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 21.03.2025**

**Бизнес-кейс «Umbrella»**

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

Преподаватель: Босенко Т.М.

Москва

2025

**Цель работы:** реализация конвейера ETL для решения бизнес-кейса «Umbrella» с использованием Apache Airflow.

**Задачи:**

- Развернуть ВМ ubuntu\_mgpu.ova в VirtualBox.

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

- Запустить контейнер с кейсом, изучить и описать основные элементы интерфейса Apache Airflow.

- Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес кейс Umbrella в draw.io.

- Результаты работы представить в виде файла ФИО.pdf, выгрузить в учебный портал moodle.

**Ход работы:**

Для работы был запущен контейнер с Airflow на рисунке 1.

Изображение выглядит как текст, Шрифт, число, линия

Контент, сгенерированный ИИ, может содержать ошибки.

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.

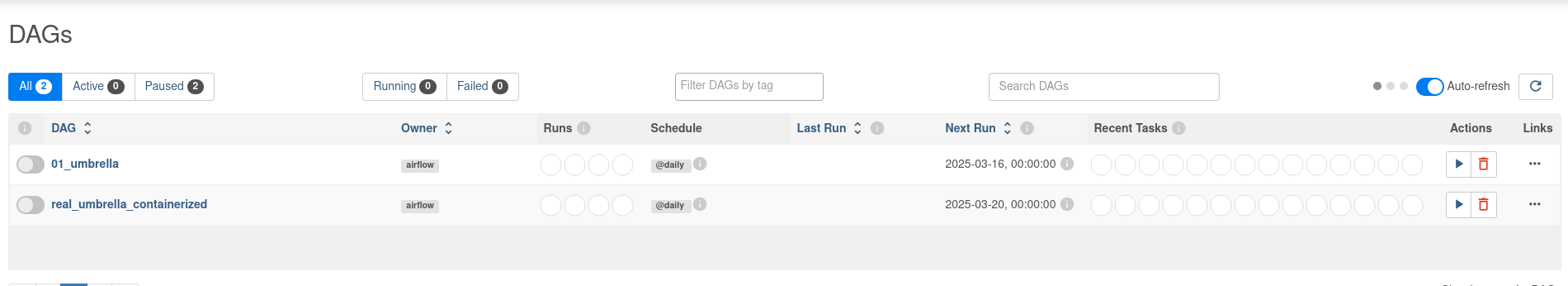


Рисунок 2 – Запуск Airflow

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.). Их можно также с помощью фильтра фильтровать по тегу, статусу, владельцу. Кнопка "Trigger Dag"(изображена стрелкой справа) позволяет вручную запустить выполнение DAG. Кнопка "Refresh"(правее стрелки) обновляет список DAG. На данный момент на экране можно увидеть DAGs с именем 01\_umbrella и real\_umbrella. Всего 2 DAGs и пока не запущены.

Graph View - графическое представление DAG, где задачи отображаются в виде блоков, а зависимости между ними — в виде стрелок. Оно состоит из задач, которые представлены в виде прямоугольника (цвет прямоугольника указывает на статус задачи (например, зеленый — успешно выполнена, красный — ошибка)). Стрелки между задачами показывают порядок выполнения. Вверху страницы есть легенда, которая объясняет, что означают цвета и статусы. Для DAG 01\_umbrella задачами являются fetch\_weather\_forecast, fetch\_sales\_data, clean\_forecast\_data и т. д., соединённые стрелками, как видно на рисунке 3.

Изображение выглядит как текст, снимок экрана, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Графическое представление DAG

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 4.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Древовидное представление DAG

Task Duration - детальная информация о выполнении конкретной задачи. Далее идёт Code, в котором можно просмотреть исходный код DAG на рисунке 5.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 – Исходный код DAG

Task Tries — это механизм, который позволяет задаче повторно выполняться в случае сбоя. Landing Times — это метрика, которая показывает, когда задача фактически завершила выполнение. А также есть диаграмма Ганта, по которой также можно отслеживать выполнение DAG и представлена на рисунке 6.

