CONSUMO DE UNA API:

De la Tierra a las Estrellas con Postman

Realizado por:
Diana Mª Pascual García

Contenidos

Situación.....3

Recorrido con API 1: IDEALISTA 4

Recorrido con API 2: Star Wars.....5

Recorrido con API 3: JSONPlaceholder.....6

Reflexión.....7

Documentación consultada y enlaces.....8

Situación

El objetivo de esta práctica era seleccionar una API que esté publicada en internet, de una temática de mi preferencia, de consumo gratuito, para ser probada con Postman.

Mi propósito desde el primer momento, era seleccionar una API que para mi resultase valiosa, de modo que convirtiera la práctica no en un mero trabajo rutinario, sino en algo que despertara mi interés. Por ello, tenía muy claro que elegiría o una API que se ajustara a mis necesidades actuales o que estuviera vinculada a mis gustos y preferencias.

En primer lugar, inspeccioné el listado proporcionado en las instrucciones y me encontré con la dificultad de que todas las que me interesaban ya estaban fuera de servicio. A pesar de ello, esto me ayudó a considerar alternativas interesantes.

Por mi situación actual, puede que en algún momento tenga que cambiar de vivienda. Es una dificultad que suele rondarme la cabeza y creo que por ello fue por lo que pensé en buscar si IDEALISTA contaba con API, y así era. Tenía que ponerme en contacto con ellos para poder obtener las claves necesarias, previa descripción del motivo de mi petición. Solo así, me mandaría también la documentación que describe su uso. Antes de hacerlo, investigué si era posible usarla con Postman y parece que no había problemas.

El problema, es que tardan en contestar a este tipo de consultas, por lo que me planteé iniciar mi práctica con otra API orientada a mis aficiones. Esta me permitiría familiarizarme con Postman, iniciarme en el uso de las request, etc. y así coger más habilidad para cuando me llegaran las credenciales de IDEALISTA. Entre mis muchas y variadas aficiones, me encanta todo lo relacionado con el mundo de STAR WARS. Me he criado viendo sus películas y mi afición continúa gracias al interés que sigue despertando a día de hoy por todo el material que se sigue creando (Series de animación y no animadas, comics, juegos de mesa, etc.). Descubrí SWAPI.dev, una interfaz de programación de aplicaciones, que proporciona datos relacionados con el universo de Star Wars.

A continuación, describiré toda esta odisea, de la que dudo que haya un final, incluso después de que termine esta práctica.

Recorrido con API 1: IDEALISTA

Elegí esta API (https://api.idealista.com) con el deseo de indagar por ella y descubrir los precios de los alquileres y compras de viviendas en Sevilla y sus alrededores.

Contacté por los medios indicados en la página para que me enviaran las claves pertinentes. Tras unos 10 días, recibí en mi correo las claves y un par de páginas de manual de uso (la documentación se adjunta en “Documentación consultada”).

Como puede verse en el documento exportado de Postman sobre Idealista, conseguí sin demasiados problemas obtener el token oportuno para poder acceder. Sin embargo, al iniciar las pruebas con las peticiones, no paraba de darme errores. Normalmente el 400: Bad request, incluso copiando y pegando los ejemplos de la documentación y el 404. Me di cuenta que en la documentación había ejemplos señalaban a una versión 3.5 de la API y otro a a la 3. Por tanto, volví a ponerme en contacto con la empresa. Les solicité información actualizada y orientación sobre el error que pudiera estar cometiendo. Respondieron muy amablemente.

Mi error es que estaba añadiendo los parámetros en el body como Content-Type: application/json. Necesitaba enviarlos con Content-Type: application/x-www-form-urlencoded:

```
curl --location 'https://api.idealista.com/3.5/es/search' \
--header 'Signature: {signature}' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Bearer {bearer}' \
--data-urlencode 'propertyType=homes' \
--data-urlencode 'operation=sale' \
--data-urlencode 'locationId=0-EU-ES-28'
```

Realicé los cambios oportunos y como por arte de magia, los datos comenzaron a aparecer (desde ahora soy fan del estado 200).

