

SD PRACTICA

Adil Akhazzan

Contenido

Introducción: 2

Arquitectura Conceptual 2

Arquitectura Técnica: 3

Casos de Usos..... 3

Rutas..... 4

Servicios: 7

GUIA DE DESPLIEGUE 9

Aspecto Relevantes 12

Introducción:

Para empezar esta guía debemos de indicar los programas que vamos utilizar estos serán:

Postman: Para simular nuestro front-end y poder ir probando nuestras apis

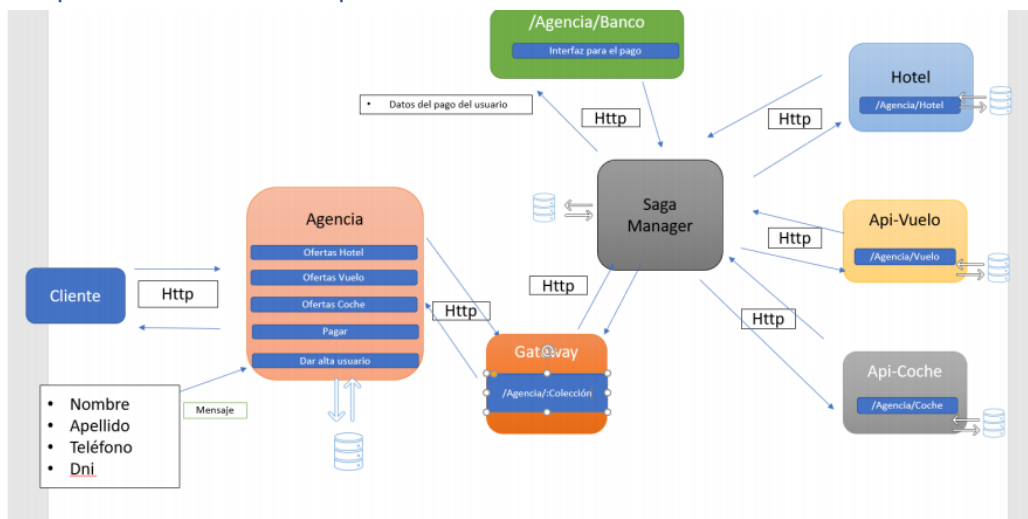
Node : Sera el entorno JavaScript que emplearemos para el desarrollo de nuestra practica.

Express: Esta biblioteca nos facilitare la gestión de los métodos y recursos HTTP

MongoDB: Sera nuestra base de datos

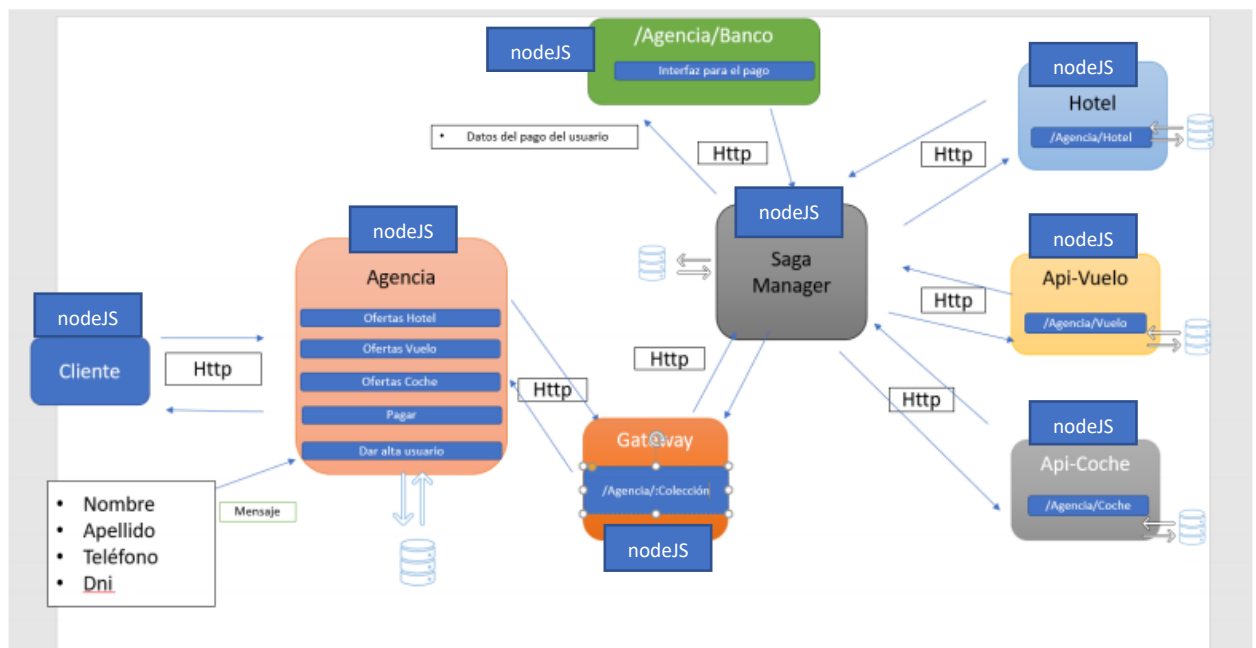
Visual Code : Sera nuestra herramienta para programar nuestro código.

