

UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN  
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT  
DEPARTAMENTO DE INFORMÁTICA – DI

André Luiz de Souza Franco

**UMA API RESTFUL PARA OFERTA E DEMANDA DE SERVIÇOS  
PROFISSIONAIS**

MOSSORÓ - RN

2019

André Luiz de Souza Franco

**UMA API RESTFUL PARA OFERTA E DEMANDA DE SERVIÇOS  
PROFISSIONAIS**

Monografia apresentada à Universidade do Estado do Rio Grande do Norte como um dos pré-requisitos para obtenção do grau de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Sebastião Emídio Alves Filho.

MOSSORÓ - RN

2019



André Luiz de Souza Franco

**UMA API RESTFUL PARA OFERTA E DEMANDA DE SERVIÇOS  
PROFISSIONAIS**

Monografia apresentada como pré-requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovado em: 14 / 05 / 2019

Banca Examinadora

---

Prof. Dr. Sebastião Emídio Alves Filho (Orientador)  
Universidade do Estado do Rio Grande do Norte – UERN

---

Prof. Dr. Maximiliano Araújo da Silva Lopes  
Universidade do Estado do Rio Grande do Norte – UERN

---

Prof. Allysson Mendes de Oliveira  
Universidade do Estado do Rio Grande do Norte – UERN

## **AGRADECIMENTOS**

Agradeço, primeiramente, a Deus, que sem Ele nada disso seria possível. Agradeço também a minha família que sempre esteve do meu lado me apoiando e a minha namorada Ligia Maria, que sempre me ajudou e nunca me deixou desistir.

Meus agradecimentos também vão para meu orientador Prof. Dr. Sebastião Alves que esteve comigo desde o começo e me ajudando muito, e a Prof. Dra. Cicilia Maia que me apoio e ajudou.

*“Quando eu contei meus sonhos para alguém  
Me disseram: São grandes demais pra você,  
Mas com Deus foi bem diferente, Ele me  
disse: Vá em frente, eu contigo estou”  
(Leandro Borges)*

## RESUMO

O crescente aumento dos índices do setor de serviços, que impacta fortemente a economia brasileira, foi o responsável em 2018 pela alta do PIB. Para que este segmento continue a evoluir, a qualidade da prestação de serviços é imprescindível. Vários softwares foram desenvolvidos visando atender a este setor, mas cada um utiliza tecnologias próprias. Isto cria uma dependência com plataformas específicas e dificulta o compartilhamento de informações entre plataformas distintas. Para isso, este trabalho tem por objetivo contribuir na melhoria dos processos de contratação e oferta de serviços através da implementação de uma API RESTful que torna mais simples e ágil o desenvolvimento de sistemas para esta finalidade. Seguiu-se os princípios do estilo arquitetural REST e fez uso de Web Services, permitindo que aplicações implementadas em diferentes plataformas e linguagens de programação consumam os serviços da API RESTful. A implementação da API fez uso da linguagem de programação PHP, juntamente com a *framework* Laravel para facilitar e aumentar a velocidade do desenvolvimento. Utilizou-se o banco de dados relacional MySQL. A validação foi feita através de testes auxiliados pela ferramenta POSTMAN, onde obteve-se os resultados esperados comprovando a viabilidade e eficiência da API.

**Palavras chave:** Web Services, RESTful, Laravel, Oferta de serviços, Demanda de serviços.

## ABSTRACT

The growing increase in the indices of the services sector, which strongly impacts the Brazilian economy, was responsible in 2018 for the rise in GDP. For this segment to continue to evolve, the quality of service provision is imperative. Several software were intended to serve this sector, but were used in the use of proprietary technologies. This creates a dependency on specific stacks and makes it difficult to share information between the different platforms. To achieve this goal, this work aims to help in the improvement of the hiring process and the service offer through the implementation of an API RESTful, which makes it simpler and faster to develop the system with this objective. We followed the principles of REST architectural style and we used the Web Services, allowing developed applications in diverse platforms and programmatic language to consume API RESTfu services. The API developing used PHP programmatic language, along with Laravel framework to smooth and increase developing speed. We used the relational MySQL database. The confirmation was made through a series of tests assisted by POSTMAN tool, in which we got the expected results, confirming API viability and efficiency.

**Keywords:** Web Services , RESTful, Laravel, Service provision, Services demand.



## LISTA DE SIGLAS

API	Application Programming Interface
GPS	Global Positioning System
HATEOAS	Hypermedia As The Engine Of Application State
HAL	Hypertext Application Language
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IBGE	Instituto Brasileiro de Geografia e Estatística
JSON	Javascript Object Notation
MVC	Model-View-Controller
NoSQL	Not Only SQL
ORM	Object-relational mapping
PHP	PHP Hypertext Preprocessor
PIB	Produto Interno Bruto
REST	Representational State Transfer
RPC	Remote Procedure Call
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
TI	Tecnologia da Informação
URI	Uniform Resource Identifier
XML	Extensible Markup Language

## LISTA DE FIGURAS

Figura 1: Vagas de trabalho intermitente e parcial

Figura 2: Diagrama de Classes

Figura 3: Diagrama de Objetos

Figura 4: Diagrama de Componentes

Figura 5: Funcionamento de uma aplicação PHP

Figura 6: Exemplo de rota

Figura 7: Exemplo de Controller

Figura 8: Exemplo de Model

Figura 9: Tabelas do Banco de Dados

Figura 10: Teste da rota do cadastro do usuário

Figura 11: Resultado ao deletar um perfil

Figura 12: Cadastro de um serviço

Figura 13: Busca de um serviço através do ID

Figura 14: Busca realizada por filtros

Figura 15: Avaliação de um serviço

Figura 16: Listagem das avaliações de um serviço

Figura 17: Contratação de um serviço

Figura 18: Apresentação de um contrato

Figura 19: Contratos Solicitados

## **LISTA DE TABELAS**

Tabela 1: URI's

Tabela 2: Rotas dos Usuários

Tabela 3: Rotas dos Serviços

Tabela 4: Rotas dos Avaliações

Tabela 5: Rota dos contratos

## SUMÁRIO

<b>LISTA DE SIGLAS</b> .....	<b>9</b>
<b>LISTA DE FIGURAS</b> .....	<b>10</b>
<b>SUMÁRIO</b> .....	<b>16</b>
<b>1 INTRODUÇÃO</b> .....	<b>18</b>
1.1 Justificativa .....	18
1.2 Objetivos .....	20
1.3 Metodologia.....	20
1.4 Estrutura do Documento.....	21
<b>2 FUNDAMENTAÇÃO TEÓRICA</b> .....	<b>22</b>
<b>2.1 Setor de Serviços na Economia</b> .....	<b>22</b>
2.2 Aplicações Tecnologias .....	<b>Erro! Indicador não definido.</b>
2.3 Arquitetura de Serviços.....	24
<b>2.4 Web Services</b> .....	<b>26</b>
2.5 Arquitetura REST.....	26
2.6 Trabalhos Relacionados .....	30
2.6.1 APIS.....	30
2.6.2 Aplicativos.....	31
<b>3 Aplicação Proposta</b> .....	<b>32</b>
3.1 Visão Geral .....	32
3.2 Diagrama de Classes .....	32
3.3 Diagrama de Objetos .....	33
3.5 Diagrama de Componentes .....	34
<b>4 Implementação e Validação</b> .....	<b>36</b>
4.1 Tecnologias Utilizadas .....	36
4.1.2 Laravel.....	36
4.1.2 Banco de Dados .....	39
4.1.3 OAuth .....	40
4.2 Validação e Implementação.....	42

<b>5 CONSIDERAÇÕES FINAIS .....</b>	<b>52</b>
5.1 Trabalhos Futuros.....	52
<b>REFERÊNCIAS.....</b>	<b>54</b>
<b>APÊNDICES .....</b>	<b>58</b>
Apêndice A – Validação de rotas utilizando o POSTMAN.....	58

## 1 INTRODUÇÃO

O setor de serviços busca atender a uma necessidade pessoal ou uma necessidade empresarial. A prestação de serviços pode ser aplicado de várias formas, no qual as empresas e profissionais podem demonstrar as qualidades na prestação de serviços aos clientes (KOTLER; KELLER, 2012). O melhor serviço que uma empresa pode prestar ao seu cliente é resolver o problema do mesmo, com rapidez e eficiência tendo em vista a fidelização (TORRES, 2013).

Este segmento corresponde às seguintes subcategorias: serviços prestados às famílias, serviços de informação e comunicação, serviços profissionais, administrativos e complementares, serviços de transportes e correios, atividades imobiliárias, serviços de manutenção e reparação e outras atividades de serviços (serviços voltados para a agricultura, cultura, financeiro, comercial e pessoal) (IBGE, 2016).

“O conjunto dessas informações constitui a mais completa fonte de estatísticas sobre a estrutura produtiva do setor empresarial de serviços não financeiros no Brasil, fornecendo aos órgãos das esferas governamental e privada subsídios para o planejamento e a tomada de decisões, e, aos usuários em geral, informações para estudos setoriais mais aprofundados.” (IBGE, 2016).

Entretanto, os serviços não são considerados atividades industriais e nem de agropecuária, por esse motivo, o setor agrega uma série de atividades bastante heterogêneas (Souza et al., 2011). Dessa forma, é possível que os segmentos do setor sejam mais dinâmicos e apresentem maiores ganhos de produtividade, contribuindo para o crescimento da produtividade agregada da economia.

### 1.1 Justificativa

O setor de serviços no Brasil tem ganhado importância nos últimos anos. Este segmento engloba desde os serviços de baixo valor adicionado (como serviços de

limpeza e manutenção predial) até atividades com conteúdo tecnológico maior (como serviços de tecnologia associados às empresas) (SILVA et al, 2016).

Em 2018, o setor de serviços foi o grande responsável pelo crescimento Produto Interno Bruto (PIB) brasileiro, a alta foi de 1,3%. Este processo ocorreu devido a estabilização da economia, inflação e juros baixos, e um menor índice de desemprego comparando ao ano de 2017, o qual teve uma alta apenas de em 2017 havendo um avanço de 0,5 % (IBGE, 2018).

A tecnologia pode ser uma grande aliada para impulsionar ainda mais o crescimento do setor de serviços. Em alguns casos os clientes realizam as contratações às cegas, onde o mesmo não conhece bem o profissional contratado, e por vezes é feita através de alguma indicação. As plataformas de oferta e demanda de serviços existentes são fechadas e restritas apenas a aplicativos ou sites. Diante disso, este trabalho tem como objetivo principal contribuir na melhora dos processos de contratação e oferta de serviços profissionais através da implementação de uma API RESTful que torna mais simples e ágil o desenvolvimento de sistemas para esta finalidade.

Além disso, inclui a possibilidade de avaliar os serviços a fim de passar uma segurança e confiabilidade para o cliente ao realizar uma contratação. Uma API resulta numa interface de programação de aplicações, possibilita a integração de sistemas proporcionando uma segurança dos dados, monetização de acessos e facilidade na troca de informações dentre linguagens distintas.

A avaliação da qualidade dos serviços é muito importante, pois pode-se observar alguns aspectos, tais como: a confiabilidade que visa observar se os prestadores de serviço estão prestando o serviço oferecido de forma confiável e cuidadosa, segurança: conhecimento e atenção dos profissionais ao exibir a habilidade, a aparência dos serviços fornecidos, e empatia, ou seja, a atenção individualizada.

Esta avaliação da satisfação deve ser feita de forma ativa, perguntando e comunicando-se com os clientes. a qualidade é uma das características forte de um produto ou serviço que conferem aptidão para satisfazer necessidades explícitas ou implícitas (LOBO, 2010). Além disso, gera entre os usuários uma colaboração,

dispondo da vontade de ajuda mútua a proporcionar uma melhor experiência com os prestadores de serviços e clientes.

Sabe-se que os dispositivos móveis não são apenas uma alternativa de comunicação, mas também para resolver problemas do dia a dia, como por exemplo uma solicitação de serviços para a locomoção. De acordo com o IBGE (2010), 94.6% dos brasileiros fazem uso de aparelhos móveis com conexão à internet. Diante de todo o contexto exposto, faz-se viável o desenvolvimento desta monografia a fim de disponibilizar uma interface que torne mais ágil e eficiente a implementação de sistemas distribuídos para proporcionar a sociedade um meio digital de oferta e demanda de serviços de forma confiável.

## 1.2 Objetivos

O trabalho proposto tem como objetivo contribuir na melhora dos processos de contratação e oferta de serviços através da implementação de uma API RESTful que torna mais simples e ágil o desenvolvimento de sistemas para esta finalidade.

Tem-se como objetivos específicos:

- Permitir a utilização da API por sistemas implementados em plataformas distintas;
- Criar uma classificação dos serviços de acordo com o tipo
- Permitir que os serviços de profissionais cadastrados recebam avaliações;
- Validar a API com o POSTMAN e através de uma aplicação *mobile*.

## 1.3 Metodologia

Inicialmente foi realizado um estudo sobre o estilo arquitetural REST e os pacotes do Laravel (*framework* utilizada no desenvolvimento da API) a fim de implementar a API e seus recursos. Em seguida, foi feita uma pesquisa sobre a influência do setor de serviço dentro da economia brasileira, algumas aplicações tecnológicas desenvolvidas para o setor de serviços, arquitetura de serviços, *Web services* e arquitetura REST. Essa revisão tem como objetivo a formulação teórica com rigor científico em todas as etapas desenvolvidas e o estudo dos principais conceitos.



Foi feita a modelagem da aplicação, incluindo diagramas de classes, de objetos, componentes, além da modelagem do Banco de Dados. A API foi desenvolvida na linguagem orientada a objetos, PHP com o auxílio do *framework* Laravel visando acelerar o desenvolvimento e utilizou-se o banco de dados relacional MySQL, por ter um alto desempenho e facilidade de uso, além de conter capacidade para armazenar grandes quantidades de dados. Por fim, a validação de todos os *end-points* (pontos de acesso da API) foi realizada através de testes na ferramenta POSTMAN e em uma aplicação *mobile* intitulada como ProService.

