



# **Desenvolvimento de aplicação para comunicação com o *Openstack***

Trabalho Laboratorial nº 1 (TL1)  
Laboratório de Tecnologias de Informação  
LEI 2020/21

Grupo 5

Filipe Alexandre Coelho Almeida, 2171302

Gonçalo Miguel Conceição Vicente, 2172131

Leiria, abril de 2021



# Resumo

O presente documento insere-se no âmbito da disciplina de Laboratório de Tecnologias de Informação (LTI), pertencente à Licenciatura em Engenharia Informática e descreve a implementação de uma aplicação Web para gerir os recursos existentes através de uma plataforma *Openstack*.

De forma a implementar este projeto, utilizamos a *framework Laravel* juntamente com a *framework Vue.js*, para construir a aplicação Web. Através da *framework Vue.js* é possível comunicar com o servidor Linux, onde se encontra instalado o *Openstack*, através de pedidos *RESTFull* (*GET, POST, PUT, DELETE*).

O trabalho laboratorial descrito neste documento, encontra-se dividido em duas partes distintas. A primeira parte, está associada a serviços existentes ao *Core* da plataforma *Openstack*. A segunda parte, está associado a um serviço que é necessário instalar e vai ser abordado a questão dos *Containers*.

**Palavras-chave:** *Openstack, RESTfull, Laravel, Vue.js, Virtualização*

# Abstract

The following document is part of the discipline of Information Technology Laboratory (LTI), belonging to the Degree in Computer Engineering and explains the implementation of a Web application to manage existing resources through an Openstack platform.

In order to implement this project, we used the Laravel framework simultaneously with the framework Vue.Js, to build the Web application. Through the Vue.Js framework it is possible to communicate with the Linux server, where Openstack is running, through RESTFull requests. (GET, POST, PUT, DELETE).

The laboratory work described in this document is divided into two distinct parts. The first one is associated with existing services to the Core of the OpenStack platform. The second part is associated with a service that needs to be installed and where the Containers will be referred.

**Keywords:** *Openstack, RESTfull, Laravel, Vue.Js, Virtualization*

# Índice

<b>Resumo.....</b>	<b>iii</b>
<b>Abstract.....</b>	<b>iv</b>
<b>Lista de Figuras.....</b>	<b>vi</b>
<b>Lista de Tabelas.....</b>	<b>vii</b>
<b>1. Introdução .....</b>	<b>1</b>
<b>2. Trabalho Desenvolvido.....</b>	<b>2</b>
<b>2.1. Parte I .....</b>	<b>2</b>
<b>2.2. Parte II.....</b>	<b>6</b>
<b>3. Análise crítica e proposta de melhorias.....</b>	<b>8</b>
<b>4. Conclusão .....</b>	<b>9</b>
<b>Bibliografia .....</b>	<b>10</b>

# Lista de Figuras

Figura 1 - Componente definirIP.vue.....	3
Figura 2 - Componente login.vue.....	3
Figura 3 - Componente instancias.vue .....	4
Figura 4 - Componente imagens.vue.....	5
Figura 5 - Componente volumes.vue .....	5
Figura 6 - Componente dashboard.vue.....	6

# Lista de Tabelas

Tabela 1 - Lista de todos os métodos/URI/descrição .....	7
--	---





# 1. Introdução

O presente documento insere-se na licenciatura em Engenharia Informática, mais concretamente na unidade curricular Laboratório de Tecnologias de Informação decorrente no ano letivo 2020/2021.

Neste documento, é descrito como foi implementado o trabalho laboratorial 1 bem como as funcionalidades existentes.

O trabalho laboratorial encontra-se dividido em duas partes distintas. Na primeira parte, os objetivos a serem cumpridos eram: instalação da plataforma *Openstack*, criação de uma aplicação que comunicasse através das APIs e que permitisse gerir os recursos disponibilizados pela plataforma, nomeadamente: identificar qual a plataforma *Openstack* a utilizar, acesso via user/pwd, visualização de instâncias criadas por projeto, mudar de projeto, criar/alterar/eliminar VMs, criar volumes de armazenamento e adição de imagens de sistemas operativos. Na segunda parte, o objetivo era utilizar um serviço que não fosse utilizado na primeira parte bem como implementar algumas das funcionalidades que esse serviço permite.

