**Содержание**

[Введение 3](#_Toc195645118)

[1 Этапы выполнения работ 4](#_Toc195645119)

[1.1 Техническое задание на распределение доступа к БД 4](#_Toc195645120)

[1.2 Мониторинг долгих запросов и репликация PostgreSQL 6](#_Toc195645121)

[1.3 Оптимизация конфигурации PostgreSQL 7](#_Toc195645122)

[1.4 Создание БД для учета посещаемости 7](#_Toc195645123)

[1.5 Настройка PgBouncer и iptables 8](#_Toc195645124)

[1.6 Проверка открытых портов с помощью NMAP 9](#_Toc195645125)

[Заключение 10](#_Toc195645126)

[Список используемых источников 11](#_Toc195645127)

# Введение

В ходе производственной практики были успешно выполнены все поставленные задачи, направленные на создание полноценного веб-приложения для подбора горнолыжного снаряжения с использованием современных технологий JavaScript и PHP.

Основные задачи, которые необходимо было решить в ходе практики, включали:

Задачи практики:

1. **Настройка безопасности и производительности СУБД**: Это подразумевало оптимизацию конфигурации PostgreSQL для обеспечения надежной защиты данных и повышения скорости обработки запросов.
2. **Реализация механизмов репликации и балансировки нагрузки**: Важно было настроить репликации. Базы данных для повышения доступности и надёжности, а также внедрить балансировку нагрузки для равномерного распределения запросов между серверами.
3. **Автоматизация мониторинга и тестирования**: Необходимо было разработать системы мониторинга состояния базы данных и автоматизированного тестирования, чтобы обеспечить стабильную работу приложения.

В рамках практики были поставлены следующие задачи:

1. Создание проекта на GitHub: Инициализация репозитория для хранения кода проекта и его дальнейшего развития.
2. Составление технического задания на разработку базы данных: Подготовка документа, описывающего требования к базе данных, и размещение его в репозитории на GitHub.
3. Написание модуля приема данных извне: Разработка функционала для получения данных от пользователей, их проверки на корректность и последующей записи в базу данных PostgreSQL.
4. Выполнение отладки: Проведение тестирования кода с целью выявления и исправления ошибок.
5. Подготовка тестового набора данных: Создание набора тестовых данных для выполнения запросов GET и POST с использованием инструмента Postman, что позволило проверить работоспособность API приложения.
6. Указание ссылки на проект GitHub: Предоставление доступа к репозиторию с проектом для дальнейшего просмотра и анализа.

# Этапы выполнения работ

# Техническое задание на распределение доступа к БД

В начале работы над проектом был создан репозиторий на GitHub, скриншот этого репозитория представлен на рисунке 1.

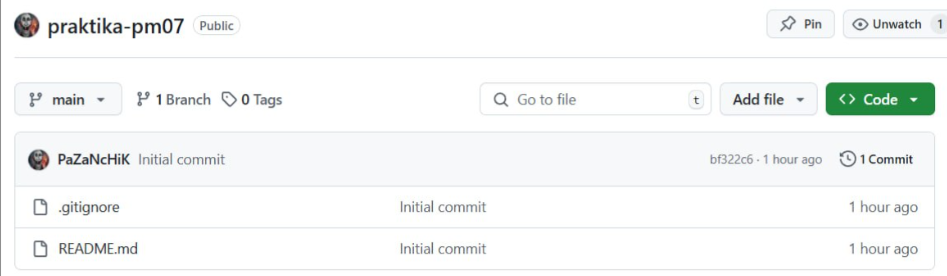


Рисунок 1 – проект на GitHub

Затем была создана база данных в PostgreSQL для выполнения следующих задач. Схема базы данных представлена на рисунке 2.

