Universidade: Universidade Europeia
Faculdade: IADE - Faculdade de Design, Tecnologia e Comunicação
Curso: Engenharia Informática

Índice

Elementos do Grupo:

- Pedro Dias
- Leonardo Nguyen
- Silésio Pipa
- Fausto Bettencourt

1. Introdução

1.1 Contexto

A crescente oferta de eventos tecnológicos (conferências, meetups, workshops) torna cada vez mais difícil para utilizadores encontrarem iniciativas relevantes de forma rápida e eficaz. As plataformas tradicionais de eventos recorrem sobretudo a filtros rígidos e pesquisas baseadas em palavras-chave, o que limita a expressividade da intenção do utilizador e conduz frequentemente a resultados pouco relevantes.

1.2 Problema

Os utilizadores enfrentam dificuldades em encontrar eventos alinhados com os seus interesses, disponibilidade temporal e contexto pessoal, devido a mecanismos de pesquisa pouco flexíveis e à ausência de personalização.

1.3 Objetivos

O objetivo deste projeto é desenvolver uma plataforma web que permita:

- Descoberta inteligente de eventos tecnológicos;
- Pesquisa em linguagem natural suportada por Inteligência Artificial;
- Personalização da experiência do utilizador;
- Autenticação segura e gestão de perfis.

1.4 Visão Geral da Solução

A confAI é uma aplicação web que integra tecnologias modernas de frontend e backend, combinadas com um componente de Inteligência Artificial (CSP) para interpretação semântica de consultas em linguagem natural, autenticação via Clerk e uma arquitetura escalável baseada em serviços.

ID	Nome	Ator Principal
UC1	Autenticar utilizador	Utilizador
UC2	Pesquisar eventos	Utilizador
UC3	Pesquisa inteligente (linguagem natural)	Utilizador
UC4	Visualizar detalhes de evento	Utilizador

2.2 Caso de Utilização Principal — Pesquisa Inteligente de Eventos

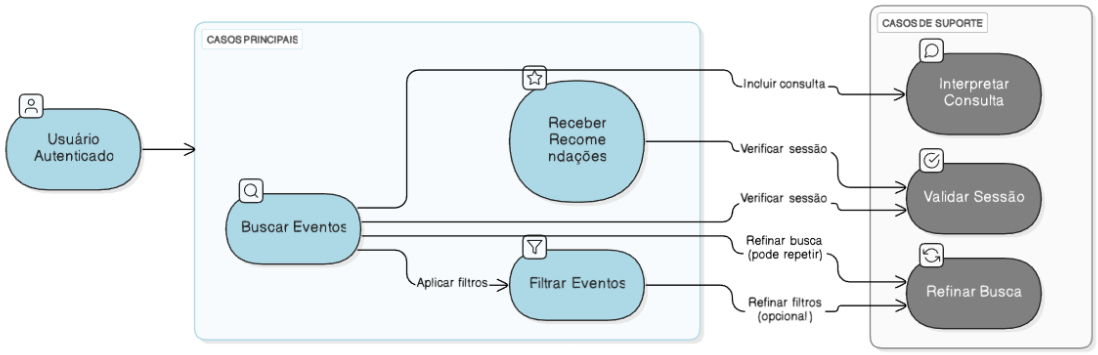
Ator: Utilizador autenticado
Objetivo: Encontrar eventos relevantes através de linguagem natural
Pré-condições: Utilizador autenticado via Clerk

Pós-condições: Lista de eventos relevantes apresentada ao utilizador
Fluxo Principal:

1. O utilizador insere uma consulta em linguagem natural.
2. O frontend valida a sessão do utilizador.
3. O backend envia a consulta ao componente de IA.
4. A IA interpreta a intenção e gera filtros estruturados.
5. O sistema executa uma pesquisa híbrida (semântica + filtros).
6. Os eventos são ordenados por relevância e apresentados.

Fluxo Alternativo:
Se a consulta for ambígua, o sistema solicita refinamento ao utilizador.

