INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Conversor HL7 para FHIR baseado em mapeamentos

RUI PEDRO COSTA MEIRELES
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Conversor HL7 para FHIR baseado em mapeamentos

Rui Pedro Costa Meireles

Porto, outubro 2019

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software, orientado pelo Professor Doutor Nuno Silva.
Supervisor: Pedro Pinto

Resumo

Nos dias de hoje, é crucial para uma organização que desenvolve produtos de *software* manter o seu sistema atualizado e preparado para reagir ao mercado competitivo em que se insere.

A Glintt-HS é uma organização que necessita de se adaptar às necessidades dos seus clientes e às constantes mudanças que vão surgindo no seu negócio, sendo por isso necessário o estudo continuo do mercado.

A maior cota de mercado desta empresa é a área da saúde, e atualmente o protocolo mais usado a nível internacional para a troca de informação entre sistemas é o HL7. Contudo, as tecnologias estão em constante evolução e para responder às necessidades atuais impostas pelas aplicações móveis, a comunidade HL7 criou o protocolo FHIR.

A Glintt tem como objetivo principal, estudar e desenvolver uma solução capaz de transformar as mensagens do protocolo HL7, nos *resources* do protocolo FHIR. Pois atualmente não existe nenhuma solução capaz de o fazer.

Palavras-chave: ESB, Mensagens HL7, FHIR *resources*, *message broker*, integração de sistemas

Abstract

Nowadays, it is crucial for an organization that develops software products to keep its system up to date and ready to respond to the competitive market in which it operates.

Glintt-HS is an organization that needs to adapt to the needs of its customers and to the constant changes that are coming up in their business. Therefore, the continuous study of the market is necessary.

The company's largest market share is healthcare, and currently the most widely used international protocol for information exchange between systems is HL7. However, technologies are constantly evolving and to meet the current needs imposed by mobile applications, the HL7 community has created the FHIR protocol.

Glintt's main objective is to study and develop a solution capable of transforming HL7 protocol messages into FHIR protocol resources, given the fact that currently there is no solution capable of doing it.

Keywords: ESB, Message Broker, HL7 messages, FHIR resources, system integration

Agradecimentos

Gostaria de começar por agradecer aos meus pais, que sempre acreditaram em mim, por todo o apoio, amor e carinho que me deram. Pelo esforço que fizeram para que tudo isto fosse possível de realizar. Sem dúvida que se não fosse tudo isso, não tinha chegado até onde cheguei. Por tudo isso o meu obrigado!

À minha namorada que foi um dos grandes pilares nesta longa jornada, esteve comigo nos bons e nos momentos menos bons. Nunca me deixou desistir, deu-me sempre muita força, carinho, amor e fez-me sempre acreditar em mim mesmo. Dedicou horas a ajudar-me, a apoiar-me, o simples facto de estar presente neste percurso fez toda a diferença. Foi uma verdadeira companheira em tudo isto. Obrigado por fazeres parte de mais uma conquista na minha vida.

Ao meu irmão André quero agradecer por todo o carinho, apoio demonstrado e por acreditar em mim! Por me acompanhar em todas as conquistas da minha vida! Por me ter acompanhado até hoje em tudo! Obrigado!

Quero agradecer ao Dr. Moura Mendes, por todo o trabalho que tem tido comigo, traçando objetivos, obrigando assim a que eu não perdesse o meu rumo, mostrando confiança nas minhas capacidades.

Aos meus amigos, Tiago Tavares, Tiago Moreira, Tiago Esteves, Bruno, Pedro, Diogo, Gameiro, Rui, André, Paulo e Maria por me terem ajudado tanto. Acompanharam todo este meu caminho, deram-me muita força e sempre se demonstraram preocupados e prontos a ajudar. Obrigado!

À minha equipa dos MCDT o meu obrigado, demonstraram sempre intenção em ajudar e sempre que era preciso ficavam mais tempo na empresa para que eu pudesse trabalhar neste projeto. Agradecer ainda ao Carlos Silva e José Melo por toda ajuda no desenvolvimento do projeto e esclarecimento de dúvidas.

Agradecer ao Pedro Pinto, pela orientação e por toda a ajuda disponibilizada durante o projeto. Ao Francisco Vicente e ao Francisco Correia por me dispensarem tempo para o desenvolvimento do projeto no meu horário laboral.

Por fim quero agradecer ao meu orientador, Nuno Silva, pela disponibilidade, preocupação e pela confiança que me deu durante todo este projeto. Foi crucial toda a paciência que teve comigo e confiança.

Gratidão a todos!

Índice

Índice de figuras

Índice de Tabelas

Lista de acrónimos

NCD	*New Concept Development*
ESB	*Enterprise Service Bus*
MLLP	*Minimal Lower Layer Protocol*
REST	*Representational State Transfer*
SOAP	*Simple Object Access Protocol*
API	*Application Programming Interface*
AS	Arquitetura de software
MOM	*Message Oriented Middleware*
AMQP	*Advanced Message Queuing Protocol*

1 Introdução

Neste capítulo são apresentados:

- o contexto do projeto-estágio;
- os problemas identificados;
- os objetivos do projeto-estágio:
- abordagem técnico-científica;
- estrutura do documento.

1.1 Contexto

Esta secção apresenta os conceitos importantes inerentes ao projeto, como a interoperabilidade e sua importância para o negócio. Além disso, é explicado o modelo de negócio da *Glintt* e o modo como operam no desenvolvimento de *software* e em particular o foco do projeto a desenvolver.

1.1.1 Glintt

Tal como refere a própria empresa: "*A Glintt – Global Intelligent Technologies - é uma multinacional de tecnologia e consultoria, cotada na Euronext*." [1]

A *Glintt Healthcare Solutions*, S.A. (doravante referida como Glintt) surge a 11 de junho de 2008 em consequência de uma fusão entre duas grandes empresas que se encontravam a operar no setor da saúde em Portugal: a ParaRede e a Consiste. No que diz respeito à sua estrutura física, a *Glintt* opera em diferentes países tendo os seus escritórios sediados em localidades como: Portugal, Espanha, Angola, Brasil, Reino Unido e Irlanda.

Apesar do seu início em 2008 a *Glintt* apresenta mais de 20 anos de experiência em consultoria e serviços tecnológicos na área da saúde devido ao *know-how* que é trazido das empresas anteriores à sua fusão.

De forma concreta, esta empresa traz soluções a mais de 200 hospitais e clínicas em diferentes países, por outro lado, no que diz respeito ao setor farmacêutico, mais de 1600 farmácias na península ibérica usam o software de gestão da *Glintt*.

1.1.2 Negócio

A *Glintt* possui um grande leque de aplicações para responder às necessidades de todos os seus clientes. Como tal, oferece aplicações com grande interoperabilidade, capazes de comunicar com aplicações externas.

Atualmente, nas unidades hospitalares existem várias aplicações ligadas a um ramo específico da saúde (e.g. Laboratórios, prescrição de exames, administrativos, etc.). Estas aplicações podem ou não ser desenvolvidas pela Glintt.

1.2 Problema

Atualmente, o protocolo utilizado para a troca de informação é o HL7 v2. O HL7 v2 é um protocolo reconhecido a nível internacional para a troca de dados entre sistemas de informação ligados à saúde. Na secção 2.2.1.1.1 vai ser detalhado este protocolo.

O protocolo HL7 v2 apenas permite a comunicação através de *event-driven*, e o desenvolvimento de aplicações utilizando HL7 é muito custoso e de difícil implementação. A comunidade HL7, criou assim o FHIR, criando novas abordagens no tipo de comunicação entre sistemas, através de uma abordagem REST.

O FHIR é um protocolo utilizado para troca de informação tal como HL7 v2, permite também comunicação através de *event-driven*. Contudo, a sua principal abordagem é através de REST. Este protocolo irá ser descrito detalhadamente na secção 2.2.1.1.2.

Atualmente, a Glintt está a desenvolver uma API FHIR, uma API é um software intermediário que permite que duas aplicações comuniquem entre si. Nesta API vai existir um repositório de *resources* FHIR, que inicialmente apenas irá ser alimentado por aplicações que comuniquem através de FHIR. Contudo, é pretendido que haja uma conversão entre mensagens HL7 e *resources* FHIR, para assim tornar possível alimentar este repositório através de aplicações *legacy* (que usam HL7) e aplicações atuais (que usam FHIR) porque os dois protocolos não são, nativamente, compatíveis.

Atualmente não existe nenhum software capaz de fazer esta conversão, pelo que será necessário estudar a forma de o fazer, e desenvolver uma solução capaz de dar resposta a essa necessidade.

1.3 Objetivos

O objetivo é o desenvolvimento dum software que permita efetuar conversões entre diferentes tipos de protocolos de transferências de informação na área da saúde.

Como um dos objetivos da *Glintt* é expandir o seu mercado de atuação a nível nacional e internacional, é pretendido que a solução arquitetural a desenvolver consiga dar resposta quer ao mercado nacional, quer ao mercado internacional.
É por isso necessário que a arquitetura seja adaptável (no sentido em que deva permitir extensão de integração) de outros protocolos. Contudo, neste projeto-estágio, pretende-se concretizar a arquitetura do sistema com a integração entre HL7 v2 e FHIR.
Existem ainda requisitos não funcionais a serem cumpridos:

- Confiabilidade;
- Processamento assíncrono;
- Alta performance;
- Adaptabilidade;
- Segurança;
- Processamento da transformação de forma assíncrona.

1.4 Abordagem

A abordagem arquitetural preconizada utilizará um sistema de integração intermédio capaz de receber e interpretar os diferentes tipos de mensagens dos vários protocolos suportados, e converter as mesmas para o formato desejado.
Em particular, neste projeto, os protocolos fonte é o HL7 v2 e protocolo destino é o FHIR.
A transformação de mensagens será conduzida por mapeamentos declarativos definidos por peritos no domínio, e aplicado em tempo de execução pelo componente de transformação do sistema de integração intermédio.
Para que haja garantia de entrega, adotou-se ainda um sistema de filas de mesnagens, capaz de as armazenar e garantir o seu processamento.

1.5 Estrutura do documento

No primeiro capítulo é descrito todo o âmbito em que projeto está envolvido e é feita uma breve apresentação da organização Glintt. É ainda descrito o problema, objetivos e abordagem adotada.
No segundo capítulo é feito um enquadramento técnico e teórico acerca das tecnologias e boas praticas adotadas, que permite um melhor entendimento de conceitos-chave e ajuda a aferir o conhecimento necessário para a leitura dos capítulos seguintes.

No terceiro capítulo é realizada uma análise de valor quanto ao projeto a desenvolver.
No quarto capítulo são dados a conhecer as mensagens HL7 e os *resources* que irão ser estudados para a conversão.
No quinto capítulo é apresentado o design arquitetural do software, sendo que algumas situações serão demonstradas com mais detalhe, recorrendo ao modelo de representação 4+1 e a diagramas em notação UML a nível de sistema e a nível de componentes. É também apresentado neste capítulo um design concetual quanto ao mapeamento e à transformação necessário entre os dois protocolos anteriormente mencionados.
No sexto capítulo sintetiza-se o trabalho de implementação do sistema descrito no capítulo 5, descrevendo boas práticas adotadas e detalhes de implementação.
No sétimo capítulo é descrito a maneira como foram abordados e executados os testes à aplicação.
No último capítulo é feita uma síntese de todo o trabalho onde são descritos todos os objetivos atingidos, conclusões obtidas com a implementação, limitações encontradas e trabalho futuro.

2 Contexto

Neste capítulo é apresentada uma visão geral do projeto, onde são discutidos conceitos e tecnologias importantes relacionados com o problema, numa perspetiva de negócio e numa perspetiva tecnológica.

2.1 Contexto de Negócio

Nesta secção irá ser averiguada a área de negócio onde o projeto está inserido e como a tecnologia tem influenciado a área da saúde. Será feita uma análise à evolução da tecnologia na área da saúde e de que forma este projeto irá contribuir para o desenvolvimento de aplicações nesta área.

2.1.1 Saúde

A área da saúde relaciona-se com a prestação de serviços, no âmbito privado ou público, de apoio ao diagnóstico, educação para a saúde, proteção e prevenção, recuperação e reabilitação e gestão realizados por profissionais das diferentes subáreas.
Atualmente em Portugal, a área da saúde é constituída por três sistemas:

- Serviço Nacional de Saúde (SNS), é o principal sistema criado pelo estado com o objetivo de garantir o direito à proteção da saúde de todos os cidadãos e dos estrangeiros que visitem ou residam no país [2];
- Subsistemas públicos e privados, sendo que os primeiros são responsáveis pelos cuidados de saúde aos empregados públicos e os segundos aos membros de organizações privadas. O subsistema público mais importante é a ADSE, a qual cobre todos os funcionários públicos não cobertos por qualquer outro subsistema especial. Possuí mais de 1,3 milhões de beneficiários, incluindo servidores de órgãos da administração pública central, regional e local [2].
  Os subsistemas privados de saúde cobrem quer o conjunto de membros de certas profissões, independentemente do seu empregador, quer o conjunto de empregados de grandes empresas ou outras organizações privadas.
- Seguros de saúde, complementam tanto o SNS como os subsistemas de saúde. Caracterizam-se por serem privados e de adesão normalmente voluntária, apesar de existirem também seguros de saúde obrigatórios [2].

A informática na saúde ou informática médica, surgiu por volta de 1974, caracterizando a relação entre as ciências da informação, da computação e da medicina com o objetivo de investigação de dispositivos, tecnologias e métodos de tratamento de informação relacionada com a prestação de cuidados de saúde. Esta disciplina é essencial para a criação de sistemas de informação na área clínica, usados para o conhecimento da realidade, para o planeamento e para a avaliação de políticas de saúde.

2.1.2 Sistemas de informação

"Um sistema de informação, automatizado ou manual, abrange pessoas, máquinas e/ou métodos que são organizados para processar e transmitir dados que representam informação para o utilizador. É um conjunto de componentes integrados para recolher, armazenar e processar dados e providenciar informação, conhecimento e produtos digitais" [3].

Para uma organização é estritamente necessário o uso de um sistema de informação. Pois este permite a gestão das suas operações, promove a interação entre clientes e fornecedores, e aumenta a competitividade no mercado.

Existem várias vantagens associadas à utilização de sistemas de informação por parte das organizações:

- Maior segurança de acesso à informação;
- Maior estabilidade;
- Maior integridade;
- Redução de custos operacionais;
- Otimização de fluxo de informação.

Apesar das vantagens enumeradas, é necessário perceber que estes sistemas possuem igualmente desvantagens:

- Altos custos que por muitas vezes não compensam o custo/benefício;
- Excesso de controlo sobre as pessoas;
- São vulneráveis a fraudes e a invasões de sistema.

Os sistemas de informação podem ser classificados de acordo com a informação que é processada [4]:

- Operacional – Gerem a informação referente a transações da organização;

- Tático – Agrupam e sintetizam os dados das operações da organização de modo a facilitar a tomada de decisão dos gestores;
- Estratégicos – Integram dados de fontes internas e externas à organização, utilizando ferramentas de análise e comparação complexas.

Os sistemas acima enunciados, representam uma hierarquia em pirâmide, tendo níveis diferenciados dependendo da sua importância. No topo da lista encontra-se o sistema estratégico, pois este lida com informações altamente privilegiadas e de grande impacto na organização. Em seguida, encontra-se o sistema de gestão tático, este lida com informações classificadas como nível de média importância. Por fim, o sistema operacional que trabalha com informações operacionais [5]. A figura ilustra os níveis referidos:

Figura 1 - Hierarquia Sistemas de Informação [5]

Esta hierarquia ilustrada na Figura 1, é a utilizada na Glintt.

2.1.3 Sistemas de informação na saúde

"Os sistemas de informação na saúde agregam um conjunto de dados, informações e conhecimentos de acordo com uma determinada área de Saúde específica, apoiando o planeamento e processo de tomada de decisão dos profissionais e dos utentes" [6].

A implementação destes sistemas de informação na área da saúde tem normalmente pelo menos um dos seguintes objetivos:

- Administrativos: pretende-se registar os dados demográficos dos doentes e também dados do funcionamento da instituição;
- Clínicos: pretende-se registar os dados de saúde dos utentes;
- Financeiros: pretende-se registar os dados financeiros como custos ou receitas de serviços prestados;
- Stocks: pretende-se gerir os stocks de uma instituição.

2.2 Contexto Tecnológico

Nesta secção irá ser averiguada a área de negócio onde o projeto está inserido e como a tecnologia tem influenciado a área da saúde. Será feita uma análise à evolução da tecnologia na área da saúde e de que forma este projeto irá contribuir para o desenvolvimento de aplicações nesta área.

2.2.1 Protocolos

Esta secção irá ser dividida em duas grandes secções:

- Protocolos de representação de informação: aqui irão ser descritos os dois protocolos que são utilizados para representar a informação que se pretende comunicar;
- Protocolos de transferência de informação: nesta irão ser apresentados e detalhados os protocolos utilizados para a transferência da informação.

2.2.1.1 Protocolos de representação de informação

Ao longo do documento têm vindo a ser mencionados os protocolos HL7 e FHIR. Nesta secção irão ser dados a conhecer cada um deles detalhadamente.

2.2.1.1.1 Protocolo HL7

"O HL7 é um organização que desenvolve padrões de dados para armazenamento e troca de informação em toda a indústria *healthcare*. Tem como missão fornecer uma *framework* e standards de troca, integração, partilha e recuperação de informação eletrónica de saúde que apoie a prática clínica e a gestão, entrega e avaliação de serviços de saúde" [7]. Criar serviços flexíveis, padrões económicos, diretrizes e metodologias de forma a permitir uma maior interoperabilidade e partilha de dados existentes num determinado sistema de informação de saúde.

"Os padrões HL7 abrangem todos os aspetos clínicos e administrativos presentes no setor da saúde, incluindo laboratórios, registros médicos, atendimento de um paciente, farmácia, relatórios de saúde pública, estudos regulamentados, contas e faturação, reclamações e reembolso, administração do paciente e agendamento de consultas" [7].

As mensagens HL7 v2 contêm várias informações do âmbito clínico tais como: consultas médicas, resultados de exames de laboratório, observações do paciente, etc. Na Figura 2 está ilustrado um exemplo de uma mensagem onde se encontram as informações dos resultados de um teste laboratorial:

```
MSH|^~\&|||||19941122100053||ORU^M01|
EVN|M01|1994111181141|
PID|||661041||GARDNER^REED^M|
PV1||I|E7^703^^LDS|
OBR||^A000520|LYTES^Serum Electrolytes|
OBX|1|NM|NAS^Serum Sodium|1|138|mmol/L|
OBX|2|NM|K^Serum Potassium|1|3.2|mmol/L|
OBX|3|NM|CL^Serum Chloride|1|114|mmol/L|
OBX|4|NM|CO2^Serum CO2|1|24|mmol/L|
```

Figura 2 - Exemplo de uma mensagem HL7 v2 [7]

Cada mensagem consiste em vários segmentos, representados por linhas tal como é demonstrado na figura. Cada segmento é iniciado por um código composto de três letras que define qual o tipo de segmento [7]. Os segmentos HL7 v2 são divididos em campos de dados separados por "|", em que cada campo contém os valores dos campos. Estes podem estar em conformidade com qualquer um dos vários tipos de dados, incluindo caracteres de texto, como por exemplo o nome do paciente ou informações como um resultado de um exame ao sangue. Tipos de dados complexos, incluído dois ou mais dados, também podem ser definidos. Os campos podem ser definidos como "Obrigatório" ou "Opcional" e podem ser repetidos várias vezes.

Analisando a figura é possível verificar o seguinte conteúdo:

1. O primeiro segmento é caracterizado o cabeçalho da mensagem, denotado por MSH (*Message Header*). Neste segmento é especificado qual o tipo de mensagem que está a ser enviada, que neste caso é do tipo ORU (*Observation Result*).
2. O segmento EVN (*Event Type*) é onde se encontra toda a informação quanto ao tipo de evento gerado.
3. O segmento PID (*Patient Identifier*) contêm a informação relativa a um paciente.

4. O segmento PV1 (*Patient Visit*), contém informação relativa à visita de um paciente.
5. Por fim o segmento OBX (*Observation*) contém a informação das observações do paciente. Existem quatro segmentos de observação (OBX1-4).

Na Tabela 1 descreve em detalhe o significado de cada símbolo utilizado nas mensagens HL7 v2.

Tabela 1 - Tabela ilustrativa do significado de cada caractere usado nas mensagens (Cohen & Ram, Consultado em 14 fevereiro 2019)

Caractere	Significado
|	Separador dos campos da mensagem.
^	Separador de componentes num campo.
~	Separador de repetição num campo.
\	Separador usado para ignorar carateres especiais.
&	Separa componentes dentro de outros componentes.

2.2.1.1.1.1 Vantagens

A grande vantagem em utilizar o protocolo HL7 v2 na troca de informação entre aplicações do ramo da saúde, é que este é o mais utilizado a nível mundial. Sendo que é o mais normalizado e com definições cimentadas aos dias de hoje.

2.2.1.1.1.2 Desvantagens

O maior problema segundo diz a comunidade HL7 é "o padrão HL7 exige adaptação em cada aplicação em que é usado. Um dos principais desafios nessa área é o alinhamento de vocabulários e modelos de dados. A maneira como as informações são descritas e formatadas nas aplicações - mesmo aqueles projetados para usar o HL7 desde o primeiro dia - pode representar um desafio para sua interoperabilidade bem-sucedida. Isso é especialmente verdade naquelas áreas dentro do padrão em que vários métodos aceitáveis de implementar o mesmo gatilho de evento ou outro componente estão em uso" [8, p. 7].

