Adapter Seven: Um Adaptador para Health Level 7 em Sistemas de Saúde

Lorena Souza Santana Lima
Instituto Federal da Bahia - Campus Salvador
Brasil - Bahia
lorena.lima@ifba.edu.br

Antonio Carlos S. Souza
Instituto Federal da Bahia - Campus Salvador
Brasil - Bahia
AC@ifba.edu.br

***Resumo*—Nas últimas décadas, houve um grande avanço na tecnologia, principalmente na área da saúde. Alguns países conseguiram acompanhar e outros não. No Brasil, por exemplo, alguns setores de saúde ainda utilizam sistemas antigos e proprietários, podendo provocar erros nas integrações de diversos outros sistemas de saúde e, consequentemente, problemas no compartilhamento de dados dos usuários. Baseado nisso, este trabalho tem como objetivo construir um adaptador que padronize o compartilhamento de dados de pacientes, utilizando o padrão Health Level Seven (HL7) na versão 2.5.1, para sistemas e equipamentos biomédicos. O HL7 está diretamente associado a interoperabilidade. Assim, foi escolhido um tipo de mensagem, ADR_A19, e realizado todos os tratamentos utilizando bibliotecas para o HL7. Mesmo sendo apenas um tipo de mensagem, os resultados mostraram que é possível alcançar a interoperabilidade. Desse modo, não é impossível utilizar o padrão HL7 nos setores de saúde sendo necessário dedicação, planejamento, estudo e que estejam abertos para mudanças.**

***Palavras-Chave*—HL7, compartilhamento de dados, interoperabilidade**

I. Introdução

A tecnologia está em constante evolução, principalmente na área da saúde. Alguns países conseguem acompanhar essa mudança e outros não, sendo importante compreender e estudar o que já tem e o que precisa melhorar. E, em conjunto a isso, tem-se o aumento no consumo e compartilhamento de dados entre sistemas. Segundo [1], diversos serviços utilizam as tecnologias de informação para agilizar processos e facilitar a vida dos atores envolvidos. Contudo, as especificidades de cada sistema pode interferir na interoperabilidade entre diferentes sistemas, aplicações, plataformas e ecossistemas.

A interoperabilidade é a capacidade de troca de informações entre infraestruturas, aplicações, sistemas que seguem um conjunto de regras e conceitos em comum [2]. No setor da saúde, este tópico vem sendo debatido e possibilitando perceber a necessidade de modernizar a infraestrutura de hospitais, clínicas, operadoras de saúde, entre outros, com objetivo de proporcionar um atendimento mais personalizado ao paciente [3]. Se essa padronização existisse, as informações dos pacientes poderiam ser acessadas de qualquer unidade de saúde, seja ela pública ou privada. Além do mais, se as informações fossem provenientes de um único banco de dados, facilitaria o acesso às funções estratégicas e resultaria em ganhos de produtividade, eliminação de duplicidades dentre outros [4].

Entre os padrões existentes no setor da saúde, o *Health Level Seven* (HL7) pode ser visto como uma solução na padronização do compartilhamento de dados dos pacientes. De acordo com [5], o HL7 é um conjunto de padrões flexíveis, diretrizes e metodologias onde vários sistemas de saúde podem se comunicar entre si, permitindo o compartilhamento e processamento de informações de maneira uniforme e consistente. Além disso, tem ainda o HL7 FHIR (*Fast Healthcare Interoperability Resources*), o padrão mais avançado e que utiliza APIs (*Application Programming Interface*) e demais terminologias aceitas internacionalmente [3].

Diante deste cenário, este projeto tem o objetivo de construir um adaptador, API (*Application Programming Interface*) REST (*Representational State Transfer*), que padronize o compartilhamento de dados de pacientes, utilizando o padrão HL7 na versão 2.5.1, para sistemas e equipamentos biomédicos.

II. Fundamentação Teórica

Nesta seção são apresentados os conceitos relacionados ao projeto.

A. Informática em Saúde

A Informática em Saúde é definida como "um campo de rápido desenvolvimento científico que lida com armazenamento, recuperação e uso da informação, dados e conhecimentos biomédicos para a resolução de problemas e tomadas de decisão" [6]. Tendo em consideração os EUA e a Europa, a informática na saúde chegou tarde ao Brasil. Os primeiros passos ocorreram na década de 70 em alguns centros universitários, como no Hospital da Universidade Federal do Rio de Janeiro (UFRJ), no Instituto do Coração e nos Hospitais das Clínicas da USP, em São Paulo e Ribeirão Preto [7]. Passou por um grande ímpeto a partir de 1983, com a criação de novos grupos para a área de pesquisa e ensino, e o divisor de águas da informática em Saúde nacional ocorreu em 1986, onde fundaram em novembro a Sociedade Brasileira de Informática em Saúde, durante o I Congresso Brasileiro de Informática em Saúde [7].

Com base em [6], o crescimento da informática em saúde ocorreu em virtude: dos avanços das Tecnologias Digitais da Informação e Comunicação (TDIC), a certeza de que o conhecimento médico e os dados dos pacientes não podem ser gerenciados pelos métodos tradicionais, também conhecido como papel, e a convicção de que o acesso ao conhecimento e a tomada de decisão é o ponto chave na Medicina moderna. Contudo, para facilitar a criação e o uso de dados de informação e de conhecimento da saúde, a interoperabilidade

é um dos quesitos essenciais, suportando e viabilizando todos os aspectos do sistema de saúde. Segundo [4], há uma grande diversidade nas variáveis internas e externas referentes aos processos saúde-doença e na administração de programas, serviços e unidades de saúde, evidenciando a necessidade da padronização dos dados.

A Organização Mundial de Saúde (OMS) reconhece que o uso das TDIC na Saúde é um componente estratégico para oferecer saúde universal com qualidade [8]. É vital a difusão do uso de novas TDIC na área da saúde para ampliar o acesso, facilitar os procedimentos, melhorar a qualidade e garantir a eficiência dos serviços prestados [9]. Porém, no setor de saúde no Brasil, ainda há muitos desafios que precisam ser vencidos. Um deles é a escassez de líderes na área da Informática em Saúde nas diferentes esferas do Sistema Único de Saúde (SUS) e da Saúde Suplementar [8]. É necessário líderes que compreendam a complexidade dos programas, a estimativa de duração dos projetos, tenham experiência e conhecimento dos processos de concepção e implementação de sistemas de informação (SI) em saúde, em ambientes complexos [8], a fim de propor estratégias que possibilite resultados positivos, a longo prazo, e que estejam de acordo com os tempos políticos.

Outro desafio maior é a falta de interoperabilidade, pois para que aconteça é dividida em várias camadas que se complementam e comunicam entre si [10], como mostra a figura 1.

Figura 1: Níveis de Interoperabilidade. Fonte: [10].

Cada camada tem uma definição:

- **Tecnológica:** tem como foco o transporte de dados entre os sistemas.
- **Semântica:** envolve a utilização de códigos e identificadores de forma que o sistema A e B se comuniquem sem ambiguidade, interpretando a informação que é transmitida da mesma forma [10].
- **Organizacional:** coordena os processos de trabalho, permitindo realizar a atividade em conjunto dos processos nas instituições de saúde.

Além do mais, a interoperabilidade tem como vantagens: eliminação de deficiências através da automatização de tarefas, maior controle e agilidade do *workflow* (fluxo de trabalho), otimização do tempo, redução de custos, retrabalhos e erros humanos, melhor experiência de atendimento ao paciente [6], garantias de escalabilidade, sustentabilidade do projeto [8], *standards* uniformizados, garantia de segurança e acessos controlados e menor redundância de informação [10]. Todavia, por não conhecer o processo de implementação da interoperabilidade, muitas organizações de saúde ainda não aplicaram. Esse processo possui três etapas e segundo [6] são: desenvolvimento e implementação do registro eletrônico de saúde (EHR), desenvolvimento e implementação de medidas de desempenho administrativos sensíveis para certificação e a definição e adoção de um padrão único de linguagem entre sistemas.

É fundamental que a comunicação entre os diversos SI obedeçam os protocolos *standard* para alcançar a interoperabilidade, e dentre eles tem-se: *Digital Imaging and Communications in Medicine* (DICOM), ASC X12, Troca de Informação em Saúde Suplementar (TISS) e HL7. Desses, o destaque vai para o HL7 por ser adotado por grandes sistemas, está em constante evolução, é considerada a mais adaptável em interoperabilidade na saúde e por possuir fortes potencialidades relacionados a problemas de distribuição e comunicação [9].

B. HL7

O HL7 é um padrão internacional utilizado no setor da saúde. Foi desenvolvido e gerenciado pela organização *Health Level Seven International*, fundada em 1987. Um grupo sem fins lucrativos [11] e dedicado ao desenvolvimento de normas para troca de informação hospitalar [9], no qual está presente em mais de 55 países. Nos Estados Unidos (EUA), mais de 90% das unidades de saúde adotaram este padrão [12] que é formado por um conjunto de normas do HL7, produzindo protocolos de transmissão de mensagens entre os equipamentos, base de dados e sistemas de gestão médicos. Essas normas ficam centralizadas na camada de aplicação, denominada de "camada 7", no modelo OSI (*Open Systems Interconnect*) de comunicação entre computadores adaptados para o sistema [13].

A criação deste padrão, teve o intuito de favorecer o compartilhamento de dados clínicos e administrativos, de forma confiável, entre os diversos SI em saúde [12] sem precisar especificar os detalhes técnicos. A estrutura e o *design* das mensagens possibilita que apenas dados específicos sejam enviados, permitindo e potenciando a aplicação do HL7 numa arquitetura cliente-servidor [9]. Algumas formas de transferências de dados como *e-mail*, *downloads*, FTP (Protocolo de Transferência de Arquivo), são considerados insuficientes pelo setor da saúde devido a individualidade do usuário e da unidade de atenção, além da necessidade de contexto da informação tornar a troca de dados um grande desafio [14].

O HL7 está relacionado a interoperabilidade e no Brasil, essa questão ainda está sendo discutida, pois o grau de maturidade para a interoperabilidade ainda é baixo e nem todas as unidades de saúde são informatizadas ou realizam o correto cadastramento de dados [3]. É necessário o alinhamento entre os profissionais de saúde, provedores, instituições e gestores, para que esses sistemas padronizados seja implantados.

Os objetivos do padrão HL7, de acordo com [10] são:

- Ser o mais abrangente possível;
- Ser flexível;
- Uma norma aberta;
- Fornecer formatos e protocolos na comunicação entre aplicações;
- Permitir a integração de diversas aplicações num SI;
- Fornecer um meio de atingir a interoperabilidade;

- Com a integração dos dados, melhorar o processo de decisão clínica;
- *Workflows* otimizados;
- Reduzir conflitos e problemas de comunicação na transmissão de mensagens;

O desenvolvimento do HL7 foi fundamentado na tese que a cada ação que ocorra, referente aos dados na saúde, é gerado um *trigger event*, um evento que permite o envio da informação entre os sistemas. Quando fala em ação, está associado a criação ou atualização de um dado relacionado ao paciente inserida na mensagem no formato HL7.

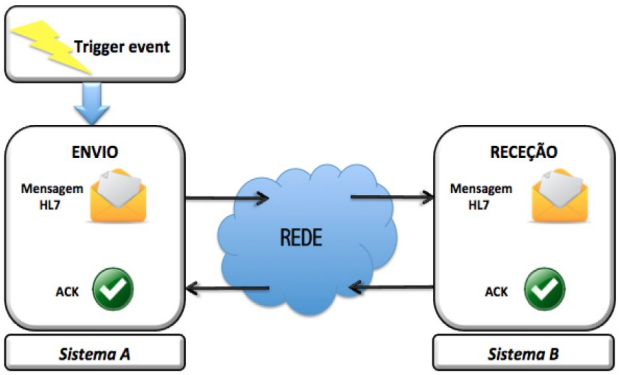


Figura 2: Troca de informação através de mensagens HL7, entre sistemas, gerado por um *trigger event*. Fonte: [9].

O padrão HL7 é divida em duas categorias:

1) Versão 2

Versão mais utilizada e compatível com qualquer versão 2.x anterior [11]. Possui um formato plano e é composta por campos de tamanhos variados com os seguintes elementos:

- **Delimitadores:** separam os campos de informação [15].

Delimitador	ASCII	Descrição
<CR>	13	Segmento
|	124	Separador (*Pipes*)
^	94	Componente
~	126	Repetição
&	38	Subcomponente
\	92	Carácter de Escape

Tabela I: Delimitadores. Fonte: [15], [16].

