














DOCUMENTACIÓN API-Test

Field Name	Field Type	Size / Precision	Scale	Not Null	Defa...	Description
  id	TINYINT	11	0	<input checked="" type="checkbox"/>	Null	id del usuario
 first_name	VARCHAR	128	0	<input checked="" type="checkbox"/>	Null	nombre del usuario
 last_name	VARCHAR	128	0	<input checked="" type="checkbox"/>	Null	apellido del usuario
 email	VARCHAR	64	0	<input checked="" type="checkbox"/>	Null	email del usuario, se utiliza el campo para el login
 password	VARCHAR	1000	0	<input checked="" type="checkbox"/>	Null	contraseña del usuario para el login
 token	VARCHAR	1000	0	<input checked="" type="checkbox"/>	-1	token generado por JWT, para la autenticación del usuario. se genera en el momento que se crea el usuario y en el login
 age	INTEGER	99	0	<input type="checkbox"/>	Null	Edad del usuario
 image	VARCHAR	255	0	<input type="checkbox"/>	Null	Dirección de la imagen
 description	VARCHAR	255	0	<input type="checkbox"/>	Null	Descripción del usuario
 created_at	DATETIME	0	0	<input type="checkbox"/>	Null	Fecha en la cual se realizo la creación del usuario
 updated_at	DATETIME	0	0	<input type="checkbox"/>	Null	Ultima feha en la que se hizo una modificación en datos del usuario
 active	TINYINT	4	0	<input checked="" type="checkbox"/>	1	Estado del usuario: 1- activo 2- eliminado

Estructura de la tabla de usuarios: users.

