

PRÁCTICA REST

Previo

Nuestra aplicación corre en el puerto 6789

Los datos de servidor, base de datos, username, password y conexión son los siguientes:

```
server = '127.0.0.1,6789'
database = 'ssdd'
username = 'sa'
password = 'icaiSQL2019'
cnxn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server}; \
                      SERVER='+server+';DATABASE='+database+'; \
                      UID='+username+';PWD='+password)
```

Consultar la cantidad de fechas disponibles en la base de datos

URL→ <http://127.0.0.1:6789/api/v1.0/data/cantidad-fechas>

Resultado:

	fecha
0	0001-01-01
1	0002-02-02
2	2000-05-24
3	2019-01-06
4	2019-02-06
5	2019-03-06
6	2019-04-06
7	2019-05-06
8	2019-06-06
9	2019-07-06
10	2019-08-06
11	2019-09-06
12	2019-10-06
13	2019-11-06
14	2019-12-06
15	2020-01-06
16	2020-02-06
17	2020-03-06
18	2020-04-06
19	2020-05-06
20	2020-06-06
21	2020-07-06
22	2020-08-06
23	2020-09-06
24	2020-10-06
25	2020-11-06
26	2020-12-06
27	2021-01-06
28	2021-02-06
29	2021-07-05

Consultar los códigos de las estaciones de origen

URL → <http://127.0.0.1:6789/api/v1.0/data/estaciones-origen>

Resultado:

	idunplug_base
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	20
20	21
21	22
22	23
23	24
24	25
25	26
26	27
27	28
28	29
29	30

Consultar los códigos de las estaciones de destino

URL → <http://127.0.0.1:6789/api/v1.0/data/estaciones-destino>

	idplug_base		
0	1	10	11
1	2	11	12
2	3	12	13
3	4	13	14
4	5	14	15
5	6	15	16
6	7	16	17
7	8	17	18
8	9	18	19
9	10	19	20
		20	21
		21	22
		22	23
		23	24
		24	25
		25	26
		26	27
		27	28
		28	29
		29	30

Consultar los movimientos para un determinado día

Mostramos un ejemplo de una URL para que sea más sencillo ver la ejecución

URL → <http://127.0.0.1:6789/api/v1.0/data/movimientos-dia/2021-07-05>

Resultado:

Mostramos únicamente unos cuantos movimientos de todos los que muestra:

	fecha	ageRange	user_type	idunplug_station	idplug_station	idunplug_base	idplug_base	travel_time	Fichero
0	2021-07-05	0	1	90	66	8	21	219	201906
1	2021-07-05	4	1	71	136	19	19	359	201906
2	2021-07-05	4	1	39	38	7	17	375	201906
3	2021-07-05	5	1	66	90	21	4	264	201906
4	2021-07-05	4	1	152	166	13	3	367	201906
5	2021-07-05	5	1	55	53	4	14	174	201906
6	2021-07-05	0	1	133	129	6	2	308	201906
7	2021-07-05	4	1	153	169	3	21	462	201906

Consultar todos los movimientos para un determinado día que hagan un trayecto iniciado en una determinada estación origen

Mostramos un ejemplo de una URL para que sea más sencillo ver la ejecución

URL → <http://127.0.0.1:6789/api/v1.0/data/movimientos-origen/2019-01-06&90>

Resultado:

Mostramos únicamente unos cuantos movimientos de todos los que muestra:

	fecha	ageRange	user_type	idunplug_station	idplug_station	idunplug_base	idplug_base	travel_time	Fichero
0	2019-01-06	4	1	90	83	10	7	291	201906
1	2019-01-06	0	1	90	75	12	21	251	201906
2	2019-01-06	4	1	90	114	20	4	327	201906
3	2019-01-06	5	2	90	124	4	18	2431	201906
4	2019-01-06	0	1	90	135	24	27	950	201906
5	2019-01-06	0	1	90	91	2	5	478	201906
6	2019-01-06	0	1	90	83	20	20	300	201906
7	2019-01-06	0	1	90	90	6	20	741	201906

Consultar todos los movimientos para un determinado día que hagan un trayecto finalizado en una determinada estación de destino