2.3 Diagrama de Casos de Uso

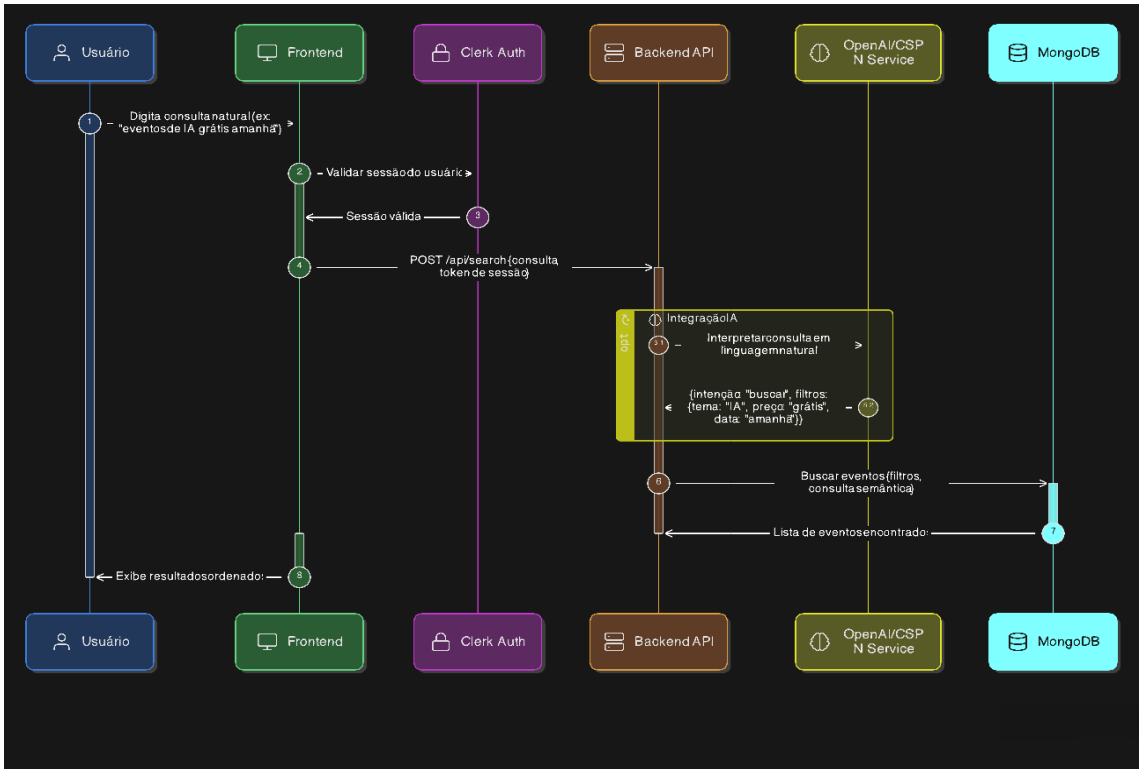


O caso de uso descreve a pesquisa por linguagem natural.

3. Diagramas do Sistema

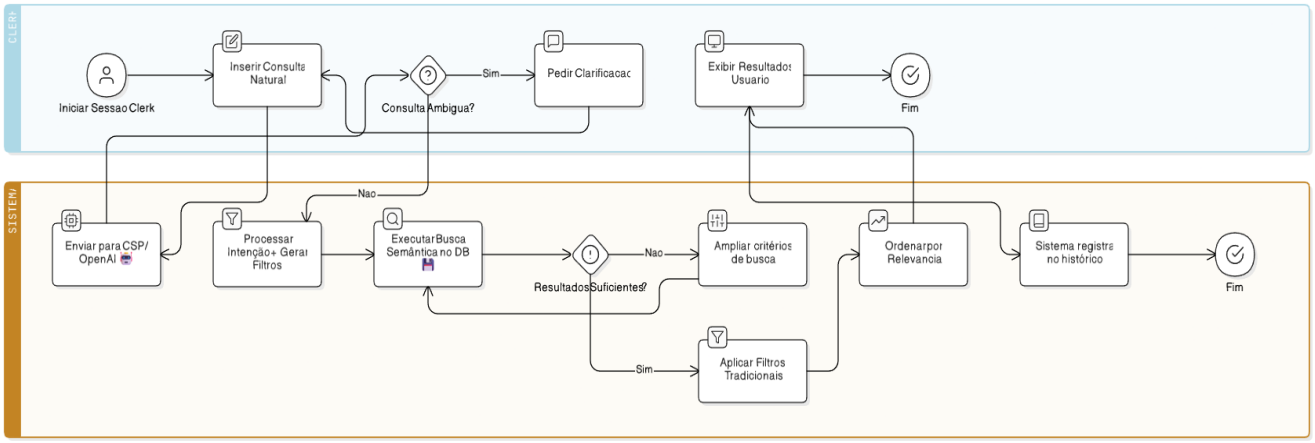
Esta secção apresenta os principais diagramas UML utilizados para modelar o comportamento e a estrutura do sistema, de acordo com as boas práticas de Engenharia de Software.