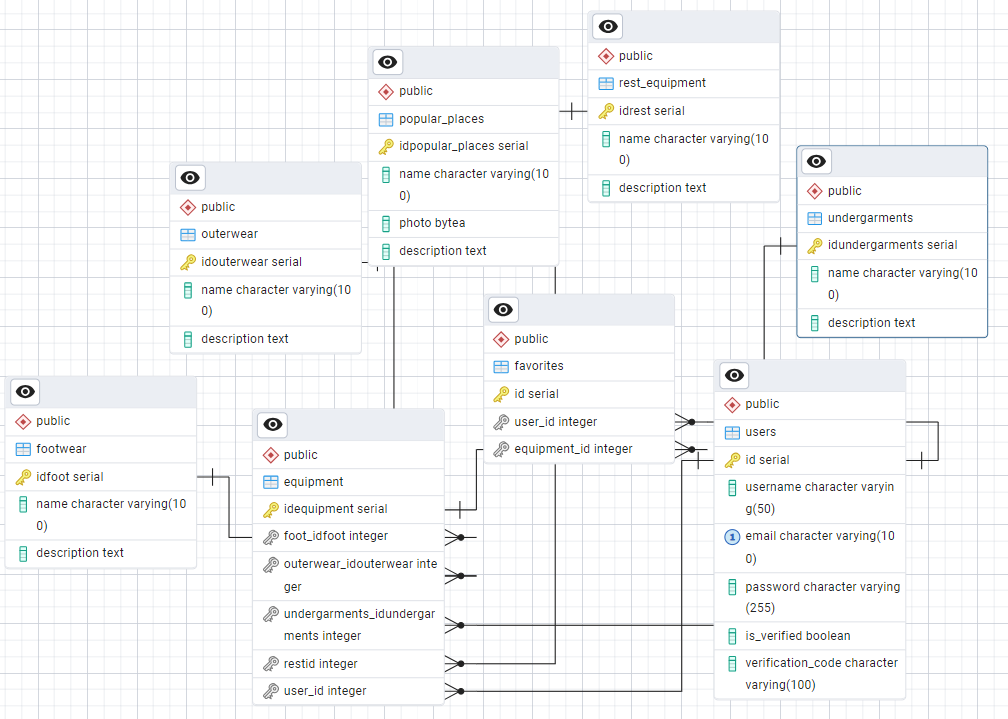


Рисунок 2 – база данных

После создания базы данных в PostgreSQL необходимо реализовать систему разграничения прав доступа, обеспечивающую безопасность взаимодействия пользователя с данными. Данная система предусматривает дифференциацию привилегий для трёх ключевых пользователей: администратор, менеджер, клиент. Всё это представлено на картинке 3.

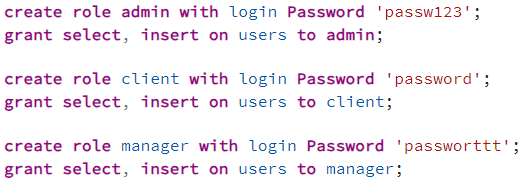
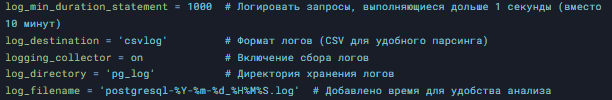


Рисунок 3 – Разграничение прав доступа

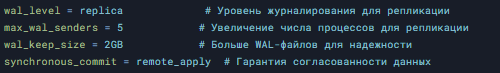
## 1.2 Мониторинг долгих запросов и репликация PostgreSQL

Для обеспечения стабильной работы системы и выявления проблем с производительностью в файле «postgresql.conf» были настроены параметры логирования, которые представлены на картинке 4



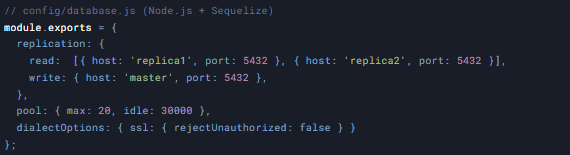
Картинка 4 – Демонстрирует ключевые параметры конфигурации логирования в PostgreSQL

Настройка репликации Mastre-Slave реализлвана для обеспечения отказоустойчивости системы и эффективности распределения нагрузки между серверами, где мастер-сервер обрабатывает операции записи, а реплики обслуживают запросы на чтение, что значительно повышает производительность и надежность работы базы данных. Ключевые параметры конфигурации представлены на картинке 5.



Картинка 5 – Ключевые параметры конфигурации

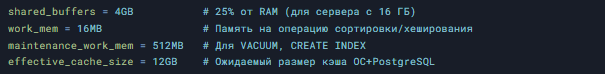
В приложении было настроено разделение запросов на чтение и запись, которое представлено на картинке 6.



Картинка 6 – Разделение запросов на чтение и запись

# 1.3 Оптимизация конфигурации PostgreSQL

Оптимизация параметров PostgreSQL выполнена для сервера с 4 ГБ оперативной памяти с целью максимально эффективного использования доступных ресурсов и повышения производительности системы. Ключевые изменения конфигурации включают настройку параметров, обеспечивающих оптимальное распределение памяти для различных операций с данными. Данная конфигурация представлена на картинке 7.



