Conexión y consumo de APIs

Sesión Conceptual 1

{desafío} latam_


Inicio
Desarrollo
Cierre
Objetivo de la sesión
Activación de conceptos clave
Conceptualización
Ejercicios
Quiz
Cierre

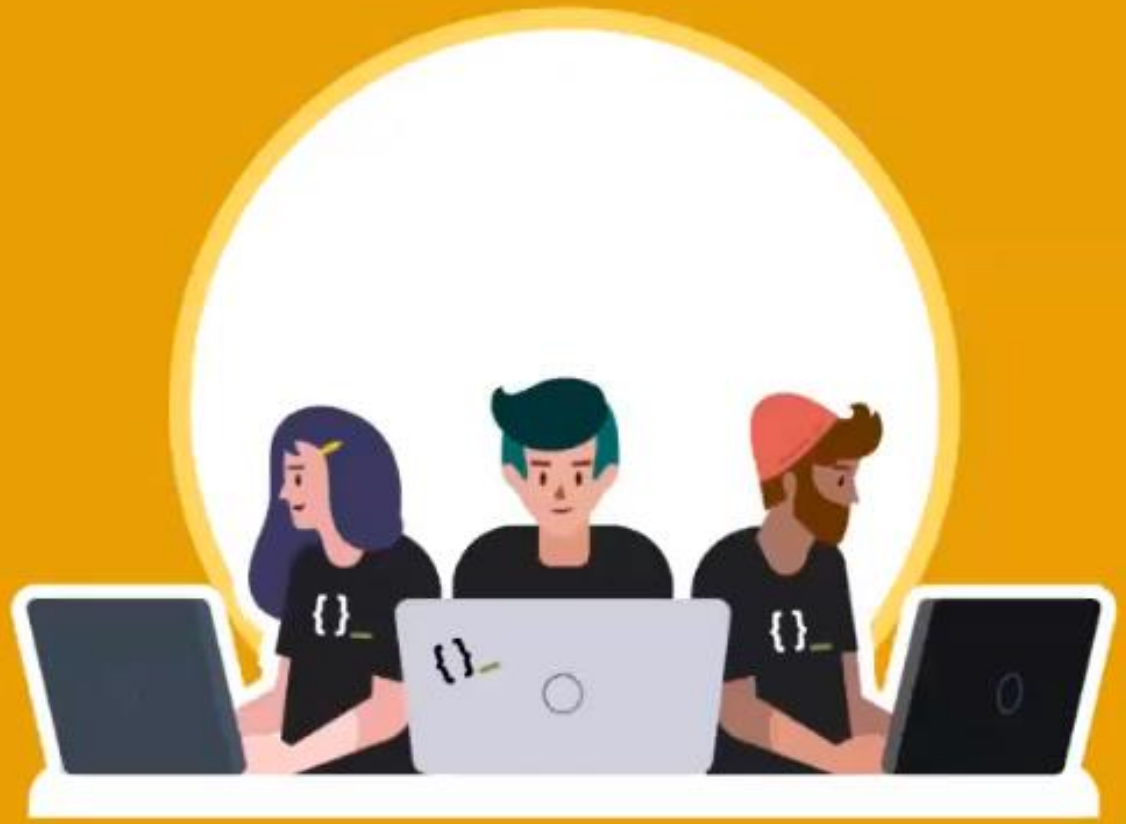
Inicio

10 minutos

{desafío} latam_

Objetivo

- Conectar a APIs mediante Python .
- Extraer información relevante para tu aplicación.



