

# **Universidade da Beira Interior**

## **Departamento de Informática**



### ***WebServices* em Sistemas Distribuídos**

Elaborado por:

**Tiago Roxo nº 37032**

**Joana Costa nº 37606**

**Professor Doutor Valderi Leithardt**

**Professora Doutora Alexandra Mendes**

29 de maio de 2019

# Conteúdo

<b>Conteúdo</b>	<b>i</b>
<b>1 Tutorial de Funcionamento</b>	<b>1</b>
<b>2 Modelação de Sistema</b>	<b>5</b>
<b>3 Características do Sistema</b>	<b>9</b>
3.1 Compartilhamento de recursos . . . . .	9
3.2 Heterogeneidade . . . . .	9
3.3 Escalabilidade . . . . .	10
3.4 Transparência . . . . .	11
3.5 Concorrência . . . . .	12
3.6 Tolerância a Falha . . . . .	13
3.7 Segurança . . . . .	14

# Acrónimos

<b>CS</b>	<i>C Sharp</i>
<b>HTTPS</b>	<i>Hypertext Transfer Protocol Secure</i>
<b>IIS</b>	<i>Internet Information Services</i>
<b>IP</b>	<i>Internet Protocol</i>
<b>JSON</b>	<i>JavaScript Object Notation</i>
<b>PBKDF2</b>	<i>Password-Based Key Derivation Function 2</i>
<b>PHP</b>	<i>Hypertext Preprocessor</i>
<b>SQL</b>	<i>Structured Query Language</i>
<b>SSL</b>	<i>Secure Sockets Layer</i>
<b>TCP</b>	<i>Transmission Control Protocol</i>
<b>UBI</b>	Universidade da Beira Interior

# Capítulo 1

## Tutorial de Funcionamento

A interface inicial contém no lado esquerdo da janela, no topo, a secção dedicada à pesquisa por aluno, tendo a opção de pesquisa por número ou nome. A figura 1.1 exibe o menu inicial da aplicação. Pesquisando por número exige a introdução de **a** antes do número, caso seja um número de licenciatura, **m** mestrado e **d** doutoramento. Existe uma expressão regular associada a este campo de modo a garantir a correta introdução de valores para pesquisa. A procura de alunos por nome irá promover uma pesquisa de todos os alunos que contenham o nome inserido, independentemente do grau. Esta pesquisa obriga à procura de informação em todos os servidores e a apresentar a informação dos alunos encontrados, bem como o nome destes (interação com servidor de tipo I).

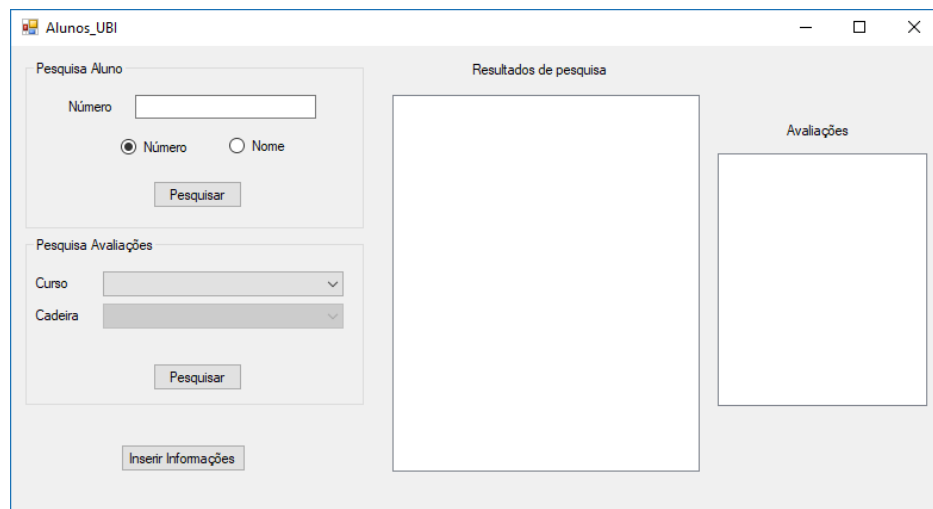


Figura 1.1: Menu inicial de aplicação.

A pesquisa com sucesso de um aluno promove o aparecimento do seu nú-

mero, nome e curso, na `listBox` central; caso a pesquisa resulte em vários alunos, aparecerão tantas linhas quantos os resultados da pesquisa. Para visualizar as diferentes disciplinas de cada aluno, basta fazer duplo clique sobre o nome deste, modificando a `listBox` central para que esta apresente as disciplinas, limpando o conteúdo anteriormente presente nesta (informação de alunos).

