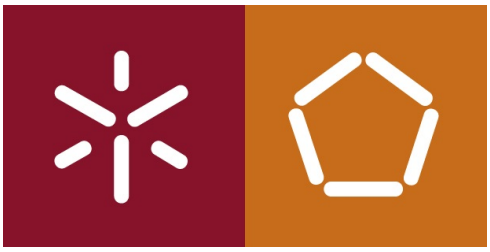

Universidade do Minho
Mestrado Integrado em Engenharia Informática
4ºano - 2º Semestre

Interoperabilidade Semântica

Ficha 2

Elemento do Grupo 01

A83732 – Gonçalo Rodrigues Pinto

30 de abril de 2021

Conteúdo

Lista de Figuras

1 Introdução

No 2º semestre do 1º ano do Mestrado em Engenharia Informática da Universidade do Minho, existe uma unidade curricular enquadrada no perfil Engenharia do Conhecimento denominada por Interoperabilidade Semântica, que tem como objetivo a introdução ao conceito de interoperabilidade semântica.

A Interoperabilidade Semântica é um termo essencial para qualquer sistema, a interoperabilidade é a capacidade de um sistema de comunicar de forma transparente com outro sistema, a semântica é a disciplina da linguística que se ocupa do estudo do significado das expressões linguísticas, bem como, das relações de significado que essas expressões estabelecem entre si e com o mundo. Assim, podemos concluir que a interoperabilidade semântica, nomeadamente na área da informática, é a capacidade que um sistema informático deve ter para comunicar consoante uma lista de regras semânticas de forma a ser possível a troca de informação com significado fiável, mantendo a consistência.

Uma das soluções utilizada na integração de sistemas e na comunicação entre aplicações diferentes passa pelo desenvolvimento de *Web Services* que são componentes que permitem às aplicações enviar e receber dados, onde cada uma pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, um formato intermediário como *JSON, XML*, etc. [9]

Utilizando a tecnologia *Web Service*, uma aplicação pode utilizar outra para efetuar tarefas simples ou complexas mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Por outras palavras, os *Web Services* fazem com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos pelo *Web Service*.

Com o objetivo de consciencializar e abordar este tema, surgiu o presente trabalho que se enquadra na unidade curricular de Interoperabilidade Semântica e pretende que os alunos desenvolvam *Web Services REST*, que é um estilo de arquitetura que define um conjunto de restrições e propriedades baseados em *HTTP*.

2 Descrição do Problema

Neste trabalho pretendeu-se a simulação de um sistema de saúde de um determinado estabelecimento, recorrendo a *Web Services REST* desenvolvidos pelos alunos.

Este sistema de saúde é composto por duas aplicações. Para ambas, foi pedida a criação de uma Base de Dados, bem como, um conjunto de operações que permita a realização de um conjunto de requisitos. Relativamente à primeira aplicação (aplicação A), foi pedido que esta permitisse:

1. Pedir um exame e agendar o mesmo para um determinado dia/hora;
2. Disponibilizar a lista de exames a realizar num determinado dia;
3. Disponibilizar um serviço para receber notificações de exames realizados e relatórios.

Em relação à segunda aplicação (aplicação B), o conjunto de requisitos pedidos foi o seguinte:

1. Carregar a lista de todos os exames a realizar no presente dia;
2. Realizar os exames e disponibilizar os relatórios.

Qualquer partilha de informação entre as duas aplicações apenas foi feita através do envio e receção de mensagens *JSON*[7] utilizando o protocolo *HTTP*[8].

3 Resolução do Problema

De forma a que as duas aplicações conseguissem realizar os requisitos apresentados e trocar informação foi necessário efetuar um levantamento de caraterísticas e estudo das mesmas, com o objetivo de perceber a melhor estrutura aplicacional, bem como, as melhores tecnologias para cada aplicação a desenvolver.

3.1 Estrutura aplicacional

Os *Web Services REST* foram construídos utilizando o conceito denominado *microservices* [5] que segue uma abordagem arquitetónica e organizacional, cada um é composto por pequenos serviços independentes que comunicam usando *API's* (*Application Programming Interface*[10]). Esta abordagem facilitou a escalabilidade e agilizou o desenvolvimento dos *Web Services*, habilitando a inovação e acelerando o tempo de execução do trabalho.

Usando uma arquitetura de *microservices*, as aplicações foram criadas como componentes independentes que executam cada processo da aplicação como um serviço. Esses serviços comunicam por meio de uma interface usando *API's* leves. Os serviços criados realizam uma única função e são executados de forma independente, cada serviço pode ser atualizado, implementado e escalado para atender à procura de funções específicas de cada uma das aplicações.

