Healthgate – Plataforma de integração de microsserviços no domínio da saúde

Lucas Daniel Barboza
Programa de Pós-graduação em Tecnologias Sustentáveis, Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo - Vitória, ES, Brasil

Rodrigo Varejão Andreão
Programa de Pós-graduação em Tecnologias Sustentáveis, Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo - Vitória, ES, Brasil

Celso A.S. Santos
Departamento de Informática – Universidade Federal do Espírito Santo (UFES) – Vitória – ES – Brasil

Jordano Ribeiro Celestrini
Departamento de Informática – Universidade Federal do Espírito Santo (UFES) – Vitória – ES – Brasil

Rafael Pereira
Programa de Pós-graduação em Tecnologias Sustentáveis, Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo - Vitória, ES, Brasil

***Abstract*—Este trabalho apresenta o desenvolvimento de uma plataforma de integração de microsserviços para e-health, denominada HealthGate. A arquitetura proposta para a plataforma, baseada em microsserviços, visa auxiliar implementações contextualizadas em e-health, buscando atingir pontos críticos relacionados ao domínio, como a heterogeneidade de serviços e especificações. Propomos a utilização do padrão FHIR para integração de microsserviços na plataforma, contribuindo para a interoperabilidade entre sistemas. Para validar a arquitetura proposta e os requisitos definidos, foram realizados três estudos de caso, implantando um microsserviço que detecta episódios de fibrilação atrial em segmentos de ECG, um microsserviço que calcula o Escore de Framingham e um microsserviço que realiza a predição de diagnóstico de diabetes baseando-se em dados clínicos.**

Keywords — Microsserviços, Healthcare, FHIR.

I. INTRODUÇÃO

A execução dos serviços de e-health demanda uma grande quantidade de dados que devem estar constantemente atualizados e disponíveis, a fim de subsidiar a tomada de decisão médica e melhorar a qualidade do atendimento. Além disso, diferentes serviços de e-health geram e trocam dados entre si para que os pacientes possam receber tratamento adequado. Como em outros domínios, o aumento exponencial na quantidade de dados armazenados eletronicamente, a variedade de serviços e sistemas a serem disponibilizados, a necessidade de resposta rápida às regras de negócios e alguns requisitos específicos do domínio tornam a integração das TICs na área da saúde um tarefa complexa. [1] citam alguns desses requisitos particulares no domínio da saúde, a saber: o fornecimento de alta disponibilidade, para mitigar problemas relacionados à interrupção dos serviços de e-health; confiabilidade dos dados, com o objetivo de torná-los sempre disponíveis e não suscetíveis a falhas; e escalabilidade, para que os serviços de saúde em possam se adaptar a um número incerto de usuários.

Diante desse cenário, a gestão e manutenção da infraestrutura de tecnologia da informação, ao invés de implicar redução de custos, pode ter o efeito contrário, pois requer não apenas diversos insumos para a implantação dessa infraestrutura, mas também a disponibilidade de local adequado, instalação de componentes de hardware e equipamentos de rede, além da implementação de políticas de segurança, mas também a disponibilidade de uma equipe técnica treinada para gerenciá-la [4].

A computação em nuvem tem a capacidade de agilizar a manutenção, o suporte e a escalabilidade da infraestrutura de TIC, fornecendo novas maneiras de gerenciar e compartilhar dados médicos. O emprego de containers na computação em nuvem para desenvolver e implantar aplicações neste molde de disponibilidade favoreceu o uso crescente de um novo modelo arquitetônico de engenharia de software, conhecido como microsserviços.

Segundo [12], a integração de serviços heterogêneos no domínio da saúde tem a capacidade de apoiar aspectos relacionados às tarefas clínicas e administrativas, redução de erros médicos, redução de custos, segurança de dados, integração de processos de saúde e utilização de sistemas legados. No entanto, a integração dos serviços em si tem sido uma tarefa crítica, pois a redundância de dados e funcionalidades, a diversidade de tecnologias e a falta de padronização dos procedimentos de comunicação entre as aplicações são problemas comumente encontrados que afetam a qualidade do atendimento ao paciente. Para resolver essas deficiências, a arquitetura distribuída com foco em e-health pode ser capaz de lidar com esses fatores críticos. Esta arquitetura pode ser baseada em SOA (Service-Oriented Architecture) ou em microsserviços.

