Rotas e Funcionalidades - Vivere+

1. Rotas do Sistema

1.1. Página Principal

- `GET /home` → Retorna à página inicial da aplicação.

1.2. Autenticação *(Recurso protegido) (Funcionalidade Adicional)*

- `POST /login` → Realiza a autenticação do usuário.
- `POST /register` → Registra um novo usuário no sistema.

1.3. Experiências

- `POST /home/experiencia` → Cria uma nova experiência.
- `GET /home/experiencia` → Lista todas as experiências disponíveis.
- `GET /home/experiencia/<id_experiencia>` → Retorna os detalhes de uma experiência específica.
- `PUT /home/experiencia/<id_experiencia>` → Atualiza uma experiência existente.
- `DELETE /home/experiencia/<id_experiencia>` → Remove uma experiência.

1.4. Compras *(Recurso protegido) (Funcionalidade Adicional)*

- `POST /home/compra` → Realiza a compra de um ticket de experiência.

1.5. Categorias

- `GET /home/categoria/<id_categoria>` → Retorna todas as experiências dentro de uma determinada categoria.

1.6. Popular Banco

- `GET /populate` → Popula banco de dados para ver como a página está ficando

---

2. Modificações no Front-end

2.1. Página Inicial (Home)

- Criar requisição para listar as experiências por categoria.
- Criar requisição para exibir a experiência em destaque (aquela que não pertence a nenhuma categoria específica).

2.2. Componente de Categoria (Pequeno e Grande)

- Implementar requisição para que, ao clicar em uma categoria, o usuário seja redirecionado para a página com as experiências correspondentes.
- Padronizar informações com banco de dados

2.3. Registro de Experiência

- Implementar requisição para enviar as informações da nova experiência para o banco de dados.
- Ajustar a visualização de prévia do cadastro (se necessário).
- Ajustar informações inseridas e verificar novamente quais são obrigatórias e opcionais
- Ajustar a compra de ingressos para adequar ao padrão do banco de dados, por exemplo, caso o ingresso seja gratuito não permitir inserir preço, já nos outros tipos permitir a inserção

2.4. Página de Categorias

- Criar requisição para listar todas as experiências pertencentes a uma determinada categoria, similar à listagem da página inicial.

2.5. Navegação entre Rotas

- Garantir a navegação fluida entre todas as páginas e funcionalidades do sistema por meio do header.

2.6. Página da Experiência

- Implementar requisição para editar a experiência (tomando cuidado com informações opcionais ou obrigatórias)
- Implementar requisição para deletar a experiência
- Implementar requisição para pegar o id e exibir as informações da experiência

2.7. Conexão do Front-End com os Arquivos do Back-End

- Implementar a conexão das funções do front com o back

---

3. Funcionalidades Adicionais

3.1. Usuário e Autenticação

- Criar uma janela/modal para login e cadastro, garantindo autenticação segura e armazenamento dos dados no banco.

3.2. Paineis Separados para Admin e Cliente

- **Admin**: Página com todas as funções de CRUD para experiências (sem compra).
- **Cliente**: Página voltada para compra (sem CRUD de experiência), incluindo:
    - Implementação de um **carrinho de compras**.
    - Criação do **processo de checkout** para finalização da compra.

3.3. Testes Unitários

- Desenvolver um arquivo dedicado para testar todas as requisições da API, garantindo que os endpoints funcionem corretamente.

3.4. Barra de Busca

- Implementar uma barra de pesquisa para que os usuários possam **buscar experiências pelo nome** de forma rápida e eficiente.
    - Permitir que o usuário digite o nome (ou parte dele) de uma experiência desejada.
    - Realizar uma consulta no banco de dados para **filtrar eventos** que contenham o termo pesquisado/
    - Criar um **endpoint de busca** no back-end: `GET /home/experiencia?search=<termo>`
    - O endpoint deve buscar no banco de dados eventos cujo título contenha o termo pesquisado.
    - A resposta deve ser uma **lista de eventos filtrados**.

3.5. Upload de Imagem

- Implementar uma forma do banco de dados receber imagens enviadas do próprio usuário ao invés de apenas receber um url.

4. Modelos do Banco de Dados

4.1. Categoria (`Categoria`)

A tabela Categoria armazena os diferentes tipos de eventos disponíveis.

Campos:

- **`id` (Int, Primary Key, Auto Increment) → Identificador único da categoria.**
- **`nome` (String) → Nome da categoria.**
- **`eventos` (Relação com a tabela Evento) → Define a relação entre categorias e eventos (uma categoria pode ter vários eventos).**

🔹 **Relacionamento: 1:N (Uma categoria pode conter vários eventos).**

4.2. Evento (`Evento`)

A tabela Evento armazena todas as informações sobre um evento específico.

Campos:

- **`id` (Int, Primary Key, Auto Increment) → Identificador único do evento.**
- **`titulo` (String) → Nome do evento.**
- **`descricao` (String) → Descrição detalhada do evento.**
- **`endereco` (String) → Local onde o evento ocorrerá.**
- **`dataInicio` (DateTime, Opcional) → Data e hora de início do evento.**
- **`dataTermino` (DateTime, Opcional) → Data e hora de término do evento.**
- **`ticketType` (Enum `TicketType`) → Tipo de ingresso do evento (Padrão: `INGRESSO`).**
- **`imagemUrl` (String, Opcional) → URL da imagem representativa do evento.**
- **`categoriaId` (Int, Foreign Key) → Identificador da categoria associada ao evento.**
- **`categoria` (Relação com a tabela Categoria) → Define a categoria à qual o evento pertence.**

4.3. Enumeração de Tipos de Ingresso (`TicketType`)

O banco de dados define um Enum chamado `TicketType`, que determina os tipos de ingressos disponíveis para os eventos.

Opções disponíveis:

- **`INGRESSO` → Tipo padrão de ingresso.**
- **`VIP` → Ingressos especiais para acesso VIP.**
- **`GRATUITO` → Eventos sem custo.**

Alterações e Melhorias no Banco de Dados

1. Transformar `Categoria` em um `Enum` caso as categorias sejam fixas.

- Shows e Entretenimento
- Workshops e Aulas
- Viagens e Turismo
- Aventura e Adrenalina
- Relaxamento e Bem-Estar

- Gastronomia e Degustações
- Infantil e Familiar
- Experiências Personalizadas

2. Manter a FK de categoria no evento se precisar de CRUD dinâmico para categorias.
3. Adicionar um modelo `Usuario` para armazenar dados de autenticação e perfil.
4. Adicionar um modelo `Carrinho` para suportar múltiplos itens antes da compra.
5. Incluir um campo null com default 0 `preco` na tabela `Evento`, já que eventos pagos precisam ter um valor.
6. Permitir `null` em `dataInicio` e `dataTermino` caso o evento não tenha data definida ainda.
7. Adicionar valor default para `imagemUrl` com um placeholder.
8. Adicionar valor default para preço do ticket como 0 caso o ingresso seja gratuito