



ИС Метаданных эксперимента SPD

Концепции, планы и разработка

Выполнил: Шкерин Н.В.,
Государственный университет "Дубна"
Руководители:
Прокошин Ф.В.,
Лаборатория ядерных проблем им. В. П. Джелепова
Объединенный институт ядерных исследований
Дорохин В.А.,
Государственный университет "Дубна"

16.04.2024

Ускорительный комплекс NICA

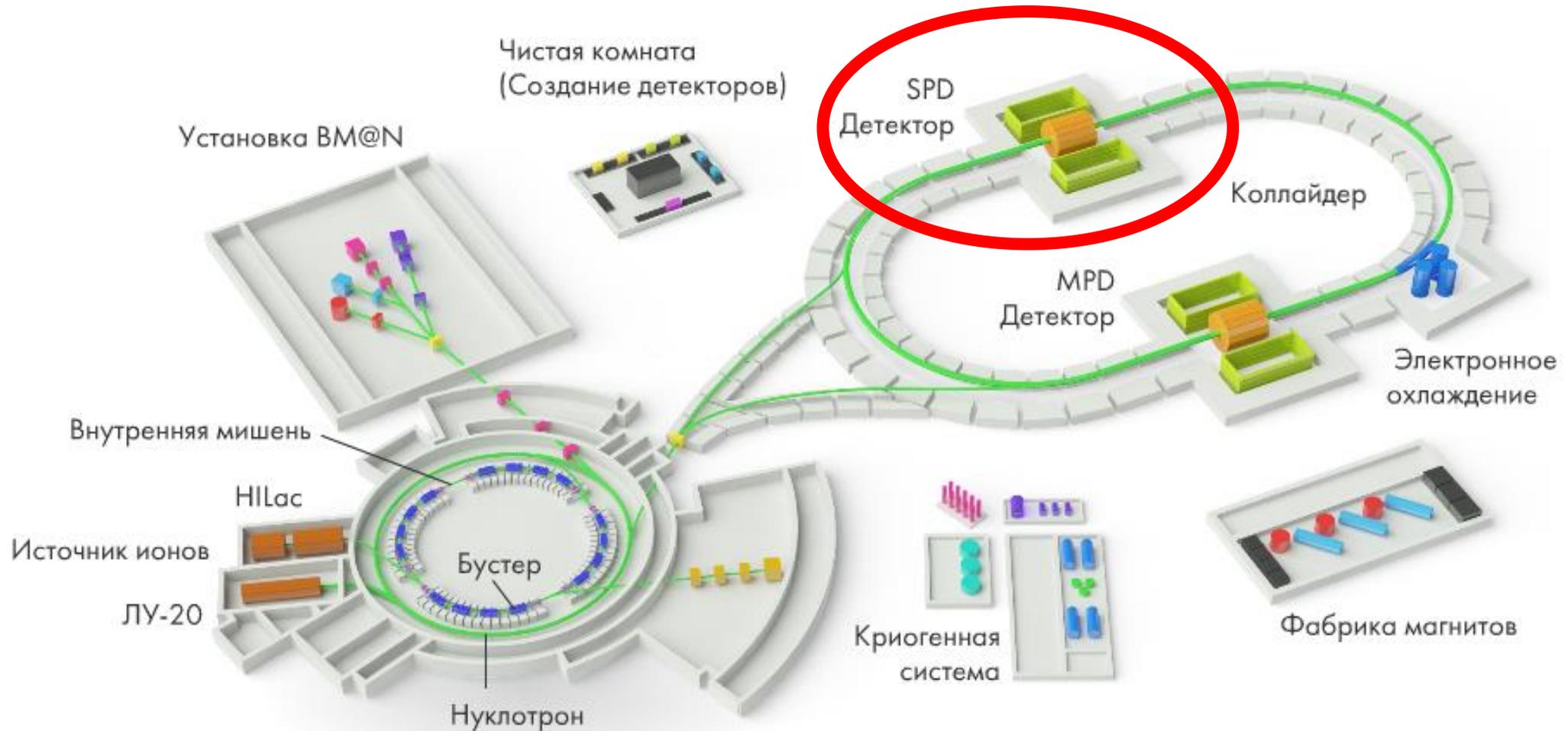


Схема ускорительного комплекса NICA

Установка SPD как источник данных

- Частота событий ~ 3 МГц
- При проектной светимости:
 - Поток данных 20 ГБ/с
 - 200 ПБ/год (сырые данные)
 - $2 \cdot 10^{13}$ событий в год
- Данные столкновений
- Моделированные данные
- Вспомогательная информация
- Данные о конфигурации и состоянии детектора