#### **1.4 Estrutura do Documento**

O trabalho encontra-se organizado da seguinte forma:

- Capítulo 2: exhibe a fundamentação teórica para o trabalho proposto, expondo a influência do setor de serviços na economia, aplicações tecnológicas que buscam solucionar problemas neste setor. Assuntos mais específicos também foram abordados: o que são *Web Services*, arquitetura REST e o que é preciso para a API se tornar RESTful, trabalhos relacionados.
- Capítulo 3: aborda a aplicação proposta através do modelo de banco de dados usado na implementação, diagrama de classes, diagrama de objetos e diagrama de componentes.
- Capítulo 4: Descreve as tecnologias utilizadas no desenvolvimento da API proposta, e mostra os resultados dos testes realizados utilizando o POSTMAN e através de uma aplicação *mobile* intitulada como ProService.
- Capítulo 5: São apresentadas as considerações finais e as perspectivas de trabalhos futuros;

## 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados todos os assuntos relacionados ao trabalho proposto, iniciando pela relevância do setor de serviços na economia, alguns sistemas que buscam o mesmo propósito, e daí são apresentados conceitos em que se baseia este trabalho.

### 2.1 Setor de Serviços na Economia

O setor de serviços abrange todas as atividades econômicas cujo produto não é um bem físico, ou seja, imaterial. Um serviço é um bem consumível no momento em que é produzido, fornecendo então um valor agregado em diversas outras formas (por exemplo: conveniência, diversão, oportunidade, conforto ou saúde) que então representa um interesse intangível de seu comprador (CAZETI, 2013).

O setor que mais cresce na economia mundial é o segmento de serviços. No Brasil, representa cerca de 60% do PIB (Produto Interno Bruto), enquanto nos Estados Unidos, este segmento corresponde a mais de 75% do PIB; nos países latino-americanos, como a Argentina e a Colômbia, é superior a marca brasileira de 60% (Marcos Cobra, 2004).

Este segmento além de assumir uma posição de destaque na economia brasileira, segue uma tendência histórica de crescimento na geração de emprego. De acordo com o relatório de dados do IBGE juntamente com o comércio, os setores de serviços são responsáveis por mais de 75% das vagas de empregos criadas desde a reforma trabalhista (IBGE, 2018).

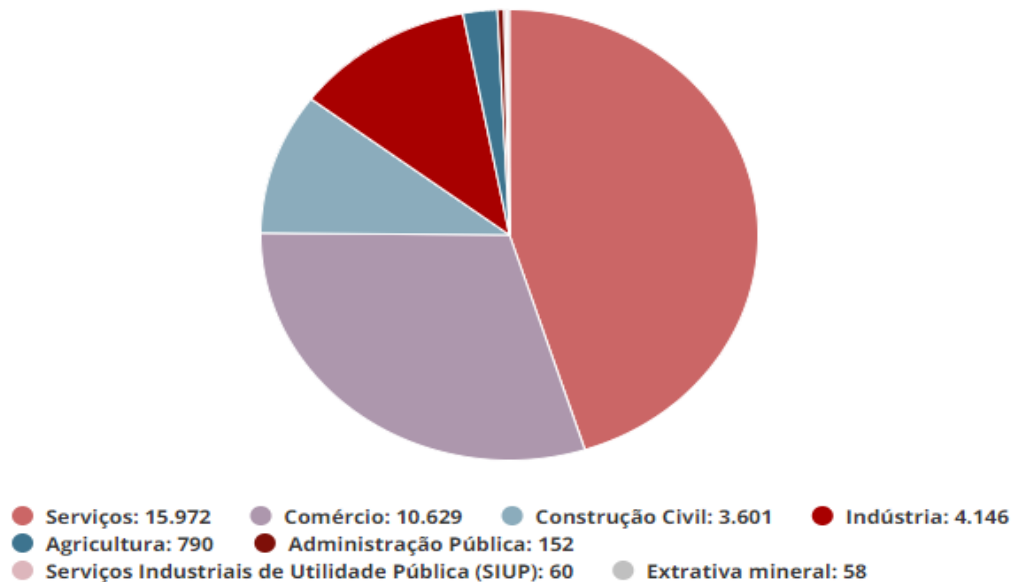
Em São Paulo o número de vagas de emprego neste setor dobrou em 2018 em relação ao mesmo período de 2014, além disso, nos oito primeiros meses de 2018, 131 mil vagas foram preenchidas. Estudos realizados em 2017 sobre a produtividade dos serviços descrevem o setor com produtividade relativamente elevada, composto por segmentos bastante heterogêneos (IG, 2018).

Na Figura 1, pode-se observar que no intervalo de tempo entre novembro de 2017 a junho de 2018, o número de vagas de trabalho intermitente e parcial no segmento de serviços corresponde a 15.992 vagas, proporcional a 45,15% do valor total. Houve um aumento de 35 mil vagas, sendo que mais de 15 mil foram advindas do setor de serviços em relação ao ano de 2016.

Figura 1: Vagas de trabalho intermitente e parcial

### Criação de vagas de trabalho intermitente e parcial

De novembro de 2017 a junho de 2018



Fonte: IBGE

A fim de aumentar a produtividade do setor de serviços o Sebrae (2018) afirma que é interessante promover soluções tecnológicas para facilitar a interação dos prestadores de serviços aos contratantes. Possibilitando assim, a realização das atividades de forma simples e rápida tornando os processos de forma mais confiável através das indicações realizadas por outros clientes.

## 2.2 Tecnologias Web

A evolução da internet provocou uma crescente expansão da tecnologia dos *smartphones* possibilitando a conexão dos colaboradores à redes sociais de forma mais espontânea. Passou a ser uma ferramenta para facilitar as atividades diárias e casuais, como por exemplo o GPS (Global Positioning System, que em português significa Sistema de Posicionamento Global), um aplicativo que auxilia no deslocamento fornecendo as direções para alcançar uma coordenada, melhorando o acesso e o compartilhamento de informações relevantes de tempo e espaço (Dickinson et al., 2015).

As tecnologias simplificaram também o funcionamento da economia colaborativa ou do compartilhamento. Belk (2010) e Schor (2014), afirmam que essa

economia assume um papel destacado e promissor globalmente. Permitem que os indivíduos percebam os benefícios de acesso e do uso dos serviços e produtos, ao invés da sua posse, economizando espaço, tempo, dinheiro e ainda tem a oportunidade de gerar novos relacionamentos com um consumo mais ativo e consciente (BOTSMAN;2011).

O compartilhamento de transportes, bicicletas e até mesmo hotéis e quartos tem sido um aspecto de expansão da economia colaborativa. O Airbnb é um serviço online comunitário disponível em múltiplas plataformas, visando a conexão dos anfitriões com os inquilinos. Os anfitriões disponibilizam seus imóveis para alugar e os indivíduos que viajam a passeio ou a trabalho podem reservarem acomodações de diversos preços, estilos e tamanhos através do aplicativo ou do site visando facilitar os trâmites dos viajantes.

O Uber por sua vez é uma nova modalidade de interligar motoristas com consumidores provocou uma mudança na dinâmica do mercado de transporte individual desde seu lançamento em diversos países em 2010. Uma grande inovação para facilitar ainda mais a vida do usuário mediante a correria foi no método de pagamento, o qual é feito de forma automática, clara e segura.

Dentre os setores que influenciam na economia colaborativa, há o setor de serviços onde a tecnologia passou a agregar novas características proporcionando facilidades à vida dos prestadores de serviços, o surgimento do autoatendimento é um exemplo disto. Unida com a internet possibilitou a realização das atividades mais rápidas, tornando as comunicações móveis, flexíveis, rápidas e diretas (Fitzsimmons et al., 2010, p. 19).

### **2.3 Arquitetura de Serviços**

De acordo com Hurwitz et. al. (2007), o termo arquitetura descreve a concepção global e a estrutura de um sistema de computador quando aplicado a ciência da computação. “A arquitetura de software de um programa ou sistema de computação é a estrutura ou estruturas do sistema, que compreendem os elementos de software, as propriedades visíveis externamente desses elementos e as relações entre eles”. tradução livre Bass(2003).

Dentre os modelos arquiteturais há a Arquitetura Orientada a Serviço ou *Service Oriented Architecture* (SOA). NEWCOMER (2004) define uma Arquitetura Orientada a Serviços (SOA) como um estilo de projeto que guia todos os aspectos de criação e utilização dos serviços de negócio durante seu ciclo de vida, definindo uma infraestrutura de TI que permite diferentes aplicações atuarem conjuntamente, trocando informações e participando do processo do negócio. Bean (2010) afirma que que o surgimento no meio da tecnologia do SOA foi a partir da evolução das arquiteturas RPC (*Remote Procedure Call*) e de protocolos de objetos distribuídos.

SOA é um estilo de projeto que trata das definições e do provisionamento da infra-estrutura de TI (Tecnologia da informação) permitindo que diferentes aplicações troquem dados e participem de processo de negócio independente dos sistemas operacionais onde estas aplicações estão executando ou linguagens de programação utilizadas para suas implementações (Newcomer e Lomow, 2005). É uma das propriedades dos serviços que espelham atividades de negócio do mundo real.

A associação entre os processos e os serviços é fundamental para o SOA atingir o alinhamento das plataformas aos negócios. Torna-se possível adequar os requisitos de tempo e agilidade existentes nas mudanças que ocorrem no negócio, sendo capaz de direcionar as mudanças em busca da redução de custos, melhorias de processos organizacionais, maior qualidade nos produtos gerados, e entre outros.

Destacam-se nas propostas sobre este modelo arquitetural, o foco na padronização de interfaces, reutilização de código e artefatos desenvolvidos, e disposição dos sistemas de informação em componentes (Mezo et al., 2008; Stal, 2002; Marks & Bell, 2006). Dentre os benefícios prometidos na literatura, encontramos flexibilidade e agilidade (Crawford et al., 2005; Kumar et al., 2007; Ciganek et al., 2005), a mitigação de dificuldades tecnológicas oriundas da obsolescência dos sistemas legados (Mezo et al., 2008) ou de problemas de utilização de outras tecnologias de integração (Corba, Dcom, COM+) (Kumar et al., 2007) e a maior facilidade de substituição de componentes e pacotes de software (Puschmann & Alt, 2001; Serman, 2007) por causa da menor dependência dos serviços em relação aos sistemas que os suportam.

## 2.4 Web Services

A capacidade de duas aplicações se comunicarem em máquinas diferentes, executadas em sistemas operacionais distintos atuando em conjunto, é denominada interoperabilidade (OLIVEIRA, 2018). Diante desta necessidade, no final da Década de 90 surgiu uma nova nomeação para sistemas distribuídos que permite a troca de dados na internet, chamados por *Web services*. De acordo com o W3C (*World Wide Web Consortium*), um *Web service* é um sistema de software projetado para suportar interações de máquina para máquina em uma rede (BOOTH et al., 2004).

Oliveira (2018) afirma que é uma aplicação típica de cliente-servidor, os clientes para consumirem os serviços precisam, apenas enviar requisições para os servidores onde encontram-se os recursos a serem manuseados. Os recursos proporcionam um alto grau de manutenibilidade, além disso, é capaz de separar a lógica da integração de negócios da implementação para que um desenvolvedor de possa focar na montagem de um aplicativo integrado, não sendo necessário se prender aos detalhes da implementação.

O *web service* é muito utilizado por existir a necessidade da integração de ambientes totalmente diferentes para integrar com os diferentes sistemas (Vigo e Oliveira 2014). Funcionam com a utilização do protocolo da camada de aplicação HTTP para o transporte dos dados, os quais são transferidos no formato XML (*eXtensible Markup Language*) e encapsulados pelo protocolo SOAP. No arquivo descritor possui as informações necessárias para que outros componentes interajam com o serviço, incluindo o formato das mensagens para as chamadas aos métodos e os protocolos de comunicação além das formas de localização.

## 2.5 Arquitetura REST

A arquitetura REST (*Representational State Transfer*) é uma coleção de princípios e restrições arquiteturais para o desenvolvimento de aplicações distribuídas na Web. Adota o paradigma cliente-servidor, onde as requisições partem inicialmente do cliente e são respondidas pelo servidor (FIELDING, 2000). É uma abordagem leve para o desenvolvimento de *Web Services*, que buscam simplicidade e baixo acoplamento.

O estilo arquitetural REST enfatiza a escalabilidade das interações entre componentes, a generalidade das interfaces, o desenvolvimento independente de componentes e a utilização de componentes intermediários que reduzem a latência das interações, reforçam a segurança e encapsulam sistemas legados (Fielding 2000). Para uma maior definição sobre o que consiste a arquitetura REST, alguns autores como Firno (2012) e Fredrich (2012) afirmam que a teoria gira em torno de alguns princípios básicos da arquitetura Web, tais como:

- **Interface Uniforme:** Define como funciona uma interface de comunicação entre clientes e servidores dentro da arquitetura utilizando o protocolo HTTP. Segundo Allamaraju (2010, p.3), neste protocolo é usado uma interface baseada nos métodos OPTIONS, GET, HEAD, POST, PUT, DELETE e TRACE, onde cada método opera apenas com um recurso, não mudando o comportamento, nem a sintaxe. Com isso é garantido que as informações que trafegam pela rede mantenham-se íntegras.
- **Múltiplas Representações:** Allamaraju (2010, p.262) afirma que “o documento que um servidor retorna a um cliente é a representação de um recurso ou serviço. A representação é o encapsulamento da informação em uma codificação específica, seja XML, JSON ou HTML”. O conceito de representações está em como as respostas de um serviço podem ser manipuladas. Estas respostas possuem um conjunto de informações de uma operação que foi ou que será executada. Este conjunto inclui a informação que será enviada, o serviço que será executado e a codificação em que as informações estão. Portanto, uma informação deve ter uma ou mais representações, no qual clientes e servidores usam o conceito de *Media Types*.
- **Armazenamento de informações:** Tendo em vista que um serviço REST é naturalmente *Stateless* (sem estado), serviços programados devem ter mecanismos que identifiquem se as informações das respostas de um servidor devem ou não serem armazenadas para posteriores utilizações, eliminando a necessidade de carregamento de informações com os mesmos conteúdos em determinadas partes do sistema WATSON (2011).