2.2.1.1.2 Protocolo FHIR

Em 2011 a comunidade HL7 reconheceu que o HL7 versão 3 não estava a ir de encontro às necessidades do mercado. Com o objetivo de melhorar a interoperabilidade, foi criada uma

especificação *Resources For Health* para o HL7, denominada por "*Fast Healthcare Interoperability Resources*" (FHIR) [9].

O foco do FHIR é o fornecimento de uma API simples, fácil de implementar e gerir. É inspirada nas API web contemporâneas. Foram definidos modelos para suportar a troca de informação, ou seja, foram criados modelos baseados no tipo de mensagem como por exemplo o *Patient.*

A API FHIR é uma interface HTTP contemporânea orientada a recursos FHIR para efetuar operações CRUD (*create, read, update* e *delete*), e usa XML e JSON para serializar os recursos.

Estes recursos podem representar dados clínicos, administrativos e de infraestrutura, incluindo consultas de pacientes únicos, consultas a nível de população. Cada recurso representa uma estrutura de dados, expressa em campos e tipos de dados bem definidos [9].

As definições destes recursos clínicos são relativas a conceitos concretos e intuitivos, como prescrição de medicamentos, reações adversas, prescrição de exames, etc. Por exemplo, um *resource MedicationPrescription* possuí informação quanto ao prescritor (FHIR *Practitioner*), ao paciente (FHIR *Patient*) e qual o medicamento prescrito (FHIR *Medication*).

```
<?xml version="1.0" encoding="UTF-8"?>
<Patient xmlns="http://hl7.org/fhir">
  <id value="f201"/>
  <identifier>
  <!--    The identifier for the person as this patient (fictive)    -->
    <use value="official"/>
    <type>
      <text value="BSN"/>
    </type>
    <system value="urn:oid:2.16.840.1.113883.2.4.6.3"/>
    <value value="123456789"/>
  </identifier>
<!--    Demographics    -->
  <identifier>
  <!--    The identifier for this individual    -->
    <use value="official"/>
    <type>
      <text value="BSN"/>
    </type>
    <system value="urn:oid:2.16.840.1.113883.2.4.6.3"/>
    <value value="123456789"/>
  </identifier>
<!--    Indicates that the patient is not part of a multiple birth    -->
  <active value="true"/>
  <name>
  <!--    The name associated with the individual (fictive)    -->
    <use value="official"/>
    <text value="Roel"/>
    <family value="Bor"/>
    <given value="Roelof Olaf"/>
    <prefix value="Drs."/>
    <suffix value="PDEng."/>
  </name>
  <telecom>
  <!--    The mobile contact detail for the individual    -->
    <system value="phone"/>
    <value value="+31612345678"/>
    <use value="mobile"/>
  </telecom>
  <telecom>
  <!--    The home contact detail for the individual    -->
    <system value="phone"/>
    <value value="+31201234567"/>
    <use value="home"/>
  </telecom>
  <gender value="male"/>
  <birthDate value="1960-03-13"/>
<!--    The date and time of birth for the individual    -->
  <deceasedBoolean value="false"/>
<!--    Indicates that the individual is not deceased    -->
```

Figura 3 - Exemplo de um resource Patient

Na Figura 3 é possível observar um excerto de um exemplo de um *resource Patient*. Verificamos que é composto por vários campos e conseguimos ter uma perceção de quais os campos existentes, ao contrário do que acontece no HL7.

Contudo, em FHIR não são descritos modelos detalhados e exaustivos para cada aspeto de um recurso/conceito, mas, por contraponto, fornece um mecanismo de extensibilidade incorporado para enriquecer as definições de recursos existentes à medida das necessidades do negócio específico.

Utilizando o exemplo anteriormente apresentado, existem informações do paciente que não estão presentes no *resource Patient* como é o caso da Nacionalidade.

```
<!-- nationality -->

<extension xmlns="http://hl7.org/fhir"
     url="http://hl7.org/fhir/StructureDefinition/patient-nationality" >
 <-- extension sliced by value:url  in the specified orderOpen-->
 <extension url="code"> 0..1 Extension  <!-- 0..1 Nationality Code -->
  <valueCodeableConcept><!-- 0..1 CodeableConcept
      Value of extension --></valueCodeableConcept>
 </extension>
 <extension url="period"> 0..1 Extension  <!-- 0..1 Nationality Period -->
  <valuePeriod><!-- 0..1 Period
      Value of extension --></valuePeriod>
 </extension>
</extension>
```

Figura 4 - Exemplo de uma extensão para Patient

Na Figura 4 é possível observar o exemplo de uma extensão criada e definida pela comunidade HL7 FHIR para a nacionalidade do paciente.

FHIR adota o padrão/paradigma arquitetural *RESTful* como principal paradigma. Contudo, este também possuí *event-driven*.

"Os sistemas REST, conforme descrito por Fielding (geralmente chamados de arquiteturas *RESTful*), foram recentemente adotados como abstração de informações dominantes da *WorldWideWeb*" [10, p. 3].

A orientação a recursos e *statelessness* do protocolo HTTP são responsáveis pela natureza altamente escalável descentralizada da *web*.

Recentemente, as principais plataformas de serviços de internet migraram para serviços *RESTful*, afastando-se de outras opções de arquitetura, como as abordagens SOAP. Uma implementação de REST segue quatro princípios básicos de design:

- Usa métodos HTTP explicitamente;
- É *stateless*;
- Expõem a estrutura de diretórios como URI aos recursos;
- Serializa os recursos em linguagem XML ou JSON.

No contexto do FHIR, a especificação descreve os seguintes atributos de recursos:

- Devem ter um limite claro, que corresponda a um ou mais *scopes* de transação lógica;
- Devem diferir uns dos outros em termos de significado, não apenas no uso (por exemplo, maneiras diferentes de usar um relatório de laboratório não devem resultar em recursos diferentes);
- Precisam de ter uma identidade natural;
- Devem ser muito comuns e usados em muitas transações diferentes;
- Devem ser específicos ou detalhados o suficiente para impedir o suporte a uma ampla gama de transações comerciais;
- Devem ser mutuamente exclusivos;
- Devem usar outros recursos, mas devem ser mais do que apenas composições de outros. Cada recurso deve introduzir um novo conteúdo;
- Devem ser grandes o suficiente para fornecer contexto significativo. Os recursos que contêm apenas alguns atributos provavelmente são muito pequenos para fornecer valor comercial significativo.

"As vantagens práticas das arquiteturas RESTful incluem interfaces leves que permitem transmissão e processamento mais rápidos de estrutura de dados e mais adequadas para dispositivos móveis. Estas interfaces permitem ainda ciclos de desenvolvimento mais fáceis e mais rápidos através das suas estruturas simples" [10, p. 3].

2.2.1.1.2.1 Críticas ao FHIR

O FHIR é orientado a recursos, permite que a implementação seja muito simples de artefactos básicos, a sua transmissão e persistência de informação. No entanto, visto que é um protocolo recente e em crescimento, há muita pouca orientação sobre como esses recursos básicos devem ser construídos em *collections* e *relationships* de maior escala.
Além disso, há pouco ou nenhum suporte para o fluxo de trabalho e comportamento dinâmico além das operações básicas de CRUD (*Create, Read, Update* e *Delete*). Isto pode fazer com que se torne numa área de divergência, podendo levar à falta de interoperabilidade [10].
Existes ainda muitas dúvidas entre agregações de recursos e documentos que ainda não estão definidos.

2.2.1.1.3 Comparação entre HL7 v2 e HL7 FHIR

Na Tabela 2 é possível observar as semelhanças e diferenças entre o protocolo HL7 v2 e o HL7 FHIR, que são os dois protocolos estudados no âmbito desta dissertação:

Tabela 2 - Comparação entre HL7 v2 e FHIR resource [10]

Critérios de comparação	**HL7 v2**	**HL7 FHIR**
Ano de iniciação	1987	2011
Processo de desenvolvimento/metodologias	*Bottom-up/ ad hoc*	Iterativa e incremental
Paradigmas de arquitetura	Mensagens e registos	*RESTful*
Sobrecarga de aprendizagem	Ordem de semanas	Ordem de semanas
Extensibilidade	Fornece um mecanismo através do uso *"Z-segments"*. As extensões devem ser restritas a elementos de dados que não afetam o significado dos segmentos padrão.	Extensões do FHIR, podem aparecer em qualquer nível (incluindo nos tipos de dados).
Compatibilidade entre versões	Possuí processos rigorosos para manter a compatibilidade com versões anteriores e posteriores. O conteúdo só pode ser adicionado no final de campos, componentes, etc. É esperado que as aplicações ignorem o conteúdo ou repetições inesperadas.	Promete regras de compatibilidade semelhantes. O caminho para um elemento dentro de uma instância do FHIR permanecerá inalterado em versões futuras.
Leitura humana	De uma forma geral, não é legível para o ser humano o conteúdo de uma mensagem.	Exige que o conteúdo seja legível ao ser humano.
Opcionalidade e Perfis	Maioria dos elementos de dados é opcional. Incluí muitos elementos que são usados apenas em	Maioria dos elementos de dados é opcional. Recursos FHIR são mais limitados quanto aos termos de quais elementos estão

	circunstâncias muito limitadas.	incluídos na especificação principal. Para responder a esse problema são utilizadas extensões.
São necessárias ferramentas especializadas?	Sim	Não
Diretamente consumível?	Sim	Sim
Tamanho da especificação	Centenas de páginas	Centenas de páginas
Exemplos de implementação na especificação	Sim	Sim
Referencias de implementações disponibilizadas pelo HL7	Não	Sim
Suporte da comunidade	Forte	Muito recente
Inerentemente adequado para dispositivos móveis	Não	Sim
Número de tipos de mensagem	?	30
Grau de adoção	Muito alto	Não conclusivo
Tipo de modelo de informação	*Ad hoc*	Não conclusivo
Suporte internacional de caracteres	Não (ASCII)	Sim (UTF8)
Formato de mensagem internacional	Suporta padrão global único	Suporta padrão global único

Um dos maiores desafios da interoperabilidade do HL7 v2 é a variação da implementação. Mesmo quando os cenários são idênticos, os elementos de dados suportados podem variar, até mesmo o local em que um determinado elemento é colocado. Como resultado, definir regras de mapeamento consistentes entre HL7 v2 e FHIR a um nível internacional ou até mesmo regional é de elevada dificuldade. Os mapeamentos FHIR fornecem um ponto de partida a ter em consideração, mas estes geralmente necessitam de ser feitos consoante a base de implementação [11].

2.2.1.2 Protocolos de transferência de informação

Neste projeto foram estudados dois protocolos de transferência de informação:

- RESTful;

- SOAP.

A comunicação feita através do WS HL7 pode ser feita através de qualquer um dos protocolos mencionados anteriormente. Por sua vez, a comunicação feita para o envio de informação para o WS FHIR será sempre através de REST.

2.2.1.2.1 RESTful

Os sistemas RESTful, são caracterizados por serem *stateless* e separam a lógica do cliente e do servidor. Estes sistemas utilizam o estilo arquitetural REST (*Representational State Transfer*). Este estilo fornece padrões entre sistemas de computadores na *web*, facilitando a comunicação entre os mesmos [12].

No estilo arquitetural REST, a implementação do cliente e a implementação do servidor podem ser feitas de forma independente, sem que nenhuma das partes tenha conhecimento da outra. Isto permite que o código no lado do cliente pode ser alterado sem que tenha qualquer impacto no lado do servidor. Para garantir a modularidade e a separação entre cliente e servidor apenas é necessário que cada um conheça o formato de mensagens a enviar para o outro.

Tal como foi mencionado anteriormente, os sistemas que seguem o paradigma REST são *stateless*, o que significa que o servidor não precisa de saber nada sobre o estado do cliente e vice-versa. Desta forma, o servidor e o cliente podem receber e compreender qualquer mensagem, sem a necessidade de ver o estado das mensagens anteriores.

Existem vantagens associadas à utilização de REST:

- Permite o formato JSON, XML, entre outros. Com o JSON que normalmente funciona melhor com dados e oferece análise mais rápida, o REST é geralmente considerado mais fácil de trabalhar;
- O REST fornece desempenho superior, principalmente por meio do armazenamento em cache de informações que não são alteradas e não são dinâmicas;
- É o protocolo mais usado em grandes serviços como Yahoo, Ebay, Amazon e até mesmo a Google;
- O REST geralmente é mais rápido e usa menos largura de banda.

Quanto às desvantagens é possível enumerar as seguintes:

- O REST apenas trabalha com o protocolo HTTP;

2.2.1.2.2 SOAP

"O SOAP (*Simple Object Access Protocol*) é um protocolo de mensagens que permite que programas executados em sistemas operacionais diferentes (como *Windows* e *Linux*) comuniquem através de HTTP (*Hypertext Transfer Protocol*) e XML (*Extensible Markup Language*)" [13].

Visto que os protocolos da web estão instalados e disponíveis para uso por todas as principais plataformas de sistemas operativos, o HTTP e o XML fornecem uma solução disponível que permite que os programas executados em diferentes sistemas operativos numa rede comuniquem entre si. "Apesar de que o seu emparelhamento frequente seja feito com HTTP, é possível utilizar outros protocolos de transporte"[13]

Existem vantagens associadas à utilização do SOAP:

- É uma plataforma e linguagem independente;
- Fornece comunicações simplificadas através de *proxies* e *firewalls*;
- Tem capacidade de aproveitar diferentes protocolos de transporte, incluindo HTTP, SMTP e outros.

Quanto às desvantagens é possível enumerar as seguintes:

- Normalmente é muito mais lento que os outros tipos de padrões *middleware*.
- Tem diferentes níveis de suporte, dependendo da comunidade/linguagem/plataforma utilizada. Por exemplo, o suporte SOAP junto da comunidade *Python* e PHP não é tão forte quanto é nas comunidades Java e .NET.

2.2.2 Sistemas de integração

"Um sistema de integração (SI) é um processo de engenharia relacionado com a junção de diferentes subsistemas ou componentes, num único sistema. Certifica-se que cada subsistema integrado funciona conforme a necessidade do mesmo" [14].

Permite valorizar o sistema existente com novas funcionalidades, provenientes da conexão de funções de diferentes sistemas.

"Na última década, a agregação de sistemas ou subsistemas de componentes diferentes que cooperam para fornecer uma funcionalidade completa tem sido o foco de indústrias que usam a tecnologia" [14].

Existem dois estilos principais são utilizados em integração de sistemas:

- Integração horizontal, envolve a criação de um único subsistema com o objetivo de ser a interface única entre todos os outros subsistemas, garantido que haja apenas uma interface entre qualquer subsistema, em que qualquer um pode ser substituído por outro sem afetar os restantes;
- Integração vertical, os subsistemas são integrados de acordo com a funcionalidade, criando "silos" de entidades funcionais. Começando com a função mais básica inferior para cima (vertical). Este método é muito rápido e envolve apenas alguns fornecedores e desenvolvedores, mas com o passar do tempo torna-se mais caro implementar novas funcionalidades, novos silos devem ser criados;

Nas secções seguintes, são apresentadas tecnologias de suporte às descrições dos pontos anteriores.

2.2.2.1 Enterprise Service Bus (ESB)

"O termo *Enterprise Service Bus* foi dado por Gartner em 2002 e apresentado pelo analista Roy Schulte para descrever uma categoria de produtos de software que ele observou estarem disponíveis no mercado naquele momento. Atualmente, ainda não existe uma definição universal de ESB, sendo que esta varia consoante a fonte ou o negócio onde se encontra inserido" [15]. A definição que mais se adequa a esta dissertação é a seguinte: "É um sistema arquitetural que possibilita a interoperabilidade entre ambientes heterogêneos" [15].

Na figura seguinte pode-se observar uma arquitetura que ilustra a ligação entre um ESB com os respetivos sistemas ou aplicações.

Um ESB é essencialmente uma arquitetura, definida por um conjunto de regras e princípios para integrar diferentes aplicações. Observando a figura anterior, verifica-se que o ESB é a peça central de todo o sistema, que permite integrar um sistema com outras aplicações, serviços diversos, comunicar com base de dados de diferentes tipos, etc. Com a utilização de um ESB, os sistemas ficam totalmente pouco desacoplados entre si, permitindo, apesar disso, que comuniquem bilateralmente, com menores dependências ou conhecimento dos outros sistemas que fazem parte do ESB, e dependendo menos de alterações que ocorrem nos restantes.

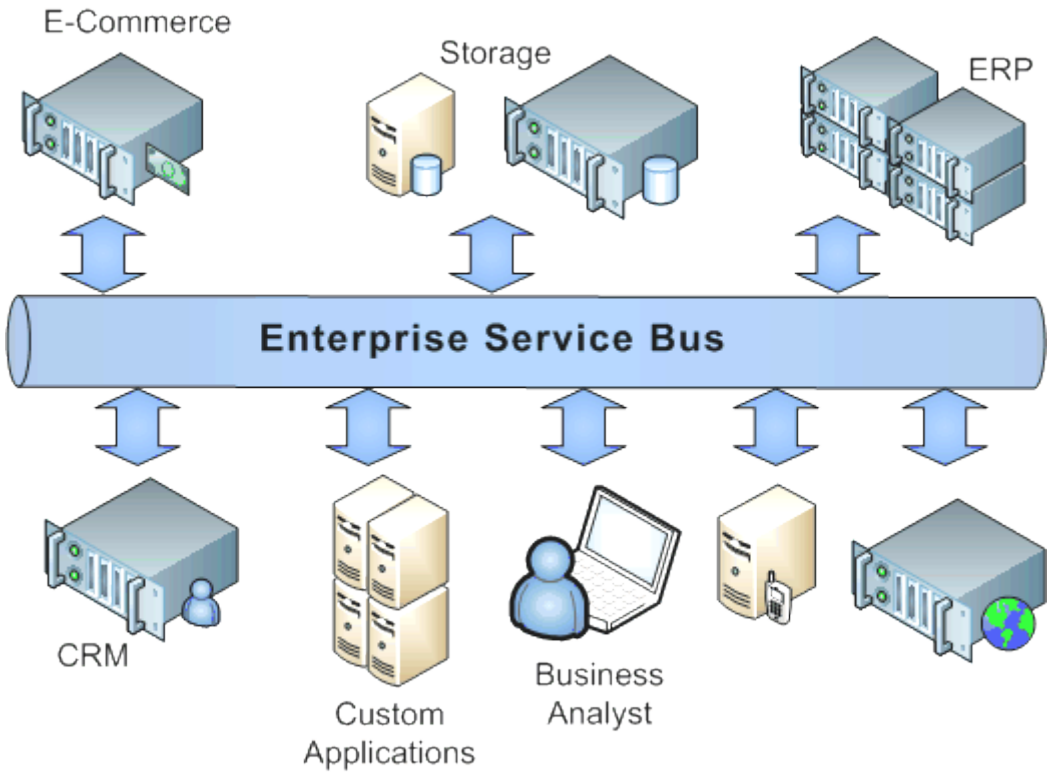

Figura 5 - Arquitetura de um sistema com utilização de um ESB [16]

Foi pela necessidade de se afastar a integração ponto-a-ponto, que este conceito surgiu. A integração ponto-a-ponto resulta num código de integração personalizado espalhado entre aplicações sem nenhuma maneira central de monitorizar ou solucionar problemas. Com o tempo torna-se frágil e difícil de gerir toda esta arquitetura, é muitas vezes referido pela como metáfora "point-to-point hell" semelhante à metáfora *"spaghetti code"* em codificação/programação., pois torna-se difícil de escalar o sistema dadas as dependências rígidas entre os diferentes componentes.

Tal como já foi mencionado anteriormente, "a principal razão da utilização de um ESB é aumentar a agilidade organizacional, reduzindo o tempo e facilitando a integração de novos sistemas. Permite que as empresas melhorem os seus produtos com as vantagens de comunicação e transformação de mensagens fornecidas pelos ESB. Esta agilidade é suportada pela capacidade do ESB de integrar perfeitamente aplicações, serviços, dados e processos em sistemas na nuvem, dispositivos móveis, etc." [15].

Na implementação de um ESB existem alguns conceitos que se deve ter em conta, no momento de escolher a arquitetura a adotar. Em seguida são descritos alguns desses conceitos agrupados pelos seguintes termos:

- **Orquestração**, criar um serviço composto por vários componentes existentes de granularidade fina. Existe um processo central que controla e coordena os restantes processos, ao contrário da coreografia onde não existe um processo central a coordenar, cada serviço necessita de interagir com outros serviços de forma ordenada. Na orquestração é promovida a reutilização e a capacidade de gestão dos componentes subjacentes;
- **Coreografia,** "descreve uma colaboração entre um conjunto de serviços para atingir um objetivo comum. Captura as interações nas quais os serviços participantes se envolvem para atingir essa meta e as dependências entre as interações. Também incluem dependências de fluxo de controle (por exemplo, uma determinada interação deve ocorrer antes de outra)" [17];

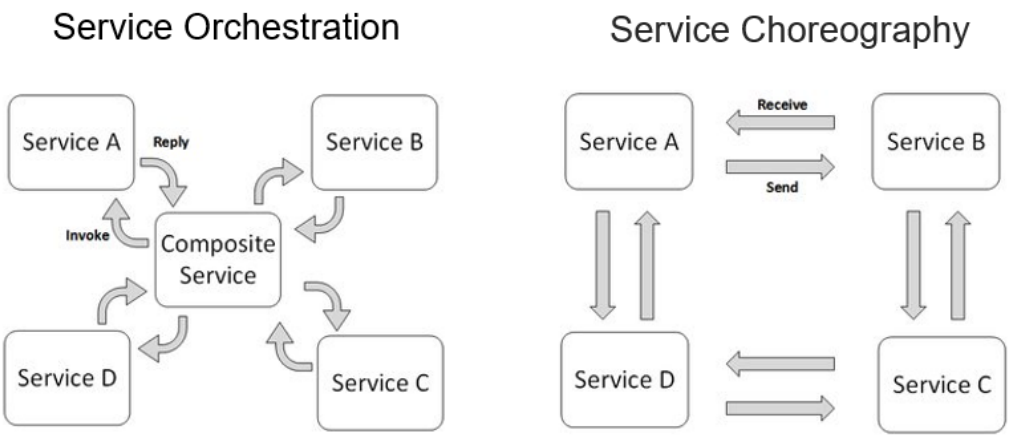

Figura 6 - Orquestração vs. Coreografia [18]

- **Transformação**, transformação de dados entre formatos de dados canônicos e formatos de dados específicos exigidos por cada conector ESB, como por exemplo uma simples transformação de XML para JSON. Os formatos de dados canônicos podem simplificar consideravelmente os requisitos de transformação associados a uma grande implementação ESB, onde há muitos consumidores e provedores, cada um com seus próprios formatos e definições de dados;
- **Transporte**, negociação do protocolo de transporte entre vários formatos (como HTTP, JMS, JDBC, etc.);
- **Mediação**, fornecer interfaces múltiplas com objetivo de suportar várias versões de um serviço para compatibilidade com versões anteriores ou, alternativamente, permitir múltiplos canais para a mesma implementação de componente subjacente.

