TÉCNICO LISBOA

Ser Músico em Portugal

Publicação e Visualização de Dados com Solução Mediawiki e Arquitetura MVC

Tiago José Ribeiro Teixeira

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Orientadores: Prof. Helena Sofia Andrade Nunes Pereira Pinto
Prof. José Luís Brinquete Borbinha

Júri

Presidente: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Orientador: Prof. Helena Sofia Andrade Nunes Pereira Pinto
Vogal: Prof. António Manuel Ferreira Rito da Silva

Dezembro 2020

Agradecimentos

Quero começar por agradecer aos meus professores orientadores, prof. Helena Sofia Pinto e o prof. José Borbinha pela oportunidade de trabalhar num projeto tão interessantes como este, e por todo o apoio, conselhos e ajuda no desenvolvimento do mesmo. Quero também agradecer à prof. Maria João Albuquerque e a todos os envolvidos no projeto PROFMUS pela ajuda, informação, ensinamentos e disponibilidade durante todo o processo, principalmente no momento de testarem a aplicação desenvolvida. Gostava também de agradecer ao Instituto de Etnomusicologia - Centro de Estudos em Música e Dança (INET-MD - NOVA FCSH) pela bolsa providenciada no âmbito do projeto PROFMUS (PTDC/ART-PER/32624/2017).

Quero agradecer aos meus pais, pelo apoio e motivação que sempre me deram desde que este percurso académico se iniciou e também pelos valores que me incutiram e que construíram a pessoa que sou hoje. Esta tese foi desenvolvida em honra a todos os esforços que fizeram por mim.

À minha avó materna pela boa disposição e conselhos sempre tão sábios.

À minha namorada por ter sido o principal pilar neste percurso onde esteve sempre disposta a dar-me toda a motivação e ajuda principalmente nos momentos mais difíceis. Agradecer-lhe pelo amor, pela delicadeza e pela prontidão em transmitir boas energias. Um obrigado nunca será suficiente.

À restante família e amigos.

No decorrer deste mestrado, partiram duas das pessoas mais importantes da minha vida. Com elas cresci, brinquei e aprendi valores que não vou esquecer nunca. Sei que estejam onde estiverem, estão e vão estar sempre a torcer por mim. Daqui de baixo, resta me homenagear-vos todos os dias. Esta conquista, é para vocês.

Um especial obrigado,

Até já...

Abstract

This project aims to develop an information system that allows to manage all the information about musicians and professionals of this type of art, who lived in Portugal, between 1750 and 1985. Currently, this information is spread over several digital platforms and also handwritten ones, which does not turn easy the user's access to it. For this reason, the primary goal of this project is to create an information system on the Web that aggregates all the available dataset but preserving all the existing information in its original shape. In order to create this information system, an ontology will be built using Wikibase and Mediawiki technologies. While creating and defining the ontology, two web applications will be developed, one of which will be assigned to the management of the ontology and the other to view the data, a task that will be developed by me as the main focus of my dissertation.

Keywords

Music, Musicians, Musical Art, Information Systems, Ontologies, Web

Resumo

O presente projeto ambiciona desenvolver um sistema de informação que permita num só local gerir toda a informação sobre músicos e profissionais desta arte, que tenham vivido em portugal, entre 1750 e 1985. Atualmente, estas informações estão dispersas por várias plataformas digitais e também manuscritas, o que não facilita o acesso ao utilizador. Por essa mesma razão, o objetivo primordial deste projeto visa então criar um sistema de informação na Web que agregue todo o *dataset* disponível mas preservando toda a informação pré-definida no seu estado original. Para a criação deste sistema de informação vai ser construída uma ontologia recorrendo às tecnologias da Wikibase e da Mediawiki. Simultaneamente à criação e definição da ontologia vão ser desenvolvidas duas aplicações Web, em que uma estará atribuída à gestão da ontologia e outra para a visualização dos dados, tarefa esta que será desenvolvida por mim como principal foco da minha dissertação.

Palavras Chave

Música, Músicos, Arte Musical, Sistemas de Informação, Ontologias, Web;

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens

Acrónimos

API Application Programming Interface

FRBR Functional Requirements for Bibliographic Records

FRBRoo Functional Requirements for Bibliographic Records-Object Oriented

IST Instituto Superior Tecnico

FRBR Functional Requirements for Bibliographic Record

SaaS Software as a Service

MVC Model-View-Controller

HTTP Hypertext Transfer Protocol

URL Uniform Resource Locator

HTML HyperText Markup Language

REST API Representational State Transfer Application Programming Interface

1

Introdução

Conteúdo

1.1 Motivação

O contexto histórico da música em Portugal incide particularmente nas obras criadas e desenvolvidas por compositores portugueses ou estrangeiros que, em algum momento das suas vidas, passaram por Portugal. Contudo, existe uma lacuna evidente no conhecimento geral relativo às instituições musicais e aos músicos em si, quer no que diz respeito à sua vida pessoal, quer no modo como esse contexto pessoal poderá estar relacionado com Portugal.

Em termos concretos, a motivação para este projeto representa uma recolha contínua no tempo de uma coleção de dados, existentes e novos, sobre pessoas que, em portugal e entre os anos de 1750 e 1985, tiveram como atividade principal ou profissão a música. Um sistema deste tipo trará vantagens sociais e culturais enormes já que vai agregar num só espaço toda a informação dispersa. Vai permitir também uma autocorreção da mesma (a maior exposição pública vai permitir um cruzamento de informação maior e mais preciso) e a criação de um perfil mais completo para cada elemento já que cada base de dados pode conter informações diferentes e complementares sobre a mesma pessoa.

Mais em concreto, a motivação para este projeto de tese será a contrução, configuração e publicação de vistas. As vistas são conjuntos de dados definidos por algum tipo de príncipío e o objetivo principal das vistas são as de agregar informação conforme os requisitos que os utilizadores pretendam. A publicação destes conjuntos de dados, segundo os coordenadores do projeto, serão bastante úteis para a comparação e para efeitos estatísticos entres os elementos musicais pertencentes a cada conjunto de dados, sejam eles referentes a um músico, a um compositor, a um espaço musical, entre outros.

O desenvolvimento deste projeto conta com a colaboração do Laboratório de Sistemas de Informação e Apoio à Decisão (IDSS), do Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento (INESC-ID) e do Instituto De Etnomusicologia - Centro de Estudos em Música e Dança (INET-md).

1.2 Problema

A pouca informação existente acerca da condição social desta classe, e a sua dispersão por variadíssimos *datasets*, em estruturas também elas bastante diferenciadas, criam um vazio de conhecimento significativo que dificulta o acesso ao histórico de um certo elemento musical, seja um músico ou um espaço físico, no período temporal em questão. De forma a tentar colmatar esta lacuna, é necessário identificar e desenvolver uma solução que permita o recurso à Web semântica, através da construção e desenvolvimento de uma ontologia (definição no próximo capítulo), para a criação de um sistema que facilite a interação simples e prática de todas as bases de dados individualmente e em simultâneo.

As principais dificuldades na construção da ontologia serão encontrar o meio mais adequado para conseguir agregar tamanha quantidade de informação nos mais variados formatos, desde imagens, registos em papel ou até mesmo bases de dados digitalizadas e guardadas em formatos tecnologicamente retrógrados.

Posteriormente à estruturação da ontologia, é necessário desenvolver aplicações onde se possa facilmente fazer a gestão da informação, assim como a visualização da mesma. Com efeito, esta investigação tem como principal meta a conceção e desenvolvimento de uma aplicação de visualização e publicação dos dados, que possibilite aos administradores dos mesmos, criar uma interface gráfica personalizada, na qual lhes seja permitido manipular vários elementos visuais e partilhar essa interface com o utilizador comum.

1.3 Resultado

Atualmente, fruto de duas dissertações de mestrado no âmbito do projeto, estão desenvolvidas duas aplicações que vão trabalhar os dados recolhidos. Inicialmente, um outro aluno foi responsável pelo desenvolvimento da aplicação de gestão dos dados. Esta aplicação permite aos utilizadores importarem os dados que já recolheram através de formulários próprios, exportar esses mesmos dados, apagar objetos já importados e permite ainda a criação de subgrupos de informação, denominadas ”Vistas”, que são conjuntos de objetos que correspondam a um certo conjunto de critérios.

A segunda aplicação, desenvolvida no âmbito da minha dissertação, trata precisamente a publicação destes conjuntos de informação, ou seja, vai permitir que os utilizadores ”construam”um site para as suas vistas. O objetivo principal da publicação é a de poder existir um termo de comparação entre as várias propriedades existentes nos vários objetos. Deste modo, a aplicação foca-se na oferta de vários elementos gráficos que permitem a comparação referida, desde um mapa geográfica a uma linha temporal, a presença também de uma tabela configurável e ainda a possibilidade de visualização de todos os dados referentes a cada objeto. Todos estes elementos são configuráveis pelo administrador da vista, podendo aceder a um conjunto de definições para cada elemento ou para a página em geral. No final, basta guardar as configurações e partilhar o URL que representa o site da vista em questão.

Em relação às duas aplicações, há seguramente ainda trabalho futuro a realizar, mas apenas com o uso um pouco mais intensivo das mesmas, como por exemplo através da importação de uma grande quantidade de dados reais, é que se poderá apurar melhor as funcionalidades que justificam uma melhoria, ou a implementação de novas funcionalidades.

1.4 Organização do documento

A organização deste documento segue o seguinte padrão:

- no segundo capítulo vão ser abordados e apresentados projetos e tecnologias relacionados e que possam ser de alguma maneira úteis para este projeto. Nomeadamente vão ser analisadas tecnologias existentes para o desenvolvimento de aplicações Web e das suas componentes, assim como vai ser abordada a aplicação desenvolvida para gestão dos dados;
- no terceiro capítulo vai ser feita uma análise ao problema a resolver, definindo os requisitos e os casos de estudo e vai ser ainda apresentada a solução funcional para a sua resolução;
- no quarto capitulo será apresentada a arquitetura da solução e pormenorizada a sua implementação;
- no quinto capítulo será demonstrado o funcionamento da solução.
- no sexto capítulo será apresentada a avaliação ao projeto recorrendo à análise dos resultados dos testes feitos a utilizadores reais;
- no sétimo e último capítulo será feita uma análise conclusiva aos aspectos principais do projeto e como seria possível realizar no futuro algum tipo de trabalho que complementaria os desenvolvimentos já realizados;

2

Trabalho Relacionado

Conteúdo

O projeto em causa tem como principal intenção a criação de uma simbiose entre uma ontologia e um sistema de informação. Por esta razão é importante abordar ambos os conceitos e as tecnologias já existentes que permitam uma interligação mais fácil entre ambos, principalmente relacionados com o contexto da música.

Apesar do foco desta tese não incidir sobre a construção da ontologia, torna-se crucial ter algumas noções bem definidas e estruturadas acerca deste tema e dos seus envolventes, uma vez que são a base de todo o projeto em si. Posto isto, verifica-se mais ainda a importância de analisar as tecnologias e projetos que possam de alguma maneira ser úteis para o desenvolvimento de uma interface Web num sistema de informação.

Para a construção da ontologia vão ser usadas as tecnologias *Wiki* tais como a *MediaWiki* e a *Wikidata*. Os conceitos destas tecnologias vão ser resumidamente detalhados numa fase posterior deste mesmo capítulo. Para o desenvolvimento da aplicação Web de visualização de dados vai ser usada a *framework* Django[1] (especificação no próximo capítulo) juntamente com as tecnologias elementares Web tais como HTML, CSS e Javascript.

2.1 Ontologias como representação de informação

Com o exponencial crescimento da Web, foi necessário otimizar as pesquisas feitas através da associação e dedução dos termos concretos presentes na Web com o contexto temático em que se inserem.

Segundo Guarino (1998) [2], "uma ontologia é uma teoria lógica que explica o significado pretendido de um vocabulário formal, ou seja, o seu compromisso ontológico com uma conceptualização particular do mundo" .

De acordo com Gruber (1993) [3] as principais componentes das ontologias são:

- Items: Objetos básicos
- Classes: conjuntos, coleções ou tipo de objetos
- Atributos: recursos ou parâmetros que os objetos podem conter
- Relações: laços lógicos que relacionam objetos com outros objetos
- Eventos: mudanças nos atributos ou relações

2.2 Ontologias no contexto musical

A digitalização do conteúdo artístico e cultural tem sido uma realidade constante nos últimos anos. Sendo a música uma das artes mais populares, também esta tem sofrido grandes alterações por forma

[1] https://www.djangoproject.com

a alcançar uma maior digitalização em todas as vertentes, desde simples notas musicais até ao produto musical final. Nesta nova realidade, foram desenvolvidas algumas ontologias com foco precisamente na arte musical.

2.2.1 DOREMUS

A ontologia DOREMUS [2] baseia-se no modelo *Functional Requirements for Bibliographic Record-object oriented* (FRBRoo) e tem o seu principal foco na descrição de obras de música clássica e tradicional e também das suas interpretações. Tem como principais objetivos: publicar e ligar dados musicais na Web, desenvolvimento de ferramentas de auxílio para a seleção de obras musicais e construir e validar ferramentas educacionais que permitam a implementação de padrões, vocabulários e de tecnologias de Web Semântica. Harchichi disse na sua publicação "DOREMUS: Doing Reusable Musical Data (2015)" [4] que as obras musicais são objetos complexos que para uma eficaz compreensão exigem a descrição das suas manifestações físicas (gravações e partituras) e dos eventos que as definem (criação, publicação e performance). Neste tipo de modelo, as obras resultam sempre de uma atividade ou de um evento que acabam por estar presentes integralmente em cada fase do processo de modelação. As atividades são executadas por agentes (pessoas ou coletividades) num determinado ponto no tempo e no espaço.

Choffé e Leresche (2016) [5] explicam que o modelo é composto na sua base pelos elementos: obra, expressão, evento, tipo, motivo, ator e algumas propriedades básicas.

Os vocabulários controlados utilizados pelo DOREMUS foram extraídos do Musicbrainz [3], da Wikidata [4] e da DBpedia [5] em conjunto com outros vocabulários tais como: ISNI[6], para pessoas e entidades coletivas; RAMEAU[7], para grupos étnicos, géneros musicais e meios de performance, UNIMARC e RDA para funções, IAML[8] para géneros musicais e meios de performance e MIMO, um thesaurus multilingue para os instrumentos de música ocidental, para meios de performance.

Além do modelo conceptual e dos vocabulários controlados, o projeto DOREMUS disponibiliza ferramentas e metodologias para o tratamento de dados utilizando tecnologias de Web Semântica.

2.2.2 BIBFRAME e Performed Music Ontology

Este projeto conta com a colaboração da Universidade de Stanford, e as bibliotecas da Columbia, Cornell, Harvard, Princeton e a biblioteca do congresso dos Estados Unidos.

[2] https://www.doremus.org/
[3] https://musicbrainz.org
[4] https://www.wikidata.org/
[5] https://wiki.dbpedia.org
[6] https://isni.org/
[7] https://rameau.bnf.fr/informations/rameauenbref
[8] https://www.iaml.info/

A Performed Music Ontology [9] pretende criar uma extensão do BIBFRAME [10] de modo a descrever registos de áudio de performances em todos os formatos recomendando vocabulários de domínios específicos relativos ao suporte físico em análise desde os primeiros formatos analógicos até aos formatos digitais mais recentes incluindo registos musicais em vídeo. Também foram acrescentados identificadores para fazer a relação entre o registo áudio da performance com os catálogos temáticos dos compositores interpretados. Foi também criada uma classe para representar a organização sonora utilizada na obra gravada permitindo assim a descrição de outros sistemas e notações musicais não estando limitado às organizações da música ocidental.

A Performed Music Ontology é uma ontologia cujo foco incide sobre a descrição de performances e por essa razão mais relacionada com a descrição de performances logo implica uma preocupação maior na descrição do acontecimento e não tanto da obra em si.

2.2.3 Music Ontology Framework

A *Music Ontology Framework* é uma tentativa de relacionar conceitos comuns sobre o domínio da música. Divide-se em três clusters representando respetivamente a informação editorial, informação de produção e anotações temporárias/decomposição de eventos. O primeiro cluster representa informação editorial através de conceitos como “álbum”, “track”, “band” e “audio file”. A *Music Ontology* usa outras ontologias como base para estes conceitos tais como a ontologia FOAF (Friend-of-a-Friend), a ontologia FRBR, a ontologia *Timeline* e a ontologia *Event*.

A ontologia FOAF é usada para representar pessoas, relações entre pessoas, grupos de pessoas e organizações. Já a ontologia FRBR fornece duas abstrações que servem de base para informações editoriais. Nesta ontologia um item é uma entidade concreta e uma manifestação abrange todos os objetos físicos que possuem as mesmas características tanto no seu conteúdo real como na sua forma física. As ontologias *Timeline* e *Event* permitem descrever eventos musicais num contexto temporal estabelecendo uma relação entre as diferentes linhas temporais com os eventos e os seus agentes passivos ou ativos.

2.2.4 Studio Ontology Framework

A *Studio Ontology Framework* estende da *Music Ontology* e é mais dedicada para a partilha e gestão de informação sobre produção musical num estúdio de gravação. Facilita o registo de todos os detalhes inerentes da produção de uma música em estúdio fornecendo uma conceptualização explícita do ambiente de estúdio. A *Studio Ontology* consegue descrever cenários de gravação do mundo real

[9] http://performedmusicontology.org
[10] https://www.loc.gov/bibframe/

envolvendo hardware ou uma pós-produção em computador. Além de estender da *Music Ontology*, usa também as ontologias de tempo e de eventos assim como o mesmo modelo de produção musical.

2.3 Wikidata,Wikimedia and Mediawiki

O Mediawiki[11] é a tecnologia base do nosso projeto. É uma tecnologia gratuita preparada para instalar num servidor na internet e para receber milhões de acessos por dia. Foi desenvolvida pela Fundação Wikimedia[12] em 2002 para dar suporte à Wikipédia e ainda hoje é utilizado não só na Wikipédia[13] mas em todos os websites da Wikimedia. Usa a tecnologia PHP[14] para processar e apresentar os dados guardados na sua base de dados MySQL[15].

O Mediawiki foi optimizado para lidar eficientemente com projetos de grande escala na ordem dos terabytes e com centenas de milhares de acessos por segundo. Um dos principais fatores para sua utilização em milhares de projetos é a relativa facilidade da sua configuração, bastando um servidor remoto para hospedar o site e uma base de dados MySql configurada e *online*. O MediaWiki e as suas extensões contêm um tutorial de configuração que o utilizador tem que seguir na primeira vez que acede ao repositório. Após o tutorial, o repositório fica pronto a ser usado.

