UNIVERSIDADE DE TAUBATÉ

Eduardo Augusto Rodrigues de Oliveira

Renan dos Santos

USO DA ARQUITETURA ORIENTADA A SERVIÇO (SOA) APLICADA NA ÁREA DE SAÚDE

Taubaté - SP

2019

UNIVERSIDADE DE TAUBATÉ

Eduardo Augusto Rodrigues de Oliveira

Renan dos Santos

USO DA ARQUITETURA ORIENTADA A SERVIÇO (SOA) APLICADA NA ÁREA DE SAÚDE

Trabalho de Conclusão de Curso apresentado como requisito parcial para a conclusão do Curso de Engenharia de Computação do Departamento de Informática da Universidade de Taubaté. Orientação: Prof. Dr. Márcio Augusto Ernesto de Moraes.

Taubaté - SP

2019

Ficha catalográfica elaborada pelo
SIBi – Sistema Integrado de Bibliotecas / UNITAU

O48u Oliveira, Eduardo Augusto Rodrigues de
Uso da arquitetura orientada a serviço (SOA) aplicada na área de saúde / Eduardo Augusto Rodrigues de Oliveira, Renan dos Santos. - 2019.
60f. : il.

Monografia (graduação) - Universidade de Taubaté, Departamento de Informática, 2019.
Orientação: Prof. Dr. Márcio Augusto Ernesto de Moraes, Departamento de Informática.

1. *Web services*. 2. Arquitetura orientada a serviços (computador). 3. *Middleware*. I. Santos, Renan dos. II. Universidade Taubaté. III. Título.

CDD 005.3

Ficha catalográfica elaborada pela Bibliotecária Regina Márcia Cuba – CRB 8a/7416

Eduardo Augusto Rodrigues de Oliveira

Renan dos Santos

USO DA ARQUITETURA ORIENTADA A SERVIÇO (SOA) APLICADA NA ÁREA DE SAÚDE

Trabalho de Conclusão de Curso apresentado como requisito parcial para a conclusão do Curso de Engenharia de Computação do Departamento de Informática da Universidade de Taubaté.

Data: 23/09/2019

Resultado: APROVADO

BANCA EXAMINADORA

Prof. Dr. Márcio Augusto Ernesto de Moraes
Universidade de Taubaté.

Prof. Me. Antonio Esio Marcondes Salgado
Universidade de Taubaté.

Prof. Esp. Edgar Israel
Universidade de Taubaté.

Prof. Me. Dawilmar Guimarães Araújo
Universidade de Taubaté.

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

LISTA DE SIGLAS

SOA – Service Oriented Architecture ou Arquitetura Orientada a Serviços

HTTP – Hypertext Transfer Protocol ou Protocolo de Transferência de hipertexto

HTTPS – Hypertext Transfer Protocol Secure ou Protocolo de Transferência de hipertexto Segura

WSLD – Web Service Description Language ou Linguagem de Descrição de Serviços Web

XML – Extensible Markup Language ou Linguagem extensível de Marcação

REST – Representational State Transfer ou Transferência Representacional de Estado

RESTful – Resentational State Transfer ou Transferência Representacional de Estado

SOAP – Simple Object Access Protocol ou Protocolo Simples de Acesso a Objeto

PEP – Prontuário Eletrônico do Paciente

UML – Unified Modelling Language ou Linguagem de Modelagem Unificada

RESUMO

Nos dias de hoje, com o aumento da procura por consultórios médicos, consultórios e hospitais vem pensando em diversas formas de facilitar o atendimento do paciente com o médico. Essa situação acontece também nas áreas de informática, onde software e sistemas podem otimizar tempo e recursos. O projeto utiliza uma Arquitetura Orientada a Serviços com o objetivo de apresentar a tecnologia Web Service para clínicas e hospitais da rede de saúde privada, facilitando a comunicação entre clínicas e a vida do paciente. Para o desenvolvimento iniciou pelo entendimento dos conceitos das principais tecnologias abordadas: SOA, Web Service, WSLD, XML, REST, RESTful, SOAP e Middleware. O desenvolvimento em si foi baseado nas práticas da engenharia de software, partindo pelo levantamento dos requisitos até a implementação, este trabalho fez uso das tecnologias supracitadas orientando-se pela segurança e desempenho. O middleware, um programa em Python Django, foi desenvolvido com os conceitos como uma proposta de Web Service, cuja finalidade é facilitar o atendimento do paciente com o médico. Ao final desse projeto, conclui-se que com o uso de Web Service é possível facilitar alguns processos em clínicas e hospitais.

Palavras-chave: Web-service, Arquitetura Orientada a Serviços, Middleware, Troca de Mensagens

ABSTRACT

Nowadays with the increase in doctor's office search, hospitals and doctor's offices are thinking in many ways to facilitate the attendance of a patient with a medic. This situation also happens in the computing fields, where software and systems can optimize time and resources. The project uses a Service Oriented Architecture with the objective of presenting the Web Service technology to clinics and hospitals of the private health net, facilitating the communication between clinics and the patient's life. This work will show the involved technologies, which approach the concepts of SOA, Web Service, WSLD, XML, REST, RESTful, SOAP and Middleware. The development was based on software engineering practices, starting with requirements gathering until the implementation, this work made use of the technologies above oriented by security and performance. The middleware, a Python Django program, was developed with the concepts above as a proposal of Web Service, whose goal is to facilitate the treatment between patient and medic. At the end of this project it was concluded that with the use of a Web Service it's possible to facilitate some processes in hospitals and doctor's offices.

Keywords: Web Service, Service Oriented Architecture, Middleware, Message Exchange

1. INTRODUÇÃO

Serviços de saúde são serviços que visam a manutenção e restauração da saúde das pessoas. Eles não servem apenas para diagnóstico e tratamento de doenças, mas também para prevenção dessas doenças.

Para complementar as anáises clínicas, exames clínicos com a finalidade de verificar o estado de saúde ou invetigar doenças são utilizados pelos medicos para obter um melhor diagnóstico e acompanhar as condições de saúde do paciente.

Nos dias de hoje, para atender as necessidades das pessoas são contratados terceiros para ajudá-las. Essa situação acontece também nas áreas de tecnologia e informática. Softwares e sistemas podem otimizar tempo e recursos quando utilizam-se funções implementadas por terceiros, as quais damos o nome de serviços.

O que é Arquitetura Orientada a Serviços?

O conceito de Arquitetura Orientada a Serviços ou SOA (*Service-Oriented Architecture*) foi proposto em 1996, pelos pesquisadores Roy Schulte e Yefim Nates.

Arquitetura Orientada a Serviços (SOA) é uma arquitetura que integra serviços reutilizáveis, considerando-se que cada serviço é bem definido dentro de suas propostas e são fracamente acopladas, ou seja, um não depende do outro, mas podem trabalhar em conjunto.

O foco principal do projeto é criar um sistema voltado para prontuários eletrônicos, ou seja, quando um paciente precisar fazer um exame em alguma clínica ou hospital, ao invés de ter que levar um encaminhamento dizendo quais exames serão feitos, o médico envia pelo sistema quais os exames a serem realizados, e após feito, recebe o resultado mais rápido, agilizando todo esse processo.

Quando o médico pedir exames ao paciente ele mesmo envia pelo sistema quais as clínicas e quais os exames serão realizados. Dessa forma as clínicas sabem previamente que o exame será realizado naquele local e o paciente precisa levar somente seus documentos pessoais e na clínica já consegue ter acesso aos exames que serão realizados e qual o médico que solicitou.

Quando os exames estiverem prontos, as clínicas enviam pelo sistema os resultados ao médico que solicitou, facilitando para o médico que consegue ver todos os exames e os locais onde foram realizados e facilitando para o paciente que não precisa guardar os resultados e leva-los para o médico depois.

1.1. MOTIVAÇÃO

Na rede pública de saúde, todos os dados, de todas as clínicas de saúde, estão disponíveis para cada uma das clínicas da rede pública, de forma que cada clínica pode acessar as informações de

outro, caso seja necessário. Isso é possível pois todos os dados estão armazenados num mesmo servidor, no qual todos possuem acesso.

Na rede privada de saúde, cada clínica de saúde armazena seus dados unicamente para si. Caso seja necessário acessar os dados de outra clínica, seria preciso obter acesso ao sistema daquela clínica. Se fosse necessário passar por quatro unidades, seriam quatro acessos diferentes e assim por diante.

A motivação do desenvolvimento desse projeto tem como objetivo facilitar a comunicação entre clínicas da rede privada de saúde, de forma a facilitar o atendimento ao paciente, pois o mesmo não precisaria carregar documentos, o processo de agendamento médico e entrega de resultados de exames seriam otimizados, uma vez que seria possível o envio de dados entre uma clínica e outra.

1.2. JUSTIFICATIVA

A cada ano, vários pacientes passam por hospitais e clínicas médicas para realizarem consultas e fazer exames. Em cada consulta realizada em uma determinada clínica, o médico tem que ter acesso aos dados do paciente e algumas vezes acesso a última consulta realizada por aquele paciente para poder dar um melhor atendimento ao paciente. Digamos que, por exemplo, após realizar uma consulta, o médico peça uma série de exames ao paciente, entretanto os exames não são feitos nesse consultório, mas sim, em outros clínicas e hospitais. O procedimento seria o médico entregar um papel com as informações sobre os exames e o paciente levaria aos outros consultórios e hospitais. Após os exames serem realizados, o paciente pega os resultados, reagenda no consultório que pediu os exames e entrega os exames ao médico. Nesse meio tempo entre o pedido e entrega de exames podem acontecer várias coisas com todos esses papéis e exames, como por exemplo, o paciente pode amassa-los, perde-los ou esquecer algum dos exames no retorno ao médico que os pediu.

A intenção desse trabalho é facilitar a vida de médicos e pacientes, transformando toda essa documentação em um documento eletrônico, ou seja, agilizar o processo após a realização de exames, pois o médico teria acesso aos exames logo após o paciente realizá-los, não precisando esperar o paciente retornar ao consultório para poder vê-los, além de evitar uma possível perda dos exames caso o paciente estivesse carregando-os, tornando o processo mais seguro.

1.3. OBJETIVOS

O objetivo do projeto é desenvolver um sistema em uma arquitetura orientada a serviços, visando melhorar os agendamentos médicos através de um sistema web na rede de saúde privada.

1.3.1. OBJETIVOS ESPECÍFICOS

Têm-se como objetivos específicos do projeto:

- Facilitar a comunicação entre clínicas de saúde;
- Agilizar o processo de encaminhamento e realização de exames médicos;
- Facilitar o acesso aos dados pelos usuários do sistema;
- Permitir uma comunicação segura;
- Acessível através da Web;
- Dispensar o uso de papéis.

1.4. DESCRIÇÃO DO PROJETO

O foco do projeto é criar um sistema de prontuários eletrônicos, ou seja, quando um paciente tiver que fazer um exame em alguma clínica, ao invés de levar um encaminhamento marcando quais exames serão feitos, o médico envia pelo sistema quais serão realizados, e após feito, acaba tendo o resultado mais rápido, agilizando todo o processo.

Assim que o médico solicitar exames ao paciente ele mesmo envia pelo sistema quais as clinicas e quais exames serão realizados, assim as clínicas já ficam cientes que o exame será feito ali. O paciente precisa levar apenas seus documentos pessoais consigo para realizar o exame, pois a clínica consegue ter acesso aos exames que devem ser realizados e qual médico solicitou.

Após a realização dos exames as clínicas enviam pelo sistema os resultados ao médico que solicitou, assim o paciente não precisa guardá-los e leva-los no retorno ao médico. Por fim, quando retornar ao médico ele consegue ver todos os exames que foram feitos, onde foram realizados e os resultados de uma forma mais fácil, sem precisar de vários encaminhamentos e nem de correr o risco de perder algum exame se tivesse que leva-los ao médico no retorno.

1.5. ORGANIZAÇÃO DO TRABALHO

Para um melhor entendimento deste trabalho, ele será dividido da seguinte forma:

- Capítulo 1 - Introdução: Uma breve descrição do que será tratado no trabalho, objetivos e motivação.
- Capítulo 2 – Descrição do Projeto: Serão apresentados os principais conceitos sobre prontuário eletrônico, vantagens, desvantagens e dificuldades de se implantar um.
- Capítulo 3 – Metodologia de desenvolvimento: Será feita uma revisão de várias tecnologias que serão utilizadas no projeto como SOA, Web Services, WSLD, XML, REST, RESTful, SOAP e Middleware.
- Capítulo 4 - Tecnologia e Ferramentas a serem Utilizadas: Apresentação das tecnologias e ferramentas que serão utilizadas para a implementação do sistema.
- Capítulo 5 - Análise e Projeto: Análise e representação do ciclo do projeto.
- Capítulo 6 - Conclusão: Conclusão do projeto.