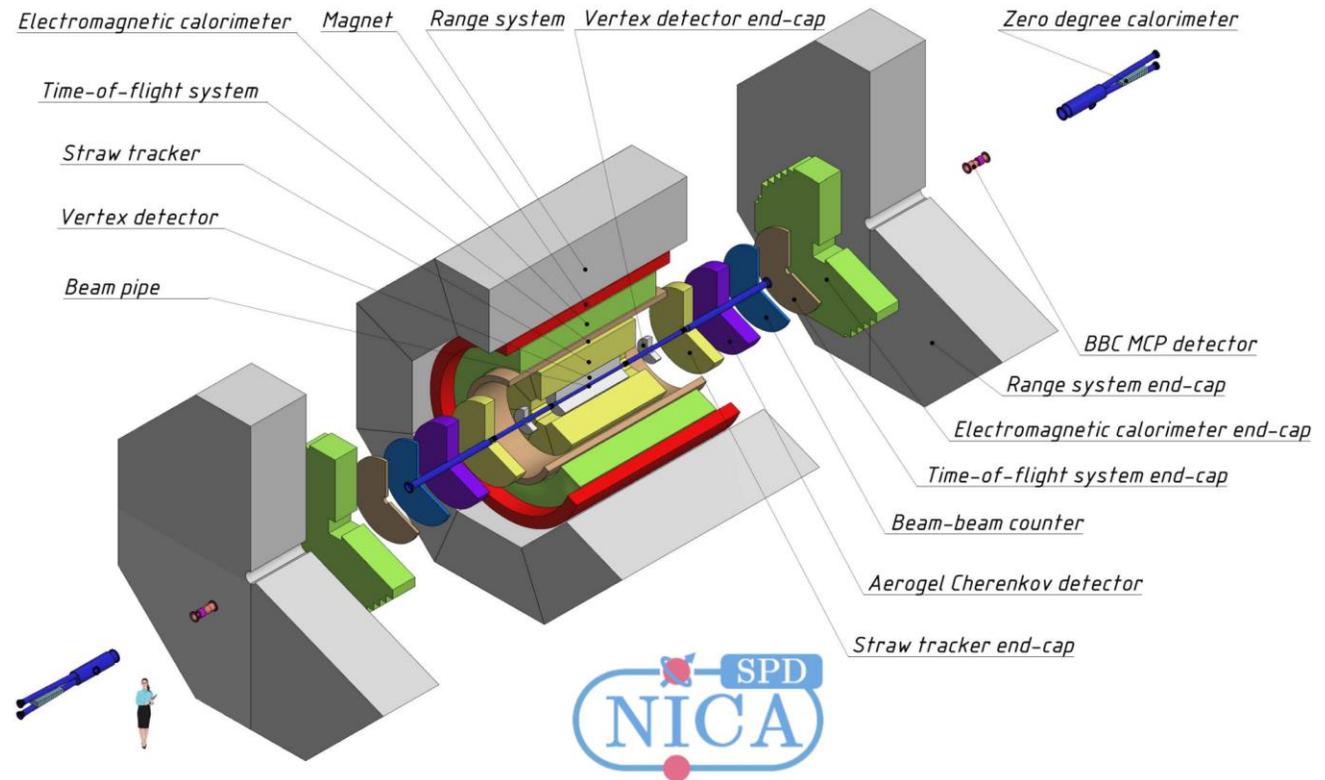


Схема детектора SPD

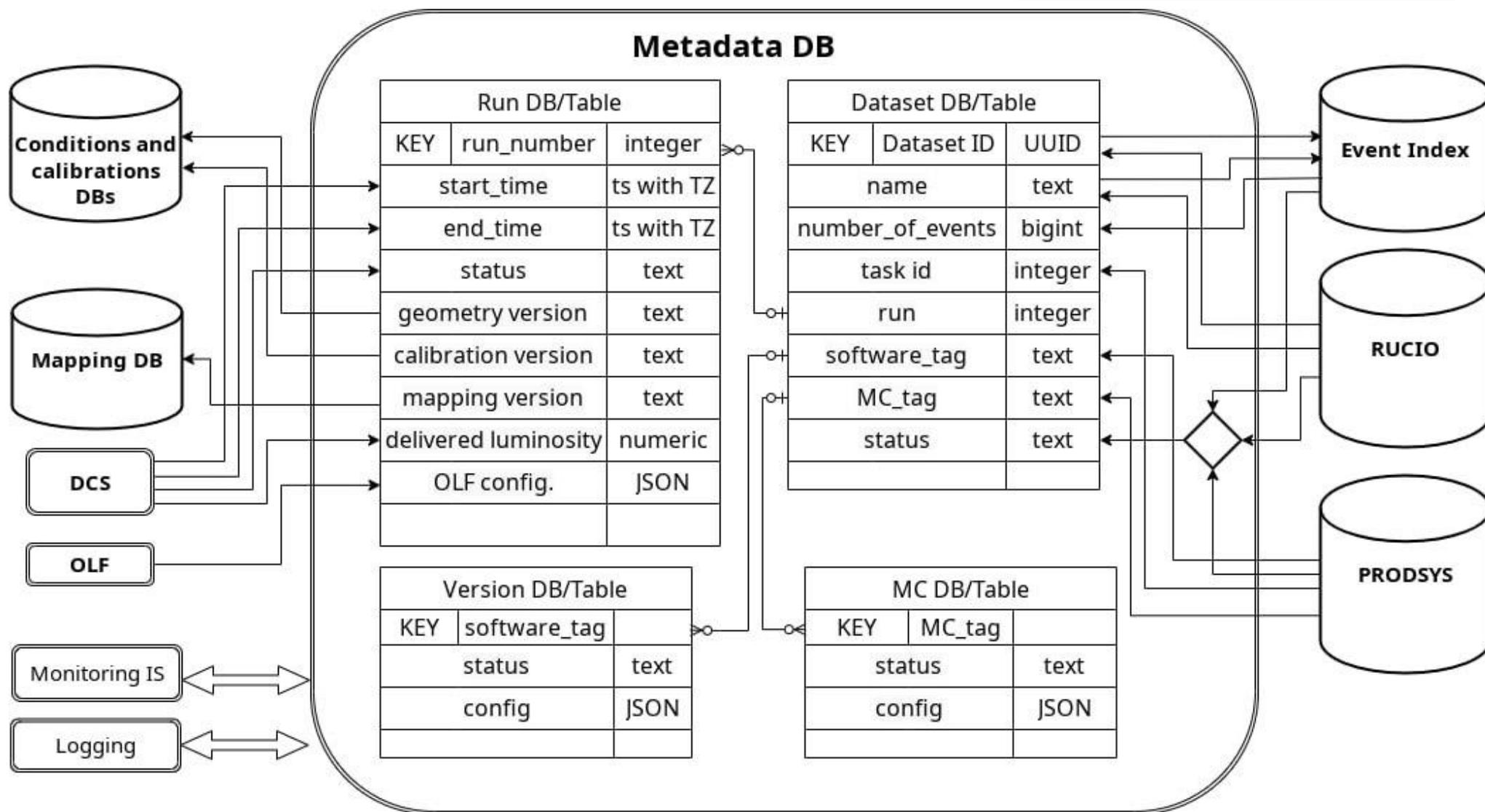
Описание проблемной ситуации

Данные в SPD хранятся в различных форматах и в разных системах: распределенные файловые хранилища, мета-информация, базы данных, текстовые файлы на устройствах, etc. Часто объемы этих данных велики. В то же время, для эффективного управления набором, восстановлением и анализом данных часто необходима информация из множества гетерогенных источников.

