МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики
Кафедра киберфотоники

Отчет по лабораторной работе №1

Дисциплина: «Инженерия данных»

Выполнила: Цибикина А.И.

Группа: 6233-010402D

Самара 2025

Архитектура:

Схема пайплайна:

Prefect (оркестратор) → Open-Meteo API (источник) → MinIO (сырые данные), → ClickHouse (агрегированные таблицы) → Telegram Bot (уведомления).

Какие инструменты и почему:

- Prefect 2.14.21 — для оркестрации, управления зависимостями и расписания. Выбор стабильной версии 2.x обусловлен совместимостью с зависимостями и простотой настройки.
- MinIO — S3-совместимое объектное хранилище для сырых JSON-ответов от API.
- ClickHouse — высокопроизводительная колоночная СУБД.
- Telegram Bot API — канал для доставки push-уведомлений. Прост в интеграции

Источник данных:

Бесплатный API Open-Meteo.

Эндпоинт: https://api.open-meteo.com/v1/forecast

Параметры запроса:

- latitude, longitude — точные координаты Москвы и Самары, жестко заданные в коде.
- hourly=temperature_2m,precipitation,windspeed_10m,winddirection_10m — набор необходимых почасовых метрик.
- daily=temperature_2m_max,temperature_2m_min,precipitation_sum,windspeed_10m_max — набор дневных агрегатов для расчета сводки.
- timezone=Europe/Moscow — для получения времени в нужном часовом поясе.
- start_date, end_date — устанавливаются на дату "завтра", чтобы запросить прогноз строго за один день.

Extract → Transform → Load:

Extract: Пайплайн запускается по расписанию, после чего асинхронно отправляется HTTP-запросы к Open-Meteo API для каждого города. Сырые JSON-ответы сохраняются в MinIO в структурированные папки по городу и дате.

Transform: Почасовые данные из JSON разворачиваются в плоскую структуру для таблицы weather_hourly. Параллельно рассчитываются дневные агрегаты (min/max/avg температура, сумма осадков) для таблицы weather_daily.

Load: Преобразованные данные пакетом загружаются в таблицы ClickHouse. В параллельной задаче формируется и отправляется текстовое уведомление в Telegram с прогнозом и алертами на основе дневных данных.

Качество данных:

Проверки:

- Валидация HTTP-статуса ответа API через response.raise_for_status().
- Механизм повторных попыток (retries=3) при извлечении данных.
- Неявная проверка структуры данных при трансформации.

Точки сбоя:

- Недоступность или изменение формата ответа у Open-Meteo API.
- Недоступность сервисов MinIO или ClickHouse из-за проблем с сетью или конфигурацией Docker.
- Отсутствие или некорректность секретов для Telegram (токен, chat_id).

Результаты работы пайплайна:

1. Содержимое в ClickHouse

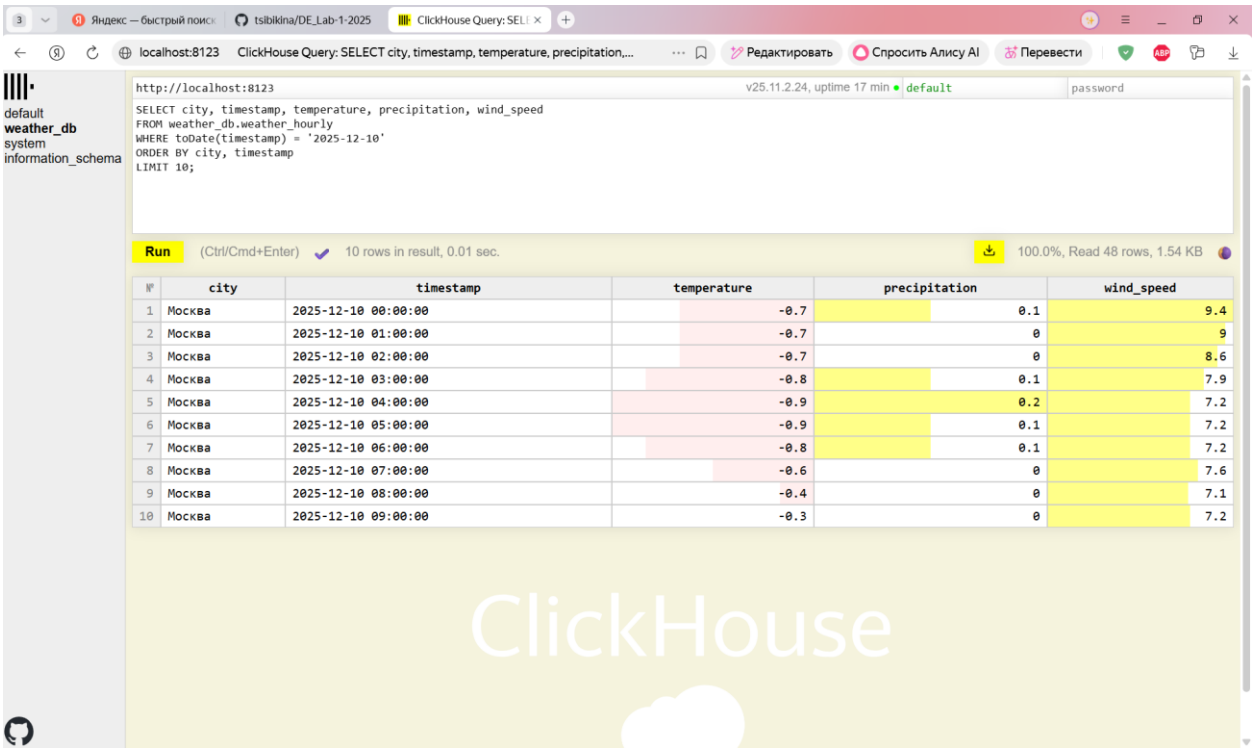

Рисунок 1 - Содержимое запроса для таблицы weather_hourly

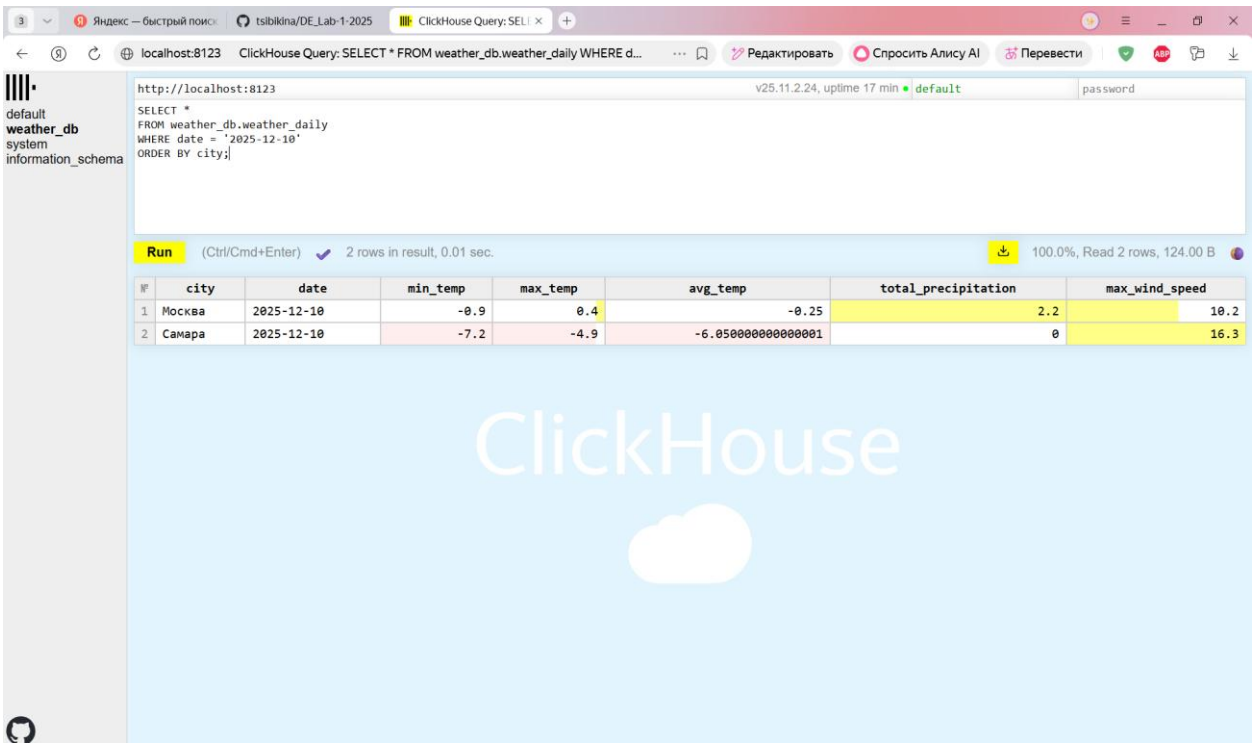

Рисунок 2 - Содержимое для запроса таблицы weather_daily