Tal como se apresenta na figura 1 existe apenas uma interface para controlar o processo de interação, existindo um outro serviço que também foi criado denominado Serviço de Dados que é uma *API* que efetua o registo de novas informações na base de dados, bem como, fornece essas mesmas. Desta forma, o utilizador limita-se a interagir com a interface construída, assim estes serviços recebem os pedidos nas rotas previamente definidas através do protocolo de comunicação *HTTP* e respondem a esses pedidos num determinado formato.

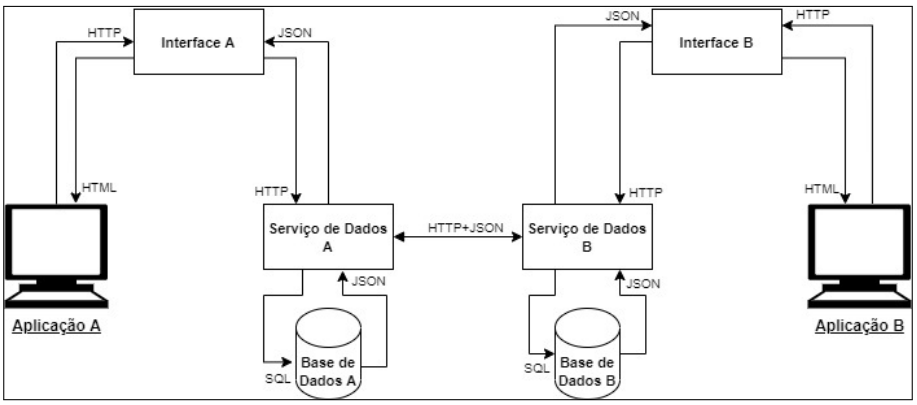

Figura 1: Estrutura aplicacional.

3.2 Bases de Dados

Ambas as aplicações desenvolvidas partilham dados, assim ambos os sistemas de registo de episódios, pacientes, tipos de exames e pedidos precisam de armazenar a mesma informação, por isso ambas as bases de dados são iguais, mas poderiam muito bem ter uma estrutura diferente que permitisse guardar a mesma informação.

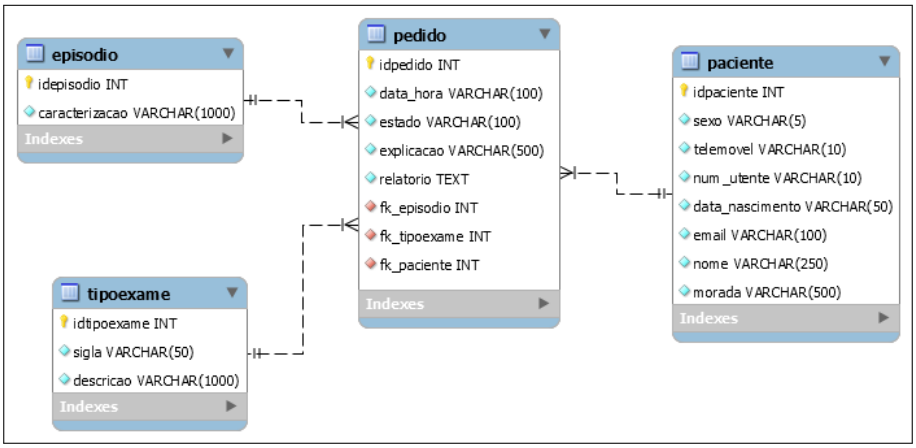

Figura 2: Base de dados implementada.

Como pode ser observado na figura 2 foram consideradas as seguintes entidades (tabelas) com os seguintes atributos (colunas):

- **Episódio** contextualiza um determinado pedido que foi feito ou irá ser feito;
- **Tipo de Exame** referente ao contexto do pedido. Possui um identificador, uma sigla e uma descrição do tipo de exame em causa;
- **Paciente** descreve o doente identificado por um número sequencial, um género, um telemóvel, um número de utente, uma data de nascimento, um email, um nome e uma morada;
- **Pedido** descreve o exame a realizar através de um identificador único, de uma data e hora, de um estado, de uma explicação que fundamente o pedido, o respetivo relatório e o tipo de exame, o episódio e o paciente em causa, através de uma chave estrangeira referente à entidade respetiva.

