

## Sistema de gestão de eventos desportivos

João Pedro Mendes Ribeiro  
(nrº 23795, regime diurno)

Orientação de  
Sandro Carvalho

LICENCIATURA EM ENGENHARIA EM SISTEMAS INFORMÁTICOS  
ESCOLA SUPERIOR DE TECNOLOGIA  
INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE

## **Identificação do aluno**

João Pedro Mendes Ribeiro

Aluno número 23795, regime diurno

Licenciatura em Engenharia em Sistemas Informáticos

## **Orientação**

Sandro Carvalho

## Resumo

O presente projeto teve como objetivo a análise, especificação e implementação de um sistema de gestão de eventos desportivos, utilizando tecnologias modernas como .NET para o back-end, React para o front-end e PostgreSQL para a base de dados. A solução foi projetada para atender às necessidades tanto dos promotores quanto dos participantes de eventos, oferecendo funcionalidades essenciais para a criação, gestão e participação nesses eventos.

No desenvolvimento da plataforma, foram implementadas funcionalidades que permitem aos promotores inserir, editar, remover eventos e visualizar o número de inscritos. Aos utilizadores, foi proporcionada a possibilidade de visualizar eventos disponíveis, inscrever-se, acompanhar os eventos em que estão inscritos e consultar o histórico de eventos passados. Além disso, foi garantido um sistema de autenticação seguro e um design de interface intuitivo para otimizar a experiência do usuário.

Como conclusão, o projeto demonstrou a eficácia das tecnologias escolhidas na criação de uma solução robusta e eficiente para a gestão de eventos desportivos.



## Abstract

The present project aimed to analyze, specify, and implement a sports event management system using modern technologies such as .NET for the back-end, React for the front-end, and PostgreSQL for the database. The solution was designed to meet the needs of both event organizers and participants, offering essential functionalities for event creation, management, and participation.

In the development of the platform, features were implemented to allow organizers to add, edit, remove events, and view the number of registered participants. For users, the platform provided the ability to view available events, register, track the events they are enrolled in, and consult participation history. Additionally, a secure authentication system and an intuitive user interface design were ensured to optimize the user experience.

In conclusion, the project demonstrated the effectiveness of the chosen technologies in creating a robust and efficient solution for sports event management. The developed platform fills an existing gap in the market, offering tools that simplify administrative processes, enhance user experience, and enable effective analysis of historical data. This work reinforces the potential of technological solutions to transform the organization and participation in sports events.



## Agradecimentos

Gostaria de aproveitar este espaço para expressar a minha gratidão a todos aqueles que, de alguma forma, contribuíram para a realização deste projeto.

Em primeiro lugar, agradeço ao meu orientador, Sandro Carvalho, pelo apoio constante, pelas orientações valiosas e pela paciência ao longo deste percurso. A sua dedicação, profissionalismo e disponibilidade foram fundamentais para ultrapassar os desafios encontrados e para a concretização deste projeto.

Agradeço profundamente à minha família, que sempre esteve ao meu lado, oferecendo o suporte emocional necessário em todas as fases deste projeto.

Quero também deixar um agradecimento especial aos meus amigos, que estiveram sempre presentes, mesmo à distância, prontos para oferecer palavras de apoio, momentos de descontração e a leveza necessária para equilibrar o esforço e o descanso. A vossa amizade foi um pilar fundamental durante esta jornada.

Além disso, não posso deixar de mencionar todos aqueles que, direta ou indiretamente, contribuíram para este projeto — colegas, professores e outros profissionais que, através de conversas, partilhas e sugestões, ajudaram a moldar o resultado final.

Cada pessoa mencionada, de forma direta ou indireta, deixou a sua marca neste projeto, e por isso, a todos, deixo o meu mais sincero e profundo agradecimento.

Obrigado por acreditarem em mim e por fazerem parte desta jornada!



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Objetivos . . . . .	1
1.2	Contexto . . . . .	2
1.3	Estrutura do documento . . . . .	3
<b>2</b>	<b>Análise ao problema</b>	<b>4</b>
2.1	Desafios na Gestão de Eventos Desportivos . . . . .	4
2.2	Estado da Arte: Soluções Existentes no Mercado . . . . .	4
2.2.1	SportsEngine . . . . .	4
2.2.2	Active Network . . . . .	5
2.2.3	TeamSnap . . . . .	5
2.3	Lacunas Identificadas . . . . .	6
2.4	Comparação dos diferentes softwares . . . . .	7
<b>3</b>	<b>Análise e Modelação do Sistema</b>	<b>8</b>
3.1	Levantamento de requisitos . . . . .	8
3.1.1	Requisitos Funcionais . . . . .	8
3.1.2	Requisitos não Funcionais . . . . .	9
3.1.3	Atores . . . . .	10
3.2	Arquitetura técnica do sistema . . . . .	10
3.3	Diagrama de casos de uso . . . . .	11
3.3.1	User Stories . . . . .	12
3.4	Diagrama Entity-relationship (ER) . . . . .	12
3.4.1	Tabela Eventos . . . . .	13
3.4.2	Tabela Users . . . . .	14
3.4.3	Tabela Inscrições . . . . .	14
3.4.4	Tabela __EFMigrationsHistory . . . . .	14
3.4.5	Relacionamentos . . . . .	15

3.5	Mockups . . . . .	15
3.5.1	Página de Login . . . . .	15
3.5.2	Página de Registo . . . . .	16
3.5.3	Página Inicial . . . . .	17
3.5.4	Página de Eventos . . . . .	18
3.5.5	Página de Visualização de um evento . . . . .	19
3.5.6	Página de Eventos inscritos . . . . .	20
3.5.7	Página de Histórico . . . . .	21
3.5.8	Página do Promotor . . . . .	22
3.5.9	Página de eventos do promotor . . . . .	23
<b>4</b>	<b>Implementação</b>	<b>24</b>
4.1	Tecnologias utilizadas . . . . .	24
4.1.1	.NET Core . . . . .	24
4.1.2	React . . . . .	24
4.1.3	PostgresSQL . . . . .	24
4.2	Ferramentas utilizadas . . . . .	24
4.2.1	Visual Paradigm . . . . .	25
4.2.2	GitLab . . . . .	25
4.2.3	Rider . . . . .	25
4.2.4	VisualStudio Code . . . . .	25
4.2.5	Postman . . . . .	25
4.2.6	pgAdmin 4 . . . . .	25
4.3	Autenticação . . . . .	25
4.3.1	Registo . . . . .	26
4.3.2	Login . . . . .	26
4.3.3	Logout . . . . .	28
4.4	Back-end . . . . .	28
4.4.1	Criação de um evento . . . . .	28
4.4.2	Listagem de eventos . . . . .	30
4.4.3	Visualização detalhada de um evento . . . . .	30
4.4.4	Exclusão de um evento . . . . .	31
4.4.5	Edição de um evento . . . . .	33
4.4.6	Visualização de eventos inscritos . . . . .	34
4.4.7	Inscrição em eventos . . . . .	36
4.4.8	Visualização do numero de inscritos num evento . . . . .	38

4.4.9	Visualização do histórico de eventos . . . . .	39
4.4.10	Upload de imagens . . . . .	40
4.5	Front-end . . . . .	41
4.5.1	Configuração das rotas . . . . .	41
4.5.2	Utilização de Hooks . . . . .	42
4.5.3	Integração com o Back-end . . . . .	43
4.5.4	CSS . . . . .	44
4.5.5	Testes . . . . .	45
<b>5</b>	<b>Análise de resultados</b>	<b>57</b>
5.1	Página de Login . . . . .	57
5.2	Página de Registo . . . . .	58
5.3	Página Inicial . . . . .	59
5.4	Página de Eventos . . . . .	60
5.5	Página de Visualização de um evento . . . . .	61
5.6	Página de Eventos inscritos . . . . .	62
5.7	Página de Histórico . . . . .	63
5.8	Página do Promotor . . . . .	64
5.9	Página de Eventos do Promotor . . . . .	65
5.10	Avaliação final . . . . .	66
<b>6</b>	<b>Conclusão</b>	<b>67</b>



# **Lista de Figuras**

3.1	Arquitetura técnica do sistema . . . . .	10
3.2	Diagrama de casos de uso . . . . .	11
3.3	Diagrama Entity-relationship (ER) . . . . .	13
3.4	Mockup da página de login . . . . .	16
3.5	Mockup da página de registo . . . . .	16
3.6	Mockup da página inicial . . . . .	17
3.7	Mockup da página de eventos . . . . .	18
3.8	Mockup da página de visualização de um evento . . . . .	19
3.9	Mockup da página de eventos inscritos . . . . .	20
3.10	Mockup da página de histórico . . . . .	21
3.11	Mockup da página do promotor . . . . .	22
3.12	Mockup da página de eventos do promotor . . . . .	23
4.1	Teste no postman para o registo . . . . .	46
4.2	Teste no postman para o login . . . . .	46
4.3	Teste no postman para o logout . . . . .	47
4.4	Teste no postman para criar evento . . . . .	48
4.5	Teste no postman para a listagem de eventos . . . . .	49
4.6	Teste no postman para a visualização detalhada de um evento . . . . .	50
4.7	Teste no postman para a exclusão de um evento . . . . .	51
4.8	Teste no postman para a visualização de eventos inscritos . . . . .	52
4.9	Teste no postman para a inscrição em eventos . . . . .	53
4.10	Teste no postman para a visualização do numero de inscritos num evento .	54
4.11	Teste no postman para a visualização do histórico de eventos . . . . .	55
4.12	Teste no postman para dar upload a imagens . . . . .	56
5.1	Página de login . . . . .	57
5.2	Página de registo . . . . .	58
5.3	Página inicial . . . . .	59

5.4	Página de eventos . . . . .	60
5.5	Página de visualização de um evento . . . . .	61
5.6	Página de eventos inscritos . . . . .	62
5.7	Página de histórico . . . . .	63
5.8	Página do promotor . . . . .	64
5.9	Página de eventos do promotor . . . . .	65

## Lista de Tabelas



# Listagens de Código

4.1	Endpoint para efetuar o registo no sistema.	26
4.2	Endpoint para efetuar login no sistema.	27
4.3	Endpoint para efetuar logout do sistema.	28
4.4	Endpoint para criar um evento no sistema.	28
4.5	Endpoint para listar todos os eventos do sistema.	30
4.6	Endpoint para visualizar os detalhes de um evento perante o ID	30
4.7	Endpoint para excluir um evento do sistema	31
4.8	Endpoint para editar um evento do sistema	33
4.9	Endpoint para visualizar eventos inscritos	34
4.10	Endpoint para realizar a inscrição em eventos	36
4.11	Endpoint para visualizar o numero de inscritos num evento	38
4.12	Endpoint para visualizar o histórico de eventos já realizados	39
4.13	Endpoint para dar upload de imagens	40
4.14	Código da configuração das rotas no react	42
4.15	Exemplo de código para a utilização da hook useState	42
4.16	Exemplo de código para a utilização da hook useEffect	43
4.17	Exemplo de código para a utilização da hook useParams	43
4.18	Exemplo de código para a utilização da hook useNavigate	43
4.19	Exemplo de código para a utilização da biblioteca Axios	44
4.20	Exemplo do css utilizando flexbox	45



# Siglas & Acrónimos

**API** Application Programming Interface. 10, 11, 25

**CRUD** Create, Read, Update, Delete. 28

**ER** Entity-relationship. ix, xiii, 3, 8, 12, 13

**HTTP** Hypertext Transfer Protocol. 25, 44

**IDE** Integrated Development Environment. 25

**RF** Requisito Funcional. 8, 9

**RNF** Requisito não Funcional. 9

**UML** Unified Modeling Language. 3

**US** User Stories. 12



# Glossário

**.NET** É uma plataforma de desenvolvimento de software da Microsoft que suporta a criação de uma ampla variedade de aplicações, incluindo desktop, web, móveis e jogos, utilizando linguagens como C#, F# e Visual Basic . x, 1, 2, 10, 11, 24

**Axios** É uma biblioteca popular em JavaScript/TypeScript para realizar requisições HTTP assíncronas. 43

**Back-end** Refere-se à parte de um sistema de software que lida com a lógica e processamento dos dados, geralmente localizado no servidor. É responsável pelo funcionamento interno da aplicação, como o processamento de dados, segurança, integrações e interações com a base de dados. xi, 1, 3, 10, 11, 25, 28, 43, 44

**DevOps** É a união de pessoas, processos e tecnologias para fornecer continuamente valor aos clientes. 25

**Entity Framework** É uma ferramenta da Microsoft, pertencente ao pacote ADO.NET, que permite trabalhar com base de dados relacional de maneira abstrata. 14, 29–32, 37, 39

**Flexbox** Tem a função de organizar elementos dentro de containers de forma flexível. 44, 45

**Framework** É um conjunto de ferramentas e convenções que facilitam o desenvolvimento de software ao fornecer uma estrutura pré-definida para organizar e executar tarefas comuns. 10, 24

**Front-end** É a parte visível para os utilizadores finais de uma aplicação ou site. É a interface gráfica com a qual os utilizadores interagem diretamente. O front-end é desenvolvido usando linguagens de marcação, estilos e scripts que são renderizados e executados no navegador do utilizador, possibilitando a interação com o back-end através de solicitações HTTP. 1, 3, 10, 11, 24, 25, 43, 44

**Full-stack** Junção do front-end e back-end, ou seja, desenvolvimento de código para cliente e servidor em simultâneo. 1, 10

**Git** É um sistema de controlo de versões. 25

**GitLab** É um gestor de repositório de software baseado em git. x, 25

**Hash** É a transformação de uma grande quantidade de dados numa pequena quantidade de informações. 14

**Hooks** São funções que permitem ligar aos recursos de state e ciclo de vida do React a partir de componentes funcionais. xi, 42

**JavaScript** É uma linguagem de programação amplamente utilizada para desenvolvimento web. É conhecida por ser interpretada pelos navegadores, permitindo interatividade dinâmica em páginas web. 10, 24

**JetBrains** É uma empresa de desenvolvimento de software que produz ferramentas para desenvolvimento de software. 25

**Mockups** São representações visuais realistas de um produto ou design, usadas para demonstrar a sua aparência e funcionalidade antes da produção final. x, 15, 17, 19, 21, 23

**MoSCoW** Método de priorização de funcionalidades e/ou requisitos. 8, 9

**Open-source** Tem o seu código aberto, ou seja, que pode ser visualizado por qualquer pessoa. 24

**PostgresSQL** É uma base de dados relacional open-source conhecido pela sua robustez e recursos avançados. x, 1, 2, 10, 24–26

**React** Framework de código aberto front-end para desenvolvimento web. Permite criar interfaces para o utilizador recorrendo a componentes. x, 1, 2, 10, 11, 24, 25, 43

**RESTful** Refere-se a um estilo de arquitetura de software baseado nos princípios REST. É utilizado para projetar sistemas de software que são simples e escaláveis pela Internet. 10, 11, 24, 44

# 1. Introdução

Este projeto enquadra-se na área dos Sistemas Informáticos, especificamente no Desenvolvimento Web, que envolve a criação e programação de websites e aplicações web. Dentro do desenvolvimento web, há uma divisão comum entre o Front-end e o Back-end. O Front-end diz respeito à interface visual com a qual os utilizadores interagem, enquanto o Back-end refere-se às funcionalidades e processos que sustentam essa interface, mas que geralmente não são visíveis para o utilizador. Durante este projeto, foi desenvolvido um sistema Full-stack, englobando tanto o desenvolvimento do Front-end quanto do Back-end, com o objetivo de criar uma plataforma de gestão de eventos desportivos abrangente e eficiente.

Na atualidade, a gestão de eventos desportivos abrange várias atividades, desde a criação e promoção dos eventos até à inscrição e acompanhamento dos participantes. A implementação de sistemas informáticos específicos para este fim permite uma gestão centralizada e eficiente, automatizando processos manuais e reduzindo erros humanos.

Este projeto proporciona funcionalidades tanto para os promotores de eventos como para os utilizadores. Os promotores têm a capacidade de inserir, remover e editar eventos, bem como visualizar o número de inscritos em cada evento. Os utilizadores podem visualizar eventos, inscrever-se nos mesmos e acompanhar os eventos em que estão inscritos, além de consultar o histórico de eventos anteriores. Para garantir a segurança e a personalização das experiências, tanto promotores como utilizadores necessitam de se registar e efetuar login.

Neste contexto, a criação de uma plataforma integrada para a gestão de eventos desportivos não só otimiza a organização e a participação nos eventos, mas também contribui para um maior engajamento e satisfação dos utilizadores. O desenvolvimento deste sistema exemplifica a aplicação prática dos conhecimentos em Sistemas Informáticos, mais concretamente em desenvolvimento Web, demonstrando como estas áreas podem ser utilizadas para resolver problemas reais e melhorar processos existentes.

## 1.1 Objetivos

Os objetivos iniciais deste projeto foram a análise, especificação e implementação de um conjunto de funcionalidades prioritárias para um sistema de gestão de eventos desportivos, desenvolvido com recurso às tecnologias .NET para o Back-end, React para o Front-end e PostgreSQL para a base de dados. Estes objetivos incluem:

- Desenvolver uma plataforma que permita aos promotores de eventos:

- Inserir novos eventos desportivos.
  - Remover eventos existentes.
  - Editar eventos existentes.
  - Visualizar o número de inscritos em cada evento.
- Criar funcionalidades para os utilizadores da plataforma, incluindo:
    - Visualizar a lista de eventos disponíveis.
    - Inscrever-se em eventos.
    - Visualizar os eventos em que estão inscritos.
    - Consultar o histórico de eventos passados.
- Garantir um sistema de autenticação seguro, onde tanto promotores quanto utilizadores necessitam de se registar e efetuar login.
  - Assegurar uma experiência de utilizador intuitiva e eficiente através de um design de interface de fácil utilização.
  - Avaliar e experimentar as funcionalidades das tecnologias escolhidas (.NET, React e PostgreSQL), garantindo que suportam de forma eficiente as necessidades do projeto.

## 1.2 Contexto

O desenvolvimento deste projeto insere-se no contexto da crescente necessidade de soluções tecnológicas eficientes para a gestão de eventos desportivos. Com o aumento da popularidade dos desportos e da realização de eventos em várias escalas, torna-se imperativo dispor de sistemas que facilitem tanto a organização quanto a participação nesses eventos.

A plataforma desenvolvida destina-se a preencher uma lacuna no mercado, onde muitos organizadores de eventos desportivos ainda dependem de métodos manuais ou de sistemas não integrados para gerir as suas atividades. Ao fornecer uma solução digital centralizada, este projeto visa:

- **Facilitar a Organização de Eventos:** Simplificar o processo de criação e gestão de eventos desportivos.
- **Melhorar a Experiência dos Participantes:** Permitir aos utilizadores inscrever-se em eventos de forma fácil e rápida, além de poder visualizar os eventos em que estão inscritos.

- **Automatizar Processos Administrativos:** Reduzir o tempo e o esforço necessários para gerir as inscrições.
- **Fornecer Acesso a Dados Históricos:** Permitir tanto aos promotores quanto aos utilizadores consultar o histórico de eventos, o que pode ser útil para análises e melhorias futuras.

Este projeto representa uma iniciativa que tem como objetivo criar uma solução completa e eficiente para a gestão de eventos desportivos. O desenvolvimento desta plataforma demonstra como a aplicação de tecnologias modernas pode transformar a maneira como eventos desportivos são organizados e geridos, oferecendo uma ferramenta valiosa para promotores e participantes.

### 1.3 Estrutura do documento

O presente documento apresenta um estudo detalhado sobre o desenvolvimento de uma plataforma de gestão de eventos desportivos, focando os aspectos de análise, design, implementação e avaliação dos resultados obtidos. Inicialmente, a introdução contextualiza o projeto dentro da área dos Sistemas Informáticos e define os objetivos principais do trabalho.

Além desta introdução inicial, o documento está estruturado da seguinte forma:

- **Análise ao Problema:** Este capítulo inclui uma análise detalhada dos riscos para a gestão eficiente de eventos desportivos. Serão identificados os desafios enfrentados pelos promotores de eventos e as lacunas existentes nos sistemas atuais de gestão de eventos.
- **Análise e Modelação do Sistema:** Este capítulo inclui todos os requisitos levantados (requisitos funcionais e requisitos não funcionais), a arquitetura técnica do sistema, a apresentação de alguns diagramas Unified Modeling Language (UML) (diagrama de casos de uso, diagrama Entity-relationship (ER)) e, por fim, a apresentação dos mockups para as diferentes páginas web.
- **Implementação:** Este capítulo, inclui a discussão sobre a implementação prática da plataforma, detalhando as tecnologias e as ferramentas escolhidas. Serão abordados aspectos técnicos relacionados com o desenvolvimento do sistema (Front-end e Back-end). Serão, também, discutidos os testes realizados para validar a funcionalidade e desempenho da plataforma.
- **Análise de Resultados:** Este capítulo inclui uma análise dos resultados obtidos durante a implementação.
- **Conclusão:** Finalmente, a conclusão apresentará uma reflexão final sobre o projeto desenvolvido.

## 2. Análise ao problema

A gestão eficaz de eventos desportivos envolve uma série de desafios que afetam diretamente a experiência dos organizadores e participantes.

### 2.1 Desafios na Gestão de Eventos Desportivos

A organização de eventos desportivos envolve uma complexa rede de tarefas que vão desde a criação e promoção dos eventos até à gestão de inscrições e logística durante o evento. Entre os desafios comuns encontram-se:

- **Complexidade na Gestão de Inscrições:** Muitos sistemas atuais não oferecem uma maneira integrada e eficiente de gerir o processo de inscrição, o que pode resultar em erros, atrasos e frustração para os participantes.
- **Dificuldade na Promoção de Eventos:** A falta de ferramentas adequadas para a promoção eficaz dos eventos pode limitar a visibilidade e o alcance dos mesmos, afetando o número de participantes.
- **Gestão de Recursos e Logística:** Para eventos que envolvem múltiplas localizações, a gestão eficiente de logística e recursos pode ser complexa. Isso inclui desde a gestão de instalações desportivas até a alimentação para os participantes.

### 2.2 Estado da Arte: Soluções Existentes no Mercado

Existem algumas soluções disponíveis no mercado para a gestão de eventos desportivos, cada uma com suas características distintas e focos específicos. Abaixo, serão apresentadas análises detalhadas das plataformas SportsEngine, Active Network e TeamSnap, destacando suas funcionalidades principais, pontos fortes e limitações.

