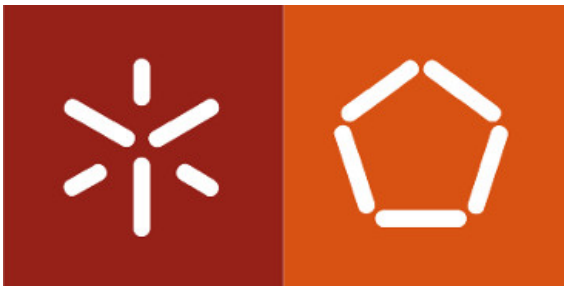

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Engenharia do Conhecimento

Interoperabilidade Semântica

Trabalho de Grupo

Grupo 01

Gonçalo Pinto, A83732

João Diogo Mota, A80791

José Gonçalo Costa, PG42839

José Nuno Costa, A84829

20 de junho de 2021

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 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 e 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 utilizadas 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. Utilizando a tecnologia *Web Service*, um sistema pode utilizar outro para efetuar tarefas simples ou complexas, mesmo que o outro sistema esteja num local diferente e escrito numa outra linguagem. [1]

Com o objetivo de abordar este tema, surgiu o presente trabalho que se enquadra na unidade curricular de Interoperabilidade Semântica e pretendeu que os alunos implementassem um sistema onde fossem aplicados métodos de interoperabilidade, em particular arquiteturas baseadas em serviços Web.

O sistema desenvolvido visa gerir *Electronic Health Records* (EHR) baseados no modelo clínico *OpenEHR*. Para isso, o grupo decidiu construir um *Web Service REST* que utiliza um estilo de arquitetura que segue um conjunto de restrições e propriedades baseados em HTTP.

Capítulo 2

Contextualização e Trabalhos Relacionados

2.1 Contextualização

Um *Electronic Health Record* (EHR) é uma versão digital do histórico clínico de um paciente. EHRs são registos, em tempo real, centrados no paciente que tornam as informações disponíveis de forma instantânea e segura para os utilizadores autorizados. [2]

Embora um EHR contenha os históricos médicos e de tratamento dos pacientes, um EHR contém outros dados além dos dados clínicos registados numa consulta, podendo incluir uma visão mais ampla do histórico ao paciente. EHRs são considerados uma parte vital da tecnologia da informação no tema da saúde e podem:

- conter o histórico médico, diagnósticos, medicamentos, planos de tratamento, datas de imunização, alergias, imagens de radiologia e resultados de exames laboratoriais do paciente;
- permitir o acesso a ferramentas baseadas em evidências que os fornecedores dos serviços podem usar para tomar decisões sobre o atendimento de um paciente;
- automatizar e agilizar o fluxo de trabalho dos fornecedores dos serviços.

Um dos principais recursos de um EHR é que as informações de saúde possam ser criadas, geridas por fornecedores autorizados, bem como, partilhadas com outros fornecedores em mais de uma instituição de saúde. Os EHRs são construídos para compartilhar informações com outras instituições de saúde e organizações - como laboratórios, especialistas, clínicas, farmácias, hospitais - para que cada um contenha informações de todos os médicos e procedimentos envolvidos no cuidado de um paciente.

2.2 Trabalhos Relacionados

Um dos trabalhos mais relacionados com EHR é o *OpenEHR*, uma tecnologia para saúde electrónica, que consiste em especificações, em modelos clínicos e em software que podem ser utilizados para criar padrões e construir soluções de informação e interoperabilidade para a saúde. [3]

Este projeto surgiu principalmente pela necessidade de os pacientes terem que levar consigo os seus registos clínicos, não havendo a partilha de informação entre serviços de saúde. Outro dos grandes problemas que pretende resolver é na prestação de cuidados, na medida de colmatar a falta de suporte existente em várias instituições de saúde, como por exemplo referências perdidas, transferências de pacientes, etc. Este tipo de problemas acaba sempre por prejudicar a pessoa que necessita dos serviços. Nesse sentido o objetivo do *OpenEHR* é melhorar a qualidade da interoperabilidade semântica na área da saúde.

As especificações *OpenEHR* incluem modelos de informação para dados de saúde, incluindo várias opções, entre elas uma especificação para uma API aberta. Sendo esta um conjunto de rotas e padrões estabelecidos para a utilização de funcionalidades por parte das aplicações que não pretendem saber os detalhes da implementação do software, mas apenas usar os serviços.

Capítulo 3

Metodologia, Tecnologias Utilizadas e Estrutura do EHR

3.1 Metodologia

No sentido de alcançar os objetivos definidos que foi gerir EHRs, o sistema construído teve por base a API fornecida pelo *OpenEHR*, quer a nível de rotas, quer a nível de conteúdo que é enviado e devolvido em cada uma destas.

