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

ДИСЦИПЛИНА:

Проектный практикум по разработке ETL-решений

Лабораторная работа 5.2

Разработка алгоритмов для трансформации данных. Airflow DAG

Выполнила: Сергеева А. И., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

Цель работы: разработать и настроить DAG в Apache Airflow для автоматизации процесса загрузки данных о предстоящих запусках ракет и скачивания изображений, связанных с этими запусками. Результаты работы DAG должны быть выгружены на основную операционную систему, а также построена архитектура аналитического решения.

Задачи:

- Клонировать на ПК задание Бизнес-кейс «Rocket» в домашний каталог ВМ.

- Запустить контейнер с кейсом, изучить основные элементы DAG в Apache Airflow.

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

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

- Спроектировать архитектуру DAG Бизнес-кейса «Rocket» в draw.io.

- Построить диаграмму Ганта работы DAG в Apache Airflow.

Ход работы:

download_rocket_launches.py предназначен для загрузки данных о предстоящих запусках ракет, скачивания изображений, связанных с запуском и для подготовки уведомлений о количестве скачанных изображений. listing_2_10.py аналогичен предыдущему, listing_2_02.py выполняет задачи без расписания, listing_2_03.py не содержит задач, запускается вручную, listing_2_04.py только загружает данные о ракетах, listing_2_06.py ещё скачивает изображения, но без уведомления. Для работы был запущен контейнер с Airflow на рисунке 1.