Os microsserviços consistem em um estilo arquitetural, no qual as aplicações de maior tamanho e complexidade são decompostas em um ou mais pequenos serviços. Esses serviços são desenvolvidos de forma independente e têm a responsabilidade de executar com eficiência uma única tarefa [4]. Este modelo surgiu como uma alternativa ao modelo de desenvolvimento de software monolítico, onde uma aplicação é construída na forma de um único grande bloco e que apresenta problemas como a dificuldade de desenvolver e entender um único grande bloco de código, a dificuldade de integração entre sistemas e a dificuldade de expandir ou alterar componentes que já foram implantados sem afetar a aplicação como um todo [6].

Nesse contexto, este artigo apresenta uma proposta de plataforma de integração de serviços de e-health baseada na arquitetura de microsserviços. Para possibilitar a interoperabilidade entre os microsserviços e a plataforma, propõe-se a utilização do padrão FHIR para especificação e descrição dos microsserviços. Assim, mesmo que sejam implementados em tecnologias diferentes, os serviços que serão implementados na plataforma devem seguir um único esquema de especificação. Essa abordagem facilita a integração de microsserviços na plataforma e permite que outras entidades que usam o padrão FHIR se comuniquem com a plataforma de forma padronizada.

Para demonstrar que a abordagem proposta é válida, três microsserviços foram implementados. Todo o processo de implementação do microsserviço na plataforma foi realizado em comunicação máquina-máquina, através da especificação FHIR, que serviu para descrever as características dos serviços, bem como as suas entidades de entradas e saídas.

II. ARQUITETURA DE MICROSSERVIÇOS

O modelo arquitetural baseado em microsserviços se tornou muito popular nos últimos anos, principalmente porque grandes empresas como Netflix e Amazon o utilizam no desenvolvimento de suas aplicações. Este modelo visa proporcionar maior facilidade de implementação, manutenção e escalabilidade para as aplicações. Como o serviço é implementado isoladamente, ele pode ser aprimorado e expandido com mais facilidade.

As vantagens de usar a arquitetura de microsserviços se devem à sua abordagem granular. Desta forma, novos componentes podem ser adicionados à aplicação sem afetar o funcionamento dos componentes em execução e os vários componentes podem ser escritos usando diferentes tecnologias, desde que ofereçam um mecanismo de troca de mensagens, geralmente uma interface de programação de aplicativo (API).

De acordo com [5] os microsserviços têm três características principais:

- Tamanho, sendo relativamente pequeno;
- Contexto limitado, pois as suas funcionalidades são implementadas como um único serviço;
- Independência, pois cada microsserviço é independente.

Ainda de acordo com [5], aplicações baseadas na arquitetura de microsserviços devem ser flexíveis, capazes de acompanhar mudanças de domínio e modificações necessárias, devem ser modulares, compostas por componentes isolados que contribuem para o funcionamento geral do sistema e devem ser evolutivas, capazes de constantemente evoluir com a adição de novos recursos.

A arquitetura de microsserviços surgiu na comunidade de desenvolvimento ágil, que enfrentou dificuldades no desenvolvimento de aplicações na arquitetura monolítica tradicional. Nessa arquitetura, como as aplicações monolíticas são construídos na forma de um único grande bloco, contendo todas as funções necessárias, torna-se mais difícil corrigir erros e adicionar novos recursos à medida que a aplicação cresce [2].

[5] enfatiza que as aplicações desenvolvidos em uma arquitetura monolítica são difíceis de manter e evoluir devido à sua complexidade. Qualquer alteração que precise ser feita resultará na indisponibilidade de toda a aplicação. Portanto, as aplicações monolíticas têm pouca escalabilidade e forçam os desenvolvedores a usar a mesma tecnologia em todo a aplicação.

A arquitetura de microsserviços surgiu como uma forma de resolver os problemas comumente encontrados na arquitetura monolítica. Assim, a independência dos microsserviços torna a computação em nuvem a plataforma natural para as aplicações desenvolvidas nesta arquitetura, pois permite a alocação de recursos de acordo com as necessidades de cada microsserviço [6].

Considerando a implementação de microsserviços baseados em computação em nuvem, os principais benefícios contextualizados em aplicações de saúde, segundo [7] são a dispensabilidade de aquisição de equipamentos, a dispensabilidade de instalação e manutenção de serviços, escalabilidade, permitindo que os serviços possam ser expandidos conforme a necessidade, e maior rapidez na implantação.