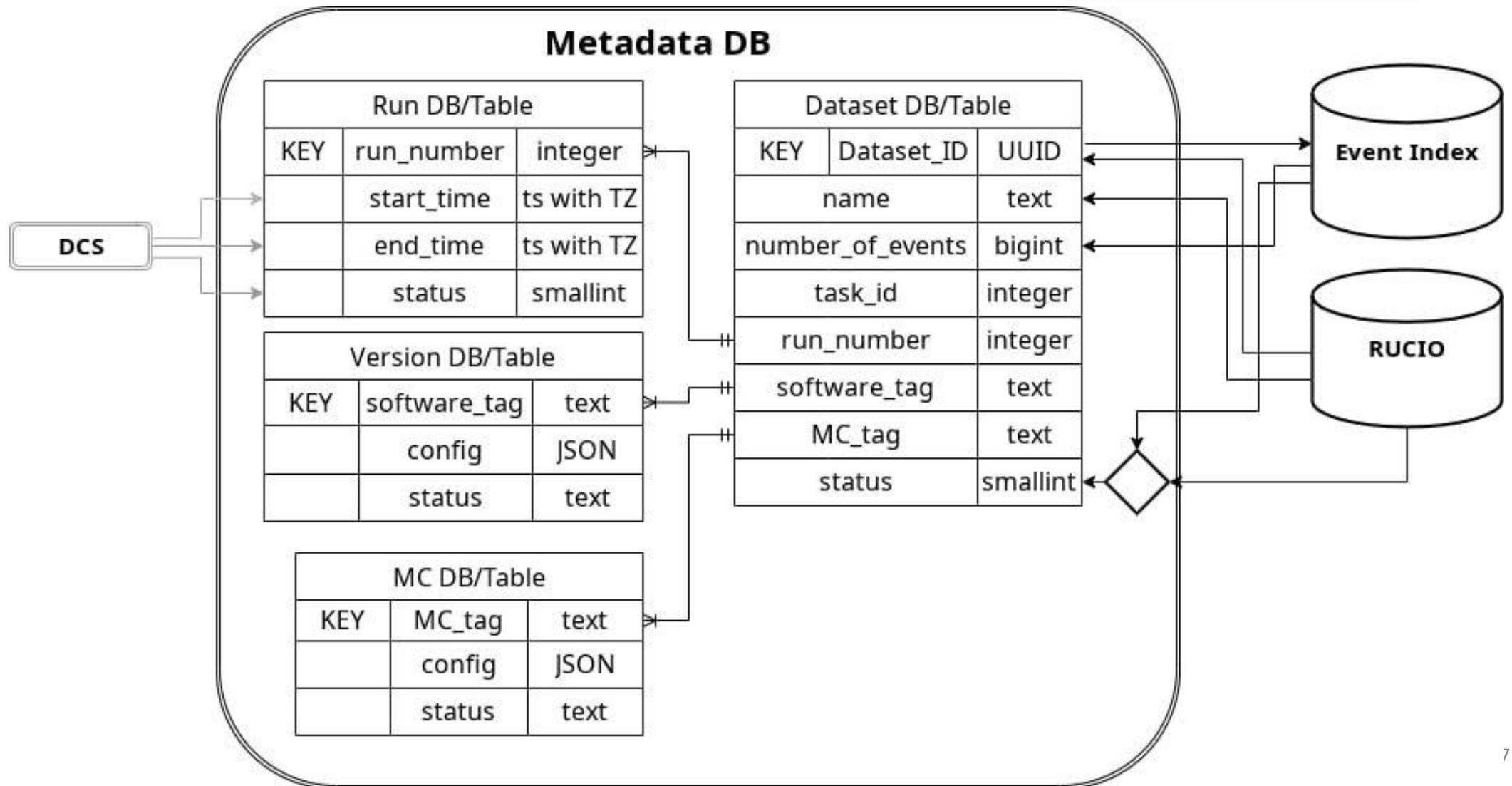
Постановка задачи

1. Изучение проектируемых информационных систем эксперимента SPD, модели данных, а также реализации аналогичных систем в других экспериментах.
2. Разработка схемы хранения информации в БД
3. Выбор набора программных средств и разработка API для обмена данными с пользователями и другими ИС.
4. Разработка средств получения метаданных, их транспортировки и записи в БД. Разработка интерфейсных приложений.
5. Создание веб интерфейса.

Концептуальная схема ИС Метаданных

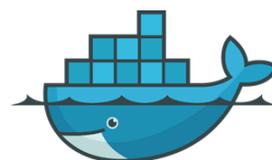


Прототип ИС Метаданных



Используемые технологии

- FastAPI
- Pydantic
- PyTest + Locust
- Sphinx python documentation
- Gitlab
- Docker
- SQLAlchemy + alembic + asyncpg + PostgreSQL
- InfluxDB + Grafana
- Redis



docker



Microservice architecture

Microservice

1

API

2

Services

3

Data Base

Microservice architecture: API

Pydantic schemas validate

URI Versioning

Валидация данных от клиента
Валидация данных на уровне сервиса
Валидация данных из БД



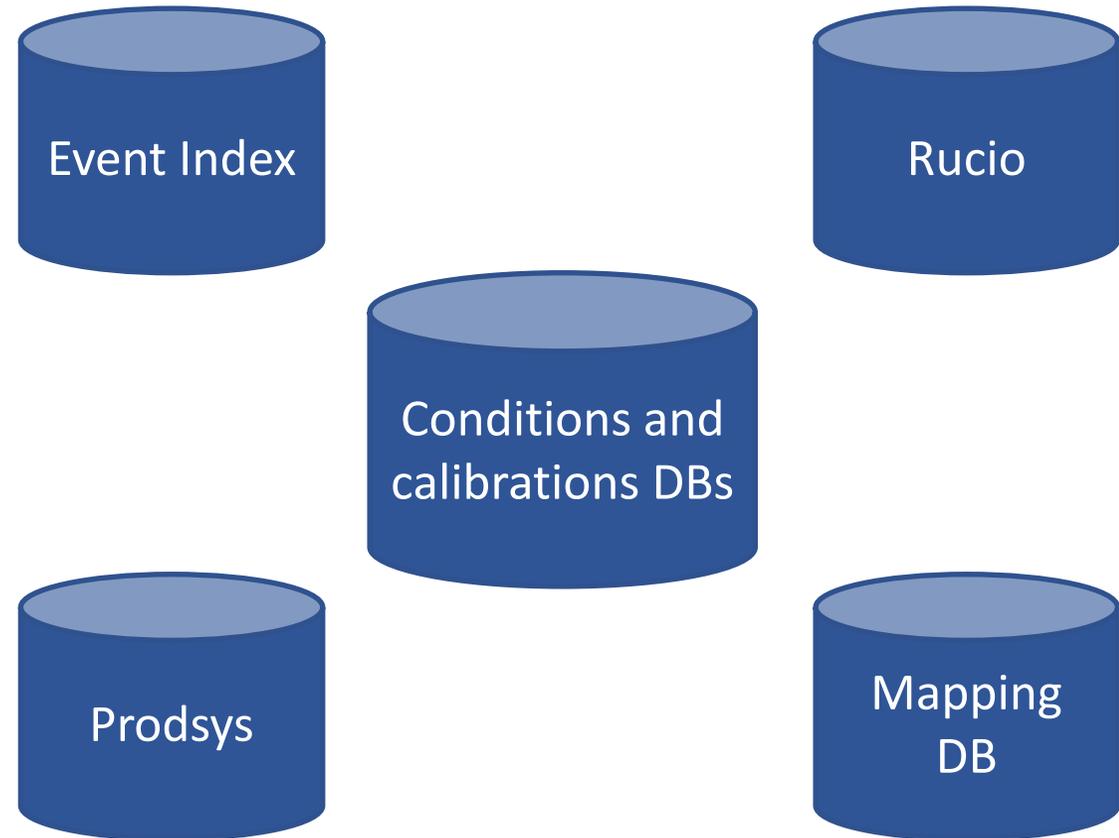
<https://127.0.0.1:8000/v1/testExample/get/2>