El archivo JSON adjunto muestra mi historial de búsquedas. Comencé buscando casas en venta, pero luego amplié mi búsqueda para incluir alquileres e incluso habitaciones. Los precios están desorbitados, no puedo descartar ninguna opción.

Es cierto, que he comprobado que, mientras no hay problemas a la hora de solicitar las búsquedas a través de los campos definidos como requeridos en la documentación, otras variables que también deberían filtrar, no actúan como deberían. Dejo de muestra una de las solicitudes con la función de ordenamiento, que debería permitir organizar el listado en orden ascendente (asc) o descendente (desc).

Recorrido con API 2: Star Wars

Swapi.dev es una API (Interfaz de Programación de Aplicaciones, por sus siglas en inglés) que proporciona datos relacionados con el universo de Star Wars. Permite acceder a información sobre personajes, películas, naves espaciales, vehículos, especies y planetas presentes en las películas, series, libros y otros medios relacionados con Star Wars. Es una herramienta útil para fanáticos que desean acceder a información detallada sobre el universo de la saga o para creadores de contenidos y entretenimiento.

La documentación de esta API se encuentra colgada directamente en su web de información. Contempla la posibilidad de descargarla en Postman (*Véase en el proyecto de Postman la primera petición guardada*).

Realicé algunas consultas para probar diferentes estrategias de filtrado de la información. Creé parámetros que me permitieron agilizar mis búsquedas guardando la variable y el valor específico. Tras esto, decido buscar a mi personaje favorito: Ahsoka Tano... Pero no está incluída, ¿Cómo puede ser?

Compruebo que sí están los datos de su planeta de origen e incluso su especie (son las búsquedas específicas realizadas en estas categorías).

Por tanto, me dispongo a realizar una llamada Post para incorporar a este personaje. Para ello diseño un body en Json que se adapta fielmente a los parámetros que se definen en el manual. Sin embargo, la API no lo permite. Un poco tarde comprendí que efectivamente es lógico que una API que cuenta con una base de datos impida la incorporación de datos a cualquier consumidor que pueda acceder a ella.

El propio manual indica que la variable de filtrado de cada categoría es el “name”, no pudiéndose hacer desde otra categoría diferente. Aun así, dejo una muestra del error que sale al buscarlo por otra categoría. También apuntaría como propuesta de mejora en la experiencia de búsqueda por filtrado, que no compartieran la misma denominación en los diferentes listados. Al menos, cambiarlo a “namePeople”, “namePlanet”. De este modo, podrías guardar todas las variables de búsqueda e ir modificando sus valores sin tener que machacar los resultados, pudiendo ordenar mejor tu documentación.

Pero mi reivindicación por la incorporación de este personaje no podía quedar así. De ahí que optara por recurrir a una tercera API.

Recorrido con API 3: JsonPlaceholder

JSONPlaceholder es una API que proporciona datos simulados en formato JSON para propósitos de desarrollo y pruebas.

No requiere de registro ni autenticación. Incluye datos simulados que representan diferentes tipos de recursos, como usuarios, publicaciones, comentarios, álbumes, fotos, etc. Estos datos simulados son útiles para realizar pruebas de integración, prototipos rápidos y ejercicios de desarrollo.

Cuenta con una documentación clara y concisa en la propia web que describe todos los endpoints disponibles, los parámetros de las solicitudes y la estructura de los datos de respuesta.

¿Por qué la elegí? Porque podía “mockear” una situación de petición de incorporación de datos en una API a través de un POST. Para ello, seleccione la estructura de un request como si fuera una usuaria que realiza una reclamación para que se incorpore al persona que en la aplicación SWAPI y aporta una foto de la misma para demostrar su existencia.

Para ello he tenido que buscar el último “id” que estaba registrado (era el 100) y añadirle un dígito más. Sin embargo, en el caso de esta API no hubiera hecho falta porque si pones un id que ya existe, automáticamente te asigna el número del primero que esté libre.

Como puede observarse en la documentación del proyecto, la petición es todo un éxito.

Reflexión

Durante mi trabajo con las APIs, he tenido la oportunidad de sumergirme en el mundo del desarrollo y la integración de aplicaciones. Cada una de ellas me ofreció una experiencia única y desafiante, pero también gratificante. Nunca pensé que pudiera llegar a disfrutar con esta práctica. Confieso que al principio me sentí un poco aturdida y dudé de mis posibilidades.

