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

Вебинар 28-03-2025

Бизнес-кейс «Rocket»

Выполнила: Сачкова Г.Г., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

Задачи:

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

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

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

4. Построить диаграмму Ганта работы DAG в Apache Airflow.

На рисунке 1 показана верхнеуровневая архитектура аналитического решения Rocket.

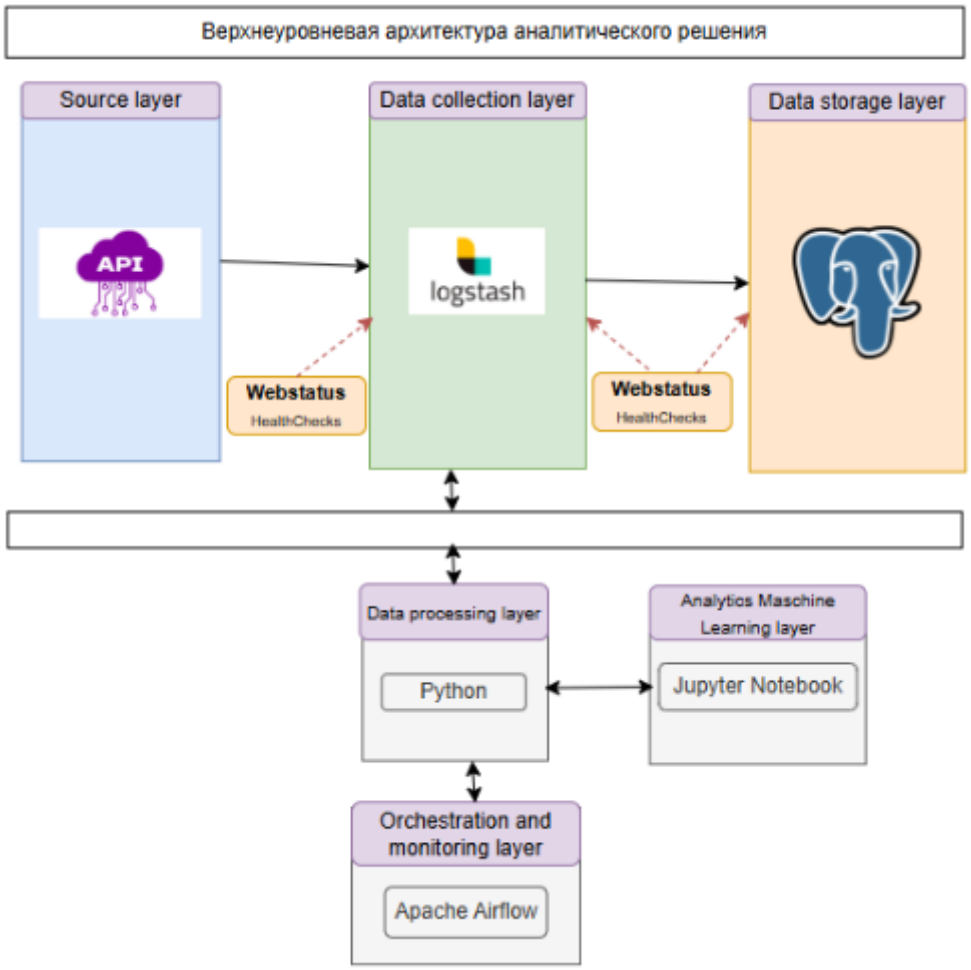


Рисунок 1 – Верхнеуровневая архитектура

Общий процесс

1. Сбор данных: API Rocket извлекает данные.
2. Сбор и обработка данных: Logstash обрабатывает и отправляет данные в PostgreSQL.
3. Хранение данных: Данные сохраняются в PostgreSQL.
4. Обработка данных: Python обрабатывает данные из PostgreSQL.
5. Анализ и машинное обучение: Jupyter Notebook используется для анализа и обучения моделей.
6. Оркестрация и мониторинг: Apache Airflow управляет всем процессом и отслеживает выполнение задач.

