# API REST

El API REST permite realizar operaciones de creación, lectura y modificación, utilizando los métodos del protocolo http y algunas implementaciones particulares para los endpoints. En la siguiente sub secciones de explicaran cada una de las funciones del API REST.

Nota: Recuerde que toda solicitud sin importar su método debe ser en formato JSON y debe contener al menos la version de “JSONRCP”, es decir, {"jsonrcp": "2.0"}.

Nota: Para los ejemplos y las consultas se utiliza la librería requests de python y se realizan las solicitudes con una instancia de sesiones para mas comodidad.

import requests  
session = requests.Session()

## Lectura (GET)

Existen dos formas de lectura, la primera es en forma de lista (list) y la segunda es en forma de detalle (detail). Para ambas formas el método http utilizado es el GET.

### Lista

La forma del endpoint seria $HOSTNAME/api/path

#### Ejemplo:

* Para hacer una lista completa de contactos

**Entrada:**

session.get(  
 "http://localhost:8014/api/contact",  
 json={"jsonrcp": "2.0"},  
 headers={"Content-Type": "application/json"},  
).json()

**Salida:**

{  
 "jsonrpc": "2.0",  
 "id": null,  
 "result": [  
 {  
 "id": 13,  
 "name": "000",  
 "phone": false,  
 "mobile": false,  
 //...  
 },  
 //...  
 ]  
}

También se puede utilizar filtros personalizados, pasando parámetros adicionales:

#### Ejemplo:

* Filtrar por nombre de contacto

**Entrada:**

session.get(  
 "http://localhost:8014/api/contact",  
 json={"jsonrcp": "2.0", "params": {"name": "admin"}},  
 headers={"Content-Type": "application/json"},  
).json()

**Salida:**

{  
 "jsonrpc": "2.0",  
 "id": null,  
 "result": [  
 {  
 "id": 3,  
 "name": "Administrator",  
 //...  
 }  
 ]  
}

Hay ocasiones en las cuales no desea requerir todos los campos que extrae el API. Para eso existe el parámetro fields al cual se le pasa una lista con los nombres de los campos.

#### Ejemplo:

* Listar los contactos con solo el nombre del usuario

**Entrada:**

session.get(  
 "http://localhost:8014/api/contact",  
 json={"jsonrcp": "2.0", "params": {"fields": ["name"]}},  
 headers={"Content-Type": "application/json"},  
).json()

**Salida:**

{  
 "jsonrpc": "2.0",  
 "id": null,  
 "result": [  
 {  
 "id": 13,  
 "name": "000"  
 },  
 //...  
 ]  
}

Cada consulta devuelve una serie de campos por defecto si no se le pasa el parámetro fields. Si en cambio desea mostrar todos los campos habilitados, ignorando los campos por defecto, puede pasar el valor ALL por el parámetro fields, eso es: {"fields": "ALL"}.

### Detalle

La forma del endpoint seria $HOSTNAME/api/path/<int:id>. La idea principal es obtener los datos de un registro en especifico a la cual se le pasa su ID al final del endpoint. También admite el parámetro fields para extraer campos específicos. Cualquier otro parámetro es ignorado.

#### Ejemplo:

* Detallar el usuario con el ID = 13

**Entrada:**

session.get(  
 "http://localhost:8014/api/contact/13",  
 json={"jsonrcp": "2.0"},  
 headers={"Content-Type": "application/json"},  
).json(),

**Salida:**

{  
 "jsonrpc": "2.0",  
 "id": null,  
 "result": {  
 "id": 13,  
 "name": "000",  
 "phone": false,  
 "mobile": false,  
 //...  
 }  
}

## Creación (POST)

La creación de nuevos registros se usa el mismo enlace para en listar los campos solo con la diferencia que se utiliza el método POST. Es decir, su forma seria $HOSTNAME/api/path. Y se le pasan los campos con los valores a crear como parámetros de la consulta y se retornara el nuevo registro creado con los campos por defecto o los requeridos por el parámetro fields. Adicional a esto, si se ejecuta un proceso extra en la respuesta de la solicitud debe estar un campo con el nombre response\_actions con los datos adicionales de la acción.

#### Ejemplo:

* Crear un nuevo contacto

**Entrada:**

session.post(  
 "http://localhost:8014/api/contact",  
 json={  
 "jsonrcp": "2.0",  
 "params": {  
 "name": "New",  
 "phone": "+1234567890",  
 "country\_id": 1,  
 },  
 },  
 headers={"Content-Type": "application/json"},  
).json(),

**Salida:**

{  
 "jsonrpc": "2.0",  
 "id": null,  
 "result": {  
 "id": 20,  
 "name": "New",  
 "phone": "+1234567890",  
 "mobile": false,  
 "contact\_address": "\n\n \nAndorra",  
 //...  
 "country\_id": [  
 1,  
 "Andorra"  
 ],  
 "company\_type": "person",  
 //...  
 }  
}

## Modificación (PUT, PATCH)

Para modificar los campos de un registro su usa el mismo endpoint que se usa para detallar un registro. Es decir, tiene una forma $HOSTNAME/api/path/<int:id>. Y se pasa la solicitud http o con el método PUT o con el método PACTH. Se pasan los parámetros a cambiar y se retornara el registro modificado. También admite el parámetro fields para obtener campos específicos.