É uma tecnologia disponível em mais de 300 línguas, com mais de 900 definições de configuração diferentes e com a possibilidade de incorporar mais de 1900 extensões que permitem ativar novas funções ou fazer alterações em funções já existentes. Uma das extensões que vamos utilizar no nosso projeto será a Wikidata[16].

A Wikidata caracteriza-se essencialmente através de itens em que cada um dos quais descrito com um rótulo, uma descrição e por um ou mais nomes alternativos.As declarações descrevem as caraterísticas de um elemento através de uma propriedade e de um valor. Por exemplo, uma pessoa pode adicionar uma propriedade a indicar onde se educou e indicando o valor do local onde foi educado, em relação a um edifício, pode-se criar propriedades para as coordenadas geográficas, especificando os valores de longitude e latitude. No domínio do nosso projeto podemos definir uma propriedade “data de nascimento” que terá como valor a data de nascimento do músico em causa. Uma propriedade que estabeleça a ligação com uma base de dados externa é denominada um identificador.

[11] https://www.mediawiki.org/
[12] https://www.wikimedia.org
[13] https://www.wikipedia.org
[14] https://www.php.net
[15] https://www.mysql.com
[16] https://www.wikidata.org/

2.3.1 Mediawiki API

O MediaWiki quando é instalado, vem implementado com uma REST API[17] (definição no próximo subcapítulo) que permite gerir informação do repositório através de vários *endpoints*. Para o desenvolvimento da interface de visualização de dados, a API vai assumir um dos papéis mais importantes no sistema, sendo responsável por todo o tipo de extração de informação.

O lado do cliente faz um pedido de informação ao servidor através do protocolo HTTP e recebe a resposta num formato padrão, normalmente no formato JSON[18]. Um pedido consiste num *endpoint* e num conjunto de parâmetros que servirão para filtrar a informação. Existem dois tipos de pedido que podem ser feitos: GET e POST. Para uma pedido GET, um parâmetro seria uma *string* de consulta presente no URL. Para um pedido POST, os parâmetros são formatados no formato JSON e incluídos no pedido. Na *string* de consulta presente no URL, adiciona-se o parâmetro *action*. Este parâmetro indica à API qual a ação a ser executada. A ação mais popular é a de consulta (a URL tem de conter então *action* = *query*), que permite extrair dados de um sistema wiki. Em conjunto com o parâmetro de ação, um dos três seguintes parâmetros tem de ser adicionado:

- **prop**: obter propriedades das páginas
- **list**: obter lista de páginas que correspondem a um determinado critério
- **meta**: obter meta informações sobre o wiki e o utilizador

Por fim, inclui-se o parâmetro de formato, que informa a API em que formato pretendemos obter os resultados. O formato recomendado é JSON. A API ofereceu suporte a outros formatos de *output* no passado, mas não são recomendados.

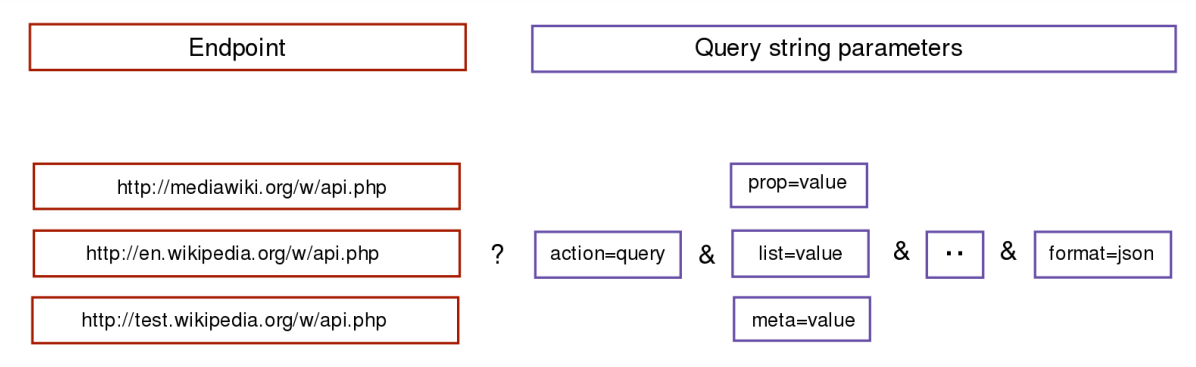


Figura 2.1: Formato do pedido

[17] https://www.mediawiki.org/wiki/API:Main$_{page}$
[18] https://www.json.org/json-en.html

2.3.2 Pedido HTTP: GET

Os pedidos HTTP com o método GET são usados maioritariamente para requisitar algum tipo de informação, ao invés do método POST, usado para acrescentar informação. Nos pedidos GET, os dados e critérios do pedido, se se aplicar, são anexados ao endereço URL.

Vai ser agora apresentado um exemplo mais concreto de uma consulta simples para entender como seria um pedido GET e consequente resposta, típicas no contexto da API da Mediawiki. No pedido seguinte vamos fazer uma pesquisa por um título num sistema wiki. Vai ser usado o módulo *search*[19] da API.

http://en.wikipedia.org/w/api.php?action=query&list=search&srsearch={*title*}&format=json

Explicação de cada parte do URL:

- **http://en.wikipedia.org/w/api.php**: representa o *endpoint* principal. Neste caso, é a Wikipedia em inglês.
- **action=query**: representa a extração de dados do repositório wiki.
- **list=search**: significa obter uma lista de páginas que correspondem a um critério.
- **srsearch={*title*}**: indica o título da página ou conteúdo a ser pesquisado.
- **format=json**: indica o formato de *output* como JSON, que é o formato recomendado.

Listagem 2.1: Exemplo parcial de uma resposta de um pedido GET à API da Wikipédia

```
1  {
2      "batchcomplete": "",
3      "continue": {
4          "sroffset": 10,
5          "continue": "-||"
6      },
7      "query": {
8          "searchinfo": {
9              "totalhits": 5060
10         },
11         "search": [
12             {
13                 "ns": 0,
```

[19] https://www.mediawiki.org/wiki/API:Search

```
                "title": "Nelson Mandela",
                "pageid": 21492751,
                "size": 196026,
                "wordcount": 23664,
                "snippet": "<span class=\"searchmatch\">Nelson</span> Rolihlahla",
                "timestamp": "2018-07-23T07:59:43Z"
            }
        ]
    }
}
```


2.4 Exemplos de projetos que usam a Mediawiki API

A organização responsável pela Mediawiki e no geral pelas tecnologias wiki disponibiliza um portal[20] com alguns projetos desenvolvidos e já estabilizados e que usam a API do Mediawiki para gestão da informação. A maioria dos projetos desenvolvidos tem uma arquitetura algo semelhante com a aplicação que foi desenvolvida neste projeto. Por ser um número já elevado de projetos apenas vão ser apresentados alguns.

2.4.1 *Wikifeedia - A news feed of featured articles*

A página inicial da Wikifeedia[21] consiste num *feed* de notícias de artigos em destaque. Os artigos em destaque são considerados os melhores artigos da Wikipédia. Ver 2.2.

2.4.2 *Sightseer Map - Features a map with interesting places to visit around Athens*

A aplicação[22] apresenta um mapa do centro de Atenas e marcadores nos principais locais. Ao clicar num marcador do mapa, são exibidas informações sobre o local. Estão também disponíveis uma lista dos locais e um filtro de pesquisa. Ao clicar num local na lista, são exibidas informações sobre o local e surge uma animação no seu marcador no mapa. Ver 2.3.

[20] https://apps-gallery.toolforge.org/
[21] https://wikifeedia.vishnuks.com/
[22] https://desinas.github.io/sightseer-map/

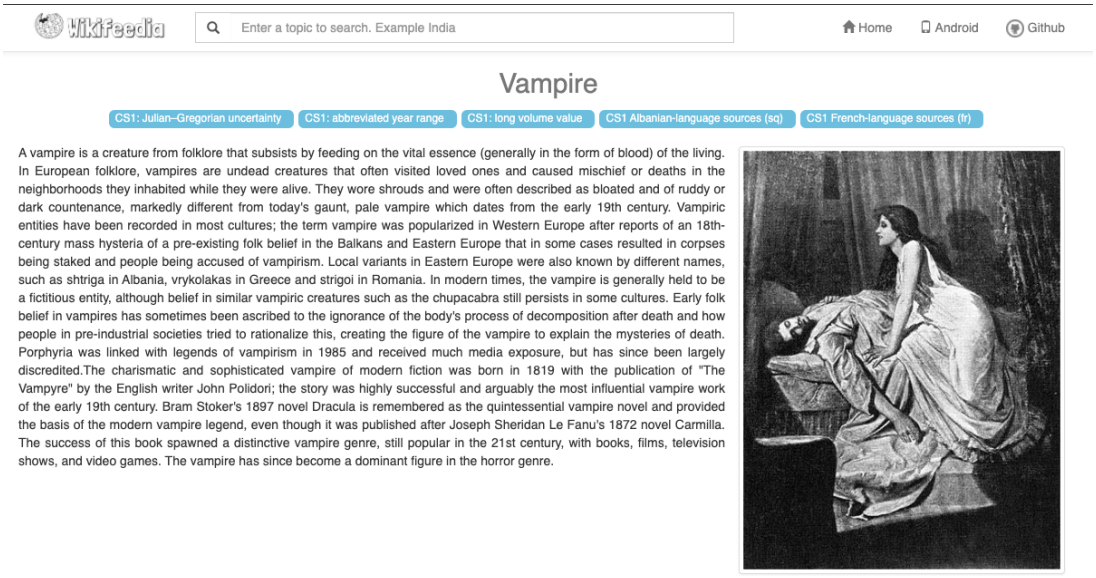


Figura 2.2: Página inicial do Wikifeedia

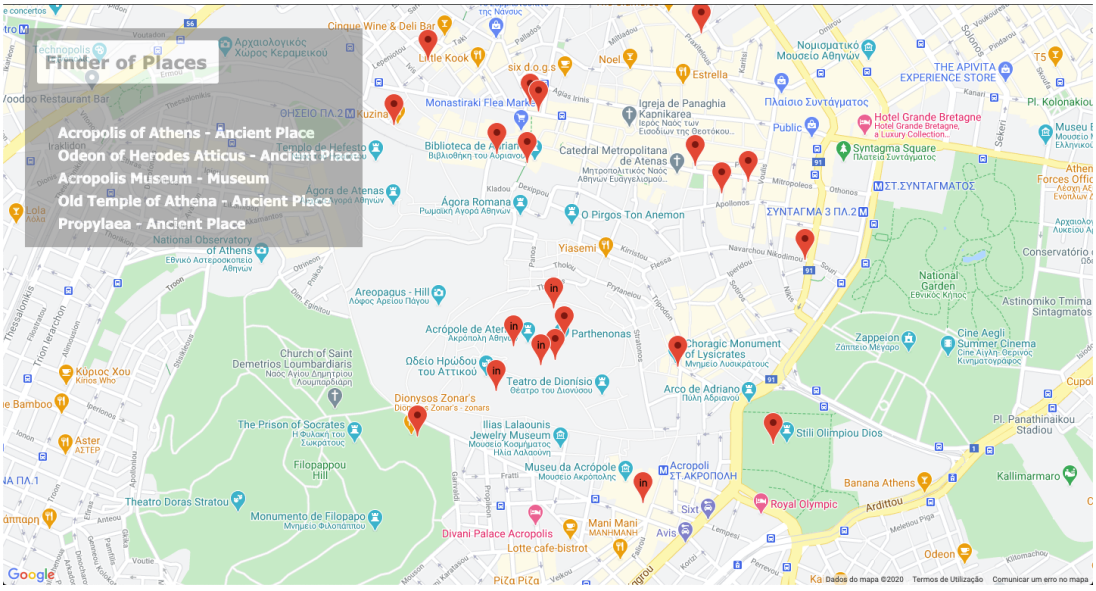


Figura 2.3: Página inicial do *Sightseer Map*

2.4.3 *User Contributions - Fetches top 50 edits made by a user*

Esta aplicação[23] usa a API para mostrar as ultimas 50 edições realizadas por um certo utilizador. Ver 2.4.

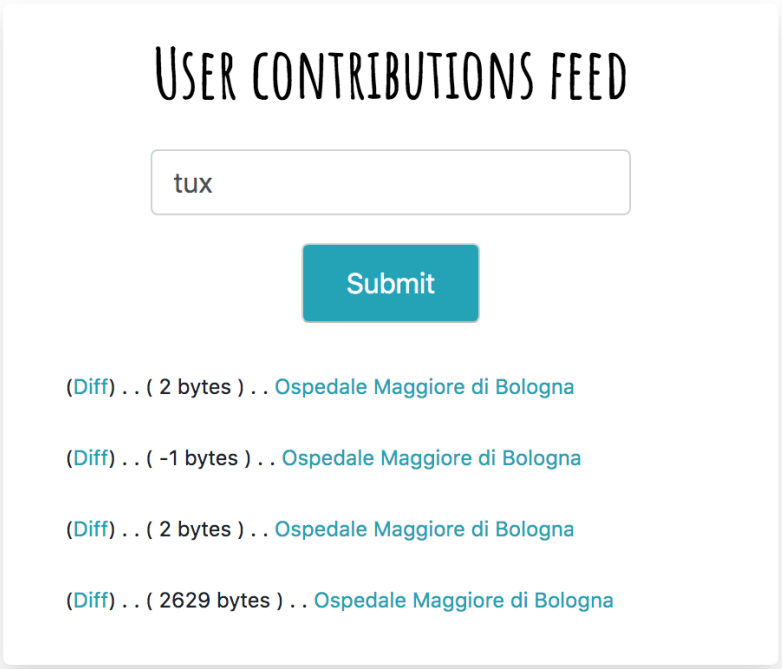


Figura 2.4: Página inicial do *User Contributions*

2.4.4 *MediaWiki Benchmarker*

Esta aplicação consegue fazer uma análise *benchmark* a uma instalação Mediawiki através da API. Usa gráficos de barra e diagramas de caixa e permite a exportação dos dados para um ficheiro CSV ou JSON. Os gráficos podem também ser descarregados no formato SVG. Esta aplicação torna-se bastante útil para repositórios em larga escala e com um número muito elevado de acessos, de forma a se poder avaliar a eficiência do sistema. Esta ferramenta poderá ser bastante útil na avaliação futura do nosso repositório, principalmente quando a quantidade de dados for significativa, assim como o número de utilizadores a utilizar as aplicações. Ver fig. 2.5.

[23] https://user-contributions-feed.toolforge.org/

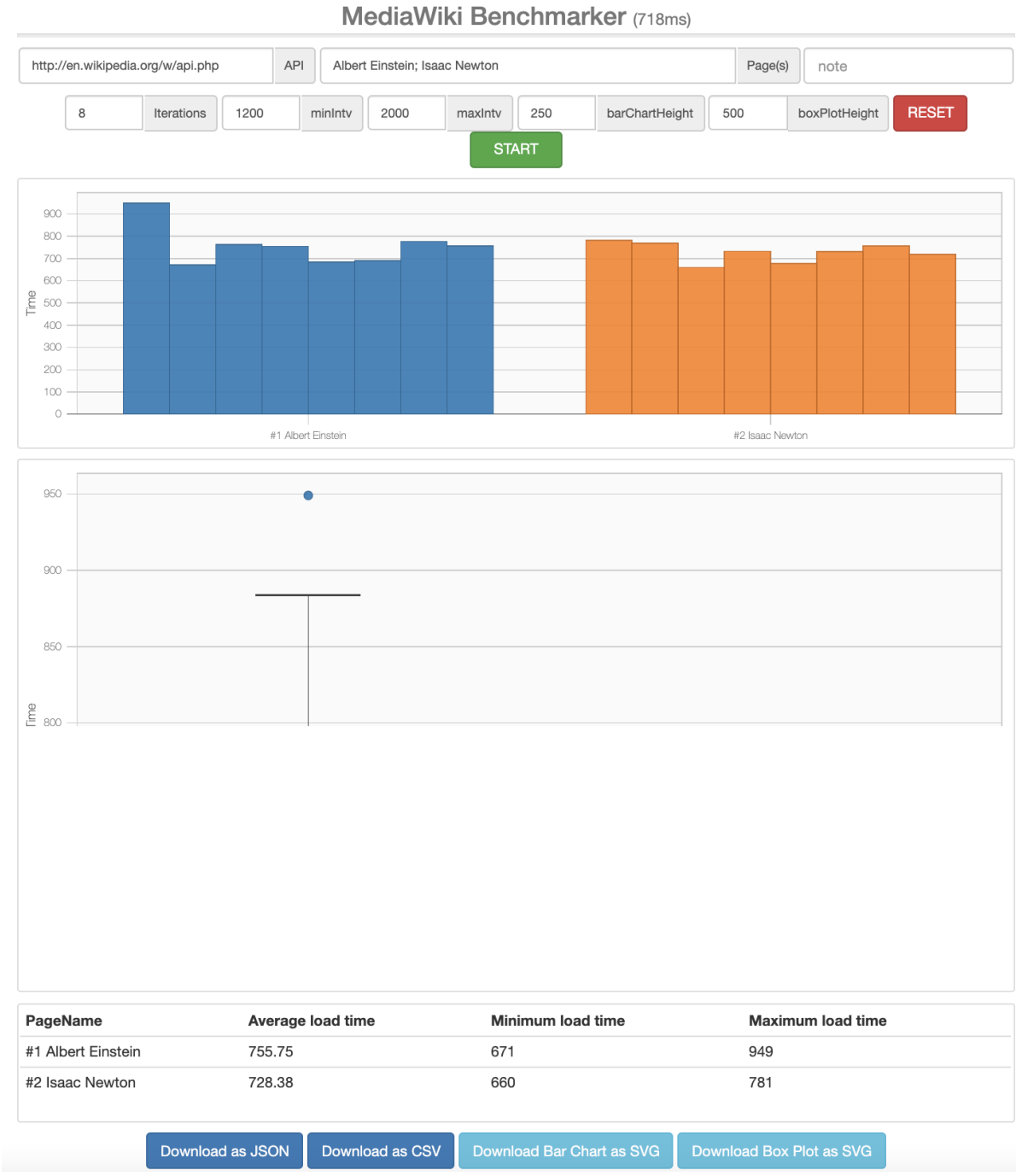


Figura 2.5: Mediawiki Benchmarker

2.5 Tecnologias de Desenvolvimento Web

A Web dos dias de hoje acabou com o antigo modelo de programação baseado inteiramente numa arquitetura *single-box*, em que basicamente o objetivo era a representação estática e única dos dados, não havendo muita capacidade para uma interação dinâmica com a informação.