На рисунке 2 показана архитектура дага.

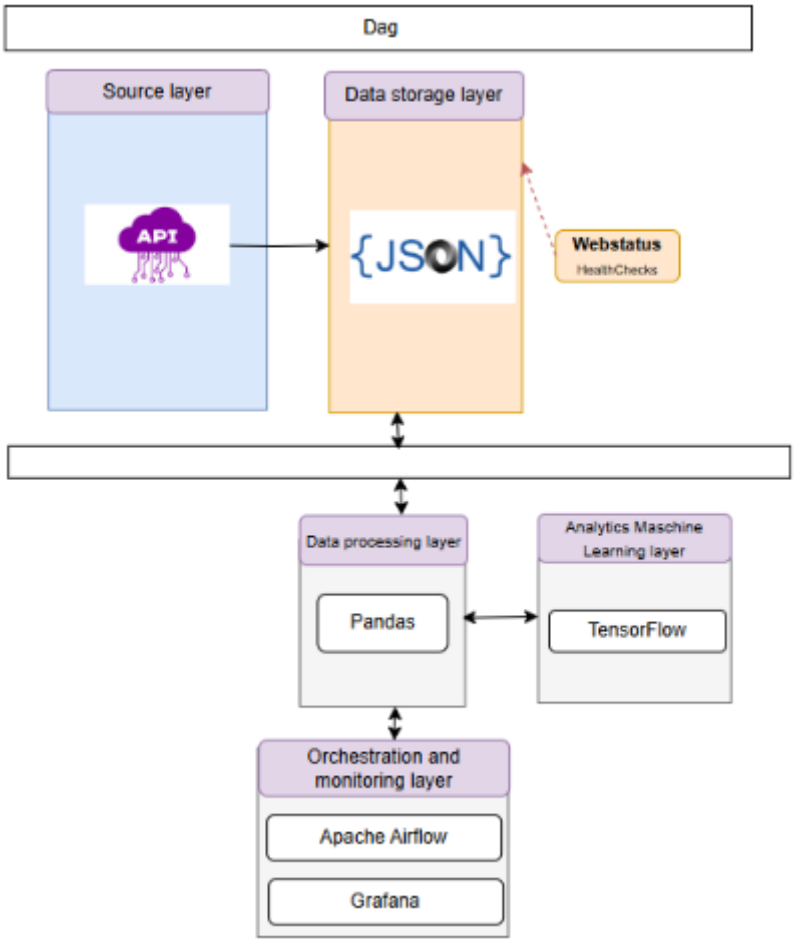


Рисунок 2 – Архитектура DAG

На рисунке 3 показан граф дага.

Рисунок 3 – Граф DAG

На рисунке 4 показана диаграмма Ганта

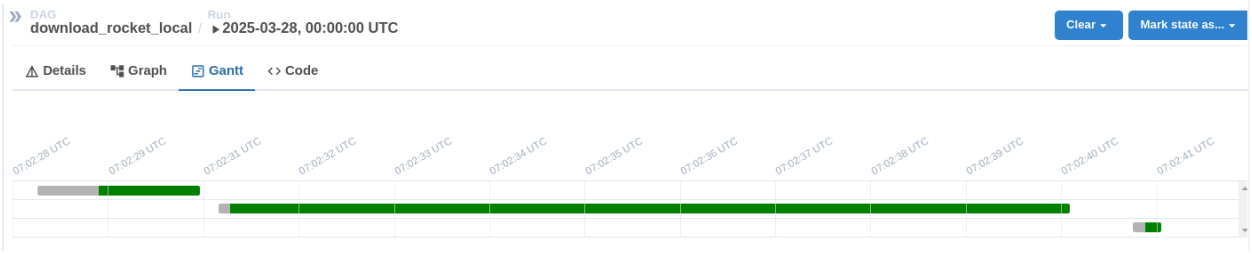


Рисунок 4 – Диаграмма Ганта

На рисунке 5 продемонстрирован исполняемый файл sh для автоматической выгрузки данных из контейнера в основную ОС данных, полученные в результате работы DAG в Apache Airflow.

```
#!/bin/bash
CONTAINER_NAME="business_case_rocket_25-scheduler-1"
CONTAINER_PATH="/opt/airflow/data/images"
HOST_PATH="/home/andreyn/papka"
mkdir -p "$HOST_PATH"
docker cp "$CONTAINER_NAME:$CONTAINER_PATH/." "$HOST_PATH"
echo "The papka has been uploaded"
```