#### 2.2.1 SportsEngine

A sportsengine é uma plataforma robusta dedicada à gestão de torneios e ligas desportivas. Destaca-se pela sua capacidade de suportar competições de alto nível, oferecendo funcionalidades avançadas como gestão detalhada de equipas, horários de jogos e estatísticas. É amplamente utilizada por organizações que requerem um sistema sofisticado

para administrar competições complexas. No entanto, a sua interface pode ser complexa para utilizadores iniciantes e pequenas organizações desportivas.

- **Pontos fortes:**

- Especialização em gestão de torneios e ligas desportivas;
- Funcionalidades avançadas para gestão de equipas e horários de jogos.

- **Pontos fracos:**

- Complexidade na utilização para pequenas organizações e eventos de menor escala;
- Foco limitado em eventos não competitivos.

### 2.2.2 Active Network

A Active Network é reconhecida pela sua plataforma de inscrição online e gestão operacional de eventos. Esta solução permite aos organizadores configurar facilmente processos de inscrição, gerir pagamentos e comunicar informações cruciais aos participantes. Apesar de oferecer relatórios básicos para análise de desempenho do evento, a sua especialização em eventos recreativos e comunitários pode limitar a sua adaptação para competições desportivas de alto nível. Além disso, a personalização das funcionalidades pode ser restrita, dificultando a adaptação para diferentes modalidades desportivas e requisitos específicos de eventos desportivos.

- **Pontos fortes:**

- Ferramentas robustas para inscrição online e gestão operacional de eventos;
- Relatórios básicos para análise de desempenho do evento.

- **Pontos fracos:**

- Menor especialização em competições desportivas de alto nível;
- Personalização limitada para diferentes modalidades desportivas.

### 2.2.3 TeamSnap

A TeamSnap é conhecida pela sua eficiência na gestão de equipas desportivas, facilitando a comunicação entre membros da equipa, organização de eventos e acompanhamento de calendários. Com uma interface intuitiva e fácil de usar, esta plataforma é ideal para treinadores, pais e atletas que procuram uma solução simples para coordenação de atividades desportivas. No entanto, as suas funcionalidades são mais adequadas para gestão de equipas do que para eventos individuais, e a personalização das ferramentas pode ser limitada para necessidades específicas de eventos desportivos mais complexos.

- **Pontos fortes:**

- Gestão simplificada de equipas e comunicação entre membros;
- Interface intuitiva e fácil de usar.

- **Pontos fracos:**

- Funcionalidades mais adequadas para gestão de equipas do que para eventos individuais;
- Limitações em funcionalidades avançadas de gestão de eventos.

## 2.3 Lacunas Identificadas

Apesar das opções disponíveis, algumas lacunas significativas permanecem no mercado de gestão de eventos desportivos:

- **Falta de Personalização Avançada:** Embora as plataformas existentes ofereçam opções de personalização, muitas vezes essas personalizações são limitadas e não abrangem todas as necessidades específicas de diferentes modalidades desportivas e tipos de eventos. Isso pode resultar em dificuldades para adaptar completamente a plataforma às exigências únicas de cada evento, limitando a capacidade dos organizadores de eventos em proporcionar uma experiência personalizada aos participantes.
- **Complexidade na Utilização:** Alguns sistemas disponíveis no mercado podem ser complexos demais para utilizadores não técnicos. Isso pode levar a curvas de aprendizagem prolongadas e dificuldades na utilização diária da plataforma. Uma interface complexa pode resultar em erros de utilização e frustração por parte dos utilizadores, especialmente aqueles que não têm experiência prévia com tecnologias avançadas.
- **Segurança e Privacidade de Dados:** Com a crescente preocupação com segurança cibernética e privacidade de dados, algumas plataformas podem não fornecer medidas de segurança robustas o suficiente para proteger informações sensíveis dos participantes, como dados pessoais.
- **Escalabilidade e Desempenho:** Em eventos de grande escala ou com picos de tráfego, algumas plataformas podem enfrentar desafios de escalabilidade e desempenho. Isso pode resultar em tempos de carregamento lentos, falhas no sistema durante picos de uso e uma experiência geral inconsistente para os utilizadores.

Ao destacar essas lacunas, torna-se evidente a oportunidade para uma plataforma de gestão de eventos desportivos que ofereça uma personalização avançada, uma interface intuitiva e um suporte abrangente para competições de alto nível, visando preencher as lacunas identificadas no mercado existente.

## 2.4 Comparação dos diferentes softwares

- **SportsEngine:** Especializada em gestão de torneios e ligas desportivas, oferecendo funcionalidades avançadas para competições de alto nível. ([sportsengine](#))
- **Active Network:** Focada em inscrição online e gestão operacional de eventos, com relatórios básicos para análise de desempenho. ([Active Network](#))
- **TeamSnap:** Destinado principalmente à gestão de equipas e comunicação, com uma interface intuitiva e fácil de usar. ([TeamSnap](#))
- **Plataforma Desenvolvida:** Adaptada para preencher as lacunas identificadas nas soluções existentes.

Para comparar as características das diferentes soluções de software mencionados no subcapítulo 2.2, foi criada a seguinte Tabela 2.1:

Funcionalidade	SportsEngine	Active Network	TeamSnap	Plataforma Desenvolvida
Gestão de Inscrições	Sim	Sim	Básico	Sim
Personalização	Limitada	Limitada	Limitada	Alta
Usabilidade	Complexa	Boa	Boa	Excelente
Suporte a diferentes desportos	Sim	Básico	Básico	Sim
Foco em eventos desportivos	Elevado	Limitado	Médio	Total
Complexidade do website	Alta	Média	Média	Baixa
Inscrição em eventos	Sim	Sim	Sim	Sim
Consulta de histórico de eventos	Básica	Básica	Básica	Média/Avançada

Tabela 2.1: Comparação das funcionalidades das soluções existentes no mercado com a plataforma desenvolvida.

# 3. Análise e Modelação do Sistema

Este capítulo apresenta a análise e modelação do sistema desenvolvido onde são apresentados o levantamento de requisitos 3.1, a arquitetura técnica do sistema 3.2, o diagrama de casos de uso 3.3, o diagrama Entity-relationship (ER) 3.4 e, por fim, o desenho dos mockups 3.5.

## 3.1 Levantamento de requisitos

O levantamento de requisitos é uma etapa crítica no desenvolvimento do sistema de gestão de eventos desportivos, pois define as funcionalidades e características essenciais que a plataforma deve possuir para atender às necessidades dos seus utilizadores.

Nos dois subcapítulos seguintes, 3.1.1 e 3.1.2, são apresentados todos os requisitos para o desenvolvimento do projeto, divididos em requisitos funcionais e não funcionais, respetivamente. Os requisitos foram priorizados utilizando o método MoSCoW, que consiste em categorizar cada requisito em um dos quatro grupos seguintes:

- **M - Must have (Deve ter):** Requisitos essenciais que são críticos para o funcionamento do sistema. Sem esses requisitos, o sistema não poderá ser considerado funcional;
- **S - Should have (Deveria ter):** Requisitos importantes, mas não essenciais. Esses requisitos são altamente desejáveis e melhoram significativamente a experiência do utilizador, mas o sistema ainda funcionará sem eles;
- **C - Could have (Poderia ter):** Requisitos desejáveis que podem ser incluídos se houver tempo e recursos disponíveis. Esses requisitos são considerados melhorias ou extras que aumentam a satisfação do utilizador;
- **W - Won't have (Não terá):** Requisitos que foram considerados, mas decididos como não prioritários para esta fase do projeto. Esses requisitos podem ser revisitados em futuras atualizações ou versões do sistema.

### 3.1.1 Requisitos Funcionais

- **RF01:** O sistema deve permitir que os utilizadores se registem fornecendo informações como nome, endereço de email e password. **MoSCoW:** Must have

- **RF02:** Os promotores devem poder adicionar, remover e editar eventos do sistema. **MoSCoW:** Must have
- **RF03:** Os utilizadores devem poder visualizar uma lista dos eventos disponíveis no sistema. **MoSCoW:** Must have
- **RF04:** A lista de eventos deve ser apresentada de forma clara e organizada, destacando informações essenciais como título, data e localização. **MoSCoW:** Should have
- **RF05:** Os utilizadores devem poder visualizar um evento com mais detalhe, clicando no mesmo. **MoSCoW:** Must have
- **RF06:** Os utilizadores devem poder filtrar a lista de eventos por tipo de desporto, facilitando a procura por eventos específicos de interesse. **MoSCoW:** Should have
- **RF07:** Os utilizadores devem poder filtrar a lista de eventos por data (mês), facilitando a procura por eventos específicos de interesse. **MoSCoW:** Should have
- **RF08:** Os utilizadores devem poder realizar a inscrição em eventos disponíveis no sistema. **MoSCoW:** Must have
- **RF09:** Os utilizadores devem poder visualizar todos os eventos em que estão inscritos. **MoSCoW:** Must have
- **RF10:** Os utilizadores devem poder visualizar o histórico de eventos que já aconteceram. **MoSCoW:** Must have

### 3.1.2 Requisitos não Funcionais

- **RNF01 - Compatibilidade:** O sistema deve ser compatível com o Firefox, Chrome e navegadores derivados. Qualquer sistema operacional com compatibilidade com os navegadores mencionados deve ser capaz de interagir com o sistema. **MoSCoW:** Must have
- **RNF02 - Segurança:** Apenas promotores devidamente autenticados terão autorização para inserir, editar e remover eventos desportivos, bem como para visualizar os inscritos em cada evento. **MoSCoW:** Must have
- **RNF03 - Segurança:** Apenas utilizadores autenticados devem conseguir inscrever-se nos eventos desportivos. **MoSCoW:** Must have
- **RNF04 - Desempenho:** O sistema deve permitir ser utilizado por vários utilizadores em simultâneo. **MoSCoW:** Must have
- **RNF05 - Portabilidade:** O sistema deve funcionar em diferentes sistemas operacionais. **MoSCoW:** Should have

### 3.1.3 Atores

Neste projeto, foram identificados dois atores que interagem com o sistema: utilizadores e promotores. Cada um desempenha um papel fundamental na utilização e administração do sistema de gestão de eventos desportivos.

- **Utilizador:** um utilizador no sistema pode inscrever-se em eventos desportivos, visualizar eventos disponíveis e consultar o histórico de eventos passados.
- **Promotor:** é responsável por criar e gerir eventos desportivos na plataforma. Eles podem adicionar novos eventos definindo detalhes como localização, data e requisitos específicos. Além disso, têm a capacidade de editar e remover eventos já existentes, bem como consultar os participantes inscritos em cada evento.

## 3.2 Arquitetura técnica do sistema

A arquitetura planeada para este sistema corresponde a uma aplicação web em Full-stack. Na figura 3.1 é possível observar a ilustração da arquitetura técnica deste sistema.

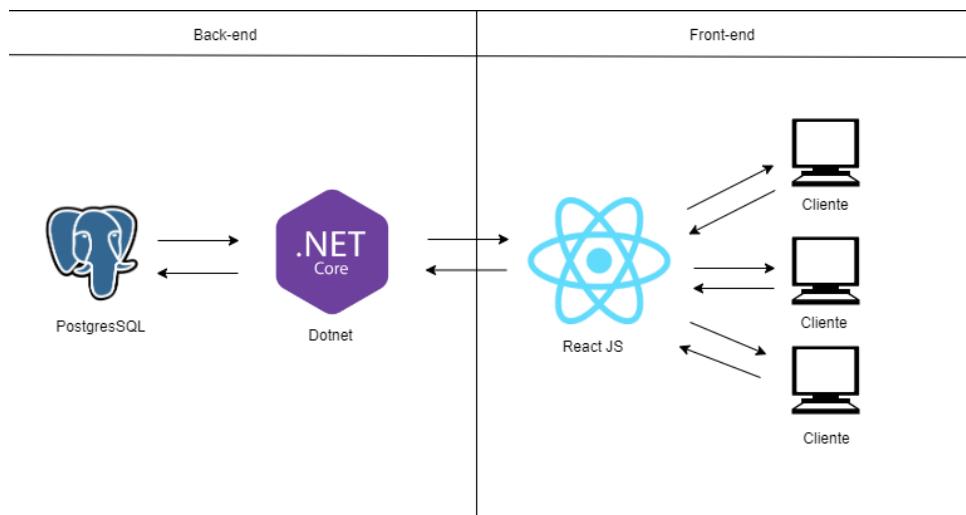


Figura 3.1: Arquitetura técnica do sistema

- **Back-end:** o backend do sistema utiliza o PostgreSQL como base de dados e a plataforma .NET para o desenvolvimento da lógica de negócios e das APIs de comunicação. O PostgreSQL foi escolhido devido à sua robustez, capacidade de escalabilidade e suporte a transações complexas. A plataforma .NET oferece uma ampla gama de bibliotecas e Frameworks que facilitam o desenvolvimento de APIs RESTful seguras e eficientes.
- **Front-end:** no lado do Front-end, o sistema utiliza React, uma biblioteca JavaScript amplamente adotada para a criação de interfaces interativas e responsivas. React permite uma experiência fluida ao interagir com os eventos desportivos, facilitando a navegação intuitiva e a visualização dos dados em tempo real.

- **Integração e Comunicação:** a comunicação entre o Back-end e o Front-end é realizada através de APIs RESTful fornecidas pelo Back-end .NET. Essas APIs garantem a troca segura e eficiente de dados entre o servidor e o cliente, permitindo que o Front-end React apresente informações atualizadas e precisas aos utilizadores finais.

### 3.3 Diagrama de casos de uso

O diagrama de casos de uso representa visualmente as interações entre os atores e o sistema de gestão de eventos desportivos, tal como se pode ver na figura 3.2. Ele descreve as principais funcionalidades oferecidas pelo sistema e como os atores interagem com essas funcionalidades para atingir os seus objetivos. Este diagrama é essencial para compreender os requisitos funcionais do sistema e guiar o desenvolvimento de software de forma a satisfazer as necessidades dos utilizadores.



Figura 3.2: Diagrama de casos de uso

### 3.3.1 User Stories

- **US01:** Como utilizador, quero criar conta no sistema.
- **US02:** Como utilizador, quero efetuar login no sistema.
- **US03:** Como promotor, quero criar conta no sistema.
- **US04:** Como promotor, quero efetuar login no sistema.
- **US05:** Como promotor, quero inserir diferentes eventos no sistema.
- **US06:** Como utilizador, quero visualizar a lista de eventos que se encontram disponíveis.
- **US07:** Como utilizador, quero visualizar as informações detalhadas de cada evento.
- **US08:** Como utilizador, quero filtrar a pesquisa dos diferentes eventos por tipo de desporto.
- **US09:** Como utilizador, quero filtrar a pesquisa dos diferentes eventos por data (mês).
- **US10:** Como utilizador, quero realizar a inscrição num determinado evento.
- **US11:** Como promotor, quero excluir diferentes eventos do sistema.
- **US12:** Como promotor, quero verificar quantos inscritos tem os diferentes eventos.
- **US13:** Como utilizador, quero consultar os eventos em que estou inscrito.
- **US14:** Como utilizador, quero consultar o histórico de eventos já realizados.
- **US15:** Como promotor, quero editar diferentes eventos do sistema.

## 3.4 Diagrama Entity-relationship (ER)

O diagrama Entity-relationship (ER) apresentado é uma representação visual fundamental para a modelação de uma base de dados destinada à gestão de eventos e inscrições de utilizadores, tal como se pode ver na figura 3.3. Este diagrama detalha a estrutura das tabelas que compõem a base de dados, os campos que cada tabela contém e os relacionamentos entre essas tabelas.

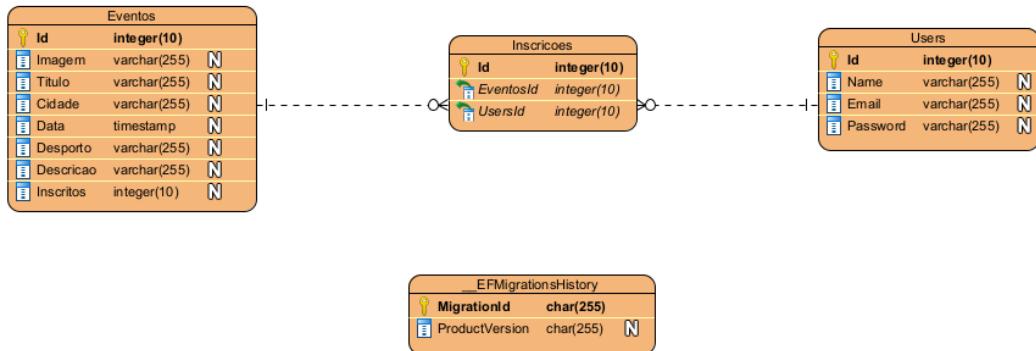


Figura 3.3: Diagrama Entity-relationship (ER)

### 3.4.1 Tabela Eventos

A tabela "Eventos" é central para esta base de dados, pois armazena todas as informações pertinentes aos eventos que serão geridos. Cada evento é identificado por um campo "Id", que é a chave primária da tabela. Este campo é um número inteiro de até 10 dígitos. Além do identificador único, a tabela contém os seguintes campos:

- **Imagen:** Um campo varchar de 255 caracteres que armazena o URL da imagem associada ao evento. Esta imagem pode ser usada em interfaces para representar visualmente o evento.
- **Titulo:** Um campo varchar de 255 caracteres que armazena o título do evento. Este título deve ser breve e descritivo, fornecendo uma visão geral do evento.
- **Cidade:** Um campo varchar de 255 caracteres que indica a cidade onde o evento ocorrerá. Este campo é crucial para a localização geográfica do evento.
- **Data:** Um campo timestamp que regista a data e a hora em que o evento ocorrerá. Este campo permite a ordenação cronológica dos eventos e a gestão de horários.
- **Desporto:** Um campo varchar de 255 caracteres que especifica o tipo de desporto associado ao evento. Este campo é útil para categorizar os eventos por modalidade desportiva.
- **Descrição:** Um campo varchar de 255 caracteres que fornece uma descrição detalhada do evento

- **Inscritos:** Um campo integer de até 10 dígitos que regista o número de participantes inscritos no evento. Este campo é importante para monitorizar a participação e a popularidade dos eventos.

### 3.4.2 Tabela Users

A tabela "Users" é essencial para a gestão dos utilizadores que interagem com os eventos. Cada utilizador é identificado por um campo "Id", que é a chave primária da tabela e um número inteiro de até 10 dígitos. Os demais campos desta tabela incluem:

- **Name:** Um campo varchar de 255 caracteres que armazena o nome do utilizador. Este campo é utilizado para identificar o utilizador de maneira mais simples.
- **Email:** Um campo varchar de 255 caracteres que armazena o endereço de email do utilizador.
- **Password:** Um campo varchar de 255 caracteres que armazena a senha do utilizador. Este campo é armazenado de maneira segura para proteger a privacidade e a segurança dos utilizadores (Hash).

### 3.4.3 Tabela Inscrições

A tabela "Inscrições" faz a ligação entre os utilizadores e os eventos, registando as inscrições dos utilizadores nos eventos. Cada inscrição é identificada por um campo "Id", que é a chave primária da tabela e um número inteiro de até 10 dígitos. Os campos adicionais incluem:

- **EventosId:** Um campo integer de 10 dígitos que atua como chave estrangeira, referenciando o evento na tabela "Eventos". Este campo estabelece o vínculo entre uma inscrição e um evento específico.
- **UsersId:** Um campo integer de 10 dígitos que atua como chave estrangeira, referenciando o utilizador na tabela "Users". Este campo estabelece o vínculo entre uma inscrição e um utilizador específico.

### 3.4.4 Tabela \_\_EFMigrationsHistory

A tabela "\_\_EFMigrationsHistory" é utilizada pelo Entity Framework para manter o histórico de migrações da base de dados. Esta tabela contém os seguintes campos:

- **MigrationId:** Um campo char de 255 caracteres que identifica a migração. Cada migração representa uma alteração na estrutura da base de dados.
- **ProductVersion:** Um campo char de 255 caracteres que regista a versão do produto. Este campo é utilizado para compatibilidade e controlo de versões.

### 3.4.5 Relacionamentos

Os relacionamentos entre as tabelas são fundamentais para a integridade e funcionalidade da base de dados:

- **Eventos e Inscricoes:** Existe um relacionamento um-para-muitos entre "Eventos" e "Inscricoes". Cada evento pode ter várias inscrições associadas, o que significa que muitos utilizadores podem inscrever-se num único evento. No entanto, cada inscrição está vinculada a um único evento específico.
- **Users e Inscricoes:** Existe um relacionamento um-para-muitos entre "Users" e "Inscricoes". Cada utilizador pode inscrever-se em vários eventos, o que significa que um utilizador pode participar em muitos eventos diferentes. No entanto, cada inscrição está vinculada a um único utilizador específico.

## 3.5 Mockups

Os mockups desempenham um papel crucial no desenvolvimento do sistema de gestão de eventos desportivos, fornecendo uma representação visual das interfaces e funcionalidades do sistema antes da implementação final. Eles ajudam a comunicar a aparência e a experiência do utilizador, permitindo que todas as partes interessadas, incluindo programadores, designers e utilizadores finais, compreendam o fluxo das interfaces do sistema.

### 3.5.1 Página de Login

A Figura 3.4 representa a página de login do sistema, onde o utilizador insere o seu email e a sua palavra-passe para efetuar o respetivo login. Caso o utilizador não possua conta, será redirecionado para a página de registo para criar uma nova conta.

