Arquitectura da Aplicação

Equipa
Pando [PL5]

Disciplina
Engenharia de Software

Data
11.11.2018

Autores
Membros da Unidade de ENV:
Francisco Monteiro, Coordenador de Ambiente
Rodrigo Cardoso, Vice-Coordenador de Ambiente
Carolina Bandeira, Colaboradora de Ambiente
João Cunha, Colaborador de Ambiente

Com o apoio da Unidade de IMP:
Tiago Martins, Coordenador de Implementação
Diogo Isidoro, Vice-Coordenador de Implementação
Alexandre Faria, Colaborador de Implementação
João Almeida, Colaborador de Implementação
Diogo Loureiro, Colaborador de Implementação
Daniel Santos, Colaborador de Implementação

Índice

Glossário

ORCID	Open Researcher and Contributor ID
Framework	Software que fornece uma forma padrão de desenvolver e implementar uma aplicação
SGBD	Sistema de Gestão de Base de Dados
PK	Primary Key (Chave Primária)
API	Application Programming Interface
REST	Representation State Transfer
MVC	Model-View-Controller
HTTP	Hypertext Transfer Protocol
SQL	Structured Query Language (Linguagem de Consulta Estruturada)

Referências

ES2017_AD_BRTAG_v2.0.pdf

Ch6_Architectural_design.pdf

https://dzone.com/articles/an-introduction-to-restful-apis

Link para o sítio, no gitHub, onde se encontra este mesmo documento

https://github.com/pl5es/ES/tree/master/information/Ambiente

Introdução

No ciclo de vida do desenvolvimento de software podemos encontrar diversas etapas e todas com seus propósitos muito bem definidos, porém, uma que podemos dizer com segurança ser de extrema importância é a arquitetura do software.

Este documento apresenta a arquitetura utilizada na realização do nosso programa. É apresentada através de um conjunto de perspetivas que juntas visam cobrir os principais aspectos técnicos relativos ao desenvolvimento e implementação do sistema em questão. O objetivo é capturar e formalizar as principais decisões tomadas em relação à arquitetura do sistema.

Possibilidades de Software

1)Possíveis Frameworks

No desenvolvimento de um site, é necessário um conjunto de componentes que permitam a sua criação, de forma a ser estável e permita um rápido desenvolvimento. Para o efeito foram criadas as frameworks, que disponibilizam os componentes prontos a usar.

1.1 - Ruby on Rails

Foi projetado para ajudar os developers a levar os aplicativos desde o conceito à conclusão o mais rápido possível sem precisar de reinventar.
Ruby é uma linguagem dinâmica, open source com foco na simplicidade e na produtividade. Tem uma sintaxe elegante de leitura natural e fácil escrita. Ruby on Rails é um dos frameworks existentes projetado para tornar os aplicativos web de programação mais fáceis, fazendo suposições sobre o que o developer precisa para começar, de modo a escrever menos código.
Tem como base o padrão de arquitetura MVC.

2)Possíveis SGBD

2.1 - MySQL

Este SGBD tem como base a linguagem SQL, funciona através da interação diretamente com a consola interna. O seu sucesso deve-se ao facto de ser fácil de utilizar e ter ainda uma grande portabilidade e compatibilidade. Para além disso, possui ainda a possibilidade de integrar interfaces gráficas, como o MySQL Toolkit.

2.2 - PostgreSQL

Este SGBD também tem como base a linguagem SQL e é muito semelhante ao MySQL. Para além disso, uma das vantagens relativamente ao MySQL é o facto de esta ser open-source.

Software Escolhido

Para o desenvolvimento da *user interface* utilizamos o **React** por ser uma biblioteca em JavaScript sendo que é principalmente destinada ao desenvolvimento de páginas web.

Na escolha da framework não houve dúvidas na altura da escolha.
A elaboração da *web framework* foi feita num framework livre, **Ruby on Rails,** que foi escolhido pela sua facilidade e rapidez no desenvolvimento de sites orientados a banco de dados. É um projeto de código aberto escrito na linguagem de programação Ruby.

O **PostgreSQL** é um dos SGBDs (Sistema de Gestão de Bases de Dados) de código aberto mais avançados, contando com recursos como:

- Consultas complexas
- Chaves estrangeiras
- Integridade transacional
- Linguagem Procedimental em várias linguagens
- Facilidade de Acesso

Sendo este o utilizado pela nossa equipa de Implementação para gerir as bases de dados do nosso sistema.

Descrição da Arquitetura

Entidades

User

Entidade referente a cada utilizador na rede.

