**Практическое задание: Разработка Task Manager на React 19 + TypeScript + Ant Design**

Цель задания:  
Создать полнофункциональное веб-приложение для управления задачами, используя современный стек технологий: React 19, TypeScript и UI-библиотеку Ant Design (antd).

**Требования к приложению**

**1. Структура данных (TypeScript)**

Создайте интерфейс Task со следующими полями:

* id: string — уникальный идентификатор (используйте crypto.randomUUID() или Date.now().toString()).
* title: string — название задачи.
* description?: string — описание (необязательное поле).
* status: 'todo' | 'in-progress' | 'done' — статус задачи.
* priority: 'low' | 'medium' | 'high' — приоритет.
* createdAt: Date — дата создания.
* tags: string[] — массив тегов (например: ['frontend', 'bug']).

**2. Функциональность (CRUD + Фильтрация)**

Приложение должно позволять пользователю:

* Создавать новую задачу через форму (в модальном окне).
* Просматривать список всех задач.
* Редактировать существующую задачу (открывать форму с предзаполненными данными).
* Удалять задачу (с подтверждением через Modal.confirm).
* Искать задачи по названию и описанию (в реальном времени).
* Фильтровать задачи по:
  + Статусу (todo, in-progress, done)
  + Приоритету (low, medium, high)
  + Тегам (мультивыбор)

**3. Использование компонентов Ant Design**

Обязательно используйте следующие компоненты antd:

* Layout, Header, Content — для структуры страницы.
* Card — для отображения каждой задачи.
* List — для рендеринга списка задач.
* Form, Input, TextArea, Select, Radio, DatePicker, Checkbox — для формы задачи.
* Button (с иконками PlusOutlined, EditOutlined, DeleteOutlined).
* Modal — для форм создания и редактирования.
* Input.Search — для поисковой строки.
* Tag — для отображения тегов и приоритета (цветные: красный — high, синий — medium, зеленый — low).
* Badge — для отображения количества задач в каждом статусе.
* message.success / message.error — для уведомлений.
* Space, Divider, Typography.Title — для компоновки элементов.

**4. Дополнительные требования**

* Сохранение данных: Используйте localStorage для сохранения списка задач между перезагрузками страницы.
* Оптимизация: Примените useMemo для мемоизации отфильтрованного списка задач и useCallback для обработчиков.
* Адаптивность: Приложение должно корректно отображаться на разных экранах.
* Визуальное оформление: Карточки задач должны визуально выделяться по приоритету (например, цветная левая граница).