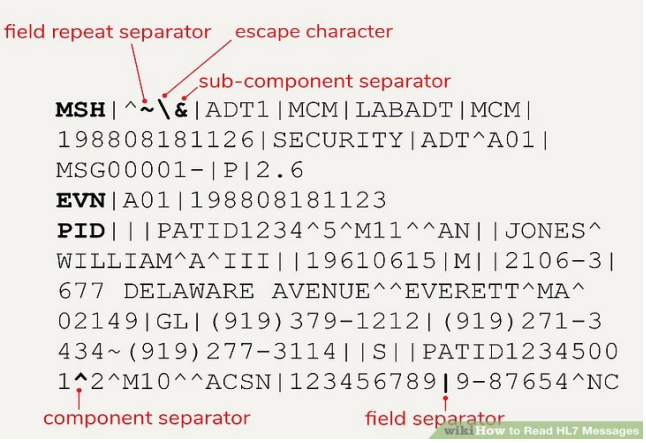


Figura 3: Delimitadores na mensagem HL7 versão 2.x. Fonte: [17].

- **Segmentos:** código de três caracteres que define a sua função [18].

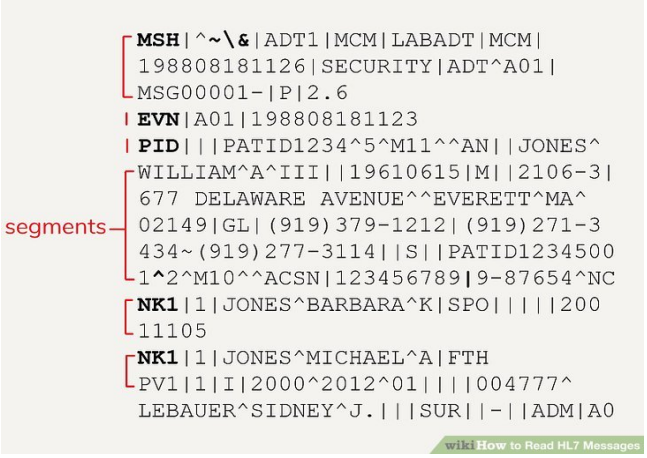


Figura 4: Segmentos na mensagem HL7 versão 2.x. Fonte: [17].

- **Campos:** fazem parte do segmento e cada um deles contém um tipo de dado [15].

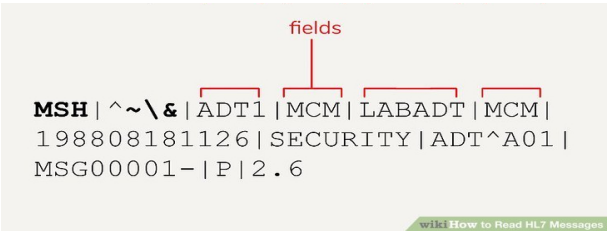


Figura 5: Campos de um segmento na mensagem HL7 versão 2.x. Fonte: [17].

- **Tipos de Dados:** simples (contém apenas um valor único) ou complexos (podem conter mais de um subelemento) [15].

Cada segmento contém um grupo de campos com diferentes tipos de dados que, como são independentes, podem ser utilizados em várias outras mensagens em sequências diferentes [11]. É obrigatório que todas as mensagens comecem com o segmento "MSH", onde contém a informação sobre o tipo da mensagem.

Segmento	Descrição
MSH	*Message header*
PID	*Patient identity*
PV1	*Patient visit information*
NK1	*Patient's Next of Kin*
EVN	*Event type*
OBX	*Observation/result*
AL1	*Allergy information*
DG1	*Diagnosis information*
DRG	*Diagnosis related group*
PR1	*Procedures*
ROL	*Role*
IN1	*Insurance*
ACC	*Accident information*
PDA	*Patient death and autopsy*

Tabela II: Exemplos de segmentos. Fonte: [19].

O tipo da mensagem determina quais segmentos a formaram, sendo que, para cada um existe diferentes subtipos.

Por exemplo, uma mensagem pode ser do tipo **ADT** com o subtipo **A01**, sendo registrada como **ADT_A01**. O exemplo da figura 6, criada na linguagem JAVA, mostra a estrutura da mensagem **ADR_A19**, usada neste projeto.

```
ADR_A19 (start)
   MSH - MSH|^~\&|Simulador Multiparamétrico||||20210821||ADR^A19|||2.5.1||||AL
   [ { SFT } ] - Not populated
   MSA - Not populated
   [ ERR ] - Not populated
   [ QAK ] - Not populated
   QRD - Not populated
   [ QRF ] - Not populated
   QUERY_RESPONSE (start)
   {
      [ EVN ] - Not populated
      PID - PID|||456879||Del Bairro^Maria||20000801|F||||||||||78984696251
      [ { OBR } ] (non-standard) - OBR|1|||^Sinais Vitais^LN
      [ PD1 ] - Not populated
      [ { ROL } ] - Not populated
      [ { NK1 } ] - Not populated
      PV1 - Not populated
      [ PV2 ] - Not populated
      [ { ROL2 } ] - Not populated
      [ { DB1 } ] - Not populated
      [ { OBX } ] - OBX|1|NM|PESO CORPORAL||59|Kg|||||F
                    OBX|2|NM|DOR||1||0-10||||F
      [ { AL1 } ] - Not populated
      [ { DG1 } ] - DG1||||Asma
      [ DRG ] - Not populated
      PROCEDURE (start)
      [{
         PR1 - Not populated
         [ { ROL } ] - Not populated
      }]
      PROCEDURE (end)
      [ { GT1 } ] - Not populated
      INSURANCE (start)
      [{
         IN1 - Not populated
         [ IN2 ] - Not populated
         [ { IN3 } ] - Not populated
         [ { ROL } ] - Not populated
      }]
      INSURANCE (end)
      [ ACC ] - Not populated
      [ UB1 ] - Not populated
      [ UB2 ] - Not populated
   }
   QUERY_RESPONSE (end)
   [ DSC ] - Not populated
ADR_A19 (end)
```

Figura 6: Segmentos permitidos na mensagem ADR-A19. Fonte: Figura do Autor

Na mensagem da figura 6, os colchetes ([ ]) indicam que um segmento ou o grupo de segmentos é opcional, enquanto que as chaves ({ }) indicam a repetição do segmento ou grupo de segmentos [20].

Tipo da Mensagem	Descrição
ADT	*Adimission Transfer Discharge*
ORM	*Order (Pharmacy/Treatmet)*
ORU	*Observation Result*
BAR	*(Add/Change) Billing Account*
ACK	*General Acknowledgement*
DFT	*Detailed Financial Transaction*
MDM	*Medical Document Management*
MFN	*Master Files Notification*
RAS	*Pharmacy/ Treatment Administration*
RDE	*Pharmacy/ Treatment Encoded Order*
RGV	*Pharmacy/ Treatment Give*

Tabela III: Exemplos de tipos das mensagens. Fonte: [19].

Como pode perceber na tabela III, cada tipo está relacionado a uma descrição ou tarefa. Assim, tomando como base os tipos citados anteriormente, o **ADT_A01** é relativo a admissão do paciente e o **ADR_A19** a uma consulta de um paciente. Segundo [10], o HL7 suporta as seguintes tarefas:

- Admissão, Transferência, Alta;
- Pedidos de Laboratório;
- Pedidos de Radiologia;
- Gestão de Ordem;
- Finanças;
- Observação;

Mesmo que as regras de compatibilidade entre as versões ofereçam suporte à evolução das interfaces, é necessário que as versões subsequentes não incluam estruturas que invalidem as versões anteriores [20].

2) Versão 3

Escrita em XML, tem um formato totalmente diferente e não compatível com versões anteriores, como a V2 [11], sendo impossível a comunicação se não houver as modificações consideráveis. Logo, [11] ainda acredita que não seja implementada devido aos investimentos já realizados nas versões 2.x, na sua dificuldade de implementação, nos altos custos e na próxima proposta do mais recente padrão da HL7, o FHIR.

```
<?xml version="1.0" encoding="UTF-8"?>
<PRPA_IN101001UV01 ITSVersion="XML_1.0" xmlns="urn:hl7-org:v3"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id extension="3948375" root="2.16.840.1.113883.19.10.700363.2288"/>
  <creationTime value="20060501140010"/>
  <versionCode code="NE2006"/>
  <!-- Interaction is a notification of a person registration -->
  <interactionId extension="PRPA_IN101001UV01" root="2.16.840.1.113883.1.6"/>
  <processingCode code="P"/>
  <processingModeCode code="T"/>
  <acceptAckCode code="ER"/>
  <receiver>
    <device>
    <id extension="922" root="2.16.840.1.113883.19.9"/>
    <name>Master MPI</name>
      <asAgent>
        <representedOrganization>
          <id extension="1002003" root="2.16.840.1.113883.19.200"/>
          <name>Alpha Hospital</name>
        </representedOrganization>
      </asAgent>
    </device>
  </receiver>
  <sender>
    <device>
      <id extension="1" root="2.16.840.1.113883.19.9"/>
    </device>
```

Figura 7: Estrutura da mensagem HL7 na versão 3. Fonte: [11].

C. Equipamentos Biomédicos

Os equipamentos biomédicos são essenciais para o ser humano. Eles podem ser usados para auxiliar os profissionais da saúde no diagnóstico, tratamento, monitoração, suporte à vida, entre outros. Em relação à sua complexidade, pode ser:

- **Baixa**: não requer recursos humanos especializados [21]. Exemplo: esfigmomanômetro.
- **Média**: profissionais precisam ter uma formação e treinamento adequado para garantir a segurança pessoal e do paciente [21]. Exemplo: monitor multiparamétrico.
- **Alta**: exige profissionais altamente qualificados e com treinamento técnico de alto nível especializado [21]. Exemplo: tomógrafo.

A alta relevância das tecnologias na saúde, tem contribuído para a sua complexidade que, consequentemente, vem acompanhado de consequências podendo afetar a segurança do paciente devido ao aumento dos riscos de erros humanos. Um desses erros está associado as características do equipamento, como o *design* da interface. Sendo assim, os

equipamentos devem ser bem projetados e de boa qualidade para assegurar a segurança e atendimento apropriado aos pacientes [21].

1) Monitor Multiparamétrico

Equipamento de média complexidade, responsável por monitorar as seguintes variáveis: frequência cardíaca com traçado de eletrocardiograma, saturação de oxigênio no sangue (SpO2), capnografia (a depender da marcar e modelo) referente a concentração ou pressão parcial de dióxido de carbono (CO2) nos gases respiratórios, pressão arterial (não invasiva ou invasiva), temperatura e frequência respiratória [21]. Como são captadas em tempo real, permite uma resposta rápida ao tratamento e novas intervenções, caso o paciente precise.

No ambiente hospitalar, é utilizado nos setores de emergências, ambulatórios e pronto-socorro. Ademais, também tem sido utilizado em outros ambientes como academias, clínicas especializadas, atendimento *homecare* e centro de fisioterapia. Os valores das variáveis são apresentados na telas, além possibilitar a configuração de alarmes.

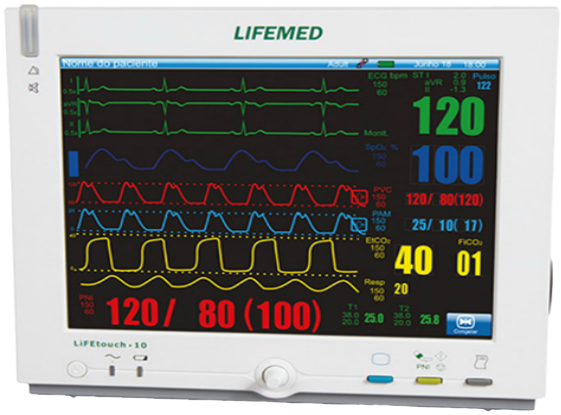


Figura 8: Monitor Multiparamétrico. Fonte: [22]

III. Trabalhos Relacionados

Nesta seção constam alguns trabalhos acadêmicos que abordam temas similares ao estudo proposto. Cinco projetos foram analisados, abordando temas referentes aos sinais vitais, utilização do padrão HL7, validação e transmissão das mensagens em HL7, assim como desenvolvimento de aplicações para legitimar os resultados.

[23] não utilizou o padrão HL7 no seu estudo, mas focou na questão da monitoração dos sinais vitais. Foi desenvolvido um modelo de monitoração de pacientes em leitos de Unidade de Terapia Intensiva (UTI), utilizando o sistema embarcado Micro Servidor Web (MSW). Este capturava, armazenava e disponibilizava, via sistema *web* (*World Wide Web*), os sinais vitais adquiridos por monitores padronizados conectados aos pacientes por meios de sensores. Dos cinco sinais vitais escolhidos, foram utilizados 3: temperatura corporal, frequência cardíaca e saturação. A pressão arterial sistólica e diastólica não foram possíveis de serem obtidos pela falta de equipamentos. A aquisição e distribuição dos dados foi amparada na arquitetura cliente/servidor e no protocolo de comunicação de dados HTTP (*HyperText Transfer Protocol*), oferecendo agilidade e segurança das informações. Ao final, o autor obteve resultados positivos, pois conseguiu monitorar remotamente o paciente, além de criar um modelo de baixo custo.