Segundo Drobi [6], com o surgimento do *Software as a Service (SaaS)* e dos *Web Services*, a distribuição dos dados passou a ser a regra e não a excepção. A aplicação e a distribuição de dados levam a representações e formatos de dados heterogéneos. Normalmente, no código fonte de uma aplicação, há blocos de código exclusivamente dedicados à produção e ao consumo destes formatos, isto é, juntá-los e transformá-los num modelo que seja importante para uma determinada funcionalidade ou serviço. Um exemplo comum seria fazer pedidos para um qualquer *Web service* para obter informação num formato popular tal como XML ou JSON, pegar nesta informação e representá-la numa página HTML.

2.5.1 Arquiteturas de Software para a Web

A abordagem padrão na arquitetura *single-box* era formatar o modelo de armazenamento de dados ao formato do modelo de serviço. Esta abordagem foi a mais comum por permitir ter um controlo total sobre os dados e também por permitir poder armazená-los num só lugar.

Nos novos modelos de desenvolvimento Web, existem ferramentas e soluções que permitem a fácil manipulação e transformação dos dados.

A arquitetura *single-box* pressupõe que todos os componentes do sistema estão sempre disponíveis ou, se não estiverem, que não podemos fazer nada para gerir o erro resultante, excepto relatar ao utilizador a ocorrência de um erro interno e esperar que a componente em falha volte a ter um comportamento normal. Esta suposição não se pode aplicar no desenvolvimento da Web moderno. Na verdade, um *Web Service* pode ter vários sistemas distribuídos. Alguns são essenciais para o resultado final do *Web Service*, outros são opcionais e alguns estão disponíveis em vários servidores (através da replicação). Continuar a enviar um erro interno do servidor para o utilizador do quando qualquer um desses serviços estiver em falha tornou-se irracional. Em vez disso, devemos ser mais explícitos ao lidar com os sinais que os diferentes serviços emitem. Consideremos, por exemplo, um pedido HTTP: pode devolver um ”200 OK”como resposta mas também pode devolver um ”400 Not Found”(indicando que o recurso que foi pedido não existe), um ”400 Bad Request”ou qualquer outro *status* HTTP que ajude o *Web Service* a lidar com a situação. Dependendo das circunstâncias, o sistema pode tentar novamente, deduzir o erro relacionado e devolvê-lo ao utilizador, ou até mesmo ignorar o erro.

2.5.2 Modelo *Model-View-Controller (MVC)*

A arquitetura MVC foi inicialmente desenvolvida para interfaces de utilizador em aplicações implementadas com a linguagem de programação *Smalltalk*[24]. Nesta abordagem, o sistema é dividido em três componentes:

- **modelo**: expressa o conhecimento do domínio (*Model*)
- **visualização**: interface do utilizador (*View*)
- **controlador**: responsável por atualizar a interface do utilizador (*Controller*)

O modelo MVC tem demonstrado os seus benefícios para aplicações interativas permitindo múltiplas representações da mesma informação, promovendo a reutilização de código e ajudando os programadores a focarem-se num único aspeto da aplicação.

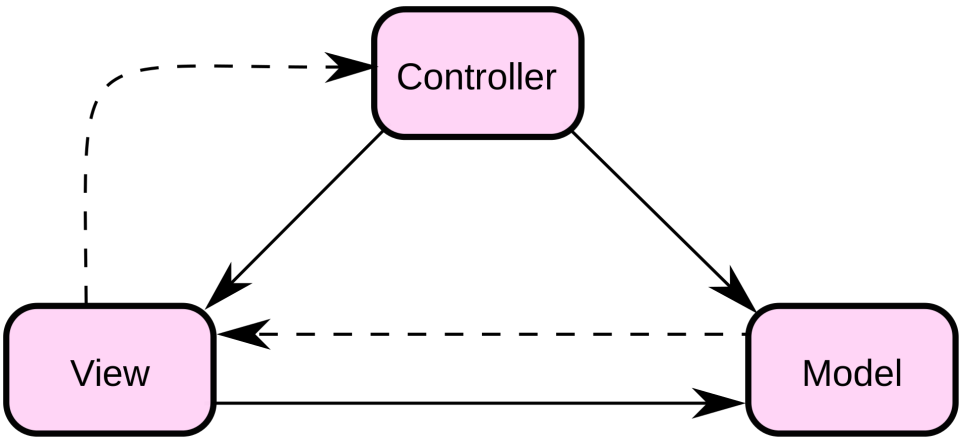


Figura 2.6: Diagrama do modelo MVC

Cada uma das três componentes é responsável por uma tarefa. O modelo contém os dados da aplicação e faz a gestão do seu *core* principal. A visualização faz a gestão da exibição visual do modelo e do feedback dado ao utilizador. O controlador interpreta os *inputs* do utilizador e é responsável por transmitir estes *inputs* aos outros dois componentes de modo a que estes possam reagir apropriadamente. No sub capítulo seguinte vamos analisar a *framework* Django, que segue precisamente esta arquitetura, e é a tecnologia usada para desenvolver a nossa aplicação.

2.5.3 Django: *Framework* de desenvolvimento Web

Django é uma framework para desenvolvimento web através da linguagem de programação Python e foi criado para o desenvolvimento rápido de sites baseados em bases de dados.

[24] https://squeak.org

Como referido na secção anterior, ”dentro”do Django está o estilo MVC, um conjunto de bibliotecas *opensource* e programadas em Python. O lema desta tecnologia é ”Don't repeat yourself”, que traduzindo significa, ”Não te repitas”. Assim como o Python, o Django foca-se na ineficiência, permitindo que o programador faça quase todas as tarefas com o mínimo de esforço em termos de código. Além das vantagens já referidas, o Django também é escalável, robusto e rápido tendo uma grande comunidade de *developers* e um conjunto robusto de componentes integrados. Tanto pode aceder ou criar ficheiros de dados JSON e/ou XML e lidar com sistemas de base de dados relacionais tais como Oracle, MySQL, SQLite e PostgreSQL. [7]

Num site tradicional, uma aplicação web espera por pedidos HTTP do *browser* ou de outro ”cliente”. Quando um pedido é recebido, a aplicação calcula o que é necessário executar com base no URL e, possivelmente, nas informações presentes nos dados POST ou GET. Dependendo do que for necessário, a aplicação pode ler ou gravar informações de uma base de dados ou executar outras tarefas para responder ao pedido. A aplicação vai então devolver uma resposta ao cliente, normalmente criando dinamicamente uma página HTML para o *browser* exibir, inserindo os dados gerados nos espaços certos da página. As aplicações Django normalmente agrupam o código que lida com cada uma das tarefas em ficheiros separados. [8]

O Django tem incluído uma aplicação própria de administração que usa os modelos implementados para construir automaticamente uma espécie de área de testes da nossa aplicação para que se possa criar, visualizar, atualizar e apagar registos. Esta aplicação permite poupar muito tempo durante o desenvolvimento, tornando muito mais fácil e prático testar os modelos em desenvolvimento. A aplicação de administração também pode ser útil para gerir dados em produção, dependendo do tipo de site. O projeto Django recomenda apenas para gestão de dados internos, ou seja, apenas para uso de administradores ou pessoas internas ao projeto, já que a abordagem centrada no modelo não é necessariamente a melhor interface possível para todos os utilizadores e expõe muitos detalhes desnecessários sobre os modelos.

- **URLs**: Apesar de ser possível processar pedidos de cada URL numa única função python, é muito mais eficiente e menos complexo escrever uma função *view* (vista) para processar cada pedido correspondente a cada URL. O mapeamento de URLs é usado para redirecionar pedidos HTTP para a *view* apropriada com base no URL do pedido. O mapeamento de URLs também pode fazer corresponder padrões específicos de *strings* ou dígitos que aparecem num URL e transmiti-los a uma função *view* como dados.

- **View (Vista)**: As vistas são responsáveis por tratar dos pedidos, recebendo pedidos HTTP e devolvendo respostas HTTP. As vistas acedem aos dados necessários de modo a satisfazer o pedido através dos *models* (modelos) e encarregam os *templates* da formatação da resposta.

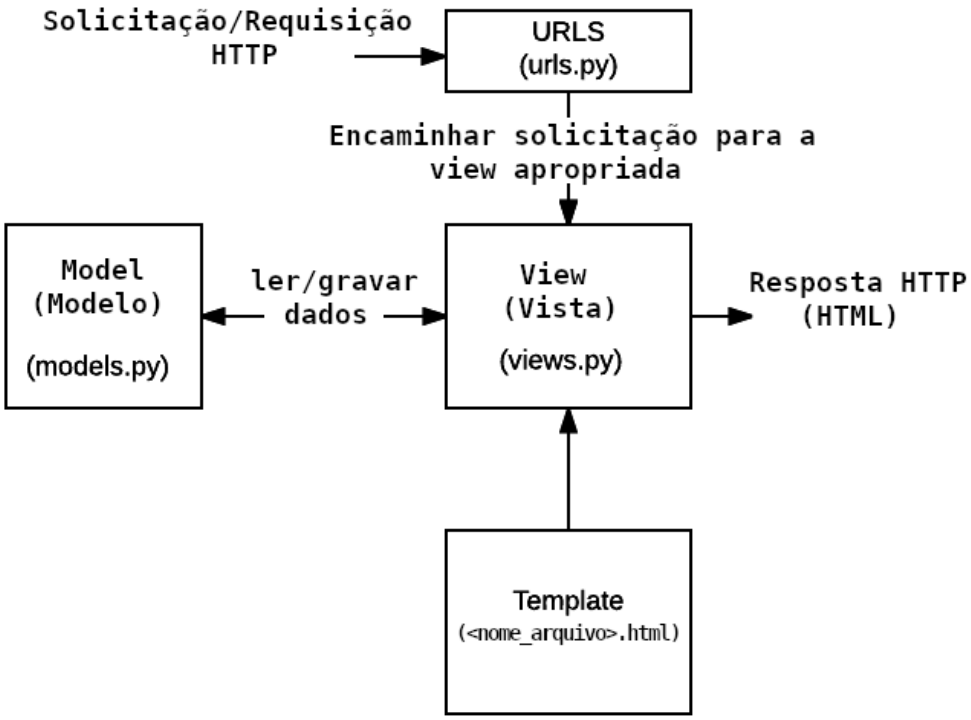


Figura 2.7: Arquitetura de uma aplicação Django

- **Models (Modelos)**: Os modelos são objetos python que definem a estrutura dos dados de uma aplicação, e que fornecem mecanismos para fazer a gestão da informação presente na base de dados (adicionar, editar e eliminar).

- **Templates**: Um *template* é um ficheiro de texto que define a estrutura ou o *layout* de um ficheiro como por exemplo de uma página HTML, com espaços reservados usados para representar a informação real. Uma *view* pode criar dinamicamente uma página HTML usando um template HTML e preenchendo-a com dados de um *model*(modelo).

2.5.4 REST API

Nos últimos anos, uma REST API tornou-se uma arquitetura comum, já que o seu conceito é proporcionar um método simples de aceder a *web services*. [9] Hoje, uma quantidade substancial de dados na web é trocada através de APIs da Web que expõem dados em formatos convenientes, como JSON ou XML. Quase todos os 100 melhores sites da Alexa[25] fornecem as suas próprias APIs, o que é um indicador da preferência dos *web services* para a comunicação com outras componentes externas. Na maioria dos casos, as respostas estruturadas destas APIs seguem um formato sintático comum, con-

[25]https://www.alexa.com/topsites

forme é característico do estilo da arquitetura REST. [10] As restrições do estilo de arquitetura REST afetam as seguintes propriedades:

- desempenho nas interações de componentes, que pode ser o fator dominante no desempenho percebido pelo utilizador e na eficiência da rede;
- escalabilidade permitindo suportar um grande número de componentes e de interações entre componentes. Roy Fielding [11] descreve o efeito da REST na escalabilidade como: A separação de interesses cliente-servidor da REST simplifica a implementação dos componentes, reduz a complexidade da semântica da ligação, melhora a eficácia do ajuste de desempenho e aumenta a escalabilidade dos componentes do servidor. As restrições do sistema em camadas permitem que as camadas intermédias tais como proxies, gateways e firewalls, sejam introduzidos em vários pontos da comunicação sem alterar as interfaces entre os componentes, permitindo-lhes auxiliar na tradução da comunicação ou melhorar o desempenho através de uma cache partilhada em grande escala. O REST permite o processamento intermédio ao restringir as mensagens a serem auto descritivas: a interação é *stateless* entre os pedidos, métodos padrão e são usados alguns tipos de media para indicar a semântica e trocar informações e as respostas indicam explicitamente a capacidade de armazenamento em cache;
- simplicidade de uma interface uniforme;
- capacidade de modificação dos componentes para atender à necessidade de uma mudança (mesmo quando a aplicação está em execução);
- visibilidade da comunicação entre componentes por agentes de serviço;
- portabilidade dos componentes movendo o código do programa com os dados;
- confiabilidade na resistência a falhas no nível do sistema na presença de falhas em componentes, ligações ou dados.

2.5.5 Aplicação de Gestão dos dados

Como referido no capítulo anterior, a primeira aplicação desenvolvida neste projeto focou-se na gestão da informação e da ontologia (Mediawiki). Sendo uma aplicação desenvolvida no âmbito do mesmo projeto, houve uma óbvia colaboração entre mim e o outro aluno envolvido, que também desenvolveu a sua dissertação final de mestrado. Esta aplicação visou corresponder aos seguintes casos de uso:

- Criar conta e iniciar sessão: Um utilizador cria uma conta, inicia sessão e tem acesso aos objetos no sistema. Neste ponto, apenas algumas das funcionalidades estarão disponíveis. O utilizador terá permissões de escrita até que a sua permissão seja alterada por um administrador.

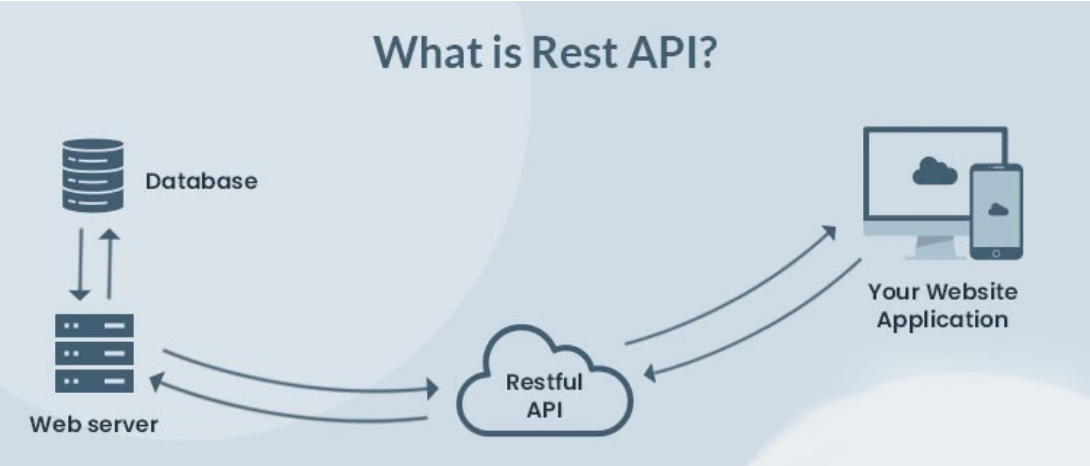


Figura 2.8: Arquitetura com REST API como componente intermediária [1]

- Importar conjunto de dados: Um utilizador importa um ficheiro com dados. Os dados são organizados em colunas e linhas de um ficheiro de folhas de cálculo, onde cada linha representa um objeto e cada coluna representa os valores de uma propriedade. Isso pode levar, ou não, à criação de um formulário normalizado.
- Exportar conjunto de dados: Um utilizador seleciona um conjunto de objetos para exportar e o sistema cria um ficheiro que pode ser descarregado com os objetos selecionados e as propriedades que contêm valores para esses objetos.
- Criar formulário normalizado: Para facilitar a importação de dados para a base de dados, tanto para o próprio utilizador como para terceiros, é criado um formulário normalizado importando um ficheiro que vai servir de *template* do formulário a ser utilizado. Ele será guardado no sistema e poderá ser exportado a qualquer momento por utilizadores com permissão para lhe aceder.
- Criar vista: Um utilizador pretende criar uma coleção de objetos, chamada de vista, que pode ser gerida e publicada. Assim, o utilizador seleciona os objetos ou carrega um ficheiro com os objetos que pertencerão à vista, dá um nome, uma descrição e define a sua privacidade. Uma nova vista será criada com a sua própria página web.
- Exportar vista: Um utilizador pode exportar os itens de uma vista para um ficheiro Excel. Este ficheiro pode então ser usado para editar a vista ou gerar uma nova.
- Gerir itens: Quando os itens são guardados na base de dados, eles podem ser geridos por utilizadores com permissão para tal. Esta gestão inclui:
  - Apagar itens do sistema, selecionando da lista de itens ou importando um arquivo com os QIDs dos itens a serem excluídos

- Recuperar itens da lista de itens excluídos. A Wikibase não permite que itens sejam verdadeiramente removidos do sistema. Portanto, por exemplo, se um item foi excluído por engano, ele pode ser facilmente recuperado mais tarde.
- Editar os itens já importados para o sistema. As alterações feitas no sistema são feitas usando a interface fornecida pelo MediaWiki.
- Gerir permissões: Para alterar as permissões de um utilizador específico, um administrador pode escolher o conjunto de permissões mais adequado que o utilizador deve ter.

3

Análise do Problema e Solução Funcional

Conteúdo

Podemos agora apresentar a metodologia escolhida para o desenvolvimento deste projeto. Primeiro que tudo, o projeto pode ser dividido em três fases principais:

- Recolha de informação e construção da ontologia, através da tecnologia *Wikidata*;
- Desenvolvimento de uma aplicação Web para popular e gerir a informação da ontologia;
- Estabilização da aplicação de gestão dos dados e desenvolvimento da aplicação Web para gestão da visualização da informação;

Como já foi explicado nos capítulos anteriores, esta tese vai-se focar essencialmente no terceiro passo (Estabilização da aplicação de gestão dos dados e desenvolvimento da aplicação Web para gestão da visualização da informação).

3.1 Requisitos

A aplicação vai ser maioritariamente usada por utilizadores com poucas habilidades técnicas, por isso, todas as operações têm que ter um nível baixo de dificuldade e complexidade para serem executadas.