Изображение выглядит как текст, линия, Шрифт, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – Диаграмма Ганта

Раздел Admin позволяет управлять настройками Airflow. Элементы:

* Connections: Настройка подключений к внешним системам (например, базам данных, API).
* Variables: Управление переменными, которые могут использоваться в DAG.
* Pools: Управление пулами ресурсов для выполнения задач.
* Users: Управление пользователями и их правами (если включен RBAC — Role-Based Access Control).

Browse - раздел для просмотра различных данных и метаданных. Docs - встроенная документация Airflow.

Сам DAG состоит из нескольких пустых операторов (DummyOperator) и иллюстрирует процесс ETL. Данные извлекаются, очищаются, объединяются, происходит обучение модели и ее развёртывание.

В рамках **12 варианта** необходимо было получить прогноз в Амстердаме на 7 дней, отфильтровать дни с температурой <0°C и посчитать среднюю температуру. Для этого для начала необходимо было пройти регистрацию на сайте weather.api и получить уникальный api, как видно на рисунке 7.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7 – Регистрация на сайте weather.api

Затем необходимо было внести изменения в DAG real\_umbrella.py, иллюстрирующий вариант использования бизнес кейса "Umbrella", который автоматизирует весь pipeline обработки погодных и продажных данных, обучения модели машинного обучения и её "развёртывания", а именно в части fetch\_weather\_forecast(), где получаются данные о погоде. Нужно зафиксировать город Амстердам и 7 дней, как показано на рисунке 8.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8 – Получение данных о погоде за 7 дней из Амстердама

Также были немного изменены значения в продажах. Теперь можно запускать DAG на рисунке 9, который выполнился успешно.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 9 – Запуск DAG real\_umbrella

На рисунке 10 видно, что все нужные нам файлы успешно загрузились.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 10 – Загруженные файлы

Далее было изучено содержимое каждого файла на рисунках 11–15.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11 – Очищенные данные о продажах

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12 – Очищенные данные о погоде

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 13 – Объединённые данные

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 – Сгенерированные данные о продажах

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 15 – Выгруженные данные о погоде

Стояла задача фильтрации данных, которые меньше 0 градусов, тут все температуры положительные, поэтому были отредактированы данные продаж. но код фильтрации все равно будет составлен. Модель была выгружена на рисунке 16.

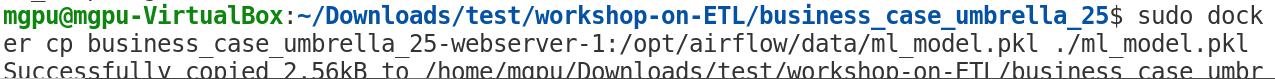


Рисунок 16 – Выгруженная модель

Фильтрация и вычисление средней температуры показаны на рисунке 17, а также пример предсказания продаж при 13 градусах на 18 рисунке.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 13 – Фильтрация и вычисление средней температуры

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 – Пример предсказания

Данные берутся из API погодных данных (WeatherAPI, который предоставляет доступ к информации о текущей погоде, прогнозах, исторических данных), а также синтетические данные из таблицы. Для хранения данных был выбран PostgreSQL, он отлично подходит для обработки транзакционных задач, таких как оплаты и обработка заказов, может хранить JSONB, что позволяет хранить полуструктурированные данные и использовать их в продажах (например, детальную информацию о покупке). MLflow Model Registry позволяет хранить модели машинного обучения и была предложена как возможный вариант в архитектуре. При анализе и обработке данных используется Python. Готовая архитектура представлена на рисунке 15.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 15 – Архитектура для бизнеса по продаже зонтов

**Выводы:**

В рамках работы были изучены основные элементы Apache Airflow. Разработана архитектура аналитического решения, включающая слои источников данных, их обработки, хранения и доступа для бизнес-пользователей, а также при просмотре дага была выявлена функция, отвечающая за машинное обучение, поэтому был добавлен и этот слой и зафиксирован Airflow в рамках отдельного слоя. Данная архитектура демонстрирует, что данные берутся о погоде и продажах, анализируются и обучаются модели на Python, хранятся не только сами данные в базе, но и версионность моделей машинного обучения. Был выполнен прогноз погоды на 7 дней для Амстердама и анализ его результатов. Как видно, погоды с минусовыми температурами нет, т. к. в Амтсердаме тепло и сейчас весна, так что это нормально. Цель и задачи были выполнены.