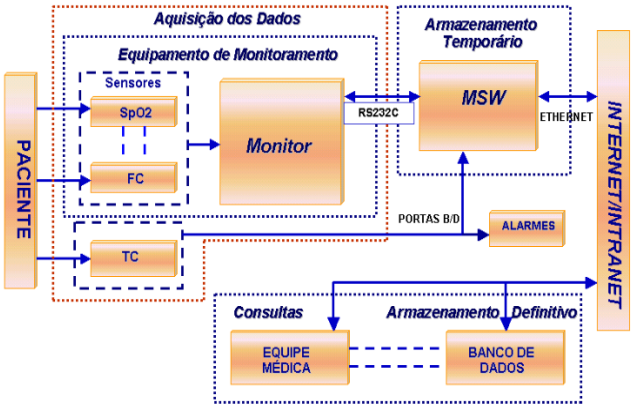


Figura 9: Modelo de Implementação do estudo. Fonte: [23]

No relatório para Mestrado, [16] descreve o projeto realizado na empresa *ALERT Life Sciences Computing* S.A. ao longo de 20 semanas. O projeto consistia em desenvolver uma aplicação que permitisse a validação e interpretação de mensagens HL7 versão 2, permitindo a troca de informação clínica entre a aplicação e outros SI. Foi utilizado a linguagem JAVA e, para simular o ciclo, criou-se o cliente e servidor. A validação da mensagem ocorreu em duas etapas: primeiro verificou se a mensagem era composta por um conjunto válido de segmentos, baseada no seu tipo, e a segunda analisou a estrutura e conteúdo de cada segmento. O autor teve seus objetivos atingidos e superados, pois foram criadas algumas ferramentas extras e o seu trabalho seria colocado em funcionamento em várias instituições de vários continentes.

Segundo [24], o fato de instituições de saúde possuírem sistemas legados e proprietários pode explicar a baixa interoperabilidade no Brasil, atrapalhando a integração dos sistemas. Baseado nisso, o mesmo implementou um Sistema de Prontuário Eletrônico do Paciente (PEP) com o objetivo de permitir a inserção, edição, visualização e exportação dos dados do prontuário em um padrão de transmissão de informações médicas. Foi criado um protótipo PEP, na forma de uma aplicação *web* JEE, e um serviço que consultaria às informações do PEP utilizando a ferramenta HAPI, que é uma API para programar aplicações com suporte à troca de mensagens em HL7. Após testes com um conjunto mínimo de informações, o autor obteve resultados positivos.

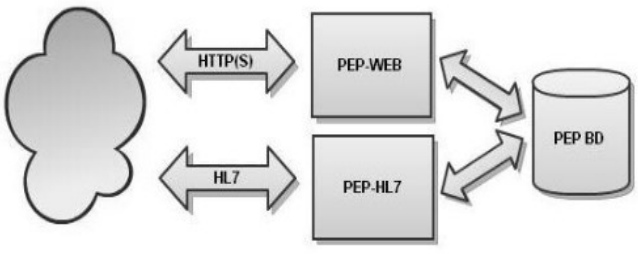


Figura 10: Visão geral do sistema. Fonte: [24].

A integração entre os SI de uma unidade é um assunto que merece atenção, pois caso ocorra algum erro pode gerar sérios problema no compartilhamento dos dados. O estudo [10] aborda esse quesito. Foi feita uma análise do estado das integrações entre os SI de hospitais Portugueses e os autores citaram que o elevado número de SI, a especificidade e a utilidade de melhoria da prestação de serviços de saúde, origina um problema sério quando existem erros nas integrações entre os mesmos. Para validar o estudo, investigaram os SI dos departamentos de análises laboratoriais e radiologia e suas integrações com o SI central do hospital. As mensagens trocadas pelos SI seguiam o padrão HL7, versão 2.4, e foram observados segmentos específicos (MSH, PID, PVI, OBR, ORC e MSA). Os resultados mostraram que a qualidade da integração é semelhante nos dois hospitais, porém um pouco pior no departamento de radiologia e, especificamente, nos segmentos OBR e ORC, concluindo assim que a utilização de determinados campos dos segmentos nas mensagens variam com o departamento e instituição de saúde onde é efetuada a análise, tendo que com o passar do tempo alterar as interfaces clínicas e transmitir novos dados.

O estudo [25] seguiu a mesma abordagem que o estudo anterior, fez uma análise dos estados das integrações de instituições hospitalares focando nas mensagens. Foi usado um mecanismo de integração para receber, processar e reencaminhar mensagens HL7 que passavam por uma validação para analisar sua qualidade. Das 1.048.576 mensagens recolhidas e processadas, identificaram nomes de usuários indevidos e erros sistemáticos causados por alguns SI, concluindo que o padrão HL7 necessita ser usado corretamente e com menor número de erros possível.

IV. Solução

Esta seção discorre sobre o desenvolvimento da solução, distribuída nos subseções de: etapas de desenvolvimento , as tecnologias utilizadas, a visão geral do sistema, modelagem, api e segurança de dados.

A. Etapas de desenvolvimento

O desenvolvimento da solução seguiu as seguintes etapas:

1) Escolha das tecnologias;
2) Teste das bibliotecas que oferecem suporte para o padrão HL7;
3) Modelagem do Banco de Dados;
4) Construção da API;
5) Modelagem das aplicações;
6) Construção do simulador;
7) Construção da aplicação *web*;

B. Tecnologias

A escolha das tecnologias foi fundamentada em: o quanto de conhecimento possuía sobre as mesmas, disponibilidade de documentação aliadas à facilidade de aprendizagem e quais ofereciam melhor suporte e consistência nos dados durante a manipulação das mensagens no formato HL7. Elas foram separadas em linguagens de programação, *frameworks*, *softwares*, banco de dados e bibliotecas.

Dentre as linguagens de programação, foram utilizadas: **JAVA**, **Javascript** e **Typescript**. Cada uma possui suas características. O JAVA é uma linguagem orientada a objetos, funciona independente da plataforma, é tipada, imperativa e o bom entendimento das suas regras a torna simples. Diferente desta, o Javascript é uma linguagem interpretada, conhecida como linguagem de *script* para páginas *web*. Além disso, pode ser utilizada em ambientes sem relação com o *browser* para construções de API's. O Typescript não diferente muito do Javascript, pois foi de onde se originou. Na hora de compilar o código, o Typescript é convertido/transpilado para o Javascript [26], e isso ocorre porque *browser* não compreender outra linguagem que não seja Javascript. Sua principal vantagem é a tipagem forte, servindo como validação durante a compilação, além da orientação a objetos.

Dos *frameworks*, utilizou-se o **Express.js**, criado para otimizar a construção de aplicações *web* e API's com o **Node.js**. Este último é *software* que permite a execução de código Javascript a nível de *front-end* e *back-end* [27], além de possibilitar uma melhor escalabilidade, desempenho aprimorado e custos otimizados, sendo utilizado por grandes empresas como a NASA [28].

A fim de facilitar os testes dos serviços RESTful da API mediante requisições HTTP, simples e complexas, foi utilizado o *software* **Postman**, principalmente por possibilitar a geração da documentação da API. E para permitir que os dados dessas requisições fossem salvos, foi usado o banco de dados **PostgresSQL**. Este é um sistema que atua como um gerenciamento de banco de dados relacionais. Permite a utilização da linguagem SQL (*Structured Query Language*) e é muito aplicado devido a sua praticidade e compatibilidade com diferentes padrões de linguagem [29]. Funciona bem para aplicações que tenha intensidade de acessos. Os usuários podem realizar consultas de forma simples, sem precisar acessar o banco de dados [29]. Apresenta risco reduzido se houver necessidade de mover o banco de dados local para a nuvem, ou da nuvem para o local, ou mesmo de uma plataforma de nuvem para outra [30]

Por fim, como biblioteca foi empregado o **ReactJS**, construída em Javascript para criação de interfaces de usuário [31]. É baseada em componentes, onde cada um administra seu estado e pode ser combinado com outros para criação de UI's (*User Interface*) mais complexas. É declarativo, fazendo com que o código fique mais simples de ser depurado, e a atualização e renderização ocorre apenas nos componentes em que houver mudanças.

C. Visão Geral do Sistema

A figura 11 apresenta a Visão Geral do Sistema, consistindo em:

- Um simulador para inserir, atualizar e consultar dados de pacientes e sinais vitais.
- Um aplicação *web* para a visualização dos dados dos pacientes inseridos no simulador, além de possibilitar a pesquisa individual.
- Uma API que tem a função de intermediar a comunicação entre as aplicações e o Banco de Dados.
- Um Banco de Dados que armazena os dados que vem das aplicações.

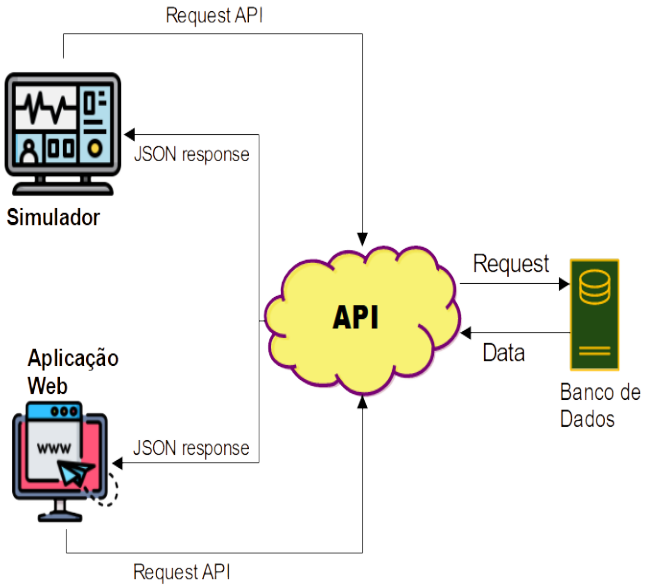


Figura 11: Visão geral do Sistema. Fonte: Figura do autor

D. Modelagem

Nesta subseção consta casos de usos referente ao simulador e o usuário, no caso o profissional, além das tabelas utilizadas no banco de dados.

1) Casos de Uso

O sistema possui dois atores:

- Simulador: envia dados do simulador e do paciente. Esse envio está relacionado ao seu registro na base de dados, pois a partir disso é gerado um *token* para possibilitar permissões ao acesso a API. Já o envio dos dados do paciente, é no formato HL7 em *string* num objeto JSON (*JavaScript Object Notation*).

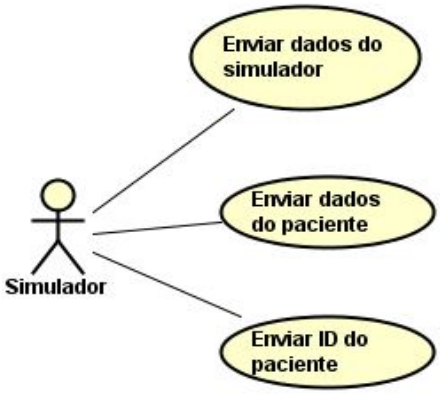


Figura 12: Diagrama de Caso de Uso do Simulador. Fonte: Figura do Autor

- Profissional: pode cadastrar seus dados, fazer login, alterar senha e consultar pacientes. Após ter feito o cadastro, tem acesso a todos os sinais vitais dos pacientes cadastrados no simulador, além de poder consultar um paciente específico pelo nome. Contudo, isso só é possível após o login, pois é aplicado a mesma ação do simulador referente ao *token*. E se esquecer a senha, é permitido altera-lá assim como o *e-mail*.

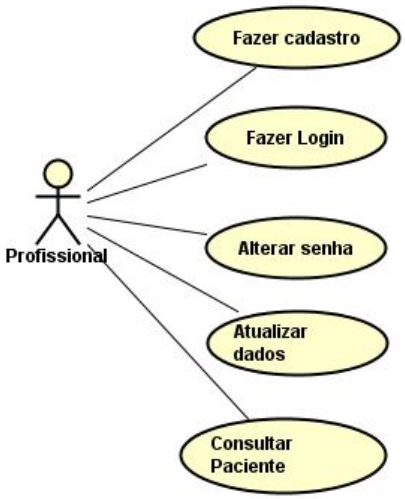


Figura 13: Diagrama de Caso de Uso do Profissional. Fonte: Figura do autor.