Рисунок 5 – Исполняемый файл

На рисунке 6 показано выполнение этого файла.

```
andreyn@Herrifly:~/PycharmProjects/workshop-on-ETL/business_case_rocket_25$ ./bash.sh
                                        Successfully copied 1.71MB to /home/andreyn/papka
The papka has been uploaded
```

Рисунок 6 – Выполнение исполняемого файла

На рисунке 7 показано, что все файлы скопировались.

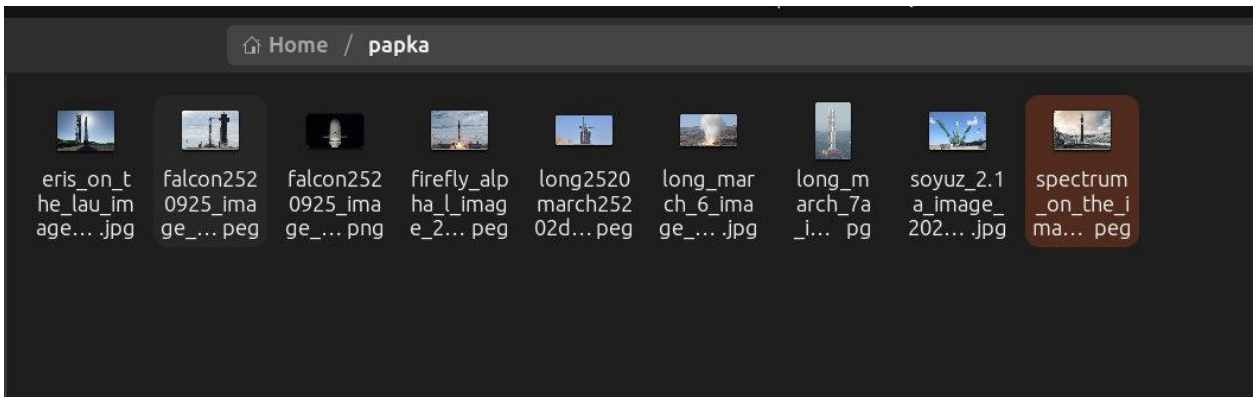


Рисунок 7 – Все изображения скопировались

Выполнение индивидуального задания. Вариант 11:

1) Разработать метод анализа доступности изображений на сервере по URL
2) Оценить производительность текущего кода на больших объемах данных
3) Создать план по улучшению скорости скачивания изображений с учетом возможных ошибок

На рисунке 8 показан код созданного дага для выполнения задания.

```python
def _get_pictures():  1 usage
    import os
    import requests
    from urllib.parse import urlparse
    # Ensure the images directory exists
    images_dir = "/opt/airflow/data/images"
    pathlib.Path(images_dir).mkdir(parents=True, exist_ok=True)

    # Load data from launches.json
    with open("/opt/airflow/data/launches.json") as f:
        launches = json.load(f)

    image_urls = [launch["image"] for launch in launches["results"] if launch["image"]] * 20

    def is_image_available(image_url):
        try:
            response = requests.head(image_url, timeout=10)

            if response.status_code == 200:
                print(f"Image available at {image_url}")

            return response.status_code == 200 and 'image' in response.headers.get('Content-Type', '')
        except requests.RequestException:
            return False

    def download_image(image_url, target_file):
        try:
            response = requests.get(image_url, stream=True, timeout=30)
            response.raise_for_status()
            with open(target_file, 'wb') as f:
                for chunk in response.iter_content(chunk_size=8192):
                    f.write(chunk)
            print(f"Downloaded {image_url} to {target_file}")
            return True
```