Como o propósito deste projeto é a pesquisa de avaliações de alunos, criou-se, do lado direito da janela, uma outra `listBox`, onde serão apresentadas estas. Assumindo que foram encontrados alunos, e foi selecionado um, apresentando as disciplinas deste na `listBox` central, podemos aceder às avaliações desse aluno, numa determinada disciplina, por duplo clique na disciplina que pretendemos verificar. Ao realizar estas ações, serão apresentadas na `listBox` do lado direito da janela, todas as avaliações que cumpram os requisitos anteriormente referidos. Caso o aluno não tenha nenhuma avaliação na disciplina escolhida, será apresentada na `listBox` do lado direito da janela a mensagem "Aluno sem avaliações".

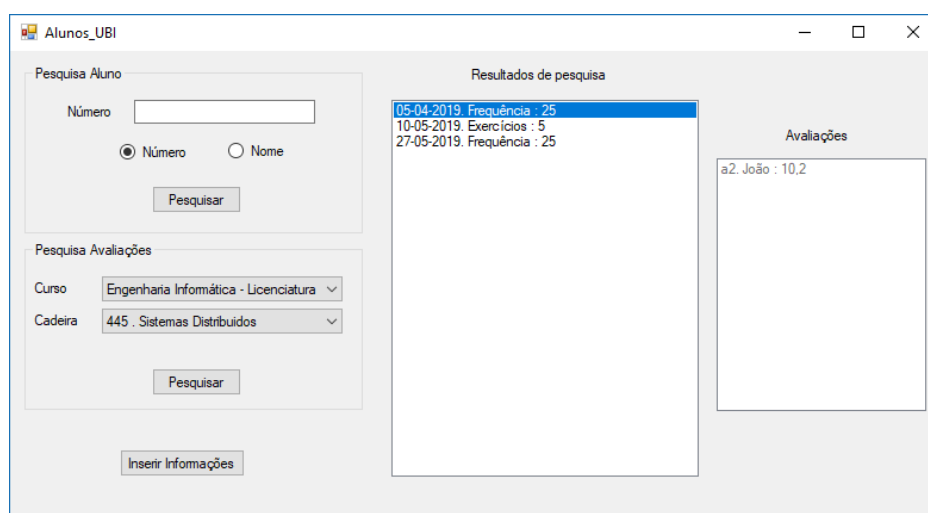


Figura 1.2: Apresentação de notas de avaliações de disciplina.

Alternativamente à pesquisa de alunos, podemos pesquisar por disciplinas, sendo para tal necessário a escolha de um curso. A figura 1.2 exibe um exemplo destes. A escolha de um curso, promove o carregamento de todas as disciplinas deste e a seleção de uma disciplina fará com que a `listBox` central carregue todos os alunos inscritos nessa disciplina. À seleção de um aluno da `listBox` central, por duplo clique neste, será feito o carregamento de todas as avaliações deste, apresentando a informação destas na `listBox` do lado direito da janela. De referir que, aquando do carregamento de variadas informações, o *WebService* associado ao servidor de tipo I irá realizar pedidos a bases de dados de servidores de tipo II, mediante o pedido feito (se o aluno ou curso é de licenciatura ou mestrado, tem

de se pesquisar em servidores diferentes) e mediante a situação dos servidores que contêm as bases de dados (servidor com base de dados de licenciatura pode não estar disponível e portanto temos de procurar esta informação no servidor alternativo).

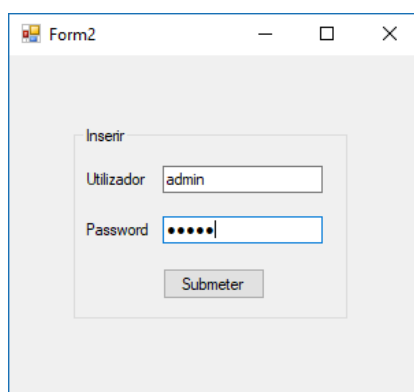
The image shows a web browser window with a title bar that says 'Form2'. Inside the window, there is a form with a title 'Inserir'. Below the title, there are two input fields. The first is labeled 'Utilizador' and contains the text 'admin'. The second is labeled 'Password' and contains seven dots, indicating a masked password. Below these two fields is a button labeled 'Submeter'.

Figura 1.3: Menu de autenticação para inserção de informação em bases de dados.

Ao seleccionar o botão para "Inserir Informações"(visível em 1.1), migramos para um outro menu onde será preciso inserir as credenciais, para que possamos inserir nova informação nas bases de dados. Um exemplo da utilização deste menu pode ser visto na figura 1.3. Ao introduzirmos a *password* correcta podemos evoluir para um terceiro menu, onde teremos, sob a forma de tabulações, todas as inserções possíveis de realizar neste sistema. Podemos inserir, desde pessoa, curso, aluno em cadeira, atribuir nota de aluno em avaliação ou cadeira ou mesmo inserir nova cadeira em curso.