#### Ejemplo:

**Entrada:**

session.put(  
 "http://localhost:8014/api/contact/20",  
 json={  
 "jsonrcp": "2.0",  
 "params": {"name": "Anybody"},  
 },  
 headers={"Content-Type": "application/json"},  
).json()

**Salida:**

{  
 "jsonrpc": "2.0",  
 "id": null,  
 "result": {  
 "id": 20,  
 "name": "Anybody",  
 "phone": "+1234567890",  
 //...  
 }  
}

# Endpoints

A continuación se presentan los endpoint habilitados por el modulo.

## Autentificación

La autentificación por defecto de Odoo utiliza Cookies. Por tal motivo en python se usan las sesiones de la biblioteca requests y para javascript se utiliza el modulo de axios, pero existen muchas otras herramientas.

* Endpoint: /api/authenticate
* Métodos: GET, POST

#### Ejemplo:

* Autentificarse

session.get(  
 "http://localhost:8014/api/authenticate",  
 data={  
 "jsonrcp": "2.0",  
 "params": {"db": "test\_proyect", "login": "admin", "password": "\*\*\*\*\*\*"},  
 },  
)

## Contactos

* Endpoint: /api/contact
* Modelo: res.partner
* Métodos: GET, POST, PUT, PATCH
* Filtros:
  + name: ilike
  + type: =
* Campos por defecto:
  + name
  + phone
  + mobile
  + contact\_address
  + x\_studio\_driver\_license
  + x\_studio\_date\_of\_birth\_1
  + bank\_ids
  + street
  + street2
  + city
  + zip
  + country\_id
  + company\_type
  + property\_account\_receivable\_id
  + property\_account\_payable\_id

## Facturas

Para manejar las facturas se usa dos endpoints.

### Factura

* Endpoint: /api/invoice
* Modelo: account.move
* Métodos: GET, POST, PUT, PATCH
* Filtros:
  + name: ilike
  + partner\_id: =
* Campos por defecto:
  + name
  + partner\_id
  + payment\_reference
  + invoice\_date
  + invoice\_payment\_term\_id
  + journal\_id
  + currency\_id
  + line\_ids

### Linea de Factura

* Endpoint: /api/invoice/line
* Modelo: account.move.line
* Métodos: GET, POST, PUT, PATCH
* Filtros:
  + name: ilike
  + product\_id: =
* Campos por defecto:
  + name
  + product\_id
  + quantity
  + price\_unit
  + tax\_ids

## Firmas

Para manejar las firmas se utilizan dos endpoints.

### Firma

* Endpoint: /api/signature
* Modelo: sign.request
* Métodos: GET, PUT, PATCH
* Filtros:
  + display\_name: ilike
  + state: =
  + subject: ilike

### Enviar firma

* Endpoint: /api/signature/send
* Modelo: sign.send.request
* Metodos: GET, POST
* Filtros:
  + display\_name: ilike
  + filename: ilike
* Campos por defecto:
  + signer\_ids
  + follower\_ids
  + subject
  + filename
  + template\_id

#### Ejemplo:

* Crear una solicitud de firma desde plantilla

**Entrada:**

session.post(  
 "http://localhost:8014/api/signature/send",  
 json={  
 "jsonrcp": "2.0",  
 "params": {  
 "signer\_id": 1,  
 "signer\_ids": [1],  
 "subject": "Message",  
 "filename": "File name",  
 "template\_id": 2,  
 "has\_default\_template": True,  
 },  
 },  
 headers={"Content-Type": "application/json"},  
).json()

Nota: Para crear la solicitud se requiere del parámetro {"has\_default\_template": true} para poder generar la firma correctamente.

**Salida:**

{  
 "jsonrpc": "2.0",  
 "id": null,  
 "result": {  
 "id": 18,  
 "response\_actions": [  
 {  
 "id": 13  
 }  
 ]  
 }  
}

Para consultar el estado de la solicitud puede rastrear por medio del id en la lista del parámetro response\_actions.

#### Ejemplo

* Obtener el estado del ultimo registro creado

**Entrada:**

session.get(  
 "http://localhost:8014/api/signature/13",  
 json={  
 "jsonrcp": "2.0",  
 "params": {"fields": ["state"]},  
 },  
 headers={"Content-Type": "application/json"},  
).json()

Nota: Para crear la solicitud se requiere del parámetro {"has\_default\_template": true} para poder generar la firma correctamente.

**Salida:**

{  
 "jsonrpc": "2.0",  
 "id": null,  
 "result": {  
 "id": 13,  
 "state": "sent"  
 }  
}

## Endpoints adicionales

| Endpoint | Modelo | Metodos permitidos | Filtros |
| --- | --- | --- | --- |
| /api/bank | res.partner.bank | GET, POST, PUT, PATCH | acc\_number, partner\_id |
| /api/country | res.country | GET | name, code |
| /api/account | account.account | GET, POST, PUT, PATCH | name, company\_id |
| /api/payment\_term | account.payment.term | GET | name |
| /api/journal | account.journal | GET | display\_name |
| /api/currency | res.currency | GET | full\_name, symbol |
| /api/product | product.product | GET | display\_name, code, description |
| /api/tax | account.tax | GET | display\_name, description |
| /api/signature/template | sign.template | GET | display\_name |