<https://127.0.0.1:8000/v2/testExample/get/2>



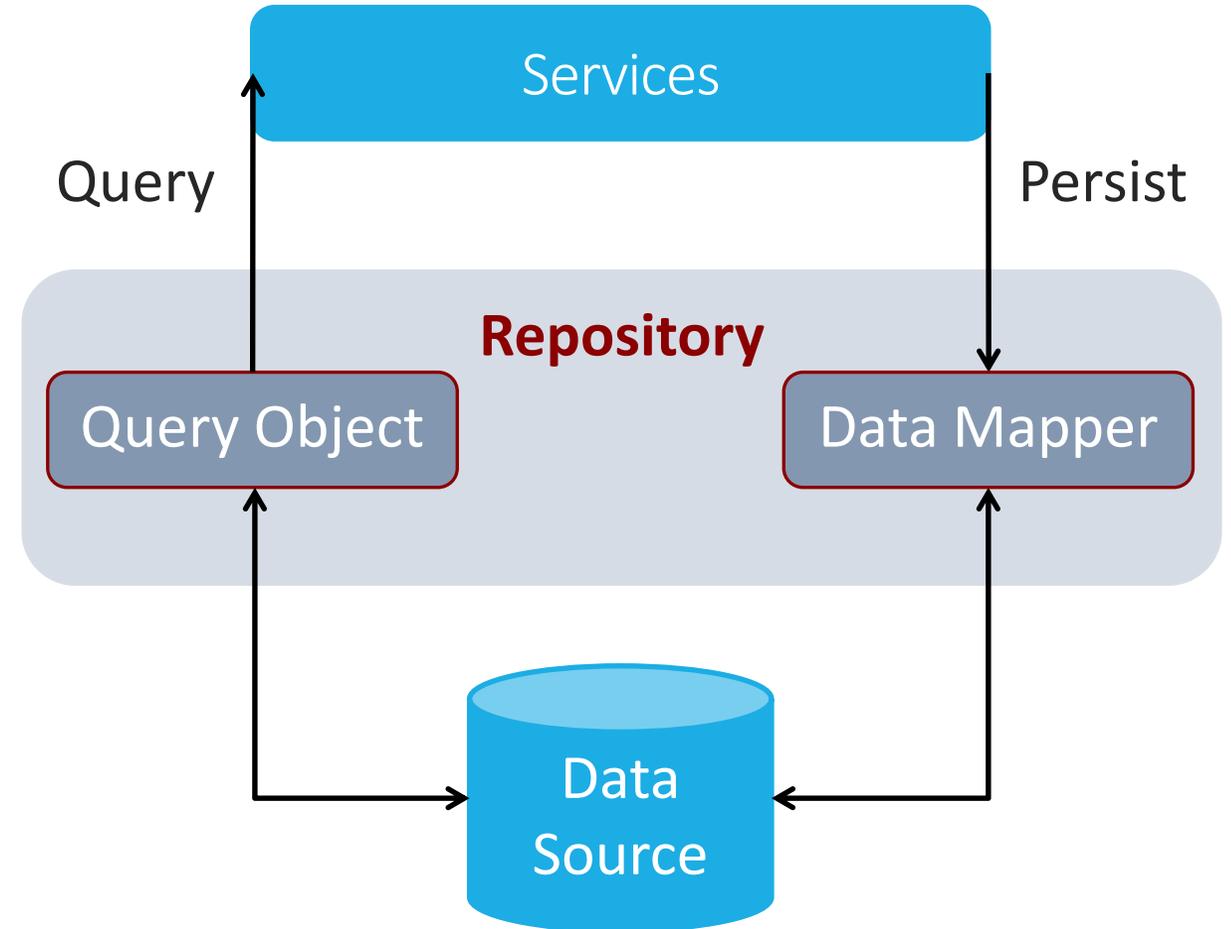
Microservice architecture: Services

- Запросы ко внешним системам
- Обработка данных
- Любая иная бизнес-логика

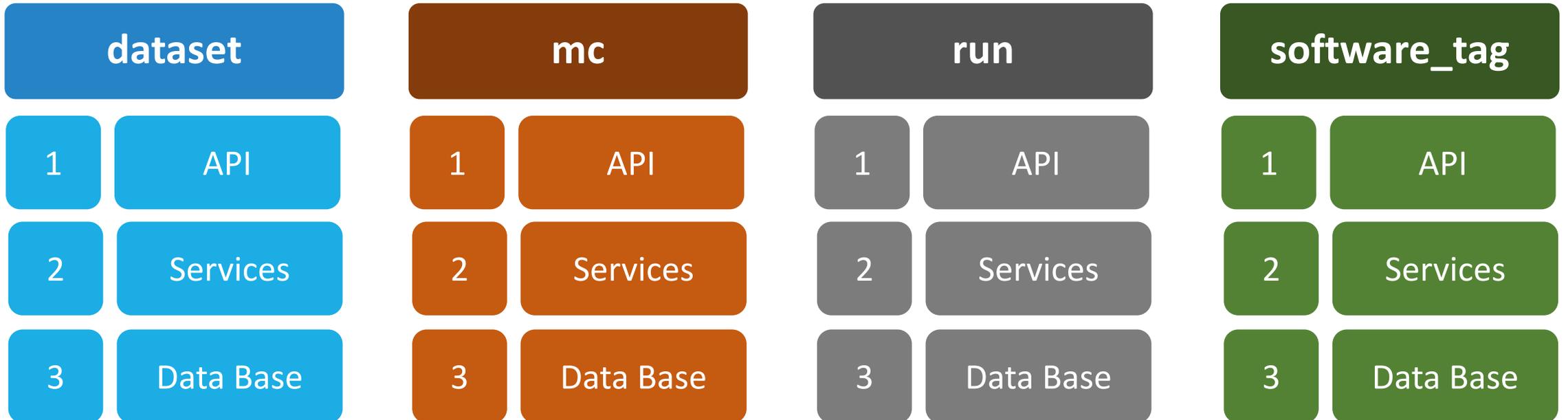


Microservice architecture: Data Base

- Настройка подключений к базам данных
- Запись и чтение в Postgres и Redis
- Миграции Postgres
- Паттерн «Repository»



Back-end microservices



Один микросервис = Одна схема БД

dataset_microservice

mc_microservice

run_microservice

software_tag_microservice

 Schemas (4)

