CBIS'24 XX Congresso Brasileiro de Informática em Saúde
08/10 a 11/10 de 2024 - Belo Horizonte/MG - Brasil

IF-Cloud: API FHIR para integração de projetos de saúde digital

IF-Cloud: FHIR API for integrating digital healthcare projects

IF-Cloud: API FHIR para integración de proyectos de salud digital

Juliano Machado Vieira[1], Jeremias Piontkoski de Abreu[1], Juliano Costa Machado[1], Fábio Pires Itturriet[2], André Luís Del Mestre Martins[1]

1 IFSul campus Charqueadas (RS), Brasil.
2 Departamento Acadêmico de Eletrotécnica, UTFPR, Curitiba (PR), Brasil

Autor correspondente: André Luís del Mestre Martins
E-mail: andremartins@ifsul.edu.br

Resumo

Objetivo: Desenvolver IF-Cloud, uma API para prototipagem e integração de dispositivos IoT e aplicações web de saúde interoperáveis. **Método:** O estudo de caso é um ecossistema de saúde digital para monitoramento de biossinais que realiza apenas operações Criar/Ler/Atualizar/Excluir (CRUD). IF-Cloud recebe novas operações através do *upload* de *scripts python* em uma interface gráfica. IF-Cloud utiliza os dados dos recursos FHIR provenientes de alguma API de CRUD e retorna um outro recurso FHIR com os dados processados pelos *scripts*. **Resultados:** um biossinal foi cadastrado na API de CRUD para os experimentos. A compressão de dados e o cálculo de frequência cardíaca são operações incluídas no ecossistema utilizando IF-Cloud. Uma aplicação para visualização de biossinais se beneficia da adição ao exibir a frequência cardíaca e o biossinal simultaneamente. **Conclusão:** IF-Cloud permite a inclusão de novas funcionalidades em ecossistemas de saúde digital mediante um *upload* de arquivos de *script*.

Descritores: Saúde Digital; Interoperabilidade da Informação em Saúde; Computação em Nuvem

Abstract

Objective: Develop IF-Cloud, an API for prototyping and integrating IoT devices and web applications for healthcare. **Method:** The case study is a healthcare ecosystem for biosignal monitoring that only performs Create/Read/Update/Delete (CRUD)

operations. IF-Cloud receives new operations by uploading python scripts into a graphical user interface. IF-Cloud uses data of FHIR resources coming from some CRUD API and returns another FHIR resource with the data processed by the scripts. **Results:** a biosignal was registered in the CRUD API for the experiments. Data compression and heart rate calculation are operations included in the ecosystem using IF-Cloud. An application for visualizing biosignals benefits from the addition by displaying heart rate and biosignal simultaneously. **Conclusion:** IF-Cloud allows the inclusion of new functionalities in healthcare ecosystems by uploading script files.
Keywords: Digital Health; Health Information Interoperability; Cloud Computing

Resumen

Objetivo: desarrollar IF-Cloud, una API para la creación de prototipos e integración de dispositivos IoT y aplicaciones web sanitarias interoperables. **Método:** El caso de estudio es un ecosistema de salud digital para el monitoreo de bioseñales que solo realiza operaciones de Crear/Leer/Actualizar/Eliminar (CRUD). IF-Cloud recibe nuevas operaciones cargando scripts de Python en una interfaz gráfica. IF-Cloud utiliza datos de recursos FHIR provenientes de alguna API CRUD y devuelve otro recurso FHIR con los datos procesados por los scripts. **Resultados:** se registró una bioseñal en la API CRUD para los experimentos. La compresión de datos y el cálculo de la frecuencia cardíaca son operaciones incluidas en el ecosistema mediante IF-Cloud. Una aplicación para visualizar bioseñales se beneficia de la adición al mostrar la frecuencia cardíaca y la bioseñal simultáneamente. **Conclusión:** IF-Cloud permite la inclusión de nuevas funcionalidades en los ecosistemas digitales de salud mediante la carga de archivos script.
Descriptores: Salud Digital; Interoperabilidad de la Información en Salud; Nube Computacional

Introdução

O ecossistema de sistemas de informação em saúde está cada vez mais heterogêneo, integrando dispositivos *Internet of Things* (IoT – Internet das Coisas), computação de borda, computação em nuvem e aplicativos.[1] Algumas das pesquisas envolvem aplicações e dispositivos de hardware IoT e necessitam de alguma infraestrutura de nuvem para fins de prototipação, experimentação e validação,[2] pois

é a camada de computação em nuvem que realiza a interface entre dispositivos IoT e aplicações de usuário.