3.3 Serviço de Dados

A apresentação das diversas informações ao utilizador requer uma componente capaz de comunicar com a base de dados, interpretando-os para a interface da aplicação respectiva num formato de fácil interpretação.

Para este efeito, tal como foi apresentado na figura 1 criou-se um serviço baseado na *framework* **Express** [2] disponibilizada pelo **Node.js** [4] que permitiu construir uma *API*, deste modo a informação que foi apresentado na figura 2 é armazenada numa base de dados relacional **MySQL** [3] que permite a separação dos diferentes conceitos.

Assim, cada aplicação possui um serviço de dados que permite efetuar as típicas operações *GET* e *POST* dos *Web Services REST*, em que a primeira consiste na obtenção da informação presente na base de dados, o que equivale a um *SELECT* sobre a mesma enquanto que, a segunda consiste no registo de dados, isto pode ser um *INSERT ou UPDATE* na base de dados mediante a operação solicitada. É de realçar que esta interface apenas responde a pedidos *HTTP* nas rotas criadas e devidamente identificadas e devolve sempre a informação no formato *JSON*.

3.4 Interface

O utilizador apenas pode executar as operações se existir um meio que permite apresentar a informação de uma forma intuitiva e compreensível, para esse efeito e tal como foi descrito foi construído uma interface para cada uma das aplicações.

Cada uma das interfaces foi baseada na *framework* **Express** disponibilizada pelo **Node.js** contudo nesta, em vez de, devolver a informação no formato *JSON* devolve páginas *HTML* construídas a partir de **Pug** [6]. Para a interface de cada aplicação apresentar a informação presente na base de dados respetiva, necessita de solicitar a informação desejada ao serviço de dados respetivo, assim, utilizou-se a *framework* ***Axios*** [1] que permite comunicar através de um protocolo com o Serviço de Dados devido. Deste modo, a interface com esta última *framework* efetua pedidos *HTTP*, processa a informação recebida em *JSON* e transforma em *HTML*.

3.4.1 Aplicação A

Nesta aplicação, o utilizador ao aceder à página inicial é apresentada uma descrição da aplicação, também é possível observar pela figura 3 uma barra superior que possui o conjunto de operações permitidas nesta aplicação, onde cada opção é um *link* para a página que permite efetuar a operação desejada.

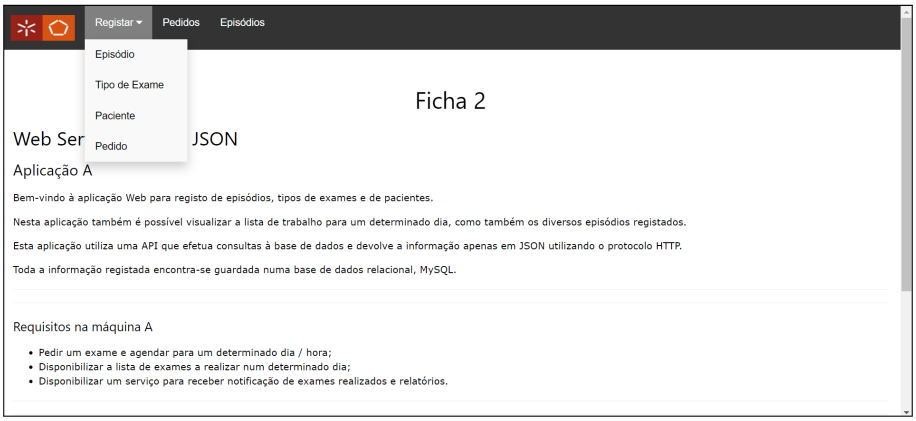

Figura 3: Menu Inicial da Aplicação A.

Tal como o nome indica, as operações de "Registar Episódio", "Registar Tipo de Exame"e "Registar Paciente", permitem o registo de novos dados. Na figura 4 é apresentada a página dedicada ao registo de um novo paciente, devendo este preencher todos os campos requeridos.