Desarrollo

70 minutos

{desafío}
latam_

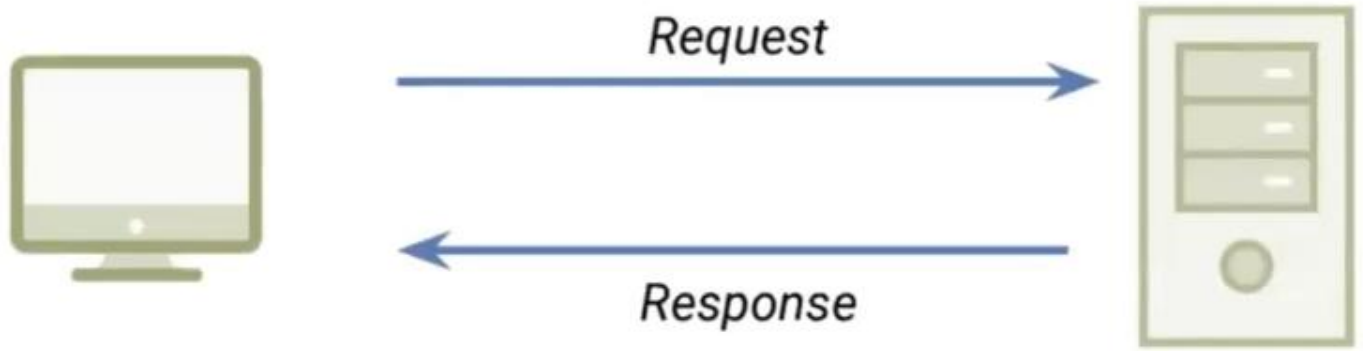
/* Introducción a APIs */

{desafío}
latam_

APIs

- **API**, acrónimo de Application Program Interface, es una interfaz de acceso que permite comunicar programas. Particularmente hablaremos de API.

- Es una interfaz que, al darle ciertas instrucciones, permite manipular datos, enviando o recibiendo a un computador externo, que actúa como "**servidor**".



Tipos de APIs existentes

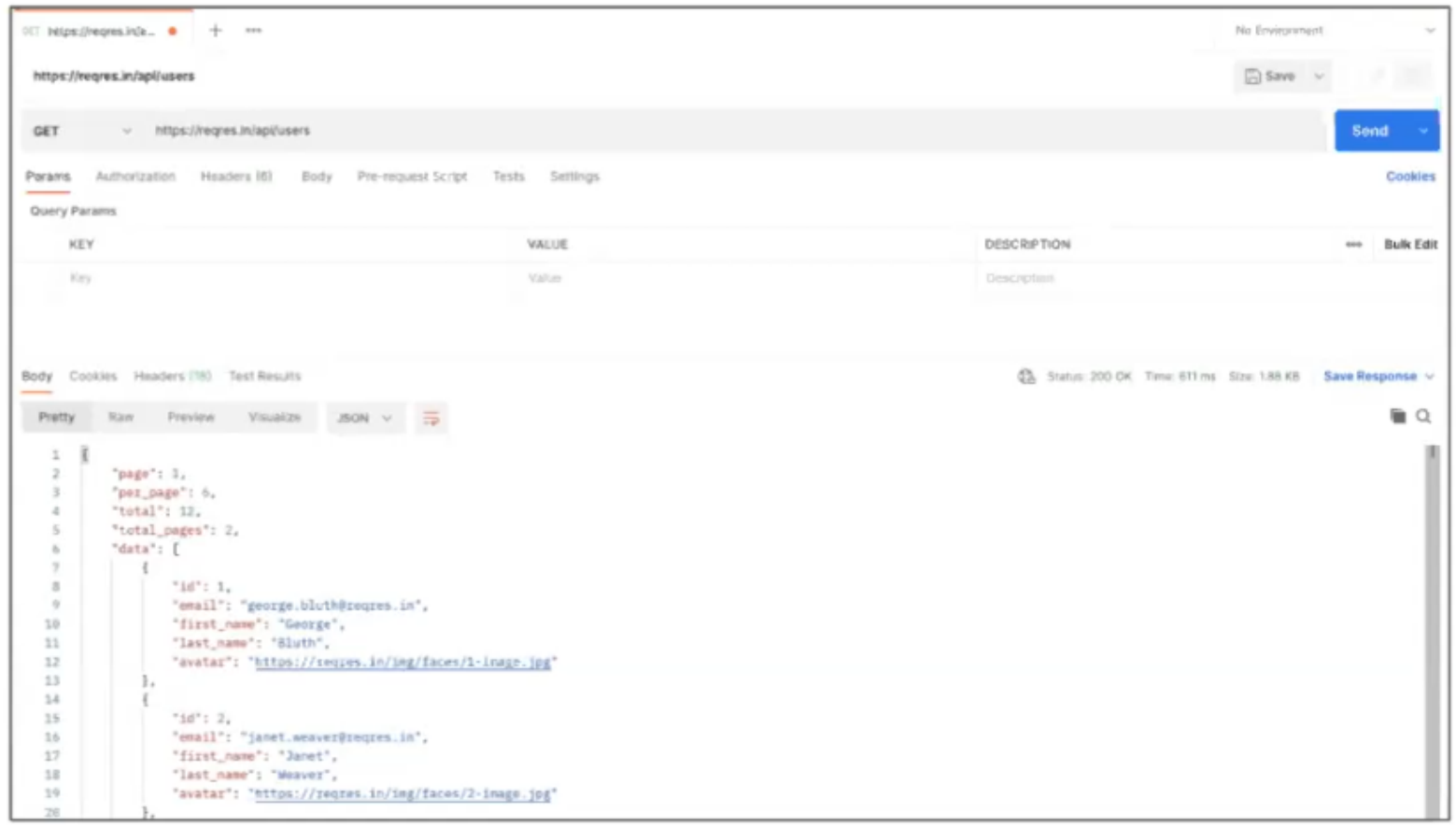
- En internet existen API muy diversas, entres las cuales se encuentran:
  - Clima y temperatura.
  - Cambio de monedas.
  - Indicadores económicos.
  - Servicios para subir archivos.
  - Compra y venta de criptomonedas.
  - Servicios de geolocalización, como Google Maps.
  - Personalizadas, disponibilizadas por internet.

