МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное образовательное
учреждение высшего образования
«ИТМО»

Факультет Программной Инженерии и компьютерной
Техники

# Разработка информационной системы для удалённого управления данными магниторезонансного томографа

Выполнили: студенты группы №P3315
Деревгин Егор
Мацюк Владимир

Санкт-Петербург
2024 г.

# Оглавление

# 1. Введение

В современном медицинском оборудовании магнитно-резонансная томография (МРТ) занимает важное место благодаря своей высокой точности и возможности создания детализированных изображений внутренних структур организма. Однако эффективное управление данными, полученными в процессе сканирования, требует использования специализированных информационных систем. Данная работа направлена на разработку информационной системы для удалённого управления данными МРТ, которая позволит автоматизировать процессы сбора, хранения, обработка медицинских данных, обеспечивая при этом безопасность и конфиденциальность информации.

# 2. Подробное текстовое описание предметной области

## 2.1. Магнитно-резонасная томография (МРТ)

Магнитно-резонансная томография (МРТ) является неинвазивным методом медицинской диагностики, использующим сильные магнитные поля и радиоволны для создания детализированных изображений внутренних структур организма. МРТ широко применяются для диагностики заболеваний центральной нервной системы, опорно-двигательного аппарата, сердечно-сосудистой системы, а также для исследования мягких тканей и органов.

### 2.2. Удалённое управление данными МРТ

Удалённое управление данными МРТ включает в себя сбор, хранение, обработку данных, полученных во время сканирования пациентов. Информационная система для такой задачи должна обеспечивать безопасный доступ к данным, а также предоставлять инструменты для визуализации результатов исследований.

### 2.3. Основные компоненты предметной области

- **Магнитно-резонансный томограф**: аппаратное обеспечение, генерирующее и собирающее данные сканирования.
- **Данные сканирования**: изображения и метаданные, связанные с процедурой сканирования (время, параметры сканирования, информация о пациенте).
- **Медицинский персонал**: врачи, радиологи и технический персонал, взаимодействующие с системой для проведения и анализа сканирований.
- **Пациенты**: субъекты, проходящие процедуру МРТ сканирования. Информационные системы: базы данных, серверы и программное обеспечение для обработки и хранения данных.
- **Безопасность и конфиденциальность**: меры по защите данных пациентов.

## 3. Назначение информационной системы и решаемые задачи

### 3.1. Назначение информационной системы

Назначение информационной системы для удалённого управления данными МРТ заключается в автоматизации процессов сбора, хранения и доступа к данным сканирований, а также в обеспечении эффективного взаимодействия медицинского персонала с данными для диагностики состояния пациентов.

### 3.2. Основные задачи системы

1) **Централизованное хранение данных**:
   - Обеспечение единого хранилища для всех данных сканирований.
2) **Удалённый доступ и управление**:
   - Позволяет медицинскому персоналу получать доступ к данным сканирований.
   - Управление параметрами сканирования и расписанием томографов

3) **Визуализация данных**:
    - Инструменты для визуализации изображений МРТ .
4) **Обеспечение безопасности и конфиденциальности**:
    - Контроль доступа к данным.

# 4. Функциональные и нефункциональные требования

## 4.1. Функциональные требования

1) **Управление пользователями**:
    - Регистрация и аутентификация пользователей.
    - Роли и права доступа (администраторы, врачи, техперсонал).
2) **Хранение данных**:
    - Хранение dicom-изображений в базе данных Orthanc.
    - Хранение данных ИС в базе данных PostgreSQL.
3) **Управление исследованиями**:
    - Планирование и расписание исследований.
    - Настройка параметров сканирования:
        (a) **TR** (время повторения) - число
        (b) **TE** (время эхо) - число
        (c) **FOV** (поле обзора) - число
        (d) **Матричный размер** - пара чисел
        (e) **Толщина среза** - число
        (f) **Число накоплений** - число
4) **Доступ к данным**:
    - Просмотр и поиск исследований по критериям: пациент, дата.
    - Загрузка и скачивание изображений.
5) **Визуализация**:
    - Окно для отображения изображений.
    - Инструменты для визуализации изображений:
        (a) **Перемещение** - позволяет перемещать линейно изображения в выбранном окне относительно центра окна кнопкой мыши.
        (b) **Вращение** - позволяет вращаться изображения в выбранном окне относительно центра окна кнопкой мыши.
        (c) **Окно просмотра** - позволяет регулировать параметры отображения изображений в окнах (яркость, контраст). Использования инструмента окно просмотра не происходит изменение значений в пикселях изображения, меняется только диапазон отображения градаций серого на мониторе.