Esta entidade contém os seguintes atributos:

- ID - Número inteiro identificador único de cada utilizador
- Description - Descrição do utilizador
- Email - Endereço eletrónico do utilizador
- Institution - Instituição à qual o utilizador pertence
- Name - Nome do utilizador
- orcid - Código alfanumérico para identificar exclusivamente um investigador
- research_area - Área de investigação do investigador
- username - Username do utilizador
- interests - Lista de interesses do utilizador
- created_at - Data de criação
- updated_at - Data de atualização
- avatar - Lista de avatares do utilizador
    - url - url respectivo da fotografia

A entidade user tem as seguintes relações:

- UserHasFolder - Um utilizador pode ter zero ou mais pastas
- UserHasBookmark - Um utilizador pode ter zero ou mais favoritos

Folder

Entidade referente às pastas que guardam os favoritos.

Esta entidade contém os seguintes atributos:

- ID - Número inteiro identificador único de cada pasta
- title - Título/nome da pasta
- bookmarks - Lista de favoritos
- created_at - Data de criação
- updated_at - Data de atualização

A entidade folder tem as seguintes relações:

- FolderHasBookmark - Uma pasta pode ter zero ou mais favoritos
- UserHasFolder - Um utilizador pode ter zero ou mais pastas

Bookmark

Entidade referentes aos favoritos.

Esta entidade tem os seguintes atributos:

- ID - Número inteiro identificador único de cada favorito
- title - Título/nome do favorito
- url - url do bookmark
- interests - Lista de interesses

A entidade bookmark tem as seguintes relações:

- UserHasBookmark - Um utilizador pode ter zero ou mais favoritos
- FolderHasBookmark - Uma pasta pode ter zero ou mais favoritos

Interest

Entidade referente aos interesses.

Esta entidade tem os seguintes atributos:

- ID - Número inteiro identificador próprio de cada interesse
- hashtag - Hashtag relativa ao interesse
- created_at - Data de criação
- updated_at - Data de atualização

A entidade interest tem as seguintes relações:

- UserHasInterest - Um utilizador pode ter zero ou mais interesses
- BookmarkHasInterest - Um favorito pode ter zero ou mais interesses

