



UNIVERSIDADE
LUSÓFONA

Agregador Multitemático Trabalho Final de curso

Autores:

João Cardoso, nº 21704574

Rafael Reto, nº 21702113

Orientador:

Fernando Teodósio

Trabalho Final de Curso | LEI | 26/06/2020

www.lusofona.pt

Direitos de cópia

Agregador Multitemático, Copyright de *João Cardoso e Rafael Reto*, ULHT.

A Escola de Comunicação, Arquitectura, Artes e Tecnologias da Informação (ECATI) e a Universidade Lusófona de Humanidades e Tecnologias (ULHT) têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Índice

Índices de Quadros, Figuras, Tabelas, Equações, source code exemplo	4
Índice de Anexos.....	5
Resumo.....	6
Abstract	7
Introdução	8
1. Identificação do Problema.....	9
2. Levantamento e análise de Requisitos	10
3. Viabilidade e Pertinência	11
4. Solução Desenvolvida	12
5. Benchmarking	14
a) Identificação de soluções existentes em mercado e análise comparativa com a solução proposta, indispensável para valorização do critério de avaliação de inovação	14
b) Enquadramento teórico e estado da arte (com revisão bibliográfica)	14
<input type="checkbox"/> O que é a Web 3.0?	14
<input type="checkbox"/> O que é o RDF?.....	15
<input type="checkbox"/> O que é o SparQL.....	16
<input type="checkbox"/> O que é o XML?.....	17
<input type="checkbox"/> Posicionamento da solução relativamente à competição:	18
6. Método e Planeamento	18
7. Resultados	20
8. Conclusão e Trabalhos Futuros	25
Calendário	26
Bibliografia	27
Anexos	28

Índices de Quadros, Figuras, Tabelas, Equações, source code exemplo

Figura 1 - Arquitetura esperada.....	12
Figura 2 - Arquitetura Adaptada face aos problemas encontrados	13
Figura 3 - Web 3.0	15
Figura 4 - Exemplo de RDF	16
Figura 5- Exemplo de implementação de SparQL	16
Figura 6- Exemplo de Query de SparQL no Projecto	17
Figura 7- Exemplo de Query de SparQL no Projecto	17
Figura 8 - Exemplo de ficheiro XML	18
Figura 9- Ideia inicial do calendário	19
Figura 10- Calendário utilizado depois dos ajustes dos problemas	19
Figura 11 - Home page Com Query	20
Figura 12- Resultado da Query	21
Figura 13- Parte do XML usado no Projecto	21
Figura 14 - Ler o XML relativo aos restaurantes	22
Figura 15- Parse da Query para perceber o tipo de serviço	22
Figura 16- Parse da Query para perceber a cidade.....	22
Figura 17- Parse da Query para perceber a data de partida/inicio	23
Figura 18- Guardar na Lista restaurantes relativos à pesquisa feita.....	23

Índice de Anexos

Anexo 1- calendarioTFC.xlsx	28
Anexo 2 - TFC1920_P02Req_21702113;21704574.xlsx (Requirement 1).....	28
Anexo 3- TFC1920_P02Req_21702113;21704574.xlsx (Requirement 2).....	29
Anexo 4- TFC1920_P02Req_21702113;21704574.xlsx (Requirement 3).....	29
Anexo 5 - TFC1920_P02Req_21702113;21704574.xlsx (Requirement 4).....	30
Anexo 6 - TFC1920_P02Req_21702113;21704574.xlsx (Requirement 5).....	30

Resumo

Agregador Multitemático é uma aplicação *web* capaz de aceder a dados contidos na base de dados em que são devolvidos resultados consoante o *input (query)* inserido pelo utilizador, ou seja, faz uso da pesquisa natural, permitindo assim ao utilizador e ao computador cooperarem entre si. A *web* semântica interliga os significados de palavras com a finalidade de atribuir um significado. Isto faz com que seja compreensível tanto pelo utilizador como pelo computador.

Esta aplicação poderá ser usada para diversos tipos de comércio/empresa, no entanto queríamos basear-nos na área da hotelaria e turismo para conseguirmos aprofundar um pouco mais e mostrar o nosso objetivo.

Para o desenvolvimento deste TFC usamos uma tecnologia chamada *Web 3.0 (Web Semântica)*. Esta *Web Semântica* faz com que o utilizador possa inserir naturalmente na *searchbar* o que pretende procurar e a aplicação devolve um resultado, e esta pesquisa poderá ser algo do tipo literal por exemplo: “Quero um hotel em Coimbra para dia 26/06”. Com esta *Web 3.0* facilita a pesquisa ao *user* e é mais intuitivo. A nossa aplicação tem presente uma *Front-End*, uma *Back-End* e uma Base de Dados onde são armazenados os dados para serem mostrados à posterior na *Front-End*.

O que difere esta aplicação das mais comuns existentes na internet é o uso desta pesquisa adaptativa a cada utilizador que reduz a complexidade de pesquisa e permite uma pesquisa mais eficaz e mais centralizada nos resultados esperados.

Abstract

Multithematic Aggregator is a web application capable of accessing data contained in the database that are returned as a result or query (query) entered by the user, that is, it makes use of natural search, thus allowing the user and the computer cooperates with each other. A semantic interconnection of the Web with meanings of words allowed to assign meaning. This makes it understandable by both the user and the computer.

This application can be used for different types of commerce / company, however, you should look for an area of hospitality and tourism to get more details about a little more and show our objective.

For the development of this TFC, use a technology called Web 3.0 (Semantic Web). This Semantic Web makes it possible for the user to naturally insert into the search bar or to search for and apply a result, and this search can be something of the literal type, for example: "I want a hotel in Coimbra for 06/26". With this Web 3.0 it facilitates a search for the user and is more intuitive. An application of ours features a Front-End, a Back-End and a Database, where the data is stored to be shown later in the Front-End.

What differentiates this application from the most common ones on the internet is the use of this adaptive search to each user, which reduces the complexity of the search and allows a more effective search and more focused on the expected results.

Introdução

A *internet* já teve diversas fases, começando pelo *web 1.0* e seguindo até ao que vamos desenvolver neste TFC, o *web 3.0*.

Introduzindo rapidamente como chegamos a esta etapa da *internet* e começando pelo início, o *web 1.0* era hipoteticamente uma página que quase não interagira com o utilizador, os seus conteúdos eram meramente institucionais. Já o *web 2.0* foi revolucionária pois trouxe os tão famosos *chats* e redes sociais bem como o uso abundante de blogs tudo isto produzido pelos utilizadores da web. Por fim e relativamente mais importante visto ser a razão pela qual este TFC está a ser desenvolvido, a *web 3.0*, esta inovação da Internet visa mostrar informação de forma organizada e que não apenas os utilizadores humanos possam compreender mas também as máquinas, assim estas poderão ser uma mais valia para nós, humanos, respondendo a pesquisas e perguntas com uma solução concreta, personalizável e ideal para o *user*. Esta é uma ideia não se trata de inteligência artificial, mas sim de um sistema onde o computador consegue ler um bloco de informação.