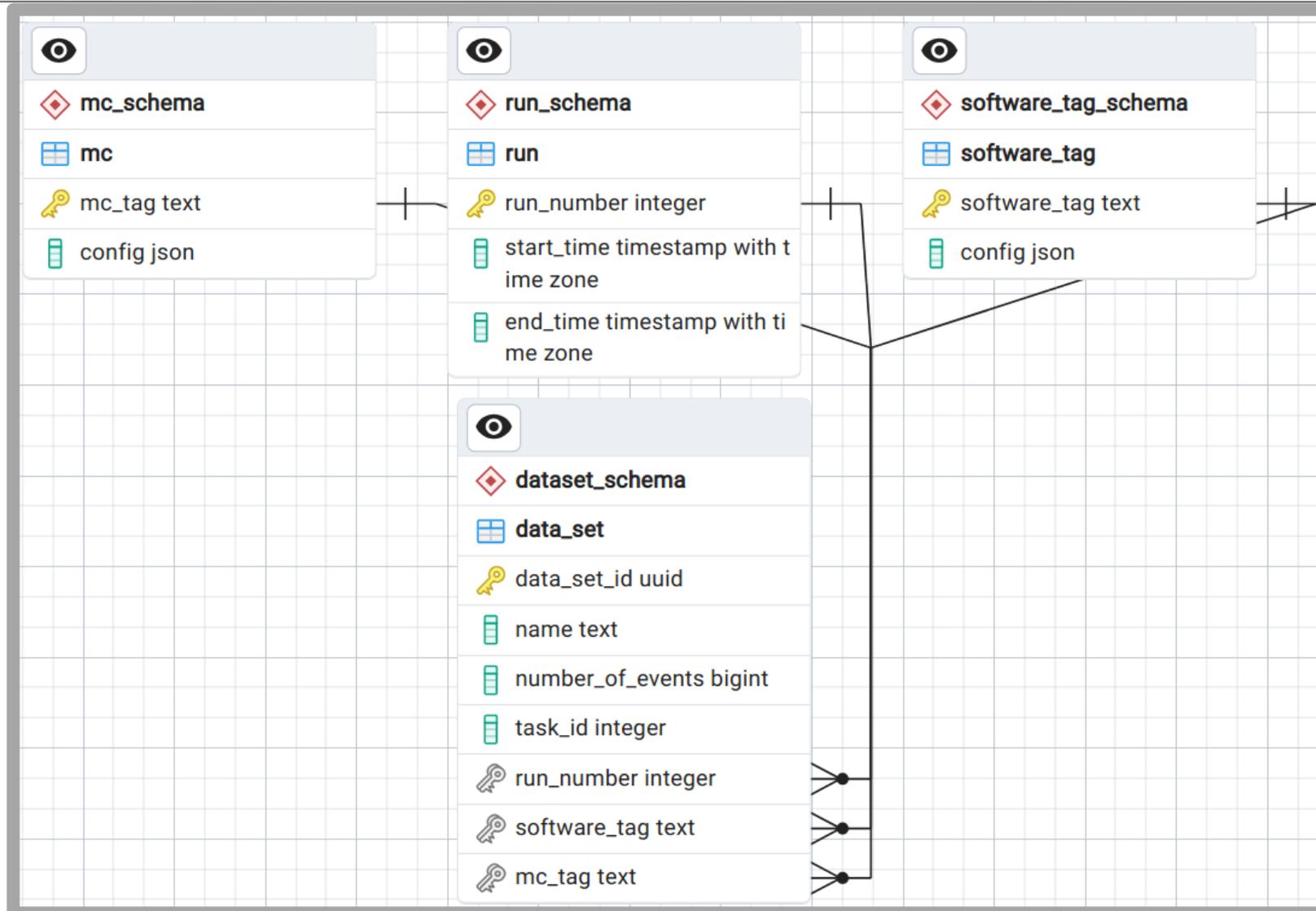
>  dataset_schema

>  mc_schema

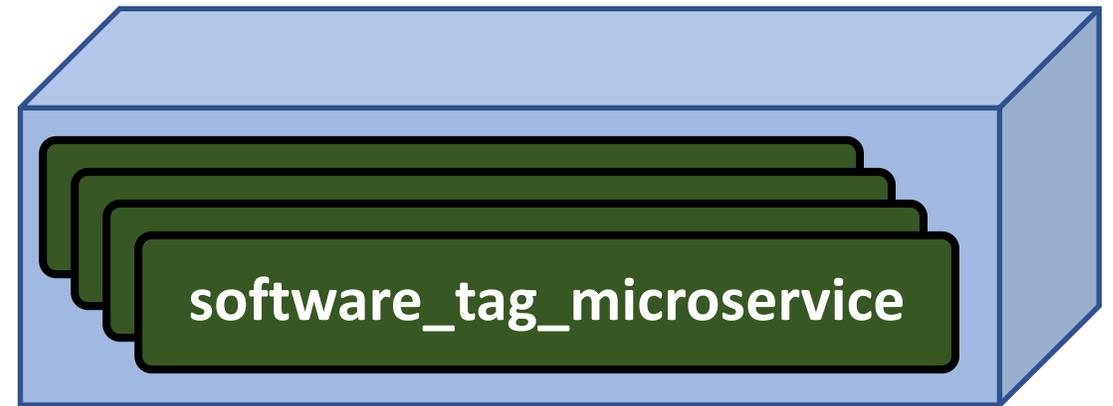
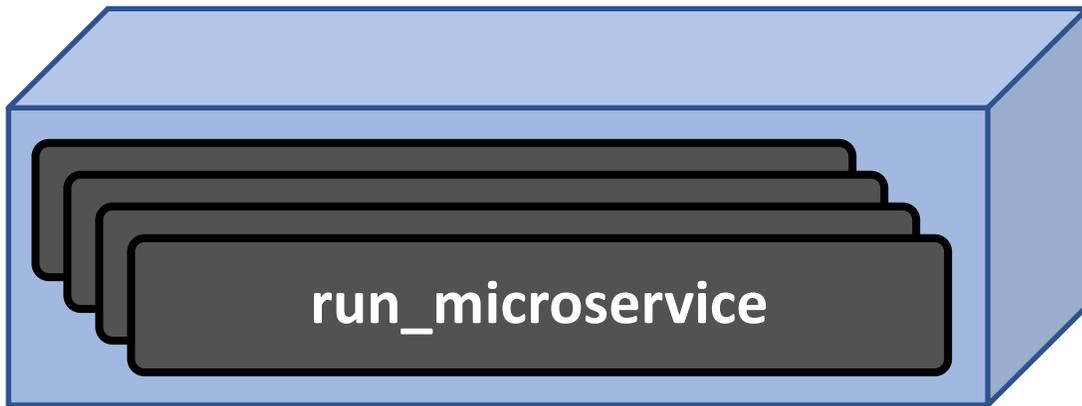
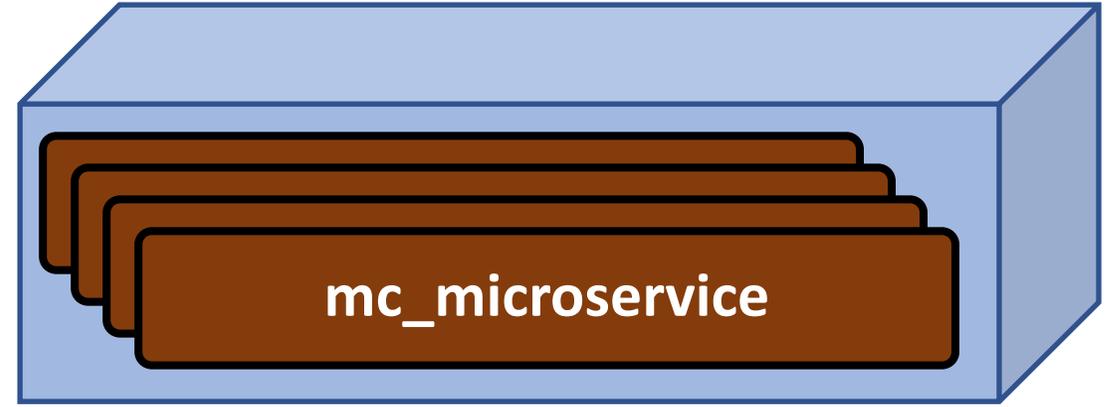
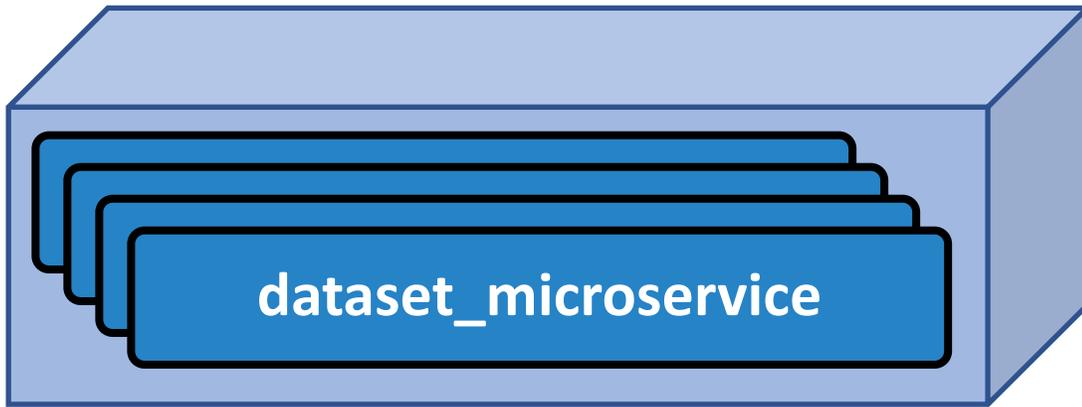
>  run_schema