2. REVISÃO BIBLIOGRÁFICA

A metodologia do desenvolvimento visa estabelecer uma abordagem contendo detalhes para desenvolver ou melhorar sistemas, informando cada um de suas tarefas, técnicas e ferramentas utilizadas durante o desenvolvimento do projeto. A metodologia proposta para este trabalho é a Arquitetura Orientada a Serviços (*Service Oriented Architecture*), que é uma abordagem de arquitetura que define e integra serviços reutilizáveis.

3. METODOLOGIA DE DESENVOLVIMENTO

Neste item será feito uma revisão de todas as tecnologias utilizadas no projeto.

3.1. ARQUITETURA ORIENTADA A SERVIÇOS (SOA)

SOA é criado com base em padrões reconhecidos e suportados por provedores de TI, como *Web Services*. Empresas podem trocar dados e informações, independentemente de suas infraestruturas, economizando tempo, espaço, recursos financeiros e mão-de-obra. Isso possibilita eficiência e reutilização, além de uma capacidade de alinhar a parte de gestão com a tecnologia da informação e aumentar produtividade, competitividade e segurança da empresa no mercado.

De acordo com ERL (2004; apud Campos, Donadel, Todesco, Varvákis, de Souza Bermejo, 2006) SOA introduz uma nova camada de lógica (serviços) de integração como uma interface de comunicação entre as aplicações (interface de serviços), permitindo-as interagirem oferecendo e consumindo serviços uma da outra.

De acordo com Barry (2003; apud Mabrouk, 2008): “Um serviço é uma função bem definida, auto-contida e não depende no contexto ou estado de outros serviços. Um serviço é definido como uma unidade de trabalho a ser realizado em nome de uma entidade de computação, como um usuário humano ou outro programa.”

A identificação de serviços depende do tipo de projeto e das informações disponíveis e há diferentes fontes para identificar os serviços candidatos. Estas fontes não se excluem mutualmente, podendo-se utilizar mais de uma delas em um mesmo projeto (MACKENZIE et al., 2006; apud de Souza, 2016).

SOA é algo novo no mercado, e é possível que algumas pessoas não compreendam corretamente o que ele é. A seguir uma lista de coisas que um SOA não é:

- Não é uma tecnologia;
- Não é um produto;
- Não é um Web Service;

- Não é um projeto de TI;
- Não é um software;
- Não é um framework;
- Não é uma metodologia;
- Não é uma solução de negócio;
- Não é um middleware;
- Não pode ser comprada;
- Não é um serviço;
- Não é uma ferramenta de produtividade.

Pode-se dizer que: Não é fácil mudar algo que já existe, principalmente em empresas, e não é diferente com SOA, que também possui vantagens e desvantagens.

Vantagens do SOA:

- **Reutilização:** O serviço é reutilizável. Promove flexibilidade, dando margem para que os desenvolvedores consigam inovar. Dá ao consumidor uma extensa escolha de suprimentos;
- **Produtividade:** Os serviços são reutilizados em outros projetos, diminuindo o tempo de desenvolvimento;
- **Padronizado:** É utilizado com base em padrões;
- **Interoperabilidade:** Os serviços são disponibilizados independente da infraestrutura, plataforma e tecnologia;
- **Manutenibilidade:** Fácil manutenção dos serviços;
- **Integração:** Integração com outros sistemas e aplicativos.

Desvantagens do SOA:

- **Complexidade:** Grande quantidade de dados a ser gerenciado;
- **Desempenho:** O bom funcionamento depende do servidor de rede;
- **Disponibilidade:** Caso haja queda na rede, os serviços ficam indisponíveis;
- **Testabilidade:** Debug é um grande problema para os desenvolvedores;
- **Segurança:** Os serviços são disponíveis na rede. Qualquer aplicativo pode ter acesso, pois os dados são trafegados e podem ser interceptados.

De acordo com ENDREI (2004; apud Campos, Donadel, Todesco, Varvákis, de Souza Bermejo, 2006) Nessa arquitetura são definidos os seguintes papéis:

- **Consumidor de serviço:** é constituído por uma aplicação, um módulo de software ou outro serviço que requer um serviço. O comunicador executa o serviço segundo o contrato de interface;
- **Provedor de Serviço:** é uma entidade que aceita e executa solicitações de consumidores. Ele publica os serviços e contrata a interface ao registro de serviço de modo que o consumidor de serviço possa descobrir e acessar o serviço;
- **Registro de serviço:** é o facilitador para a descoberta de serviço. Ele contém um repositório de serviços disponíveis e leva em conta para a pesquisa de serviço interfaces de provedor para consumidores que se interessarem pelo serviço.

O desenvolvimento de aplicações tem sido direcionado ao acesso direto do usuário através de interfaces gráficas e essa abordagem é centrada nos fluxos do usuário e nas funcionalidades por ele esperadas (Juric et al., 2008; apud Pereira Siqueira, Nogueira de Oliveira e Aparecida de Oliveira, Mai./Ago. 2016). Todos esses sistemas funcionam de maneira isolada, devido à falta de padronização e às informações serem armazenadas em pequenos sistemas incompatíveis entre si, formando verdadeiras “Ilhas de Informaçåo” (Wang et al., 2010; apud Pereira Siqueira, Nogueira de Oliveira e Aparecida de Oliveira, Mai./Ago. 2016). Como eliminar essas ilhas de informação e como realizar a interoperabilidade de forma efetiva são problemas desafiadores na indústria médica (Hammami et al., 2014; apud Pereira Siqueira, Nogueira de Oliveira e Aparecida de Oliveira, Mai./Ago. 2016).

3.2. WEB SERVICES

Web service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

Web Service nada mais é que uma classe em *Python* ou outra linguagem. Essa classe pode ser consumida por um *Web Form* ou até mesmo por outros *Web Services*, independente da linguagem em que foram criados.

Segundo Leandro (2005), conforme citado por Peracci e Almeida Bessa (2014) o Web Service tem sido usado em diversos tipos de aplicações, como utilização dos serviços Web em interfaces de sistemas legados ou desktop devido à necessidade de tornar os processos de negócio mais acessíveis.

Segundo Miranda (2005), conforme citado por Peracci e Almeida Bessa (2014), podem ser destacados, dentre os benefícios existentes a independência de plataforma e linguagem de

programação, permitindo que programas de diferentes linguagens e em diferentes plataformas se comuniquem um com o outro de uma forma padrão, isto é, se um empresário cria um *Web Service* em *Python* contendo todas as regras da sua empresa, onde suas aplicações *Web Form* irão consumi-lo e futuramente pretende-se disponibilizá-las para seus parceiros ou clientes, não haverá problemas, mesmo que seus parceiros possuam aplicações em Java, VB.NET, C-Sharp ou qualquer outra linguagem que existe ou venha a existir e tenha a capacidade para consumir um *Web Service*.

Web Service é um serviço disponível na Web e acessível através de protocolos da Web como HTTP, HTTPS, etc. Pode ser acessado também por outras plataformas, como dispositivos móveis.

3.3. WEB SERVICE DESCRIPTION LANGUAGE (WSLD)

WSDL é uma linguagem baseada em XML e utilizada para descrever Web Services funcionando como um "contrato" do serviço. Este contrato descreve o serviço, especifica como acessá-lo e quais operações ou métodos estão disponíveis, o que ele pode fazer. Ele descreve completamente a interface de um serviço.

Através do WSDL, podemos descrever os serviços externos ou interfaces que são oferecidos por uma determinada aplicação, independentemente de sua plataforma ou linguagem de programação. Para ser feita a troca de informação entre a aplicação e o Web Service, é necessário que essa aplicação conheça o funcionamento do mesmo. Isso é feito por meio da leitura do WSDL. Ele tem como principal objetivo descrever as interfaces apresentadas e apontar a localização dos seus serviços, disponíveis em um local previsível e bem conhecido na rede, o qual permite que o cliente acesse de maneira confiável.

Para enxergar o valor do WSDL, imagine que você quer invocar um método SOAP, que é fornecido por seus parceiros de negócio. Você pede alguns exemplos de mensagens SOAP e escreve sua aplicação a fim de produzir e consumir mensagens que se parecem com os exemplos, porém isso pode gerar alguns erros. Um exemplo, você pode informar que um campo é um inteiro quando na verdade se trata de uma *string*. O WSDL especifica o que a mensagem de requisição deve conter e como vai ser a resposta, em uma notação não ambígua.

Os elementos básicos do WSDL são (José Silva, 2018):

- <definições> elemento deve ser o elemento raiz de todos os documentos WSDL.Ele define o nome do serviço web.
- <tipos> elemento se encarrega de definir os tipos de dados que são usados pelo serviço web.Tipos são documentos XML, ou partes do documento.

- <mensagem> elemento descreve os dados que estão sendo trocados entre os prestadores de serviços web e os consumidores.
- <portType> combina vários elementos da mensagem para formar um one-way completa ou operação de ida e volta.
- <binding> elemento fornece detalhes específicos sobre como uma operação vai realmente ser transmitida através do fio.
- <port> define um terminal individual, especificando um endereço único para uma ligação.
- <service> define as portas suportadas pelo serviço web. Para cada um dos protocolos suportados, há um elemento de entrada. O elemento de serviço é um conjunto de portas.

Um documento WSDL também pode conter outros elementos, como elementos de extensão e um elemento de serviço que torna possível agrupar as definições de vários serviços da web em um único documento WSDL único.

3.4. EXTENSIBLE MARKUP LANGUAGE (XML)

XML, ou Linguagem de Marcadores Extensíveis, é uma linguagem muito utilizada atualmente em diversas aplicações, devido ao fato de ser possível criar marcadores de acordo com a necessidade corrente, permite que representamos muitos tipos de dados. A facilidade disponibilizada pela extensibilidade dos marcadores torna possível criarmos marcadores que realmente trazem um valor semântico à mensagem e, assim, facilitam imensamente o tratamento dos dados.

XML é uma linguagem baseada em texto e é totalmente independente de plataforma. Deste modo, esta linguagem é ideal para a troca de mensagens entre plataformas, que é algo muito comum de ocorrer entre serviços, que provavelmente estão em plataformas distintas.

Com o XML, a informação é organizada de uma forma hierárquica, parecida com uma árvore, onde os marcadores mais internos são imediatamente dependentes do marcador que está no nível superior a estes. Este tipo de estrutura, aliada aos marcadores com valor semântico, permite que os algoritmos responsáveis por analisar a mensagem sejam simples e eficientes.

Além disso, quando utilizamos o XML, criamos mensagens que são facilmente interpretadas tanto por humanos quanto por computadores, e podemos incluir praticamente qualquer informação em qualquer linguagem, devido ao suporte a Unicode, que é um formato de representação de caracteres que permite incluirmos os caracteres ocidentais comuns, caracteres não-romanos, entre outros.

Em contrapartida, o uso de texto plano pode representar um grande impacto no consumo de banda necessário à transmissão de mensagem e trazer problemas de segurança, uma vez que o dado está completamente aberto. Para solucionar o primeiro problema pode-se utilizar compressão HTTP (usualmente compressão *gzip*). Já o segundo pode ser solucionado com o uso de criptografia de dados, criando um canal seguro entre cliente e provedor de serviço (Siqueira Marques de Souza, 2006).

3.5. REST

REST (*Representational State Transfer*) é um estilo de arquitetura baseado no design do protocolo HTTP, construído para servir aplicações em rede, o qual possui diversos mecanismos para representar recursos como código de status, representação de tipo de conteúdo, etc. O REST surgiu como uma alternativa ao SOAP, uma vez que ele é mais leve e transmitir dados diretamente via HTTP.

REST foi mencionado pela primeira vez em 2000, na tese de doutorado de Roy Fielding. A aplicação mais comum de REST é a própria *World Wide Web*, que utilizou REST para o desenvolvimento do HTTP 1.1. O próprio Roy disse que REST enfatiza a escalabilidade das interações do componente, a generalidade das interfaces, a implementação independente de componentes e componentes intermediários para reduzir a latência da interação, forçar a segurança e encapsular sistemas legados" (Plansky, 2014).