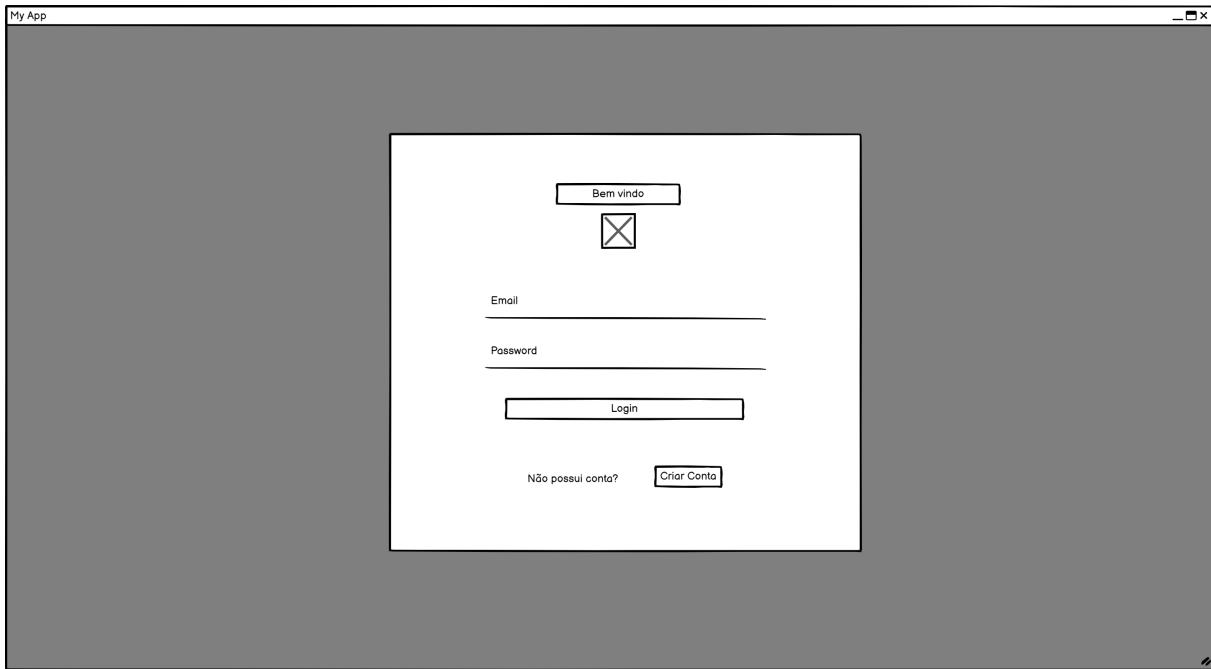


Figura 3.4: Mockup da página de login

### 3.5.2 Página de Registo

A Figura 3.5 representa a página de registo do sistema, onde o utilizador insere o seu nome, email e palavra-passe para criar uma nova conta. Caso o utilizador já possua conta, será redirecionado para a página de login para se autenticar no sistema.

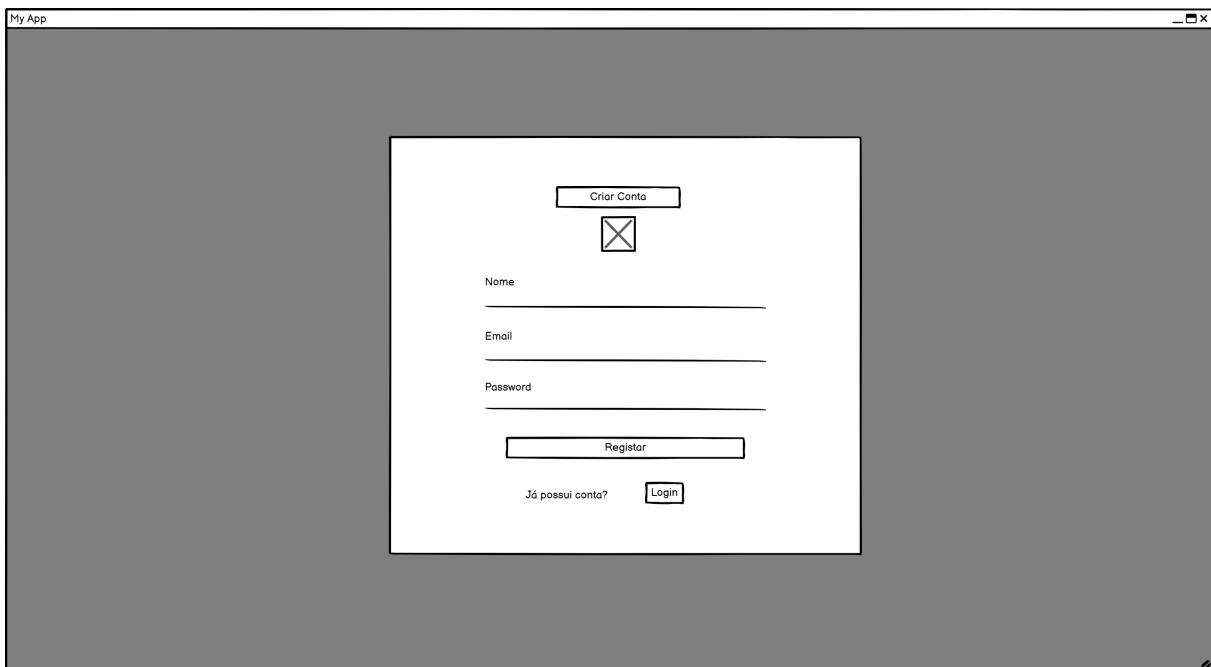


Figura 3.5: Mockup da página de registo

### 3.5.3 Página Inicial

A Figura 3.6 representa a página inicial do sistema, onde um conjunto de imagens é apresentado em slideshow. No Header, é possível ver diferentes botões, como "HomePage"(página atual), "Eventos"e "Histórico". Além disso, é possível visualizar os eventos em que o utilizador está inscrito e fazer logout do sistema.

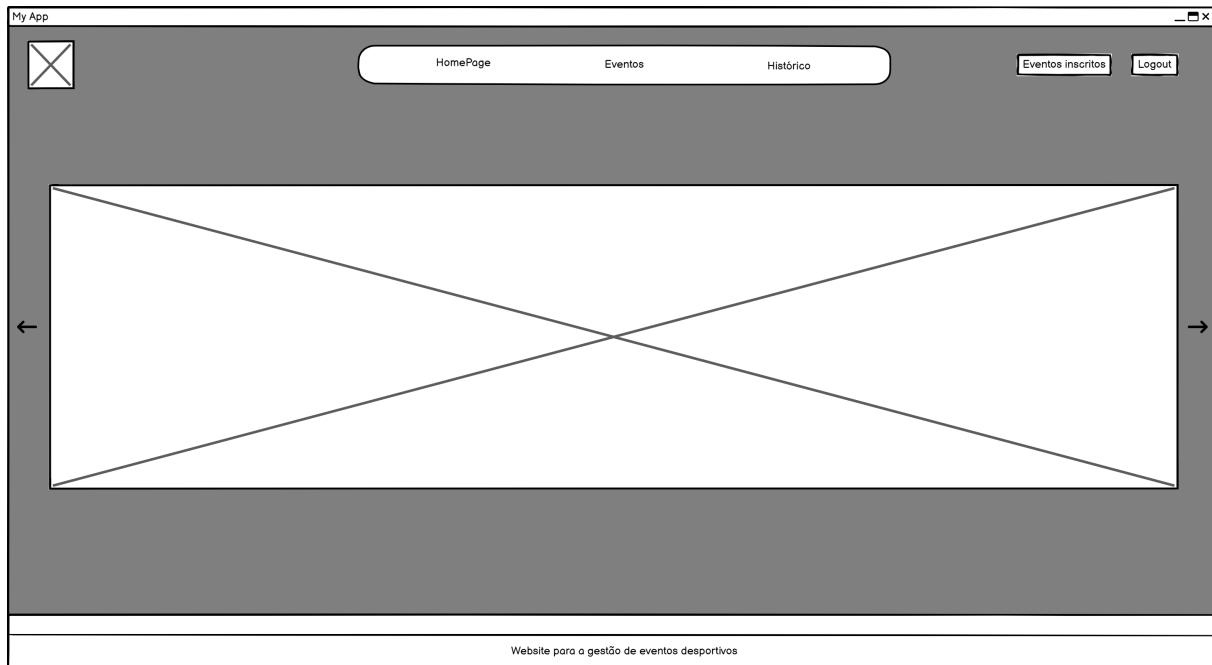


Figura 3.6: Mockup da página inicial

### 3.5.4 Página de Eventos

A Figura 3.7 representa a página de eventos do sistema, onde são apresentados todos os eventos que estão inseridos no sistema. No Header, é possível ver diferentes botões, como "HomePage", "Eventos"(página atual) e "Histórico". Além disso, é possível fazer logout do sistema.

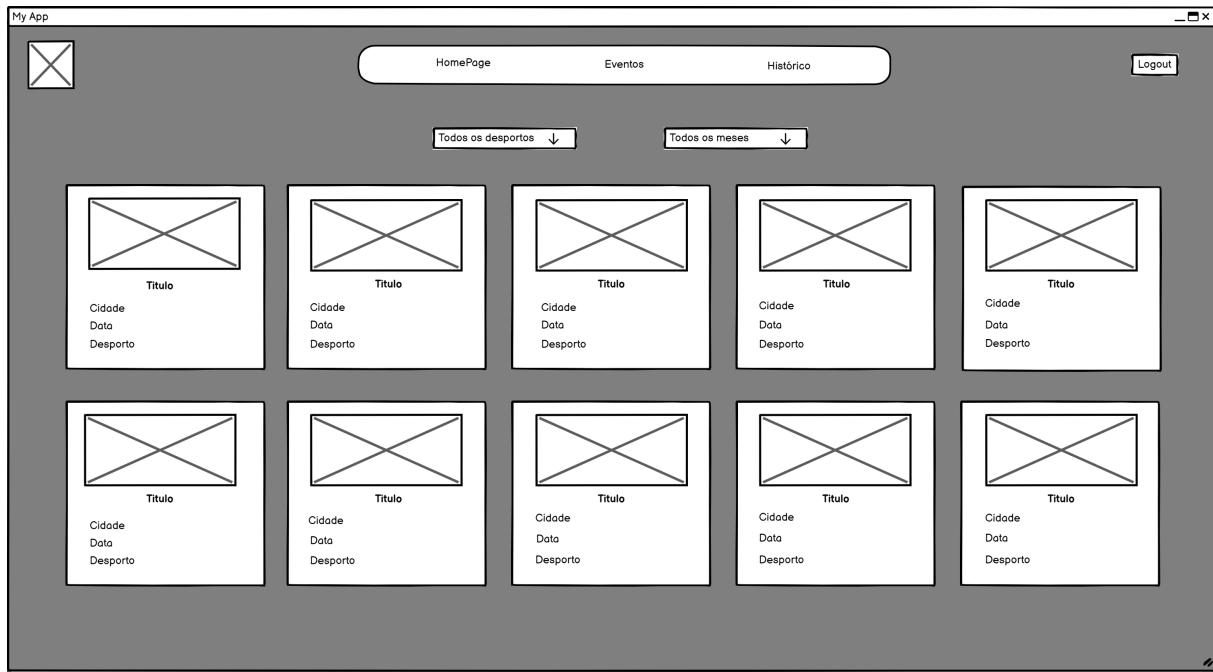


Figura 3.7: Mockup da página de eventos

### 3.5.5 Página de Visualização de um evento

A Figura 3.8 representa a página de visualização de um evento do sistema, onde toda a informação relativa ao evento é apresentada, incluindo a possibilidade de realizar a inscrição no evento. No Header, é possível ver diferentes botões, como "HomePage", "Eventos" e "Histórico". Além disso, é possível fazer logout do sistema.

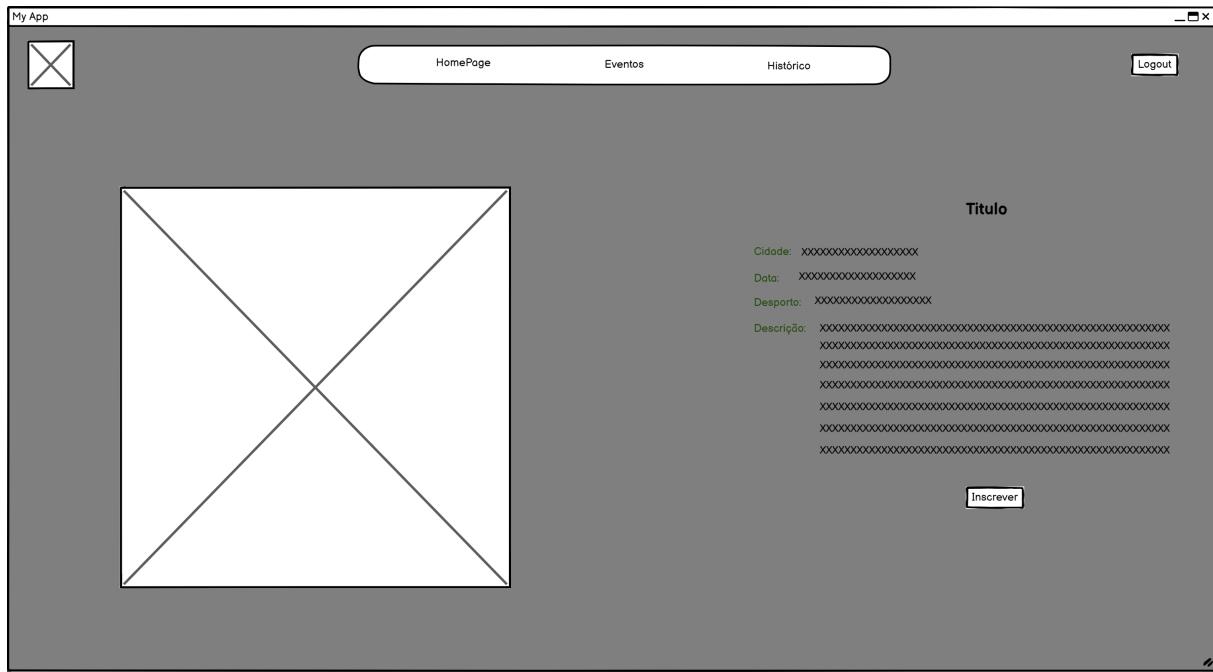


Figura 3.8: Mockup da página de visualização de um evento

### 3.5.6 Página de Eventos inscritos

A Figura 3.9 representa a página de visualização de todos os evento do sistema em que o utilizador está inscrito. No Header, é possível ver diferentes botões, como "HomePage", "Eventos" e "Histórico". Além disso, é possível fazer logout do sistema.

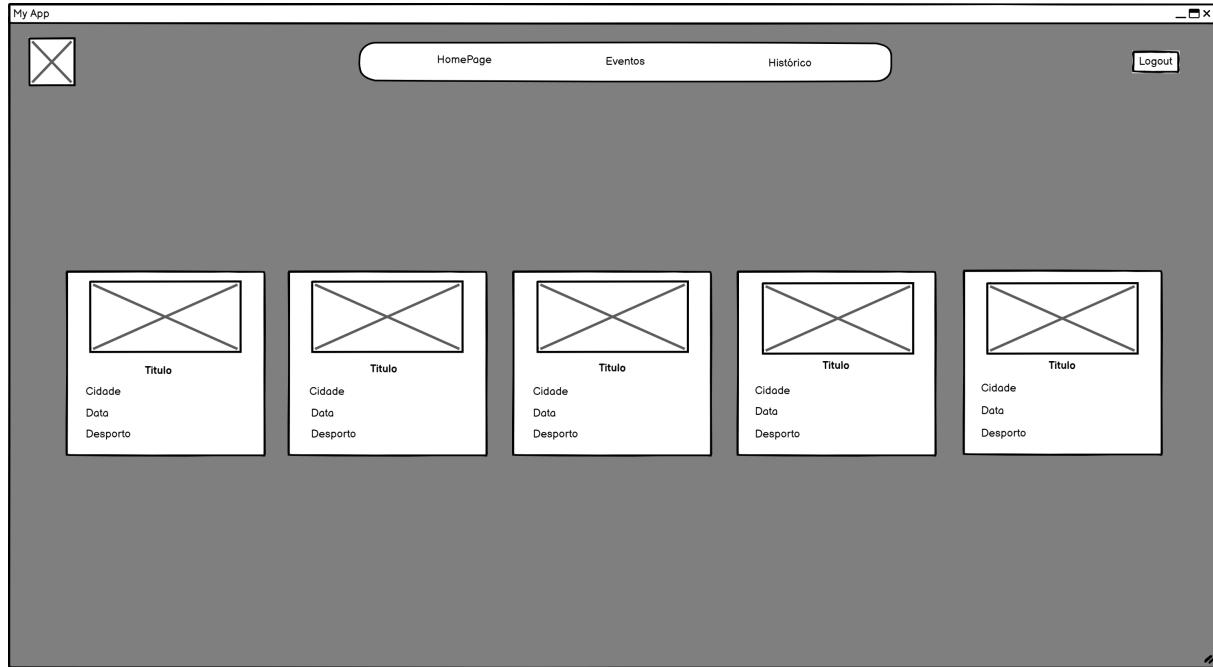


Figura 3.9: Mockup da página de eventos inscritos

### 3.5.7 Página de Histórico

A Figura 3.10 representa a página de histórico de todos os evento do sistema que já ocorreram. No Header, é possível ver diferentes botões, como "HomePage", "Eventos" e "Histórico"(página atual). Além disso, é possível fazer logout do sistema.

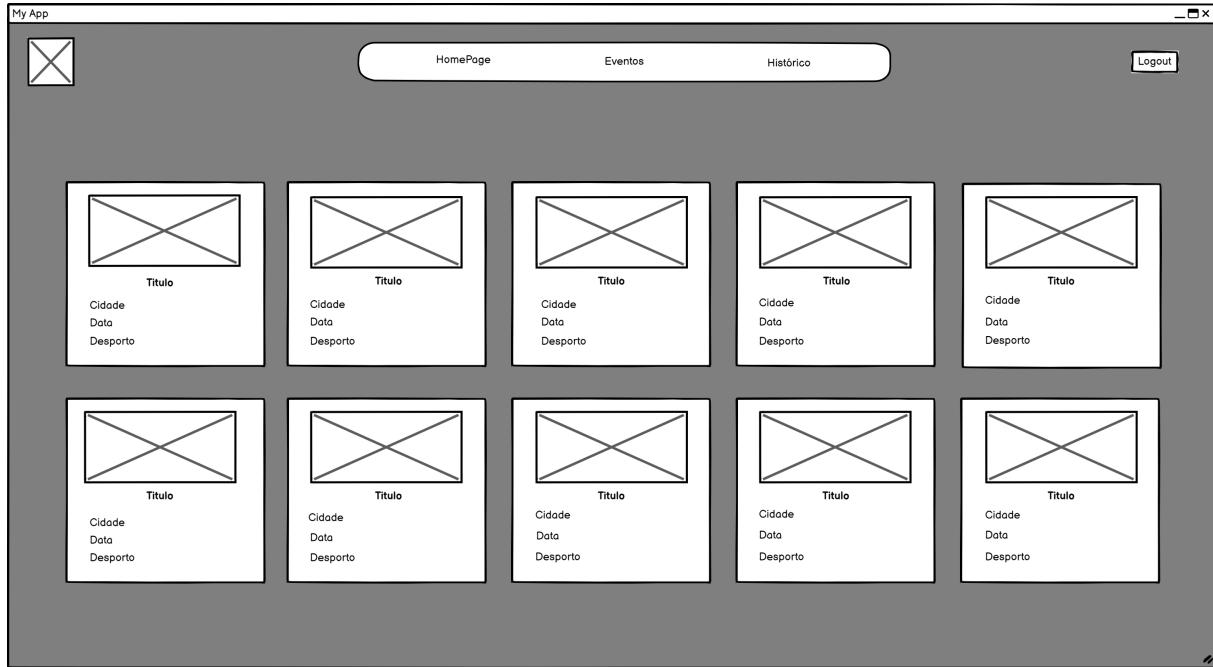


Figura 3.10: Mockup da página de histórico

### 3.5.8 Página do Promotor

A Figura 3.11 representa a página do promotor, onde este pode inserir e remover eventos, assim como verificar quantos inscritos tem um determinado evento. No Header, é possível fazer logout do sistema.

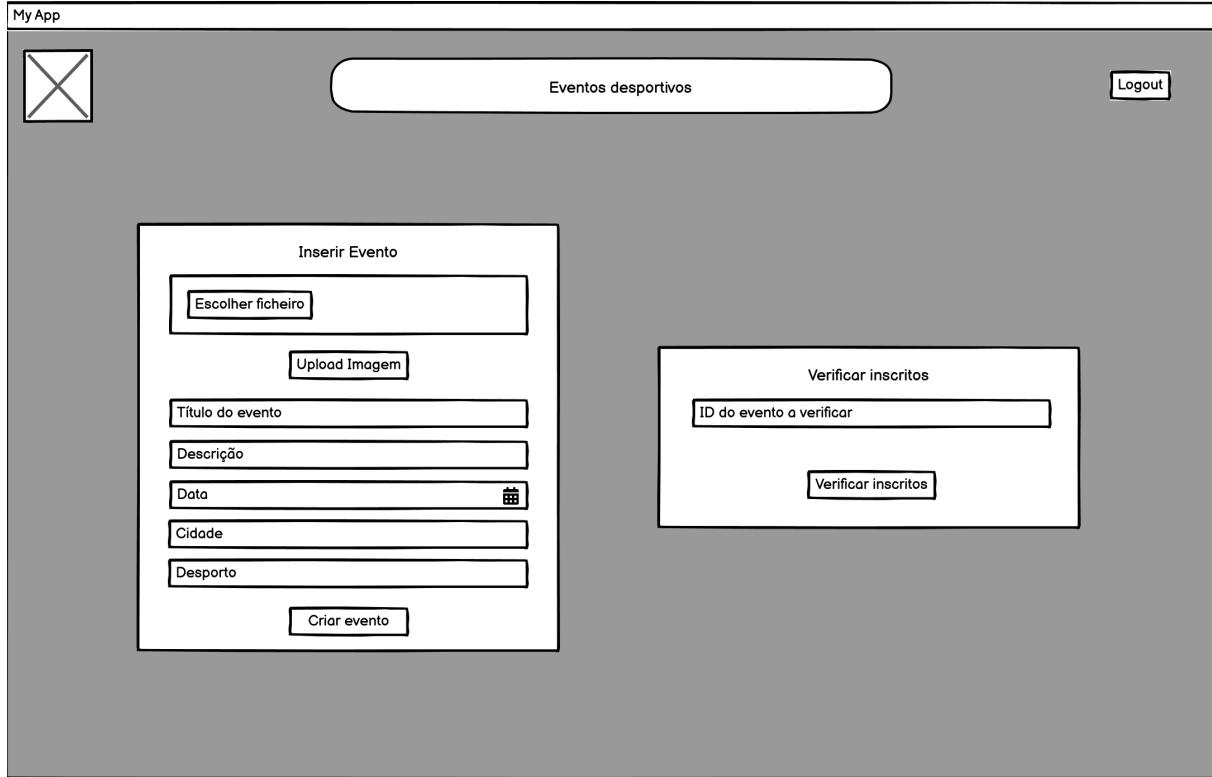


Figura 3.11: Mockup da página do promotor

### 3.5.9 Página de eventos do promotor

A Figura 3.12 representa a página de eventos do promotor, onde este pode remover e editar eventos, assim como verificar o ID associado a cada evento. No Header, é possível fazer logout do sistema.