Em conversa com os investigadores e coordenadores do projeto PROFMUS, tentámos perceber e estabelecer as funcionalidades essenciais de desenvolver. Os investigadores focaram-se muito em poderem comparar estatisticamente os dados de cada objeto. Atendendo que a aplicação vai trabalhar em ”cima”das vistas, este aspeto da comparação tornou-se essencial.

Como os objetos contêm diversos tipos de dados, desde coordenadas geográficas, datas ou simples *strings*, os elementos de visualização das vistas teriam que se adaptar a estes formatos. Assim sendo, optou-se pela inclusão de um mapa onde as coordenadas geográficas pudessem ser projetadas, uma linha temporal onde as datas pudessem também ser mostradas (num formato de ordem da data mais antiga para a mais recente), uma tabela com uma visualização geral de todos os objetos e respetivas propriedades e ainda a possibilidade de consultar todas as propriedades de um objeto com os valores bem destacados.

Também foi bem determinado que todos estes elementos teriam que ser o mais flexíveis e configuráveis possível. Deste modo, teriam que ser implementadas funcionalidades onde os utilizadores pudessem configurar cada elemento ao seu critério. Esta configuração contempla desde a escolha dos objetos a projetar no mapa e linha temporal, as cores associadas a cada objeto no mapa, a ordem de mostragem dos elementos referidos, entre outros.

3.2 Restrições

A aplicação vai extrair toda a informação através da API da *Mediawiki*, o que representa algumas dificuldades já que a documentação referente não é a mais completa e algumas operações teoricamente simples representam soluções mais complexas. A solução mais eficaz de extração de informação seria através de um sistema de execução de *queries* diretamente no nosso repositório, algo que é possível, por exemplo, no repositório que dá suporte à Wikipédia[1], mas que para configurar num repositório local como o nosso não há documentação concreta de como o fazer. Assim sendo, estamos limitados às capacidades e às operações disponíveis pela API. Para as operações a que a API não consegue corresponder, tiveram que ser desenvolvidas soluções alternativas e mais complexas, naturalmente mais lentas e menos eficientes. A extração e visualização de grandes conjuntos de dados pode então tornar-se um pouco lento.

3.3 Solução Conceptual

Uma vista é um sub conjunto de todos os objetos existentes no sistema, em que cada objeto é constituído por um conjunto de propriedades e dos respetivos valores. Por exemplo, podemos considerar como propriedade ”Data de Nascimento”e o seu respetivo valor uma data que represente a data de nascimento do objeto correspondente.

As vistas podem ser criadas e geridas pelos utilizadores do sistema, e podem posteriormente ser publicadas significando que cada vista terá o seu site exclusivo, podendo este ser partilhado com quem se quiser. Por exemplo, se um utilizador pretende um site com todos os músicos italianos que chegaram a Portugal num determinado período de tempo, o utilizador pode criar uma vista com todos os itens que correspondam a esse critério e depois publicá-la.

Quando o ”dono”(administrador) de uma vista decide publicá-la, pode definir um conjunto de opções quanto à sua apresentação estética no site, seleccionando um vasto leque conjunto de opções que no fim resultam no ”site oficial”da vista. A aplicação desenvolvida visa precisamente servir a configuração da publicação da vista, definindo uma interface visual de visualização da mesma.

A solução escolhida e idealizada teria que corresponder aos requisitos indicados pelos utilizadores, implementando os casos de uso identificados. Considerando que a total configurabilidade da aplicação era uma necessidade fulcral, toda a solução foi desenvolvida em redor deste aspecto.

Como referido nos capítulos anteriores, os dados são guardados num repositório *Mediawiki* e todas as operações e interações com os dados são executadas através da API embutida neste sistema. A criação das ”vistas”, para posterior publicação, é feita numa outra aplicação também desenvolvida no âmbito deste projeto. É também nesta aplicação de gestão dos dados que é feita a ”ligação”à aplicação

[1] https://www.wikipedia.org

de visualização das vistas. Atendendo ao nível de habilidades técnicas dos utilizadores, este processo tem de ser o mais simples e acessível possível.

3.4 Casos de estudo

Com os requisitos especificados na secção 3.1, foram definidos os seguintes casos de uso relevantes para posterior implementação na aplicação. Todos os casos de uso são focados nos elementos de visualização de dados e na sua flexibilidade. Todos os casos de estudo estarão ao acesso dos administradores dos seus sub conjuntos de dados (Vistas).

- Configurar nomes, títulos e sub títulos: O utilizador administrador da vista pode configurar o titulo desta, o sub titulo e ainda os títulos de cada uma das secções de informação (Informações, Tabela, Mapa e Gráfico Temporal).
- Escolher as propriedades a serem visualizadas na tabela: Uma das secções da aplicação é uma tabela onde as colunas são as várias propriedades e em que cada linha representa os valores das propriedades relativos a cada objeto. O administrador tem a possibilidade de definir que propriedades pretende ver na tabela.
- Configurar a ordem de mostragem da informação: O administrador da vista pode definir a ordem com que os elementos visuais podem aparecer. Por exemplo, pode definir que a tabela deve aparecer primeiro que o gráfico temporal e primeiro que o mapa.
- Escolher as propriedades geográficas a serem projetadas num mapa: De todas as propriedades com condições de serem projetadas no mapa, o administrador pode escolher as que quer que sejam visíveis no mapa.
- Escolher os objetos a serem projetados no mapa: De todos os objetos com propriedades que possam ser projetados no mapa, esta definiçao permite escolher quais aqueles que efetivamente vão ser mostrados.
- Escolher a cor do marcador geográfico correspondente a cada propriedade geográfica de um determinado objeto : Das propriedades que o administrador escolheu projetar no mapa, pode para cada objeto, definir um marcador com cor diferente.
- Escolher os objetos a serem mostrados na linha temporal: Esta definição permite escolher os objetos cujo as datas relacionadas se queiram consultar na linha temporal.

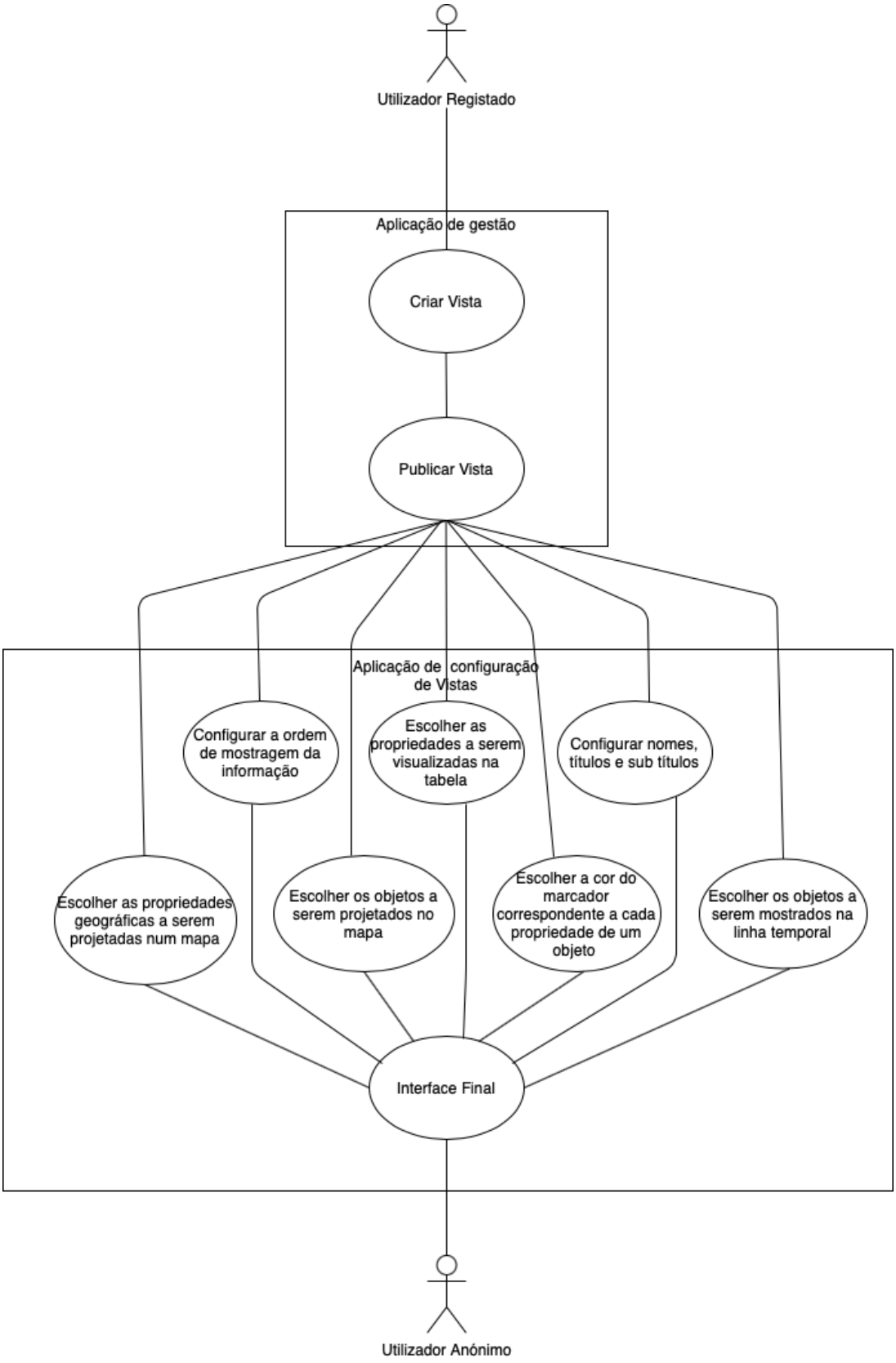


Figura 3.1: Ordem de execução dos casos de estudo

4

Arquitetura e Implementação

Conteúdo

4.1 Arquitetura

Na nossa arquitetura, o *Object Controller* (controlador de objetos) é feita por uma implementação do *Mediawiki* que vai gerir e dar suporte ao *Object Repository* (repositório de objetos). Todo o tipo de informações adicionais como os elementos das vistas ou as definições estéticas de cada vista serão guardadas numa base de dados *SQLite*. Estes elementos são coordenados pelo *Application Controller* (controlador da aplicação). Este controlador consiste numa aplicação Django que se liga à Mediawiki via protocolo HTTP e à base de dados através de uma API interna. O controlador da aplicação é também responsável pela interface mostrada ao utilizador através de um servidor web Apache[1].

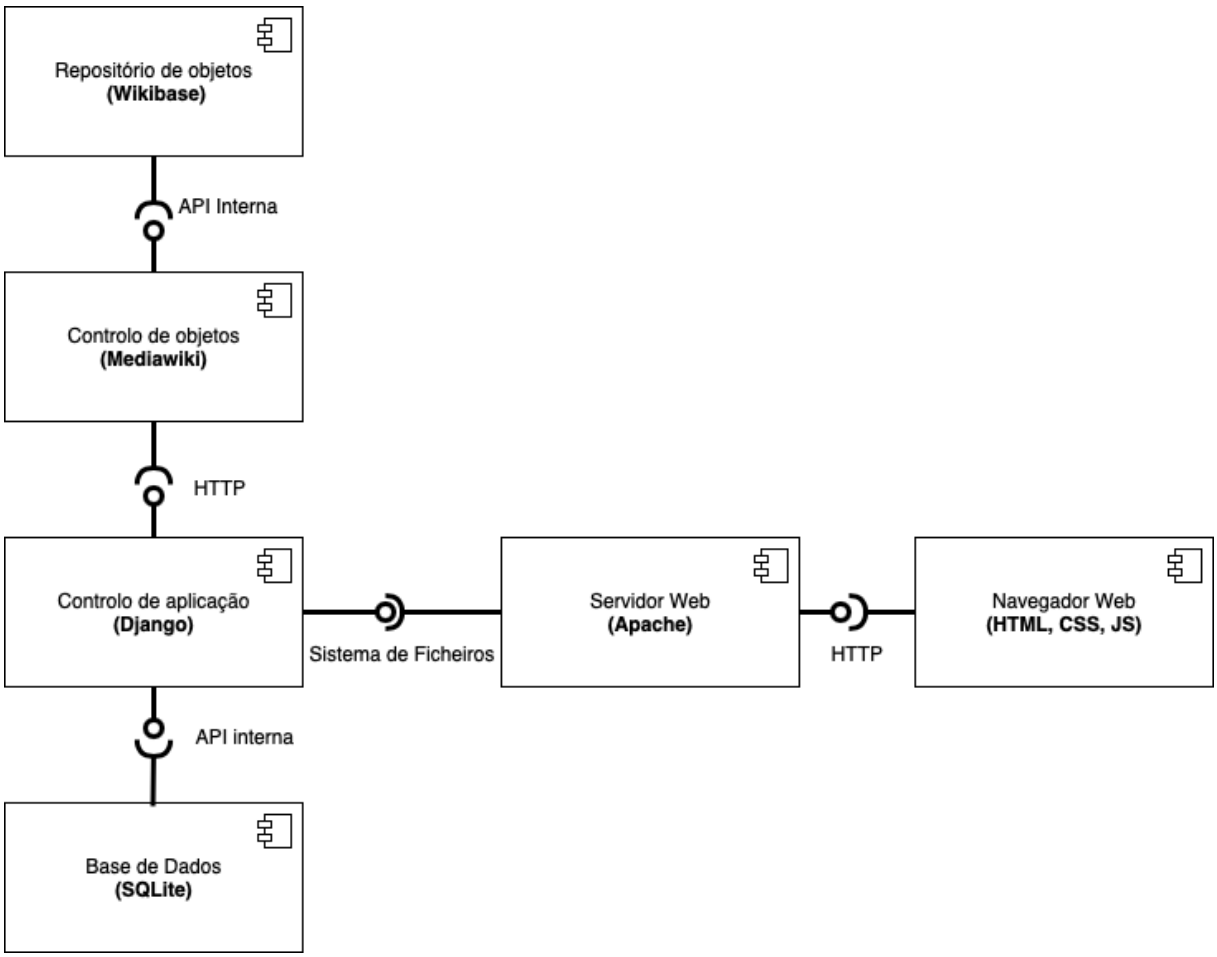


Figura 4.1: Arquitetura conceptual da aplicação

[1] https://www.apache.org

4.1.1 Controlador da Aplicação

O controlador da aplicação contém toda a lógica da aplicação tanto para o *front-end* como para o *back-end*. Contém também a interface gráfica com a qual os utilizadores vão interagir. Esta interface é gerada dinamicamente recorrendo às informações guardadas na base de dados e recorrendo ao repositório da Mediawiki para extrair todas as informações acerca dos objetos das vistas. Estas informações são depois inseridas nos ficheiros HTML através de uma *syntax* específica da *framework* Django.

4.1.2 Base de dados

A base de dados *SQLite* está ligada à *Application Controller* através de uma API interna e guarda todas as informações acerca das vistas, desde os objetos de cada vista, até às configurações estéticas da visualização de cada uma. No Django, um modelo é uma classe Python onde cada coluna da base de dados é representada como um atributo da classe. Optou-se pela tecnologia *SQLite* pois é uma base de dados leve e representada através de um único ficheiro, o que permite por exemplo que possa ser movida com facilidade. Apesar de leve, atende às necessidades do projeto, principalmente em termos de mobilidade.

4.1.3 Controlador de Objetos e Repositório de Objetos

O controlador de objetos representa uma implementação Mediawiki e é onde estão guardados todas as propriedades e valores de todos os objetos. Permite também fazer uma gestão dos dados, mas de um modo mais complexo e ”bruto”na perspetiva do utilizador. Deste modo, este tipo de gestão será mais adequado a um administrador do sistema e não tanto ao utilizador comum. O prinicipal modo de interação com o repositório de objetos é através da API do Mediawiki que fornece um conjunto de *endpoints* para filtrar a informação extraída. Uma implementação deste tipo (Mediawiki) requer uma base de dados MySQL onde no momento da configuração inicial do repositório, são criadas as várias tabelas necessárias para posteriormente ser guardada a informação importada.

4.2 Implementação

Nesta secção vai ser descrito todo o processo de implementação da aplicação. As principais componentes a serem abordadas serão a configuração das vistas e a construção da interface final de visualização das vistas.