2) Banco de Dados

Foram definidas 8 tabelas:

- *users*: armazena dados dos usuários e se relaciona com a tabela *users_roles*. A nível didático, foram escolhidos poucos atributos, sendo que pode ser acrescentado outros. Os atributos dessa entidade são: *id, username, name, email* e *password*.
- *roles*: armazena os papéis dos usuários e se relaciona com a tabela *users_roles*. Foi escolhido 2 papéis para serem atribuídos aos usuários: *admin* e *users*. Cada papel concede permissões diferenciadas aos dados. Os atributos dessa entidade são: *id* e *name*.
- *users_roles*: originada pelo relacionamento entre as tabelas *users* e *roles*. Armazena apenas os *id's* das tabelas e seus atributos são: *user_id* e *role_id*.

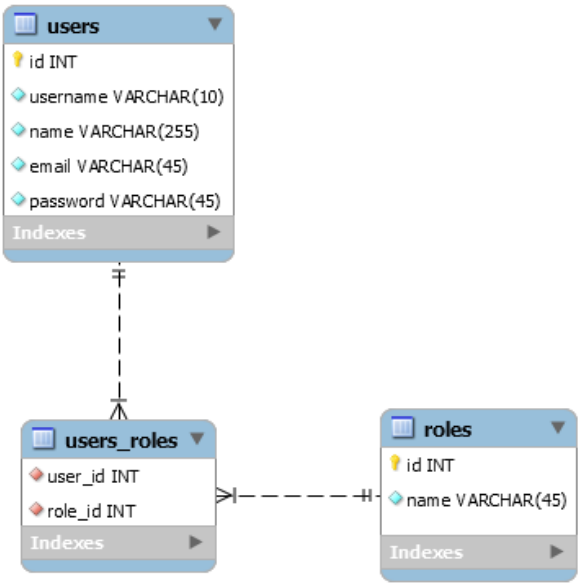


Figura 14: Tabelas referente aos usuários. Fonte: Figura do autor.

- *equipaments*: armazena dados dos simuladores. Foi designado para cada simulador um número de séries, de 6

dígitos, gerado randomicamente. Possui como atributos: *id, n_series, email* e *password*.

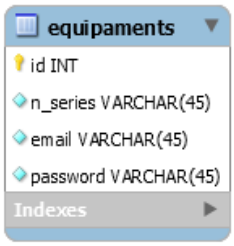


Figura 15: Tabela de equipamentos. Fonte: Figura do autor.

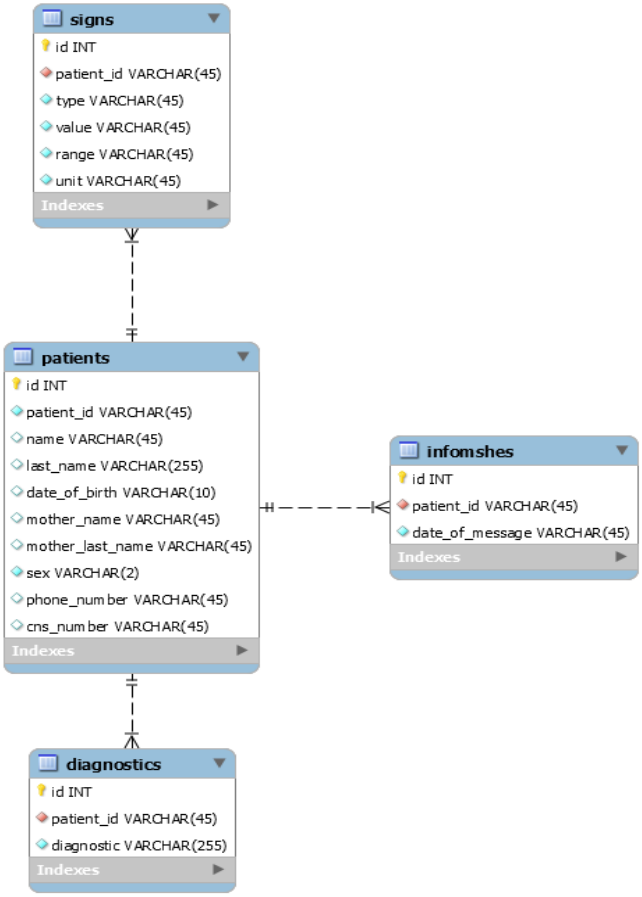


Figura 16: Tabelas referente aos pacientes. Fonte: Figura do autor.

- *signs*: armazena dados relacionados aos sinais vitais dos pacientes e se relaciona com a tabela *patients*. Possui como atributos: *id, patient_id, type, value, range* e *unit*.
- *patients*: armazena dados de identificação dos pacientes. Os atributos desta entidade são: *id, patient_id, name, last_name, data_of_birth, mother_name, mother_last_name, sex, phone_number* e *cns_number*.
- *infomshes*: armazena o dados relativo a data de registro do paciente. Este dado encontra-se no segmento MSH da mensagem no formato HL7. Se relaciona com a tabela *patients* e possui como atributos: *id, patient_id* e *date_of_message*.
- *diagnostics*: armazena dados dos diagnósticos dos pacientes, logo se relaciona com a tabela *patients*. Possui como atributos: *id, patient_id* e *diagnostic*.

A tabela *equipaments* não se relaciona com nenhuma outra, pois não houve necessidade durante o desenvolvimento do projeto. Na parte dos usuários, foi aplicado a normalização com a criação da tabela *users_roles*. Na tabelas para os dados dos pacientes, também preferiu-se separar de acordo com os dados específicos. Ambas interferências, tanto na parte de usuários como na de pacientes, foram aplicadas com o objetivo de manter a organização, garantir a flexibilidade, eliminar a redundância e dependência inconsistente.

E. API

Desenvolvido com a finalidade de ser o adaptador que intercede a comunicação entre as aplicações, simulador e *web*, e o banco de dados por meio de requisições HTTP, enviando os dados em objetos JSON. Foi desenvolvido na linguagem Javascript, com o *software* Node.js e o *framework* Express.js. Através de configurações foi conectado com o banco de dados PostgresSQL, e a documentação foi gerada utilizando o Postman.

Por ser uma solução que aborda um tipo de mensagem HL7 e tomando como base a modelagem do banco de dados, preferiu-se utilizar uma arquitetura do tipo Monolítica por ser mais simples de desenvolver, testar, fazer *deploy* e de escalar. Neste tipo, todas as funções do negócio estão implementadas em um único processo [32]. Além disso, no código foi aplicado o padrão MVC (*Model View Controller*) com o intuito de separar a parte lógica da física, permitindo fazer alterações de forma independente [33].

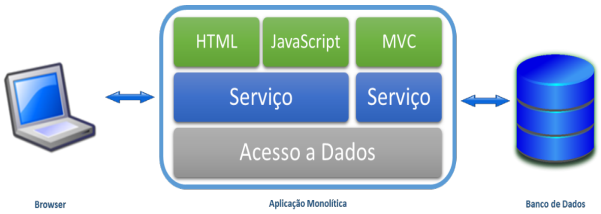


Figura 17: Aplicação Monolítica em comunicação com o *Browser* e Banco de Dados. Fonte: [32]

Como o simulador, abordado na seção de Avaliação das Ferramentas, envia e recebe mensagens no formato HL7, a API é capaz de tratar essas mensagens, tanto desagrupando os dados e salvando no banco de dados, como agrupando para ser enviado em formato HL7. Para isso, foram criados *endpoints* específicos, rotas pelo qual o serviço é acessado por uma aplicação cliente [34], para enviar e receber dados em HL7.

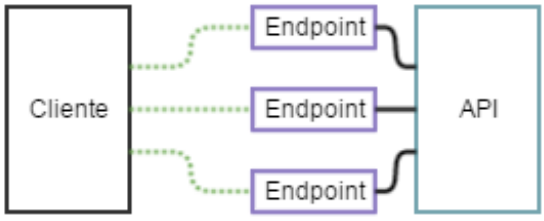


Figura 18: Cliente fazendo uso dos *endpoints* para acessar a API. Fonte: [34]

F. Segurança de Dados

A Segurança dos Dados é um assunto que vem sendo muito discutido nos últimos anos, principalmente pelo fato de ter ocorrido grandes vazamentos de dados e ataques cibernéticos. Considerando que neste projeto há manipulação de dados confidencias, foi aplicado na API o JSON *Web Token* (JWT) e adicionada configurações ao CORS (*Cross-Origin Resource Sharing*), como formas de garantir a segurança.

O JWT é um padrão aberto que define uma forma consistente e independente para transmitir informações com segurança, entre as partes, como um objeto JSON [35]. Ele pode ser assinado usando um segredo ou um par de chaves pública/privada [35]. Para este projeto, os *tokens* gerados pela API, no momento do *login*, são assinados por uma chave que se encontra em um arquivo chamado .env, onde é atribuído a uma variável e importada para as funções relacionadas a criação do *token*. Além dessa chave, é adicionado um tempo de expiração de 1 dia. Logo, se passar desse tempo é feito um *refresh token*, gerando um novo *token*.