Uma API, sigla para *Application Programming Interface*, pode definir-se, tal como o nome indica, como sendo uma interface de programação de aplicações, ou seja, uma API é um conjunto de normas que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos. A função de uma API é, de forma simplificada, facilitar o trabalho dos programadores, oferecendo um conjunto de funcionalidades desenvolvidas previamente, visando garantir sempre o máximo de segurança possível, com a capacidade de bloquear acesso e permissões a dados de software e hardware que algumas aplicações não podem usar.

Uma API REST, também chamada de API RESTful, é uma API que está em conformidade com as restrições do estilo arquitetural REST, permitindo a interação com serviços web RESTful. REST é a sigla em inglês, para transferência representacional de estado e define um conjunto de restrições a serem usadas para a criação de *Web Services* [4]. Estes serviços denominados *Web Services* RESTful, fornecem interoperabilidade entre sistemas, algo essencial no contexto do projeto proposto.

3.2 Tecnologias Utilizadas

No contexto referido, criou-se um serviço baseado na *framework Open Source, Express,* disponibilizada pelo *Node.js*. Esta *framework* é rápida, criada com o intuito de otimizar o desenvolvimento de aplicações *Web e APIs*. A sua utilização conjunta com o *Node.js* facilita o desenvolvimento de aplicações *back-end* e, em conjunto com sistemas de *templates*, aplicações *full-stack*. Esta *framework* permitiu construir uma *API* baseada na referida API fornecida pelo *OpenEHR*.

Deste modo a informação é inserida numa base de dados documental que armazena cada registo e todos os dados associados num único documento. Cada documento é composto por dados semiestruturados, dados esses, que podem ser consultados usando várias ferramentas de consulta e análise do Sistema de Gestão de Base de Dados (SGBD).

Nas bases de dados documentais os dados são armazenados como um valor que, por sua vez, é um documento e a chave associada é um identificador exclusivo para esse valor. A escolha recaiu neste paradigma de base de dados, uma vez que, fornece grande flexibilidade sobre os dados a inserir. Enquanto nas bases de dados relacionais, é necessário efetuar um modelo lógico antes da inserção de qualquer dado, as documentais não exigem esse requisito apenas é necessário carregar todos os dados sem usar qualquer esquema predefinido, sendo este construído à medida que inserimos os dados.

O SGBD orientado a documentos utilizado, *MongoDB*, foi concebido especialmente para os dados que podem ser representados num único documento, não sendo adequado para dados relacionados. Desta forma, o *MongoDB* é mais apropriado para realizar consultas aos EHRs.

3.3 Estrutura do EHR

Tendo em consideração o modelo clínico apresentado pelo *OpenEHR*, o grupo decidiu armazenar no *MongoDB* apenas uma coleção de documentos onde cada um corresponde apenas a um único EHR. Desta forma considerou-se armazenar em cada EHR, as seguintes informações:

- *system_id*, identificador do sistema que gere os EHRs;
- *ehr_id*, identificador único de cada EHR;
- *ehr_status*, objeto único por cada EHR que contém várias informações e configurações do estado, incluindo:
  - *subject*, um objeto que contém a referência direta ao assunto num serviço demográfico ou à identidade;
  - *is_modifiable*, é verdadeiro se for um registo que pode ser alterado;
  - *is_queryable*, é verdadeiro se for um registo considerado ativo;
- *ehr_access*, uma referência ao objeto *ehr_access* que controla o acesso ao EHR;
- *time_created*, a data e hora de criação do EHR;
- *folders*, uma lista opcional para cada EHR. Se existir, cada objeto desta lista contém referências a outros objetos e por isso cada uma é composta por:
  - *items*, as referências a outros objetos;
  - *folders*, *sub-folders* para esta *folder*;
- *directory*, é um objeto opcional para cada EHR. Se existir, é uma referência ao primeiro elemento do atributo *folders*;
- *contributions*, é a lista das contribuições que causaram alterações a um EHR. Cada *contribution* contém:
  - *versions*, lista de versões, que pode incluir referências a qualquer número de VERSION instâncias. Desse modo considerou-se que existem *versioned_objects* e que cada um deles tem as sua *versions*;
  - *audit*, corresponde a objeto que contém informação de quem submeteu esta *contribution*;

O objetivo do sistema desenvolvido era gerir EHRs por esse motivo, não foram consideradas nenhumas operações de manipulação do objeto *ehr_status*, bem como, das *compositions*. Contudo, o primeiro objeto é um elemento obrigatório de cada EHR logo sempre que é criado um é necessário também criar um *ehr_status*, enquanto que, as *compositions* são elementos opcionais.

