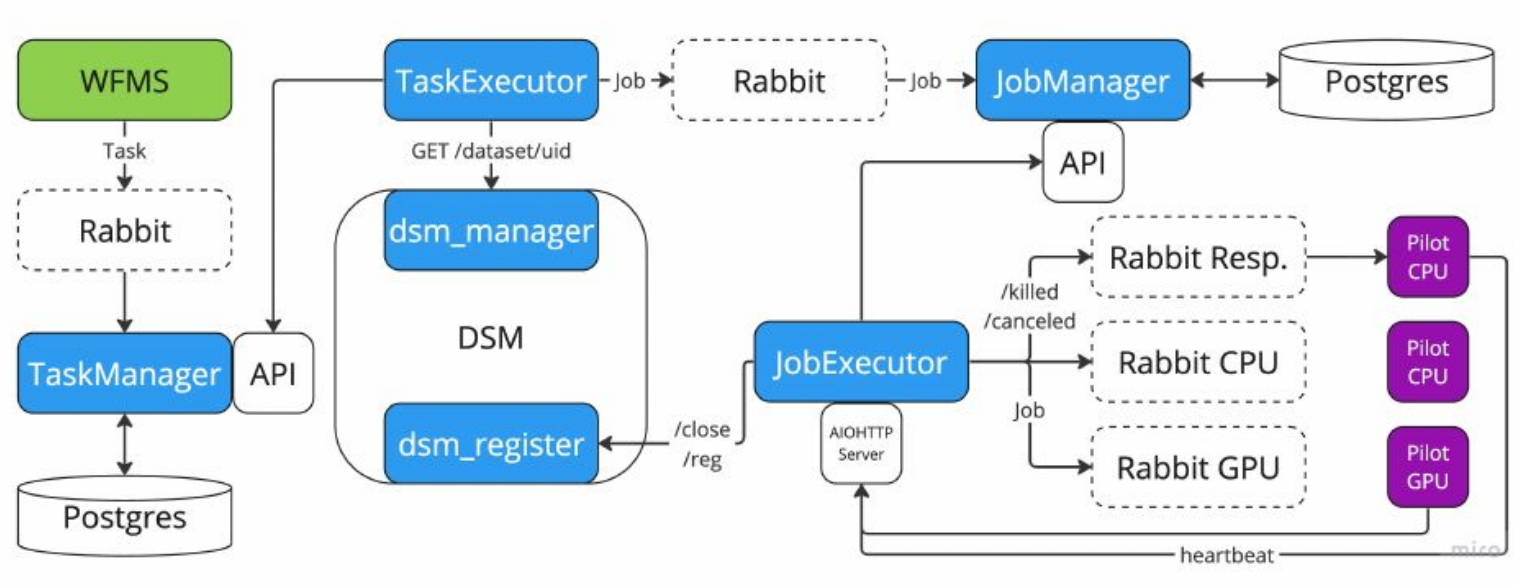
Workload Management System

Планы на ближайшее будущее

Общая архитектура

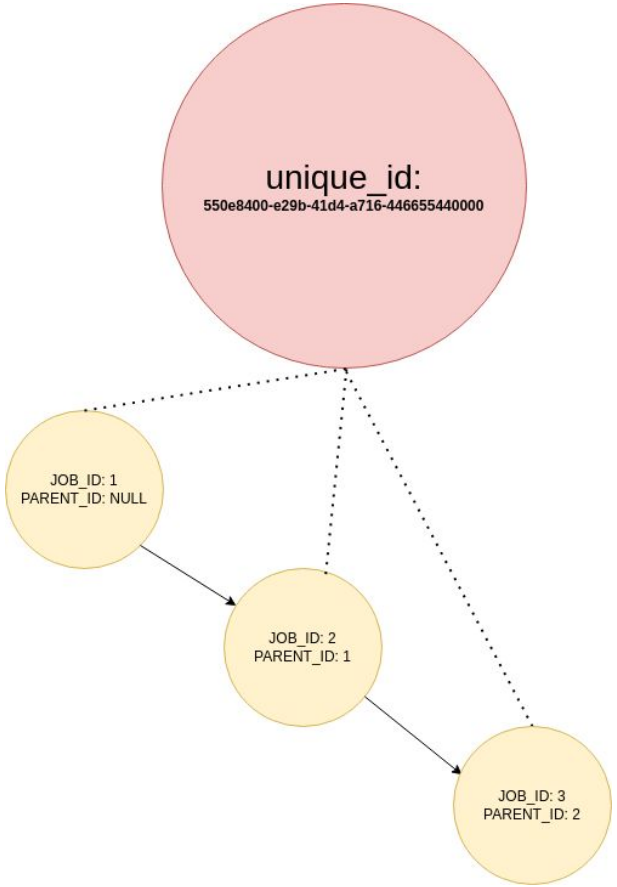


Текущий статус WMS

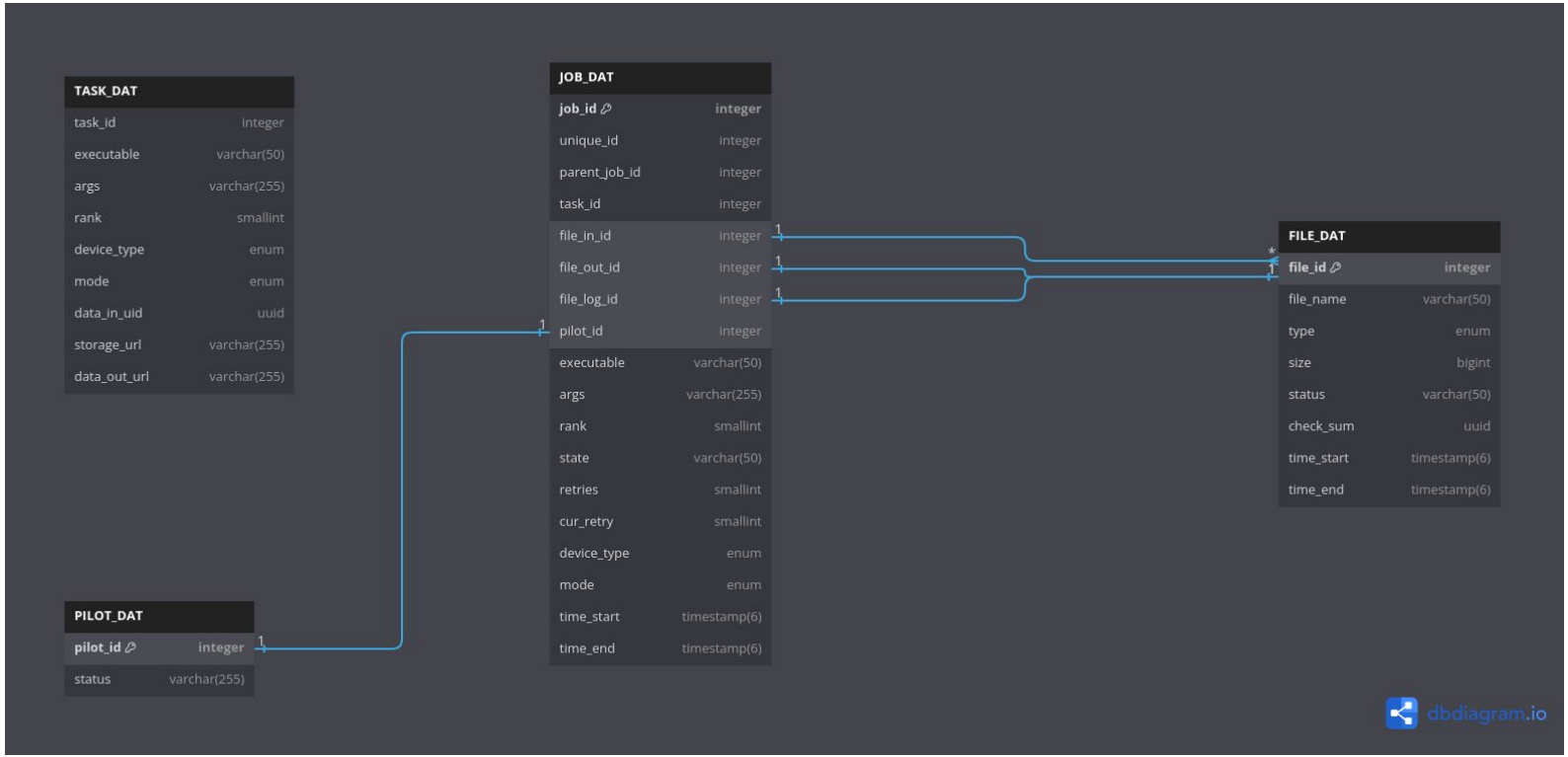
- Ключевые компоненты REST API сервиса для job-manager’а и task-manager’а реализованы.
- Получение ‘метрик’ с пилота и обновление статуса задач.
- Набор пачки задач job-executor’ом и их передача в очередь (пока одну) на пилот (первая реализация job-executor).
- *Unit тесты методов записи/чтения с БД*
- *Введена тестовая СУБД в отдельном контейнере для прогона тестов и миграций*
- *Выполнен переход на poetry*