Рисунок 8 – Код созданного DAG

На рисунке 9 показан граф дага

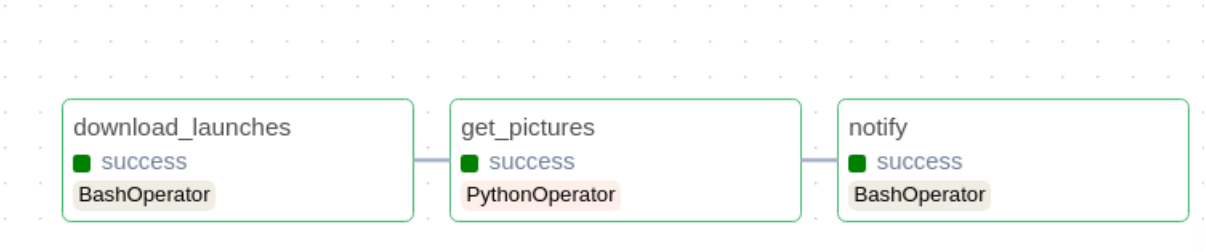


Рисунок 9 – Граф DAG

На рисунке 10 диаграмма Ганта

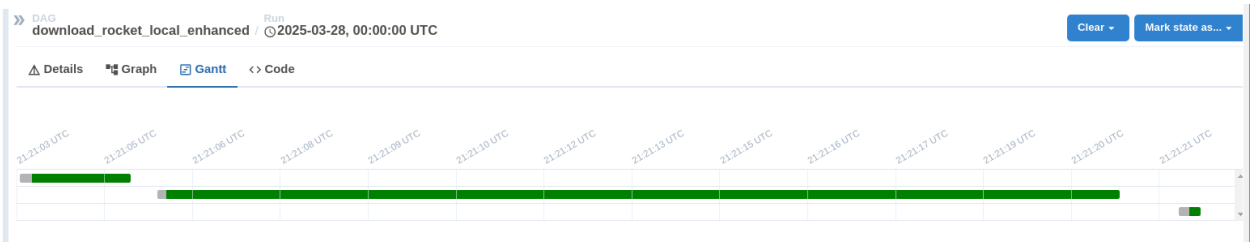
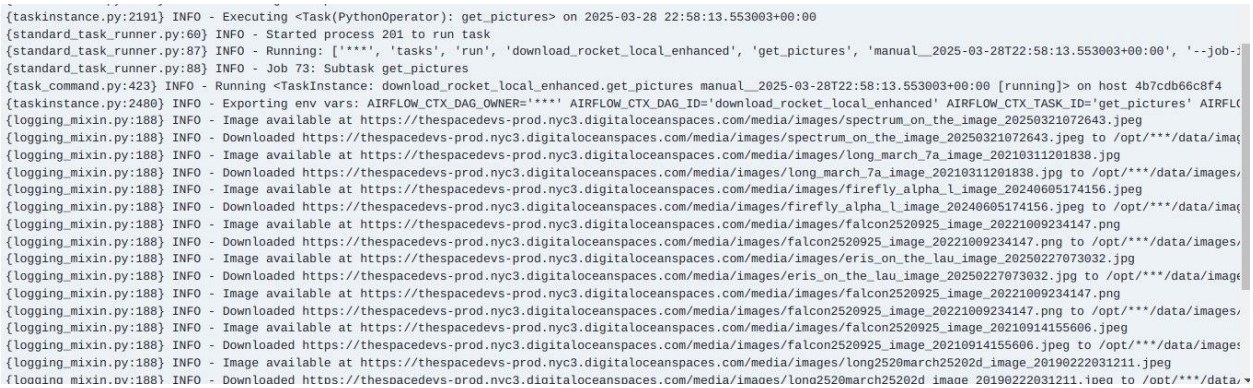