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5
        cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6
        ImNhZGVuUGFnYWMiLCJuYW1lIjoiQ2
        FkZW4gUGFnYWMgUG90dGVyIiwiZW1
        haWwiOiJjYWRlbi5wYWdhY0
        BldGhlcmVhbC5lbWFpbCIsInJvbGUiOiJ
        1c2VycyIsImlhdCI6
        MTYzNjUwMjMyNywiZXhwIjoxNjM2NTg4
        NzI3fQ.e3yUD2PczsjbnviwJSgzJyD3Ji
        7Sz0i4sREkHiR5Usk"
}
```

Listing 1: *Token* gerado pela API.

Figura 19: *Token* colorido por partes no site [35]. Fonte: [35].

Conforme a figura 19, o *token* possui 3 partes, separados por pontos, que são:

- ***Header***: indica qual tipo e algoritmo de assinatura é usado.

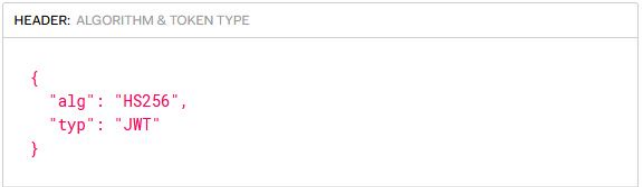


Figura 20: *Header*. Fonte: [35]

- ***Payload***: pode conter informações sobre os usuários e outros dados adicionais.

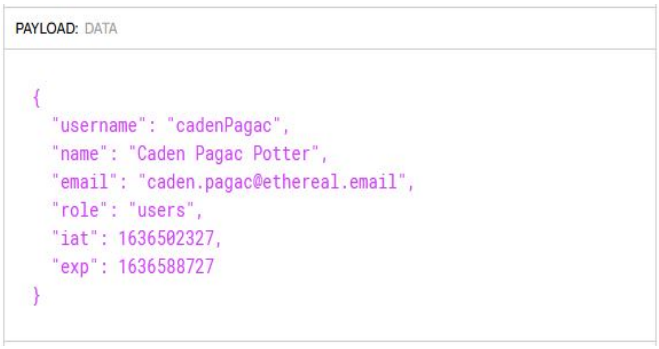


Figura 21: *Payload*. Fonte: [35]

- ***Signature***: é a assinatura de fato e, para cria-lá, precisa do *header*, *payload* e o segredo, que é a chave.

Figura 22: *Signature*. Fonte: [35]

Esse *token* é utilizado para acessar as rotas internas da API, pois somente os usuários autenticados podem ter acesso aos dados. Então, tudo que estiver relacionado a manipulação dos dados do paciente e atualização e exclusão de dados pessoais, é necessário passar o *token* no *header* durante a requisição. Além disso, ele é decodificado para pegar informações, do usuário ou simulador, na parte do *payload* e disponibiliza-las para uso na parte de atualização dos próprios dados.

A outra forma adotada, o CORS, é um modo de usar cabeçalhos adicionais HTTP a fim de conceder permissões a uma origem (domínio), como exemplo uma URL(*Uniform Resource Locator*) de uma aplicação *Web*, para acessar dados de um servidor em outra origem [36]. Se uma origem distinta tentar acessar o servidor, é bloqueado pelo CORS e nenhum dado é disponibilizado. Além do mais, para os métodos de requisição HTTP que conseguem provocar efeitos colaterais nos dados do servidor, a especificação impõe que navegadores "pré-enviem"a requisição, também conhecida como *preflight*, requisitando os métodos suportados pelo servidor e, após a "aprovação", o servidor envia a requisição verdadeira, com o método de requisição HTTP correto [36].

Na figura 23, ilustra o funcionamento do CORS entre a aplicação *web* e a API, onde foi adicionado ao *Access-Control-Allow-Origin*, na API, as origens das aplicações desenvolvidas. Quando a *web* faz uma solicitação ao servidor, ocorre o *preflight* para comprovar que está na "lista"de permissões para ter acesso aos dados. Após confirmação, finalizado o fluxo do *preflight*, ocorre a solicitação real e os dados retornados.

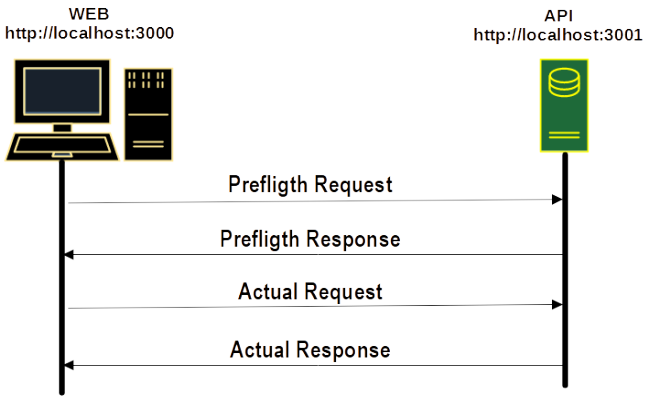


Figura 23: Funcionamento do CORS. Fonte: Figura do autor.

Se outra origem tentar acessar o servidor, o CORS bloqueia como pode ser visto na figura 24. A origem foi trocada por **http://localhost:3004**, a fim de simular o bloqueio no momento do login do usuário

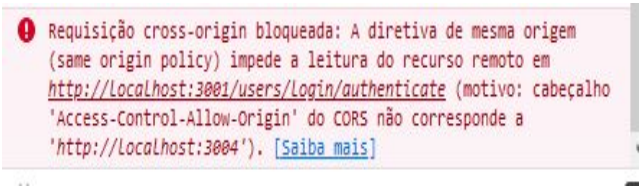


Figura 24: Simulação do bloqueio do CORS. Fonte: Figura do autor.

V. Avaliação das Ferramentas

Nesta seção, são apresentadas as aplicações do simulador, da *web* e a utilizada para a validação das mensagens no formato HL7.

A. Simulador

O Simulador foi construído tendo como base o monitor multiparamétrico, com o propósito de representar o equipamento biomédico. Assim, além dos sinais vitais que são monitorados pelo monitor, relatados na subseção do Monitor Multiparamétrico, foram adicionados a temperatura corporal, em *fahrenheit*, e a dor. E, a fim de possuir mais dados, para incluir nos segmentos da mensagem em HL7, foi incorporado alguns dados do paciente e diagnóstico.

Está aplicação foi desenvolvida na linguagem JAVA e no código também foi aplicado a arquitetura MVC, com a mesma finalidade da API. Como precisa de autorização para manipular dos dados do paciente, é necessário criar uma conta e fazer *login*, pois apenas assim o *token* é enviado pela API.

Os dados dos pacientes são enviados e recebidos, apenas, no formato HL7, como *string* no objeto JSON. Para que essas mensagens fossem tratadas, foram instalados pacotes para o padrão HL7 na versão 2.5.1. A seguir, segue as interfaces do simulador.

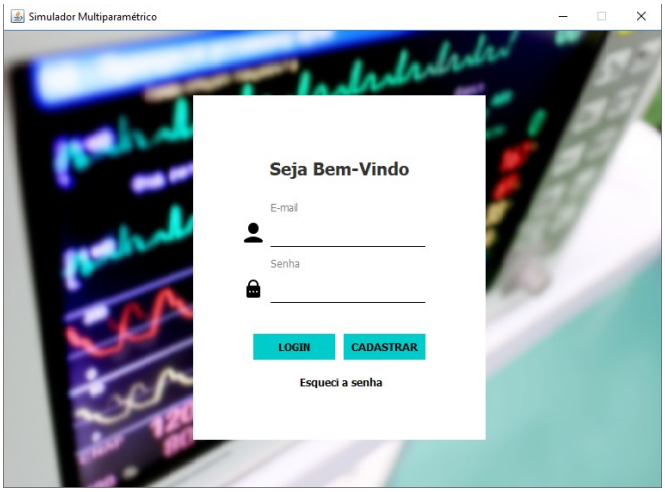


Figura 25: Tela de login. Fonte: Figura do autor.

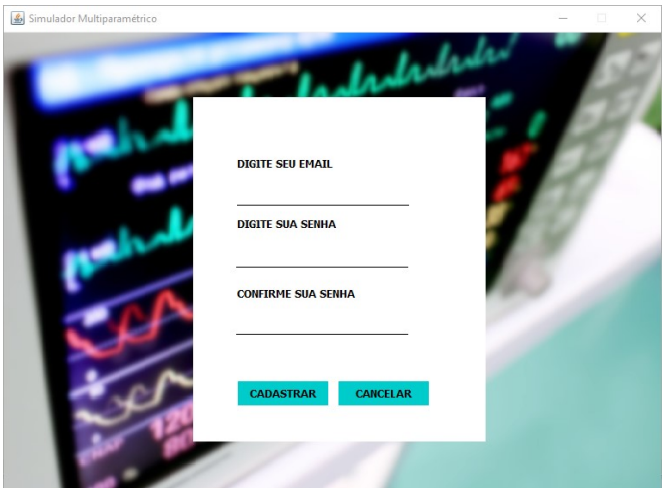


Figura 26: Tela de registro dos dados. Fonte: Figura do autor.

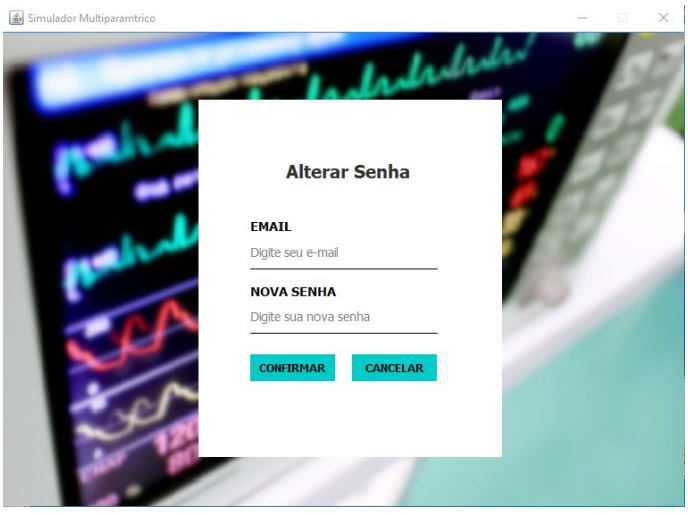


Figura 27: Tela de esqueci a senha. Fonte: Figura do autor.

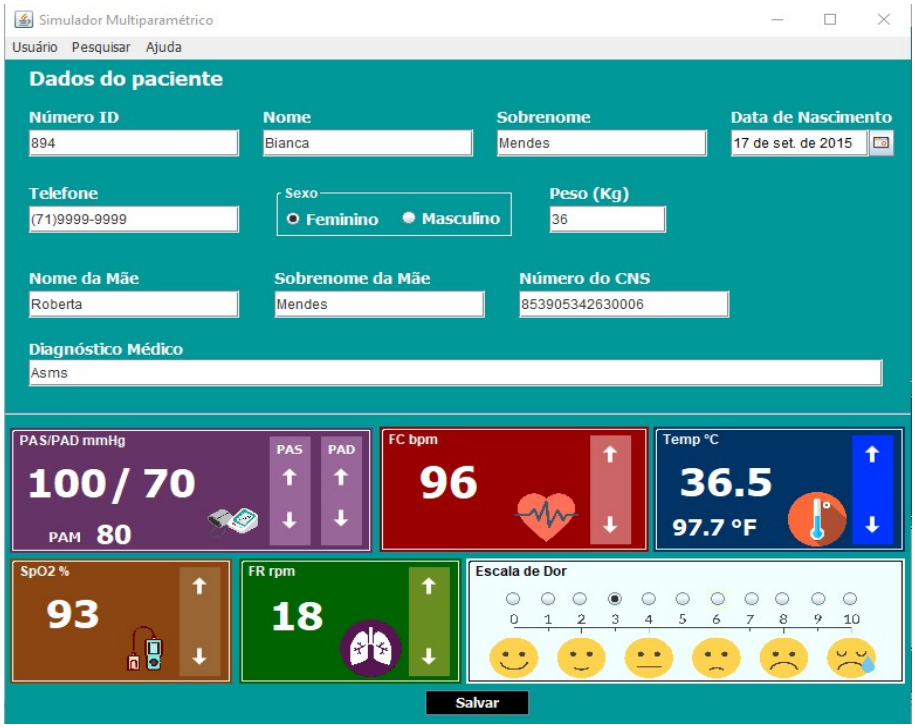


Figura 28: Tela de registro dos dados do paciente. Fonte: Figura do autor.

Figura 29: Resposta ao salvar os dados do paciente. Fonte: Figura do autor.

B. WEB

O site *web* foi construído com a finalidade de representar um PEP apenas para visualizar e consultar os pacientes cadastrados no simulador, e esses dados recebidos não precisam de tratamento para o formato HL7. Ademais, foi desenvolvido com a linguagem Typescript e a biblioteca ReactJS.

Para visualizar as informações dos pacientes, é preciso fazer um registro e *login*, assim como acontece no simulador, pois é necessário autorização. A seguir, segue as interfaces da *web*.

Figura 30: Modal de consulta do paciente pelo id. Fonte: Figura do autor.

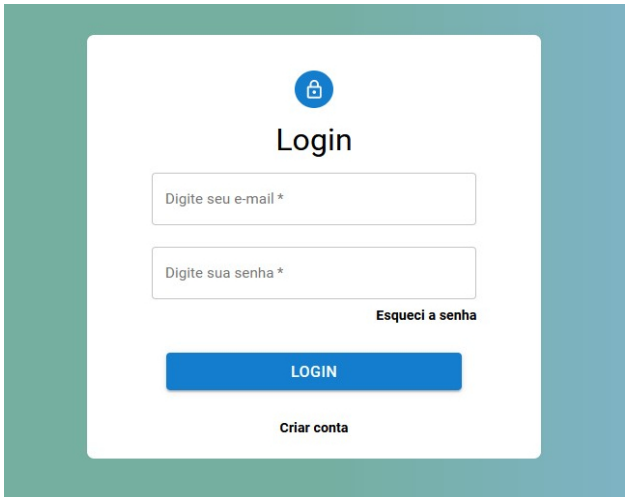


Figura 31: Tela de login. Fonte: Figura do autor.

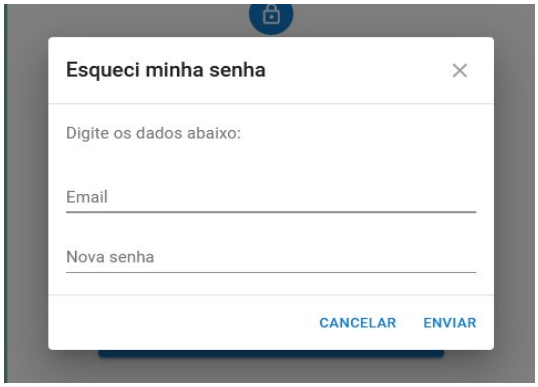