O REST possui restrições. Veja (Plansky, 2014):

- **Client-Server**: restrição mais básica para uma aplicação REST. Essa divisão separa a arquitetura e responsabilidades em dois ambientes. O cliente não se preocupa com comunicação entre banco de dados, gerenciamento de cache, etc., e o servidor não se preocupa com interface, experiência do usuário, etc. O servidor espera pelas requisições do cliente executa as requests e envia uma resposta;
- **Stateless**: um mesmo cliente pode enviar várias requisições para o servidor, mas cada uma delas precisa ser independente. Toda requisição deve conter informações o suficiente para que o servidor consiga entendê-la e processá-la;
- **Cacheable**: vários clientes acessam o servidor ao mesmo tempo e, algumas vezes, solicitando os mesmos recursos. As respostas precisam ser "cacheadas", o que evita processamentos desnecessários;
- **Uniform Interface**: um contrato para comunicação client-server. São regras simples, com o objetivo de deixar o componente o mais simples possível.
    - **Identificando o resource**: cada resource possui uma URI específica para poder ser acessado;

    - **Representação do resource**: forma como o resource é enviado de volta para o cliente. Tal representação pode ser enviada por HTML, XML, JSON, etc;
    - **Resposta auto-explicativa**: também é necessário a passagem de metadados na requisição e na resposta, como código HTTP da resposta, Host, Content-Type, etc.
- **Hypermedia**: retorna as informações necessárias na resposta, para que o cliente saiba navegar e ter acesso aos resources da aplicação;
- **Layered System**: a aplicação é composta por camadas, e essas camadas precisam ser o mais simples possível para inserção, edição e remoção. O cliente nunca deve chamar diretamente o servidor da aplicação sem antes passar por um intermediário, ou seja, alguma máquina que faça a interface com os servidores, pois isso garante que o cliente se preocupe apenas com a comunicação com o intermediário, e o intermediário fica responsável por distribuir as requisições nos servidores;
- **Code-On-Demand (Opcional)**: permite que o cliente execute um código sob demanda, ou seja, estenda parte da lógica do servidor para o cliente.

3.6. REST*ful*

Há uma certa confusão quando o assunto é REST / *RESTful*. Ambos representam os mesmos princípios, mas há uma ligeira diferença entre eles (Pires, 2017):

REST: conjunto de princípios de arquitetura.

RESTFUL: capacidade de determinado sistema aplicar os princípios de REST.

Eis alguns casos onde o REST funcionaria bem:

- Situações em que há limitação de recursos e de largura de banda: a estrutura de retorno é em qualquer formato definido pelo desenvolvedor e qualquer navegador pode ser usado. Isso acontece porque o REST usa o padrão de chamadas GET, PUT, POST e DELETE. O REST também pode usar objetos XMLHttpRequest (a base do velho AJAX) que a maioria dos navegadores modernos suportam;

- Operações totalmente sem-estado: se uma operação precisa ser continuada, o REST não será a melhor opção. No entanto, se forem necessárias operações de CRUD *stateless* (Criar, Ler, Atualizar e Excluir), o REST seria a melhor alternativa;
- Situações que exigem cachê: se a informação pode ser armazenada em cache, devido à natureza da operação stateless do REST, esse seria um cenário adequado para a tecnologia.

O REST alavanca aspectos do protocolo HTTP como pedidos GET e POST. Até mesmo esses pedidos são mapeados com CRUD (*Create. Read, Update e Delete*, ou Inserir, Ler, Atualizar e Deletar).

3.7. SOAP

SOAP (*Simple Object Access Protocol*, ou Protocolo Simples de Acesso a Objetos) é um protocolo de troca de informações entre plataformas, via HTTP. É normalmente utilizado em Web Services.

Para suas mensagens, ele se baseia em XML. É independente de plataforma e independente de implementação. Permite baixo acoplamento (pouca ou nenhuma dependência de outros serviços para realizar sua tarefa) entre requisitante e provedor e permite comunicação entre serviços de diferentes organizações (Furtado, Pereira, Azevedo, Baião, Santoro; 2009).

Tal protocolo é composto de três partes: um envelope, que define o conteúdo da mensagem e como processá-la, conjunto de regras codificadas que expressam instâncias dos tipos de dados definidos na aplicação e uma convenção para representar chamadas de eventos e respostas.

SOAP é algo bem maduro, definido e contém uma especificação completa, sendo extremamente útil em situações como as mencionadas abaixo:

- Processamento e chamada assíncronos: se o aplicativo precisa de um nível garantido de confiabilidade e segurança para a troca de mensagens, então o SOAP 1.2 oferece padrões adicionais para este tipo de operação como, por exemplo, o WSRM;
- Contratos formalizados: se os dois lados (fornecedor e consumidor) têm que concordar com o formato de troca de informações, então o SOAP 1.2 fornece especificações rígidas para esse tipo de interação.
- Operações *stateful*: caso o aplicativo precise de informação contextual e gerenciamento de estado com coordenação e segurança, o SOAP 1.2 possui uma especificação adicional em sua estrutura que apoia essa necessidade (segurança, transações, coordenação etc.).

3.8. MIDDLEWARE

É um programa que faz mediação entre softwares. Utilizado para mover ou transportar informações entre programas, ocultando as diferenças de protocolo de comunicação, plataformas e dependências do software.

O papel do *middleware* é interligar diferentes aplicações entre diferentes sistemas operacionais em diferentes computadores. Por ele ser o software de conectividade, consiste em um conjunto de serviços que permite que múltiplos processos executando em uma ou mais máquinas interajam a partir de uma rede.

Middleware de uma forma básica seria quem faz a "ponte" entre os leitores e os sistemas:

Identificador -> Antena -> Leitor -> *Middleware* -> Sistema.

Um exemplo é o *middleware* que é usado para conectar um sistema de banco de dados com um servidor web, isso permite que o usuário solicite informações do banco de dados usando formulários exibidos no próprio navegador web.

Serviços de *middleware* podem ser usados também para:

- Segurança: Verifica um programa cliente, em particular algum componente do sistema para verificar, por exemplo, se o programa cliente e seu usuário são realmente quem eles dizem que são.
- Gerenciamento de transação: Garante que um sistema ou banco de dados não sejam corrompidos se ocorrerem problemas.
- Filas de mensagens: De uma forma bem simples permite que os sistemas de baixo ou alta plataforma possam enviar mensagens entre si. Essas mensagens podem desencadear ações ou transações posteriormente.
- Servidor de aplicativos: um servidor que hospeda uma interface de programação de aplicações (API), que expõe os processos de lógica de negócios, para que outros aplicativos possam usar a lógica e os processos compartilhados.
- Servidor Web: um servidor que é responsável por aceitar solicitações de navegadores da Web, bem como o envio de respostas e conteúdo para esses navegadores – geralmente páginas da Web, tais como documentos HTML e objetos vinculados, como imagens.

3.9. PRONTUÁRIO ELETRÔNICO DO PACIENTE (PEP)

Ter todas as informações de um paciente disponíveis eletronicamente vem sendo perseguido por provedores de saúde. O Prontuário Eletrônico do Paciente (PEP) não é, no entanto, um sistema de informação trivial. Considerando que uma pessoa ao longo de sua vida recebe atendimento em diversas instituições de saúde, sendo que cada uma destas instituições armazena uma parte das informações de saúde do indivíduo, um PEP se torna um sistema distribuído e heterogêneo. Aliando a isto o fato de que a informação em saúde é complexa e pouco estruturada, a construção de um Prontuário Eletrônico que reúna todas as informações de uma pessoa desde o seu nascimento até a sua morte é um dos maiores desafios na área de sistemas de informação em saúde.

Segundo Costa(2001) e Massad (2003), conforme citado por Perim (2011), dentre as informações contidas, podem ser acessados: dados pessoais, histórico familiar, doenças anteriores, hábitos de vida, alergias, imunizações, medicamentos e terapêuticas prescritos, dentre outros.

As principais desvantagens do prontuário em papel são: (1) só pode estar em um lugar ao mesmo tempo – pode não estar disponível ou mesmo ser perdido. (2) conteúdo é livre, variando na ordem, algumas vezes é ilegível, incompleto e com informação ambígua. (3) para estudos científicos, o conteúdo precisa ser transcrito, o que muitas vezes predispõe ao erro. (4) as anotações em papel não podem disparar lembretes e alertas aos profissionais. As vantagens do prontuário em papel e baseado em registro eletrônico da seguinte forma: (1) prontuário em papel: pode ser facilmente carregado; maior liberdade de estilo ao fazer um relatório, facilidade para buscar um dado; não requer treino especial, não "sai do ar" como ocorre com computadores. (2) prontuário eletrônico: simultâneo acesso em locais distintos; legibilidade; variedade na visão do dado; suporte de entrada de dado estruturada; oferece apoio à decisão; apoio a análise de dados; troca eletrônica de dados e compartilha o suporte ao cuidado.

A tecnologia utilizada no projeto de um PEP não é o problema para se fazer a integração de sistemas de saúde e sim, a solução. Dos recentes recursos computacionais disponíveis, que favorecem o desenvolvimento de um prontuário eletrônico, destacam-se a Internet e seu alto poder de conectividade que permite instituições geograficamente distantes, compartilhar dados clínicos e até mesmo chegar aos lares dos pacientes; os softwares de navegação na Internet, pela facilidade de acesso à informação presente na Internet permitem a busca, a pesquisa e a transferência de informação da rede para o microcomputador pessoal de forma rápida e eficiente.

Segundo Massad (2003), conforme citado por Perim (2011): "o PEP foi cirado para que médicos e enfermeiros recordassem de forma sistemática dos fatos e eventos clínicos ocorridos em um indivíduo, de forma que os demais profissionais da saúde envolvidos no processo de atenção pudessem ter acesso a estas informações.

Assim, ele é o mais importante veículo de comunicação entre os membros de uma equipe de saúde responsável pelo atendimento".

No projeto proposto, quando o médico solicitar exames ao paciente ele mesmo envia pelo sistema quais as clínicas e quais exames serão feitos, assim as clínicas já ficam cientes que o exame será feito ali. O paciente precisa levar apenas seus documentos pessoais e na clínica ou hospital já conseguem ter acesso aos exames que devem ser realizados e qual o médico que solicitou. Após a realização de exames as próprias clínicas e hospitais enviam pelo sistema o resultado do exame ao médico que solicitou, assim o paciente não precisa guardar os resultados e ter que esperar o retorno para levar ao médico. E finalmente quando retornar ao médico ele consegue ver todos os exames que foram realizados, quais os locais e os resultados mais facilmente, sem precisar de vários encaminhamentos e nem de perder ou esquecer algum.

4. TECNOLOGIAS E FERRAMENTAS A SEREM UTILIZADAS

4.1. Astah

Astah Community é uma ferramenta gratuita voltada para a modelagem de diagramas UML (*Unified Modelling Language*). A ferramenta *Astah Community* é conhecida por sua praticidade e simplicidade em elaborar diagramas, como por exemplo: diagramas de classe, caso de uso, sequência, atividade, comunicação, máquina de estado, componentes, implantação, estrutura decomposição, objetos e pacotes.

Figura 1 - Astah

Fonte: astah.net

4.2. Python Django

Django é um *framework* gratuito e de código aberto para a criação de aplicações web, escrito em *Python*. É um framework web, ou seja, é um conjunto de componentes que ajuda a desenvolver para a web de forma mais rápida e mais fácil.

Figura 2 - Python Django

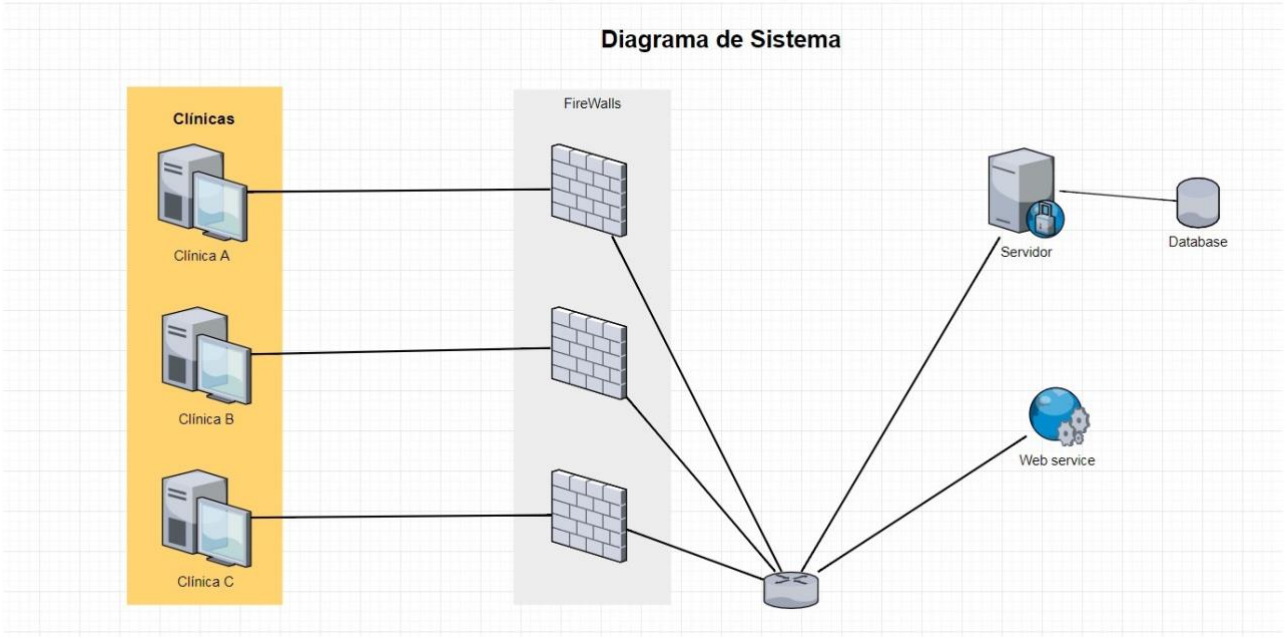
Fonte: theodo.com

5. ANÁLISE E PROJETO

5.1. DIAGRAMA DE SISTEMA

Representação das interfaces entre o sistema e as entidades externas.