Capítulo 4

Implementação

No presente capítulo pretendemos descrever as funcionalidades implementadas no sistema de gestão de EHRs desenvolvido. Desse modo, em cada uma das próximas secções irá ser detalhado quais as ações permitidas de efetuar na API, bem como, o seu funcionamento e ainda, os parâmetros e conteúdo que cada uma é expetável que receba.

4.1 Gestão de EHRs

Nesta secção foram desenvolvidas rotas que permitissem tanto criar EHRs, como a possível consulta destes.

4.1.1 **Criar EHR** - *POST /ehr*:

De forma a poder criar um novo EHR, o primeiro passo a executar é extrair do *header - prefer* do pedido a informação que diz respeito ao tipo de representação que o utilizador quer no final, caso o processo ocorra com sucesso. Posteriormente é verificado se o utilizador enviou o *subject* relativo a cada *ehr_status*.

- caso envie o *subject*, é necessário que no atributo *external_ref* exista um identificador e um *namespace* para se garantir que estes dois valores são únicos de cada EHR. Após verificados estes campos é criado um novo EHR apenas com os campos obrigatórios do modelo clínico *openEHR* (*system_id, ehr_id, ehr_status, ehr_access e time_created*);
- caso não se envie o *subject*, decidiu-se que iria ser gerado um identificador para o *ehr_status* baseado na hora de criação. O *namespace* associado a este, seria um por defeito e os atributos *is_modifiable* e *is_queryable* pertencentes também ao *ehr_status* são criados como verdadeiros;

Criado e inserido cada EHR com base nas informações recebidas é retornada alguma informação mediante o tipo de representação que o utilizador indicou.

4.1.2 **Criar EHR com id** - *PUT /ehr/{ehr_id}*:

O processo é idêntico ao de criação do ponto 4.1.1. A diferença é que neste caso é efetuada uma validação prévia no sentido de verificar que o *ehr_id* enviado por parâmetro é do formato UUIDv4 e que ainda não existe nenhum EHR com esse identificador. Caso não exista o processo de criação do EHR é idêntico ao apresentado anteriormente apenas o identificador do EHR é o valor passado como parâmetro.

4.1.3 **Obter um sumário do EHR pelo id** - *GET /ehr/{ehr_id}*

Nesta funcionalidade é efetuada uma consulta a toda coleção de documentos armazenados, no sentido de verificar se existe algum *ehr_id* idêntico ao que o utilizador indicou. Caso exista, é retornada a informação sobre os valores *o system_id, do ehr_id, o ehr_status* onde neste apenas é apresentado o *id, type e namespace, o ehr_access e o time_created.*

4.1.4 **Obter um sumário do EHR pelo *subject* id** - *GET /ehr{?subject_id,subject_namespace}*

Nesta rota é efetuada uma consulta a toda coleção de documentos armazenados no sentido de verificar se existe algum *subject.external_ref.id.value e subject.external_ref.namespace* idênticos aos que o utilizador indicou. Caso exista é retornada a mesma informação do ponto 4.1.3.

4.2 Gestão das diretorias dos EHRs

4.2.1 **Criar diretoria** - *POST /ehr/{ehr_id}/directory*:

Com o objetivo de criar uma nova diretoria num determinado EHR, é necessário obrigatoriamente indicar qual o registo a que se pretende adicionar. Para isso deve ser fornecido o *ehr_id* deste. Antes de qualquer operação é verificado se este identificador se encontra no formato devido e se identifica algum EHR. Se assim for é construído um novo objeto *folder* com base na informação que o utilizador enviou, podendo esta ser constituída por *items*, onde estes são estruturados numa lista de objetos de referências, ou *folders* que podem conter *sub-folders*. Desse modo é construído de forma recursiva os objetos devidos até que não exista nenhuma lista.

Terminado este processo, os objetos construídos são agregados na *folder* principal, onde por sua vez é acrescentada à lista de *folders* do EHR. Previamente é verificado se o conteúdo desta lista é vazio. Se assim for, é necessário, além de acrescentar, também adicionar um novo objeto ao campo *directory* desse EHR onde o identificador deste é o mesmo que a *folder* principal.

4.2.2 **Atualizar diretoria** - *PUT /ehr/{ehr_id}/directory*:

Nesta rota considerou-se que a informação enviada pelo utilizador vai substituir uma determinada *folder* de um EHR. Para isso, o utilizador necessita de indicar o identificador deste, bem como, enviar pelo *header - If-Match* o identificador da *folder* a substituir. Deste modo efetua-se uma verificação do *ehr_id* recebido no sentido de encontrar o EHR respetivo. Caso encontre é criada uma nova *folder* tal como no ponto 4.2.1. Contudo, o identificador desta *folder* é previamente atualizado, isto é, o último número deste é incrementado.