Figura 32: Modal de esqueci a senha. Fonte: Figura do autor.

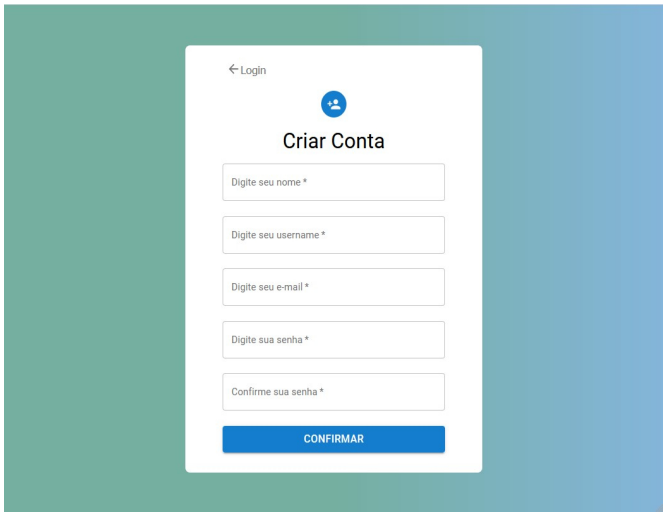


Figura 33: Tela de registro. Fonte: Figura do autor.

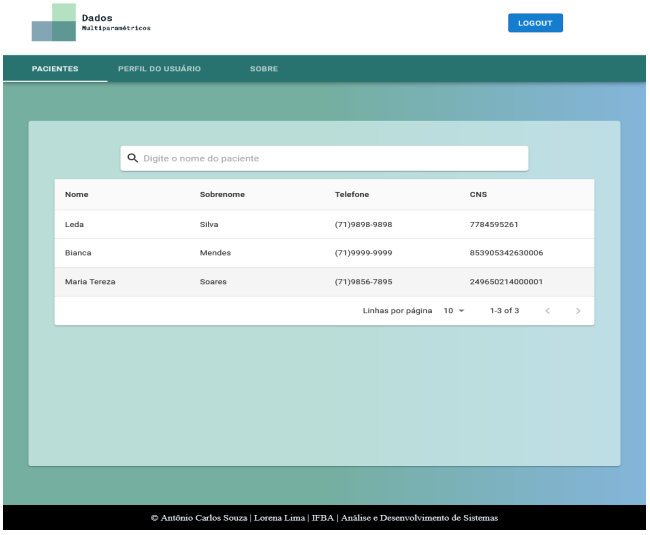


Figura 34: Sessão inicial. Fonte: Figura do autor.

Na figura 34, mostra uma lista de pacientes com o nome, sobrenome, telefone e número do CNS (Cartão Nacional de Saúde).

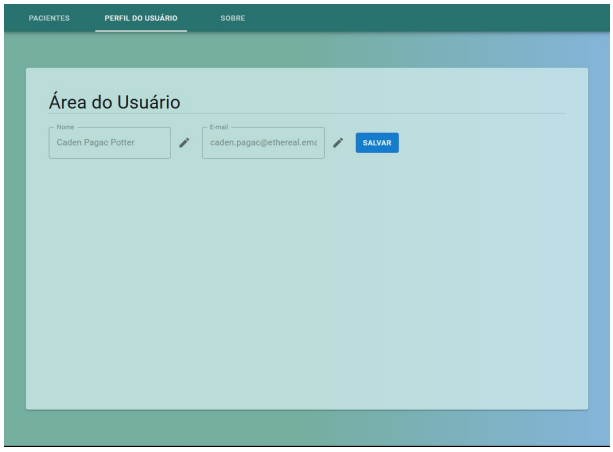


Figura 35: Sessão do perfil do usuário. Fonte: Figura do autor.

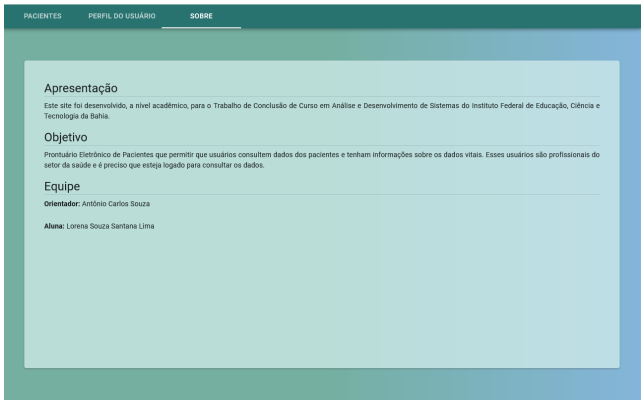


Figura 36: Sessão que fala sobre o projeto. Fonte: Figura do autor.

Na sessão Sobre, faz uma breve apresentação do projeto, com destaque, também, para o objetivo e equipe desenvolvedora.

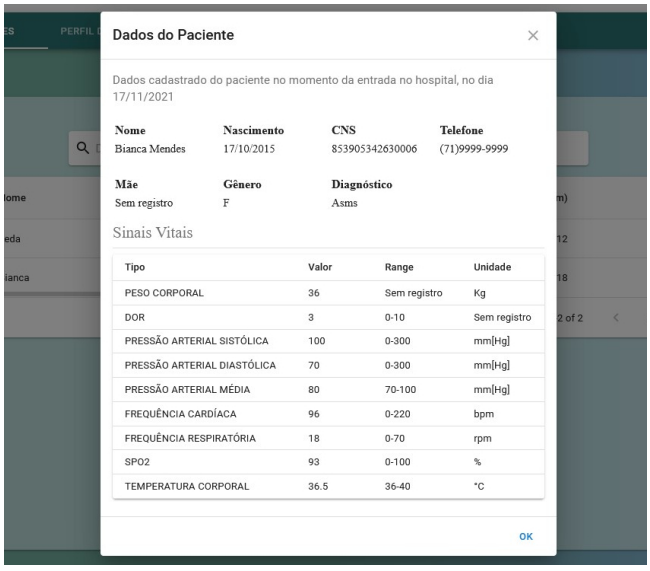


Figura 37: Modal com os dados do paciente. Fonte: Figura do autor.

Para que abra o modal com os dados do paciente, basta clicar em algum paciente na tabela.

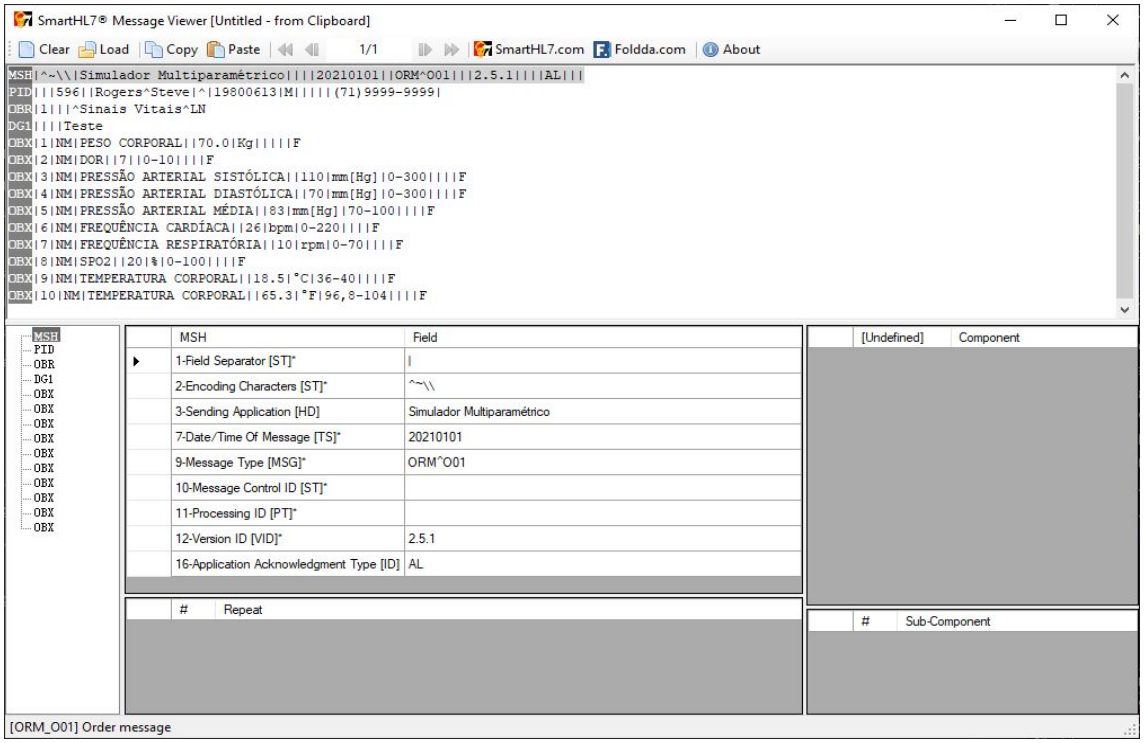


Figura 38: Ferramenta SmartHL7 Message Viewer. Fonte: Figura do autor.

E para que esses dados fossem repassados entres os diversos componentes de maneira simples, foi utilizado o *Context* API, do próprio React, como um gerenciado de estado.

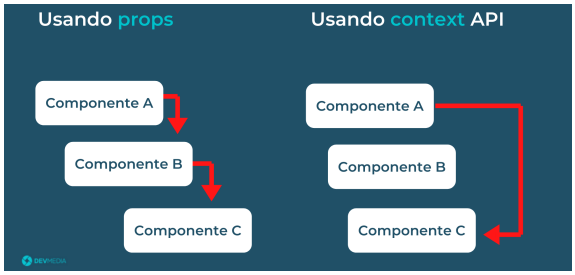


Figura 39: Uso do props X context API [37].

A figura 39 mostra a distribuição dos dados de duas formas: props e *context* API. Utilizando os props, os dados seriam transmitidos por meio da árvore de componentes, um componente depois do outro [37]. Com o *context*, os dados ficam disponíveis e podem ser compartilhados com qualquer componente.

C. Validação das Mensagens

Como já foi visto na seção da Fundamentação Teórica, existe muitos campos na estrutura da mensagem no formato HL7. Cada campo representa um tipo de informação. Sendo assim, foi necessário uma avaliação criteriosa a cada mensagem gerada durante o desenvolvimento, garantido que os dados estivessem nos lugares correspondentes. Para ajudar essa avaliação, foi utilizado a ferramenta *SmartHL7 Message Viewer* da empresa *Foldda Integrator* [38], vista na figura 38.

No *SmartHL7*, existe um espaço onde é colada a mensagem no formato HL7, como mostra na figura 40. Imediatamente após isso, é exibido os segmentos, e os campos e valores daquele segmento que estiver em destaque. Como exemplo, na figura 40, a linha referente ao segmento PID está em destaque, logo os campos que foram adicionados neste segmento são mostrados juntamente com os seus valores, exibidos na figura 41.

É possível notar que, na figura 41, além dos campos e valores, é exibido o tipo de mensagem e sua descrição no canto inferior esquerdo, onde mostra ADR_19 e *Patient query*. Assim, todas as mensagens, que transitaram entre as aplicações, foram passadas por essa ferramenta para validar se os valores estavam nos campos correspondentes. Além disso, foi utilizada a documentação [39] para uma segunda validação e um estudo mais aprofundado sobre os detalhes dos campos, disponível nos apêndices.

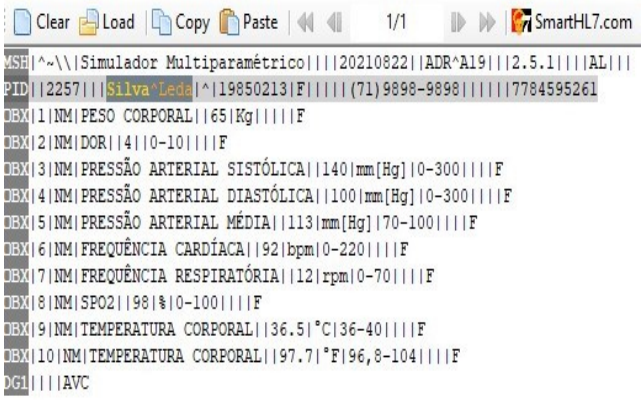


Figura 40: Mensagem no padrão HL7 no *SmartHL7*. Fonte: Figura do autor.

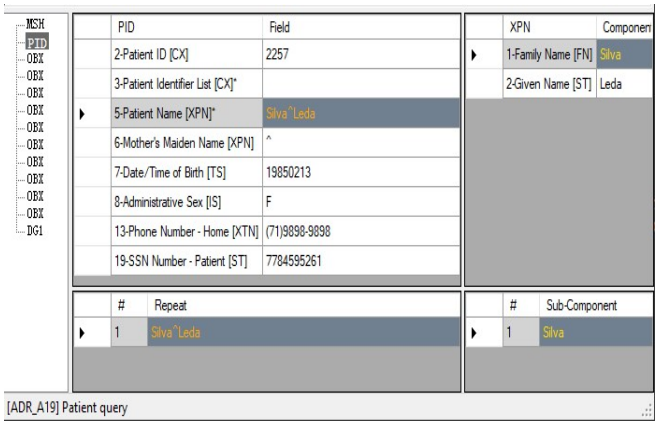


Figura 41: Campos do segmento PID no SmartHL7. Fonte: Figura do autor.

Figura 42: Representação do Simulador com os dados a serem registrados. Fonte: Figura do autor.

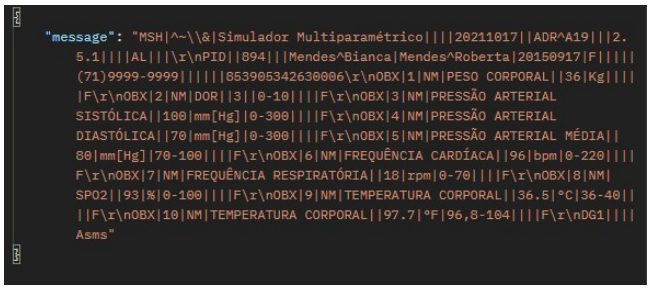


Figura 43: Resposta da consulta do paciente registrado na figura acima. Fonte: Figura do autor.

VI. Resultados Obtidos

Todos os dados utilizados durante avaliação da ferramenta (testes e validações) das mensagens em HL7, foram geradas a partir do simulador do equipamento biomédico e do PEP. Sendo assim, eles foram cadastrados, atualizados e deletados manualmente a fim de ter mais controle sobre os mesmos. Como foi mencionando na sessão de **Avaliação das Ferramentas**, no tópico **Validação das Mensagens**, todas as mensagens geradas pelo simulador e API passaram pela ferramenta *SmartHL7* com o intuito de conferir se as informações estavam nos campos certos, pois isso era a garantia de permitir o envio padronizado entre as diferentes linguagens, JAVA e Javascript.

A utilização das bibliotecas, para o padrão HL7, nas linguagens de programação escolhidas foram essenciais tanto na construção, como na decomposição da mensagem no formato HL7, permitindo uma melhor administração do tempo durante o desenvolvimento das aplicações. As mensagens foram enviadas em formato de *string*, com cada tipo de dado alocado em seu respectivo segmento sem haver inconsistências, mesmo sendo em diferentes linguagens. Com isso, foi possível observar que a interoperabilidade funcionou para essa simulação de sistema. A documentação da API pode ser acessada neste *link* [40].

A figura 42 mostra a interface do Simulador, onde os dados foram gerados e enviados para a API, no formato que é evidenciado na figura 43. Assim, com o tratamento, as informações foram extraídas dos campos dos segmentos e salvas no banco de dados, para serem enviados ao PEP e exibidas na tela inicial como é vista na figura 44.

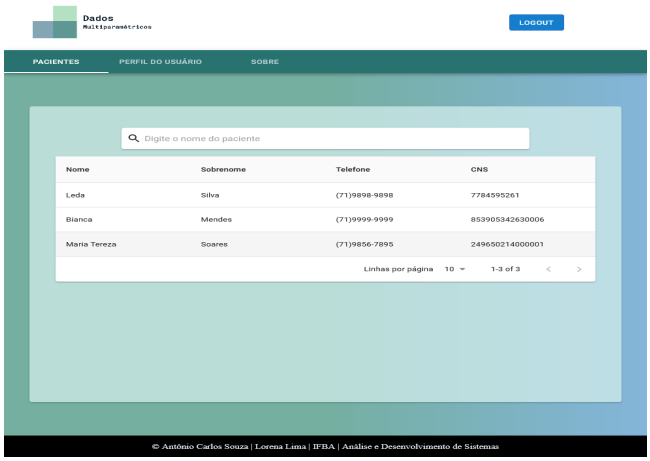


Figura 44: Representação dos dados no site Web. Fonte: Figura do autor.

Mesmo com a facilidade que as bibliotecas proporcionaram na geração das mensagens, houve uma pequena dificuldade em como usar. Na documentação, referente a linguagem JAVA, não estava claro como faria a conexão entre os segmentos para que a mensagem fosse criada, principalmente em relação ao tipo de mensagem, já que para cada uma existe uma estrutura. Além do mais, se algum segmento fosse colocado na ordem errada ou não fizesse parte da estrutura do tipo, ora a mensagem aparecia sem a