Aunque te he contado mi historia, el inicio de mi viaje fue con JSONPlaceholder. Esta API me permitió familiarizarme con los conceptos básicos, así como practicar la realización de solicitudes HTTP y la manipulación de datos JSON. Su facilidad de uso y documentación clara fueron de gran ayuda para mí mientras exploraba diferentes escenarios de prueba.

Mi experiencia con la API SWAPI fue con la que más disfruté. Aunque inicialmente me encontré con algunos desafíos al comprender la estructura de la API y sus limitaciones, la documentación detallada y creo que mi persistencia me ayudaron a superar estos obstáculos.

Finalmente, mi trabajo con la API de Idealista me brindó la oportunidad de interactuar con datos del mundo real. Aunque esta API requería un proceso de autenticación y algunas consultas más complejas, pude aprovechar al máximo mi experiencia previa con las otras APIs para enfrentar estos desafíos.

En general, creo que trabajar con estas tres APIs me ha permitido comprender la relevancia que tiene su uso y las posibles implicaciones que tiene aprender a manejarlas. Creo que he adquirido habilidades prácticas en la realización de solicitudes, manipulación de datos y resolución de problemas.

En este viaje, tiene sentido aquello de que “el miedo es el camino hacia el lado oscuro”. Al superar el miedo, he encontrado un nuevo camino hacia el crecimiento y la realización, que me ha hecho hasta disfrutar de la práctica. ¡Que la fuerza esté con aquellos que se atreven a enfrentar lo desconocido! ;)

Documentación consultada y enlaces

Enlaces:

- https://jsonplaceholder.typicode.com/
- Acceso directo a guía: https://jsonplaceholder.typicode.com/guide/
- https://swapi.dev/
- Guía para uso de Postman: https://www.redsauce.net/blog/es/postman-quick-guide

OAuth 2.0 Authentication

Description

Request a **Bearer** token for OAuth authentication.

Resource URL

POST **/oauth/token**

Headers

name	type	description	example
Authorization	string	Your Api key and secret, encoded. See below for more info.	`Basic your_base64_credentials`
Content-Type	string	Request content type	`application/x-www-form-urlencoded;charset=UTF-8`

Encoding your authorization header:

1. URL encode your API key and secret according to RFC 1738
2. Concatenate the encoded API key, a colon character ":", and the secret into a single string
3. Base64 encode the string from the previous step

For example, if your API key is "abc" and your secret "123", the `base64("abc:123")` is `YWJjOjEyMw==`

So, your Authorization header should contain:

`Basic YWJjOjEyMw==`

Parameters

This call accepts the following parameters, `x-www-form-urlencoded`:

name	type	description	notes
grant_type	string	Value: `client_credentials`	
scope	string	Values: `read` or `write`	This field is optional.

Response

name	type	description	notes
access_token	string	Your **Bearer** token	
token_type	string		
expires_in	number	Number of seconds until token expires	
scope	string	Token scope	

Example:

JSON response

```
{
  "access_token": "your_token",
  "token_type": "bearer",
  "expires_in": 43199,
  "scope": "read"
}
```

Errors

Status code	error	error description	meaning
401	unauthorized	Full authentication is required to access this resource	You are probably missing the **Authorization** header, or its format is not correct (It should start with `Basic`)
401	unauthorized	Bad credentials	Your Authorization header format is correct, but the base64 of your apikey and secret is wrong
400	invalid_request	Missing grant type	The parameter **grant_type** is missing

Example:

JSON response

```
{
  "error": "unauthorized",
  "error_description": "Full authentication is required to access this resource"
}
```

Complete request example

CURL

```
curl -X POST -H "Authorization: Basic YWJjOjEyMw==" -H "Content-Type:
application/x-www-form-urlencoded" -d 'grant_type=client_credentials&scope=read'
"https://api.idealista.com/oauth/token" -k
```

HTTP

```
POST /oauth/token HTTP/1.1
Host: api.idealista.com
Authorization: Basic YWJjOjEyMw==
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&scope=read
```

Note: Replace `YWJjOjEyMw==` with your encoded credentials, as explained in the Headers section above.