Fielding(2000) definiu seis restrições para determinar o estilo arquitetural REST, nomeadas como:

- **Cliente e Servidor:** A arquitetura cliente e servidor é bastante conhecida e utilizada em aplicações web e no REST ela tem o objetivo de separar as responsabilidades. Assim, o cliente não precisa se preocupar com tarefas como banco de dados, cache, escalabilidade, entre outros. Isso permite a evolução independente de cada um dos módulos facilitando a manutenção (COSTA et al., 2014).
- **Stateless:** Essa restrição defende que cada requisição seja independente, e que todas as informações necessárias para que o pedido seja atendido estejam na própria requisição. Assim, aumentando a escalabilidade e a confiança. Stateless significa que toda solicitação HTTP ocorre em um isolamento completo e o servidor nunca pode armazenar as informações das solicitações anteriores (RICHARDSON; RUBY, 2008).
- **Interface Uniforme:** É a principal restrição do REST. A utilização de uma interface uniforme é o que o distingue de todos os outros web services. O uso de interface uniforme tem o foco na simplicidade, acessibilidade, interoperabilidade e na capacidade de descoberta de recursos (COSTA et al., 2014). O importante sobre a interface uniforme é garantir que todo serviço use a interface do HTTP do mesmo modo, para que possamos ter serviços mais facilmente compreensíveis (RICHARDSON; RUBY, 2008).
- **Cacheable:** Os clientes podem armazenar em *cache* as respostas. As respostas devem, portanto, implícita ou explicitamente, definirem-se como sendo ou não cacheable, para evitar que os clientes reutilizem dados obsoletos ou inadequados em resposta a solicitações adicionais. Uma boa gestão da cache elimina parcial ou completamente algumas interações cliente-servidor, melhorando a escalabilidade e o desempenho.
- **Sistema em camadas:** A restrição do sistema em camadas define uma arquitetura REST como camadas hierárquicas de componentes, limitados à comunicação com os seus vizinhos imediatos, separando diferentes funcionalidades com o intuito de aperfeiçoar o requisito de escalabilidade.
- **Código sob Demanda:** É a única restrição opcional proposta por Fielding, tendo o objetivo de aumentar a flexibilidade do lado do cliente. Essa restrição permite que o cliente possa executar algum código sob demanda, dessa forma



diferentes clientes se comportam de maneiras distintas, mesmo utilizando exatamente os mesmos serviços oferecidos pelo servidor (FIELDING, 2000).

O REST diferentemente dos serviços da SOAP que trabalha com os conceitos de métodos e serviços, é movida através dos recursos onde são manipulados através do protocolo HTTP. Dessa forma utilizam os principais métodos como o GET (solicita recursos ao servidor), POST (envia dados ao servidor), PUT (realiza operações de escrita no servidor) e DELETE (solicita a exclusão de recursos ao servidor). É exatamente esse conceito de recursos que torna esta arquitetura tão simples, uma vez que cada recurso tem toda informação requerida, além de simplificar o acesso através de URLs lógicos, que são designados neste contexto por URIs.

De acordo com Matos (2013), os URIs (*Uniform Resource Identifier*) são responsáveis por tornar os recursos endereçáveis por todos os outros componentes do sistema e identificam um recurso. A tabela 1 apresenta que uma URI é composta por várias partes, começando pelo esquema utilizado (como http ou https), o nome do host seguido do caminho para um recurso.

Tabela 1: URIs

<b>Método</b>	<b>URI</b>	<b>Resposta</b>
GET	<a href="http://www.proservice.com.br/services">www.proservice.com.br/services</a>	Retorna todos os serviços
POST	<a href="http://www.proservice.com.br/services">www.proservice.com.br/services</a>	Insere um serviço no banco de dados
PUT	<a href="http://www.proservice.com.br/services/14">www.proservice.com.br/services/14</a>	Atualiza dados de um serviço com ID 14
DELETE	<a href="http://www.proservice.com.br/services/14">www.proservice.com.br/services/14</a>	Deleta um serviço do banco de dados

Fonte: Autoria própria

A arquitetura REST baseia-se nos níveis 0 a 2 da maturidade de de Richardson, Amundsen e Ruby (2013), no nível 0 é preciso a implementação do POX, “uma maneira de programar sem ter a preocupação com a definição dos recursos e o uso correto dos métodos e dos códigos de estado”.(ANDRADE, 2019). No nível 1 a API deve ser modelada conforme os recursos disponíveis e no nível 2 é composto pela implementação dos métodos do HTTP (GET, POST, PUT, DELETE).

A arquitetura RESTful é conhecida pela simplicidade, reaproveitando normas REST existentes, baseada em uma infra-estrutura generalizada. Uma API só é considerada RESTful se atingir o nível 3 da maturidade de Richardson conquistando a glória do REST, que consiste na inserção do HATEOAS (Hypermedia As The Engine Of Application State), o controle de hiperlinks possibilitando uma maior interação com a aplicação sem a necessidade do usuário conhecer a documentação (FIELDING, TAYLOR, 2000).

## **2.6 Trabalhos Relacionados**

Esta seção apresenta os trabalhos estudados similares a esta monografia, a qual se encontra subdividida em duas partes: mostrando as APIs desenvolvidas com o estilo arquitetural REST que atendem a contextos distintos e os sistemas que estão inseridos no mercado relacionados à oferta e demanda de serviços profissionais.

### **2.6.1 APIS**

Andrade (2019) em sua monografia desenvolveu uma API REST utilizando a linguagem JavaScript e a plataforma NodeJs, com auxílio da framework Express e do banco de dados ArangoDB. Foi desenvolvida para atender ao contexto da realização de reportes urbanos referente a cidade de Mossoró, utilizando geolocalização para que o usuário tenha a oportunidade de completar repórteres, realizar novos e visualizar os mais próximos. A aplicação é responsável por enviar as requisições dos cidadãos aos órgãos capazes de resolver os problemas das mais diversas categorias, além de possibilitar uma melhor interação entre os cidadãos e o poder público.

Silva (2018) desenvolveu em seu trabalho de conclusão de curso uma API na arquitetura REST em Node.js utilizando o framework Express.js com o banco de dados ArangoDB, oferece uma estrutura base para o implementação de aplicações cliente sem a necessidade de mudanças em seu núcleo, no contexto do agendamento dos serviços de reservas.

Oliveira (2018) desenvolveu em sua monografia uma aplicação distribuída com o estilo arquitetural REST desenvolvida em JavaScript auxiliado pela plataforma NodeJs utilizando dois bancos de dados NoSQL. A API foi utilizada no embasamento da plataforma Language Adviser que tem como objetivo de estabelecer o marketing

digital através do ensino de idiomas, promovendo publicidade para seus associados, atendendo a todos os requisitos para facilitar o ensino e aprendizado de um novo idioma.

### 2.6.2 Aplicativos

- **GetNinjas:** É um sistema web e mobile que visa aproximar os prestadores de serviços com os clientes. Utiliza geolocalização para enviar orçamentos dos profissionais mais próximos, o cliente realiza uma requisição da sua necessidade e a plataforma envia 3 orçamentos distintos a fim de que o mesmo escolha a melhor proposta.
- **Triider:** É uma plataforma web e mobile de oferta e demanda de serviços da cidade de Porto Alegre-RS, distingue as aplicações dos clientes e dos prestadores de serviço. Há uma maior rigorosidade para um cidadão tornar-se um prestador, é necessário uma análise profunda dos documentos e antecedentes criminais.
- **Linnuz:** É uma plataforma web e mobile com o intuito de ajudar os profissionais a divulgar seus serviços possibilitando os clientes a realizarem contratações online. Possui funcionalidades complexas em relação a efetuação do pagamento do serviço.
- **SosMarido:** É um aplicação móvel que possibilita a contratação de profissionais para serviços de manutenção de residências. O interessante é que o cliente ao selecionar os itens desejados o preço já é calculado automaticamente.
- **Bicos:** É um sistema web que possibilita a busca e a contratação de serviços profissionais, os profissionais têm a oportunidade de expor seus serviços através da adição de fotos em um portfólio, e os clientes têm a alternativa de avaliar os profissionais e adicionar comentários.
- **Sivirinos:** É um aplicativo que possibilita a contratação de serviços de forma muito rápida, com poucos clientes e de maneira muito simples através da realização de buscas pelo serviço desejado.
- **Bico certo:** É uma plataforma web e mobile específica da cidade de Montes Claros-MG, que possibilita aos usuários anunciarem sua profissão e os serviços que são capazes de oferecer. Não inclui contratação de serviços.

### **3 Aplicação Desenvolvida**

Este capítulo traz uma visão geral do objetivo da API abordando as funcionalidades que esta oferece e como encontra-se estruturada, exibindo os principais requisitos através dos diagramas de classes e de objetos, além do diagrama de componentes a fim de mostra a arquitetura detalhadamente.

#### **3.1 Visão Geral**

Como abordado nos capítulos anteriores, os índices do setor de serviço no Brasil têm crescido e possuído destaque na economia. Há várias aplicações voltadas para este contexto, visando aumentar mais ainda a taxa e proporcionar uma aproximação entre os clientes e prestadores de serviço. Para isto, é importante realizar a contratação com uma segurança e maior confiabilidade de que o prestador realmente é um bom profissional.

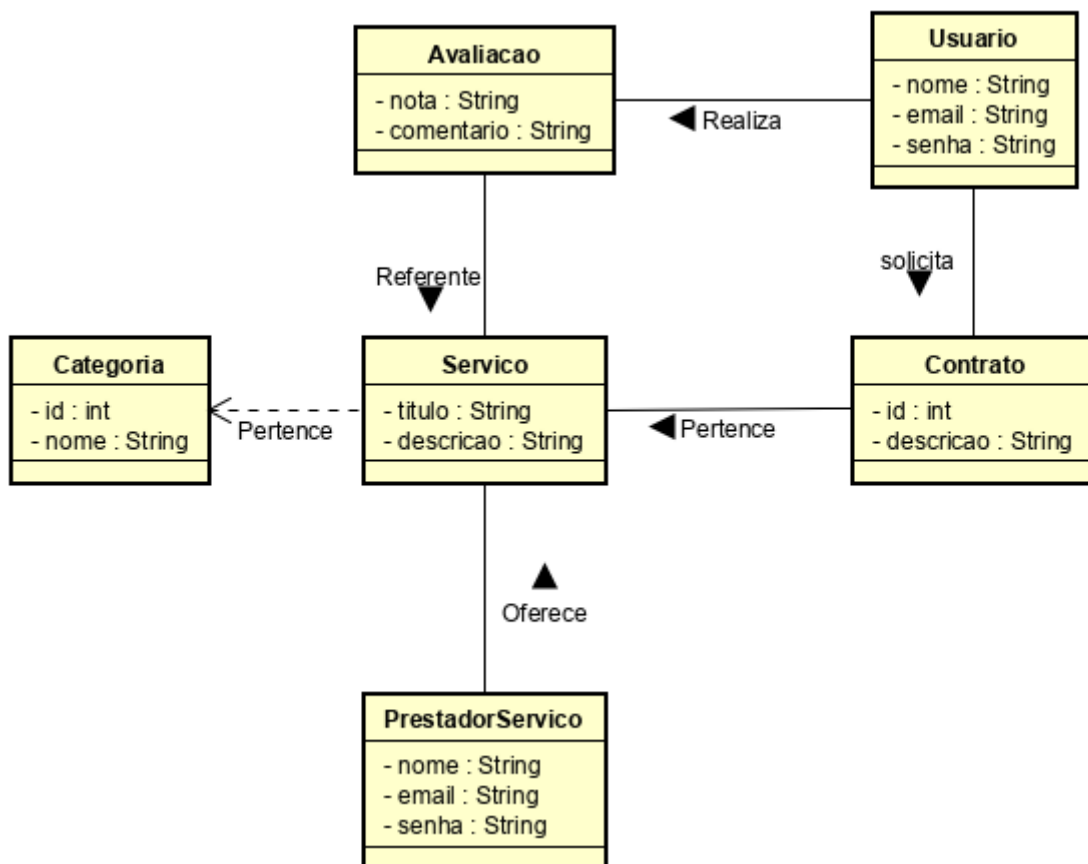
Para este fim, a API possibilita que os usuários realizem avaliações dos serviços contratados para colaborar com os demais clientes. Além disso, a API está implementada de forma genérica para atender a outros contextos que incluem as funcionalidades de realizar a solicitação e a contratação de um serviço, avaliação, buscas com filtros e os cadastros de usuários e serviços que serão ofertados.

#### **3.2 Diagrama de Classes**

Uma classe é a descrição de um tipo de objeto, onde todos os objetos são instâncias de classes, e a classe descreve as propriedades e comportamentos de um objeto determinado. Assim, um diagrama de classes demonstra a estrutura estática das classes de um sistema onde estas representam as entidades que são gerenciadas pela aplicação modelada. As classes podem se relacionar com outras através de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares).

A Figura 2 exibe as classes, os seus relacionamentos e os principais atributos da API implementada. A classe usuário se refere aos clientes que irão contratar os serviços dos Prestadores de Serviços, como já foi exposto neste capítulo a autenticação é realizada pelas credenciais do usuário, no caso o email e a senha. Um usuário pode realizar a busca de um serviço e oferecer que terá como campos o título e a descrição, este contém uma categoria. Após realizar a busca, o usuário pode solicitar um contrato e efetuar uma contratação, e ao final o cliente pode avaliar o serviço realizado pelo prestador. Novos atributos ou classes podem ser inseridos neste diagrama conforme o andamento da implementação.

Figura 2: Diagrama de classes



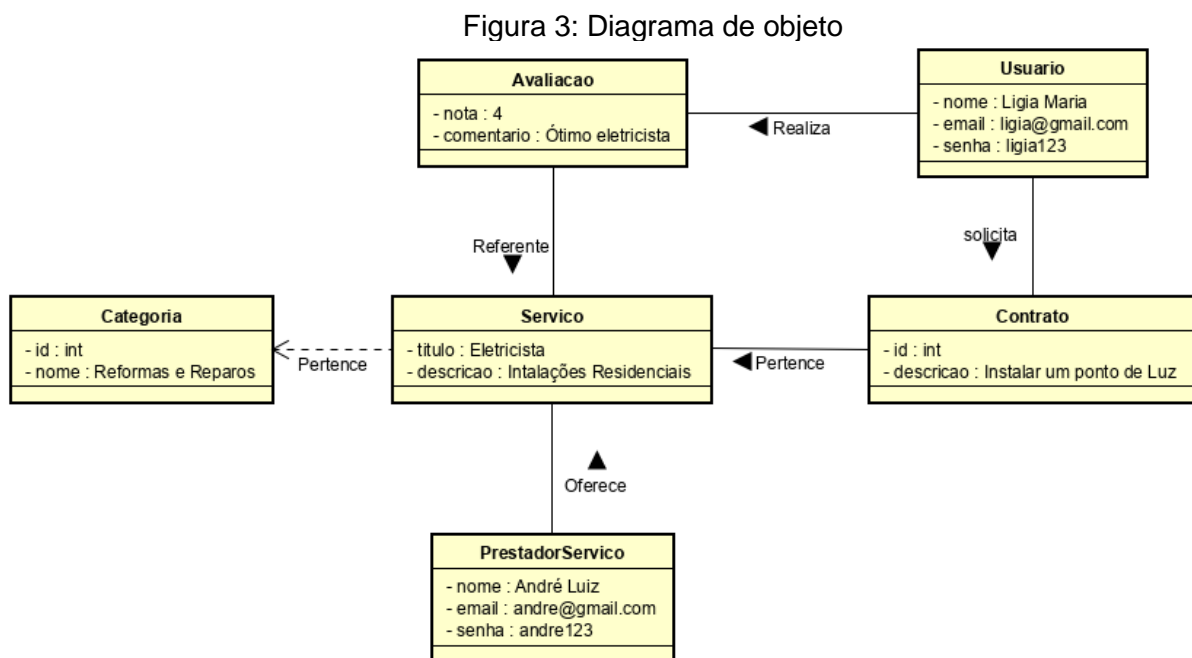
Fonte: Autoria própria

### 3.3 Diagrama de Objetos

Um objeto é um elemento que pode ser manipulado, acompanhando o comportamento. Desta forma, um diagrama de objetos exibe os objetos que foram

instanciados das classes. É como se fosse o perfil do sistema em um certo momento de sua execução. São muito úteis para exemplificar os diagramas de classes ajudando muito na compreensão. Diagramas de objetos também são usados como parte dos diagramas de colaboração, onde a colaboração dinâmica entre os objetos do sistema é mostrada.