Posteriormente, é tentada a atualização da *folder* pretendida. Caso ocorra com sucesso, é verificado se o identificador presente na *directory* é o mesmo que foi atualizado. Se assim for, é igualmente atualizado este objeto para uma versão mais recente. Caso não atualize a informação, é porque o identificador da *folder* indicada não existe ou encontra-se já numa noutra versão. De forma a que o utilizador fique o mais esclarecido possível é verificada qual das situações ocorreu e um erro diferente é retornado para cada uma destas.

4.2.3 **Eliminar diretoria** - *DELETE /ehr/{ehr_id}/directory*:

Tal como a rota anterior, para eliminar uma determinada diretoria é necessário enviar o identificador do EHR e da *folder* através do *header - If-Match* onde ambos são verificados. Para eliminar uma *folder* de um EHR é tentada a eliminação desta do EHR pretendido. Caso ocorra com sucesso é verificado se o identificador presente na *directory* é idêntico ao fornecido podendo ocorrer dois cenários: o primeiro é existirem outras *folders* além da que se eliminou, e nesse caso a *directory* passa a referir a primeira nova *folder*; o outro cenário possível de ocorrer é quando a *folder* eliminada é única. Deste modo a *directory* é colocada como um objeto vazio. Por outro lado, se nenhuma *folder* foi eliminada é porque o identificador da *folder* indicado não existe ou encontra-se já numa noutra versão. De forma a que o utilizador fique o mais esclarecido possível, é verificada qual das situações ocorreu, e um erro diferente é retornado para cada uma destas.

4.2.4 **Obter uma diretoria pela versão desta** - *GET ehr/{ehr_id}/directory/{version_uid}{?path}*:

Na presente funcionalidade pretendeu-se disponibilizar um método que permitisse ser possível obter uma determinada *folder* de um EHR. Para isso o utilizador deve fornecer o identificador desse EHR e da *folder* pretendida. Como foi apresentado, uma *folder* pode possuir *sub-folders*. Nesse sentido é facultada uma pesquisa sobre essas, para esse fim. Na *query* desta rota, deve ser indicado o *path* pretendido, onde cada nome da *folder* deve ser dividido por /.

Assim sendo, é verificado a existência do EHR. Posteriormente é procurado nas *folders* deste alguma que seja identificada pelo *version_uid* fornecido. Caso se encontre uma *folder* e não seja indicado nenhum *path* de pesquisa na *query*, é retornada a informação relativa à *folder* em questão. Por outro lado, pode ser indicada a intenção de uma procura por *sub-folders*. Então, estas são percorridas, comparando o nome de cada uma, com a parte respetiva do *path* pretendido e a informação retornada neste caso é a última *sub-folder* presente no caminho estabelecido.

4.2.5 **Obter a diretoria** - *GET ehr/{ehr_id}/directory{?path}*:

Enquanto que na rota do ponto 4.2.4. era permitido uma pesquisa por qualquer *folder* de um determinado EHR existindo a necessidade de especificar qual é através do identificador, na presente rota apenas é permitido uma pesquisa pela primeira *folder*, ou seja, o objeto referido pelo *directory*.

4.3 Gestão das contribuições dos EHRs

4.3.1 **Criar contribuição** - *POST /ehr/{ehr_id}/contribution*:

Para adicionar uma nova *contribution* a um determinado EHR é necessário, antes de mais, enviar na rota desta funcionalidade qual o identificador do EHR sendo este posteriormente validado e verificado. Caso o EHR indicado exista, é verificado o conteúdo do pedido pois o utilizador pode indicar opcionalmente o *uid* e/ou *system_id*. Caso estes valores sejam enviados, é necessário no caso do primeiro, garantir que não existe nenhuma *contribution* com esse *uid* e no segundo caso, que este valor seja idêntico ao da API construída.

Posteriormente, após verificados estes campos, é construído a lista de *versions*, de modo, a garantir que cada elemento que a compõem é uma referência para um determinado objeto. De seguida é construído o objeto *contribution* em si. Após inserido, mediante o tipo de representação final que o utilizador pretenda, é enviada alguma informação a este.

4.3.2 **Obter contribuição pelo id** - *GET /ehr/{ehr_id}/contribution/{contribution_uid}*:

Nesta rota, o objetivo é obter uma determinada contribuição através do identificador gerado aquando da criação. Nesse sentido, é necessário que o utilizador indique qual o EHR que pretende consultar através do *ehr_id* e da contribuição através do *contribution_uid*. Para isso, efetua-se uma verificação inicial, com o intuito de procurar esse EHR. Se existir, então é procurada na lista de *contributions* alguma contribuição com o *uid* fornecido.