No entanto, o uso da arquitetura de microsserviços traz consigo uma série de desafios, causados principalmente por sua característica granular e desacoplada. Um desses desafios é a especificação de contratos de microsserviços, visto que podem usar diferentes tecnologias, a especificação de contratos pode usar diferentes abordagens, e algumas das tecnologias mais atuais não possuem uma linguagem para especificação de microsserviços [5]. Outro desafio envolve a confiabilidade e segurança dos microsserviços, uma vez que o uso de APIs para fornecer serviços os torna mais expostos no nível da rede, além disso, conforme a quantidade de microsserviços aumenta, também aumenta a dificuldade de depuração e monitoramento desses microsserviços [5]

O isolamento e a independência dos microsserviços também exigem mecanismos de descoberta, controle e acesso, garantindo que cada microsserviço desempenhe sua função, possa ser descoberto e consumido de forma dinâmica, segura e controlada [10].

Apesar das vantagens da arquitetura de microsserviços em relação à arquitetura tradicional, de acordo com [10], existem alguns desafios associados ao uso de microsserviços, dentre os quais se destacam:

- A descoberta de novos serviços pelas aplicações deve ser feita de forma dinâmica , uma vez que os microsserviços são isolados e independentes;
- O roteamento da solicitação deve ser feito de forma controlada e precisa, garantindo que cada microsserviço desempenhe sua função;
- O acesso aos microsserviços deve ser realizado de forma segura e controlada.

Proxies, balanceadores de carga e mecanismos de descoberta de serviço são soluções bem conhecidas para lidar com esses três desafios. Para orquestrar a comunicação entre esses componentes, pode-se utilizar um *gateway*, responsável por controlar as APIs fornecidas pelos microsserviços. Considerando que um aplicativo pode fornecer funcionalidade a clientes de diferentes naturezas, como desktops, smartphones ou tablets, um *gateway* pode armazenar diferentes APIs implementadas para diferentes conjuntos de clientes. Como resultado, ao oferecer um único

ponto de entrada, o *gateway* também pode fornecer ferramentas para descoberta de serviço, balanceamento de carga, monitoramento e segurança.

Considerando o papel do *gateway*, o mecanismo de descoberta do serviço pode ser implementado de forma acoplada ou não. A descoberta de serviço é usada por entidades externas que desejam consumir os serviços disponíveis. Assim, sempre que um novo serviço for disponibilizado, ele deverá fornecer uma descrição que possibilite o seu registro. Da mesma forma, quando um cliente deseja consumir um serviço, ele deve consultar o registro do serviço para descobrir seus pontos de entrada, modelos de dados e outras informações necessárias. Sempre que o *gateway* receber um pedido dirigido a um determinado serviço, deverá consultar o registo de serviços para encaminhar o pedido ao serviço responsável.

III. PADRÃO HL7 FHIR

O HL7 FHIR é um padrão que define um conjunto de recursos que são utilizados em processos de saúde em diferentes contextos [14]. Ele foi projetado principalmente para fornecer interoperabilidade de sistemas de dados de registros médicos. O padrão FHIR representa dados clínicos como recursos, onde cada recurso consiste em campos e tipos de dados. Os recursos FHIR são intuitivos, por exemplo, o recurso *MedicationPrescription* faz referência ao seu prescritor (um médico FHIR), seu paciente (um paciente FHIR) e o medicamento prescrito (um medicamento FHIR).

O FHIR fornece uma interface HTTP contemporânea e orientada a recursos para pesquisar, criar, ler, atualizar e excluir recursos FHIR que representam várias estruturas relacionadas ao domínio da saúde.

Conforme mencionado por [14], a utilização do padrão HL7 FHIR se justifica por ser um padrão baseado em tecnologias web, permitindo rápida implementação, além de oferecer bibliotecas open source. Essas características tornam o desenvolvimento de aplicações de telemedicina menos dispendiosas em comparação com outros modelos mais abrangentes, como o openEHR [8].

De acordo com [14], o FHIR pode ser utilizado para descrever os recursos de um microsserviço, fornecendo, por exemplo, a descrição das entradas e saídas. Neste trabalho, o FHIR foi utilizado para prover interoperabilidade entre os microsserviços e a plataforma proposta. O recurso *HealthcareService* foi utilizado, por sua vez, para descrever as características dos microsserviços.

O recurso *HealthcareService* é usado para descrever um serviço de saúde fornecido por uma organização. Embora, de acordo com a especificação FHIR, eles não representem serviços relacionados a computadores (como SOA), o esquema FHIR pode ser usado para descoberta de serviços em uma arquitetura de microsserviços [14].