Com isto, o nosso projeto tem como funcionalidades usar a *web 3.0* para pesquisar serviços na parte do turismo, ou seja, o utilizador irá usufruir de um método de pesquisa e o computador irá responder com vários resultados dependendo da vertente escolhida pelo utilizador, sendo estas estadia, voos, restauração e pontos de interesse ou atividades, esta pesquisa poderá ser ainda filtrada por preços, datas, locais e número de pessoas. De acordo com isto, ao ser dada a resposta do computador a partir de uma lista, o utilizador verifica qual o resultado que se enquadra mais com o que ele pretende e poderá efetuar uma reserva desse serviço. Irá ser também facultado um campo para ser feita a avaliação do serviço após este ser realizado a partir de um comentário, que será exposta em cada serviço para ajudar outros utilizadores nas suas decisões.

Dito isto, pretendemos fazer uso de um certo conjunto de técnicas para preparar “terreno” para que a implementação final do projeto seja mais fácil e intuitiva. Em primeiro lugar identificamos o problema que queremos resolver, depois apresentamos o plano de resolução e finalmente apresentamos a resolução,

ou parte dela, e por final concluímos com o que poderá ser o uso deste projeto daqui para a frente.

1. Identificação do Problema

Todos os projetos pretendem resolver um problema, sendo o nosso resolver o uso de preenchimento de diferentes campos para chegar a um resultado o que poderá tirar a paciência dos utilizadores e assim fazer a empresa perder receita de negócio.

Apesar da evolução das pessoas à capacidade que têm de navegar pela internet, a mesma tem vindo a melhorar a sua capacidade de interação com o utilizador, e isto deve-se ao uso de *web* semântica. Em vez de obrigar o utilizador a preencher vários campos para chegar a um resultado, ele apenas faz uma pergunta ao browser e a internet devolve os resultados esperados.

É exatamente esse o problema, o problema é a falta de simplificação para o utilizador puder fazer as suas pesquisas. Um exemplo real é procurar por um hotel em Lisboa e ter que preencher pelo menos dois campos para além dos dias. O que a solução oferece seria preencher apenas uma input box com a seguinte pergunta “Quero um hotel em Lisboa para 26-06-2020”, o que facilitaria rapidamente o problema e o utilizador não era tao impaciente por ter que preencher variados campos.

2. Levantamento e análise de Requisitos

Antes de Proceder à implementação do projeto em si é necessário preparar como vai ser formado e, portanto, usamos uma técnica usada por todos os que desenvolvem software. O uso de requisitos permite informar o que será feito.

Dito isto, para o desenvolvimento deste Projeto, realizamos 4 requisitos principais, sendo estes os mais necessários para que o utilizador consiga usufruir da aplicação.

O primeiro requisito implementa o Registo de um utilizador, neste registo é necessário que sejam preenchidos os campos de um formulário, sendo alguns deles opcionais e o resto obrigatório. É ainda validado se estes campos são preenchidos da forma certa, caso o sejam o registo será aceite, caso contrário será pedido que os corrija.

No segundo requisito é apresentado o *Login* do Utilizador onde o mesmo terá de preencher o formulário com *e-mail* e *password*, caso estes campos coincidam o login será feito, se isto não acontecer o utilizador será informado do erro e que corrija esses campos.

O terceiro requisito refere-se à Pesquisa, este requisito é opcional, ou seja, o utilizador faz uma pesquisa, caso este queira a efetuar tendo feito login ou não. Ao introduzir a *query*, é apresentada uma lista de resultados ao utilizador.

O quarto requisito representa a Reserva de um serviço, para que isto ocorra, o utilizador terá de preencher os dados referentes de cada passageiro. É também um requisito opcional, pois os utilizadores podem acabar por não concluir a reserva.

O quinto requisito visa representar o processo que ocorre no *back-end* do projeto, aqui é recebido o input do utilizador, e este input é verificado para perceber o tipo de serviço e cidade correspondente, depois consulta a base de dados em *SPARQL* e devolve uma lista dos resultados possíveis.

3. Viabilidade e Pertinência

O nosso projeto no lado frontal apenas abrange a área do turismo, mas após a conclusão deste TFC será possível acrescentar mais áreas como comercialização de diversos produtos (veículos, imóveis, etc), visualização de entretenimento (filmes, séries e documentários), entre muitos outros. Num lado de *back-end* a pertinência deste projeto é algo muito mais importante e apesar do utilizador não ver será a parte fulcral deste projeto, isto porque não será uma técnica de pesquisa que entrará em declínio e a sua estimativa de vida tem uma longevidade enorme. As técnicas aplicadas neste *software* permitem validar como possível conceito académico a ser ensinado apesar da sua complexidade, visto ser uma ferramenta duradoura e intuitiva quando aplicada. Achamos que a nossa ideia é algo vantajoso para quem o usa, pois consegue ajudar a encontrar serviços económicos ou de grande necessidade com grande facilidade e rapidez.

Este projeto tem como publico alvo, pessoas que pretendam viajar, no entanto este conjunto pode abranger uma variação de idades muito variada, portanto o site está planeado para que este défice não tenha qualquer implicação na exploração do site. Esta pesquisa semântica que está a ser implementada tem como objetivo isso mesmo, facilitar o uso do mesmo. De momento, apenas foram feitos testes funcionais, ou seja, os testes foram feitos durante o processo de desenvolvimento do projeto. Estes testes ajudaram-nos a perceber se as funcionalidades estavam corretas e funcionais num espaço de tempo curto e com melhor qualidade.

Pegando no dito acima temos algumas empresas pela qual nos guiamos para a construção desta aplicação, uma delas é a *Teoturis*: <http://teoturis.gagicrc.com/>, esta pela qual nos foi encomendado o projeto. Uma referência que tivemos em mente foi uma junção entre o site: <https://www.tripadvisor.pt/> e o site <https://www.wolframalpha.com/>, que um se relaciona com o tema a tratar, ou seja a parte relativa ao turismo e outro faz uso da área que nos propusemos a desenvolver, a pesquisa semântica, respetivamente.

4. Solução Desenvolvida

Conhecida já a área de inserção decidimos então fazer o planeamento da arquitetura a dispor no projeto.

Face ao proposto conseguimos desenvolver parte da arquitetura inicialmente esperada. Abaixo segue a arquitetura que foi desenhada para a implementação final do projeto, coisa que não foi possível devido aos problemas encontrados durante o ano.

