ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 2](#_Toc151717463)

[1 Аналитический раздел 3](#_Toc151717464)

[1.1 Описание проблемы 3](#_Toc151717465)

[1.2 Постановка задачи 3](#_Toc151717466)

[1.3 Анализ существующих решений 3](#_Toc151717467)

[2 Конструкторский раздел 4](#_Toc151717468)

[2.1 Предложенное решение 4](#_Toc151717469)

[2.2 Описание протокола 4](#_Toc151717470)

[3 Технологический раздел 5](#_Toc151717471)

[3.1 Используемый стек технологий 5](#_Toc151717472)

[3.2 Реализация протокола 5](#_Toc151717473)

[4 Исследовательский раздел 6](#_Toc151717474)

[4.1 Демонстрация работы протокола 6](#_Toc151717475)

[5 ЗАКЛЮЧЕНИЕ 7](#_Toc151717476)

[6 БИБЛИОГРАФИЧЕСКИЙ СПИСОК 8](#_Toc151717477)

# ВВЕДЕНИЕ

# Аналитический раздел

(Здесь будет описание глобальной проблемы, существующих решений, описание того, что именно мы хотим решить из глобальной проблемы и идея решения)

## Описание проблемы

Здесь описать какие есть виды и задачи репликации. Для чего и когда применяется.

## Постановка задачи

Здесь описать, что именно мы хотим сделать (формальное описание master-master репликации)

## Анализ существующих решений

Описание того, какие виды репликации есть в Postgres (физическая и логическая)

# Конструкторский раздел

(Здесь будет описываться (формально) протокол, который мы реализуем для решения проблемы)

## Предложенное решение

Решение программно реализовать логическую репликацию с возможностью синхронизации таблиц баз данных

## Описание протокола

А вот здесь нужно подумать, как это дело описать, до сюда я ещё не дошел. Здесь описание внутреннего взаимодействия команд с примерами.

# Технологический раздел

(Здесь будет описание реализации протокола)

## Используемый стек технологий

## Реализация протокола

# Исследовательский раздел

(Здесь будет демонстрация работы)

## Демонстрация работы протокола

# ЗАКЛЮЧЕНИЕ

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Здесь я буду собирать материалы, которые нам понадобятся

1. <https://studfile.net/preview/8867121/#2> – дается базовое описание взаимодействия программ прикладного уровня (сама статья устаревшая, тк использует Delfi). Можно взять использование сокеты для взаимодействия с протоколами сетевого уровня. Плюс поднимается вопрос про открытость системы (с этим нужно разобраться, как сделать систему открытой)
2. <https://vc.ru/u/1263669-vasiliy-volgin/648641-pro-migraciyu-i-sinhronizaciyu-bazy-dannyh-dlya-nachinayushchego-testirovshchika-2023> - базовое понимание, что такое миграция, что такое синхронизация, в чем их отличия
3. <https://www.ibm.com/docs/ru/i/7.1?topic=system-synchronization-planning-procedures> – про синхронизацию систем (больше подходит под понятие миграции с одной на другую). Предлагается поддерживать две системы (с разными данными), которые раз в какой-то промежуток будут обмениваться ими. В конце получим две системы с одинаковыми данными и можно будет перейти на вторую.
4. <https://selectel.ru/blog/tutorials/how-to-set-up-replication-in-postgresql/> - описание репликаций в Postgresql.   
   Идет описание WAL (журнал предзаписи транзакций) – перед тем как сохранить информацию на диске команды записываются сюда, чтобы:   
   1) если будет сбой в сервере можно восстановить незафиксированные данные;   
   2) этот журнал используется и при репликации.   
   Дальше о видах репликации:   
   1) потоковая (на реплики передаются WAL) – есть ограничения, нужно иметь идентичные базы (версия, ОС, архитектура). Есть асинхронная (сначала применит у себя, потом отправит WAL), есть синхронная (сначала отправляется WAL (хотя бы на одну реплику), потом сохраняется на первый сервер).   
   2) логическая – оперирует записями в таблицах. То есть отправляет команды на реплики. Можно одни таблицы реплицировать на один сервер, другие на другой. (WAL работает на физическом уровне уже, заменяются байты, а не выполняются команды). Ограничения: реплицировать структуру необходимо отдельно, вручную (ЗОНА РОСТА).  
   Ну и дальше идет описание как это все настроить на сервере постгрес.
5. <https://sysadminium.ru/replikaciya_v_postgresql/> - повторяет предыдущую статью (описывает физическую и логическую репликацию). Кроме этого дают задачи и виды репликации.
6. <https://postgrespro.ru/docs/postgresql/10/logical-replication> - описание логической репликации на оф.сайте
7. <https://timeweb.cloud/tutorials/postgresql/logicheskaya-replikaciya-postgresql> - описание логической (и немного физической) репликации постгрес.
8. <https://habr.com/ru/companies/otus/articles/710956/> - описание того, почему и зачем используются репликации. Также говорится о видах репликации. Есть примеры работы репликаций.

Мысли по реализации:

Два идентичных сервиса (бэкенда), которые работают с посгрес внутри. Обеспечивают master-master синхронизацию. Наверное, на логическом уровне, чтобы можно было реплицировать и структуру, и данные.

То есть. Сервис по http принимает команды к СУБД (внутри будет какая-нибудь Sqlalchemy для простоты), а через сокеты отправлять данные на реплику. (которая в свою очередь тоже может отправлять изменения на первую).

Состав тестового композа:

1. Postgres 1 (образ postgres)
2. Postgres 2 (образ postgres)
3. Сервер 1 (образ сервиса, адрес запуска 8080, работает с Postgres 1)
4. Сервер 2 (образ сервиса, адрес запуска 8050, работает с Postgres 2)

Функционал API:

1. Настроить репликацию (подключение двух сервисов, какие-нибудь настройки)
2. Запустить репликацию
3. Остановить репликацию
4. Команды для работы с бд: (не допустить SQL инъекции :)
   1. Создать таблицу
   2. Добавить запись
   3. Получить запись
   4. Обновить запись
   5. Удалить запись