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket$ sudo docker compose up -d
WARN[0000] /home/dev/Downloads/workshop-on-ETL/business_case_rocket/docker-compose.yml: the
[+] Running 4/4
 ✔ Container business_case_rocket-postgres-1   Started
 ✔ Container business_case_rocket-scheduler-1  Started
 ✔ Container business_case_rocket-init-1       Started
 ✔ Container business_case_rocket-webserver-1  Started
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket$
```

Рисунок 1 – Запуск контейнера

Для начала необходимо было ознакомиться с основными элементами интерфейса Apache Airflow. Для перехода в интерфейс необходимо было перейти на адресу http://localhost:8080/ на рисунке 2. Пароль и логин admin.

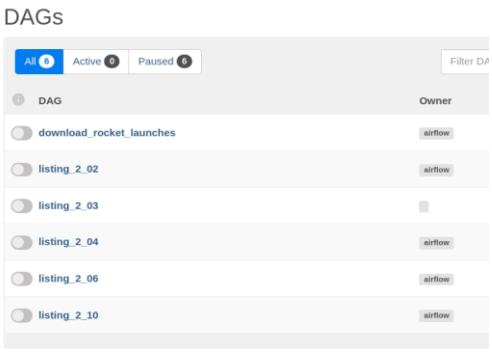


Рисунок 2 – Запуск Airflow

Первый основной элемент - DAGs (Directed Acyclic Graphs). Это главная страница, на которой отображаются все доступные DAG (Directed Acyclic Graphs). DAG — это набор задач, которые выполняются в определенном порядке. На данной вкладке есть список всех доступных DAG с их статусами (например, "Running", "Failed" и т. д.). Всего 6 DAGs и пока не запускались.

Graph View - графическое представление DAG, где задачи отображаются в виде блоков, а зависимости между ними — в виде стрелок. Все DAGs были запущены с разными параметрами, чтобы посмотреть на различия в выполнении. Для DAG графическое представление видно на рисунках 3-8.

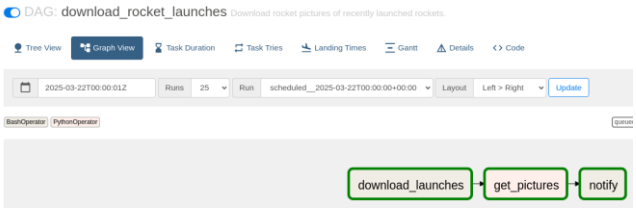


Рисунок 3 – Графическое представление DAG download_rocket_launches