Na camada de computação em nuvem dos projetos IoT em saúde digital, está assumido que existem algumas funcionalidades: processamento e análise de dados produzidos por múltiplos tipos de dispositivos; monitoramento contínuo de dados de pacientes; disponibilidade e acessibilidade de dados em escala, entre outros.[3] Em todos os casos citados, os trabalhos focados em dispositivos IoT assumem que a nuvem composta por um ou mais serviços e APIs (*Application Programming Interface* - Interface de Programação de Aplicação) executa algoritmos e softwares para suportar as funcionalidades desejadas. Entretanto, a prototipagem de uma infraestrutura de nuvem para testar dispositivos IoT geralmente não é realizada porque trata-se de um desenvolvimento oneroso e não é o foco desses trabalhos.

Além disso, um requisito que qualquer aplicação de saúde precisa respeitar é a interoperabilidade entre os diferentes sistemas.[4] Dentre os diversos padrões para troca de dados já propostos visando a questão da interoperabilidade, o FHIR - Fast Health Interoperability Standards[5] - se destaca pela crescente adoção mundial devido a sua característica de focar na integração por serviços web e na compatibilidade com aplicações RESTful. Desenvolver projetos IoT e aplicações web que se comunicam utilizando o padrão FHIR facilita a integração por qualquer sistema proveniente de qualquer desenvolvedor que também use FHIR.[6]

Por causa da abrangência do ecossistema de aplicações de saúde digital, esta proposta foca em um ecossistema especializado no monitoramento contínuo de biossinais, onde os dispositivos IoT são chave na aquisição, transmissão, armazenamento, processamento e visualização de dados de pacientes.[2] Neste trabalho, biossinal refere-se a sinais bioelétricos variantes no tempo como eletrocardiograma (ECG), fotopletismografia (PPG), entre outros.

O problema de pesquisa deste trabalho é viabilizar o desenvolvimento e a integração dos projetos IoT e aplicações web de saúde, onde novas soluções computacionais possam ser testadas e prototipadas em nuvem com curva de aprendizagem mínima, evitando ou minimizando que o tempo de desenvolvimento e testes seja alocado com implantação de nuvem para as soluções. Por exemplo, o processo de compressão de dados é bem vindo no processamento de qualquer biossinal porque reduz a necessidade de armazenamento e acelera a transmissão dos

dados, o que eventualmente contribui para a utilização eficiente da largura de banda e redução de custos.[7] Outro exemplo, a aquisição de biossinais por dispositivos IoT pode exigir processamento de sinais em software como filtragem em software de sinais com ruído, reconhecimento de padrões (exemplo: identificar picos R de uma onda ECG), obtenção de dados a partir do sinal adquirido (exemplo: cálculo de saturação de oxigênio a partir de um PPG).[8] Os projetos IoT usualmente não se preocupam em demonstrar as operações de processamento na nuvem, pois fazem de forma local ou integrada ao dispositivo.[9] Assim, a hipótese deste trabalho é que um servidor web na nuvem permita o *upload* de *scripts* para prototipar projetos IoT e aplicações de saúde com troca de mensagens padronizada e sem a necessidade de desenvolver infraestrutura de nuvem para integração do ecossistema de aplicações. A justificativa desta pesquisa é que a inexistência de uma solução em nuvem abrangente o suficiente para projetos IoT e aplicações web é um gargalo muito relevante para o avanço das pesquisas em saúde digital.