Figura 3- Diagrama de Sistema



Fonte: Autores

Elementos do diagrama (Figura 3)

Database: Banco de dados do sistema, contendo a coleção de dados que são acessadas pelas clínicas.

Servidor: Programa que provê a funcionalidade do sistema que é acessado pelas clínicas.

Web Service: Um serviço oferecido pelo Servidor Web para outros dispositivos eletrônicos, como computadores e celulares. Nesse caso um serviço de prontuário médico é oferecido.

Firewall: Sistema de seguraça de rede que monitora e controla o tráfego de rede baseado em regras de segurança pré-determinadas.

Clínica: são os clients, isto é, computadores que acessam informações do sistema de prontuário que está armazenado no servidor.

5.2. Lista de Eventos

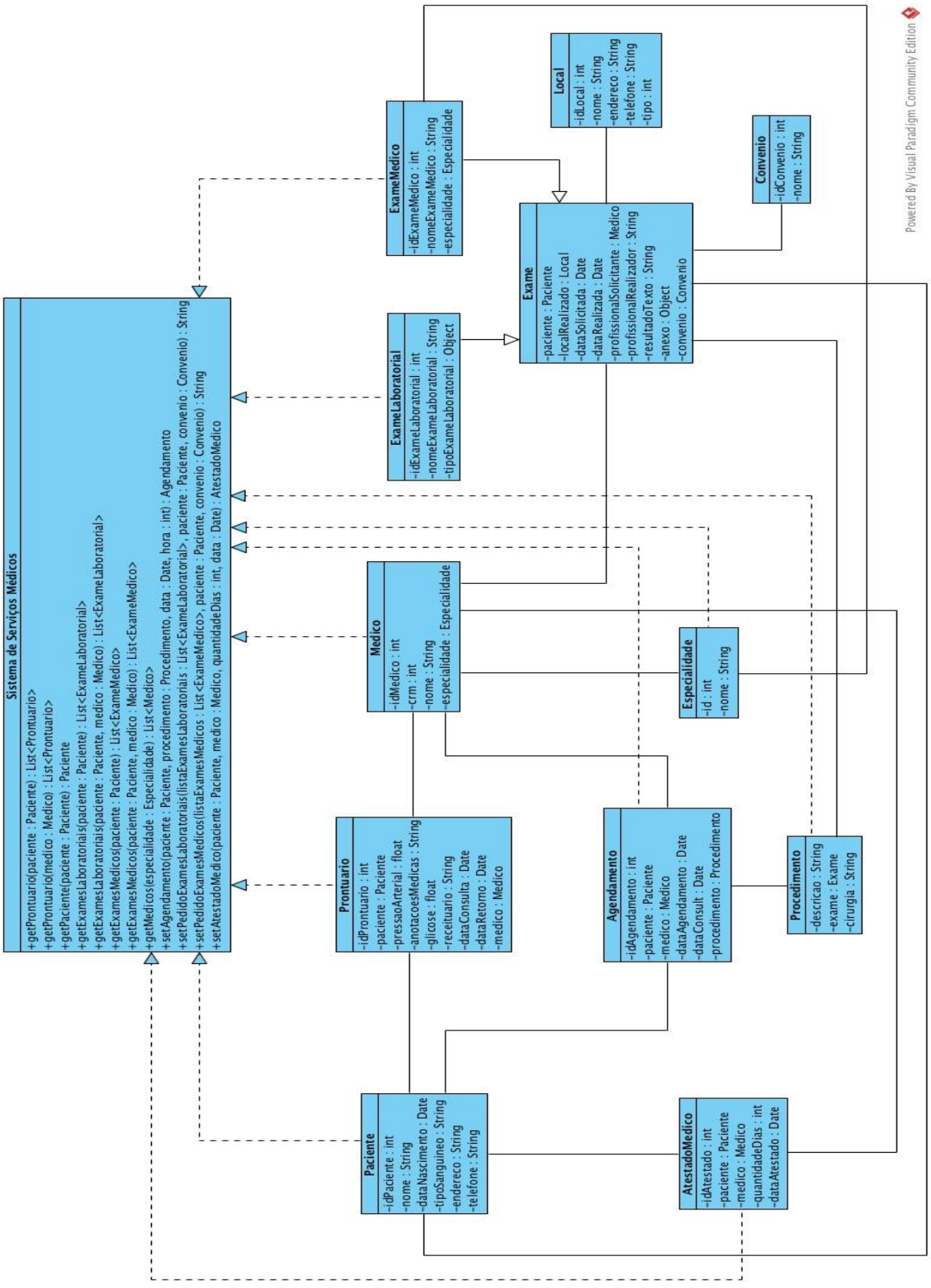
Tabela 1 - Lista de Eventos

Nº	Descrição	Evento	Caso de Uso	Resposta
1	Sistema recupera Prontuários por Paciente	dadosPaciente	Recuperar Prontuários por Paciente	R01
2	Sistema recupera Prontuários por Médico	dadosMedico	Recuperar Prontuários por Médico	R02
3	Sistema recupera Paciente	dadosPaciente	Recuperar Paciente	R03
4	Sistema lista Exames Laboratoriais por Paciente	dadosPaciente	Listar Exames Laboratoriais por Paciente	R04
5	Sistema lista Exames Laboratoriais por Paciente e Médico	dadosBusca	Listar Exames Laboratoriais por Paciente e Médico	R05
6	Sistema lista Exames Médicos por Paciente	dadosPaciente	Listar Exames Médicos por Paciente	R06
7	Sistema lista Exames Médicos por Paciente e Especialidade	dadosBusca	Listar Exames Médicos por Paciente e Especialidade	R07
8	Sistema lista Exames Médicos por Paciente e Médico	dadosBusca	Listar Exames Médicos por Paciente e Médico	R08
9	Sistema lista Médicos por Especialidade	dadosEspecialidade	Listar Médicos por Especialidade	R09
10	Sistema realiza Agendamento	dadosAgendamento	Realizar Agendamento	R10
11	Sistema realiza Pedido de Exame Laboratorial	dadosPedido	Realizar Pedido de Exame Laboratorial	R11
12	Sistema realiza Pedido de Exame Médico	dadosPedido	Realizar Pedido de Exame Médico	R12
13	Sistema obtêm Atestado Médico	dadosAtestado	Obter Atestado Médico	R13

Fonte: Autores

5.3. Diagrama de Classes

Figura 4 - Diagrama de Classes



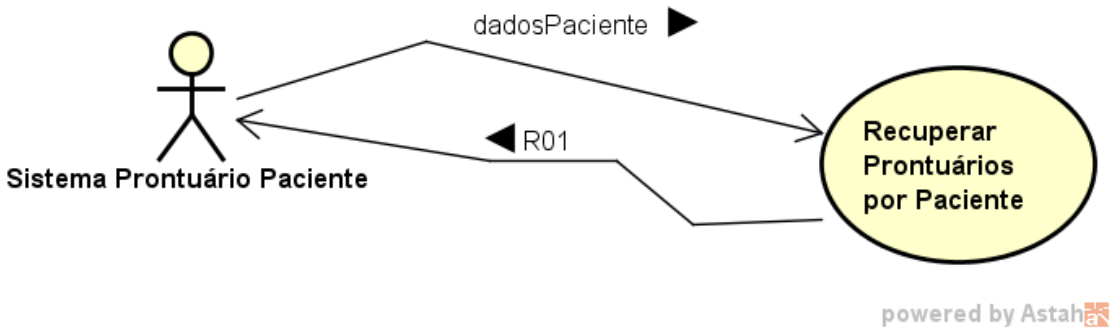
Fonte: Autores

5.4. Casos de Uso

5.4.1. Caso de Uso - Sistema Recupera Prontuários por Paciente

5.4.1.1. Diagrama de Caso de Uso

Figura 5 - Caso de Uso - Sistema Recupera Prontuários por Paciente



Fonte: Autores

5.4.1.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Recuperar Prontuários por Paciente

Curso Normal

1 – Sistema chama funcionalidade Recuperar Prontuário passando dados do Paciente;

2 – SSM valida paciente;

3 – SSM seleciona lista de Prontuários a partir do Paciente;

4 – SSM valida dados do Médico;

5 – SSM retorna lista de Prontuários

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Não encontrou Prontuários

3.1 – SSM retorna mensagem “Nenhum Prontuário Encontrado”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Médico Inválido

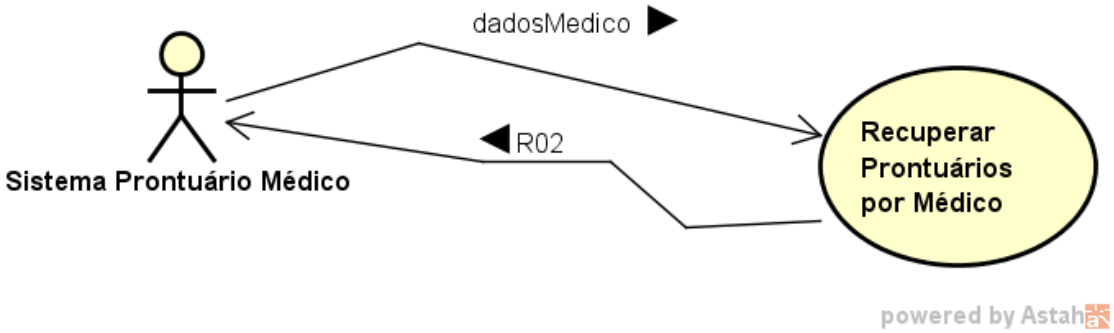
4.1 – SSM retorna mensagem “Médico Inválido”

4.2 - Finalizar Caso de Uso

5.4.2. Caso de Uso - Recuperar Prontuários por Médico

5.4.2.1. Diagrama de Caso de Uso

Figura 6 - Caso de Uso - Recuperar Prontuários por Médico



Fonte: Autores

5.4.2.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Recupera Prontuários por Médico

Curso Normal

1 – Sistema chama funcionalidade Recuperar Prontuário passando dados do Médico

2 – SSM valida médico;

3 – SSM seleciona Prontuários por Médico;

4 – SSM valida Paciente;

5 – SSM retorna lista de Prontuários

Curso Alternativo 01

Caso 2: Médico Inválido

2.1 – SSM retorna mensagem “Médico inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Nenhum Prontuário Encontrado

3.1 – SSM retorna mensagem “Nenhum Prontuário Encontrado”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Paciente Inválido

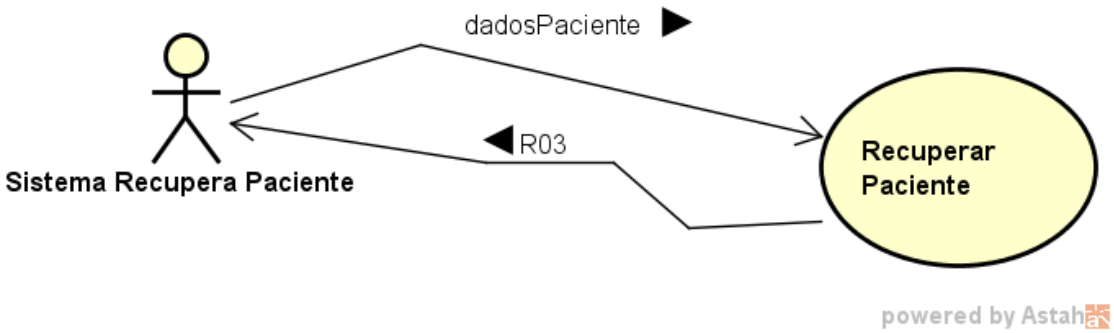
4.1 – SSM retorna mensagem “Paciente Inválido”

4.2- Finalizar Caso de Uso

5.4.3. Caso de Uso - Sistema Recupera Paciente

5.4.3.1. Diagrama de Caso de Uso

Figura 7 - Caso de Uso - Sistema Recupera Paciente



Fonte: Autores

5.4.3.2. Descrição do Curso Normal e Alternativo do Caso de Uso –Recuperar Paciente

Curso Normal

1 – Sistema chama funcionalidade Recuperar Paciente passando dados do Paciente;

