Esquema do projeto

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Requisitos técnicos

Front End

* Bootstrap (versão 5.2.3);
* Html 5;
* Css;
* OpenStreetMap (Leaflet).

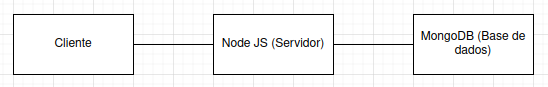
Back End

* Cors (versão 2.8.5);
* bodyParser (versão 1.20.1);
* Jsonwebtoken (versão 8.5.1);
* Mongodb (versão 6.3.0);
* Nodemailer (versão 6.9.7);
* Bcrypt (versão: 5.1.1);
* Dotenv (versão 16.0.3);
* Express (versão 4.18.2);
* Fs;
* nodeJS (versão 20.10.0).

Arquitetura

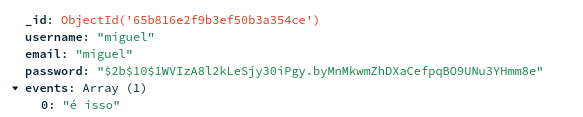
O nosso *website* assenta sobre uma arquitetura 3 camadas, onde temos na camada de apresentação, os clientes, Node Js na camada de servidor e Mongo DB para a camada de dados.

Todos os pedidos entre o cliente e a base de dados, são todos geridos pela camada servidor.



Base de dados

O esquema da nossa base de dados é composta por duas coleções:

* Users
* events



Servidor – Server.js

Configuração Inicial:

Servidor Express com leitura de variáveis de ambiente (.env).

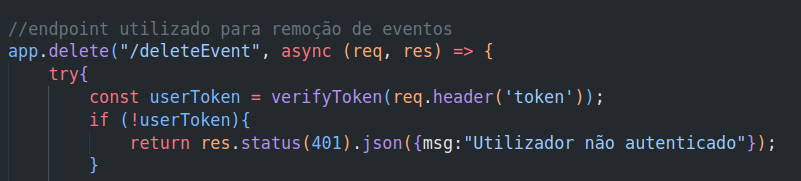
Utilização de CORS para permitir partilha de recursos entre diferentes origens e análise de corpos de pedidos HTTP em formato JSON e com codificação URL.

Autenticação e Registo:

A autenticação está a ser realizada através de tokens JWT onde cada token contém no *payload* o objeto do utilizador, sendo que este objeto é composto pelo *username*  e  *password, d*esta forma, nos seguintes pedidos é nos possível obter estes dados de forma segura.

A nível da criação do token estamos a utitilizar a tag {expiresIn: "900s"} que equivale a 15 minutos para a sessão expirar.

A nível dos pedidos seguintes que o utilizador faça; para os endpoints que seja necessário verificar se o mesmo está ou não autenticado, fazemos sempre a verificação do token da seguinte forma:



**Endpoints**

/signUp: Ponto de acesso para registo de um novo utilizador. Verifica se o nome de utilizador já existe; se não, encripta a palavra-passe e insere o novo utilizador na base de dados, enviando um e-mail de confirmação.

/login: Ponto de acesso para autenticação de um utilizador. Confirma as credenciais do utilizador e, se estiverem corretas, gera e envia um token JWT para sessões futuras.

Gestão de Eventos:

/addEventToUser: Ponto de acesso para adicionar um evento à lista de um utilizador. Verifica a autenticação do utilizador através de um token, atualiza a lista de eventos na base de dados e devolve a confirmação da operação.

/registeredEvents: Ponto de acesso para listar todos os eventos registados na base de dados, sem necessidade de autenticação.

/searchForEvents: Ponto de acesso que permite realizar uma pesquisa de eventos por nome, data, hora ou localização, usando expressões regulares.

/eventDetails: Ponto de acesso para obter detalhes específicos de um evento pelo nome.

/deleteEvent: Ponto de acesso para eliminar um evento. Exige autenticação. Remove o evento da base de dados e atualiza as listas de eventos dos utilizadores associados.

/addEvent: Ponto de acesso para criar um novo evento. Exige autenticação. Insere os detalhes do evento na base de dados e atualiza a lista de eventos do utilizador que o criou.

Gestão de Utilizadores:

/userInfoUpdate: Ponto de acesso para atualizar informações do utilizador, como descrição ou contactos.

/postInfoUpdate: Ponto de acesso para atualizar detalhes de um evento específico, como título, data, hora, descrição e imagem.

/myEvents: Ponto de acesso para listar eventos criados por um utilizador.

/myFavorites: Ponto de acesso para listar os eventos favoritos de um utilizador, verificando autenticação e compilando uma lista com base nos favoritos armazenados.

/verifyIfUserIsLoggendIn: Ponto de acesso para verificar a autenticação do utilizador.