(d) **Масштаб** - позволяет приближать и отдалять изображение в окне.
(e) **Аннотации** - позволяет скрывать и показывать текстовую информацию на изображениях.

6) **Мониторинг оборудования**:
    - Возможность присваивания меток(комментариев) о состоянии МРТ.

### 4.2. Нефункциональные требования

1) **Производительность**:
    - Время отклика системы при взаимодействие пользователя с ней не должно превышать 10 секунд.
    - Возможность одновременной работы нескольких (более 20) пользователей.
2) **Безопасность**:
    - Использование алгоритмов шифрования данных при работе с данными пользователей.
    - Механизмы предотвращения неавторизируемого доступа.
3) **Локализация**:
    - Интерфейс пользователя должен поддерживать русский и английский языки.
4) **Совместимость**:
    - Доступ к пользовательскому интерфейсу должен быть доступен при использовании браузеров Google Chrome (от 126 версии), Firefox (от 130 версии).
5) **Безопасность**:
    - Шифрование и защита данных от несанкционированного доступа.
    - Аудит доступа и действий пользователей.

## 5. Модели основных прецедентов и их описание

Для информационной системы удалённого управления данными МРТ можно выделить следующие основные прецеденты (use cases):

### 5.1. Прецедент: Авторизация пользователя

#### 5.1.1. Описание

- Пользователь системы вводит свои учетные данные для доступа к системе. Система проверяет корректность данных и предоставляет доступ в зависимости от роли пользователя.

#### 5.1.2. Акторы

- Пользователь (администратор, врач, техперсонал)

#### 5.1.3. Основной поток событий

1) Пользователь открывает страницу авторизации.
2) Вводит логин и пароль.
3) Нажимает кнопку "Войти".
4) Система проверяет данные и определяет роль пользователя.
5) Предоставляет доступ к соответствующим функциям системы.

### 5.2. Прецедент: Просмотр данных исследования

#### 5.2.1. Описание

- Врач или другой уполномоченный пользователь просматривает результаты МРТ исследования пациента.

#### 5.2.2. Акторы

- Врач
- Администратор

#### 5.2.3. Основной поток событий

1) Пользователь входит в систему
2) Переходит в раздел "Сканирования".
3) Выполняет поиск по критериям (по имени пациента, дате, по имени врача).
4) Выбирает нужное сканирование из списка.
5) Открывает изображение и просматривает результаты.

### 5.3. Прецедент: Просмотр данных исследования

#### 5.3.1. Описание

- Врач вручную загружает данные исследования в центральное хранилище.

#### 5.3.2. Акторы

- Врач

#### 5.3.3. Основной поток событий

1) Пользователь входит в систему.
2) Врач заполняет атрибуты исследования (время, пациент).
3) Врач выбирает файлы исследования со своего устройства.
4) Врач нажимает кнопку "Загрузить".
5) Исследование сохраняется в хранилище и врач получает сообщение об успешной загрузке.

### 5.4. Прецедент: Управление расписанием сканирований

#### 5.4.1. Описание

- Администратор или уполномоченный пользователь планирует и управляет расписанием работы томографа.

#### 5.4.2. Акторы

- Врач
- Администратор
- Техперсонал

#### 5.4.3. Основной поток событий

1) Пользователь входит в систему.
2) Переходит в раздел "Расписание".
3) Добавляет новое сканирование, указывая врача, время, пациента и параметры сканирования.
4) Сохраняет изменения.
5) Система обновляет расписание и сообщает об успешном изменении расписания.

## 6. Предложение архитектуры будущей системы

### 6.1. Общая схема архитектуры

Архитектура информационной системы для удалённого управления данными томографа будет основана на многослойной архитектуре с разделением на фронтенд, бэкенд и хранилища данных. Учитывая необходимость демонстрации на сервере helios, предлагается следующая архитектура:

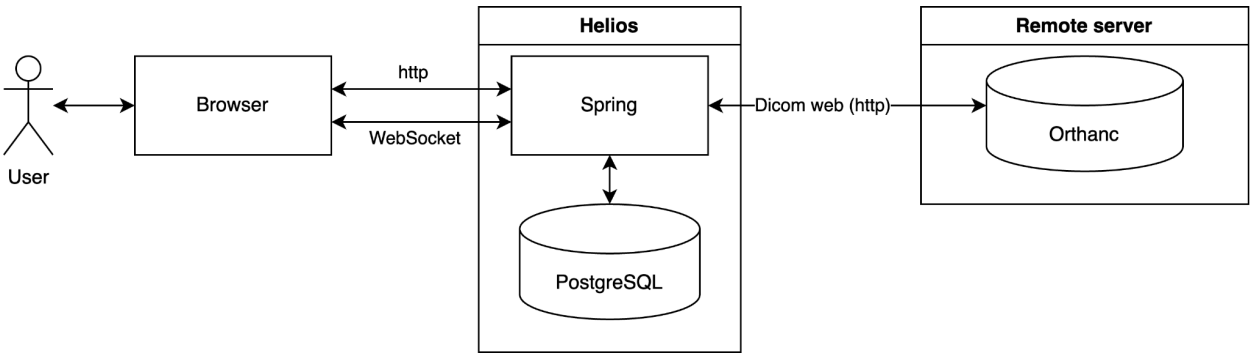



Рис. 6.1: Схема

1) **Клиентский уровень (Frontend)**:
   - Технологии: Svelte.
   - Обеспечивает интерактивный пользовательский интерфейс.

- Взаимодействует с сервером через REST API.

2) **Серверный уровень (Backend)**:
   - Технологии: Spring.
   - Реализует бизнес-логику приложения.
   - Обеспечивает взаимодействие с хранилищами данных и пользовательским интерфейсом.
3) **Уровень данных (DataBase)**:
   - Технологии: PostgreSQL, Orthanc.
   - Хранение данных исследований, пользователей, расписаний и т.д.

## 6.2. Детализация компонентов

### 6.2.1. Frontend (Svelte)

- **Компоненты интерфейса**: Разработка модульных компонентов для различных частей системы (авторизация, просмотр исследований, управление расписанием).
- **Маршрутизация**: Реализация маршрутов для навигации между страницами.
- **Взаимодействие с API**: Использование Axios для общения с бэкендом.

### 6.2.2. Backend (Spring)

- **Контроллеры**: Обработка HTTP-запросов и маршрутизация к соответствующим сервисам.
- **Сервисы**: Реализация бизнес-логики (управление пользователями, сканированиями, расписанием и т.д.).
- **Репозитории**: Взаимодействие с базой данных через Spring Data JPA.
- **Безопасность**: Настройка Spring Security для обеспечения аутентификации и авторизации.

### 6.2.3. База данных (PostgreSQL)

- **Моделирование данных**: Создание схемы базы данных, включающей таблицы для пользователей, сканирований, расписаний, ролей.
- **Оптимизация**: Индексация и нормализация для обеспечения производительности и целостности данных.

### 6.2.4. База данных (Orthanc)

- **Хранение и доступ к dicom-изображениям**: Хранение dicom изображений при помощи стандартных возможностей Orthanc.

#### 6.2.5. Безопасность и соответствие требованиям

- **Шифрование**: Шифрование конфиденциальных данных в базе данных.
- **Аутентификация и авторизация**: Реализация системы прав доступа

#### 6.2.6. Развертывание и обслуживание

- **Сервер helios**: Настройка среды для развертывания приложения (установка необходимых зависимостей, настройка сервера приложений).
- **Дополнительных Linux-Сервер**: Настройка дополнительного сервера для развертывания Orthanc.

### 6.3. Выбор технологий и фреймфорков

#### 6.3.1. Frontend

- **Svelte**: Выбран за его гибкость, удобство и любовь разработчиков. Позволяет создавать динамичные и отзывчивые пользовательские интерфейсы

#### 6.3.2. Backend

- **Spring**: Обеспечивает мощные возможности для разработки веб-приложений интеграции с базами данных и реализации безопасных API.

#### 6.3.3. База данных

- **PostgreSQL**: Надёжная и масштабируемая реляционная СУБД с богатым набором функций, подходящая для хранения различных данных.
- **Orthanc**: Широко используемое решение для хранения dicom-файлов.

#### 6.3.4. Дополнительные технологии

- **JWT** (JSON Web Tokens): Для реализации безопасной аутентификации и авторизации.
- **Axios**: Для осуществления HTTP-запросов с фронтенда на бэкенд.

# 7. Инфологическая модель

## 7.1. Сущности и их атрибуты

1) **Person (Человек)**
   - **ID** (PK) — уникальный идентификатор
   - **FirstName** — имя

- **LastName** — фамилия
- **MiddleName** — отчество
- **DateOfBirth** — дата рождения
- **Gender** — пол
- **PhoneNumber** — номер телефона
- **Email** — электронная почта
- **Address** — адрес

2) **User (Пользователь)**
   - **ID** (PK) — уникальный идентификатор пользователя
   - **PersonID** (FK) — ссылка на **Person(ID)**
   - **Username** — логин
   - **PasswordHash** — хэш пароля
   - **DateRegistered** — дата регистрации