Todas as alterações a realizar exigem a escolha de elementos, por recurso a *comboboxes*, ou a inserção de informação em campos, no caso de criação de novos cursos ou disciplinas. Todas as ações a realizar são intuitivas, pelo que não será descrito um exemplo concreto nesta secção. Apenas será referido que as *comboboxes* de cursos são preenchidas automaticamente e, mediante a escolha de curso do utilizador, as *comboboxes* de disciplinas são também preenchidas automaticamente, de acordo com o curso escolhido. Este preenchimento automático provém da pesquisa do *WebService* associado ao servidor I, que verificará as ligações entre os diferentes servidores de tipo II e fará pedidos a estes para apresentação de informação ao utilizador. Um exemplo de utilização do menu 3, visível em 1.4, permite perceber este preenchimento automático. Neste caso, o curso foi automaticamente preenchido, após seleção de aluno, e as cadeiras e avaliações, associadas à escolha de cadeira, são preenchidas automaticamente em *comboboxes*, aquando da escolha do utilizador. O método de seleção de alunos neste menu, segue a mesma lógica do menu inicial da aplicação (duplo clique).

The screenshot shows a web application window titled "Form3" with a tabbed interface. The active tab is "NotaAlunoAvaliacao". The form contains the following fields and controls:

- Nome de Aluno:** A text input field containing the letter "o". Below it is a "Pesquisar" button.
- Curso:** A dropdown menu currently showing "Engenharia Informática - Licenciatura".
- Cadeira:** A dropdown menu currently showing "Sistemas Distribuidos".
- Avaliação:** A dropdown menu currently showing "05-04-2019. Frequência : 25".
- Nota:** A text input field containing the value "15".
- Search Results:** A list box below the "Nome de Aluno" field displays the following items:
  - a2. João Engenharia Informática - Licenciatura
  - a3. Bernardo Informática Web - Licenciatura
  - d1. José Biomédicas - Doutoramento
- Buttons:** At the bottom of the form are two buttons: "Confirmar" and "Limpar".

Figura 1.4: Exemplo de utilização de menu de inserção de informação em bases de dados.

## Capítulo 2

# Modelação de Sistema

O trabalho começou a ser desenvolvido na modelação da base de dados a usar. De acordo com os requisitos do enunciado, e tendo em consideração a nossa experiência enquanto alunos da Universidade da Beira Interior (UBI), o modelo entidade-associação de base de dados da figura 2.1 foi o usado no desenvolvimento do projeto.

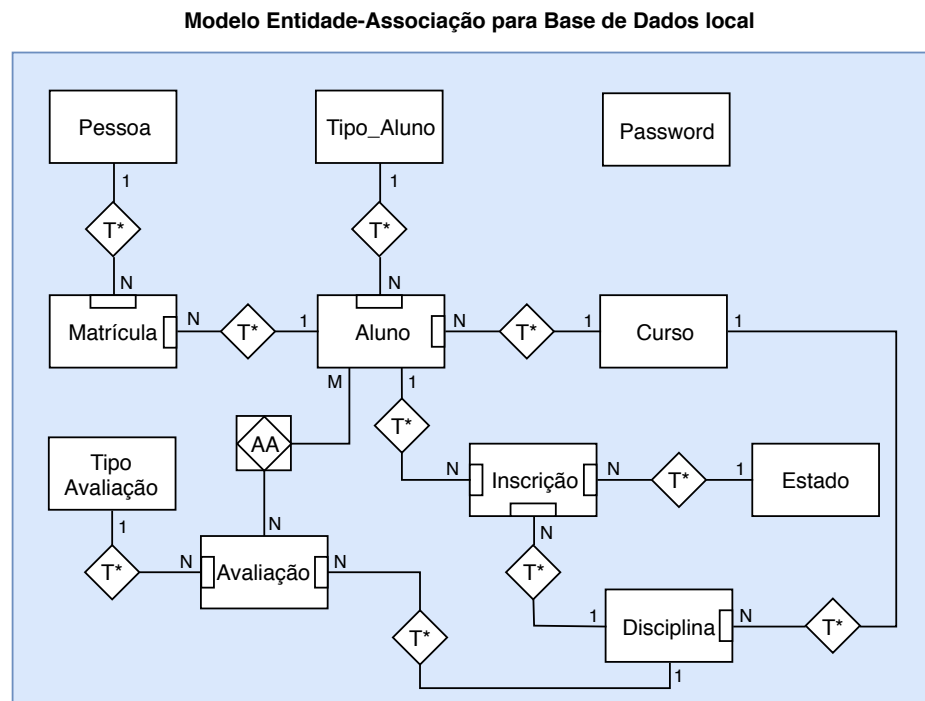


Figura 2.1: Modelação entidade-associação de base de dados usando apenas um servidor.



