федеральное государственное автономное образовательное
учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по Лабораторной работе №3

«Структурная схема с обоснованием»

по дисциплине «**Облачные и туманные вычисления**»

Авторы: Кулаков Н.В. P34312

Лысенко Д.С. P34121

Факультет: ПИиКТ

Преподаватель: Перл О.В.

Санкт-Петербург, 2023

Содержание

Решение

Будет разработано веб-приложение с простым интерфейсом, в котором пользователь сможет создавать и использовать уже существующие пайплайны, шаги которого последовательно обрабатываются нейросетевыми моделями.

Используемые сервисы

1. Virtual Machine. Нейросетевые модели для инференции (изображений и текстов). Для генерации различных типов ответов будут созданы отдельные нейросетевые сервисы.

1.2. Virtual Machine. Сервер для обработки запросов пользователей, также осуществляющий сохранение результатов генерации и метаданных (backend + frontend).

2. YDB для хранения информации о пользователях, метаданные о сгенерированных записях и ссылки на бакеты/данные в S3.

3. Yandex Object Storage для хранения текста и изображений.

4. TLS Certificate Manager. Сервис для управления TLS сертификатами для домена (фронта).

5. Yandex Message Queue для управления инференцией (процесс генерации) и распределения задач между нейросетевыми нодами, генерации очереди пользователей.

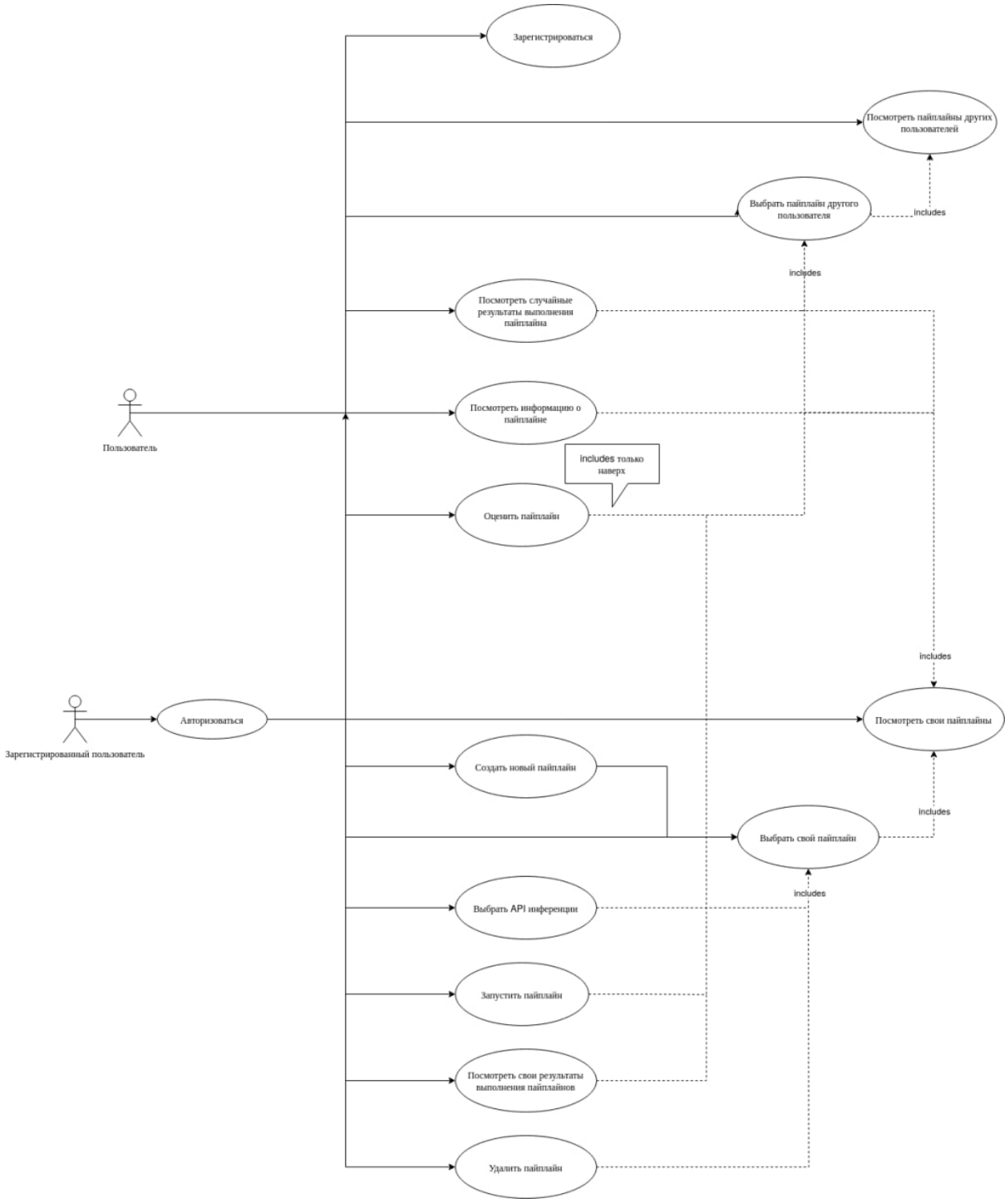
6. Yandex Monitoring для мониторинга основных метрик сервиса, например, средняя скорость инференции, использование оперативной памяти и количество сгенерированных изображений и так далее.

