Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Câmpus Bragança Paulista - Av. Major Fernando Valle, 2013 - São Miguel - Bragança Paulista - SP, Brasil

Relatório Final De Desenvolvimento
PETHUB – plataforma para donos de Pets

Aluno: Matheus Mendes Silva

Orientadora: Talita de Paula C. de Souza

Coorientadora: Ana Cristina Gobbo César

[janeiro – novembro de 2018]

RESUMO

O Brasil tem 30 milhões de animais abandonados, sendo 20 milhões de cães e 10 milhões de gatos, segundo levantamento da Organização Mundial da Saúde (OMS), no ano de 2014. De acordo com o Instituto Fess'Kobbi, apenas 41% dos cães com lar são adotados, número esse que cresce para 85% quando se trata de felinos, dados que evidenciam a procura pela aquisição de animais. Para mudar esta realidade, podemos estabelecer uma relação entre as pessoas que querem doar um animal, com as que estão atrás de um novo companheiro e ajudar a dar lar aos animais desabrigados.Sendo assim, foi proposto o desenvolvimento de um aplicativo que atendesse a grande demanda de animais esperando para serem adotados. Grande parte dos animais adotados voltam às ruas, isso se deve pela falta de conscientização dos donos. A partir disso surgiu a possibilidade de agregar mais funcionalidades para o aplicativo, assim criando a ideia de uma plataforma central para donos de Pets, onde estes poderão ter acesso a informação confiável e bem estruturada através de filtros de busca, para simplificar a procura, e promover fácil acesso à informação.

Palavras-chave: Aplicativo, Doações, Pets, Plataforma, Central.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

Diariamente pessoas são bombardeadas por um enorme volume de informações vindas dos portais de informação, grupos e sites da Internet. Dentre essa diversidade de personas, estão os donos de Pets, buscando por ajuda, informações e interação com outros donos de Pets. Com esse grande volume de informações frente ao usuário, é difícil de separar o que é confiável, já que a internet é repleta de informações conturbadas e opiniões alheias.

De acordo com dados coletados em 2014 pela Organização Mundial da Saúde, somente no Brasil há 30 milhões de animais abandonados, sendo – em números aproximados – 20 milhões de cães e 10 milhões de gatos (Bom Dia Região, 2018).

Analisando estes dois problemas, surgiu a possibilidade de desenvolver uma solução que pudesse resolver, além destes, mais alguns outros aspectos relacionados à necessidade informacional desse público.

Os dados já apontam o enorme número de animais abandonados, mas a questão é que grande parte dos animais que são doados acabam voltando para as ruas. Assim, observamos que além de incentivar a adoção, devemos incentivar a concientização por parte de quem está adotando um novo animal, para então conceguir diminuir a taxa de animais desabrigados e abandonados.

Assim, desenvolvemos o PetHub, uma plataforma que além de combater o abandono e a aquisição de animais, tem o objetivo de facilitar a procura dos usuários por informações deste tópico, e promover dicas e guias de bons cuidados aos animais.

O diferencial que o PetHub irá proporcionar, é ter todas estas funções unificadas em uma só plataforma, para que o usuário encontre tudo que deseja em um único lugar. Além de prezar a transparência dos dados e estruturar as informações por filtros de busca, para simplificar a procura e promover fácil acesso à informação.

OBJETIVOS

A presente pesquisa propõe o desenvolvimento de uma plataforma central composta por três funcionalidades principais que, unificadas, tem o objetivo de facilitar o acesso à informação, além de combater o abandono de animais por meio da conscientização dos usuários e incentivo da adoção de animais.

Neste sentido, os principais objetivos desta pesquisa são:

- Desenvolver uma solução a partir das principais ferramentas utilizadas atualmente no mercado de desenvolvimento;
- Definir uma arquitetura para o desenvolvimento;
- Tornar público os dados relacionados aos animais disponíveis para adoção através de uma API (Pires, 2018);
- Suportar cadastro de cães e gatos para adoção e, disponibilizar filtros para melhorar a busca do usuário

2. FERRAMENTAS E TECNOLOGIAS

A seguir serão apresentadas as tecnologias disponíveis no mercado, assim como seus benefícios, para que no fim dessa sessão, seja compreendido e justificado as tecnologias utilizadas para o desenvolvimento do presente projeto.