A figura 2.1 contém as principais relações entre as várias entidades consideradas, diferenciando informações de pessoas, alunos, disciplinas, cursos, avaliações nas tabelas respetivas. Todas as tabelas exibem a relação esperada entre as entidades, sendo no entanto salientadas de seguida aquelas com maior interesse.

**Matricula** relaciona a informação entre pessoa e aluno, ou seja, associa o identificador de aluno à pessoa em causa, e **Password** contém as *passwords* associadas a contas de utilizadores com privilégios para modificar a base de dados. É evidente que ambas as tabelas referidas, bem como **Pessoa**, desempenham um papel que não se relaciona diretamente com a universidade, sendo por isso passíveis de serem separadas das restantes tabelas. Esta consideração será a base do ajuste deste modelo na implementação de bases de dados num sistema distribuído, como é visível em 2.2.

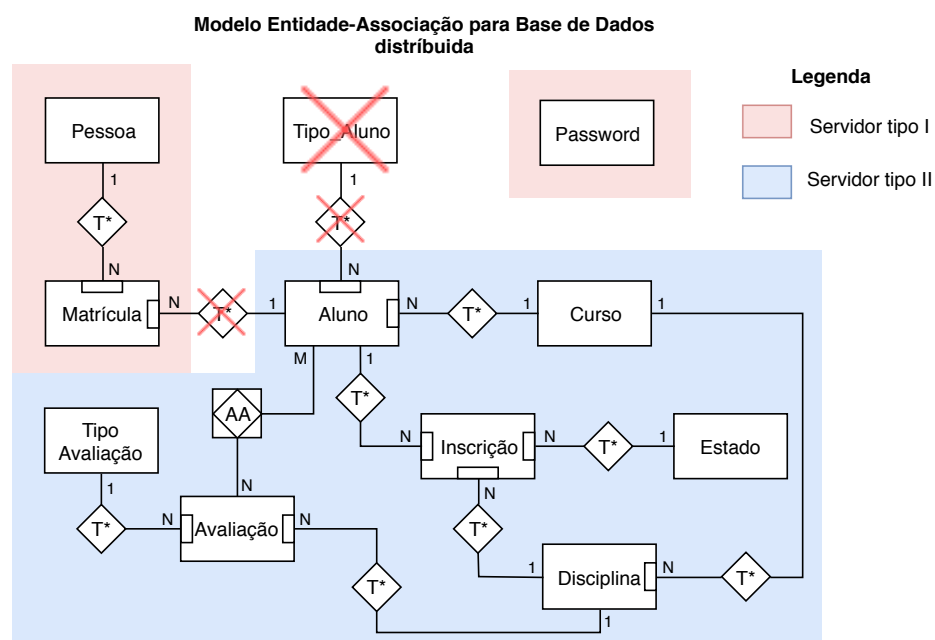


Figura 2.2: Modelação entidade-associação de base de dados num sistema distribuído.

Na figura 2.2 é possível observar as diferenças entre a modelação de base de dados local e a de num sistema distribuído. Primeiramente dividiram-se as bases de dados em dois tipos: I e II. A de tipo I é a responsável por conter informação de *passwords* de contas e informação de pessoas. Ambas estas informações não estão diretamente relacionadas com a universidade logo farão parte de uma base de dados num servidor dedicado a esta informação. As restantes tabelas na região azul, farão parte de um tipo de base de dados que será replicada em diferentes servidores, contendo informação de licenciatura, mestrado e doutoramento.

As tabelas presentes nesta região partilham características comuns aos diferentes graus de curso, razão pela qual são o modelo a usar nestes servidores (de tipo II).

A eliminação de tabelas, aquando da transição de modelo de base de dados local para distribuída, é evidenciada pelas cruzes vermelhas. Essencialmente, tabelas que indiquem qual o tipo de aluno (se de licenciatura, mestrado ou doutoramento) ou que relacionem alunos com pessoas são descartáveis. A razão para tal prende-se no facto de informações sobre alunos de licenciatura (por exemplo) estarem em servidores e base de dados específicos e portanto não é necessário, dentro da base de dados, identificar o aluno pelo seu grau; sabemos se é de licenciatura mediante a base de dados ou servidor com o qual estamos a comunicar.

Apresentada a modelação dos dois tipos de servidores utilizados (diferenciados pelo modelo de base de dados usado, tipo I ou II), é possível exibir, com maior clareza, o esquema do sistema distribuído implementado. Na figura 2.3 é apresentado esse esquema, com apresentação de legenda dos símbolos utilizados nesta, do lado direito.