Nos dias de hoje, a utilização de virtualização tem vindo a crescer cada vez mais, porém a plataforma *Openstack* não é apenas uma plataforma de gerenciamento de virtualização, embora sejam soluções parecidas. Tanto o *Openstack* como as plataformas de virtualização utilizam recursos virtualizados e possuem capacidade para detetar, informar e automatizar processos em ambientes de fornecedores diferentes. No entanto, enquanto as plataformas de virtualização facilitam a forma como são manipulados os recursos virtuais, o *Openstack*, usa os recursos virtuais para executar uma combinação de diversas ferramentas, criando um ambiente de *Cloud*.

## 2. Trabalho Desenvolvido

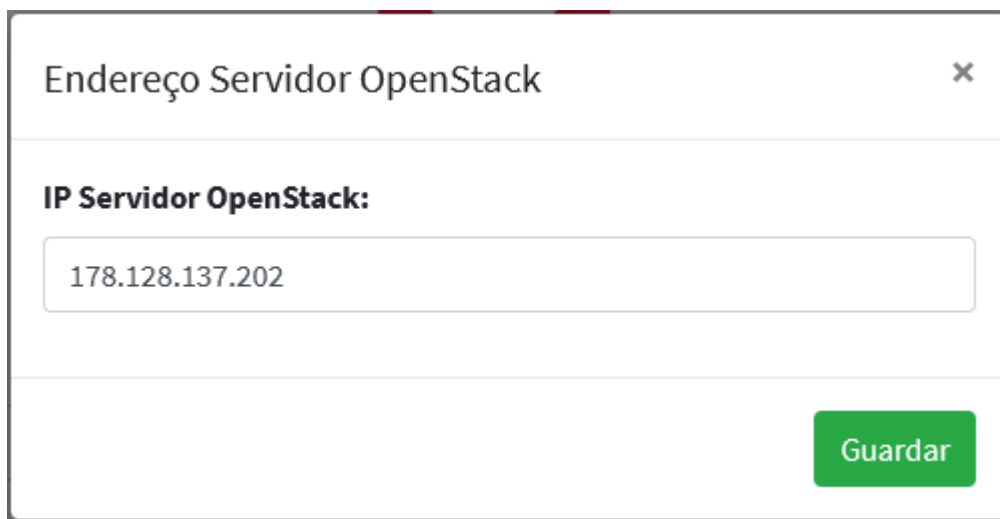
De forma a interligar a plataforma *Openstack*, com o nosso projeto, implementamos uma aplicação Web, utilizando as frameworks *Laravel* e *Vue.js* para assegurar a comunicação com as APIs de cada serviço através de pedidos *RESTFull*.

Durante a realização deste trabalho laboratorial, deparamo-nos com alguns problemas relativamente ao cabeçalho CORS. Resolvemos esses problemas com o uso do *browser Firefox* e com a instalação de duas extensões: *CORS Everywhere* e *Cross Domain – CORS*.

### 2.1. Parte I

Após se encontrar instalado o servidor Linux com a plataforma *Openstack* instalada e o projeto *Laravel* criado é necessário definir, programaticamente o endereço IP do servidor Openstack que se pretende utilizar.