Existem vantagens associadas com a utilização de um ESB:

- Monitorização;
- Conversão de protocolos;
- Segurança;
- Roteamento e mediação.

Assim como existem vantagens, também existem desvantagens, uma das maiores críticas utilizadas quanto à utilização de um ESB, é que estes podem-se tornar um "*single point of failure*", visto que é o centro de toda a comunicação entre os serviços, caso ele deixe de funcionar a comunicação simplesmente para. "Contudo, os ESB modernos já possuem uma estrutura distribuída e já não correm esse risco" [18] .
Apresenta-se de seguida uma breve descrição de dois ESB amplamente adotados.

2.2.2.1.1 Mule ESB

Mule é um ESB *open source* desenvolvido sobre um plugin do Eclipse e é baseado em Java. Permite a integração de novos sistemas de uma forma muito rápida e intuitiva, podendo ser implantado em qualquer lugar. É muito fácil de configurar e instalar tornando-o muito simples de utilizar, "é altamente escalável, permitindo que se comece pequeno e conecte mais aplicações ao longo do tempo. O ESB gere todas as interações entre aplicativos e componentes de forma transparente, independentemente de existir na mesma máquina virtual ou através da internet, e independentemente do protocolo de transporte subjacente utilizado" [19].
Abaixo são apresentadas algumas características do Mule ESB:

- **Criação e hospedagem de serviços**: exposição e hospedagem de serviços reutilizáveis, utilizando o ESB como um *container* de serviços;
- **Mediação de serviços**: serviços independentes de formatos e protocolos de mensagens, lógica de negócios separada de mensagens e chamadas a serviços independentes da localização;
- **Roteamento de mensagens**: roteamento, filtração, agregação e reordenação de mensagens com base em conteúdo e regras;
- **Transformação de dados**: troca de dados em diferentes formatos (JSON para XML e vice-versa por exemplo).

Este ESB possuí conetores, que representam o código do ESB de uma forma visual, que ajudam na integração de aplicações e na transformação de dados, existindo um leque muito vasto de conetores pré-definidos e permitindo a criação de novos. Estes conetores podem depois serem reutilizados por todas as aplicações.
Algumas características interessantes:

- Mule possuí um vasto leque de conectores pré-definidos, como por exemplo: HL7, *Salesforce, Facebook, Twitter, SAP*, etc. Isto permite que sejam feitas integrações sem ter de ser realizado qualquer tipo de instalação. Tudo está disponível com a instalação do Mule;
- Permitem a reutilização significativa de componentes. Os componentes não requerem nenhum código específico para ser executado e não é necessária nenhuma API programática. A lógica de negócios é mantida completamente separada da lógica de mensagens;
- O Mule pode ser implantado numa variedade de topologias, não apenas ESB. Pode diminuir drasticamente o tempo de mercado e aumentar a produtividade de projetos para fornecer aplicativos seguros e escaláveis que sejam adaptáveis às mudanças;
- Utiliza um conceito de *design-first* para a exposição de serviços web, recorrendo a RAML (*RESTful API Modeling Language*). RAML permite a definição de uma API completa de uma maneira simples e baseado na linguagem YAML.

2.2.2.1.2 WSO2

É um ESB *open source*, definido como um mecanismo de mensagens baseado em padrões que fornece o valor das mensagens sem escrita de código. Fornece recursos de integração de dados e além dos fluxos de integração de curta duração e sem estado, também pode ser usado para gerir processos de negócios de longa data controlando o seu estado. Fornece recursos para intermediários de mensagens que podem ser usados para transmitir mensagens de uma forma confiável, bem como recursos para executar micro serviços para fluxos de integração. "Possuí uma arquitetura que facilita a comunicação entre aplicações e ajuda a conectar sistemas de *software* com diferentes formatos de dados numa arquitetura integrada unificada" [22].
Fornece um vasto leque de conetores pré-definidos e permite a comunicação via diferentes protocolos, assim como a transformação de mensagens de uma forma muito simples e intuitiva. Possui mecanismos de monitorização e análise de acessos, performance e

estatísticas de vários tipos, permitindo identificar *bottlenecks* e a criação de *dashboards* com a informação recolhida.

À semelhança do Mule ESB apresenta características muito idênticas, fornecendo capacidades de mediação de serviço, roteamento de mensagens, transformação de dados, hospedagem de serviços, etc., apresentando-se como um ESB muito capaz de responder às necessidades do mercado.

Uma das grandes vantagens deste sistema é que possui uma quantidade de software que opera dentro do ambiente WSO2, que ao serem integrados com o ESB permitem desenvolver soluções muito robustas, com capacidades de análise de dados, autenticação e autorização, tudo sem necessidade de utilização de *software* externo.

2.2.2.1.3 Comparação entre Mule ESB e WSO2

Na Tabela 3 é possível observar uma comparação entre Mule ESB e o WSO2 ESB, que são as duas ferramentas estudadas para responder à necessidade de integrar dois mundos diferentes.

Tabela 3 - Comparações de tecnologias - ESB

CARACTERÍSTICAS	MULE ESB	WSO2 ESB
IMPLANTAÇÃO	Fácil de implantar	Fácil de implantar
DESCRIÇÃO DE API	Utiliza RAML para definir uma API. Conceito de desenhar primeiro a API e só depois desenvolver	Utilizando o WSO2 API *Management*, também permite a definição de APIs baseada no *swagger*, via interface gráfica
CONECTORES PRÉ-DEFINIDOS	Possui um grande leque de conetores pré-definidos: HL7, *Salesforce*, *Slack*, *Amazon*, *Facebook*, *Magento*, *MongoDB*, SAP, *Twitter*, entre outros. Seno que grande parte destes conetores são suportados pela própria *Mulesoft*	Também possui um grande leque de conetores pré-definidos suportados pela comunidade

COMUNIDADE	Pequena. Em ascensão	Grande e bastante participativa
UTILIZADO POR	*Spotify*, *Netflix*, *Mastercard*, *Unicef*, *Cisco*, *Salesforce*, *Coca-cola*, etc.	*Ebay*, *Motorola*, *verifone*, *subhub*, etc.
TRANSFORMAÇÕES E FILTRAGEM	Conetores avançados de transformações, recorrendo a *dataweave* (linguagem baseada em *scala*) para transformar mensagens, que para além de converter mensagens entre diferentes formatos também permite a realização de operações de filtragem com o *dataweave*. Filtros e transformações realizados no mesmo conetor	Conetores pouco intuitivos e complexos para transformar mensagens. Normalmente são usados conetores *filter* para realizar filtros e de dados e conetores *DataMapper* para realizar mapeamentos entre diferentes formatos de dados. Os conetores *filters* são definidos em XML
ALTA DISPONIBILIDADE	Fornece ferramentas de alta disponibilidade	Fornece ferramentas de alta disponibilidade
LICENCIAMENTO	Solução *open source* é bastante limitada. Modelo de licenciamento bastante limitado.	Solução completamente *open source.*
CONETORES	Bastante intuitivos e com uma aparência moderna	Menos intuitivos e com uma aparência mais antiga

2.2.2.2 API Gateway

Uma API *Gateway* é um serviço que resolve o problema de ter clientes de diferentes naturezas. É um ponto de entrada que permite o acesso a várias API, tal como é demonstrado na Figura 7:

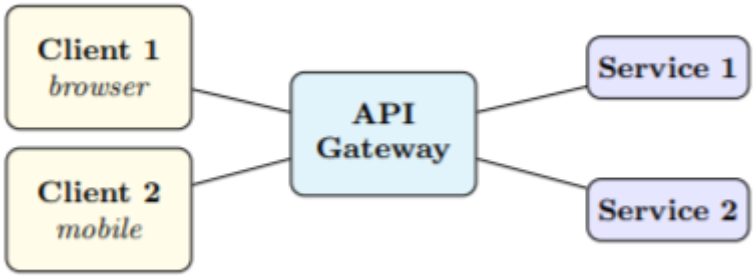

Figura 7 - Arquitetura de uma API GATEWAY

Esta arquitetura providencia funcionalidades tais como:

- Publicação de múltiplas API, em que cada uma é dedicada a um diferente tipo de cliente;
- Atualização de API existentes em tempo real.

Visto que é um ponto de entrada para os diferentes serviços, é natural que esta possua por exemplo: *Service Discovery*, balanceador de carga, monitorização e segurança. É ideal que visto a sua posição ser tão importante no sistema, que seja adotado o padrão de *Circuit Breaker*.

O *Service Discovery*, tal como o nome indica, permite realizar a descoberta de serviços. Uma aplicação que pretenda consumir um determinado serviço, não precisa de ser diretamente onde se encontra a instância do *gateway*, nem o *gateway* precisa de saber a localização de todos os serviços existentes. Para isso existe o *Service Discovery*, que permite que os diferentes serviços sejam registados e encontrados. Caso um serviço seja migrado para outro servidor apenas o registo no *Service Discovery* terá de ser alterado para a localização correta.

O *Circuit Breaker* é um padrão que tem como objetivo evitar que uma única falha num componente entre em cascata além dos seus limites, e assim fazer com que todo o sistema falhe. A ideia é falhar o mais rápido possível: quando um determinado serviço deixa de responder, os seus invocadores devem parar de esperar por uma resposta, assumindo assim o pior e começar a lidar com o facto de que o serviço com falha pode ficar indisponível. Com isto, o *Circuit Breaker* contribuí para a estabilidade e resiliência dos clientes e serviços: limitando assim a tentativa de os clientes desperdiçarem recursos na tentativa de aceder aos serviços que não estão a responder, fazendo com que estes mesmos serviços que se encontrem sobrecarregados tenham a possibilidade de recuperar. Terminando algumas das tarefas que estão a processar num determinado momento [23].

À semelhança dos ESB, a API Gateway possuí vantagens e desvantagens. Quanto às vantagens podemos enumerar as seguintes:

- Consegue resolver grande parte dos problemas das empresas;
- Permite a separação de preocupações:
  - Os micro serviços focam-se nas funcionalidades de negócio;
  - A API Gateway fornece proteção/camada de recurso comum.

Existem também desvantagens associadas à utilização de uma API Gateway:

- Possibilidade de SPOF (*Single Point of Failure*);
- Necessidade de gerir regras de roteamento ou API;
- Custo de desenvolver uma API Gateway;
- Custo adicional no uso de *Hardware/Network.*

2.2.3 Message Oriented Middleware

"As tecnologias *Message Oriented Middleware* (MOM) são usadas para interligar sistemas de software independentes, para criar um único sistema" [22]. Este facto permite que "os sistemas de software em rede sejam desenvolvidos usando diferentes tecnologias e plataformas que são executadas em diferentes configurações de hardware e / ou software." [13]

Um sistema de filas de mensagens permite que aplicações troquem dados e informações. A sua maior vantagem é a capacidade de escalonar para responder a cargas crescentes e continuar a executar mesmo quando ocorre falha.

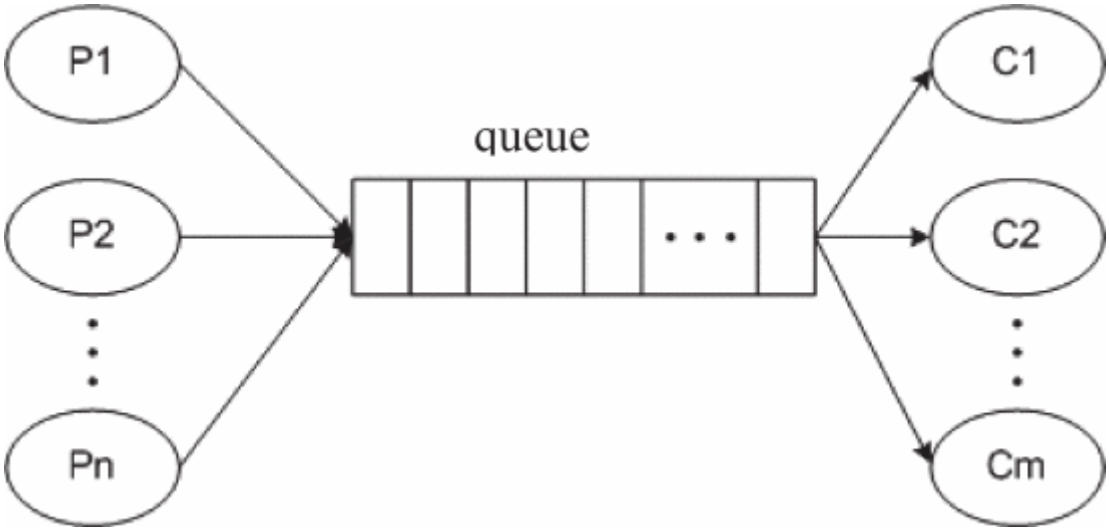

Figura 8 - Modelo genérico de message queue [22]

No modelo de sistema de filas de mensagens, há vários objetos específicos, que serão explicados nos pontos seguintes:

- *Connection Factory* – são responsáveis por criar a ligação ao emissor de mensagens, permitindo assim a configuração da conexão;
- Um objeto de conexão é usado para criar uma ligação entre os clientes e o produtor, ou seja, um ou vários canais;
- O emissor é um objeto criado por uma sessão (canal) para enviar dados para um determinado destino;
- O consumidor é a aplicação que recebe as mensagens;
- A *queue* é um *buffer* no servidor que armazena as mensagens, esta pode ser configurada para ser persistente ou não;
- A mensagem é o formato de dados base que é flexível para acomodar plataformas heterogêneas [22].

Para este projeto a desenvolver, que usa filas de mensagens para entregar as mensagens de origem até ao destino, foi escolhido o RabbitMQ como aplicação que será responsável por essa função, uma vez que esta tecnologia já está adotada na empresa.

2.2.3.1 RabbitMQ

"O RabbitMQ é um *message broker* que implementa o protocolo padrão Advanced Message Queuing Protocol (AMQP)." [22], que possuí várias funcionalidades importantes que contribuíram para o seu uso generalizado.

Assim sendo, vários servidores RabbitMQ de uma rede local podem ser agrupados, formando um *broker* lógico, este facto permite a implementação de recursos como balanceamento de carga e tolerância a falhas.

Por outro lado, o RabbitMQ tem como vantagem o uso do protocolo AMQP, que por sua vez, tem como funcionalidade importante o facto de aceitar conexões entre diferentes plataformas. [22], [23]

2.2.3.2 Advanced Message Queuing Protocol

"O AMQP é um protocolo avançado de *message queue*, que é uma camada aplicacional para serviços de *middleware* orientados a mensagens" [23].

O AMQP permite que as aplicações enviem e recebam mensagens, funcionando desta forma como mensagens instantâneas ou email. Em conseguinte, este torna-se bastante diferente de outras soluções disponíveis, pois permite especificar quais as mensagens que podem ser recebidas e como compensar segurança, confiabilidade e desempenho.

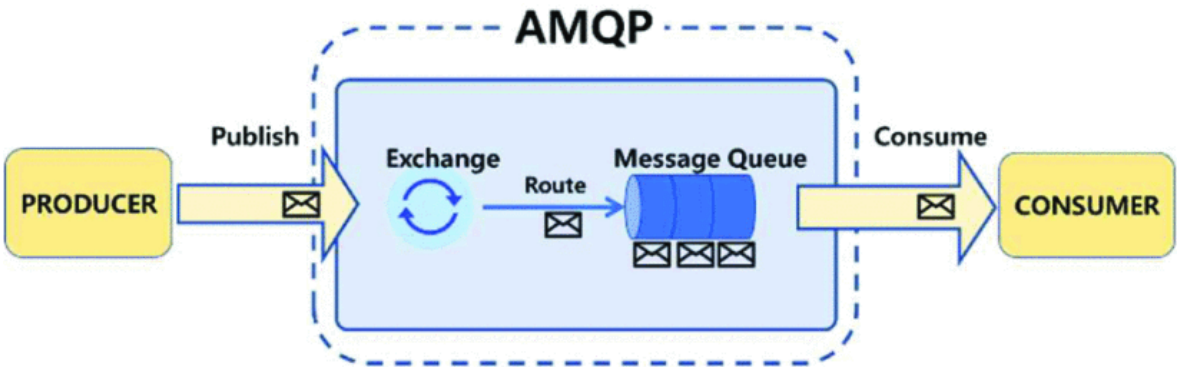

Figura 9 - Arquitetura AMQP

As mensagens enviadas pela aplicação de publicação são primeiro recebidas pelo *Exchange* definido e são roteadas para a fila de mensagens de acordo com determinadas regras. A fila de mensagens armazena as mensagens até que elas sejam completamente processadas pelo consumidor.

2.3 Síntese

Todos os conhecimentos adquiridos através do estudo de cada subsecção deste capítulo vão ser tidos em conta durante o desenvolvimento do projeto.

É possível perceber a importância do desenvolvimento deste projeto através da leitura deste capítulo. Com o aumento da utilização de dispositivos móveis na área da saúde, verifica-se

que as aplicações passarão a utilizar o FHIR como protocolo para as trocas de informação. Visto que este é o único inerentemente adequado para as aplicações móveis.

3 Análise de valor

Este capítulo apresenta uma análise de valor do projeto, com o objetivo de provar o seu valor de mercado, para a organização e para os clientes.

Esta análise fará uso do modelo NCD (*New Concept Development*) para perceber quais são as necessidades, as ideias geradas e selecionadas a fim de chegar ao conceito de negócio final. É ainda utilizado o Modelo CANVAS para facultar a visão geral do negócio.

Atualmente, a *Glintt* possui uma carteira de clientes espalhada pelo mundo, tal como já foi mencionado anteriormente. Contudo pretende alargá-la, e vê neste projeto a chave para esse objetivo.

3.1 Modelo New Concept Development

"O modelo NCD é um modelo iterativo que permite esclarecer a visão e cultura de uma organização. Com este modelo é possível perceber quais são os fatores de influência do ambiente externo, que consiste nas capacidades organizacionais, estratégia de negócio e mundo exterior (canais de distribuição, clientes e concorrentes)"[24].

Este modelo permite relação entre 5 diferentes processos:

- Identificação da oportunidade;
- Análise de oportunidade;
- Geração de ideias;
- Seleção de ideias;
- Definição de conceito.

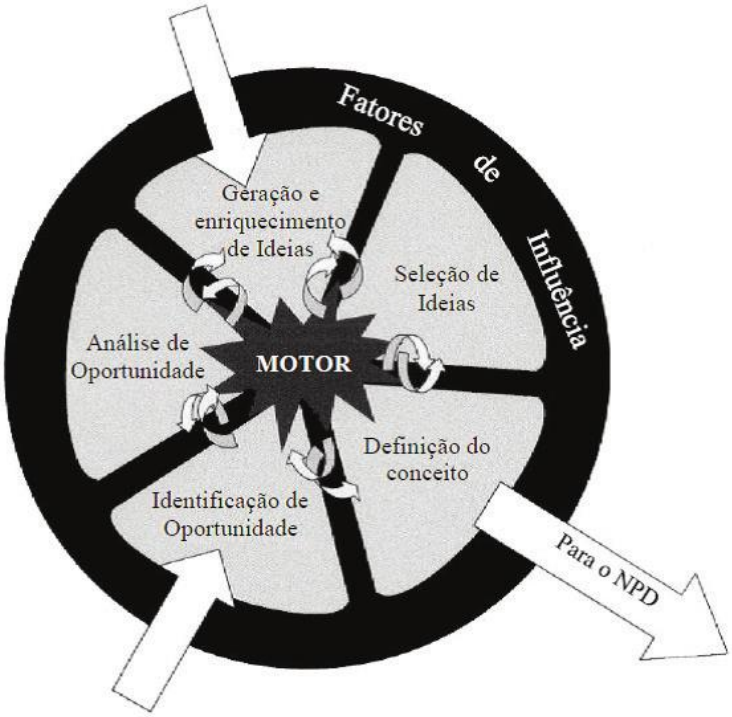

Figura 10 - Modelo NCD [24]

A Figura 10 ilustra o funcionamento do modelo NCD. Este possui uma forma circular que permite verificar que há interação entre os 5 elementos. As setas que apontam em direção ao modelo representam os pontos de entrada, que indica que um projeto pode começar através da geração de ideias ou identificação de oportunidade. A seta de saída do modelo, representa como todos os conceitos do modelo dão origem ao modelo de desenvolvimento de produto (NPD) [25].

3.2 Identificação de oportunidade

Nesta secção são identificadas e descritas as oportunidades que irão permitir o crescimento de mercado e eficiência à empresa.

