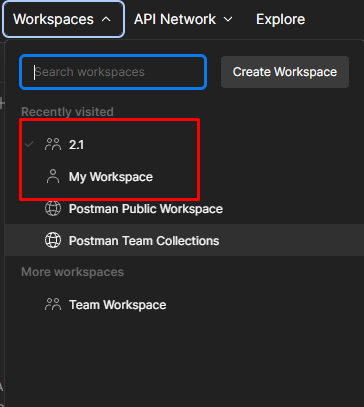
**Seccion 2.1 – Testing API’s using PostmanTool**

**2. Postman for API Testing, Introduction ,Setup & Basic HTTP Requests in Postman:**

* Instalacion
* Creamos un nuevo workspace para esta nueva sección, el workspace se llama 2.1
* Tenemos entonces 2 workspaces



* En este workspace podemos crerar collecciones, request, etc
* Para eliminar el workspace se va a las settings y delete
* Collection: donde se guardan los test cases, request, etc. Contiene un numero de folders y request http
* Se pueden crear cualquier numero de colecciones dentro de un workspace
* La colección se puede correr completa con los casos de prueba que se tienen adentro.

Request ------------API -------------Responde

* Nos encargaremos de velidar als respuestas que nos da la API
* **HTTP Request:**

GET: Recupera el recurso de la base de datos

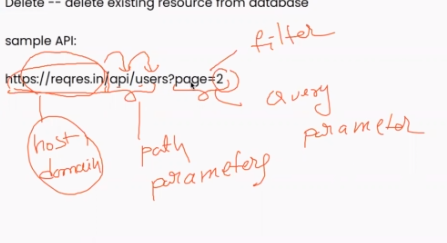
Post: Crea un nuevo recurso en la base de datos

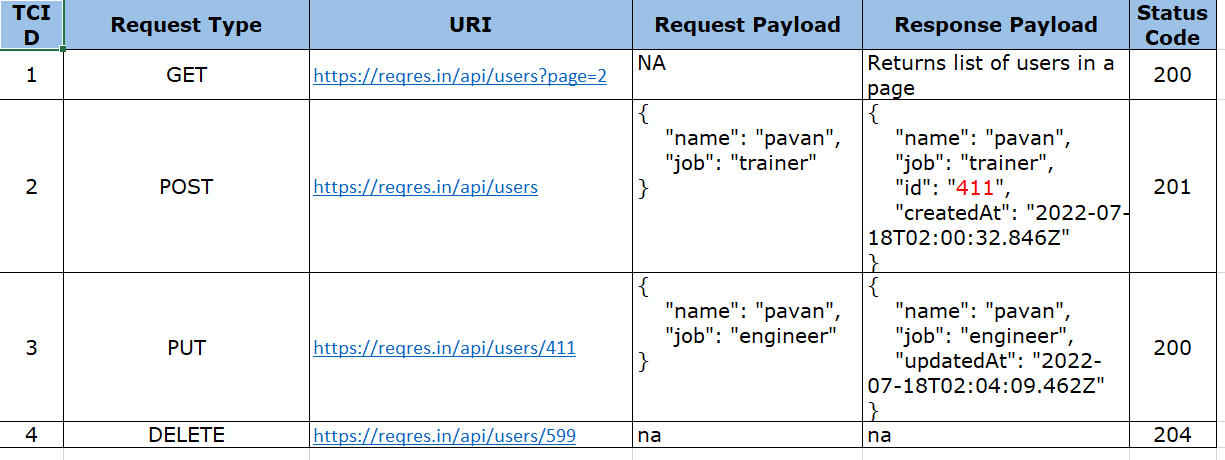
Put: Acutualiza un recurso en la base de datos

Patch: Actualiza parcialmente o detalles de un recurso

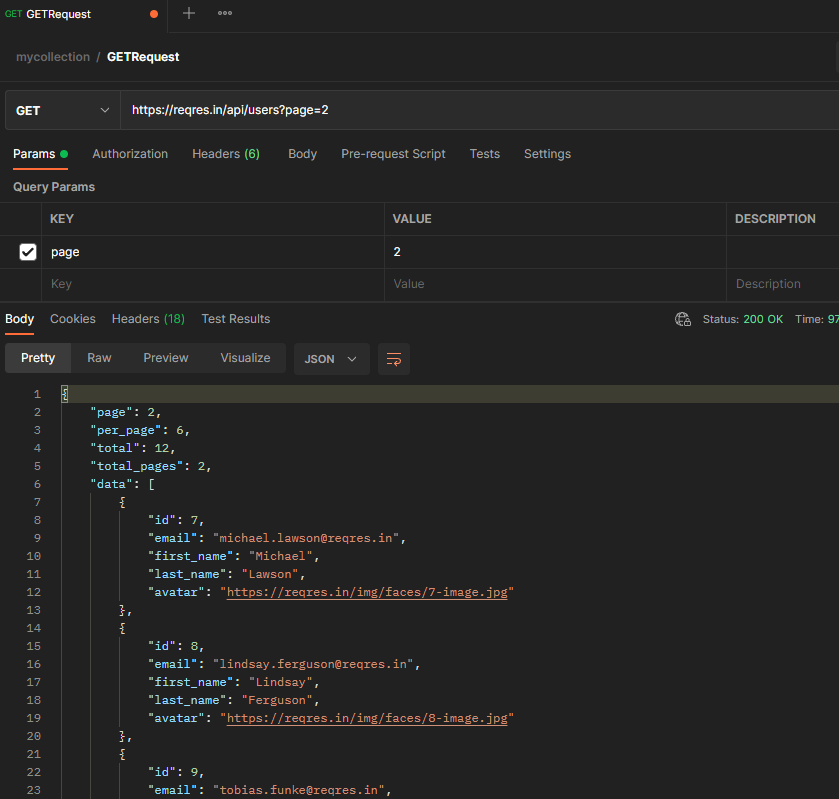
Delete: Elimina un recurso en la base de datos

* Importamos la colección de los recursos de la clase. Se usa la API Reqres
* En la documentación hay un Excel con los Test cases a usar

