

{desafío}
latam_

API _

Sesión Experimental 2



Itinerario



Activación de conceptos



Desarrollo Prueba



Panel de discusión

/* Activación de conceptos */

¿Cómo cambiaría la nota de Antonia por un 4?

```
notas = {"Camila": 7, "Antonio": 5, "Felipe": 6, "Antonia": 7}
```

1. notas[3] = 4
2. notas.insert("Antonia", 4)
3. notas["Antonia"] = 4
4. notas = {"Antonia": 4}
5. notas[4] = "Antonia"

Al iterar sobre un diccionario usando 2 iteradores, y la función `items()`, el primer iterador representa _____ y el segundo iterador representa _____

1. El valor, la clave
2. El índice, la clave
3. La clave, el índice
4. La clave, el valor
5. No se puede usar 2 iteradores

¿Cómo se pueden unir 2 listas en un diccionario, siendo la primera la lista de claves y la segunda los valores?

1. La única forma es iterando ambas listas y usar sus elementos para formar el nuevo diccionario
2. Usando la función zip sobre ambas listas, y luego iterar el zip para armar el diccionario
3. Usando la función zip, y luego la función dict sobre el zip
4. Usando la función dict sobre ambas listas
5. Solo usando la función zip

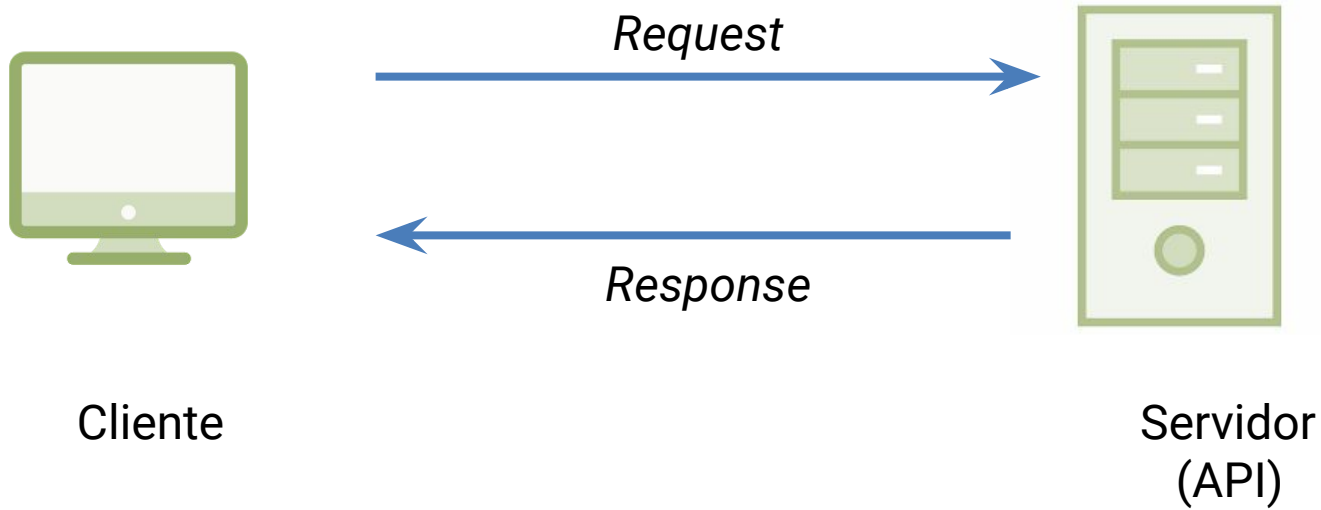
¿Qué se requiere para implementar groupby correctamente?