Capítulo 5

Complexidade do Sistema

Analisar a capacidade da API produzida é de extrema importância, uma vez que permite antecipar futuros problemas, perceber a capacidade real do sistema e corrigir problemas antes do sistema estar disponível para a utilização de um elevado número de utilizadores. Desta forma, a fase de testes e análise da complexidade do sistema é uma prática que deve ser aplicada ao desenvolvimento de qualquer tipo de *software*, dado permitir prever possíveis *bugs*, *bottlenecks*, entre outros.

Foram utilizados testes de carga, uma vez que permitem quantificar e avaliar algumas métricas gerais que devem ser consideradas quando se produz *software*, sendo elas:

- **Confiança**, se o sistema é resistente a falhas durante a execução, isto é, não entra em ciclo, não interrompe a execução por falta de recursos, entre outros;
- **Funcionalidade**, se o sistema apresenta um comportamento de acordo com o esperado e definido nos requisitos;
- **Performance**, se o sistema apresenta tempos de resposta adequados e aceitáveis, mesmo quando submetido a um volume de processamento próximo de situações reais ou de alta procura.

Para avaliar o desempenho da aplicação, foram realizados testes de carga, recorrendo ao *Apache JMeter*. Esta ferramenta permite testar o desempenho em recursos estáticos e dinâmicos, simulando várias situações de pressão (uma carga pesada num servidor, grupo de servidores, rede ou objeto) para analisar o desempenho geral. [5]

Para a realização dos testes de carga, é relevante o manuseamento das seguintes propriedades:

- *Number of threads*, o número de utilizadores concorrentes num grupo de *threads*;
- *Ramp-up period*, período de *warm-up* de cache do sistema. O *JMteter* não contabiliza este período para o desempenho;
- *Loop count*, número de iterações que serão executadas por *thread*.

Em todos os testes realizados, foi utilizado um *Ramp-up period = 10* segundos e um *Loop count = 1*, fazendo apenas variar o *Number of threads*. Em seguida, serão apresentados os diferentes testes de carga realizados sobre a API.

Para a realização dos testes de carga, foram executados quatro testes diferentes.

5.1 Tempos de Execução na criação de EHRs

O primeiro teste realizado consiste na criação de novos registos EHR. No conteúdo destes pedidos não foi enviado nenhum *ehr_status*. Assim, tal como foi apresentado previamente, o sistema desenvolvido gera um identificador para o *ehr_status* baseado na hora de criação. O *namespace* é um por defeito e os atributos *is_modifiable* e *is_queryable* são criados como verdadeiros.

Por outro lado, como um dos elementos obrigatórios era a preferência de representação através do *header*, apesar deste elemento ser irrelevante para testes na medida em que o objetivo não é a visualização mas sim a performance.

Figura 5.1: Gráfico do tempo de resposta para 100 *threads* a criar EHRs.

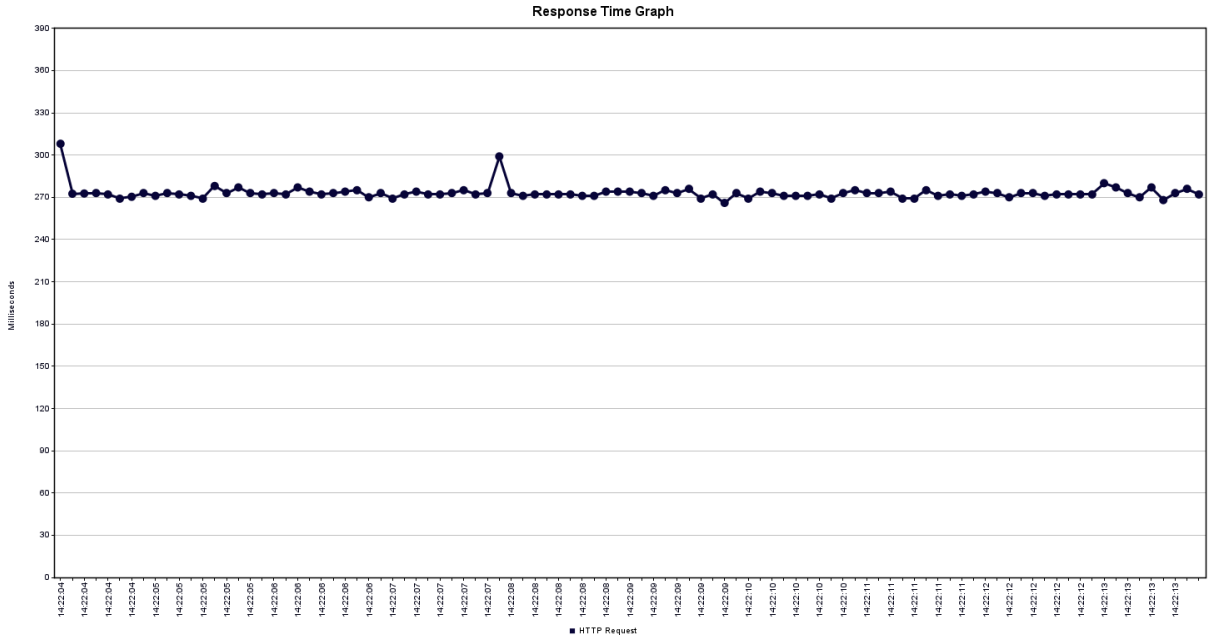

Tabela 5.1: Tempos de Execução na criação de EHRs.