Arquitectura Conceptual

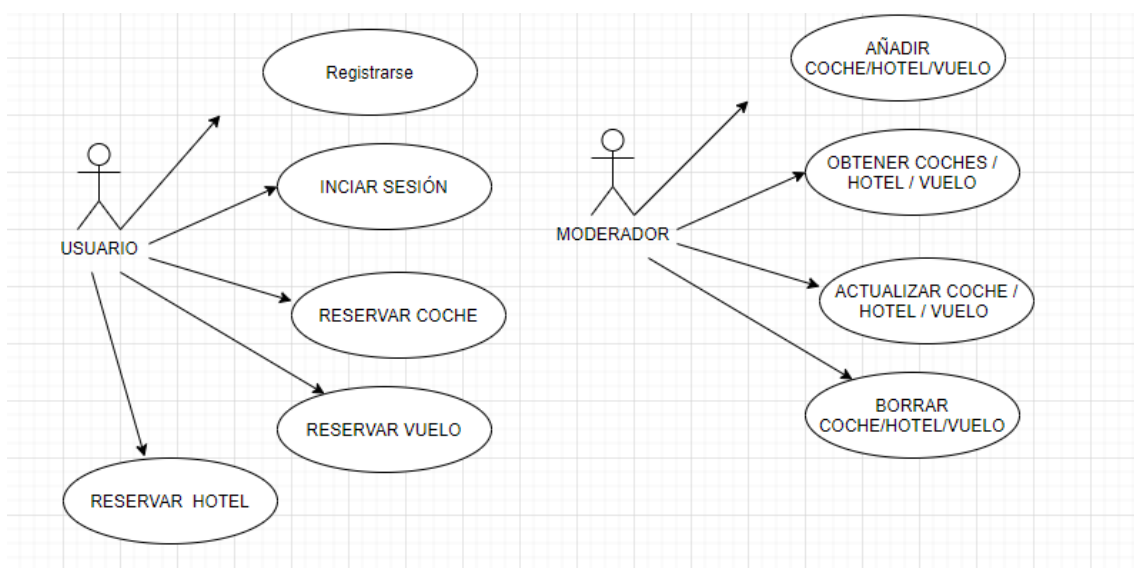


En nuestro caso nos hemos decantado por una GateWay que conecta con nuestro gestor de transacciones , y este con el resto de módulos .Esta GateWay haría de función de registro en “/Agencia/Registro” propia de un modelo SOA. De esta manera nos ahora crear un modelo tipo registro. Los servicios están desacoplados y es la Gateway la que se encarga de conectar con cada proveedor. El Saga Manager es nuestro gestor de transacciones que se encargará de solventar los problemas que puedan llegar a ocurrir en las transacciones concurrentes y distribuidas.

Arquitectura Técnica:



Casos de Usos



Los usuarios pueden registrarse, iniciar sesión y realizar las reservar que desea.

El moderador será quien controla cada proveedor para añadir, modificar o borrar los productos que desee

Rutas

Aquí explicaremos las rutas que se deben de introducir en nuestro Portman para poder utilizar nuestro servicios.