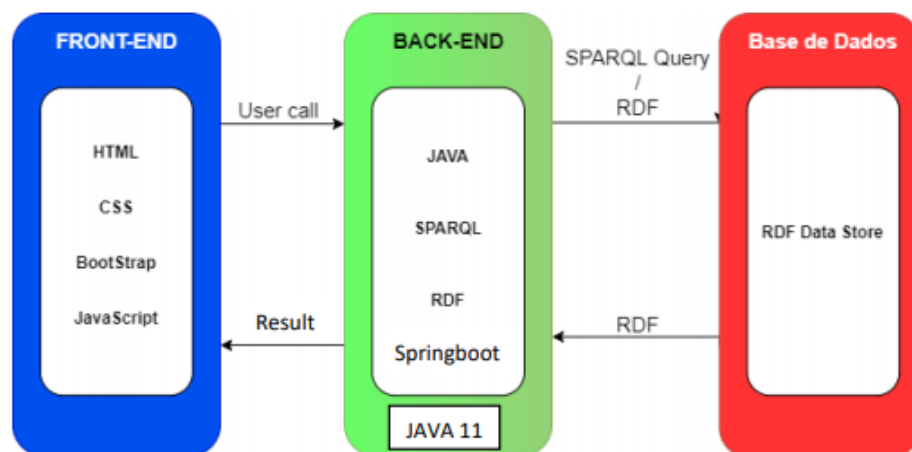


Figura 1 - Arquitetura esperada

Na figura acima o funcionamento é o seguinte: No *Front-End* o utilizador comunica através do UI usando as tecnologias HTML CSS e JavaScript. Para além disto usamos ainda *Bootstrap* para tornar o site mais apelativo. Passando agora para o *Back-End* usa *springboot* para dar *host* ao servidor, para além disso o uso da linguagem JAVA para proceder à lógica relacionada com a *query* recebida e era suposto o *Back-End* enviar a *query* ao à Base de Dados e devolver um *RDF* com os resultados, o que não foi possível.

Em relação à base de dados vimo-nos sem tempo de desenvolver tanto um servidor para o site como um servidor para a base de dados, portanto decidimos “simular” a base de dados para que o website pudesse funcionar com alguma normalidade.

Dito isto era suposto enviar uma *SPARQL Query* e receber um *RDF* com a informação relativa. Como decidimos deixar a base de dados de fora, já não era viável enviar uma *SPARQL Query* e, portanto, decidimos enviar apenas uma

Query do tipo *string* com o input do utilizador onde separávamos por espaços. Depois disto teríamos de devolver o resultado, aqui era também suposto receber um *RDF*, mas um *RDF* utiliza *URI* e nós não tínhamos nenhum face ao problema da base de dados. Portanto adotamos a solução mais perto do *RDF* que é o *XML*, sendo estes dois muito similares. A partir daqui conseguimos responder às *querys* do utilizador com o resultado esperado.

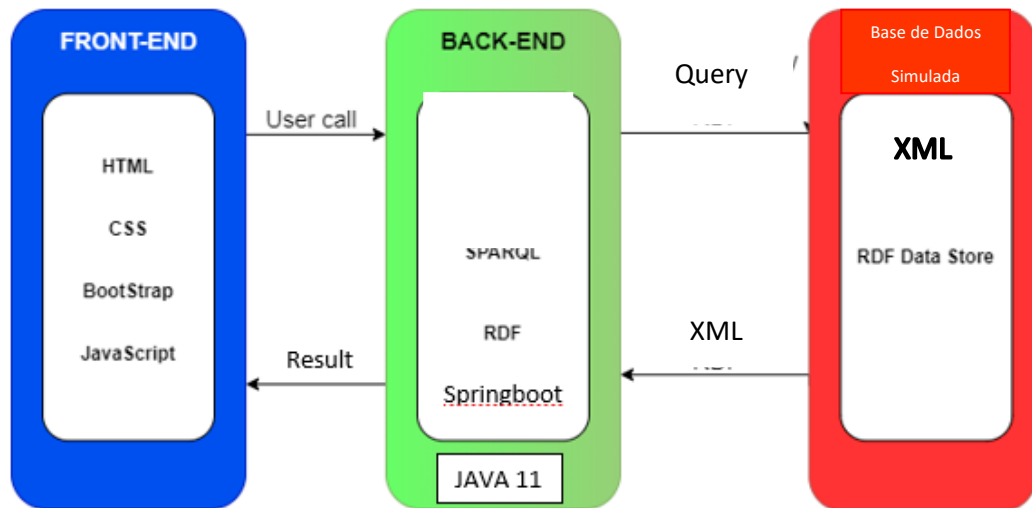


Figura 2 - Arquitetura Adaptada face aos problemas encontrados

As alterações relativas à arquitetura inicial (Fig.1) comparando à arquitetura implementada (Fig.2) situam-se na base de dados que passou a ser uma base de dados simulada por um XML e, portanto, o uso do SPARQL nem do RDF foram implementados para utilizar.

5. Benchmarking

Neste segmento fazemos uma análise aos principais competidores e fazemos um aprofundamento teórico sobre o que é que compõe a área, ainda nos situamos relativamente à competição

a) Identificação de soluções existentes em mercado e análise comparativa com a solução proposta, indispensável para valorização do critério de avaliação de inovação

Olhando para o mercado averiguámos que há uma aplicação parecida com o objetivo do nosso TFC, sendo esta a *tripadvisor*:

<https://www.tripadvisor.pt/> mas que não faz uso da pesquisa semântica.

Comparando a nossa solução pretendida com este site, uma das diferenças mais impactantes seria o uso da *Web 3.0* para o nosso TFC onde a pesquisa do utilizador seria muito mais intuitiva e responsiva.

b) Enquadramento teórico e estado da arte (com revisão bibliográfica)

- **O que é a Web 3.0?**

A *Web 3.0* é um software autodidata, ou seja, a partir do conteúdo que encontra na Internet, tem a capacidade de analisar a popularidade dos vários conteúdos e obtém conclusões. Isto é, as pessoas não têm de refinar os termos de pesquisa, porque a própria *Web 3.0* consegue-o fazer sozinha. A *Web 3.0* tem duas principal tecnologia envolvidas são elas o *RDF* e o *XML*.

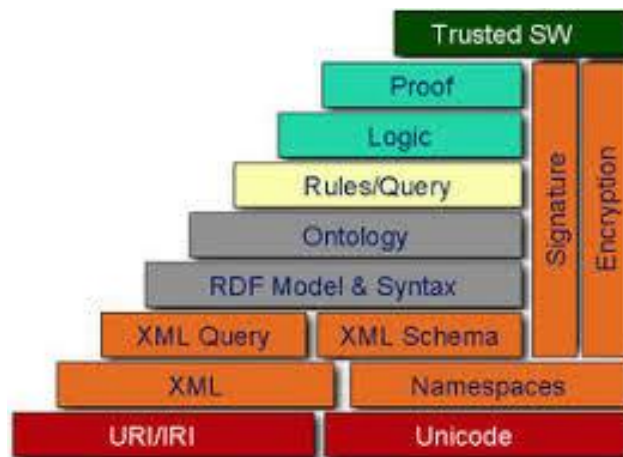


Figura 3 - Web 3.0

- **O que é o RDF?**

O *RDF* tem como objetivo ajudar o desenvolvimento de metadados, ou seja, facilitar a comunicação transparente entre os sistemas que compartilham informações e que sejam entendidas por outros sistemas da internet. O *RDF* é um grande apoio da *Web Semântica*, isto porque os metadados representados em *RDF* dão um significado aos recursos da *Web Semântica*, facilitando a manipulação e a compreensão feita por computadores. O conceito de *RDF* é formado por três tipos de objetos sendo eles os recursos, as propriedades e os valores.

- **Recursos**

Os recursos são descritos por uma expressão *RDF* e são todos identificados por um *URI (Uniform Resource Identifier)*, ou seja, tudo o que contenha um *URI* é um recurso, incluindo as páginas *Web* e os elementos de um documento *XML*.