informação, ora dava erro. Então, com o JAVA foi necessário muitos testes e pesquisas para chegar ao formato final da mensagem. Já em JavaScript, foi necessário testar várias bibliotecas disponíveis até encontrar aquela que proporcionasse maior consistência no tratamento dos dados. Contudo, a manipulação da mensagem foi mais fácil do que em JAVA.

VII. Conclusão

Este projeto proporcionou um estudo mais aprofundado sobre o padrão HL7. Um padrão que está associado à interoperabilidade, e a sua implementação, nas diversas unidades de saúde, permitiria o compartilhamento de dados dos pacientes, controlaria os erros humanos, otimizaria o atendimento dos pacientes, reduziria os custos, retrabalhos e garantiria escalabilidade. No entanto, há muitos desafios a serem vencidos, principalmente sobre a interoperabilidade, que ainda é pouco discutida comparada aos países dos EUA e Europa. Juntamente com esse ponto, ainda existe muitos sistemas legados, o que pode ser uma das suas principais causas dificultando a integração entre sistemas.

A escolha das linguagens de programação, *frameworks*, *softwares* e bibliotecas, foi baseada no quanto tinha em domínio e consistência durante o desenvolvimento do código e na utilização no mercado, pois quanto mais visibilidade é colocada sobre algo, mais fácil fica a consulta. Logo, foi uma forma de mostra que é possível desenvolver algo inovador com o que tem de mais atual no mundo da tecnologia.

A mensagem no padrão HL7 pode ser de vários tipos, mas a preferência de utilizar apenas um tipo permitiu direcionar a atenção a quesitos importantes como: a criação e decomposição da mensagem, quais segmentos devem ser incorporados, a ordem a seguir e como a *string* deve ser adicionada ao JSON enviado no HTTP. Desse modo, foi possível atingir o objetivo proposto, com duas aplicações construídas em linguagens diferentes e compartilhando dados de pacientes por mensagens no formato HL7.

Portanto, fica evidente que é possível proporcionar a interoperabilidade entre sistemas. Mesmo tendo desenvolvido pequenas aplicações para este estudo, com dedicação, organização, mais estudo, planejamento, entre outros, é possível expandir para grandes sistemas e automatizar alguns processos como a escolha de outros tipos de mensagens baseado no atendimento do paciente.

VIII. Trabalhos Futuros

Durante a análise do projeto, foi possível observar que algumas melhorias poderiam ser implementadas nas aplicações, visando o crescimento e uso numa situação real. Diante deste contexto, destacam-se:

- Mudança na arquitetura da API, de monolítica para microsserviços com o intuito de facilitar a integração, o desenvolvimento e as manutenções, caso o projeto cresça de maneira acentuada.
- Possibilitar a criação de mensagens em HL7 de diferentes tipos, dinamicamente.
- Substituir o login, na parte do equipamento, por inserção de chaves ou códigos criptografadas, geradas durante o cadastro, facilitando a conexão de diversos outros equipamentos.
- Adicionar mais cargos na API, voltados para áreas da saúde, com o intuito de organizar as permissões de acesso aos dados e melhorar a segurança.
- Permitir que médicos possam gerenciar dados dos pacientes no PEP.
- Criar uma sessão para que usuários externos, como familiares, possam ter acesso ao boletim dos pacientes que são parentes.
- Implementação do *HTTPOnly* na API para aumentar a segurança ao acesso aos dados.
- Adicionar um sistema de mensagens que permita um fluxo de dados contínuos, como o Apache Kafka, com o intuito de ter os dados dos pacientes sempre atualizados e disponíveis para consulta.

Referências

[1] R. Alves, R. S. Maciel, and J. M. David, "Interoperabilidade social e colaboração: um mapeamento sistemático da literatura," in *Anais do XVII Simpósio Brasileiro de Sistemas Colaborativos*. Porto Alegre, RS, Brasil: SBC, 2021, pp. 83–94. [Online]. Available: https://sol.sbc.org.br/index.php/sbsc/article/view/16023

[2] J. G. Fernandes, "Interoperability between health information systems in the hospital context," Master's thesis, Instituto Universitário de Lisboa, oct 2020. [Online]. Available: http://hdl.handle.net/10071/21633

[3] L. Almeida. Interoperabilidade: intercâmbio seguro de dados do paciente. [Online]. Available: https://nexxto.com/interoperabilidade-intercambio-seguro-de-dados-do-paciente/

[4] O. J. N. Bittar, M. Biczyk, M. I. Serinolli, M. C. Z. Novaretti, and M. M. N. de Moura, "Sistemas de informação em saúde e sua complexidade," *Revista de Administração em Saúde*, vol. 18, no. 70, mar 2018.

[5] A. C. F. Leslie Yedidya Valencia Ramón, "Expediente clínico electrónico: Estado del arte," *Revista UNITEPC*, vol. 8, no. 1, jun 2021.

[6] A. M. Sciarra and J. Rondina, "Informática em saúde e a interoperabilidade nos sistemas hospitalares," *Arquivos de Ciências da Saúde*, vol. 25, no. 2, pp. 2–2, 2018.

[7] R. Sabbatini. História da informática em saúde no brasil. [Online]. Available: http://sbis.org.br/informatica-em-saude/

[8] B. de Faria Leão, "O desafio brasileiro para o uso de padrões em informática em saúde," *Journal of Health Informatics*, apr 2017.

[9] F. D. C. Moreira, "Interoperabilidade em sistemas de informação na saúde usando hl7," Master's thesis, Universidade do Minho, dec 2014.