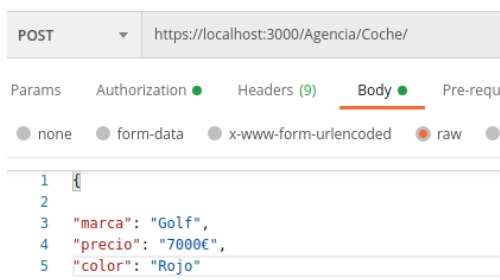
La nomenclatura en Postman es : **https://localhost:3000/**

Verbo HTTP	Ruta	Descripción
GET	/Agencia/{colección}	Obtendrás toda la información de la colección introducida (Coche,Vuelo,Hotel) almacenado en la base de datos
GET	/Agencia/{colección}/{id}	Obtendrás la información del producto seleccionado por la id
GET	/Agencia/Registrarse	Obtiene todos los usuarios registrados en la base de datos
POST	/Agencia/{colección}/	Introducirás un nuevo producto en la colección deseada
PUT	/Agencia/{colección}/{id}	Actualizaras la información del producto seleccionado por la id
DELETE	/Agencia/{colección}/{id}	Borraras la información del producto seleccionado por la id
POST	/Agencia/Registrar	Registra un usuario
POST	/Agencia/Sesión	Iniciar sesión con email y contraseña de un usuario. Se obtiene el token
PUT	/Agencia/Reservar/{colección}/{id}	Se reservar el producto de la colección indicada

Estructuras a de las ruta POST

POST	/Agencia/{colección}/
------	-----------------------

Coche



Hotel

POST

https://localhost:3000/Agencia/Hotel/

Params

Authorization

Headers (9)

Body

Pre-request Script

none

form-data

x-www-form-urlencoded

raw

binary

3

"nombre": "NX",

4

"precio": "7000€",

5

"direccion": "Santa Mateo 18 /P5"

6

Vuelo

POST

https://localhost:3000/Agencia/Vuelo

Params

Authorization

Headers (9)

Body

none

form-data

x-www-form-urlencoded

1

{

2

3

"nombre": "NewYork",

4

"precio": "120€",

5

"fecha" : "25/06/2022"

6

}

Usuario

POST	/Agencia/Registrar
GET	/Agencia/Registrarse

POST

https://localhost:3000/Agencia/Registrarse

Params

Authorization

Headers (9)

Body

F

none

form-data

x-www-form-urlencoded

raw

1

{

2

"nombre": "Ramon",

3

"email": "ramon@gmail.com",

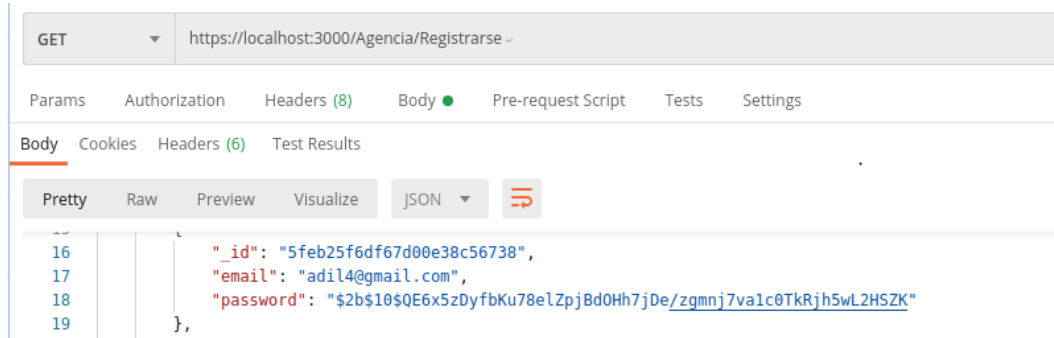
4

"password": "r123€"

5

La contraseña después se encripta.

GET



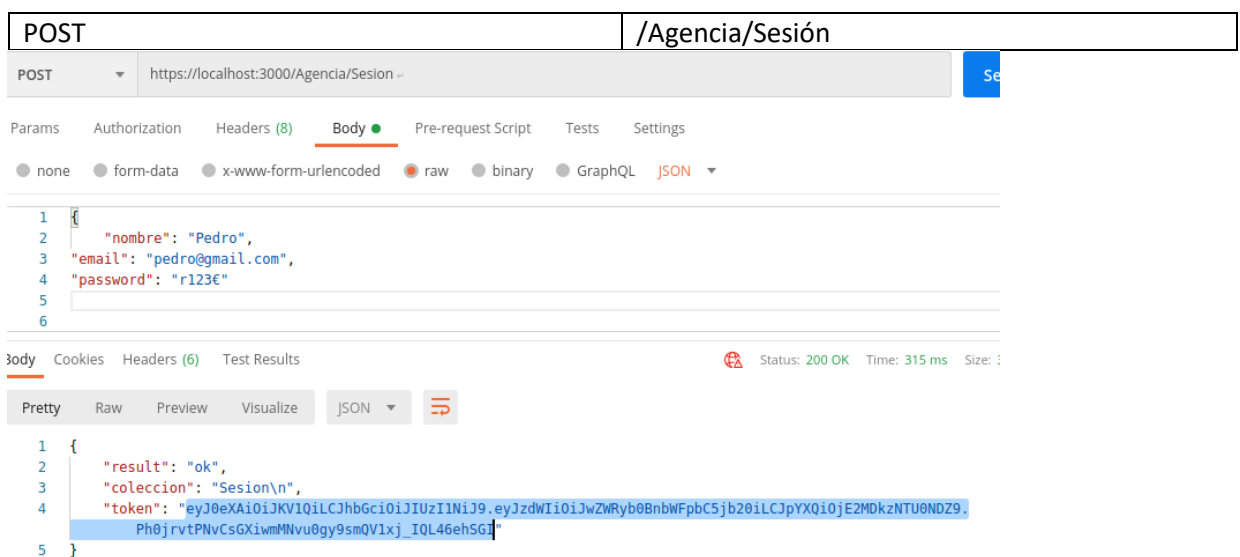
GET <https://localhost:3000/Agencia/Registrarse>