Desenvolvimento nativo ou híbrido

Para atingirmos um dos objetivos de desenvolvimento, que é suporte a diversas plataformas – como smartphones, iPhones, navegadores etc. – há resumidamente duas opções: desenvolver uma solução individual para cada plataforma, ou desenvolver uma aplicação multiplataforma.

Para a primeira opção, necessitaríamos o domínio de diversas linguagens de programação – como Java, Swift, C#, dentre outras – que, além de terem alta curva de aprendizado, tornaria o desenvolvimento inviável, em quesito de tempo de desenvolvimento, já que sempre teríamos que iniciar um novo projeto do zero.

Já uma solução multiplataforma, isto é, híbrida, possibilita que um único código fonte seja transformado em aplicativos compatíveis com diversas plataformas como Android, IOS e WindowsPhone, utilizando a mesma arquitetura, SDK e IDE (Felix, 2018).

Isso é possível através do Apache Cordova, uma plataforma que fornece APIs JavaScript para desenvolvimento mobile, além de fornecer plug-ins que dão acesso às funções nativas do aparelho, como GPS, câmera, bussola, contatos etc.

Toda parte gráfica é gerada a partir das tecnologias web HTML, CSS e JavaScript. Na prática, teremos um grande contêiner chamado de WebView, que é basicamente um navegador minimalista, que pode ser publicado nas lojas de aplicativos e funciona tão bem quanto aplicações nativas.

Figura 1 - Representação de diferentes arquiteturas (Savvycom, 2018)

Apresentado esse breve comparativo, fica claro que para este desenvolvimento, dado os recursos e tempo disponível, optamos por uma aplicação híbrida. O aplicativo híbrido será desenvolvido com o Ionic framework, que além de adotar como base o Apache Cordova, traz consigo outro framework e linguagem para construção de soluções de mais alto nível, que são o Angular e o TypeScript.

Assim temos os recursos do Cordova e os componentes que o Angular fornece em um só framework para construir aplicativos híbridos. Isso fez do Ionic a opção escolhida para o nosso desenvolvimento.

Restful Web Services

Para os dados chegarem no aplicativo, eles devem estar disponíveis para consumo, por meio de um Web Service, que é uma solução usada na integração de sistemas e na comunicação entre aplicações. Assim, as nossas aplicações, e aplicações de terceiros poderão ter acesso aos dados, facilitando este tipo de integração.

Há diversas arquiteturas utilizadas, como SOAP, WSDL, REST dentre outras. Nesta sessão, iremos abordar sobre o REST (Rozlog, 2018).

O Rest é uma arquitetura mais recente em relação às supracitadas, e permite que o sistema solicitante acesse e manipule dados a partir de um conjunto predefinido de operações sem estado. Ele se comunica através do protocolo HTTP (Redação Oficina, 2018), acessando rotas definidas e identificando a operação utilizada na requisição, para então devolver a resposta ao serviço solicitante.

As informações são transmitidas pelo formato de notação de objetos do JavaScript – JSON – que é leve, e otimiza o consumo de banda no caso dos dispositivos móveis.

Os métodos HTTP das requisições serão traduzidos em operações no server. Resumidamente:

- GET - lista dados de um recurso
- POST - cria um novo recurso
- PUT- atualiza um recurso
- DELETE – apaga algum recurso

O recurso, por sua vez, é indicado na URL da requisição, ou seja, na rota:

Exemplo: http://54.233.88.185/api/doacao

Podem haver parâmetros, passados na própria URL:

Exemplo: http://54.233.88.185/api/doacao/73

Neste caso, é buscado um registro com identificador 73.

Laravel Framework

Atualmente há diversas tecnologias que auxiliam o desenvolvimento de uma API Restful, algumas opções mais conhecidas são ASP.NET, Node.JS e alguns frameworks do PHP.

A proposta do Laravel (Laravel, 2018), é promover o código mais organizado, bem estruturado e seguindo os padrões mais comuns de desenvolvimento no mercado. Dentre tudo isso está o grande suporte que ele oferece para o desenvolvimento de APIs, tendo até uma vertente – Laravel Lumen – dedicado à criação das mesmas. Mais detalhes no artigo (Lycan, 2018).

Assim, partindo da afinidade com a linguagem PHP, a vasta documentação encontrada no próprio site do framework e a simplicidade para criar uma API, seguimos o desenvolvimento com ele.