Estas oportunidades podem levar a uma nova direção de negócio ou atualizar produtos já existentes.

A oportunidade da solução documentada nesta dissertação surgiu do problema identificado na *Glintt* relacionado com a comunicação entre os dois protocolos, e que lhe permitirá expandir a sua carteira de clientes.

Tal como já foi mencionado anteriormente, os protocolos de troca de informação ao nível da saúde. Surge assim uma oportunidade de implementar uma solução que permita não só a

comunicação entre o protocolo HL7 e o FHIR, mas sim com qualquer outro protocolo com FHIR.

3.3 Análise de oportunidade

Nesta secção, é reunida informação adicional capaz de traduzir as oportunidades identificadas em oportunidades específicas de negócio e tecnologia para a organização, e como estas estão relacionadas com as estratégias da organização, percebendo se valerá seguir as mesmas.

A *Glintt* pretende comercializar uma solução que já possuí, que se trata de uma API FHIR. Tal como já foi mencionado ao longo deste documento, esta tecnologia está a revolucionar a forma de como é feita a troca de informação na área da saúde. Contudo há ainda dificuldades em comercializar essa solução, pois ainda são poucas as unidades hospitalares que possuem aplicações capazes de comunicar com em FHIR, apesar da tendência em aumentarem.

Aos dias de hoje ainda não existe nenhum software capaz de ligar um protocolo standard HL7 e FHIR. A *Glintt* pretende assim criar uma camada que seja capaz de permitir essa ligação, aumentando assim a interoperabilidade das aplicações numa unidade de saúde.

3.4 Geração de ideia

Nesta seção descreve-se como o nascimento e desenvolvimento da oportunidade até atingir o patamar da ideia concreta. Não se pretende que seja perfeita no momento da criação, passa por várias iterações, onde é melhorada através de discussões e análises à volta da mesma. Na geração são utilizadas sessões de *brainstorming*.

Durante todo o processo de desenvolvimento, é expectado que existam sessões de *brainstorming* constantes, para que a ideia possa estar em constante processo de aperfeiçoamento, onde todos os intervenientes no processo de desenvolvimento desta solução contribuam com as suas opiniões.

Neste processo foram geradas três ideias quanto à arquitetura a ser utilizada no desenvolvimento da solução:

- Ideia 1 - Utilizando um ESB;
- Ideia 2 - Utilizando uma API *Gateway*;
- Ideia 3 - Utilizando um ESB e uma API *Gateway*.

Todos os pontos mencionados, possuem o mesmo prepósito que é receber uma mensagem de um certo tipo e convertê-la num FHIR *resource*. A diferença está na forma como é implantada a solução.
Cada tipo de arquitetura mencionada anteriormente irá estar detalhada na secção 5.

3.5 Seleção de ideia

Nesta fase, considerando que várias ideias surgiram, torna-se importante decidir quais são as que vão ser seguidas para trazer maior valor de negócio. A seleção pode incluir uma escolha individual ou mais formalizada, utilizando um método de portefólio.
Para implementar a solução apresentada para o problema em questão, existem diferentes tipos de arquiteturas e tecnologias que podem ser usadas. É importante analisar os diferentes tipos de arquiteturas e tecnologias e decidir qual é a que se adequa melhor à organização.
Com o objetivo de decidir qual a melhor ideia a selecionar, irá ser utilizado o método *Analytic Hierarchy Process* (AHP).

3.6 Analytic Hierarchy Process (AHP)

O método *Analytic Hierarchy Process* (AHP), foi criado pelo professor Thomas L. Saaty em 1980. O principal objetivo deste método é dividir o problema de decisão em níveis hierárquicos, tornando mais fácil a sua compreensão e avaliação. Neste método é permitido o uso de critérios qualitativos bem como quantitativos no processo de avaliação.
De acordo com as ideias que foram geradas na secção 3.4, foi construída a árvore hierárquica de decisão.

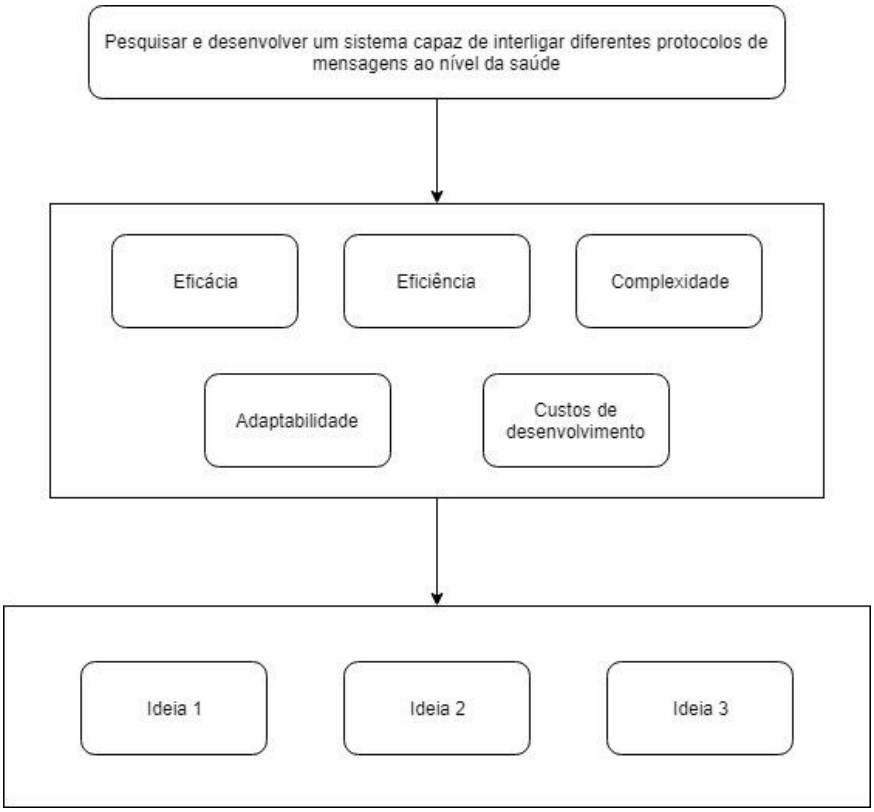

Figura 11 - Árvore hierárquica de decisão

Como observado na Figura 11, a primeira camada representa o objetivo principal (pesquisar e desenvolver um sistema capaz de interligar diferentes protocolos de mensagens ao nível da saúde).

Os critérios envolvidos para avaliar qual a melhor ideia para chegar à solução são:

- Eficiência – de que forma a solução cumpre os requisitos mencionados;
- Eficácia – se a solução cumpre os requisitos;
- Adaptabilidade – se a solução permite adaptar novos protocolos de mensagem, para estes serem igualmente convertidos para FHIR;
- Complexidade – se a solução é fácil ou complexa de desenvolver e atingir o objetivo;
- Custos de desenvolvimento – relação entre o tempo/recursos necessários para a implementação de determinada solução.

Através dos critérios anteriormente mencionados, é possível avaliar qual é a melhor ideia para atingir o objetivo principal. É igualmente necessário definir uma escala de valores para a comparação, que não deve exceder um total de nove fatores.

Utilizando a escala fundamental que Saaty definiu, é possível observar os níveis de importância de comparações através da Tabela 4:

Tabela 4 - Escala fundamental - Níveis de importância de comparações

Nível de importância	Definição	Explicação
1	Igual importância	As duas atividades contribuem igualmente para o objetivo.
3	Fraca importância	A experiência e o julgamento favorecem levemente uma atividade em relação à outra
5	Forte importância	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra
7	Muito forte importância	Uma atividade é muito fortemente favorecida em relação a outra
9	Importância absoluta	A evidência favorece uma atividade em relação a outra com o mais alto grau de certeza
2,4,6,8	Valores intermediários	Quando se procura uma condição de compromisso entre duas definições

A Tabela 5 inclui a atribuição de pesos para cada critério, segundo a escala mencionada na Tabela 4.

Tabela 5 - Tabela de avaliação AHP

Critérios de avaliação	Eficiência	Eficácia	Comple-xidade	Adaptabili-dade	Custo de desenvolvimento
Eficiência	1	7	7	7	3
Eficácia	1/7	1	2	5	2
Complexidade	1/7	1/2	1	7	7
Adaptabilidade	1/7	1/5	1/7	1	3
Custo de desenvolvimeto	1/3	1/2	1/7	1/3	1
Total	37/21	46/5	72/7	61/3	16

Depois de ser obtida a matriz com os valores correspondentes, é possível normalizar os mesmos com o objetivo de obter as prioridades gerais dos critérios. A normalização pode ser aplicada efetuando a soma dos valores de cada coluna, e depois dividindo cada célula pelo valor total de cada coluna. Por fim é possível obter as prioridades calculando a média de cada linha.

Tabela 6 - Matriz normalizada AHP

Critérios de avaliação	Eficiência	Eficácia	Complexi-dade	Adaptabilidade	Custo de desenvolvi-mento	Média
Eficiência	0,568	0,761	0,680	0,344	0,188	0,508
Eficácia	0,081	0,109	0,194	0,246	0,125	0,151
Complexidade	0,081	0,054	0,097	0,344	0,438	0,203
Adaptabilidade	0,081	0,022	0,014	0,049	0,188	0,078
Custo de desenvolvimen-to	0,189	0,054	0,014	0,016	0,062	0,067
Total	1	1	1	1	1	1

Utilizando os valores anteriormente calculados, é possível decidir quais os critérios são os mais prioritários:

Tabela 7 - Prioridade de critérios

Posição da prioridade	Critério	Percentagem de Prioridade
1	Eficiência	50,8 %
2	Complexidade	20,3 %
3	Eficácia	15,1 %
4	Adaptabilidade	7,8 %
5	Custo de desenvolvimento	6,7 %

É possível concluir que a eficiência, complexidade e eficácia são os principais fatores a ter em consideração na escolha da melhor solução. A adaptabilidade e o custo de desenvolvimento são relevantes, mas não são considerados critérios de elevada importância.

Analisando os resultados obtidos, é possível assumir que a ideia 1 é considerada a melhor solução. Tem maior eficiência, menor complexidade algorítmica e é mais eficaz.

3.7 Definição do conceito

Por fim, é desenvolvido um caso de negócio baseado nos seguintes fatores: potencial de mercado, necessidade dos clientes, investimentos necessários, avaliações da concorrência, tecnologia necessária e risco do projeto.

Após a seleção da arquitetura e tecnologias corretas a seguir, um caso de negócio é definido na *Glintt*, através de reuniões com gestores e *stakeholders*, a fim de definir os riscos da solução, necessidades do cliente e do mercado, investimentos e esforço necessário.

O objetivo principal desse projeto é fornecer uma solução capaz de fazer a ponte entre diferentes protocolos, que nativamente não incompatíveis. Como já foi referido, irá ser

desenvolvida uma solução escalável em que seja possível introduzir diferentes protocolos de *input*, com a finalidade de os transformar em FHIR.

3.8 Proposta de valor

A proposta de valor é essencial para alertar novos clientes quanto ao valor dos produtos e serviços da organização. Como Alexander Osterwalder escreveu, é uma visão geral de uma empresa dos serviços e produtos que são valiosos para o cliente. (Osterwalder, 2004). A proposta de valor deve clarificar os produtos/serviços que a empresa fornece, quais os clientes alvo, o inerente valor fornecido e único.

O produto documentado nesta dissertação tem como objetivo principal tornar a *Glintt* capaz de alargar a sua área de intervenção, quanto à escala geográfica.

A *Glintt* pretende vender este produto como um serviço de integração entre sistemas que os clientes já possuam, criando assim uma ponte entre eles. Os dados irão estar armazenados apenas em FHIR, onde será indiferente qual o tipo de protocolo de entrada. Este produto irá enriquecer a organização, pois a interoperabilidade entre aplicações internas ou externas, vai aumentar.

3.9 Modelo Canvas

O modelo *Canvas* ou *Business Model Canvas*, é uma ferramenta de planeamento estratégico que permite desenvolver e desenhar novos modelos de negócio ou melhorar modelos existentes. Permite ter uma melhor perceção de todos os intervenientes do processo.

Tabela 8 - Modelo CANVAS representativo do projeto

Sistema de integração entre diferentes protocolos				
Parcerias principais (Key partners) Clientes utilizadores de aplicações de saúde	Atividades-chave (Key activities) Conhecer os diferentes tipos de protocolos e saber como converter um tipo noutro. Levantar as necessidades para	Proposta de valor (Key propositions) Oferecer a possibilidade de integrar aplicações com uso de diferentes tipos de	Relacionamento com clientes (Customer relationships) Suporte	Segmento de clientes (Customer segments) Colaboradores da Glintt-HS Clientes consumidores

	a conversão e implementar.	protocolos num único sistema.		
	Recursos principais (Key resources) Desenvolvedores Conversor		Canais (Channels) Sistema de integração	
Estrutura de custos (Cost structure) Desenvolvedores/programadores;		Fontes de receita (Revenue streams) Venda do serviço de integração; Venda de um repositório de dados;		

Como se pode observar através do modelo *Canvas* (Tabela 8), ao usar técnicas de conversão e desenvolvedores para implementar a conversão entre diferentes tipos de protocolo é possível que a proposta de valor seja atingida. O conhecimento adquirido quanto à conversão entre diferentes tipos de protocolo é a atividade-chave do projeto, bem como o levantamento de necessidades para implementar essa conversão.

As fontes de receita expectáveis do projeto advêm do aumento do número de clientes a utilizarem serviços desenvolvidos pela *Glintt* para troca de informação. Ao estabelecer novos contratos com clientes (aplicações externas), as formas de receita também aumentam.

4 Análise

Esta secção tem como objetivo sistematizar e analisar os tipos de mensagem selecionados para uso de prova de conceito, e que se especula como sendo representativos da grande maioria, se não na totalidade das mensagens a transformar.

4.1 Análise de Mensagens HL7

O HL7 dispõe de um conjunto diversificado de tipos de mensagens, do qual foram selecionados para análise os seguintes:

- ADT-A01 – Admissão de paciente;
- ADT-A02 – Transferência de paciente;
- ADT-A03 – Descarga de paciente;
- ADT-A04 – Registo de paciente;
- ADT-A08 – Atualização de informação do paciente;
- ADT-A34 – Combinação da informação de paciente.

A análise consiste em conhecer os segmentos existentes em cada tipo de mensagem, assim como o significado de cada um. A tabela que se segue descreve os segmentos que compõem cada mensagem mencionada anteriormente.

Tabela 9 - Segmentos existentes nos tipos de mensagem a transformar

Segmentos	Tipos de Evento/Mensagem
MSH (Message Header) – define a origem, o objetivo e o destino da mensagem.	Todos
EVN (*Event*) – é usado para comunicar qual o evento gerado, assim como o emissor e recetor.	Todos
PID (*Patient Identification*) – fornece informação do paciente. Devido à natureza das informações encontradas neste segmento, é improvável que mude com frequência.	Todos
PV1 (*Patient Visit*) – contém a informação da visita do paciente, como por exemplo: localização do paciente, admissão do médico, o número de visita, centro de atendimento, entre outros.	ADT-A01, ADT-A02, ADT-A03, ADT-A04, ADT-A08
PV2 (*Patient Visit* – informação adicional) - informação adicional do PV1	ADT-A01
PR1 (*Procedures*) - contém informações relativas a vários tipos de procedimentos que podem ser executados num paciente, por exemplo: raio-x ressonância, entre outros.	ADT-A01, ADT-A04, ADT-A08
IN1 (*Insurance*) - contém as informações de cobertura da apólice de seguro necessárias para produzir as contas de pacientes e seguros adequadamente aprovadas.	ADT-A01, ADT-A04, ADT-A08
MRG (*Merge Patient Information)* – combinação da informação de paciente.	ADT-A34

4.2 Análise de Mensagens FHIR

No FHIR também existe a troca de informação por *event-driven* à semelhança do que acontece no HL7. Contudo, o FHIR foi utilizado como *Restful* API, que é o seu prepósito. Existe assim os seguintes métodos HTTP utilizados:

- POST – utilizado para a criação de *resources*;
- GET – utilizado para obter os *resources*;
- PUT – utilizado para atualizar informação dos *resources*;
- DELETE – utilizado para apagar um determinado *resource*.

A análise consiste em perceber quais os *resources* necessários para dar resposta às transformações necessárias. Na tabela seguinte estão representados os *resources* estudados, assim como o nível de maturação dos mesmos.

Tabela 10 - Segmentos existentes nos tipos de mensagem a transformar

Resources	Nível de maturidade
MessageHeader – define a origem, o objetivo e o destino da mensagem.	4
Patient - fornece informação do paciente.	5
Encounter - contém a informação da visita do paciente, como por exemplo: localização do paciente, admissão do médico, o número de visita, centro de atendimento, entre outros.	2
InsurancePlan - descreve uma oferta de seguro de saúde composta por uma lista de benefícios cobertos, custos associados a esses benefícios (por exemplo, o plano	0
Procedure - fornece informações resumidas sobre a ocorrência do procedimento e não se destina a fornecer instantâneos em tempo real de um procedimento à medida que ele se desenrola, embora para procedimentos de longa duração, como psicoterapia, possa representar informações resumidas sobre o progresso geral	3

O nível de maturidade presente na Tabela 10, representa o grau de maturidade dos *resources* apresentados, ou seja, quanto maior for o nível, maior será a maturidade e menor será a probabilidade de mudança.

4.3 Mapeamento HL7-FHIR

Nas secções anteriores, foram apresentados os tipos de mensagens estudados e os segmentos que compõem as mesmas, assim como os *resources* FHIR que irão ser utilizados na transformação.

Na tabela seguinte é possível observar quais os segmentos existentes em cada tipo de mensagem mencionado anteriormente e quais os *resource* FHIR a que estes dão origem.

Tabela 11 - Resources FHIR e quais os segmentos HL7 usados para a sua obtenção

Resource	Segmentos usados para transformação
MessageHeader	MSH, EVN
Patient	PID, PV1, PV2
Encounter	PID, PV1, PV2
InsurancePlan	IN1?
Procedure	PR1?

Tal como já foi mencionado previamente, não existe nenhuma especificação ou mesmo recomendação para a transformação de um protocolo para o outro. Desta forma, existem ainda muitos segmentos presentes no protocolo HL7 que não têm qualquer correspondência do lado do FHIR, pelo que será necessário criar extensões para guardar essa informação, evitando assim a perda de dados.

Cada segmento presente numa mensagem HL7 possuí vários subsegmentos, sendo que cada um desses subsegmentos pode ser utilizado para a criação de um *resource* FHIR.
É no *resource Message Header* que está presente a informação referente ao tipo de evento gerado, ao emissor da mensagem e ao recetor da mesma.

Tabela 12 - Resource Message Header e mapping do HL7 [26]

Message Header	Mínimo/Máximo	Segmentos HL7
event[x]	1..1	MSH-9.2
destination	0..*	
name	0..1	MSH-5
target	0..1	
endpoint	1..1	MSH-25 ou MSH-6
receiver	0..1	PRT-5:PRT-4='WAYR' / PRT-8:PRT-4='WAYR'
sender	0..1	PRT-5:PRT-4='WAYR' / PRT-8:PRT-4='WAYR'
enterer	0..1	EVN-5 / ORC-10 / PRT-5:PRT-4='EP' / ROL where ROL.3 is EP or ORC.10

author	0..1	ORC-19 / PRT-5:PRT-4='AUT' / ROL where ROL.3 is IP or ORC.12
source	1..1	
name	0..1	MSH-3
software	0..1	SFT-3 (+SFT-1)
version	0..1	SFT-2
contact	0..1	(MSH-22?)
endpoint	1..1	MSH-24
responsible	0..1	ORC-12 / PRT-5:PRT-4='OP' / PRT-8:PRT-4='OP' / ROL where ROL.3 is RO or ORC.11
reason	0..1	EVN.4 / ORC.16 / OBR-31-reason for study / BPO-13-BP indication for use / RXO-20-indication / RXE-27-give indication / RXD-21-indication / RXG-22-indication / RXA-19-indication
response	0..1	MSA
identifier	1..1	MSA-2
code	1..1	MSA-1
details	0..1	ERR
focus	0..*	
definition	0..1	

Na Tabela 12 é possível observar os mapeamentos definidos pela comunidade HL7 FHIR entre os dois protocolos para o *MessageHeader*. Contudo, verifica-se ainda que existem segmentos no FHIR que não têm correspondência no lado do HL7.

Na Tabela 13 estão representados os mapeamentos definidos pela comunidade HL7 FHIR entre os dois protocolos para o *resource Patient*. No entanto, é possível verificar que para este *resource* também não existe mapeamento total dos segmentos entre um protocolo e o outro, à semelhança do que acontece no *MessageHeader*.