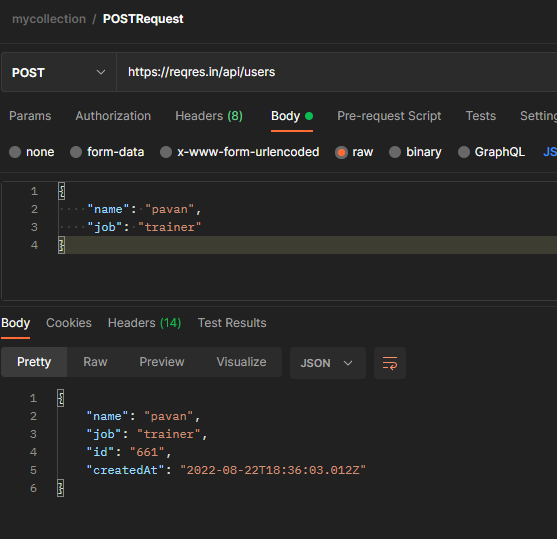




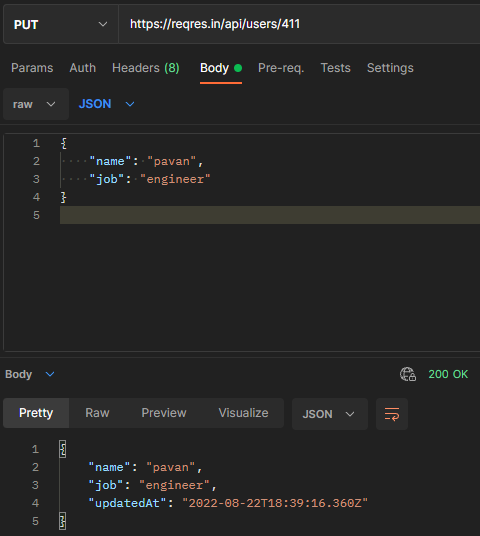
* GET:



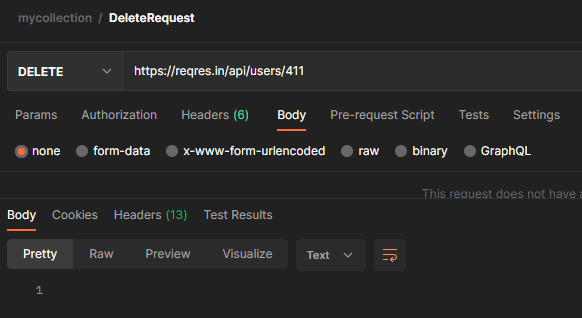
* Validaciones:
* Status code
* Time
* Size data
* Reponse body
* Cookies headers
* POST: con este se debe enviar un cuerpo con la información necesaria.



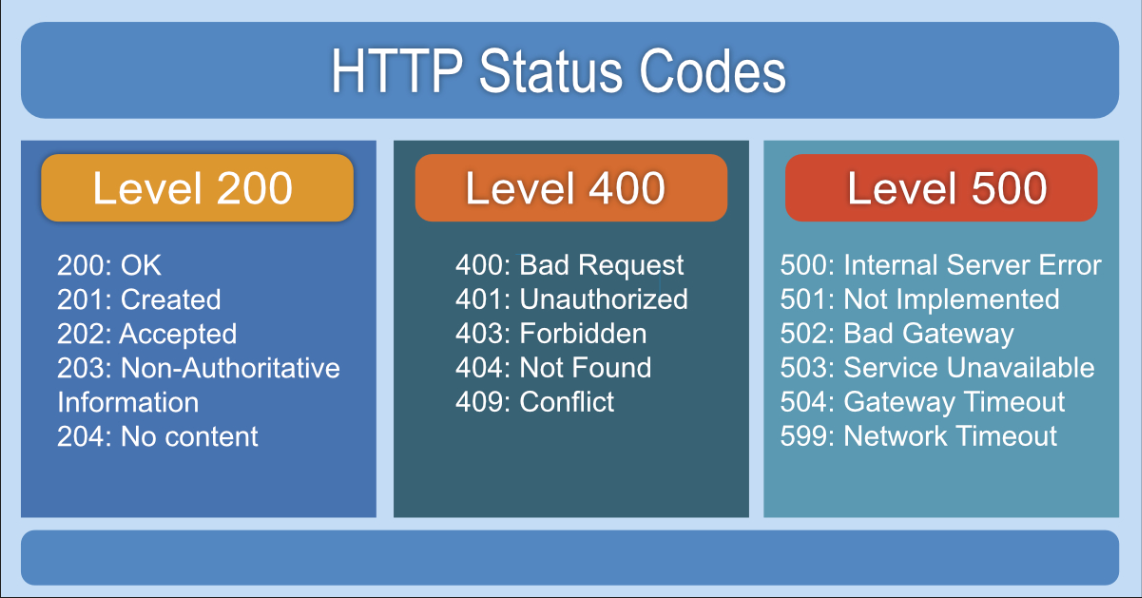
* PUT:



* Delete:



HTTP Status codes:



**3. How To Create Dummy API's, Basics of JSON & JSON Path**

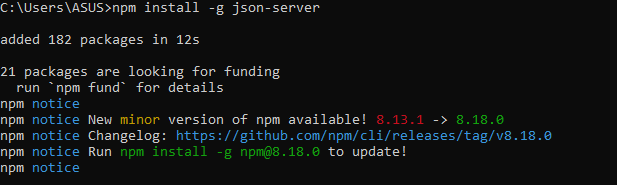
* Debemos instalar NodeJS
* Instalar npm – node package manager (viene con node js )
* Instalar JSON SERVER
* Descargar node JS de la pagina oficial
* Instalar node js. Se da next en todo
* Revisamos las variables de entorno para revisar si están las de node. Generalmente se ponen automaticamenet
* Revisamos en CMD



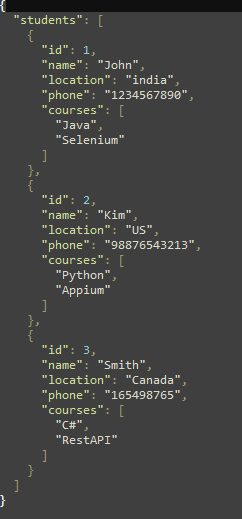


* Para instalar JSon Server se usa el comando

Npm install -g json-server



* Para crear la api se debe crear un archivo. Lo crearemos en la carpeta de la clase. Students.json . La extensión tiene que ser -json



* Ahora con este archivo se puede crear la API
* Se abre el CMD en esa carpeta y se usa el comando

Texto

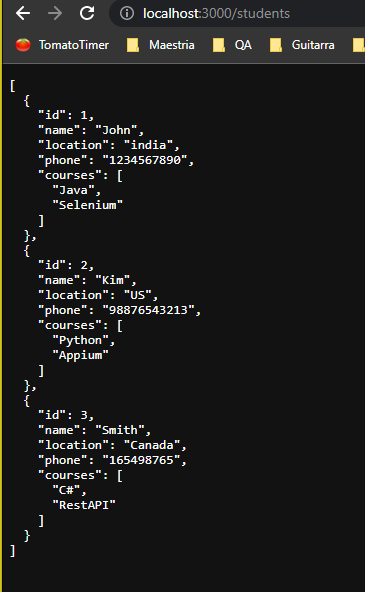
Descripción generada automáticamente

* Al correrlo se ve así. Si se cierra la ventana se cierra la API

Texto

Descripción generada automáticamente

* AL ir a la URL que sale nos muestra la API



* En el CMD no muestra que se ha pedido o ejecutado

Texto

Descripción generada automáticamente

* Desde Postman se puede acceder a esta API
* Se importa la colección de la clase con las request necesarias
* Cada que se hace un request aparece en el Cmd el request realizado junto a otra información
* Con el PUT- crear un nuevo estudiante, se actualiza el archivo .json con el nuvo registro
* JSON – Java Script Object Notation. – Es un formato de datos para la comunicación entre el cliente y el servidor. Es el usado por los servicios REST.