- **Propriedades**

As propriedades são as características que se usam para descrever um recurso.

- **Valores**

Os valores são a junção de recursos e propriedades, ou seja, consiste no valor de uma propriedade possuída por um recurso, podendo este valor ser um recurso ou um tipo primitivo definido por *XML*.

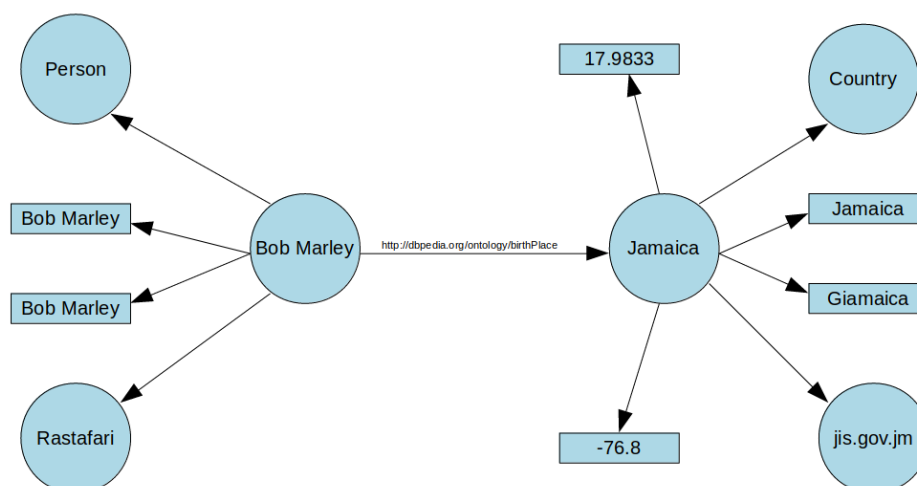


Figura 4 - Exemplo de *RDF*

- **O que é o SparQL**

SPARQL é um acrónimo recursivo do inglês *SPARQL Protocol and RDF Query Language*. Trata-se de uma linguagem padronizada para a consulta de grafos *RDF*, padrão pelo *RDF Data Access Working Group (DAWG)* do *World Wide Web Consortium (W3C)*. É uma tecnologia básica no desenvolvimento da *web* semântica que se constituiu como recomendação oficial do *W3C* a 15 de janeiro de 2008, sendo atualizada à versão 1.1 em 2013.

Num princípio *SPARQL* unicamente incorpora funções para a recuperação de parágrafos *RDF*. No entanto, algumas propostas também incluem operações para a manutenção (criação, modificação e apagar) de dados.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { <http://exemplo.org/livros> dc:title ?title
```

Figura 5- Exemplo de implementação de *SparQL*

Na figura acima apresenta-se uma pequena *query* em *SparQL* onde em *PREFIX* seleciona o *URI* da base de dados e o associa a uma variável *dc* onde pretende pesquisar, dado isto nas duas linhas que se seguem faz um *SELECT* pelo *title* e procura todos os livros com aquele *title*, guardando-o assim num outro *URI* associativo aos livros.


```

public static String getRestauranteQuery() {
    return "PREFIX data:<http://example.org/data/>" +
        "SELECT ?restaurante ?cidade\n" +
        "WHERE{\n" +
        " ?group a data:restaurante; data:cidade ?cidade .\n" +
        "}";
}

```

Figura 6- Exemplo de *Query* de *SparQL* no Projecto

Acima apresenta-se uma outra *query* em *SparQL* esta com o intuito de ser usada no nosso projeto, mas devido aos problemas encontrados não nos foi possível fazê-lo.

Como apresentado na *query* anterior o *PREFIX* seleciona o *URI* referente à base de dados on-line e daqui faz-se a seleção do que pretendemos procurar, aqui a *query* tem o intuito de procurar os restaurantes onde está naquela cidade. Um exemplo de um input para correr esta *query* seria: “Quero um restaurante em Coimbra”.

```

public static String getHotelQuery() {
    return "PREFIX data:<http://example.org/data/>" +
        "SELECT ?hotel ?cidade\n" +
        "WHERE{\n" +
        " ?group a data:hotel; data:cidade ?cidade .\n" +
        "}";
}

```

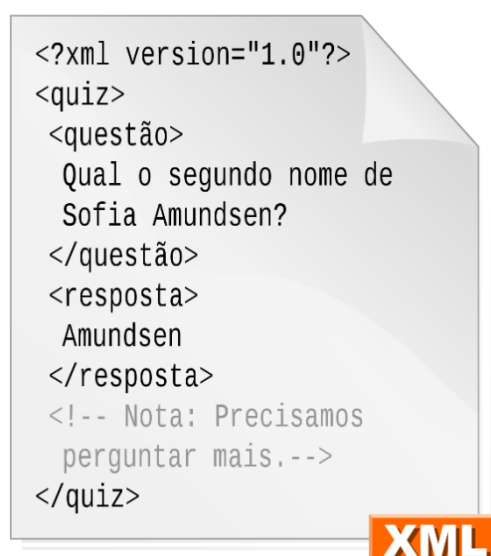
Figura 7- Exemplo de *Query* de *SparQL* no Projecto

Mais um exemplo, mas desta vez referente a um hotel.

- **O que é o XML?**

O *XML* (*Extensible Markup Language*) é um gerador de linguagens de marcação em formato de texto muito flexível derivado do *SGML* (*Standart Generalized Markup Language*). Foi criado com o objetivo de enfrentar desafios de publicações eletrónica em larga escala. O *XML* tem um papel muito fundamental nos dias de hoje e cada vez mais importante

na troca de uma ampla variedade de dados na *Web*. O *XML* contém algumas linguagens baseadas nele, como por exemplo *XHTML*, *RDF*, *SDMX* e *SMIL*.

A imagem mostra um exemplo de um ficheiro XML. O conteúdo do ficheiro é exibido dentro de um ícone de documento com uma aba dobrada no canto superior direito. O código XML contém uma declaração de versão, um elemento raiz <quiz>, um elemento <questão> com o texto 'Qual o segundo nome de Sofia Amundsen?', um elemento <resposta> com o texto 'Amundsen', e um comentário <!-- Nota: Precisamos perguntar mais.-->. Um botão laranja com o texto 'XML' em branco está localizado na base direita do ícone.