Para uma melhor especificação, a Figura 3 modela através do diagrama de objeto um exemplo de um usuário realizando a contratação de um serviço. Propõe-se o seguinte cenário: Um usuário chamado André oferece um serviço de eletricista, o qual corresponde à categoria de reformas e reparos. A usuária Ligia maria, precisa do serviço de um eletricista, e solicita um contrato de um serviço oferecido por André Luiz. Após ter sido executado o contrato, Ligia Maria realiza uma avaliação referente ao serviço contratado.



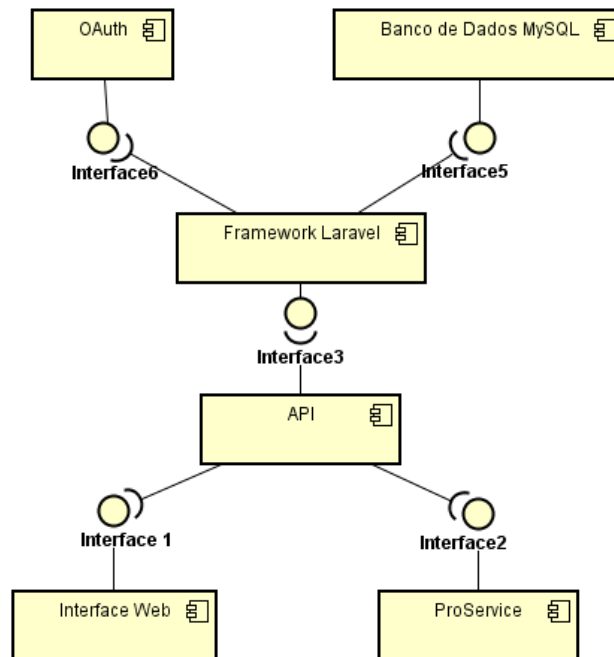
Fonte: Autoria própria

### 3.4 Diagrama de Componentes

Um componente pode ser tanto um código em linguagem de programação como um código executável já compilado. A Figura 4 corresponde ao Diagrama de componentes utilizado para apresentar a arquitetura evidenciando como a API está estruturada, além de exibir a interação entre os mesmos.

O componente API é a proposta deste trabalho, para a implementação utilizou-se a Framework Laravel, que através do *Eloquent ORM*, faz a conexão com o banco de dados e todas as consultas. O módulo OAuth é onde é feito todo o gerenciamento e manipulação dos tokens de acesso para cada usuário. A API é responsável por receber, analisar e enviar dados por meio do protocolo HTTP, sendo assim, qualquer tipo de aplicação que possa fazer requisições HTTP é possível consumir dados da API.

Figura 4: Diagrama de componentes



Fonte: Autoria própria

## 4 Implementação e Validação

Neste capítulo serão apresentadas as tecnologias que foram utilizadas para o desenvolvimento da API RESTful direcionada para a oferta e demanda de serviços profissionais e quais as suas principais vantagens, além de expor a validação e os testes que foram realizados no POSTMAN.

### 4.1 Tecnologias Utilizadas

Nesta seção são exibidas as tecnologias utilizadas neste trabalho, são elas a *framework* Laravel implementado na linguagem de programação PHP, a autenticação foi feita através do protocolo OAuth. Será apresentado as vantagens e os motivos pelos quais essas ferramentas foram escolhidas.

#### 4.1.1 Laravel

O PHP (*PHP Hypertext Preprocessor*) é uma linguagem de criação de *scripts*, ou seja, todo o processo de análise e processamento de informações é feito através de um interpretador, sendo assim, não é necessário compilar o código. Foi projetada especificamente para o desenvolvimento Web, é estruturada e orientada a objetos, atualmente encontra-se na versão 7.3. Esta linguagem não exige uma compreensão profunda das mais importantes linguagens de programação. [Converse e Joyce 2007]

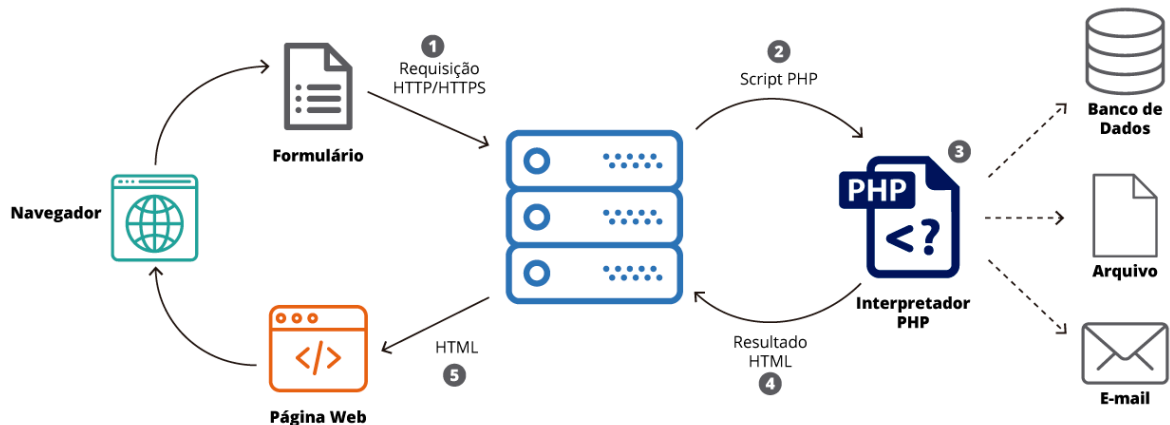
A Figura 5 mostra que o funcionamento de uma aplicação PHP, onde o cliente (navegador) envia requisições para o servidor utilizando protocolos de comunicação como HTTP ou HTTPS. Dentro do servidor essa requisição é processada pela aplicação PHP, fazendo as consultas necessárias, seja no banco de dados, arquivos ou até mesmo emails. Depois desta etapa, o interpretador retorna o resultado para o servidor em formato de HTML, e daí o servidor envia o resultado para a página web do navegador.

Dentre as *frameworks* existentes PHP, escolheu-se o Laravel por proporcionar uma escalabilidade robusta, de modo a melhorar a eficiência do desenvolvimento. Além de possuir bibliotecas específicas que facilitam o desenvolvimento de uma API RESTful, permite também que o processo de produção seja padronizado, automaticamente, permitindo que os desenvolvedores se concentrem na



implementação da lógica de negócios. Este *framework* utiliza o padrão de arquitetura MVC (*Model-View-Controller*), o mesmo proporciona benefícios na produção de um código mais limpo, facilidades para realizar atualizações.

Figura 5: Funcionamento de uma aplicação PHP



Fonte: Autoria Própria

A arquitetura MVC divide as responsabilidades para os *models* (modelos), *view* (visualização) e os *controllers* (controladores). Os *models* gerenciam os modelos de dados da aplicação, persistência da informação e acesso aos dados em si. As *view's* são as interfaces gráficas, ou seja, as telas da aplicação propriamente ditas, sendo a camada de comunicação com o usuário, manipulação e exibição dos dados. Os *controllers* realizam a comunicação da *view* com o sistema, recebendo e redirecionando as requisições, decidindo quais as *view's* exibir e realizando as requisições para o modelo. [Aihara 2009]

A Figura 6 mostra um exemplo de rota usado no laravel, no qual faz o roteamento de um método GET com a URI /home para um método *index* dentro da classe *HomeController*. A Figura 7 expõe como funciona o *controller* que recebeu esta rota. O controller é onde é feita toda lógica de negócio, neste case o método *index* chama o *model* Departamento e então pega todos os departamentos cadastrados. A Figura 8 apresenta o *model* Departamento, onde é definida a tabela do banco de dados que aquele *model* irá fazer as consultas. No *fillable* é definido quais os campos preenchíveis. Após a consulta, o *controller* chama a *view*, passando os dados necessários.

Figura 6: Exemplo de rota

```
Route::get('/home', 'HomeController@index');
```

Fonte: Autoria própria

Figura 7: Exemplo de *Controller*

```

HomeController.php x
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Departamento;
6  use Illuminate\Http\Request;
7
8  class HomeController extends Controller
9  {
10     /**
11      * Show the application dashboard.
12      *
13      * @return \Illuminate\Contracts\Support\Renderable
14      */
15     public function index()
16     {
17         $departamentos = Departamento::all();
18         return view( view: 'home', compact( varname: 'departamentos' ));
19     }
20 }

```

Fonte: Autoria própria

Figura 8: Exemplo de *Model*

```

Departamento.php x
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Departamento extends Model
8  {
9      protected $table = 'departamento';
10
11     protected $fillable = ['nome'];
12 }

```

Fonte: Autoria própria

Para se comunicar com o banco de dados o Laravel utiliza uma implementação simples do *ActiveRecord* chamada de *Eloquent ORM* que é uma ferramenta que traz várias funcionalidades para facilitar a inserção, atualização, busca e exclusão de registros em tabelas. Conta com uma configuração simples e pequena, com pouco código é possível configurar a conexão com banco de dados e trabalhar com os recursos que o mesmo oferece.

#### 4.1.2 MySQL

Banco de Dados são coleções de informações que se relacionam e que são armazenadas a fim de garantir a persistência dos dados. Foi utilizado o MySQL no desenvolvimento da API, no qual é um banco de dados relacional, ou seja, armazena os dados em tabelas organizadas em colunas, onde cada coluna contém um tipo de dado. Cada tabela contém uma chave para uma melhor identificação e rapidez na busca, e para realizar os relacionamentos.

O MySQL é um servidor e gerenciador de banco de dados (SGBD) relacional, de licença dupla (sendo uma delas de software livre) projetado inicialmente para trabalhar com aplicações de pequeno e médio porte, mas hoje atendendo a aplicações de grande porte e com vantagens destacantes comparados a outros bancos de dados. Possui todas as características que um banco de dados de grande porte precisa sendo reconhecido por algumas entidades como o banco de dados *open source* com maior capacidade para concorrer com programas similares de código fechado, tais como *SQL Server*(da Microsoft) e *Oracle*. [milani,2006].

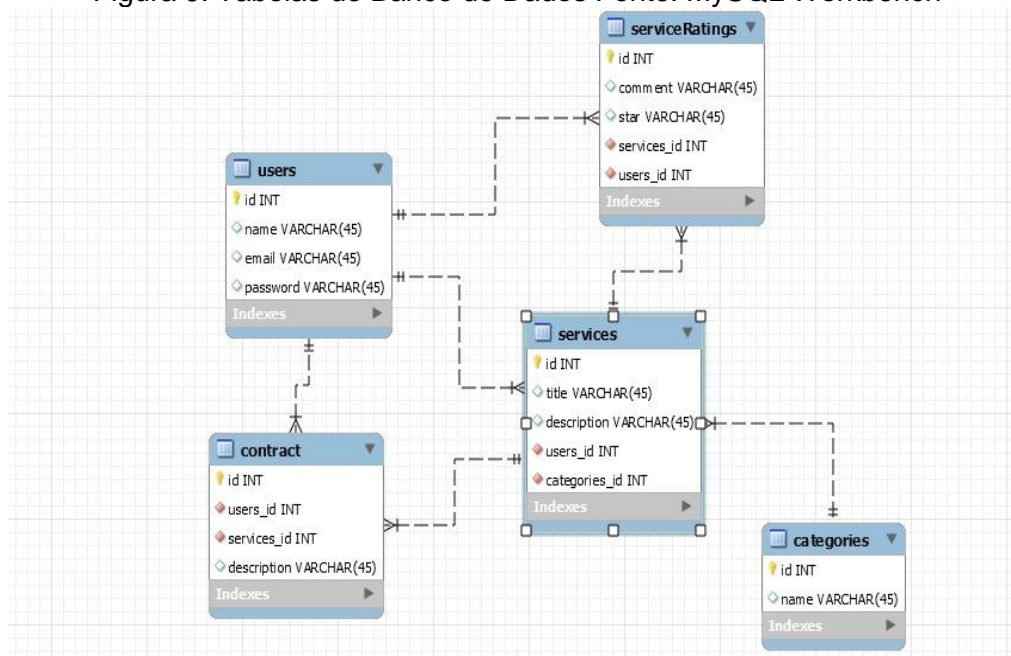
O servidor MySQL é mais rápido e flexível o suficiente para permitir armazenar logs e figuras. As principais vantagens são o desempenho, segurança por conter integridade referencial, *backup* e *restore*, controle de usuários e acessos, e ainda, se necessário, verificação e correção de corrompimento de tabelas, é um banco robusto e fácil de usar.

Resumidamente, a Figura 9 apresenta as tabelas do Banco de Dados usado na API. A tabela de usuários é onde vai ficar armazenada todas as informações dos usuários/prestadores de serviço. Um usuário pode cadastrar vários serviços, mas o serviço só pode estar relacionado com um usuário, isto caracteriza uma relação de um para muitos. A tabela de categorias serve para complementar a de serviços, em

cada serviço só é permitido estar relacionado com uma categoria e uma categoria a vários serviços.

A relação entre um usuário e os contratos também é de um para muitos, onde o usuário poderá fazer várias contratações de serviços. O serviço é capaz de ter várias avaliações, em que cada avaliação está relacionada a um serviço e ao usuário que fez a avaliação. Para a autenticação foi utilizado os campos email e senha, os níveis de permissão foi gerenciado por um pacote do Laravel chamado Passport, o qual tem implementado o protocolo OAuth.