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
16 {
17   "_id": "5feb25f6df67d00e38c56738",
18   "email": "adil4@gmail.com",
19   "password": "$2b$10$QE6x5zDyfbKu78elZpj8d0Hh7jDe/zgmnj7valc0TkRjh5wL2HSZK"
20 }
```



POST <https://localhost:3000/Agencia/Sesión>

POST <https://localhost:3000/Agencia/Sesión>

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON

```
1 {
2   "nombre": "Pedro",
3   "email": "pedro@gmail.com",
4   "password": "r123€"
5 }
6
```

body Cookies Headers (6) Test Results Status: 200 OK Time: 315 ms Size: 100 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "result": "ok",
3   "coleccion": "Session\n",
4   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwZWRyb08nbWVpbC5jb20iLCJpYXQiOiJlZ2MDkzNTU0NDZ9.
5   Ph0jrvtPNvCsGXimMNvu0gy9smQV1xj_IQL46ehSGJ"
6 }
```

Ese token será necesario para poder utilizar el resto de servicios:

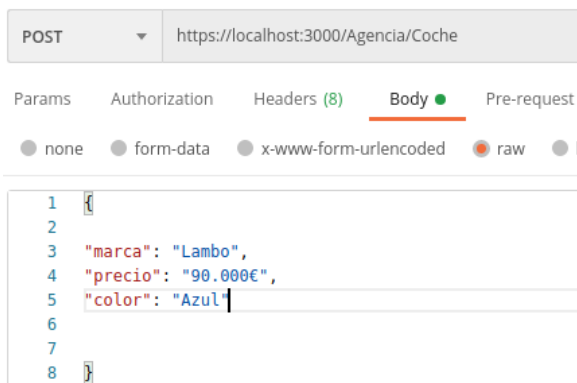
Servicios:

Hemos implementado 5 servicios estos son : Coche,Vuelo,Hotel ,Usuario y GateWay.

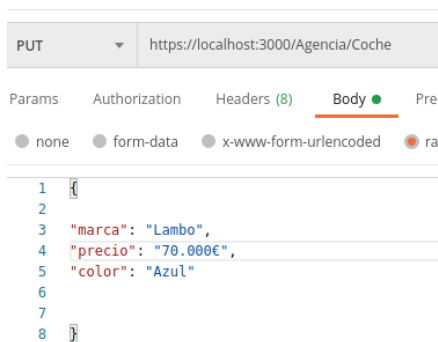
Proveedores:

Todos los proveedores tiene una estructura interna similar. En cada uno de ellos se puede realizar lo siguiente:

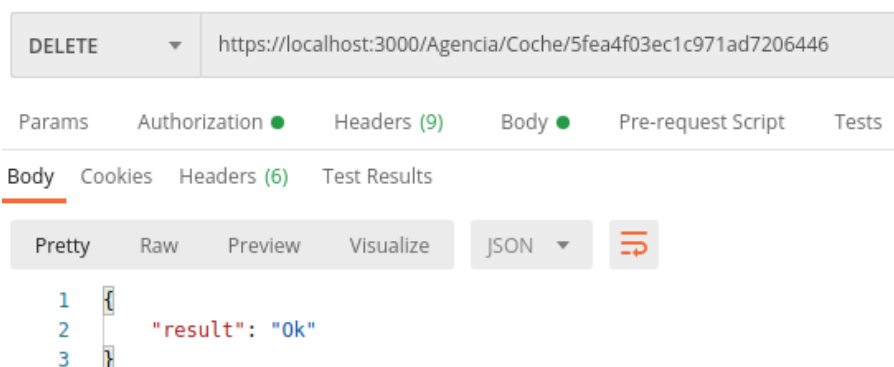
Post: Nos permite añadir un nuevo coche,vuelo o hotel.



