



UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS



MANUAL TÉCNICO

APIrest en Golang

Judith Alejandra Candelaria Sánchez 1723541
Gretel Gabrielle González Rodríguez 1805315
Marco Antonio Vázquez Rivera 1678576

Para hacer un uso correcto de esta API hay que cumplir con los siguientes requerimientos:

- Instalar [SQL Server Management Studio](#) para la base de datos.
- Instalar [Visual Studio Code](#) para el proyecto de Go de la API.
- Instalar [Git](#) para instalar librerías necesarias para el código del proyecto de Go de la API.
- Instalar [Golang](#) para el proyecto de Go de la API.
- Descargar o clonar el [proyecto desde el repositorio](#).
- Ejecutar el [script de la base de datos](#) (carpeta database) en SQL Server Management Studio.
- Ejecutar el [proyecto de Go de la API](#) (carpeta backend) en Visual Studio Code.
- Instalar [Postman](#) para realizar pruebas de las peticiones.
- Importar la [colección de peticiones en Postman](#) (carpeta request collection postman) para realizar pruebas.

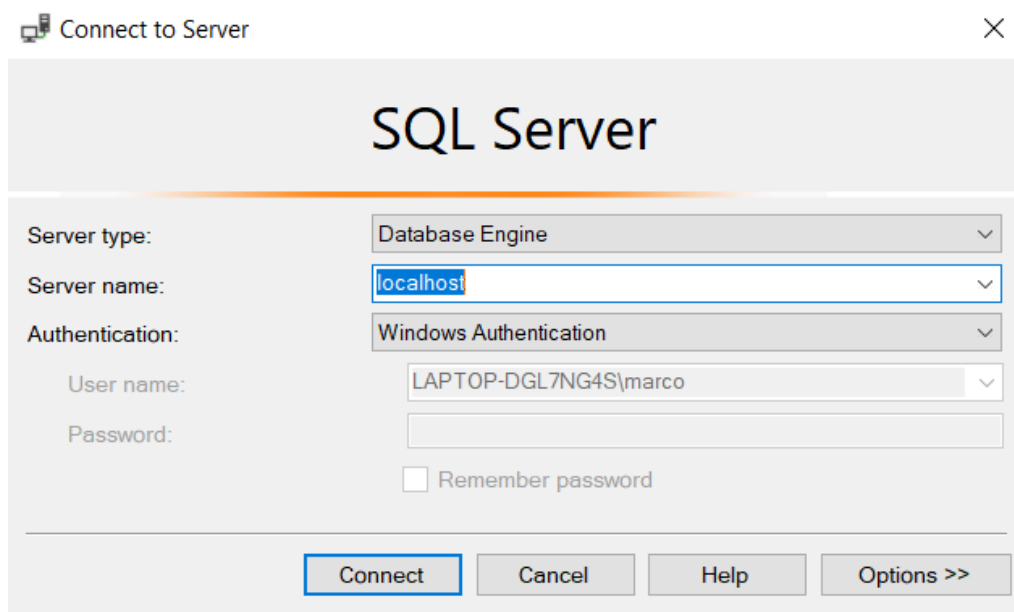
Todos los links de descargas necesarios están adjuntados en cada punto en forma de hipervínculo.

Es necesario que corra el servidor de la base de datos y el servidor de la API al mismo tiempo. A continuación, se explicará cómo hacerlo.

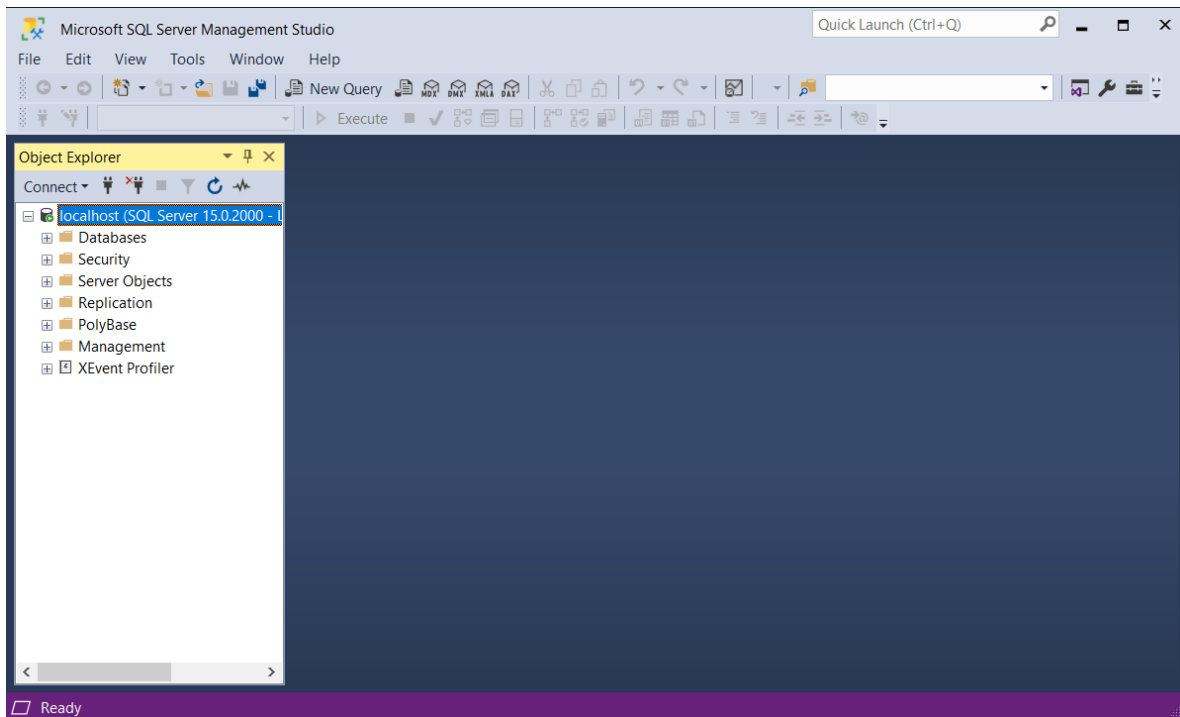
Procedimiento levantar el servidor de la base de datos y ejecutar el script de la base de datos.

Una vez que hayamos descargado el proyecto desde el repositorio, del cual facilitamos el vínculo para descargarlo en los puntos anteriores, procederemos a abrir SQL Server Management Studio.