Mostramos un ejemplo de una URL para que sea más sencillo ver la ejecución

URL → <http://127.0.0.1:6789/api/v1.0/data/movimientos-destino/2019-01-06&90>

Resultado:

Mostramos únicamente unos cuantos movimientos de todos los que muestra:

	fecha	ageRange	user_type	idunplug_station	idplug_station	idunplug_base	idplug_base	travel_time	Fichero
0	2019-01-06	0	1	79	90	8	7	308	201906
1	2019-01-06	0	1	97	90	24	23	311	201906
2	2019-01-06	4	1	113	90	21	20	246	201906
3	2019-01-06	4	1	9	90	12	10	425	201906
4	2019-01-06	0	1	9	90	22	4	563	201906
5	2019-01-06	0	1	90	90	6	20	741	201906
6	2019-01-06	0	1	90	90	14	14	5601	201906
7	2019-01-06	0	1	91	90	1	26	380	201906

Consultar todos los movimientos para un determinado día que hagan un trayecto iniciado en una determinada estación origen y finalizado en una determinada estación de destino

Mostramos un ejemplo de una URL para que sea más sencillo ver la ejecución

URL → <http://127.0.0.1:6789/api/v1.0/data/movimientos-inicio-destino/2019-01-06&90&83>

Resultado:

	fecha	ageRange	user_type	idunplug_station	idplug_station	idunplug_base	idplug_base	travel_time	Fichero
0	2019-01-06	4	1	90	83	10	7	291	201906
1	2019-01-06	0	1	90	83	20	20	300	201906
2	2019-01-06	0	1	90	83	7	14	620	201906
3	2019-01-06	4	1	90	83	19	11	370	201906
4	2019-01-06	0	1	90	83	18	5	646	201906
5	2019-01-06	0	1	90	83	5	15	426	201906
6	2019-01-06	0	1	90	83	23	1	560	201906

Consultar todos los movimientos para un determinado día que hagan un trayecto iniciado en una determinada estación origen y finalizado en una determinada estación de destino y con una duración de trayecto inferior a un determinado valor

Mostramos un ejemplo de una URL para que sea más sencillo ver la ejecución

URL → <http://127.0.0.1:6789/api/v1.0/data/movimientos-inicio-destino-tiempo/2019-01-06&90&83&400>

Resultado:

	fecha	ageRange	user_type	idunplug_station	idplug_station	idunplug_base	idplug_base	travel_time	Fichero
0	2019-01-06	4	1	90	83	10	7	291	201906
1	2019-01-06	0	1	90	83	20	20	300	201906
2	2019-01-06	4	1	90	83	19	11	370	201906

Introducir una nueva línea a la base de datos con todos los campos rellenos y el valor de fichero = "00000"

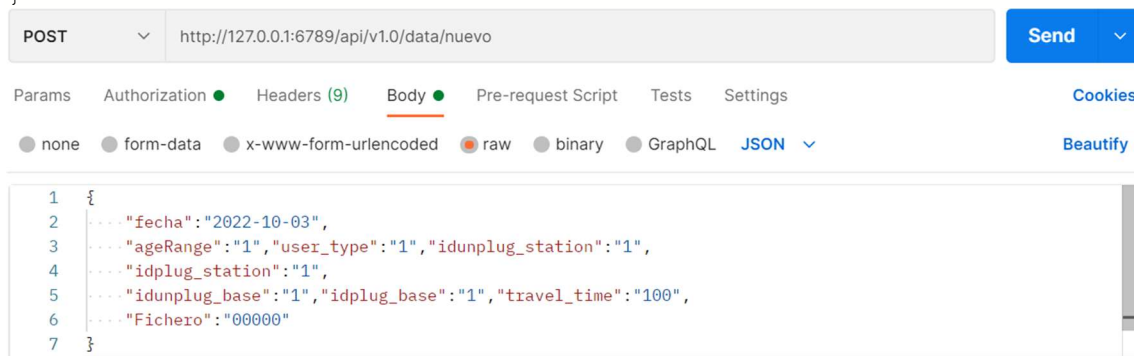
Este servicio en añadido debe estar validado, es decir, que solo aquellas personas que introduzcan usuario y contraseña y estos estén recogidos como válidos podrán ejecutar la acción que desean.