Figura 9: Tabelas do Banco de Dados Fonte: MySQL Workbench



Fonte: MySQL Workbench

### 4.1.3 OAuth

O OAuth 2 é uma estrutura de autorização que permite que as aplicações obtenham acesso limitado às contas de usuários de um determinado serviço, através do protocolo HTTP. Empresas mundiais disponibilizam deste protocolo para várias aplicações, como a Facebook, Google, GitHub, e entre outras. Este protocolo foi utilizado na API proposta, pois proporciona grande segurança e facilidade na geração e gerenciamento de tokens de acesso.

Esta estrutura se concentra na simplicidade do desenvolvimento da parte do cliente e fornece fluxos de autorização específicos para aplicativos da Web, desktop

e telefones celulares. Esta especificação e suas extensões estão sendo desenvolvidas dentro do IETF *OAuth Working Group* (RFC,6749).

O escopo é um mecanismo do OAuth 2.0 para limitar o acesso de um aplicativo à conta de um usuário. Um aplicativo pode solicitar um ou mais escopos, essas informações são então apresentadas ao usuário na tela de consentimento e o token de acesso emitido para o aplicativo será limitado aos escopos concedidos. A especificação OAuth permite que o servidor de autorização ou usuário modifique os escopos concedidos ao aplicativo em comparação com o que é solicitado, embora não haja muitos exemplos de serviços que fazem isso na prática.

OAuth não define nenhum valor específico para escopos, pois é altamente dependente da arquitetura interna e das necessidades do serviço. Nesta estrutura são definidos quatro papéis:

- *Resource Owner*(Proprietário do recurso): É o usuário que autoriza uma aplicação a acessar sua conta. Nem sempre é uma pessoa. Quando ele é uma pessoa ele é o usuário final.
- *Resource Server*(Servidor de Recursos): o servidor que possui os recursos protegidos e recebe as requisições para acessar esses recursos.
- *Authorization Server*(Servidor de Autorização): é um servidor que verifica a identidade do usuário e daí emite tokens de acesso para a aplicação.
- *Client*(Cliente): é a aplicação que acessa os recursos no *resource server* em nome do usuário. O "*client*" pode ser qualquer tipo de aplicação que faça isso.

Com essas definições, o OAuth estabelece quando uma aplicação "*client*" precisa acessar um recurso protegido no "*resource server*" ele deve obter um "token de acesso". Esse "token de acesso" contém as informações que caracterizam o acesso que o *resource owner* permitiu aos recursos protegidos.

Percebe-se que isso resolve os problemas especificados, pois o "*client*" não precisa das credenciais do "*resource owner*", somente do token gerado pelo "*authorization server*", que é uma aplicação confiável. Através do token, o "*resource owner*" pode especificar um acesso personalizado, e permitir ao "*client*" o acesso somente a um subconjunto dos recursos protegidos. Além disso, é fácil revogar o

acesso, pois basta invalidar o token concedido, sem falar que os tokens possuem um tempo de vida específico e não podem ser usados para sempre.

## 4.2 Validação

Os testes realizados para validar a API foram feitos através da ferramenta POSTMAN, tendo como principal objetivo observar o comportamento dos recursos implementados na API. Em cada requisição foi verificada a estrutura de dados enviada, verificando se está conforme o que foi estabelecido e se o retorno da requisição contém as informações necessárias, como por exemplo o status da resposta.

O POSTMAN é uma ferramenta muito simples e intuitiva, usada por grande parte dos desenvolvedores de API's. Com ela é possível testar rotas e requisições através de verbos HTTP (GET, POST, PUT, DELETE) de maneira muito simples. Além disso, pode-se realizar o teste de desempenho de uma API executando um conjunto de solicitações para verificar seu desempenho de resposta.

As rotas são os *end points* da API, através delas é possível acessar todos os recursos disponibilizados. Cada rota possui um verbo HTTP e uma URI. Cada *end point* precisa ser único para que a API possua uma interface uniforme, que foi uma das restrições propostas por Fielding e Taylor (2000).

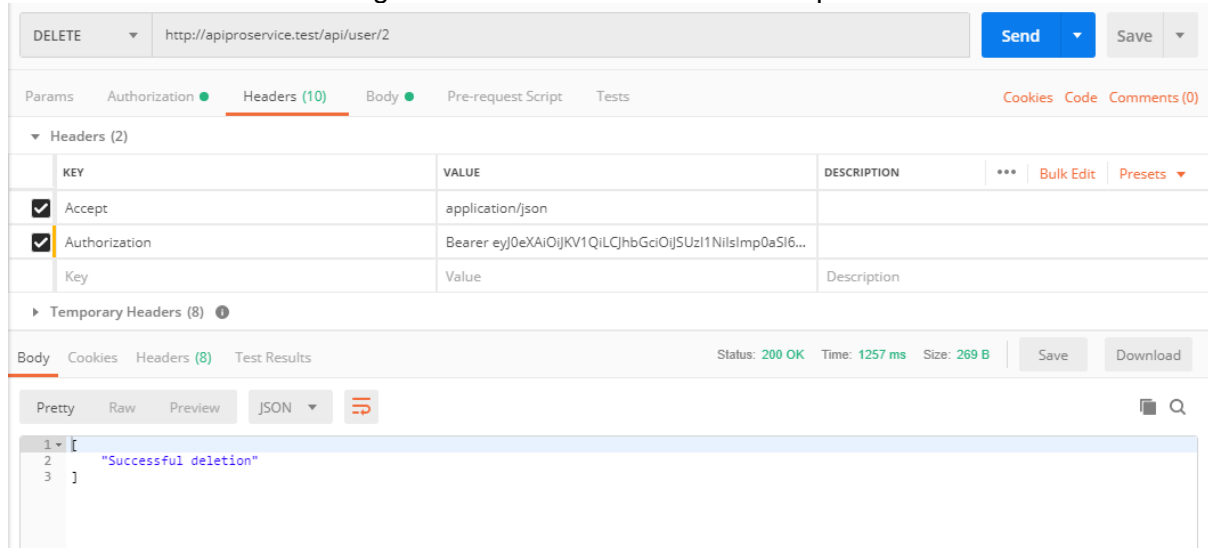
A API proposta foi dividida em, basicamente, 4 recursos: Usuários, Serviços, Avaliações e os Contratos. A implementação atingiu o nível 3 da maturidade de Richardson conquistando a glória do REST, o qual consiste na inclusão dos HATEOAS equivalente ao controle das hipermídias tornando este trabalho uma API RESTful.

Para cada recurso existem várias rotas e para que todas as funcionalidades sejam executadas o usuário necessita estar autenticado no sistema. A Tabela 2 apresenta as rotas referente ao recurso dos usuários, os métodos que foram utilizados e a descrição de cada rota. A primeira rota é para realizar o cadastro dos usuários no sistema, para isto, é necessário preencher os campos com o nome, email, senha e confirmação de senha, como resposta é retornado um JSON utilizando o padrão HAL(Hypertext Application Language) com o token de autenticação gerado através do OAuth e o nome do usuário autenticado, como mostra a Figura 10.



Foi implementado também a opção de listar os usuários, realizar a buscar pelo nome, atualizar os dados do perfil e deletar o usuário do sistema. Na Figura 11 mostra o comportamento da API ao deletar um perfil.

Figura 11: Resultado ao deletar um perfil



Fonte: POSTMAN

A tabela 3 exibe as rotas implementadas pertencente aos serviços, qualquer usuário poderá cadastrar um serviço, para isto é necessário estar logado no sistema e preencher devidamente os campos obrigatório que são o título, o ID da Categoria relacionada, e a descrição que é um campo considerado opcional. Na Figura 12 mostra a resposta da API para a realização de um cadastro.

Tabela 3: Rotas dos Serviços

URI	MÉTODO	DESCRIÇÃO
api/services	POST	Cadastra um serviço, relacionado ao usuário autenticado
api/services/{id}	GET	Busca serviço pelo ID
api/services/	GET	Lista todos os serviços
api/services/{id}	DELETE	Deleta um serviço
api/services/{id}	PUT	Edita dados de um serviço

Fonte: Autoria própria

Figura 12: Cadastro de um serviço



POST http://apiproservice.test/api/services

Params Authorization Headers (10) **Body** Pre-request Script Tests Cookies Code Comments (0)

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> title	Mecanico	
<input checked="" type="checkbox"/> description	Mecanica automotiva	
<input checked="" type="checkbox"/> category_id	4	
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 5271 ms Size: 368 B Save Download

Pretty Raw Preview JSON

```

1 {
2   "Success": "Service registered successfully",
3   "_links": {
4     "self": {
5       "href": "http://apiproservice.test/api/services/204"
6     }
7   }
8 }

```

Fonte: POSTMAN

A fim de realizar buscas dos serviços é preciso passar como parâmetro o id do serviço e tendo a possibilidade de realizar uma filtragem indicando qual a categoria pertencente e o id do profissional. A Figura 13 exibe uma busca realizada com a inserção do id do serviço e a Figura 14 apresenta a resposta da busca feita com a inclusão de filtros.

Figura 13: Busca de um serviço através do ID

GET http://apiproservice.test/api/services/204

Body Cookies Headers (8) Test Results Status: 200 OK Time: 2146 ms Size: 615 B Save Download

Pretty Raw Preview JSON

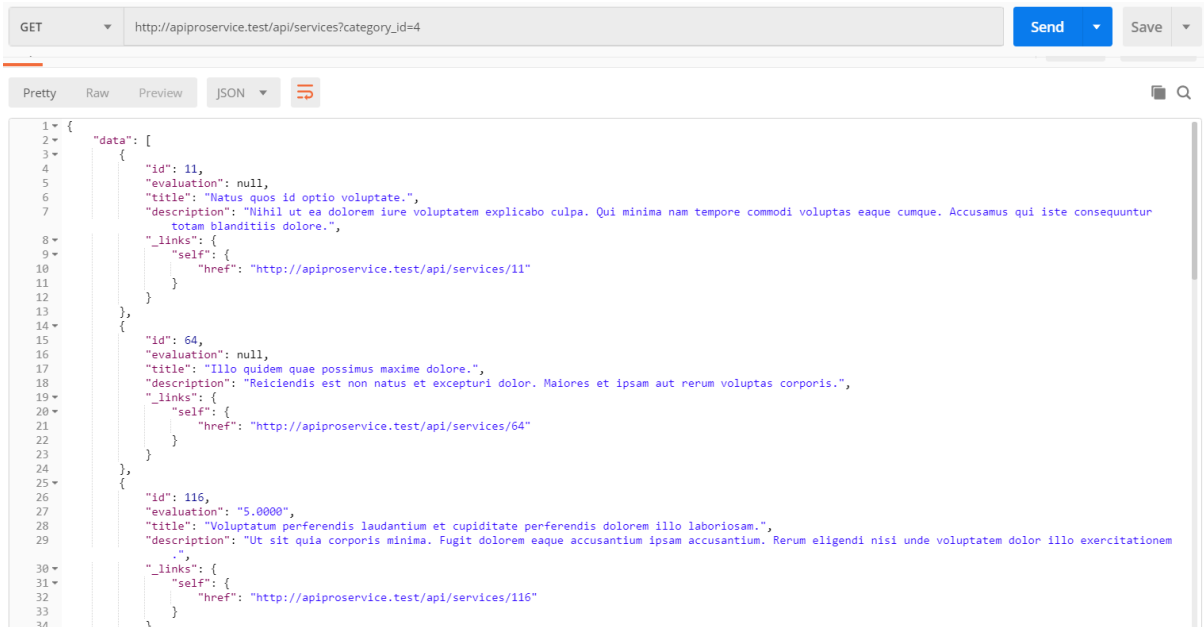
```

1 {
2   "data": {
3     "type": "Service",
4     "id": 204,
5     "user_id": 101,
6     "avaliation": null,
7     "title": "Mecanico",
8     "description": "Mecanica automotiva",
9     "category": "dolorem",
10    "_links": {
11      "self": {
12        "href": "http://apiproservice.test/api/services/204"
13      }
14    },
15    "_embedded": {
16      "category": {
17        "id": 4,
18        "name": "dolorem"
19      },
20      "Ratings": {
21        "links": {
22          "self": {
23            "href": "http://apiproservice.test/api/services/204/ratings"
24          }
25        }
26      }
27    }
28  }
29 }

```

Fonte: POSTMAN

Figura 14: Busca realizada por filtros



```

1 {
2   "data": [
3     {
4       "id": 11,
5       "evaluation": null,
6       "title": "Natus quos id optio voluptate.",
7       "description": "Nihil ut ea dolorem iure voluptatem explicabo culpa. Qui minima nam tempore commodi voluptas eaque cumque. Accusamus qui iste consequuntur totam blanditiis dolore.",
8     },
9     {
10      "self": {
11        "href": "http://apiproservice.test/api/services/11"
12      }
13    },
14  ],
15  {
16    "id": 64,
17    "evaluation": null,
18    "title": "Illo quidem quae possimus maxime dolore.",
19    "description": "Reiciendis est non natus et excepturi dolor. Maiores et ipsam aut rerum voluptas corporis.",
20  },
21  {
22    "self": {
23      "href": "http://apiproservice.test/api/services/64"
24    }
25  },
26  {
27    "id": 116,
28    "evaluation": "5.0000",
29    "title": "Voluptatum perferendis laudantium et cupiditate perferendis dolorem illo laboriosam.",
30    "description": "Ut sit quia corporis minima. Fugit dolorem eaque accusantium ipsam accusantium. Rerum eligendi nisi unde voluptatem dolor illo exercitationem .",
31  },
32  {
33    "self": {
34      "href": "http://apiproservice.test/api/services/116"
35    }
36  }
37  ]
38  }

```

Fonte: POSTMAN

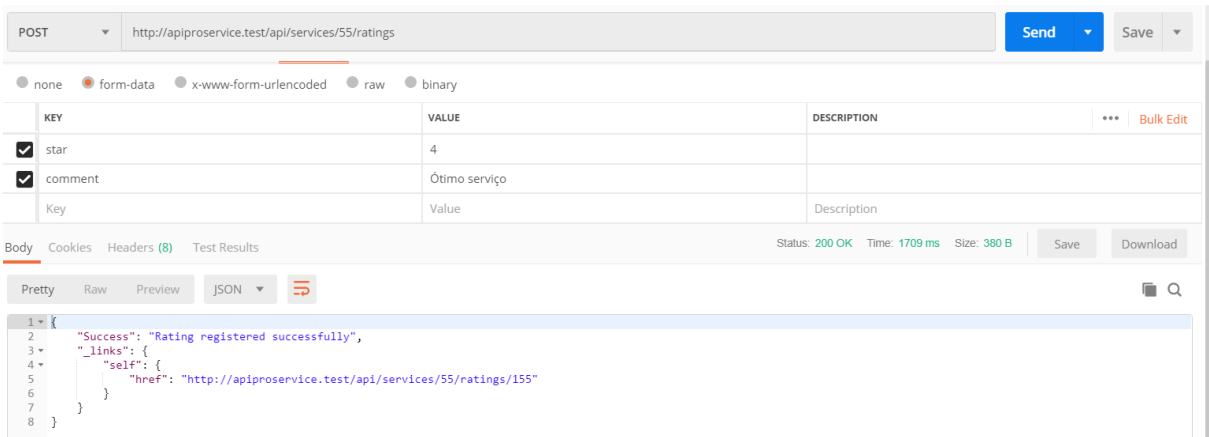
No recurso dos serviços também há a possibilidade de efetuar a exclusão de um serviço, atualização do título e a descrição. No recurso das avaliações foi implementada as rotas que estão descritas na Tabela 4. Para avaliar um serviço é preciso que o mesmo esteja contratado, é realizada através de cliques nas estrelas sendo enumeradas de 1 a 5, sendo 1 considerada como péssimo e 5 como um ótimo serviço.