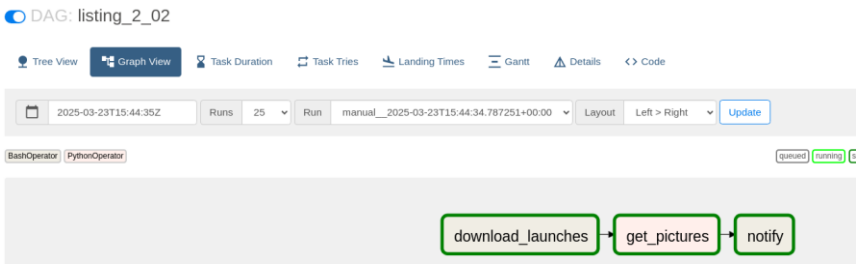


Рисунок 4 - Графическое представление DAG listing_2_02

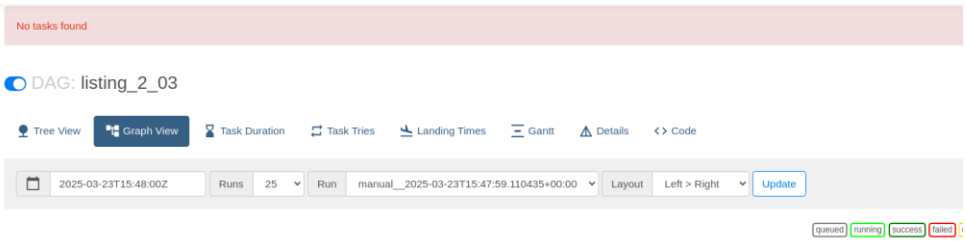


Рисунок 5 - Графическое представление DAG listing_2_03 без задач

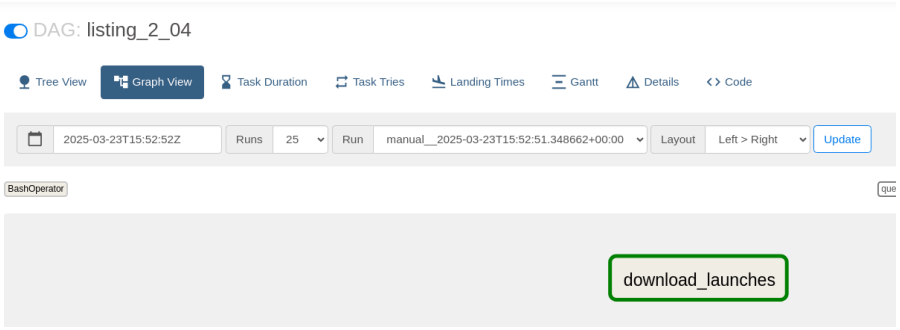


Рисунок 6 - Графическое представление DAG listing_2_04

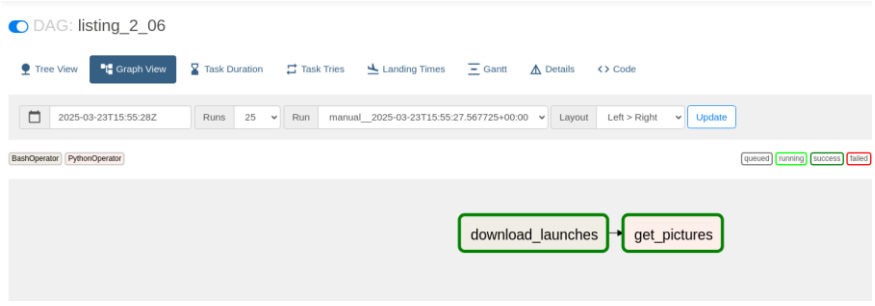


Рисунок 7 - Графическое представление DAG listing_2_06

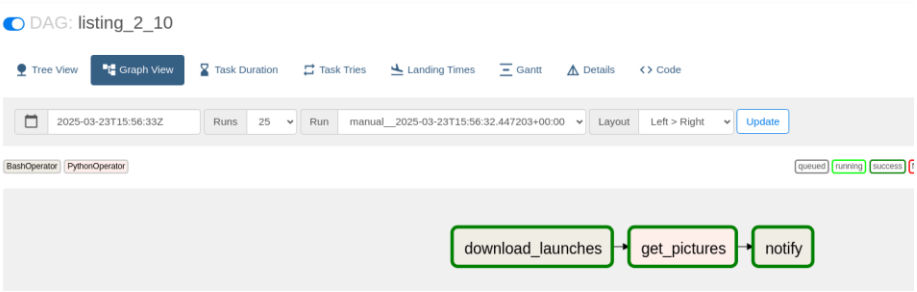


Рисунок 8 - Графическое представление DAG listing_2_10

Tree View - древовидное представление выполнения DAG за разные периоды времени. Каждый кружок представляет собой задачу, а также есть возможность фильтровать задачи по статусу или дате, что продемонстрировано на рисунке 9 для примера DAG download_rocket_launches.

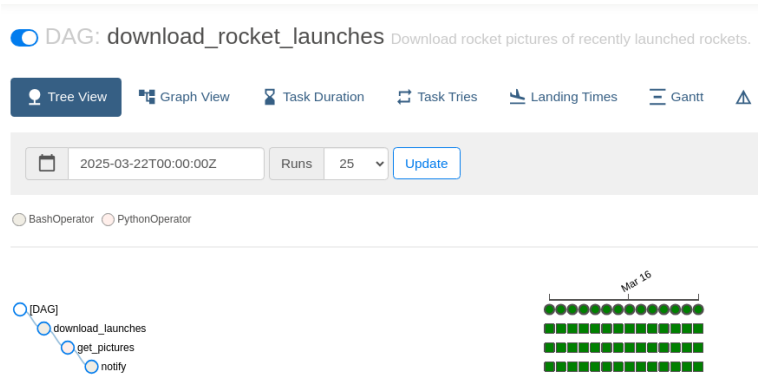


Рисунок 9 – Древовидное представление DAG download_rocket_launches

А также есть диаграмма Ганта, по которой также можно отслеживать выполнение DAG и представлена на рисунках 10-14.

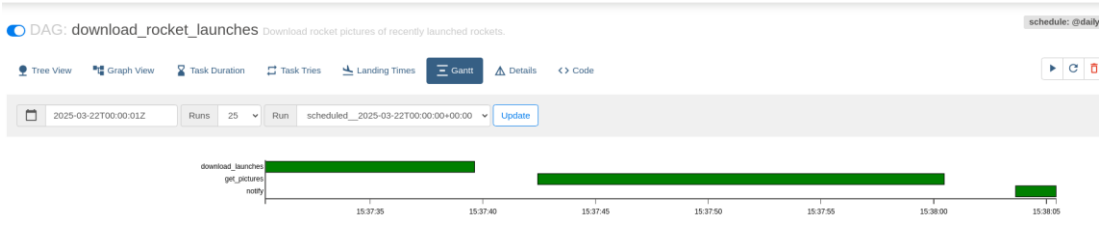


Рисунок 10 – Диаграмма Ганта DAG download_rocket_launches

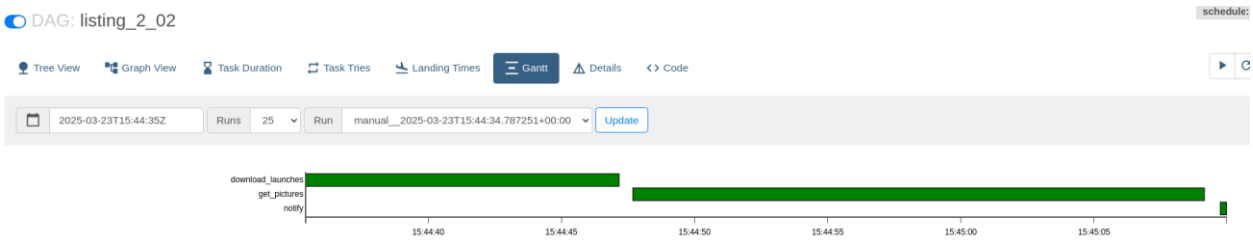


Рисунок 11 - Диаграмма Ганта DAG listing_2_02

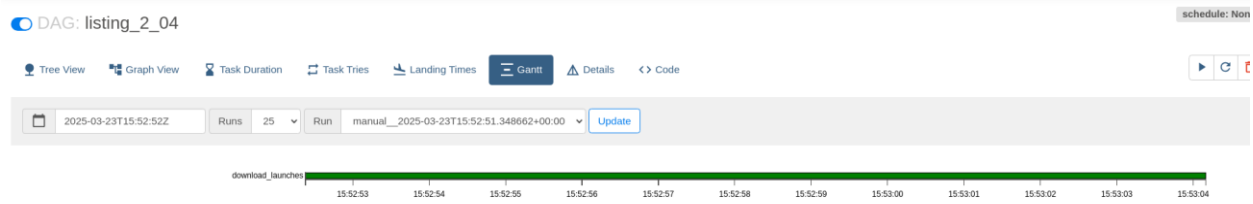


Рисунок 12 - Диаграмма Ганта DAG listing_2_04

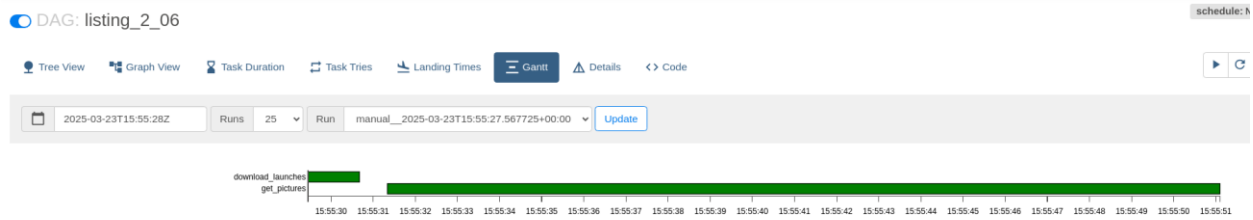


Рисунок 13 - Диаграмма Ганта DAG listing_2_06

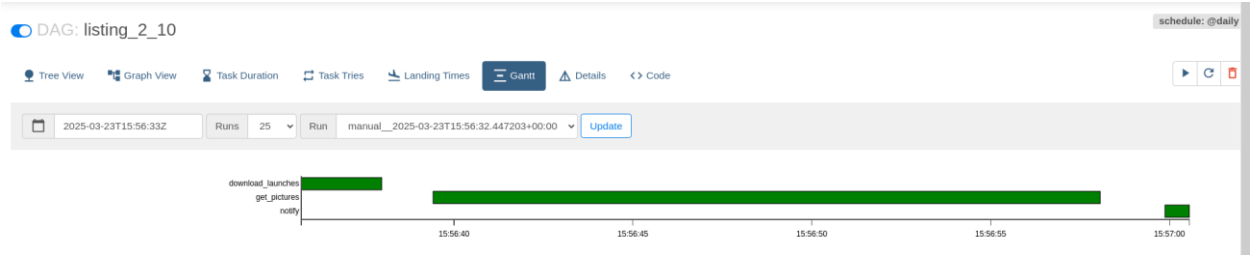
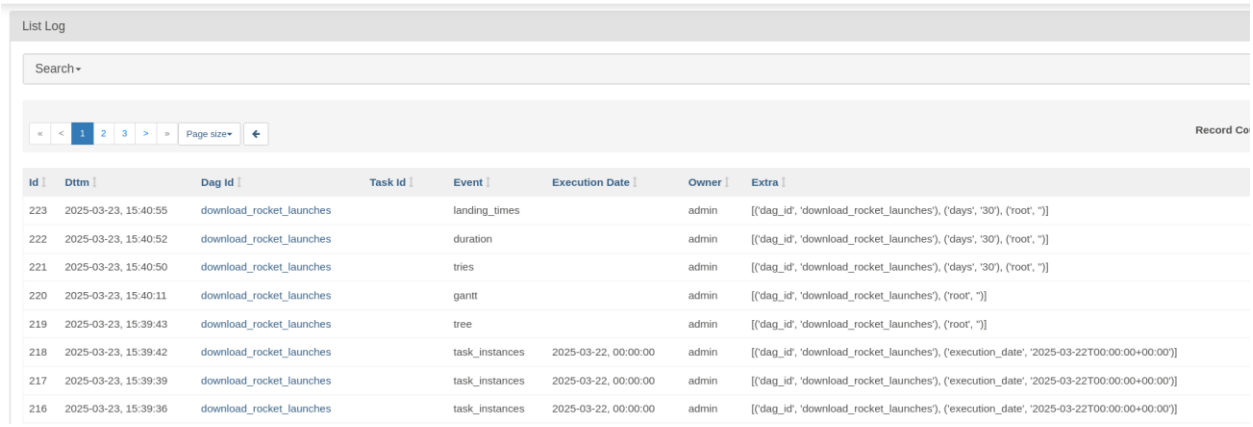


Рисунок 14 - Диаграмма Ганта DAG listing_2_10

Далее был просмотрен лист логов на рисунке 15.