Funções de Base de Dados:

insertLinesOnDatabase: Função para inserir documentos na base de dados.

updateObjectField: Função para atualizar um campo específico de um documento.

updateUserInfo: Função para atualizar informações de um utilizador.

updatePost: Função para atualizar informações de um evento.

findOneResult: Função para encontrar um documento único baseado num filtro.

findAll: Função para encontrar todos os documentos que correspondam a um filtro.

deleteEvent: Função para apagar um evento e atualizar os utilizadores que têm esse evento listado.

Utilitários:

verifyToken: Função para verificar a validade de um token JWT.

Envio de Correio Eletrónico:

sendEmail: Função que configura e envia um e-mail usando o serviço outlook com as credenciais fornecidas nas variáveis de ambiente.

Inicialização do Servidor:

Configuração para servir ficheiros estáticos e iniciar a escuta de pedidos na porta definida nas variáveis de ambiente.

**Cliente**

**Páginas HTML**

**Index.html**

Esta página é construída com uma estrutura HTML5 e é projetada para ser responsiva em vários dispositivos, algo evidenciado pelas meta tags de viewport e pelo uso do Bootstrap. A página tem um esquema de navegação que inclui ligações para as secções principais do site, como "Eventos" e "Os meus eventos", e também oferece opções de "Login" e "Registar" para a gestão de sessão do utilizador.

Seguem-se três secções de conteúdo que destacam os principais apelos do site: descobrir eventos culturais locais, conectar-se com a comunidade e experienciar eventos memoráveis. Cada secção é acompanhada de imagens e descrições que convidam o utilizador a explorar mais.

No rodapé, encontra-se informação de direitos de autor e uma ligação para a página "Sobre Nós", que contém informações adicionais sobre os criadores do website.

Além disso, o site utiliza vários scripts para melhorar a experiência do utilizador após o carregamento da página. O design da página é complementado com ícones da Font Awesome e fontes personalizadas do Google.

**Register.html**

A página utiliza vários JavaScripts para a gestão de utilizadores e validações, bem como folhas de estilo CSS para o design. A barra de navegação no cabeçalho permite o acesso fácil a outras partes do site.

O corpo principal da página consiste num formulário de registo, onde os utilizadores podem inserir o seu email, nome de utilizador e palavra-passe. O design do formulário é simples e direto, focado na usabilidade. Um botão de registo inicia o processo de verificação e criação de conta.

**Login.html**

O formulário de login está centrado na página e pede ao utilizador o seu nome de utilizador e palavra-passe. O design do formulário é consistente com o estilo geral do site, utilizando CSS para estilização e scripts para funcionalidades adicionais, como a validação de login.

Um botão de "Entrar" inicia o processo de autenticação.

**Events.html**

O ficheiro events.html é dedicado à listagem e pesquisa de eventos no website "Event Eagle". Inclui uma barra de navegação no cabeçalho, um campo de pesquisa para encontrar eventos e uma área para exibir a lista de eventos.

Além disso, a página contém uma funcionalidade de popup para detalhes dos eventos, onde os utilizadores podem ver mais informações sobre cada evento, incluindo imagens, descrições e um mapa de localização integrado com a API do OpenStreetMap (Leaflet).

O design da página segue o estilo geral do site, com um rodapé consistente. Scripts JavaScript são utilizados para carregar e mostrar os eventos, lidar com a pesquisa e gerir a interatividade com os detalhes dos eventos. A página está desenhada para ser funcional e amigável ao utilizador, facilitando a descoberta e o acesso a informações sobre eventos.

**Myevents.html**

O ficheiro myevents.html é uma página personalizada onde os utilizadores podem gerir os seus eventos. Esta inclui funcionalidades como a criação de novos eventos, visualização e edição dos eventos existentes, e uma secção para os eventos favoritos do utilizador.

A página é estruturada com uma barra de navegação, um campo de pesquisa para eventos, e áreas dedicadas para a listagem dos eventos criados pelo utilizador e os seus eventos favoritos. Há também uma funcionalidade de popup que permite editar detalhes dos eventos.

**AboutUs.html**

A página aboutUs.html do site "Event Eagle" é dedicada à apresentação da equipa responsável pelo projeto, nós! Inclui uma barra de navegação no topo e um cabeçalho principal onde estão listados os nomes e números de estudante dos membros da equipa.

**Javascripts**

**editMyEvents.js**

O script editMyEvents.js é utilizado para gerir eventos criados e favoritos do utilizador. Inclui as seguintes funções:

* myEvents(): Carrega e exibe os eventos criados pelo utilizador.
* myFavorites(): Carrega e exibe os eventos favoritos do utilizador.
* myEventSearcher(): Permite a pesquisa de eventos na lista de eventos do utilizador.
* createMyEventsList(eventsList): Cria e mostra uma lista de eventos criados pelo utilizador.
* createMyEventsFavorite(eventsList): Cria e mostra uma lista dos eventos favoritos do utilizador.
* myEventDetails(eventName): Exibe detalhes de um evento específico, incluindo localização no mapa.
* deleteMyEvent(eventName): Remove um evento da lista do utilizador.
* saveNewContent(): Salva as edições feitas num evento pelo utilizador.