В качестве облачных провайдеров были выбраны Yandex Cloud и Selectel (для машин с GPU)

Функциональные требования

1. Приложение должно предоставлять возможность авторизации и регистрации пользователей
2. Приложение должно предоставлять возможность просматривать существующие пайплайны пользователя, либо других пользователей
3. Приложение должно предоставлять возможность создавать новый пайплайн
4. Приложение должно предоставлять возможность удалять уже доступный пайплайн
5. Приложение должно предоставлять возможность просмотра оценки пайплайна
6. Приложение должно предоставлять возможность ставить оценку пайплайну другого пользователя
7. Приложение должно предоставлять возможность выбирать любой доступный в системе пайплайн для его запуска
8. Приложение должно предоставлять возможность выбора API инференции при использовании пайплайна
    a. Приложение должно предоставлять доступ к следующим API:
        i. Собственное API (инференция на виртуальной машине с GPU)
        ii. OpenAI
        iii. AI Horde
9. Приложение должно предоставлять возможность запуска пайплайна
10. Приложение должно предоставлять возможность сохранения результатов пайплайна на сервере
11. Приложение должно предоставлять возможность просмотра результатов пайплайна на сервере
12. Приложение должно предоставлять возможность удаления результатов пайплайна на сервере

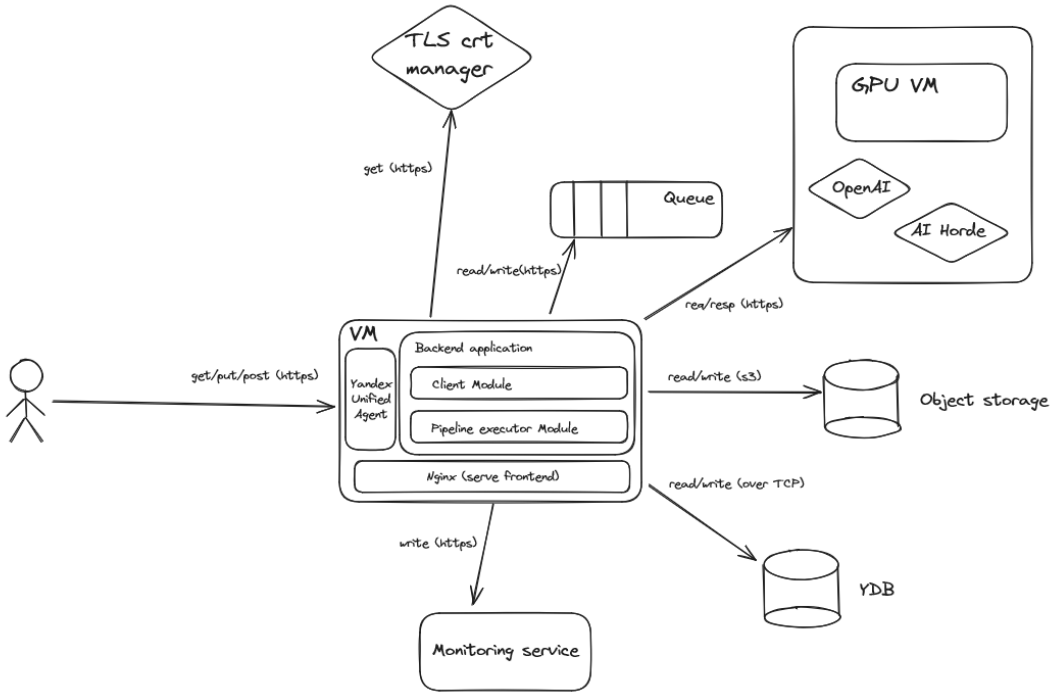
Use-case diagram



Роли

В системе будет существовать только одна роль - зарегистрированный пользователь. В отличии от обычного пользователя (гостя) у него не будет возможности запускать пайплайны и содавать свои.