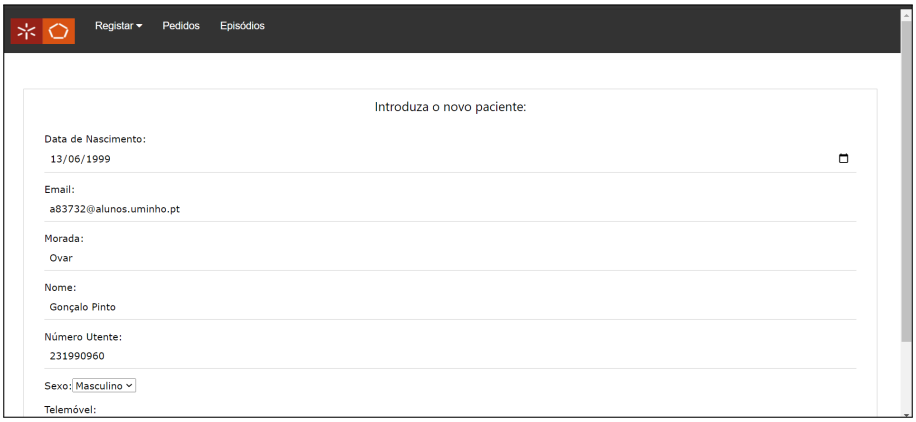

Figura 4: Registo de um novo paciente.

Visto que o trabalho se centra em pedidos e realização dos mesmos, na figura 5 está apresentada a página relativa ao registo de um novo pedido.

Figura 5: Registo de um novo pedido.

Através da página presente na figura 3 é possível aceder aos pedidos que estão por realizar, neste *link* o utilizador é encaminhado para uma tabela presente na figura 6. Nesta tabela é possível pesquisar por data, sendo facultado ao utilizador a filtragem dos pedidos para um determinado dia ou hora. Em cada linha que representa um pedido é possível cancelá-lo através de um botão presente na última coluna, bem como, é possível visualizar mais informação acerca do pedido através de um outro botão que encaminha para uma página com mais detalhes sobre o mesmo (figura 7). Na página de um pedido é apresentado apenas o nome e número de utente do paciente, contudo é possível observar os detalhes do paciente (figura 8) através de um botão criado para o efeito.

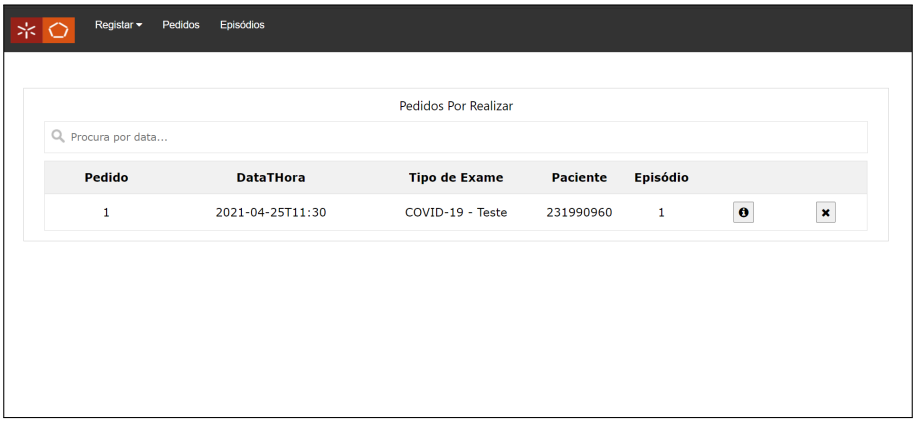

Figura 6: Lista de pedidos por realizar.

Figura 7: Página detalhada de um pedido.

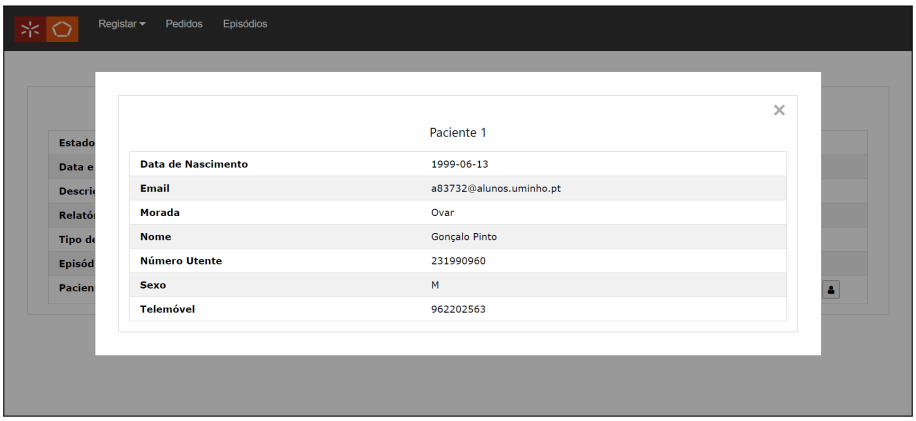

Figura 8: Página detalhada de um paciente.