Tabela 13 - Resource Patient e mapping do HL7 [27]

Patient	Mínimo/Máximo	Segmentos HL7
Identifier	0..*	PID-3
Active	0..1	
name	0..*	PID-5, PID-9
telecom	0..*	PID-13, PID-14, PID-40
gender	1..1	PID-8
birthDate	0..1	PID-7
deceased[x]	0..1	PID-30 (bool), PID-29 (datetime)
address	0..*	PID-11
maritalStatus	0..1	PID-16
multipleBirth[x]	1..1	PID-24 (bool), PID-25 (integer)
photo	0..*	OBX-5 – *needs a profile*
contact	0..*	SFT-3 (+SFT-1)
relationship	0..*	NK1-7, NK1-3
name	0..1	NK1-2
telecom	0..*	NK1-5, NK1-6, NK1-40
address	0..1	NK1-4
gender	0..1	NK1-15
organization	0..1	NK1-13, NK1-30, NK1-31, NK1-32, NK1-41
period	0..1	
communication	0..*	
language	1..1	PID-15, LAN-2
preferred	0..1	PID-15
generalPractioner	0..*	PD1-4
link	0..*	
other	1..1	PID-3, MRG-1
type	1..1	

No *resource Patient* não existe nenhum campo para guardar a informação quanto à nacionalidade do paciente, ao contrário do que existe no segmento PID existente no HL7. Neste caso específico a comunidade HL7 FHIR disponibiliza extensões já predefinidas e estruturadas por ela.

```
<!-- nationality -->

<extension xmlns="http://hl7.org/fhir"
     url="http://hl7.org/fhir/StructureDefinition/patient-nationality" >
 <-- extension sliced by value:url  in the specified orderOpen-->
 <extension url="code"> 0..1 Extension  <!-- 0..1 Nationality Code -->
  <valueCodeableConcept><!-- 0..1 CodeableConcept
      Value of extension --></valueCodeableConcept>
 </extension>
 <extension url="period"> 0..1 Extension  <!-- 0..1 Nationality Period -->
  <valuePeriod><!-- 0..1 Period
      Value of extension --></valuePeriod>
 </extension>
</extension>
```

Figura 12 - Extensão de nacionalidade

Na Figura 12 está presente um exemplo de uma extensão para capturar informação da nacionalidade do paciente.

Em suma, é possível perceber que existem muitos segmentos que não têm qualquer correspondência em FHIR, sendo que nesses casos é necessário utilizar extensões. No entanto não existem extensões criadas pela comunidade para todos os casos, sendo necessário criar as próprias extensões.

5 Design

Este capítulo tem como objetivo apresentar e debater alternativas de design arquitetural do software a ser implementado.

5.1 Design Concetual

Nesta secção serão apresentadas as diferentes fases necessárias para a implementação do mapeamento entre os diferentes protocolos.

Existem dois momentos distintos quanto à transformação entre os dois protocolos:

- Mapeamento entre HL7 e FHIR – nesta fase é quando um perito do domínio executa todo o mapeamento necessário entre um protocolo e o outro. Sendo que esta fase irá sempre desenvolvida por um perito, pois ele tem o conhecimento do domínio para executar o mapeamento necessário.
- Execução do mapeamento – é nesta fase que é utilizado o mapeamento criado para a execução da transformação entre a mensagem HL7 e o(s) *resource* FHIR.

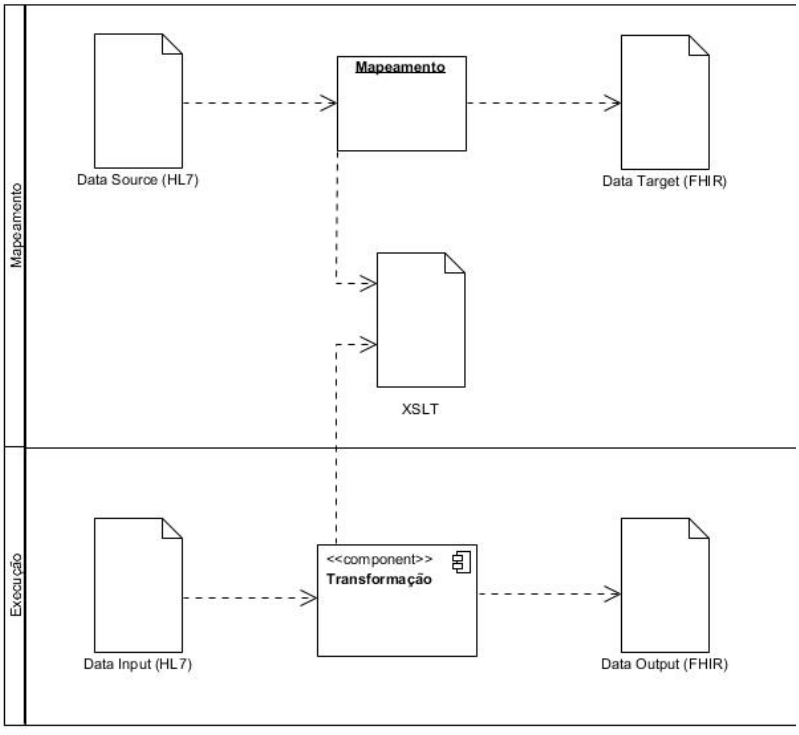

Figura 13 - Design Concetual do mapeamento e transformação

Na Figura 13 é possível observar os dois momentos distintos mencionados anteriormente. Na fase de mapeamento verificamos que tal como mencionado anteriormente, o perito de domínio possuí um *data source* (HL7) e um *data target* (FHIR). O *data source* (HL7)

corresponde a um ou vários segmentos de uma determinada mensagem HL7, enquanto o *data target* (FHIR) corresponde a um ou vários *resources* que esses segmentos poderão dar origem. É nesta fase que o perito de domínio cria o mapeamento entre o *data source* e o *data target*, dando origem a um ficheiro XSLT.

Na fase de execução, esta recebe um *data input* (HL7), ou seja, uma mensagem HL7 e depois usando o mapeamento criado na fase de mapeamento, é executada a transformação dessa mensagem. Dando origem a um *data output* (FHIR), este pode corresponder a um ou vários *resources* FHIR.

Podemos concluir que estas duas fases são imprescindíveis para o projeto a desenvolver, sendo que é na primeira que é feito e definido todo o mapeamento, por um perito. E na segunda é onde é executada a transformação desse mesmo mapeamento.

5.2 Arquitetura de software

"A arquitetura de software é uma estrutura geral de um sistema de software, que consiste num grupo de componentes que interagem e as conexões entre os mesmos, além das restrições que se aplicam a essas conexões." [28]

A arquitetura de software (AS) é extremamente importante para o entendimento, análise, verificação e evolução de um sistema de software.

Após a análise do protótipo a desenvolver, foi realizado o estudo de alternativas da AS e o seu processo de desenvolvimento.

A descrição do *design* arquitetural fará uso de dois níveis de abstração, (i) de sistema e (ii) de componente (como sugerido pelo modelo C4 [29, p. 4]) e das vistas (i) lógica, (ii) de processo, (iii) de implantação e (iv) de implementação (como sugerido pelo modelo 4+1 [30]).

5.2.1 Design arquitetural de nível 1 (Sistema)

Nesta secção irá ser apresentado o modelo 4+1 ao nível de sistema. Irão ser dados a conhecer os componentes envolvidos no projeto, assim como a interação entre os mesmos.

5.2.1.1 Vista lógica

A vista lógica descreve o sistema num conjunto de abstrações principais, retiradas principalmente do domínio do problema. Essas abstrações podem ser objetos, classes de objetos, entre outros. Para além de ajudar na análise funcional, a decomposição identifica mecanismos e elementos de design comuns em todo o sistema [30], [31].

Nesta vista são normalmente utilizados diagramas de componentes para uma melhor compreensão dos componentes envolvidos no projeto.

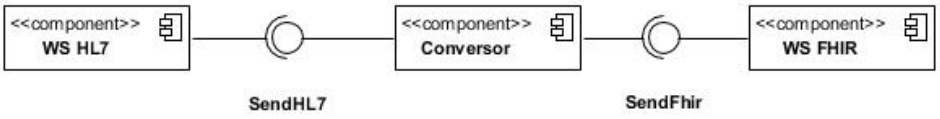

Figura 14 - Diagrama de componentes dos Sistemas envolvidos

Os componentes representados no diagrama da Figura 14 têm as seguintes responsabilidades:

- **WS HL7 –** é responsável por processar os eventos gerados por uma ação de um utilizador, criando uma mensagem HL7 e enviar a mesma para o Conversor;
- **Conversor –** é responsável por converter a mensagem recebida em *resources* FHIR e enviar os mesmos para o WS FHIR;
- **WS FHIR –** é responsável por receber os *resources* FHIR e guardá-los.

As aplicações representadas pelos componentes WS HL7 e WS FHIR no diagrama anterior já se encontram desenvolvidos, sendo que para este projeto irá ser desenvolvido o componente Conversor.

5.2.1.2 Vista de Processo

Nesta vista são utilizados diagrama de sequência para a compreensão da interação entre todos os processos envolvidos.

Na Figura 15 está representada a interação que existe entre os diferentes componentes mencionados na vista lógica. Como exemplo foi utilizado o fluxo de admissão de um paciente, que neste caso é uma mensagem do tipo ADT-A01.

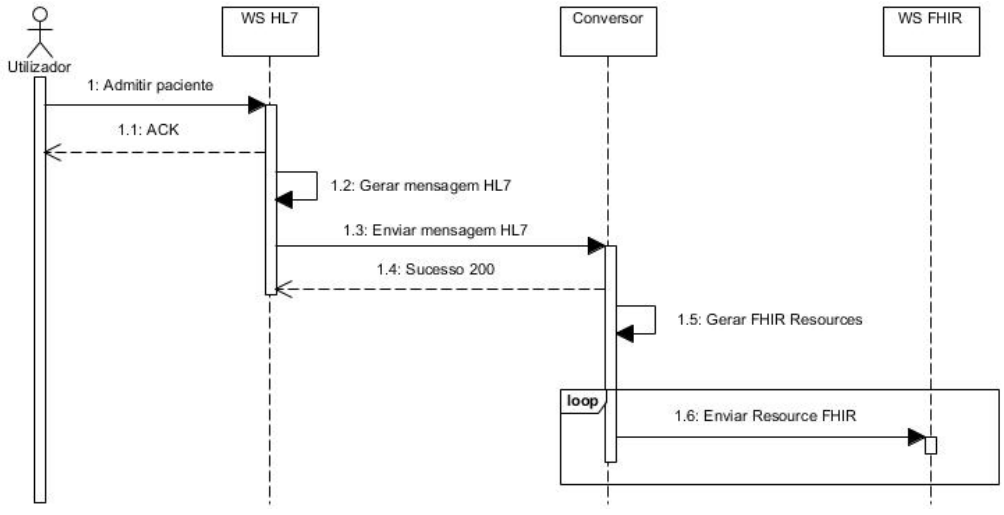

Figura 15 - Diagrama de sequência de alto nível

O fluxo começa por uma ação do utilizador, que admitindo um paciente irá gerar uma mensagem HL7 (ADT-A01) no componente WS HL7. De seguida, esta mensagem é enviada para o Conversor, que após receção da mesma envia uma resposta de aceitação (ACK) para o WS HL7. É neste componente que irá ser transformada a mensagem HL7, num ou vários *resource* FHIR e enviado(s) de seguida para o WS FHIR.

5.2.1.3 Vista de implementação

A vista de implementação tem como principal foco a organização dos módulos de software reais no contexto de desenvolvimento de software [30]. O software é dividido em diferentes pacotes, sendo que cada um destes pode corresponder a bibliotecas de programas ou subsistemas.

É nesta vista que é suportada a alocação de requisitos e trabalho para as equipas, a avaliação de custos, planeamento, monitorização do progresso do projeto e raciocínio sobre reutilização de software, portabilidade e segurança. Para dar suporte a esta vista, são utilizados os diagramas de componentes [30]

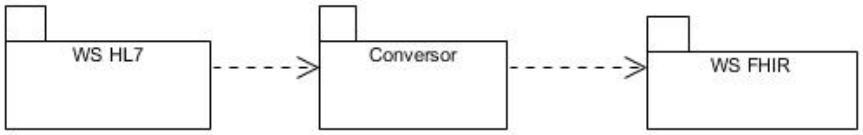

Figura 16 - Diagrama de implementação

A Figura 16 descreve a estrutura de implementação do projeto de software, alinhado com o que foi proposto nas vistas anteriores. É possível verificar quais os *packages* existentes, assim como as dependências entre os mesmos.

Existem assim 3 packages dentro do Conversor, sendo eles:

- **WS HL7 –** Este já se encontra desenvolvido e tem como responsabilidade tratar e enviar as mensagens HL7 para o ESB;
- **WS FHIR –** Ao momento tempo do desenvolvimento do projeto, o WS FHIR não se encontra desenvolvido, sendo que em alternativa irá ser utilizada uma API pública disponível para validação de *resource* FHIR, o HAPI (descrito detalhadamente na secção 6.1)
- **Conversor –** Projeto que será desenvolvido para responder aos requisitos apresentados anteriormente.

5.2.1.4 Vista de implantação

É na vista de implantação/física que são considerados vários dos requisitos não funcionais do sistema a desenvolver [30].

Desta forma, é na vista física que são utilizados os diagramas de implantação que têm o intuito de ilustrar quais os diferentes intervenientes existentes no sistema, assim como onde estes se encontram [30].

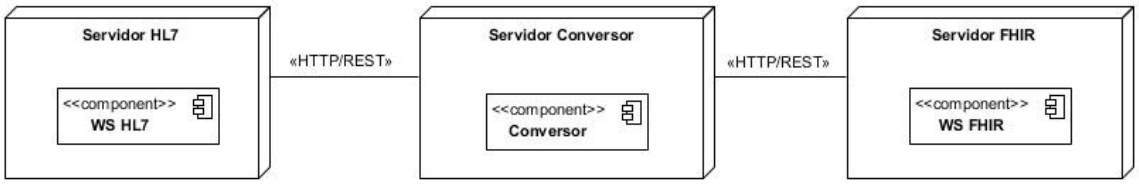

Figura 17 - Diagrama de Implantação

Na Figura 17 podemos observar como está definida a arquitetura do sistema a implementar, assim como os sistemas que já existem.

- No Servidor HL7 está presente o sistema referente ao protocolo HL7, que já foi explicado previamente;
- No Servidor FHIR é onde se encontra o sistema referente ao protocolo FHIR, que de igual forma se encontra explicado anteriormente;
- No Servidor Conversor encontra-se o sistema responsável pela conversão do protocolo HL7 para o protocolo FHIR.

5.2.2 Design arquitetural de nível 2 (componentes)

Nesta secção irá ser apresentado o modelo 4 +1 ao nível de componentes. É nesta secção que será dado mais foco ao projeto a desenvolver, ou seja, ao conversor.

5.2.2.1 Vista lógica

Na vista lógica irão ser apresentados os componentes que fazem parte do conversor. Os componentes são os seguintes:

- **ESB –** este é responsável por receber a mensagem HL7 e enviar a mesma para o *MessageBroker*;
- **SGBD –** é neste componente que fica armazenado todo o histórico das mensagens recebidas no ESB;

- ***MessageBroker*** **–** é neste componente que são colocadas em *queue* as mensagens recebidas e posteriormente consumidas.

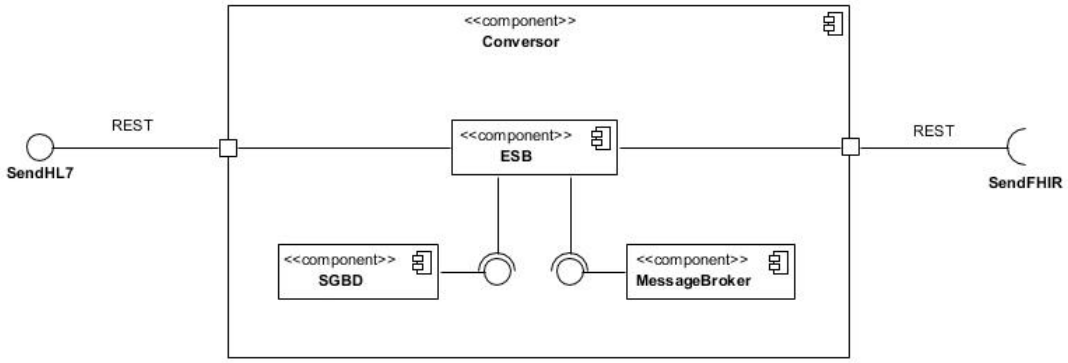

Figura 18 - Diagrama de Componentes (Detalhe do Conversor)

Na Figura 18 é possível observar quais os componentes presentes no sistema Conversor. Verificamos assim que o ponto de entrada é no componente ESB. A comunicação para este componente é feita através de REST, assim como a comunicação para o exterior.

O ESB tem a responsabilidade de comunicar com o componente SGBD, com o intuito de gravar todos os estados das mensagens ao longo do processo de transformação. Este também comunica com o MessageBroker, para que este guarde todas as mensagens numa *queue*. Posteriormente e consoante a disponibilidade do sistema, o ESB é responsável por fazer *polling* das mensagens e transformá-las. Uma vez concluído, envia o resultado dessas transformações para o FHIR, através de pedidos REST.

Em alternativa à proposta anteriormente apresentada, foi elaborado outro diagrama.

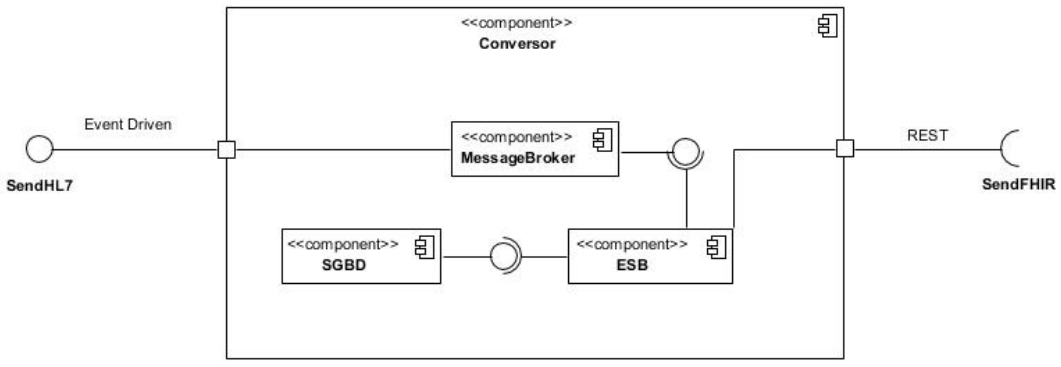

Figura 19 - Diagrama de componentes Alternativa

Na Figura 19 é possível observar que apesar de a comunicação para o WS FHIR ser a mesma do modelo apresentado na Figura 18, ou seja, REST, a comunicação com o sistema Conversor passa a ser *event driven*. Nesta alternativa, o componente de entrada passaria a ser o

MessageBroker, sendo que o restante fluxo seria o mesmo apresentado anteriormente. Para ser possível implementar esta solução é necessário que o componente WS HL7 comunique com o conversor usando uma abordagem *event driven*. Contudo, o WS HL7 não tem esta capacidade, pelo que obrigaria a alterar a forma como este comunica para o exterior.
Assim sendo, a alternativa adotada é a descrita/representada no diagrama da Figura 18.

5.2.2.2 Vista de processo

Na vista de processo irá ser apresentado com maior detalhe o processo referido anteriormente. A primeira responsabilidade do conversor é converter o protocolo HTTP para o protocolo AMQP, ou seja, no momento em que é recebida uma mensagem, é enviada uma resposta de sucesso para o emissor tornando assim o resto do processo assíncrono. A mensagem é enviada para o componente MessageBroker, sendo que este tem a responsabilidade de a colocar numa *Queue* (fila) e guardar o estado da mesma num SGBD. Em seguida o ESB faz *polling* da mensagem que existe na *queue* e envia-a para o pipeline de transformação da mensagem. É neste pipeline que é feita a transformação de uma mensagem HL7 num ou vários *resource* FHIR e igualmente guardado o estado, a mensagem, o(s) segmento(s) gerados da transformação no SGBD. Após ser(em) obtido(s) o(s) *resource* FHIR, este(s) é (são) enviado(s) para o WS FHIR, no qual será feita a validação do(s) *resource* gerado(s) e guardados em caso de sucesso.

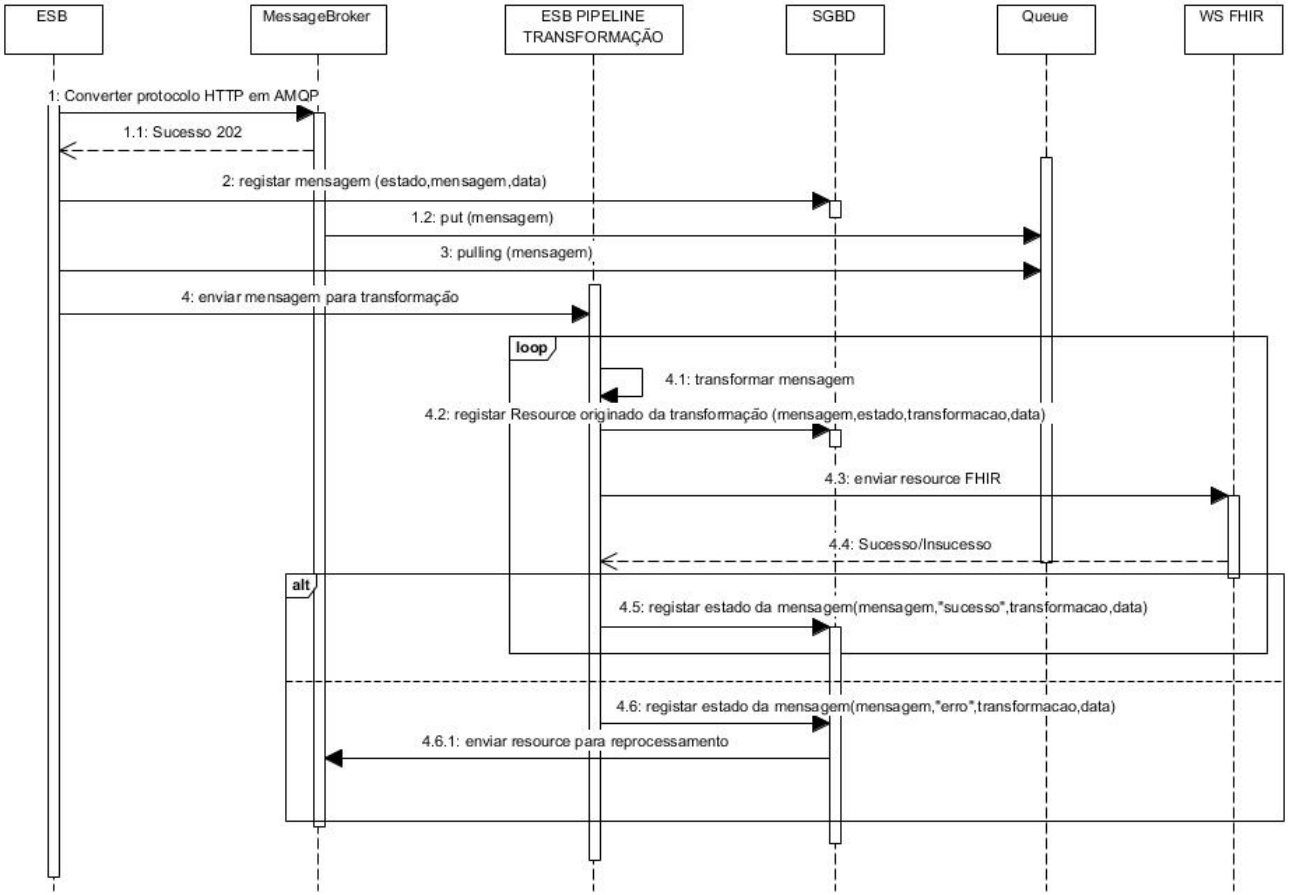

Figura 20 - Processo de transformação de mensagem

Na Figura 20 está representado a interação existente entre os componentes que compõem o Conversor. É possível observar que existem dois componentes que não foram mencionados anteriormente nos diagramas. O componente ESB Pipeline Transformação e o Queue. Estes componentes fazem parte do componente ESB e MessageBroker, foram representados neste diagrama para melhor compreensão da interação existente.