2 – SSM valida Paciente;

3 – SSM retorna os dados do Paciente

Curso Alternativo 01

Caso 2: Paciente Inválido

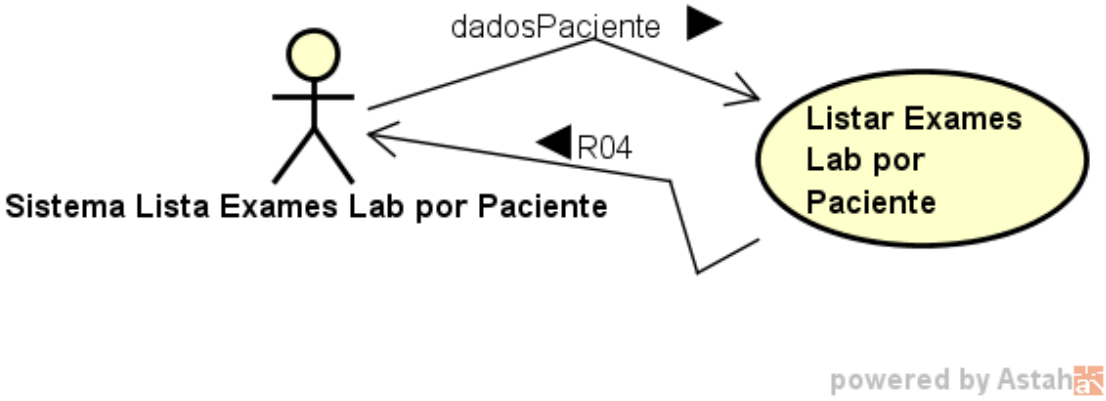
2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

5.4.4. Caso de Uso - Sistema Lista Exames Lab por Paciente

5.4.4.1. Diagrama de Caso de Uso

Figura 8 - Caso de Uso - Sistema Lista Exames Lab por Paciente



Fonte: Autores

5.4.4.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Listar Exames Lab por Paciente

Curso Normal

1 – Sistema chama funcionalidade getExamesLaboratoriais();

2 – SSM valida paciente por selectById(paciente:Paciente);

3 – SSM seleciona paciente numa lista de exames laboratoriais;

4 – SSM seleciona médico por id;

5 – SSM retorna dados do profissional solicitante;

6 – SSM seleciona local por id;

7 – SSM retorna dados do local;

8 – SSM seleciona convênio por id;

9 – SSM retorna dados do convênio;

10 – SSM retorna lista de exames laboratoriais

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Nenhum Exame Laboratorial Encontrado

3.1 – SSM retorna mensagem “Nenhum Exame Laboratorial Encontrado”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Médico Inválido

4.1 – SSM retorna mensagem “Médico Inválido”

4.2 - Finalizar Caso de Uso

Curso Alternativo 04

Caso 5: Local Inválido

5.1 – SSM retorna mensagem “Local Inválido”

5.2 - Finalizar Caso de Uso

Curso Alternativo 05

Caso 6: Convênio Inválido

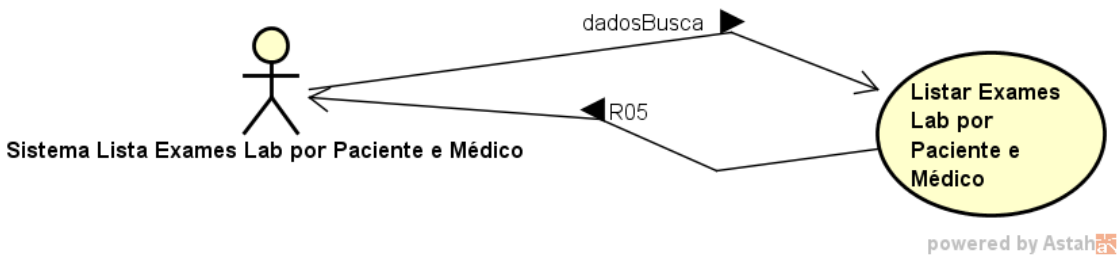
6.1 – SSM retorna mensagem “Convênio Inválido”

6.2 - Finalizar Caso de Uso

5.4.5. Caso de Uso - Sistema Lista Exames Lab por Paciente e Médico

5.4.5.1. Diagrama de Caso de Uso

Figura 9 - Caso de Uso - Sistema Lista Exames Lab por Paciente e Médico



Fonte: Autores

5.4.5.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Listar Exames Laboratoriais por Paciente e Médico

Curso Normal

1 – Sistema chama funcionalidade Recuperar Exames Laboratoriais passando dados do Paciente e do Médico;

2 – SSM valida Paciente;

3 – SSM valida Médico;

4 – SSM seleciona exames laboratoriais por Paciente e Médico;

5 – SSM valida Local;

6 – SSM valida Convênio;

7 – SSM retorna lista de Exames Laboratoriais

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Médico Inválido

3.1 – SSM retorna mensagem “Médico Inválido”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Nenhum Exame Laboratorial Encontrado

4.1 – SSM retorna mensagem “Nenhum Exame Laboratorial Encontrado”

4.2 - Finalizar Caso de Uso

Curso Alternativo 04

Caso 5: Local Inválido

5.1 – SSM retorna mensagem “Local Inválido”

5.2 - Finalizar Caso de Uso

Curso Alternativo 05

Caso 6: Convênio Inválido

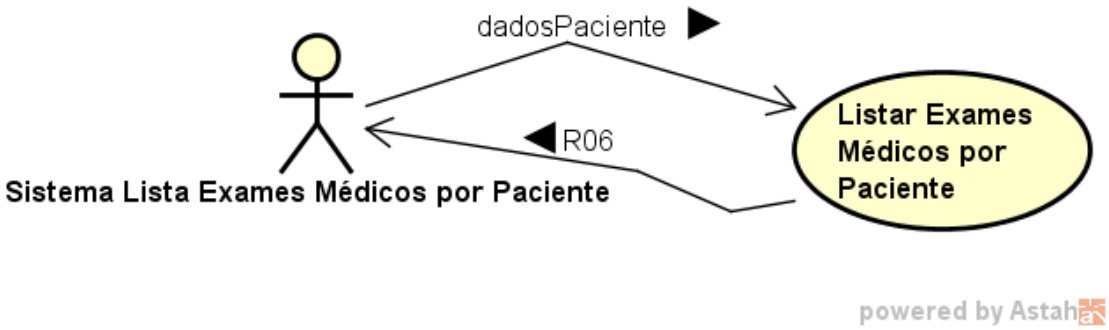
6.1 – SSM retorna mensagem “Convênio Inválido”

6.2 - Finalizar Caso de Uso

5.4.6. Caso de Uso - Sistema Lista Exames Médicos por Paciente

5.4.6.1. Diagrama de Caso de Uso

Figura 10 - Caso de Uso - Sistema Lista Exames Médicos por Paciente



Fonte: Autores

5.4.6.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Listar Exames Médicos por Paciente

Curso Normal

1 – Sistema chama funcionalidade Recuperar Exames Médicos passando dados do Paciente;

2 – SSM valida paciente;

3 – SSM seleciona Exames Médicos por Paciente;

4 – SSM valida Médico;

5 – SSM valida Local;

6 – SSM valida Convênio;

7 – SSM valida Especialidade;

8 - SSM retorna lista de Exames Médicos;

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Nenhum Exame Médico Encontrado

3.1 – SSM retorna mensagem “Nenhum Exame Médico Encontrado”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Médico Inválido

4.1 – SSM retorna mensagem “Médico Inválido”

4.2 - Finalizar Caso de Uso

Curso Alternativo 04

Caso 5: Local Inválido

5.1 – SSM retorna mensagem “Local Inválido”

5.2 - Finalizar Caso de Uso

Curso Alternativo 05

Caso 6: Convênio Inválido

6.1 – SSM retorna mensagem “Convênio Inválido”

6.2 - Finalizar Caso de Uso

Curso Alternativo 06

Caso 6: Especialidade Inválida

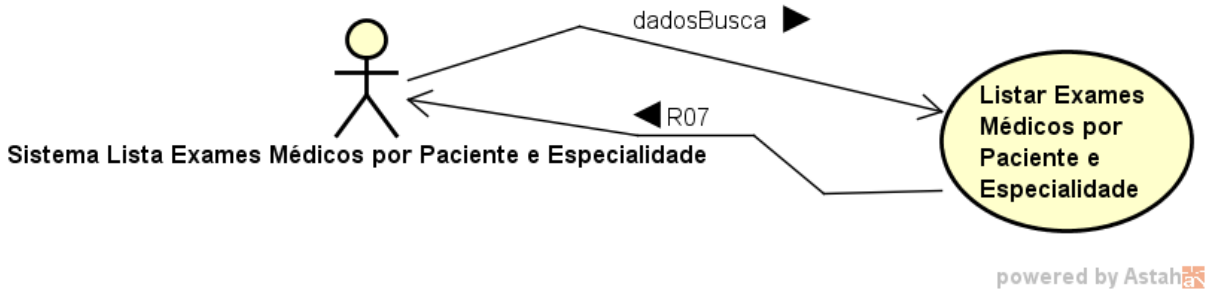
6.1 – SSM retorna mensagem “Especialidade Inválida”

6.2 - Finalizar Caso de Uso

5.4.7. Caso de Uso - Sistema Lista Exames Médicos por Paciente e Especialidade

5.4.7.1. Diagrama de Caso de Uso

Figura 11 - Caso de Uso - Sistema Lista Exames Médicos por Paciente e Especialidade



Fonte: Autores

5.4.7.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Listar Exames Médicos por Paciente e Especialidade

Curso Normal

1 – Sistema chama funcionalidade Recuperar Exames Médicos passando dados do Paciente e da Especialidade;

2 – SSM valida Paciente;

3 – SSM valida Especialidade;

4 – SSM seleciona Exames Médicos por Paciente;

5 – SSM valida Médico;

6 – SSM valida Local;

7 – SSM valida Convênio;

8 - SSM retorna lista de Exames Médicos;

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Especialidade Inválida

3.1 – SSM retorna mensagem “Especialidade Inválida”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Nenhum Exame Médico Encontrado

4.1 – SSM retorna mensagem “Nenhum Exame Médico Encontrado”

4.2 - Finalizar Caso de Uso

Curso Alternativo 04

Caso 5: Médico Inválido

5.1 – SSM retorna mensagem “Médico Inválido”

5.2 - Finalizar Caso de Uso

Curso Alternativo 05

Caso 6: Local Inválido

6.1 – SSM retorna mensagem “Local Inválido”

6.2 - Finalizar Caso de Uso

Curso Alternativo 06

Caso 7: Convênio Inválido

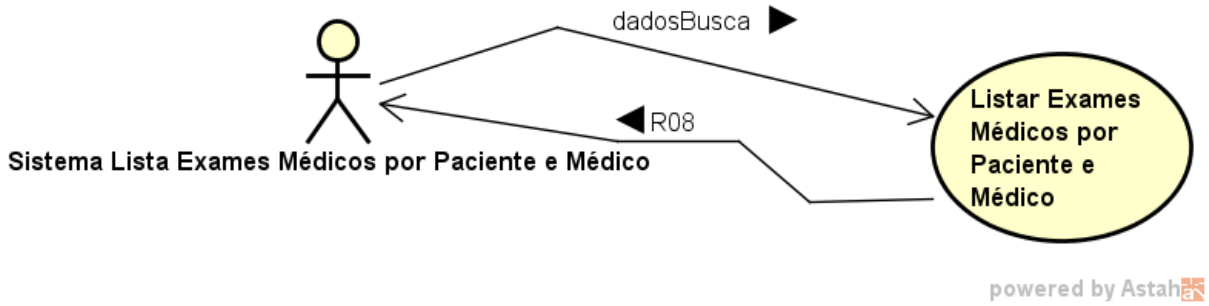
7.1 – SSM retorna mensagem “Convênio Inválido”

7.2 – Finalizar Caso de Uso

5.4.8. Caso de Uso - Sistema Lista Exames Médicos por Paciente e Médico

5.4.8.1. Diagrama de Caso de Uso

Figura 12 - Caso de Uso - Sistema Lista Exames Médicos por Paciente e Médico



Fonte: Autores

5.4.8.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Listar Exames Médicos por Paciente e Médico

Curso Normal

1 – Sistema chama funcionalidade Recuperar Exames Médicos passando dados do Paciente e do Médico;

2 – SSM valida Paciente;

3 – SSM valida Médico;

4 – SSM seleciona Exames Médicos por Paciente;

5 – SSM valida Local;

6 – SSM valida Convênio;

7 – SSM valida Especialidade;