3. DESENVOLVIMENTO

API Restful

É importante primeiro, entender como o Laravel abstrai a camada de dados junto a classe PDO do PHP, que facilita a comunicação do PHP com o banco de dados relacional através da orientação a objetos. Ele basicamente cria migrações no banco de dados a partir da camada Modelo da aplicação. Assim, ele abstrai toda a manipulação de dados utilizando métodos dinâmicos, não sendo necessário escrever uma linha de código SQL.

Após compreendermos isso, devemos iniciar um novo projeto Laravel da forma que está descrita na documentação, e criarmos as novas classes Modelos gerando junto as migrações. Inicialmente a classe ‘Usuário’, e ‘Doacao’.

No arquivo de migração, é onde deve-se modelar as entidades, inserindo os campos, chaves e os tipos de dados. O sistema de gerenciamento de banco de dados pode ser escolhido de forma transparente nos arquivos de configurações do projeto, podendo ser alterado facilmente sem ter de reescrever as consultas e manipulações no banco de dados no projeto todo.

Com as migrações feitas e atualizadas, podemos configurar os recursos disponibilizados pela API, no arquivo de configuração de rotas.

Podemos criar os métodos básicos da API, através do método resource, que associa os métodos do HTTP diretamente ao método correspondente na classe de controle.

- GET – método index | GET com parâmetro – método show
- POST – método store
- PUT – método update
- DELETE – método destroy

Estes métodos resource fazem as quatro operações básicas do banco de dados:

- Index – recupera todos os recursos
- Show – recupera um registro específico
- Store – cria um novo recurso
- Put – Atualiza um recurso existente
- Delete – Exclui um recurso específico

A partir disso, essas quatro operações já podem ser realizadas através da rota definida. Só isso já é interessante, mas para nosso sistema precisamos de chamadas que processem outros dados, e contenham as regras de negócio.

Para isso definimos rotas especificas que direcionam para métodos específicos da classe de controle do recurso.

Para a API deste projeto, criamos uma rota que possibilitasse a busca de recurso a partir de filtros, que são passados como parâmetros na url desta forma:

api/busca/doacao/{cidade}/{porte?}/{pelagem?}/{castrado?}/{tipo?}

Os parâmetros com '?' são opcionais, os parâmetros são numéricos, e funcionam como filtros da consulta.

Como exemplo, uma consulta que retorna todos os registros de cães na cidade 'Braganca', com porte grande, pelagem longa, castrados ou não castrados: api/busca/doacao/Braganca/3/2/0/1

Valores de parâmetros:

Porte – 1:Pequeno, 2:Médio, 3:Grande;

Pelagem – 1:Curto, 2:Longo;

Castrado – 1:Não, 2:Sim

Tipo – 1:Cão, 2:Cachorro

Em todo caso, o valor 0 desativa o filtro.

Autenticação

Outra questão importante é a autenticação, como devemos proteger as rotas para que somente os usuários autenticados possam tem acesso? Há duas opções principais, o Laravel PassPort, e o JWT (Bemfica, 2018) que é muito usado, e tem suporte amplo para diversas tecnologias, assim como Laravel.

Ele funciona de maneira muito simples, a partir da requisição ele verifica os campos usuário e senha, se caso existirem então o usuário é autenticado, e como resposta é enviado um token, que é composto por 3 partes:

Header: contém informações sobre o token;

Payload: contém os dados, como id do dono do token, expiração e etc.;

Signature: Uma assinatura digital que garante que o token não foi alterado manualmente.

Exemplo de token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

Cada ponto final separa uma das partes

A partir do momento que as rotas são protegidas com o middleware do JWT, só é possível fazer o acesso enviando um token valido na URL através de query string, ou no header da requisição.

Quando há uma requisição para deletar um registro de doação por exemplo, ele necessita do token, pois para deletar um registro deve-se estar autenticado. Após ele entrar no método destroy, a classe controle recupera o token recebido, e verifica o identificador do dono na parte do Payload do token, assim ele executa um delete no banco de dados no registro que possui id igual o id passado como parâmetro, e com chave estrangeira 'usuario_dono' igual ao id retirado do token.