El Json recibe tipos de datos como:

* Number
* String
* Boolean
* Null
* Object
* Array

Los tipos de datos no se tiene que especificar, solo el Key y valor

{

“firstname”:”John”,

“SecondName”:null,

“age”:30,

“phone”:[234566,234234],

“status”: True

}

{

“Students” [

{

Sid” : 101,

“sname”:”John”

“Grade”:””A”

},

{

“Sid” : 102,

“sname”:”Kevin”

“Grade”:””B”

},

]

}

* Revisar los documentos de clase para mejor comprension y ejemplos
* Con el uso de Json path se puede acceder a alguna información en especifico dentro del Json

Por ejemplo: students[0].sname ---- > John

* Se puede hacer uso de herramientas para la búsqueda del json path

[www.Jsonpathfinder.com](http://www.Jsonpathfinder.com)

[www.Jsonpathfinder.com](http://www.Jsonpathfinder.com)

* Con el json path validamos los datos en especifico de un campo

**4. Validating Response in Postman & JavaScripting**

* **Response validations**
* Status code
* Headers
* Cookies
* Response time
* Response body
* Se deben agregar assersions que no son mas de un punto de validación
* Se usa la librería pm y funciones de javascript que ya viene en postman
* Hay 2 tipos de funciones, normales y arrow
* Se usará la colección de la clase
* Veremos primero el ccomo validar el status code

Texto

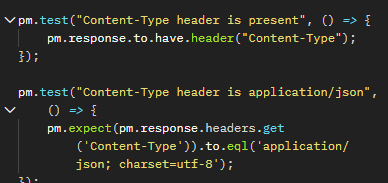
Descripción generada automáticamente

También se puede validar si el status code es uno entre varios, se usa un arreglo.

Texto

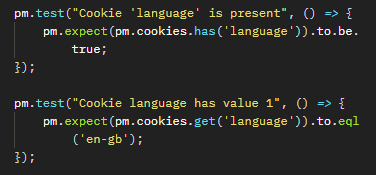
Descripción generada automáticamente con confianza media

* Los headers también se pueden chequear



Se puede verificar si un tipo de header está o también se puede verificar si un header tiene un value en especifico.

* Las cookies se puden también validar pero estas puden cambiar entre cada request

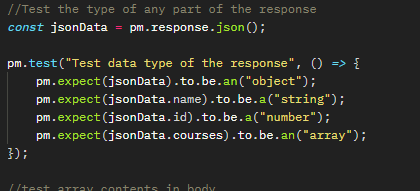


* Response time

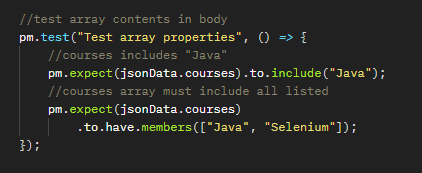
Texto

Descripción generada automáticamente

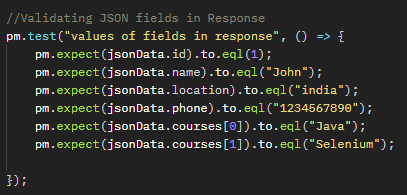
* Responde body
* En este caso se puede validar los tipos de datos que se tienen
* Primero se debe capturar la respuesta
* Luego se especifica que tipo de dato tiene cada campo, si es un numero, String, objeto, etc…



* Se puede revisar lo que un arreglo traer



* Validar los campos en la respuesta con un valor en especifico



* **JSON Schema**

Se puede ir a Google y buscar generate json Schema , ennel cual poner la respuesta y retorna el esquema. Con esto se valida como está construido el JSON



* AL colección completa se puede correr, se debe ir a la colección , click ne los 3 puntos y luego run copllection
* Esto mostrara todas las reqeust dentro de las colecciones, se puede seleccionar cuales correr.
* Se puede seleccionar el numero de iteraciones y el delay, y se muestra cada uno en la consola.

**5. Postman Scripts & Variables**

* Hay 2 tipos de scripts
  + Test scripts (clase pasada)
  + Pre request scripts
* Los test de la clase pasada se crearon bajo la pestaña de TESt
* Ahora los crearemos bajo la pestaña Pre-req Scrips. Estos se ejecutaran antes de enviar la request

Pre request scrips ------ Request ----- Response ---- Test Script

* Los pre request scrips y los test están en 3 niveles:
  + Collection
  + Folder
  + Request
* Dentro de una collecion podemos tener un folder

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* El orden de ejecución es el siguiente

Diagrama, Gráfico en cascada

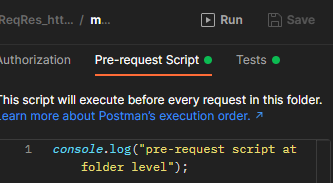
Descripción generada automáticamente

* Usamos las colecciones importadas de la clase. REqRest
* Primero revisamos la colección y sus pre req

Captura de pantalla de un celular

Descripción generada automáticamente

* Lo que se pone en el console.log se muestra en la consola como un mensaje
* Luego a nivel de foder



* Por ultimo a request level

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

* Lo mismo sucede con los test, se imprimirá un mensaje en cada nivel pero luego de realizar la request (TEst)
* Al correr la colección se muestra en la consola

Texto

Descripción generada automáticamente

* Los pre test se usan por ejemplo cuando se quiere enviar data dinámicamente. Se puede generar random y enviarla en un post
* **POSTMAN VARIABLES:**
  + Donde podemos guardar algún dato para poder reutilizarlo
* Por ejemplo se puede meter en una variable el URL para no repetirla en cada request
* Estas también se pueden crear en diferentes niveles
* **Scope:**
  + Global
  + Collection
  + Enviroment
  + Local
  + Data
* **Global:** Se pueden acceder desde cualquier parte del workspace.