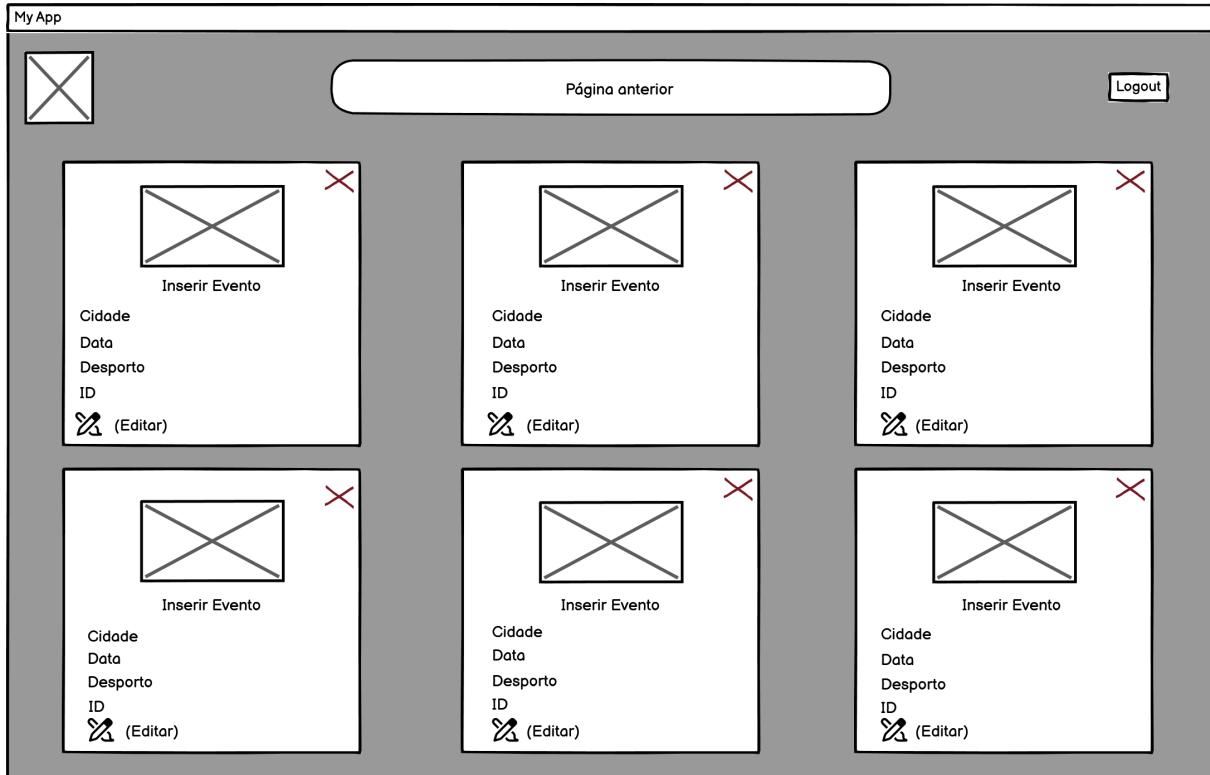


Figura 3.12: Mockup da página de eventos do promotor

# 4. Implementação

Neste capítulo é apresentado o trabalho realizado no desenvolvimento das funcionalidades propostas.

## 4.1 Tecnologias utilizadas

Para o desenvolvimento deste projeto, foram utilizadas as seguintes tecnologias:

### 4.1.1 .NET Core

.NET Core é uma Framework de desenvolvimento Open-source, multiplataforma, mantida pela Microsoft. É utilizada para construir aplicações modernas, escaláveis e de alto desempenho. No projeto, o .NET Core foi utilizado para desenvolver a lógica do servidor e a API RESTful, permitindo uma comunicação eficiente entre o Front-end e a base de dados. (.Net Core)

### 4.1.2 React

React é uma biblioteca JavaScript amplamente utilizada para a construção de interfaces interativas e dinâmicas. Mantida pelo Facebook, permite a criação de componentes reutilizáveis que facilitam o desenvolvimento de interfaces complexas. No projeto, React foi utilizado para desenvolver a interface (Front-end), proporcionando uma experiência fluida e responsiva. (React)

### 4.1.3 PostgresSQL

PostgresSQL é um sistema de gestão de base de dados objeto-relacional, conhecido pela sua robustez, desempenho e extensibilidade. É uma das bases de dados Open-source mais avançadas disponíveis. No projeto, PostgresSQL foi utilizado para armazenar e gerir todos os dados relacionados aos eventos, utilizadores e inscrições, garantindo integridade e consistência dos dados. (PostgresSql)

## 4.2 Ferramentas utilizadas

Para o desenvolvimento deste projeto, foram utilizadas as seguintes ferramentas:

#### 4.2.1 Visual Paradigm

Visual Paradigm é uma ferramenta para a criação de diagramas, fluxogramas e gráficos de forma intuitiva. (Visual Paradigm)

#### 4.2.2 GitLab

GitLab é uma plataforma completa para DevOps, fornecendo repositório Git, integração contínua, entrega contínua, e outras funcionalidades de gestão de projetos. (GitLab)

#### 4.2.3 Rider

Rider é um ambiente de desenvolvimento integrado (IDE) desenvolvido pela JetBrains, especializado em desenvolvimento .NET e outras tecnologias. No projeto, Rider foi utilizado para o desenvolvimento de código em .NET Core. (Rider)

#### 4.2.4 VisualStudio Code

Visual Studio Code é um editor de código-fonte leve, mas poderoso, disponível para Windows, macOS e Linux. No projeto, Visual Studio Code foi utilizado para o desenvolvimento do Front-end em React, beneficiando das suas extensões e funcionalidades de depuração. (Visual Studio Code)

#### 4.2.5 Postman

Postman é uma ferramenta popular para teste de APIs, permitindo enviar solicitações HTTP, testar respostas e automatizar testes. No projeto, Postman foi utilizado para testar e validar as APIs desenvolvidas em .NET Core, garantindo que todas as funcionalidades do Back-end funcionavam corretamente. (Postman)

#### 4.2.6 pgAdmin 4

PgAdmin 4 é uma ferramenta de administração e desenvolvimento para PostgreSQL, oferecendo uma interface gráfica para gerir bases de dados PostgreSQL. No projeto, pgAdmin 4 foi utilizado para a administração da base de dados, incluindo a criação de esquemas, consulta de dados e monitorização do desempenho da base de dados. (pgAdmin 4)

### 4.3 Autenticação

No desenvolvimento deste projeto, a implementação da autenticação não foi primariamente focada na criação de um sistema extremamente seguro e robusto. O objetivo

central não residia na fortificação máxima contra possíveis intrusões ou ataques cibernéticos avançados. Em vez disso, foi priorizada a funcionalidade geral e a usabilidade do sistema, procurando garantir uma experiência fluida e acessível para os utilizadores.

### 4.3.1 Registo

No processo de registo implementado no projeto, o objetivo é permitir que novos utilizadores se inscrevam na plataforma fornecendo informações básicas, como nome, email e password. O endpoint `HttpPost("register")` (listagem 4.1) no controlador `UserController` é responsável por receber os dados do utilizador através do objeto `RegisterDto`, que contém os campos necessários para o registo.

```
[HttpPost("register")]
public IActionResult Register(RegisterDto dto)
{
    var user = new User
    {
        Name = dto.Name,
        Email = dto.Email,
        Password = BCrypt.Net.BCrypt.HashPassword(dto.Password)
    };

    return Created("Sucesso!", _repository.Create(user));
}
```

Listagem 4.1: Endpoint para efetuar o registo no sistema.

- Criação do Utilizador:** O método `Register` recebe um objeto `RegisterDto` com os dados de nome, email e password. Estes dados são utilizados para criar uma nova instância da classe "User".
- Encriptação da Password:** A password fornecida pelo utilizador é encriptada utilizando a biblioteca `BCrypt.Net` antes de ser armazenada na base de dados. A função `BCrypt.Net.BCrypt.HashPassword(dto.Password)` garante que a password do utilizador não seja armazenada em texto simples, aumentando assim a segurança dos dados.
- Persistência na Base de Dados:** O utilizador criado é então passado para o método `Create` do repositório `_repository`, que é responsável por persistir os dados no PostgreSQL.
- Resposta de Sucesso:** Após a criação bem-sucedida do utilizador, uma resposta HTTP 201 (Created) é retornada juntamente com os detalhes do utilizador criado.

### 4.3.2 Login

O endpoint `HttpPost("login")` (listagem 4.2) no controlador `UserController` é responsável por autenticar utilizadores na plataforma, permitindo-lhes acesso às funcionalidades

protegidas mediante credenciais válidas.

```
[HttpPost("login")]
public IActionResult Login(LoginDto dto)
{
    var user = _repository.GetByEmail(dto.Email);

    if (user == null) return BadRequest
    (new { message = "Invalid credentials" });

    if (!BCrypt.Net.BCrypt.Verify(dto.Password, user.Password))
    {
        return BadRequest
        (new { message = "Invalid credentials" });
    }

    bool isPromoter = dto.Email.EndsWith("@promotor.com");

    _httpContextAccessor.HttpContext.Session.SetString
    ("UserId", user.Id.ToString());

    return Ok
    (new { userType = isPromoter ? "promoter" : "user" });
}
```

Listagem 4.2: Endpoint para efetuar login no sistema.

- **Validação de Credenciais:** O método Login começa por procurar o utilizador na base de dados utilizando o método `_repository.GetByEmail(dto.Email)`. Se o utilizador não existir, é retornada uma resposta HTTP 400 (BadRequest) com a mensagem "Credenciais inválidas".
- **Verificação de Password:** Utilizando a biblioteca BCrypt.Net, a password fornecida no DTO é verificada em relação à password armazenada no utilizador. Se não corresponderem, é retornada uma resposta HTTP 400 (BadRequest) com a mensagem "Credenciais inválidas".
- **Identificação do Tipo de Utilizador:** O sistema verifica se o email termina com "@promotor.com". Se sim, o utilizador é identificado como um promotor.
- **Gestão de Sessão:** Após um login bem-sucedido, o ID do utilizador é armazenado na sessão HTTP utilizando `_httpContextAccessor.HttpContext.Session.SetString("UserId", user.Id.ToString())`. Esta abordagem permite rastrear o estado da sessão e autenticar o utilizador em pedidos subsequentes.
- **Resposta de Sucesso:** A resposta HTTP 200 (OK) retorna um objeto JSON indicando o tipo de utilizador ("promotor" ou "utilizador"), dependendo do sufixo do email.

### 4.3.3 Logout

O endpoint `HttpPost("logout")` (listagem 4.3) no controlador `UserController` é responsável por encerrar a sessão de um utilizador autenticado na plataforma, limpando os dados armazenados na sessão HTTP.

```
[HttpPost("logout")]
public IActionResult Logout()
{
    _httpContextAccessor.HttpContext.Session.Clear();

    return Ok(new { message = "Logout bem-sucedido." });
}
```

Listagem 4.3: Endpoint para efetuar logout do sistema.

- **Limpeza da Sessão:** Ao ser acionado, o método `Logout` utiliza `_httpContextAccessor.HttpContext.Session.Clear()` para remover todos os dados armazenados na sessão HTTP do utilizador atual. Isso inclui o ID do utilizador e quaisquer outras informações relacionadas à sessão.
- **Resposta de Sucesso:** Após limpar a sessão com sucesso, o método retorna uma resposta HTTP 200 (OK) com um objeto JSON contendo uma mensagem indicando que o logout foi bem-sucedido.

## 4.4 Back-end

No desenvolvimento do Back-end, foi implementado um sistema Create, Read, Update, Delete (CRUD) para a gestão de eventos.

### 4.4.1 Criação de um evento

O endpoint `HttpPost` (listagem 4.4) no controlador de eventos é responsável por criar um novo evento na plataforma, garantindo que a data do evento seja ajustada ao fuso horário UTC antes de ser armazenada na base de dados.

```
[HttpPost]
public async Task<ActionResult<Evento>>
PostEvento(Evento evento)
{
    var userId = HttpContext.Session.GetString("UserId");
    if (userId == null)
    {

        return Unauthorized(new
        { message = "Utilizador não autenticado." });
    }
```

```

    }

    evento.Data = evento.Data.ToUniversalTime();

    _context.Eventos.Add(evento);
    await _context.SaveChangesAsync();

    return CreatedAtAction(nameof(GetEventos),
        new { id = evento.Id }, evento);
}

```

Listagem 4.4: Endpoint para criar um evento no sistema.

- **Recuperação do ID do Utilizador da Sessão:** A linha `var userId = HttpContext.Session.GetString("UserId");` tenta recuperar o valor do ID do utilizador armazenado na sessão HTTP. Isso é usado para identificar o utilizador autenticado numa aplicação.
- **Verificação da Existência do ID do Utilizador:** A seguir, o código verifica se `userId` é null. Se for null, significa que o ID do utilizador não foi encontrado na sessão, o que indica que o utilizador não está autenticado.
- **Resposta de Não Autorizado:** Se o ID do utilizador não estiver presente (ou seja, o utilizador não está autenticado), o método retorna uma resposta HTTP 401 (Unauthorized) com um corpo contendo uma mensagem explicativa: `return Unauthorized(new message = "Utilizador não autenticado.");`. Essa resposta informa o cliente que a requisição não pode ser concluída porque o utilizador não está autenticado, e é fornecido um motivo para isso.
- **Ajuste da Data para UTC:** A linha `evento.Data = evento.Data.ToUniversalTime();` assegura que a data do evento seja convertida para o fuso horário UTC, garantindo consistência temporal independentemente do fuso horário em que o evento foi criado.
- **Adição do Evento ao Contexto:** O método `_context.Eventos.Add(evento);` adiciona o novo evento ao contexto do Entity Framework, preparando-o para ser persistido na base de dados.
- **Salvação Assíncrono:** Utilizando `await _context.SaveChangesAsync();`, o método assegura que as alterações sejam salvas de forma assíncrona, melhorando a performance e a escalabilidade da aplicação.
- **Retorno da Resposta:** Após salvar o evento, o método retorna uma resposta HTTP 201 (Created) utilizando `return CreatedAtAction(nameof(GetEventos), new id = evento.Id , evento);`, indicando que o evento foi criado com sucesso e fornecendo a localização do recurso recém-criado.

#### 4.4.2 Listagem de eventos

O endpoint `HttpGet` (listagem 4.5) no controlador de eventos é responsável por retornar todos os eventos presentes na base de dados. Essa funcionalidade permite que os utilizadores visualizem uma lista completa de eventos disponíveis no sistema.

```
[HttpGet]
public async Task<ActionResult<IEnumerable<Evento>>>
GetEventos()
{
    return await _context.Eventos.ToListAsync();
}
```

Listagem 4.5: Endpoint para listar todos os eventos do sistema.

- **Anotação `HttpGet`:** A anotação `[HttpGet]` indica que este método responderá a requisições HTTP GET, que são normalmente utilizadas para recuperação de dados.
- **Recuperação de Eventos:** O método `GetEventos` utiliza o Entity Framework para aceder à base de dados e recuperar todos os registo da tabela `Eventos`. A chamada `await _context.Eventos.ToListAsync();` executa a consulta de forma assíncrona, retornando uma lista de todos os eventos.
- **Retorno da Resposta:** O método retorna a lista de eventos dentro de um objeto `ActionResult`. Utilizar `ActionResult<IEnumerable<Evento>>` permite flexibilidade no tipo de resposta que pode ser retornada, facilitando o tratamento de diferentes cenários (ex: retorno de erros).

#### 4.4.3 Visualização detalhada de um evento

O endpoint `HttpGet` (listagem 4.6) com parâmetro `id` no controlador de eventos é responsável por retornar os detalhes de um evento específico com base no seu identificador único (ID). Essa funcionalidade permite que os utilizadores acedam a informações detalhadas sobre um evento individual.

```
[HttpGet("{id}")]
public async Task<ActionResult<Evento>>
GetEvento(int id)
{
    var evento = await _context.Eventos.FindAsync(id);

    if (evento == null)
    {
        return NotFound();
    }

    return evento;
}
```

---

Listagem 4.6: Endpoint para visualizar os detalhes de um evento perante o ID

- **Anotação `HttpGet` com Parâmetro:** A anotação `[HttpGet("id")]` indica que este método responderá a requisições HTTP GET e aceitará um parâmetro na URL, representado por `id`. Este parâmetro é utilizado para identificar de forma única o evento que se deseja recuperar.
- **Procura do Evento:** O método `GetEvento` utiliza o Entity Framework para procurar o evento na base de dados utilizando o identificador fornecido. A chamada `await _context.Eventos.FindAsync(id);` executa a procura de forma assíncrona, retornando o evento correspondente ao `id` fornecido.
- **Verificação de Existência:** Após a tentativa de recuperação do evento, o método verifica se o evento foi encontrado. Se evento for `null`, significa que não existe um evento com o identificador fornecido, e a resposta retornada é `NotFound()`.
- **Retorno da Resposta:** Se o evento for encontrado, ele é retornado dentro de um objeto `ActionResult<Evento>`. Isso permite que o evento seja encapsulado na resposta HTTP, facilitando a visualização dos detalhes do evento pelo cliente.

#### 4.4.4 Exclusão de um evento

O endpoint `HttpDelete` (listagem 4.7) com parâmetro `id` no controlador de eventos é responsável por excluir um evento específico da base de dados com base no seu identificador único (ID). Esta funcionalidade permite que promotores de eventos removam eventos que não são mais necessários ou que foram cancelados.

```
[HttpDelete("{id}")]
public async Task<IActionResult> DeleteEvento(int id)
{
    var userId = HttpContext.Session.GetString("UserId");
    if (userId == null)
    {
        return Unauthorized(new
        { message = "Utilizador_nao_autenticado." });
    }

    var evento = await _context.Eventos.FindAsync(id);
    if (evento == null)
    {
        return NotFound();
    }

    _context.Eventos.Remove(evento);
    await _context.SaveChangesAsync();
}
```

```
    return NoContent();
}
```

Listagem 4.7: Endpoint para excluir um evento do sistema

- **Recuperação do ID do Utilizador da Sessão:** A linha `var userId = HttpContext.Session.GetString("UserId");` tenta recuperar o valor do ID do utilizador armazenado na sessão HTTP. Isso é usado para identificar o utilizador autenticado numa aplicação.
- **Verificação da Existência do ID do Utilizador:** A seguir, o código verifica se `userId` é null. Se for null, significa que o ID do utilizador não foi encontrado na sessão, o que indica que o utilizador não está autenticado.
- **Resposta de Não Autorizado:** Se o ID do utilizador não estiver presente (ou seja, o utilizador não está autenticado), o método retorna uma resposta HTTP 401 (Unauthorized) com um corpo contendo uma mensagem explicativa: `return Unauthorized(new message = "Utilizador não autenticado.");`. Essa resposta informa o cliente que a requisição não pode ser concluída porque o utilizador não está autenticado, e é fornecido um motivo para isso.
- **Anotação `HttpDelete` com Parâmetro:** A anotação `[HttpDelete("id")]` indica que este método responderá a requisições HTTP DELETE e aceitará um parâmetro na URL, representado por `id`. Este parâmetro é utilizado para identificar de forma única o evento que se deseja excluir.
- **Procura do Evento:** O método `DeleteEvento` utiliza o Entity Framework para procurar o evento na base de dados utilizando o identificador fornecido. A chamada `await _context.Eventos.FindAsync(id);` executa a procura de forma assíncrona, retornando o evento correspondente ao `id` fornecido.
- **Verificação de Existência:** Após a tentativa de recuperação do evento, o método verifica se o evento foi encontrado. Se evento for null, significa que não existe um evento com o identificador fornecido, e a resposta retornada é `NotFound()`. Isso evita tentativas de exclusão de eventos inexistentes.
- **Remoção do Evento:** Se o evento for encontrado, o método `_context.Eventos.Remove(evento);` remove o evento do contexto da base de dados.
- **Confirmação da Exclusão:** A chamada `await _context.SaveChangesAsync();` persiste a alteração na base de dados, confirmando a exclusão do evento.
- **Resposta de Sucesso:** Após a remoção bem-sucedida, o método retorna `NoContent()`, que é um código de status HTTP 204, indicando que a requisição foi bem-sucedida, mas não há conteúdo a ser retornado.

#### 4.4.5 Edição de um evento

O endpoint `HttpPut` (listagem 4.8) com parâmetro `id` no controlador de eventos é responsável por editar um evento específico da base de dados com base no seu identificador único (ID). Esta funcionalidade permite que promotores de eventos editem eventos.

```
[HttpPost("{id}")]
public async Task<IActionResult>
PutEvento(int id, Evento evento)
{
    var eventoExistente =
        await _context.Eventos.FindAsync(id);
    if (eventoExistente == null)
    {
        return NotFound(new
        { message = "Evento_nao_encontrado." });
    }

    evento.Data = evento.Data.ToUniversalTime();

    eventoExistente.Titulo =
        evento.Titulo ?? eventoExistente.Titulo;
    eventoExistente.Cidade =
        evento.Cidade ?? eventoExistente.Cidade;
    eventoExistente.Data =
        evento.Data != default ? evento.Data :
        eventoExistente.Data;
    eventoExistente.Desporto =
        evento.Desporto ?? eventoExistente.Desporto;

    try
    {
        await _context.SaveChangesAsync();
        return Ok(eventoExistente);
    }
    catch (Exception ex)
    {
        return StatusCode(500, new
        { message = "Erro ao atualizar evento",
          error = ex.Message });
    }
}
```

Listagem 4.8: Endpoint para editar um evento do sistema

- **Definição do Método HTTP PUT:** O método está decorado com o atributo `[HttpPost("id")]`, indicando que ele será responsável por lidar com as requisições HTTP PUT. Esse tipo de requisição é utilizado para atualizar um recurso existente. O parâmetro `id` na URL representa o identificador único do evento que será atualizado.