Примерная архитектура решения



TLS Manager - с помощью этого сервиса будет генерироваться Let’s Encrypt сертификат по нашему домену, на котором будет висеть наше приложение. Автоматически будет осуществляться продление сертификата при подходе к окончанию срока действия, на стороне виртуалки предполагается, что при наступлении этого события будет выгружаться сертификат из менеджера и замещать старый.

Monitoring Service - на backend-е будет создан эндпоинт метрик, затем Yandex Unified Agent будет отправлять метрики в сервис мониторинга. При помощи заранее настроенных дешбордов можно будет посмотреть на визуализированные метрики по параметрам.

Queue - используется как очередь pipeline-ов на выполнение. От пользователя приходит запрос на выполнение пайплайна, который помещается в нее. Эти задачи считываются

отдельным модулем backend-а, который осуществляется последовательное делегирование этапов пайплайна на GPU или внешнее API.

Object Storage - используется для хранения файлов пайплайнов (json-ы), результатов выполнения пайплайнов (изображения или текст)

YDB - для хранения информации о пользователях, всех созданных пайплайнов (ссылок на Object Storage), а так же результатов выполнения пайплайно (ссылок)

Диаграмма разворачивания

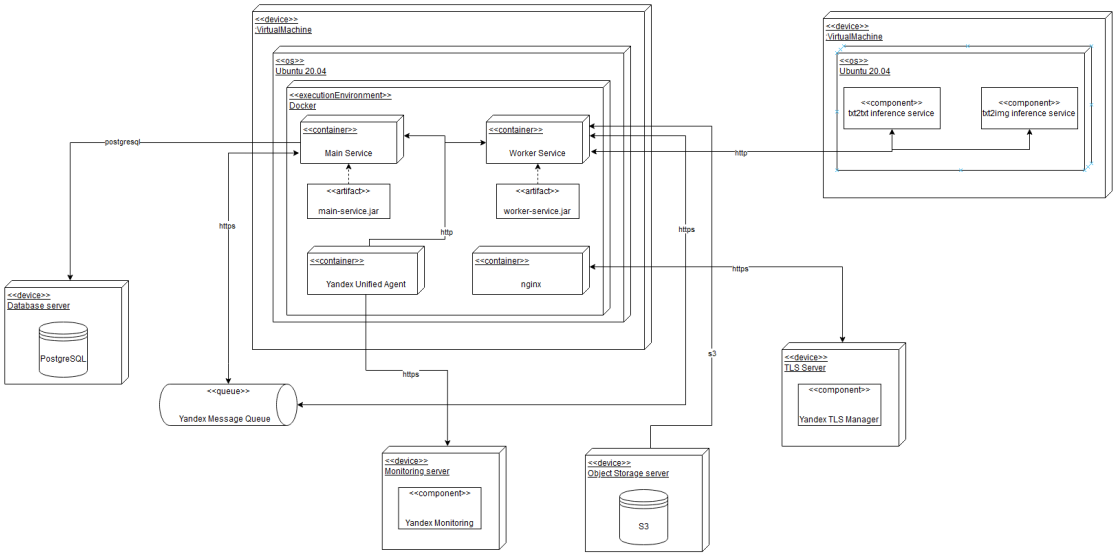
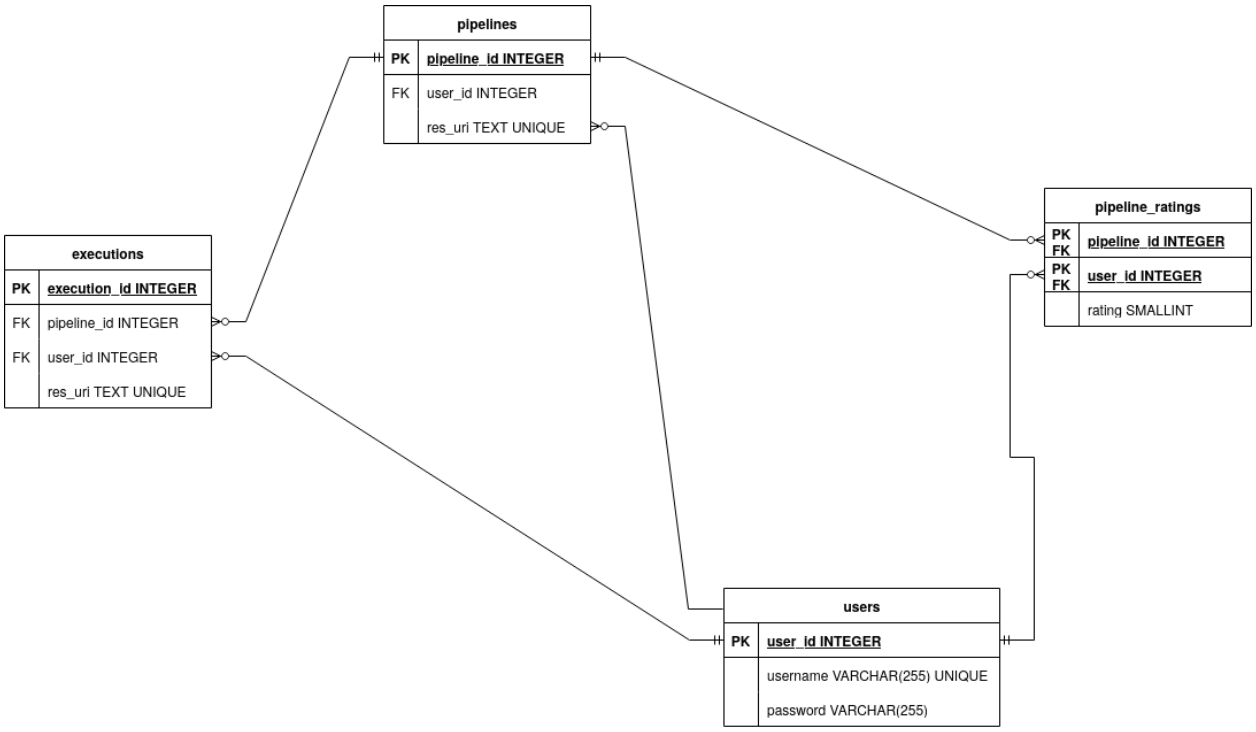