Property search

Description

This is the main call for property search purposes, depending on property type and other filters

- Description
- Authentication
- Filters
    - Garage specific filters
    - Premise specific filters
    - Home specific filters
    - Offices specific filters
    - Room specific filters
- Response
    - elementList
    - parkingSpace
    - detailedType
    - Errors

Api version: 3.5

Resource URL

POST https://api.idealista.com/3.5/{country}/search

Authentication

Search api calls are protected with application-only authentication (oauth2) , see documentation for details

Filters

Allowed parameters:

name	data type	description	additional info
country (required)	string	• es - idealista.com • it - idealista.it • pt - idealista.pt	`values: es, it, pt`
operation (required)	string		`values: sale, rent`
propertyType (required)	string		`values: homes, offices, premises, garages, bedrooms`
center*	string	geographic coordinates (WGS84) (latitude, longitude)	example: "40.123,-3.242"
locale	string	search language for summary	`values: es, it, pt, en, ca`
distance*	double	distance to center, in metres (ratio)	
locationId*	string	idealista location code	
maxItems	integer	items per page	50 as maximun allowed

numPage	integer	page number (for pagination)	(1,2,3..n)
maxPrice	double	maximun price in response	
minPrice	double	minimun price in response	
sinceDate	sinceDate	property age	W:last week, M: last month, T:last day (for rent except rooms), Y: last 2 days (sale and rooms)
order	string		`allowed values by property type:` • **`garages`**: `distance, price, street, photos, publicationDate, modificationDate (rent only), weigh, pricedown` • **`premises`**: `distance, price, street, photos, publicationDate, modificationDate, size, floor, ratioeurm2 (rent only), weigh, pricedown` • **`offices`**: `distance, price, street, photos, publicationDate, modificationDate, size, floor, ratioeurm2, weigh, pricedown` • **`homes`**: `distance, price, street, photos, publicationDate, modificationDate, size, floor, rooms, ratioeurm2 (sólo alquiler), weigh, pricedown` • **`rooms`**: `distance, price, street, photos, publicationDate, modificationDate, floor`
sort	string		`values: asc, desc`
adIds	Integer[]	filter by adid	multivalued field
hasMultimedia	Boolean	retrieve properties with pictures or video or virtual tour	

* you must specify a center + distance or locationId in each request

Garage specific filters

name	data type	description	additional info
bankOffer	boolean	owner is a bank	works for **sale in spain**
automaticDoor	boolean	with automatic door	
motorcycleParking	boolean	parking for motorbikes	
security	boolean	with security	

Premise specific filters

name	data type	description	additional info
minSize	double	min size	(from 60 m2 to 1000m2)
maxSize	double	max size	(from 60 m2 to 1000m2)
virtualTour	boolean	with virtual tour	
location	string		`values: shoppingcenter, street, mezzanine, underground, others`
corner	boolean		
airConditioning	boolean		
smokeVentilation	boolean		
heating	boolean		
transfer	boolean		
buildingTypes	string	String[]	`values: premises, industrialBuilding`

bankOffer	boolean	owner is a bank	works for **sale in spain**

Home specific filters

name	data type	description	additional info
minSize	double	min size	(from 60 m2 to 1000m2)
maxSize	double	maxSize	(from 60 m2 to 1000m2)
virtualTour	boolean	virtual tour	
flat	boolean	property is a flat	
penthouse	boolean		
duplex	boolean		
studio	boolean		
chalet	boolean		
countryHouse	boolean		
bedrooms	string	bedroom number (multivalued field)	0,1,2,3,4: , bedroom number separated by commas. examples: "0", "1,4", "0,3", "0,2,4". 4 means "4 or more"
bathrooms	string	bathroom number	0,1,2,3: , bedroom number separated by commas. examples: "0", "0,3", "0,2,3". 3 means "3 or more"
preservation	string	property preservation	`values: good, renew`
newDevelopment	boolean	if true, return only new development properties	
furnished	string	`furnished, furnishedKitchen`	different meanings depending on sale or rent: • sale: furnishedKitchen included in furnished
bankOffer	boolean	owner is a bank	works for **sale in spain**
garage	boolean	has garage	
terrance	boolean	has terrance	
exterior	boolean		**spain** only
elevator	boolean		
swimmingPool	boolean		
airConditioning	boolean		
storeRoom	boolean		
clotheslineSpace	boolean		
builtinWardrobes	boolean		
subTypology	String	chalet subtypology (for propertyType = homes and chalet = true)	`values: [independantHouse, semidetachedHouse, terracedHouse]`