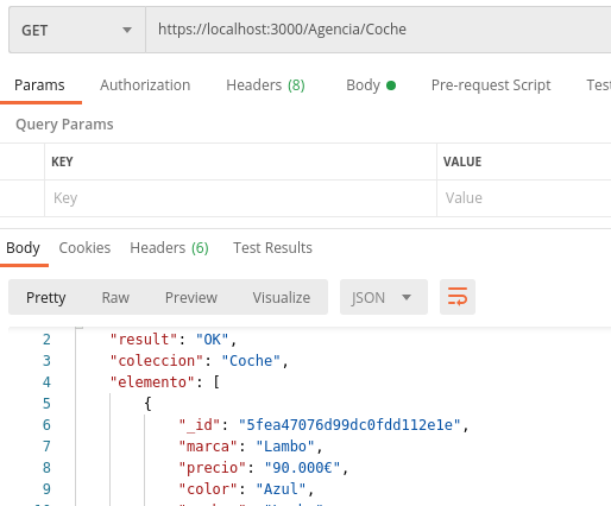
Put: Nos permite actualizar la información de nuestro proveedores.



Delete: Nos permite borrar un coche, vuelo o hotel. Debemos de indicar la id del producto a borrar.



Get :Nos permite obtener todos los productor que ofrece un proveedor.



GateWay

Este servicio se encargará de redirigirnos a cada uno de los proveedores de los cuales deseamos realizar alguna acción.

Usuarios

Aquí cada usuario se puede registrar e iniciar sesión. Cuando inicia sesión se le entrega un token para que pueda realizar las reservas que desea.

GUIA DE DESPLIEGUE

Hemos realizado la practica en Ubuntu

Primero descomprimos el archivo, encontraremos 5 carpetas cada una de ellas hace referencia a un servidor .

Hoteles_SD: Servidor de hoteles

Vuelos: Servidor de vuelos

Coche: Servidor de coche

Usuario: Servidor de usuario

GateWay: Se encarga de redireccionar las peticiones.

Para iniciar un servidor entramos en la carpeta deseada y escribimos npm start. Cabe destacar que debemos de tener instalado node para que esto funcione. En nuestra prueba al iniciarlo en CentOS sale automáticamente la opción para instalar el node.

En caso de que la carpeta node_modules no este en algún servidor nuestro, lo podemos instalar con **npm i**.

La GateWay es la que se encarga de redirigir nuestras peticiones a los distintos servidores, por tanto , debemos de modificar esta parte :

```
const IP_Coche = "localhost";
const IP_Hotel = "localhost";
const IP_Vuelo = "localhost";
const IP_Usuario = "localhost";
```

Aquí introducimos la ip de nuestros servidores.

La base de datos utilizada esta en la nube y nos conectamos a ella desca los proveedores de esta manera :

```
var db = mongojs("mongodb+srv://adil:966889991@cluster0.d6ppj.mongodb.net/SD?retryWrites=true&w=majority");
```

CentOs:

```
index.js node_modules package.json package-lock.json package-lock.json
[root@localhost Coches]# npm start

> sd@1.0.0 start /root/Coches
> nodemon index.js

[nodemon] 2.0.6
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
API REST ejecutándose en http://localhost:3002/Agencia/Coche/:id
```

Realizamos una petición desde nuestro Portman y se procesa la solicitud:

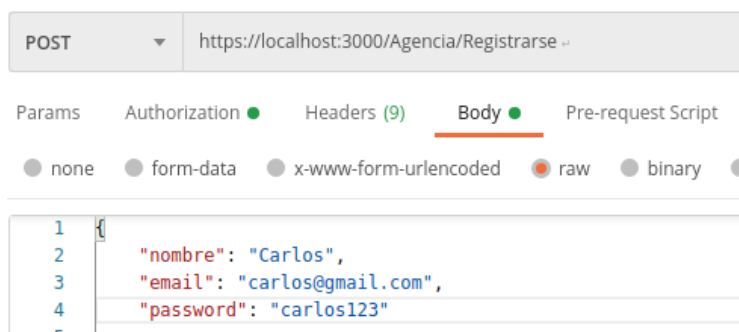
```
API REST ejecutándose en http://localhost:3002/Agencia/Coche/:id
param /Agencia/Coche
{ Coche: 'Coche' }
GET /Agencia/Coche 200 17663.216 ms - 311
```

Puede tardar algunos segundos en procesar la petición debido a que debe de acceder a la base de datos almacenada en la nube.