Отношение задач

- Необходимо прописать исключения и статусы задач на этом этапе
- Логи задач?

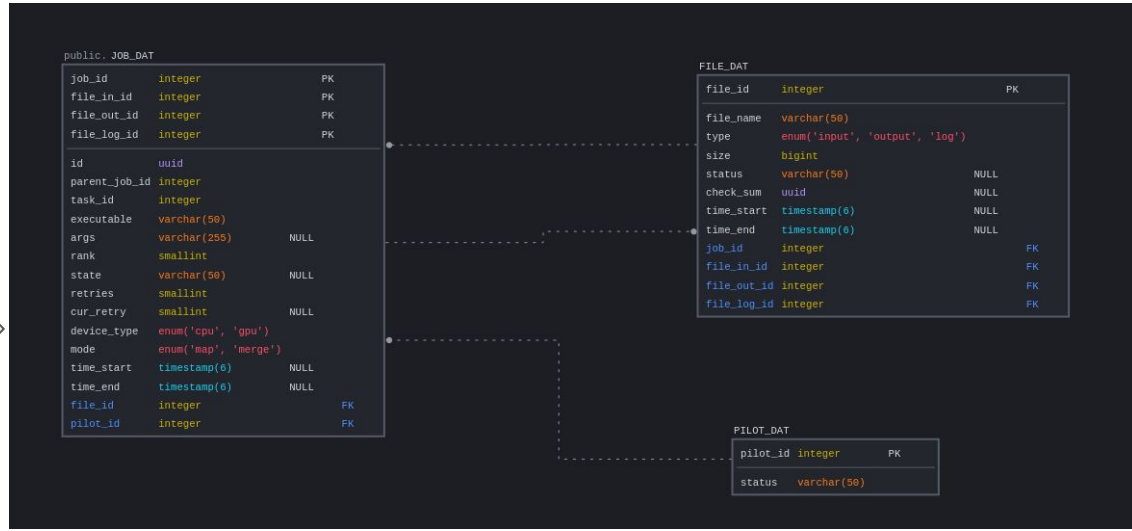


Схема

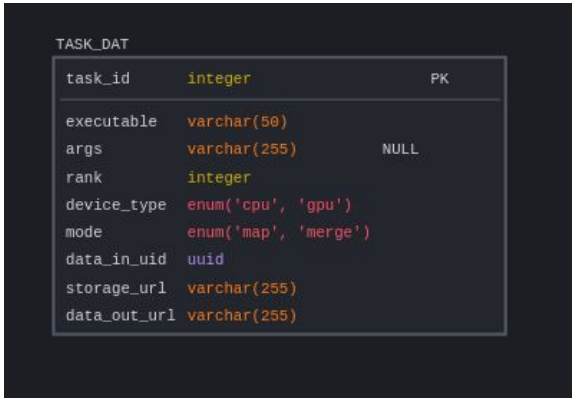

TASK_DAT
task_id integer
executable varchar(50)
args varchar(255)
rank smallint
device_type enum
mode enum
data_in_uid uuid
storage_url varchar(255)
data_out_url varchar(255)
JOB_DAT
job_id integer
unique_id integer
parent_job_id integer
task_id integer
file_in_id integer
file_out_id integer
file_log_id integer
pilot_id integer
executable varchar(50)
args varchar(255)
rank smallint
state varchar(50)
retries smallint
cur_retry smallint
device_type enum
mode enum
time_start timestamp(6)
time_end timestamp(6)
FILE_DAT
file_id integer
file_name varchar(50)
type enum
size bigint
status varchar(50)
check_sum uuid
time_start timestamp(6)
time_end timestamp(6)
PILOT_DAT
pilot_id integer
status varchar(255)
dbdiagram.io

Текущая схема

Сервер Postgres#1
job-manager



Сервер Postgres#2:
task-manager



Milestone №1 ~ 1-2 недели

- Завершить введение логирования (logging, structlog, colorlog).

```
app.main: 2024-01-25 08:51:33,645 | INFO | main.py:41 | 10 >>> FastAPI initialization
INFO:app.main:FastAPI initialization
app.main: 2024-01-25 08:51:33,645 | INFO | main.py:44 | 10 >>> Adding routes
```