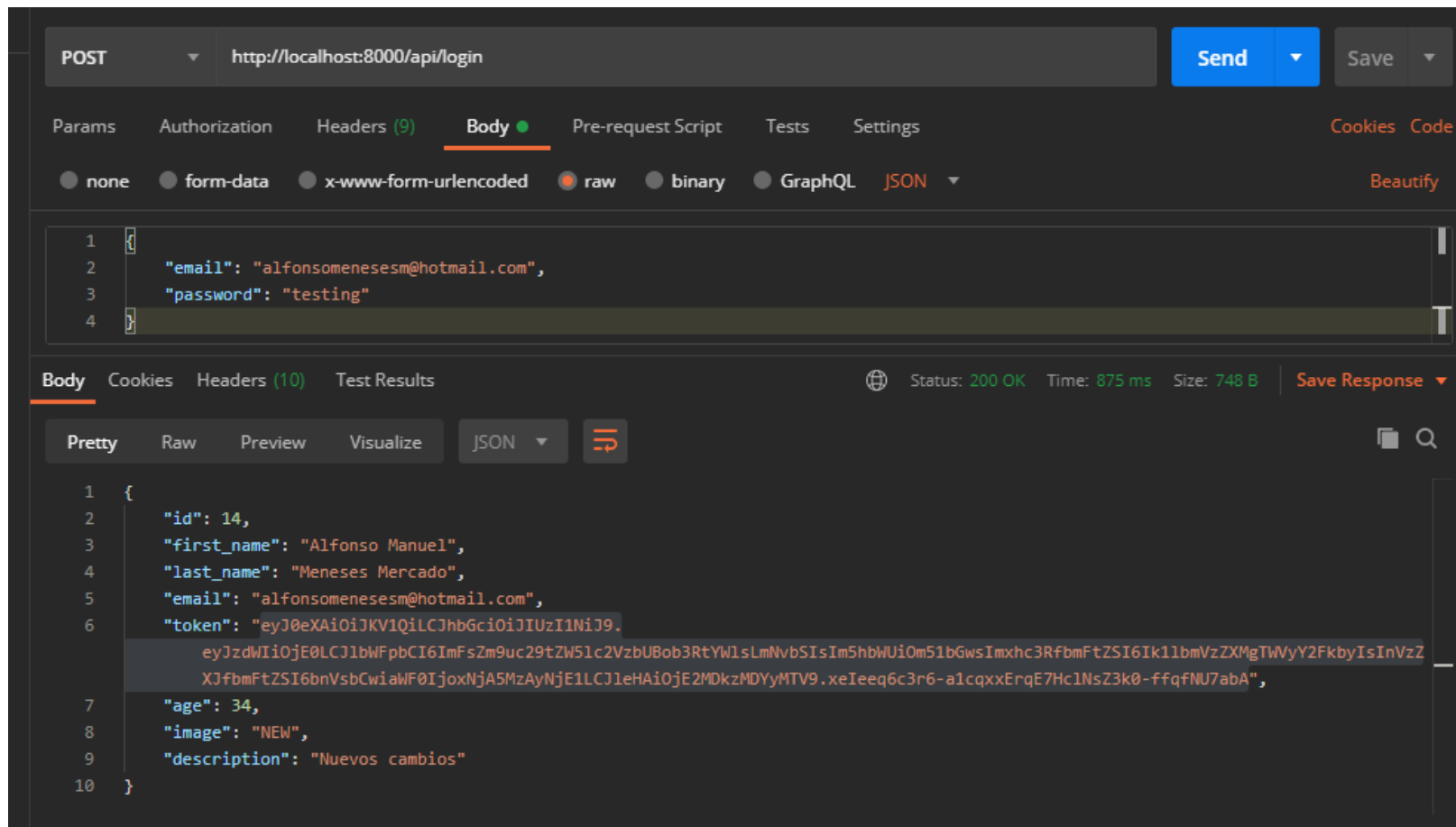
Endpoints

Endpoint	Método	URI	Payload
Login	POST	api/login	email: string -requerido
			password: string – requerido
Crear usuario	POST	api/users	first_name: string – requerido
			last_name: string -requerido
			email: string -requerido
			password: string – requerido

			age: integer
			image: string
Edición general de usuario	PUT	api/users/{id}	id: integer – requerido
			password: string – requerido
			first_name: string – requerido
			last_name: string -requerido
			email: string -requerido
			password: string – requerido
			age: integer – requerido
			image: string – requerido
Edición parcial de un usuario	PATCH	api/users/{id}	id: integer – requerido
			password: string
			first_name: string
			last_name: string
			email: string
			password: string
			age: integer
			image: string
Listar usuarios	GET	api/users	per_page: integer - opcional (cantidad de datos para la paginación)
			page: integer - opcional el numero de la lista en la paginación)
Listar usuario	GET	api/users/{id}	id: integer – requerido
Borrar usuario	DELETE	api/users/{id}	id: integer – requerido

Pruebas desde Postman

Login



POST

http://localhost:8000/api/login

Send

Save

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

Code

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

{

2

"email": "alfonsomeneses@hotmail.com",

3

"password": "nose"

4

}

Body

Cookies

Headers (10)