Hacemos uso de postman.

URL → <http://127.0.0.1:6789/api/v1.0/data/nuevo>

En el body ponemos los siguientes datos:

```
{
  "fecha": "2022-10-03",
  "ageRange": "1", "user_type": "1", "idunplug_station": "1",
  "idplug_station": "1",
  "idunplug_base": "1", "idplug_base": "1", "travel_time": "100",
  "Fichero": "00000"
}
```



Jaime Arana y Javier Álvarez

Y falta añadir la validación del usuario y la contraseña. En Authorization, Basic auth ponemos:

Username → javi

Password → 123456789

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:6789/api/v1.0/data/nuevo`. The 'Authorization' tab is selected, showing 'Basic Auth' type. The 'Username' field contains 'javi' and the 'Password' field contains '123456789'. A 'Send' button is visible in the top right.

Para comprobar que se ha ejecutado correctamente podemos usar la siguiente URL:

URL → <http://127.0.0.1:6789/api/v1.0/data/verintroducida/2022-10-03>

Resultado :

	fecha	ageRange	user_type	idunplug_station	idplug_station	idunplug_base	idplug_base	travel_time	Fichero
0	2022-10-03	1	1	1	1	1	1	100	0

Implementar una función de tipo PUT que permita actualizar el campo de travel time para una determinada entrada con valores fecha, ageRange, user_type, idunplug_station, idplug_station, idunplug_base, idplug_base

Vamos a actualizar la fila introducida previamente y además para ejecutar este servicio también será necesario validarse previamente:

La validación es como anteriormente:

Añadir la validación del usuario y la contraseña. En Authorization, Basic auth ponemos:

Username → javi

Password → 123456789

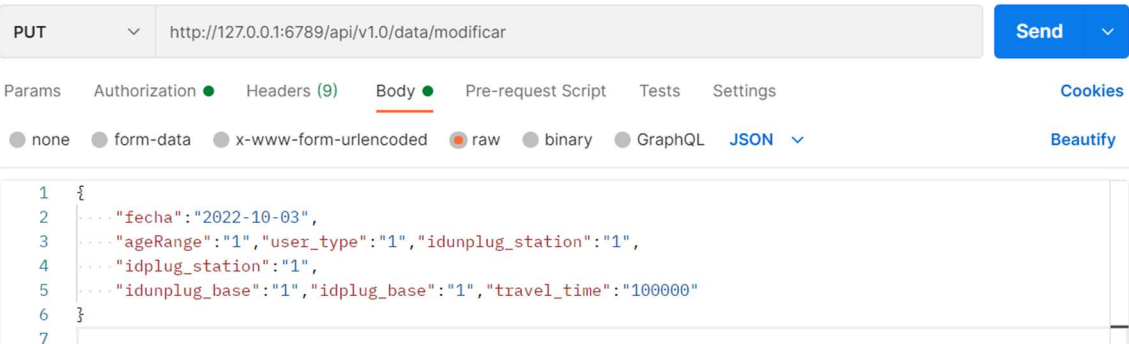
The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:6789/api/v1.0/data/modificar`. The 'Authorization' tab is selected, showing 'Basic Auth' type. The 'Username' field contains 'javi' and the 'Password' field contains '123456789'. A 'Show Password' checkbox is checked. A 'Send' button is visible in the top right.

La URL para el postman es:

URL → <http://127.0.0.1:6789/api/v1.0/data/modificar>

En el body ponemos los siguientes datos:

```
{
  "fecha": "2022-10-03",
  "ageRange": "1", "user_type": "1", "idunplug_station": "1",
  "idplug_station": "1",
  "idunplug_base": "1", "idplug_base": "1", "travel_time": "100000"
}
```



Para comprobar que se ha ejecutado correctamente podemos usar la siguiente URL:

URL → <http://127.0.0.1:6789/api/v1.0/data/veractualizada/2022-10-03>

Resultado :

	fecha	ageRange	user_type	idunplug_station	idplug_station	idunplug_base	idplug_base	travel_time	Fichero
0	2022-10-03	1	1	1	1	1	1	100000	0

