Тестовое задание для Fullstack разработчика (подробное)

Система управления пациентами и визитами (МИС)

Описание

Создать систему управления пациентами с возможностью ведения учета их визитов к врачу. Система должна включать базовый функционал для добавления, просмотра и редактирования данных пациентов и их визитов.

Требования к реализации

Backend (NestJS + Prisma + PostgreSQL):

- Создать API для управления пациентами и визитами
- Реализовать CRUD операции для сущностей Patient и Visit
- Добавить валидацию данных с помощью class-validator
- Настроить Prisma схему для PostgreSQL
- Реализовать связи между пациентами и визитами
- Добавить фильтрацию визитов по датам и статусам

Frontend (React + Zustand + TypeScript):

- Создать интерфейс для отображения списка пациентов
- Реализовать формы добавления и редактирования пациента
- Создать страницу детального просмотра пациента с историей визитов
- Добавить форму создания нового визита
- Использовать Zustand для управления состоянием
- Реализовать базовый поиск по пациентам

Модели данных:

```
interface Patient {
  id: string;
  firstName: string;
  lastName: string;
  dateOfBirth: Date;
  phoneNumber: string;
  email?: string;
  createdAt: Date;
  updatedAt: Date;
```

```
  visits?: Visit[];
}

interface Visit {
  id: string;
  patientId: string;
  visitDate: Date;
  diagnosis: string;
  treatment: string;
  status: 'scheduled' | 'completed' | 'cancelled';
  notes?: string;
  createdAt: Date;
  updatedAt: Date;
  patient?: Patient;
}
```

Основные страницы:

1. **Список пациентов** - таблица с поиском и пагинацией
2. **Карточка пациента** - детальная информация + история визитов
3. **Добавление/редактирование пациента** - форма с валидацией
4. **Добавление визита** - форма создания нового визита
5. **Календарь визитов** (опционально) - просмотр запланированных визитов

Дополнительные требования:

- Использовать UI-библиотеку shadcn/ui
- Реализовать обработку ошибок и loading состояний
- Добавить базовую пагинацию для списков
- Настроить Docker для развертывания БД
- Время выполнения: 3-5 дней

API эндпоинты (минимальный набор):

Пациенты:

- `GET /patients` - получить список пациентов с пагинацией
- `GET /patients/:id` - получить пациента по ID с визитами
- `POST /patients` - создать нового пациента
- `PUT /patients/:id` - обновить пациента
- `DELETE /patients/:id` - удалить пациента
- `GET /patients/search?q=query` - поиск пациентов

Визиты:

- `GET /visits` - получить список визитов с фильтрацией
- `GET /visits/:id` - получить визит по ID
- `POST /visits` - создать новый визит
- `PUT /visits/:id` - обновить визит
- `DELETE /visits/:id` - удалить визит
- `GET /patients/:id/visits` - получить визиты пациента

Минимальный функционал для MVP:

1. Добавление/редактирование пациентов
2. Просмотр списка пациентов с поиском
3. Просмотр карточки пациента
4. Добавление/редактирование визитов
5. Просмотр истории визитов пациента
6. Базовая валидация форм
7. Обработка ошибок

Опциональный функционал (если останется время):

- Фильтрация визитов по статусам и датам
- Статистика по визитам
- Экспорт данных
- Unit-тесты для критичных методов
- Более сложная UI с таблицами и модальными окнами

Стек и инструменты:

- **Backend**: NestJS, Prisma, PostgreSQL, class-validator
- **Frontend**: React, Zustand, TypeScript, любая UI-библиотека
- **Инфраструктура**: Docker для БД, Git для версионирования

Инструкции по выполнению:

1. Создать GitHub репозиторий
2. Структура проекта: monorepo или отдельные папки client/server
3. Добавить README с инструкциями по запуску
4. Предоставить docker-compose.yml для БД
5. Заполнить БД тестовыми данными (seed)