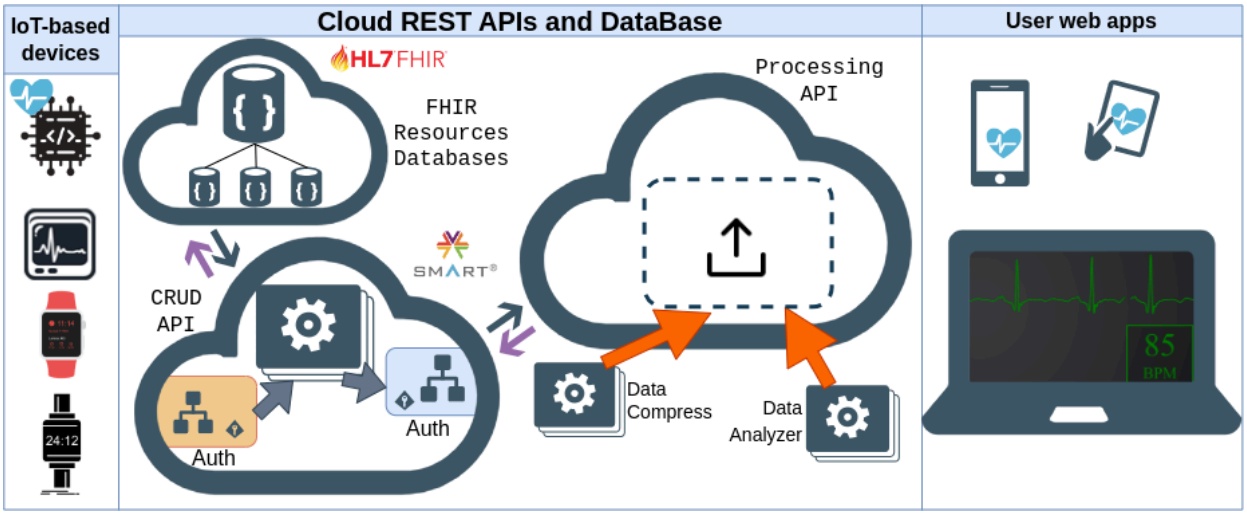
O objetivo deste trabalho é desenvolver, testar e disponibilizar **IF-Cloud**, uma API em **nuvem** para a prototipagem e **I**ntegração de projetos IoT e aplicações web de saúde compatível com o padrão de interoperabilidade **F**HIR.

O principal diferencial deste trabalho é que IF-Cloud habilita a integração e prototipação de diferentes funcionalidades dentro de um mesmo ecossistema de aplicações de saúde. Outras contribuições são: a disponibilização aberta e gratuita do código-fonte de IF-Cloud e a integração com aplicações FHIR de terceiros.

Métodos

Servidores FHIR definem um conjunto de interações comuns realizadas em um repositório de recursos digitais. Essas interações seguem o paradigma RESTful de gerenciamento de estado por ações Criar/Ler/Atualizar/Excluir (CRUD - Create/Read/Update/Delete) no conjunto de recursos. Embora os CRUDs resolvam muitos casos de uso, há algumas funcionalidades que exigem a execução de algum tipo de computação ou algoritmo em cima dos recursos FHIR.

Figura 1 - Visão geral do ecossistema de saúde digital para monitoramento contínuo de biossinais com troca de recursos no padrão FHIR. As funcionalidades das APIs em nuvem para suportar projetos IoT e aplicações web de biossinais podem ser resumidas em CRUD e processamento. IF-Cloud compõe o ecossistema como API de processamento.



A Figura 1 ilustra duas APIs FHIR em nuvem que suportam o ecossistema de monitoramento contínuo assumido neste trabalho. Os dispositivos IoT realizam a aquisição e o envio dos biossinais para a nuvem. As aplicações web recebem os dados da nuvem e os dispobilizam para visualização. Diversas implementações de APIs FHIR em nuvem[10] estão disponíveis para realizar as operações CRUD. IF-Cloud é a API para processamento de dados e que precisa, necessariamente, realizar operações na nuvem de CRUD para consultar os dados a serem processados. Ou seja, IF-Cloud não se conecta nem gerencia banco de dados diretamente, pois IF-Cloud realiza requisições em outras APIs para acessar os recursos a serem operados. Está assumido que todos os softwares e dispositivos a do ecossistema recebem e enviam dados eletrônicos de saúde seguindo a norma FHIR, obrigatoriamente.

Registro de Biossinais em FHIR

Dentre os mais de 150 recursos FHIR para descrever os diversos tipos de registro eletrônicos de saúde, o tipo *Observation* é utilizado para representar diversos biossinais dentro do padrão FHIR.[5] Quando o recurso FHIR é representado como um JSON, diversas chaves descrevem um *Observation*. Por exemplo, utilizando o biossinal do eletrocardiograma (ECG), as chaves mais relevantes representá-lo (Quadro 1) são:

- `resourceType` - campo obrigatório (*Observation*);
- `component` - array de objetos JSON, onde cada JSON descreve uma derivação do ECG.
  - `code` - código da derivação do ECG;

- `period` - taxa de amostragem, em milissegundos, entre os pontos que compõem a onda do ECG;
- `data` - string que contém cada amostra que descreve a onda do ECG para determinada derivação.
- `lower/upperLimit` - valores amostrais máximos e mínimos encontrados na chave de `"data"` (essenciais para fins de exibição).