¿Cómo se usa una API?

- Se requiere algún programa que actúe como cliente, realice un **request** (pedido) y pueda procesar el **response** (respuesta), como por ejemplo:
  - Postman
  - Python

Probando una API con Postman

- Ingresa a Postman:
  - Selecciona método **GET**
  - Ingresa URL del API
  - Presiona Send
- Obtendremos como resultado un objeto de **Json**.



{desafío} latam_

Probando una API con Python

- En la pestaña Code snippet, puedes generar de forma automática el código en Python.
  - Se requiere la librería request.
  - En variables, definir:
    - url del API.
    - payload, para enviar datos.
    - headers, para autenticarse.
    - response, resultado de la función request.

  - La variable response, almacenará la respuesta del API.

El código del response

- Según el número con el que se inicia la respuesta, será de distinto tipo:
  - 1xx: Información (Espera!)
  - 2xx: Respuesta correcta (Todo bien!)
  - 3xx: Redirección (No es aquí!)
  - 4xx: Error del cliente (Lo hiciste mal!)
  - 5xx: Error del servidor (No eres tu!)

```
print(response)
```

El contenido del response

- El response, para que lo podamos procesar en Python, se debe de pasar a tipo Json, que nos permitirá transformarlo en objetos manipulables por Python, normalmente listas y diccionarios.

```python
import requests, json

url = "https://reqres.in/api/users"
headers = {'Content-Type': "application/json"}
payload = ""
response = requests.request("GET", url, data=payload, headers=headers)
results = json.loads(response.text)
print(type(results))    # resultado diccionario
print(results)          # contenido
```