5.2.2.3 Vista de Implementação

Nesta vista irá ser apresentado o diagrama de implementação referente ao sistema Conversor, seguindo o que já foi sido mencionado nas vistas anteriores.

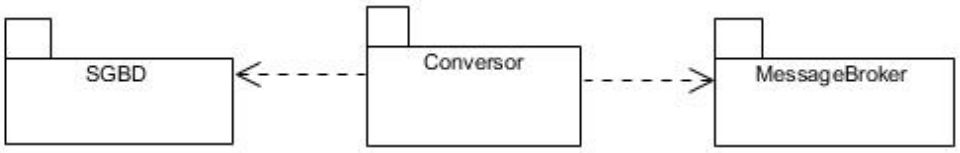

Figura 21 - Diagrama de implementação

Na Figura 21 podemos observar a estrutura do diagrama de implementação inerente ao projeto segundo o que foi apresentado nas vistas anteriores. É possível verificar quais os *packages* existentes, assim como as dependências entre os mesmos.
Existem assim 3 packages dentro do Conversor, sendo eles:

- **ESB –** Este é o orquestrador de todo o sistema do Conversor, este tem dependência dos restantes componentes;
- **SGBD –** É neste package que se encontra a base de dados do sistema do Conversor, ou seja, toda a informação quantos as mensagens processadas.
- **MessageBroker –** Este package é responsável por gerir todo o sistema de filas criado no sistema.

5.2.2.4 Vista de Implantação

Nesta vista será apresentado em detalhe o diagrama de implantação do sistema Conversor.

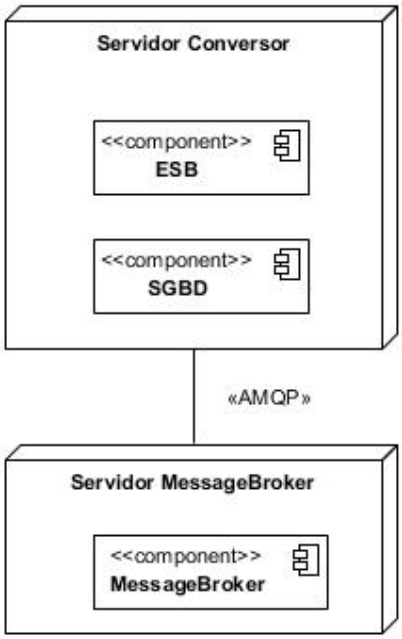

Figura 22 - Diagrama de Implantação

Na Figura 22 podemos observar como está definida a arquitetura do sistema a implementar, assim como os sistemas que já existem.

- No Servidor Conversor encontra-se o sistema responsável pela conversão do protocolo HL7 para o protocolo FHIR, assim como a conversão entre o protocolo HTTP no protocolo AMQP. É igualmente nesta máquina que se encontra o SGBD que é responsável por guardar todos os estados das mensagens;
- No Servidor *MessageBroker* é onde está o sistema de filas existentes, para o suporte do sistema de conversão.

5.3 **Síntese**

Neste capítulo foi descrito a abordagem concetual, que sugere que, para a realizar a conversão entre uma mensagem HL7 e um *resource* FHIR, são necessárias as duas fases apresentadas na secção 5.1. No design arquitetural de nível 1 (sistema) descrevem-se os principais componentes existentes no sistema: WS HL7, WS FHIR e o Conversor. O conversor será o componente que vai ser desenvolvido no projeto, sendo que este irá ser composto por 3 componentes: ESB, SGBD e *MessageBroker* tal como é descrito no design arquitetural de nível 2 (componentes).
Foi decidido usar um ESB porque a integração com um *MessageBroker* é simples e torna assim fácil a sua implementação. Um dos objetivos é registar todos os estados de cada fase de uma mensagem, desde que esta é recebida no ESB. Como tal, o uso do *MessageBroker* é crucial 5.1.

6 Implementação

Neste capítulo é sintetizada a implementação/construção do projeto onde irão ser apresentados quais os tipos de mensagens estudados e quais as transformações que foram realizadas. Neste capítulo serão também apresentados e detalhados os pipelines que foram criados para dar resposta aos objetivos apresentados anteriormente.

6.1 HAPI FHIR

No momento de desenvolvimento do projeto, a Glintt ainda não possuía uma API FHIR desenvolvida. Como tal, foi necessário usar uma API pública disponível para testar e validar os *resources* obtidos nas transformações criadas.

A API pública disponível para a validação dos *resources* é o HAPI [32].

O HAPI é também uma biblioteca de especificação de HL7 FHIR para JAVA. É possível utilizar esta biblioteca para implementar uma API FHIR, pois já possuí os modelos para os *resources* FHIR conhecidos.

Visto que a Glintt ainda não possuí o componente WS FHIR, esta API foi crucial para o desenvolvimento do projeto, pois só assim foi possível perceber os problemas que foram encontrados nos mapeamentos realizados.

Uma das limitações encontradas ao utilizar esta API para obter a validação dos *resources* FHIR gerados, é o facto de a base de dados (BD) presente no WS HL7 e nesta API serem diferentes. Uma das validações que é feita nesta API, é por exemplo: no *resource Patient* existe uma propriedade denominada por *link*, nesta propriedade é possível referenciar pacientes, entre outros. Quando na mensagem HL7 existe uma referência desse tipo, e a mapeamos para essa propriedade, a API vai validar se esse paciente existe na BD. Como as BD são diferentes, ela vai retornar uma mensagem de erro.

6.2 Pipelines, Mensagens e Transformações

Na implementação do projeto, foi decidido utilizar um ESB para o desenvolvimento do projeto, sendo que este é responsável por orquestrar todas as ações necessárias para realizar o objetivo proposto. O ESB pode ser desenvolvido utilizando diversas tecnologias como por exemplo: WSO2, Mulesoft, entre outros. Para este projeto foi escolhido o Mulesoft, uma vez que a empresa possui licença de utilização e os autor (e demais equipa) está mais contextualizado(a) com o mesmo.

Na secção 5.2 foram apresentadas diversas vistas que sustentam a arquitetura do software que foi desenvolvido. Assim esta secção tem como objetivo detalhar toda essa análise feita previamente.

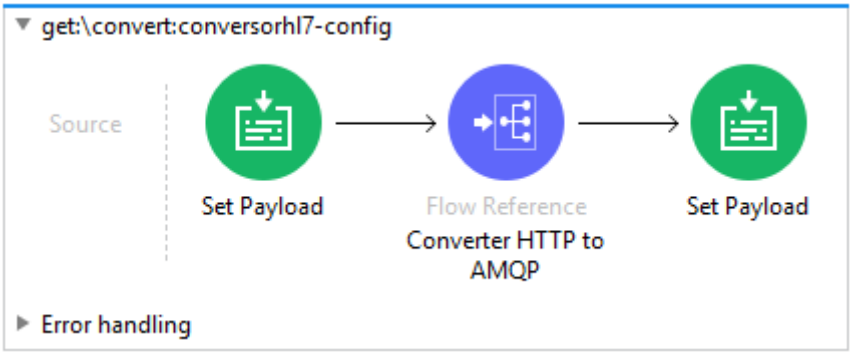

Figura 23 - Ponto de entrada no ESB

Na Figura 23 está representado o ponto de entrada no ESB, considerando assim que o primeiro passo do pipeline é converter o protocolo HTTP no AMQP e enviar uma resposta para o emissor notificando-o do sucesso da operação. Desta forma o fluxo passa a ser assíncrono.

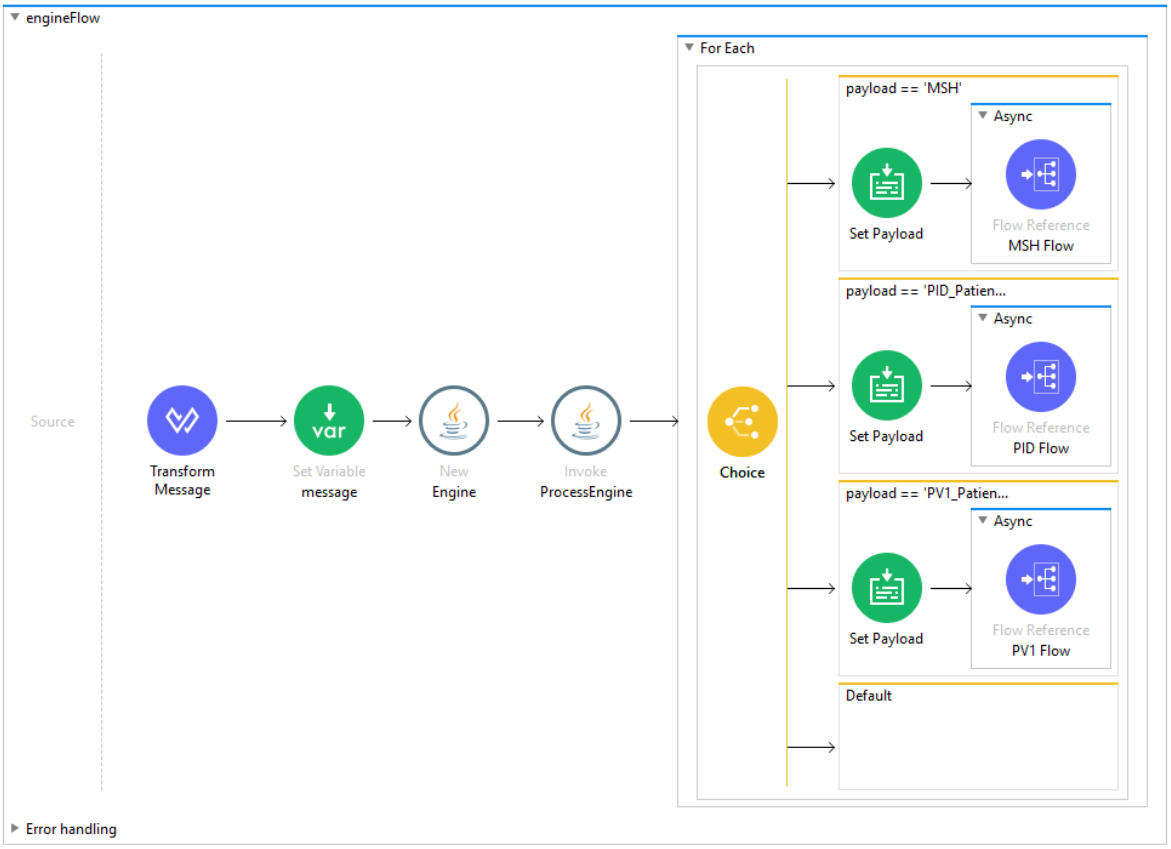

Figura 24 - Engine para escolha de criação de Resource

O passo ilustrado na Figura 24 é ler o XML da mensagem HL7 recebida e compreender quais as transformações que terão de ser feitas para obter os *resources* FHIR desejados. Tal como já foi explicado anteriormente, uma mensagem HL7 possuí diferentes segmentos e cada um deles poderá dar origem a um ou vários *resources* FHIR. Quando a mensagem constituí um segmento MSH, consequentemente será invocado o pipeline de conversão desse segmento (MSH Flow). O mesmo se aplica ao segmento PID e PV1. Sempre que for necessário acrescentar uma transformação de um novo segmento, será neste pipeline que será acrescentada.

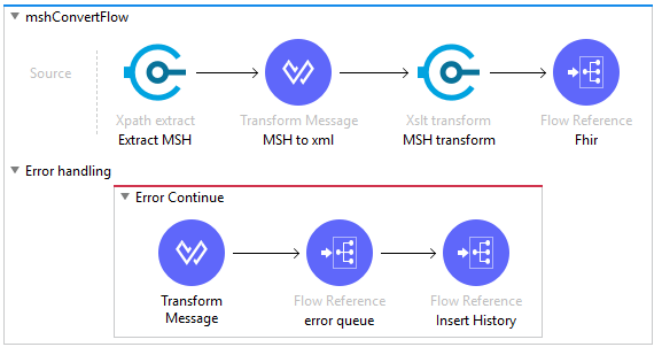

Figura 25 - MessageHeader Pipeline de transformação

Na Figura 25 podemos observar um exemplo de uma pipeline de transformação, que neste caso trata o segmento MSH. O primeiro passo é a extração do segmento MSH da mensagem recebida. Em seguida é convertido para XML esse segmento e utilizado o XSLT para concretizar a transformação. Em caso de sucesso é enviado o conteúdo da transformação para a API FHIR, mas, por outro lado, caso ocorra um erro é enviado para *queue* de reprocessamento e registado o estado da mensagem e o respetivo erro na SGBD.

```
<MSH>
    <MSH.2_EncodingCharacters>^~\&amp;</MSH.2_EncodingCharacters>
    <MSH.3_SendingApplication>
        <HD.0_NamespaceId>GH</HD.0_NamespaceId>
    </MSH.3_SendingApplication>
    <MSH.4_SendingFacility>
        <HD.0_NamespaceId>TESTEAPP</HD.0_NamespaceId>
    </MSH.4_SendingFacility>
    <MSH.5_ReceivingApplication>
        <HD.0_NamespaceId>HEPIC</HD.0_NamespaceId>
    </MSH.5_ReceivingApplication>
    <MSH.6_ReceivingFacility>
        <HD.0_NamespaceId>TESTEAPP</HD.0_NamespaceId>
    </MSH.6_ReceivingFacility>
    <MSH.7_DateTimeOfMessage>
        <TS.1>20190524172049</TS.1>
    </MSH.7_DateTimeOfMessage>
    <MSH.9_MessageType>
        <CM_MSG.0_MessageType>ADT</CM_MSG.0_MessageType>
        <CM_MSG.1_TriggerEvent>A01</CM_MSG.1_TriggerEvent>
        <CM_MSG.2_MessageStructure/>
    </MSH.9_MessageType>
    <MSH.10_MessageControlId>1138126691</MSH.10_MessageControlId>
    <MSH.11_ProcessingId>
        <PT.0_ProcessingId>P</PT.0_ProcessingId>
    </MSH.11_ProcessingId>
    <MSH.12_VersionId>
        <VID_0_VersionId>2.4</VID_0_VersionId>
    </MSH.12_VersionId>
    <MSH.15_AcceptAcknowledgmentType>AL</MSH.15_AcceptAcknowledgmentType>
    <MSH.16_ApplicationAcknowledgmentType/>
    <MSH.18_CharacterSet/>
</MSH>
```

Figura 26 - Exemplo do segmento MSH

Na Figura 26 podemos observar um exemplo do segmento MSH do protocolo HL7, que neste caso corresponde a uma mensagem do tipo ADT-A01. Tal como foi ilustrado na Figura 25, o segundo passo do pipeline de transformação é a aplicação do XSLT criado.

Para a criação do XSLT foi utilizada a ferramenta ALTOVA MapForce onde dado um *Data Source* e um *Data Target*, é possível definir o mapeamento entre os dois, gerando assim um

XSLT.

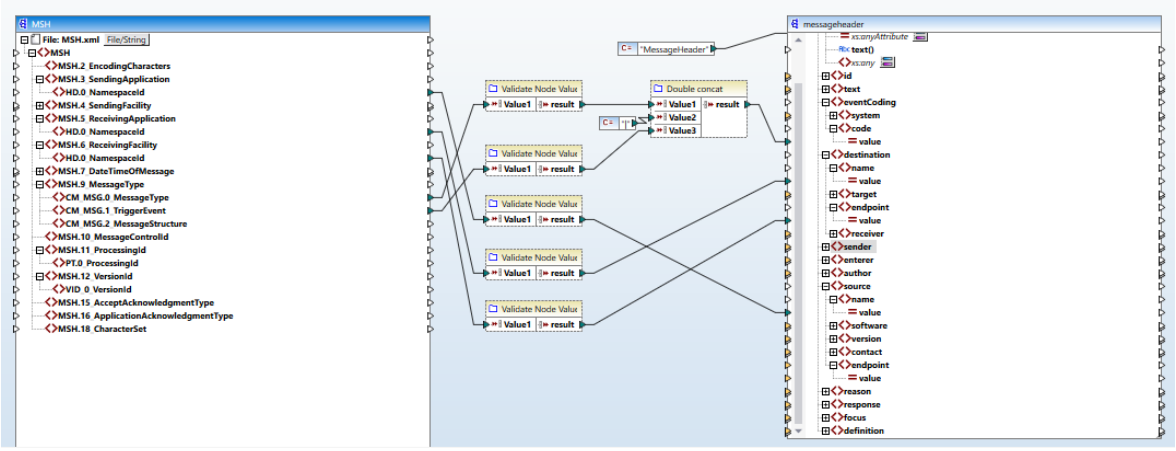

Figura 27 - Altova MapForce mapeamento

Na Figura 27 é possível observar o mapeamento feito entre o HL7 e o FHIR. Utilizando o Altova MapForce para o mapeamento, é obtido um XSLT. O XSLT estará presente na secção 10.1. Podemos verificar ainda que existem segmentos no lado de HL7 que não têm qualquer correspondência no lado do FHIR. Para ser possível a validação no HAPI foi aplicada a transformação demonstrada na Figura 27. Contudo, foi feita uma alternativa a essa transformação com a finalidade de cumprir um dos objetivos mencionados anteriormente: a não perda de dados da mensagem origem.

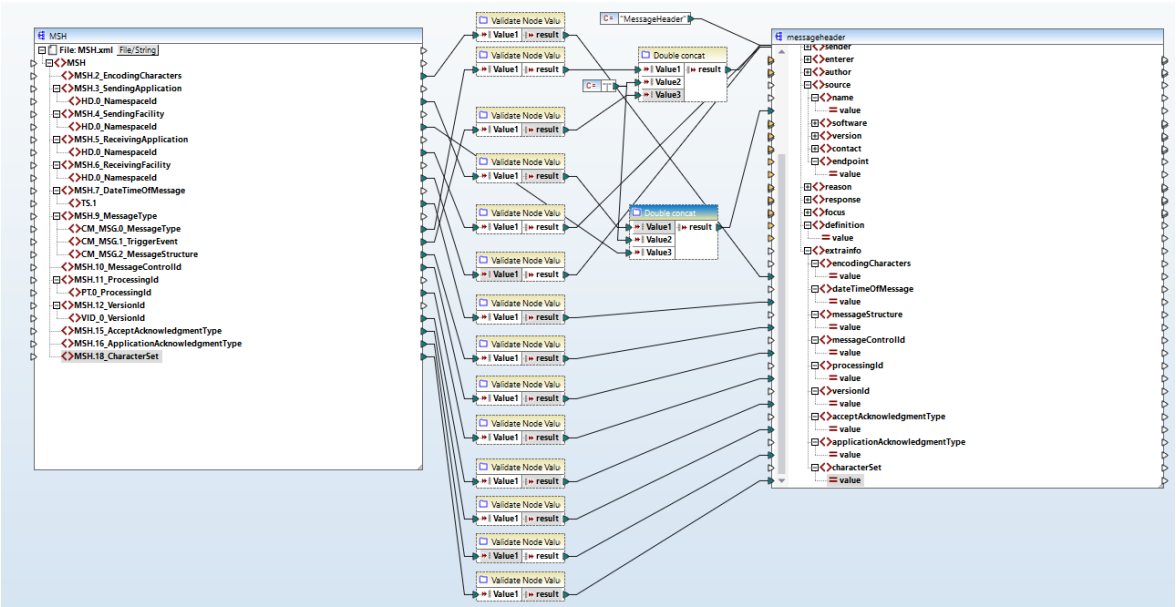

Figura 28 - Altova MapForce mapeamento total do MSH

A Figura 28 ilustra a alternativa feita para a não perda de dados mencionada anteriormente.

```
<MessageHeader xmlns="http://hl7.org/fhir"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <resourceType>MessageHeader</resourceType>
   <eventCoding>
      <code value="ADT|A01"/>
   </eventCoding>
   <destination>
      <name value="HEPIC"/>
      <endpoint value="TESTEAPP"/>
   </destination>
   <source>
      <name value="GH|TESTEAPP"/>
   </source>
   <messageDatetime value="20190524172049"/>
   <messageControlId value="1138126691"/>
   <processingId value="P"/>
   <versionId value="2.4"/>
   <acceptAcknowledgmentType value="AL"/>
</MessageHeader>
```

Figura 29 - Resultado da transformação

Na Figura 29 está representado o resultado da transformação do segmento MSH (HL7) para *MessageHeader* (FHIR), aplicando a primeira transformação mencionada na Figura 27. Depois de ser feita a obtenção do *resource*, este é enviado para o pipeline de envio para o WS FHIR.