Por fim, foi adicionada uma nova página que permite uma visão global sobre os episódios existentes, como se apresenta na figura 9. Em cada episódio é possível aceder à lista de pedidos associados a ele clicando no botão criado para o efeito, esta lista de pedidos vai ser idêntica à da figura 6 onde o estado pode ser qualquer um.

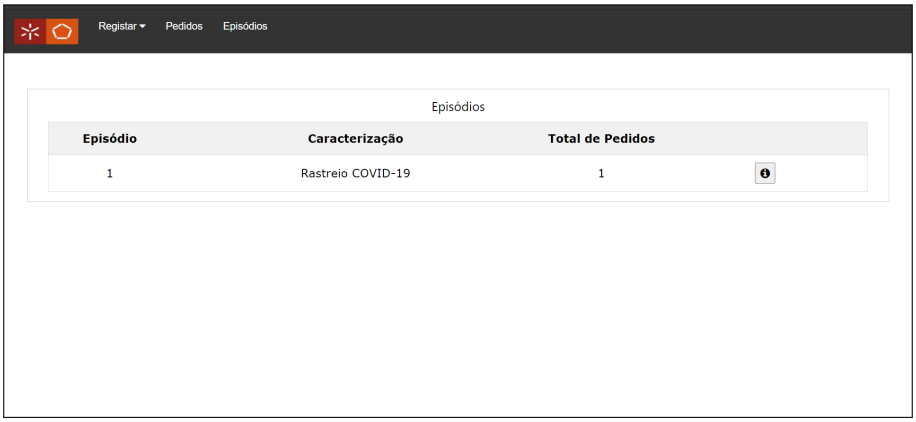

Figura 9: Lista de episódios.

3.4.2 Aplicação B

Também nesta aplicação, o utilizador ao aceder à página inicial é apresentada uma descrição da aplicação, também é possível, observar na figura 10 uma barra superior que possui o conjunto de operações permitidas nesta aplicação, onde cada opção é um *link* para a página que permite efetuar a operação desejada.

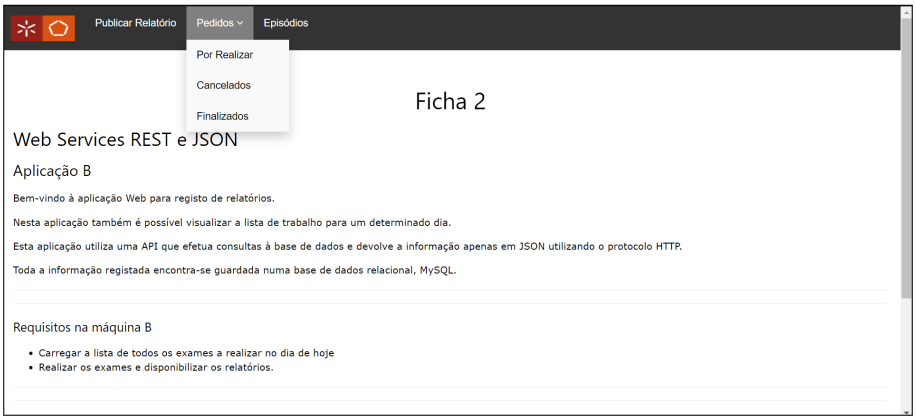

Figura 10: Menu Inicial da Aplicação B.

Uma das funções definidas como essenciais é a escrita de relatórios para os pedidos. Como tal, é apresentada um a página que permite a seleção do exame e a escrita do relatório deste.

Figura 11: Geração do relatório de um pedido.

Uma outra operação presente nesta aplicação é a possibilidade de visualizar os pedidos mediante um determinado estado que se pretenda. Deste modo, tal como retratado na figura 10, a plataforma permite apresentar os pedidos por realizar, finalizados e cancelados.

Figura 12: Lista de pedidos finalizados.

Na lista de pedidos por realizar, além de alguns dados sobre estes, é facultado ao utilizador desta aplicação a possibilidade de cancelar um determinado pedido. Por outro lado, tanto na lista de pedidos finalizados, figura 12, como na lista de pedidos cancelados não existe essa opção, limitando-se apenas à visualização dos mesmos, contudo em todas as listas é possível observar mais detalhes sobre um determinado pedido através de um botão criado para o efeito, a informação referida é a que se apresenta na figura 13.

Figura 13: Página detalhada de um pedido.

Por fim, existe uma outra operação nesta aplicação que é visualizar a lista de episódios de modo idêntico ao que se apresenta na aplicação A, presente na figura 9. Em cada episódio pode-se visualizar os pedidos associados a este e consequentemente observar detalhadamente cada um destes, como se demonstra na figura 14.