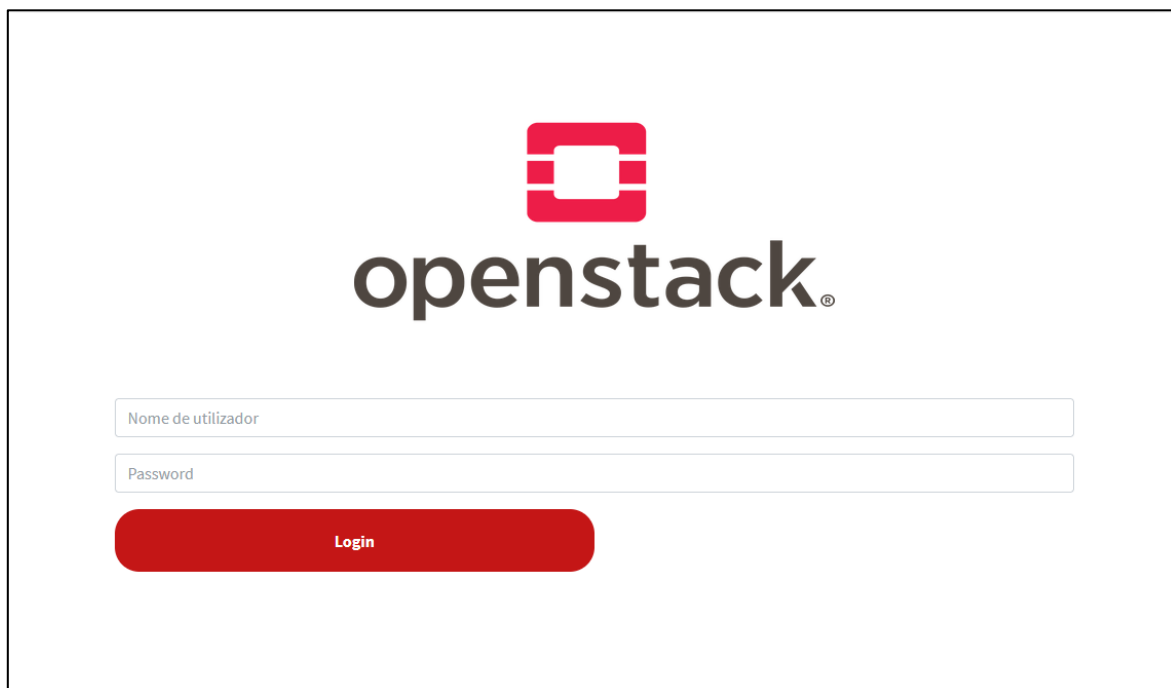
Foi criado um componente, o *definirIP.vue*(Figura 1), onde se recorre um componente *Bootstrap Modal* para pedir ao utilizador que informe qual o endereço IP a que se pretende ligar. Depois de submetido, era necessário submeter essa informação de forma que seja possível utilizar em outros componentes. Assim, instalou-se a aplicação *Vuex* que permite armazenar na *storage* local do *browser*, determinadas informações. Deste modo, assim que é submetido o endereço IP do servidor Openstack que se pretende utilizar, essa informação é colocada na *storage* com o auxílio da função *setUrl()*, ficando, deste modo acessível em toda a restante aplicação Web. Através da função *setIP()*, é guardado na *storage* local o endereço IP utilizado, para que na próxima tentativa de *login* esse endereço IP aparecer definido de forma a evitar que o utilizar tenha que estar a alterar.



The image shows a modal window titled "Endereço Servidor OpenStack" with a close button (X) in the top right corner. Inside the modal, there is a label "IP Servidor OpenStack:" followed by a text input field containing the IP address "178.128.137.202". At the bottom right of the modal, there is a green button labeled "Guardar".

Figura 1 - Componente definirIP.vue

Estando decidido qual o servidor que deveria ser utilizado, é necessário fazer login. Assim, no componente login.vue(Figura 2), é solicitado ao utilizador as suas credencias, neste caso *username* e *password*. Obtida a informação, é necessário guardar um *Token* para este ser utilizado enquanto o utilizador estiver autenticado, e para poder ser utilizado em toda a aplicação web recorreu-se ao *Vuex*. Assim, através das funções *setPassword()*, *setToken*, *setProject*, *setUser()*, são guardas informações importantes para utilizar ao logo da sessão.

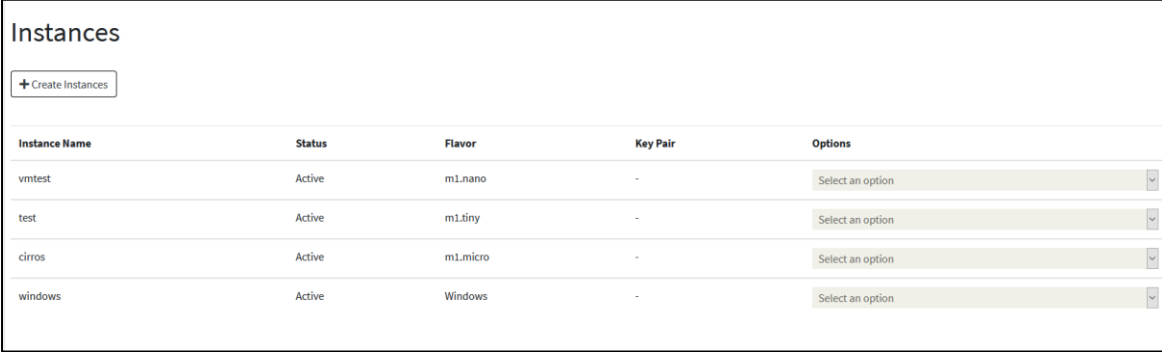


The image shows the OpenStack login interface. At the top center is the OpenStack logo, which consists of a red square icon with a white 'O' inside, followed by the word "openstack" in a dark grey sans-serif font. Below the logo, there are two input fields: the first is labeled "Nome de utilizador" and the second is labeled "Password". Below these fields is a red button with the text "Login" in white.

