**План выполнения тестового задания: API для коротких URL**

**1. Определение технологии и стека**

* **Backend**: NestJS (TypeScript), PostgreSQL, TypeORM
* **Frontend**: React (TypeScript), Vite, Tailwind CSS
* **Контейнеризация**: Docker, Docker Compose
* **Тестирование**: Jest (Backend), React Testing Library (Frontend)

**2. Разработка Backend**

**2.1 Инициализация проекта**

* Создать новый проект NestJS (nest new backend)
* Установить зависимости: TypeORM, PostgreSQL, dotenv, class-validator, class-transformer

**2.2 Создание структуры проекта**

* **Модули**:
  + UrlModule: управление сокращёнными URL
  + AnalyticsModule: сбор статистики
* **Сущности (Entities)**:
  + UrlEntity: id, originalUrl, shortUrl, alias, expiresAt, createdAt, clickCount
  + AnalyticsEntity: id, shortUrl, ip, accessedAt

**2.3 Реализация API**

1. **Создание короткой ссылки (POST /shorten)**
   * Проверка корректности URL
   * Генерация shortUrl (если alias не указан)
   * Сохранение в БД
2. **Переадресация (GET /{shortUrl})**
   * Поиск shortUrl
   * Логирование перехода (IP, дата)
   * Редирект на originalUrl
3. **Получение информации (GET /info/{shortUrl})**
   * Возвращает информацию о ссылке (originalUrl, createdAt, clickCount)
4. **Удаление ссылки (DELETE /delete/{shortUrl})**
   * Удаляет запись из БД
5. **Аналитика (GET /analytics/{shortUrl})**
   * Количество переходов
   * Последние 5 IP-адресов

**2.4 Тестирование**

* Написать тесты для API (Jest, Supertest)
* Проверка создания ссылки и редиректа

**3. Разработка Frontend**

**3.1 Инициализация проекта**

* Создать новый проект (vite create frontend)
* Установить зависимости: React, Axios, React Router, TailwindCSS, Zustand (стейт-менеджмент)

**3.2 Создание компонентов**

1. **Главная страница**
   * Форма ввода URL
   * Кнопка "Сократить"
2. **Список ссылок**
   * Таблица с данными (сокращённая ссылка, оригинальная, клики, кнопки "Удалить" и "Информация")
3. **Страница информации**
   * Данные о ссылке + аналитика переходов

**3.3 Подключение API**

* Использование Axios для запросов
* Обработка ошибок

**3.4 Тестирование**

* Тестирование компонентов (React Testing Library)

**4. Контейнеризация**

**4.1 Написание Dockerfile**

* Для Backend (NestJS + Node.js)
* Для Frontend (React + Nginx)
* Для PostgreSQL

**4.2 Docker Compose**

* Запуск всех сервисов одной командой (docker-compose up)

**5. Завершение**

* Проверка работы API и фронта
* Написание README с инструкцией по запуску
* Финальное тестирование

**Дополнительно (по желанию)**

* Аутентификация (JWT)
* Админ-панель
* Настройка CI/CD (GitHub Actions)