Quadro 1 - Exemplo de um recurso FHIR Observation no formato JSON descrevendo um ECG de 2 derivações.


```
{
  "resourceType": "Observation",
  "id": "63f7f7a39c173811e7128d4c",
  "component": [
    {
      "code": {
        "coding": [{"display": "MDC_ECG_LEAD_AVR"}]
      },
      "valueSampledData": {
        "period": 10,
        "lowerLimit": -3300,
        "upperLimit": 3300,
        "data": "2041 2043 2037 ... 2027 2034 2033"
      }
    },
    {
      "code": {
        "coding": [{"display": "MDC_ECG_LEAD_AVL"}]
      },
      "valueSampledData": {
        "period": 10,
        "lowerLimit": -2000,
        "upperLimit": 2000,
        "data": "934 932 960 ... 920 950 960"
      }
    }
  ]
}
```


Desenvolvimento de IF-Cloud

IF-Cloud foi desenvolvido em linguagem Node.js para automatizar scripts escritos em Python. A escolha do suporte inicial à automação de scripts em Python é justificada pelo fato de (i) ser uma linguagem de fácil acesso e implementação, (ii) por contar com diversas bibliotecas para fins de *big data* e inteligência artificial (NumPy, TensorFlow, etc...), (iii) por ser largamente utilizada por desenvolvedores e (iv) pela facilidade de integração e automação dentro da API.

Durante a etapa de configuração de IF-Cloud, os *scripts* Python responsáveis por fazer processamento de dados podem ser carregados por meio de uma interface de usuário. Todos os *scripts* Python carregados ficam armazenados em um diretório específico e podem ser executados a qualquer tempo quando requisitados. Após a etapa de configuração, os demais *endpoints* de IF-Cloud estão relacionados ao processamento de dados por meio da execução dos scripts e a realização de requisições na API de CRUD de onde os dados para processamento são provenientes. As próximas Subseções detalham como configurar IF-Cloud, a interface de usuário e os principais *endpoints*.

Configuração de IF-Cloud para executar scripts

IF-Cloud necessita do preenchimento de um JSON de configuração para fins de controle do fluxo de dados da interface e da automação dos *scripts* Python.

Quadro 2 - JSON de configuração de IF-Cloud. Este NÃO é um recurso FHIR.

```
1  {
2     "resourceType": "Observation",
3     "id": "63f7f7a39c173811e7128d4c",
4     "scriptName": "huff.py",
5     "component":
6     {
7        "index": 2,
8        "changeField": "data",
9        "returnOnlyFieldsComponent": false
10    }
11 }
12
```

Quadro 2 mostra um exemplo de JSON com as informações importantes para a configuração de IF-Cloud conforme segue:

- `resourceType` - tipo de Recurso FHIR que IF-Cloud deverá buscar na API de CRUD;
- `id` - identificador do Recurso FHIR que a aplicação deverá buscar na API de CRUD;
- `scriptName` - nome do script disponível no diretório *Python SRC* a ser executado.
- `component` - Configura qual a chave do Recurso FHIR a ser buscado deve ser alterado ou retornado pelo script configurado em `scriptName`.
    - `changeField` - determina qual a chave do Recurso FHIR deverá ser alterada;
    - `index` - índice da chave `changeField` a ser alterado no Recurso FHIR;
    - `returnOnlyFieldsComponent` - se IF-Cloud irá retornar somente os campos alterados ou todo o Recurso FHIR para o solicitante.

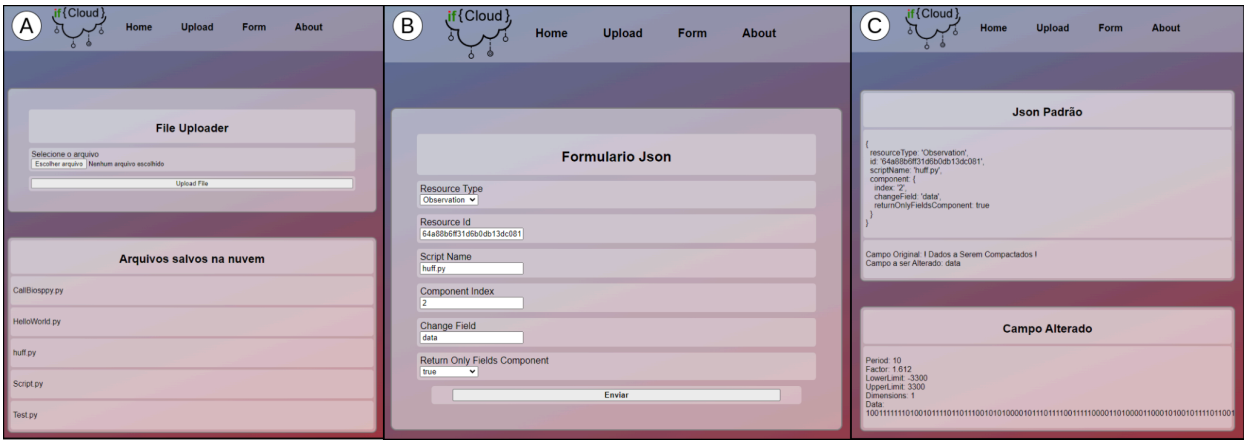
Detalhando como configurar o JSON de IF-Cloud, o desenvolvimento considera recursos FHIR *Observation* no Quadro 2, pois é o tipo de recurso onde são descritos os biossinais, estudo de caso deste trabalho. No exemplo do recurso *Observation* FHIR descrevendo um ECG (Quadro 1), podem haver até 12 chaves `data`, logo a configuração da chave `index` no Quadro 2 indica que a terceira chave `data` do recurso *Observation* identificado com o número `63f7f7a39c173811e7128d4c` vai ser a entrada do script em Python `huff.py` e, após a execução do script, a terceira chave `data` terá o conteúdo sobrescrito com a saída de `huff.py`.