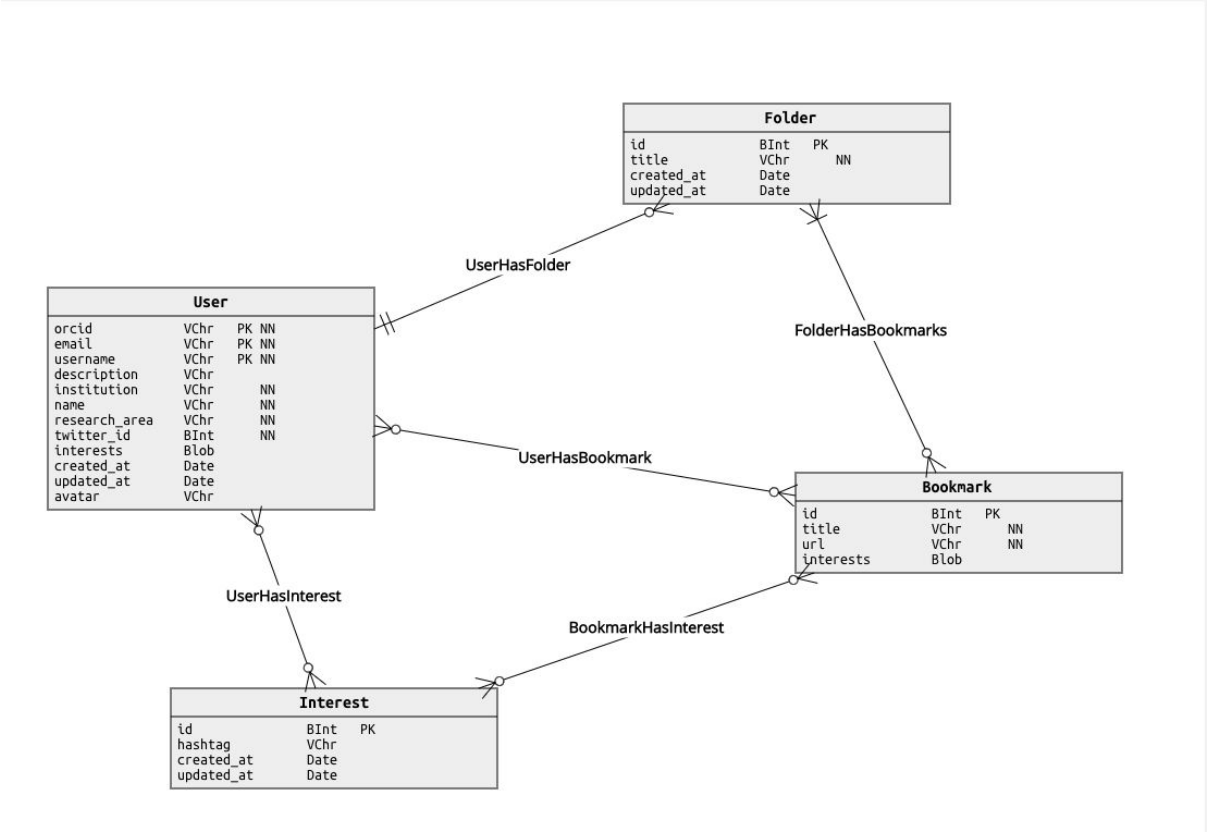

Figura 1 - Diagrama ER

REST API

Este estilo de arquitetura opera seguindo um conjunto de restrições e propriedades que obedecem ao protocolo HTTP. Um dos pontos fortes deste tipo de arquitetura é que os serviços que o utilizam possuem uma grande capacidade e simplicidade de comunicar entre si.

De uma forma geral, uma REST API funciona similarmente a um website. O cliente começa por fazer um pedido (call) para o servidor e este responde-lhe através do protocolo HTTP.

Esta abordagem tem como base os “HTTP Request Methods” (Métodos de pedido do HTTP). Os métodos a ser utilizados são: GET, PUT, POST e DELETE.

GET

É um dos métodos HTTP mais comuns, é usado para fazer “request” (pedido) de “data” (informação) de um determinado “resource” (recurso).

PUT

O PUT é usado para enviar “data” para o servidor de forma a criar ou atualizar um determinado “resource”.

POST

A diferença entre o PUT e o POST é que, chamar o mesmo método PUT múltiplas vezes terá sempre o mesmo resultado. Em contrapartida, chamar o método POST repetidamente, tem o mesmo efeito que criar o mesmo “resource” diversas vezes.

DELETE

Este método apaga um determinado “resource”.

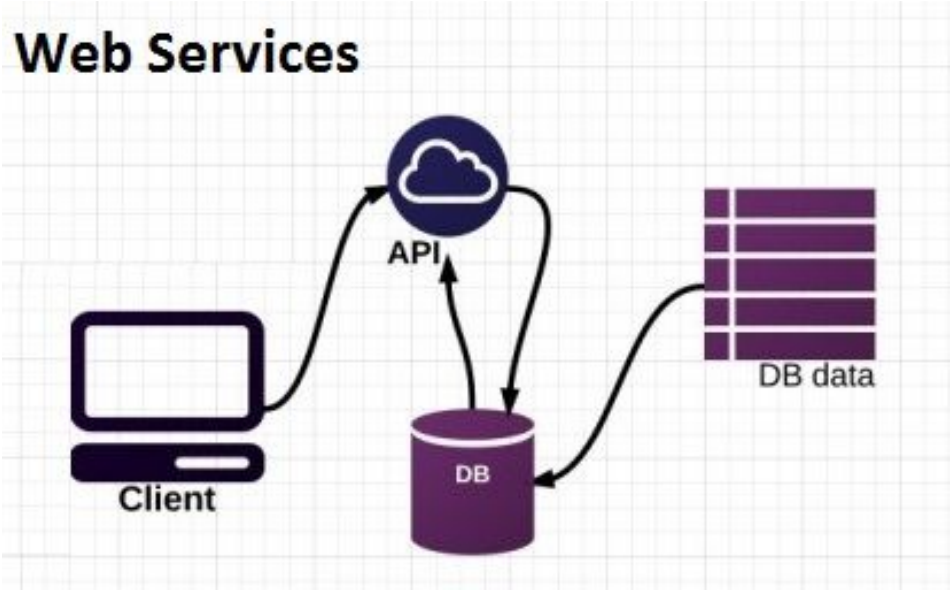

Figura 2 - REST API

SPA

Utilizou-se um sistema Single-page Application (SPA) na utilização da aplicação na web, que utiliza recursos como HTML+CSS+JavaScript.
O SPA de uma forma geral, reescreve as páginas web dinamicamente em vez de carregar novas páginas do servidor. Esta abordagem torna-se muito útil e tem bastantes vantagens quando utilizada.

Alguns pontos positivos de utilizar este sistema são:

- Redução do tempo de espera de carregar a página web;
- Não será necessário fazer reloads da página web;
- Pode facilitar o desenvolvimento da aplicação para o programador;
- Simplifica a utilização do utilizador;

Tendo também alguns pontos negativos de usar este sistema como:

- Se o utilizador desativar o JavaScript do seu browser não vai ser possível ver a web corretamente;

Conclusão

As decisões sobre a arquitetura do trabalho foram tomadas pela equipa de implementação, de modo a facilitar o seu trabalho e ao mesmo tempo garantir que o software tenha qualidade.