Threads	***Throughput/sec***	***Tempo de Resposta(ms)***	***Error(%)***
100	10,0	273	0,00
500	50,3	277	0,00
1000	101,2	276	0,00
5000	156,4	8266	1,26
10000	88,4	17776	40,92

5.2 Tempos de execução na obtenção de um EHR

O segundo teste, por sua vez, é referente à obtenção do sumário de um determinado EHR pelo seu identificador, por esse motivo foi estabelecido um identificador fixo, ou seja, as *threads* encontram-se a aceder ao mesmo EHR.

Figura 5.2: Gráfico do tempo de resposta para 500 *threads* a obter um sumário do EHR pelo id.

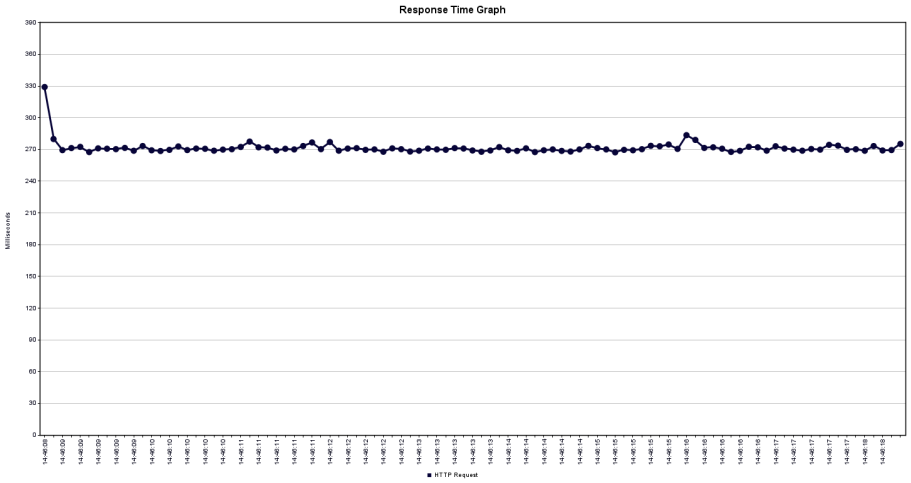

Tabela 5.2: Tempos de execução na obtenção de um EHR.

Clientes	***Throughput/sec***	***Tempo de Resposta(ms)***	***Error(%)***
100	10,0	272	0,00
500	50,2	272	0,00
1000	101,3	274	0,00
5000	146,8	7689	0,92
10000	129,1	17154	44,99

5.3 Tempos de execução na obtenção de uma diretoria

O terceiro teste realizado teve como objetivo a obtenção da primeira *folder*. Ou seja, o objeto referido pelo *directory* de um determinado EHR pelo seu identificador. Por esse motivo, foi estabelecido um identificador fixo, isto é, as *threads* encontram-se a aceder ao mesmo EHR e diretoria.

Figura 5.3: Gráfico do tempo de resposta para 1000 *threads* a obter uma diretoria de um EHR.

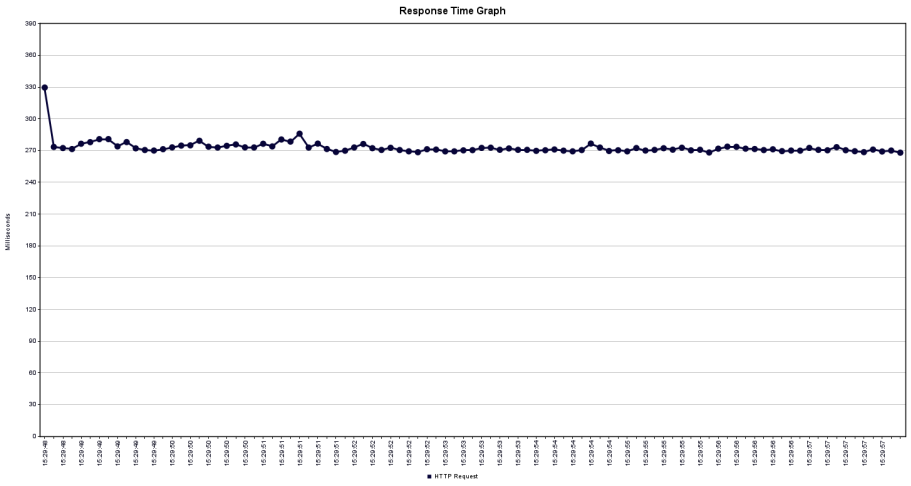

Tabela 5.3: Tempos de Execução na obtenção de uma diretoria.