Para crearla ese va a la parte derecha

Texto

Descripción generada automáticamente

Desde allí se puede añadir

Vamos a añadir el URL

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Para poderla user se debe user la sintaxis {{url\_local}}

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Esta URL se usa para todas las request de la colección

* **Collection:** Solo son accesibles dentro de la colección

Para crearla se da en la colección y en la pestaña ‘variables’

Captura de pantalla de un celular

Descripción generada automáticamente

Para acceder a esta variable se usa de la misma manera, con el {{variable\_name}}

* **Enviroment:** Se pueden tener varios ambientes, por ejemplo de DEV, QA and Prod

Se crean de la misma manera que los globales

Primero se debe crear un nuevo enviroment para poder crear las variables

Para accederlas se usan el mismo método {{}} pero se debe tener en cuenta en que ambiente se está

Accesible dentro del mismo enviroment y se debe estar apuntando a el

* **Local:** Se crean y se acceden a nivel del request

Se crean en el pre request test

Texto

Descripción generada automáticamente

Desde acá se pueden crear variables en los otros niveles.

* También se puede borrar una variable que se creo, esto se debe a hacer en el test, es decir después del request

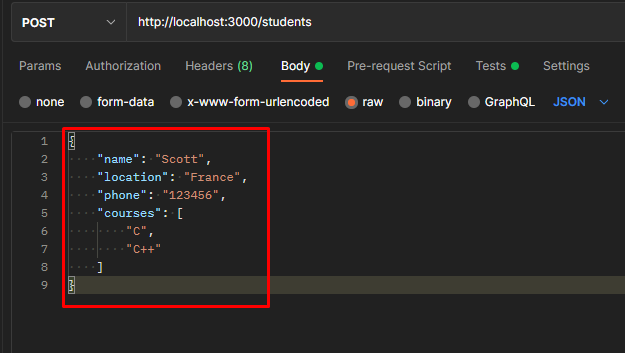
Texto

Descripción generada automáticamente

* Hay 3 metodos muy importantes:
  + Set
  + Unset
  + get

**6. Chaining Requests:**

* Esto sucede cuando de una respuesta de una request se debe usar algún valor para usarlo en otra request
* Para esto se debe guardar la respuesta en varias variables para poder usarla después, en la siguiente API
* Importamos las colecciones de la clase.
* Primero haremos un post para poder crear un nuevo estudiante y luego usamos el id para consultar
* Debemos correr la Api Studesnt
* Agregamos el body



* Y en la pestaña de TEST se agrega para poder scar la variable

Texto

Descripción generada automáticamente

* Con el JSON.parse(respondeBody), se obtiene todo el body de la respuesta y el la siguiente línea se coge lo que se necesita con el JSONPath
* Ahora en la siguiente request se usa esa variable

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

* Se corre la colección y funciona como es esperado
* Para otro ejemplo importaremos un endpoint llamado Gorest
* Para usar esta API necesitamos un token. Se consigue yendo a la pagina <https://gorest.co.in>

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Se debe tener en cuenta como es el body que se debe enviar en el POST: Esta info esta tanto en el documento de la clase como en la pagina de la API
* Se crearan datos random desde el pre request y se usarán en el body



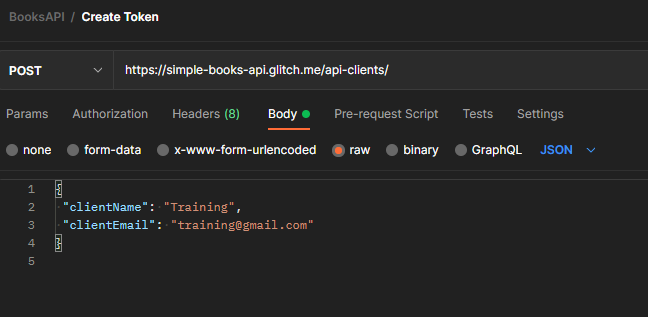
Texto

Descripción generada automáticamente

* Es importante tener en cuenta que para correr la colección completa y hacer el chaining, se debe tener algún Ambiente configurado para que las variables se puedan guardar y se puedan usar desde las variables del ambiente

**7. Data Driven Testing using JSON & CSV files**

* Se extraen los datos a usar en los request de archivos como TXT, XML, etc..
* Usaremos Books API, la importamos de los documentos de la clase
* Para algunas de las request necesitamos un token
* Para algunas se crea el token con una misma request dentro de la colección



Interfaz de usuario gráfica, Texto, Aplicación, Teams

Descripción generada automáticamente

* Este token nos servirá para algunas de las request.
* PARA el DDT - Data driven testing, podemos pasar varios datos para ejecutar las request en varias iteraciones con diferentes datos
* Haremos un post al cual le agregaremos los datos con DDTY y luego encadenaremos la respuesta para un get y un delete.
* Primero debemos preparar el formato de datos. Uaremos el archivo Excel de la clase

Tabla

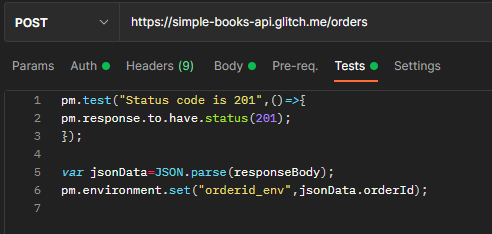
Descripción generada automáticamente

* Ahora debemos ponerlos como parte del POST.

Captura de pantalla de un celular

Descripción generada automáticamente

* Además agregamos una validación y capturamos el Id

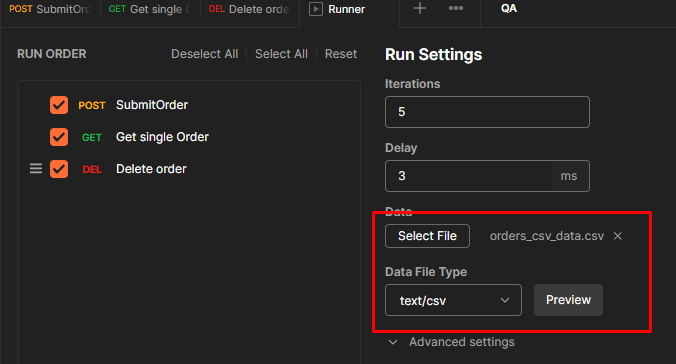


* En el GET usamos la variable capturada en la respuesta del POST

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Para poder enviart las request con el DDT, el archivo externo, debemos correr la colección completa y en esa pantalla es donde e agrega el archivo a usar.