Interface de Usuário do IF-Cloud

Para facilitar a utilização, IF-Cloud disponibiliza uma interface de usuário (UI - User Interface). Esta UI serve para *(i)* fornecer informações e instruções de uso do

IF-Cloud, *(ii)* realizar o *upload* de arquivos e *(iii)* preencher o formulário para configuração de IF-Cloud.

Figura 2 - Principais telas da UI de IF-Cloud são (A) o formulário de *upload* de *scripts*, (B) o formulário de configuração de IF-Cloud e (C) a tela de visualização da configuração após o envio do formulário de configuração.



Ao clicar no menu *Upload* na UI de IF-Cloud (Figura 2-A), o usuário pode realizar o *upload* de um *script* Python para utilização via IF-Cloud. Abaixo do formulário de *upload*, consta a lista de *scripts* disponíveis em IF-Cloud para processamento de dados e fornece uma visão dos arquivos que foram previamente carregados na nuvem. Arquivos fora da extensão `.py` não serão carregados por IF-Cloud e uma mensagem de erro informa caso arquivos de outros formatos tentem ser carregados. A versão atual de IF-Cloud também não suporta *scripts* compostos por múltiplos arquivos python, ou seja, somente um arquivo pode ser carregado por vez.

Uma das formas de carregar o JSON de configuração de IF-Cloud (Quadro 2) é utilizando o formulário para configuração e execução de FHIR Operation da interface gráfica de IF-Cloud (Figura 2-B). Ao clicar no menu *Form* da UI de IF-Cloud, o usuário acessa o formulário para configuração, onde os campos a serem preenchidos servem para montar o JSON de configuração de IF-Cloud (Quadro 2).

Ao enviar o formulário de configuração, IF-Cloud redireciona uma página informativa, que mostra em duas caixas de texto. A caixa *Json Padrão* mostra IF-Cloud configurado para substituir o conteúdo original da terceira chave `data` pela saída do script `huff.py`. A caixa *Campo Alterado* exibe todos os campos da chave `component` de um determinado índice de um *Observation*. No exemplo da Figura 2-C, a UI de IF-Cloud indica ao usuário que o conteúdo da chave `data` de índice 2 de um Recurso FHIR *Observation* foi substituído de "! Dados a Serem Compactados !" por

"10011...11001" enquanto os demais itens da chave `component` são exibidos inalterados (os demais itens correspondem às chaves do Recurso FHIR *Observation* exemplificados no Quadro 1).

Atualmente, IF-Cloud consegue processar scripts apenas com os dados em chaves dentro do vetor de objetos `component` de recursos FHIR *Observation*. A decisão de priorizar este campo é prática, pois as amostras de biossinais estarão, de fato, registradas como *Observation* e dentro de objetos JSON da chave `component`, onde o ecossistema de aplicações de saúde que motivam este trabalho (Figura 1) tende a atuar.

Endpoints Principais de IF-Cloud

Alternativamente a UI, o JSON de configuração também pode ser enviado para IF-Cloud no corpo da seguinte requisição POST:

- POST *{URL_IFCloud}/ifcloud/myForm*

onde *URL_IFCloud* é o endereço web onde IF-Cloud está implantado.