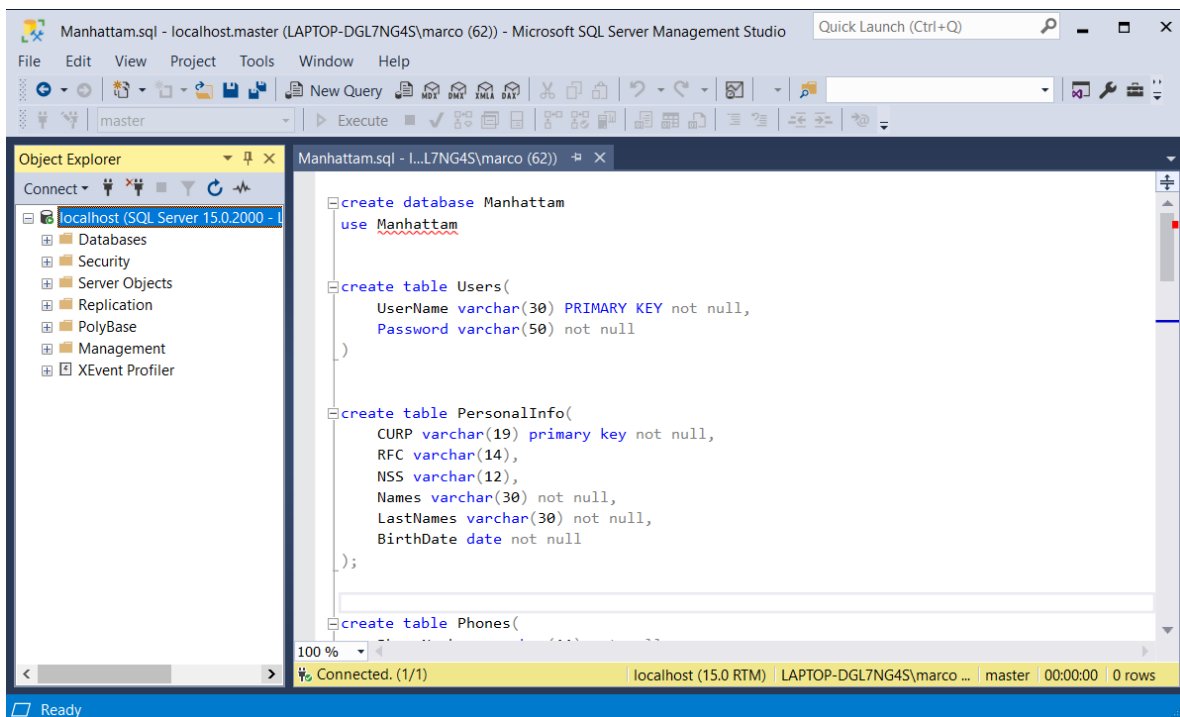
Aparecerá esta ventana de login, hay que asegurarnos que el nombre del servidor sea localhost. El User name es diferente para cada computadora, no prestar atención al User name aquí mostrado



Procedemos a dar clic en Connect para levantar el servidor de la base de datos. Si todo sale bien, se desplegará la siguiente ventana.

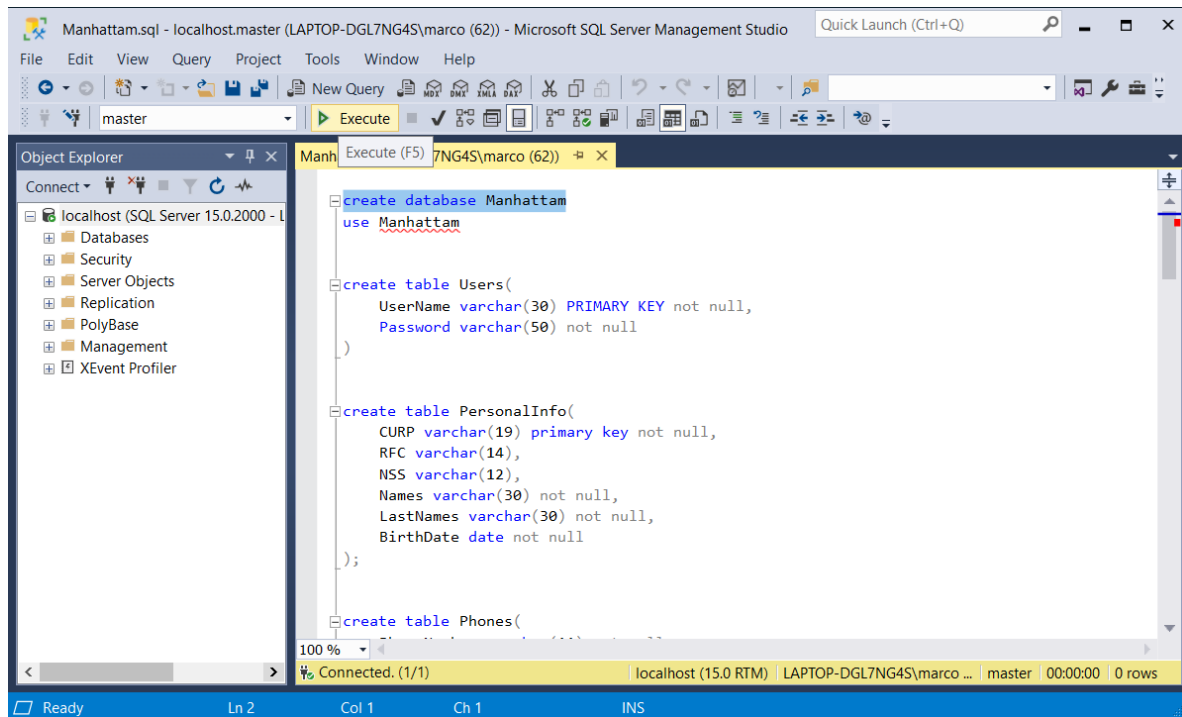


Para abrir el script de la base de datos que descargamos, podemos arrastrar el archivo (script dentro de carpeta database del proyecto) al espacio de trabajo azul y se abrirá automáticamente. Otra opción para abrir el script es dar clic y navegar en File/Open/File, navegar hasta encontrar el script de la base de datos (dentro de carpeta database del proyecto), seleccionarlo y dar clic en Open. La siguiente captura muestra el script abierto en el espacio de trabajo.