3) **Role (Роль)**
   - **ID** (PK) — уникальный идентификатор роли
   - **RoleName** — название роли
   - **Description** — описание роли

4) **UserRole (Роль пользователя)**
   - **UserID** (PK, FK) — ссылка на **User(ID)**
   - **RoleID** (PK, FK) — ссылка на **Role(ID)**

5) **Patient (Пациент)**
   - **ID** (PK) — уникальный идентификатор пациента
   - **PersonID** (FK) — ссылка на **Person(ID)**
   - **InsurancePolicyNumber** — номер страхового полиса

6) **Device (Оборудование)**
   - **ID** (PK) — уникальный идентификатор оборудования
   - **DeviceSN** — серийный номер устройства

- **Location** — местоположение
- **Status** — статус устройства

7) **Study (Исследование)**
   - **ID** (PK) — уникальный идентификатор исследования
   - **PatientID** (FK) — ссылка на **Patient(ID)**
   - **UserID** (FK) — ссылка на **User(ID)** (пользователь, проводивший исследование)
   - **DeviceID** (FK) — ссылка на **Device(ID)**
   - **DicomRef** — URL для доступа к dicom изображениям в Orthanc
   - **Status** — статус исследования (запланировано, отменено, проведено)
   - **Notes** — заметки

8) **DeviceComment (Комментарий к оборудованию)**
   - **ID** (PK) — уникальный идентификатор комментария
   - **DeviceID** (FK) — ссылка на **Device(ID)**
   - **UserID** (FK) — ссылка на **User(ID)**
   - **CommentText** — текст комментария
   - **Timestamp** — дата и время комментария

9) **Schedule (Расписание)**
   - **ID** (PK) — уникальный идентификатор записи расписания
   - **StartTime** — запланированная дата и время начала исследования
   - **EndTime** — запланированная дата и время конца исследования
   - **StudyID** (FK) — ссылка на **Study(ID)**
   - **ScheduledByUserID** (FK) — ссылка на **User(ID)**
   - **Comments** — комментарии

10) **AuditLog (Журнал действий)**
    - **ID** (PK) — уникальный идентификатор записи журнала
    - **UserID** (FK) — ссылка на **User(ID)**
    - **ActionType** — тип действия

- **Entity** — тип сущности
- **EntityID** — идентификатор сущности
- **Timestamp** — время действия
- **Details** — подробности действия

## 7.2. Связи между сущностями

1) **Person** 1 — 1 **User**
    - **Связь**: Один к одному
    - **Описание**: Каждый пользователь соответствует одному человеку. Связь осуществляется через поле **User.PersonID**.

2) **Person** 1 — 1 **Patient**
    - **Связь**: Один к одному
    - **Описание**: Каждый пациент соответствует одному человеку. Связь через поле **Patient.PersonID**.

3) **User** M — N **Role** (через **UserRole**)
    - **Связь**: Многие ко многим
    - **Описание**: Пользователь может иметь несколько ролей, роль может быть присвоена нескольким пользователям. Связь реализована через промежуточную таблицу **UserRole**.

4) **Patient** 1 — M **Study**
    - **Связь**: Один ко многим
    - **Описание**: Один пациент может иметь множество исследований. Связь через поле **Study.PatientID**.

5) **User** 1 — M **Study**
    - **Связь**: Один ко многим

- **Описание**: Один пользователь (врач или техник) может проводить множество исследований. Связь через поле **Study.UserID**.

6) **Device** 1 — M **Study**
   - **Связь**: Один ко многим
   - **Описание**: Одно устройство может использоваться во многих исследованиях. Связь через поле **Study.DeviceID**.

7) **Device** 1 — M **DeviceComment**
   - **Связь**: Один ко многим
   - **Описание**: Одно устройство может иметь множество комментариев. Связь через поле **DeviceComment.DeviceID**.

8) **User** 1 — M **DeviceComment**
   - **Связь**: Один ко многим
   - **Описание**: Один пользователь может оставлять множество комментариев об оборудовании. Связь через поле **DeviceComment.UserID**.

9) **User** 1 — M **Schedule**
   - **Связь**: Один ко многим
   - **Описание**: Один пользователь может создавать множество записей в расписании. Связь через поле **Schedule.ScheduledByUserID**.

10) **Study** 1 — 0..1 **Schedule**
    - **Связь**: Один к нулю или одному
    - **Описание**: Одно исследование может быть связано с одной записью в расписании или не иметь связи. Связь через поле **Schedule.StudyID**.