4.2.1 Configuração das vistas

Quando na aplicação de gestão dos dados um utilizador cria uma vista, fica automaticamente como administrador da mesma. Ao ser administrador, permite-lhe fazer a publicação da vista. Ver fig.4.2

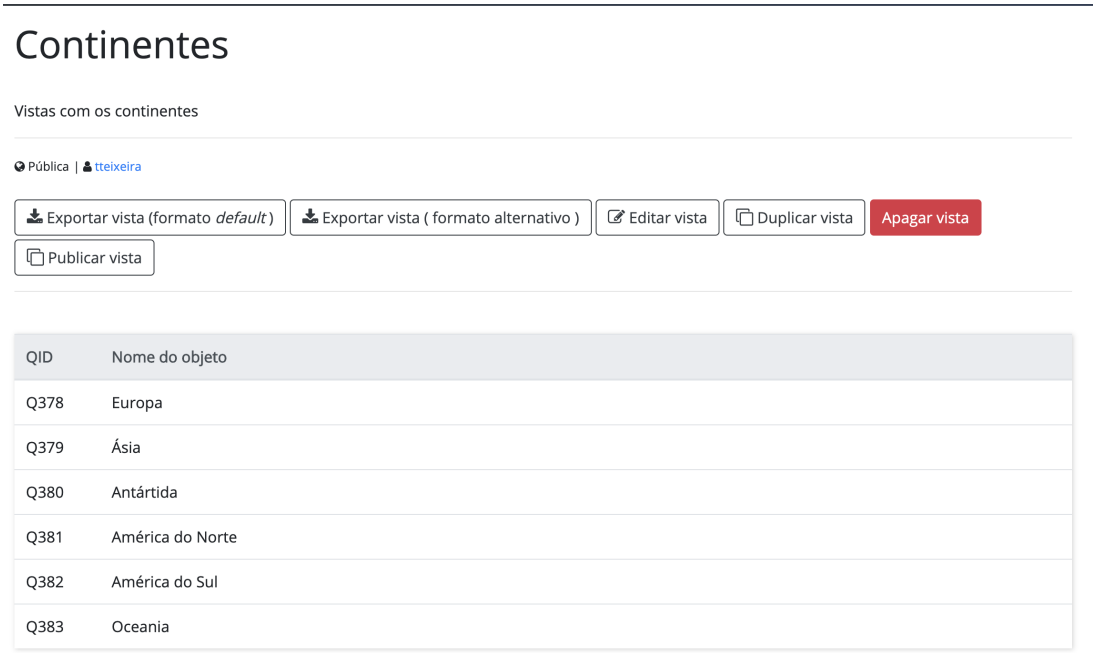


Figura 4.2: Opção de publicação da vista na aplicação de gestão dos dados

Quando é feita a publicação da vista por parte do administrador, este pode definir um conjunto vasto de opções visuais. Para guardar todas as informações acerca do estilo da vista, foi criada uma tabela nova na base de dados chamada *ViewStyle* em que cada coluna representa um campo estético da visualização da vista. Como referido anteriormente, na arquitetura Django cada tabela na base de dados é representada através uma classe *Model* em que os seus atributos representam as colunas da tabela na base de dados . Após a publicação, o administrador é ”confrontado”com uma série de escolhas que pode fazer de modo a configurar a vista. Pode imediatamente configurar os títulos e sub títulos tanto da vista como de cada secção da página. No fundo da página, o administrador tem acesso a um conjunto de definições adicionais tais como:

- Cores do *header*/*footer*
- Escolher as propriedades da tabela
- Escolher as propriedades no mapa
- Escolher as cores de cada propriedade presente no mapa
- Escolher as cores de cada objeto projetado no mapa

- Escolher a ordem de mostragem de informação

Listagem 4.1: Código Python referente ao *Model* que representa a tabela *ViewStyle*

```
class ViewStyle(models.Model):
    vista = models.ForeignKey(Vista, on_delete=models.CASCADE,null=True)
    titulo=models.CharField(max_length=100, null=True)
    subtitulo=models.CharField(max_length=100, null=True)
    font_family = models.CharField(max_length=100)
    header_color = models.CharField(max_length=100, default="#2c3e50")
    primary_color = models.CharField(max_length=100, default="#1abc9c")
    nome_informacoes=models.CharField(max_length=100, default="Informacoes")
    nome_tabela=models.CharField(max_length=100, default="Tabela")
    nome_mapa=models.CharField(max_length=100, default="Mapa")
    nome_timeline=models.CharField(max_length=100, default="Linha Temporal")
    ordem_informacoes=models.IntegerField(default=1)
    ordem_tabela=models.IntegerField(default=2)
    ordem_mapa=models.IntegerField(default=3)
    ordem_timeline=models.IntegerField(default=4)
    propriedades_tabela=models.CharField(max_length=100, null=True)
    propriedades_mapa=models.CharField(max_length=100, null=True)
    marcadores_mapa=models.CharField(max_length=100, null=True)
    ultima_edicao = models.DateTimeField(null=True)
```

4.2.2 Construção da Interface

Para a construção da interface final de visualização das vistas, o *Application Controller* extrai da base de dados todas as definições guardadas pelo administrador, e aplica-as na página HTML base, que serve como base para todas as visualizações. Por ser uma framework Web, o Django precisa de uma maneira conveniente de gerar código HTML dinamicamente. A abordagem mais comum depende dos *templates*. Um *template* contém as partes estáticas do *output* HTML desejado, bem como alguma sintaxe especial que descreve como vai ser inserido o conteúdo dinâmico. Um *template* Django é um documento de texto ou uma *string* Python marcada usando a linguagem de *template* Django. Algumas construções são reconhecidas e interpretadas pelo mecanismo dos *templates*. Os principais são variáveis e *tags*. Um modelo é renderizado com um *context*, no formato de um dicionário Python, que contém as informações que vão substituir as variáveis presentes no *template* pelos valores devidos.

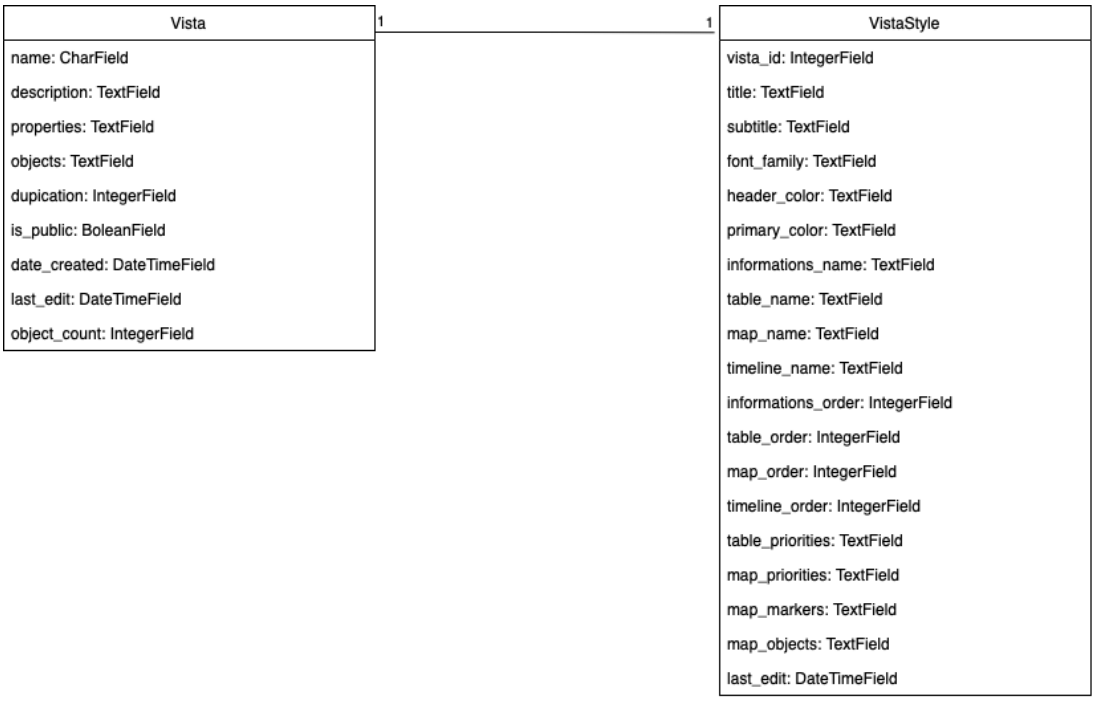


Figura 4.3: Modelo de Domínio

O primeiro passo feito pela aplicação para a construção da interface, começa por extrair da base de dados um *array* com os id's dos objetos pertencentes à vista em causa. Após a extração dos id's, a aplicação vai fazer um pedido HTTP à API da Mediawiki e recebe uma resposta no formato JSON que contém todas as propriedades e os respetivos valores, assim como possíveis qualificadores caso existam, para todos os objetos. O passo seguinte passa por iterar a resposta JSON e para cada objeto ir guardando em dicionários Python os valores de cada propriedade principal e caso hajam qualificadores, também são guardados do mesmo modo. No processo de criação dos dicionários, há uma filtragem especial caso os dados sejam uma data ou uma coordenada geográfica. Para estes dois tipos de dados, o conjunto $<Propriedade>:<Valor>$ é guardado numa estrutura à parte para que posteriormente, no *template*, estes dois dicionários sejam projetados em estruturas específicas como são um mapa e um gráfico temporal. Um dos elementos da interface é uma tabela onde por *default* são mostradas todas as propriedades principais (o administrador pode depois alterar estas propriedades escolhendo as que pretende). Ao clicar numa das linhas da tabela, são apresentadas todas as propriedades e qualificadores com os seus respetivos valores.

4.2.3 Integração MediaWiki e Wikibase

Para comunicar com a API do Mediawiki usámos maioritariamente a biblioteca Python Wikibase-api[2]. Por padrão esta biblioteca faz pedidos para o *endpoint* da Wikidata mas permite-nos indicar o URL do nosso repositório como padrão. Para cada tipo de informação, existe um formato específico nos sistemas Wikibase. Através da documentação oficial, foi possível perceber quais são os formatos que o sistema Wikibase suporta, principalmente nos tipos de dados mais abstractos tais como datas e coordenadas geográficas. As *strings* comuns não precisam de nenhum tratamento especial. Eis algumas das especificidades tidas em conta:

- As datas devem ser interpretadas no formato ”yyyy-mm-dd”. Se o valor do dia e mês não for conhecido, estes são substituídos por zero. Significa que uma data pode ser ”yyyy-mm-dd”, ”yyyy-mm-00” e ”yyyy-00-00”. Cada um tem uma precisão associada no objeto Wikibase.

- As coordenadas geográficas estão guardadas no formato ”latitude,longitude,precisão” onde cada valor é um decimal. A latitude e a longitude são expressas em graus decimais e os valores devem variar entre -360 e 360. O valor da precisão indica a precisão dos valores importados (latitude e longitude) e pode ser representado pelos seguintes valores:

  - 0.000001
  - 0.00001
  - 0.0001
  - 0.001
  - 0.01
  - 0.1
  - 1
  - 10

4.3 Produção

Nesta secção, será descrito o processo envolvido na configuração *online* de cada componente do sistema.

[2]https://wikibase-api.readthedocs.io/ /downloads/en/latest/pdf/

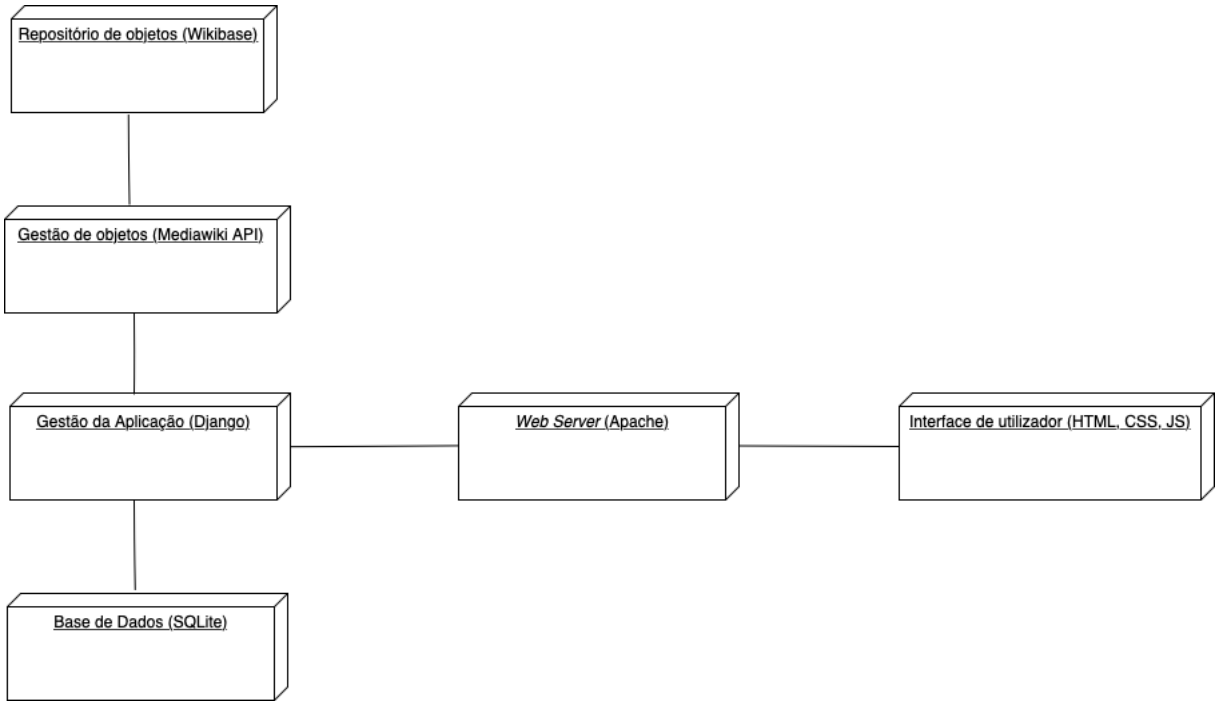


Figura 4.4: Arquitetura do sistema em produção

4.3.1 Servidor Web

O sistema está instalado numa máquina Debian GNU/Linux 10 e usa um servidor Apache HTTP (v2.4.38). O servidor físico está alojado pelo Instituto Superior Técnico (IST) e foi disponibilizado especificamente para o projeto Profmus. Assim que forem reunidas as condições oportunas, as aplicações vão ser transferidas para um servidor próprio da Universidade Nova de Lisboa.

4.3.2 *Application Controller*

A aplicação Django é executada num ambiente virtual com Django (v2.2.7) e Python (v3.7.3) instalados. A implementação foi executada usando WSGI e seguindo o tutorial disponível na página web do Django.

4.3.3 *Object Repository and Object Controller*

Para poderem ser configurados, requerem o PHP instalado assim como uma base de dados MySQL. Instalámos então o PHP (v7.4.4) e uma base de dados MariaDb (v10.3.17). Durante o processo de instalação do Mediawiki, incluímos a extensão Wikibase quando questionados sobre quais as extensões a instalar. A instalação da Wikibase também pode ser feita após a instalação do Mediawiki.

4.4 Estabilização da Aplicação de Gestão dos Dados

Como referido nos capítulos anteriores, fazem parte deste projeto duas aplicações, uma para gerir os dados e outra para visualizá-los. O foco da minha tese é a aplicação de visualização de dados, ainda assim, sou também responsável pela transação desde o momento em que o aluno responsável pela aplicação de gestão entregou e defendeu a sua tese, até esta começar a ser utilizada pelos utilizadores reais. Nesta secção, vou indicar as implementações e correções de algumas funcionalidades que efetuei na aplicação. A maior parte das correções surgem de *bugs* que surgiram ao testar a aplicação as primeiras vezes com dados reais que me foram fornecidos. Atendendo que são os primeiros testes com informação real, é perfeitamente normal surgirem estas pequenas falhas.

4.4.1 Exportação dos dados mediante qualquer formulário existente no sistema

Uma das funcionalidades que foi implementada, foi a opção dos utilizadores poderem exportar dados mediante qualquer formulário que já exista no sistema. Até aqui, apenas podiam exportar mediante o mesmo formato padrão que o sistema permitia, não havendo flexibilidade para que o utilizador pudesse exportar mediante um outro formulário (preferencialmente importado por si) presente no sistema. Esta funcionalidade traz total liberdade a uma das funções mais importantes, a de exportação de dados.

4.4.2 Correção na importação de dados

Na primeiras tentativas de importação de dados reais, foram surgindo alguns erros que impediam a correta importação dos mesmos. Todos os erros estavam associados com o tipo de dados presentes no ficheiro xlsx. Se uma coluna da tabela fosse referente a uma data, o excel mudava automaticamente o tipo de dados dessa coluna para ”data”e com números inteiros acontecia o mesmo, mas neste caso mudava para o tipo ”número”. Todos os dados que não estivessem no formato ”texto”ou ”geral”, a aplicação não os conseguia ler tendo como consequência o mal funcionamento da aplicação. Tendo em conta que o excel assume estes tipos de dados automaticamente, não fazia sentido incutirmos aos utilizadores a responsabilidade de verificarem constantemente os tipos de dados presentes no ficheiro, para cada coluna. Assim, as alterações a fazer teriam que ser ao nível da aplicação para que o processo de importação fosse o mais robusto possível. Neste sentido, após uma análise complexa ao código funcional responsável pela importação dos dados, e após vários *debugs*, foi possível tornar este processo independente do tipo de dados importados e por consequência foi possível importar um primeiro bloco dos dados reais fornecidos.

5

Demonstração

Conteúdo

Após todos os passos de desenvolvimento descritos nos capítulos anteriores, obtivemos então uma aplicação funcional que cumpre com os requisitos pressupostos. No *timing* de desenvolvimento desta tese, a aplicação ainda não vai ser usada intensivamente como se espera a médio/longo prazo. Tal se deve aos vários processos de recolha de informação ainda pendentes e também à importação destes dados para a aplicação de gestão, aplicação que também ainda não está totalmente estável, estando agora a ser feito os primeiros testes e as correções e implementações adicionais que advêm dos testes. Nos próximos sub capítulos vão ser descritos, em pormenor, com o auxílio de imagens da aplicação, as funcionalidades existentes.

5.1 Publicar Vista

Esta função faz a ”ponte”entre a aplicação de gestão dos dados e a aplicação de visualização dos dados. Quando a vista é publicada é criada uma base de visualização da vista para a posterior configuração pelo administrador.

5.2 Alterar título e sub título

Esta função permite ao administrador alterar o titulo e sub titulo da vista. Assim, permite que as duas principais *labels* da vista sejam configuradas sempre que o administrador pretenda. Na fig.5.2 é possível ver a opção de alteração destes títulos e na fig.5.1.

Figura 5.1: Visualização final do título e sub título na vista.

Figura 5.2: Opção de alteração do título e sub título da vista.

5.3 Alterar título das secções de informação

Esta função permite ao administrador alterar os títulos de cada secção de informação. Inclusive permite que o administrador possa, por exemplo, mudar os títulos para outra língua.

Figura 5.3: Opção de alteração do título de uma secção.

5.4 Informações Adicionais

Uma das secções na visualização da vista permite ver alguns detalhes adicionais acerca da construção da vista tal como o número de objetos, a data de criação, a data da última edição e o número de classes diferentes nos objetos da vista. De momento o administrador da vista não pode fazer nenhuma escolha acerca desta secção. Ver fig. 5.4.

Figura 5.4: Informações adicionais da vista na aplicação

5.5 Definições acerca da tabela

Esta secção tem as principais escolhas possíveis de fazer pelo administrador em relação à tabela. Neste caso, podem ser escolhidas as propriedades referentes a cada objeto que se quer mostrar na tabela. Na tabela em si ainda é possível ordenar os valores segundo uma das colunas, por ordem crescente ou decrescente, além de ser possível fazer uma pesquisa aberta por um qualquer valor, sendo esta pesquisa em *real time* filtrando os resultados da tabela. A tabela ainda oferece a funcionalidade de consultar mais informação sobre os objetos. bastando para tal clicar numa das linhas da tabela. Na figura 5.5 é possível ver o resultado final da tabela na aplicação. Cada coluna representa uma propriedade e a caixa no canto superior direito representa o local onde se pode fazer uma pesquisa livre. Esta última funcionalidade, que podemos ver na fig. 5.6, foi também um requisito importante para a comunidade do projeto Profmus. Na fig. 5.7 é possível ver o resultado quando se carrega numa das linhas da tabela. Permite consultar todas as propriedades e valores desse objeto em específico.