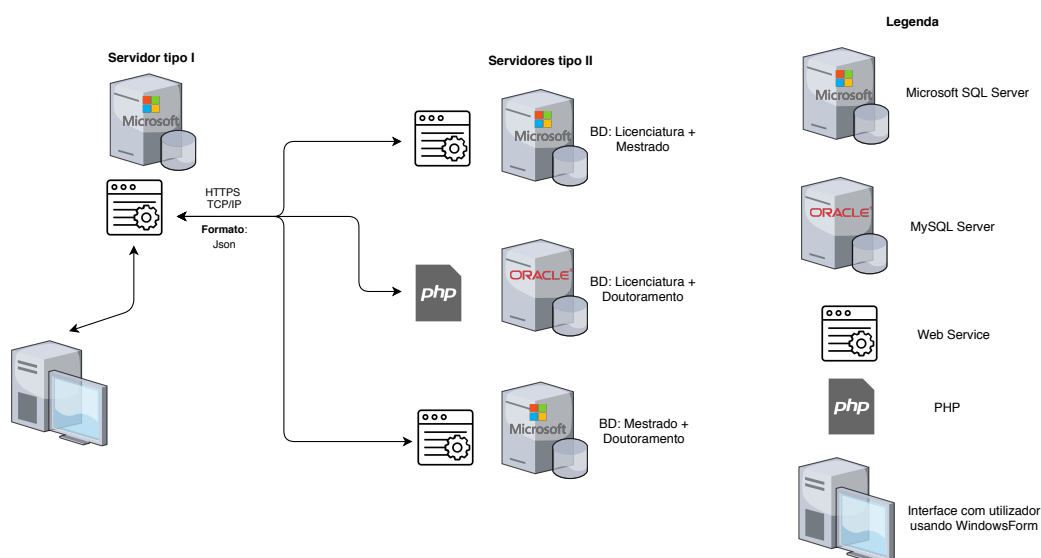


Figura 2.3: Esquema do sistema distribuído implementado.

O esquema apresentado demonstra a interação entre a interface (usando *WindowsForm*), responsável por receber os *inputs* do utilizador e os encaminhar para o *WebService*, associado ao servidor de tipo I. Este *WebService* será o responsável, em primeira instância, de conferir se a *password* introduzida confere com uma presente na base de dados do servidor, e desempenhará também funções de associar informação de pessoas, caso o pedido do utilizador assim exija. Um exemplo deste caso é se o utilizador pesquisar por número de aluno; o sistema deverá apresentar o nome (informação de pessoa), número e curso (informação de

aluno/curso) resultantes da pesquisa. De referir que, tal como o enunciado exigia, o sistema comporta a capacidade de distinguir entre utilizador com privilégios de alteração de dados (têm *password* presente na base de dados de servidor de tipo I) e os restantes utilizadores, apenas capacitados para consultar informação.

O *WebService* associado ao servidor de tipo I terá também a responsabilidade de encaminhar os pedidos de informação para os servidores correspondentes. Para simplicidade de denominação de servidores de tipo II, denominaremos o primeiro servidor com *WebService*, *Structured Query Language* (SQL) *Server* e base de dados contendo informação de Licenciatura e Mestrado como  $\alpha$ , o imediatamente abaixo deste com *Hypertext Preprocessor* (PHP), *MySQL* como  $\beta$  e o último servidor de tipo II, abaixo de  $\beta$ , como  $\gamma$ . Caso o pedido de utilizador seja respeitante a alunos/cursos de licenciatura, este *WebService* terá de comunicar com o  $\alpha$  ou  $\beta$ . Será dada preferência a  $\beta$  e, caso este esteja indisponível, será contactado  $\alpha$ . De forma análoga, funcionam os pedidos para os restantes graus. A informação recolhida, pelo *WebService* associado ao servidor de tipo I, será organizada e apresentada ao utilizador, tornando transparente a este toda a distributividade existente no sistema.

Outra característica deste sistema, associada à distributividade das bases de dados, é a necessidade de garantir consistência nestas aquando da alteração de informação. Visando manter a consistência será feita, pelo *WebService* associado ao servidor de tipo I, uma verificação de conexão entre os servidores envolvidos, aquando da inserção/alteração de dados. A título de exemplo, se o utilizador quiser introduzir um novo curso de Licenciatura, esta introdução deverá ocorrer em  $\alpha$  e  $\beta$ . Para garantir que tal ocorre, antes de inserir, *WebService* associado ao servidor de tipo I, fará ligação a ambos os servidores e, apenas no caso de ambos responderem, fará o pedido de inserção do novo curso em ambas as bases de dados associadas aos servidores.

Toda a comunicação do sistema é feita em *Hypertext Transfer Protocol Secure* (HTTPS) sob *Transmission Control Protocol* (TCP) sob *Internet Protocol* (IP), e o formato de dados usado, aquando de comunicação entre *WebServices* e PHP, foi *JavaScript Object Notation* (JSON).

## Capítulo 3

# Características do Sistema

O trabalho desenvolvido compreendeu todas as características de sistemas distribuídos, propostas no enunciado do trabalho prático. Nas subsecções seguintes será feita a descrição das medidas implementadas que contribuíram para que essas características estivessem presentes neste projeto.