Рисунок 10 – Диаграмма Ганта

1) Разработать метод анализа доступности изображений на сервере по URL

На рисунке 11 показано выполнение первого задания, где реализован метод доступности изображений. Как можно увидеть, сначала идет проверка, что изображения доступны, а только потом скачивание.



```
{taskinstance.py:2191} INFO - Executing <Task(PythonOperator): get_pictures> on 2025-03-28 22:58:13.553003+00:00
{standard_task_runner.py:60} INFO - Started process 201 to run task
{standard_task_runner.py:87} INFO - Running: ['***', 'tasks', 'run', 'download_rocket_local_enhanced', 'get_pictures', 'manual__2025-03-28T22:58:13.553003+00:00', '--job-i
{standard_task_runner.py:88} INFO - Job 73: Subtask get_pictures
{task_command.py:423} INFO - Running <TaskInstance: download_rocket_local_enhanced.get_pictures manual__2025-03-28T22:58:13.553003+00:00 [running]> on host 4b7cdb66c8f4
{taskinstance.py:2480} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='***' AIRFLOW_CTX_DAG_ID='download_rocket_local_enhanced' AIRFLOW_CTX_TASK_ID='get_pictures' AIRFLO
{logging_mixin.py:188} INFO - Image available at https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/spectrum_on_the_image_20250321072643.jpeg
{logging_mixin.py:188} INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/spectrum_on_the_image_20250321072643.jpeg to /opt/***/data/imag
{logging_mixin.py:188} INFO - Image available at https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long_march_7a_image_20210311201838.jpg
{logging_mixin.py:188} INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long_march_7a_image_20210311201838.jpg to /opt/***/data/images/
{logging_mixin.py:188} INFO - Image available at https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/firefly_alpha_l_image_20240605174156.jpeg
{logging_mixin.py:188} INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/firefly_alpha_l_image_20240605174156.jpeg to /opt/***/data/imag
{logging_mixin.py:188} INFO - Image available at https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png
{logging_mixin.py:188} INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png to /opt/***/data/images/
{logging_mixin.py:188} INFO - Image available at https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/eris_on_the_lau_image_20250227073032.jpg
{logging_mixin.py:188} INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/eris_on_the_lau_image_20250227073032.jpg to /opt/***/data/image
{logging_mixin.py:188} INFO - Image available at https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png
{logging_mixin.py:188} INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png to /opt/***/data/images/
{logging_mixin.py:188} INFO - Image available at https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20210914155606.jpeg
{logging_mixin.py:188} INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20210914155606.jpeg to /opt/***/data/images
{logging_mixin.py:188} INFO - Image available at https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long2520march25202d_image_20190222031211.jpeg
{logging_mixin.py:188} INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long2520march25202d_image_20190222031211.jpeg to /opt/***/data/
```

Рисунок 11 – Метод доступности изображений

2) Оценить производительность текущего кода на больших объемах данных

Для проведения производительности были загружено разное количество изображений 10 и 200 штук. На рисунке 12 показана скорость их загрузки в секундах.