IF-Cloud executa um *script* de acordo com as configurações previstas no JSON Padrão (vide Quadro 2) e modifica o conteúdo de alguma chave de um recurso FHIR proveniente da API de CRUD na realização da seguinte requisição:

- GET *{URL_IFCloud}/run_script/operation/{id}*

onde o *id* é o identificador do recurso FHIR vindo da API de CRUD. Esta requisição GET sempre retorna um recurso FHIR como resposta. Esta é a rota que implementa a principal funcionalidade de IF-Cloud onde as aplicações web e os dispositivos IoT que necessitam processamento de dados devem requisitar.

Sempre que IF-Cloud receber uma requisição na rota principal, é necessário buscar os dados salvos na API de CRUD para serem sobrescritos com processamento intermediado pelos *scripts* salvos em IF-Cloud. Na arquitetura assumida neste trabalho (Figura 1), IF-Cloud realiza a seguinte requisição FHIR de consulta na API de CRUD:

- GET *{URL_API_CRUD}/{resource_type}/{id}*

onde *URL_API_CRUD* é o endereço web da API FHIR que vai fornecer os Recursos FHIR para IF-Cloud, *resource_type* e *id* são provenientes do JSON de configuração (chaves `resourceType` e `id` no Quadro 2). *URL_API_CRUD* é configurado como variável de ambiente na fase de implantação de IF-Cloud.

Resultados e Discussão

Esta Seção apresenta dois experimentos práticos para comprovar o funcionamento de IF-Cloud. O primeiro experimento foi conduzido utilizando algoritmo de Huffman, que é uma técnica de compactação de dados para reduzir seu tamanho sem perdas. Algoritmo de compactação é um dos casos de uso de aplicações que motivam este trabalho. O segundo experimento é mais específico e apresenta um ecossistema de aplicações hipotético para realização de eletrocardiogramas onde o papel de IF-Cloud é realizar o cálculo da frequência cardíaca.

IF-Cloud foi implantado no nível gratuito do serviço de nuvem da Amazon AWS[11] em uma instância do serviço EC2, onde uma máquina virtual Ubuntu foi configurada com a instalação de diversas bibliotecas Python para suporte aos *scripts* feita de forma manual e sob demanda. Outras alternativas de implantação de IF-Cloud na nuvem utilizando abordagem serverless ou containers são possíveis, mas não são o foco deste trabalho.O código-fonte de IF-Cloud está disponível em repositório público e gratuito com controle de versão[1].

IF-Cloud executando Compressão de Dados

O algoritmo de Huffman[12] é amplamente conhecido e uma versão em Python foi carregada em IF-Cloud usando a UI no menu *Upload*, conforme o exemplo da Figura 2. O Quadro 3 mostra o recurso FHIR original utilizado neste experimento proveniente de uma API de CRUD e o recurso FHIR alterado com o valor retornado da execução do *script* da codificação de Huffman. O valor da chave `data` originalmente era `"! Dados a Serem Compactados !"` e foi alterado para o valor binário compactado apresentado em forma de string `"10011…11001"`. Aparentemente, a string binária de saída é maior que o valor original, logo não houve compactação. Entretanto, o número binário resultante da operação é mostrado para fins didáticos, pois o mesmo binário codificado como ASCII (mesma codificação do valor original de entrada) seria uma sequência de 12 caracteres ilegíveis, o que é menor que os 29 caracteres originais da entrada. É importante ressaltar que o dado original é consultado da API de CRUD e modificado em IF-Cloud sem salvar nada no banco de dados original, ou seja, se uma aplicação quiser consultar o mesmo recurso FHIR sem alteração, basta realizar a requisição direta API de CRUD.

[1] https://github.com/if4health/ifcloud

Quadro 3 - Recurso FHIR *Observation* retornado de uma requisição GET na **(A)** API de CRUD e na **(B)** API IF-Cloud, mas com a chave `data` alterada pelo script *huff.py*. Chaves irrelevantes para este exemplo foram omitidas.

(A)

```
{
  "id": "64a88b6ff31d6b0db13dc081",
  "resourceType": "Observation",
  "component": [
    {      ...      },
    {      ...      },
    {
      "valueSampledData": {
        "data": "! Dados a Serem Compactados !"

      }
    }
  ]
}
```

(B)

```
{
  "id": "64a88b6ff31d6b0db13dc081",
  "resourceType": "Observation",
  "component": [
    {      ...      },
    {      ...      },
    {
      "valueSampledData": {
        "data": "1001111111010010111101101110010101000010111011110011111000011010000110001010010111101100 1"
      }
    }
  ]
}
```

IF-Cloud calculando Frequência Cardíaca de um ECG

Este experimento demonstra como IF-Cloud pode ser usado para integrar e prototipar projetos IoT, conforme o título deste trabalho. A realização de um ECG de forma remota é o estudo de caso escolhido para esta demonstração e, de acordo com o ecossistema de saúde digital estabelecido (Figura 1):