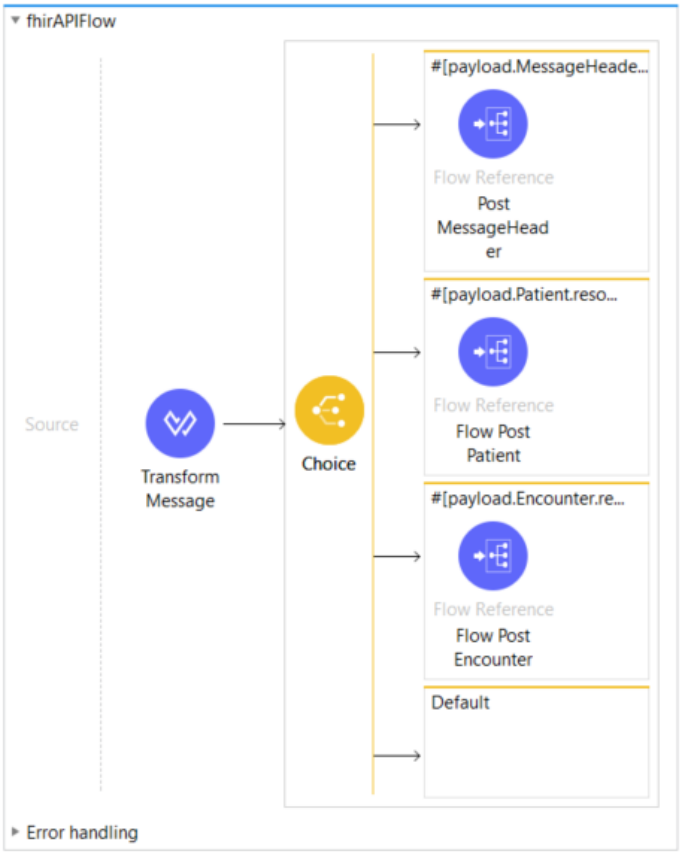

Figura 30 - Pipeline de envio do resource FHIR

Na Figura 30 está representado o pipeline de envio para o WS FHIR. À semelhança do que é feito no pipeline *Engine*, é necessário perceber qual o *resource* obtido neste pipeline, e enviar o mesmo para o pipeline definido para esse *resource*.

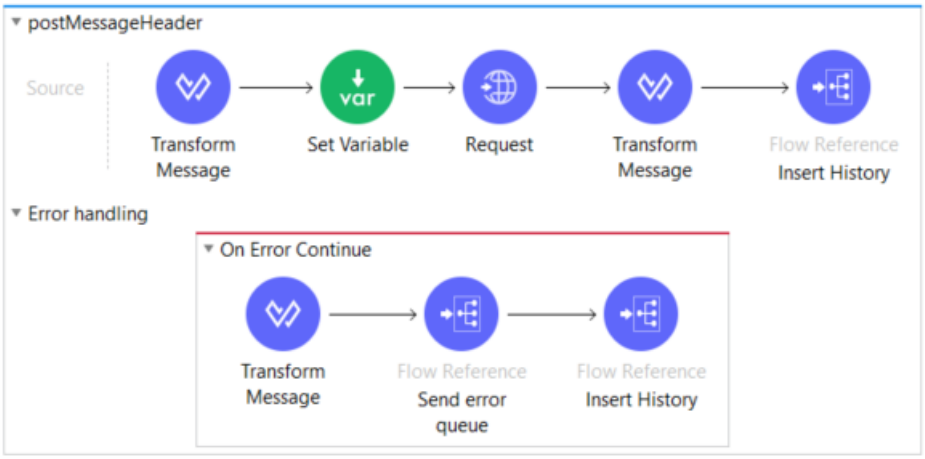

Figura 31 - Pipeline de Envio do MessageHeader para o WS FHIR

Na Figura 31 é possível observar o pipeline definido para o envio do *resource MessageHeader*. Todavia, empregando a API FHIR para validar a transformação alcançada, devido ao facto de essas extensões não serem suportadas pelo FHIR, a validação dá *false*.

```
PID_PatientIdentification>
   <PID_3_PatientIdentifierList>
      <CX_0_Id>60736492</CX_0_Id>
      <CX_3_AssigningAuthority>
         <CX_3_0_NamespaceId>JMS</CX_3_0_NamespaceId>
      </CX_3_AssigningAuthority>
      <CX_4_IdentifierTypeCodeId>NS</CX_4_IdentifierTypeCodeId>
   </PID_3_PatientIdentifierList>
   <PID_3_PatientIdentifierList>
      <CX_0_Id>799566</CX_0_Id>
      <CX_3_AssigningAuthority>
         <CX_3_0_NamespaceId>CUFP</CX_3_0_NamespaceId>
      </CX_3_AssigningAuthority>
      <CX_4_IdentifierTypeCodeId>NS</CX_4_IdentifierTypeCodeId>
   </PID_3_PatientIdentifierList>
   <PID_4_AlternatePatientIdPid>
      <CX_0_Id>206435150</CX_0_Id>
      <CX_3_AssigningAuthority>
         <CX_3_0_NamespaceId>NIF</CX_3_0_NamespaceId>
      </CX_3_AssigningAuthority>
      <CX_4_IdentifierTypeCodeId>PT</CX_4_IdentifierTypeCodeId>
   </PID_4_AlternatePatientIdPid>
   <PID_5_PatientName>
      <XPN_0_FamilyName>
         <XPN_0_0_Surname>SIRNAME</XPN_0_0_Surname>
      </XPN_0_FamilyName>
      <XPN_1_GivenName>JO\XC3\O</XPN_1_GivenName>
      <XPN_2_SecondAndFurtherGivenNamesOrInitialsThereof>JOAQUIM LOPES</XPN_2_SecondAndFurtherGivenNamesOrInitialsThereof>
   </PID_5_PatientName>
   <PID_7_DateTimeOfBirth>
      <TS_0_TimeOfAnEvent>19901025000000</TS_0_TimeOfAnEvent>
   </PID_7_DateTimeOfBirth>
   <PID_8_AdministrativeSex>M</PID_8_AdministrativeSex>
   <PID_11_PatientAddress>
      <XAD_0_StreetAddressSad>
         <XAD_0_0_StreetOrMailingAddress>SEM NOME</XAD_0_0_StreetOrMailingAddress>
      </XAD_0_StreetAddressSad>
      <XAD_1_OtherDesignation>COIMBRA</XAD_1_OtherDesignation>
      <XAD_2_City>COIMBRA</XAD_2_City>
      <XAD_3_StateOrProvince>6</XAD_3_StateOrProvince>
      <XAD_4_ZipOrPostalCode>3030-168</XAD_4_ZipOrPostalCode>
      <XAD_5_Country>1</XAD_5_Country>
      <XAD_7_OtherGeographicDesignation/>
   </PID_11_PatientAddress>
   <PID_13_PhoneNumberHome>
      <XTN_3_EmailAddress>teste@gmail.com</XTN_3_EmailAddress>
      <XTN_6_PhoneNumber>964078974</XTN_6_PhoneNumber>
   </PID_13_PhoneNumberHome>
   <PID_14_PhoneNumberBusiness>
      <XTN_6_PhoneNumber/>
   </PID_14_PhoneNumberBusiness>
   <PID_16_MaritalStatus>
      <CE_0002_0_IdentifierSt/>
      <CE_0002_1_Text/>
   </PID_16_MaritalStatus>
   <PID_19_SsnNumberPatient>281668601</PID_19_SsnNumberPatient>
   <PID_23_BirthPlace/>
   <PID_28_Nationality>
      <CE_0212_0_IdentifierSt>1</CE_0212_0_IdentifierSt>
      <CE_0212_1_Text>PORTUGAL</CE_0212_1_Text>
   </PID_28_Nationality>
   <PID_29_PatientDeathDateAndTime>
      <TS_0_TimeOfAnEvent/>
   </PID_29_PatientDeathDateAndTime>
   <PID_30_PatientDeathIndicator>N</PID_30_PatientDeathIndicator>
/PID_PatientIdentification>
```

Figura 32 - Segmento PID exemplo

Na Figura 32 está representado um exemplo de um segmento PID constituído por diversos segmentos.