1. Importar groupby desde numpy
2. Se aplica al objeto como lista.groupby()
3. Se debe ordenar previamente los datos de la estructura
4. Retorna un array unidimensional
5. Solo se puede aplicar a strings

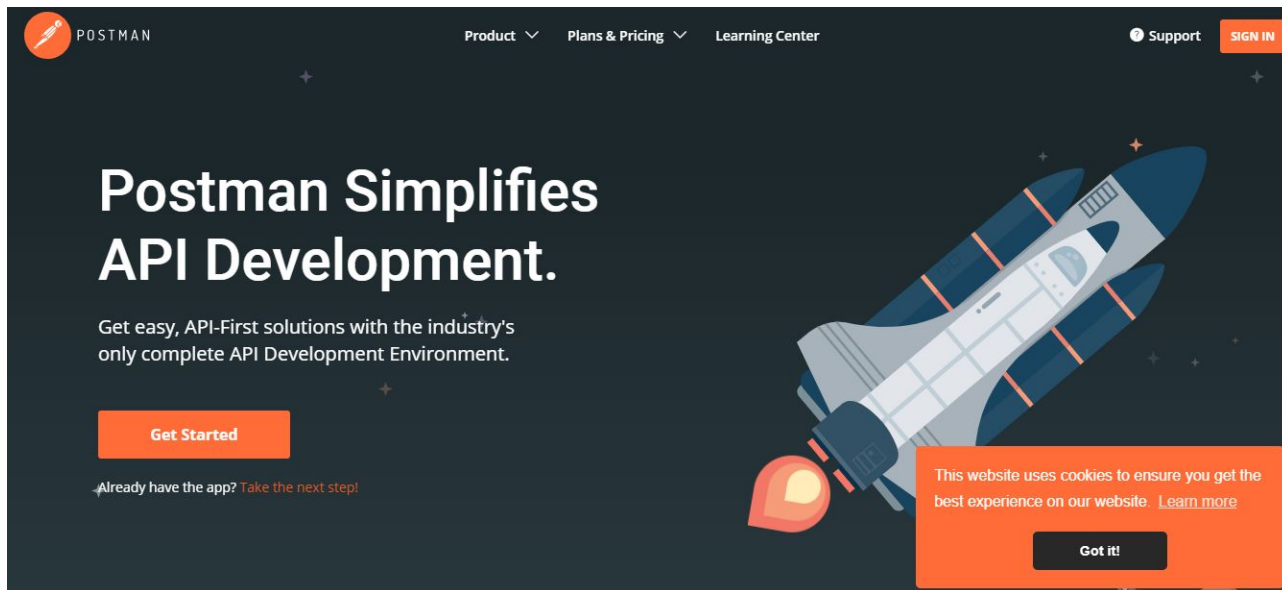
Tipos de API existentes

- Clima y temperatura
- Cambio de monedas
- Indicadores económicos
- Servicios para subir archivos
- Compra y venta de criptomonedas
- Servicios de geolocalización, como Google Maps

Funcionamiento de una API



Postman

The image shows the Postman website's landing page. The background is dark blue with a stylized rocket ship illustration on the right side. The rocket is white with blue and orange accents, and it has a large orange flame at the bottom. The text is white and orange. The navigation bar at the top includes the Postman logo (an orange circle with a white pipette icon) and the word "POSTMAN". To the right of the logo are links for "Product", "Plans & Pricing", and "Learning Center". Further right are links for "Support" and a "SIGN IN" button. The main heading "Postman Simplifies API Development." is in large white font. Below it is a subheading "Get easy, API-First solutions with the industry's only complete API Development Environment." and a "Get Started" button. At the bottom left, there is a link "Already have the app? Take the next step!". At the bottom right, there is a cookie notice and a "Got it!" button.

POSTMAN

Product ▾ Plans & Pricing ▾ Learning Center

Support SIGN IN

Postman Simplifies API Development.

Get easy, API-First solutions with the industry's only complete API Development Environment.