>  software_tag_schema

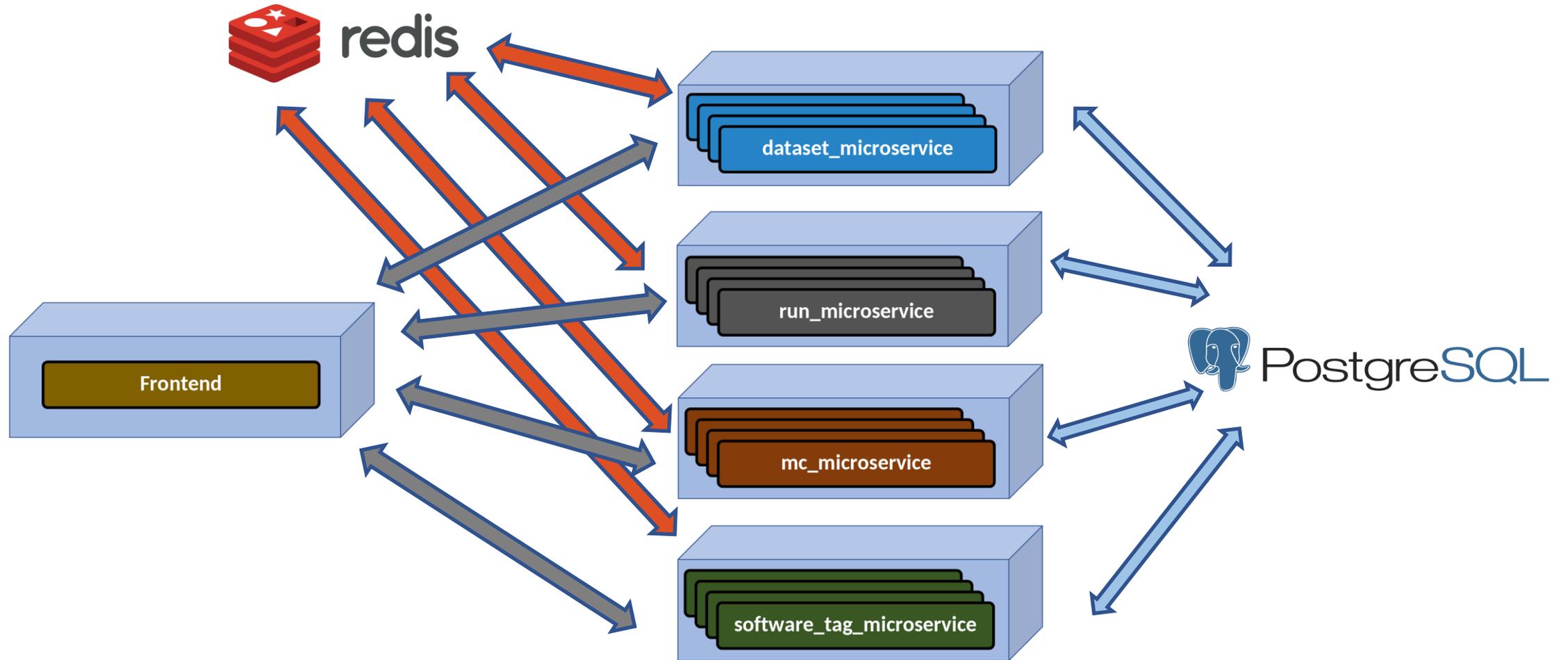
Схема хранения информации в БД



Docker



Microservices pattern «API Composer»

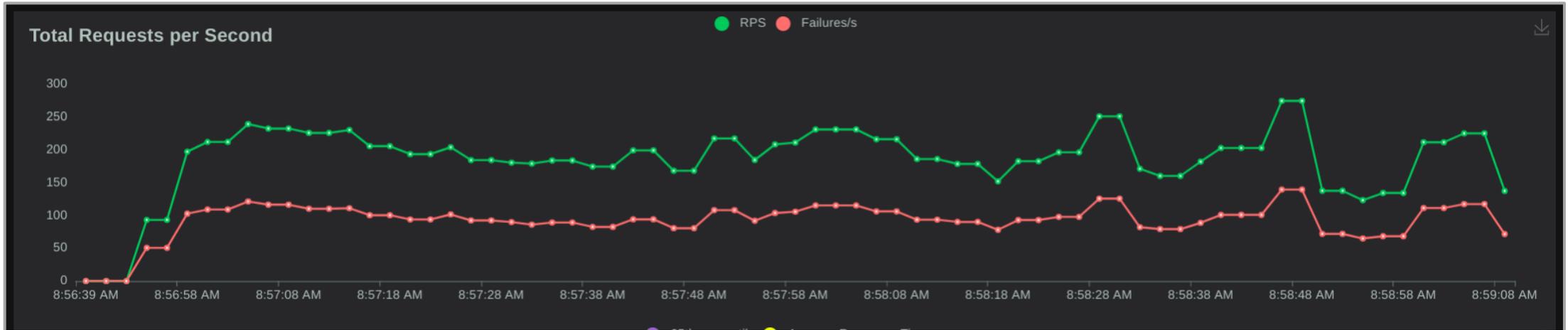


Нагрузочное тестирование

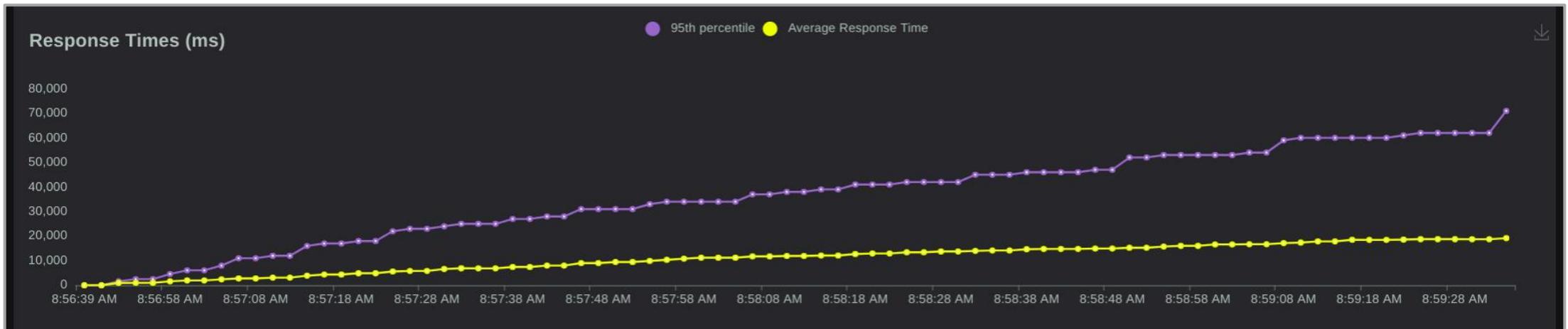
Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)
GET	/v1/	8005	11	7700	34000	37000
GET	/v1/testExample/get/2	7754	7754	7700	34000	37000
	Aggregated	15759	7765	7700	34000	37000

Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
12725.3	23	80858	20.97	73.8	0
12704.08	98	83895	20.96	78	78
12714.82	23	83895	20.97	151.8	78