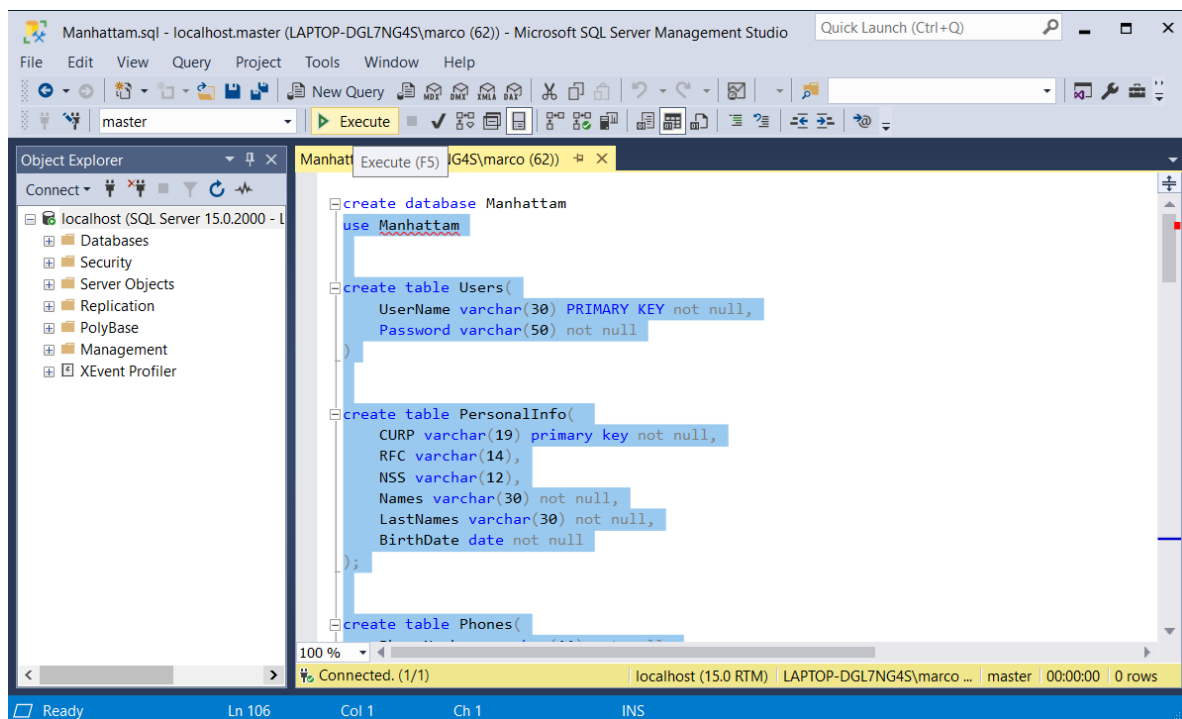


Una vez que hayamos abierto el script, procederemos a ejecutarlo de la siguiente manera.

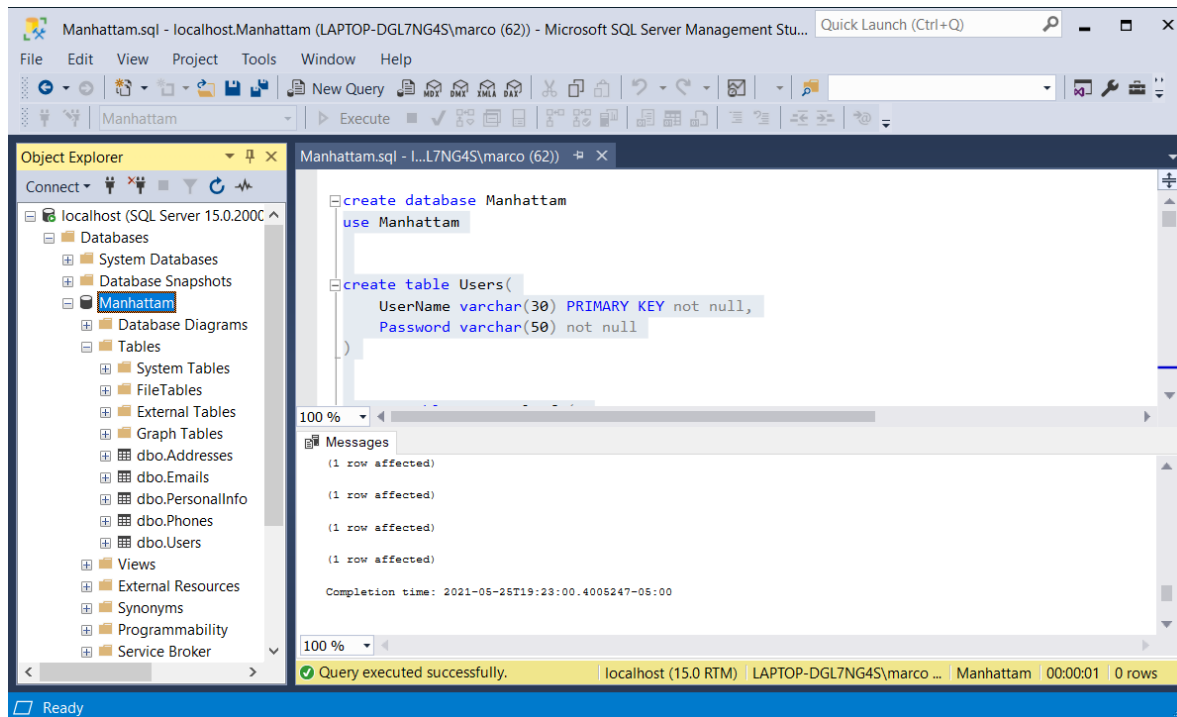
Seleccionaremos únicamente la primera línea del script y daremos clic en Execute:



Después de ejecutar la primera línea, seleccionaremos el resto del código, desde la segunda línea hasta la línea final del script y daremos clic en Execute:

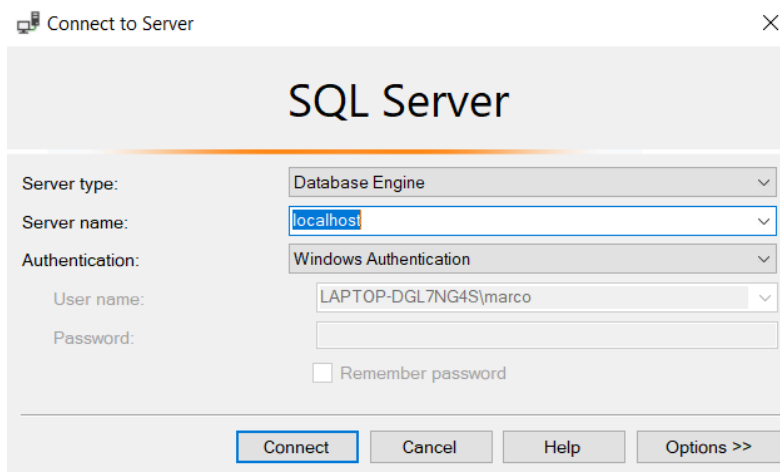


Una vez ejecutado el script correctamente nuestra consola tendrá los siguientes mensajes y la base de datos y sus entidades aparecerán en el Object Explorer, como se muestra a continuación.