3.1 Diagrama de Sequência



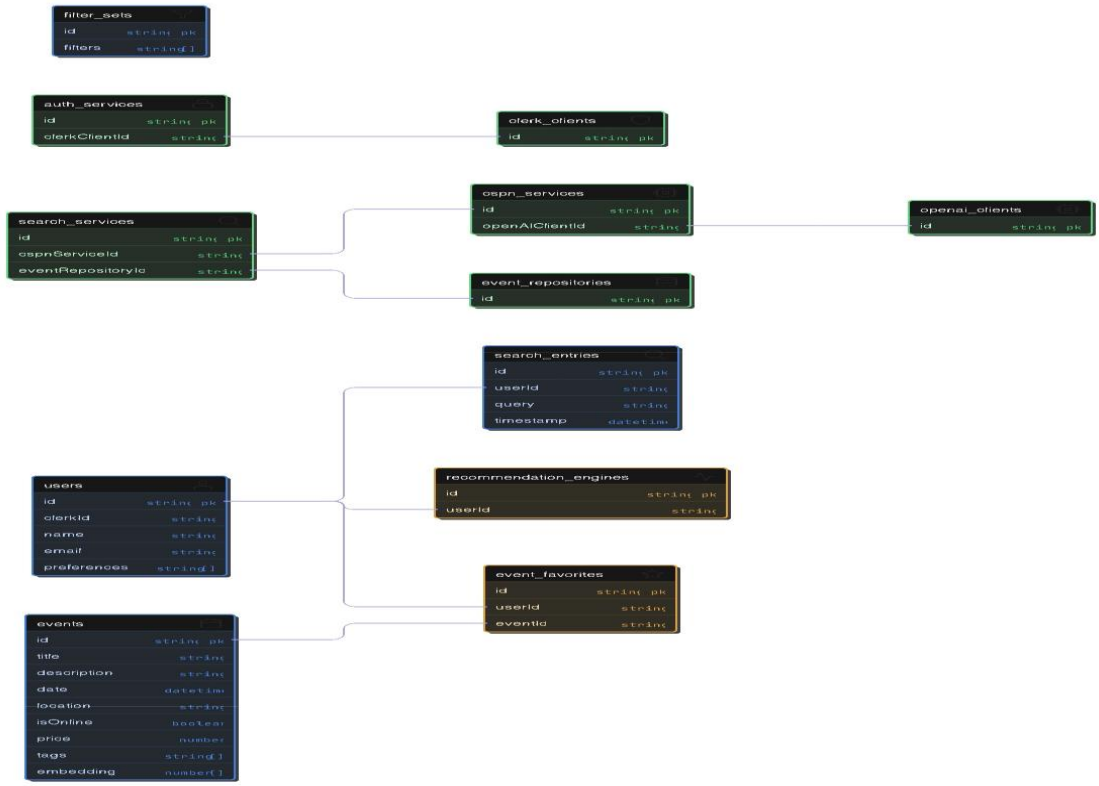
Representa o fluxo principal da busca inteligente.

3.2 Diagrama de Atividades



Mostra o workflow da busca inteligente de eventos.

3.3 Diagrama de Classes



4. Arquitetura da Solução

4.1 Visão Geral

A arquitetura da ConfAI segue um modelo modular e escalável, baseado numa separação clara de responsabilidades entre frontend, backend, serviços externos e base de dados.