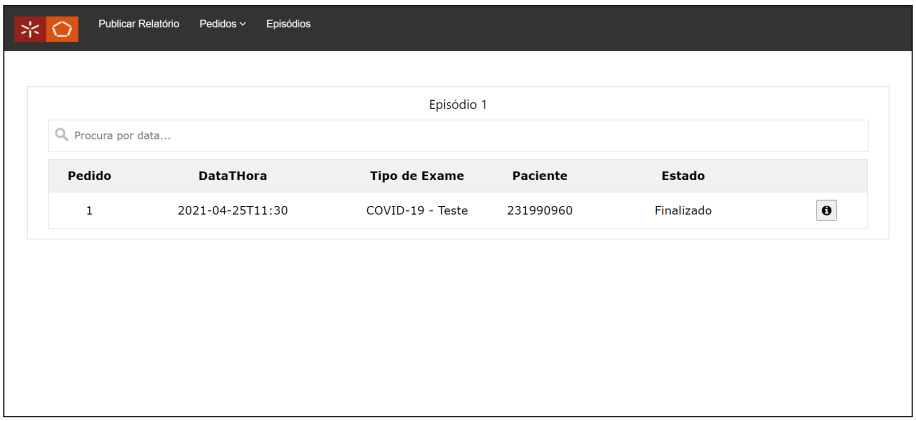

Figura 14: Pedidos associados a um episódio.

3.5 Comunicação entre aplicações

A comunicação entre os dois sistemas é, naturalmente, uma peça fundamental, tendo em conta que o tema principal do projeto proposto é a Interoperabilidade Semântica. Assim, foi efetuada uma comunicação entre os dois sistemas onde os dados a enviar foram convertidos em *JSON* e enviados utilizando o protocolo de comunicação *HTTP*.

Uma vez que a linguagem de programação utilizada foi *JavaScript* que utiliza o formato *JSON* para armazenar e transportar dados, a questão de conversão foi bastante facilitada uma vez que, a sintaxe deste formato é derivada da sintaxe de notação de objeto da linguagem. É de realçar que o formato *JSON* é apenas texto podendo ser escrito e lido em qualquer linguagem de programação, o que permite também que, os dados a enviar possam adotar a estrutura que se entender fornecendo assim uma liberdade para estruturação e conversão dos dados.

De forma a comunicar os dados foi utilizado novamente *framework Axios* pois tal como foi previamente referido utiliza *HTTP* para efetuar pedidos *GET, POST, etc.*

Uma vez que as duas aplicações possuem cada uma *API* que fornece rotas de forma a comunicar a informação com a aplicação respetiva, o grupo decidiu que a comunicação iria ser efetuada entre os serviços de dados de cada aplicação, desta forma é possível distinguir a camada de dados da camada de visualização de cada aplicação. Uma outra decisão tomada foi que, sempre que exista nova informação inserida pelo utilizador, independente da aplicação é comunicada à outra aplicação, isto garante que a informação se encontra sempre atualizada.

3.5.1 Aplicação A para Aplicação B

Em relação à aplicação A após a interface comunicar ao serviço de dados desta aplicação a informação registar, o serviço efetua a inserção na base de dados respetiva e posteriormente comunica a informação ao outro serviço de dados que por sua vez insere na base de dados a que tem acesso. Assim, vão existir as mesmas rotas de *POST* em ambas aplicações, no entanto a aplicação A possui um passo extra que é o de enviar a informação para o outro serviço.

De seguida apresentamos as mensagens que são trocadas no registo de um paciente e de um pedido, respetivamente.

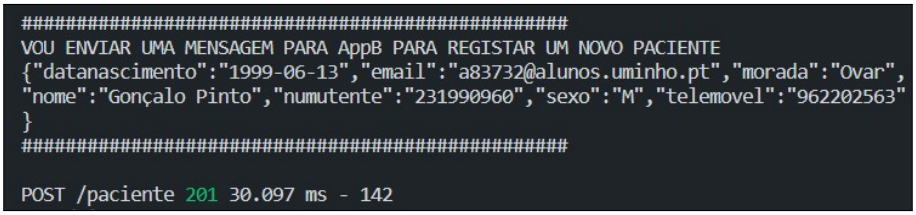

Figura 15: Mensagem enviada pela aplicação A à aplicação B para registar um paciente.