A partir de este momento, **el servidor ya está corriendo, la base de datos ya está creada y ya tenemos registros** dentro de ella (el script ya nos proporciona registros con la sentencia INSERT INTO). El servidor se encuentra corriendo el **puerto 1433** y el **nombre de la base de datos es Manhattan**.

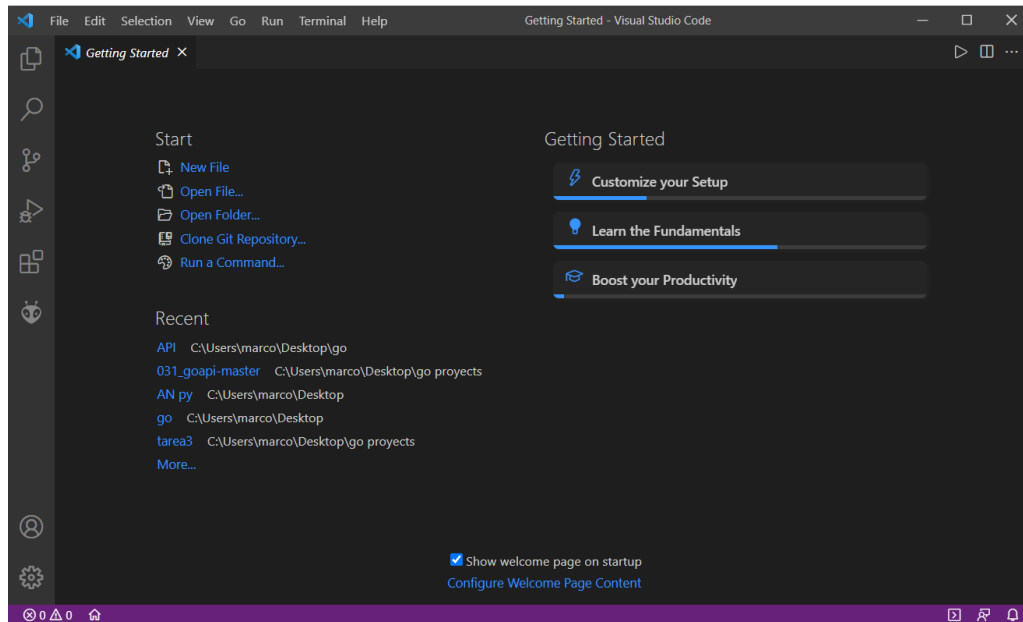
Al momento de cerrar SQL Server Management Studio, el servidor se desconecta, por lo que para volver a levantar el servidor solo hay que volver a abrir SQL Server Management Studio e iniciar sesión como el primer paso.



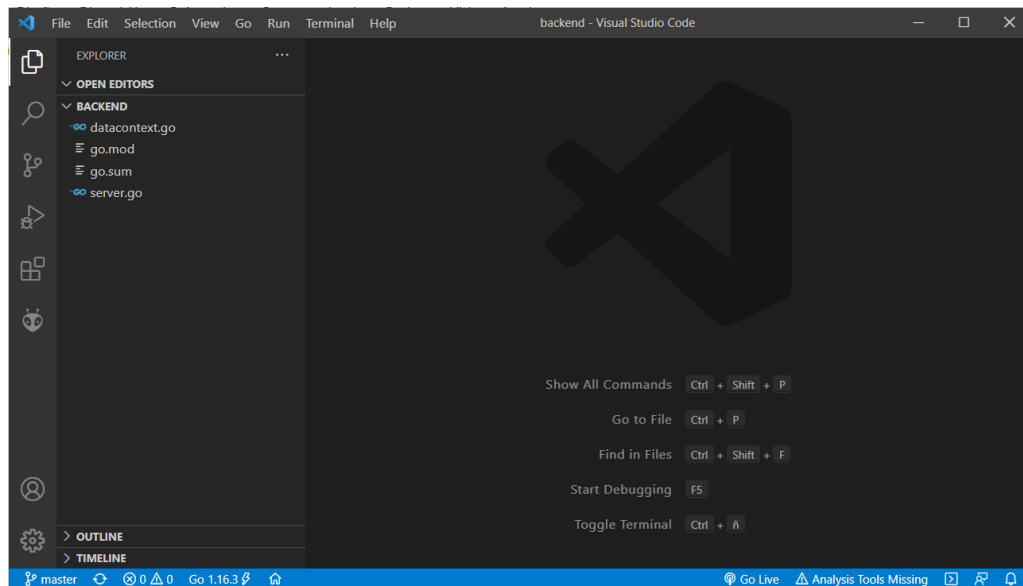
No es necesario realizar todo el procedimiento anterior, ya que la base de datos ya ha sido creada.

Procedimiento conectar la API con la base de datos, ejecutar el código y levantar el servidor de la API.

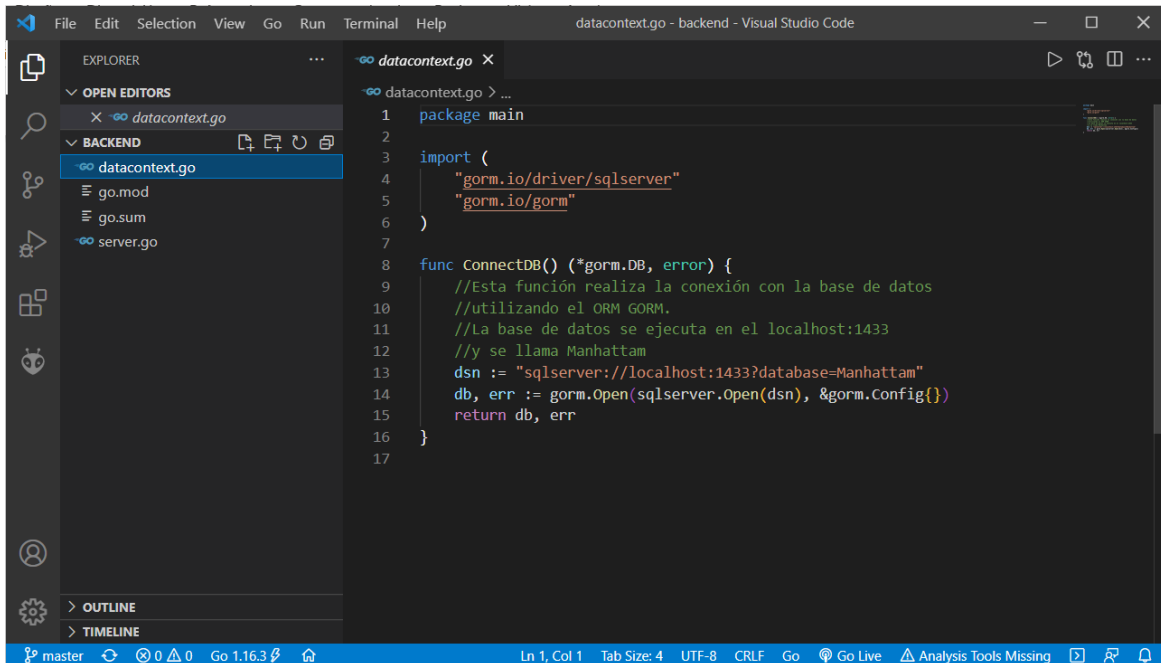
Una vez que hayamos levantado el servidor de la base de datos y ejecutado el script de la base de datos, procederemos a abrir Visual Studio Code para ejecutar el proyecto de Go de la API y levantar el servidor de la API.