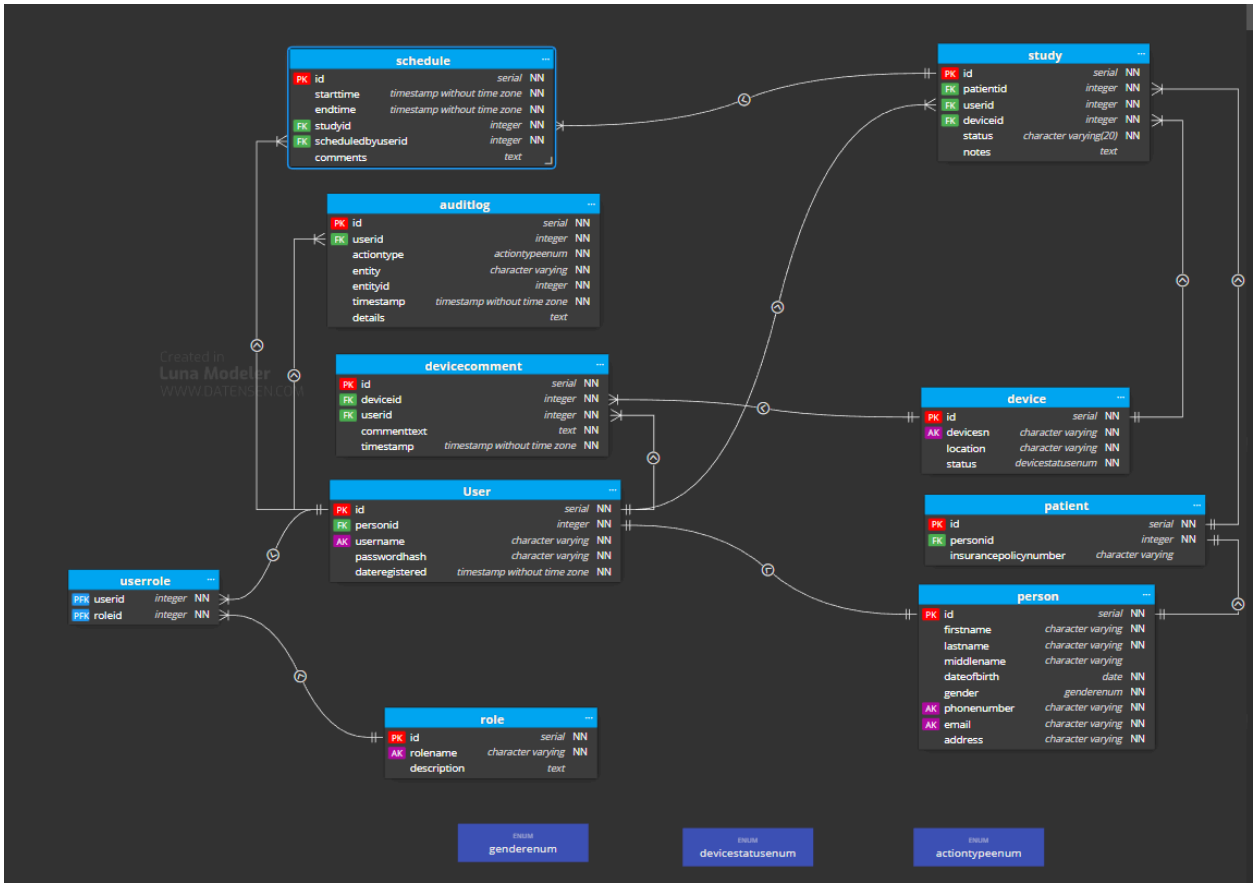
11) **User** 1 — M **AuditLog**
    - **Связь**: Один ко многим

- **Описание**: Один пользователь может иметь множество записей в журнале действий. Связь через поле **AuditLog.UserID**.

## 7.3. Диаграмма (текстовое представление)

[Person] 1 --- 1 [User]
[Person] 1 --- 1 [Patient]
[User] M ---< [UserRole] >--- M [Role]
[Patient] 1 --- M [Study]
[User] 1 --- M [Study]
[Device] 1 --- M [Study]
[Device] 1 --- M [DeviceComment]
[User] 1 --- M [DeviceComment]
[User] 1 --- M [Schedule]
[Study] 1 --- 0..1 [Schedule]
[User] 1 --- M [AuditLog]

