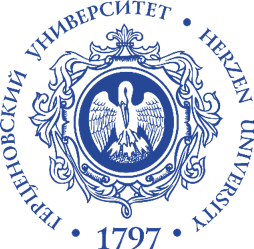
МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ

ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»



Направление подготовки   
09.03.01 Информатика и вычислительная техника

Направленность (профиль)

«Технологии разработки программного обеспечения»

**ОТЧЁТ**

по реализации проекта для дисциплины «Базы данных»

Разработка системы управления медицинскими услугами

Преподаватель: к.ф-м.н., доцент кафедры ИТиЭО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Жуков Н. Н.)

Студенты 2 курса:

Адаменко С. С. \_\_\_\_\_\_\_\_\_\_\_

Суворов Р. М. \_\_\_\_\_\_\_\_\_\_\_

Гневнов А. Е. \_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург  
2024

Оглавление

[Ответственные 3](#_Toc169187805)

[Предметная область 3](#_Toc169187806)

[Ход выполнения нормализации 3](#_Toc169187807)

[Объяснение выбранной СУБД 4](#_Toc169187808)

[ER–диаграмма 4](#_Toc169187809)

[Исходный текст запросов 5](#_Toc169187810)

[Таблицы 5](#_Toc169187811)

[Индексы 7](#_Toc169187812)

[Триггеры и функции 7](#_Toc169187813)

[Фронт проект 8](#_Toc169187814)

[Бэкенд проект 8](#_Toc169187815)

[Оптимальность использования NoSQL баз данных для моделирования выбранной предметной области 9](#_Toc169187816)

# Ответственные

Адаменко С.С. – разработчик проекта. В обязанности входил процесс создания фронтенд части приложения. Вёл кураторскую деятельность по созданию бэкенд части приложения.

Суворов Р.М. – разработчик проекта. В обязанности входил процесс создания бэкенд части приложения.

Гневнов А.Е. – разработчик проекта. Работа с PostgreSQL.

# Предметная область

Система управления медицинскими услугами, обеспечивающая автоматизацию процесса записи на приём к врачам.

Основные функции системы включают:

− Личный кабинет: создание и редактирования персональных данных;

− Запись на приём: пользователи могут записываться на прием к врачам, а также в будущем отменять или подтверждать, что встреча прошла успешно;

− Администрирование системы: администратор может просматривать все встречи за всё время и редактировать информацию о врачах;

− Кроссплатформенность: адаптация приложения на все операционные системы и устройства;

− Навигация и фильтрация: простая навигация и фильтры

# Ход выполнения нормализации

После выделения конкретных сущностей, был сформирован список атрибутов каждой сущности:

Сущности для медицинской системы:

Пользователь (User):

Каждый пользователь является физическим лицом и имеет следующие атрибуты: полное имя, электронная почта, возраст, пол и пароль. Ключевой атрибут – `id`. Все атрибуты данной сущности обязательны к заполнению.

Врач (Doctor):

Каждый врач имеет следующие атрибуты: имя, университеты, количество пациентов, опыт, аватар, описание, специализация и расписание (список). Ключевой атрибут - `id`.

Назначение (Appointment):

Каждое назначение имеет следующие атрибуты: пользователь, врач, дата, время и статус. Ключевой атрибут - `id`. Связано с пользователем через `user\_id` (внешний ключ к сущности `Users`), с врачом через `doctor\_id` (внешний ключ к сущности `Doctors`).

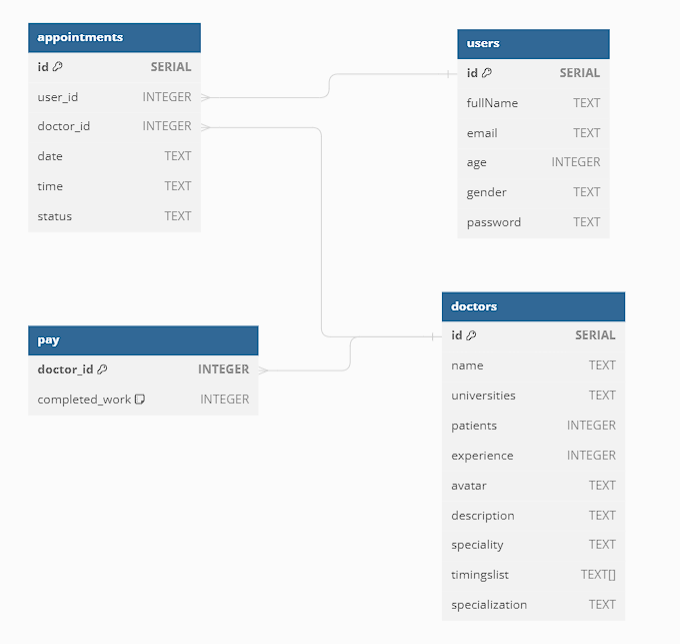
Зарплата (Pay):

Каждая зарплата имеет следующие атрибуты: выполненная работа и врач. Внешний ключ `doctor\_id.