8 - SSM retorna lista de Exames Médicos;

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Médico Inválido

3.1 – SSM retorna mensagem “Médico Inválido”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Nenhum Exame Médico Encontrado

4.1 – SSM retorna mensagem “Nenhum Exame Médico Encontrado”

4.2 - Finalizar Caso de Uso

Curso Alternativo 04

Caso 5: Local Inválido

5.1 – SSM retorna mensagem “Local Inválido”

5.2 - Finalizar Caso de Uso

Curso Alternativo 05

Caso 6: Convênio Inválido

6.1 – SSM retorna mensagem “Convênio Inválido”

6.2 - Finalizar Caso de Uso

Curso Alternativo 06

Caso 6: Especialidade Inválida

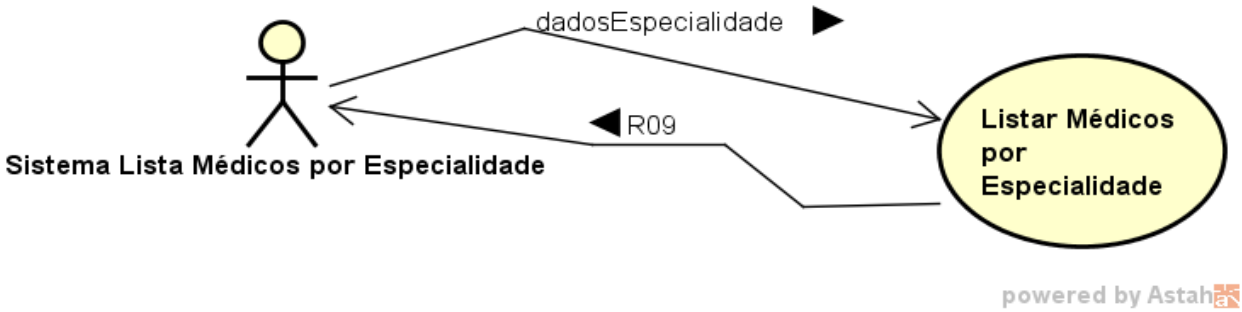
7.1 – SSM retorna mensagem “Especialidade Inválida”

7.2 - Finalizar Caso de Uso

5.4.9. Caso de Uso - Sistema Lista Médicos por Especialidade

5.4.9.1. Diagrama de Caso de Uso

Figura 13 - Caso de Uso - Sistema Lista Médicos por Especialidade



Fonte: Autores

5.4.9.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Listar Médicos por Especialidade

Curso Normal

1 – Sistema chama funcionalidade Listar Médicos passando dados da Especialidade;

2 – SSM valida Especialidade

3 – SSM retorna lista de Médicos por Especialidade

Curso Alternativo 01

Caso 2: Especialidade Inválida

2.1 – SSM retorna mensagem “Especialidade Inválida”

2.2 - Finalizar Caso de Uso

Caso Alternativo 02

Caso 3: Nenhum Médico Encontrado

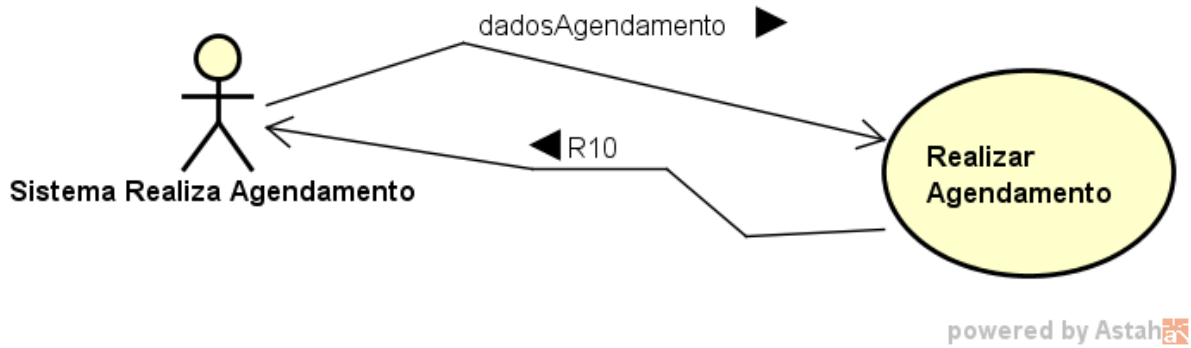
3.1 – SSM retorna mensagem “Nenhum Médico Encontrado”

3.2 - Finalizar Caso de Uso

5.4.10. Caso de Uso - Sistema Realiza Agendamento

5.4.10.1. Diagrama de Caso de Uso

Figura 14 - Caso de Uso - Sistema Realiza Agendamento



Fonte: Autores

5.4.10.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Realizar Agendamento

Curso Normal

1 – Sistema chama funcionalidade Realizar Agendamento passando dados do Agendamento;

2 – SSM valida Paciente;

3 – SSM valida Procedimento;

4 – SSM insere Agendamento

5 – SSM retorna dados do Agendamento

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Procedimento Inválido

3.1 – SSM retorna mensagem “Procedimento Inválido”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Não Foi Possível Realizar o Agendamento

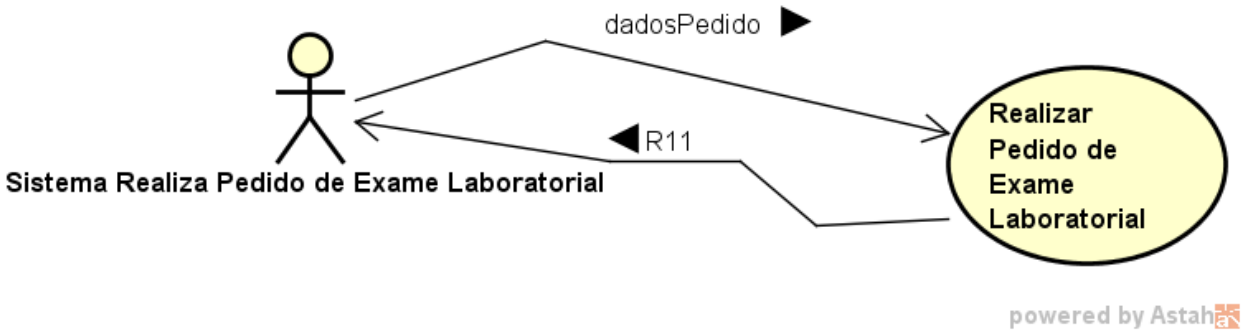
3.1 – SSM retorna mensagem “Não Foi Possível Realizar o Agendamento”

3.2 - Finalizar Caso de Uso

5.4.11. Caso de Uso - Sistema Realiza Pedido de Exame Laboratorial

5.4.11.1. Diagrama de Caso de Uso

Figura 15 - Caso de Uso - Sistema Realiza Pedido de Exame Laboratorial



Fonte: Autores

5.4.11.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Realizar Pedido de Exame Laboratorial

Curso Normal

1 – Sistema chama funcionalidade Realizar Pedido de Exames Laboratoriais passando dados do Pedido

2 – SSM valida Paciente;

3 – SSM valida Convênio;

4 – SSM valida lista de Exames Laboratoriais;

5 – SSM insere Pedidos de Exames;

6 – SSM retorna mensagem “Pedido realizado”

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Convênio Inválido

3.1 – SSM retorna mensagem “Convênio Inválido”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Lista de Exames Laboratoriais Inválida

4.1 – SSM retorna mensagem “Lista de Exames Inválida”

4.2 - Finalizar Caso de Uso

Curso Alternativo 04

Caso 5: Não Foi Possível Realizar o Pedido

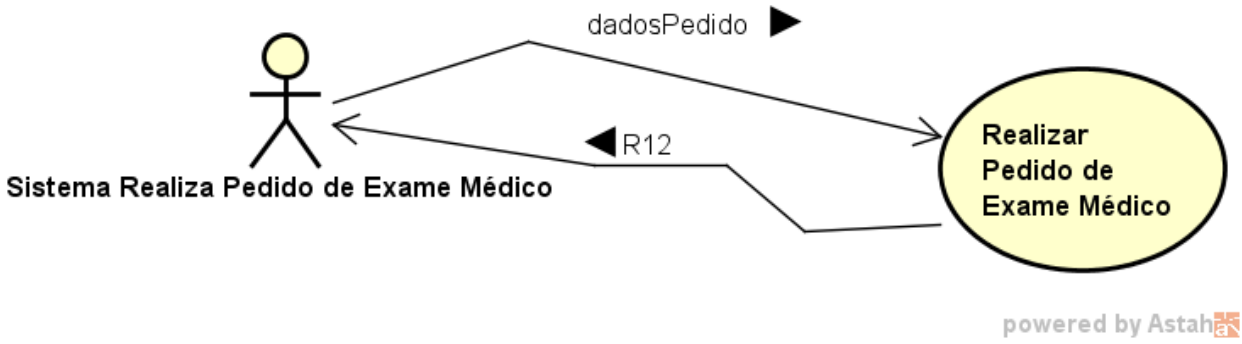
5.1 – SSM retorna mensagem “Não Foi Possível Realizar o Pedido”

5.2 - Finalizar Caso de Uso

5.4.12. Caso de Uso - Sistema Realiza Pedido de Exame Médico

5.4.12.1. Diagrama de Caso de Uso

Figura 16 - Caso de Uso - Sistema Realiza Pedido de Exame Médico



Fonte: Autores

5.4.12.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Realizar Pedido de Exame Médico

Curso Normal

1 – Sistema chama funcionalidade Realizar Pedido de Exames Médicos passando dados do Pedido

2 – SSM valida Paciente;

3 – SSM valida Convênio;

4 – SSM valida lista de Exames Médicos;

5 – SSM insere Pedidos de Exames;

6 – SSM retorna mensagem “Pedido realizado”

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Convênio Inválido

3.1 – SSM retorna mensagem “Convênio Inválido”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Lista de Exames Médicos Inválida

4.1 – SSM retorna mensagem “Lista de Exames Inválida”

4.2 - Finalizar Caso de Uso

Curso Alternativo 04

Caso 5: Não Foi Possível Realizar o Pedido

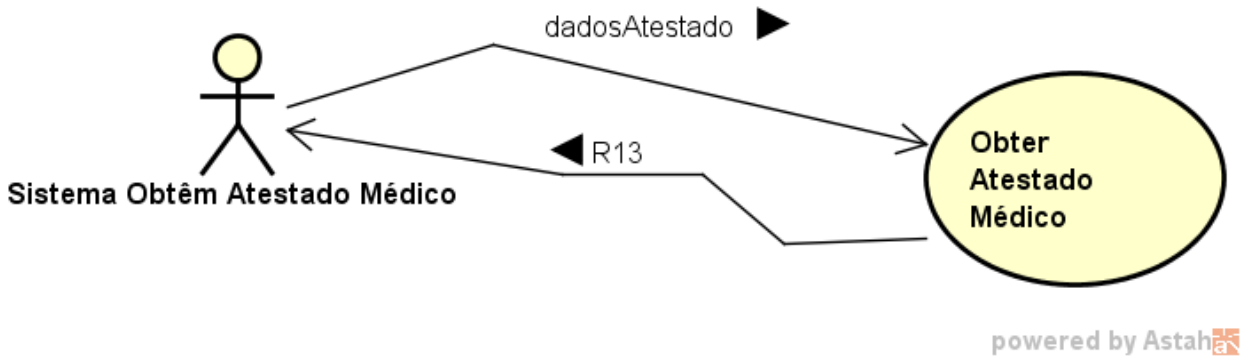
5.1 – SSM retorna mensagem “Não Foi Possível Realizar o Pedido”

5.2 - Finalizar Caso de Uso

5.4.13. Caso de Uso - Sistema Obtêm Atestado Médico

5.4.13.1. Diagrama de Caso de Uso

Figura 17 - Caso de Uso - Sistema Obtêm Atestado Médico



Fonte: Autores

5.4.13.2. Descrição do Curso Normal e Alternativo do Caso de Uso – Obter Atestado Médico

Curso Normal

1 – Sistema chama funcionalidade Registrar Atestado Médico

2 – SSM valida Paciente;

3 – SSM valida Médico;

4 – SSM insere Atestado Médico

5 – SSM retorna dados do atestado

Curso Alternativo 01

Caso 2: Paciente Inválido

2.1 – SSM retorna mensagem “Paciente Inválido”

2.2 - Finalizar Caso de Uso

Curso Alternativo 02

Caso 3: Médico Inválido

3.1 – SSM retorna mensagem “Médico Inválido”

3.2 - Finalizar Caso de Uso

Curso Alternativo 03

Caso 4: Não Foi Possível Obter Atestado Médico

3.1 – SSM retorna mensagem “Não Foi Possível Obter Atestado Médico”