# 8. Даталогическая модель



# 9. Реализация СУБД в PostgreSQL

```
-- Drop existing tables if needed (optional)
-- Be cautious with DROP statements in a production environment
DROP TABLE IF EXISTS AuditLog;
DROP TABLE IF EXISTS Schedule;
DROP TABLE IF EXISTS DeviceComment;
DROP TABLE IF EXISTS Study;
DROP TABLE IF EXISTS Device;
DROP TABLE IF EXISTS Patient;
DROP TABLE IF EXISTS UserRole;
DROP TABLE IF EXISTS Role;
DROP TABLE IF EXISTS "User";
DROP TABLE IF EXISTS Person;
DROP TYPE IF EXISTS GenderEnum;
DROP TYPE IF EXISTS DeviceStatusEnum;
DROP TYPE IF EXISTS ActionTypeEnum;

-- Create ENUM types
CREATE TYPE GenderEnum AS ENUM ('M', 'F', 'O');
CREATE TYPE DeviceStatusEnum AS ENUM ('Работает', 'Неисправно', 'В обслуживании');
CREATE TYPE ActionTypeEnum AS ENUM ('Создание', 'Изменение', 'Удаление', 'Вход', 'Выход');

-- Create the Person table
CREATE TABLE Person (
  ID SERIAL PRIMARY KEY,
  FirstName VARCHAR NOT NULL,
  LastName VARCHAR NOT NULL,
  MiddleName VARCHAR,
  DateOfBirth DATE NOT NULL,
  Gender GenderEnum NOT NULL,
  PhoneNumber VARCHAR UNIQUE NOT NULL,
  Email VARCHAR UNIQUE NOT NULL,
  Address VARCHAR NOT NULL
);

-- Create the User table
CREATE TABLE "User" (
  ID SERIAL PRIMARY KEY,
  PersonID INTEGER NOT NULL UNIQUE REFERENCES Person(ID) ON DELETE CASCADE,
```

```
  Username VARCHAR UNIQUE NOT NULL,
  PasswordHash VARCHAR NOT NULL,
  DateRegistered TIMESTAMP NOT NULL DEFAULT NOW()
);

-- Create the Role table
CREATE TABLE Role (
  ID SERIAL PRIMARY KEY,
  RoleName VARCHAR UNIQUE NOT NULL,
  Description TEXT
);

-- Create the UserRole table (Many-to-Many relationship between User and Role)
CREATE TABLE UserRole (
  UserID INTEGER NOT NULL  REFERENCES "User"(ID) ON DELETE CASCADE,
  RoleID INTEGER NOT NULL  REFERENCES Role(ID) ON DELETE CASCADE,
  PRIMARY KEY (UserID, RoleID)
);

-- Create the Patient table
CREATE TABLE Patient (
  ID SERIAL PRIMARY KEY,
  PersonID INTEGER NOT NULL UNIQUE REFERENCES Person(ID) ON DELETE CASCADE,
  InsurancePolicyNumber VARCHAR
);

-- Create the Device table
CREATE TABLE Device (
  ID SERIAL PRIMARY KEY,
  DeviceSN VARCHAR UNIQUE NOT NULL,
  Location VARCHAR NOT NULL,
  Status DeviceStatusEnum NOT NULL
);

-- Create the Study table
CREATE TABLE Study (
  ID SERIAL PRIMARY KEY,
  PatientID INTEGER NOT NULL REFERENCES Patient(ID) ON DELETE CASCADE,
  UserID INTEGER NOT NULL  REFERENCES "User"(ID) ON DELETE SET NULL, -- User who performed the
study
  DeviceID INTEGER NOT NULL  REFERENCES Device(ID) ON DELETE SET NULL,
  Status VARCHAR(20) NOT NULL DEFAULT 'Planned',
```

```
  CONSTRAINT chk_status CHECK (Status IN ('Planned', 'Canceled', 'Successed'))
  Notes TEXT
);

-- Create the DeviceComment table
CREATE TABLE DeviceComment (
  ID SERIAL PRIMARY KEY,
  DeviceID INTEGER NOT NULL REFERENCES Device(ID) ON DELETE CASCADE,
  UserID INTEGER NOT NULL REFERENCES "User"(ID) ON DELETE CASCADE,
  CommentText TEXT NOT NULL,
  Timestamp TIMESTAMP NOT NULL DEFAULT NOW()
);

-- Create the Schedule table
CREATE TABLE Schedule (
  ID SERIAL PRIMARY KEY,
  StartTime TIMESTAMP NOT NULL,
  EndTime TIMESTAMP NOT NULL,
  StudyID INTEGER NOT NULL  REFERENCES Study(ID) ON DELETE SET NULL,
  ScheduledByUserID INTEGER NOT NULL REFERENCES "User"(ID) ON DELETE CASCADE,
  Comments TEXT
);

-- Create the AuditLog table
CREATE TABLE AuditLog (
  ID SERIAL PRIMARY KEY,
  UserID INTEGER NOT NULL REFERENCES "User"(ID) ON DELETE CASCADE,
  ActionType ActionTypeEnum NOT NULL,
  Entity VARCHAR NOT NULL,
  EntityID INTEGER NOT NULL,
  Timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
  Details TEXT
);
```