O recurso *Endpoint*, que é incorporado ao recurso *HealthcareService*, foi usado para descrever os tipos de entradas utilizados pelos microsserviços, bem como descrever os tipos de saídas geradas. De acordo com a especificação FHIR, um recurso *Endpoint* FHIR descreve os detalhes técnicos de um local que pode ser conectado para a entrega e recuperação de informações [15].

O atributo *payloadType* foi utilizado em nossa abordagem para especificar dados estruturados, permitindo o uso de entradas com base no esquema json, e o atributo *payloadMimeType* foi usado para especificar o tipo de arquivo que será enviado na solicitação, como application/ fhir + xml ou application / fhir + json.

IV. MÉTODOS

O desenvolvimento da plataforma de integração de microsserviços seguiu uma abordagem tradicional de engenharia de software, com etapas de especificação, projeto e implementação e validação.

Na etapa de especificação, foram definidos os requisitos funcionais e não funcionais da plataforma proposta, conforme ilustrado na tabela 1.

TABLE I. REQUISITOS FUNCIONAIS E NÃO-FUNCIONAIS

RF 01. Registrar microsserviço Informações: O sistema deve possibilitar o registro de microsserviços por meio de uma descrição utilizando o esquema FHIR.	RNF 01. Aplicação Informações: A plataforma será implementada em arquitetura web
RF 02. Listar microsserviços Informações: O sistema deve permitir que o usuário consulte a lista de microsserviços registrados e publicados.	RNF 02. Acesso Informações: A base de dados deverá ser protegida, contendo o acesso apenas de usuários devidamente autorizados.
RF 03. Pesquisar microsserviço Informações: O sistema deve fornecer uma ferramenta de pesquisa de microsserviço para o usuário	RNF 03. Implementação Informações: O sistema será implementado em linguagem Javascript com banco de dados não relacional com gerenciador MongoDB, utilizando frameworks NodeJS, ExpressJS, Angular.
RF 04. Consumir microsserviço Informações: O sistema deve ser capaz de implementar automaticamente a interface do usuário permitindo o uso do microsserviço em questão.	RNF 04. Responsividade Informações: O sistema deve ser acessível a partir de qualquer dispositivo móvel, sem afetar seu funcionamento.

Na fase de projeto e implementação, foi definida uma arquitetura para a plataforma baseada em quatro camadas principais: aplicações cliente, *gateway*, servidor FHIR e microsserviços. Essa arquitetura aproveita a característica autônoma dos microsserviços, permitindo que sejam implementados e implantados independentemente da plataforma ou linguagem de programação, bastando ter uma camada de acesso implementada na forma de uma API REST. Da mesma forma, os serviços podem ser disponibilizados para uma variedade de clientes diferentes sem que eles tenham conhecimento dos detalhes de implementação. A Figura 1 mostra uma visão geral da arquitetura proposta chamada HealthGate.

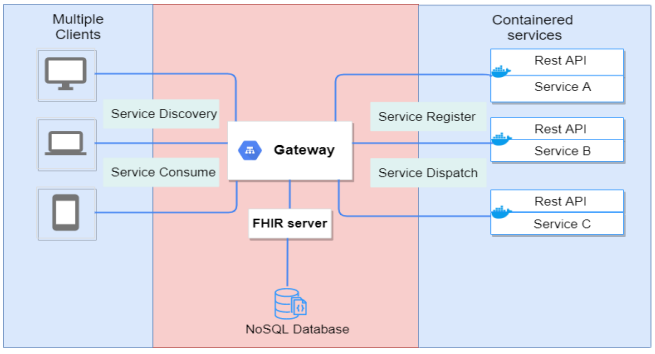

Figura 1. Proposta de arquitetura

A Figura 2 apresenta o diagrama de sequência ilustrando a troca de mensagens entre a interface de usuário do HealthGate, o *gateway*, o servidor FHIR e as entidades de microsserviços. A primeira operação a ser realizada é a publicação do serviço. Após esta etapa, o cliente passa a conhecê-lo e consumi-lo conforme sua descrição.