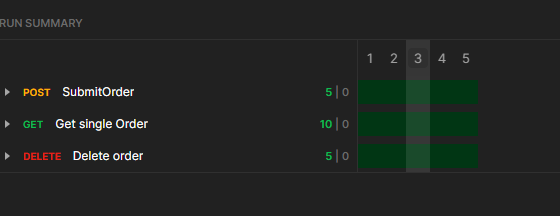


* Si se da click en preview se pueden ver los datos a usar

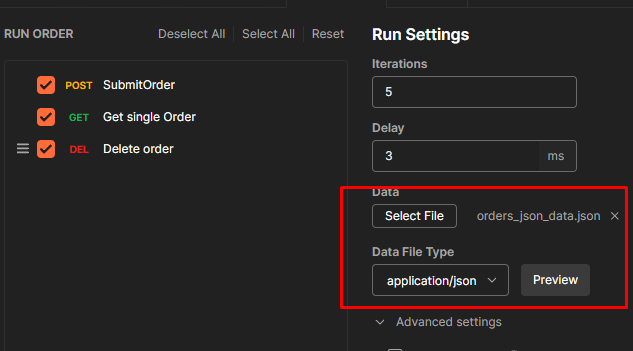
Imagen que contiene Texto

Descripción generada automáticamente

* Se da en correr la colección y se ven los resultados



* De igual manera se hace con archivos tipo JSON, se agregan de la misma manera



Interfaz de usuario gráfica, Aplicación

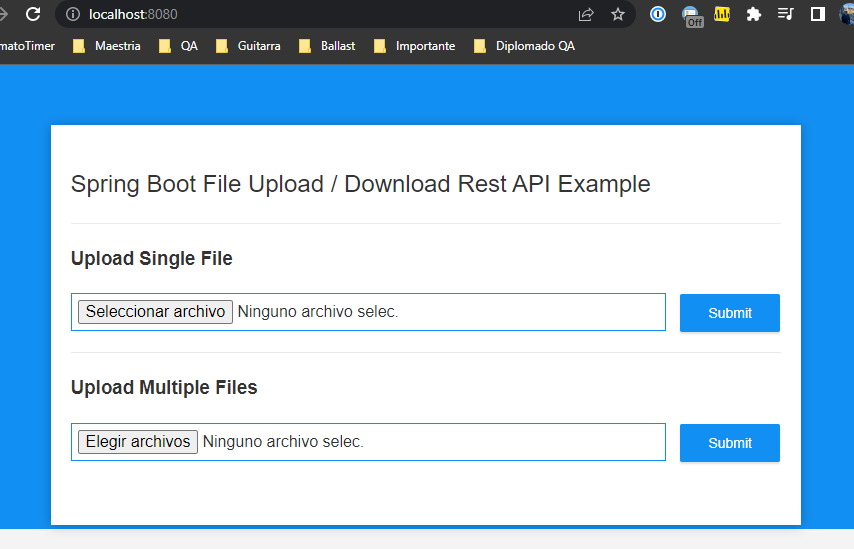
Descripción generada automáticamente

**8. File Upload & Download API, Authorization Types, cURL**

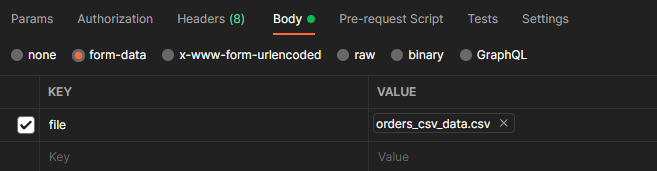
* Para poder subir un documento, vamos a correr el archivo -jar que esta en la documentación de la clase, para esto vamos a donde lo tenemos guardado, abrimos consola y corremos el siguiente comando



* Se debe tener en cuenta el puerto en que se va a correr en esta caso el 8080 y ver que no este corriendo nada mas en este. Si hay algo mas hay que matar ese proceso desde cmd como admin



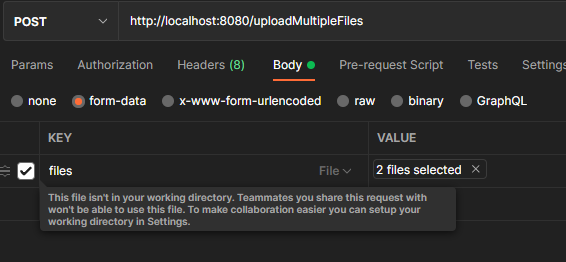
* Vamos entonces enviar un POST, en esta API se pueden enviar un solo archivo o enviar múltiples archivos
* Para poder enviar un archivo se va a el body, form-data y se agrega un key llamada ‘File’, a continuación se selecciona el file que se quiere



Captura de pantalla de un celular

Descripción generada automáticamente

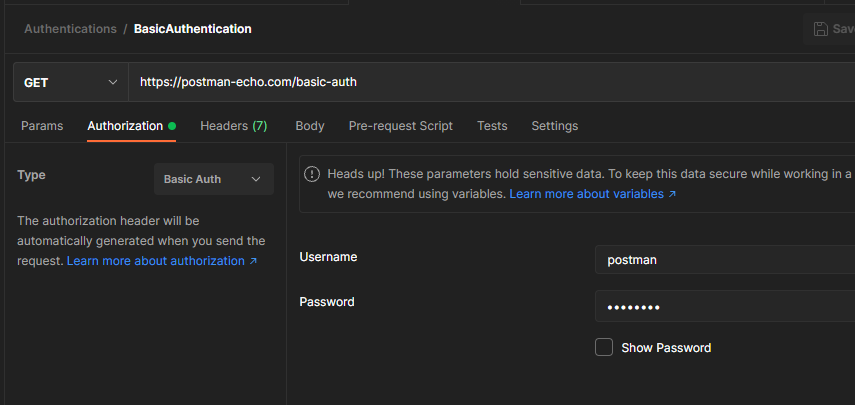
* Para enviar varios archivos se usa el otro API
* EN el body key se selecciona ‘files’ y se suben los necesarios

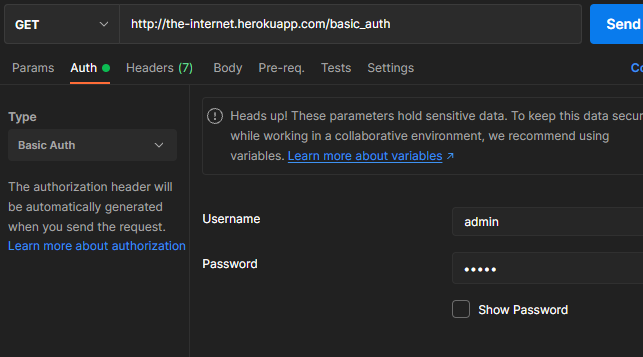


Texto

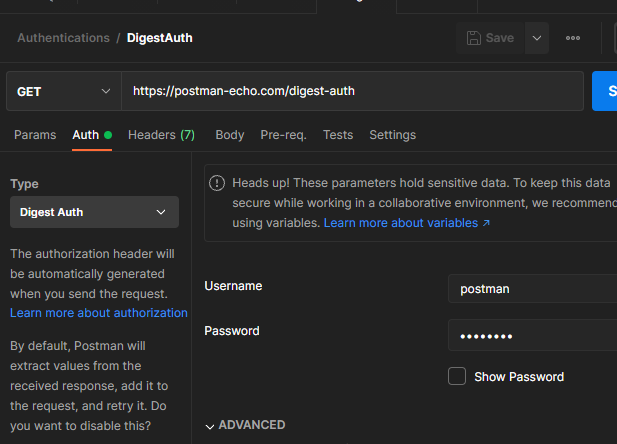
Descripción generada automáticamente

* **Authentication Types**
* Hay muchos tipos de autenticación y esto depende de como esté construida la Api.
* En basic authenticacion and digest authentication, en ambos se necesita un usuario y una contraseña. La única diferencia es que una es mas segura que la otra
* Para las autenticaciones se usará la colección de la clase
* Basic Authentication: usuario y contraseña