4.2 Componentes Principais

Frontend

- Implementado com Next.js
- Interface responsiva
- Comunicação com backend via HTTPS

Backend

- API Routes em Next.js
- Responsável por lógica de negócio, integração com IA e base de dados

Componente de IA

- OpenAI para processamento de linguagem natural
- CSP para interpretação contextual e geração de filtros
- Suporte a pesquisa semântica e híbrida

Autenticação

- Clerk para autenticação e gestão de sessões
- OAuth e gestão segura de utilizadores

Base de Dados

- MongoDB
- Armazenamento de eventos, utilizadores e embeddings semânticos

4.3 Containerização com Docker

De forma a garantir consistência entre ambientes de desenvolvimento, teste e produção, o projeto recorre à utilização de **Docker** para a containerização dos principais componentes do sistema.
A utilização de Docker permite encapsular a aplicação e as suas dependências num ambiente isolado, reduzindo problemas associados a diferenças de configuração entre máquinas e facilitando o processo de instalação e execução do sistema.

4.3.1 Motivação para a utilização de Docker

O Docker foi utilizado pelos seguintes motivos principais:

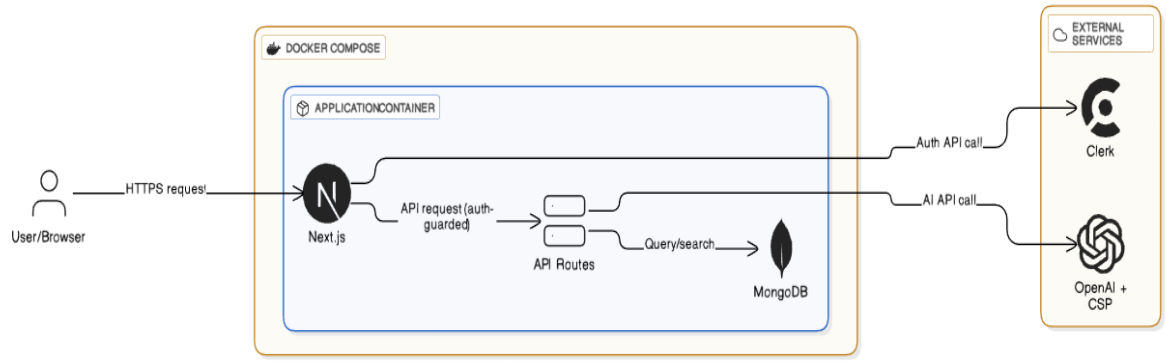
- **Portabilidade:** a aplicação pode ser executada em qualquer sistema que suporte Docker, independentemente do sistema operativo.
- **Reprodutibilidade:** todos os membros da equipa utilizam exatamente o mesmo ambiente de execução.
- **Facilidade de deploy:** simplifica o processo de publicação da aplicação em ambientes de produção ou cloud.
- **Isolamento:** evita conflitos de dependências entre o projeto e o sistema anfitrião.
- **Escalabilidade futura:** facilita a adoção de arquiteturas baseadas em microserviços.

4.3.2 Arquitetura baseada em Containers

A arquitetura do sistema é organizada em múltiplos containers, cada um com uma responsabilidade bem definida:

- **Container Frontend/Backend (Next.js):**
  Responsável pela interface do utilizador, API routes e lógica de negócio.
- **Container da Base de Dados (MongoDB):**
  Armazena dados de eventos, utilizadores e embeddings semânticos.
- **Integrações Externas:**
  Serviços como OpenAI e Clerk são consumidos via APIs externas seguras, não sendo containerizados diretamente, mas integrados no fluxo da aplicação.
  A comunicação entre os containers é realizada através de uma rede interna definida pelo Docker, garantindo segurança e isolamento

4.4 Diagrama de Arquitetura



5. Componente de Inteligência Artificial

5.1 Objetivo do Uso de IA

O principal objetivo da utilização de Inteligência Artificial no projeto é permitir a pesquisa de eventos através de **linguagem natural**, ultrapassando as limitações das pesquisas baseadas exclusivamente em palavras-chave e filtros rígidos. A IA possibilita interpretar a intenção do utilizador, mesmo quando a consulta é ambígua ou pouco estruturada.

5.2 Arquitetura do Componente de IA

O componente de IA está integrado no backend da aplicação e é composto por dois elementos principais:

- **Modelo de Linguagem (OpenAI):** responsável pelo processamento de linguagem natural.
- **CSP:** responsável por estruturar o raciocínio, interpretar o contexto e transformar a consulta do utilizador em filtros utilizáveis pelo sistema.

Este componente comunica de forma síncrona com o backend através de chamadas HTTPS seguras.