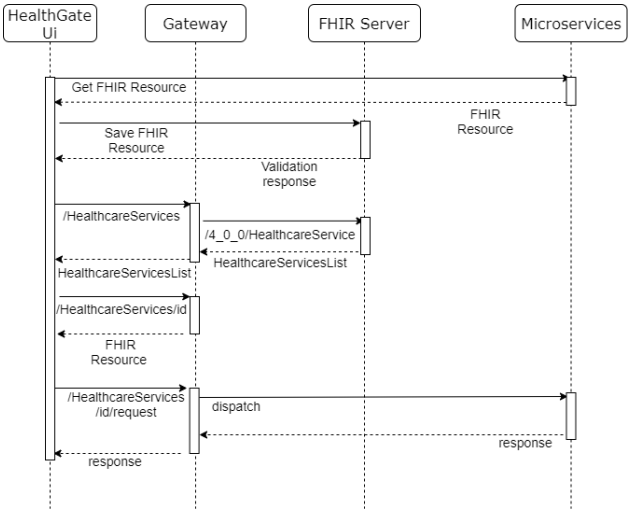

Figure 2. Diagrama de sequência

A plataforma foi implementada usando a pilha de ferramentas MEAN, composta pelo banco de dados não relacional MongoDB, o framework de back-end Express.js, o framework de front-end Angular e o interpretador Javascript Node.js. Essa pilha de ferramentas usa uma única linguagem de programação para desenvolvimento de *front-end* e *back-end*, proporcionando maior produtividade no desenvolvimento de aplicações web. A camada lógica da aplicação, denominada *gateway*, foi desenvolvida com o auxílio do framework Express. Para permitir aos clientes acessar a lista de serviços e consumi-los, foi implementada uma API REST, que tem seus endpoints mostrados na tabela 2.

TABLE II. DESCRIÇÃO DOS ENDPOINTS

Endpoint	**Ator**	**Método**	**Descrição**
/publish	Microservice	Post	Publica o serviço após receber sua descrição.
/services	Cliente	Get	Retorna a lista de serviços disponíveis.
/services/:serviceId	Cliente	Get	Retorna a descrição detalhada do serviço
/services/:serviceId	Cliente	Post	Encaminha requisição de serviço específico.

A interface da aplicação, implementada com o auxílio do framework Angular, foi desenhada para que o cliente possa acessar a lista de serviços disponíveis na plataforma, bem como também é responsável por construir todos os campos utilizados como entrada para o microsserviço, com base na descrição das entradas fornecidas pelo provedor de microsserviços, baseada no padrão FHIR.

V. ESTUDOS DE CASO

Para validar os requisitos definidos na tabela 1 e a arquitetura proposta, além de levantar novos requisitos específicos que possam surgir em um ambiente de e-health, foram avaliados três cenários com a implantação de três microsserviços. O primeiro microsserviço é baseado no trabalho de [11]. Os autores usaram uma rede neural profunda baseada na topologia *long-short term memory* (LSTM) para identificar episódios de fibrilação atrial em registros de 10 segundos de eletrocardiogramas (ECGs). A figura 3 exibe a rede LSTM com 6 camadas utilizada pelos autores. Esse serviço tem potencial para ser utilizado em ambientes de telecardiologia, contribuindo para o diagnóstico mais rápido da fibrilação atrial no processo de triagem de pacientes.

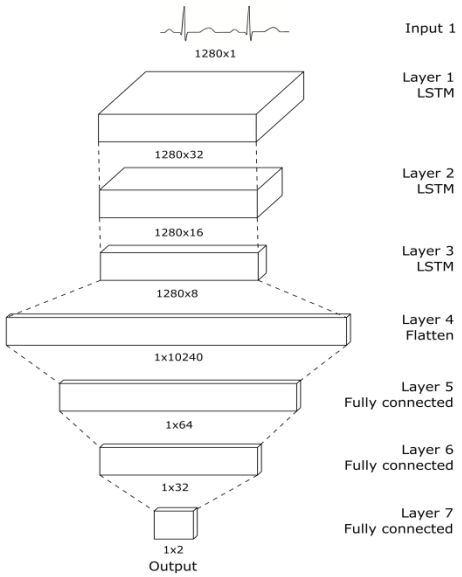

Figure 3. Rede LSTM com múltiplas entradas

O segundo microsserviço realiza o cálculo do escore de Framingham. Consiste em uma métrica para estimar o risco de um evento cardiovascular em um determinado indivíduo a partir de algumas variáveis como idade, nível de colesterol e pressão arterial [9]. A utilização de um serviço que faça esse cálculo contribui para que ações preventivas sejam tomadas em indivíduos com potencial para desenvolver doenças cardiovasculares.