[10] D. Lajas, D. N. Gonçalves-Ferreira, R. Oliveira, and R. Cruz-Correia, "Analysis and comparison of hl7 messages in a hospital services," in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, 2017, pp. 1–6.

[11] InteOpera. O HL7. [Online]. Available: http://interopera.esy.es/wp-content/uploads/2017/04/O-HL7-PRIME-1.pdf

[12] Wikipédia. Health Level 7. [Online]. Available: https://pt.wikipedia.org/wiki/Health_Level_7

[13] (2020, sep) Interoperabilidade em saúde: entenda o que é e quais os benefícios. [Online]. Available: https://medilab.net.br/2020/09/01/interoperabilidade-em-saude-entenda-o-que-e-e-quais-os-beneficios/

[14] InteOpera. O HL7 é um mistério para você? [Online]. Available: http://interopera.esy.es/wp-content/uploads/2016/12/HL7_101_V2_15122016.pdf

[15] M. J. L. Pais, "Mapeamentos de hl7-v2 para fhir," Master's thesis, Faculdade de Ciências, 2019.

[16] B. R. da Silva Pereira, "Implementação de um parser hl7 parainteração do alertr© com aplicações terceiras," Master's thesis, FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO, https://repositorio-aberto.up.pt/bitstream/10216/59893/1/000129616.pdf, jul 2008.

[17] wikiHow. (2020, aug) How to read hl7 messages. [Online]. Available: https://www.wikihow.com/Read-HL7-Messages

[18] T. A. Batista, "Análise Comparativa de Padrões de Interoperabilidade em Sistemas Hospitalares." 2016, monografia (Bacharel em Ciência da Computação), UNIJUI (Universidade Regional do Noroeste do Estado do Rio Grande do Sul), Santa Rosa, Brasil. [Online]. Available: https://bibliodigital.unijui.edu.br:8443/xmlui/bitstream/handle/123456789/4200/Tatiane%20Aparecida%20Batista.pdf?sequence=1&isAllowed=y

[19] V. Nathan. Hl7-intro. [Online]. Available: https://medium.com/@mmayuravaani/hl7-intro-6bcdcec8de0

[20] M. Ohlinger, K. Sharkey, and S. Cai. (2017, aug) Hl7 message structure. [Online]. Available: https://docs.microsoft.com/en-us/biztalk/adapters-and-accelerators/accelerator-hl7/hl7-message-structure

[21] C. G. L. BARRETO, "Avaliação de usabilidade de um monitor multiparamétrico utilizado em um estabelecimento assistencial de saúde público," Master's thesis, Universidade Federal de Uberlândia, http://clyde.dr.ufu.br/bitstream/123456789/24983/5/Avalianov 2018.

[22] [Online]. Available: http://www.lifemed.com.br/produto/lifetouch-10

[23] T. A. A. E. Shheibia, "Um modelo de monitoração de pacientes na uti usando micro servidor web," Master's thesis, Universidade Federal de Santa Catarina, https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/84846/196910.pdf?sequence=1&isAllowed=y, oct 2003.

[24] L. W. GONÇALVES, "Prontuário eletrônico do paciente adotando padrões para a telemedicina no brasil," Master's thesis, UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, https://www.lume.ufrgs.br/bitstream/handle/10183/26348/000757803.pdf?sequence=1?, jun 2010.

[25] A. C. de Loureiro e Nogueira, "Validação da identificação de utentes num integrador de mensagens hl7 para monitorização e portabilidade de dados na área da saúde," Master's thesis, Universidade do Porto, sep 2019.

[26] P. H. A. Cavalcante. (2021, may) Introdução a typescript: o que é e como começar? [Online]. Available: https://blog.geekhunter.com.br/introducao-a-typescript/

[27] A. P. de Andrade. (2021, jan) O que é o express.js? [Online]. Available: https://www.treinaweb.com.br/blog/o-que-e-o-express-js

[28] M. Gajda. (2020, jul) Why use node.js web development? scalability, performance and other benefits of node based on famous web applications. [Online]. Available: https://tsh.io/blog/why-use-nodejs/

[29] I. de Souza. (2020, aug) Postgresql: saiba o que é, para que serve e como instalar. [Online]. Available: https://rockcontent.com/br/blog/postgresql/

[30] T. Graham. (2019, nov) 6 things for developers to know about postgres. [Online]. Available: https://thenewstack.io/6-things-for-developers-to-know-about-postgres/

[31] *React*, Facebook, https://pt-br.reactjs.org/, 2021.

[32] O. Software. (2021, mar) Micro serviços: Qual a diferença para a arquitetura monolítica? [Online]. Available: https://www.opus-software.com.br/micro-servicos-arquietura-monolitica/

[33] J. LUCIANO and W. J. B. ALVES. (2011) PadrÃo de arquitetura mvc: Model-view- controller. [Online]. Available: https://unifafibe.com.br/revistasonline/arquivos/revistaepeqfafibe/sumario/20/16112011142249.pdf

[34] G. Vitulo. (2017, oct) O que é, como são feitas e qual a importância das web apis? [Online]. Available: https://inovacaodigital.blog/o-que-e-api/

[35] (2014) Introduction to json web tokens. [Online]. Available: https://jwt.io/introduction

[36] M. contributors. (2021, nov) Cross-origin resource sharing (cors). [Online]. Available: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/CORS

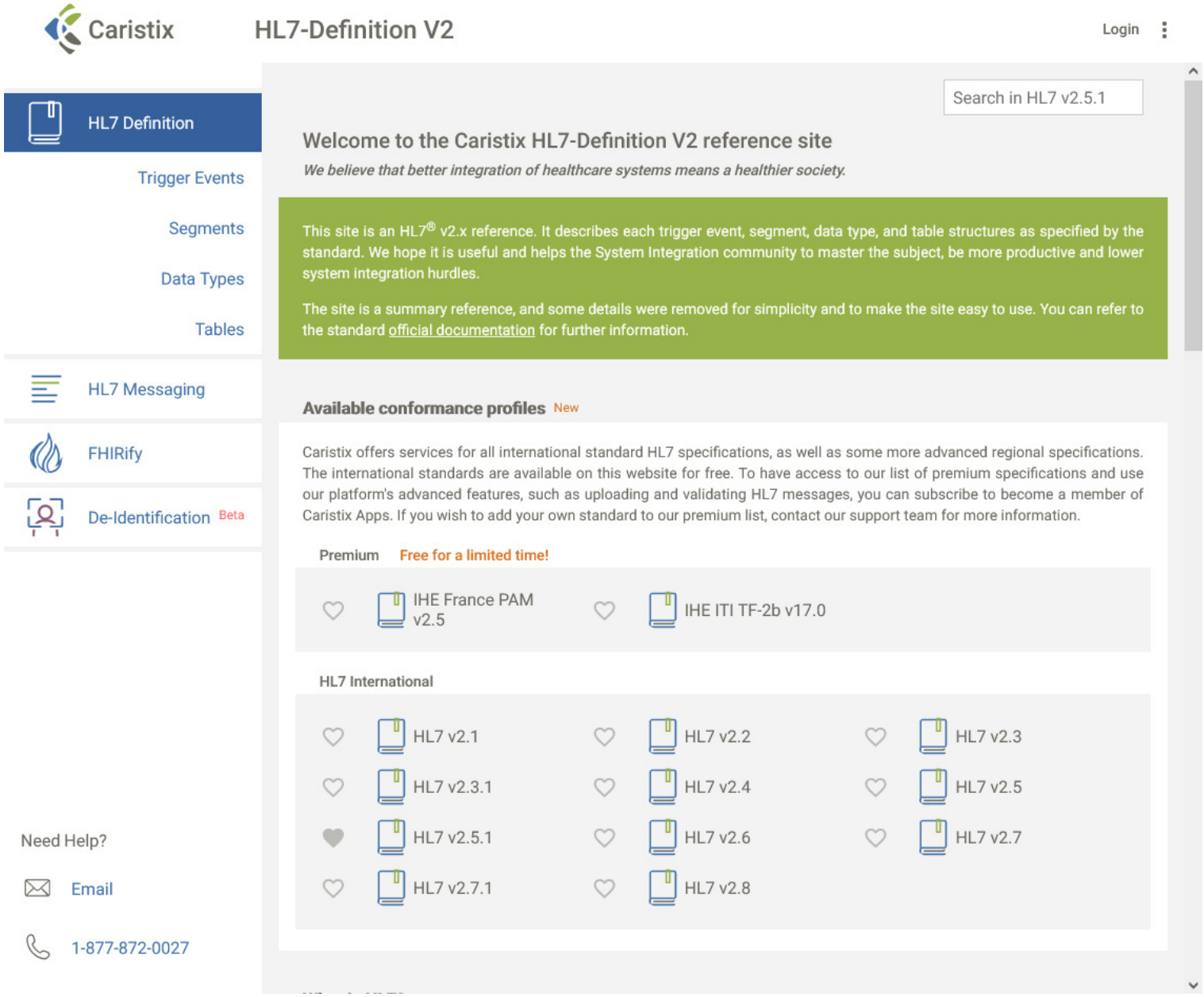
[37] R. Sousa. (2021) React.js: Passando dados com context api. [Online]. Available: https://www.devmedia.com.br/react-js-passando-dados-com-context-api/42904

[38] M. Chen. (2021, aug) Hl7 message viewer. [Online]. Available: https://foldda.com/2021/08/01/hl7-message-viewer/

[39] Caristix. (2021) Caristix apps. [Online]. Available: https://apps.caristix.com/console/Home

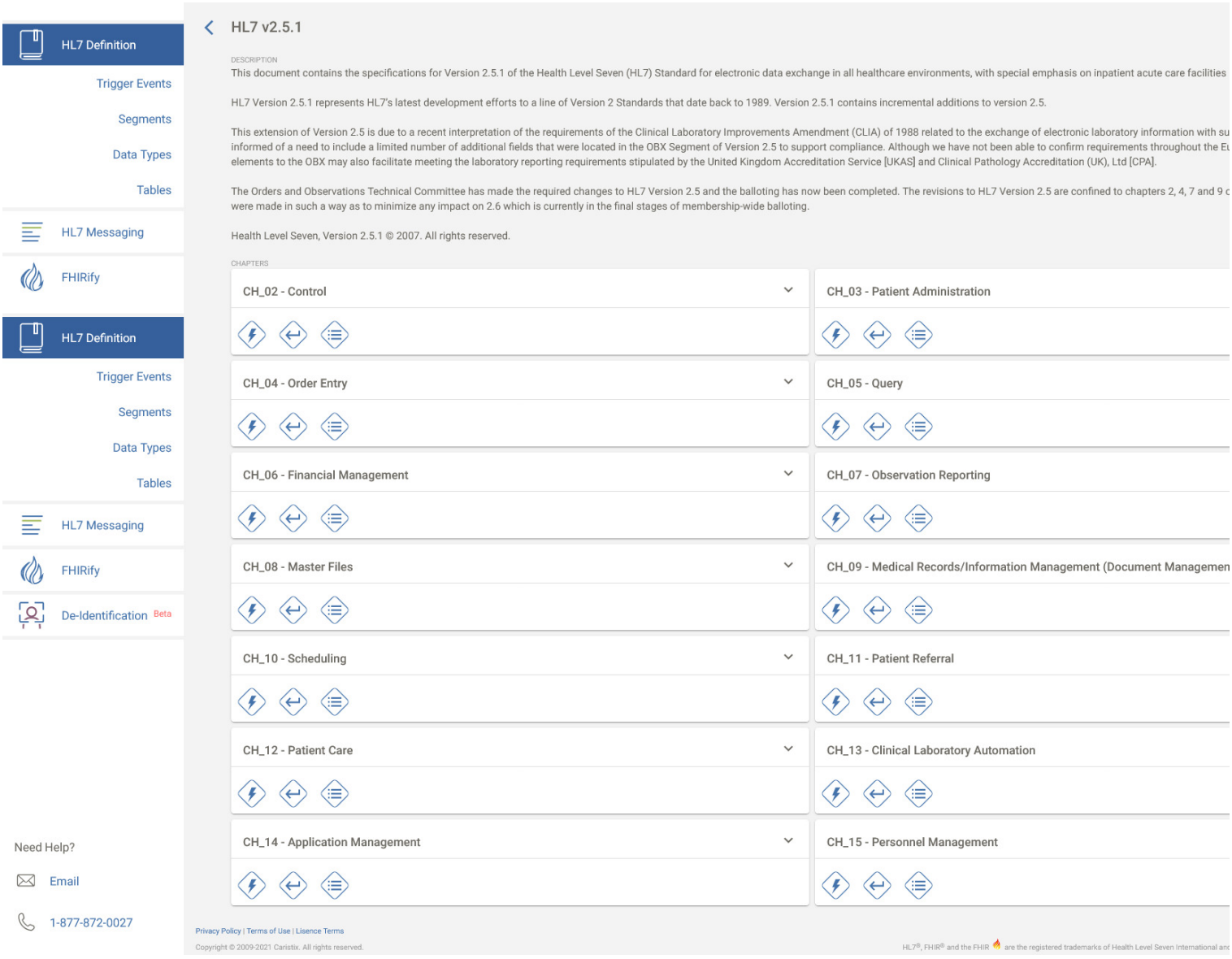
[40] L. S. S. Lima. (2021, aug) Hl7-api. [Online]. Available: https://documenter.getpostman.com/view/5841921/TVsskUBE

Apêndice A
Versões do padrão HL7

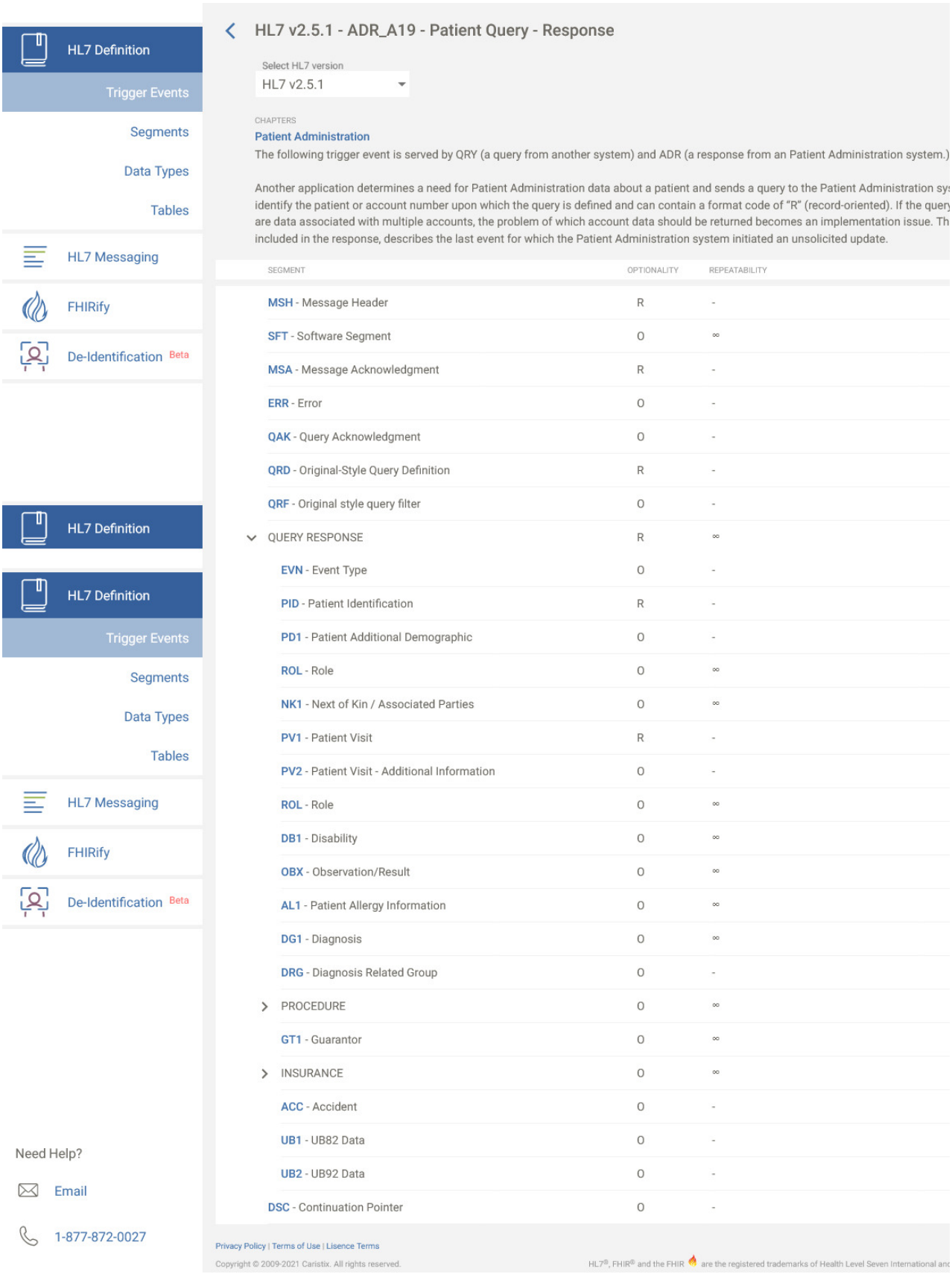


Apêndice B

Especificações da versão 2.5.1 do padrão HL7 utilizada no projeto



Apêndice C
Segmentos permitidos no tipo de mensagem ADR_A19



Apêndice D

Exemplo: Especificações do segmento MSH