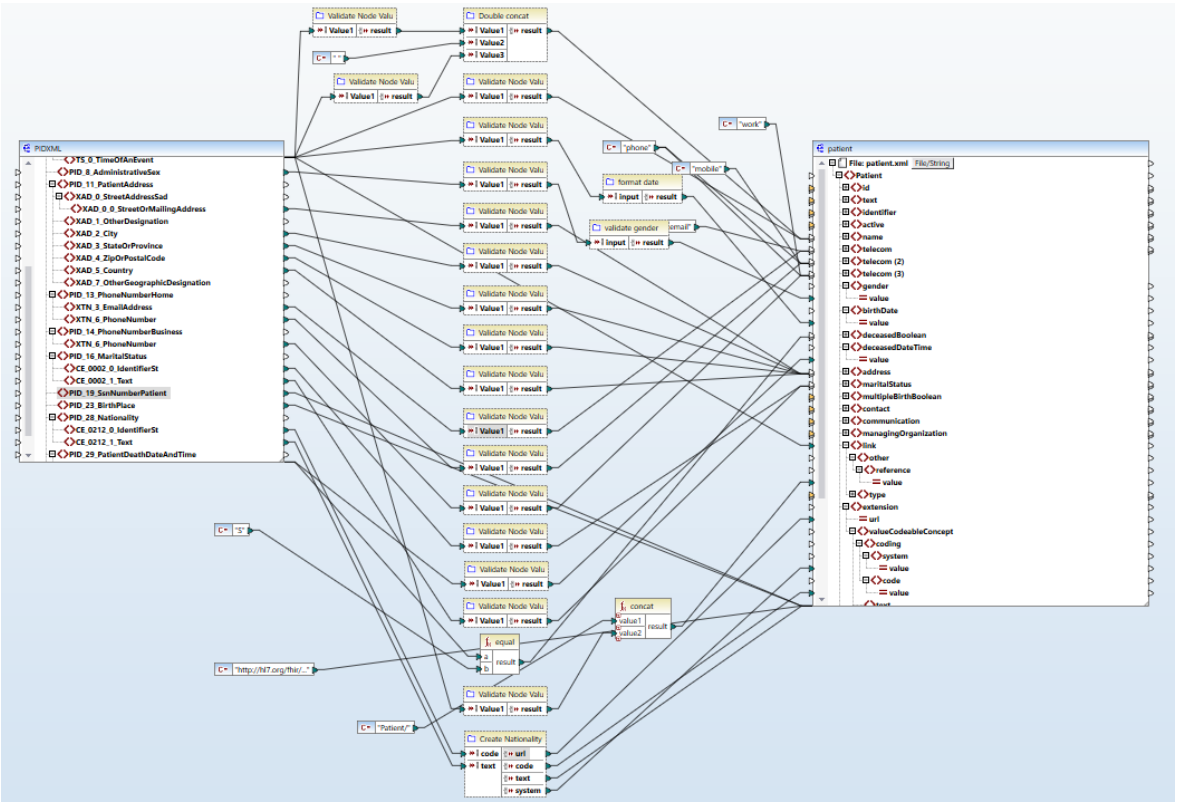

Figura 33 - Altova MapForce mapeamento do segmento PID

É possível observar na Figura 33 o mapeamento feito entre o segmento PID e o *resource Patient*.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<Patient xmlns="http://hl7.org/fhir">
    <resourceType>Patient</resourceType>
    <name>
        <family value="SIRNAME"/>
        <given value="JOXC3O JOAQUIM LOPES"/>
    </name>
    <telecom>
        <system value="email"/>
        <value value="teste@gmail.com"/>
    </telecom>
    <telecom>
        <system value="phone"/>
        <value value="964078974"/>
        <use value="mobile"/>
    </telecom>
    <telecom>
        <system value="phone"/>
        <value/>
        <use value="work"/>
    </telecom>
    <gender value="male"/>
    <birthDate value="1990-10-25"/>
    <deceasedBoolean value="false"/>
    <deceasedDateTime/>
    <address>
        <line value="SEM NOME"/>
        <city value="COIMBRA"/>
        <state value="6"/>
        <postalCode value="3030-168"/>
        <country value="1"/>
    </address>
    <maritalStatus>
        <coding>
            <code/>
        </coding>
        <text/>
    </maritalStatus>
    <link>
        <other>
            <reference value="Patient/60736492"/>
        </other>
    </link>
    <link>
    <extension url="http://hl7.org/fhir/StructureDefinition/patient-nationality">
        <valueCodeableConcept>
            <coding>
                <system value="urn:iso:std:iso:3166"/>
                <code value="1"/>
            </coding>
            <text>PORTUGAL</text>
        </valueCodeableConcept>
    </extension>
</Patient>
```

Figura 34 - Resource Patient obtido exemplo

Na Figura 34 está ilustrado o *resource Patient* obtido da transformação e verificamos que existe um campo denominado por *link*. Esse campo é utilizado segundo a comunidade FHIR, para listar referências a pacientes existentes que podem vir ou não na mensagem HL7. Neste caso em concreto essa informação está presente na mensagem e por isso é transformada.

Como as bases de dados são diferentes no componente WS HL7 e no componente WS FHIR, durante a validação deste *resource* irá dar erro. É devolvida uma mensagem de erro informando que os pacientes referenciados naquele campo não existem no sistema. Isto não invalida que a transformação não tenha sido feita da forma correta, mas não é possível obter sucesso do WS FHIR.

Anteriormente apenas foram apresentados dois exemplos, mas foram desenvolvidas mais transformações neste projeto.

6.3 MessageBroker

O RabbitMQ foi seleccionado como *messagebroker* para o desenvolvimento deste projeto, pois é um *messagebroker* já conhecido pela empresa, contudo, poderia ter sido escolhido qualquer outro *messagebroker* como por exemplo o Kafka.

O RabbitMQ apresenta-se como uma solução simples de utilizar e configurar, assim, para a implementação deste projeto foram abordadas as diferentes funcionalidades que o mesmo apresenta.

Na implementação deste projeto foram definidos dois *exchanges* distintos. Um para o envio das mensagens para as *pipelines* de transformação e o outro para reprocessar as mensagens que obtiveram erros no envio para a API FHIR.

O RabbitMQ provê quatro tipos de *exchanges* [22]:

- *Fanout* - envia mensagens para todas as filas de mensagem que estão à escuta neste *Exchange*;
- *Direct* - apenas envia mensagens para as filas de mensagens que foram discriminadas através de uma chave de roteamento;
- *Topic* – é semelhante ao *direct*, porém permite que a chave de roteamento utilize os *wildcards* "#" e "*" para encontrar as filas de mensagens. O "#" representa zero ou mais palavras e o "*" representa apenas uma palavra;
- *Headers*: também semelhante ao *direct*, mas ao invés de uma chave de roteamento, utiliza um conjuto de chave-valores para decidir quais as filas de mensagens que vão receber as mensagens. Permite que o envio de mensagens para uma fila seja mais filtrado.

Os dois *exchanges* configurados são do tipo *direct*, porque é pretendido que todos os eventos recebidos sejam apenas consumidos por um consumidor definido. É possível também

configurar se o *Exchange* e a fila de mensagens devem sobreviver quando um nó do RabbitMQ reinicia.

Nesta implementação foi determinado que não se deve perder os *exchanges* quando o sistema é reiniciado e as filas de mensagens apenas são apagadas quando o último consumidor é desligado.

Tal como foi explicado anteriormente, o *messagebroker* possuí um papel crucial no projeto desenvolvido, pois é nele que é através dele que é feita toda a gestão dos estados da mensagem recebida.

6.4 Síntese

Em suma, para cada processo existe um pipeline bem definido e com uma determinada função. Existem 3 grandes processos:

- Conversão do protocolo HTTP para AMQP;
- Conversão de mensagem HL7 para *resource* FHIR;
- Envio do *resource* obtido para o WS FHIR.

Foram feitas várias transformações entre um protocolo e outro, de alguns segmentos. Contudo, não foi possível em todos eles obter uma validação correta do WS FHIR.

7 Experiências e avaliação

Neste capítulo são descritos os processos de experimentação e avaliação usados para avaliar a solução desenvolvida. No ponto seguinte, irá ser definida a abordagem utilizada para avaliar a solução final do projeto e com base nessa abordagem será possível avaliar a qualidade de todo o trabalho realizado.

7.1 Abordagem

Para realizar o processo de avaliação foram definidas duas grandezas que irão ajudar a avaliar a solução aqui documentada:

- **Desempenho:** utilizada para medir o desempenho da solução desenvolvida. É esperado que o tempo de resposta da solução seja inferior a 20 ms.
- **Eficácia:** verificar que os recursos FHIR obtidos são aceites pelo validador/API FHIR.

Para avaliar a grandeza e testar a hipótese referida anteriormente é necessária a definição de metodologias de avaliação:

- **Monitorização do desempenho da arquitetura:** definição de rotinas ou planos de monitorização para averiguar se o desempenho da arquitetura vai de encontro ao esperado. Existem várias ferramentas que permitem avaliar o desempenho de sistemas;
- **Monitorização do sucesso/falha na validação dos recursos FHIR:** validação dos recursos FHIR obtidos após a transformação.

7.2 Resultados de desempenho

Para testar a arquitetura e compreender se a mesma satisfaz os objetivos que se pretende atingir, foram definidas as seguintes métricas para avaliar o desempenho:

- Número médio de mensagens que são processadas por segundo (publicação e consumo);
- Tempo médio de resposta ao utilizador.

Além das métricas mencionadas anteriormente foi ainda definida uma variante para estes testes:

- Quantidade de pedidos efetuados num determinado espaço de tempo (10 segundos). Enviando 50, 100, 1000, 5000 e 10000 pedidos.

Para avaliar as referidas métricas foram efetuados testes de carga a todo o sistema usando ferramentas de terceiros. Para tal, foram avaliadas algumas dessas ferramentas:

- ***HttpPerf*:** simples de usar, os resultados são obtidos através de linha de comandos;
- ***SmartMeter*:** apenas está disponível a versão trial;

- ***Apache JMeter*:** é bastante utilizado e é desenvolvida pela Apache. Possuí uma interface gráfica para desenhar os testes e gera gráficos e tabelas com os resultados obtidos.

Para realizar os testes de carga e também avaliar algumas das métricas definidas previamente, foi escolhida a ferramenta *JMeter*, tendo sido também utilizada a *RabbitMQ API Management*, disponibilizada pelo próprio *RabbitMQ*. A *RabbitMQ* oferece uma interface gráfica que permite retirar métricas relacionadas com as filas de mensagem definidas.

No *JMeter* é possível definir planos de teste através da sua interface gráfica. Desta forma, foi pormenorizado um período de tempo de 10 segundos para efetuar pedidos ao sistema implementado. Para além deste período de tempo, foi também definido o número de processos que vão ser corridos, ou seja, o número de pedidos que vão ser efetuados durante esse período. Na Figura 35 podemos observar a configuração do plano de testes.

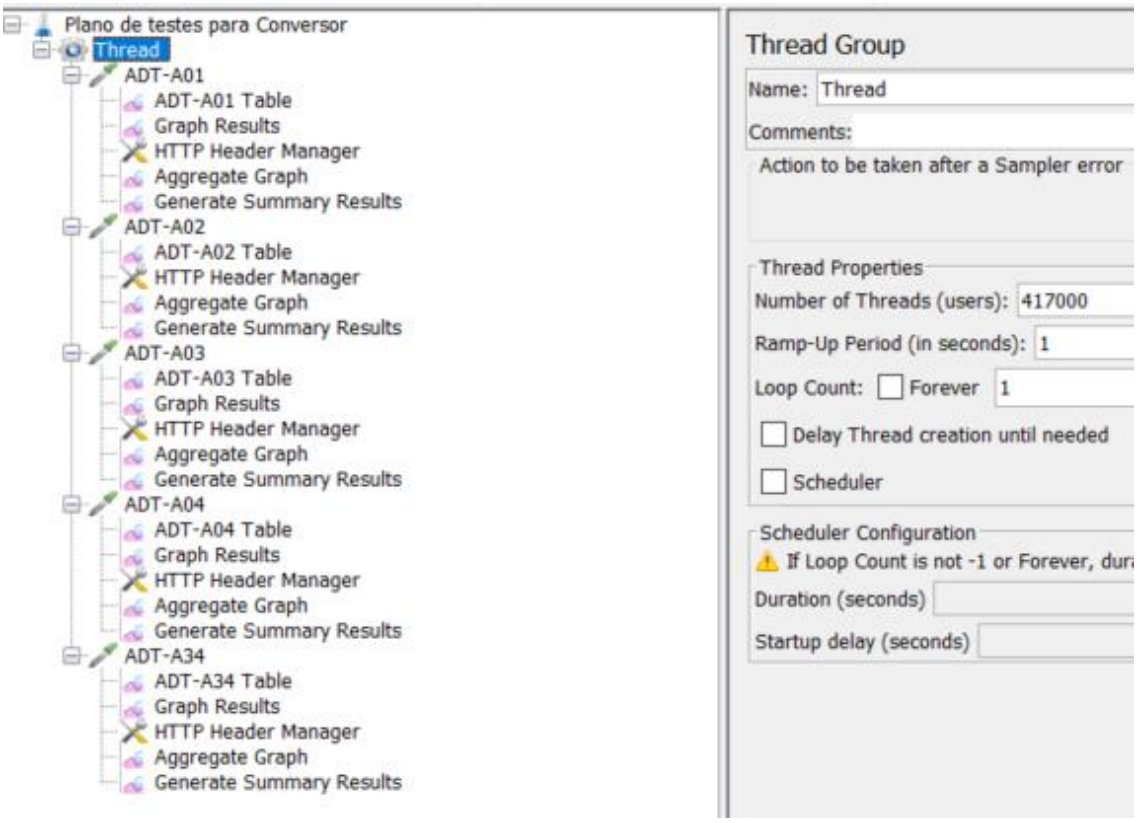

Figura 35 - Configuração do plano de testes no JMeter

Os resultados dos testes efetuados podem ser observados na interface gráfica fornecida pelo *JMeter* na Tabela 14.

Tabela 14 - Resultado dos testes de carga por tipo de mensagem

Tipo Mensagem	# Mensagens	Tempo de envio (ms)	Taxa de erro (%)
ADT-A01	50	7	0

ADT-A01	100	7	0
ADT-A01	1000	10	0
ADT-A02	50	8	0
ADT-A02	100	8	0
ADT-A02	1000	8	0
ADT-A03	50	6	0
ADT-A03	100	6	0
ADT-A03	1000	6	0
ADT-A04	50	7	0
ADT-A04	100	7	0
ADT-A04	1000	8	0
ADT-A34	50	8	0
ADT-A34	100	8	0
ADT-A34	1000	15	0

Na Tabela 14 é possível observar o tempo de envio em milissegundos (ms) em média por cada tipo de mensagem, influenciando o número de pedidos feitos ao conversor. Verificamos que a percentagem de erro é de 0%, o que já era expectável visto que após ser rececionada uma mensagem, o conversor envia essa mensagem para o MessageBroker, tornando o resto do processo assíncrono, e apenas envia uma resposta de sucesso para o emissor do pedido.

Para além dos testes de carga demonstrados anteriormente, foram feitos testes através do POSTMAN para a validação dos *resources* FHIR obtidos na conversão.

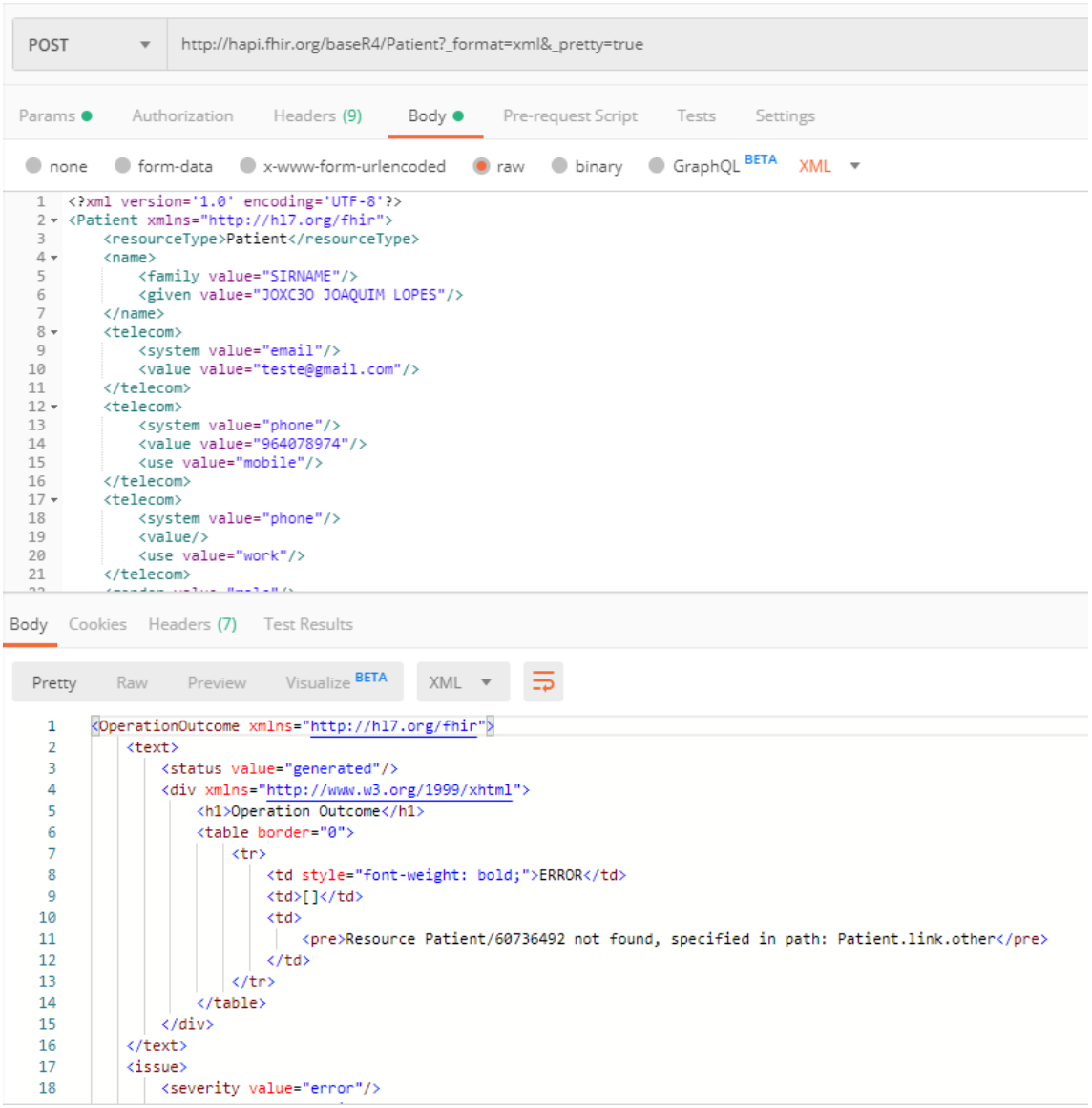

Figura 36 - Pedido POST para criação de paciente erro

Na Figura 36 é possível observar um exemplo de um erro de validação feito pelo HAPI. Este problema deve-se ao facto de que o paciente referenciado no *resource Patient* não existe na SGBD do HAPI. Isto acontece porque as BD do emissor e do recetor não são as mesmas, tal como já foi explicado anteriormente.

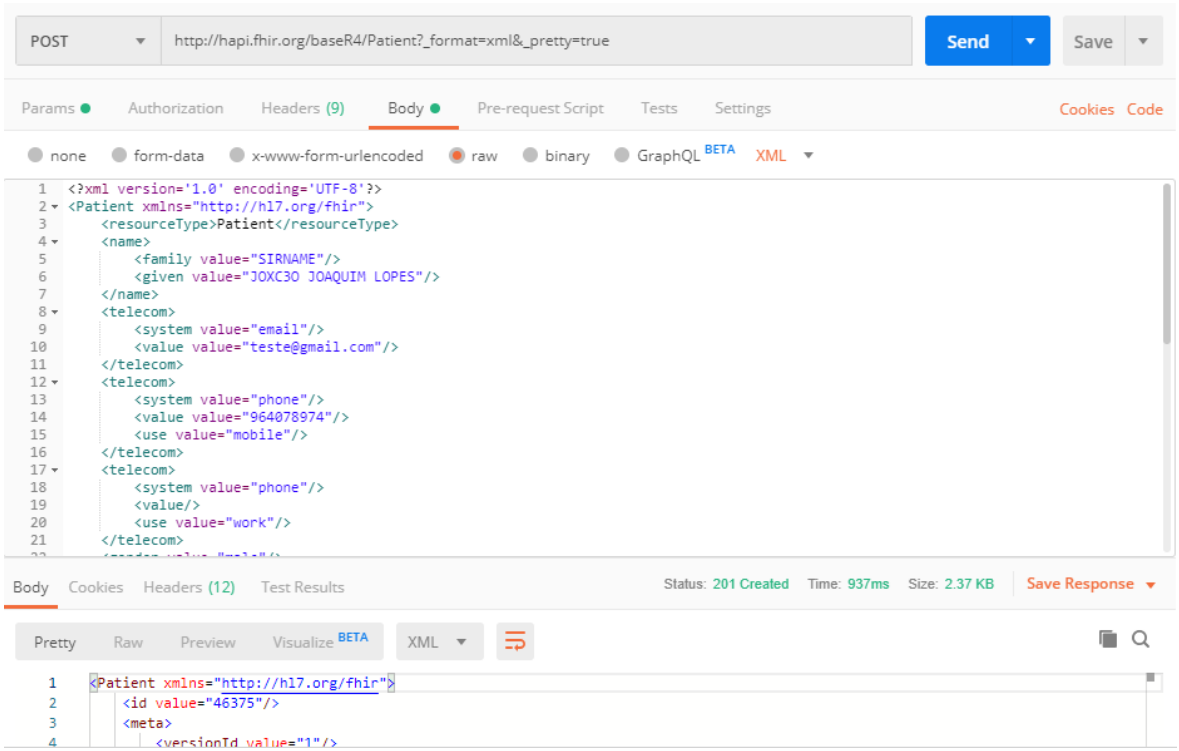

Figura 37 - Pedido POST para criação de paciente sucesso

Na Figura 37 é possível observar um exemplo de sucesso na criação de paciente.

7.3 Síntese

Durante os testes de carga foram detetados problemas na arquitetura do conversor, sendo que estes foram corrigidos para ser possível atingir os valores apresentados na Tabela 14. Foi por isso crucial a execução destes testes para compreender possíveis falhas quanto ao aumento de carga.

Foram ainda feitos testes através dum cliente HTTP (i.e. Postman) para validar as respostas da validação das conversões efetuadas. Através destes testes foram detetados alguns problemas de mapeamento que foram posteriormente corrigidos.

8 Conclusões

Neste capítulo serão apresentadas as conclusões finais de todo o trabalho desenvolvido, onde são referidos os objetivos alcançados, assim como os não alcançados. Também será dado a conhecer o trabalho futuro que se espera realizar.

8.1 Objetivos Alcançados

O objetivo principal do projeto desenvolvido consistiu no desenvolvimento de uma solução que permitisse transformar o conteúdo de uma mensagem HL7 num *resource* FHIR. Da mesma forma, também foi pretendido que a manutenção da solução e a sua evolução fossem simples, e para que isso acontecesse foram criados pipelines com responsabilidades singulares dependendo do seu propósito.
De um modo geral, o objetivo foi cumprido:

- existem pipelines bem definidos para a execução da transformação;
- é guardado o histórico de todas as fases da mensagem desde o momento da receção da mesma;
- é obtido um *resource* FHIR.

Neste projeto apenas foram aplicados alguns exemplos de mensagens existentes em HL7 como prova de conceito. Porque em muitos casos não existem regras de mapeamento entre um protocolo e outro, foi necessário analisar os dois esquemas e propor o mapeamento, que nalguns casos implica o desenvolvimento de extensões ou utilização de outras propostos pela comunidade.
Quando o componente WS FHIR estiver desenvolvido pela Glintt e a BD for comum quer do lado WS HL7, quer do lado WS FHIR, será muito mais fácil a integração entre os dois protocolos.

8.2 Objetivos Não Alcançados

A solução apresentada não dá resposta a todos os casos apresentados, pelo que não foi possível converter todos os segmentos presentes na mensagem.
De facto, usando extensões, a validação das mensagens é mais complicada ou impossível, porquanto não existem validadores automáticos predefinidos nas ferramentas, como é o caso da aplicação FHIR usada: o HAPI.

Existe ainda muita falta de informação quanto aos *resources* que deverão ser utilizados para a conversão. Tudo isto torna difícil o desenvolvimento de uma solução capaz de dar resposta a diferentes aplicações. Pois não existe nenhum modelo internacional suficientemente maduro para a definição das conversões necessárias.

Existem ainda muitas dúvidas quanto ao tipo de pedido REST necessário fazer consoante o tipo/evento de mensagem, por exemplo: na admissão de doentes foi assumido que seriam pedidos POST. Contudo, não significa que sejam desse tipo, pois o paciente pode já existir e não será criado um novo.

Nesta fase, o conversor está muito dependente da forma como está implementada a API FHIR, sendo que o objetivo é que este componente seja autónomo e genérico o suficiente para integrar qualquer sistema.

8.3 Trabalho Futuro

Como trabalho futuro, seria interessante implementar uma conversão inversa, ou seja, uma conversão de FHIR para HL7. Tirando assim partido de um conversor bidirecional.

É também expectável que sejam criadas mais transformações para os restantes segmentos existentes no HL7, e criar assim um conversor sólido e completo de HL7 para FHIR.

É expectável que seja explorado ainda mais a abordagem concetual aplicada, pois existe uma grande margem de progressão e potencial, no sentido em que ainda não existe solução para este problema capaz de dar resposta a todos os casos possíveis.

9 Referências bibliográficas

[1] Glintt, «Sobre a Glintt». [Em linha]. Disponível em: https://www.glintt.com/pt/o-que-somos/sobreaglintt/Paginas/default.aspx. [Acedido: 24-Fev-2019].

[2] J. Arriscado Nunes, «Centro de Estudos Sociais, Faculdade de Economia da Universidade de Coimbra», *Rech. En Anthropol. Au Port.*, vol. 4, n. 1, pp. 61–64, 1992.

[3] V. Zwass, «information system | Definition, Examples, & Facts», *Encyclopedia Britannica*. [Em linha]. Disponível em: https://www.britannica.com/topic/information-system. [Acedido: 24-Fev-2019].

[4] E. D'Souza, «Sistemas de Informação: Tipos de Sistemas de Informação», *Sistemas de Informação*, 2009. .

[5] «Tipos de Sistemas de Informação - PDF». [Em linha]. Disponível em: https://docplayer.com.br/722346-Tipos-de-sistemas-de-informacao.html. [Acedido: 24-Fev-2019].

[6] «Faculty of Business and Economics (HEC Lausanne)». [Em linha]. Disponível em: https://www.unil.ch/hec/en/home.html. [Acedido: 24-Fev-2019].

[7] S. Cohen e R. Ram, «(54) METHODOLOGY FOR MAPPING HL7 V2», p. 21.

[8] HL7, «Health Level Seven International». [Em linha]. Disponível em: https://www.hl7.org/documentcenter/. [Acedido: 24-Fev-2019].

[9] J. C. Mandel, D. A. Kreda, K. D. Mandl, I. S. Kohane, e R. B. Ramoni, «SMART on FHIR: a standards-based, interoperable apps platform for electronic health records», *J. Am. Med. Inform. Assoc.*, vol. 23, n. 5, pp. 899–908, Set. 2016.

[10] D. Bender e K. Sartipi, «HL7 FHIR: An Agile and RESTful approach to healthcare information exchange», em *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, 2013, pp. 326–331.

[11] «Comparison-v2 - FHIR v4.0.0». [Em linha]. Disponível em: https://www.hl7.org/fhir/comparison-v2.html. [Acedido: 24-Fev-2019].

[12] «What is REST?», *Codecademy*. [Em linha]. Disponível em: https://www.codecademy.com/articles/what-is-rest. [Acedido: 11-Out-2019].

[13] M. Rouse, «What is SOAP (Simple Object Access Protocol)? - Definition from WhatIs.com», *SearchMicroservices*. [Em linha]. Disponível em: https://searchmicroservices.techtarget.com/definition/SOAP-Simple-Object-Access-Protocol. [Acedido: 24-Fev-2019].

[14] «What is System Integration (SI)? - Definition from Techopedia». [Em linha]. Disponível em: https://www.techopedia.com/definition/9614/system-integration-si. [Acedido: 24-Fev-2019].

[15] J. Kress, «Enterprise Service Bus», 2013. [Em linha]. Disponível em: https://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html. [Acedido: 24-Fev-2019].

[16] Richard, «Hub-Spoke Integration And ESB Explained For Any Business Owner». .

[17] Z. Madani e N. Nematbakhsh, «A logical formal model for verification of Web Service Choreography», em *2009 12th International Conference on Computers and Information Technology*, 2009, pp. 448–453.

[18] F. Rissetto, «ESBs, o que são, do que se alimentam», *Fabrício Rissetto*. [Em linha]. Disponível em: http://www.fabriciorissetto.com/blog/ESBs/. [Acedido: 24-Fev-2019].

[19] «What is an ESB?», *MuleSoft*, 21-Mai-2015. [Em linha]. Disponível em: https://www.mulesoft.com/resources/esb/what-esb. [Acedido: 24-Fev-2019].

[20] «WSO2 | The Open Source Technology for Digital Business». [Em linha]. Disponível em: https://wso2.com/. [Acedido: 24-Fev-2019].

[21] F. Montesi e J. Weber, «Circuit Breakers, Discovery, and API Gateways in Microservices», *ArXiv160905830 Cs*, Set. 2016.

[22] V. M. Ionescu, «The analysis of the performance of RabbitMQ and ActiveMQ», em *2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER)*, 2015, pp. 132–137.

[23] X. J. Hong, H. S. Yang, e Y. H. Kim, «Performance Analysis of RESTful API and RabbitMQ for Microservice Web Application», em *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 257–259.
[24] A. Martikainen, «(PDF) Front End of Innovation in Industrial Organization», *ResearchGate*. [Em linha]. Disponível em: https://www.researchgate.net/publication/324208591_Front_End_of_Innovation_in_Industrial_Organization. [Acedido: 24-Fev-2019].
[25] «(PDF) Cross-border shipment route selection utilizing analytic hierarchy process (AHP) method», *ResearchGate*. [Em linha]. Disponível em: https://www.researchgate.net/publication/323905554_Cross-border_shipment_route_selection_utilizing_analytic_hierarchy_process_AHP_method. [Acedido: 24-Fev-2019].
[26] «MessageHeader - FHIR v4.0.0». [Em linha]. Disponível em: https://www.hl7.org/fhir/messageheader-mappings.html. [Acedido: 15-Set-2019].
[27] «Patient - FHIR v4.0.0». [Em linha]. Disponível em: https://www.hl7.org/fhir/patient-mappings.html. [Acedido: 28-Set-2019].
[28] J. Ding, «An approach for modeling and analyzing dynamic software architectures», em *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, Changsha, China, 2016, pp. 2086–2092.
[29] «O modelo C4 de documentação para Arquitetura de Software», *InfoQ*. [Em linha]. Disponível em: https://www.infoq.com/br/articles/C4-architecture-model/. [Acedido: 13-Out-2019].
[30] P. B. Kruchten, «The 4+1 View Model of architecture», *IEEE Softw.*, vol. 12, n. 6, pp. 42–50, Nov. 1995.
[31] A. Staveley, «The “4+1” View Model of Software Architecture - DZone Java». [Em linha]. Disponível em: https://dzone.com/articles/%E2%80%9C41%E2%80%9D-view-model-software. [Acedido: 19-Ago-2019].
[32] «HAPI FHIR - The Open Source FHIR API for Java». [Em linha]. Disponível em: http://hapifhir.io/. [Acedido: 29-Set-2019].

10 Anexos

10.1 Anexo A XSLT de Paciente (excerto)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:fn="http://www.w3.org/2005/xpath-functions" exclude-result-prefixes="xs fn">
    <xsl:output method="xml" encoding="UTF-8" byte-order-mark="no" indent="yes"/>
    <xsl:template match="/">
        <xsl:variable name="var26_PID_PatientIdentification" as="node()?" select="PID_PatientIdentification"/>
        <Patient xmlns="http://hl7.org/fhir">
            <resourceType>Patient</resourceType>
            <name>
                <family>
                    <xsl:for-each select="$var26_PID_PatientIdentification">
                        <xsl:variable name="var1_XPN_____Surname" as="node()" select="*:PID_5_PatientName[fn:namespace-uri() eq
                        '']/*:XPN_0_FamilyName[fn:namespace-uri()
                        eq '']/*:XPN_0_0_Surname[fn:namespace-uri() eq '']"/>
                        <xsl:attribute name="value" namespace="">
                            <xsl:choose>
                                <xsl:when test="(fn:translate(fn:string($var1_XPN_____Surname/@xsi:nil), 'true ', '1') = '1')">
                                    <xsl:sequence select="''"/>
                                </xsl:when>
                                <xsl:otherwise>
                                    <xsl:sequence select="fn:string($var1_XPN_____Surname)"/>
                                </xsl:otherwise>
                            </xsl:choose>
                        </xsl:attribute>
                    </xsl:for-each>
                </family>
                <given>
            </name>
            <telecom>
            <telecom>
            <telecom>
            <gender>
            <birthDate>
            <deceasedBoolean>
            <deceasedDateTime>
            <address>
            <maritalStatus>
            <xsl:for-each select="$var26_PID_PatientIdentification/*:PID_3_PatientIdentifierList[fn:namespace-uri() eq '']">
            <extension>
                <xsl:attribute name="url" namespace="" select="xs:string(xs:anyURI('
                http://hl7.org/fhir/StructureDefinition/patient-nationality'))"/>
                <valueCodeableConcept>
                    <coding>
                        <system>
                        <code>
                            <xsl:for-each select="$var26_PID_PatientIdentification">
                                <xsl:variable name="var24_CE_______IdentifierSt" as="node()" select=
                                "*:PID_28_Nationality[fn:namespace-uri() eq '']/*:CE_0212_0_IdentifierSt[fn:namespace-uri() eq '']"/>
                                <xsl:attribute name="value" namespace="">
                                    <xsl:choose>
                                </xsl:attribute>
                            </xsl:for-each>
                        </code>
                    </coding>
                    <xsl:for-each select="$var26_PID_PatientIdentification">
                </valueCodeableConcept>
            </extension>
        </Patient>
    </xsl:template>
</xsl:stylesheet>
```