5.3 Funcionamento

O funcionamento do componente de IA segue os seguintes passos:

1. O utilizador introduz uma consulta em linguagem natural.
2. O backend envia a consulta ao serviço de IA.
3. O CSP interpreta a intenção, contexto e restrições temporais ou geográficas.
4. A IA devolve uma estrutura de dados contendo filtros e parâmetros de pesquisa.
5. O backend executa uma pesquisa híbrida (semântica + filtros).
6. Os resultados são ordenados por relevância e apresentados ao utilizador.

5.4 Testes Efetuados

Foram realizados testes para avaliar o comportamento e eficácia do componente de IA:

- **Testes funcionais:** validação da correta interpretação de consultas simples e complexas.
- **Testes de integração:** verificação da comunicação entre IA, backend e base de dados.

- **Testes comparativos:** comparação entre pesquisa tradicional e pesquisa com IA.

5.5 Métricas de Desempenho

Métrica	Resultado
Taxa de interpretação correta	~85%
Tempo médio de resposta	< 2 segundos
Taxa de sucesso da pesquisa	~80%
Redução de interações do utilizador	Significativa

5.6 Discussão da Eficácia

Os resultados demonstram que a utilização de IA melhora significativamente a experiência do utilizador, reduzindo o esforço necessário para encontrar eventos relevantes. No entanto, consultas altamente ambíguas ainda requerem refinamento adicional, o que representa uma oportunidade de melhoria futura.

6. Segurança

6.1 Autenticação e Autorização

A autenticação é realizada através do serviço **Clerk**, que fornece mecanismos seguros de login, gestão de sessões e OAuth. Apenas utilizadores autenticados podem aceder às funcionalidades principais da aplicação.

6.2 Segurança das Comunicações

Todas as comunicações entre frontend, backend e serviços externos são realizadas através de **HTTPS**, garantindo confidencialidade e integridade dos dados transmitidos.

6.3 Boas Práticas de Código Seguro

Foram adotadas várias boas práticas, incluindo:

- Utilização de variáveis de ambiente para dados sensíveis;
- Validação de inputs do utilizador;
- Proteção das rotas da API através de middleware;
- Não exposição de chaves de API no repositório.

6.4 Considerações Futuras

Como trabalho futuro, poderão ser adicionados mecanismos de:

- Rate limiting;
- Auditoria de acessos;
- Monitorização de tentativas de acesso indevido.

7. Base de Dados

7.1 Modelo de Dados

A base de dados utilizada é **MongoDB**, escolhida pela sua flexibilidade e boa.
No projeto original, a base de dados é criada dinamicamente pela aplicação aquando da inserção de dados, seguindo a abordagem típica de aplicações MongoDB. No entanto, para efeitos de documentação e reprodutibilidade, foram disponibilizados scripts de

inicialização da base de dados no repositório, permitindo a criação explícita das coleções, definição de índices e inserção de dados de teste.

7.3 Exemplos de Pesquisas
As pesquisas incluem filtros por data, localização, preço e similaridade semântica, garantindo resultados relevantes e eficientes.

8. Manual do Utilizador
8.1 Autenticação
1. Aceder à aplicação.
2. Efetuar login ou registo através do sistema de autenticação.
3. Após autenticação, o utilizador é redirecionado para a página principal.

8.2 Pesquisa de Eventos
1. Introduzir uma consulta em linguagem natural.
2. Visualizar os resultados apresentados.
3. Aplicar filtros adicionais, se necessário.

8.3 Gestão de Eventos
- Visualizar detalhes de um evento;
- Guardar eventos como favoritos;
- Consultar eventos guardados.

9. Gestão do Projeto
9.1 Metodologia
Foi utilizada uma abordagem **ágil**, inspirada em Scrum, com sprints semanais, reuniões regulares e entregas incrementais.

9.2 Ferramentas
- GitHub (controlo de versões)
- Docker (containerização)

10. Conclusão e Trabalho Futuro

O projeto ConfAI atingiu os objetivos propostos, demonstrando a aplicação prática de conceitos de Engenharia de Software, Inteligência Artificial e Segurança. A integração de IA revelou-se eficaz na melhoria da experiência do utilizador.
Como trabalho futuro, destaca-se:
- Melhoria da precisão da IA;
- Expansão do sistema de recomendações;
- Escalabilidade para um maior número de utilizadores.