# Объяснение выбранной СУБД

Для данного проекта в качестве СУБД был выбран сервис Supabase, которая основывается на PostgreSQL, поскольку она предоставляет удобный интерфейс для работы с базами данных и обладает широким спектром функций, необходимых для реализации проекта. Supabase поддерживает стандарт SQL, что облегчает разработку и поддержку приложения. Также Supabase предоставляет удобный API для работы с базой данных, что упрощает взаимодействие с ней из различных языков программирования. Ещё немаловажным плюсом является, что она имеет бесплатный тариф, благодаря чему мы можем сделать REST API.

# ER–диаграмма

****

# Исходный текст запросов

## Таблицы

-- Создание таблицы users

CREATE TABLE IF NOT EXISTS users (

    id SERIAL PRIMARY KEY,

    fullName TEXT,

    email TEXT,

    age INTEGER,

    gender TEXT,

    password TEXT

);

-- Создание таблицы pay

CREATE TABLE IF NOT EXISTS pay (

    doctor\_id INTEGER PRIMARY KEY REFERENCES doctors(id),

    completed\_work INTEGER DEFAULT 0

);

-- Создание таблицы doctors

CREATE TABLE IF NOT EXISTS doctors (

    id SERIAL PRIMARY KEY,

    name TEXT,

    universities TEXT,

    patients INTEGER,

    experience INTEGER,

    avatar TEXT,

    description TEXT,

    speciality TEXT,

    timingslist TEXT[],

    specialization TEXT

);

-- Создание таблицы appointments

CREATE TABLE IF NOT EXISTS appointments (

    id SERIAL PRIMARY KEY,

    user\_id INTEGER REFERENCES users(id),

    doctor\_id INTEGER REFERENCES doctors(id),

    date TEXT,

    time TEXT,

    status TEXT

);

## Индексы

CREATE INDEX ON appointments (user\_id);

CREATE INDEX ON appointments (doctor\_id);

## Триггеры и функции

Пример функции:

CREATE OR REPLACE FUNCTION increment\_completed\_work()

RETURNS TRIGGER AS $$

BEGIN

    UPDATE pay

    SET completed\_work = completed\_work + 1

    WHERE doctor\_id = NEW.doctor\_id;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

Пример Тригерра:

CREATE TRIGGER update\_completed\_work

AFTER UPDATE OF status ON appointments

FOR EACH ROW

WHEN (NEW.status = 'completed')

EXECUTE FUNCTION increment\_completed\_work();

## Фронт проект

github:



Проект:



## Бэкенд проект

github:



# Оптимальность использования NoSQL баз данных для моделирования выбранной предметной области

В случае системы управления медицинскими услугами, основной акцент обычно делается на обработку транзакционных данных, таких как записи пациентов на прием к врачам, управление расписаниями врачей и т. д.

Исходя из следующих факторов, которые следует учитывать при оценке оптимальности использования NoSQL баз данных (для примера взята MongoDB) для нашей предметной области:

1. Гибкость схемы данных: MongoDB предоставляет гибкую схему данных, известную как схема по требованию (schema on demand). Это означает, что вы можете хранить документы различных структур в одной коллекции, что может быть полезно для хранения данных о пациентах, врачах и записях на приём, которые могут иметь различные наборы атрибутов.
2. Масштабируемость: MongoDB хорошо масштабируется горизонтально благодаря шардингу и репликации. Это означает, что вы можете распределить данные по нескольким серверам для обработки больших объемов данных и повышения отказоустойчивости системы.
3. Высокая производительность: MongoDB обеспечивает высокую производительность при выполнении запросов благодаря встроенной поддержке индексов, кэшированию и эффективной обработке запросов в формате JSON.
4. Гибкие запросы и агрегации: MongoDB предоставляет мощные инструменты для выполнения запросов и агрегаций, позволяя выполнять сложные запросы и аналитику данных.
5. Поддержка ACID-транзакций: С версии MongoDB 4.0 появилась поддержка ACID-транзакций для операций чтения и записи в одной коллекции или нескольких коллекциях, что обеспечивает целостность данных в транзакционных операциях.

Исходя из этих особенностей, MongoDB может быть хорошим выбором для моделирования системы управления медицинскими услугами, особенно если необходима гибкость в схеме данных, масштабируемость и высокая производительность при работе с большими объемами данных.

