Задание 3: Клиент – Серверное взаимодействие

💡 Цель:

Глубже изучить клиент - серверное взаимодействие.

🛠️ Технические требования

1. Стек технологий:

- (Node.js + Express)
- Клиент использовать ранее разработанное SPA-приложения

2. Особенности реализации:

✅ Реализация

Вариант 1 Сервер (Node.js + Express):

- Реализовать REST API для управления списком задач (CRUD).
- Использовать TypeScript на сервере.
- Данные хранить в памяти (массив объектов).
- Методы получения всех задач, поиск по названию и/ или дате

Вариант 2 Fake api (Функция имитирующая запросы, но хранящая данные на клиенте)

- Реализовать Функцию имитирующую REST API для управления списком задач (CRUD).
- Использовать TypeScript.
- Данные хранить в indexedDB/localStorage.
- Методы получения всех задач, поиск по названию и/ или дате

✅ Маршруты Сервер/Fake api

- Get /tasks получение всех задач
- Get /tasks/:id получение задачи по ид
- Delete /tasks/:id удаление
- Patch /tasks/:id обновление
- Post /tasks создание

Рекомендации по предварительной настройке проекта

Чтобы сделать процесс разработки более комфортным и стандартизированным, рекомендуется выполнить следующую настройку перед началом реализации:

tsconfig.json

Убедитесь, что в проекте есть корректно настроенный **tsconfig.json** файл.

ESlint + Prettier

Добавьте в проект систему проверки кода и форматирования.

Что нужно сдать?

1. **Ссылка на GitHub-репозиторий** с решением
2. **README.md** , содержащий:
    - Описание реализованного функционала
    - Инструкцию по запуску проекта
    - Используемые технологии и подходы
    - Краткое описание применённой архитектуры
1. **Комментарии в коде:** добавить комментарии в формате JSDoc к интерфейсам компонентов или для объяснения каких-то нетривиальных решений