Já o terceiro microsserviço consiste em um modelo *Support Vector Machine* (SVM), treinado para classificar dados de pacientes, realizando a predição se os pacientes possuem ou não possuem diabetes. O modelo foi treinado utilizando o dataset disponibilizado em [16], o qual é formado por dados apenas de pacientes mulheres, possuindo como atributos o número de gestações da paciente, seu IMC, nível de insulina, idade e etc. Para este trabalho, a maior acurácia alcançada foi de 75%, utilizando uma configuração de 50% de dados para treino e 50% para teste. Apesar de ser um modelo básico, estudos mais avançados podem ser úteis em sistemas de monitoramento remoto de pacientes.

Para abstrair as preocupações sobre as dependências de hardware e software, o primeiro serviço de estudo, que utiliza aprendizado profundo, foi implementado com o auxílio do ambiente de execução Google Colab [3], que possui várias ferramentas necessárias para executar modelos de aprendizado de máquina, incluindo configurações de GPU. Ambos os serviços foram expostos na forma de API REST, com o auxílio da linguagem de programação Python e do framework web Flask.

No primeiro microsserviço, foram implementados dois endpoints: "/genfa", que recebe pelo método POST o arquivo contendo os segmentos de ECG, e endpoint "/spec", que é responsável por retornar a especificação FHIR do microsserviço, e é acessado por meio de uma solicitação do tipo GET. Portanto, para implantar o microsserviço, uma solicitação GET deve ser feita no endpoint "/spec". No processo de pré-validação, a descrição FHIR retornada pelo microsserviço é decomposta na forma dos dois recursos FHIR usados neste trabalho, *HealthcareService* e *Endpoint*. O servidor FHIR valida a especificação dos recursos, persistindo os recursos utilizados nos diferentes registros. Como é utilizado um banco de dados não relacional, existe um documento para registro do recurso *Endpoint* e outro documento para registro do recurso *HealthcareService*. Todo esse processo pode ser realizado em comunicação máquina a máquina, desde que o usuário forneça a URL do microsserviço contendo a especificação FHIR.

Com a persistência da especificação dos recursos, o usuário pode visualizar o microsserviço na lista de microsserviços cadastrados na plataforma, e pode consumi-lo. A interface para usar o microsserviço foi construída com base na especificação do recurso *Endpoint*. No caso do microsserviço em estudo, foi especificado que ele recebe um arquivo do tipo multipart/form-data. Assim, a interface foi construída para permitir o upload do arquivo contendo os trechos do ECG.

O processamento de solicitações entre o *gateway* e a interface do usuário é realizado por meio do ID do microsserviço persistido no banco de dados. Ao receber a solicitação POST contendo o arquivo a ser analisado, o *gateway* verifica qual microsserviço irá processar a solicitação. Essa abordagem com um *gateway* permite o controle central das solicitações, proporcionando maior segurança aos microsserviços, além de fornecer um ponto central para chamadas de funções.

Após encaminhar a solicitação para o microsserviço, a resposta é encaminhada pelo *gateway* para a interface do usuário. O processo de utilização do serviço é ilustrado na Figura 4, por meio de um diagrama de atividades, demonstrando a troca de informações entre as entidades e os processos que são realizados.

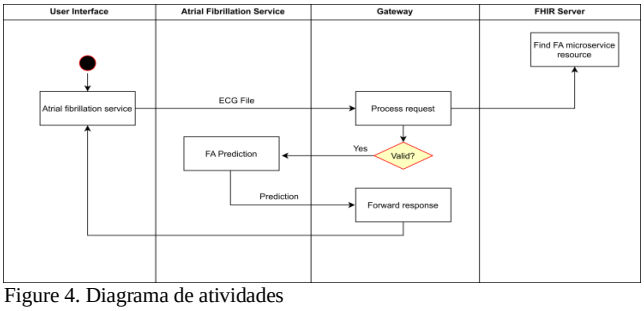

Figure 4. Diagrama de atividades

No segundo microsserviço, foram implementados dois endpoints, "/genrisk", que recebe via solicitação POST um arquivo json contendo as entradas necessárias para o cálculo da pontuação, e "/spec", responsável por retornar a especificação FHIR por meio de uma solicitação GET. O processo de implementação do microsserviço na plataforma segue o mesmo padrão adotado no microsserviço de detecção de fibrilação atrial. No entanto, a especificação do recurso *Endpoint* definido para este microsserviço estabelece que o endpoint "/genrisk" recebe como entrada dados do tipo application/json. A partir dessa definição, a plataforma cria campos customizados que devem ser preenchidos pelo usuário para consumir o serviço. Esses campos são exibidos em formato de formulário e, após o preenchimento, a plataforma é responsável por construir o arquivo json com os dados inseridos no formulário e encaminhar a solicitação a ser calculada pelo microsserviço.