- Новая схема базы данных (**JOB_DAT**, **FILE_DAT**, **TASK_DAT**, **PILOT_DAT**).
- Собственный классы-исключения.
- task-executor/spitter - активация/деактивация .

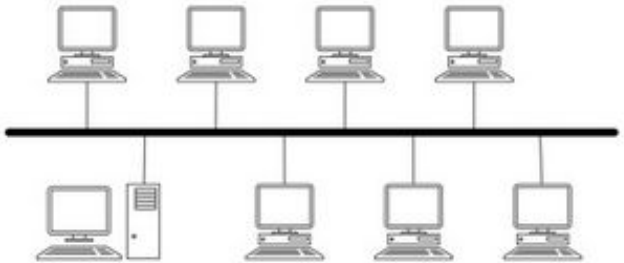
Milestone №2 (обработка одного датасета) ~ 2-4 недели

1) Управление шиной событий с помощью пилота.

2) Регистрация наборов данных в DSM (просмотреть формат в swagger).

3) Task-Executor (каркас) - как это будет выглядеть (описать, может быть, не слишком беспокоясь об алгоритме).

4) Способы обработки сообщений RabbitMQ (прямая операция записи в RDBMS / другие проблемы и т. п.).



Milestone №3 (обработка серии датасетов) ~ 2-3 недели

- Предварительная регистрация нескольких наборов данных в DSM.
- Примитивный спиттер с заданиями в очередь для дальнейшей обработки Task-Executor.

Планы в ближайшее будущее на пилота

1. Развертывание второго пилота (config???) для отладки очереди с командами и взаимодействий со второй очередью (GPU/CPU).
2. Возможность обработки серии задач, находящихся в очереди (отладка демона).

Итого
Ожидаемо (~ 5-9 недель)

1) Генерация заданий.

2) Регистрация файлов.

3) Простая логика обработки закрытия набора данных.

Конечная цель по окончанию Milestone №3, идеально:

Отмена заданий/задач (цепочка обработки распространяется на все 4 системы и все 4 микросервиса **WMS**)