### **Тестовое задание для Node.js разработчика (Nest.js + Prisma)**

#### **Технические требования:**

1. **Фреймворк:** Использовать Nest.js.
2. **ORM:** Использовать Prisma для взаимодействия с базой данных.
3. **База данных:** PostgreSQL.
4. **Аутентификация и авторизация:** JWT для аутентификации и ролевая модель доступа.
5. **Валидация данных:** На стороне сервера.

#### **Задачи:**

1. **Инициализация проекта:**
   * Настройка базового Nest.js проекта.
   * Интеграция Prisma и настройка базы данных PostgreSQL.
2. **Создание схемы базы данных:**
   * Определить модели в Prisma:
     + User: пользователи системы.
       - id: уникальный идентификатор.
       - email: электронная почта.
       - password: хэшированный пароль.
       - role: роль пользователя (admin или user).
       - createdAt: дата создания.
       - updatedAt: дата обновления.
     + Post: записи, созданные пользователями.
       - id: уникальный идентификатор.
       - title: заголовок записи.
       - content: текст записи.
       - userId: внешний ключ, связанный с моделью User.
       - createdAt: дата создания.
       - updatedAt: дата обновления.
   * Реализовать связь между пользователями и записями (один ко многим).
3. **Реализация API эндпоинтов:**
   * **Пользователи:**
     + POST /auth/register: регистрация нового пользователя.
     + POST /auth/login: вход в систему, возврат JWT.
     + GET /users: список всех пользователей (доступен только для админов).
     + GET /users/:id: профиль пользователя (доступен для админов и самого пользователя).
     + PUT /users/:id: обновление профиля (доступно для админа и самого пользователя).
     + DELETE /users/:id: удаление пользователя (доступно для админов и самого пользователя).
   * **Записи:**
     + POST /posts: создание новой записи (доступно для аутентифицированных пользователей).
     + GET /posts: список всех записей (доступен всем пользователям).
     + GET /posts/:id: детальная информация о записи (доступна всем пользователям).
     + PUT /posts/:id: обновление записи (доступно для автора записи).
     + DELETE /posts/:id: удаление записи (доступно для админов и автора записи).
4. **Дополнительные задачи (опционально):**
   * Написание unit и e2e тестов.
   * Добавление функциональности фильтрации и пагинации для GET /posts и GET /users.
   * Развертывание проекта на облачной платформе, например, Heroku.

#### **Критерии оценки:**

* Качество кода и архитектура приложения.
* Соблюдение REST принципов и понимание веб-безопасности.
* Работа с базой данных и правильная реализация связей.
* Эффективность аутентификации и авторизации.

#### **Сдача работы:**

Проект должен быть представлен в виде репозитория на GitHub с README файлом, содержащим инструкции по установке и запуску.