El contenido del response

- Se puede observar un diccionario como respuesta, y en donde la clave “data” corresponde a una lista con diccionarios, en donde cada clave corresponderá a un atributo de los usuarios solicitados en la url en el llamado a la función **request**.

```
<class 'dict'>
{'page': 1, 'per_page': 6, 'total': 12, 'total_pages': 2, 'data': [{'id': 1, 'email': 'georg
e.bluth@reqres.in', 'first_name': 'George', 'last_name': 'Bluth', 'avatar': 'https://reqres.
in/img/faces/1-image.jpg'}, {'id': 2, 'email': 'janet.weaver@reqres.in', 'first_name': 'Jane
t', 'last_name': 'Weaver', 'avatar': 'https://reqres.in/img/faces/2-image.jpg'}, {'id': 3, '
email': 'emma.wong@reqres.in', 'first_name': 'Emma', 'last_name': 'Wong', 'avatar': 'https:/
/reqres.in/img/faces/3-image.jpg'}, {'id': 4, 'email': 'eve.holt@reqres.in', 'first_name': '
Eve', 'last_name': 'Holt', 'avatar': 'https://reqres.in/img/faces/4-image.jpg'}, {'id': 5, '
email': 'charles.morris@reqres.in', 'first_name': 'Charles', 'last_name': 'Morris', 'avatar'
: 'https://reqres.in/img/faces/5-image.jpg'}, {'id': 6, 'email': 'tracey.ramos@reqres.in', '
first_name': 'Tracey', 'last_name': 'Ramos', 'avatar': 'https://reqres.in/img/faces/6-image.
jpg'}], 'support': {'url': 'https://reqres.in/#support-heading', 'text': 'To keep ReqRes fre
e, contributions towards server costs are appreciated!'}}
```

Quiz

{desafío}
latam_

/* API REST */

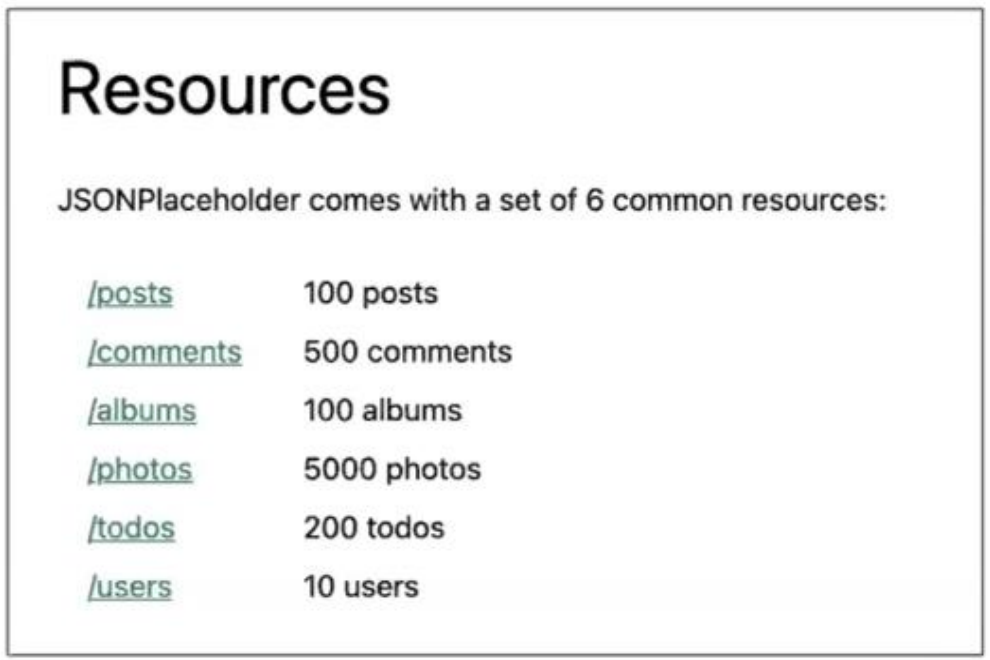
{desafío}
latam_

APIs REST

- REST (Representational State Transfer), es un modelo de arquitectura web basado en el protocolo HTTP para mejorar las comunicaciones cliente-servidor.
- Por esta razón es que las interacciones se realizan por medio de una URL.
- Estas URL's cambiarán según cada API y para cada acción que se desea realizar.
- La API, debe de entregar en su documentación el los **recursos** disponibilizados en internet para la interacción con esta.