Para abrir proyecto de Go de la API que descargamos, podemos arrastrar la carpeta del proyecto (carpeta llamada backend) a la ventana de Visual Studio Code y se abrirá automáticamente. Otra opción para abrir el proyecto es dar clic y navegar en File/Open Folder, navegar hasta encontrar la carpeta del proyecto de Go de la API (carpeta llamada backend), seleccionarlo y dar clic en Seleccionar Carpeta. La siguiente captura muestra el proyecto de Go de la API abierto en Visual Studio Code.



El primer paso es realizar la conexión al servidor de la base de datos y a la base de datos. Procederemos a abrir el archivo `datacontext.go` y localizamos la línea 13 de código.

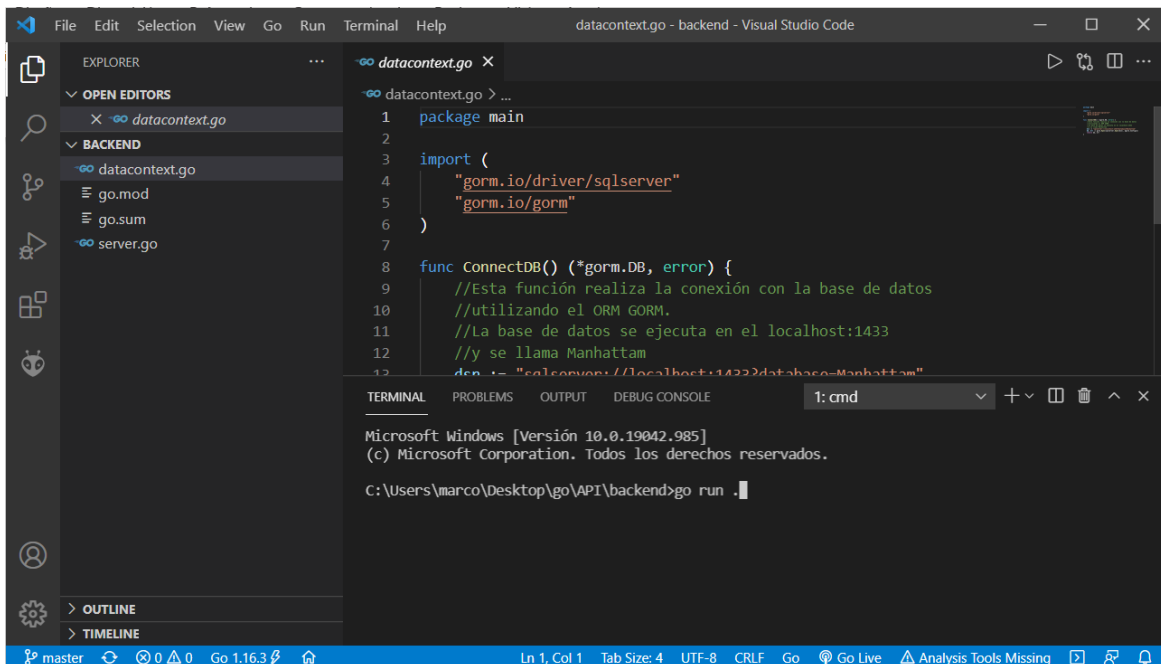


The screenshot shows the Visual Studio Code interface with the `datacontext.go` file open. The Explorer panel on the left shows the project structure with `datacontext.go` selected. The main editor displays the following Go code:

```
1 package main
2
3 import (
4     "gorm.io/driver/sqlserver"
5     "gorm.io/gorm"
6 )
7
8 func ConnectDB() (*gorm.DB, error) {
9     //Esta función realiza la conexión con la base de datos
10    //utilizando el ORM GORM.
11    //La base de datos se ejecuta en el localhost:1433
12    //y se llama Manhattam
13    dsn := "sqlserver://localhost:1433?database=Manhattam"
14    db, err := gorm.Open(sqlserver.Open(dsn), &gorm.Config{})
15    return db, err
16 }
17
```

Proporcionaremos en esta línea de código el puerto donde está ejecutándose la base de datos (por default es el puerto 1433) y el nombre de la base de datos (Manhattam).

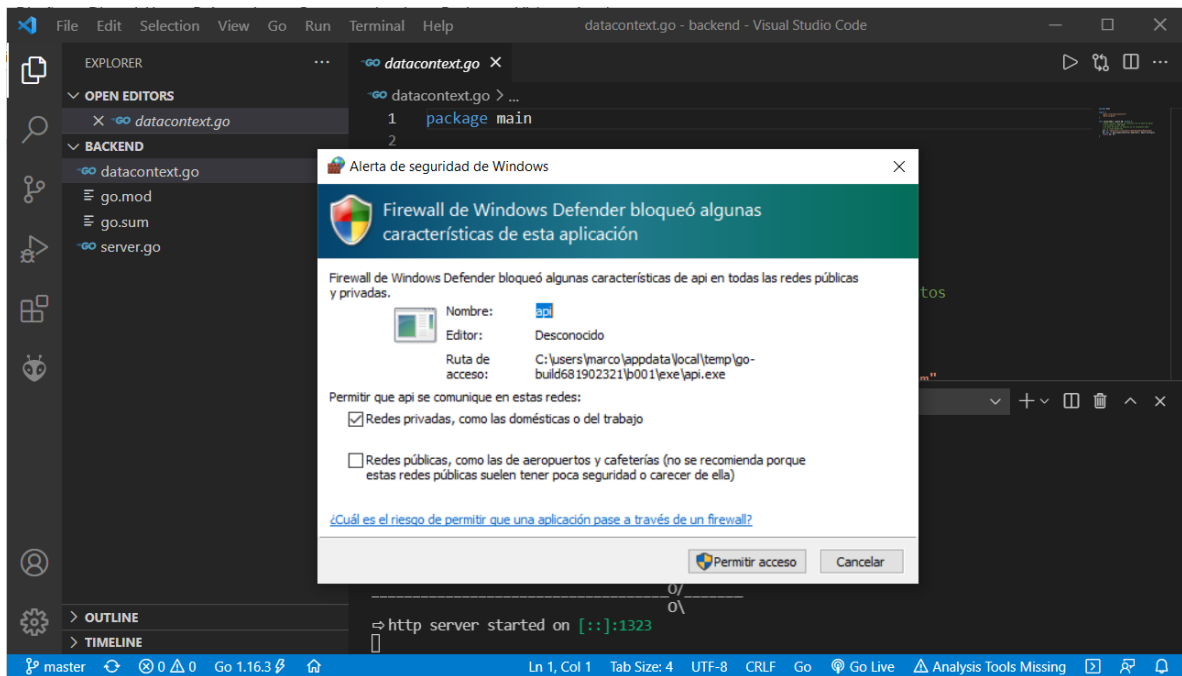
Una vez que hayamos proporcionado el puerto donde está ejecutándose la base de datos y el nombre de la base de datos, procederemos a ejecutar el proyecto, abriendo la terminal de Visual Studio Code con la combinación de teclas `Ctrl+Ñ` y escribiendo el comando `go run .` como se muestra en la siguiente captura.