Clientes	***Throughput/sec***	***Tempo de Resposta(ms)***	***Error(%)***
100	10,0	273	0,00
500	50,1	272	0,00
1000	102,0	273	0,00
5000	138,6	8261	2,82
10000	154,5	17885	42,21

5.4 Tempos de execução na obtenção de uma contribuição

Por fim, foi realizado um teste com o intuito de analisar a performance do sistema aquando de um pedido para obter uma contribuição pelo seu identificador. Para isto, foram estabelecidos identificadores fixos, ou seja, as *threads* encontram-se a aceder à mesma contribuição do mesmo EHR.

Figura 5.4: Gráfico do tempo de resposta para 5000 *threads* a obter uma contribuição de um EHR.

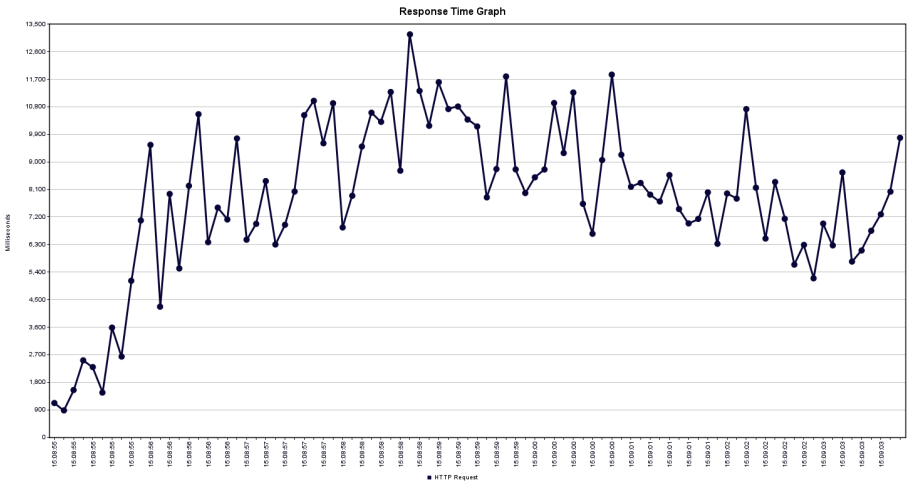

Tabela 5.4: Tempos de Execução na obtenção de uma contribuição.

Clientes	***Throughput/sec***	***Tempo de Resposta(ms)***	***Error(%)***
100	10,0	274	0,00
500	50,2	275	0,00
1000	101,1	275	0,00
5000	99,5	7568	1,58
10000	84,4	16932	43,02

Capítulo 6

Discussão

Neste projeto vários métodos foram implementados e cada um foi testado à medida do seu desenvolvimento. Todas as funcionalidades foram construídas de forma a garantir que a informação fosse o mais precisa e clara possível, o que dá confiança suficiente para utilizar e recomendar o sistema de gestão de EHRs construído. É importante referir que o EHR foi baseado no modelo clínico do *OpenEHR*, bem como, o sistema desenvolvido assenta na API desta organização.

Depois de todos os testes e devidamente analisados, é possível observar que a arquitetura e *deploy* elegidos para a nossa API correspondem aos requisitos propostos no início do projeto. Através dos gráficos e tabelas previamente apresentados no capítulo 5, identifica-se um padrão independente do teste que foi realizado. Ou seja, com o aumento do número de *threads* (clientes) no sistema, o tempo de resposta vai aumentando e o débito deixa de atingir o seu melhor valor, apesar disso ser uma certeza mesmo antes de se realizarem os testes. Além de se confirmar o bom funcionamento da arquitetura e da aplicação, os testes serviram também para explorar o limite do cliente ao realizar certas operações em simultâneo na API. Relativamente a este ponto, é possível afirmar que o ponto de rutura da API situa-se entre os **5000 e 10000** clientes, dado que é nesse momento que começam a surgir percentagens de erros significativas.

Através do sistema desenvolvido, é possível gerir os EHRs e seus recursos como diretorias e contribuições. Contudo, o aspeto que o grupo sentiu mais dificuldades no desenvolvimento da API descrita ao longo do relatório, diz respeito à falta de documentação clara da API do *OpenEHR*. A questão principal é que para o entendimento de como a API desenvolvida funciona e se deve utilizar, é necessário existir documentação.