# 10. Обеспечение целостности данных при помощи средств языка DDL и триггеров.

```
-- Создание функции триггера для проверки конфликтов в расписании
CREATE OR REPLACE FUNCTION check_schedule_conflict()
RETURNS TRIGGER AS $$
DECLARE
  conflicting_count INTEGER;
  new_device_id INTEGER;
BEGIN
  IF NEW.StudyID IS NULL THEN
    -- Если StudyID не задан, пропускаем проверку
    RETURN NEW;
  END IF;
  -- Получаем DeviceID из таблицы Study
  SELECT DeviceID INTO new_device_id FROM Study WHERE ID = NEW.StudyID;
  IF new_device_id IS NULL THEN
    -- Если DeviceID не найден, пропускаем проверку
    RETURN NEW;
  END IF;
  -- Проверяем на наличие конфликтов в расписании
  SELECT COUNT(*) INTO conflicting_count
  FROM Schedule S
  JOIN Study ST ON S.StudyID = ST.ID
  WHERE
    ST.DeviceID = new_device_id
    AND S.ID <> NEW.ID
    AND (
      (NEW.StartTime, NEW.EndTime) OVERLAPS (S.StartTime, S.EndTime)
    );

  IF conflicting_count > 0 THEN
    RAISE EXCEPTION 'Конфликт в расписании: устройство уже занято в это время.';
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Создание триггера для вызова функции проверки конфликтов
CREATE TRIGGER trigger_check_schedule_conflict
```

```
BEFORE INSERT OR UPDATE ON Schedule
FOR EACH ROW EXECUTE FUNCTION check_schedule_conflict();
```

# 11. Скрипт заполнения базы тестовыми данными.

```
-- Заполнение таблицы Person
INSERT INTO Person (FirstName, LastName, MiddleName, DateOfBirth, Gender, PhoneNumber, Email, Address)
VALUES
-- Пользователи
('Иван', 'Иванов', 'Иванович', '1985-05-20', 'M', '+71234567890', 'ivanov@example.com', 'ул. Ленина, д. 1'),
('Мария', 'Петрова', 'Сергеевна', '1990-08-15', 'F', '+79876543210', 'petrova@example.com', 'ул. Пушкина, д. 2'),
('Алексей', 'Смирнов', 'Алексеевич', '1978-12-30', 'M', '+79991234567', 'smirnov@example.com', 'ул. Мира, д. 3'),
-- Пациенты
('Елена', 'Кузнецова', 'Игоревна', '1995-03-10', 'F', '+79001234567', 'kuznetsova@example.com', 'ул. Советская, д. 4'),
('Дмитрий', 'Попов', 'Владимирович', '1982-11-25', 'M', '+79112223344', 'popov@example.com', 'ул. Садовая, д. 5');

-- Заполнение таблицы Role
INSERT INTO Role (RoleName, Description)
VALUES
('Администратор', 'Администратор системы'),
('Врач', 'Медицинский работник'),
('Техперсонал', 'Технический персонал');

-- Заполнение таблицы User
-- Предполагается, что ID пользователей в таблице Person начинаются с 1
INSERT INTO "User" (PersonID, Username, PasswordHash, DateRegistered)
VALUES
(1, 'ivanov', 'hashed_password1', NOW()),
(2, 'petrova', 'hashed_password2', NOW()),
(3, 'smirnov', 'hashed_password3', NOW());

-- Заполнение таблицы UserRole
INSERT INTO UserRole (UserID, RoleID)
VALUES
(1, 1), -- Иванов - Администратор
(2, 2), -- Петрова - Врач
(3, 3); -- Смирнов - Техперсонал

-- Заполнение таблицы Patient
-- ID пациентов в таблице Person начинаются с 4
INSERT INTO Patient (PersonID, InsurancePolicyNumber)
VALUES
(4, '1234567890'),
```

```
(5, '0987654321');

-- Заполнение таблицы Device
INSERT INTO Device (DeviceSN, Location, Status)
VALUES
('SN-001', 'Клиника №1', 'Работает'),
('SN-002', 'Клиника №2', 'В обслуживании');

-- Заполнение таблицы Study
INSERT INTO Study (PatientID, UserID, DeviceID, Params, Notes)
VALUES
(1, 2, 1, '{"TR": 500, "TE": 20, "FOV": 250, "MatrixSize": [256, 256], "SliceThickness": 5, "NumberOfAverages": 2}',
'Исследование головного мозга'),
(2, 2, 2, '{"TR": 600, "TE": 25, "FOV": 300, "MatrixSize": [512, 512], "SliceThickness": 3, "NumberOfAverages": 1}',
'Исследование позвоночника');

-- Заполнение таблицы DeviceComment
INSERT INTO DeviceComment (DeviceID, UserID, CommentText, Timestamp)
VALUES
(1, 3, 'Проведено плановое обслуживание', NOW() - INTERVAL '1 day'),
(2, 3, 'Необходимо заменить деталь', NOW());

-- Заполнение таблицы Schedule
INSERT INTO Schedule (StartTime, EndTime, StudyID, ScheduledByUserID, Comments)
VALUES
(NOW() + INTERVAL '1 hour', NOW() + INTERVAL '2 hours', 1, 2, 'Запланировано исследование для пациента
Иванова'),
(NOW() + INTERVAL '3 hours', NOW() + INTERVAL '4 hours', 2, 2, 'Запланировано исследование для пациента
Петрова');

-- Заполнение таблицы AuditLog
INSERT INTO AuditLog (UserID, ActionType, Entity, EntityID, Timestamp, Details)
VALUES
(1, 'Создание', 'User', 1, NOW() - INTERVAL '2 days', 'Создан пользователь Иванов'),
(1, 'Создание', 'User', 2, NOW() - INTERVAL '2 days', 'Создан пользователь Петрова'),
(1, 'Создание', 'User', 3, NOW() - INTERVAL '2 days', 'Создан пользователь Смирнов'),
(2, 'Создание', 'Study', 1, NOW() - INTERVAL '1 day', 'Создано исследование для пациента Кузнецова'),
(2, 'Создание', 'Study', 2, NOW(), 'Создано исследование для пациента Попов');
```