### 3.1 Compartilhamento de recursos

Segundo a definição de partilha de recursos, esta característica está associada à ideia da existência de recursos, num sistema distribuído, que pode ser acedido, localmente ou de forma remota, a partir de variados domínios. Neste sentido, a existência das diferentes bases de dados, em diferentes servidores, acedíveis a partir de diferentes vias (comunicação entre *WebServices* ou por PHP) evidencia a existência de compartilhamento de recursos no sistema desenvolvido.

A comunicação com *WebService*, situados em diferentes servidores, pode ser também analisada do ponto de vista do compartilhamento de recursos. Ao pedido de informação, usando a interface, o sistema terá que aceder às diferentes bases de dados, usando como intermediários os *WebService* e PHP. Neste sentido, não só as bases de dados são compartilhadas entre todo o sistema, mas também as vias de acesso a estas, ou seja, *WebService* e PHP.

### 3.2 Heterogeneidade

Uma evidência da existência de heterogeneidade neste trabalho é o facto de este contemplar a comunicação entre servidores, contendo diferentes tipos de bases de dados. Tendo a figura 2.3 como referência, podemos observar que existem dois tipos de bases de dados existentes, *MySQL* e *SQL Server*, usados em diferentes servidores.

O facto de terem sido usados quatro servidores distintos, em quatro computadores distintos, usando diferentes sistemas operativos e arquiteturas é outra evidência da variabilidade de componentes no sistema e, consequentemente, heterogeneidade. Os sistemas operativos diferentes no sistema são *Windows* 8, 10 e *Ubuntu* 18.04. Estes sistemas operativos são, respetivamente, os dos servidores  $\alpha$ ,  $\gamma$  e  $\beta$ , seguindo a definição de servidores, no parágrafo abaixo da figura 2.3.

Outra demonstração da heterogeneidade existente é o uso de diferentes tecnologias de comunicação com base de dados. No caso de comunicação com servidores de base de dados *MySQL* é usado PHP. Se considerarmos servidores com *SQL Server* a tecnologia usada é um *WebService*, criados usando a linguagem *C Sharp* (CS).

O uso de vários *WebServices* no sistema, implementados em CS, e o facto de a realização da comunicação com base de dados *MySQL* ser feita por PHP demonstra o uso de diferentes linguagens neste projeto. Este aspecto exhibe a variabilidade de linguagens no sistema, bem como mecanismos de tradução associados à comunicação entre estas, reforçando a presença de heterogeneidade. O uso de um formato JSON, comum à comunicação entre todos os *WebServices* e PHP's, evidencia o mecanismo de tradução usado para servir de comunicação entre os diferentes componentes do sistema.

### 3.3 Escalabilidade

A modelação de base de dados neste projeto permite que a inserção de novos componentes (como novos tipos de avaliação, novos tipos de cursos, entre outros) seja facilmente conseguido, por recurso a inserção de novas linhas nas tabelas respetivas nas bases de dados. Caso se quisesse incorporar este sistema para ser associado a outra universidade (cenário não contemplado pelo enunciado), tal seria possível por inserção de uma nova tabela e acréscimo de dois atributos em tabelas de *Aluno* e *Curso*. Este cenário demonstra que, mesmo para casos para o qual o sistema não foi projetado, a sua modificação para contemplar estes é possível sem necessidade de reformulação total do sistema.

Outro exemplo de escalabilidade de sistema é evidenciado aquando da introdução de um novo servidor. Da maneira como o sistema foi projetado é fácil a incorporação das funcionalidades desenhadas neste trabalho neste novo servidor. Assumindo que estamos a considerar servidores do tipo II, no caso de ser *SQL Server*, podemos reutilizar todo o *WebService* desenhado, e no caso de *MySQL*, podemos reaproveitar os PHP's criados. Em ambas as situações as únicas alterações a incorporar seria o IP do novo servidor e o nome da(s) base(s) de dados deste; ambas estas informações estão associadas a variáveis de classes, para maior comodidade na sua alteração. Assumindo que estamos a considerar servi-

dores do tipo I, bastaria reutilizar código do *WebService* correspondente (ao tipo I) e associar um novo *WebService* (criado para ficar associado ao novo servidor). Relativamente à inserção de bases de dados num novo servidor, assumindo que estas seguem a mesma modelação do sistema atual, seria facilmente exequível por recurso a *scripts*, desenhados pelo grupo, visando uma maior facilidade de implementações futuras, o que confere maior escalabilidade ao sistema.

De forma análoga à introdução de um novo servidor poderíamos pensar na introdução de um outro *WebService* ou PHP, associado a um servidor já existente. Em ambos os casos seria possível fazer uso da reutilização de código (nos *WebService* e PHP já existentes), fazendo apenas alterações pontuais para cumprir os objetivos para os quais estes novos *WebService*/PHP foram projetados. Esta evidência demonstra a capacidade do sistema em lidar com a introdução de novas funcionalidades para as quais não foi inicialmente projetada.