Já para o terceiro microsserviço, foram implementados os endpoints "/diabetes", que recebe via solicitação POST oito atributos necessários, e o endpoint "/spec", responsável por retornar via solicitação GET a descrição FHIR do serviço. Assim como realizado no segundo serviço de estudo, a plataforma cria campos customizados que devem ser preenchidos pelo usuário para consumir o serviço e monta a requisição que será encaminhada ao *gateway* e posteriormente encaminhada ao microsserviço.

Um vídeo disponibilizado no link https://www.youtube.com/watch?v=mAQNpUQ1iZc, demonstra o processo de utilização da plataforma proposta, onde são exibidas as implantações dos dois últimos microsserviços utilizados para avaliação da plataforma, exibindo as etapas para cadastro e consumo do serviço, bem como as opções de configuração.

VI. Discussão

Neste trabalho, utilizamos uma arquitetura baseada em microsserviços para desenvolver um protótipo de plataforma voltada para a integração de serviços de e-health. Esta arquitetura flexível proporciona aos serviços de saúde maior independência, uma vez que cada serviço é implementado de forma isolada, utilizando tecnologias diferentes. Essa abordagem pode levar a problemas de interoperabilidade e implantação de serviços. Para mitigar essa possibilidade, utilizamos o padrão FHIR como forma de especificar os serviços cadastrados na plataforma. Desta forma, todos os serviços cadastrados seguem a mesma especificação.

A utilização de um servidor FHIR para controlar a troca de dados padronizada por este formato, permite que outros recursos sejam associados pela plataforma. Um desses recursos pode ser o recurso *Patient*, que engloba dados relacionados à saúde em uma gama de atividades [8]. A combinação desses recursos em uma arquitetura baseada em microsserviços permite a composição de *workflows* customizados de acordo com as necessidades dos pacientes, organizações ou outros atores.

Para validar a plataforma, utilizamos três cenários onde implementamos microsserviços que realizam tarefas relacionadas ao diagnóstico de doenças cardíacas e diabetes. Para ambos os serviços, o recurso FHIR *HeathcareService* especifica características gerais, enquanto o recurso *Endpoint* especifica características técnicas que são usadas pela plataforma para controlar solicitações entre clientes e serviços.

No primeiro cenário, percebemos que o microsserviço implementado demanda alto poder computacional, pois utiliza aprendizado profundo para prever episódios de fibrilação atrial. Essa alta demanda computacional pode se tornar um problema diante de um grande número de solicitações simultâneas que podem ocorrer, ocasionando principalmente problemas de latência, porém, o uso da arquitetura baseada em microsserviços permite que o microsserviço seja implantado em um ambiente de nuvem, e seu poder computacional pode ser dimensionado de acordo com crescimento da demanda, em um processo conhecido como *serverless*.

No segundo cenário e no terceiro cenário, so microsserviços de cálculo do escore de Framingham e de predição de diabetes não necessitam de alto poder computacional, entretanto, os dados de entrada são estruturados na forma de um arquivo json que contém diversos atributos. Com a utilização do padrão FHIR, esses atributos são facilmente especificados no recurso *Endpoint* e também permite, como já mencionado, a integração da plataforma com outras soluções que utilizam este padrão.

No contexto da programação, as tecnologias utilizadas para implementação baseadas apenas na linguagem de programação Javascript permitiram maior facilidade na criação e correção de funções, além de reduzir a ocorrência de erros, como erros de sintaxe que podem ocorrer ao se utilizar diferentes linguagens.

A utilização de uma base de dados não relacional possibilitou a utilização do padrão FHIR de forma mais amigável, uma vez que os recursos são estruturados na forma de documentos, o que facilita todas as operações de inserção, atualização e remoção de registros.

A interface de usuário construída com o auxílio do framework Angular e da biblioteca Angular Material permitiu a criação de uma interface amigável que se adapta facilmente a ambientes desktop e mobile, permitindo o uso da plataforma em diferentes configurações de dispositivos e tamanhos de tela.

