['results']
total_launches = data['count']

# 1. Распределение по статусам запусков
status_counts = Counter([launch['status']['name'] for launch in launches])

# 2. Частота запусков по дням недели
weekday_counts = Counter()
for launch in launches:
    date = datetime.strptime(launch['net'], '%Y-%m-%dT%H:%M:%SZ')
    weekday_counts[calendar.day_name[date.weekday()]] += 1

# 3. Топ самых активных провайдеров
provider_counts = Counter([launch['launch_service_provider']['name'] for launch in launches])
top_providers = provider_counts.most_common(5)

# 4. Средняя продолжительность окон запуска
window_durations = []
for launch in launches:
    start = datetime.strptime(launch['window_start'], '%Y-%m-%dT%H:%M:%SZ')
    end = datetime.strptime(launch['window_end'], '%Y-%m-%dT%H:%M:%SZ')
    duration = (end - start).total_seconds() / 3600  # в часах
    window_durations.append(duration)
avg_window = sum(window_durations) / len(window_durations)

# Генерация текстового отчета
report = f"""
=== РАСШИРЕННЫЙ ОТЧЕТ О ПРЕДСТОЯЩИХ ЗАПУСКАХ ===
Всего предстоящих запусков: {total_launches}
Анализируется записей: {len(launches)}
```

```
=== РАСШИРЕННЫЙ ОТЧЕТ О ПРЕДСТОЯЩИХ ЗАПУСКАХ ===
Всего предстоящих запусков: 331
Анализируется записей: 10

1. РАСПРЕДЕЛЕНИЕ ПО СТАТУСАМ:
------------------------------
TBD: 5 (50.0%)
Go: 5 (50.0%)

2. ЧАСТОТА ЗАПУСКОВ ПО ДНЯМ НЕДЕЛИ:
------------------------------
Tuesday: 4 запусков
Saturday: 2 запусков
Sunday: 2 запусков
Monday: 2 запусков

3. ТОП-5 АКТИВНЫХ ПРОВАЙДЕРОВ:
------------------------------
SpaceX: 3 запусков
China Aerospace Science and Technology Corporation: 2 запусков
Isar Aerospace: 1 запусков
Firefly Aerospace: 1 запусков
Gilmour Space Technologies: 1 запусков

4. СРЕДНЯЯ ПРОДОЛЖИТЕЛЬНОСТЬ ОКОН ЗАПУСКА:
------------------------------
Среднее: 2.02 часов
Минимальное: 0.00 часов
Максимальное: 4.66 часов
```

Построение визуализаций на основе отчета.

```python

# Визуализация данных в виде pie-чартов
plt.figure(figsize=(18, 12))

# Цветовые палитры
status_colors = plt.cm.Pastel1(np.linspace(0, 1, len(status_counts)))
weekday_colors = plt.cm.Set3(np.linspace(0, 1, len(weekday_counts)))
provider_colors = plt.cm.Accent(np.linspace(0, 1, len(top_providers)))

# График 1: Распределение по статусам
plt.subplot(2, 2, 1)
plt.pie(status_counts.values(), labels=status_counts.keys(), autopct='%1.1f%%',
        colors=status_colors, startangle=90, shadow=True)
plt.title('Распределение по статусам', fontsize=12, pad=20)

# График 2: Дни недели
plt.subplot(2, 2, 2)
plt.pie(weekday_counts.values(), labels=weekday_counts.keys(), autopct='%1.1f%%',
        colors=weekday_colors, startangle=90, wedgeprops={'edgecolor': 'white'})
plt.title('Запуски по дням недели', fontsize=12, pad=20)

# График 3: Топ провайдеров
plt.subplot(2, 2, 3)
providers, counts = zip(*top_providers)
plt.pie(counts, labels=providers, autopct='%1.1f%%',
        colors=provider_colors, startangle=45, explode=[0.05]*len(top_providers))
plt.title('Топ-5 провайдеров', fontsize=12, pad=20)

# График 4: Продолжительность окон (оставляем гистограмму, так как это количественные данные)
plt.subplot(2, 2, 4)
n, bins, patches = plt.hist(window_durations, bins=10, color='skyblue', edgecolor='navy')
plt.title('Распределение продолжительности окон', fontsize=12)
plt.xlabel('Часы')
plt.ylabel('Количество запусков')

# Добавляем процентные значения к гистограмме
total = len(window_durations)
for i in range(len(patches)):
    plt.text(bins[i], n[i]+0.5, f'{n[i]/total:.1%}',
             ha='center', va='bottom')

plt.tight_layout(pad=3.0)
plt.savefig('launches_pie_analysis.png', dpi=300, bbox_inches='tight')
plt.show()
```





Таким образом:

- Успешность и неуспешность запуска ракет распределяется в разрезе 50/50.

- Самый частый день недели по запускам Четверг (приходится на него 4 запуска).

- Самый активный провайдер SpaceX (3 запуска).

- Средняя продолжительность окон запуска около 2 часов.

Задание 2

Оценить необходимость использования BashOperator и PythonOperator для таких задач.

DAG использует оба оператора:

- BashOperator для загрузки JSON через curl
- PythonOperator для обработки JSON и загрузки изображений

Это рациональное разделение:

- BashOperator идеален для простых HTTP-запросов (curl/wget)
- PythonOperator необходим для сложной логики (парсинг JSON, обработка изображений)

BashOperator подходит для простого скачивания файлов, работой с файловой системой, запуска CLI-утилит.

PythonOperator подходит там, где используется сложная логика: парсинг JSON, анализ данных, работа с Python-библиотеками, использование Xcom для передачи данных между тасками.

Задание 3

Изучить альтернативы для получения данных о стартах ракет с использованием API.

NASA Launch Schedule API

Источник: NASA API Portal

Особенности:

✓ Официальные данные NASA о предстоящих запусках

✓ Бесплатный доступ (до 1000 запросов/час)

✓ Включает миссии SpaceX, ULA, Rocket Lab (партнёрские запуски)

SpaceX API

Источник: GitHub / r-spacex/SpaceX-API
Особенности:
✅ Полные данные о запусках SpaceX (Falcon 9, Starship)
✅ История всех миссий с 2006 года
✅ Бесплатно и без ограничений

Launch Library 2 (альтернатива The Space Devs)

Источник: Launch Library 2
Особенности:
✅ Более стабильная версия API (v2)
✅ Подробные данные о ракетах, миссиях, площадках
✅ Бесплатно (но требуется API-ключ)

ВЫВОД:

Использование данного решения на основе AirFlow позволит компании операционную эффективность вследствие автоматизации процесса (загрузки, обработки, логгирования), также предоставляет возможность маштабировать данный процесс, подключая новые источники данных. Процесс стандартизован, что позволяет гарантированно выполнять по расписанию.

Компания будет получать улучшенное качество данных за счет логгирования, сохранения данных по шаблону (папки, наименования файлов). Вследствие чего эти данные будут использоваться для аналитики и предсказывания будущих запусков.