Todos los gets mostrados en rutas funcionan sin la necesidad de un token, pero para realiza cualquier otra acción necesitamos un token.

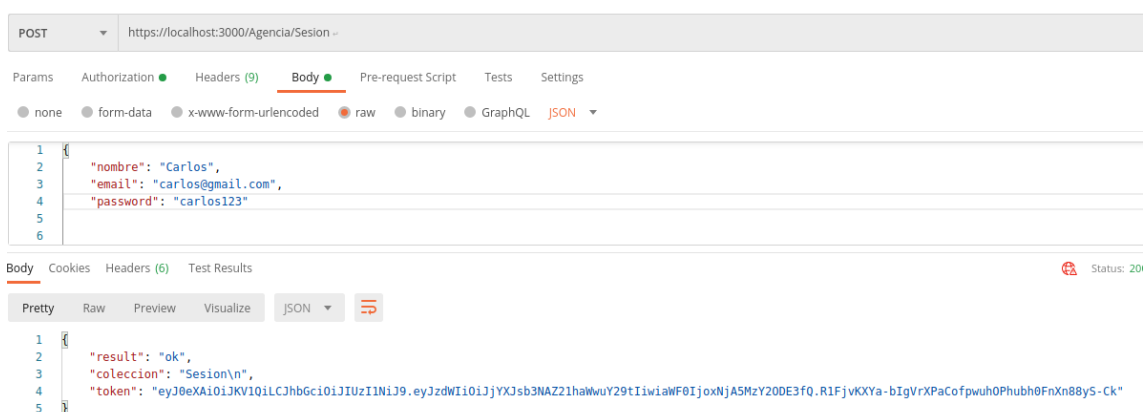
Guía de obtención de token:

Para obtener un token y poder usarlo en nuestro Postman debemos de iniciar sesión con un usuario ya almacenado en nuestra db o también podemos registrarnos para crear uno. Esta acción función sin la necesidad de un token.

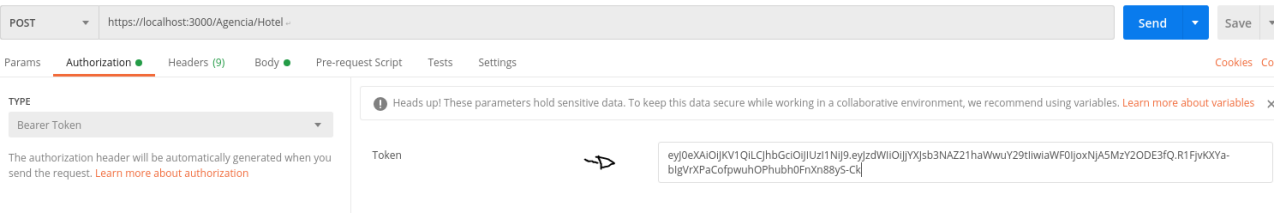


Cabe destacar que debemos de recordar la contraseña porque aunque hagamos un get y obtengamos todos los usuarios de nuestra db, sus contraseñas estarán encriptadas.

Ahora cambiamos “Registrarse” por “Sesión” y de esta manera



Ahora copiamos ese token y lo pegamos en “Authorization” del Postman.



De esta manera el resto de rutas indicadas con anterioridad funcionarán.

Aspecto Relevantes

Cabe destacar que nuestra GateWay utiliza un sistema de certificado https para garantizar su seguridad.

```
const opciones =  
{  
  key: fs.readFileSync('./certificados/key.pem'),  
  cert: fs.readFileSync('./certificados/cert.pem')  
};
```

Y de esta manera creamos nuestro servidor:

```
//Iniciamos la aplicación  
https.createServer(opciones, app).listen(port, () => {  
  console.log(`Agencia REST ejecutándose en https://localhost:\${port}/Agencia:collecciones/`\);  
}\);
```

También contamos con un sistema de Token para impedir que usuarios no registrados puedan acceder a nuestro servidores. Además las contraseñas de todos nuestros usuarios se encriptan con el sistema de hash para poder garantizar la seguridad de nuestros usuarios.