- **Procura do Evento Existente:** A linha var eventoExistente = await \_context.Eventos.FindAsync(id); tenta localizar o evento na base de dados usando o ID fornecido na URL. Isso é feito de forma assíncrona para não bloquear a execução do código.
- **Verificação da Existência do Evento:** Caso o evento não seja encontrado (ou seja, eventoExistente for null), o código retorna uma resposta HTTP 404 (Not Found) indicando que o evento com o ID fornecido não existe: return NotFound(new message = "Evento não encontrado.");.
- **Ajuste da Data para UTC:** A linha evento.Data = evento.Data.ToUniversalTime(); assegura que a data do evento seja convertida para o fuso horário UTC, garantindo consistência temporal independentemente do fuso horário em que o evento foi criado.
- **Atualização dos Campos Permitidos:** O código atualiza os campos do evento existente com os valores fornecidos no objeto evento. Para evitar sobreescriver dados que não foram alterados, os campos são atualizados de forma condicional.
- **Guardar as Alterações:** As alterações feitas ao objeto eventoExistente são guardadas na base de dados usando o método assíncrono await \_context.SaveChangesAsync();. Isso garante que as alterações sejam persistidas de forma eficiente.
- **Tratamento de erros:** O código é envolvido num bloco try-catch para capturar possíveis exceções durante o processo de guardar. Se ocorrer um erro (por exemplo, problemas de conexão com a base de dados), o método retorna uma resposta HTTP 500 (Internal Server Error) com uma mensagem de erro: return StatusCode(500, new message = "Erro ao atualizar evento", error = ex.Message);.
- **Retorno de Sucesso:** Caso o evento seja atualizado com sucesso, o método retorna uma resposta HTTP 200 (OK) com o evento atualizado no corpo da resposta: return Ok(eventoExistente);.

#### 4.4.6 Visualização de eventos inscritos

O endpoint `HttpGet("Inscricoes")` (listagem 4.9) no controlador de eventos permite que um utilizador autenticado recupere uma lista de eventos nos quais está inscrito. Esta funcionalidade é crucial para fornecer aos utilizadores uma visão clara dos eventos nos quais participam, facilitando o acompanhamento e a gestão das suas inscrições.

```
[HttpGet("Inscricoes")]
public async Task<ActionResult<IEnumerable<Evento>>>
GetEventosInscritos()
{
    var userId =
        _httpContextAccessor.HttpContext.Session.GetString
        ("UserId");

    if (userId == null)
```

```

    {
        return BadRequest
        (new { message = "User_nao_autenticado." });
    }

    var inscricoes = await _context.Inscricoes
        .Where(i => i.UserId == userId)
        .ToListAsync();

    var eventoIds =
        inscricoes.Select(i => i.EventoId).ToList();
    var eventosInscritos = await _context.Eventos
        .Where(e => eventoIds.Contains(e.Id))
        .ToListAsync();

    return eventosInscritos;
}

```

Listagem 4.9: Endpoint para visualizar eventos inscritos

- Anotação `HttpGet` com Rota Personalizada:** A anotação `[HttpGet("Inscricoes")]` indica que este método responderá a requisições HTTP GET para a rota `api/Inscricoes`, proporcionando um ponto de acesso específico para recuperar eventos inscritos pelo utilizador.
- Obtenção do `UserId` da Sessão:** O método `GetEventosInscritos` utiliza `_httpContextAccessor.HttpContext.Session.GetString("UserId")` para obter o identificador do utilizador armazenado na sessão. Este passo é essencial para garantir que a requisição está a ser feita por um utilizador autenticado.
- Verificação de Autenticação:** Se o `userId` obtido for null, significa que o utilizador não está autenticado, e o método retorna um erro `BadRequest` com uma mensagem apropriada. Esta verificação previne o acesso não autorizado à funcionalidade.
- Recuperação das Inscrições do Utilizador:** O método então procura todas as inscrições do utilizador na base de dados, utilizando o Entity Framework para executar uma consulta assíncrona sobre a tabela de inscrições (`_context.Inscricoes`). A condição `Where(i => i.UserId == userId)` garante que apenas as inscrições do utilizador autenticado sejam recuperadas.
- Recuperação dos Eventos Correspondentes:** Após obter as inscrições, o método extrai os identificadores dos eventos (`eventoIds`) e realiza uma segunda consulta para recuperar os detalhes dos eventos correspondentes (`_context.Eventos.Where(e => eventoIds.Contains(e.Id))`).
- Resposta com os Eventos Inscritos:** Finalmente, a lista de eventos nos quais o utilizador está inscrito é retornada como resposta da requisição. Este retorno é realizado de forma assíncrona, garantindo a eficiência e a responsividade do sistema.

#### 4.4.7 Inscrição em eventos

O endpoint `HttpPost("{id}/Inscricao")` (listagem 4.10) no controlador de eventos permite que um utilizador autenticado se inscreva em um evento específico. Esta funcionalidade é essencial para a interação do utilizador com a plataforma, proporcionando uma maneira eficiente e segura de participar em eventos.

```
[HttpPost("{id}/Inscricao")]
public async Task<ActionResult<Evento>>
Inscriver(int id)
{
    var evento = await _context.Eventos.FindAsync(id);
    if (evento == null)
    {
        return NotFound();
    }

    var userId =
        HttpContext.Session.GetString("UserId");
    if (userId == null)
    {
        return Unauthorized(
            new { message = "User_nao_autenticado." });
    }

    var inscricaoExistente =
        _context.Inscricoes.Any(
            i => i.EventoId == id && i.UtilizadorId == userId);
    if (inscricaoExistente)
    {
        return BadRequest(
            new { message =
                "User_ja_esta_inscrito_neste_evento" });
    }

    evento.Inscritos++;
    _context.Entry(evento).State = EntityState.Modified;

    var inscricao = new Inscricao
    {
        EventoId = evento.Id,
        UtilizadorId = userId
    };

    _context.Inscricoes.Add(inscricao);

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
```

```

    {
        if (!EventoExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return NoContent();
}

```

Listagem 4.10: Endpoint para realizar a inscrição em eventos

- **Anotação `HttpPost` com Rota Personalizada:** A anotação `[HttpPost("id/Inscricao")]` indica que este método responderá a requisições HTTP POST para a rota `api/id/Inscricao`, onde `id` é o identificador do evento. Isto permite que os utilizadores se inscrevam em eventos específicos.
- **Procura do Evento pelo ID:** O método começa a procurar o evento correspondente ao `id` fornecido na URL utilizando o Entity Framework (`_context.Eventos.FindAsync(id)`). Se o evento não for encontrado, retorna um erro `NotFound`.
- **Obtenção do UserId da Sessão:** O método utiliza `HttpContext.Session.GetString("UserId")` para obter o identificador do utilizador armazenado na sessão. Esta etapa é crucial para garantir que apenas utilizadores autenticados se possam inscrever em eventos.
- **Verificação de Autenticação:** Se o `userId` for null, o método retorna um erro `Unauthorized`, indicando que o utilizador não está autenticado.
- **Verificação de Inscrição Existente:** O método verifica se o utilizador já está inscrito no evento (`_context.Inscricoes.Any(i => i.EventoId == id && i.UtilizadorId == userId)`). Se a inscrição já existir, retorna um erro `BadRequest` com uma mensagem apropriada.
- **Incremento do Número de Inscritos:** Se todas as verificações anteriores passarem, o método incrementa o número de inscritos no evento (`evento.Inscritos++`) e marca o estado da entidade como modificado (`_context.Entry(evento).State = EntityState.Modified`).
- **Criação de Nova Inscrição:** O método então cria uma nova inscrição com o ID do evento e o ID do utilizador da sessão, adicionando-a ao contexto (`_context.Inscricoes.Add(inscricao)`).
- **Guardar as Alterações na Base de Dados:** As alterações são salvas de forma assíncrona utilizando `await _context.SaveChangesAsync()`. Se ocorrer uma exceção

de concorrência de atualização (DbUpdateConcurrencyException), o método verifica se o evento ainda existe. Se não, retorna um erro NotFound; caso contrário, a exceção é lançada novamente.

- **Resposta Sem Conteúdo:** Se a operação for bem-sucedida, o método retorna uma resposta NoContent, indicando que a inscrição foi realizada com sucesso, mas sem fornecer um corpo de resposta.

#### 4.4.8 Visualização do numero de inscritos num evento

O endpoint `HttpGet("id/Inscritos")` (listagem 4.11) permite obter o número de inscritos num evento específico. Esta funcionalidade é essencial para que os promotores possam controlar a participação nos eventos, proporcionando uma visão clara do engajamento dos utilizadores.

```
[HttpGet("{ id }/Inscritos")]
public async Task<ActionResult<int>> GetInscritos(int id)
{
    var userId = HttpContext.Session.GetString("UserId");
    if (userId == null)
    {
        return Unauthorized(new
        { message = "Utilizador não autenticado." });
    }

    var evento = await _context.Eventos.FindAsync(id);

    if (evento == null)
    {
        return NotFound();
    }

    return Ok(evento.Inscritos);
}

private bool EventoExists(int id)
{
    return _context.Eventos.Any(e => e.Id == id);
}
```

Listagem 4.11: Endpoint para visualizar o numero de inscritos num evento

- **Rota e Método GET:** O método está associado a um endpoint HTTP GET, e a URL inclui um parâmetro `id/Inscritos`. Este endpoint será utilizado para visualizar os inscritos de um evento específico, identificado pelo `id` fornecido.
- **Verificação de Autenticação:** O código tenta obter o ID do utilizador a partir da sessão com `HttpContext.Session.GetString("UserId")`. Se o `userId` for null, significa que o utilizador não está autenticado, então o método retorna um erro 401

(Unauthorized) com uma mensagem informando que o usuário não está autenticado: return Unauthorized(new message = "Utilizador não autenticado.");

- **Procura do Evento:** O evento é recuperado da base de dados com o método FindAsync(id), utilizando o ID fornecido na URL. Se o evento não for encontrado (ou seja, evento for null), o código retorna uma resposta 404 (Not Found), indicando que o evento não foi encontrado.
- **Retorno dos Inscritos:** Se o evento for encontrado, o método retorna uma resposta 200 (OK) com a lista de inscritos associada ao evento. Isso é feito com: return Ok(evento.Inscritos);
- **Método Auxiliar (EventoExists):** O método EventoExists verifica se um evento existe na base de dados com o ID fornecido. Ele retorna true se o evento com o ID informado existir, e false caso contrário.

#### 4.4.9 Visualização do histórico de eventos

O endpoint HttpGet("historico") (listagem 4.12) permite obter uma lista de eventos passados, ou seja, eventos cuja data já tenha expirado. Esta funcionalidade é fundamental para que os utilizadores possam consultar eventos anteriores nos quais participaram ou estão interessados.

```
[HttpGet("historico")]
public IActionResult GetEventosHistoricos()
{
    try
    {
        var eventosHistoricos = _context.Eventos
            .Where(e => DateTime.Compare
                (e.Data.ToLocalTime(), DateTime.Now) < 0)
            .ToList();
        return Ok(eventosHistoricos);
    }
    catch (Exception ex)
    {
        return StatusCode
            (500, $"Erro_interno_do_servidor:{ex.Message}");
    }
}
```

Listagem 4.12: Endpoint para visualizar o histórico de eventos já realizados

- **Anotação HttpGet com Rota Personalizada:** A anotação [HttpGet("historico")] indica que este método responderá a requisições HTTP GET para a rota api/historico. Esta rota é utilizada para recuperar eventos passados.
- **Filtragem de Eventos Passados:** O método começa a procurar todos os eventos cujo campo Data seja anterior à data e hora atuais. Isso é feito utilizando o Entity

Framework e uma expressão lambda (`DateTime.Compare(e.Data.ToLocalTime(), DateTime.Now) < 0`). Esta comparação converte a data do evento para o horário local antes de compará-la com a data e hora atuais.

- **Retorno dos Eventos Históricos:** Se a procura for bem-sucedida, o método retorna a lista de eventos históricos encapsulada em uma resposta Ok.
- **Tratamento de Exceções:** O bloco try-catch é utilizado para capturar e tratar possíveis exceções que possam ocorrer durante a execução do método. Se ocorrer uma exceção, o método retorna uma resposta StatusCode(500) com uma mensagem de erro detalhando o problema.

#### 4.4.10 Upload de imagens

O endpoint `HttpPost("UploadImagem")` (listagem 4.13) permite que os utilizadores enviem imagens para o servidor. Esta funcionalidade é crucial para que os promotores possam adicionar imagens aos eventos, melhorando a personalização e a experiência visual da plataforma.

```
[HttpPost("UploadImagem")]
public async Task<IActionResult>
UploadImagem([FromForm] IFormFile file)
{
    if (file == null || file.Length == 0)
        return BadRequest("Nenhum arquivo enviado.");

    try
    {
        var uploadsFolder = Path.Combine("", "media");
        if (!Directory.Exists(uploadsFolder))
            Directory.CreateDirectory(uploadsFolder);

        var uniqueFileName =
            Guid.NewGuid().ToString() + "_" + file.FileName;
        var filePath =
            Path.Combine(uploadsFolder, uniqueFileName);

        using (var stream =
            new FileStream(filePath, FileMode.Create))
        {
            await file.CopyToAsync(stream);
        }

        return Ok(new { filePath });
    }
    catch (Exception ex)
    {
        return StatusCode
            (500, $"Erro interno do servidor: {ex.Message}");
    }
}
```

```
    }  
}
```

Listagem 4.13: Endpoint para dar upload de imagens

- **Anotação `HttpPost` com Rota Personalizada:** A anotação `[HttpPost("UploadImagem")]` indica que este método responde a requisições HTTP POST para a rota `api/UploadImagem`. Esta rota é utilizada para fazer o upload de imagens.
- **Recepção e Validação do Arquivo:** O método começa recebendo o arquivo enviado através do parâmetro `[FromForm] IFormFile file`. Ele valida se o arquivo não é nulo e se possui conteúdo (`file.Length > 0`). Caso contrário, retorna um `BadRequest` com a mensagem "Nenhum arquivo enviado.".
- **Definição do Diretório de Uploads:** Define o diretório onde as imagens serão armazenadas (`media`). Se o diretório não existir, ele é criado utilizando `Directory.CreateDirectory`.
- **Criação de Nome Único para o Arquivo:** Para evitar conflitos de nomes, é criado um nome único para o arquivo utilizando `Guid.NewGuid().ToString()` concatenado com o nome original do arquivo.
- **Salvação do Arquivo no Servidor:** O arquivo é salvo no servidor utilizando um `FileStream`. O método `CopyToAsync` é utilizado para copiar o conteúdo do arquivo para o caminho especificado.
- **Retorno da Resposta de Sucesso:** Se o upload for bem-sucedido, o método retorna um `Ok` com o caminho do arquivo (`filePath`) na resposta.
- **Tratamento de Exceções:** Um bloco `try-catch` é utilizado para capturar e tratar quaisquer exceções que possam ocorrer durante o processo de upload. Em caso de erro, retorna um `StatusCode(500)` com uma mensagem de erro detalhando o problema.

## 4.5 Front-end

### 4.5.1 Configuração das rotas

O AppRouter (listagem 4.14) é um componente fundamental que utiliza o React Router para gerir as rotas da aplicação, direcionando os utilizadores para diferentes páginas com base nos URLs correspondentes.

- **Utilização do React Router:** Importa os componentes necessários do React Router (`BrowserRouter`, `Routes` e `Route`) para configurar o roteamento da aplicação.

- **Definição das Rotas:** Cada rota é definida usando o componente `<Route>`, onde `path` especifica o caminho do URL e `element` renderiza o componente correspondente quando o caminho é correspondido.

```

import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { Login } from '../pages/Login';
import { Register } from '../pages/Register';
import Principal from '../pages/Principal';
import Eventos from '../pages/Eventos';
import Historico from '../pages/Historico';
import Visualizacao from '../pages/Visualizacao';
import Admin from '../pages/Admin';
import EventosInscritosPage from '../pages/Inscritos';

export const AppRouter = () => {
  return (
    <Router>
      <Routes>
        <Route path="/login" element={<Login/>} />
        <Route path="/register" element={<Register/>} />
        <Route path="/" element={<Principal/>} />
        <Route path="/eventos" element={<Eventos/>} />
        <Route path="/historico" element={<Historico/>} />
        <Route path="/visualizacao/:id" element={<Visualizacao/>} />
        <Route path="/admin" element={<Admin/>} />
        <Route path="/inscritos" element={<EventosInscritosPage/>} />
      </Routes>
    </Router>
  );
}

```

Listagem 4.14: Código da configuração das rotas no react

### 4.5.2 Utilização de Hooks

#### useState

O hook `useState` é essencial para gerir estados locais dentro de componentes funcionais. Ele é utilizado para declarar variáveis de estado e funções para atualizá-las, proporcionando reatividade aos componentes. Na listagem 4.15, é apresentado um exemplo do uso da hook `useState` no projeto.

```

import { useState } from "react";

const [email, setEmail] = useState("");
const [password, setPassword] = useState("");

```

```
const [name, setName] = useState("");
```

Listagem 4.15: Exemplo de código para a utilização da hook useState

### useEffect

O hook useEffect é usado para realizar efeitos colaterais em componentes funcionais, como a procura de dados ou manipulação de DOM. Ele é executado após a renderização do componente e em atualizações subsequentes. Na listagem 4.16, é apresentado um exemplo do uso da hook useEffect no projeto.

```
import React, { useState, useEffect } from "react";
useEffect(() => {
    fetchEventos();
}, []);
```

Listagem 4.16: Exemplo de código para a utilização da hook useEffect

### useParams

O hook useParams é utilizado para aceder os parâmetros da URL definidos dinamicamente em rotas dentro de um componente React. Ele é particularmente útil em situações onde é preciso extrair valores específicos da URL para usar no componente. Na listagem 4.17, é apresentado um exemplo do uso da hook useParams no projeto.

```
import { Link, useParams } from 'react-router-dom';
const { id } = useParams();
```

Listagem 4.17: Exemplo de código para a utilização da hook useParams

### useNavigate

O hook useNavigate é usado para navegar entre rotas de forma programática dentro de um componente funcional. Ele fornece uma função navigate que pode ser chamada para redirecionar o utilizador para uma rota específica. Na listagem 4.18, é apresentado um exemplo do uso da hook useNavigate no projeto.

```
import { Link, useNavigate } from "react-router-dom";
const navigate = useNavigate();
```

Listagem 4.18: Exemplo de código para a utilização da hook useNavigate

## 4.5.3 Integração com o Back-end

A biblioteca que foi utilizada para realizar a conexão do Back-end com o Front-end foi o Axios. Utilizar Axios em conjunto com React simplifica a comunicação entre a aplicação

Front-end e um Back-end RESTful. Com ele, é possível configurar requisições HTTP para procurar, enviar, atualizar e excluir dados do servidor, proporcionando uma experiência fluida e responsiva para os utilizadores. Na listagem 4.19, é apresentado um exemplo do uso da biblioteca axios no projeto.

```
const handleRegister = async (event) => {
  event.preventDefault();

  try {
    const response = await axios.post("http://localhost:8000/api/register", {
      name: name,
      email: email,
      password: password
    });
    console.log(response.data);
    navigate("/login");
  } catch (error) {
    console.error("Erro ao registrar:", error);
  }
};
```

Listagem 4.19: Exemplo de código para a utilização da biblioteca Axios

#### 4.5.4 CSS

O CSS é a linguagem fundamental para estilização de componentes web em aplicações React. Ele permite definir cores, fontes, layouts e animações, garantindo uma experiência visual coesa e agradável para os utilizadores. Ao integrar o CSS com React, podemos criar interfaces dinâmicas e responsivas que se adaptam a diferentes dispositivos e tamanhos de tela.

##### Flexbox

Flexbox é um sistema de layout em CSS projetado para facilitar a organização de elementos em um contêiner, proporcionando maior controle sobre a distribuição e alinhamento dos itens. As principais vantagens do Flexbox incluem:

- **Layout Flexível e Responsivo:** Flexbox permite criar layouts fluidos que se ajustam automaticamente ao tamanho do conteúdo e às dimensões da tela. Isso simplifica a criação de interfaces responsivas, essenciais para uma experiência consistente em dispositivos variados, desde desktops até smartphones.
- **Alinhamento e Distribuição Simplificados:** Com Flexbox, é fácil alinhar itens verticalmente ou horizontalmente dentro de um contêiner, usando propriedades como justify-content, align-items e align-self.

- **Ordenação Flexível de Itens:** Flexbox permite reorganizar visualmente elementos dentro de um contêiner sem modificar a estrutura HTML, usando propriedades como `order`. Isso proporciona flexibilidade no design de interfaces e facilita ajustes na disposição dos elementos conforme necessário.
- **Suporte Amplo nos Navegadores Modernos:** Flexbox é amplamente suportado pelos navegadores modernos, o que o torna uma escolha robusta para desenvolvimento web. Ele oferece uma solução eficaz e confiável para problemas comuns de layout que os programadores enfrentam.

Na listagem 4.20, é apresentado um exemplo do uso de Flexbox no projeto.

```
.eventos-container {  
    display: flex;  
    flex-wrap: wrap;  
    justify-content: center;  
}
```

Listagem 4.20: Exemplo do css utilizando flexbox

#### 4.5.5 Testes

Neste projeto foram realizados testes às rotas da API, com recurso à ferramenta Postman, visando assegurar não só a lógica de negócio, mas também o seu correto funcionamento.

##### Registo

A Figura 4.1 mostra o teste realizado para o registo em que, o teste deu 201 o que significa que a solicitação foi atendida com sucesso e resultou na criação de um novo utilizador.