Tabela 4: Rota das avaliações

URI	MÉTODO	DESCRIÇÃO
api/services/{id}/ratings	POST	Gera uma avaliação relacionado a um serviço
api/services/{id}/ratings	GET	Retorna as avaliações de um serviço
api/services/{id}/ratings/{id}	GET	Retorna uma avaliação pelo ID
api/services/{id}/ratings/{id}	PUT	Atualiza dados de uma avaliação
api/services/{id}/ratings/{id}	DELETE	Deleta uma avaliação

Fonte: Autoria própria

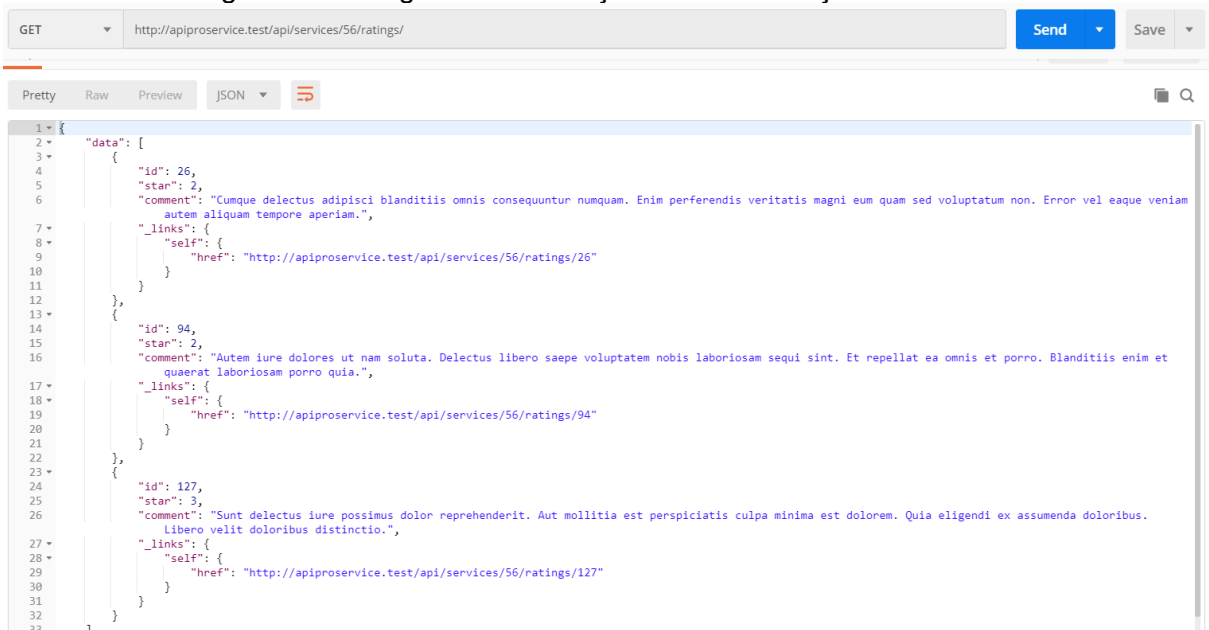
Figura 15: Avaliação de um serviço



Fonte: POSTMAN

A Figura 15 exibe o comportamento ao realizar uma avaliação de um serviço. Além desta funcionalidade o usuário é capaz de visualizar as avaliações buscando pelo ID e há a alternativa de retornar todas as avaliações relacionadas a um serviço em específico mostrando a média final do mesmo. A Figura 16 exibe a resposta da listagem das avaliações, também é possível atualizar uma avaliação, editando a nota das estrelas e os comentários.

Figura 16: Listagem das avaliações de um serviço



Fonte: POSTMAN

Na Tabela 5 exibe as rotas implementadas no recurso contrato incluindo como funcionalidades a realização de um contrato de um usuário relacionado a um serviço, contendo como campos obrigatórios o ID dos serviços adicionando uma descrição da

execução do serviço que o cliente está precisando. Na Figura 17 apresenta a resposta da API ao realizar uma contratação de um serviço.

Tabela 5: Rota dos contratos

URI	MÉTODO	DESCRIÇÃO
api/contracts	POST	Gera um contrato
api/contracts/{id}	GET	Retorna dados de um contrato pelo ID
api/contracts/made	GET	Retorna todos os contratos realizados pelo usuário autenticado
api/contracts/requested	GET	Retorna todos os contratos requisitados pelo usuário autenticado
api/contracts/{id}	PUT	Atualiza dados de um contrato
api/contracts/{id}	DELETE	Deleta um contrato

Fonte: Autoria própria

Figura 17: Contratação de um serviço

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://apiproservice.test/api/contracts
- Body:**

```

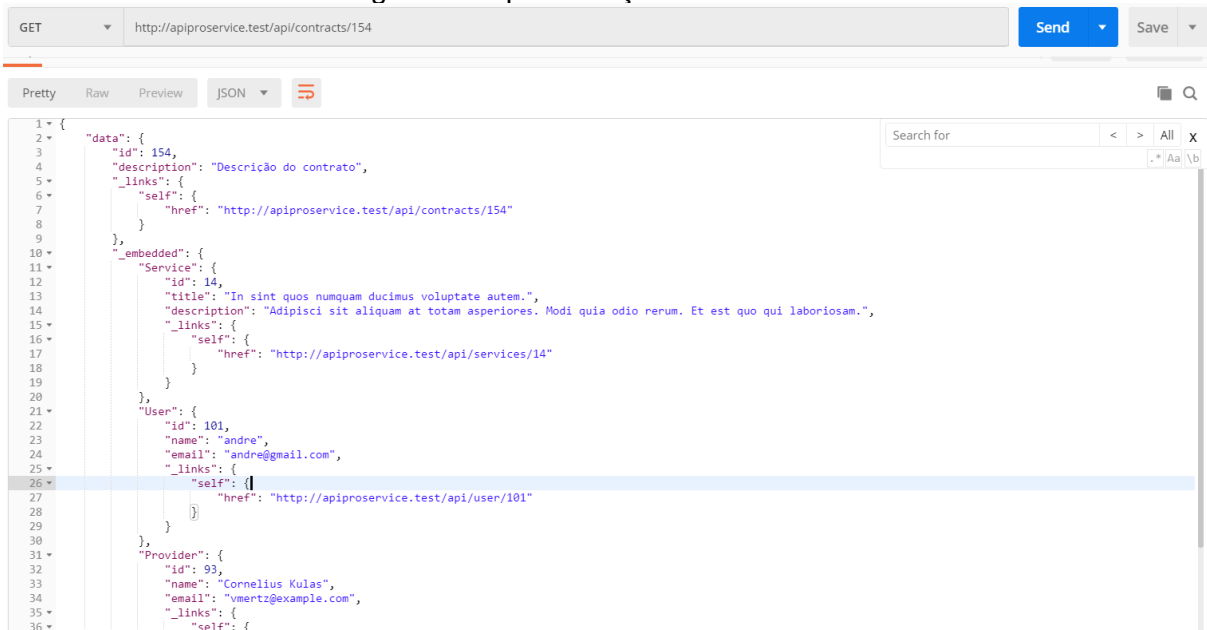
1 {
2   "Success": "Contrato registered successfully",
3   "links": {
4     "self": {
5       "href": "http://apiproservice.test/api/contracts/154"
6     }
7   }
8 }

```
- Status:** 200 OK
- Time:** 4581 ms
- Size:** 369 B

Fonte: POSTMAN

A fim de visualizar um contrato, é efetuada as requisições através do ID. Na Figura 18 apresenta a resposta da API na ferramenta POSTMAN. O usuário é capaz de verificar quais os contratos foram realizados pelo usuário autenticado, e na Figura 19 exhibe os contratos que foram solicitados.

Figura 18: Apresentação de um contrato



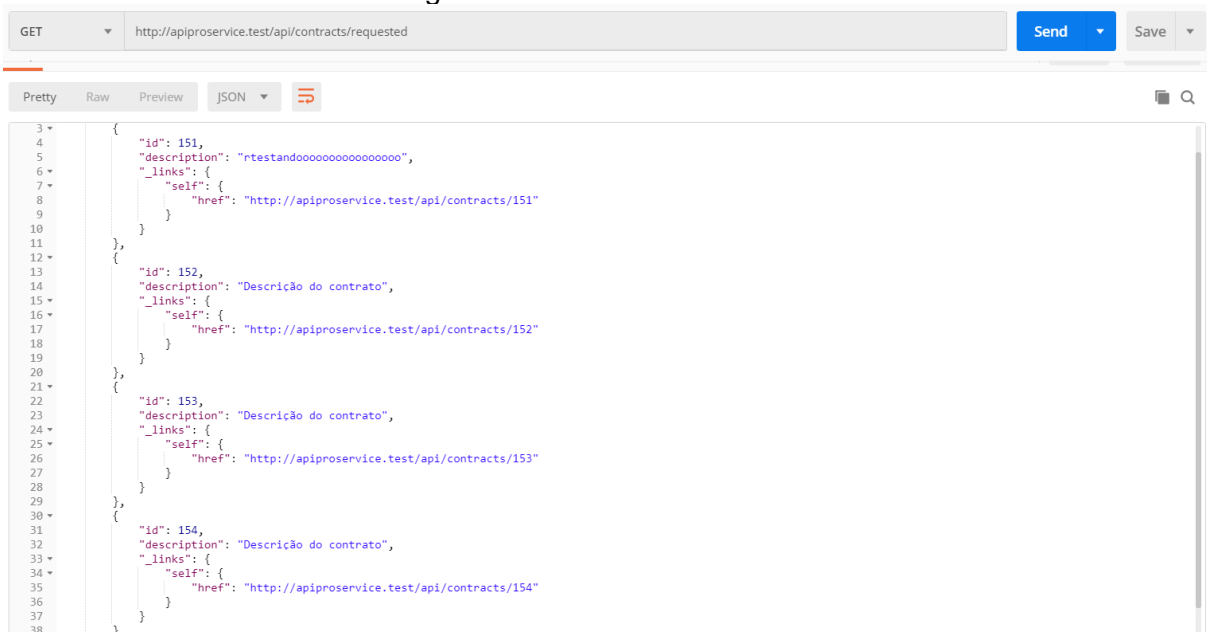
```

1 GET http://apiproservice.test/api/contracts/154
2
3 {
4   "data": {
5     "id": 154,
6     "description": "Descrição do contrato",
7     "_links": {
8       "self": {
9         "href": "http://apiproservice.test/api/contracts/154"
10      }
11    },
12    "_embedded": {
13      "Service": {
14        "id": 14,
15        "title": "In sint quos numquam ducimus voluptate autem.",
16        "description": "Adipisci sit aliquam at totam asperiores. Modi quia odio rerum. Et est quo qui laboriosam.",
17        "_links": {
18          "self": {
19            "href": "http://apiproservice.test/api/services/14"
20          }
21        }
22      },
23      "User": {
24        "id": 101,
25        "name": "andre",
26        "email": "andre@gmail.com",
27        "_links": {
28          "self": {
29            "href": "http://apiproservice.test/api/user/101"
30          }
31        }
32      },
33      "Provider": {
34        "id": 93,
35        "name": "Cornelius Kulas",
36        "email": "vmertz@example.com",
37        "_links": {
38          "self": {
39            "href": "http://apiproservice.test/api/providers/93"
40          }
41        }
42      }
43    }
44  }
45 }

```

Fonte: POSTMAN

Figura 19: Contratos Solicitados



```

1 GET http://apiproservice.test/api/contracts/requested
2
3 [
4   {
5     "id": 151,
6     "description": "rtestandoooooooooooooooooooo",
7     "_links": {
8       "self": {
9         "href": "http://apiproservice.test/api/contracts/151"
10      }
11    }
12  },
13  {
14    "id": 152,
15    "description": "Descrição do contrato",
16    "_links": {
17      "self": {
18        "href": "http://apiproservice.test/api/contracts/152"
19      }
20    }
21  },
22  {
23    "id": 153,
24    "description": "Descrição do contrato",
25    "_links": {
26      "self": {
27        "href": "http://apiproservice.test/api/contracts/153"
28      }
29    }
30  },
31  {
32    "id": 154,
33    "description": "Descrição do contrato",
34    "_links": {
35      "self": {
36        "href": "http://apiproservice.test/api/contracts/154"
37      }
38    }
39  }
40 ]

```

Fonte: POSTMAN

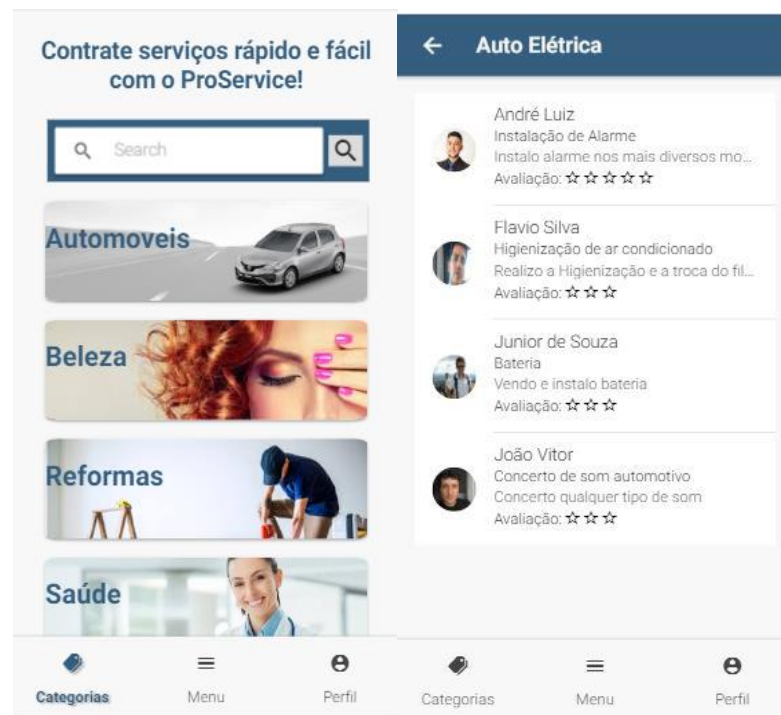
No recurso dos contratos também é possível realizar edições, porém é apenas na descrição de um contrato se for após a sua solicitação e realizar a exclusão através do ID.