[Get Started](#)

Already have the app? [Take the next step!](#)

This website uses cookies to ensure you get the best experience on our website. [Learn more](#)

[Got it!](#)

Hacer una request con Postman

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'NEW', 'Runner', 'Import', and a plus icon. Below this is a status bar with a notification: 'Bring your whole team to Postman for free! Update the app to experience the new features. [What's new?](#)'. The main interface is divided into several sections. The top section shows the URL 'https://jsonplaceholder.typicode.com/posts' and the method 'GET'. Below this are tabs for 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The 'Body' tab is selected, showing a JSON response. The response is a list of four posts, each with 'userId', 'id', 'title', and 'body' fields. The status bar at the bottom right indicates 'Status: 200 OK' and 'Time: 275 ms'.

https://jsonplaceholder.typicode.com/posts

GET

Params Send Save

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (17) Test Results

Status: 200 OK Time: 275 ms

Pretty Raw Preview JSON

```
1 {
2   "userId": 1,
3   "id": 1,
4   "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
5   "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
6 },
7 {
8   "userId": 1,
9   "id": 2,
10  "title": "qui est esse",
11  "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aper
12 },
13 {
14   "userId": 1,
15   "id": 3,
16   "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
17   "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit
18 },
19 {
20   "userId": 1,
21   "id": 4,
22   "title": "eum et est occaecati",
23   "body": "ullam et saepe reiciendis voluptatem adipisci\nsit amet autem assumenda provident rerum culpa\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\nquis sun
24 },
25 }
```

JSON

- Acrónimo de JavaScript Object Notation
- Envía información en texto plano entendible tanto por personas como por programas
- No se necesita saber javascript para usarlo
- En python, se debe importar cómo `"import json"`

Consumir una API desde Python

- Se puede obtener el código desde Postman

```
1 import requests
2
3 url = "https://jsonplaceholder.typicode.com/posts"
4
5 headers = {
6     'cache-control': "no-cache",
7     'Postman-Token': "2defb9f3-9b11-499b-be4f-82505a2f8e1a",
8 }
9
10 response = requests.request("GET", url, headers=headers)
11
12 # Cuerpo de la respuesta o "body"
13 print(response.text)
14
15 # Código de la respuesta
16 print(response)
```

Códigos de respuesta

- 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!)

Transformar la respuesta a JSON

```
1 import json
2 results = json.loads(response.text)
3 print(type(results)) # Veremos que el resultado es una lista
4 print(results[0]) # Mostramos el primer elemento
```

```
"[\n {\n   \"userId\": 1,\n   \"id\": 1,\n   \"title\": \"sunt aut facere repellat provident occaecati excepturi opti\no reprehenderit\", \n   \"body\": \"quia et suscipit\\nsuscipit recusandae consequuntur expedita et cum\\nreprehenderit m\nolestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto\" \n }, \n {\n   \"userId\": 1,\n   \"i
```



```
<class 'list'>
{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'body': 'quia et\nsuscipit\\nsuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem s\nunt rem eveniet architecto'}
```

{desafío}
latam_

Transformar request a función

```
1 import json
2 import requests
3
4
5 def request(requested_url):
6     headers = {
7         "cache-control": "no-cache",
8         "Postman-Token": "2defb9f3-9b11-499b-be4f-82505a2f8e1a",
9     }
10    response = requests.request("GET", requested_url, headers=headers)
11    return json.loads(response.text)
12
13 request('http://jsonplaceholder.typicode.com/posts')
```


Recursos

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

Note: resources have relations. For example: posts have many comments, albums have many photos, ... see below for routes examples.

Rutas

Routes

All HTTP methods are supported.

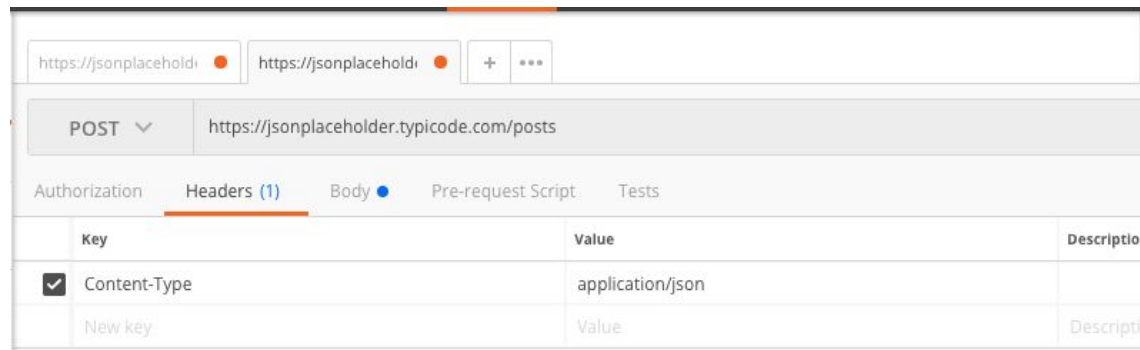
GET	/posts
GET	/posts/1
GET	/posts/1/comments
GET	/comments?postId=1
GET	/posts?userId=1
POST	/posts
PUT	/posts/1
PATCH	/posts/1
DELETE	/posts/1