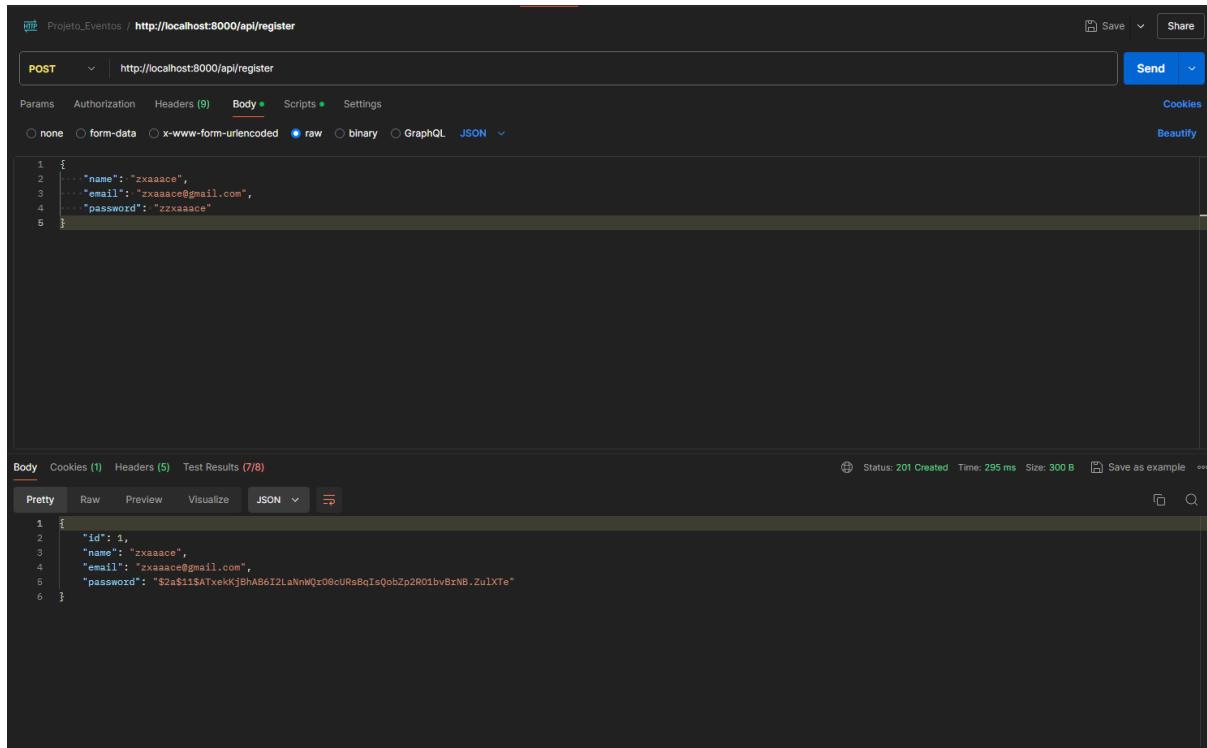


Figura 4.1: Teste no postman para o registo

## Login

A Figura 4.2 mostra o teste realizado para o login em que, o teste deu 200 o que significa que a solicitação foi atendida com sucesso e resultou no login de um utilizador.

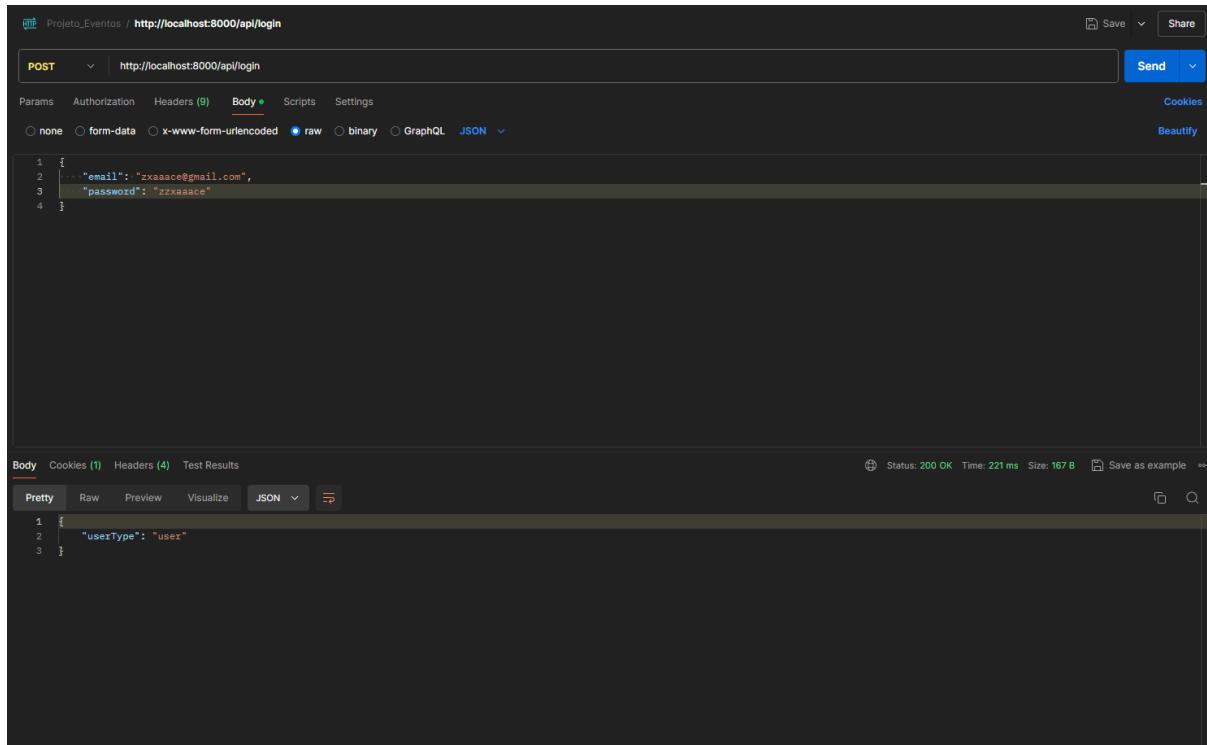


Figura 4.2: Teste no postman para o login

## Logout

A Figura 4.3 mostra o teste realizado para o logout em que, o teste deu 200 o que significa que a solicitação foi atendida com sucesso e resultou no logout do utilizador.

The screenshot shows the Postman interface with the following details:

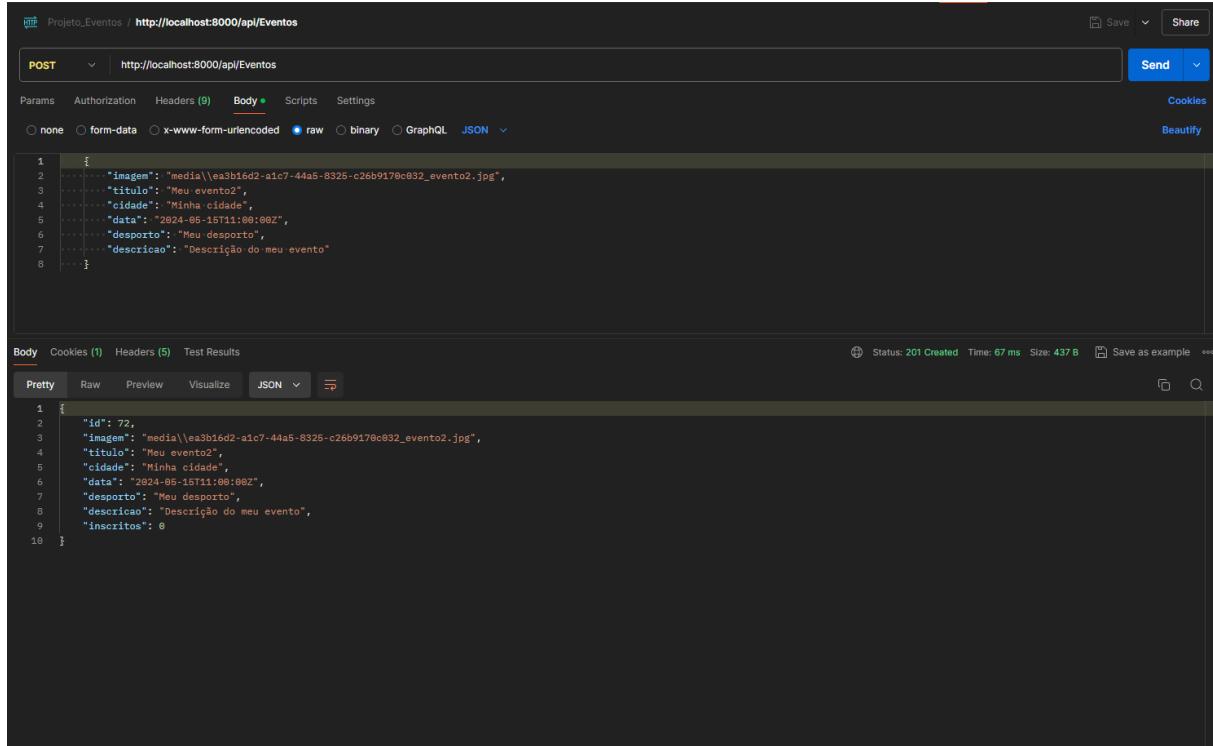
- Header:** POST, URL: http://localhost:8000/api/logout
- Params:** Key: Value, Description: Description
- Body:** JSON response:

```
1 {  
2     "message": "Logout bem-sucedido."  
3 }
```
- Status:** 200 OK, Time: 17 ms, Size: 182 B

Figura 4.3: Teste no postman para o logout

## Criação de eventos

A Figura 4.4 mostra o teste realizado para a criação de um evento em que, o teste deu 201 o que significa que a solicitação foi atendida com sucesso e resultou na criação de um novo evento.



The screenshot shows a Postman interface with the following details:

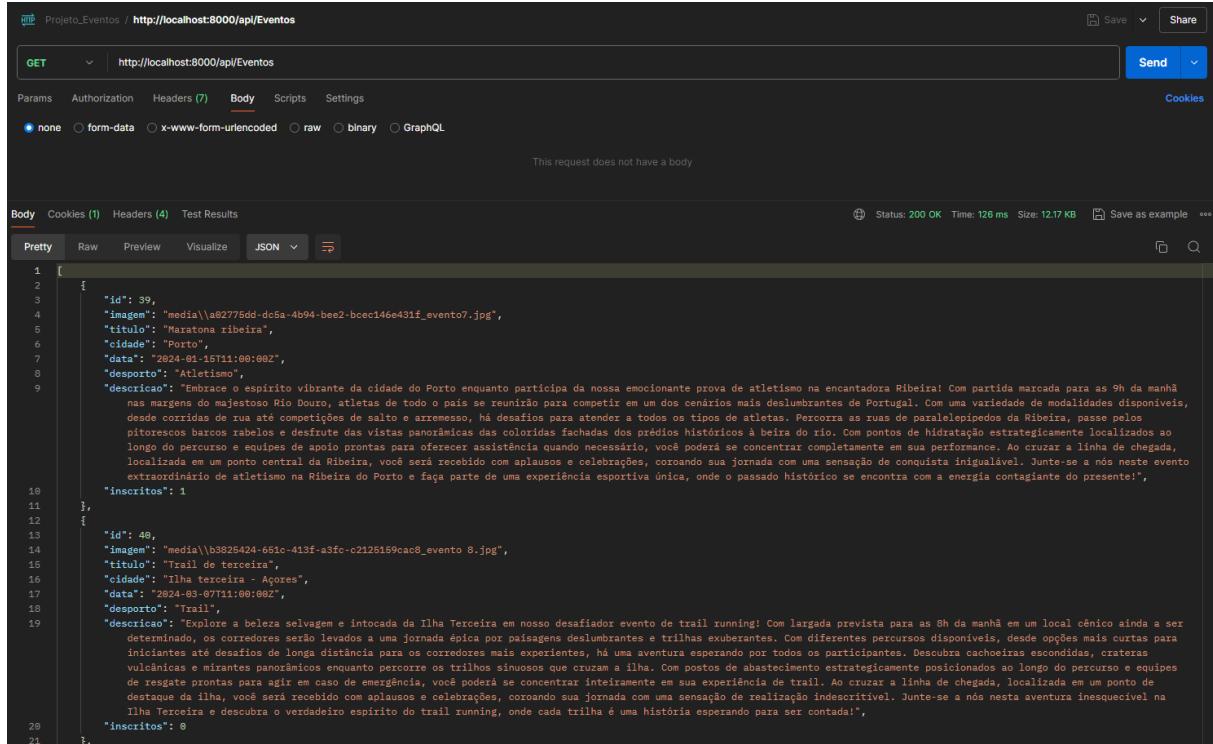
- Request URL:** http://localhost:8000/api/Eventos
- Method:** POST
- Body:** JSON (selected)
- JSON Data:**

```
1  {
2     "id": 72,
3     "imagem": "media\\ea3b16d2-a1c7-44a5-8325-c26b9170c032_evento2.jpg",
4     "titulo": "Meu evento2",
5     "cidade": "Minha cidade",
6     "data": "2024-05-15T11:00:00Z",
7     "desporto": "Meu desporto",
8     "descricao": "Descrição do meu evento",
9
10 }
```
- Response Status:** 201 Created
- Response Time:** 67 ms
- Response Size:** 437 B

Figura 4.4: Teste no postman para criar evento

## Listagem de eventos

A Figura 4.5 mostra o teste realizado para a listagem de eventos em que, o teste deu 200 o que significa que a solicitação foi atendida com sucesso e resultou na listagem de todos os eventos.



```

HTTP Projekt_Eventos http://localhost:8000/api/Eventos
GET http://localhost:8000/api/Eventos
Params Authorization Headers (7) Body Scripts Settings
None form-data x-www-form-urlencoded raw binary GraphQL
This request does not have a body
Body Cookies (1) Headers (4) Test Results
Pretty Raw Preview Visualize JSON
[{"id": 39, "imagem": "media\\a02775dd-dc5a-4b94-bee2-bce146e431f_evento7.jpg", "titulo": "Maratona ribeira", "cidade": "Porto", "data": "2024-01-15T11:00:00Z", "desporto": "Atletismo", "descricao": "Embrace o espírito vibrante da cidade do Porto enquanto participa da nossa emocionante prova de atletismo na encantadora Ribeira! Com partida marcada para as 9h da manhã nas margens do majestoso Rio Douro, atletas de todo o país se reunirão para competir em um dos cenários mais deslumbrantes de Portugal. Com uma variedade de modalidades disponíveis, desde corridas de rua até competições de salto e arremesso, há desafios para atender a todos os tipos de atletas. Percorra as ruas de paralelepípedos da Ribeira, passe pelos pitorescos barcos rabelos e desfrute das vistas panorâmicas das coloridas fachadas dos prédios históricos à beira do rio. Com pontos de hidratação estratégicamente localizados ao longo do percurso e equipes de apoio prontas para oferecer assistência quando necessário, você poderá se concentrar completamente em sua performance. Ao cruzar a linha de chegada, localizada em um ponto central da Ribeira, você será recebido com aplausos e celebrações, coroando sua jornada com uma sensação de conquista inigualável. Junte-se a nós neste evento extraordinário de atletismo na Ribeira do Porto e faça parte de uma experiência esportiva única, onde o passado histórico se encontra com a energia contagiante do presente!", "inscritos": 1}, {"id": 40, "imagem": "media\\b3825424-651c-413f-a3fc-c2125159cac8_evento 8.jpg", "titulo": "Trail de terceira", "cidade": "Ilha terceira - Açores", "data": "2024-03-07T11:00:00Z", "desporto": "Trail", "descricao": "Explore a beleza selvagem e intocada da Ilha Terceira em nosso desafiador evento de trail running! Com largada prevista para as 8h da manhã em um local cênico ainda a ser determinado, os corredores serão levados a uma jornada épica por paisagens deslumbrantes e trilhas exuberantes. Com diferentes percursos disponíveis, desde opções mais curtas para iniciantes até desafios de longa distância para os corredores mais experientes, há uma aventura esperando por todos os participantes. Descubra cachoeiras escondidas, crateras vulcânicas e mirantes panorâmicos enquanto percorre os trilhos sinuosos que cruzam a ilha. Com postos de abastecimento estratégicamente posicionados ao longo do percurso e equipes de resgate prontas para agir em caso de emergência, você poderá se concentrar inteiramente em sua experiência de trail. Ao cruzar a linha de chegada, localizada em um ponto de destaque da ilha, você será recebido com aplausos e celebrações, coroando sua jornada com uma sensação de realização indescritível. Junte-se a nós nesta aventura inesquecível na Ilha Terceira e descubra o verdadeiro espírito do trail running, onde cada trilha é uma história esperando para ser contada!", "inscritos": 0}]
  
```

Figura 4.5: Teste no postman para a listagem de eventos

## Visualização detalhada de um evento

A Figura 4.6 mostra o teste realizado para a visualização detalhada de um evento específico em que, o teste deu 200 o que significa que a solicitação foi atendida com sucesso e resultou na mostragem dos detalhes do evento pedido.

The screenshot shows a Postman interface with the following details:

- Request URL:** http://localhost:8000/api/Eventos/39
- Method:** GET
- Body:** JSON response (Pretty)
- Response Status:** 200 OK
- Response Headers:** Status: 200 OK, Time: 13 ms, Size: 1.59 KB
- Response Body (Pretty JSON):**

```

1  {
2     "id": 39,
3     "imagem": "media\\a82775dd-dc5a-4b94-bee2-bcec146e431f_evento7.jpg",
4     "titulo": "Maratona ribeirã",
5     "cidade": "Porto",
6     "data": "2024-01-15T11:00:00Z",
7     "desporto": "Atletismo",
8     "descricao": "Embrace o espírito vibrante da cidade do Porto enquanto participa da nossa emocionante prova de atletismo na encantadora Ribeira! Com partida marcada para as 9h da manhã nas margens do majestoso Rio Douro, atletas de todo o país se reunirão para competir em um dos cenários mais deslumbrantes de Portugal. Com uma variedade de modalidades disponíveis, desde corridas de rua até competições de salto e arremesso, há desafios para atender a todos os tipos de atletas. Percorra as ruas de paralelepípedos da Ribeira, passe pelos pitorescos bares rabelos e desfrute das vistas panorâmicas das coloridas fachadas dos prédios históricos à beira do rio. Com pontos de hidratação estratégicamente localizados ao longo do percurso e equipes de apoio prontas para oferecer assistência quando necessário, você poderá se concentrar completamente em sua performance. Ao cruzar a linha de chegada, localizada em um ponto central da Ribeira, você será recebido com aplausos e celebrações, coroando sua jornada com uma sensação de conquista inigualável. Junte-se a nós neste evento extraordinário de atletismo na Ribeira do Porto e faça parte de uma experiência esportiva única, onde o passado histórico se encontra com a energia contagiosa do presente!",
9   "inscritos": 1
10 }
```

Figura 4.6: Teste no postman para a visualização detalhada de um evento

## Exclusão de um evento

A Figura 4.7 mostra o teste realizado para a exclusão de um evento em que, o teste deu 204 o que significa que a solicitação foi atendida com sucesso e resultou na exclusão do evento.

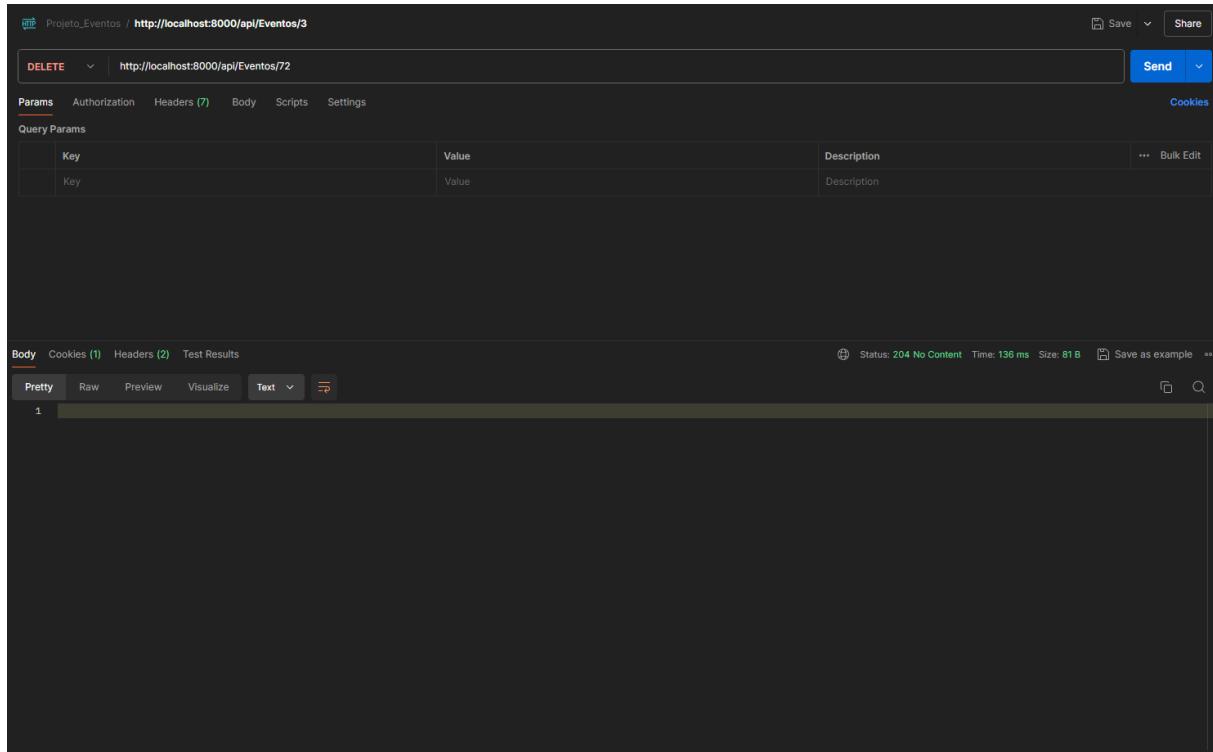
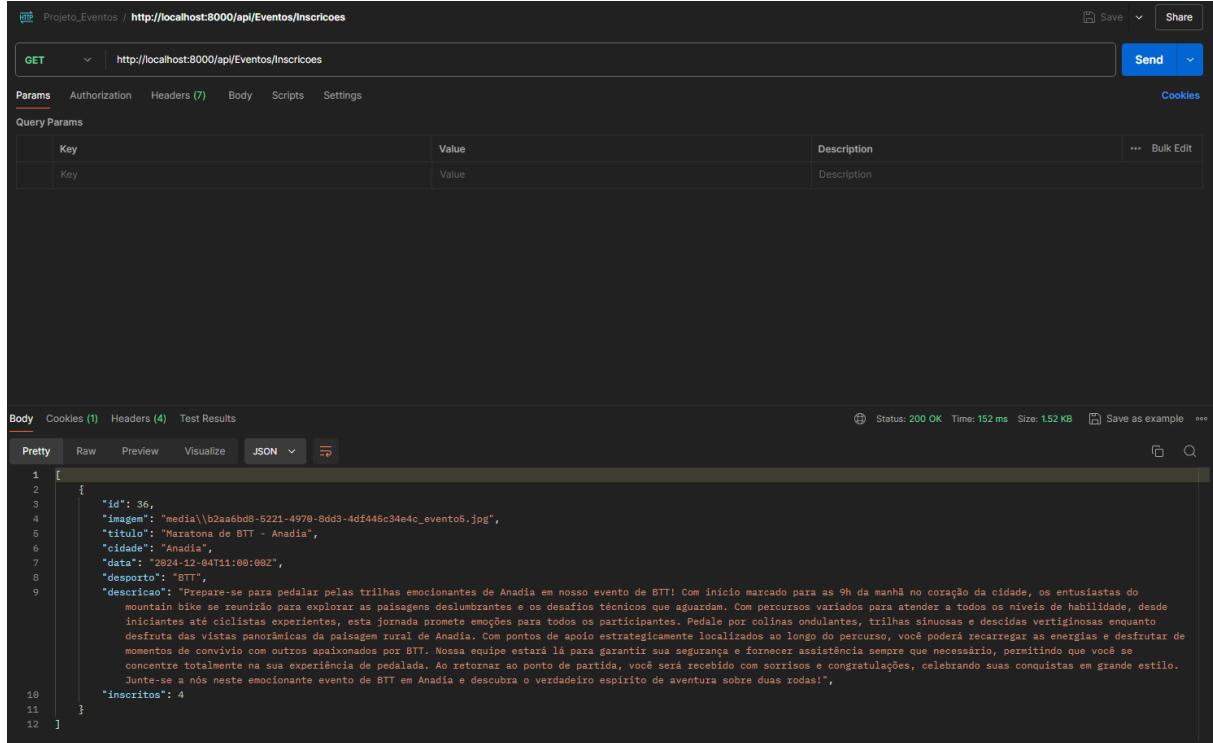