Levando em conta que os tokens se expiram rapidamente, foi necessário criar uma rota que renovasse, e enviasse como resposta uma nova chave. Assim o usuário não será desconectado por inatividade no sistema, basta programarmos para disparar essa requisição de renovação a cada período determinado de tempo.

Os token podem ser armazenados na memória local dos browsers, que mantem o usuário autenticado para a próxima vez, mesmo se ele sair da página. Um problema gerado por isso, é ter na memória um token expirado. Para corrigir isso, foi criado uma outra rota que simplesmente valida um token, e responde se o usuário continua conectado, ou deve renovar sua sessão.

Desenvolvendo a aplicação

Após os tópicos abordados, o desenvolvimento deve seguir para a parte da aplicação de fato. Como foi discutido, as tecnologias hibridas e detalhado sobre o Ionic framework que trabalha com Angular e TypeScript, achamos mais interessante por criar primeiro uma versão WEB de um projeto Angular que consumisse a API desenvolvida.

O objetivo seria ter mais produtividade através da afinidade que temos com o desenvolvimento web. E quando a solução estivesse em um nível razoável, boa parte do código seria aproveitado no Ionic, acelerando o processo de desenvolvimento.

Angular SPA

A abordagem utilizada no desenvolvimento dessa aplicação WEB, é um conceito bastante interessante conhecido como SPA – Single Page Application (Gil, 2018). Que resumidamente, é mostrar todo o conteúdo de uma aplicação em uma única página.

Isso traz benefícios durante o uso em performance, já que não há reload entre as páginas, que tira a ideia de lentidão. Assim, podemos consumir diversos

dados, e não há tempo de espera no carregamento das páginas por atraso de alguma resposta.

Na figura 2, abaixo, podemos observar melhor o comparativo entre o ciclo de uma aplicação SPA e uma aplicação tradicional.

Na aplicação tradicional, há uma requisição inicial, que traz como resposta a página a qual deseja ser acessada. E, após isso, na tentativa de enviar algum dado ao servidor, no caso através de um formulário via método POST, obtém-se uma nova página na resposta, que deverá ser renderizada pelo navegador.

Já no esquema da SPA, vemos que após a requisição inicial e o carregamento da página, quaisquer outros dados serão enviados via Ajax, que pode ser feito em background com a aplicação – sem necessidade de recarregar. Após os dados serem recebidos e processados pelo servidor, ele então envia o conteúdo da resposta no formato JSON já comentado.

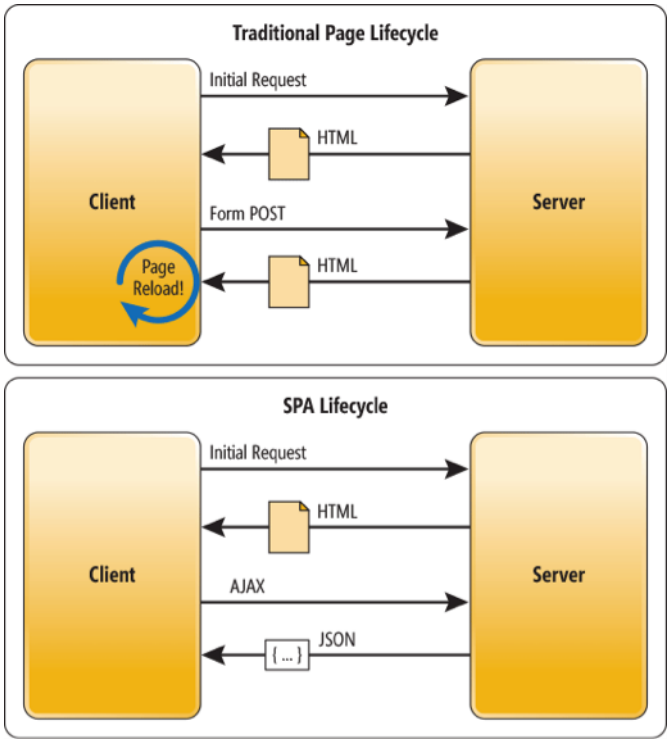

Figura 2 - Comparativo entre o funcionamento de uma SPA e um ciclo tradicional (Mendoza, 2018)

Após esse breve comparativo, podemos ver que a SPA atende muito bem nossos requisitos, já que precisaremos consumir os dados da API, que são transmitidos por JSON, e temos a necessidade do Ajax para fazer vários carregamentos na tela quase simultâneos, além de aumentar a performance geral do sistema. Portanto, foi a estratégia adotada.