Id	Dttm	Dag Id	Task Id	Event	Execution Date	Owner	Extra
223	2025-03-23, 15:40:55	download_rocket_launches		landing_times		admin	[('dag_id', 'download_rocket_launches'), ('days', '30'), ('root', '')]
222	2025-03-23, 15:40:52	download_rocket_launches		duration		admin	[('dag_id', 'download_rocket_launches'), ('days', '30'), ('root', '')]
221	2025-03-23, 15:40:50	download_rocket_launches		tries		admin	[('dag_id', 'download_rocket_launches'), ('days', '30'), ('root', '')]
220	2025-03-23, 15:40:11	download_rocket_launches		gantt		admin	[('dag_id', 'download_rocket_launches'), ('root', '')]
219	2025-03-23, 15:39:43	download_rocket_launches		tree		admin	[('dag_id', 'download_rocket_launches'), ('root', '')]
218	2025-03-23, 15:39:42	download_rocket_launches		task_instances	2025-03-22, 00:00:00	admin	[('dag_id', 'download_rocket_launches'), ('execution_date', '2025-03-22T00:00:00+00:00')]
217	2025-03-23, 15:39:39	download_rocket_launches		task_instances	2025-03-22, 00:00:00	admin	[('dag_id', 'download_rocket_launches'), ('execution_date', '2025-03-22T00:00:00+00:00')]
216	2025-03-23, 15:39:36	download_rocket_launches		task_instances	2025-03-22, 00:00:00	admin	[('dag_id', 'download_rocket_launches'), ('execution_date', '2025-03-22T00:00:00+00:00')]

Рисунок 15 – Лист логов

Далее лист логов для DAG listing_2_10 был выгружен в ОС, как видно на рисунке 16.

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket$ docker cp business_case_rocket-scheduler-1:/opt/airflow/logs/listing_2_10 /home/logs
Successfully copied 1.02kB to /home/logs
```