Figura 5.5: Exemplo da tabela na aplicação

5.6 Definições acerca do mapa

Esta secção tem as principais escolhas possíveis de fazer pelo administrador em relação ao mapa. Neste caso, podem ser escolhidas múltiplas definições como escolher as propriedades visíveis no mapa, associar uma cor a cada propriedade ou ainda mais específico, associar uma cor a cada propri-

Figura 5.6: Exemplo de pesquisa na tabela na aplicação

Figura 5.7: Exemplo do resultado quando se carrega numa das linhas da tabela na aplicação.

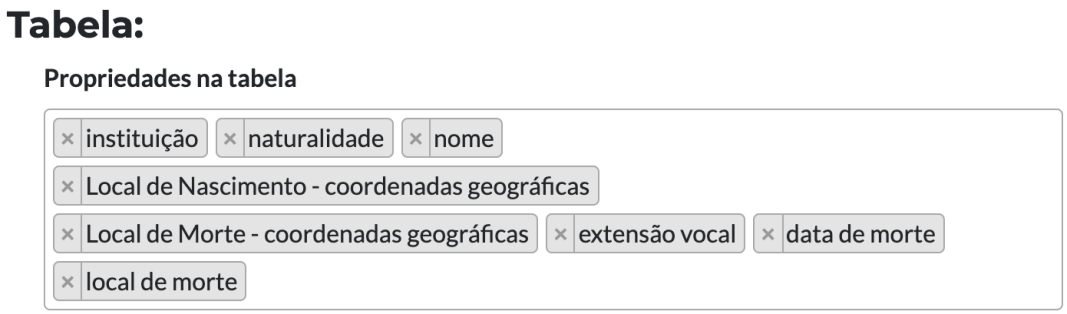


Figura 5.8: Escolha de propriedades a mostrar na tabela.

edade de cada objeto. Caso haja uma cor associada a uma propriedade e outra associada à mesma propriedade mas de um objeto específico, prevalece o nível mais baixo de especificidade. Um dos requisitos para o mapa seria a possibilidade de fazer zoom até ao nível das ruas, como é visível na fig.5.9. É um aspecto importante já que o sistema trata as localizações através de coordenadas geográficas o que permite ter uma localização com alto nível de precisão, tal como foí pedido pelos investigadores.

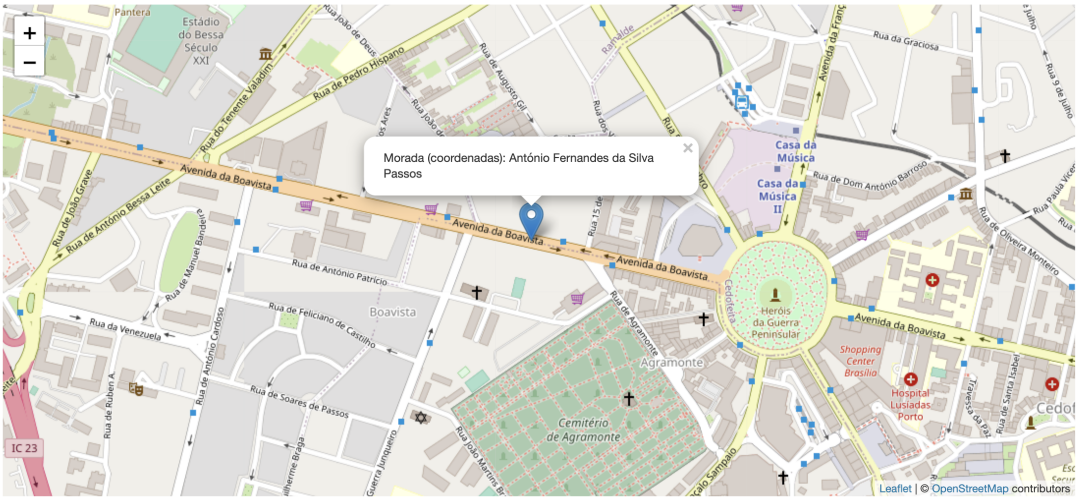


Figura 5.9: Ruas promenorizadas no mapa da aplicação.

Figura 5.10: Exemplo real da projeção no mapa das moradas de três músicos.

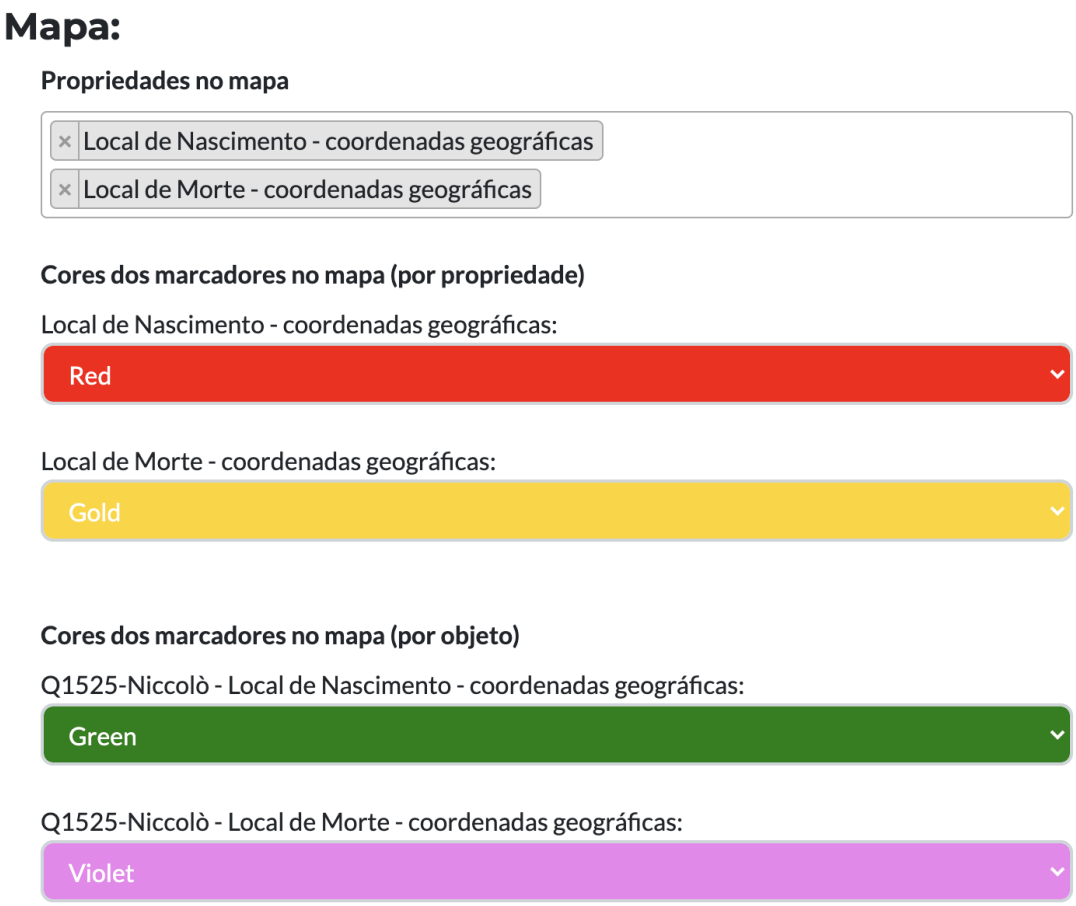


Figura 5.11: Definições acerca do mapa.

5.7 Definições acerca da linha temporal

Nesta definição o utilizador pode escolher quais os objetos (por exemplo, músicos), cujo as variadas datas quer ver no gráfico. Na figura 5.12 podemos ver uma linha temporal com dados reais do projeto. Os elementos da vista em causa apenas possuem as datas de morte registadas.

Figura 5.12: Exemplo real de um gráfico temporal na aplicação.

5.8 Ordem da Informação

Esta definição permite ao administrador ordenar ou reordenar as secções de informação arrastando os blocos para cima uns dos outros conforme se pretenda. Desta forma o administrador pode definir a ordem que pretende seja por prioridade de importância ou por mera preferência.

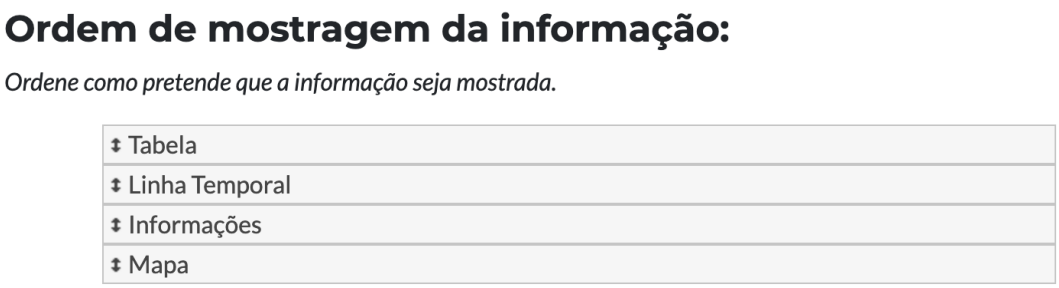


Figura 5.13: Escolha da ordem da informação.

6

Avaliação

Conteúdo

Para avaliar a interação dos utilizadores com a aplicação, bem como a sua usabilidade, realizámos uma série de sessões de avaliação qualitativa com os utilizadores alvo, ou seja, com os especialistas em música que participam no projeto Profmus.

6.1 Caracterização do utilizador

A aplicação foi desenvolvida pensando num tipo de utilizador muito específico. É um tipo de utilizador que não tem necessariamente muitas capacidades técnicas quando se trata de lidar com componentes informáticas e tecnológicas tais como computadores e aplicações, mas que domina bem o manuseamento de informação em ficheiros xlsx, principalmente informação relacionada com a música e com músicos. Assim sendo, as ações possíveis de realizar na aplicação terão que ter um aspecto e comportamento básico e simples.

6.2 Moldes dos testes

Atendendo às restrições em vigor devido à pandemia da COVID-19, todos os contactos com as pessoas envolvidas no projeto PROFMUS foram através de plataformas de vídeo-chamada. Assim sendo, as sessões de testes foram feitas através da plataforma Zoom. Os participantes apenas tinham que ter:

- um computador com um navegador relativamente recente;
- um rato e teclado;
- acesso à internet para se ligarem ao endereço da aplicação;

6.3 Metodologia

No geral, cada sessão de teste durou cerca de 30 minutos. Os testes consistiram nos seguintes passos:

- **Introdução** - Neste passo foi explicado aos utilizadores o propósito da sessão e em que contexto se inseria. Também foi neste passo que foi fornecido aos utilizadores um formulário online que servirá de guia durante o teste e onde estão descritas todas as tarefas com algumas perguntas a acompanhar cada tarefa. O formulário está disponível no apêndice A.2.
- **Realização de tarefas** - Neste passo os utilizadores realizaram as tarefas descritas no formulário, sem limitação de tempo e após a realização da tarefa eram convidados a responder a três perguntas simples sobre a tarefa que acabaram de realizar.

- **Apreciação Geral** - Neste passo, após a execução de todas as tarefas, o utilizadores foram convidados a escrever um comentário geral acerca da aplicação, onde puderam indicar todos os aspetos positivos e negativos e ainda puderam fazer recomendações sobre algo que gostavam que fosse melhorado ou implementado de raiz. Este último passo foi também acompanhado de uma conversa informal.

6.4 Avaliação

Esta secção vai explicar que componentes do sistema foram avaliados e quais foras as tarefas que os utilizadores tiveram que executar. O objetivo principal dos testes foi perceber a facilidade com que os utilizadores conseguiram executar cada tarefa, assim como também perceber quais foram as suas dificuldades. Durante os testes também foi avaliado o manual de utilizador, já que este foi fornecido aos utilizadores para possíveis dúvidas que surgissem. Considerando todos estes aspectos, foram criadas as seguintes tarefas:

- Fazer login na aplicação de gestão de dados com a conta indicada e verificar se existe uma vista criada pela mesma conta onde tem sessão iniciada. (T1)
- Clicar na vista criada “por si” e verificar se tem a opção de “publicar vista”. (T2)
- Carregar em ”publicar vista”, experimentar alterar o título da vista e clicar em guardar. Verificar se o título foi alterado. (T3)
- Experimentar alterar o sub titulo da vista e clicar em guardar. Verificar se o sub titulo foi alterado. (T4)
- Experimentar alterar o titulo da tabela e clicar em guardar. Verificar se o titulo foi alterado. (T5)
- Escolher uma das colunas da tabela e experimentar ordenar os dados segundo os valores dessa coluna. (T6)
- Experimentar pesquisar por um nome de um dos objetos presentes na tabela. (T7)
- Experimentar clicar numa das linhas da tabela. Confirmar a visualização de mais informação acerca do objeto em causa. (T8)
- No separador “Definições” experimentar mudar a cor do Cabeçalho. (T9)
- Experimentar adicionar e remover uma propriedade da tabela e do mapa. (T10)
- Experimentar mudar a cor de um dos marcadores do mapa. (T11)

- Experimentar trocar a ordem da informação. (T12)
- Copiar o link presente no navegador e experimentar abrí-lo num separador anónimo. Confirmar que deixa de conseguir editar a informação e que estão presentes todas as suas alterações. (T13)

No final de cada tarefa, os utilizadores tinham três perguntas para responder, duas obrigatórias e uma facultativa. As duas obrigatórias, ambas de escolha múltipla, pediam ao utilizador para que indicasse se a tarefa foi bem sucedida, escolhendo a opção ”sim”ou ”não”e com que facilidade foi executada. Neste caso, indicando numa escala de 1 a 7 em que 1 representa ”nada fácil”e 7 representa ”muito fácil”. A resposta facultativa é uma resposta de escrita livre, em que o utilizador pode, caso se aplique, indicar as dificuldades dignas de nota na execução da tarefa em causa. No final da sessão, os utilizadores foram confrontados com uma questão obrigatória de resposta aberta, onde puderam escrever uma apreciação mais geral sobre a aplicação.

6.5 Resultados

Nesta secção vão ser apresentados os resultados dos testes feitos aos utilizadores. A avaliação contou com 7 utilizadores, maioritariamente pessoas envolvidas no projeto Profmus. No fim de cada tarefa, cada utilizador respondeu se concluiu a tarefa com sucesso e indicou o nível de dificuldade da execução da tarefa em que:

- 1 - Extremamente Difícil
- 2 - Difícil
- 3 - Relativamente Difícil
- 4 - Dificuldade moderada
- 5 - Relativamente Fácil
- 6 - Fácil
- 7 - Extremamente Fácil

Na tabela 6.1 estão apresentados, para cada utilizador, o nível de dificuldade que relataram para cada tarefa. Analisando os resultados, é facilmente perceptível que no geral os utilizadores executaram todas as tarefas com alguma facilidade. As tarefas 8,9,10,11 e 12 foram as que ofereceram menor facilidade na sua execução na perspetiva dos utilizadores. Todas as restantes foram globalmente identificadas

Tabela 6.1: Avaliação do utilizador por tarefa

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
Utilizador 1	7	7	7	7	7	7	7	7	7	7	7	7	7
Utilizador 2	7	7	7	7	7	7	7	7	7	7	7	7	7
Utilizador 3	7	7	7	7	7	7	7	6	6	7	5	6	7
Utilizador 4	6	7	6	7	7	7	6	6	6	5	7	4	7
Utilizador 5	7	7	7	7	7	7	7	7	7	7	7	7	7
Utilizador 6	7	7	7	7	7	7	7	7	7	7	7	7	7
Média	6.83	7	6.83	7	7	7	6.83	6.66	6.66	6.66	6.66	6.33	7

como extremamente fáceis de executar. Em seguida vão ser apresentados os vários comentários gerais feitos por todos os utilizadores no fim das respetivas sessões. Cada um destes comentários engloba opiniões e sugestões sobre cada tarefa que achem necessário destacar. Engloba também uma opinião geral sobre a aplicação e sugestões para o futuro.

Para um utilizador pouco experiente a aplicação pareceu bastante simples e clara de utilizar. Senti alguma dificuldade nalgumas tarefas que foram facilmente resolvidas através de uma explicação muito breve ou da leitura do manual. Talvez a tarefa menos fácil tenha sido a pesquisa de um termo na tabela pois a caixa da pesquisa não está identificada: esta dificuldade também foi rapidamente ultrapassada depois da explicação e não teria problemas nas próximas utilizações. Após este teste posso afirmar que gostei de trabalhar com a aplicação, que torna muito simples o trabalho para o investigador e para o utilizador e a apresentação é limpa e clara. O manual de instruções também é bom e parece explicar claramente as funções.

Na óptica deste utilizador a tarefa mais difícil foi a de pesquisar um termo na caixa de pesquisa da tabela, por esta não estar bem identificada. Tal observação foi útil para corrigir este aspeto na aplicação, destacando um pouco mais a caixa de pesquisa. As restantes opiniões destacam o facto de o manual do utilizador estar bom e completo, além de observações mais gerais destacando a simplicidade do manuseamento da aplicação.

As propriedades da Tabela e do Mapa, quando são alteradas, permanecem alteradas no layout da página Definições ainda que as alterações não tenham sido guardadas. Este ponto pode ser confuso pois: se eu voltar ao separador Definições sem me lembrar que alterações havia feito, posso pensar que as propriedades que a página mostra são as que estão gravadas quando tal não é verdade. Penso que se pode ou mostrar um aviso quando as Definições não foram gravadas ou desfazer as alterações não guardadas sempre que se sai do separador Definições (talvez esta seja mais intuitiva/comum para o utilizador). Na ordem de mostragem da informação, parece-me que o grafismo não é muito explícito, ainda que um utilizador experiente perceba que funciona por "drag and drop". Penso que arranjar algum grafismo mais explícito ou comum a outros sites poderia ficar melhor. Ainda que, novamente, um utilizador experiente perceba e intuitivamente resolva facilmente o problema.

Este utilizador releva o facto de a aplicação não alertar para o facto de as alterações feitas ainda não terem sido confirmadas, através do clique no botão para o efeito. Este lapso de informação pode

induzir o utilizador em erro, levando-o a assumir que as alterações estão efetivas na sua vista, quando não estão. O utilizador em causa relatou também que o modo como o ordenamento da informação é feito não é o mais claro, não sendo totalmente perceptível. Estas observações foram tidas em conta e serão corrigidas na aplicação.

Plataforma bastante fácil e intuitiva, bonita e bem desenvolvida. Espero que a importação dos dados seja tão fácil como este teste. A minha única recomendação vai de encontro ao facto de achar mais fácil se houvesse divisórias por instituições, por ex. Irmandade de Santa Cecília, Sindicato dos Músicos, etc. Mantendo uma linha temporal que permita perceber o desenvolvimento da acção associativa da classe dos músicos.