2. Содержимое бакета в MinIO

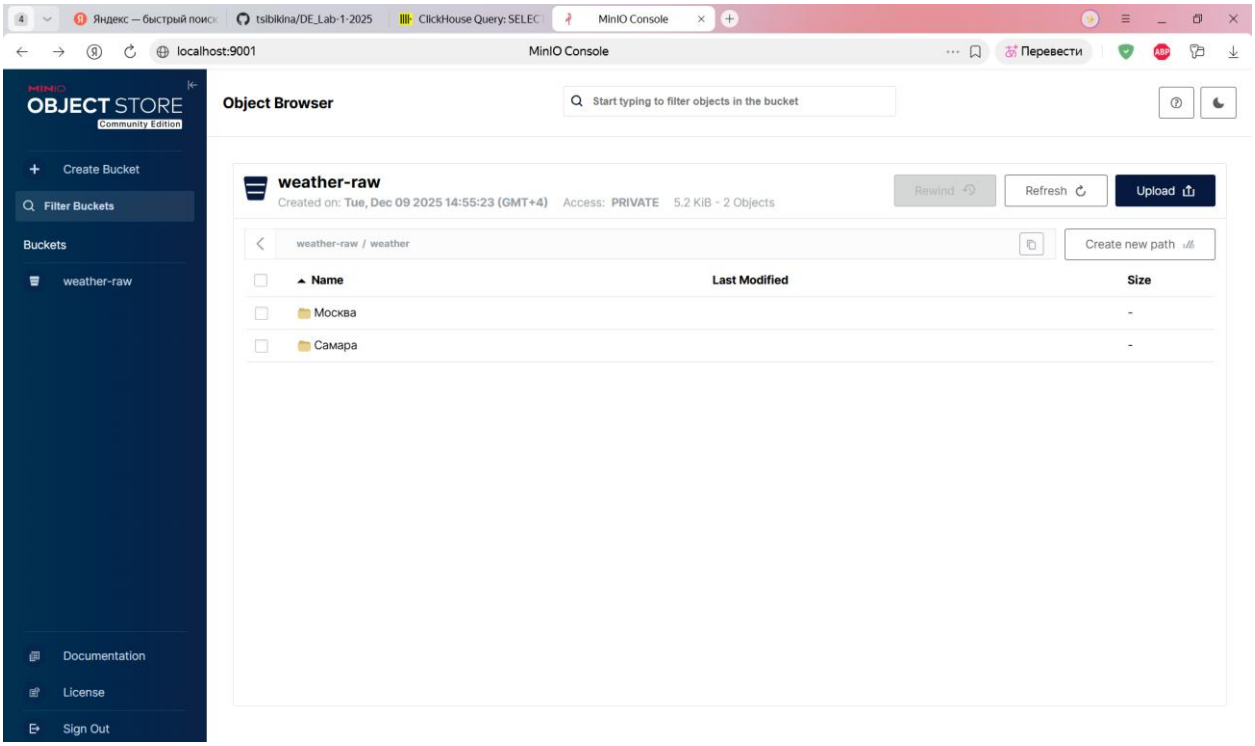

Рисунок 3 - Работа в MinIO

3. Perfect логи

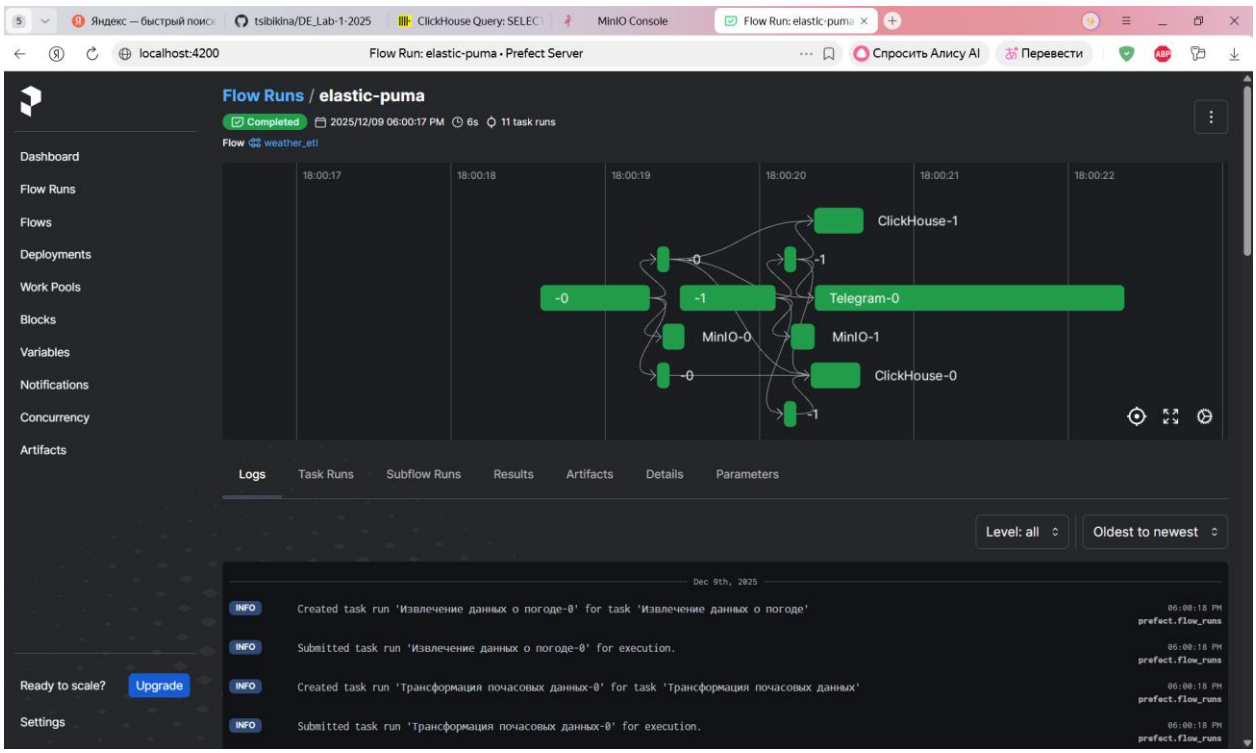

Рисунок 4 - Лог работы в Perfect