```
<?xml version="1.0"?>
<quiz>
  <questão>
    Qual o segundo nome de
    Sofia Amundsen?
  </questão>
  <resposta>
    Amundsen
  </resposta>
  <!-- Nota: Precisamos
  perguntar mais.-->
</quiz>
```

Figura 8 - Exemplo de ficheiro *XML*

- **Posicionamento da solução relativamente à competição:**

Relativamente à competição apresentado, e visto que foram apresentadas duas companhias, em relação a uma delas possuímos de certa forma e de que tenhamos conhecimento a mesma implementação ou similar do que se pretende fazer, isto respetivamente à *wolfram*. Falando agora do *tripadvisor*, não surgiu o tempo de implementar uma *front-end* capaz de apresentar tantos recursos como o *tripadvisor* apresenta nos dias de hoje, mas será um ponto a conseguir com o progredir do projeto nos anos seguintes.

6. Método e Planeamento

Inicialmente tínhamos o calendário planeado para o que pensávamos ser a estrutura indicada para o projeto, e com isto escolhemos as tecnologias que estávamos mais à vontade e que pensaríamos ser o necessário para desenvolver o projeto. Com o passar do tempo e ao descobrir mais sobre a área a explorar percebemos que não iríamos conseguir desenvolver o projeto na sua totalidade. Escolher o mais próximo da nossa

zona de conforto não foi o ideal visto haver linguagens/frameworks diferentes que poderíamos ter usado de forma mais fácil e eficaz.

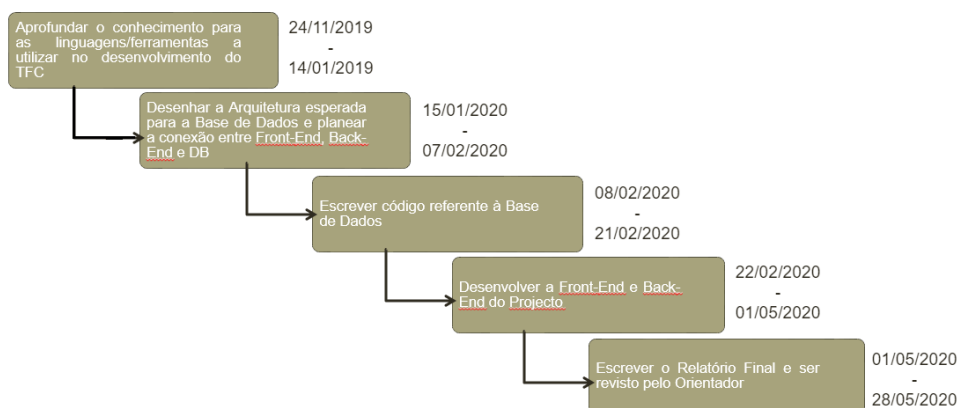


Figura 9- Ideia inicial do calendário

Para além disto e apesar de termos conseguido na maioria ficar com as linguagens/frameworks que nos sentíamos mais confortáveis, fomos ainda forçados a aprender uma linguagem de Base de Dados especifica para *Semantic Search* chamada *SPARQL*, o que mais tarde percebemos que não iríamos conseguir implementar sem um servidor dedicado à base de dados.

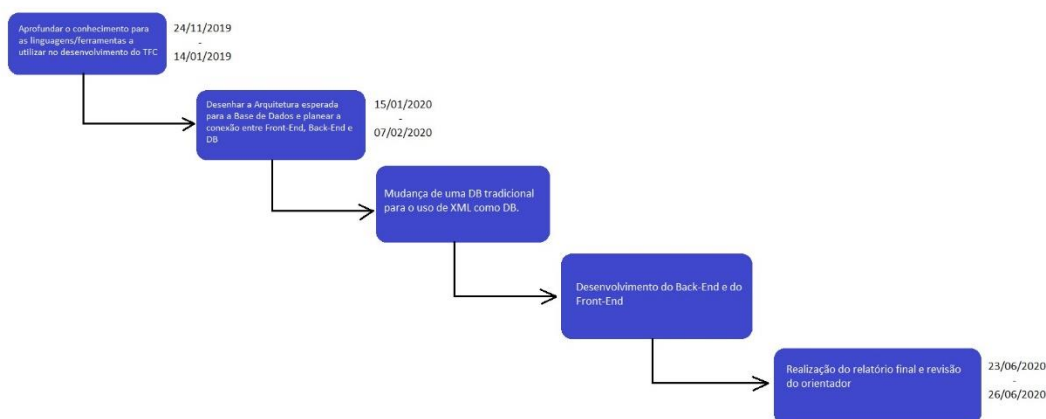


Figura 10- Calendário utilizado depois dos ajustes dos problemas

Dito isto decidimos então adotar a metodologia *AGILE* para o decorrer do resto do projeto. Com esta metodologia realizávamos sprints de 2 semanas com reuniões entre o grupo diariamente. De duas em duas semanas tínhamos uma tarefa diferente, algumas delas falharam e tivemos que adotar métodos de aproximação do problema diferentes.

7. Resultados

De seguida segue os resultados obtidos pelo grupo durante o semestre.

Primeiramente queremos apenas frisar que o nosso projeto não visa a utilização intensiva da *front-end* e, portanto, focámo-nos mais na implementação do *back-end* logo o nosso leque de resultados não é alargado. Dito isto segue o resultado:

Nas duas seguintes imagens, na Fig.11 o utilizador faz uso do seu teclado para inserir a sua pesquisa juntamente com as datas que pretende procurar, de seguida ao pesquisar é lhe devolvido o resultado (Fig.12) para todos os Hotéis e Parecidos inseridos no nosso *XML*.

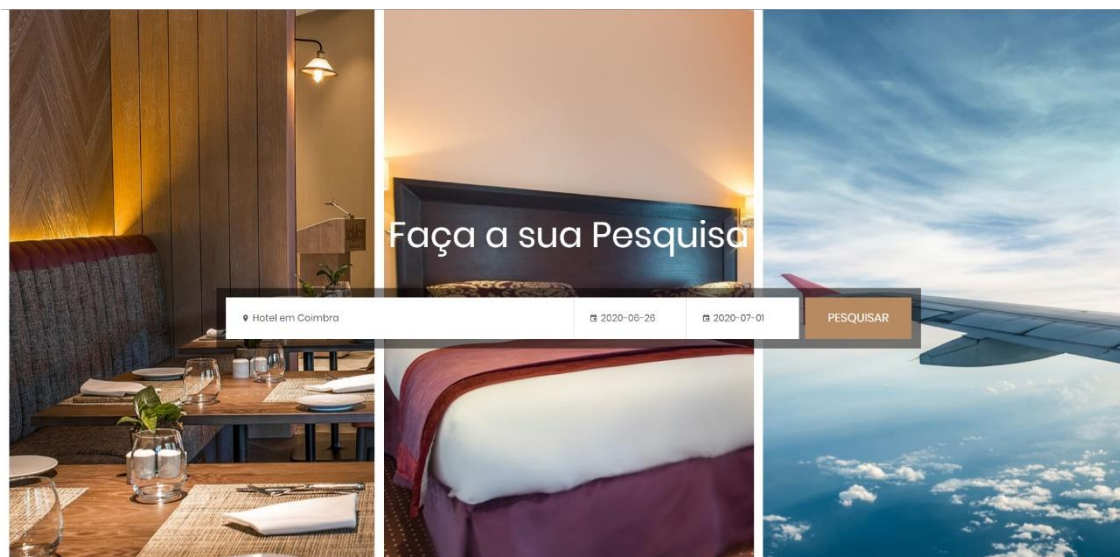


Figura 11 - *Home page Com Query*

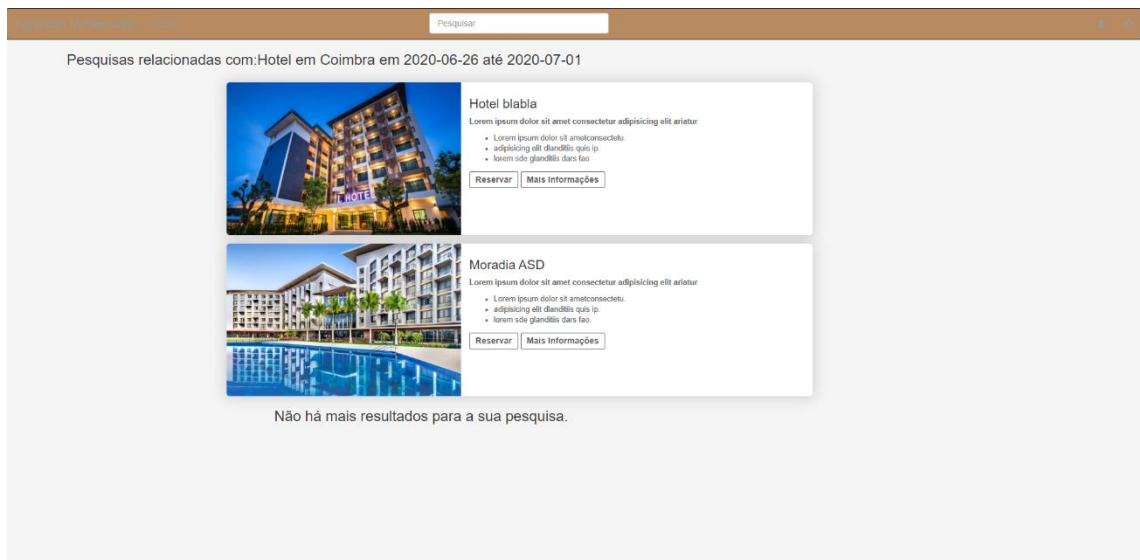


Figura 12- Resultado da *Query*

Visto que os resultados não mostram realmente o trabalho realizado durante o semestre, iremos ainda mostrar parte do que acontece por trás de todo este processo.

Como dito anteriormente simulámos a base de dados através do *XML* e portanto:

```
class>
  <restaurante>
    <id>1</id>
    <name>Comes & Bebes</name>
    <cidade>Coimbra</cidade>
    <rua>Rua dos Patrões</rua>
    <codpostal>3000-308</codpostal>
    <especialidade>Arroz de Berbigão</especialidade>
  </restaurante>
```

Figura 13- Parte do *XML* usado no Projeto

Executado a construção do nosso *XML*, temos que o ler e guardar na nossa *Back-End*. Ao ler o *XML* vamos separando os *Nodes* por tipo, por exemplo acima o node exemplificado é o “restaurante” e assim guardamos em *hashmaps* em que a *Key* será a cidade onde este Objeto reside como se pode averiguar na figura seguinte.

```

public static void readRestaurantes(NodeList nodeList) {
    for (int itr = 0; itr < nodeList.getLength(); itr++) {
        Node node = nodeList.item(itr);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            int id = Integer.parseInt(eElement.getElementsByTagName("id").item(0).getTextContent());
            String name = eElement.getElementsByTagName("name").item(0).getTextContent();
            String cidade = eElement.getElementsByTagName("cidade").item(0).getTextContent();
            String rua = eElement.getElementsByTagName("rua").item(0).getTextContent();
            String codepostal = eElement.getElementsByTagName("codpostal").item(0).getTextContent();
            String especialidade = eElement.getElementsByTagName("especialidade").item(0).getTextContent();
            Restaurante restaurante = new Restaurante(id, name, cidade, rua, codepostal, especialidade);
            restaurante_map.put(cidade, restaurante);
        }
    }
}

```

Figura 14 - Ler o XML relativo aos restaurantes

Já na pesquisa retiramos a *query* inserida pelo utilizador e fazemos o parse, como pudemos ver abaixo.

```

public static String parseSearch(String q) {
    if (q.contains("restaurante") || q.contains("esplanada") || q.contains("café") || q.contains("tasca")
        || q.contains("Restaurante") || q.contains("Esplanada") || q.contains("Café") || q.contains("Tasca"))
        return "restaurante"; // correr uma query
    if (q.contains("voos") || q.contains("voos"))
        return "voos"; // correr outra query
    if (q.contains("Hotel") || q.contains("Hotéis") || q.contains("Apartamento") || q.contains("Moradia") || q.contains("Apartamentos") || q.contains("Moradias")
        || q.contains("hotel") || q.contains("hotéis") || q.contains("apartamento") || q.contains("moradia") || q.contains("apartamentos") || q.contains("moradias"))
        return "hotel"; // correr outra query
    return "";
}

```

Figura 15- Parse da Query para perceber o tipo de serviço

Ao saber o tipo de pesquisa precisamos de pesquisar mais uma vez a cidade inserida e procedemos ao mesmo processo só que desta vez relativa a cidades, deste processo retiramos a cidade e aqui temos logo as duas palavras chaves para devolver a *query*.

```

public static String parseCidade(String q) {
    if (q.contains("Coimbra") || q.contains("coimbra"))
        return "Coimbra"; // correr uma query
    if (q.contains("Lisboa") || q.contains("lisboa"))
        return "Lisboa"; // correr uma query
    if (q.contains("Paris") || q.contains("paris"))
        return "Paris"; // correr uma query
    return "";
}

```

Figura 16- Parse da Query para perceber a cidade

Para além disto temos ainda um *parse* relativo à data para voos, abaixo separamos por espaços a *string*, e pegamos a data de partida.

```

public static String parseSearchDate(String q) {
    String[] currencias = q.split( regex: " ");
    return currencias[currencias.length - 1 - 3];
}

```

Figura 17- Parse da Query para perceber a data de partida/inicio

Finalmente para acabar este processo e como queremos passar uma lista, percorremos o *hashmap* relativo ao tipo de serviço e *key* (cidade) referente, e acrescentamos a uma lista definida para cada tipo.

```

case "restaurante": {
    Iterator iterator = restaurante_map.entrySet().iterator();
    while (iterator.hasNext()) {
        Map.Entry me2 = (Map.Entry) iterator.next();
        if (me2.getKey() == cidade) {
            Restaurante restaurantetemp = (Restaurante) me2.getValue();
            restaurante.add(restaurantetemp);
        }
    }
}

```

Figura 18- Guardar na Lista restaurantes relativos à pesquisa feita

Para não haver conflito entre tipos de variáveis passamos por fim uma lista de *strings* onde é suposto fazer a separação por “|” na *front end* do projeto e devolver o resultado.

```

public static ArrayList<String> restauranteToString(){
    ArrayList<String> novo = new ArrayList<>();
    for (int i = 0; i < restaurante.size(); i++) {
        novo.add((String) restaurante.get(i).toString());
    }
    return novo;
}

```

Figura 19- Passar Lista do tipo Restaurante para uma Lista de Strings

Para terminar temos a função que é corrida para executar o servidor do site, onde inicialmente ele lê o XML e guarda os dados em cada *HashMap* e depois sim corre o servidor com tudo preparado.