The screenshot shows the Visual Studio Code interface with the `datacontext.go` file open. The terminal panel at the bottom is active, showing the command `go run .` being executed. The output of the command is displayed in the terminal:

```
Microsoft Windows [Versión 10.0.19042.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\marco\Desktop\go\API\backend>go run .
```



Damos clic en Permitir acceso, y a partir de este momento la API ya está ejecutándose en el **puerto 1323**.

Peticiones HTTP

A continuación, mostraremos cómo realizar peticiones a la API de manera correcta, proporcionando una documentación de cada petición http y endpoints disponibles.

POST login

Loguea en la API y genera un Token.

- **URL:** localhost:1323/login?UserName=&Password=
- Esta es una petición pública.
Esta petición POST login permite iniciar sesión con un usuario y contraseña existentes en la base de datos y generar un token para la autenticación bearer token.
Este token se requiere para realizar las peticiones http a la base de datos que se mostrarán a continuación. Es necesario guardar este token y agregarlo en las peticiones en la parte de Autenticación de tipo Bearer token.
- **Request Params**
 - UserName: nombre de usuario registrado en la base de datos.
 - Password: contraseña del usuario registrado en la base de datos.

Prueba:

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** localhost:1323/login?UserName=marco&Password=marco123
- Send Button:** A blue button labeled "Send".
- Params Tab:** Shows "Query Params" with a table:

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	UserName	marco			
<input checked="" type="checkbox"/>	Password	marco123			
	Key	Value	Description		

Below the table, the **Body** tab is selected, showing the response in JSON format:

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MjIyNTcyMjEsIm5hbWUiOiJ0YXJybyIsInR5cGUiOiJBZG1pb1J9.hYDFoy01Pw0JNIR0pXMCbJErg3AVBHBnrC_f18rVWY"
3 }
```

At the top right of the response area, it shows "200 OK", "25 ms", and "296 B". There is also a "Save Response" button.

POST signUp

Sign up (Crea usuario).

- **URL:** localhost:1323/signUp?UserName=&Password=
- Esta petición es pública.
Esta petición POST signUp permite crear un nuevo usuario en la base de datos, por lo que el usuario que sea creado ya estará registrado y podrá generar un token para realizar peticiones a la base de datos.
- **Request Params**
 - UserName: nombre de usuario para registrar en la base de datos.
 - Password: contraseña del usuario para registrar en la base de datos.

Prueba:

POST localhost:1323/signup?UserName=jose&Password=jose123 Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> UserName	jose			
<input checked="" type="checkbox"/> Password	jose123			
Key	Value	Description		

Body 201 Created 82 ms 183 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "UserName": "jose",
3   "Password": "jose123"
4 }
```

GET personalInfo

Trae todos los registros en PersonalInfo.

- **URL:** localhost:1323/jwt/personalInfo
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición GET personalInfo permite consultar todos los registros de la entidad PersonalInfo de la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.

Prueba:

GET localhost:1323/jwt/personalInfo Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 200 OK 67 ms 1.28 KB Save Response

Pretty Raw Preview Visualize JSON

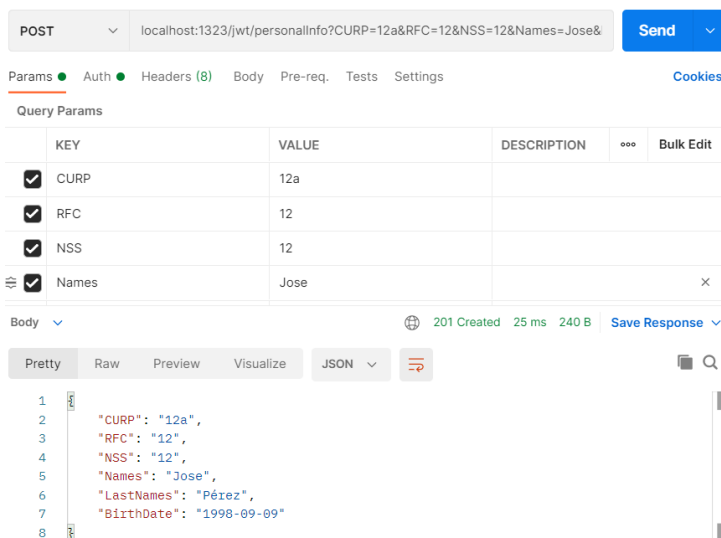
```
1 [
2   {
3     "CURP": "101",
4     "RFC": "456",
5     "NSS": "789",
6     "Names": "Abigail",
7     "LastNames": "Picapiedra",
8     "BirthDate": "1998-09-08T00:00:00Z"
9   },
10  {
11    "CURP": "123",
12    "RFC": "456",
13    "NSS": "789",
14    "Names": "Abigail",
15    "LastNames": "Picapiedra",
16    "BirthDate": "1998-09-08T00:00:00Z"
17  }
18 ]
```

POST personalInfo

Crea un registro en PersonalInfo.

- **URL:** localhost:1323/jwt/personalInfo?CURP=&RFC=&NSS=&Names=&LastNames=&BirthDate=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición POST personalInfo permite crear un registro en la entidad PersonalInfo en la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - CURP: CURP a registrar.
 - RFC: RFC a registrar.
 - NSS: NSS a registrar.
 - Names: Nombres a registrar.
 - LastNames: Apellidos a registrar.
 - BirthDate: Fecha de nacimiento a registrar.