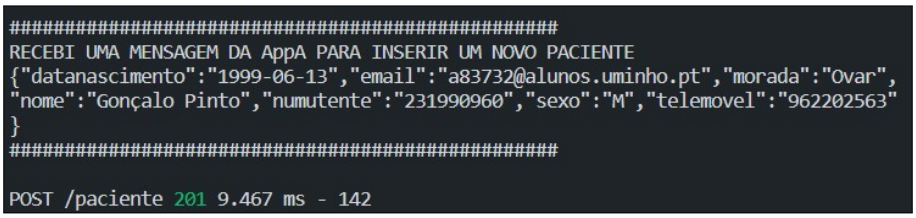

Figura 16: Mensagem recebida pela aplicação B por parte da aplicação A para registar um paciente.

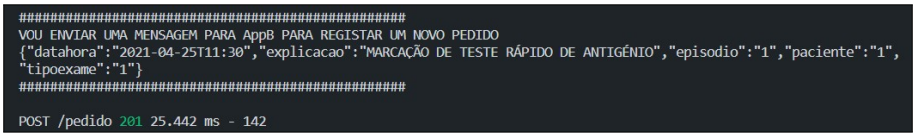

Figura 17: Mensagem enviada pela aplicação A à aplicação B para registar um pedido.

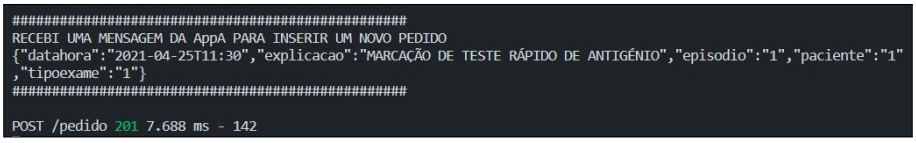

Figura 18: Mensagem recebida pela aplicação B por parte da aplicação A para registar um pedido.

3.5.2 Aplicação B para Aplicação A

Em relação à aplicação B após a interface comunicar ao serviço de dados desta aplicação a informação a registar, nesta situação apenas é registado um relatório. O serviço efetua a inserção na base de dados respetiva e posteriormente comunica a informação ao outro serviço de dados que por sua vez insere na base de dados a que tem acesso, tal como foi apresentado previamente vai existir a mesma rota de *POST* para inserir um relatório em cada serviço, contudo no da aplicação B tem a tarefa adicional de enviar para a outra aplicação.

De seguida apresentamos as mensagens que são trocadas no registo de um relatório.

```
################################################
VOU ENVIAR UMA MENSAGEM PARA AppA PARA ATUALIZAR O RELATÓRIO
{"pedido":"1","relatorio":"Declaramos para os devidos efeitos que Gonçalo Rodrigu
es Pinto, com o número de identificação 231990960 com data de nascimento 13/06/19
99 realizou o teste de deteção de antigénio pelo teste Abbott Panbio TM Covid-19
Ag Rapid Test na hora e dia\r\nde 13:55 25/04/2021, e obteve o resultado de: NEGA
TIVO"}
################################################
```

Figura 19: Mensagem enviada pela aplicação B à aplicação A para registar um relatório.

```
################################################
RECEBI UMA MENSAGEM DA AppB PARA ADICIONAR UM RELATÓRIO
{"pedido":"1","relatorio":"Declaramos para os devidos efeitos que Gonçalo Rodrigu
es Pinto, com o número de identificação 231990960 com data de nascimento 13/06/19
99 realizou o teste de deteção de antigénio pelo teste Abbott Panbio TM Covid-19
Ag Rapid Test na hora e dia\r\nde 13:55 25/04/2021, e obteve o resultado de: NEGA
TIVO"}
################################################
```

Figura 20: Mensagem recebida pela aplicação A por parte da aplicação B para registar um relatório.

3.5.3 Operações comuns

Em relação à operação de cancelamento que pode ser executada em cada uma das aplicações, criou-se em cada serviço de dados rotas que podem ter que comunicar à outra aplicação o cancelamento, como também, rotas que simplesmente atualizam o estado do pedido que foi recebido, por exemplo, se aplicação A decide cancelar um pedido, o estado deste é atualizado na base de dados A, através do serviço de dados respetivo, consequentemente este serviço envia o pedido de cancelamento ao serviço de dados da aplicação B, cabendo a este, atualizar o estado do pedido recebido. Uma vez que a informação é sempre comunicada, existe sempre garantia que a informação está presente nas bases de dados, desta forma, existe uma separação lógica das tarefas de cada serviço.