Нагрузочное тестирование

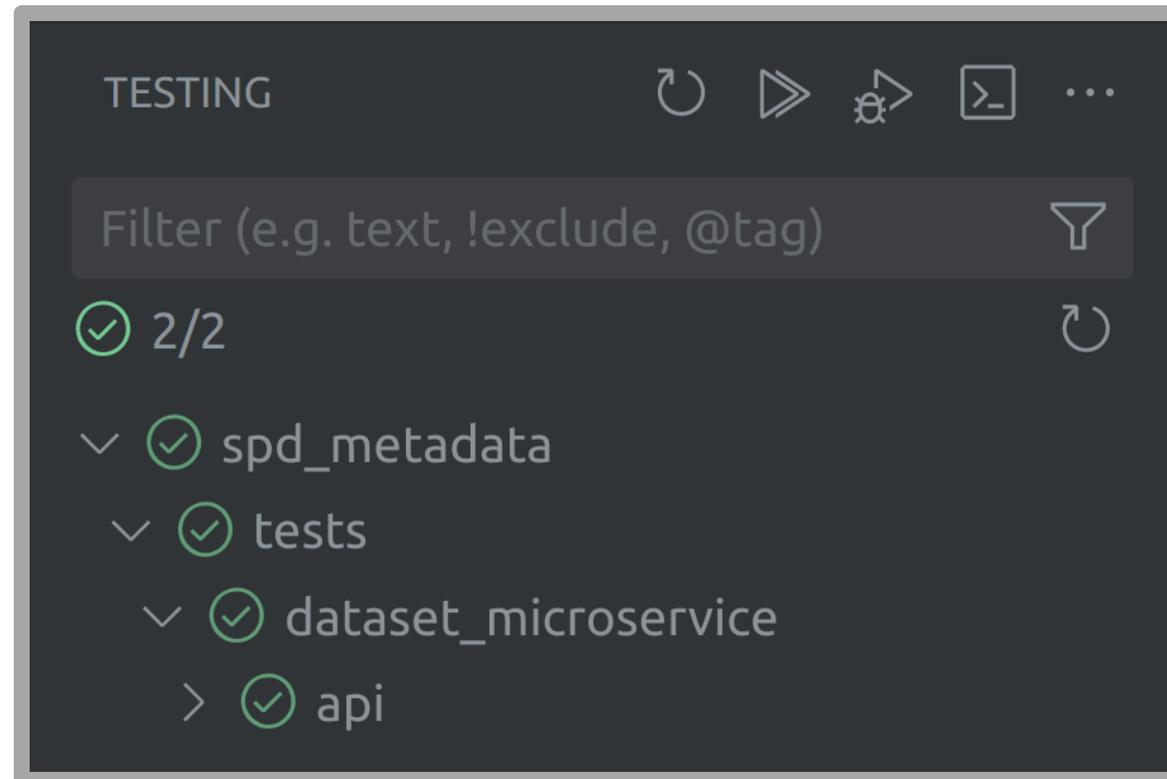


Total Requests per Second

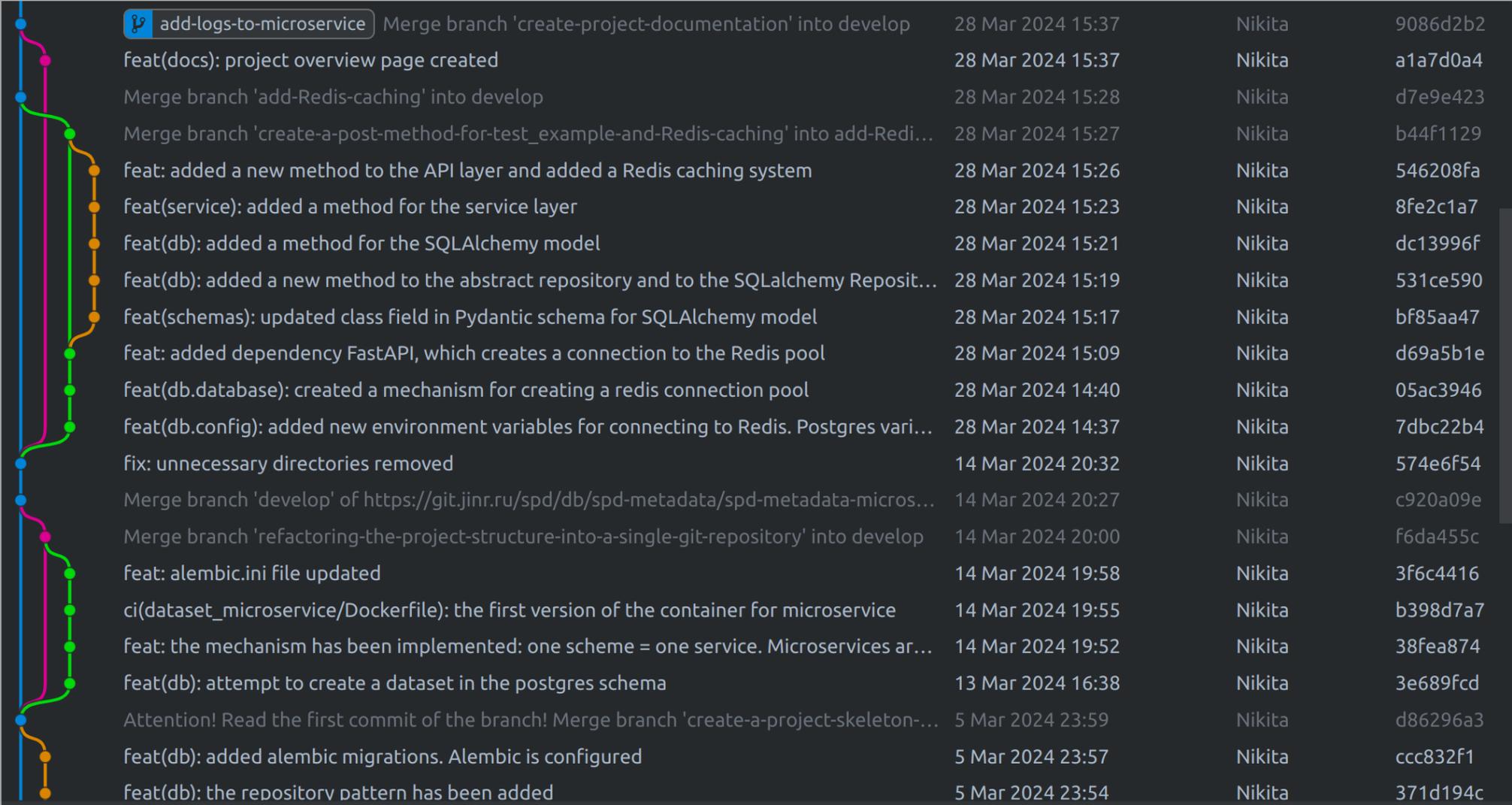


Response Times

Pytest



GitFlow



The image displays a Git commit history on the right and a corresponding branch diagram on the left. The commit history is a list of 20 entries, each with a commit message, date, time, author, and commit hash. The branch diagram shows a vertical line representing the main branch, with various colored lines (blue, pink, green, orange) representing other branches and their relationships, including merges and feature branches.