Рисунок 16 – Выгрузка логов в каталог ОС

А сами результаты работы DAG, а именно изображения, были также выгружены в ОС на рисунке 17.

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket$ sudo docker cp business_case_rocket-scheduler-1:/tmp/images /home/images
Successfully copied 1.62MB to /home/images
```

Рисунок 17 – Выгрузка результатов работы DAG в ОС

Завершающим этапом необходимо было создать исполняемый файл .sh для автоматизации выгрузки данных. С его содержимым можно ознакомиться на рисунке 18.

```
#!/bin/bash

# Параметры
CONTAINER_NAME="business_case_rocket-scheduler-1"
DAG_ID="download_rocket_launches"
TASK_ID="get_pictures"
HOST_OUTPUT_DIR="$HOME/Downloads/airflow_output"

# Создание директории для выгрузки
mkdir -p "${HOST_OUTPUT_DIR}"

echo "Поиск последних логов..."
LATEST_LOG=$(docker exec $CONTAINER_NAME bash -c "ls -td /opt/airflow/logs/${DAG_ID}/${TASK_ID}/*/ | head -1")/1.log

if [ -n "$LATEST_LOG" ]; then
    echo "Копирование логов из ${LATEST_LOG}..."
    docker cp "${CONTAINER_NAME}:${LATEST_LOG}" "${HOST_OUTPUT_DIR}/logs.zip" && \
        echo "Логи успешно скопированы в ${HOST_OUTPUT_DIR}/logs.zip" || \
        echo "Ошибка копирования логов"
else
    echo "Логи не найдены. Убедитесь, что DAG выполнялся."
fi