Portanto, concluímos que todos os requisitos do processo de modelagem foram corretamente implementados, e os cenários estudados permitem afirmar que a plataforma proposta permite a implementação de serviços voltados para e-health em diferentes cenários.

VII. TRABALHOS FUTUROS

Este trabalho demonstrou as etapas de especificação, projeto, implementação e validação de uma plataforma de integração de microsserviços para e-health denominada HeathGate. A etapa de validação da plataforma buscou explorar aspectos relacionados à interoperabilidade de aplicações na área de saúde.

Durante o processo de validação, observou-se que todos os requisitos propostos para a plataforma foram implementados e validados por meio de três estudos de caso. Esses microsserviços foram implementados na plataforma utilizando uma linguagem padronizada baseada em recursos FHIR, em um processo que pode ser realizado em comunicação máquina-máquina, contribuindo para a homogeneidade das especificações do serviço e, consequentemente, uma facilidade de integração.

Extensões futuras deste trabalho incluem a avaliação dos requisitos de segurança relacionados ao domínio e-health, incluindo a análise de problemas relacionados à disponibilidade da aplicação. Pretende-se também realizar novos estudos de caso, implementando serviços mais heterogêneos na plataforma.

AGRADECIMENTOS

Os autores agradecem o apoio financeiro da FAPES TO093/2017 e TO 598/2018 e o apoio financeiro da CAPES.

REFERÊNCIAS

[1] E. Abukhousa, N. Mohamed e J. Al-Jaroodi. (2012). e-Health Cloud: Opportunities and Challenges. Future Internet. 4. 621-645. 10.3390/fi4030621.

[2] R. Chen, S. Li e Z. Li. (2018). From Monolith to Microservices: A Dataflow-Driven Approach. In Proceedings - Asia-Pacific Software Engineering Conference, APSEC (Vol. 2017-December, pp. 466–475). IEEE Computer Society. https://doi.org/10.1109/APSEC.2017.

[3] Google Colab. Disponível em: https://colab.research.google.com/

[4] C. Esposito, A. Castiglione, C. Tudorica e F. Pop. (2017). Security and privacy for cloud-based data management in the health network service chain: a microservice approach. IEEE Commun. Mag., 55, 102-108.

[5] N. Dragoni et al. (2017). Microservices: Yesterday, today, and tomorrow. In Present and Ulterior Software Engineering (pp. 195–216). Springer International Publishing. https://doi.org/10.1007/978-3-319-67425-4_12.

[6] V. Pachghare. (2016). Microservices Architecture for Cloud Computing. Journal of Information Technology and Sciences. p. 1–13.

[7] T. Paiti. What are the opportunities and challenges of cloud computing technology in the healthcare information systems. 2013. 60 f. Master of Science Thesis – Degree Program in Information Technology, Faculty of Faculty of Industrial Engineering and Management, Lappeenranta University of Technology.

[8] FHIR Patient Resource. Disponível em: https://www.hl7.org/fhir/patient.html

[9] H. Pimenta e A. P. Caldeira. (2014). Fatores de risco cardiovascular do Escore de Framingham entre hipertensos assistidos por equipes de Saúde da Família. Ciência & Saúde Coletiva, 19(6), 1731-1739.

[10] N. Singhal, U. Sakthivel e P. Raj. (2019). Selection Mechanism of Micro-Services Orchestration Vs. Choreography. International Journal of Web & Semantic Technology, 10(1), 01–13. https://doi.org/10.5121/ijwest.2019.10101

[11] R. Pereira, R. V. Andreão e G. T. Zago. (2019). Identificação de fibrilação atrial em registros eletrocardiograma utilizando Redes Neurais Recorrentes do tipo Long Short-Term Memory. 10.17648/sbai-2019-111461.

[12] K. Khoumbati, M. Themistocleous e Z. Irani. (2005). Integration Technology Adoption in Healthcare Organisations: A Case for Enterprise Application Integration. Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 149a-149a.

[13] FHIR HealthcareService Resource. Disponível em: http://hl7.org/fhir/healthcareservice.html

[14] B. Eapen, K. Sartipi e N. Archer. (2020). Serverless on FHIR: Deploying machine learning models for healthcare on the cloud

[15] FHIR Endpoint Resource. Disponível em: http://hl7.org/fhir/endpoint.html#Endpoint

[16] Pima Indians Database. Disponível em: https://www.kaggle.com/uciml/pima-indians-diabetes-database