Test Results

Status: 401 Unauthorized

Time: 827 ms

Size: 354 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

{

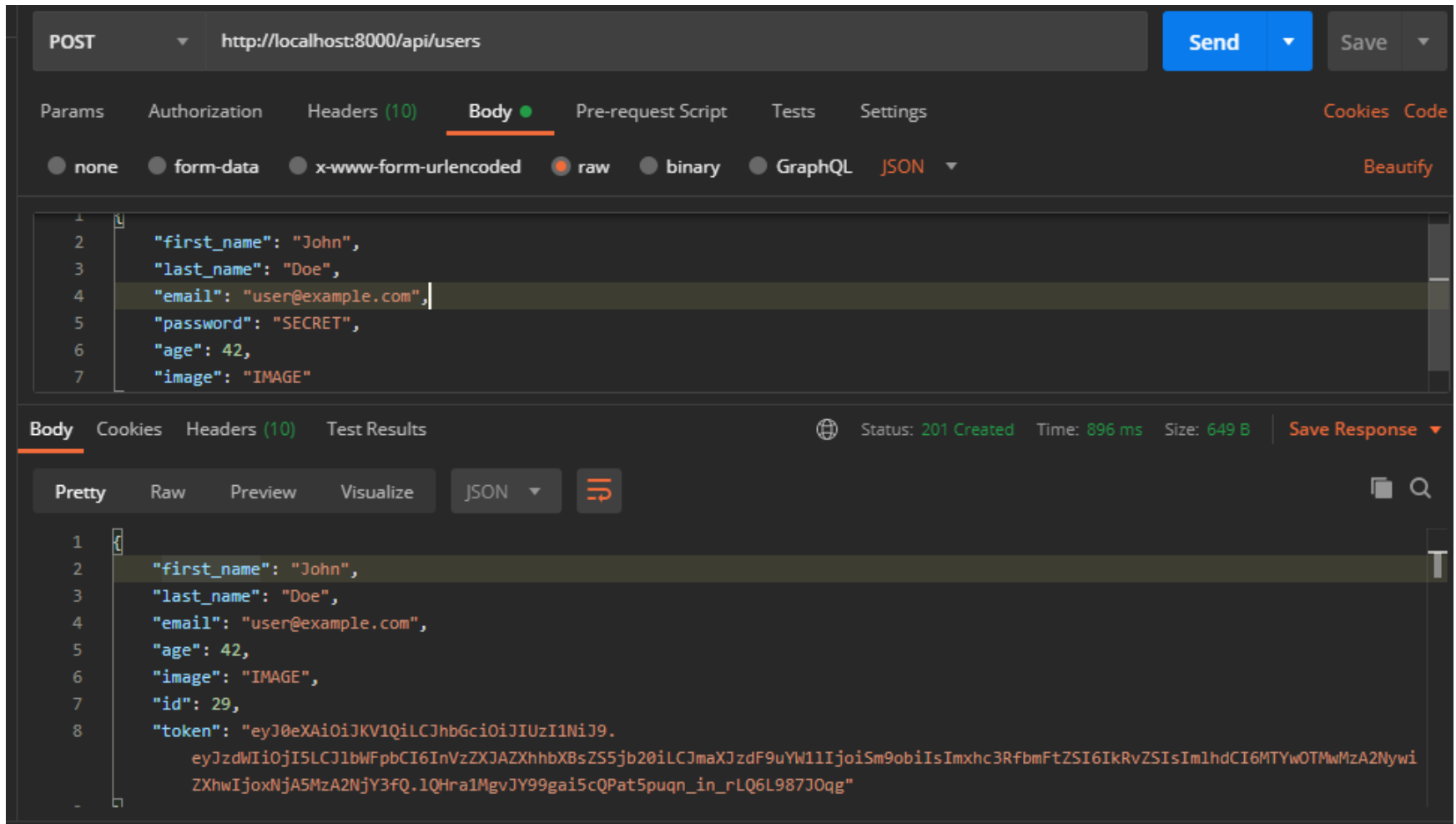
2

"error": "Error in user or password"