Figura 2 - Componente login.vue

Quando o utilizador pretende fazer logout da aplicação Web, o *Token* é destruído com o auxilio da função *clearToken()*.

Tal como pedido no enunciado do trabalho, era necessário visualizar todas as instâncias criadas bem como criar, alterar e eliminar máquinas virtuais. Assim, foi implementado o componente `instancias.vue` (Figura 3). Neste componente, existe uma tabela onde se encontram todas as instâncias existentes, sendo apresentado o seu nome, estado, *flavor*, par de chaves e as opções complementares: editar o nome, apagar uma instância e abrir a consola utilizando o *noVNC*, porém esta opção só se encontra disponível para instâncias que estejam no estado ativo. De forma a ser possível criar uma instância utilizou-se o componente *Bootstrap Modal*, onde é apresentado ao utilizador para escolher um nome que pretende para a instância, seleccionar qual a zona de disponibilidade, seleccionar qual é a imagem a utilizar, escolher o *flavor*, bem como o tipo de rede a utilizar.



Instance Name	Status	Flavor	Key Pair	Options
vmtest	Active	m1.nano	-	Select an option
test	Active	m1.tiny	-	Select an option
cirros	Active	m1.micro	-	Select an option
windows	Active	Windows	-	Select an option

Figura 3 - Componente `instancias.vue`

Também era pretendido que existisse a possibilidade de adicionar imagens de sistemas operativos para utilizar nas VMs. Para tal, foi utilizado o componente `imagens.vue` (Figura 4), onde se encontra uma listagem de todas as imagens existentes, sendo apresentado o seu nome, estado, visibilidade, se é protegida, o formato do disco e o tamanho. Para que fosse possível criar uma nova imagem, recorreu-se ao componente *Bootstrap Modal*, onde o utilizador tem de escolher um nome para a imagem, o uri de onde se encontra a imagem, o formato da imagem, o disco mínimo (em GB) e a RAM mínima em (MB), bem como o tipo de visibilidade e proteção.

Images					
+ Create Image					
Name	Status	Visibility	Protected	Disk Format	Size
windows_2	Queued	Shared	True	QCOW2	NaN undefined
Cirros	Queued	Shared	True	ISO	NaN undefined
Cirros	Queued	Private	True	ISO	NaN undefined
Micro XP	Queued	Private	True	ISO	NaN undefined
ubuntu14	Active	Shared	False	ISO	912.26 MB
windows	Active	Shared	False	QCOW2	12 GB
cirros-0.5.2-x86_64-disk	Active	Public	False	QCOW2	16.3 MB

Figura 4 - Componente imagens.vue

Um outro objetivo deste trabalho laboratorial, era existir a possibilidade de criar volumes de armazenamentos. Assim, no componente volumes.vue(Figura 5), encontra-se a listagem de todos os volumes existentes, sendo possível visualizar o nome associado ao volume, o tamanho que este possui, caso existe alguma descrição também será apresentada, caso contrário é apresentado “-”, o estado do volume, tipo, a zona de disponibilidade em que se encontra, se é *bootable* e se se encontra encriptado. Para que exista a possibilidade de criar um novo volume, recorreu-se mais uma vez ao componente *Bootstrap Modal*, onde é pedido ao utilizador que defina o nome do volume, o tamanho (em GB) e escolher uma imagem das existentes.

Volume							
+ Create Volume							
Name	Size	Description	Status	Type	Availability Zone	Bootable	Encrypted
Volume_Test2	1 GiB	ola mundo	Available	lvmdriver-1	nova	No	No
Volume_Test1	1 GiB	-	Available	lvmdriver-1	nova	No	No
4387e87f-48e3-4381-a681-e51a19262616	1 GiB	-	In-use	lvmdriver-1	nova	Yes	No
fd561f36-4703-4f9b-b685-15b8ace20726	1 GiB	-	Available	lvmdriver-1	nova	Yes	No
86df6789-60a5-4899-a4d5-e664bda2b3e3	2 GiB	-	Available	lvmdriver-1	nova	Yes	No
9b08877a-29da-4ef1-a2bc-7458d5d24ae4	2 GiB	-	Available	lvmdriver-1	nova	Yes	No
4b00c0b5-0d34-4a12-a7eb-e1cf279b24d0	2 GiB	-	Available	lvmdriver-1	nova	Yes	No
45f46ff0-1111-407f-b1a2-d2d6880764a5	1 GiB	-	Available	lvmdriver-1	nova	Yes	No
windows	40 GiB	-	In-use	lvmdriver-1	nova	Yes	No