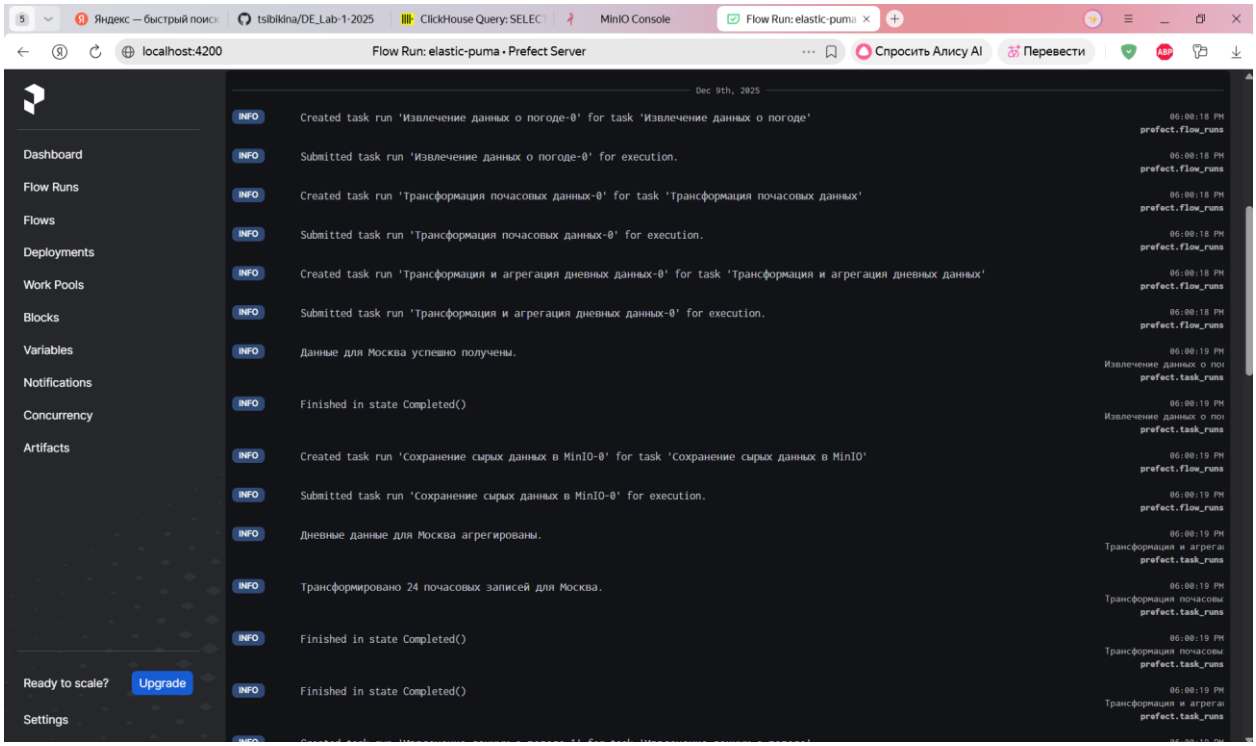

Рисунок 5 - Лог работы в Perfect

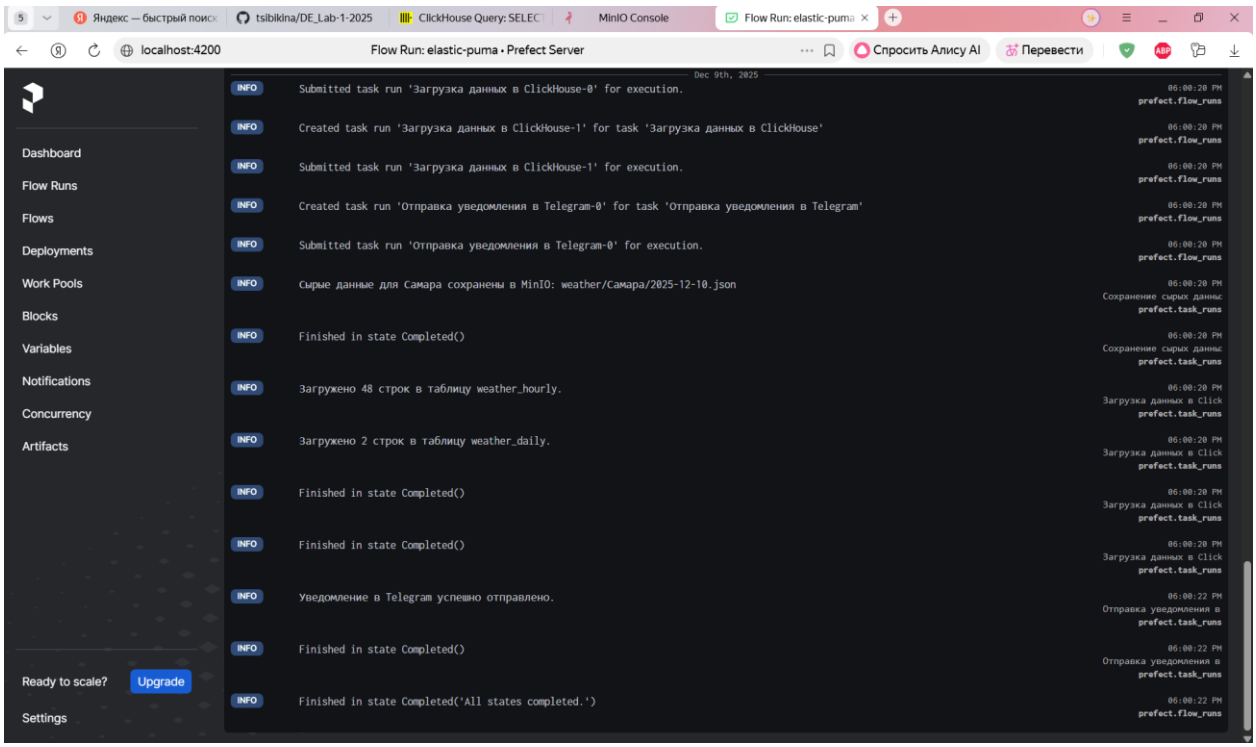

Рисунок 6- Лог работы в Perfect

4. Результат работы уведомления в Telegram

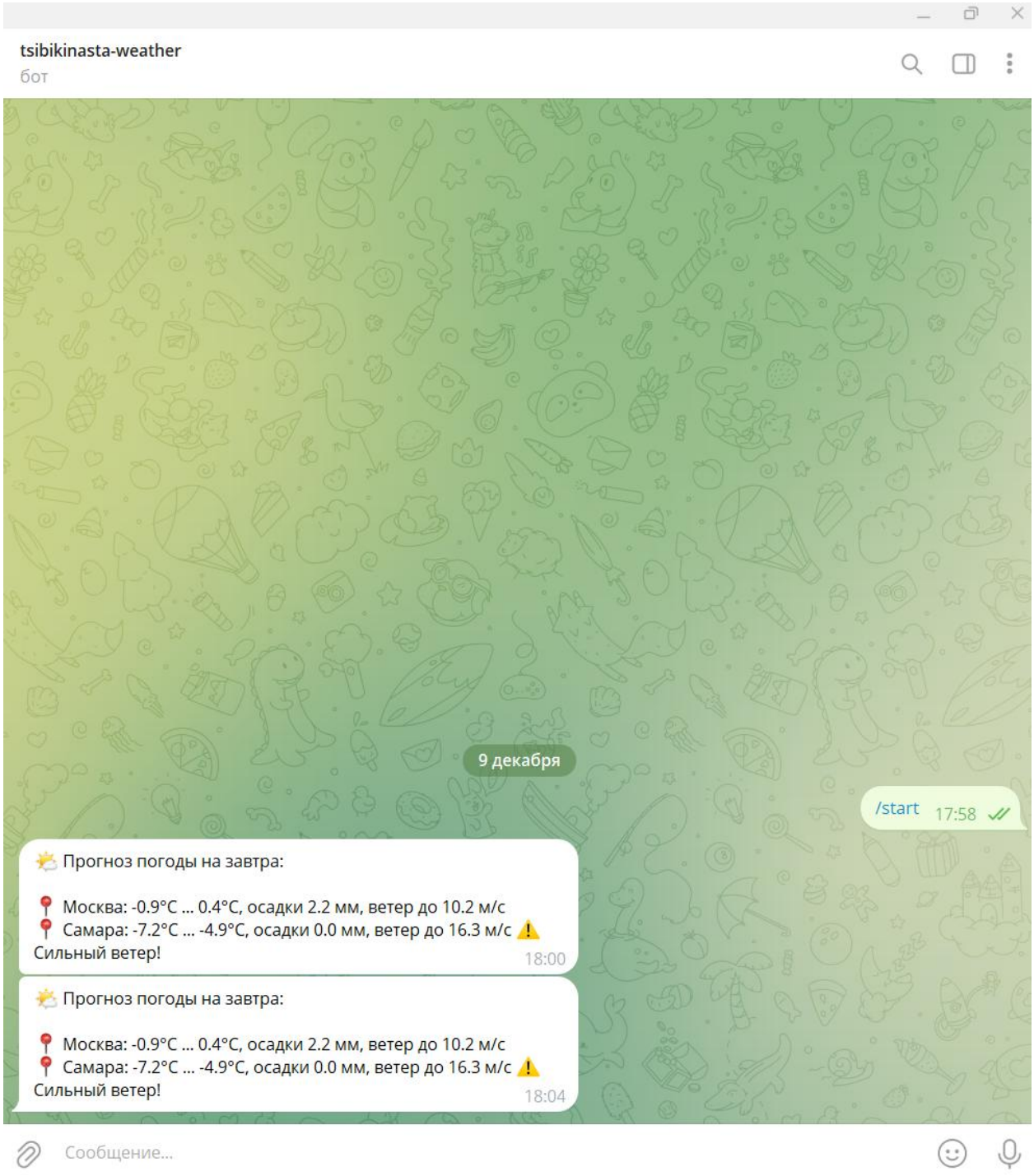

Рисунок 7 - Уведомления в Telegram

Вывод по работе и её дальнейшие улучшения.

Сложности возникли с настройкой и отладкой всего окружения, а не самой логикой ETL-процесса. Из ключевых трудностей - это конфликты версий библиотек и настройка уведомлений в telegram.

Можно улучшить программу добавив виртуальное окружения, чтобы убрать проблему с конфликтами библиотек и может сделать более детальную обработку ошибок.