Os testes mostraram resultados satisfatórios, mostrando a eficácia da API. Neste capítulo foram expostas as respostas das funcionalidades principais, no Apêndice A encontram-se as funções secundárias com maiores detalhes.

### 4.2.1 Validação com o ProService

O ProService é uma aplicação multiplataforma destinada a ofertas e demandas de serviços profissionais, onde o usuário poderá tanto contratar quanto oferecer um serviço. Conectando clientes e profissionais de forma simples e segura, por meio de buscas por serviços. Na Figura 20 mostra a tela inicial da aplicação composta pelas categorias dos serviços, redirecionando para os profissionais correspondente as categorias clicadas, exibindo as informações principais: nome, serviço ofertado, descrição do serviço e a avaliação.

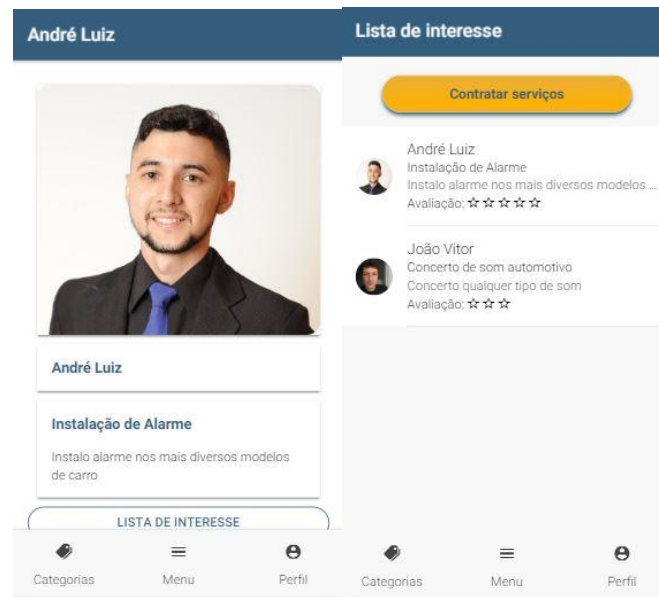
Figura 20: Buscando um serviço



Fonte: ProService

Na Figura 21 exibe as telas para a realização de uma contratação de serviços. O usuário é capaz de visualizar o perfil do profissional e ao clicar no botão adicionar na lista de interesse o serviço é adicionado em uma espécie de carrinho, sendo possível a contratação de mais de um serviço ao mesmo tempo.

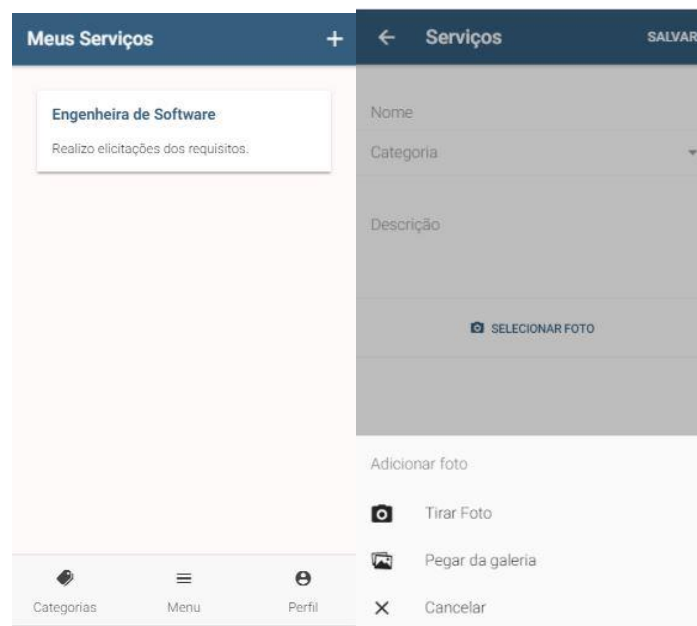
Figura 21: Contratando um serviço



Fonte: ProService

Na Figura 22 exibe as telas para a realização do cadastro de um serviço. Inicialmente o usuário poderá visualizar os que já estão cadastrados e clicando no ícone “+” é redirecionado a tela de cadastro onde é preciso inserir o nome do serviço, a descrição, selecionar a categoria correspondente e fotos para montar um pequeno portfólio.

Figura 22: Cadastro de serviços



Fonte: ProService

## 5 CONSIDERAÇÕES FINAIS

O setor de serviços abrange todas as atividades econômicas cujo produto não é um bem físico, ou seja, imaterial, contém uma participação importante na economia brasileira, o mesmo foi responsável por 75,8% do PIB em 2018. Diante o contexto exposto neste trabalho, o qual teve por objetivo geral contribuir na melhora dos processos de contratação e oferta de serviços através da implementação de uma API RESTful que torna mais simples e ágil o desenvolvimento de sistemas para esta finalidade.

Para a concretização desta monografia, primeiramente foi realizado um estudo sobre a arquitetura REST e o método de desenvolvimento da API com os pacotes do *framework* para desenvolvimento web Laravel.

A proposta do desenvolvimento da API no contexto de oferta e demanda de serviços foi validada através da ferramenta POSTMAN. A qual auxiliou todo o desenvolvimento, onde foram obtidas todas as respostas esperadas para as funcionalidades e recursos implementados. Percebendo que a proposta atingiu as expectativas e eficiência planejada. Na seção seguinte será exposto os trabalhos futuros para a complementação deste trabalho.

A API tem um destaque comparada aos trabalhos relacionados expostos no capítulo 2, além de ter utilizado uma linguagem de programação diferente, atingiu o nível 3 de implementação da maturidade de Richardson conquistando a glória do REST, ou seja, esta API é RESTful. Propõe ainda um diferencial para as aplicações, uma maneira diferente de realizar a contratação de serviços, buscas com a inclusão de filtros e a oportunidade de avaliar os serviços, além de ser capaz de ser usada para outros contextos.

### 5.1 Trabalhos Futuros

Como trabalhos futuros, pretende-se implementar *end-points* que permitam que o usuário solicite orçamento dos serviços que pretende contratar, assim como o provedor do serviço possa aceitar ou não prestar aquele serviço. Além disso, implementar o agendamento, onde o provedor possa inserir seus dias e horários de funcionamento, e a partir disto o usuário contrate e marque seu horário.



O provedor do serviço poderá avaliar o cliente, para que outros provedores possam analisar o cliente antes de aceitar o serviço com segurança. Através dessas avaliações, além da avaliação do usuário em relação ao provedor, será possível identificar usuários mal-intencionados, sendo possível denunciar o perfil para que o mesmo seja banido do sistema.

Faz-se necessário a inclusão do módulo de pagamento, para ser possível a monetização do sistema. Por fim, é interessante ser implementado a geolocalização dos serviços prestados, para que possa ser possível que os usuários localizem serviços mais próximos. Além de automatização dos testes.

## REFERÊNCIAS

- Aihara, Diogo Satoru (2009) “**Study About the Relationship Between the Model-ViewController Pattern and Usability**”. Disponível em: < <http://bit.ly/1lxZHpv>>. Acesso em: 20/03/2019. p. 14. RFC,6749.
- AIRBNB. Disponível em: <<https://www.airbnb.com/>>. Acesso em 18/04/2019
- ALLAMARAJU, Subbu. **RESTful Web Services Cookbook**. Sebastopol, CA: O'Reilly. 2010.
- Bean, J. (2003). **XML for Data Architects: Designing for Reuse and Integration**. Disponível em: <[https://www.researchgate.net/publication/291859517\\_SOA\\_and\\_Web\\_Services\\_Interface\\_Design](https://www.researchgate.net/publication/291859517_SOA_and_Web_Services_Interface_Design)>. Acesso em: 13/04/2019
- BELK, R. W. Sharing. *Journal of Consumer Research*, v. 36, n. 5, p. 715-734, 2010.
- BOOTH, David et al. **Web Services Architecture**. fev. 2004. Disponível em: <<http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>>. Acesso em: 20/04/2019.
- BOTSMAN, Rachel; ROGERS, Roo. **O que é seu é meu - como o consumo colaborativo vai mudar o nosso mundo**. Porto Alegre, Bookman Editora, 2011.
- CASETI, R. **Análises e avaliações da viabilidade de terceirização em alguns segmentos produtivos**. 2013
- Ciganek, A. P.; Haines, M. N.; Haseman, W. (2005) **Challenges of Adopting Web Services: Experiences from the Financial Industry**. In: *Proceedings of the 38th Hawaii International Conference on System Sciences*, p. 1-10
- COSTA, B. e. a. **Evaluating a representational state transfer (rest) architecture: What is the impact of rest in my architecture in: leee. software architecture (wicsa)**. p. 105–114, 2014
- Crawford, C. H.; Bate, G. P.; Cherbakov, L.; Holley, K.; Tsocanos, (2005) C. **Toward an on demand service-oriented architecture**. *IBM Systems Journal*, v. 44, n. 1, p. 81-107
- FIELDING, R. T.; TAYLOR, R. N. **Principled design of the modern web architecture**. *ACM Trans. Internet Technol.*, ACM, New York, NY, USA, v. 2, n. 2, p. 115–150, maio 2002. ISSN 1533-5399. Disponível em: <<http://doi.acm.org/10.1145/514183.514185>>. Acesso em: 05/04/2019.
- Fielding, R., Thomas. (2000) “**Representational State Transfer (REST)**”. Disponível em: <[https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.html](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.html)>. Acesso em: 04/04/2019.

FITZSIMMONS, J.; FITZSIMMONS, M. **Administração de serviços: operações, estratégia e tecnologia de informação**. 2. ed. Porto Alegre: Bookman, 2010

FREDRICH, Todd. **REST API Tutorial**. Disponível em: <<http://www.restapitutorial.com/lessons/whatisrest.html>> Acesso em: 06/03/2019

HURWITZ, J. Et al. **Service Oriented Architecture For Dummies**. Indianapolis: Willey, 2007

IBGE. Instituto brasileiro de Geografia e estatística. 2018. Disponível em: <<https://ww2.ibge.gov.br/home/>> Acesso em: 02/03/2019

IBGE. **Pesquisa Anual de Serviços - PAS**. 2016. Disponível em: <<https://www.ibge.gov.br/estatisticas/economicas/servicos/9028-pesquisa-anual-de-servicos.html?=&t=o-que-e>> acesso em 10/04/2019.

IG. **Serviços têm melhor resultado em 4 anos e geram mais de 20 mil empregos em sp**. Disponível em: <<https://economia.ig.com.br/2018-10-18/crescimento-no-setor-de-servicos-sp.html>>. Acesso em: 08/04/2019.

Kotler, P. and Keller, K.L. (2012) **Marketing Management**. 14th Edition, Pearson Education.

Kumar, S.; Dakshinamoorthy, V.; Krishnan, M. S. (2007) **Does SOA Improve the Supply Chain? An Empirical Analysis of the Impact of SOA Adoption on Electronic Supply Chain Performance**. Proceedings of the 40th Hawaii International Conference on System Sciences, p. 1-10

Len Bass, Paul Clements, and Rick Kazman. **Software Architecture in Practice**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998., 2º Edição, Abril 2003.

LINNUZ. Disponível em: <<https://linnuz.com.br>> acesso em 18/04/2019.

LOBO.M.S.C. **Aplicação da análise envoltória de dados (DEA) para apoio às políticas públicas de saúde: o caso dos hospitais de ensino**. 2010. Tese (Doutorado em Engenharia de Produção)- Universidade Federal do Rio de Janeiro, Rio de Janeiro/RJ,2010.

Marks, E. A.; Bell, M. (2006) **Service-oriented architecture: a planning and implementation guide for business and technology**. New Jersey: John Wiley e Sons, Inc,

Mezo, B. M.; Chaparro, T. S.; Heras, A. D. (2008) **Características de las empresas que utilizan Arquitectura Orientada a Servicios y de su contexto de operación**. Journal of Information Systems and Technology Management, v. 5, n. 2, p. 269-304

- Mezo, B. M.; Chaparro, T. S.; Heras, A. D. (2008) **Características de las empresas que utilizan Arquitectura Orientada a Servicios y de su contexto de operación.** Journal of Information Systems and Technology Management, v. 5, n. 2, p. 269-304
- MILANI, André. **MySQL: Guia do Programador.** São Paulo. Novatec Editora, 2006.
- Newcomer, E. and Lomow, G. (2004) **Understanding SOA with Web Services** (Independent Technology Guides). Addison-Wesley Professional, Boston.
- Newcomer, E., and Lomow, G. 2005. **Understanding Soa with Web Services.** Amsterdam: AddisonWesley Longman.
- OLIVEIRA, G.G.D.O. **Construção aplicações distribuídas utilizando-se api rest.** 2018.
- Puschmann, T.; ALT, R. (2001) **Enterprise Application Integration - The Case of the Robert Bosch Group. In: Proceedings of the 34th Hawaii International Conference on System Sciences,** p. 1-10
- RICHARDSON, L.; AMUNDSEN, M.; RUBY, S. **Restful Web Apis. Oreilly & Associates Incorporated,** 2013. ISBN 9781449358068. Disponível em: <<http://books.google.com.br/books?id=i3a7mAEACAAJ>>. Acesso em: 04/04/2019.
- SCHOR, J. B. **Debating the sharing economy. Journal of Self-Governance and Management Economics,** v. 4, n. 3, p. 7-22, 2016. Disponível em: Acesso em: 25/03/2019.
- Serman, D. V. (2007) **Estratégias de TI para a integração eletrônica da informação: um estudo sobre o estado da arte e da prática.** 2007. 119 p. Dissertação (Mestrado em Administração) –Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro
- SILVA, C; FILHO, N; KOMATSU, B. **Uma Abordagem sobre o Setor de Serviços na Economia Brasileira.** N 19. 2016.
- SILVA, M. C. d. **Estação iot para monitoramento da temperatura e umidade do interior de veículos.** 2018. Disponível em: <<https://goo.gl/XuJyMT>>. Acesso em: 09/04/2019.
- SOUZA, C. C. et al. **Risk classification in an emergency room: agreement level between a Brazilian institutional and the Manchester Protocol.** Revista Latinoamericana de Enfermagem, Ribeirão Preto, v. 19, n. 1, p. 26-33, 2011. Disponível em: <[http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S0104-11692011000100005](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-11692011000100005)>. Acesso em: 20/04/2019
- Stal, M. (20062) **Web Services: Beyond Component-Based Computing.** Communications of the ACM, v. 45, n. 10, p. 71-78

TORRES, Simone Pádua et al. **Marketing de Relacionamento: A Satisfação e Fidelização do Cliente.** Disponível em: <[http://www.iptan.edu.br/publicacoes/anuario\\_producao\\_cientifica/arquivos/revista1/artigos/Artigo\\_Simone\\_Sonia.pdf](http://www.iptan.edu.br/publicacoes/anuario_producao_cientifica/arquivos/revista1/artigos/Artigo_Simone_Sonia.pdf)>. Acesso em: 05/03/2019.

