**Diário de Bordo – Peterson Fontinhas**

**Semana:** Semana de 25/05/2025 a –/–/2025 (Prazo final a definir)

**Responsabilidades/Atividades Realizadas:**

- Fui responsável pela concepção e criação da estrutura inicial de arquivos e diretórios do projeto, visando uma organização modular conforme as especificações **(`main.c`, `eventos.c/.h`, `participantes.c/.h`, `lista.c/.h`**).
- Defini as estruturas de dados (**`structs`**) centrais que servirão de base para todo o "Sistema de Controle de Participação em Eventos Estudantis". Esta etapa envolveu a criação dos seguintes arquivos de cabeçalho e suas respectivas **`structs`**, ainda sem a implementação das funções:
    - **`participantes.h`**:
        - **`Participante`**: Estrutura para armazenar os dados de cada participante (RA e nome).
    - **`lista.h`** (para a lista de participantes de um evento):
        - **`NodeParticipante`**: Nó da lista encadeada que armazena um **`Participante`** e o ponteiro para o próximo nó.
        - **`ListaParticipantes`**: Estrutura de controle (cabeçalho) para a lista de participantes de um evento específico, contendo o ponteiro **`head`** e a **`quantidade`** de inscritos.
    - **`eventos.h`** (para o gerenciamento de eventos):
        - **`DataEvento`**: Estrutura auxiliar para armazenar data e hora de forma organizada.
        - **`Evento`**: Estrutura principal para os dados de um evento (código, nome, data, local) e, crucialmente, um campo **`ListaParticipantes inscritos`** para conectar cada evento à sua própria lista de participantes.
        - **`NodeEvento`**: Nó da lista encadeada que armazenará um ponteiro para um `Evento` e o ponteiro para o próximo evento na lista geral.
        - **`GerenciadorEventos`**: Estrutura de controle (cabeçalho) para a lista principal de todos os eventos do sistema, contendo o **`head`** para esta lista e a **`quantidadeEventos`**.
- O objetivo desta fase foi estabelecer um alicerce robusto e bem definido para as estruturas de dados, facilitando a subsequente implementação das funcionalidades e a integração do trabalho em equipe.

**Trechos de Código Explicados:**

**(Em `eventos.h`) A `struct Evento`:**

```
typedef struct {
   int codigo;
   char nome[100];
   DataEvento dataEvento;
   char localEvento[100];
   ListaParticipantes inscritos; // cada evento tem uma ListaParticipantes
} Evento;
```

- *Explicação:* "Esta **`struct`** é o coração do sistema no que tange a um evento individual. Ela agrupa todas as informações pertinentes a um evento, como seu **`codigo`** identificador, **`nome`**, **`dataEvento`** (que é outra **`struct`** para melhor organização da data e hora), e **`localEvento`**. O campo mais importante para a dinâmica do sistema é **`inscritos`**, do tipo **`ListaParticipantes`** (definida em **`lista.h`**). Isso significa que cada instância de **`Evento`** carregará consigo a cabeça da lista de todos os participantes inscritos especificamente nele, permitindo o gerenciamento individualizado dos inscritos por evento."

**(Em `eventos.h`) A `struct GerenciadorEventos`:**

```
typedef struct{
   NodeEvento* head; // NodeEvento aponta para um Evento
   int quantidadeEventos;
} GerenciadorEventos;
```

- *Explicação:* "Para gerenciar todos os diversos eventos que o sistema pode ter, criei a **`GerenciadorEventos`**. Ela atua como um cabeçalho para a lista mestra de eventos. O ponteiro **`head`** aponta para o primeiro **`NodeEvento`** (nó da lista de eventos), e **`quantidadeEventos`** rastreia o número total de eventos cadastrados. Esta estrutura será fundamental para funções como cadastrar um novo evento no sistema ou buscar por um evento existente."

**Dificuldades Encontradas e Como Foram Resolvidas:**

- "Uma consideração inicial foi como melhor representar a relação entre um evento e sua lista de participantes. Decidi por incluir a **`struct ListaParticipantes`** diretamente dentro da **`struct Evento`** (em vez de apenas um ponteiro para ela que precisaria ser alocado separadamente) para simplificar a gestão da memória de cada evento e garantir que toda **`Evento`** já 'nasça' com sua estrutura de lista de participantes pronta para ser inicializada."

**Próximos Passos (Planejados para a próxima semana/período):**

- Concentrar na implementação das funções de manipulação em **`eventos.c`**, começando pelas funcionalidades de **`inicializarGerenciadorEventos`**, **`cadastrarNovoEvento`** (incluindo a correta inicialização da **`ListaParticipantes`** interna do novo evento) e **`buscarEventoPorCodigo`**.