- **Dispositivo IoT** é um eletrocardiógrafo de 12 derivações, portátil para utilização fora de ambiente hospitalar, com conexão de internet Wi-Fi e programado para registrar e enviar registrar ECGs dentro do padrão FHIR;[13]
- **API de CRUD** é HAPI FHIR, uma API de código aberto, com servidor de testes público e que implementa completamente (todos os recursos) o padrão FHIR;[14]
- **API de processamento** é o IF-Cloud, a proposta deste trabalho;
- **Aplicação web** é um visualizador de ECG básico de funcionamento similar aos monitores multiparamétricos desenvolvido especialmente para este experimento.

Um trecho de 10 segundos de um ECG de uma pessoa saudável foi cadastrado como um Recurso FHIR *Observation* na API de CRUD. Esse trecho de ECG foi retirado do repositório de dados de pesquisas médicas Physionet.[15] Como HAPI FHIR é uma API pública para testes, enquanto a API estiver online e sem *drop* do banco de dados, o exato Recurso cadastrado para este experimento pode ser consultado com a seguinte requisição:

```
curl -X 'GET' \
  'https://hapi.fhir.org/baseR4/Observation/44712858' \
  -H 'accept: application/fhir+json'
```

O ECG cadastrado contém somente as amostras que compõem o sinal do ECG para exibição na aplicação web ou em um gráfico. Outras informações, como frequência cardíaca em batimentos por minuto (BPM), não estão presentes e precisam ser calculadas baseado no sinal disponível. Um *script* em Python chamado `calcBPM.py` para calcular a variação da frequência cardíaca foi desenvolvido utilizando as funcionalidades disponíveis no BioSPPy,[16] um processador digital de diversos biossinais. Esse *script* foi carregado em IF-Cloud utilizando a UI (Figura 2) e configurado para utilizar o Recurso *Observation* de identificador `44712858`. Diferentemente do Huffman, que não requer nenhuma biblioteca Python específica, a execução do script `calcBPM.py` requer a instalação do BioSSPy e as suas dependências na máquina virtual que hospeda IF-Cloud.

A Figura 1 destaca uma captura de tela da aplicação web durante a exibição de um ECG. O visualizador tem duas informações principais: (i) o biossinal ECG e (ii) a frequência cardíaca. A aplicação web precisa conhecer a URL de cada API e realizar duas requisições para montar a tela. Uma requisição consulta o sinal de ECG diretamente da API de CRUD que retorna um recurso FHIR com as informações necessárias para exibição, onde a chave `data` contém todas as amostras que compõem os 10 segundos do ECG (Quadro 4-A). A frequência cardíaca é calculada e fornecida por IF-Cloud, onde a chave `data` com as amostras do ECG é substituída por uma sequência de 11 números representando a variância da frequência cardíaca, em BPM, durante os 10 segundos (Quadro 4-B).

Quadro 4 - Recurso FHIR *Observation* retornado de uma requisição GET na **(A)** API de CRUD e na **(B)** API IF-Cloud, mas com a chave `data` alterada pelo script *calcBPM.py*. Chaves irrelevantes para este exemplo foram omitidas.

```
{
  "id": "44712858",
  "resourceType": "Observation",
  "component": [
    {
      "valueSampledData": {
        "data": "953.0 951.0 949.0 ... 934.0 936.0 935.0"
      }
    }
  ]
}
```

(A)

```
{
  "id": "44712858",
  "resourceType": "Observation",
  "component": [
    {
      "valueSampledData": {
        "data": "81.6 80.3 79.1 77.4 75.7 74.5 76.3 78.9 80.3 79.4 78.5"
      }
    }
  ]
}
```

(B)

A aplicação web foi desenvolvida para este experimento porque outros visualizadores de ECG online gratuitos careciam de alguma funcionalidade. Por

exemplo, o LightWave[17] da Physionet não é capaz de interpretar ECGs descritos em formato FHIR e não mostra a frequência cardíaca. O protótipo do visualizador de ECG também está disponível em repositório com código aberto[2].