Recursos

- En el API de JSON Placeholder, encontramos los recursos para acceder a la siguiente información:
    - Post
    - Comentarios
    - Álbumes
    - Fotos
    - Todos
    - Usuarios
- Los recursos formarán parte de la URL.



Resources

JSONPlaceholder comes with a set of 6 common resources:

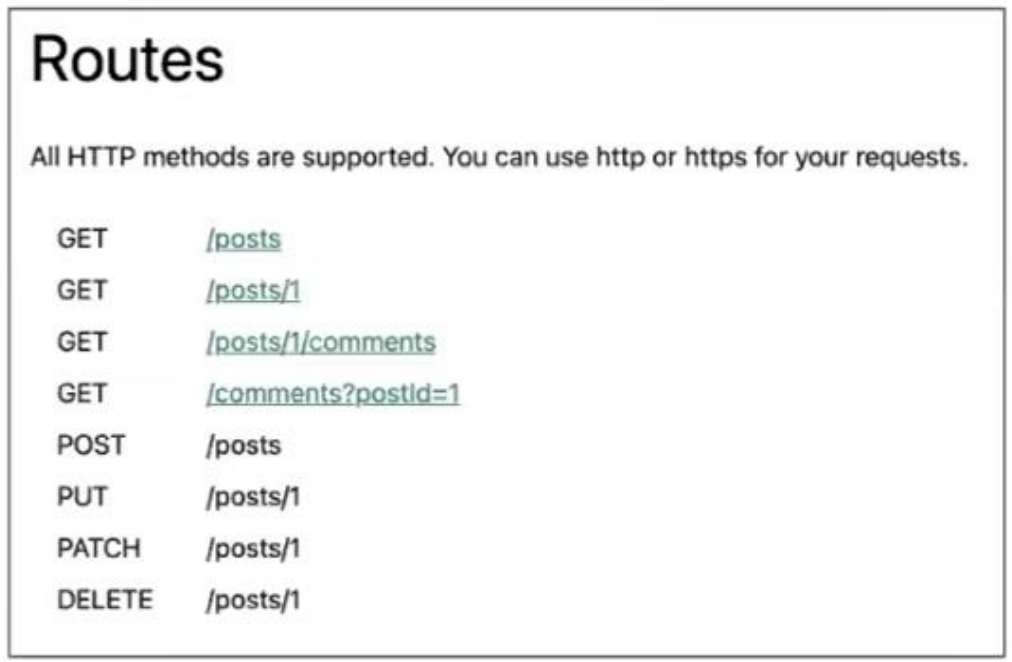
/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	200 todos
/users	10 users

Rutas

- Generalmente en la documentación de la API, estarán publicadas las rutas para que formen parte de la URL.

- La ruta dependerá de la acción que deseemos realizar:

    - GET, obtener información.
    - POST, enviar datos para un nuevo recurso.
    - PATCH, actualiza un recurso existente.
    - DELETE, eliminar un recurso existente.



Routes

All HTTP methods are supported. You can use http or https for your requests.

GET	/posts
GET	/posts/1
GET	/posts/1/comments
GET	/comments?postId=1
POST	/posts
PUT	/posts/1
PATCH	/posts/1
DELETE	/posts/1