# 2. Копирование изображений
echo "Копирование изображений из /tmp/images..."
docker cp "${CONTAINER_NAME}:/tmp/images" "${HOST_OUTPUT_DIR}/" && \
    echo "Изображения успешно скопированы в ${HOST_OUTPUT_DIR}/images" || \
    echo "Ошибка копирования изображений (возможно задача не выполнялась)"

# 3. Проверка результата
echo -e "\nСодержимое ${HOST_OUTPUT_DIR}:"
ls -lh "${HOST_OUTPUT_DIR}"
```

Рисунок 18 - .sh файл для автоматической выгрузки данных

Далее необходимо было сделать созданный файл исполняемым и запустить скрипт на рисунке 19.

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket$ chmod +x export_data.sh
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_rocket$ ./export_data.sh
Поиск последних логов...
Копирование логов из /opt/airflow/logs/download_rocket_launches/get_pictures/2025-03-24T13:55:45.176807+00:00//1.log...
Successfully copied 6.66kB to /home/dev/Downloads/airflow_output/logs.zip
Логи успешно скопированы в /home/dev/Downloads/airflow_output/logs.zip
Копирование изображений из /tmp/images...
Successfully copied 1.62MB to /home/dev/Downloads/airflow_output/
Изображения успешно скопированы в /home/dev/Downloads/airflow_output/images
```

Рисунок 19 – Запуск скрипта по загрузке результатов DAG

Данные берутся из API, который предоставляет доступ к базе данных о космических запусках, ракетах, агентствах и событиях. Пользователи могут получать информацию о предстоящих и прошедших запусках, а также сведения о ракетах и космических агентствах, участвующих в миссиях. Дополнительно можно использовать погодные API, данные о стоимости запуска, чтобы делать более точные выводы о днях, когда запуски эффективны, их успешности/неуспешности. Данные хранятся в локальном хранилище, а именно /tmp/launches.json и /tmp/images/. А их можно, например, хранить в PostgreSQL, а изображения в MinIO, т. к. изображений много и MinIO позволяет хранить неограниченное количество объектов любого размера. Объём хранилища легко масштабируется добавлением новых серверов, а также к изображениям есть быстрый доступ. В бизнес-слое может быть визуализация для демонстрации карты запусков, уведомления о предстоящих запусках, а также просмотр моделей ракет через Model Viewer от Google с открытым исходным кодом, позволяет отображать 3D-контент в интернете, в данном случае ракеты. Предложенная архитектура бизнес-кейса представлена на рисунке 20.