 add-logs-to-microservice	Merge branch 'create-project-documentation' into develop	28 Mar 2024 15:37	Nikita	9086d2b2
feat(docs): project overview page created		28 Mar 2024 15:37	Nikita	a1a7d0a4
Merge branch 'add-Redis-caching' into develop		28 Mar 2024 15:28	Nikita	d7e9e423
Merge branch 'create-a-post-method-for-test_example-and-Redis-caching' into add-Redi...		28 Mar 2024 15:27	Nikita	b44f1129
feat: added a new method to the API layer and added a Redis caching system		28 Mar 2024 15:26	Nikita	546208fa
feat(service): added a method for the service layer		28 Mar 2024 15:23	Nikita	8fe2c1a7
feat(db): added a method for the SQLAlchemy model		28 Mar 2024 15:21	Nikita	dc13996f
feat(db): added a new method to the abstract repository and to the SQLAlchemy Reposit...		28 Mar 2024 15:19	Nikita	531ce590
feat(schemas): updated class field in Pydantic schema for SQLAlchemy model		28 Mar 2024 15:17	Nikita	bf85aa47
feat: added dependency FastAPI, which creates a connection to the Redis pool		28 Mar 2024 15:09	Nikita	d69a5b1e
feat(db.database): created a mechanism for creating a redis connection pool		28 Mar 2024 14:40	Nikita	05ac3946
feat(db.config): added new environment variables for connecting to Redis. Postgres vari...		28 Mar 2024 14:37	Nikita	7dbc22b4
fix: unnecessary directories removed		14 Mar 2024 20:32	Nikita	574e6f54
Merge branch 'develop' of https://git.jinr.ru/spd/db/spd-metadata/spd-metadata-micros...		14 Mar 2024 20:27	Nikita	c920a09e
Merge branch 'refactoring-the-project-structure-into-a-single-git-repository' into develop		14 Mar 2024 20:00	Nikita	f6da455c
feat: alembic.ini file updated		14 Mar 2024 19:58	Nikita	3f6c4416
ci(dataset_microservice/Dockerfile): the first version of the container for microservice		14 Mar 2024 19:55	Nikita	b398d7a7
feat: the mechanism has been implemented: one scheme = one service. Microservices ar...		14 Mar 2024 19:52	Nikita	38fea874
feat(db): attempt to create a dataset in the postgres schema		13 Mar 2024 16:38	Nikita	3e689fcd
Attention! Read the first commit of the branch! Merge branch 'create-a-project-skeleton-...		5 Mar 2024 23:59	Nikita	d86296a3
feat(db): added alembic migrations. Alembic is configured		5 Mar 2024 23:57	Nikita	ccc832f1
feat(db): the repository pattern has been added		5 Mar 2024 23:54	Nikita	371d194c

Sphinx documentation

The screenshot shows a Sphinx-generated documentation page for the 'spd-metadata' project. The page is dark-themed and features a sidebar on the left with a search bar and a 'Contents' menu. The main content area displays a welcome message and a detailed table of contents. The right sidebar contains a 'Contents' section with a welcome message and a link to 'Indices and tables'.

spd-metadata
documentation

Search Ctrl + K

Contents:

- Project overview
- project

Welcome to spd-metadata's documentation!

Contents:

- [Project overview](#)
 - [Стек](#)
 - [Структура проекта](#)
 - [Существуют 4 основных микросервиса:](#)
 - [Структура микросервиса](#)
- [project](#)
 - [base_metadata module](#)
 - [dataset_microservice_package](#)
 - [mc_microservice_package](#)
 - [run_microservice_package](#)
 - [software_tag_microservice_package](#)

Indices and tables

- [Index](#)
- [Module Index](#)

Contents

Welcome to spd-metadata's documentation!

Indices and tables

Sphinx documentation

```
22
23 @router.get("/testExample/get/{numberr}")
24 async def get_user_details(test_example_service: Annotated[TestExampleService, Depends(test_exam
25     """Function for testing Redis. When you enter a number,
26     it displays all the data from the database. Simulating a complex user request.
27
28     Args:
29         test_example_service (Annotated[TestExampleService, Depends]): service layer
30         numberr (str): random number for testing Redis
31         redis_cache (_type_, optional): link to DB layer. Defaults to Depends(get_redis).
32
33     Returns:
34         json: all data in DB
35     """
36
37     cache = await redis_cache.get(numberr)
38     if cache is not None:
39         return {'numberr': numberr, 'data': cache, 'source': 'cache'}
40
41     data = await test_example_service.get_all_data()
42
43     # save cache in memory for 1 hour
44     await redis_cache.set(numberr, str(data), ex=3600)
45     return data
```

Sphinx documentation

The screenshot shows a Sphinx documentation page for the `dataset_microservice.api.v1.test_example` module. The page is dark-themed and includes a sidebar with a search bar and a navigation tree. The main content area displays the module name, a function signature for `get_user_details`, a description, and details about its parameters and return type. The sidebar on the left shows the project structure, with the current page selected under `dataset_microservice.api.v1.test_example`. The sidebar on the right shows a table of contents with links to submodules and module contents.

spd-metadata
documentation

Search Ctrl + K

Contents:

- Project overview
- project**
 - base_metadata module
 - dataset_microservice package**
 - dataset_microservice.api package**
 - dataset_microservice.api.v1 package**
 - dataset_microservice.db package
 - dataset_microservice.service package
 - mc_microservice package
 - run_microservice package
 - software_tag_microservice package

dataset_microservice.api.v1.router module

dataset_microservice.api.v1.test_example module

async

`dataset_microservice.api.v1.test_example.get_user_details(test_example_service: TestExampleService, numberr: str, redis_cache=Depends(get_redis))` [\[source\]](#)

Function for testing Redis. When you enter a number, it displays all the data from the database. Simulating a complex user request.

Parameters:

- `test_example_service` (*Annotated[TestExampleService, Depends]*) – service layer
- `numberr` (*str*) – random number for testing Redis
- `redis_cache` (*_type_, optional*) – link to DB layer. Defaults to `Depends(get_redis)`.

Returns:

all data in DB

Return type:

json

async `dataset_microservice.api.v1.test_example.post_user_details(text: TrainingExampleSchemaAdd, test_example_service: TestExampleService) →`

Contents

Submodules

- dataset_microservice.api.v1.dependency module
- dataset_microservice.api.v1.event_index module
- dataset_microservice.api.v1.router module
- dataset_microservice.api.v1.test_example module

Module contents

Log monitoring

The screenshot displays a log monitoring interface with a dark theme. At the top, there is a search bar with the text "Search or jump to..." and a "ctrl+k" shortcut. Below the search bar, a navigation breadcrumb shows "Home > Dashboards > Logs" with a star icon. To the right of the breadcrumb are buttons for "Add" and "Share", and a time range selector set to "2024-04-11 10:07:57 to 2024-04-11 22:07:57".

The main content area is divided into four panels, each showing log entries for a specific microservice:

- dataset_microservice:** Shows a series of INFO logs indicating successful server process starts and application startups for processes 17, 19, 18, and 16.
- run_microservice:** Shows logs for booting workers and listening at http://0.0.0.0:8000. It includes several ERROR logs indicating worker failures and SIGTERM signals.
- mc_microservice:** Shows a traceback for a ModuleNotFoundError, indicating that the module 'project.dataset_microservice' is not found. The traceback includes file paths and line numbers.
- dataset_microservice:** Shows a similar ModuleNotFoundError traceback, indicating the same module is not found in a different context.

Текущие результаты

1. Изучены проектируемые информационные системы эксперимента SPD.
2. Настроена гибкая система разработки, с минимальной связностью компонентов друг от друга. Это позволяет вносить новый функционал, и изменять старый в программный код - с минимальными затратами по времени.
3. Разработаны схемы хранения метаданных информации в БД.
4. Разработан начальный API для обмена данными с пользователями и другими ИС. Разработаны средства получения метаданных, их транспортировки и записи в БД. Разработаны интерфейсные приложения для обслуживания запросов пользователей.

Дальнейшие планы

- Сравнить Docker и Apptainer
- Заменить Redis на Valkey
- Исполтзовать паттерн Gateway API вместо API Composer
- Kubernetes, балансировка нагрузки
- Мониторинг Prometheus
- Front-end
- По мере развития других информационных систем SPD будет производится добавление новых компонент, микросервисов, и другого функционала

Спасибо за внимание!