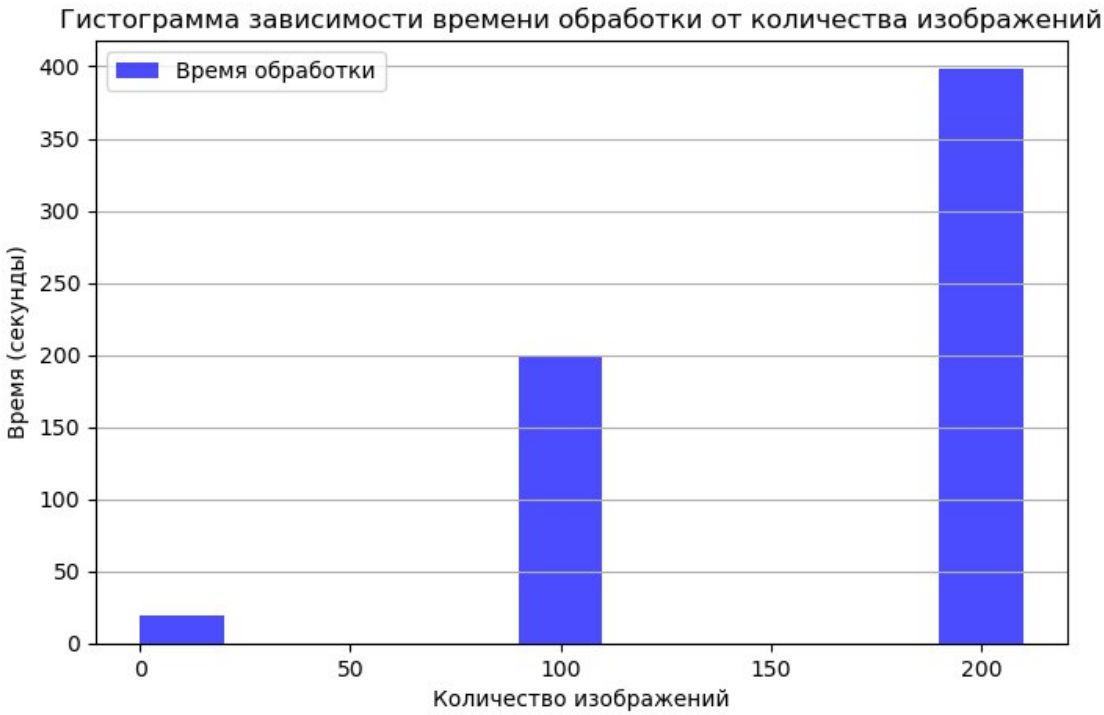


Рисунок 12 – Скорость загрузки изображений

При загрузке 10 изображений, время на скачивание 1 штуки составляет 2.2с, при скачивании 100 – 1.98с, при 200 – 1.99с. Время при различной нагрузке изменяется линейно. При повышенной нагрузке время, затраченное на загрузку одного изображения, не меняется. Объем данных не влияет на производительность. Затраченное время на загрузку примерно одинаковое.

3) Создать план по улучшению скорости скачивания изображений с учетом возможных ошибок

План улучшения скорости скачивания изображений:

1. Асинхронная загрузка:
    - Использовать библиотеки asyncio и aiohttp.
    - Создать асинхронную функцию для скачивания изображений.
    - Запускать несколько задач для одновременной загрузки.
2. Оптимизация размера чанка:
    - Увеличить размер чанка при скачивании, например, до 8192 байт.

3. Проверка Content-Type:
    - Перед загрузкой проверять заголовок Content-Type, чтобы убедиться, что файл является изображением.
4. Обработка ошибок:
    - Добавить обработку исключений requests.RequestException для обработки сетевых ошибок и таймаутов.
5. Проверка доступности перед загрузкой:
    - Использовать requests.head для быстрой проверки доступности ресурса перед загрузкой.

Выводы:

Экономическая эффективность

1. Анализ доступности изображений по URL

Важность: Проверка доступности изображений до их загрузки позволяет избежать ненужных затрат времени и ресурсов.

Экономическая целесообразность:

- Снижение трафика: минимизация количества попыток загрузки недоступных изображений уменьшает затраты на сетевой трафик.
- Повышение надежности: уменьшается вероятность попадания в систему некачественных данных, что улучшает конечный продукт и снижает затраты на их обработку.

2. Оценка производительности кода на больших объемах данных

Экономическая эффективность:

- Увеличение времени обработки: при медленной загрузке изображений требуется больше времени и ресурсов, что может привести к увеличению затрат.
- Риск потери возможностей: длительное время загрузки может привести к потере актуальности данных, что отрицательно сказывается на бизнес-процессах.

3. План улучшения скорости скачивания изображений с учетом возможных ошибок

Экономическая эффективность:

- Повышение скорости загрузки позволяет быстрее обрабатывать данные, что снижает затраты на вычислительные ресурсы.
- Снижение числа затратных повторных попыток загрузки улучшает общую эффективность системы, сокращая время и расходы на процесс в целом.