De forma a contornar o problema, o grupo decidiu utilizar o *Swagger*, que é uma aplicação *open source*, que auxilia programadores nos processos de definir, criar, documentar e consumir APIs REST. O *Swagger* acaba por ser um padrão de integração e que descreve os seguintes recursos de uma API: endpoints, dados recebidos, dados retornados, códigos HTTP e métodos de autenticação.

O *Swagger* facilita a modelação, a documentação e a geração de código pois disponibiliza um conjunto de ferramentas que auxiliam o desenvolvedor de APIs REST, simplificando tarefas como a modelagem da integração, a geração de documentação (legível) e geração de códigos do cliente e do servidor, com suporte a várias linguagens de programação.

O *Swagger* traz um ecossistema formado por várias ferramentas para criação e manipulação de APIs. Assim, a ferramenta disponibilizada pelo *Swagger* utilizada foi o *Swagger UI*. Esta ferramenta é uma interface gráfica que permite, de uma forma mais simples, explorar as definições de APIs baseadas em *Swagger*, sendo aplicado na publicação da documentação. Esta ferramenta também foi auxiliada por um conjunto de bibliotecas *javascript* usadas para o consumo de APIs especificadas com o *Swagger*, que podem ser utilizadas com aplicações clientes e *Node.js*.

A principal contribuição do *Swagger* no trabalho desenvolvido foi garantir um padrão nas interfaces de integração. Com isso, sempre que preciso, qualquer utilizador pode ter acesso aos parâmetros necessários para a correta utilização do sistema ou futura integração com outro sistema.

Capítulo 7

Conclusão e Trabalho Futuro

Os objetivos traçados para este projeto foram cumpridos, quer a nível de planeamento, quer a nível de todo o processo de desenvolvimento. Relativamente ao processo de planeamento da API, foram cumpridas as etapas de análise do modelo clínico *OpenEHR*. Ao nível do desenvolvimento, todos os métodos que permitem gerir EHRs foram cumpridos, mesmo com considerações em alguns campos.

Também no que diz respeito ao sistema final, o balanço é bastante positivo, pois a aplicação desenvolvida contém as funcionalidades propostas.

O objetivo inicial era desenvolver um sistema que permitisse gerir *Electronic Health Records* (EHR) baseados no modelo clínico *OpenEHR* que utilizasse *Web Services*. Pelo exposto anteriormente, acredita-se que esse objetivo inicial foi plenamente cumprido. Algumas das dificuldades encontradas no desenvolvimento do projeto estão relacionadas com a documentação da API do *OpenEHR*. A documentação oficial não dava resposta a todas as questões que foram surgindo e a implementação de algumas funcionalidades, numa fase inicial, foi demorada e exigiu bastante trabalho de pesquisa, até que a equipa se familiarizasse com as terminologias definidas. No entanto, apesar das dificuldades iniciais, considera-se que de um ponto de vista global, o EHR considerado facilita muitos dos aspetos do histórico clínico de um paciente em termos de interoperabilidade.

Como possíveis melhorias ao que foi realizado, refere-se a possibilidade de alguma reorganização e otimizações ao código.

Algumas decisões relativas à estrutura do EHR, por exemplo, os campos considerados, que foram tomadas no início do desenvolvimento, com mais algum conhecimento e experiência, poderiam ter sido diferentes. Um exemplo de uma dessas decisões foi relacionada com as *versions*, ao qual se considerou que existiam já versões. Um outro exemplo é relativo a alguns campos não obrigatórios do EHR, uma vez que a API do *openEHR* não explicita como esses campos eram enviados, e por esse motivo estes não foram considerados na nossa API.

Como trabalho futuro, o grupo mostra-se disponível para continuar a manter o produto, quer a nível de correção erros, implementação de novas funcionalidades e escrita de nova documentação.

Concluindo, atingindo o patamar final da elaboração deste projecto o grupo encontra-se orgulhosos com o produto final aqui desenvolvido e pensa que todo este trabalho valeu realmente a pena, na medida em que enriqueceu o conhecimento de todos os elementos do grupo.

Bibliografia

[1] *Web service: o que é, como funciona, para que serve?* 2016. URL: https://www.opensoft.pt/web-service/ (acedido em 13/06/2021) (ver p. 4).

[2] *What is an electronic health record (EHR)?* 2019. URL: https://www.healthit.gov/faq/what-electronic-health-record-ehr (acedido em 13/06/2021) (ver p. 5).

[3] *What is openEHR?* 2021. URL: https://www.openehr.org/about/vision_and_mission (acedido em 13/06/2021) (ver p. 5).

[4] *O que é API REST?* 2020. URL: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api (acedido em 13/06/2021) (ver p. 6).

[5] *Apache JMeter*. 2020. URL: https://jmeter.apache.org/ (acedido em 13/06/2021) (ver p. 11).