Note: you can view detailed examples [here](#).

Request

- URL
- HTTP method o “verbo”
 - GET: Leer datos
 - POST: Crear datos
 - PUT: Actualizar datos
 - DELETE: Eliminar datos
- Header
- Body

Crear un recurso (POST) desde Postman

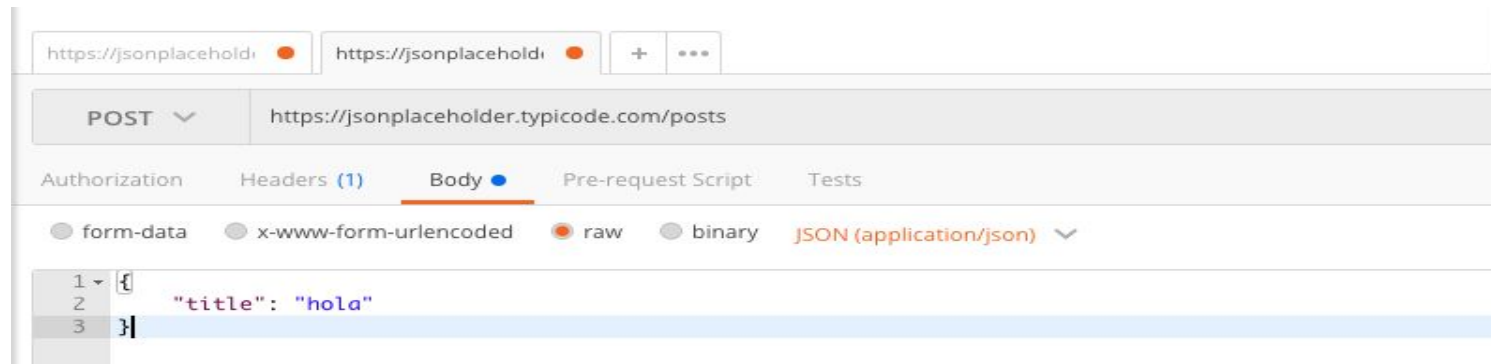


https://jsonplaceholder | https://jsonplaceholder | + ...

POST ▾ https://jsonplaceholder.typicode.com/posts

Authorization Headers (1) Body ● Pre-request Script Tests

Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/json	
New key	Value	Description



https://jsonplaceholder | https://jsonplaceholder | + ...

POST ▾ https://jsonplaceholder.typicode.com/posts

Authorization Headers (1) Body ● Pre-request Script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ▾