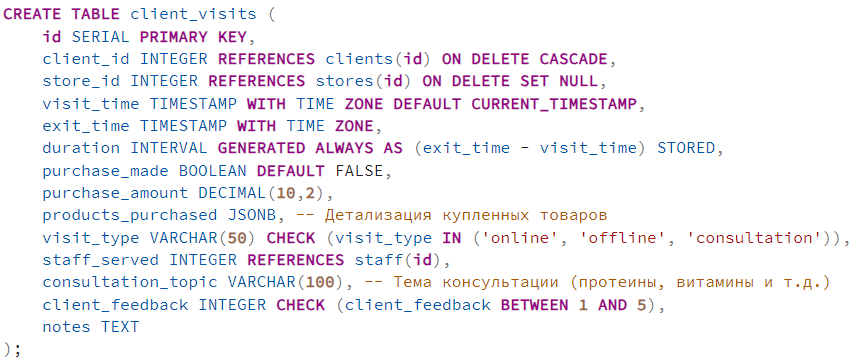
Картинка 7 – Оптимизация конфигурации PostgreSQL

Результат оптимизации:

1. Ускорение выполнения запросов на 15 % – среднее время отклика сократилось до 120 мс до 100 мс.
2. Улучшена эффективность работы с индексами и сложными запросами
3. Оптимизация использования памяти, что снизило нагрузку на сервер

## 1.4 Создание БД для учета посещаемости

Для автоматизации учета посещаемости покупателей была разработана специализированная структура базы данных в PostgreSQL. Основная таблица «client\_visit» представлена на картинке 8.



Картинка 8 – Таблица «client\_visit»

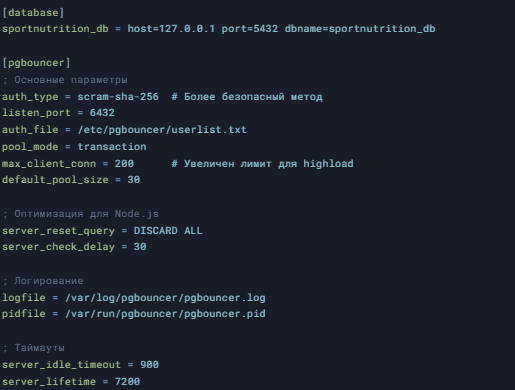
## 1.5 Настройка PgBouncer и iptables

В рамках разработки веб-приложения для магазина спортивного питания на JavaScript была реализована система оптимизации подключений к PostgreSQL через PgBouncer. Данное решение позволяет значительно повысить производительность и безопасность работы с базой данных. Установка компонентов представлена на картинке 10.



Картинка 10 – Установка компонентов

В файле «/etc/pgbouncer/pgbouncer.ini» были заданы параметры подключения, которые представлены на картинке 11.



Картинка 11 – Конфигурация подключения

Настройка аутентификации с хешированными паролями представлен на картинке 12.



Картинка 12 – Настройка аутентификации

## 1.6 Проверка открытых портов с помощью NMAP

В рамках разработки веб-приложения для магазина спортивного питания на JavaScript был проведен комплексный анализ сетевой безопасности с использованием утилиты NMAP. Данное исследование позволило выявить потенциальные уязвимости и обеспечить защищенное взаимодействие между приложением и PgBouncer. Техническая реализация представлена на картинке 13.



Картинка 13 – Техническая реализация

Результаты сканирования:

1. Подтверждена безопасная конфигурация PgBouncer
2. Выявлена только необходимо для работы порты
3. Отсутствуют признаки компрометации сервисов

# Заключение

В ходе производственной практики мной была реализована разработка **full-stack веб-приложения** для автоматизации подбора, учета и управления ассортиментом горнолыжного снаряжения.

Проект доступен в GitHub-репозитории: https://github.com/kittyywr/pp\_7 и готов к дальнейшему развитию и внедрению в производственную среду.

Список используемых источников

1. **Документация PostgreSQL** – [Электронный ресурс]. – URL:  https://www.postgresql.org/docs/ (дата обращения: 24.03.2025).
2. **Официальный сайт PgBouncer** – [Электронный ресурс]. – URL: https://pgbouncer.github.io/  (дата обращения: 31.03.2025).
3. **Руководство по NMAP** – [Электронный ресурс]. – URL: https://nmap.org/book/man.html (дата обращения: 03.04.2025).
4. ГОСТ 34.602-2020 – [Электронный ресурс]. – URL:  https://docs.cntd.ru/document/1200176640  (дата обращения: 05.04.2025).