```
public static HashMap restaurante_map = new HashMap<String, Restaurante>();
public static HashMap voos_map = new HashMap<String, Voo>();
public static HashMap hotel_map = new HashMap<String, Hotel>();

public static void main(String[] args) throws IOException {
    ReadRDF.read( tipo: "restaurante");
    ReadRDF.read( tipo: "voo");
    ReadRDF.read( tipo: "hotel");
    SpringApplication.run(Application.class, args);
}
```

Figura 20- Lê XML e armazena dados em *hashmaps* diferentes e depois corre o servidor

8. Conclusão e Trabalhos Futuros

Com o desenvolvimento deste trabalho concluímos que a web semântica é um processo mais complicado do esperado, quando abraçamos este projeto. As nossas expectativas ao início eram bastante elevadas, mas com a experiência do nosso orientador Prof. Fernando Teodósio, que nos foi mentalizando de que não seria fácil atingir essas expectativas. Ao longo do tempo, percebemos que não conseguiríamos cumprir com os nossos principais objetivos, de modo a que fomos obrigados a tomar decisões sobre o que realmente conseguimos produzir. Sendo também uma área que não foi abordada durante o nosso percurso académico e devido à escassez de exemplos levou-nos a ter bastantes dificuldades a concluir este projeto. Achamos que poderíamos adquirir bastante experiência e portfolio, caso conseguíssemos concluir na totalidade o projeto. Futuramente, o projeto pode vir a ganhar muito mais, se se conseguir implementar a arquitetura inicialmente pensada e portanto, estamos a pensar continuar o projeto no futuro, caso nós nos encontremos na disponibilidade assim o terminar em continuação da orientação que nos foi fornecida, mas de qualquer das formas deixamos as linhas precursoras para poder ser retomado no futuro por outros alunos.

Calendário

24/11/2019 – 14/01/2019	Aprofundar o conhecimento para as linguagens/ferramentas a ser usadas no desenvolvimento do TFC.
15/01/2020 – 07/02/2020	Desenhar a Arquitetura esperada para a Base de Dados e planear a conexão entre <i>FRONT-END</i> , <i>BACK-END</i> e <i>DB</i> .
08/05/2020 – 21/05/2020	Mudança de uma Base de Dados tradicional para o uso de <i>XML</i> como Base de Dados.
22/05/2020 – 23/06/2020	Desenvolvimento do <i>FRONT-END</i> e <i>BACK-END</i> do Projecto.
23/06/2020 – 26/06/2020	Realização do relatório final e revisão do orientador.

Bibliografia

Internetinnovation.[Online]. Available:

<https://www.internetinnovation.com.br/blog/entenda-o-conceito-da-web-3-0/>

Ex2.[Online]. Available: <https://ex2.com.br/blog/web-1-0-web-2-0-e-web-3-0-enfim-o-que-e-isso/>

Medium.[Online]. Available: <https://medium.com/@joazarate/por-que-a-web-3-0-%C3%A9-importante-e-voc%C3%AA-deveria-saber-sobre-isso-9eb02f003e75>

Wikipedia.[Online]. Available:

https://pt.wikipedia.org/wiki/Resource_Description_Framework

OrganicaDigital.[Online]. Available: <https://www.organicadigital.com/blog/o-que-e-web-semantica/>

Wikipedia.[Online]. Available: <https://pt.wikipedia.org/wiki/XML>

W3.[Online]. Available: <https://www.w3.org/XML/>

sqlDBM[Online]. Available: <https://sqldbm.com/Home/>

Anexos

Link referente ao Projecto e Anexos: [Google Drive Projecto e Anexos](#)

Calendário referente ao planeamento do TFC:

1	CALENDÁRIO TFC 2019/2020		
2	INICIO	FIM	OBJECTIVO:
3	24/11/2019	14/01/2019	Aprofundar o conhecimento para as linguagens/ferramentas a ser usadas no desenvolvimento do TFC.
4	15/01/2020	07/02/2020	Desenhar a Arquitetura esperada para a Base de Dados e planear a conexão entre FRONT-END, BACK-END e DB.
5	08/02/2020	21/02/2020	Escrever o código referente à base de Dados.
6	22/02/2020	01/05/2020	Desenvolver a Front-End e Back-End do Projecto.
7	01/05/2020	28/05/2020	Escrever o Relatório Final e Ser revisto pelo Orientador.
8			

Anexo 1- calendarioTFC.xlsx

Identificação					
Requisito #	1				
Requisitante	Utilizador				
Descrição	Registo do utilizador na aplicação Para que o utilizador consiga registar-se é necessário o preenchimento de um formulário: -> Email; -> Password; -> Nº Telemóvel; -> Nome Completo; -> Localidade.				
Objectivo	Permite aos utilizadores efetuarem serviços na aplicação				
Grupo/Classificador	Utilizador				
Test Case	Registo do utilizador: 1. São introduzidos todos os dados necessários -> O registo do utilizador é realizado 2. São introduzidos valores inválidos / falta do preenchimento de todos os campos obrigatórios -> O registo do utilizador não é realizado				
Prioridade	<input type="checkbox"/> Indispensável (Incluir em impressão crítica; omissão pode representar perda de valor relevante para a avaliação e/ou incumprimento de requisitos normativos) MUST DO <input checked="" type="checkbox"/> Necessário (Incluir importante; omissão representa perda de valor para a avaliação, embora não crítica) Should Do <input type="checkbox"/> Desejável (Facilitar a operação, mas não mandatória; com omissão parcial, omissão pode implicar perda de valor para a avaliação) Could Do <input type="checkbox"/> Opcional (Fator de ajuste; omissão não ocorre e não representa perda de valor para a avaliação) Nice to Do <input type="checkbox"/> Indesejado (Referência para a ação, indica o contexto em que a avaliação não deve ocorrer para que se consiga obter o melhor resultado possível) MUST NOT DO				
Descrição Use Case (ações & Processos)					
Pressupostos (incluir contextualização)					
N/A					
Processo					
#	Ação	Atores	Regra	Domínio	valido
1	Apresentação do Formulário		N/A		<input type="checkbox"/>
2	Preencher Formulário de Publicação	Utilizador	N/A		<input type="checkbox"/>
3	Validação do Formulário		N/A	Controlo da aplicação	<input type="checkbox"/>
4.1	Utilizador é registado		Formulário validado		<input type="checkbox"/>
4.2	Dados introduzidos são invalidados		O formulário não apresenta dados validos		<input type="checkbox"/>
5	Retorna a 2		N/A		<input type="checkbox"/>

Anexo 2 - TFC1920_P02Req_21702113;21704574.xlsx (Requirement 1)

Identificação																																																	
Requisito #	2																																																
Requisitante	Utilizador																																																
Descrição	<p>Login do Utilizador</p> <p>Para o utilizador aceder ao seu perfil é necessário fazer o login preenchendo os seguintes campos:</p> <p>-> Email;</p> <p>-> Password.</p>																																																
Objectivo	Permite ao utilizador aceder ao seu perfil, fazer pesquisas de serviços e reservar um serviço.																																																
Grupo/Classificador	Utilizador																																																
Test Case	<p>Login do utilizador:</p> <p>1. São introduzidos todos os dados necessários -> <input type="checkbox"/> login do utilizador é realizado</p> <p>2. São introduzidos valores inválidos / falta do preenchimento de todos os campos obrigatórios -> <input type="checkbox"/> login do utilizador não é realizado</p>																																																
Prioridade	<input type="checkbox"/> Indispensável	(Inclusão imprescindível, a omissão poderá representar perda de valor relevante para a solução e/ou incumprimento regulamentar ou normativo) MUST DO																																															
	<input checked="" type="checkbox"/> Necessário	(Inclusão importante, a omissão representa perdas de valor para a solução, embora não críticas) Should Do																																															