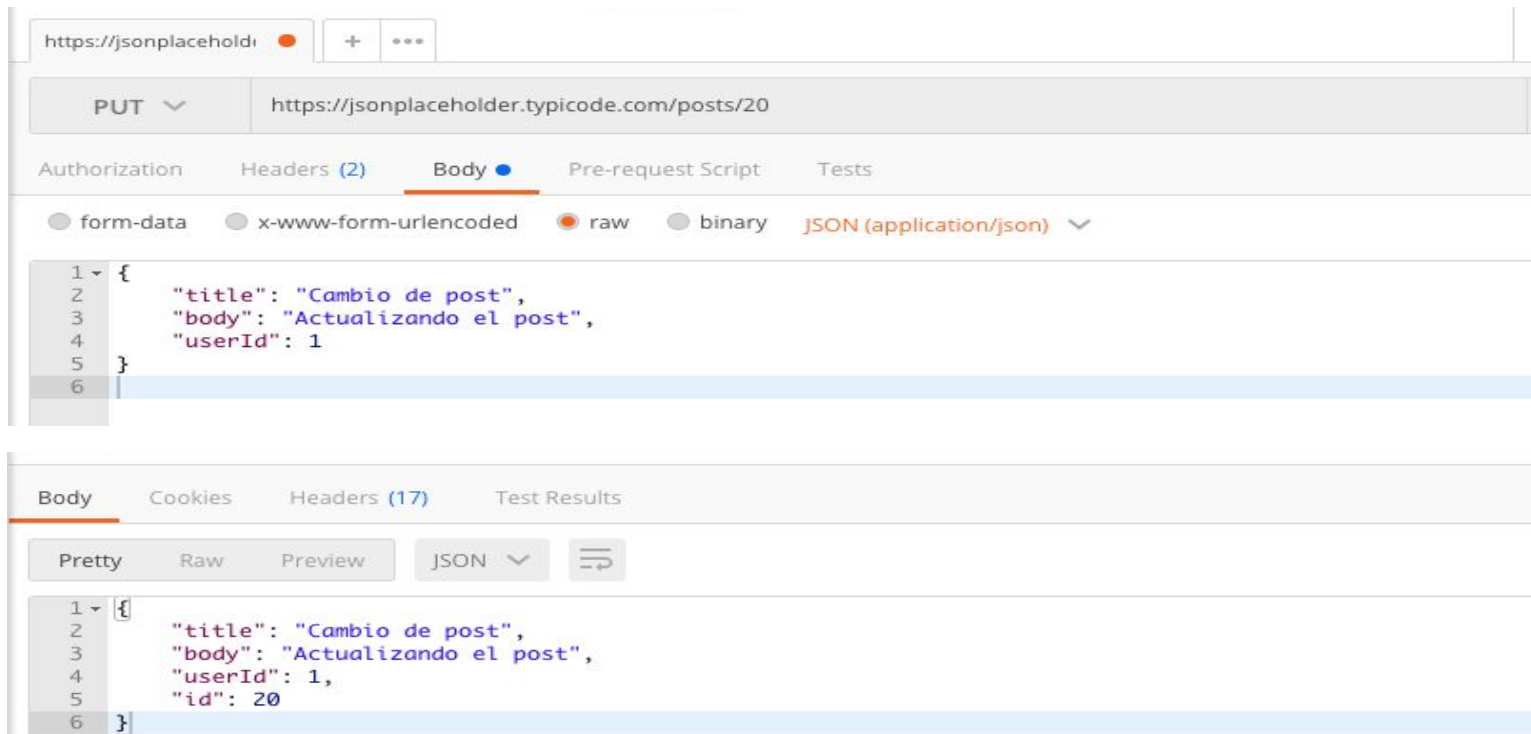
```
1 {  
2   "title": "hola"  
3 }
```

Creando un recurso desde Python

```
1 import requests
2
3 url = "https://jsonplaceholder.typicode.com/posts"
4
5 payload = "{\r\n\t\"title\": \"Post 101\",\r\n\t\"body\": \"Este es nuestro primer post\",\r\n\t\"userId\": 1\r\n}"
6 headers = {
7     'Content-Type': "application/json",
8     'cache-control': "no-cache",
9     'Postman-Token': "ac158b42-ca0e-4ada-92f0-f362962775e7"
10 }
11
12 response = requests.request("POST", url, data=payload, headers=headers)
13
14 print(response.text)
```

```
{
  "title": "Post 101",
  "body": "Este es nuestro primer post",
  "userId": 1,
  "id": 101
}
```