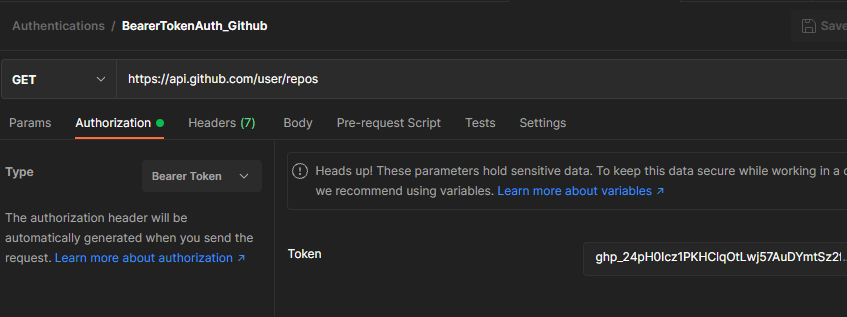




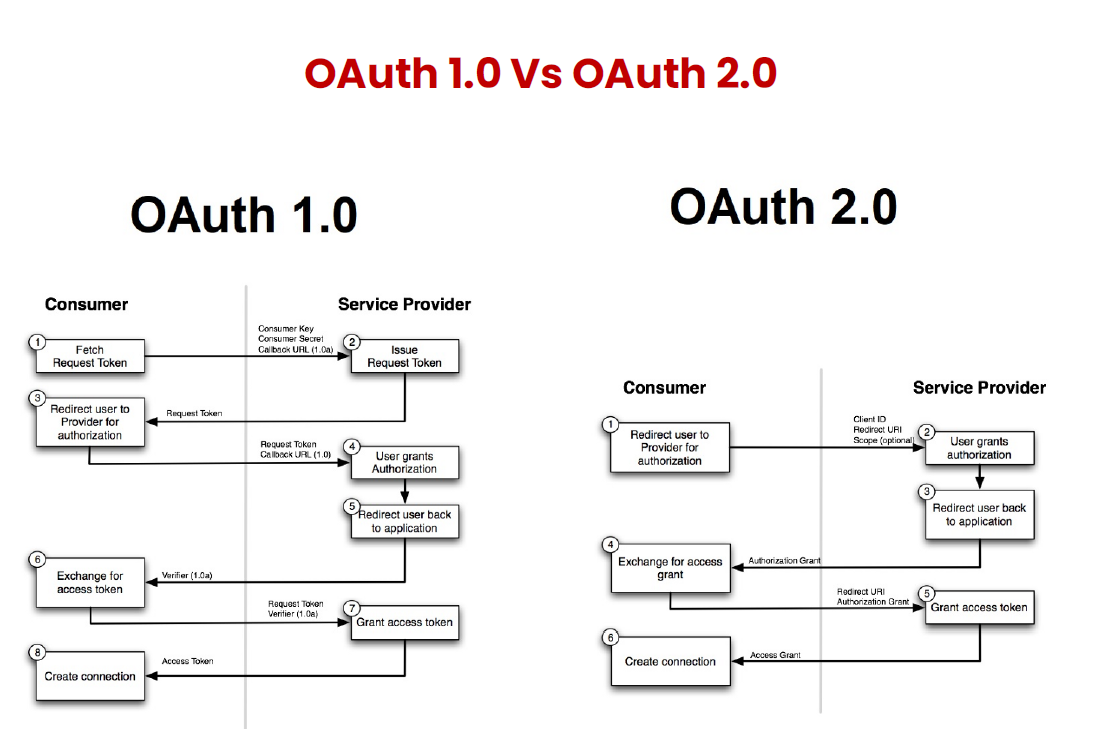
* Digest Authentication: el usuario y el password se encriptan



* En este caso el bearer token se genera en github y trae toda la información de los repositorios del tutor



* Para la última autenticación debemos crear una api key openweathermap
* La key se activa en unas horas



* Todo el proceso de OAuth2 está en la collection y en la documentación.
* **Swagger:**  Es documentación interactiva

En el siguiente link se puede ver una documentación de swagger

<https://fakerestapi.azurewebsites.net>

swagger se hace para explorar la Api porque es documentación, pero no se usa con propósitos de Testing

* **cURL:** Esto contiene el request además de los headers, autenticaciones, etc. Todo lo que se requiere está en ese curl

Este cURL se puede enviar desde el CMD

Se puede crear una request desde el curl en postman. Se da en importa , se va a la pestaña ‘raw text’ y luego se importa el cURL

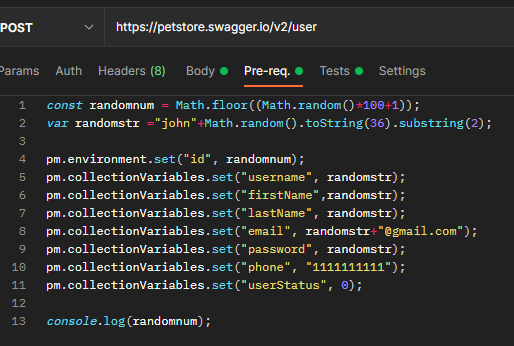
* En postman también se puede crear documentación, esto se usa generalmente luego de realizar las pruebas. Para hacerlo se da en los 3 puntos de la colección y luego en documentación
* Se muestran todas las request de la colección y se pueden agregar comentarios
* En la parte superior derecha se pyuede clickear en publish y se obtiene un URL que lleva a esta documentación

**9. Validate XML Response, Converting XML Response to JSON & Documenting &Publishing:**

* Usaremos la APi de PETstore, la cuel permite respuestas tanto en XML como en JSON

[**https://petstore.swagger.io**](https://petstore.swagger.io)

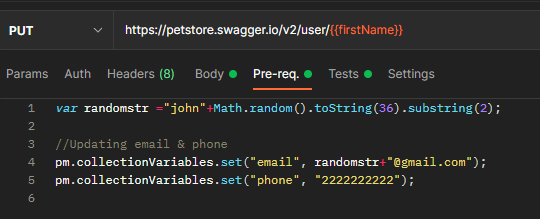
* Del swagger podemos obtener el Curl y de allí la request para postman
* Primero importaremos la colección del usuario y la uasremos
* En el post inicial tendremos los datos dinámicamente