Figura 5 - Componente volumes.vue

Uma das tarefas a ser realizada era mudar o projeto com base nos projetos disponíveis. Desta forma criámos a componente `changeProject.vue`, que tem como função utilizar o nome do projeto que se quer seleccionar para fazer um pedido POST com scope do projeto seleccionado e com as credenciais guardadas no *vuex*. De seguida as variáveis no *vuex* do *token* e do projeto são substituídas por as que forem recebidas na resposta do pedido POST.

Foi também desenvolvido uma área onde são apresentadas estatísticas, nomeadamente as quantidades de instâncias existentes, a quantidade de vCPUs existentes, a memória RAM utilizada, bem como a quantidade de volumes existentes, a quantidade de *snapshots* bem como o armazenamento de volume. Para tal, foi criada o componente `dashboard.vue`, um componente que é apresentado logo após o *login* ser efetuado corretamente.

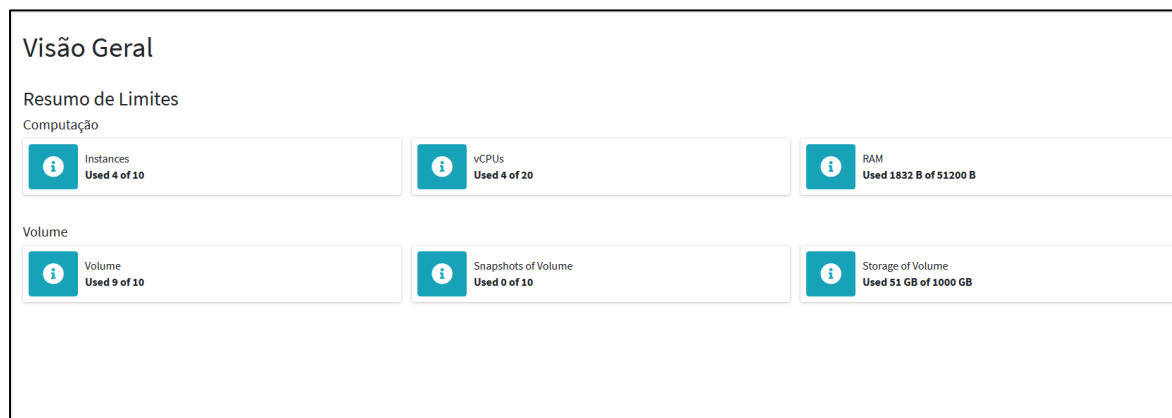


Figura 6 - Componente `dashboard.vue`

## 2.2. Parte II

A segunda parte deste trabalho laboratorial descreve a utilização de outros serviços da plataforma *Openstack*, devendo ficar disponíveis e funcionais algumas das funcionalidades que esses serviços possuem.

De forma a cumprir este requisito optamos por utilizar o *Zun*, que é um serviço *Openstack Container* e tem como objetivo fornecer um serviço de API para a execução de containers.

Para que seja possível demonstrar as funcionalidades deste serviço, resolvemos implementar, no componente `containers.vue`, a listagem de todos os containers existentes, sendo possível visualizar o nome, a imagem associada, o endereço IP que possui, o número de CPU's bem como a memória RAM. Recorrendo a um componente *Bootstrap Modal*, está também implementada, a opção de criar um novo container, sendo necessário preencher o

nome do *container*, qual a imagem que se pretende utilizar, a quantidade de CPU's, a memória RAM (em MB) e a zona de disponibilidade.

Na tabela seguinte, é possível visualizar um esquema de todos os métodos *RESTFull* utilizados e o respetivo URI, bem como uma descrição sobre os mesmos.