4 Resultados

Com a realização do presente trabalho foram obtidos alguns resultados que se consideram interessantes:

- No primeiro trabalho da unidade curricular de Interoperabilidade Semântica foi implementado o mesmo projeto só que, a comunicação entre os sistemas foi baseada em *HL7* o que acarretou um trabalho extra, devido à necessidade de alguma pesquisa por forma a entender a sua sintaxe e funcionamento, mudando a base de comunicação neste trabalho para *JSON* facilitou bastante a conversão dos dados a enviar, uma vez que como foi referido é texto o que leva a uma grande liberdade na sua representação.

- No primeiro trabalho o grupo decidiu optar por utilizar páginas *HTML* para interagir com o utilizador e *Javascript* para efetuar as operações. Esta abordagem foi muito semelhante ao deste trabalho, apenas a comunicação foi diferente o que permitiu que o desenvolvimento dos *Web Services* pretendidos fosse mais fácil, possibilitando a exploração da arquitectura *REST* implementada, separando os serviços permitiu uma melhor estruturação do projeto e a separação de conceitos.

- Atualmente, a arquitetura mais utilizada nos *Web Services* é o *REST*. Uma vez que as tecnologias utilizadas são recentes encontram-se vocacionadas para este tipo de arquitetura permitindo alguma agilidade na criação dos *Web Services*. Contudo, caso se pretendesse utilizar uma arquitetura mais antiga como *SOAP*, que é também um protocolo para troca de informações entre aplicações, que utiliza *XML* e *HTTP* a conversão seria de fácil execução. Tendo em conta, a estrutura aplicacional adoptada, *Micro Web Services REST*, caso se pretenda efetuar a migração para uma arquitetura *SOAP*, ter-se-ia apenas de alterar as mensagens que são enviadas e consequentemente a extração da informação que é recebida para o formato da mensagem específico (*JSON* para *XML*), aproveitando já o protocolo de comunicação estabelecido. Nestas mensagens seria necessário transformar a mensagem a enviar num "envelope", que define o que está na mensagem e como processá-la. Este é composto por um *header*, elemento opcional, que possui as informações adicionais, como por exemplo, se a mensagem deve ser processada por um determinado nó intermediário e no *body*, elemento obrigatório, que possui o *payload* ou a informação a ser transportada para o seu destino final.

5 Conclusão

O presente relatório descreveu, de forma sucinta, o processo de construção de um protótipo para interoperação entre os dois sistemas enquadrado num determinado caso de estudo.

Durante a elaboração deste trabalho surgiram algumas dificuldades que, com o esforço e entusiasmo do grupo pelo resultado final, acabaram por ser ultrapassadas.

Numa perspetiva futura, considero que seria interessante a implementação usando outra arquitectura de *Web Services*, como por exemplo *SOAP*.

Após a realização deste trabalho, fiquei consciente da importância que interoperabilidade entre sistemas de informação têm, esta é extremamente necessária no quotidiano pois ao existir um formato como uma linguagem de comunicação comum entre aplicações é possível desenvolver sistemas avançados que garantam que a informação não se perca e que não seja incoerente.

Considero que os objetivos propostos com a realização deste trabalho foram cumpridos, bem como, a consolidação dos conhecimentos na Interoperabilidade Semântica.

Por fim, espero que os conhecimentos obtidos e consolidados sejam de enorme utilidade tendo em conta uma perspetiva futura no mundo laboral.

Referências

[1] *Axios*. 2021. URL: https://www.npmjs.com/package/axios (acedido em 25/04/2021).

[2] *Express*. 2019. URL: https://expressjs.com/ (acedido em 25/04/2021).

[3] *Mysql*. 2020. URL: https://www.npmjs.com/package/mysql (acedido em 25/04/2021).

[4] *Node.js*. 2019. URL: https://nodejs.org/en/ (acedido em 25/04/2021).

[5] *O que são microsserviços?* 2021. URL: https://aws.amazon.com/pt/microservices/ (acedido em 25/04/2021).

[6] *Pug*. 2021. URL: https://www.npmjs.com/package/pug (acedido em 25/04/2021).

[7] *Trabalhando com JSON*. 2020. URL: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/JSON (acedido em 25/04/2021).

[8] *Uma visão geral do HTTP*. 2021. URL: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview (acedido em 25/04/2021).

[9] *Web service: o que é, como funciona, para que serve?* 2016. URL: https://www.opensoft.pt/web-service/ (acedido em 25/04/2021).

[10] *What is an API? In English, please.* 2019. URL: https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/ (acedido em 25/04/2021).