Offices specific filters

name	data type	description	additional info
minSize	double	min size	(from 60 m2 to 1000m2)
maxSize	double	maxSize	(from 60 m2 to 1000m2)
layout	string		`values: withWalls, openPlan`
buildingType	double		`values: exclusive, mixed`
garage	boolean		

hotWater	boolean		
heating	boolean		
elevator	boolean		
airConditioning	boolean		
security	boolean		
exterior	boolean		**spain** only
bankOffer	boolean	owner is a bank	works for **sale in spain**

Room specific filters

name	data type	description	additional info
housemates	string		2,3,4: , housemates number separated by commas. examples: "2", "2,4". 4 means "4 or more"
smokePolicy	String	`allowed, disallowed`	
petsPolicy	boolean	`allowed, disallowed`	
gayPartners	boolean		
newGender	string		`values: male, female`

Response

Common response:

name	data type	description	additional info
actualPage	integer	current page	
itemsPerPage	integer	items per page	
lowerRangePosition	integer		
paginable	boolean	has paginations	
summary	string array	translated search summary	
total	integer	total items	
totalPages	integer	total pages	
upperRangePosition	integer		
elementList	items array	property listing	

elementList

name	data type	description	additional info
address	string		
bathrooms	integer	bathroom number	
country	string	es, it, pt	
distance	string	distance to center	
district	string		
exterior	boolean	is exterior	
floor	string		

hasVideo	boolean		
latitude	double		
longitude	double		
municipality	string		
neighborhood	string		
numPhotos	integer	pics number	
operation	string		`sale, rent`
price	integer		
propertyCode	integer		property id
province	string		
region	string		
rooms	integer	room number	
showAddress	boolean	owner allows to show address	
size	integer	size in m2	
subregion	string		
thumbnail	string		
url	string	idealista property url	
status	string		`newdevelopment, good, renew`
newDevelopment	boolean	is new development property?	
tenantGender	string		`values: mixed, male, female` rooms only
garageType	string		`values: carAndMotorcycle, motocycle, compactCar, sedanCar, twoCars` garages only
parkingSpace	Object	information about parking space for homes	(included in price, price...)
hasLift	boolean		homes
newDevelopmentFinished	Boolean	true if ad is from New Development, and it's finished, not returned otherwise	
isSmokingAllowed	Boolean	true if smoking is allowed, not returned otherwise	rooms
priceByArea	Double	price / size	
detailedType	Object	typology and subtypology	
externalReference	String		

parkingSpace

nombre	tipo	descripción	observaciones
hasParkingSpace	Boolean		
isParkingSpaceIncludedInPrice	Boolean		
parkingSpacePrice	Double	price if has parking space and is not included	

detailedType

nombre	tipo	descripción	observaciones
typology	String		flat, chalet, countryHouse, garage,premise,room,office
subtypology	String	flats	`duplex, penthouse, studio,`
		chalets	independantHouse, semidetachedHouse, terracedHouse, andarMoradia,
		country houses	`countryHouse, castillo, palacio, masia, cortijo, casale, casaDePueblo, casaTerrera, casaMata, torre, caseron, pazo, villa, palacete, masseria, fattoria,`
			`trullo, casalicascine, baiteChalet, quinta, moinho, monteAlentejano, solar,`
		premises	`commercialProperty, industrialPremise,`
		garages	`carAndMotorcycle, compactCar, sedanCar, motorcycle, twoCars`

Examples

https://api.idealista.com/3/es/search?locale=es&maxItems=20&numPage=1&operation=sale&order=publicationDate&propertyType=garages&sort=desc&apikey={api_key}&t=1&language=es&bankOffer=true&locationId=0-EU-ES-28