Conclusão

Este trabalho apresentou IF-Cloud, uma API para prototipação e integração de softwares para a saúde baseada em nuvem interoperável em FHIR. O ponto de partida deste trabalho é um ecossistema de aplicações de saúde composto por pelo menos um dispositivo IoT, uma API que realiza as operações CRUD e uma aplicação web com interface de usuário. Por meio de uma interface de usuário simples, IF-Cloud suporta a configuração e o *upload* de *scripts python* para adicionar novas funcionalidades ao ecossistema acessíveis por requisições web em *endpoints*. Para demonstrar o funcionamento e a relevância de IF-Cloud, especificou-se o monitoramento contínuo de biossinais para o ecossistema de aplicações, onde diversas funcionalidades além do CRUD são pesquisadas e desejadas, mas que carecem da integração de ecossistema para validar as soluções. Durante os experimentos, dois scripts foram carregados em IF-Cloud para adicionar a compressão de dados e o cálculo de frequência cardíaca no ecossistema. Os testes feitos para validação do sistema demonstraram que IF-Cloud oferece recursos para acelerar a prototipagem de novas soluções em saúde digital, pois reduz o teste de novas funcionalidades a um *upload* de arquivos.

A continuidade do desenvolvimento de IF-Cloud está prevista com o foco na ampliação do suporte a outros tipos de recurso FHIR (atualmente IF-Cloud suporta apenas *Observation*) e outras linguagens de programação. Além disso, planeja-se a configuração dos endpoints do IF-Cloud como FHIR Operations, onde o nome da operação é prefixado com o caractere `"$"`. A exploração do paradigma *serverless* na arquitetura de software de IF-Cloud está no horizonte porque IF-Cloud inclui operações que são executadas esporadicamente no ecossistema de aplicações, uma característica que é aderente ao paradigma serverless.

[2] https://github.com/if4health/ecgmonitor

Referências

1. Szabó Z, Bilicki V. Access control of EHR records in a heterogeneous cloud infrastructure. Acta Cybernetica. 2021 Dec 7;25(2):485-516.

2. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials. 2015 Jun 15;17(4):2347-76.

3. Farahani B, Firouzi F, Chang V, Badaroglu M, Constant N, Mankodiya K. Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare. Future generation computer systems. 2018 Jan 1;78:659-76.

4. Benson T, Grieve G. Principles of health interoperability. Cham: Springer International. 2021:21-40.

5. FHIR Release 5 [Internet]. HL7.org. 2024. [Acessado em 15/04/2024]. Disponível em: https://www.hl7.org/fhir/

6. dos Santos LS, del Mestre Martins G, Itturriet FP, Machado JC, del Mestre Martins AL. Interoperabilidade e Segurança na Implementação de Aplicações Web de Saúde com SMART on FHIR. Journal of Health Informatics. 2023 Jul 20;15(Especial).

7. Abdelaziz AB, Rahimi MA, Alrabeiah MR, Ibrahim AB, Almaiman AS, Ragheb AM, Alshebeili SA. Photoplethysmography Data Reduction Using Truncated Singular Value Decomposition and Internet of Things Computing. Electronics. 2023 Jan 2;12(1):220.

8. Serhani MA, T. El Kassabi H, Ismail H, Nujum Navaz A. ECG monitoring systems: Review, architecture, processes, and key challenges. Sensors. 2020 Mar 24;20(6):1796.

9. Tejedor J, García CA, Márquez DG, Raya R, Otero A. Multiple physiological signals fusion techniques for improving heartbeat detection: A review. Sensors. 2019 Oct 29;19(21):4708.

10. Ayaz M, Pasha MF, Alzahrani MY, Budiarto R, Stiawan D. Correction: The Fast Health Interoperability Resources (FHIR) standard: systematic literature review of implementations, applications, challenges and opportunities. JMIR Med Inform. 2021 Aug 17;9(8):e32869.

11. AWS Free Tier [Internet]. Amazon Web Services. 2024. [Acessado em 15/04/2024]. Disponível em: https://aws.amazon.com/free/

12. Knuth DE. Dynamic huffman coding. Journal of algorithms. 1985 Jun 1;6(2):163-80.

13. Morás PL, del Mestre Martins AL, Itturriet FP. CardIoT - Eletrocardiógrafo Interoperável Baseado em Internet das Coisas. Anais do Computer on the Beach. 2023 May 3;14:416-23.

14. Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. circulation. 2000 Jun 13;101(23):e215-20.

15. HAPI FHIR - The Open Source FHIR API for Java. [Internet]. Smile CDR. 2024. [Acessado em 15/04/2024]. Disponível em: https://hapifhir.io/

16. Bota P, Silva R, Carreiras C, Fred A, da Silva HP. BioSPPy: A Python toolbox for physiological signal processing. SoftwareX. 2024 May 1;26:101712.

17. Moody GB. Lightwave: Waveform and annotation viewing and editing in a web browser. In Computing in Cardiology 2013 2013 Sep 22 (pp. 17-20). IEEE.