3.2 - Finalizar Caso de Uso

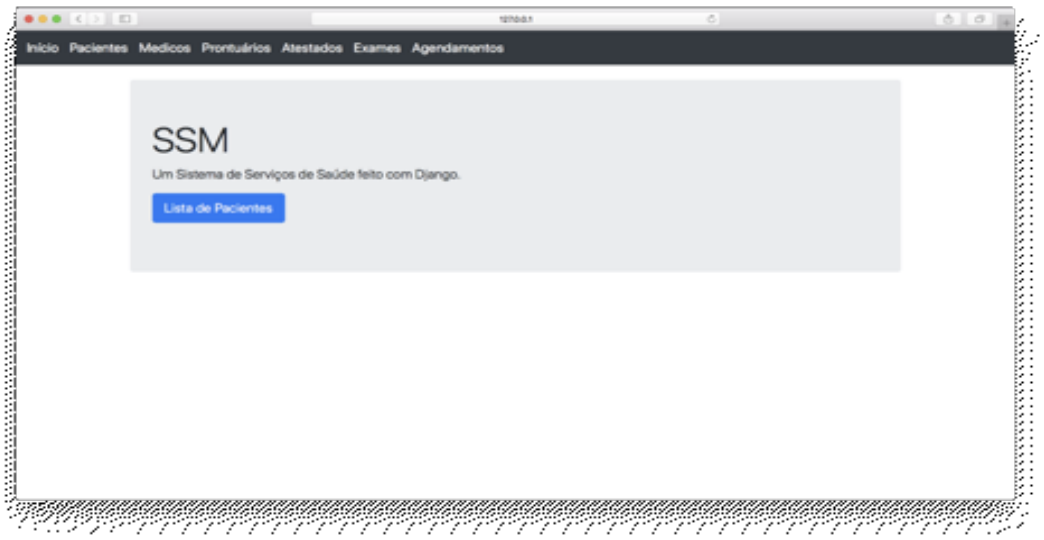
6. RESULTADOS OBTIDOS

Foi desenvolvido um sistema para atender o objetivo proposto com base nas premissas descritas nos itens 5.2, 5.3 e 5.4.

Conforme a imagem abaixo, na barra de atalhos na parte superior do sistema pode-se observar cada parte do sistema que atende as premissas dos itens 5.2 e 5.4. A seguir uma descrição de cada item:

- Início: volta para a página inicial do sistema.

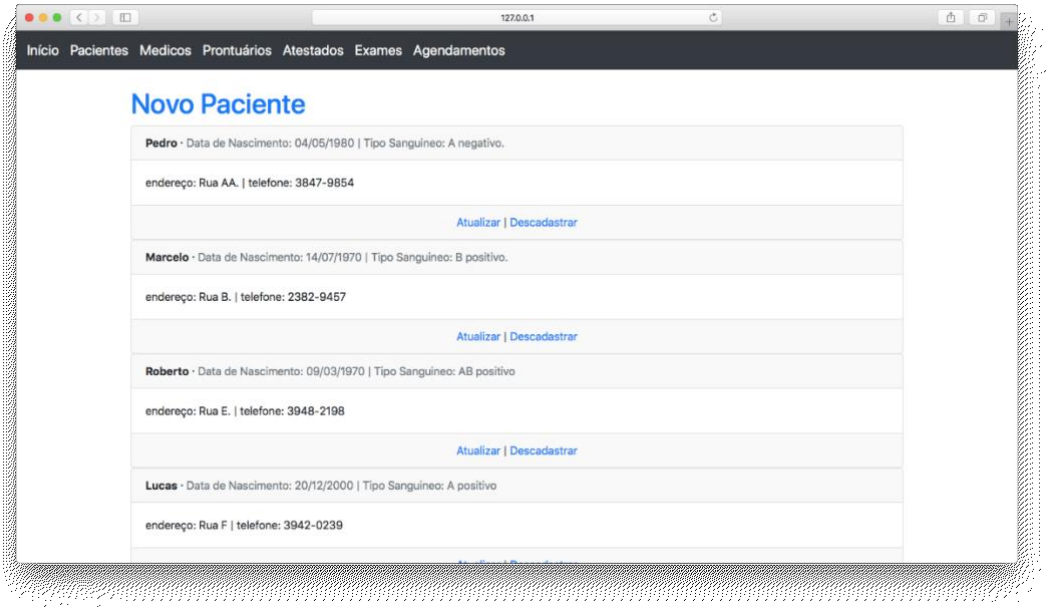
Figura 18 - Tela Inicial do Sistema



Fonte: Autores

- Pacientes: página com lista de pacientes cadastrados no sistema. Nela é possível ao médico cadastrar um novo paciente ou atualizar o cadastro de um paciente.

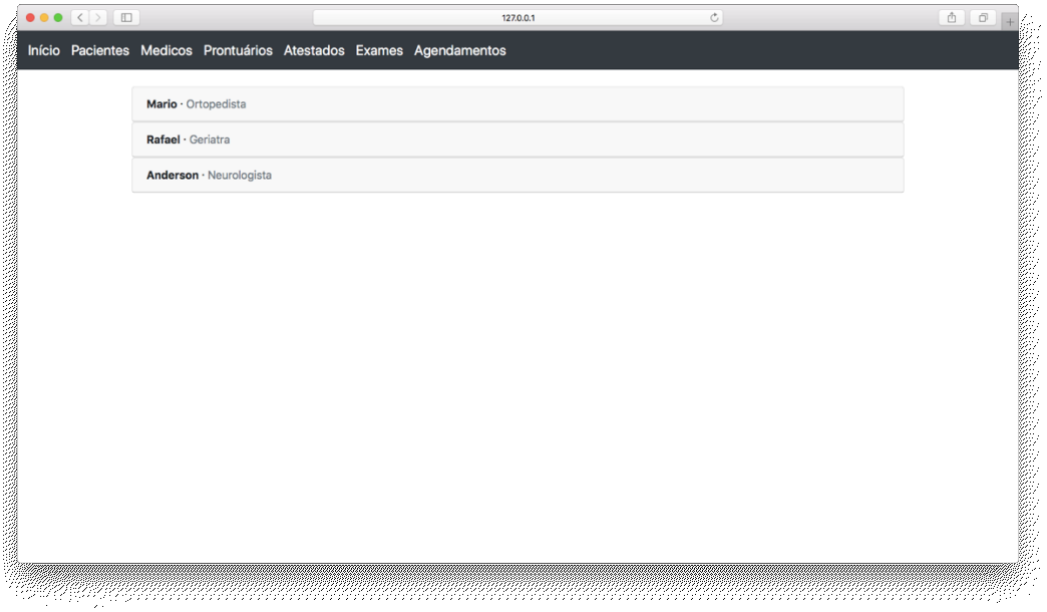
Figura 19 - Tela de Pacientes



Fonte: Autores

- Médicos: página com lista dos médicos cadastrados no sistema e suas especialidades.

Figura 20 - Tela de Médicos

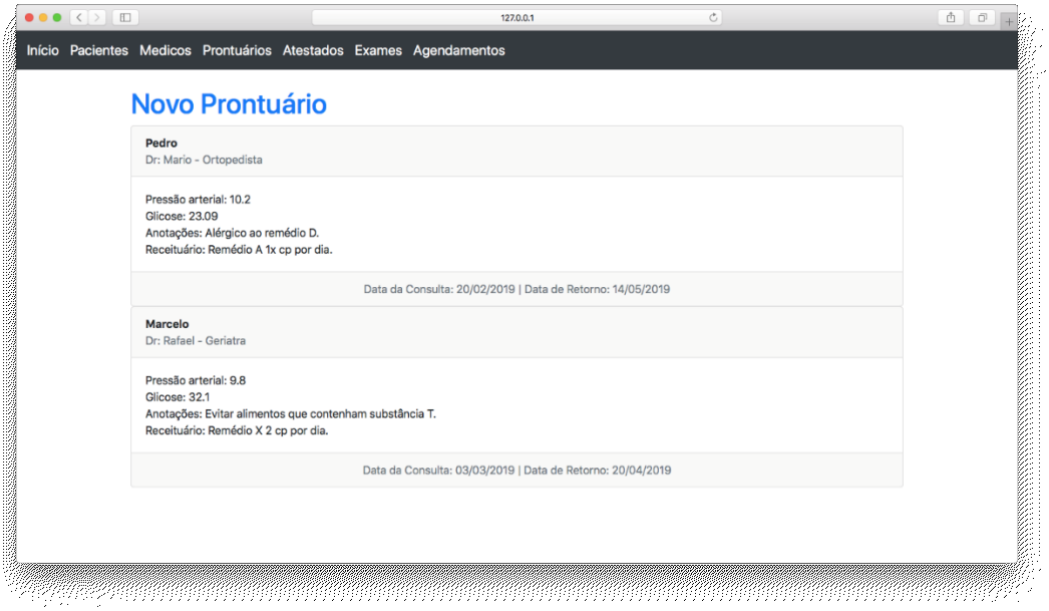


Fonte: Autores

- Prontuários: página com lista de prontuários. Cada prontuário contém o paciente sendo atendido, o médico atendente, dados do paciente como pressão, taxa de glicose,

anotações feitas pelo médico, receituário, data da consulta e data de retorno. Nessa página também é possível ao médico adicionar um novo prontuário.

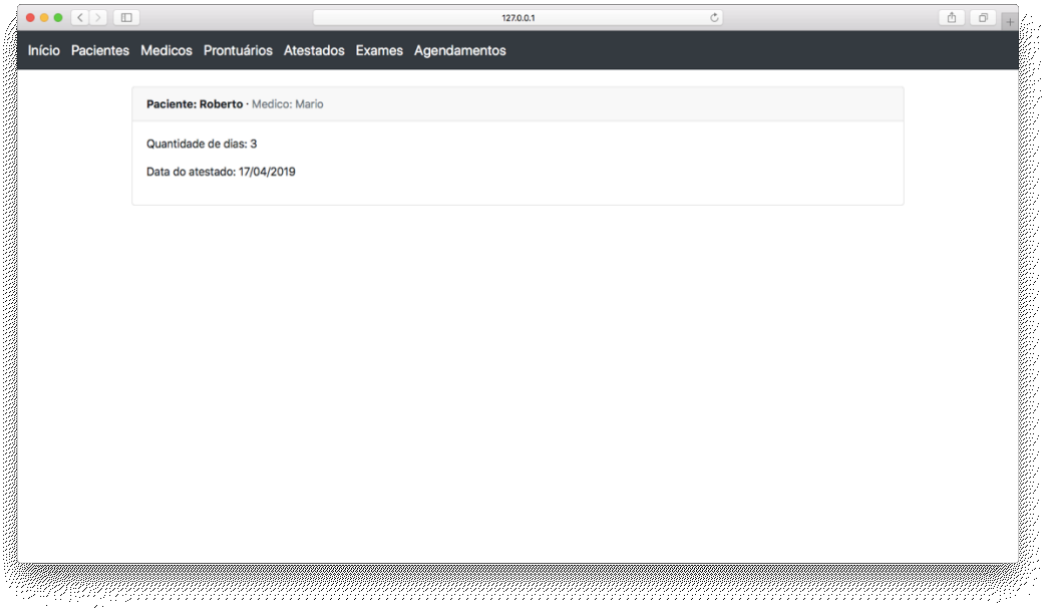
Figura 21- Tela de Prontuários



Fonte: Autores

- Atestados: página com lista de atestados. Lista de pacientes que pediram um atestado. Ao clicar em um paciente é possível ver detalhes do atestado como o paciente que pediu, o médico que deu o atestado, a data do atestado e a quantidade de dias que esse atestado vale. É possível ao médico adicionar um novo atestado nessa página. O atestado é impresso pelo próprio médico e entregue ao paciente.