	<input type="checkbox"/> Desejável	(Facilitador de operação, mas não mandatório; com enquadramento parcial, a omissão pode implicar perdas de valor para a solução) Could Do																																															
	<input type="checkbox"/> Opcional	(Factores de ajuste, a omissão não acarreta perdas relevantes do valor global da solução) Nice to Do																																															
	<input type="checkbox"/> Indesejado	(Referência por negação, indica características que a solução não deve ter; serve para gerir conflitos de requisitos enumerados por stakeholders distintos) MUST NOT DO																																															
Descrição Use Case (ações & Processos)																																																	
<p>Pressupostos (incluir contextualização)</p> <p>1. Utilizador encontra-se registado.</p>																																																	
<p>Processo</p> <table border="1"> <thead> <tr> <th>#</th> <th>Ação</th> <th>Actores</th> <th>Regra</th> <th>Domínio</th> <th>valido</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Apresenta o formulário de registo</td> <td></td> <td>N/A</td> <td></td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>2</td> <td>Introduz os dados de registo</td> <td>Utilizador</td> <td>N/A</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>3</td> <td>Valida os dados</td> <td></td> <td>N/A</td> <td>Controlo de aplicação</td> <td></td> </tr> <tr> <td>4.1</td> <td>Dados introduzidos são validados</td> <td></td> <td>Os dados introduzidos são válidos ao login</td> <td></td> <td></td> </tr> <tr> <td>4.2</td> <td>Dados introduzidos são invalidados</td> <td></td> <td>Os dados introduzidos são inválidos ao login</td> <td></td> <td></td> </tr> <tr> <td>5.2</td> <td>Retorna a 2ª</td> <td></td> <td>N/A</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>		#	Ação	Actores	Regra	Domínio	valido	1	Apresenta o formulário de registo		N/A		<input checked="" type="checkbox"/>	2	Introduz os dados de registo	Utilizador	N/A		<input type="checkbox"/>	3	Valida os dados		N/A	Controlo de aplicação		4.1	Dados introduzidos são validados		Os dados introduzidos são válidos ao login			4.2	Dados introduzidos são invalidados		Os dados introduzidos são inválidos ao login			5.2	Retorna a 2ª		N/A								<input type="checkbox"/>
#	Ação	Actores	Regra	Domínio	valido																																												
1	Apresenta o formulário de registo		N/A		<input checked="" type="checkbox"/>																																												
2	Introduz os dados de registo	Utilizador	N/A		<input type="checkbox"/>																																												
3	Valida os dados		N/A	Controlo de aplicação																																													
4.1	Dados introduzidos são validados		Os dados introduzidos são válidos ao login																																														
4.2	Dados introduzidos são invalidados		Os dados introduzidos são inválidos ao login																																														
5.2	Retorna a 2ª		N/A																																														
					<input type="checkbox"/>																																												

Anexo 3- TFC1920_P02Req_21702113;21704574.xlsx (Requirement 2)

Identificação																																											
Requisito #	3																																										
Requisitante	Utilizador																																										
Descrição	Pesquisar de serviços, podendo o utilizador reservar ou não um serviço.																																										
Objectivo	Permite diferenciar os serviços através das suas características tais como origem, destino, preço, data de partida e de chegada e número de passageiros de um voo, nome ou destino de um hotel tal como as datas de check-in e check-out, número de quartos e hóspedes, o nome de um restaurante, cidade, atração ou atividade.																																										
Grupo/Classificador	Utilizador																																										
Test Case	<p>Apresentação de resultados de pesquisa:</p> <p>1. verificação da existência de key word inserida.</p> <p>2. verificação da existência dos voos, restaurantes e hotéis apresentados.</p> <p>3. verificação de detalhes informativos.</p>																																										
Prioridade	<input type="checkbox"/> Indispensável	(Inclusão imprescindível, a omissão poderá representar perda de valor relevante para a solução e/ou incumprimento regulamentar ou normativo) MUST DO																																									
	<input type="checkbox"/> Necessário	(Inclusão importante, a omissão representa perdas de valor para a solução, embora não críticas) Should Do																																									
	<input type="checkbox"/> Desejável	(Facilitador de operação, mas não mandatório; com enquadramento parcial, a omissão pode implicar perdas de valor para a solução) Could Do																																									
	<input checked="" type="checkbox"/> Opcional	(Factores de ajuste, a omissão não acarreta perdas relevantes do valor global da solução) Nice to Do																																									
	<input type="checkbox"/> Indesejado	(Referência por negação, indica características que a solução não deve ter; serve para gerir conflitos de requisitos enumerados por stakeholders distintos) MUST NOT DO																																									
Descrição Use Case (ações & Processos)																																											
<p>Pressupostos (incluir contextualização)</p> <p>N/A;</p>																																											
<p>Processo</p> <table border="1"> <thead> <tr> <th>#</th> <th>Ação</th> <th>Actores</th> <th>Regra</th> <th>Domínio</th> <th>valido</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Apresentação formulário de pesquisa de serviço</td> <td></td> <td>N/A</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>2</td> <td>Pesquisar serviço</td> <td>Utilizador</td> <td>N/A</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>3</td> <td>Validação da pesquisa</td> <td></td> <td>N/A</td> <td>Controlo de serviços</td> <td><input type="checkbox"/></td> </tr> <tr> <td>4.1</td> <td>Apresentação de serviços pesquisados</td> <td></td> <td>Pesquisa de serviço validada</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>4.2</td> <td>Retorna a 2ª</td> <td></td> <td>A reserva não foi validada.</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		#	Ação	Actores	Regra	Domínio	valido	1	Apresentação formulário de pesquisa de serviço		N/A		<input type="checkbox"/>	2	Pesquisar serviço	Utilizador	N/A		<input type="checkbox"/>	3	Validação da pesquisa		N/A	Controlo de serviços	<input type="checkbox"/>	4.1	Apresentação de serviços pesquisados		Pesquisa de serviço validada		<input type="checkbox"/>	4.2	Retorna a 2ª		A reserva não foi validada.								
#	Ação	Actores	Regra	Domínio	valido																																						
1	Apresentação formulário de pesquisa de serviço		N/A		<input type="checkbox"/>																																						
2	Pesquisar serviço	Utilizador	N/A		<input type="checkbox"/>																																						
3	Validação da pesquisa		N/A	Controlo de serviços	<input type="checkbox"/>																																						
4.1	Apresentação de serviços pesquisados		Pesquisa de serviço validada		<input type="checkbox"/>																																						
4.2	Retorna a 2ª		A reserva não foi validada.																																								

Anexo 4- TFC1920_P02Req_21702113;21704574.xlsx (Requirement 3)