## 12. Предложить pl/pgsql-функции и процедуры, для выполнения критически важных запросов (которые потребуются при последующей реализации прецедентов).

Процедура просмотра данных исследования

```
CREATE OR REPLACE FUNCTION get_study_data(p_study_id INTEGER)
RETURNS TABLE(
  study_id INTEGER,
  patient_full_name VARCHAR,
  device_sn VARCHAR,
  params JSON,
  notes TEXT
) AS $$
BEGIN
  RETURN QUERY
  SELECT
    s.ID,
    CONCAT(p.LastName, ' ', p.FirstName, ' ', COALESCE(p.MiddleName, '')) AS patient_full_name,
    d.DeviceSN,
    s.Params,
    s.Notes
  FROM Study s
  JOIN Patient pt ON pt.ID = s.PatientID
  JOIN Person p ON p.ID = pt.PersonID
  LEFT JOIN Device d ON d.ID = s.DeviceID
  WHERE s.ID = p_study_id;
END;
$$ LANGUAGE plpgsql;
```

Функция для обновления статуса устройства

```
CREATE OR REPLACE FUNCTION update_device_status(
  p_device_id INTEGER,
  p_new_status DeviceStatusEnum,
  p_user_id INTEGER,
  p_comment TEXT
)
RETURNS VOID AS $$
BEGIN
```

```
  UPDATE Device
  SET Status = p_new_status
  WHERE ID = p_device_id;

  IF p_comment IS NOT NULL THEN
     INSERT INTO DeviceComment (DeviceID, UserID, CommentText)
     VALUES (p_device_id, p_user_id, p_comment);
  END IF;
END;
$$ LANGUAGE plpgsql;
```

## 13. Создание индексов на основе анализа использования базы данных в контексте описанных на первом этапе прецедентов.

1) **Уникальный индекс на поле Username в таблице "User"**
   a) **Описание:** Ускоряет процесс авторизации пользователей. При входе в систему поиск пользователя происходит по полю Username, и наличие уникального индекса позволяет быстро находить записи.
   b) **SQL-запрос:**
   c) CREATE UNIQUE INDEX idx_user_username ON "User" (Username);

---

2) **Добавление составного индекса на поля StartTime и EndTime в таблице Schedule**
   a) **Описание:** Ускоряет операции по поиску расписания по дате и времени. Это критично при управлении расписанием сканирований и проверке доступности томографа.
   b) **SQL-запрос:**
   c) CREATE INDEX idx_schedule_starttime_endtime ON Schedule (StartTime, EndTime);

---

3) **Добавление индекса на поле InsurancePolicyNumber в таблице Patient**
   a) **Описание:** Ускоряет операции поиска пациентов по номеру страхового полиса, что улучшает процессы идентификации пациента.
   b) **SQL-запрос:**

c) CREATE INDEX idx_patient_insurancepolicynumber ON Patient (InsurancePolicyNumber);

## 14. Диаграмма классов архитектуры системы