UBER. Disponível em: <<https://www.uber.com/br/pt-br/>> acesso em 19/04/2019.

Vigo, Leandro e Oliveira, V., Eliane. (2014) “**WEB SERVICES: INTEGRAÇÃO E PADRONIZAÇÃO DE SERVIÇOS**”. p. 37.

WATSON, W. **REST Constraints Part 5.** Disponível em: <<http://wwatson.me/2011/10/13/rest-constraints-part-5/>> Acesso em: 19/04/2019

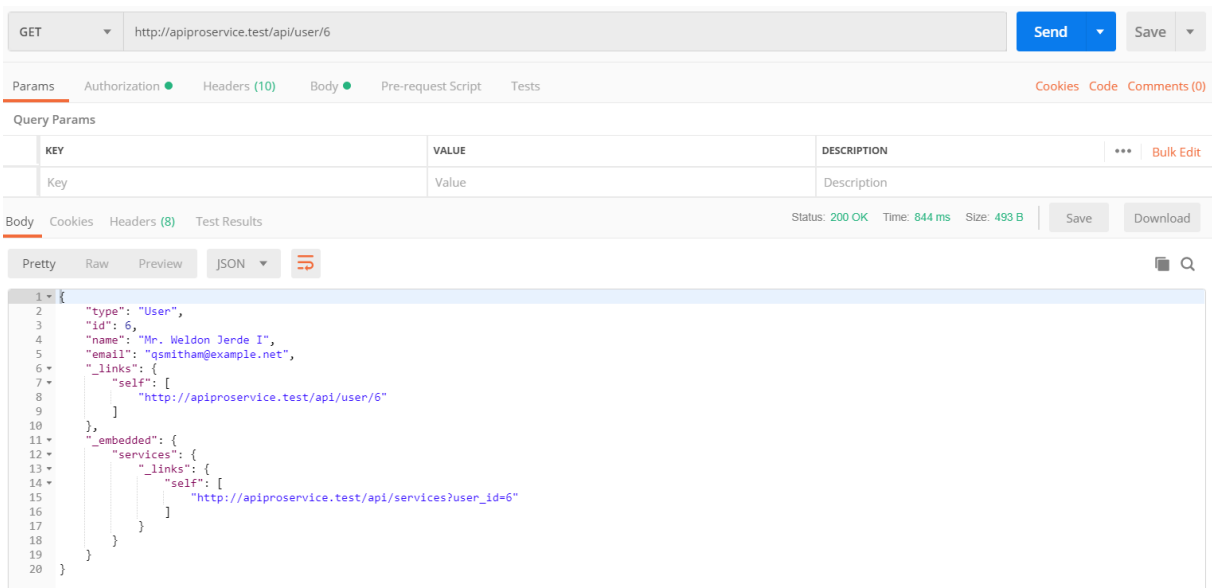
XOFER. Disponível em: <<http://meuapp.mobi/xofer/>>. Acesso em 19/04/2019

ZIPCAR. Disponível em: <<https://www.zipcar.com/>>. Acesso em 19/04/2019

99 TAXIS. Disponível em: <<https://99app.com>> Acesso em 19/04/2019



Figura 3: Mostrar usuário buscando pelo ID



GET http://apiproservice.test/api/user/6

Send Save

Params Authorization Headers (10) Body Pre-request Script Tests Cookies Code Comments (0)

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

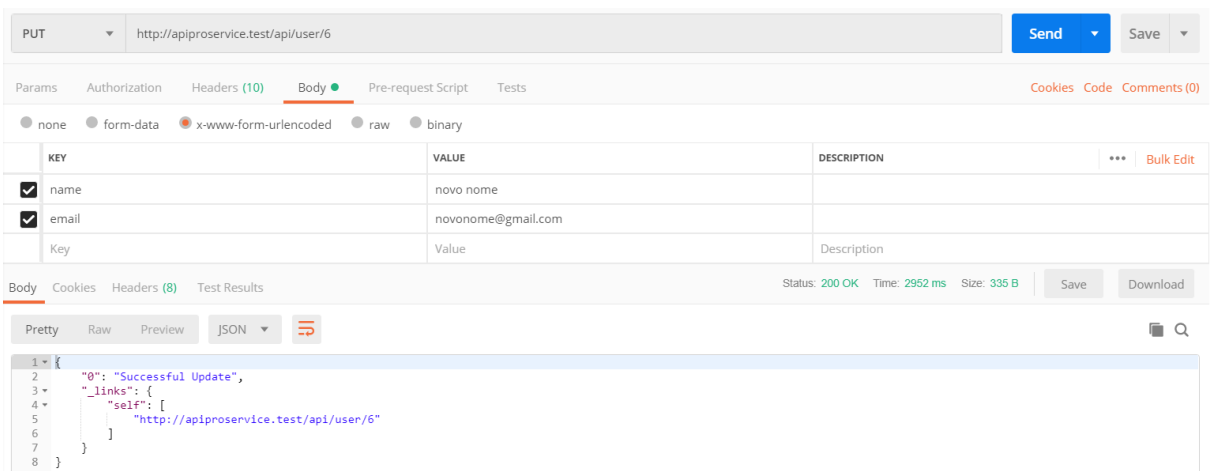
Body Cookies Headers (8) Test Results Status: 200 OK Time: 844 ms Size: 493 B Save Download

Pretty Raw Preview JSON

```
1 {
2   "type": "User",
3   "id": 6,
4   "name": "Mr. Weldon Jerde I",
5   "email": "qsmitham@example.net",
6   "_links": {
7     "self": [
8       "http://apiproservice.test/api/user/6"
9     ]
10  },
11  "_embedded": {
12    "services": {
13      "_links": {
14        "self": [
15          "http://apiproservice.test/api/services?user_id=6"
16        ]
17      }
18    }
19  }
20 }
```

Fonte: POSTMAN

Figura 4: Atualiza dados de um usuário



PUT http://apiproservice.test/api/user/6

Send Save

Params Authorization Headers (10) Body Pre-request Script Tests Cookies Code Comments (0)

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	novo nome	
<input checked="" type="checkbox"/> email	novonome@gmail.com	
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 2952 ms Size: 335 B Save Download

Pretty Raw Preview JSON

```
1 {
2   "0": "Successful Update",
3   "_links": {
4     "self": [
5       "http://apiproservice.test/api/user/6"
6     ]
7   }
8 }
```

Fonte: POSTMAN

Figura 5: Exclui um usuário buscando pelo ID

DELETE http://apiproservice.test/api/user/2

Send Save

Params Authorization Headers (10) Body Pre-request Script Tests Cookies Code Comments (0)

Headers (2)

KEY	VALUE	DESCRIPTION
Accept	application/json	
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6...	
Key	Value	Description

Temporary Headers (8)

Body Cookies Headers (8) Test Results Status: 200 OK Time: 1257 ms Size: 269 B Save Download

Pretty Raw Preview JSON

```

1- [
2-   "Successful deletion"
3- ]

```

Fonte: POSTMAN

Figura 6: Listagem de todos os serviços do sistema

GET http://apiproservice.test/api/services

Send Save

Pretty Raw Preview JSON

```

1- {
2-   "data": [
3-     {
4-       "id": 1,
5-       "evaluation": "3.3333",
6-       "title": "Debitis sunt iste deleniti voluptas quia.",
7-       "description": "Iste earum officia magnam. Corporis ex repudiandae minus quas a consequatur sint. Perspiciatis magni quis et eligendi voluptatem et deserunt eum. Velit amet repellendus sint ut quibusdam quos vel.",
8-       "_links": {
9-         "self": {
10-            "href": "http://apiproservice.test/api/services/1"
11-          }
12-        }
13-      },
14-     {
15-       "id": 2,
16-       "evaluation": null,
17-       "title": "Est hic et repudiandae voluptas.",
18-       "description": "Qui eaque ullam commodi fugiat id. Nisi perspiciatis quas veniam rerum culpa maiores consequatur. Minus eos dolores reiciendis cum.",
19-       "_links": {
20-         "self": {
21-            "href": "http://apiproservice.test/api/services/2"
22-          }
23-        }
24-      },
25-     {
26-       "id": 4,
27-       "evaluation": "1.0000",
28-       "title": "Provident doloribus vero eligendi.",
29-       "description": "Est quo voluptatem hic consequatur distinctio eligendi. Et ut dignissimos sapiente qui quia. Inventore id voluptate ut laudantium ipsam iusto .",
30-       "_links": {
31-         "self": {
32-            "href": "http://apiproservice.test/api/services/4"
33-          }
34-        }
35-      }
36-     ]
37-   }

```

Fonte: POSTMAN



Figura 7: Excluir um serviço

DELETE http://apiproservice.test/api/services/3

Params Authorization Headers (9) Body Pre-request Script Tests Cookies Code Comments (0)

▼ Headers (2)

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Accept	application/json	
<input checked="" type="checkbox"/> Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6IjkyM2Qz...	
Key	Value	Description

▶ Temporary Headers (7)

Body Cookies Headers (8) Test Results Status: 200 OK Time: 1365 ms Size: 269 B Save Download

Pretty Raw Preview JSON

```

1 [
2   "Successful deletion"
3 ]

```

Fonte: POSTMAN

Figura 8: Editar dados de um serviço

PUT http://apiproservice.test/api/services/77

Params Authorization Headers (10) Body Pre-request Script Tests Cookies Code Comments (0)

● none ● form-data ● x-www-form-urlencoded ● raw ● binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> title	Editando Título	
<input checked="" type="checkbox"/> description	Descrição Editada	
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 1631 ms Size: 363 B Save Download

Pretty Raw Preview JSON

```

1 {
2   "Success": "Service Update successfully",
3   "_links": {
4     "self": {
5       "href": "http://apiproservice.test/api/services/77"
6     }
7   }
8 }

```

Fonte: POSTMAN

Figura 9: Mostrar dados de uma avaliação

```

5   "star": 2,
6   "comment": "Magnam culpa at et facilis rem. Temporibus esse eos distinctio consequatur. Culpa fuga non quia vel.",
7   "_links": {
8     "self": {
9       "href": "http://apiproservice.test/api/services/98/ratings/54"
10    }
11  },
12  "_embedded": {
13    "service": {
14      "_links": {
15        "self": {
16          "href": "http://apiproservice.test/api/services/98"
17        }
18      }
19    },
20    "Service Provider": {
21      "name": "Juston Mullen",
22      "email": "lempl10@example.net",
23      "_links": {
24        "self": {
25          "href": "http://apiproservice.test/api/user/89"
26        }
27      }
28    },
29    "user": {
30      "name": "Nikita Stamm",
31      "email": "wolf.lily@example.org",
32      "_links": {
33        "self": {
34          "href": "http://apiproservice.test/api/user/92"
35        }
36      }
37    }
38  }
39 }
40 }

```

Fonte: POSTMAN

Figura 10: Atualizar dados de uma avaliação

PUT http://apiproservice.test/api/services/54/ratings/154

Params Authorization Headers (10) Body Pre-request Script Tests Cookies Code Comments (0)

none form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> star	5	
<input checked="" type="checkbox"/> comment	Editanco Comentário	
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 1787 ms Size: 361 B Save Download

```

1  {
2    "0": "Successful Update",
3    "_links": {
4      "self": {
5        "href": "http://apiproservice.test/api/services/55/ratings/154"
6      }
7    }
8  }

```

Fonte: POSTMAN

Figura 11: Exclui uma avaliação do sistema

DELETE http://apiproservice.test/api/services/54/ratings/154

Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Cookies Code Comments (0)

Headers (2)

KEY	VALUE	DESCRIPTION
Accept	application/json	
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImpp0aSI6jkyM2Qz...	
Key	Value	Description

Temporary Headers (7)

Body Cookies Headers (8) Test Results Status: 200 OK Time: 790 ms Size: 269 B Save Download

Pretty Raw Preview JSON

```
1 {
2   "data": [
3     "Successful deletion"
4   ]
5 }
```

Fonte: POSTMAN

Figura 12: Mostra todos os contratos realizados pelo usuário autenticado

GET http://apiproservice.test/api/contracts/made

Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Cookies Code Comments (0)

Headers (2)

Temporary Headers (6)

Body Cookies Headers (8) Test Results Status: 200 OK Time: 1582 ms Size: 393 B Save Download

Pretty Raw Preview JSON

```
1 {
2   "data": [
3     {
4       "id": 155,
5       "description": "Descrição do contrato",
6       "_links": {
7         "self": {
8           "href": "http://apiproservice.test/api/contracts/155"
9         }
10      }
11     }
12   ]
13 }
```

Fonte: POSTMAN

Figura 13: Editar dados de um contrato

The screenshot shows a Postman interface for a PUT request to `http://apiproservice.test/api/contracts/18`. The request body is a JSON object:

```

1 {
2   "Success": "Contract Update successfully",
3   "links": {
4     "self": {
5       "href": "http://apiproservice.test/api/contracts/18"
6     }
7   }
8 }

```

The response status is 200 OK, with a time of 2168 ms and a size of 364 B.

Fonte: POSTMAN

Figura 14: Excluir um contrato

The screenshot shows a Postman interface for a DELETE request to `http://apiproservice.test/api/contracts/7`. The request headers are:

KEY	VALUE	DESCRIPTION
Accept	application/json	
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsIm00aSI6IjkyMzQz...	
Content-Type	application/x-www-form-urlencoded	

The response status is 200 OK, with a time of 1541 ms and a size of 269 B. The response body is a JSON array:

```

1 [
2   "Successful deletion"
3 ]

```

Fonte: POSTMAN