Estas funções são essenciais para a interatividade na gestão de eventos no site, permitindo aos utilizadores ver, editar e organizar seus eventos e favoritos de maneira eficiente.

**eventsList.js**  
O script eventsList.js é utilizado para gerir a visualização e interação com eventos no site "Event Eagle". As principais funções incluem:

* displayEvents(): Mostra todos os eventos registados. Faz uma chamada GET ao endpoint /registeredEvents para obter a lista de eventos.
* eventSearcher(): Permite aos utilizadores pesquisar eventos. Envia uma pesquisa ao endpoint /searchForEvents e mostra os resultados.
* prepareList(eventsListSearch): Prepara a lista de eventos pesquisados para exibição.
* createEventsList(eventsList): Cria a visualização dos eventos na página, incluindo imagens e informações básicas.
* eventDetails(eventName): Exibe detalhes de um evento específico, incluindo localização no mapa.
* verifyCoordinates(eventCoordinates): Verifica se as coordenadas de um evento estão ou não válidas.
* verifyIfThereIsAUserLoggedIn(): Verifica se existe um utilizador autenticado.
* hideEventDetails(): Esconde os detalhes de um evento.
* callMapsAPI(eventLatitude, eventLongitude): Chama a API do OpenStreetMap para exibir a localização do evento.
* addToMyEvents(): Adiciona um evento aos favoritos do utilizador.
* createPosts(): Prepara a página para a criação de novos eventos.
* managePosts(): Gerencia a visualização de eventos criados pelo utilizador.
* addEvent(): Adiciona um novo evento à lista do utilizador.
* myPosts(): Carrega e exibe os posts de um utilizador.
* createList(events): Cria uma lista de eventos para exibição.

Este script é essencial para a funcionalidade interativa do site, permitindo aos utilizadores visualizar, pesquisar, editar e gerir eventos de forma eficaz.

**loginSignUpVerifications.js**

O script loginSignUpVerifications.js é utilizado no site "Event Eagle" para validar as informações fornecidas pelos utilizadores durante os processos de login e registo. As principais funções incluem:

* validateLogin(): Verifica se os campos de nome de utilizador e palavra-passe estão preenchidos durante o login. Se estiverem, chama a função de login.
* validateSignUp(): Confirma se os campos de email, nome de utilizador e palavra-passe estão preenchidos no registo. Verifica também se a palavra-passe tem cinco ou mais caracteres. Se tudo estiver correto, prossegue com o registo.
* isEmpty(): Função auxiliar para verificar se um campo está vazio. Se estiver, exibe uma mensagem de erro.

Este script é essencial para garantir que os dados fornecidos pelos utilizadores sejam válidos e completos antes de prosseguirem com o login ou o registo.

**userHandle.js**  
O script userHandle.js é responsável por lidar com várias funcionalidades relacionadas a utilizadores. Aqui estão as principais funções incluídas neste script:

* signUp(): Esta função é utilizada para registar um novo utilizador. Ela faz uma solicitação para o servidor com os dados de email, nome de utilizador e palavra-passe fornecidos pelo utilizador. Se o registo for bem-sucedido, redireciona o utilizador para a página principal.
* login(): Realiza o processo de autenticação do utilizador. Envolve o envio do nome de utilizador e palavra-passe ao servidor para verificação. Se o login for bem-sucedido, o utilizador é redirecionado para a página principal.
* saveContent(): Utilizada para salvar a descrição do utilizador. Esta função faz uma solicitação ao servidor com a informação inserida pelo utilizador e atualiza o perfil do utilizador com essa informação.
* logout(): Efetua o logout do utilizador. Remove os dados de autenticação armazenados no navegador e redireciona o utilizador para a página de login.
* logoutByExpiredToken(): Realiza o logout devido a um token expirado. Remove os dados de autenticação armazenados no navegador e redireciona o utilizador para a página de login.
* renderPageAfterLoadHtmlElements(): Esta função é chamada após a carga dos elementos HTML da página. Ela verifica se o utilizador está autenticado e ajusta a exibição dos elementos da página com base no estado de login. Por exemplo, se o utilizador estiver autenticado, o nome de utilizador é exibido e os botões de login e registo são ocultados.
* makeRequest(url, options): É uma função auxiliar utilizada para fazer solicitações HTTP ao servidor. Ela envia uma solicitação para a URL especificada com as opções fornecidas e retorna a resposta do servidor.

Este script desempenha um papel fundamental na gestão de utilizadores e autenticação no site "Event Eagle", permitindo que os utilizadores se registem, façam login, atualizem as suas informações e efetuem logout de forma segura.