Método	URI	Descrição
POST	/identity/v3/auth/tokens	Enviar o pedido de autenticação
GET	/container/v1/containers	Ir buscar informação sobre todos os containers
GET	/image/v2/images	Ir buscar informação sobre todas as imagens existentes
GET	/compute/v2.1/os-availability-zone	Ir buscar informação sobre todas as availability zones
GET	/identity/v3/auth/projects	Ir buscar informação sobre todos os projetos existentes
GET	/compute/v2.1/os-keypairs	Ir buscar informação sobre os key pairs
GET	/compute/v2.1/servers/detail	Ir buscar informação sobre todas as instâncias existentes
GET	/compute/v2.1/flavors/detail	Ir buscar informação sobre os flavors
GET	:9696/v2.0/networks	Ir buscar informação sobre as networks
GET	/identity/v3/auth/projects	Ir buscar todos os projetos existentes
GET	/volume/v3/{this.\$store.state.project}/volumes/detail	Ir buscar todos os volumes de um determinado projeto
POST	/container/v1/containers	Criar um novo container
POST	/image/v2/images	Criar uma nova imagem
POST	/image/v2/images/{image.id}/import	Enviar o link para criar uma nova imagem
POST	/compute/v2.1/servers/{instance.id}/action	Abrir uma consola para acesso à instância
POST	/compute/v2.1/servers	Criar uma nova instância
POST	/volume/v3/{this.\$store.state.project}/volumes	Criar um volume associado a um projeto
PUT	/compute/v2.1/servers/{this.instance.id}	Editar uma instância existente
DELETE	/compute/v2.1/servers/{instance.id}	Eliminar uma instância existente

Tabela 1 - Lista de todos os métodos/URI/descrição

### 3. Análise crítica e proposta de melhorias

Ao longo de todo o trabalho laboratorial, tivemos como objetivo implementar o que era pedido no enunciado, sendo que podíamos ter investido mais algum tempo no design da aplicação web.

Num trabalho desta natureza, existe sempre algo que se possa melhorar, sendo que poderia ter sido desenvolvido ferramentas de *searching* e filtros para as tabelas onde se encontram listadas as instâncias, as imagens e volumes, e mostrar mais informações sobre as mesmas.

Um outro ponto que pode vir a ser melhorado no futuro, é aprofundar mais o projeto de *networking* (*neutron*), permitindo criar e listar routers, *networks*, *security groups* e mais alguns elementos disponibilizados pelo *neutron*.

Mesmo com as melhorias que podiam ser feitas achámos que o trabalho ficou do nosso agrado, sendo que apresenta as funcionalidades básicas do *Openstack* mesmo sabendo que poderíamos integrar mais ferramentas úteis.



## 4. Conclusão

Terminada a elaboração do trabalho laboratorial, podemos afirmar que os objetivos inicialmente propostos através do enunciado se encontram funcionais, além de termos implementado mais funcionalidades de forma a enriquecer este trabalho.

Com a realização deste trabalho, tivemos conhecimento de uma realidade que não conhecíamos, e concluímos que a utilização de o ambiente virtualizado do Openstack apresenta várias vantagens face ao *hypervisores* habitualmente utilizados bem como a possibilidade de instalar vários projetos passando a aumentar ainda mais o leque de opções que podem ser utilizadas, como é o caso dos *containers*.

## Bibliografia

*Additional services.* (23 de 08 de 2019). Obtido de Openstack:  
<https://docs.openstack.org/newton/install-guide-rdo/additional-services.html>

*All-In-One Single Machine.* (18 de 06 de 2019). Obtido de Openstack:  
<https://docs.openstack.org/devstack/latest/guides/single-machine.html>

*Block Storage API V3 (CURRENT).* (16 de 04 de 2021). Obtido de Openstack:  
<https://docs.openstack.org/api-ref/block-storage/v3/index.html>

*Compute API.* (11 de 03 de 2020). Obtido de Openstack: <https://docs.openstack.org/api-ref/compute/>

*Containers Service API.* (08 de 12 de 2019). Obtido de Openstack:  
<https://docs.openstack.org/api-ref/application-container/>

*Developer Quick-Start.* (19 de 11 de 2019). Obtido de Openstack:  
<https://docs.openstack.org/zun/latest/contributor/quickstart.html>

*Identity API v3 (CURRENT).* (17 de 09 de 2020). Obtido de Openstack:  
<https://docs.openstack.org/api-ref/identity/v3/index.html>

*Image Service API v2 (CURRENT).* (18 de 03 de 2021). Obtido de Openstack:  
<https://docs.openstack.org/api-ref/image/v2/index.html>

*Networking API v2.0.* (23 de 04 de 2021). Obtido de Openstack:  
<https://docs.openstack.org/api-ref/network/v2/index.html>