Pruebas:



POST localhost:1323/jwt/personalInfo?CURP=12a&RFC=12&NSS=12&Names=Jose&

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> CURP	12a			
<input checked="" type="checkbox"/> RFC	12			
<input checked="" type="checkbox"/> NSS	12			
<input checked="" type="checkbox"/> Names	Jose			

Body 201 Created 25 ms 240 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "CURP": "12a",
3   "RFC": "12",
4   "NSS": "12",
5   "Names": "Jose",
6   "LastNames": "Pérez",
7   "BirthDate": "1998-09-09"
8 }
```

PUT personalInfo

Modifica un registro en PersonalInfo.

- **URL:** localhost:1323/jwt/personalInfo?CURP=&RFC=&NSS=&Names=&LastNames=&BirthDate=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición PUT personalInfo permite modificar un registro en la entidad PersonalInfo en la base de datos.
Los parámetros requeridos son los datos que serán modificados y los datos que no serán modificados también.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - CURP: CURP de la persona de datos a modificar.
 - RFC: RFC modificada.
 - NSS: NSS modificado.
 - Names: Nombres modificados.
 - LastNames: Apellidos modificados
 - BirthDate: Fecha de nacimiento modificada.

Prueba:

PUT localhost:1323/jwt/personalInfo?CURP=12a&RFC=123&NSS=123&Names=Jose Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	CURP	12a			
<input checked="" type="checkbox"/>	RFC	123			
<input checked="" type="checkbox"/>	NSS	123			
<input checked="" type="checkbox"/>	Names	Jose			

Body 200 OK 48 ms 238 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "CURP": "12a",
3   "RFC": "123",
4   "NSS": "123",
5   "Names": "Jose",
6   "LastNames": "Pérez",
7   "BirthDate": "1998-09-09"
8 }
```

DEL personalInfo

Elimina un registro en PersonalInfo.

- **URL:** localhost:1323/jwt/personalInfo?CURP=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición DELETE personalInfo permite eliminar un registro en la entidad PersonalInfo en la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - CURP: CURP de registro a eliminar

Prueba:

DELETE localhost:1323/jwt/personalInfo?CURP=12a Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	CURP	12a			
	Key	Value	Description		

Body 204 No Content 202 ms 78 B Save Response

Pretty Raw Preview Visualize Text

1

GET addresses

Trae todos los registros en Addresses.

- **URL:** localhost:1323/jwt/addresses
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición GET addresses permite consultar todos los registros de la entidad Addresses de la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.

Prueba:

The screenshot shows a REST client interface. At the top, a GET request is configured to `localhost:1323/jwt/addresses`. Below the request bar, the 'Query Params' section is visible, showing a table with headers 'KEY', 'VALUE', and 'DESCRIPTION'. The table contains one entry: 'Key' with 'Value' and 'Description'. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response is a single object with the following fields: 'ID' (1), 'Name' (Casa), 'Street' (Estrella), 'Number' (1208), 'Neighborhood' (Luna), 'ZipCode' (66478), 'City' (Marte), 'State' (Nuevo León), 'Country' (México), and 'CURP' (123). The status bar indicates a 200 OK response with a 56 ms duration and 1.47 KB of data.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
{
  "ID": "1",
  "Name": "Casa",
  "Street": "Estrella",
  "Number": "1208",
  "Neighborhood": "Luna",
  "ZipCode": "66478",
  "City": "Marte",
  "State": "Nuevo León",
  "Country": "México",
  "CURP": "123"
}
```

POST addresses

Crea un registro en Addresses.

- **URL:** `localhost:1323/jwt/addresses?Name=&Street=&Number=&Neighborhood=&ZipCode=&City=&State=&Country=&CURP=`
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición POST addresses permite crear un registro en la entidad Addresses en la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - Name: Nombre del lugar a registrar.
 - Street: Calle de la dirección a registrar.
 - Number: Número de la dirección a registrar.
 - Neighborhood: Colonia de la dirección a registrar.
 - ZipCode: Código postal de la dirección a registrar.
 - City: Ciudad de la dirección a registrar
 - State: Estado de la dirección a registrar.
 - Country: País de la dirección a registrar.
 - CURP: CURP de la persona a quien pertenece la dirección.

Prueba:

POST localhost:1323/jwt/addresses?Name=Escuela&Street=Pedro de Alba&Number=1&Neighborhood=Centro Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	Name	Escuela			
<input checked="" type="checkbox"/>	Street	Pedro de Alba			X
<input checked="" type="checkbox"/>	Number	1			
<input checked="" type="checkbox"/>	Neighborhood	Centro			

Body 201 Created 157 ms 324 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "ID": "",
3   "Name": "Escuela",
4   "Street": "Pedro de Alba",
5   "Number": "1",
6   "Neighborhood": "Centro",
7   "ZipCode": "123",
8   "City": "San Nicolas",
9   "State": "Nuevo León",
10  "Country": "México",
11  "CURP": "123"
12 }
```

PUT addresses

Modificar un registro en Addresses.

- **URL:** localhost:1323/jwt/addresses?ID=&Name=&Street=&Number=&Neighborhood=&Zip Code=&City=&State=&Country=&CURP=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición PUT addresses permite modificar un registro en la entidad Addresses en la base de datos.
Los parámetros requeridos son los datos que serán modificados y los datos que no serán modificados también.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params:**
 - ID: ID de dirección a modificar.
 - Name: Nombre de la dirección modificada.
 - Street: Calle modificada.
 - Number: Número modificado.
 - Neighborhood: Colonia modificada.
 - ZipCode: Código postal modificado.
 - City: Ciudad modificada.
 - State: estado modificado.
 - Country: País modificado.
 - CURP: CURP de la persona a quien pertenece la dirección.

Prueba:

PUT localhost:1323/jwt/addresses?ID=9&Name=Facultad&Street=Pedro de Alba& Number=123 Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	ID	9			
<input checked="" type="checkbox"/>	Name	Facultad			
<input checked="" type="checkbox"/>	Street	Pedro de Alba			
<input checked="" type="checkbox"/>	Number	123			

Body 200 OK 42 ms 336 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "ID": "9",
3   "Name": "Facultad",
4   "Street": "Pedro de Alba",
5   "Number": "123",
6   "Neighborhood": "Centro",
7   "ZipCode": "123",
8   "City": "San Nicolas de los Garza",
9   "State": "Nuevo León",
10  "Country": "México",
11  "CURP": "123"
12 }
```

DEL addresses

Elimina un registro en Addresses.

- **URL:** localhost:1323/jwt/addresses?ID=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición DELETE addresses permite eliminar un registro en la entidad Addresses en la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token>
- **Request Params**
 - ID: ID de dirección a eliminar.

Prueba:

DELETE localhost:1323/jwt/addresses?ID=9 Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	ID	9			
	Key	Value	Description		

Body 204 No Content 28 ms 78 B Save Response

Pretty Raw Preview Visualize Text