3

}

Crear usuario



POST

http://localhost:8000/api/users

Send

Save

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Cookies

Code

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1 {
2   "first_name": "John",
3   "last_name": "Doe",
4   "email": "user@example.com",
5   "password": "SECRET",
6   "age": 42,
7   "image": "IMAGE"
8 }
```

Body

Cookies

Headers (10)

Test Results

🌐

Status: 401 Unauthorized

Time: 744 ms

Size: 348 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

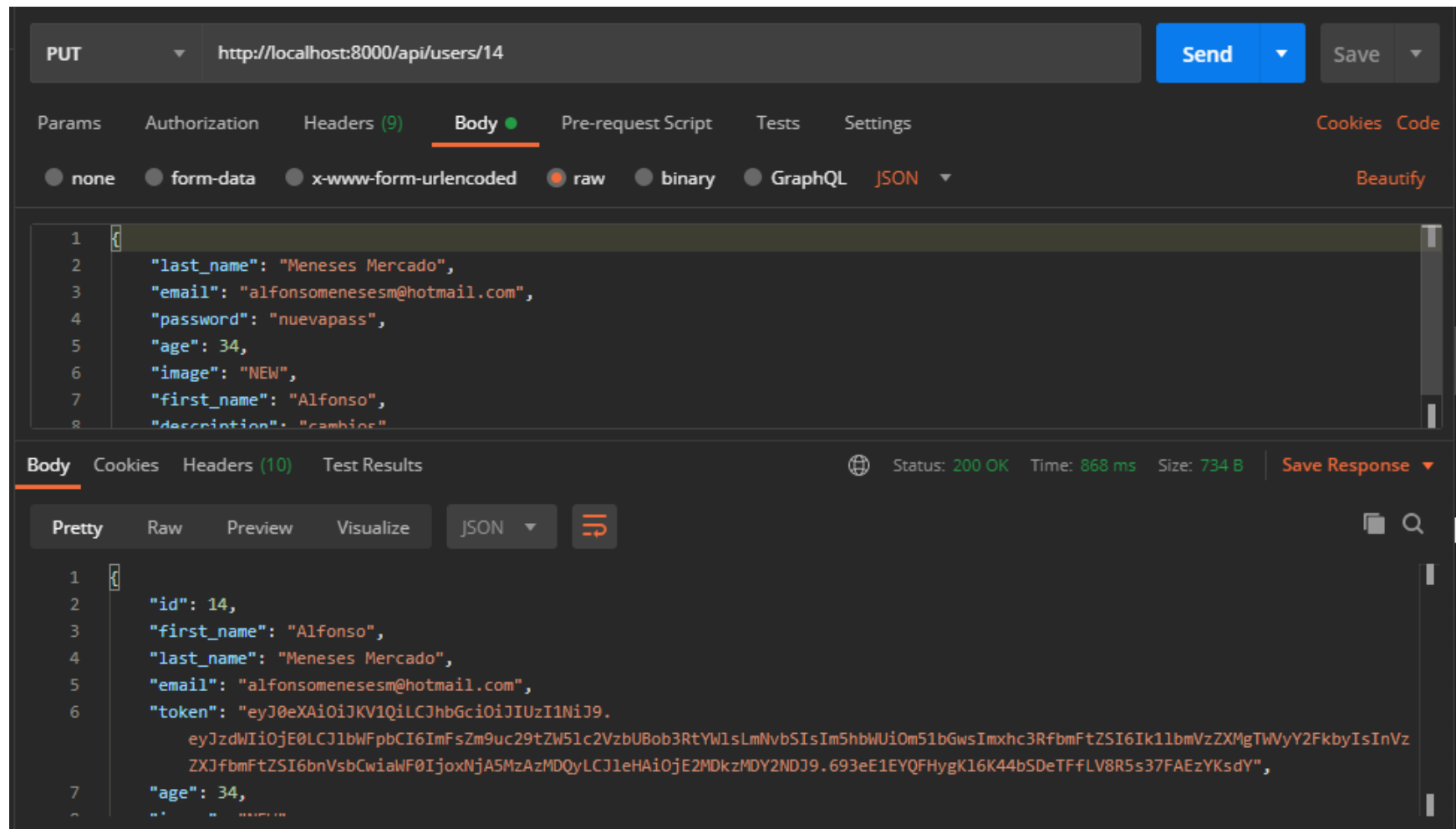
↺

📄

🔍

```
1 {
2   "error": "Invalid Auth token."
3 }
```

Edición general de usuario



Edición parcial de un usuario

The screenshot shows a REST client interface with a PATCH request to `http://localhost:8000/api/users/14`. The request body is a JSON object with `"age": 30`. The response is a JSON object with user details and the updated age.

Request:

```
PATCH http://localhost:8000/api/users/14
```

Body:

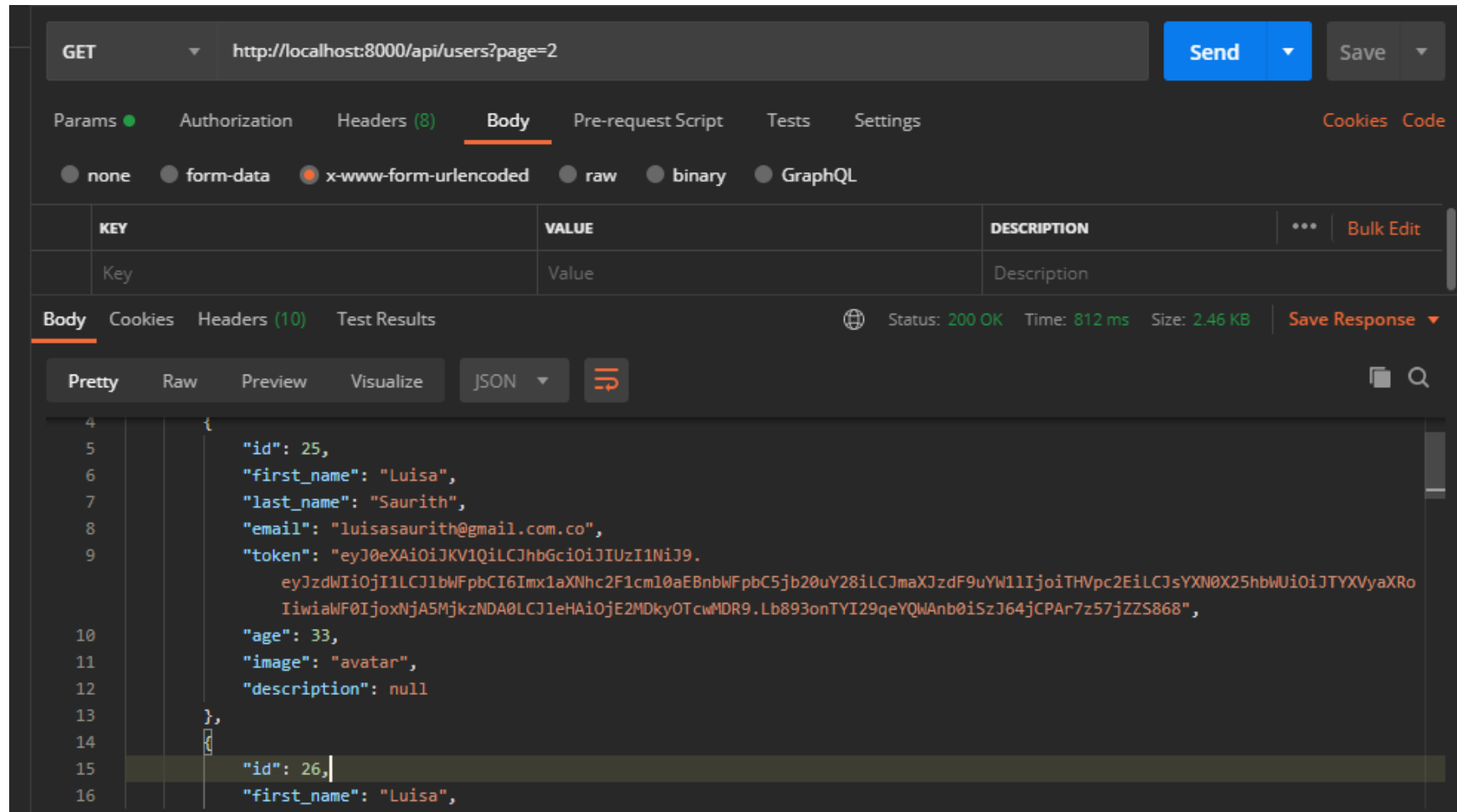
```
{
  "age": 30
}
```

Response:

```
{
  "id": 14,
  "first_name": "Alfonso",
  "last_name": "Meneses Mercado",
  "email": "alfonsomeneses@hotmail.com",
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiE0LCJlbWFPbCI6ImFsZm9uc29tZW51c2VzbUBob3RtYW1sLmNvbSI6Im5hbWUiOm51bGwsImxhc3RfbmFtZSI6Ikk1bmVzZXMGdWVyY2FkbyIsInVzZXJfbmFtZSI6bnVsbCwiaWF0IjojA5MzAzMDQyLCJleHAiOiE2MDkzMDY2NDJ9.693eE1EYQFHgK16K44bSDeTFfLV8R5s37FAEzYKsdY",
  "age": 34
}
```

Status: 200 OK **Time:** 762 ms **Size:** 734 B

Listar usuarios



Listar usuario

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/users/17`. The response is a JSON object representing a user.

Request:

- Method: GET
- URL: `http://localhost:8000/api/users/17`
- Body: x-www-form-urlencoded

Response:

- Status: 200 OK
- Time: 781 ms
- Size: 700 B

JSON Response:

```
{
  "id": 17,
  "first_name": "Luisa",
  "last_name": "Saurith",
  "email": "luisasaurith@gmail.com",
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiE3LCJlbWV4Imx1aXNhcn2F1cm10aEBnbWV4LCJ0eXN0X25hbWUiOiJTYXVyaXR0IiwidXN1c19uYW11IjpudWxsLCJpYXQiOiE2MDkyOTUwOTQsImV4cCI6MTYwOTI5ODY5NH0.4RXtjyiZSDEPqI4i4KfWYmygi9D-Rddq3_kzDoGhf2U",
  "age": 33,
  "image": "avatar",
  "description": null
}
```

Borrar usuario

The screenshot shows a REST client interface with a DELETE request to `http://localhost:8000/api/users/18`. The request is successful, returning a 200 OK status. The response body is a JSON object containing user information.

Request:

- Method: DELETE
- URL: `http://localhost:8000/api/users/18`
- Body: This request does not have a body

Response:

- Status: 200 OK
- Time: 825 ms
- Size: 696 B
- Save Response

Response Body (JSON):

```
{
  "id": 18,
  "first_name": "Luisa",
  "last_name": "Saurith",
  "email": "luisasaurith@hotmail.com",
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJE4LCjlbWFpbCI6Imx1aXNhcn2F1cm10aEBnaG90bWFpbC5jb20iLCJmaXJzdF9uYW11IjoithVpc2EiLCJzYXN0X25hbWUiOiJTYXVyaXRoIiwiaWF0IjoxNjA5MjkxOTI5LCJleHAiOiJE2MDkyOTU1MjF9.ZQKKDPHDG7yq7cC74eK8cIaMtziU1Tf8kSMdtF9DG4c",
  "age": 33,
  "image": "avatar",
  "description": null
}
```

Error 404

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

- Method:** GET
- URL:** http://localhost:8000/api/
- Buttons:** Send, Save
- Tabs:** Params, Authorization, Headers (8), **Body**, Pre-request Script, Tests, Settings, Cookies, Code
- Body Type:** x-www-form-urlencoded (selected), none, form-data, raw, binary, GraphQL
- Table:**

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		
- Body Tab:** Body, Cookies, Headers (9), Test Results
- Status:** 200 OK (green)
- Time:** 719 ms (green)
- Size:** 296 B (green)
- Buttons:** Save Response (dropdown arrow)
- Body View:** Pretty, Raw, Preview, Visualize, JSON (dropdown), Refresh icon
- Response Body (JSON):**

```
1 {  
2   "error": "Not found"  
3 }
```

No Content-Type

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

- Method:** POST
- URL:** http://localhost:8000/api/users
- Body Type:** x-www-form-urlencoded (selected)
- Status:** 403 Forbidden
- Time:** 770 ms
- Size:** 398 B

The response body is displayed in JSON format:

```
1 {
2   "error": "Request should have 'Content-Type' header with value 'application/json'"
3 }
```

KEY	VALUE	DESCRIPTION
Key	Value	Description

Sin Token

POST

http://localhost:8000/api/users

Send

Save

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Cookies

Code

Headers

8 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json				
<input type="checkbox"/>	Authorization	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWliOiJl...				
	Key	Value	Description			

Body

Cookies

Headers (10)

Test Results

Status: 401 Unauthorized

Time: 803 ms

Size: 348 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

{

"error": "Invalid Auth token."

}

Con token invalido o vencido

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

- Method:** POST
- URL:** http://localhost:8000/api/users
- Buttons:** Send, Save
- Tabs:** Params, Authorization, Headers (10), Body, Pre-request Script, Tests, Settings, Cookies, Code
- Headers Tab:** Shows 8 hidden headers. Visible headers include:

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json				
<input checked="" type="checkbox"/>	Authorization	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJl...				
	Key	Value	Description			
- Body Tab:** Shows the response status and details:
 - Status:** 401 Unauthorized
 - Time:** 724 ms
 - Size:** 348 B
 - Buttons:** Save Response
- Response Body:** Pretty, Raw, Preview, Visualize (JSON). The response is:

```
1 {  
2   "error": "Invalid Auth token."  
3 }
```

Indicaciones para correr la API y realizar pruebas

Requerimientos

Apache instalado con la versión de PHP en 7.3, 8.0 o superior.

Base de datos MySql o MariaDB.

En el modo de desarrollador, se necesita instalar el sistema de gestión de paquetes “Composer”, lo cual es requerido para instalar algunos paquetes, los cuales están en el archivo “composer.json”.

Link para instalar composer: <https://getcomposer.org/download/>

Configurar información requerida para conectarse con el servidor de base de datos, estas configuraciones se hacen en el archivo “.env”.

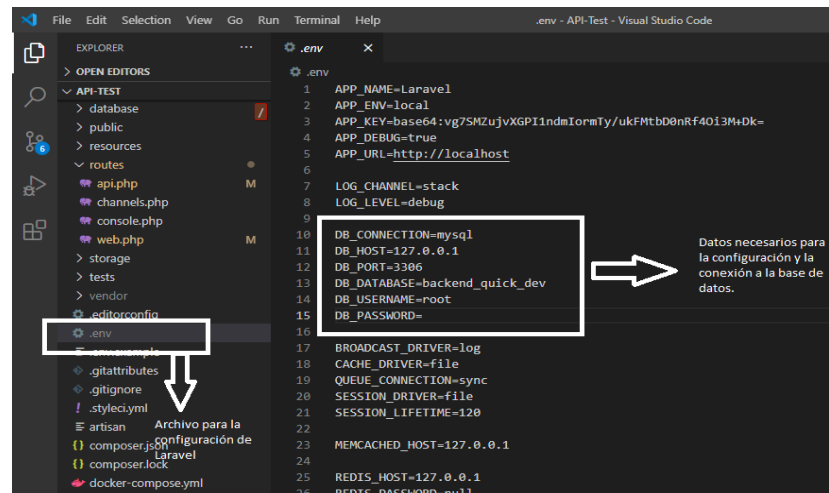
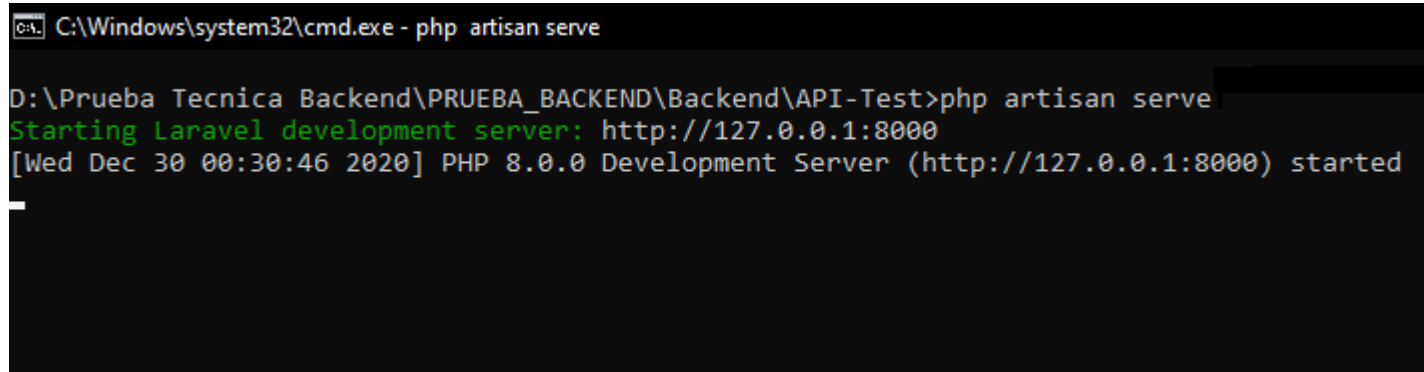


Imagen configuración conexión a la DB

Si al editar la configuración de la base de datos en el archivo “.env” y no se ha podido conectar, se debe limpiar la cache de configuración del proyecto. Esto se hace con el comando: “php artisan config:clear”.

Para la creación de la base de datos y la tabla “users”, se encuentra en el repositorio, en la carpeta DB, un archivo sql, en el cual se encuentra el script el cual realizara la creación de la base de datos y la tabla

Si se quiere hacer las pruebas o correr la API sin usar o instalar Apache (u otro servidor como Tomcat), se puede hacer desde la consola que utilice en su sistema operativo. Para esto es obligatorio haber instalado composer. Desde la consola se ejecuta el comando “php artisan serve”.



```
C:\Windows\system32\cmd.exe - php artisan serve

D:\Prueba Tecnica Backend\PRUEBA_BACKEND\Backend\API-Test>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Wed Dec 30 00:30:46 2020] PHP 8.0.0 Development Server (http://127.0.0.1:8000) started
```

API corriendo desde la consola