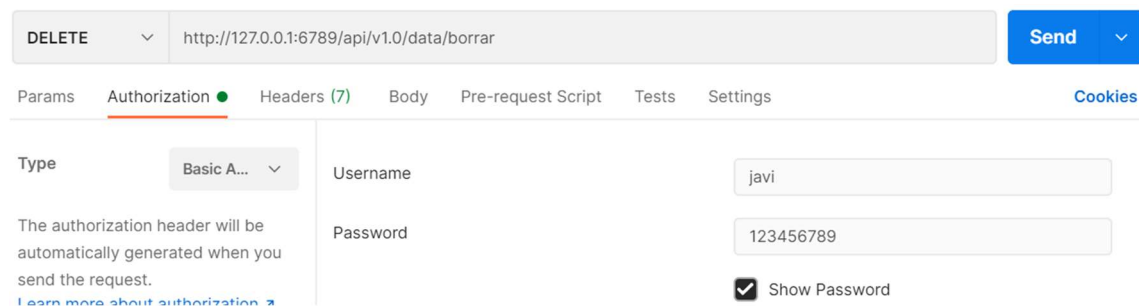
Implementar una función de tipo delete que permita eliminar líneas de la base de datos pasando una combinación de los parámetros fecha con cualquiera del resto de parámetros

Vamos a eliminar la línea introducida previamente pasando la fecha y la estación base como parámetro. Para poder ejecutar este servicio también es necesario realizar la validación, como hemos explicado previamente la manera es la siguiente:

Añadir la validación del usuario y la contraseña. En Authorization, Basic auth ponemos:

Username → javi

Password → 123456789



DELETE ▼ http://127.0.0.1:6789/api/v1.0/data/borrar Send ▼

Params **Authorization** ● Headers (7) Body Pre-request Script Tests Settings Cookies

Type Basic A... ▼

The authorization header will be automatically generated when you send the request.
[Learn more about authorization](#) ▼

Username

Password

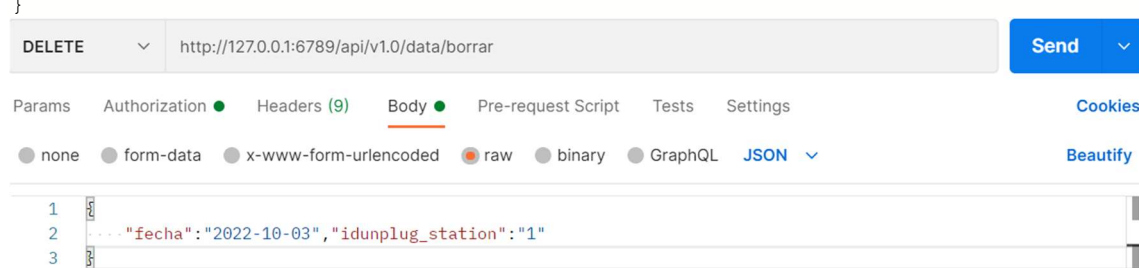
☒ Show Password

La URL para el postman es:

URL → <http://127.0.0.1:6789/api/v1.0/data/borrar>

Mostramos un ejemplo para el body:

```
{
  "fecha": "2022-10-03", "idunplug_station": "1"
}
```



DELETE ▼ http://127.0.0.1:6789/api/v1.0/data/borrar Send ▼

Params **Authorization** ● Headers (9) **Body** ● Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON ▼ Beautify

```
1 {
2   "fecha": "2022-10-03", "idunplug_station": "1"
3 }
```

Para ver que funciona correctamente como ya hemos mosrado previamente que podíamos ver la línea para la fecha 2022-10-03 si volvemos a llamar a la URL anterior no aparecerá:

URL → <http://127.0.0.1:6789/api/v1.0/data/movimientos-dia/2022-10-03>

fecha	ageRange	user_type	idunplug_station	idplug_station	idunplug_base	idplug_base	travel_time	Fichero
-------	----------	-----------	------------------	----------------	---------------	-------------	-------------	---------

Como vemos aparece vacío por lo que no está en nuestra base de datos