GET

- No se necesita utilizar la variable payload ya que solo se obtendrán datos.
- Se debe de indicar GET como argumento en función request.

```python
import requests, json

url = "https://reqres.in/api/users"
headers = {'Content-Type': "application/json"}
payload = ""
response = requests.request("GET", url, data=payload, headers=headers)
results = json.loads(response.text)
print(type(results))    # resultado diccionario
print(results)          # contenido
```

```
<class 'dict'>
{'page': 1, 'per_page': 6, 'total': 12, 'total_pages': 2, 'data': [{'id': 1, 'email': 'georg
e.bluth@reqres.in', 'first_name': 'George', 'last_name': 'Bluth', 'avatar': 'https://reqres.
in/img/faces/1-image.jpg'}, {'id': 2, 'email': 'janet.weaver@reqres.in', 'first_name': 'Jane
t', 'last_name': 'Weaver', 'avatar': 'https://reqres.in/img/faces/2-image.jpg'}, {'id': 3, '
email': 'emma.wong@reqres.in', 'first_name': 'Emma', 'last_name': 'Wong', 'avatar': 'https:/
/reqres.in/img/faces/3-image.jpg'}, {'id': 4, 'email': 'eve.holt@reqres.in', 'first_name': '
Eve', 'last_name': 'Holt', 'avatar': 'https://reqres.in/img/faces/4-image.jpg'}, {'id': 5, '
email': 'charles.morris@reqres.in', 'first_name': 'Charles', 'last_name': 'Morris', 'avatar'
: 'https://reqres.in/img/faces/5-image.jpg'}, {'id': 6, 'email': 'tracey.ramos@reqres.in', '
first_name': 'Tracey', 'last_name': 'Ramos', 'avatar': 'https://reqres.in/img/faces/6-image.
jpg'}], 'support': {'url': 'https://reqres.in/#support-heading', 'text': 'To keep ReqRes fre
e, contributions towards server costs are appreciated!'}}
```

POST

- Se debe de indicar en payload los datos del nuevo recurso.
- Se debe de indicar POST como argumento en función request.

```
import requests, json

url = "https://jsonplaceholder.typicode.com/posts"
headers = {'Content-Type': "application/json"}
# datos de nuevo recurso
payload = '''{"title": "Post 101",
            "body": "Este es nuestro primer post"}'''
# indica ruta de url
response = requests.request("POST", url, data=payload, headers=headers)
results = json.loads(response.text)
print(type(results))    # resultado diccionario
print(results)          # contenido
```

```
<class 'dict'>
{'title': 'Post 101', 'body': 'Este es nuestro primer post', 'id': 101}
```

PUT

- Se debe de indicar el recurso que se actualizará.
- Se debe de indicar en payload los datos de actualización del recurso.
- Se debe de indicar PUT como argumento en función request.

```
import requests, json

# indica post a actualizar
url = "https://jsonplaceholder.typicode.com/posts/20"
headers = {'Content-Type': "application/json"}
# datos de nuevo recurso
payload = '''{"title": "Cambio de Post",
            "body": "Este es un cambio en el post 20",
            "userId":"1"}'''
# indica ruta de url
response = requests.request("PUT", url, data=payload, headers=headers)
results = json.loads(response.text)
print(type(results))    # resultado diccionario
print(results)          # contenido
```

```
<class 'dict'>
{'title': 'Cambio de Post', 'body': 'Este es un cambio en el post 20', 'userId': '1', 'id':
20}
```

DELETE

- Se debe de indicar el recurso que se eliminará.
- Se debe de indicar el payload vacío.
- Se debe de indicar DELETE como argumento en función request.

```python
import requests, json

# indica post a eliminar
url = "https://jsonplaceholder.typicode.com/posts/20"
headers = {'Content-Type': "application/json"}
# payload sin datos
payload = ""
# indica ruta de url
response = requests.request("DELETE", url, data=payload, headers=headers)
print(response.text) # resultado de delete
```

```
<Response [200]>
```

Quiz

{desafío}
latam_

Cierre

10 minutos

{desafío} latam_

- ¿Qué es una API?
- ¿Qué es POSTMAN y para qué nos sirve?
- ¿Qué son los Métodos HTTP?
- ¿Cuántos Métodos HTTP existen y cuál es la utilidad de cada uno de ellos?
- ¿Para qué sirven los códigos de respuesta? ¿Cuáles son los más importantes de conocer?

{desafío} latam_

Academia de talentos digitales