```
1
```


GET emails

Trae todos los registros en Emails.

- **URL:** localhost:1323/jwt/emails
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición GET emails permite consultar todos los registros de la entidad Emails de la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.

Prueba:

The screenshot shows a REST client interface. At the top, a GET request is configured for the URL 'localhost:1323/jwt/emails'. Below the URL bar, tabs for 'Params', 'Auth', 'Headers (7)', 'Body', 'Pre-req.', 'Tests', and 'Settings' are visible. The 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response status is '200 OK' with a response time of '26 ms' and a size of '621 B'. The JSON data contains an array of three email records, each with 'Email' and 'CURP' fields.

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

```
1 {
2   {
3     "Email": "101@gmail.com",
4     "CURP": "101"
5   },
6   {
7     "Email": "101@hotmail.com",
8     "CURP": "101"
9   },
10  {
11    "Email": "123@gmail.com",
12    "CURP": "123"
13  }
14 }
```

POST emails

Crea un registro en Emails.

- **URL:** localhost:1323/jwt/emails?Email=&CURP=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición POST emails permite crear un registro en la entidad Emails en la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - Email: Email a registrar.
 - CURP: CURP de la persona a quien pertenece el Email.

Prueba:

POST localhost:1323/jwt/emails?Email=qwerty@gmail.com&CURP=123 Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	Email	qwerty@gmail.com			x
<input checked="" type="checkbox"/>	CURP	123			
	Key	Value	Description		

Body 201 Created 28 ms 184 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "Email": "qwerty@gmail.com",
3   "CURP": "123"
4 }
```

PUT emails

Modificar un registro en Emails.

- **URL:** localhost:1323/jwt/emails?Email=&NewEmail=&CURP=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición PUT emails permite modificar un registro en la entidad Emails en la base de datos.
Los parámetros requeridos son los datos que serán modificados y los datos que no serán modificados.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - Email: Email a modificar.
 - NewEmail: Email modificado
 - CURP: CURP de la persona a quien pertenece el Email.

Prueba:

PUT localhost:1323/jwt/emails?Email=qwerty@gmail.com&NewEmail=qwerty@hotmail.com&CURP=123 Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	Email	qwerty@gmail.com			
<input checked="" type="checkbox"/>	NewEmail	qwerty@hotmail.com			
<input checked="" type="checkbox"/>	CURP	123			
	Key	Value	Description		

Body 200 OK 84 ms 181 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "Email": "qwerty@hotmail.com",
3   "CURP": "123"
4 }
```

DEL emails

Elimina un registro en Emails.

- **URL:** localhost:1323/jwt/emails?Email=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición DELETE emails permite eliminar un registro en la entidad Emails en la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - Email: Email a eliminar.

Prueba:

DELETE localhost:1323/jwt/emails?Email=qwerty@hotmail.com Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	Email	qwerty@hotmail.com			
	Key	Value	Description		

Body 204 No Content 25 ms 78 B Save Response

Pretty Raw Preview Visualize Text 1

GET phones

Trae todos los registros en Phones.

- **URL:** localhost:1323/jwt/phones
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición GET phones permite consultar todos los registros de la entidad Phones de la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.

Prueba:

GET localhost:1323/jwt/phones Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 200 OK 61 ms 697 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "PhoneNumber": "85154651",
3   "Place": "Casa",
4   "CURP": "123"
5 },
6 {
7   "PhoneNumber": "54164616",
8   "Place": "Oficina",
9   "CURP": "123"
10 },
11 }
12
```

POST phones

Crea un registro en Phones.

- **URL:** localhost:1323/jwt/phones?PhoneNumber=&Place=&CURP=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición POST phones permite crear un registro en la entidad Phones en la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - PhoneNumber: Número de teléfono a registrar.
 - Place: Lugar del número a registrar.
 - CURP: CURP de la persona a quien pertenece el número.

Pruebas:

POST localhost:1323/jwt/phones?PhoneNumber=191919&Place=Facultad&CURP=123 Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> PhoneNumber	191919			
<input checked="" type="checkbox"/> Place	Facultad			
<input checked="" type="checkbox"/> CURP	123			
Key	Value	Description		

Body 201 Created 33 ms 199 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "PhoneNumber": "191919",
3   "Place": "Facultad",
4   "CURP": "123"
5 }
```

PUT phones

Modificar un registro en Phones.

- **URL:** localhost:1323/jwt/phones?PhoneNumber=&NewPhone=&NewPlace=&CURP=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición PUT phones permite modificar un registro en la entidad Phones en la base de datos.
Los parámetros requeridos son los datos que serán modificados y los datos que no serán modificados.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - PhoneNumber: Número de teléfono a modificar.
 - NewPhone: Número de teléfono modificado.
 - NewPlace: Lugar del número modificado.
 - CURP: CURP de la persona a quien pertenece el número.

Prueba:

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** localhost:1323/jwt/phones?PhoneNumber=191919&NewPhone=919191&NewPlace=Universidad&CURP=123
- Buttons:** Send, Params, Auth, Headers (8), Body, Pre-req., Tests, Settings, Cookies
- Params Table:**

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> PhoneNumber	191919			
<input checked="" type="checkbox"/> NewPhone	919191			
<input checked="" type="checkbox"/> NewPlace	Universidad			
<input checked="" type="checkbox"/> CURP	123			
Key	Value	Description		

- Body:** 200 OK, 156 ms, 197 B, Save Response
- Response Format:** Pretty, Raw, Preview, Visualize, JSON
- JSON Response:**

```
1 {
2   "PhoneNumber": "919191",
3   "Place": "Universidad",
4   "CURP": "123"
5 }
```

DEL phones

Elimina un registro en Phones.

- **URL:** localhost:1323/jwt/phones?PhoneNumber=&CURP=
- Esta petición requiere Bearer Token. (Iniciar sesión, generar token y utilizarlo en las peticiones).
Esta petición DELETE phones permite eliminar un registro en la entidad Phones en la base de datos.
El bearer token generado debe ser colocado en Authorization Bearer Token en la petición.
- **Authorization:** Bearer Token
- **Token:** <token> generado al logearse.
- **Request Params**
 - **PhoneNumber:** Número de teléfono a eliminar.
 - **CURP:** CURP de la persona a quien pertenece el número.

Prueba:

DELETE localhost:1323/jwt/phones?PhoneNumber=919191&CURP=123 Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> PhoneNumber	919191			
<input checked="" type="checkbox"/> CURP	123			
Key	Value	Description		

Body 204 No Content 23 ms 78 B Save Response

Pretty Raw Preview Visualize Text

1

Todas las funciones que requieren Authorization Bearer Token necesitan colocar en la petición el token generado al momento de loguearse, como se muestra a continuación.

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

Type Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

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

Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...