O autor deste comentário realça a facilidade e simplicidade de manuseamento da aplicação. A única recomendação feita assenta na gestão dos dados, com a eventual possibilidade de dividir a informação por instituições e projetando na linha temporal a evolução do contacto dos músicos com as várias associações.

Achei fácil a utilização desta aplicação e bastante intuitiva. Gostei muito da linha do tempo e do mapa. Apenas faço as seguintes sugestões: a) permitir escolher os objetos e as propriedades na linha temporal b) permitir escolher os objetos e as propriedades no mapa c) permitir escolher o zoom inicial do mapa d) a visualização da linha da tabela ser semelhante à do Reasonator e) deixar a tabela escondida e colocar no local da pesquisa o termo "Pesquisa", ou então um "visual"semelhante ao google.

Como apreciação final da aplicação, este utilizador reportou 5 sugestões muito específicas no que considera serem melhorias importantes a aplicar/corrigir. Todas foram consideradas e tidas em conta. Importa relatar também o facto de o utilizador em questão ter achado a aplicação fácil e intuitiva de interagir.

Apreciei a organização é clara e a facilidade de execução das tarefas, quase sempre intuitivas. A excepção é a modificação da "Ordem de mostragem da informação", pois é um pouco menos imediato como funciona, mas mesmo assim não oferece dificuldades de maior. Deveria tornar-se mais evidente quando as alterações ficam efectivamente guardada. Do ponto de vista gráfico (numa perspectiva puramente estética) acho que a aplicação também podia ser melhorada, mas não sei se isto é função do Tiago ou se deveríamos ter também a colaboração de um designer.

Este utilizador aponta também para a questão da aplicação não alertar que as alterações feitas ainda não estão guardadas, tal como um dos outros utilizadores. Diz também que a organização é clara e que as tarefas são facilmente executáveis. Deixa ainda uma sugestão em relação ao aspeto gráfico da aplicação, onde talvez a ajuda de um *designer* fosse uma mais valia.

Manejar a aplicação foi bastante simples e intuitivo, o que facilita muito o trabalho dos utilizadores. O público utilizará também bastante bem a plataforma e visualizará os conteúdos de forma fácil. Os layouts e aspeto gráfico dão uma ideia de clareza e acessibilidade à plataforma.

Este comentário faz uma observação mais geral em relação à aplicação destacando a simplicidade e a intuitividade da mesma. Elogia também o *layout* e o aspeto gráfico, não dando nenhuma sugestão prática.

6.6 Discussão de resultados

No geral, os resultados dos testes foram bastante positivos, resultando apenas em algumas sugestões que visam ainda melhorar mais as funcionalidades da aplicação. As primeiras tarefas do teste eram expectáveis de serem fáceis de executar, mesmo para os utilizadores menos experientes tecnicamente. As maiores dificuldades eram esperadas de aparecer a partir da tarefa 6 e confirmou-se com alguns utilizadores a reportar algumas dificuldades mínimas, como é possível verificar na secção anterior. É importante relatar que nenhum dos utilizadores sugeriu alterações significativas no manual da aplicação. No geral todos os intervenientes acharam a aplicação bastante intuitiva e fácil de utilizar, e algumas das dificuldades que surgiam eram rapidamente mitigadas com uma consulta rápida do manual de utilizador. Os questionários completos com perguntas e respostas estarão anexados a este documento apêndice A.2.2.

7

Conclusão

Conteúdo

7.1 Conclusões

Fazer parte do projeto Profmus permitiu ter a percepção real do vazio que existe na busca e manutenção do domínio histórico da música em Portugal. Ao mesmo tempo que nos apercebemos da riqueza cultural que o nosso país contém, apercebemo-nos também que esta riqueza nem sempre é bem aproveitada e estimada.

No trabalho em específico que foi desenvolvido nesta tese, deparámo-nos com a total fragmentação da informação o que tornava difícil fazer uma pesquisa mais granular ou uma comparação real entre vários músicos tendo em consideração alguma característica comum entre eles. A solução ideal passou por agrupar toda a informação recolhida num só sistema informático. Os primeiros passos foram dados com a colaboração de um outro aluno, na elaboração da sua tese final de mestrado. Ele foi responsável por coordenar a recolha de informação ainda que este processo seja progressivo no tempo e por desenvolver uma aplicação responsável pela gestão da informação recolhida. No término da sua tese, a aplicação por ele desenvolvida era responsável por todas as ações diretas nos dados como importar, exportar, eliminar, etc. Umas das ações também disponíveis é a de criação e gestão de vistas, que são sub grupos de informação. Por exemplo, um utilizador pode criar uma vista que contenha todos os músicos que nasceram em 1800. O objetivo final destas vistas é a sua publicação, que passa por criar um site para a visualização dos dados contidos na mesma.

Um dos requisitos inicialmente definidos é de que o administrador de uma vista (quem a criou) pode definir as características visuais e gráficas do ”site”da sua vista. Esta tese foca-se precisamente no desenvolvimento da aplicação responsável pela publicação destes elementos. Esta aplicação permite que o utilizador possa escolher vários elementos gráficos onde possam ser projetados os dados, tais como um mapa, uma tabela e uma linha temporal. Em cada um destes elementos há um conjunto de opções possíveis de escolher e definir.

Aquando uma primeira versão estável da aplicação, foram feitos vários testes com alguns dos investigadores do projeto, onde estes executaram um conjunto de tarefas e no fim puderam dar uma opinião geral da aplicação, dando (ou não) sugestões para a melhoria da mesma.

Após a realização dos testes com utilizadores reais, o *feedback* dos utilizadores foi bastante positivo. Todas as tarefas foram executadas com sucesso e no geral a maior parte das tarefas foi considerada ”fácil”ou ”muito fácil”de fazer. A aplicação foi também considerada intuitiva e simples de manusear. De todos os comentários realizados no âmbito dos testes, destaco a importância de dois em específico, feitos por dois dos principais coordenadores do projeto Profmus. Os elogios e sugestões inseridos nestes dois comentários, transmitiram um grande grau de satisfação e uma sensação de missão cumprida.

Já na fase final do desenvolvimento desta tese, os coordenadores do projeto tiveram a amabilidade de me convidar a integrar o projeto como investigador em definitivo, onde vou continuar a colaborar na estabilização de ambas as aplicações informáticas desenvolvidas. Este convite cimentou a excelente

relação de colaboração que sempre foi mote durante todo o processo em que trabalhámos juntos, resultando na continuação desta colaboração enquanto o projeto de investigação decorrer.

7.2 Trabalho Futuro

Em relação ao trabalho futuro, numa aplicação de visualização de dados, há sempre mais elementos que podem ser adicionados para que a visualização seja ainda mais completa. Em relação ao mapa, há várias funcionalidades extra que podem vir a ser implementadas. Por exemplo, pode vir a ser implementada uma funcionalidade em que seja possível ver a variação em termos geográficos de um músico. Imaginemos um músico que passou por vários pontos do nosso país, seja em habitação ou em compromissos profissionais. Pode vir a ser interessante fazer um *tracking* a estas variações.

Um dos investigadores, no seu teste, referiu que poderia ser uma mais valia incluir um tipo de visualização dos dados de um músico semelhante ao da aplicação Reasonator[1]. Como podemos ver na fig.7.1, é um tipo de visualização muito semelhante ao da Wikipédia. Segundo o investigador em questão, é um tipo de visualização a que as futuros utilizadores das aplicações estão habituados, logo poderia ser uma vantagem incluir no futuro uma funcionalidade deste género na aplicação desenvolvida.

O design gráfico da aplicação pode também vir a ser melhorado. Como base foi usado um *template* Bootstrap, já que não é a minha especialidade a conceção do design gráfico base de interfaces web. Assim sendo, acredito que a colaboração futura de um designer profissional possa vir acrescentar beleza estética à base da aplicação.

Para finalizar, apenas com o uso intensivo do sistema e com um maior número de dados será perceptível um outro tipo de elementos ou funcionalidades que possam vir a acrescentar mais valias, por isso, acredito que a melhoria da aplicação se prolongue no tempo.

[1] https://reasonator.toolforge.org

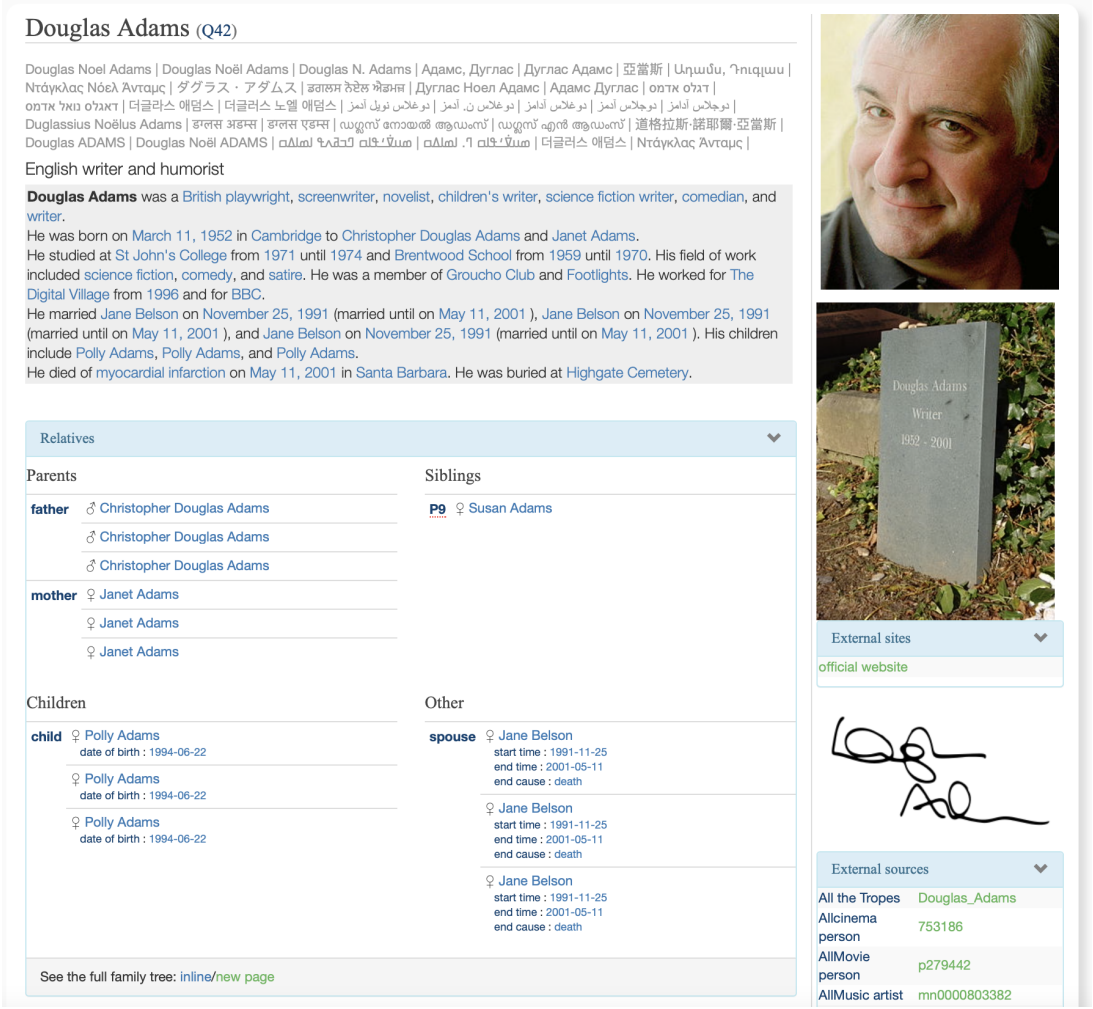


Figura 7.1: Visualização de dados na aplicação Reasonator.

Bibliografia

[1] What is rest api? [accessed 23-October-2020]. [Online]. Available: https://www.zestard.com/blog/rest-api-benefits/

[2] N. Guarino, "Formal ontologies and information systems," in *Proceedings of FOIS'98*, Trento, Italy, 1998, pp. 5–6. [Online]. Available: http://www.loa.istc.cnr.it/old/Papers/FOIS98.pdf

[3] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human - Computer Studies*, vol. 43, no. 5-6, pp. 907–928, 1995.

[4] M. Achichi, R. Bailly, C. Cecconi, M. Destandau, M. Achichi, R. Bailly, C. Cecconi, M. Destandau, K. Todorov, M. Achichi, and R. Bailly, "DOREMUS : Doing Reusable Musical Data To cite this version : HAL Id : hal-01309167 DOREMUS : Doing Reusable Musical Data," 2016.

[5] P. Choffé, "Doremus : Connecting sources , enriching eatalogues and eser experience," in *DOREMUS : Connecting Sources, Enriching Catalogues and User Experience*, 2016, pp. 1–216.

[6] S. Drobi, "Play2: A new era of web application development," *IEEE Internet Computing*, vol. 16, no. 4, pp. 89–94, 2012.

[7] "The 1st 2018 Indonesian Association for Pattern Recognition International Conference (INAPR) : proceeding : September 7, 2018, Bina Nusantara University, Alam Sutera Campus, Tangerang, Banten, Indonesia," *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, no. 1, pp. 218–222, 2018.

[8] Django documentation. [accessed 21-October-2020]. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[9] K. Boonchuay, Y. Intasorn, and K. Rattanaopas, "Design and implementation a REST API for association rule mining," in *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. IEEE, jun 2017. [Online]. Available: https://doi.org/10.1109%2Fecticon.2017.8096326

[10] D. Serrano, E. Stroulia, D. Lau, and T. Ng, “Linked REST APIs: A Middleware for Semantic REST API Integration,” *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*, pp. 138–145, 2017.

[11] Representational state transfer (rest). [accessed 23-October-2020]. [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Apêndices

A.1 Manual de Utilizador

Ser Músico em Portugal - Aplicação de Configuração de Vistas

Manual do Utilizador

Serviço desenvolvido em parceria entre:

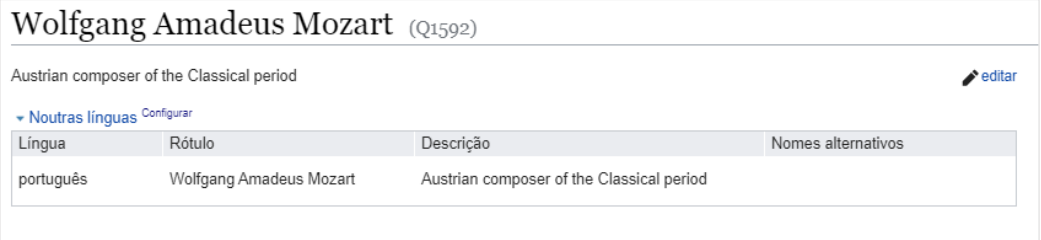
- INETmd: Projeto "PROFMUS - Ser Músico em Portugal: a condição sócio-profissional dos músicos em Lisboa (1750-1985)", co-financiado pela FCT com a referência PTDC/ART-PER/32624/2017
- INESC-ID: Laboratório de Sistemas de Informação e Apoio à Decisão (IDSS)

Índice

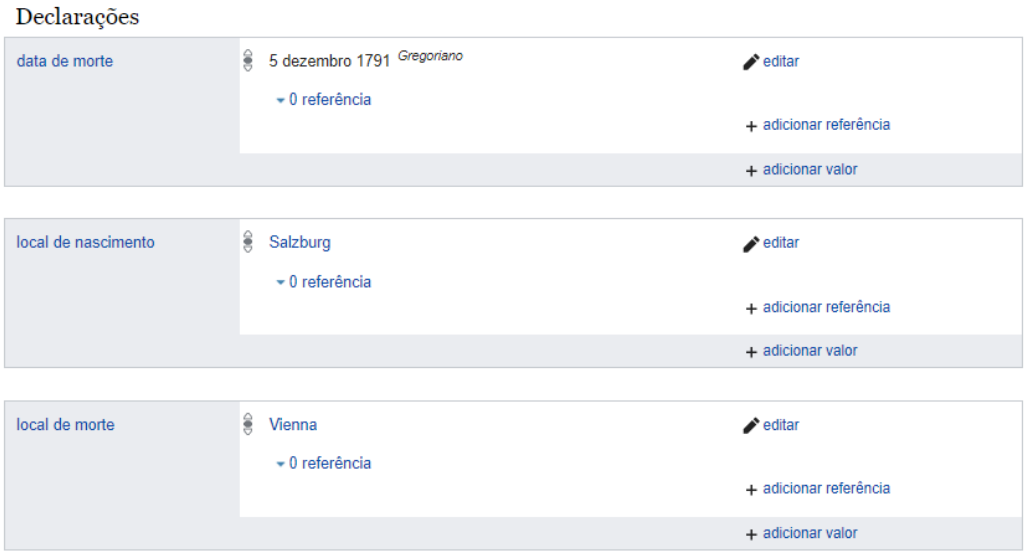
1. Descrição Geral do Sistema

Esta aplicação permite a criação de uma interface de visualização de dados guardados numa instalação MediaWiki e geridos por uma terceira aplicação desenvolvida também no âmbito deste projeto. A MediaWiki funciona de forma semelhante à Wikipédia, mas permite o armazenamento de dados estruturados. E esta aplicação faz uso precisamente dessa funcionalidade. Como tal, será explicado abaixo um pouco da estrutura dos dados quando estes são guardados.

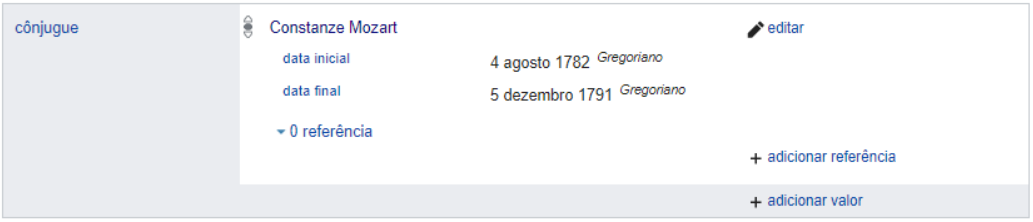
Os dados são guardados na MediaWiki sob a forma de **objetos**. Um objeto representa algo ou alguém, podendo tanto ser um músico como um instrumento, uma instituição ou uma localidade. Um objeto tem um nome e pode ter uma descrição textual.



Adicionalmente, pode depois ser caracterizado por um conjunto de **propriedades** e respetivos valores. Por exemplo, um músico pode ter uma data de nascimento e um conjugue, ou uma instituição pode ter um ou mais fundadores.