4. RESULTADOS

Da forma a qual foi trabalhado no presente projeto, é observável não só os resultados práticos conquistados, mas também os avanços intelectuais e os novos conhecimentos que me foram proporcionados, fruto de bastante esforço desempenhado nessa proposta, que se tornaram motivos para continuar o desenvolvimento desta plataforma.

API Disponível

A API está hospedada em um servidor virtual AWS, e disponível através do endereço IP: http://54.233.88.185/api.

Figura 3 - Api em funcionamento

Figura 4 – Busca de doações na cidade 'Braganca'

Plataforma WEB

A plataforma web atualmente suporta o cadastro de novos usuários, assim como o cadastro de Pets na sessão PetMatch. As outras soluções citadas anteriormente estarão disponíveis em breve.

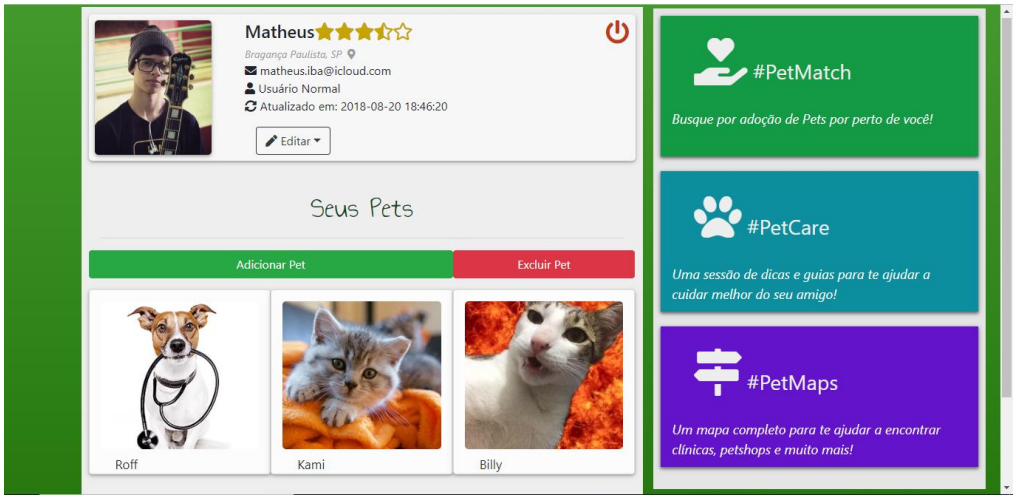

Figura 5 - Página de perfil do usuário - PetHub

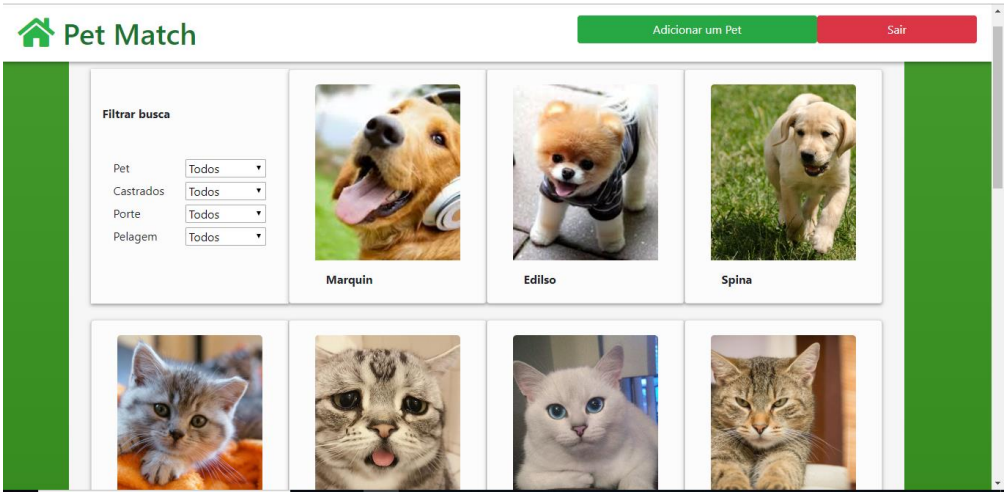

Figura 6 - Página PetMatch - PetHub

Figura 7 - Detalhes do Pet selecionado - PetHub

5. CONCLUSÃO

Ao termino deste relatório, é apropriado que façamos uma breve visão geral dos tópicos discutidos.