Actualizando un recurso (PUT) desde Postman



Actualizar un recurso desde Python

```
import requests

url = "https://jsonplaceholder.typicode.com/posts/1"

payload = "{\n\t\"title\": \"Cambio de post\",\n\t\"body\": \"Actualizando el post\",\n\t\"userId\": 1\n}\n"
headers = {
    'Content-Type': "application/json",
    'cache-control': "no-cache",
    'Postman-Token': "563a09ea-fdff-4a54-be88-7d8ec46bd01c"
}

response = requests.request("PUT", url, data=payload, headers=headers)

print(response.text)
```

Borrar un recurso (DELETE) desde Python

```
import requests

url = "https://jsonplaceholder.typicode.com/posts/1"

payload = ""
headers = {
    'Content-Type': "application/json",
    'cache-control': "no-cache",
    'Postman-Token': "fc3777ed-499a-4a9b-8250-43c1b1c8b6c3"
}

response = requests.request("DELETE", url, data=payload, headers=headers)

print(response.text)
```


SSL

- Acrónimo de Secure Sockets Layer
- Es una capa de seguridad que establece encriptación entre el navegador y el servidor
- Evita que la información que enviamos o recibimos sea leída por un tercero.

1.



Alice tiene 2 claves:

- Una **privada** para cifrar *su* mensaje: **AWETGT**
- Una **pública** para descifrar *su* mensaje: **EGEUJU**

2.



Alice se junta con Bob y le pasa *su* clave **pública** para descifrar



3.



Alice envía a Bob un mensaje cifrado con su clave privada: **AWETGT**



4.

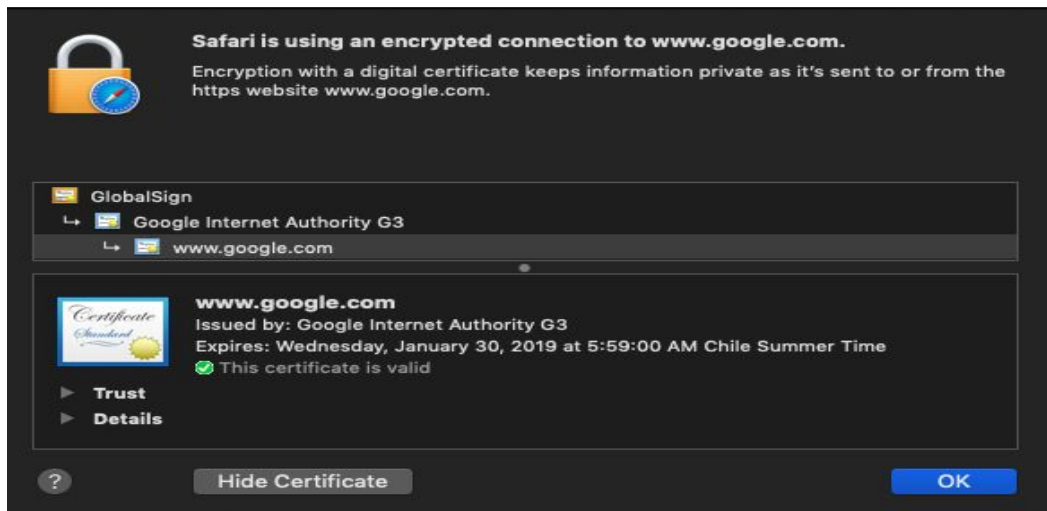
Bob utiliza la clave que Alice le pasó en el paso 2 para descifrar el mensaje



La magia que ocurre detrás

Al momento de conectarnos a un sitio web que utilice SSL con un navegador, se establece un acuerdo, llamado en inglés handshake, de forma automática.