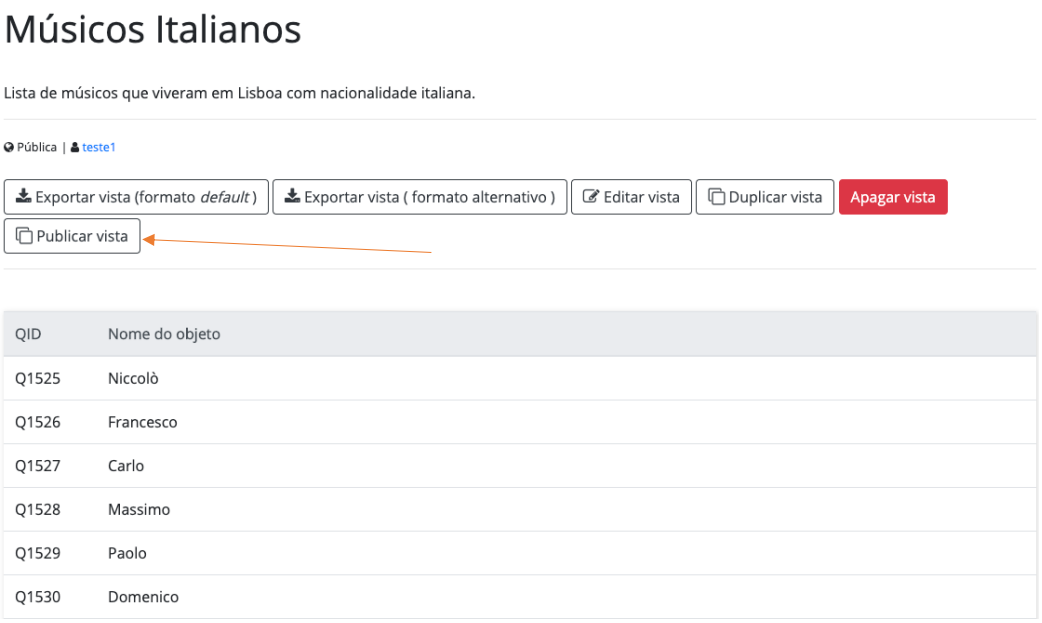
Por sua vez, as propriedades podem ter **qualitativos** que as caracterizam. Qualitativos funcionam para as propriedades como as propriedades funcionam para os objetos. Por exemplo, um músico pode ter um conjugue com o qual foi casado durante um período temporal. Assim, a propriedade "conjugue" poderá ter qualitativos, como a "data inicial", "data final", ou o local do casamento.



No sistema, foi implementada uma forma de criar coleções de objetos a que chamamos **Vistas**. Uma vista trata-se de um conjunto de objetos agrupados de forma a que possam ser posteriormente usados na criação de uma página *web* sobre os mesmos.

Estas **Vistas** podem então ser publicadas ficando acessíveis ao público em geral. É a publicação das vistas o foco desta aplicação disponibilizando para tal um conjunto de configurações onde o administrador pode definir o aspeto estético com que os utilizadores poderão consultar os dados. A informação é extraída da Mediawiki através de uma REST API já embutida na sua instalação.

2. Publicar Vista



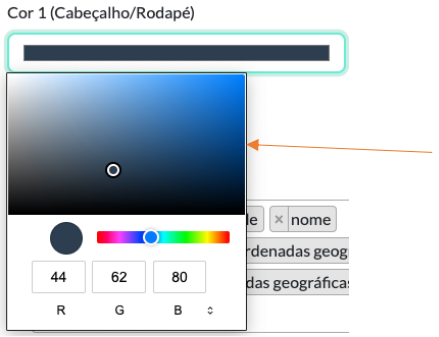
Ainda na aplicação de gestão dos dados, nas vistas criadas pelo utilizador com sessão iniciada, temos a opção de a publicar. Basta entrar no separador “vistas”, clicar numa que tenha sido criada pelo utilizador e clicar no botão “Publicar vista”. Nas vistas criadas por outros utilizadores esta opção não aparece.

3. Configurações Estéticas

Para alterar o título e subtítulo basta editar os campos de texto e clicar no botão “Guardar” do respetivo campo.

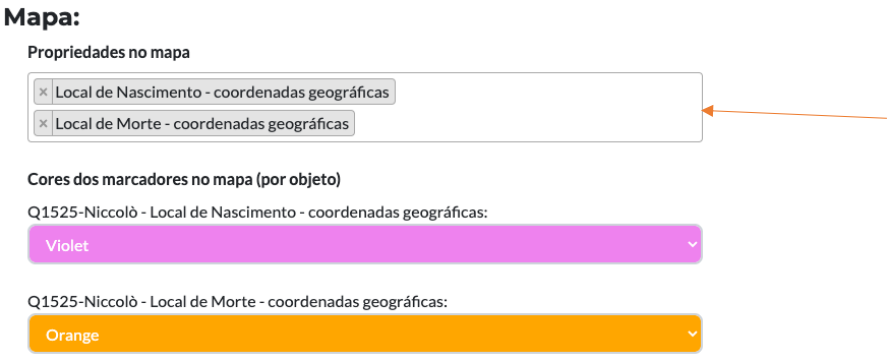
Para alterar os títulos de cada secção de informação, basta editar os campos de texto e clicar no botão “Guardar” respetivo.

4. Definições da Vista



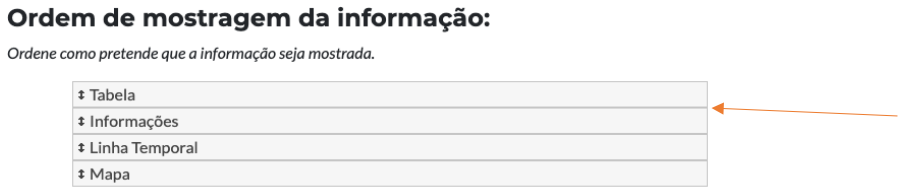
Para alterar a cor tanto do cabeçalho/rodapé como a cor primária presente no fundo de algumas secções, basta clicar na caixa de cor, e escolher a nova cor. Para guardar basta clicar no botão “Guardar” mais abaixo.

Em relação à tabela, é possível escolher as colunas presentes na mesma. Para tal, basta escolher as propriedades que pretendemos mostrar, carregando na caixa respetiva e selecionando ou desseleccionando as propriedades que pretendemos adicionar ou remover.



Em relação ao mapa podemos configurar dois aspetos diferentes. Podemos configurar, como em relação à tabela, as propriedades projetadas no mapa. Deste modo, basta clicar na caixa das propriedades e escolher as propriedades que queremos mostrar ou as que queremos esconder. Além desta configuração, podemos também

associar uma cor a cada marcador presente no mapa. Por exemplo, podemos selecionar a cor do marcador que representa o local de nascimento do cantor “Niccolò” e uma outra cor para o seu local de morte.

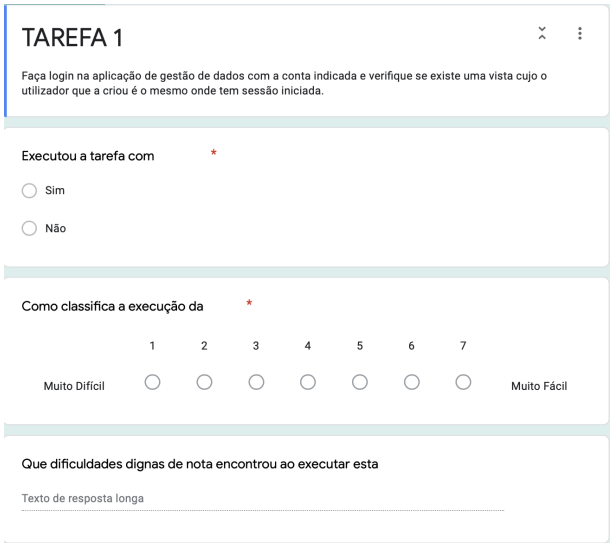


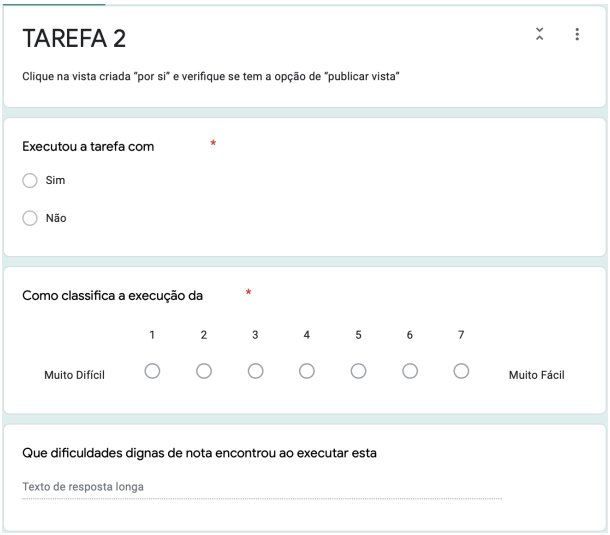
A última definição possível de alterar é a ordem de aparecimento das secções de informação. Basta ordenar ou reordenar as barras colocando na pela ordem que pretendemos.

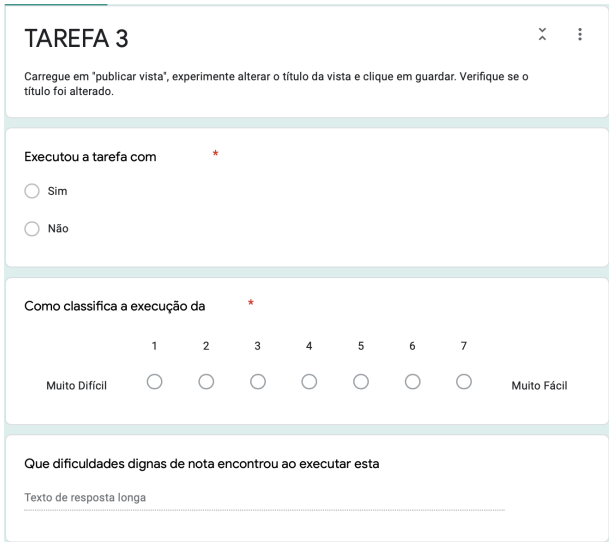
Todas estas definições são confirmadas ao carregar no botão “Guardar” no fundo da página das definições.

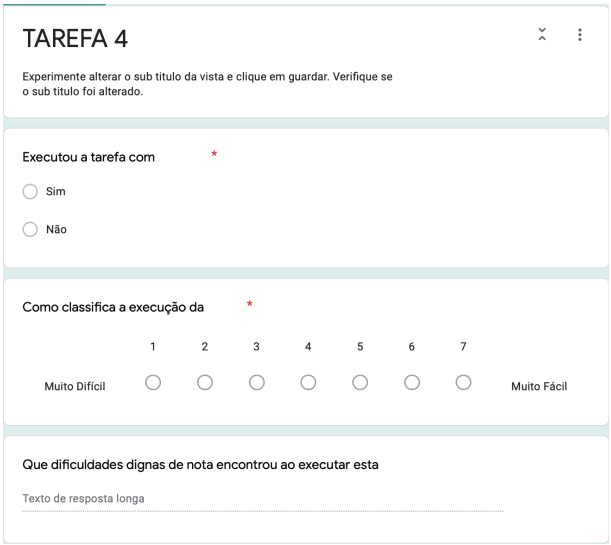
A.2 Testes Realizados

A.2.1 Tarefas









TAREFA 5

Experimente alterar o titulo da tabela e clique em guardar. Verifique se o titulo foi alterado.

Executou a tarefa com *

- Sim
- Não

Como classifica a execução da *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta

Texto de resposta longa

TAREFA 6

Escolha uma das colunas da tabela e experimente ordenar os dados segundo os valores dessa coluna.

Executou a tarefa com sucesso? *

- Sim
- Não

Como classifica a execução da tarefa? *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta tarefa?

Texto de resposta longa

TAREFA 7

Experimente pesquisar por um nome de um dos objetos presentes na tabela.

Executou a tarefa com sucesso? *

- Sim
- Não

Como classifica a execução da tarefa? *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta tarefa?

Texto de resposta longa

TAREFA 8

Experimente clicar numa das linhas da tabela. Confirme a visualização de mais informação acerca do objeto em causa.

Executou a tarefa com sucesso? *

- Sim
- Não

Como classifica a execução da tarefa? *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta tarefa?

Texto de resposta longa

TAREFA 9

No separador "Definições" experimente mudar a cor do Cabeçalho.

Executou a tarefa com sucesso? *

- Sim
- Não

Como classifica a execução da tarefa? *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta tarefa?

Texto de resposta longa

TAREFA 10

Experimente adicionar ou remover uma propriedade da tabela e do mapa.

Executou a tarefa com sucesso? *

- Sim
- Não

Como classifica a execução da tarefa? *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta tarefa?

Texto de resposta longa

TAREFA 11

Experimente mudar a cor de um dos marcadores do mapa.

Executou a tarefa com sucesso? *

- Sim
- Não

Como classifica a execução da tarefa? *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta tarefa?

Texto de resposta longa

TAREFA 12

Experimente trocar a ordem da informação.

Executou a tarefa com sucesso? *

- Sim
- Não

Como classifica a execução da tarefa? *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta tarefa?

Texto de resposta longa

TAREFA 13

Copie o link presente no seu navegador e experimente abrí-lo num separador anónimo. Confirme que deixa de conseguir editar a informação e estão presentes todas as suas alterações.

Executou a tarefa com sucesso? *

- Sim
- Não

Como classifica a execução da tarefa? *

	1	2	3	4	5	6	7	
Muito Difícil	○	○	○	○	○	○	○	Muito Fácil

Que dificuldades dignas de nota encontrou ao executar esta tarefa?

Texto de resposta longa

TAREFA 14

Descrição (opcional)

Como apreciação geral, indique o que mais gostou, o que menos gostou e que recomendações adicionais faz à aplicação. *

Texto de resposta longa

A.2.2 Resultados

Questionários Tarefa-1

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	Nenhuma
Sim	7	
Sim	6	
Sim	7	
Sim	7	

Questionários Tarefa-2

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	

Questionários Tarefa-3

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	7	
Sim	6	
Sim	7	
Sim	7	

Questionários Tarefa-4

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	

Questionários Tarefa-5

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	

Questionários Tarefa-6

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	

Questionários Tarefa-7

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	7	
Sim	6	
Sim	7	
Sim	7	

Questionários Tarefa-8

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	7	
Sim	6	
Sim	7	
Sim	7	

Questionários Tarefa-9

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	6	
Sim	6	
Sim	7	
Sim	7	

Questionários Tarefa-10

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	6	
Sim	5	
Sim	7	
Sim	7	

Questionários Tarefa-11

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	
Sim	7	

Questionários Tarefa-12

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	5	
Sim	4	
Sim	7	
Sim	7	

Questionários Tarefa-13

Executou a tarefa com sucesso?	Como classifica a execução da tarefa?	Que dificuldades dignas de nota encontrou ao executar esta tarefa?
Sim	7	
Sim	7	
Sim	6	
Sim	7	
Sim	7	
Sim	7	

Questionários Tarefa-14

Como apreciação geral, indique o que mais gostou, o que menos gostou e que recomendações adicionais faz à aplicação.
As propriedades da Tabela e do Mapa, quando são alteradas, permanecem alteradas no layout da página Definições ainda que as alterações não tenham sido guardadas. Este ponto pode ser confuso pois: se eu voltar ao separador Definições sem me lembrar que alterações havia feito, posso pensar que as propriedades que a página mostra são as que estão gravadas quando tal não é verdade. Penso que se pode ou mostrar um aviso quando as Definições não foram gravadas ou desfazer as alterações não guardadas sempre que se sai do separador Definições (talvez esta seja mais intuitiva/comum para o utilizador). Na ordem de mostragem da informação, parece-me que o grafismo não é muito explícito, ainda que um utilizador experiente perceba que funciona por "drag and drop". Penso que arranjar algum grafismo mais explícito ou comum a outros sites poderia ficar melhor. Ainda que, novamente, um utilizador experiente perceba e intuitivamente resolva facilmente o problema.
Manejar a aplicação foi bastante simples e intuitivo, o que facilita muito o trabalho dos utilizadores. O público utilizara também bastante bem a plataforma e visualizará os conteúdos de forma fácil. Os layouts e aspeto gráfico dão uma ideia de clareza e acessibilidade à plataforma.
Apreciei a organização é clara e a facilidade de execução das tarefas, quase sempre intuitivas. A excepção é a modificação da "Ordem de mostragem da informação", pois é um pouco menos imediato como funciona, mas mesmo assim não oferece dificuldades de maior. Deveria tornar-se mais evidente quando as alterações ficam efectivamente guardada. Do ponto de vista gráfico (numa perspectiva puramente estética) acho que a aplicação também podia ser melhorada, mas não sei se isto é função do Tiago ou se deveríamos ter também a colaboração de um designer (depois, a longo prazo, no âmbito do projecto).
Para um utilizador pouco experiente a aplicação pareceu bastante simples e clara de utilizar. Senti alguma dificuldade nalgumas tarefas que foram facilmente resolvidas através de uma explicação muito breve ou da leitura do manual. Talvez a tarefa menos fácil tenha sido a pesquisa de um termo na tabela pois a caixa da pesquisa não está identificada: esta dificuldade também foi rapidamente ultrapassada depois da explicação e não teria problemas nas próximas utilizações. Após este teste posso afirmar que gostei de trabalhar com a aplicação, que torna muito simples o trabalho para o investigador e para o utilizador e a apresentação é limpa e clara. O manual de instruções também é bom e parece explicar claramente as funções.
Achei fácil a utilização desta aplicação e bastante intuitiva. Gostei muito da linha do tempo e do mapa. Apenas faço as seguintes sugestões: a) permitir escolher os objetos e as propriedades na linha temporal b) permitir escolher os objetos e as propriedades no mapa c) permitir escolher o zoom inicial do mapa d) a visualização da linha da tabela ser semelhante à do Reasonator e) deixar a tabela escondida e colocar no local da pesquisa o termo "Pesquisa", ou então um "visual" semelhante ao google
Plataforma bastante fácil e intuitiva, bonita e bem desenvolvida. Espero que a importação dos dados seja tão fácil como este teste. A minha única recomendação vai de encontro ao facto de achar mais fácil se houvesse divisórias por instituições, por ex. Irmandade de Santa Cecília, Sindicato dos Músicos, etc. Mantendo uma linha temporal que permita perceber o desenvolvimento da acção associativa da classe dos músicos.