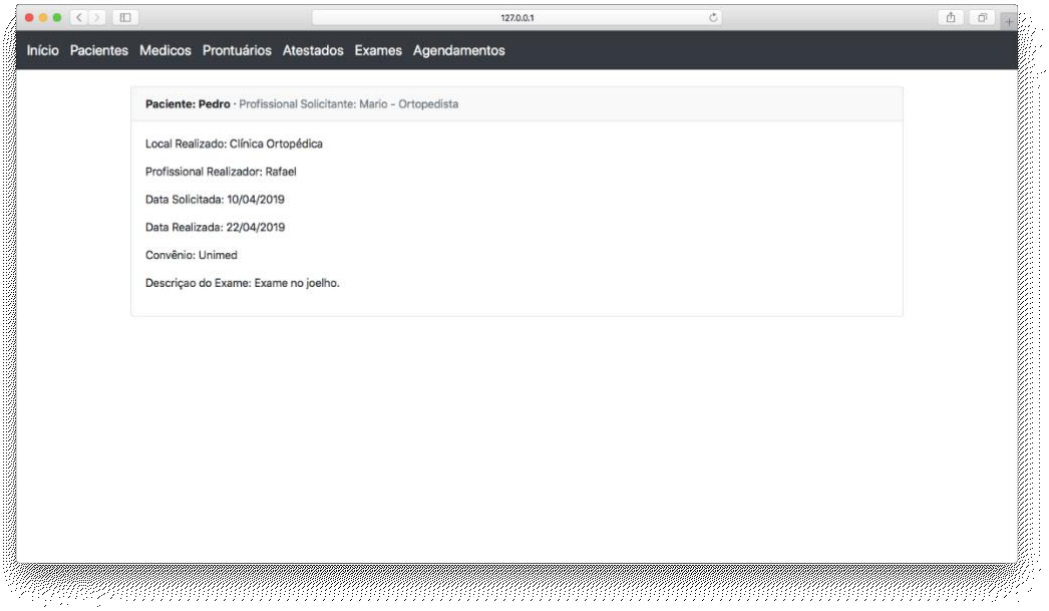
Figura 22 - Tela de Detalhes de um Atestado



Fonte: Autores

- Exames: página com lista de exames marcados e realizados. Nela é possível marcar um novo exame. Também é possível clicar em exames listados para saber detalhes como quem é o paciente, o médico solicitante, o local onde foi ou será realizado o exame, o médico que realizou ou realizará o exame, data em que foi solicitado o exame, data em que foi realizado o exame, o convênio do paciente e a descrição do exame feita pelo médico que realizou o exame.

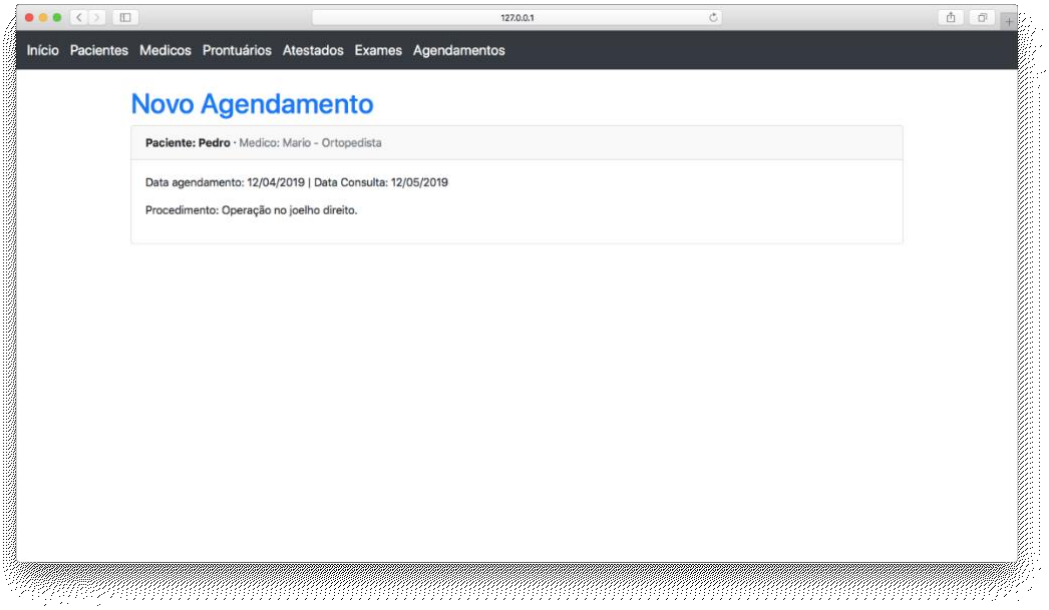
Figura 23- Tela de Detalhes de um Exame



Fonte: Autores

- Agendamentos: página com lista de agendamentos. Nela é possível ver detalhes como o paciente que agendou, o médico que atendeu esse paciente, a data do agendamento, a data da consulta e a descrição do procedimento feito na consulta. Também é possível fazer um novo agendamento.

Figura 24- Tela de Agendamentos



Fonte: Autores

Infelizmente não foi possível a implementação de todas as propostas desse projeto no sistema desenvolvido, que foram os itens 5, 7 e 8 da tabela no item 5.2.

Conforme dito no texto, o sistema em cada clínica pode estar em diferentes linguagens e para tornar possível a comunicação entre esses sistemas, é usado XML. A seguir um exemplo de modelo em código XML usado pelo sistema quando é pedido dados do paciente ao sistema:

Figura 25 - Exemplo de Estrutura de um XML

paciente.xml

```xml
<?xml version="1.0"?>

<Prontuario>
    <id>"pegarProntuario"</id>
    <paciente>
        <id>1</id>
        <nome>"Pedro"</nome>
    </paciente>
    <solicitante>
        <nome>"Mario"</nome>
        <unidade>"Clinica Mais"</unidade>
    </solicitante>
    <listaProntuario>
        <item>
            <id>1</id>
            <pressaoArterial>10.2</pressaoArterial>
            <glicose>23.09</glicose>
            <anotacoes>"Alérgico ao remédio D."</anotacoes>
            <receituario>"Remédio A 1x cp por dia."</receituario>
            <dataDaConsulta>2018-02-20</dataDaConsulta>
            <dataDeRetorno>2018-05-14</dataDeRetorno>
        </item>
    </listaProntuario>
</Prontuario>>

```

Fonte: Autores

7. CONCLUSÃO

Nos dias de hoje, com o aumento da procura por consultórios médicos, consultórios e hospitais vem pensando em diversas formas de facilitar o atendimento do paciente com o médico. O projeto tem essa finalidade: facilitar as relações entre pacientes e médicos. Dessa forma, não há necessidade de papelada excessiva, não se corre o risco de perder informação, facilita ver o exame assim que estiver pronto e sempre que um médico precisar de informações, poderá ter acesso a elas rapidamente.

Isso é possível graças ao *Web Service*, cuja função é enviar respostas que foram solicitadas, ou seja, quando alguém solicita uma informação, ele entra em ação e envia a resposta desejada. Web Service ajuda também em várias outras situações, como por exemplo quando se tem um site que precisa pegar informações de outro site.

Usando *Web Service* e outros componentes facilitaram a execução do projeto, e assim conseguimos cumprir o objetivo de facilitar alguns processos em clínicas e hospitais, permitindo que os atendimentos que se relacionam a esses serviços possam ser realizados de forma mais ágil.

7.1. TRABALHOS FUTUROS

O sistema possui muita margem para desenvolvimento e aperfeiçoamento, a começar pela implementação dos itens já mencionados anteriormente. O sistema de atestado pode ser mudado para que o médico não precise imprimir o atestado, enviando-o diretamente para a empresa na qual o paciente trabalha.

Além disso, esse projeto é passível de análises que permitirão aperfeiçoamentos, como uma análise comparativa de desempenho de sistemas que foram desenvolvidos com essa arquitetura e sistemas que foram desenvolvidos com outra arquitetura. Também seria possível estender esse sistema para outras plataformas além do PC como aplicativos para Android, iOS e outras plataformas.

REFERÊNCIAS BIBLIOGRÁFICAS

De Souza, J. L. Uma Arquitetura de Software para Implementação de um EHR Utilizando SOA Considerando a Interoperabilidade entre Sistemas Legados. 2016.

Endrei, Mark; Ang, Jenny; Arsanjani, Ali; Chua, Sook; Comte, Philippe; Krogdahl, Pal; Luo, Min; Newling, Tony. Pattems: "Service-oriented architecture and Web services". Copyright International Business Machines Corporation, 2004.

Erl, Thomas. "Service-oriented architecture: A field guide to integrating XML and Web Services". Pearson Education Inc, 2004. 536p. ISBN 0131428985.

Furtado, C.; Pereira, V.; Azevedo, L.; Baião, F.; Santoro, F. Arquitetura Orientada a Serviço - Conceituação. Relatórios técnicos do Departamento de Informática Aplicada da UNIRIO nº0012. 2009. Disponível em: <http://www.seer.unirio.br/index.php/monografiasppgi/article/view/266> Acessado em 15 Set. 2018.

José Silva, A. Introdução ao WSDL. Fevereiro, 2018. Disponível em: <https://medium.com/@alexjosesilva/introdução-ao-wsdl-abece3a8bab5> Acessado em 15 Set. 2018.

Mabrouk, M. I. SOA fundamentals in a nutshell. Disponível em: https://www.ibm.com/developerworks/webservices/tutorials/ws-soa-ibmcertified/ws-soa-ibmcertified.html. 2008. Acessado em 26 Ago. 2018.

Perim, I. L. Prontuário Eletrônico: Uma Revisão Bibliográfica. Universidade Federal de Minas Gerais. Governador Valadares, 2011.

Pereira Siqueira, O. M.; Nogueira de Oliveira, R. A.; Aparecida de Oliveira, A. Integração de Sistemas de Informação em Saúde com a Utilização de Service Oriented Architecture (SOA). Revista

de Gestão da Tecnologia e Sistemas de Informação, Vol. 13, n. 2, Mai./Ago. 2016, pp. 255-274. ISSN 1807-1775.

Pires, J. O que é API? REST e RESTful? Conheça as definições e diferenças. 2017. Disponível em: <https://becode.com.br/o-que-e-api-rest-e-restful/> Acessado em 23 Set. 2018.

Plansky, R. Definição, restrições e benefícios do modelo de arquitetura REST. Outubro, 2014. Disponível em: <https://imasters.com.br/desenvolvimento/definicao-restricoes-e-beneficios-modelo-de-arquitetura-rest/?trace=1519021197&source=single> Acessado em 06 Out. 2018.

Rouse, M. Middleware. Dezembro 2017. Disponível em: <https://searchmicroservices.techtarget.com/definition/middleware> Acessado em 22 Set. 2018.

Siqueira Marques de Souza, V. A. Uma Arquitetura Orientada a Serviços para Desenvolvimento, Gerenciamento e Instalação de Serviços de Rede. 2006.

BIBLIOGRAFIA

Barra Cordeiro, E. SOA – Arquitetura Orientada a Serviços. Outubro, 2012. Disponível em: <http://blog.iprocess.com.br/2012/10/soa-arquitetura-orientada-a-servicos/> Acessado em 25 Ago. 2018.

Bruno. Web Services REST versus SOAP. 2015. Disponível em: < https://www.devmedia.com.br/web-services-rest-versus-soap/32451> Acessado em 13 Out 2018.

Calado, R. O que é REST?. Disponível em: <http://www.rodrigocalado.com.br/o-que-e-rest-um-resumo-do-assunto-caracteristicas-conceitos-vantagens-e-desvantagens-prefiro-dizer-que-e-uma-rapida-introducao-ao-assunto/> Acessado em 01 Set. 2018.

Documentação Oficial do Django. Disponível em: <https://docs.djangoproject.com/en/2.0/> Acessado em 06 Out. 2018.

Fusion Middleware Concepts Guide. Disponível em: <https://docs.oracle.com/cd/E17904_01/core.1111/e10103/intro.htm#ASCON109> Acessado em 29 Set. 2018.

Guia do Usuário de Astah. Disponível em: <http://astah.net/manual> Acessado em 22 Set. 2018.

Hübner Brondani, C. Guia Prático de utilização da ferramenta Astah Community 6.1. Disponível em: <https://www.scribd.com/doc/139768773/Astah-Comnunity> Acessado em 23 Set. 2018.

Maycon. WSDL: Simplifique a integração de dados via Web Service. 2014. Disponível em: <https://www.devmedia.com.br/wsdl-simplifique-a-integracao-de-dados-via-web-service/30066> Acessado em 08 Set. 2018.

O. Campos, K.; Donadel, A.; Todesco, J. L.; Varvákis, G.; de Souza Bermejo, P. H. Aplicação de tecnologias de Web Services para definição de um modelo de arquitetura orientada a serviço (SOA). 2006.

O. Campos, K.; Donadel, A.; Todesco, J. L.; Varvákis, G.; de Souza Bermejo, P. H. Um modelo de Arquitetura Orientada a Serviços (SOA). Congresso Sul Brasileiro de Computação. [S.I.]: 2006. ISSN 2359-2656.

Peracci, R. F.; Almeida Bessa, G. M. Utilizando Web Service para Integração de Sistemas um Estudo de Caso: Prefeitura de Juiz de Fora. Centro de Ensino Superior de Juiz de Fora. Juiz de Fora – MG, 2014.

Vincent S., W. Django for Beginners: Build websites with Python & Django. 2018. 314p. ISBN 9781983172663.

Vincent S., W. REST APIs with Django: Build websites with Python & Django. 2018. 188p. ISBN 9781983172663.

Web Services. Disponível em: <https://www.devmedia.com.br/web-services/2873> Acessado em 02 Set. 2018.