## 3.4 Transparência

A noção de transparência neste projeto está presente na ideia em que existem diferentes *WebServices* a comunicarem entre si e a troca de informação entre os vários acontece sem que cada um saiba como os restantes operam, apenas necessitando saber que valores retornam. Uma evidência disto é um *WebService*, associado ao servidor do tipo I, pedir informações a outros relativamente a informação de licenciatura e esperar pelo resultado para o apresentar à interface do utilizador. Para este *WebService* é-lhe transparente como a base de dados de licenciatura está organizada e como o *WebService* associado a esse servidor opera. Apenas tem de saber que, se entregar a informação necessária, ser-lhe-á devolvida a informação correta. A título de exemplo, caso queira saber quais as cadeiras de um determinado curso de licenciatura, o *WebService* associado ao servidor de tipo I apenas necessita de enviar o identificador do curso para um servidor de licenciatura, recebendo como respostas todas as cadeiras associadas.

Outra noção de transparência encontra-se associada à interface usada. Para o utilizador, quando pesquisa pelo nome de alunos da UBI, é-lhe transparente onde a informação é recolhida. Neste exemplo, o servidor do tipo I irá procurar todos os nomes que contenham as palavras introduzidas pelo utilizador e relacionar os resultados obtidos com o número de aluno e os cursos de cada um. Naturalmente, informações de alunos e cursos apenas se encontram em servidores do tipo II, podendo os diferentes resultados ter informações presentes em mais que um servidor; no caso de pesquisarmos por "Maria", iremos encontrar o número de aluno de licenciatura e de mestrado desta, o que implica recolher informação de dois servidores, do tipo II, distintos. Embora todo este processo decorra em bastidores, para o utilizador a informação é-lhe apresentada toda em simultâneo, ocultando a

este toda a distributividade envolvida no processo de recolha de informação.

De modo a tornar transparente para o utilizador a queda de um servidor e a tomada de comando do servidor alternativo, optou-se por delimitar de conexão em 1000 milissegundos, entre *WebServices* e servidores. Deste modo, quando a informação for apresentada ao utilizador, este não precisa de se aperceber que o servidor eleito para apresentar informação de licenciatura, por exemplo, não foi possível de ser acedido, tendo o sistema recorrido ao servidor alternativo para tal; neste exemplo, o servidor  $\beta$  teria ficado indisponível e o  $\alpha$  tinha assumido o seu papel. Caso não tivesse sido delimitado este tempo máximo, seria perceptível a queda de um servidor, pois a interface demoraria demasiado tempo a responder.

## 3.5 Concorrência

Coulouris define concorrência em sistema distribuídos como o acesso a serviços ou aplicações em simultâneo, por clientes. Assim, faz sentido observar a concorrência existente neste sistema a nível dos recursos partilhados neste. O principal elemento, no sistema deste projeto, onde a concorrência tem de ser controlada é a nível dos *WebServices*, nomeadamente o associado ao servidor de tipo I, como exibido na figura 2.3.

Este *WebService* será o ponto de contacto entre a interface e os diferentes *Web-Service*, associados a servidores de tipo II. Para tal recorreu-se à documentação *Microsoft* para verificar se existia alguma maneira de gerir o acesso concorrente a *WebServices*. Descobrimos que, ao adicionar, no cabeçalho do ficheiro de *Web-Service*, uma definição de como se comportar, a nível da concorrência, este se comportaria de forma a que nenhum cliente ficasse em espera, nem nenhum pedido ficasse por executar, aquando de acesso concorrente.

```
[ ServiceBehavior(ConcurrencyMode=ConcurrencyMode.Multiple ,  
    InstanceContextMode = InstanceContextMode.Single ) ]  
  
public class ServicoWeb : System.Web.Services.WebService{  
    ...  
}
```

Listing 3.1: Trecho de código introduzido para promover a gestão de concorrência.

Com a introdução do trecho de código 3.1 e, segundo a documentação *Microsoft* (<https://docs.microsoft.com/en-us/dotnet/framework/wcf/samples/concurrency>), cada instância de *WebService* fica capacitada para processar múltiplas mensagens, de forma concorrente. Por defeito, *WebService* teria o modo de concorrência *Single*, o que implicaria que cada *WebService* apenas processaria uma mensagem de cada vez. Com esta mudança, visamos uma melhoria da concorrência do sistema.