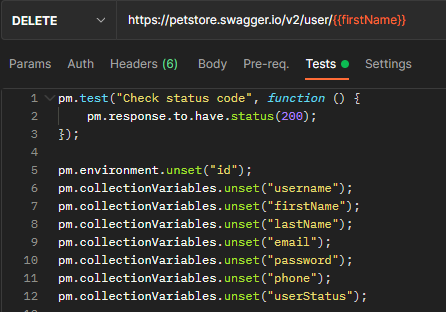
* En el GET



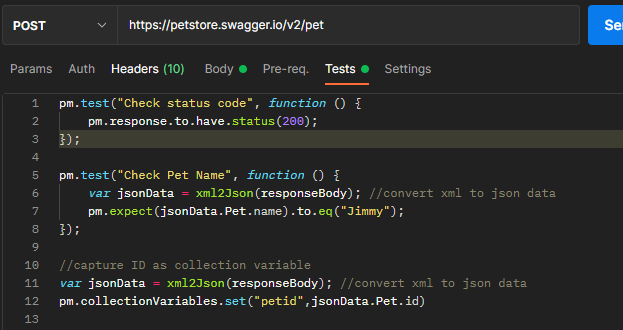
* PUT



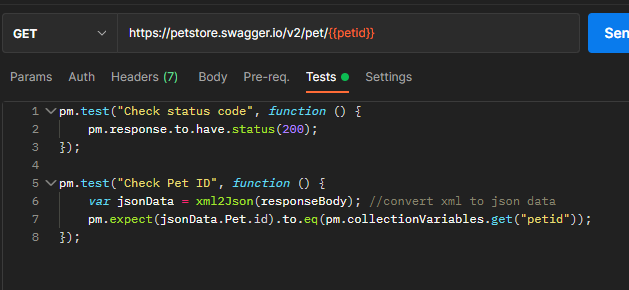
* DELETE



* Corremos la colección completa y luego vamos al lado superior derecho y exportamos los resultados como un json
* Se puede crear también la documentación
* Desde la documentación se puede correr la documentación, lo que importa la colección al local de postman para usarlo. No tiene cambios respecto a lo que se creo en la documentación
* Usamos la otra API, la de pet
* En esta desde swagger configuramos para que la respuesta venga en xml
* Las validaciones no se pueden hacer directamente en XML por lo que se convierten a JSON
* POST



* GET



* PUT

Captura de pantalla de un celular

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

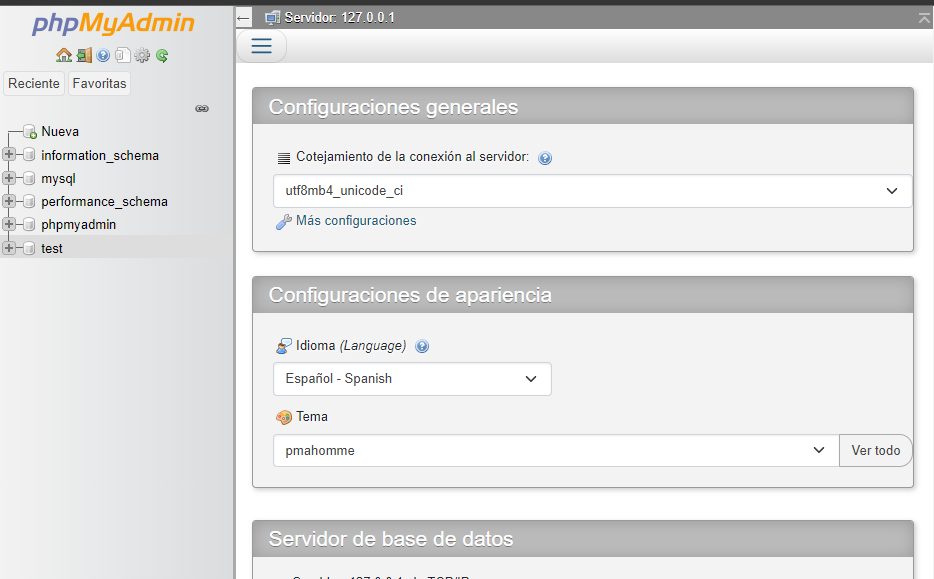
* DELETE

Interfaz de usuario gráfica

Descripción generada automáticamente

* **E-commerce Application – OpenCart**
* Primero vamos a crear la api / cuenta
* Debemos descargar la aplicación, instalarla en nuestro sistema
* Los pasos se pueden revisar en la documentación
* Debemos instalar también XAMPP, esto traerá apache, mysql y php por defecto
* Corremos XAMPP
* Open cart foplder se debe poner en htdocs dentro del XAMPP en disco C
* Luego de que este copiado se va al folder ‘upload’ de opencart
* Hay un archivo de nombre config-dist.php.. se le debe cambiar el nombre solo a config.php
* Se va a la carpeta de admin y se hace el mismo cambio
* Ahora se debe crear la DB.
* Primero se deben correr los servicios en XAMPP y luego acceder a la url (apache and mysql

<http://localhost/phpmyadmin/>



* Ahora debenmos crear una DB, vamos a la sección de DB

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

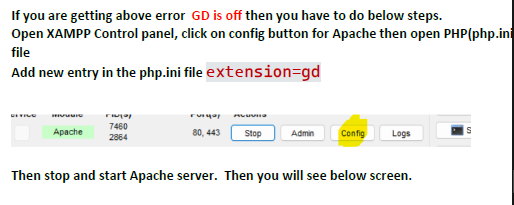
* Luego de crear la DB se va a la URL

[**http://localhost/opencart/**](http://localhost/opencart/)

* Luego aparece una pantalla con todo lo requerido, si algo está en rojo es que falta

Tabla

Descripción generada automáticamente



Tabla

Descripción generada automáticamente

* Click en continuar
* Se rellenan los datos encesatios y luego se va a

C:\xampp\htdocs\opencart\upload

* Texto

  Descripción generada automáticamenteSe bora el folder de instalación

}

* Para abrir la aplicación en el sistema local usamos:

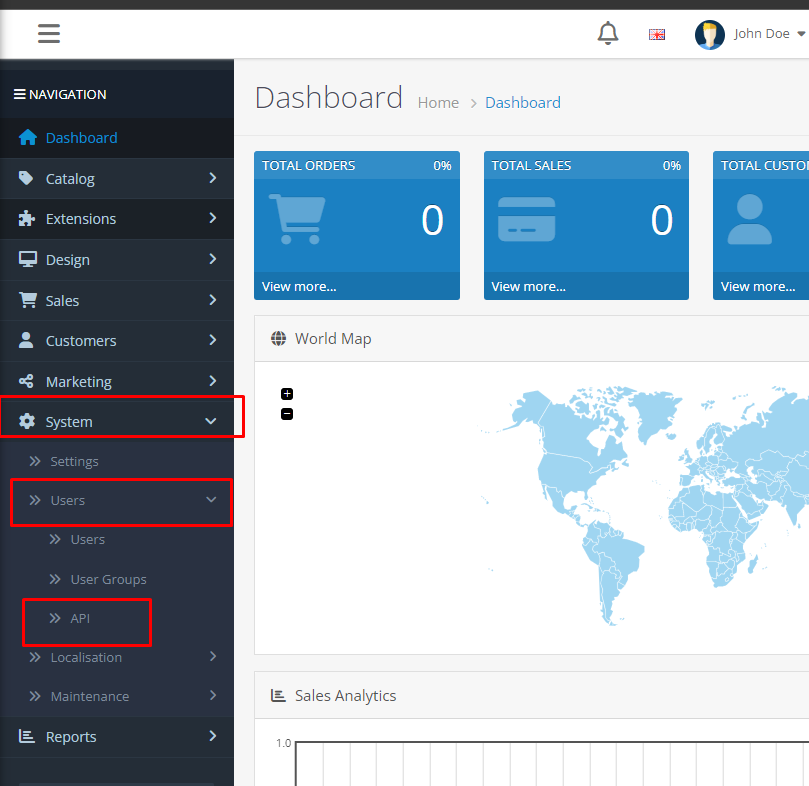
<http://localhost/opencart/>

* Para ir al back/admin, se va a :

<http://localhost/opencart/admin/>

credenciales: admin/admin

* La API se encuentra en System/ users / API



* Para crear el AP Iuser, se va a esta pestaña.
* Clcik en el + a la derecha

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Luego de crearla hay que ponerla en enable
* Luego ir a la pestaña IP address y añadir nuestra ip.
* La podemos tomar de cmd

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

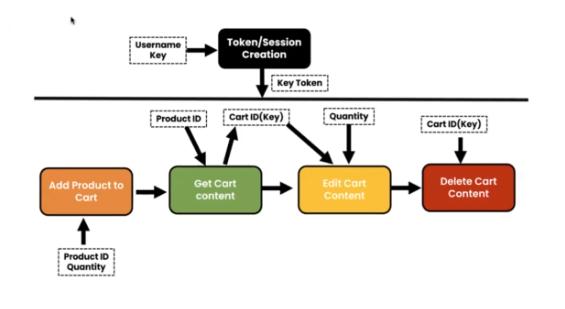
Descripción generada automáticamente

* Ahora se puede acceder a esta API desde nuestro sistema local
* Importante tener el URL de la documentación de opencart

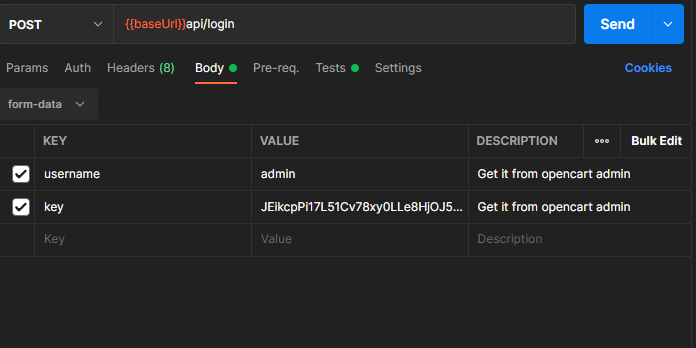
<https://docs.opencart.com/en-gb/system/users/api/>

**10. Real Time e-Commerce App API's, Sharing Collections, Reporting , CI Jenkins:**

* Usaremos lo creado en la clase pasada
* En los documentos de la clase se tiene la documentación de la Apio o se puede acceder a la documentación directamente desde opencart
* Para crear una sesión necesitamos el key y el usuario que podemos obtener desde el admin panel
* Se usarán varias validaciones y variables



* Create sesión/Token - POST

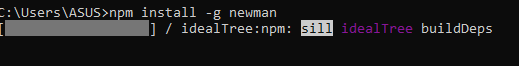


* Tener en cuenta que se esta recibiendo un HTML como respuesta y no se puede sacar el campo
* Se siguen los pasos de las request.
* **Maneras de correr una colección**

1. Localmente
2. Newman (se debe instalar el componente junto con nodejs)

* Para instalar Newman

Usamos el comando npm install -g Newman

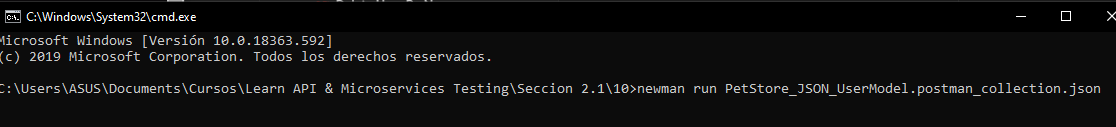


Se debe exportar la colección.

En este caso usaremos la del petstore

**Para correr al colección se tienen varias formas**

1. Se va a la locación y se corre el cmd



Texto

Descripción generada automáticamente

Podemos generar un reporte hmtl pero debemos instalar algo nuevo

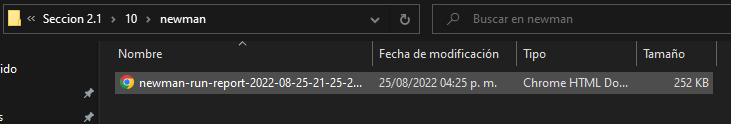
Npm install -g Newman-reporter-hmtl



Para correrla se usa el mismo copmando anterior pero al final -r html



El reporte se genera en la misma carpeta donde está la colección



Al abrirlo lleva al reporte

1. La segunda forma es mediante el URL

Para crear una URL para la colección se va a los 3 puntos y se da en share

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se muestran varias opciones, en la primera se pueden poner los correr o se les envía.

La mejor opción es mediante el JSON Link

Texto

Descripción generada automáticamente

Se da en get link y se puede compartir

Para ejecutarlo se hace desde el CMD pero copiando el link



Este reporte fue creado por alguien mas que corrió la colección remotamente

* Otra manera de correrlo es mediante Jenkins
* Jenkins es una herramienta de CI
* Descargamos e instalamos Jenkins
* Instalamos y en la carpeta de Jenkins corremos el comando

Java -jar jenking.war

El CMD nos genera una contarseña que luego debemos pegar cuando vamos a localhost/8080 (en este puerto se puso Jenkins)

D0267956c78b47398b873cf6a6cc1768

* Se empieza la instalación y configuración
* Se crea un nuevo item, en este caso la colección que correremos
* Luego se va a build y se selecciona execute comand batch
* Luego se pone el mismo comando para correr en el cmd.
* Luego se da en run now y se corre la colección añadida
* Se puede ver el resultado