JSON response


```
{
  "actualPage":1,
  "elementList":[
     {
        "address":"isabel clara eugenia 55",
        "agency":false,
        "bathrooms":0,
        "condition":"",
        "country":"es",
        "description":"",
        "distance":"9265",
        "district":"hortaleza",
        "exterior":false,
        "favComment":"",
        "favourite":true,
  "favoriteState":favorite,
        "floor":"",
        "hasVideo":false,
        "latitude":40.4962591,
        "longitude":-3.6561735,
        "municipality":"madrid",
        "neighborhood":"sanchinarro",
        "numPhotos":5,
        "operation":"rent",
        "price":8500,
        "propertyCode":"26136403",
        "propertyType":"garaje",
        "propertyTypeCode":"G",
        "province":"madrid",
        "region":"",
        "rooms":0,
        "showAddress":true,
        "size":0,
```

```
         "subregion":"",

"thumbnail":"http://im1.idealista.com/thumbs?wi=149&he=105&en=ce5ubOrxWNmH1A3mngN0U7T8
055gZnr6FRGLOPQu%2F6HU1hx3p1hrWFhNB0%2FlDIXf&ch=-1707394652",
         "url":"http://www.idealista.com/inmueble/26136403/",
         "userType":0,
   "age":"underConstruction"
         "newDevelopment": false,
         "newProperty": false,
      }
   ],
   "itemsPerPage":20,
   "lowerRangePosition":0,
   "numPaginations":0,
   "paginable":false,
   "resultSummary": [
        "comprar",
        "garajes",
        "madrid provincia",
        "entre 3.000 y 500.000 euros",
        "con video profesional"
    ],
   "total":1,
```

```
   "totalPages":1,
   "upperRangePosition":1
}
```


https://api.idealista.com/3/es/search?locale=es&maxItems=20&numPage=1&operation=rent&order=distance&propertyType=homes&sort=asc&k={api_key}&t=13690631207690.9876595329247525&language=es&locationId=0-EU-ES-28&preservation=good&newDevelopment=true&width=140&height=105

JSON response


```
{
    "elementList": [
        {
            "propertyCode": "27258694",
            "thumbnail":
"http://im1.idealista.com/thumbs?wi=149&he=105&en=ce5ubOrxWNmH1A3mngN0U7T8055gZnr6FRGL
OPQu%2F6HU1hx3p1hrWFhNB0%2FlDIXf&ch=-1707394652",
            "numPhotos": 16,
            "floor": "4",
            "price": 525,
            "propertyType": "studio",
            "operation": "rent",
            "size": 53,
            "exterior": true,
            "rooms": 0,
            "bathrooms": 1,
            "address": "lago constanza 66",
            "province": "madrid",
            "municipality": "madrid",
            "district": "ciudad lineal",
            "neighborhood": "ventas",
            "country": "es",
            "latitude": 40.4294543,
            "longitude": -3.6460803,
            "showAddress": true,
            "url": "http://www.idealista.com/27258694",
            "agency": false,
            "favourite": false,
            "hasVideo": false,
            "status": "good",
            "age": "builtInThe70s",
            "newDevelopment": true,
            "newProperty": false
        }
    ],
    "total": 30438,
    "totalPages": 30438,
    "actualPage": 1,
    "itemsPerPage": 1,
    "numPaginations": 0,
    "summary": [
        "alquilar",
        "viviendas",
        "madrid provincia",
        "de todos los precios",
        "de todos los tamaños",
        "usada / buen estado"
    ],
    "paginable": true,
    "upperRangePosition": 1,
    "lowerRangePosition": 0
}
```

Errors

Errors are returned in http header

errorCode	message	http status (header)	descripción
400	Invalid value. Accepted values for operation are: sale, rent	400	
400	operation is required	400	
404	the locationId doesn't exist	404	
500	server error	500	

examples

JSON response

```
{
    "message": "Invalid value. Accepted values for operation are: sale, rent",
     "httpStatus": 400
}

{
    "message": "server error",
     "httpStatus": 500
}
```