Схема базы данных



Описание пайплайна

Пайплайны будут храниться в Object Storage в виде файла json.

```
{
      "pipeline_id": integer
      "name": string,
      "author": string,
      "stages": [
            {
                  "type": string,
                  "inline": boolean,
                  "body": string,
                  "parameters": []
            },
            {
                  "type": string,
```

```
                "inline": boolean,
                "body": string,
                "parameters": []
            }
        ]
}
```

Сообщения, передаваемые в очередь выполнения запросов - совпадают с определением пайплайна.

- stages - этапы пайплайна.
- stages.type - тип задачи. Например, txt2img (генерация на основании изображения текста), txt2txt (генерация текста на основании текста).
- stages.inline - используется ли внутри пайплайна prompt, или ссылка на объект Object Storage. Это необходимо, поскольку Yandex Queue принимает максимально 256 KB в качестве записи, что требует использования ссылок, а не самих данных.
- stages.body - тело запроса (prompt), либо ссылка на само тело.
- stages.parameters в этапах пайплана - различные конфигурационные параметры, которые непосредственно влияют на инференцию. Например, это может быть тип выборщика, используемый при генерации токенов в текстовых моделях, температура (случайный фактор) и т.д.

Масштабирование

Узкое место в описанной системе - это виртуальная машина с GPU и backend-ом собственной API инференции. На одной машине может одновременно выполняться только одна задача - либо генерация изображений, либо генерация текста. Ситуацию может изменить:

- добавление дополнительных GPU на машины - в таком случае можно на каждый графический процессор назначить по одной задаче инференции нейронной сети.
- увеличение количества машин с GPU - каждая машина будет выбирать по задаче из очереди и обрабатывать запрос.

Если объединить эти два способа, можно добиться значительного прироста количества обрабатываемых задач за определенное время

При увеличении количества пользователей можно развернуть/настроить авто скейлинг машин с Frontend/Backend (задачи попадают только на одну случайную машину), перед ними поставить балансировщик нагрузки, например, на основе Network Load Balancer (Yandex Cloud)