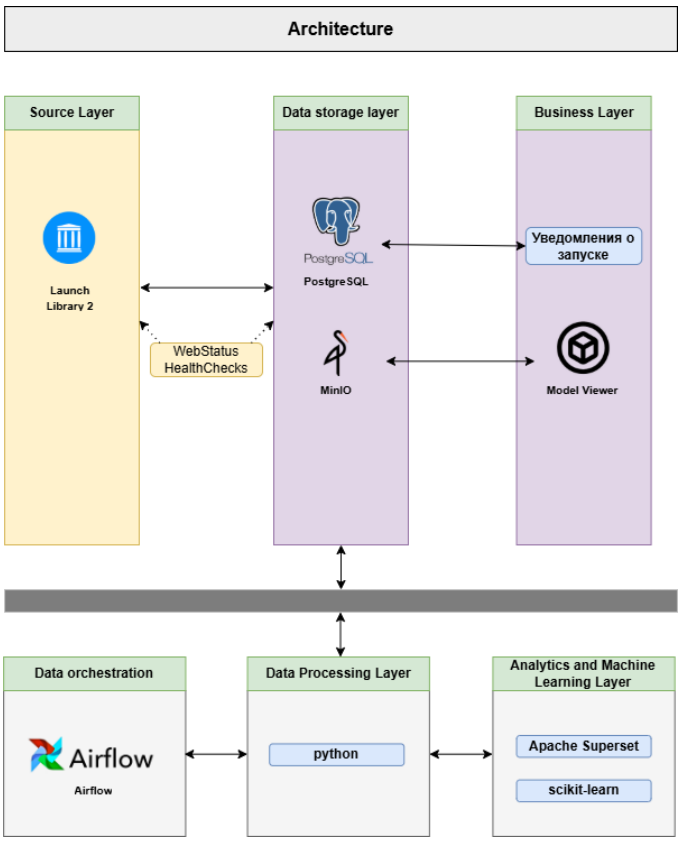


Рисунок 20 – Архитектура бизнес-кейса по ракетам

Архитектура DAG представлена на рисунке 21.

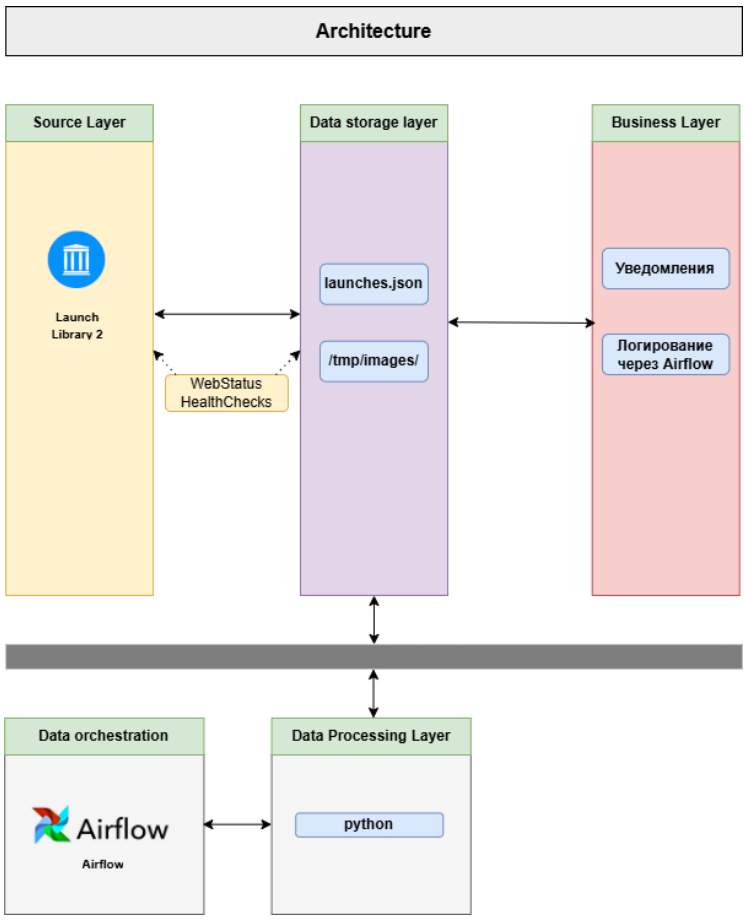


Рисунок 21 – Архитектура DAG

Выводы:

В рамках работы были изучены основные элементы Apache Airflow. Реализован ETL-процесс загрузки данных о запусках ракет и изображений, а также автоматическая выгрузка результатов. Построены 2 архитектуры: для самого DAG и для бизнес-кейса по запуску ракет, были проанализированы разные способы создания DAGs и выявлено, когда какой лучше. Для начала был DAG без задач, чтобы увидеть шапку DAG. Далее с одной задачей, лучше при создании DAG добавлять задачи поэтапно для лучшего мониторинга ошибок, как в DAG listing_2_4.py, в listing_2_6.py уже вводилась еще 1 задача, в конце создаётся Bash-команда для вывода количества изображений и конечной проверки работоспособности DAG. .sh-скрипты позволяют автоматизировать процессы и хотя Airflow поддерживает BashOperator, сложные Bash-команды лучше выносить в отдельные .sh-файлы. .sh-файл для

выгрузки результатов DAG был успешно создан. Ель и задач ибыли выполнены.