Figura 4.7: Teste no postman para a exclusão de um evento

## Visualização de eventos inscritos

A Figura 4.8 mostra o teste realizado para a visualização de eventos inscritos em que, o teste deu 200 o que significa que a solicitação foi atendida com sucesso e resultou na visualização dos eventos em que um utilizador estava inscrito.



```

1  [
2    {
3      "id": 36,
4      "imagen": "media\\b2aa6bd8-b221-4970-8dd3-4df445c34e4c_evento5.jpg",
5      "titulo": "Maratona de BTT - Anadia",
6      "cidade": "Anadia",
7      "data": "28/12/2024-08:00:00Z",
8      "desporto": "BTT",
9      "descricao": "Prepare-se para pedalhar pelas trilhas emocionantes de Anadia em nosso evento de BTT! Com início marcado para as 9h da manhã no coração da cidade, os entusiastas do mountain bike se reunirão para explorar as paisagens deslumbrantes e os desafios técnicos que aguardam. Com percursos variados para atender a todos os níveis de habilidade, desde iniciantes até ciclistas experientes, esta jornada promete emoções para todos os participantes. Pedale por colinas ondulantes, trilhas sinuosas e descidas vertiginosas enquanto desfruta das vistas panorâmicas da paisagem rural de Anadia. Com pontos de apoio estratégicamente localizados ao longo do percurso, você poderá recarregar as energias e desfrutar de momentos de convívio com outros apaixonados por BTT. Nossa equipe estará lá para garantir sua segurança e fornecer assistência sempre que necessário, permitindo que você se concentre totalmente na sua experiência de pedalada. Ao retornar ao ponto de partida, você será recebido com sorrisos e congratulações, celebrando suas conquistas em grande estilo. Junte-se a nós neste emocionante evento de BTT em Anadia e descubra o verdadeiro espírito de aventura sobre duas rodas!",
10     "inscritos": 4
11   }
12 ]

```

Figura 4.8: Teste no postman para a visualização de eventos inscritos

## Inscrição em eventos

A Figura 4.9 mostra o teste realizado para a inscrição em eventos em que, o teste deu 204 o que significa que a solicitação foi atendida com sucesso e resultou na inscrição de um utilizador num determinado evento.

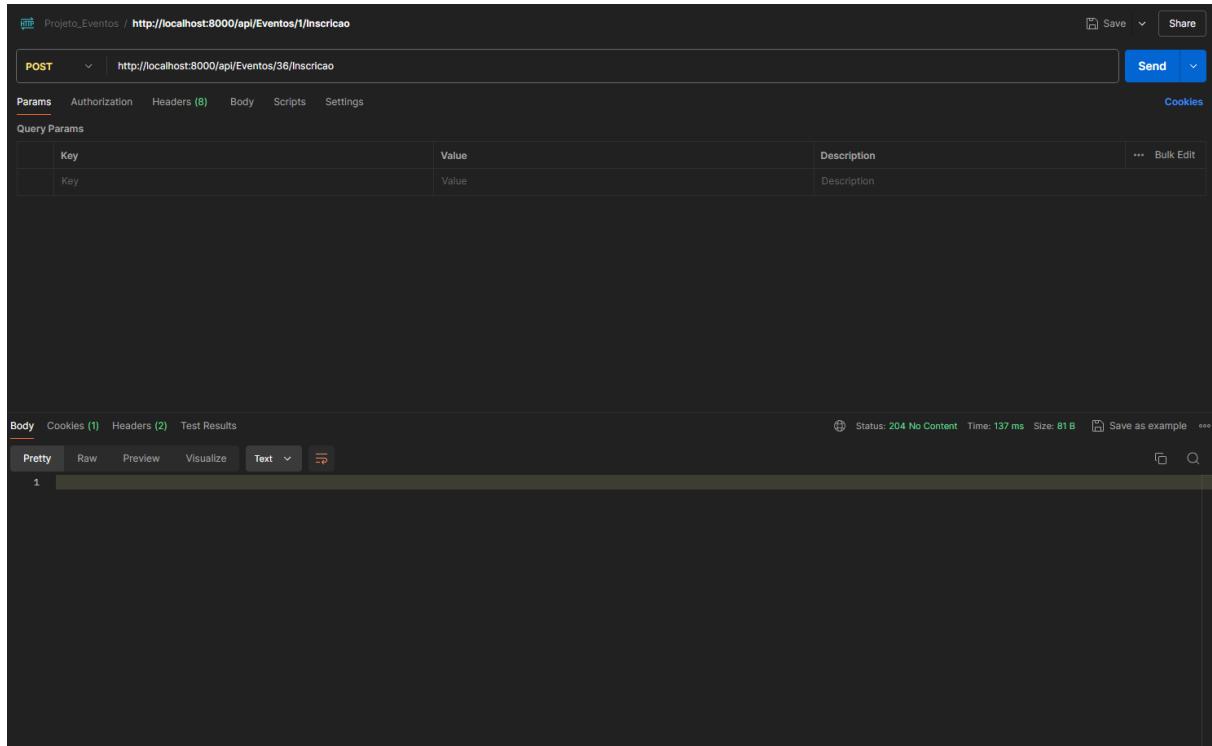


Figura 4.9: Teste no postman para a inscrição em eventos

## Visualização do numero de inscritos num evento

A Figura 4.10 mostra o teste realizado para a visualização do numero de inscritos num evento em que, o teste deu 200 o que significa que a solicitação foi atendida com sucesso e resultou na visualização do numero de inscritos num determinado evento.

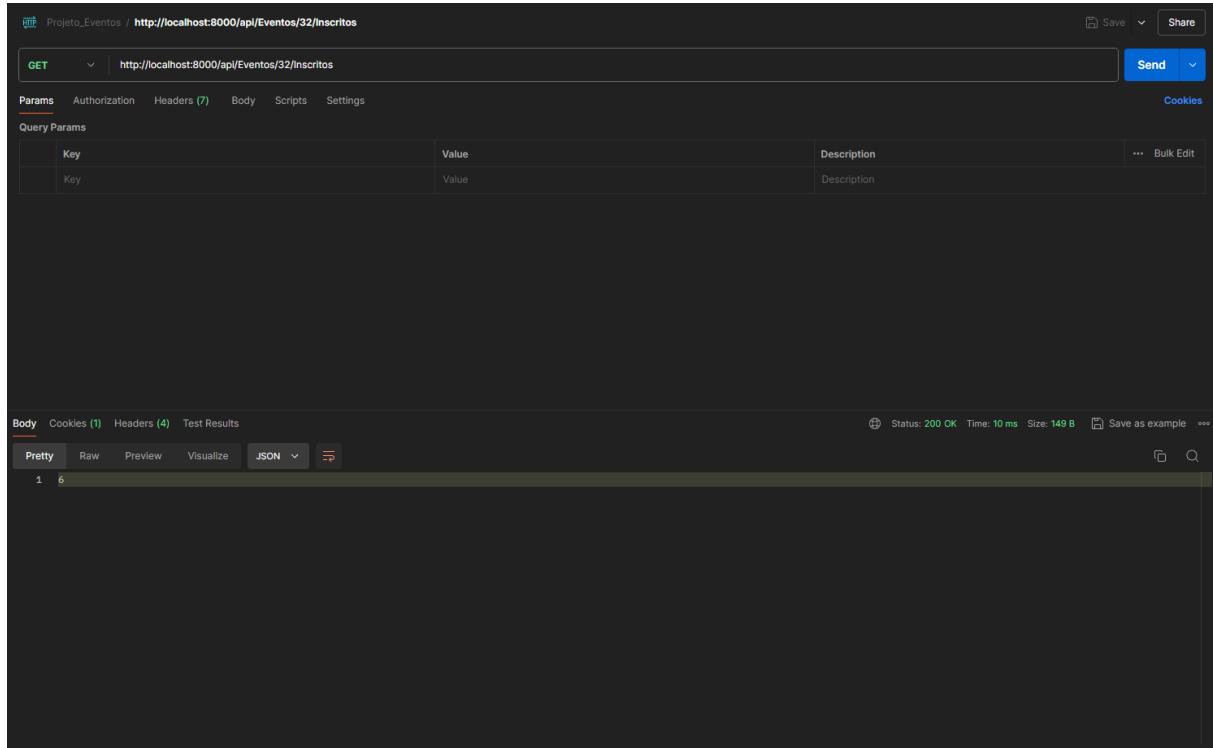
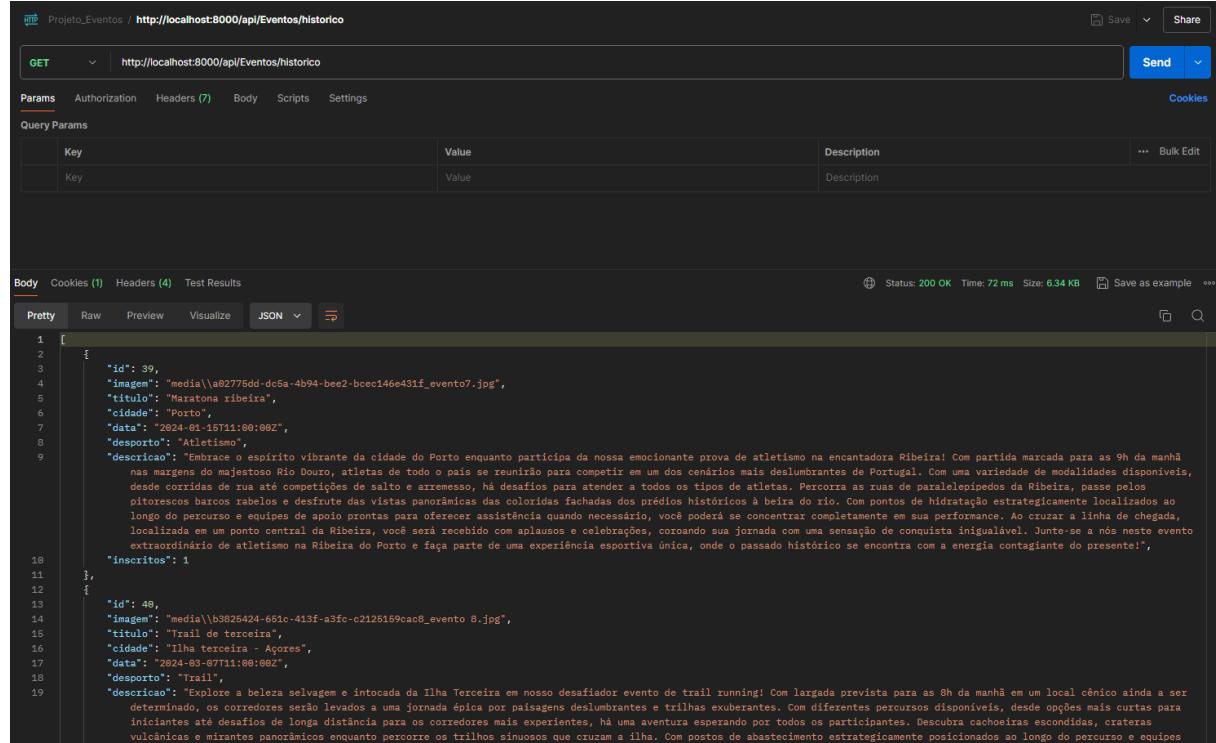


Figura 4.10: Teste no postman para a visualização do numero de inscritos num evento

## Visualização do histórico de eventos

A Figura 4.11 mostra o teste realizado para a visualização do histórico de eventos em que, o teste deu 200 o que significa que a solicitação foi atendida com sucesso e resultou na visualização do histórico de eventos que já ocorreram.



```

1 [
2   {
3     "id": 39,
4     "imagem": "media\\a82775dd-dc5a-4b94-bee2-bce146e431f_evento7.jpg",
5     "titulo": "Maratona ribeira",
6     "cidade": "Porto",
7     "data": "2024-04-15T11:00:00Z",
8     "desporto": "Atletismo",
9     "descricao": "Embrace o espírito vibrante da cidade do Porto enquanto participa da nossa emocionante prova de atletismo na encantadora Ribeira! Com partida marcada para as 9h da manhã nas margens do majestoso Rio Douro, atletas de todo o país se reunirão para competir em um dos cenários mais deslumbrantes de Portugal. Com uma variedade de modalidades disponíveis, desde corridas de rua até competições de salto e arremesso, há desafios para atender a todos os tipos de atletas. Percorra as ruas de paralelepípedos da Ribeira, passe pelos pitorescos bairros rabelos e desfrute das vistas panorâmicas das coloridas fachadas dos prédios históricos à beira do rio. Com pontos de hidratação estratégicamente localizados ao longo do percurso e equipes de apoio prontas para oferecer assistência quando necessário, você poderá se concentrar completamente em sua performance. Ao cruzar a linha de chegada, localizada em um ponto central da Ribeira, você será recebido com aplausos e celebrações, coroando sua jornada com uma sensação de conquista inigualável. Junte-se a nós neste evento extraordinário de atletismo na Ribeira do Porto e faça parte de uma experiência esportiva única, onde o passado histórico se encontra com a energia contagiante do presente!",
10    "inscritos": 1
11  },
12  [
13    {
14      "id": 48,
15      "imagem": "media\\b3825424-651c-413f-a3fc-c2125159cac8_evento 8.jpg",
16      "titulo": "Trail de terceira",
17      "cidade": "Ilha terceira - Açores",
18      "data": "2024-03-07T11:00:00Z",
19      "desporto": "Trail",
20      "descricao": "Explore a beleza selvagem e intocada da Ilha Terceira em nosso desafiador evento de trail running! Com largada prevista para as 8h da manhã em um local cênico ainda a ser determinado, os corredores serão levados a uma jornada épica por paisagens deslumbrantes e trilhas exuberantes. Com diferentes percursos disponíveis, desde opções mais curtas para iniciantes até desafios de longa distância para os corredores mais experientes, há uma aventura esperando por todos os participantes. Descubra cachoeiras escondidas, crateras vulcânicas e mirantes panorâmicos enquanto percorre os trilhos sinuosos que cruzam a ilha. Com postos de abastecimento estratégicamente posicionados ao longo do percurso e equipes

```

Figura 4.11: Teste no postman para a visualização do histórico de eventos

## Upload de imagens

A Figura 4.12 mostra o teste realizado para dar upload de imagens em que, o teste deu 200 o que significa que a solicitação foi atendida com sucesso e resultou na realização de um upload de uma determinada imagem.

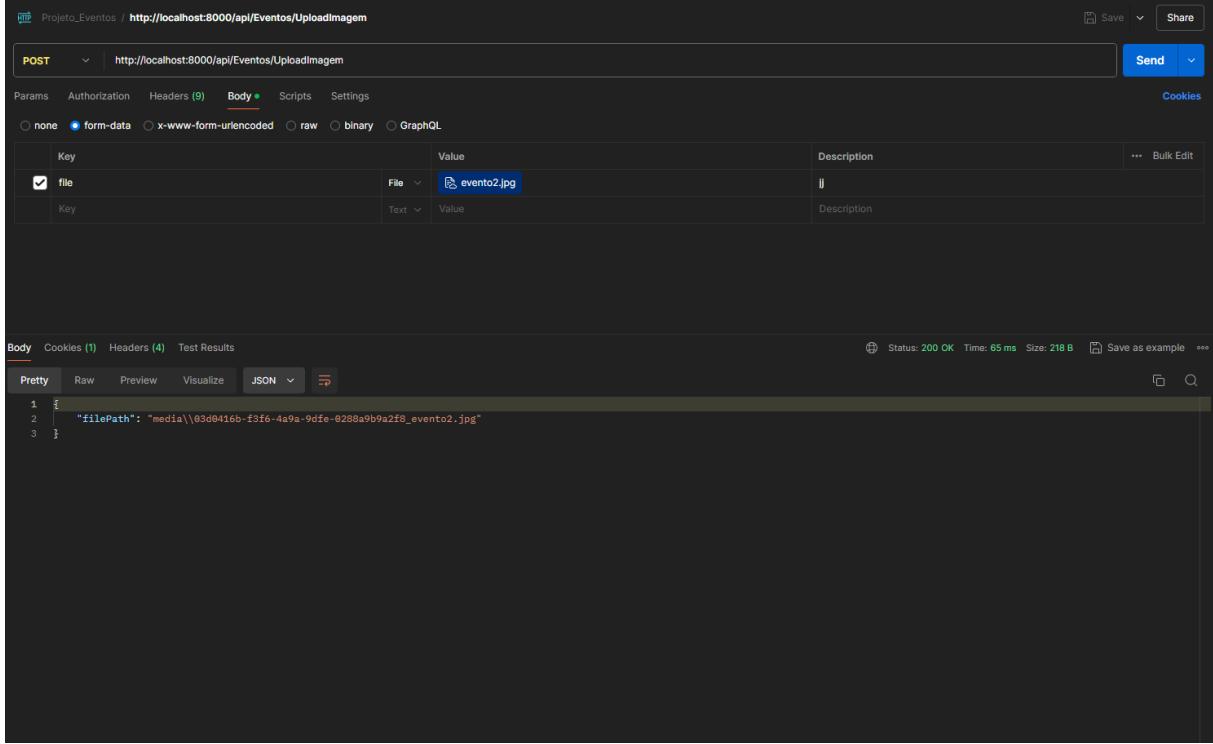


Figura 4.12: Teste no postman para dar upload a imagens

# 5. Análise de resultados

Neste capítulo são apresentados os vários ecrãs no seu estado final e uma análise crítica dos resultados obtidos durante a implementação.

## 5.1 Página de Login

A figura 5.1 representa a página de login do sistema, onde o utilizador insere o seu email e a sua palavra-passe para efetuar o respetivo login. Caso o utilizador não possua conta, será redirecionado para a página de registo para criar uma nova conta.

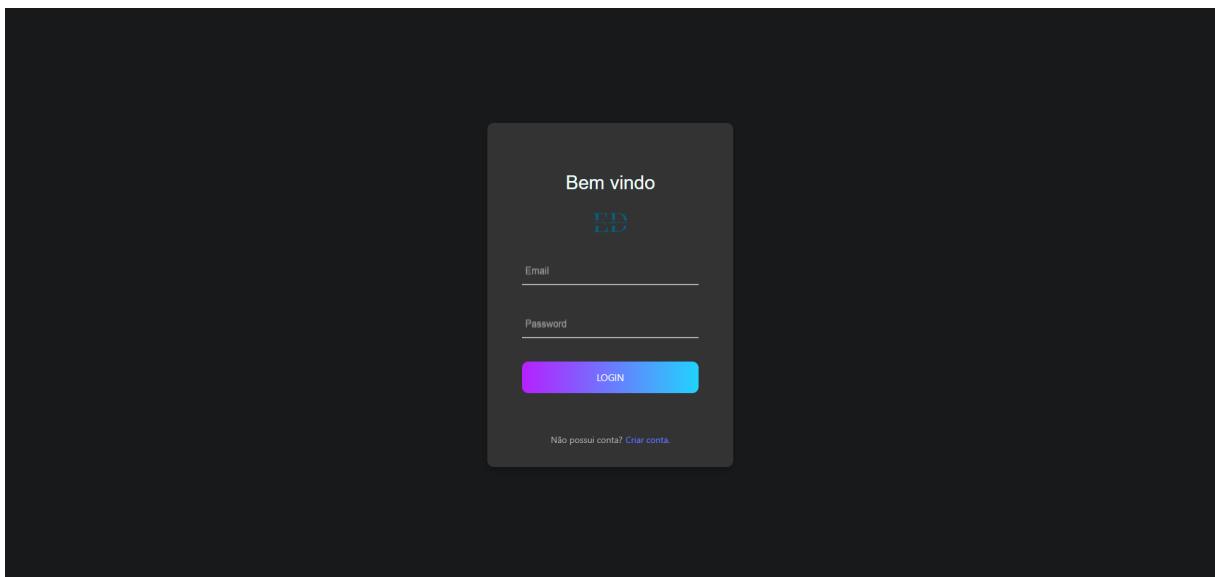


Figura 5.1: Página de login

## 5.2 Página de Registo

A figura 5.2 representa a página de registo do sistema, onde o utilizador insere o seu nome, email e palavra-passe para criar uma nova conta. Caso o utilizador já possua conta, será redirecionado para a página de login para se autenticar no sistema.

