**Objetivos:**

- Desenvolvimento de aplicação utilizando React com TypeScript;
- Centralização do estado da aplicação utilizando a Context API;
- Utilização de efeitos colaterais com o hook useEffect;
- Modularização da lógica e da interface por meio de componentes reutilizáveis;
- Estilização da interface utilizando a biblioteca styled-components;
- Persistência de dados no navegador utilizando o localStorage;
- Definição e uso adequado de tipos e interfaces com TypeScript.

**Tema da prova prática:** Lista de tarefas com persistência local

**Descrição Geral:**

Desenvolva uma aplicação React com TypeScript chamada **Tarefas do Dia**, onde o usuário poderá adicionar, listar, marcar como concluída e remover tarefas. As tarefas devem ser armazenadas localmente (por meio do localStorage). A interface da aplicação deve ser estilizada utilizando styled-components.

**Forma de entrega:**

A entrega será presencial, mediante demonstração funcional da aplicação ao professor, que poderá realizar questionamentos sobre a implementação.

**Requisitos da prova**:

1. Estrutura do projeto

   Organize o código em, no mínimo, os seguintes arquivos: App.tsx, contexts/TarefaContext.tsx, components/ListaTarefas.tsx, components/FormNovaTarefa.tsx.

2. Gerenciamento de estado

   Utilize o hook useState para controlar o estado das tarefas, seja no componente ou no contexto.

   Cada tarefa deve possuir os seguintes atributos:

   - id: string
   - descricao: string
   - concluida: boolean

3. Context API

   Implemente um contexto chamado TarefaContext, contendo:

   - Lista de tarefas;
   - Função para adicionar tarefa;
   - Função para remover tarefa;
   - Função para alternar o status de conclusão da tarefa.

Utilize createContext e useContext.

4. Efeitos colaterais com useEffect

   Utilize o hook useEffect para:

   - Carregar tarefas previamente salvas no localStorage ao montar o componente;
   - Atualizar o localStorage sempre que a lista de tarefas for modificada.

5. Estilização com styled-components

   Utilize styled-components para criar uma interface com os seguintes elementos:

   - Formulário contendo um campo de texto e um botão “Adicionar”;
   - Lista de tarefas contendo:
     - Checkbox para marcar tarefa como concluída;
     - Botão para remoção da tarefa;
     - Estilização distinta para tarefas concluídas.

6. Tipagem com TypeScript

   Tipar corretamente os dados, especialmente:

   - A interface da entidade Tarefa;
   - As propriedades (props) de cada componente;
   - Os dados compartilhados via contexto.

**Critérios de avaliação**:

| **Critério** | **Pontos** |
|---|---|
| Organização e funcionamento geral | 2,0 |
| Uso correto de componentes | 1,5 |
| Implementação de Context API | 1,5 |
| Uso apropriado de useState e useEffect | 2,0 |
| Persistência com localStorage | 1,5 |
| Estilização com styled-components | 1,5 |