A concorrência do sistema pôde ser testada usando a interface do utilizador (usando *WindowsForm*, visível em 2.3) e um *WebService* associado a um servidor de tipo II, usando *Internet Information Services* (IIS). À inserção simultânea de um novo curso, em ambos, verificou-se o *feedback* esperado, bem como a inserção deste de forma correta. A concorrência aqui testada foi o acesso ao mesmo *WebService* e aos métodos deste. Segundo os testes realizados e a documentação *Microsoft*, temos razões para acreditar que esta abordagem melhora o tratamento de concorrência no sistema desenvolvido. Esta modificação de comportamento de *WebService* foi incorporado em todos os *WebServices* presentes no sistema.

### 3.6 Tolerância a Falha

A tolerância a falhas está exibida sob a forma de replicação de base de dados entre os servidores de tipo II. Foram usados três servidores deste tipo, estando no servidor  $\alpha$ , base de dados de licenciatura e mestrado, no servidor  $\beta$  base de dados de licenciatura e doutoramento e no servidor  $\gamma$  base de dados de mestrado e doutoramento, tendo como referência a imagem 2.3. Por recurso a replicação de informação conseguimos que o sistema desenvolvido tenha sempre uma alternativa de consulta de informação, caso um servidor não esteja disponível. Como exemplo, caso o servidor  $\beta$  esteja em baixo, podemos sempre usar  $\alpha$  para consulta de informação relativa a licenciatura. Seguindo o mesmo exemplo, como um servidor assume o papel de outro, em caso de indisponibilidade, estamos a considerar também a replicação de hardware, outra das medidas de tolerância a falhas.

Com o uso de bases de dados em sistemas distribuídos temos de garantir que estas se mantenham coerentes ao longo das variadas inserções realizadas pelo utilizador. Como a mesma informação se encontra sempre em dois servidores diferentes, terá que ser garantido que a inserção numa base de dados de um servidor ocorre sempre que também possa ocorrer no outro. Para tal, antes de realizar quaisquer inserções é verificada a conexão entre ambos os servidores e, caso ambos estejam acedíveis, promove-se às respetivas alterações nas bases de dados. Deste modo garantimos que estas se mantêm consistentes o que por si só é uma abordagem de tolerância a falhas. Estamos a evitar que uma falha (inconsistência) se desenvolva num erro e, consequentemente, num defeito. As decisões tomadas visam evitar o aparecimento de incoerências, o que potencialmente poderia gerar conflitos e corromper tanto as bases de dados como a interface.



## 3.7 Segurança

Esta característica de sistemas distribuídos encontra-se presente neste trabalho, de forma natural, pela implementação de requisitos do enunciado. O facto de ser obrigatório o sistema ter dois tipos de utilizadores distintos, com diferenças relativamente ao leque de ações a realizar (utilizador do tipo `admin` tem poder para alterar bases de dados, enquanto que `aluno` apenas pode consultar dados), exige a presença de uma tabela, em base de dados, dedicada para verificar se a inserção de um utilizador, e respetiva *password*, têm permissão para entrar na interface dedicada à modificação de dados. Esta medida em si, evidencia a presença de segurança no sistema projetado pois avalia o perfil de utilizador e apenas disponibiliza as ações de modificação de base de dados caso este tenha permissões para tal. Aliado a esta avaliação de permissão está, implicitamente, a comparação entre *password* inserida pelo utilizador e a presente na base de dados. Para complementar a noção de segurança do sistema, ao invés de ser guardada a *password* em forma de texto na base de dados, guardou-se o resultado da *Password-Based Key Derivation Function 2* (PBKDF2) desta, usando 10.000 iterações, em conjunção com sal de 32 bits (aleatoriamente criado); a informação do sal usado foi também guardada em base de dados, para uso de validação de *password*.

O uso de *WebServices* em *Windows* implicou a criação de uma diretoria virtual em IIS, onde o *WebService* foi colocado para que fosse remotamente acedido. Usando este serviço, definiu-se que o acesso ao *WebServices* apenas seria possível por HTTPS, exigindo também que a comunicação fosse feita por *Secure Sockets Layer* (SSL), conferindo maior segurança na comunicação entre os diferentes *WebServices* utilizados no sistema.

Uma medida adicional de segurança, associada à gestão de base de dados foi o facto de PHP's verificarem qual o tipo de pedido, apenas executando o código caso o tipo requisitado seja concordante com o tipo esperado. A título de exemplo, caso o PHP seja responsável por devolver as cadeiras de um curso, é esperado que lhe seja dado a informação do identificador do curso sendo, portanto, um pedido do tipo `POST`; caso alguém tentasse comunicar com este PHP por pedido `GET` (simulação de possível ataque, visando obter informação da base de dados), tal não responderia. Também associado à gestão de base de dados, tanto em PHP como nos *WebServices*, todas as variáveis, aquando da execução de uma *query* SQL, são introduzidas sob a forma de parâmetros. Esta medida visa prevenir a injeção de código SQL (possível ataque), promovendo maior segurança a nível da gestão da base de dados.