O principal problema evidenciado, é de grande escala. Sendo assim, a solução proposta no presente projeto remete a uma das possibilidades para diminuição do enorme número de animais abandonados, e tem como objetivo, atuar junto a outros meios de conscientização e fiscalização.

Outro grande proveito deste projeto, foi a grande quantidade de pesquisas e busca por informações que colaboraram com o término de minha formação, de forma que eu conhecesse as tecnologias mais recentes, aproximando-me mais da realidade no mercado de trabalho.

O estado atual do sistema já é bastante satisfatório e robusto. Mas, para a solução ser realmente disponibilizada no mercado e divulgada como plataforma para donos de pets, compreendem-se as próximas etapas de desenvolvimento:

- Disponibilizar as outras duas funções, PetCare e Pet, Maps;
- Registro de domínio, e melhora do servidor de hospedagem;
- Desenvolvimento do aplicativo híbrido com Ionic Framework;
- Disponibilizar nas lojas de aplicativos;
- Pensar no plano de negócios para sustentar a plataforma de forma comercial;

6. REFERÊNCIAS BIBLIOGRÁFICAS

al, C. e. (18 de 09 de 2018). *MY PETS: APLICATIVO MOBILE PARA AUXILIAR A DOAÇÃO DE ANIMAIS ANIMAIS DE.* Fonte: sbpcnet.

Bemfica, D. F. (18 de 09 de 2018). *Entendendo o JWT*. Fonte: Imasters: https://imasters.com.br/back-end/entendendo-o-jwt

Bom Dia Região. (23 de 02 de 2018). *Mais de 30 milhões de animais abandonados esperam uma chance*. Fonte: TV Tribuna: https://redeglobo.globo.com/sp/tvtribuna/Caominhada/noticia/mais-de-30-milhoes-de-animais-abandonados-esperam-uma-chance.ghtml

Felix, W. (18 de 09 de 2018). *6 aspectos essenciais para decidir entre aplicações mobile híbridas e nativas*. Fonte: Medium: https://medium.com/@waldyrfelix/6-aspectos-essenciais-para-decidir-entre-aplica%C3%A7%C3%B5es-mobile-h%C3%ADbridas-e-nativas-51bce0dace68

Folha de S. Paulo. (18 de 09 de 2018). *Gatos adotados são maioria no Brasil; donos de cães ainda preferem comprar*. Fonte: Folha de S. Paulo: https://f5.folha.uol.com.br/bichos/2016/08/gatos-adotados-sao-maioria-no-brasil-donos-de-caes-ainda-preferem-comprar.shtml

Gil, B. (18 de 09 de 2018). *Quando e porque desenvolver uma SPA (Single Page Application)*. Fonte: Vizir: https://vizir.com.br/2016/08/quando-e-porque-desenvolver-uma-spa-single-page-application/

Laravel. (19 de 09 de 2018). *Documentation*. Fonte: Laravel: https://laravel.com/docs/5.7

Lycan, R. (18 de 09 de 2018). *Construindo uma API RESTful com Laravel - Parte 1*. Fonte: Rafaell-lycan: https://rafaell-lycan.com/2015/construindo-restful-api-laravel-parte-1/

Mendoza, M. S. (18 de 09 de 2018). *SPA-LifeCycle*. Fonte: S Mendoza: https://www.smendoza.net/mean-angularjs/spa-lifecycle/

Pires, J. (18 de 09 de 2018). *O que é API? REST e RESTful? Conheça as definições e diferenças!* Fonte: Becode: https://becode.com.br/o-que-e-api-rest-e-restful/

Redação Oficina. (18 de 09 de 2018). *O protocolo HTTP*. Fonte: Oficina da Net: https://www.oficinadanet.com.br/artigo/459/o_protocolo_http

Rozlog, M. (18 de 09 de 2018). *REST e SOAP: Usar um dos dois ou ambos?* Fonte: InfoQ: https://www.infoq.com/br/articles/rest-soap-when-to-use-each

Savvycom. (18 de 09 de 2018). *Making the right decision*. Fonte: Savvycon Software: https://savvycomsoftware.com/making-right-decision/