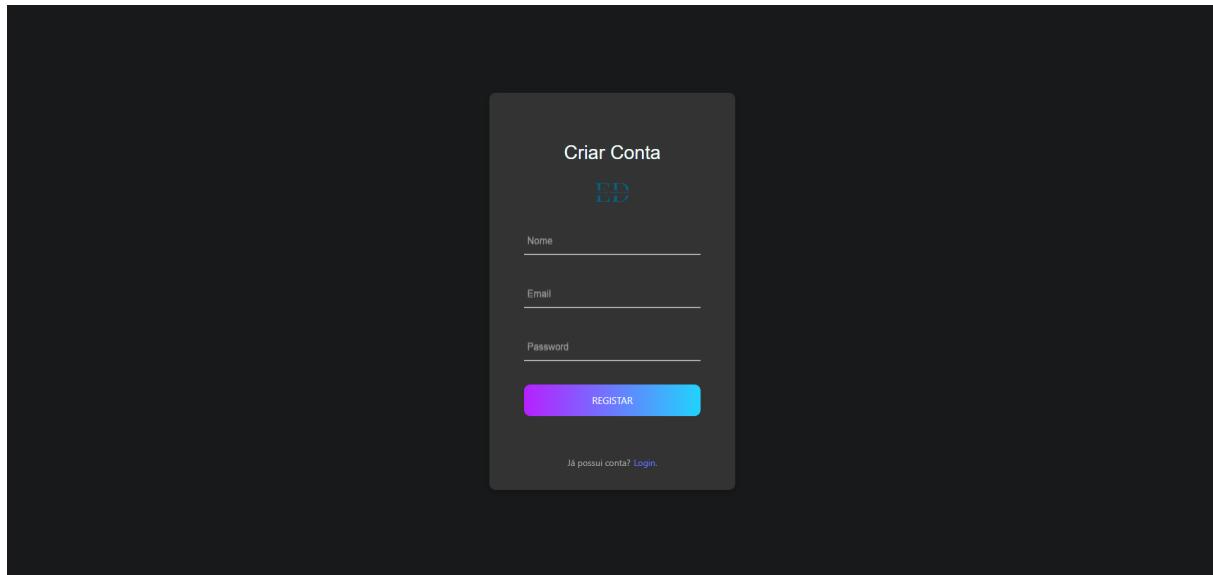


Figura 5.2: Página de registo

## 5.3 Página Inicial

A figura 5.3 representa a página inicial do sistema, onde um conjunto de imagens é apresentado em slideshow. No Header, é possível ver diferentes botões, como "HomePage"(página atual), "Eventos" e "Histórico". Além disso, é possível visualizar os eventos em que o utilizador está inscrito e fazer logout do sistema.

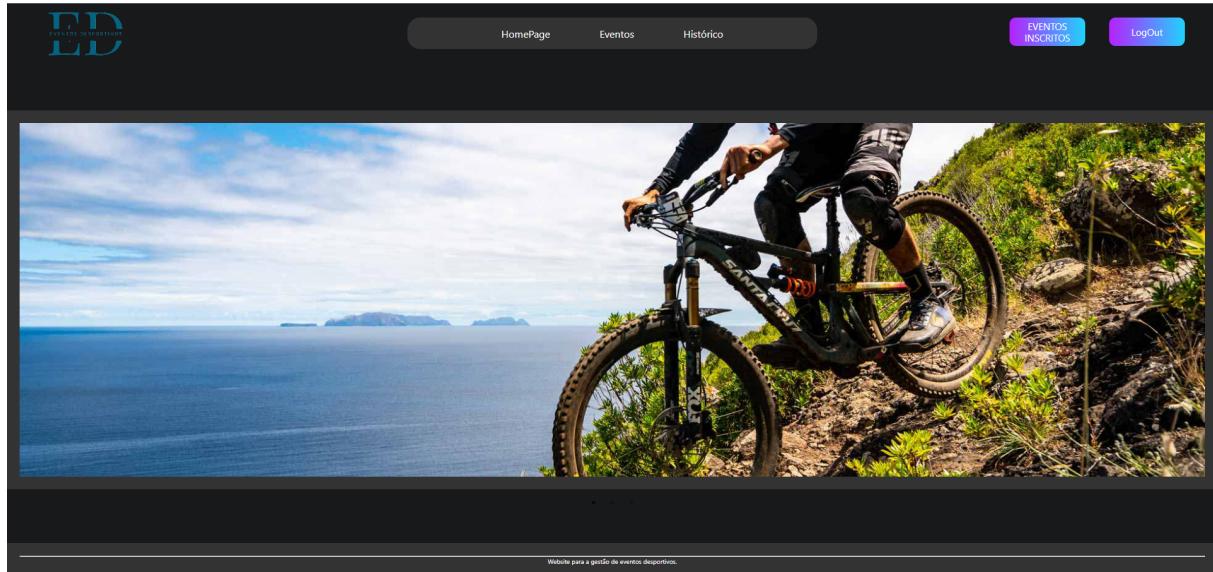


Figura 5.3: Página inicial

## 5.4 Página de Eventos

A figura 5.4 representa a página de eventos do sistema, onde são apresentados todos os eventos que estão inseridos no sistema. No Header, é possível ver diferentes botões, como "HomePage", "Eventos"(página atual) e "Histórico". Além disso, é possível fazer logout do sistema.

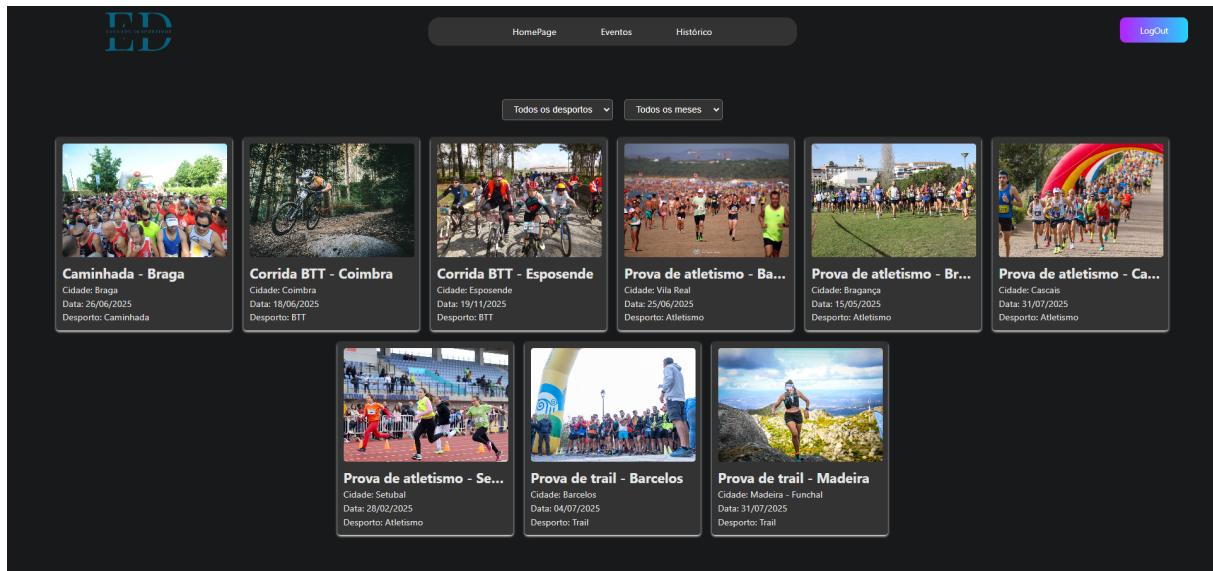


Figura 5.4: Página de eventos

## 5.5 Página de Visualização de um evento

A figura 5.5 representa a página de visualização de um evento do sistema, onde toda a informação relativa ao evento é apresentada, incluindo a possibilidade de realizar a inscrição no evento. No Header, é possível ver diferentes botões, como "HomePage", "Eventos" e "Histórico". Além disso, é possível fazer logout do sistema.

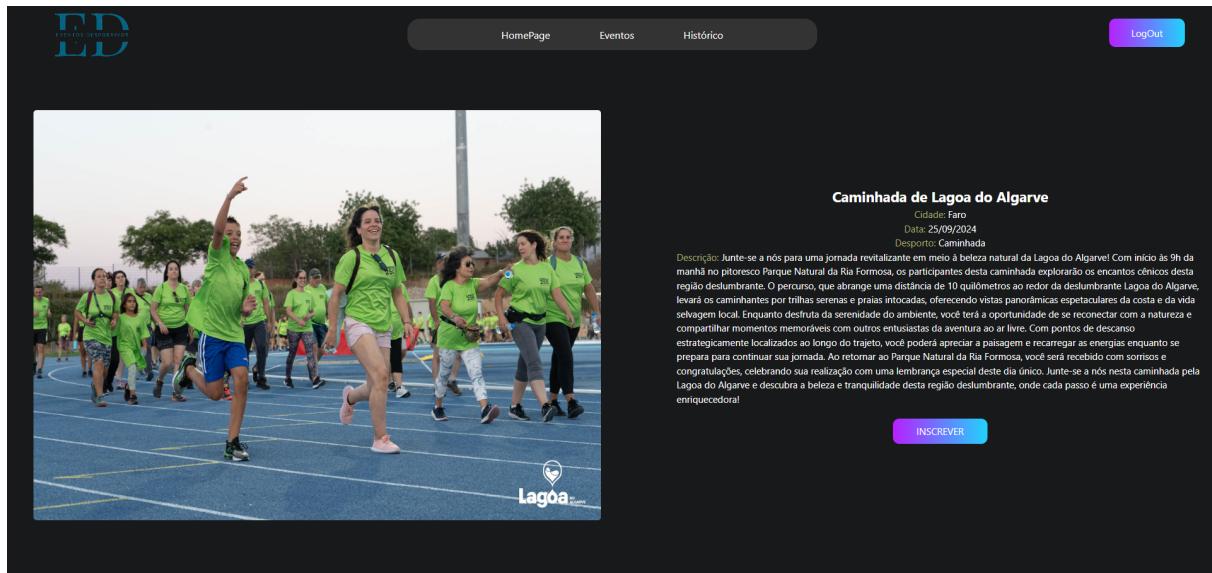


Figura 5.5: Página de visualização de um evento

## 5.6 Página de Eventos inscritos

A figura 5.6 representa a página de visualização de todos os eventos do sistema em que o utilizador está inscrito. No Header, é possível ver diferentes botões, como "HomePage", "Eventos" e "Histórico". Além disso, é possível fazer logout do sistema.

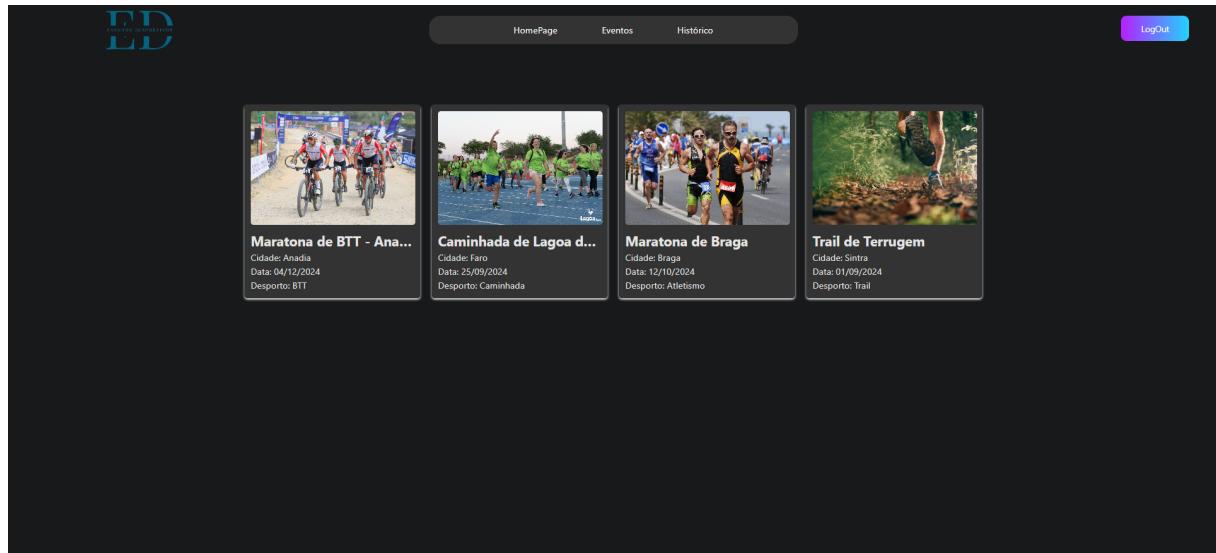


Figura 5.6: Página de eventos inscritos

## 5.7 Página de Histórico

A figura 5.7 representa a página de histórico de todos os eventos do sistema que já ocorreram. No Header, é possível ver diferentes botões, como "HomePage", "Eventos" e "Histórico"(página atual). Além disso, é possível fazer logout do sistema.

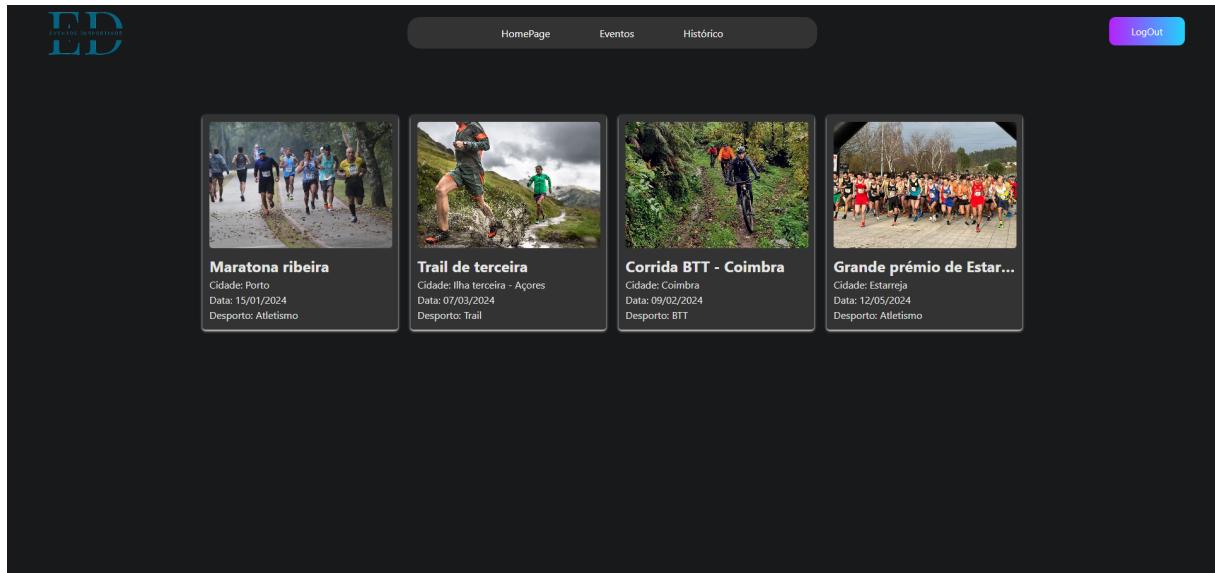


Figura 5.7: Página de histórico

## 5.8 Página do Promotor

A figura 5.8 representa a página do promotor, onde este pode inserir eventos, assim como verificar quantos inscritos tem um determinado evento. No Header, é possível fazer logout do sistema.

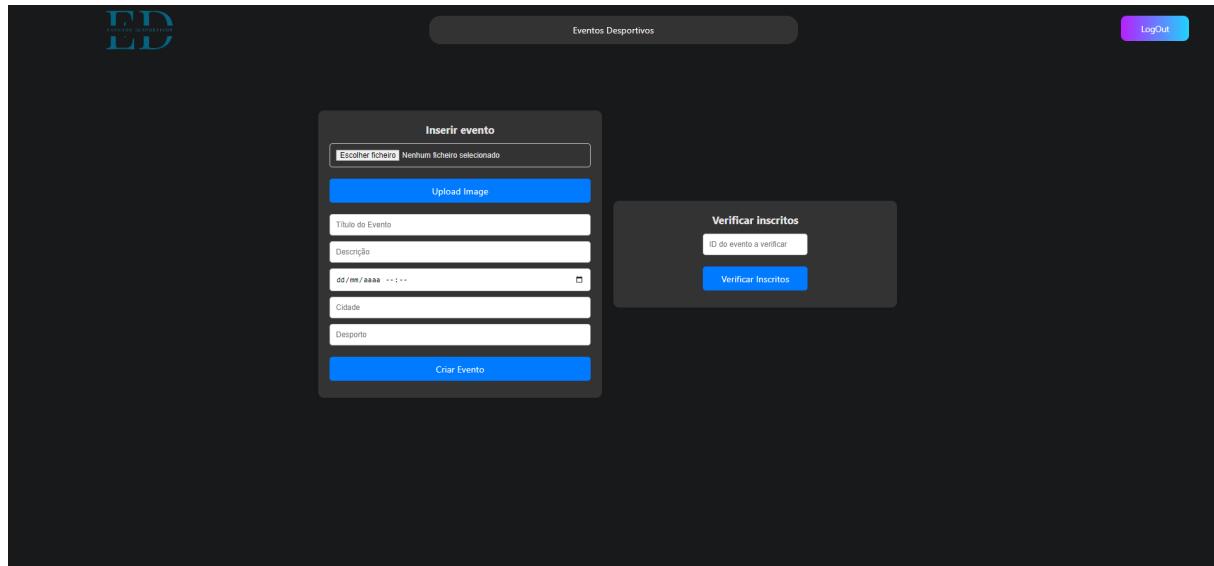


Figura 5.8: Página do promotor

## 5.9 Página de Eventos do Promotor

A figura 5.9 representa a página de eventos do promotor, onde este pode remover qualquer evento do sistema assim como edita-lo. No Header, é possível fazer logout do sistema.

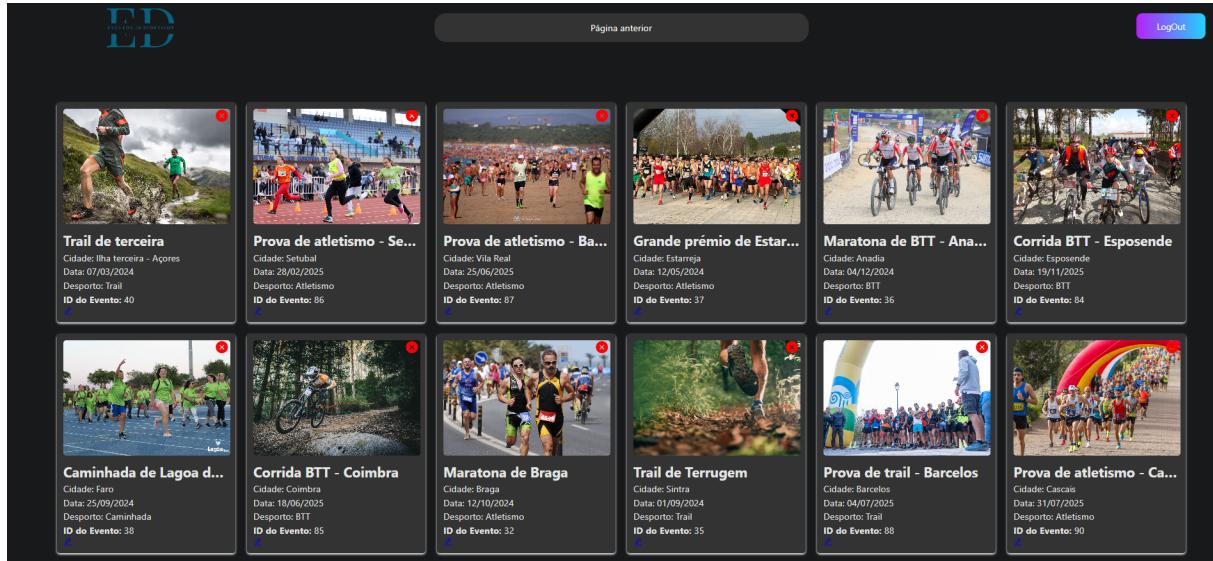


Figura 5.9: Página de eventos do promotor

## 5.10 Avaliação final

Tendo em conta os requisitos identificados, entre funcionais e não funcionais, todos foram desenvolvidos com sucesso, representando uma taxa de conclusão de 100%.

Entre os requisitos implementados, destacam-se as funcionalidades de gestão de eventos, visualização de eventos disponíveis e inscritos, consulta do histórico de eventos, bem como uma interface intuitiva e responsiva que garante uma experiência de utilizador eficiente.

O sistema encontra-se totalmente funcional, com todas as telas desenvolvidas e integradas, estando pronto para ser utilizado no contexto para o qual foi projetado.

Em relação às tecnologias utilizadas (.NET, React e PostgreSQL), a curva de aprendizagem foi desafiadora, mas extremamente enriquecedora. A experiência adquirida permitiu consolidar conhecimentos técnicos e aplicar boas práticas de desenvolvimento ao longo do projeto.

## 6. Conclusão

Este projeto representa não apenas a concretização de um objetivo académico, mas também uma experiência profundamente enriquecedora, que proporcionou uma evolução significativa a nível técnico e pessoal.

A plataforma desenvolvida reflete não só os objetivos iniciais propostos, mas também um compromisso com a entrega de uma solução robusta, intuitiva e funcional. Cada funcionalidade foi cuidadosamente planeada e implementada para garantir que o sistema atende às necessidades específicas identificadas, oferecendo valor real tanto para os promotores de eventos quanto para os utilizadores finais.

Para além da implementação técnica, este projeto permitiu também o desenvolvimento de competências interpessoais, como a capacidade de gerir prazos, resolver problemas de forma criativa e trabalhar de forma autónoma. Estas habilidades serão, sem dúvida, fundamentais para futuros desafios no mundo profissional.

A realização deste projeto também serviu para reforçar a importância do equilíbrio entre teoria e prática. O conhecimento adquirido ao longo do percurso académico foi colocado à prova e complementado com experiências práticas, resultando numa aprendizagem integrada e eficaz.

Em conclusão, este trabalho não só cumpriu os objetivos propostos, mas também superou as expectativas iniciais. O sistema desenvolvido está pronto para ser utilizado, representando uma solução tecnológica sólida para a gestão de eventos desportivos.

# Bibliografia

Active Network. ???? Active network. <https://www.activenetwork.com/>.

GitLab. ???? Gitlab. <https://about.gitlab.com/>.

.Net Core. ???? .net core. <https://dotnet.microsoft.com/en-us/download>.

pgAdmin 4. ???? pgadmin 4. <https://www.pgadmin.org/download/pgadmin-4-windows/>.

PostgresSql. ???? Postgresql. <https://www.postgresql.org/>.

Postman. ???? Postman. <https://www.postman.com/>.

React. ???? React. <https://react.dev/>.

Rider. ???? Rider. <https://www.jetbrains.com/rider/>.

sportsengine. ???? sportsengine. <https://www.sportsengine.com/>.

TeamSnap. ???? Teamsnap. <https://www.teamsnap.com/>.

Visual Paradigm. ???? Visual paradigm. <https://www.visual-paradigm.com/>.

Visual Studio Code. ???? Visual studio code. <https://code.visualstudio.com/>.