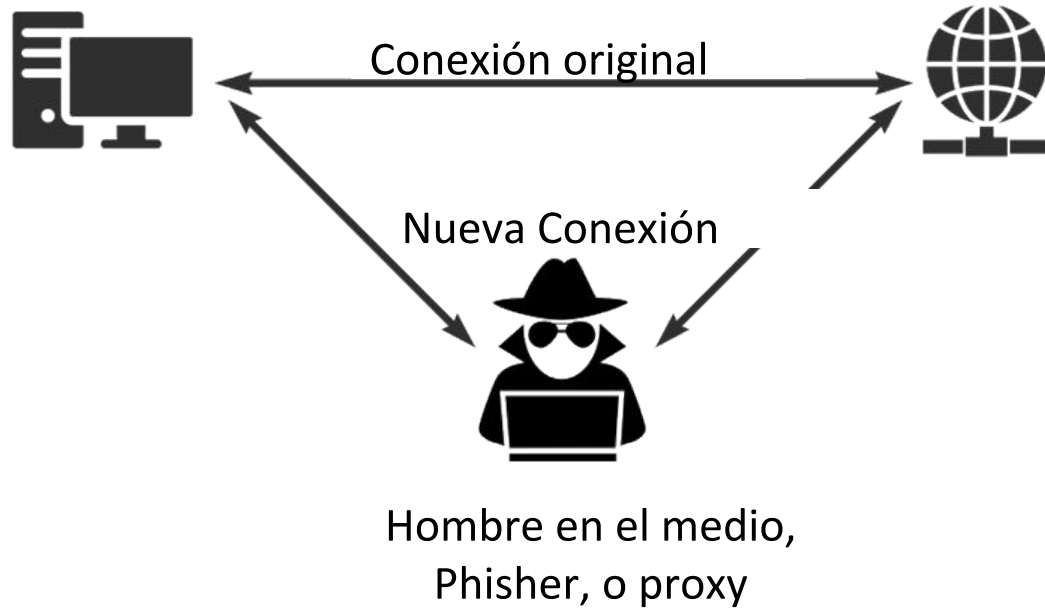
En este handshake el servidor envía un certificado que tiene el nombre del sitio y la clave pública al cliente.



Conectándose a SSL

- Postman identifica automáticamente si un *request* ocupa HTTPS y genera el código para conectarnos
- La librería requests, por defecto, tiene la verificación de certificados SSL habilitada; Si no puede verificar un certificado durante un *request*, ocurrirá un `SSLError`

Hombre en el medio



Autenticando desde Python

```
1 import json
2 import requests
3
4
5 def request(requested_url):
6     headers = {
7         "app_id": "e51457d3",
8         "app_key": "6cf09f1d0edbefd2381f937f396150a3",
9     }
10    response = requests.request("GET", requested_url, headers=headers)
11    return json.loads(response.text)
12
13
14 word = "test"
15 request("https://od-api.oxforddictionaries.com:443/api/v1/entries/en/{}".format(word))
```

Otros tipos de autenticación

- Ingresar la clave en el body
- Renovar el token en cada solicitud
- OAuth2

Siempre se lee leer la documentación de la API

Cómo manejar los tokens

- Con args al ejecutar el script
- Definiendo variables de entorno desde la terminal (Linux, OSX)
- Si se usa variables de entorno, se pueden persistir en `.bashrc` o `.bashprofile`

`/* Prueba */`

/* Panel de discusión */

Elementos clave que aprendimos

- Un algoritmo es una serie de pasos finitos para resolver un problema
- Crear el algoritmo es la parte compleja de la programación
- Para crear algoritmos disponemos de herramientas como los diagramas de flujo y pseudocódigo
- Las funciones son importantes para ordenar el código y evitar repetir varias veces lo mismo
- Los ciclos son clave para evitar repetir código y hacer el programa escalable
- Un problema puede tener más de una solución, y cuál será la mejor dependerá de cada caso
- Las estructuras de datos como listas y diccionarios nos permiten almacenar y manejar grandes cantidades de datos en una sola variable
- Podemos comunicarnos con otros programas de forma segura mediante el uso de una API, y usar estos datos para nuestros propios programas

Otras áreas interesantes

- Estructuras de datos
- Complejidad algorítmica
- Almacenamiento y recuperación de información
- Bases de datos
- QA y Construcción de pruebas automatizadas
- Arquitectura de software
- Ciencia de datos
- Inteligencia artificial

{desafío}
latam_

*Academia de
talentos digitales*

www.desafiolatam.com