

Este documento brinda información detallada sobre la implementación de tokens JWT en un servidor Express y el desarrollo de un módulo de usuarios para expandir una API existente.

JWT en EXPRESS

Luis Pascal Barros Suarez

Introducción:

Implementación de un sistema de autenticación con nombre de usuario y contraseña para restringir el acceso a ciertas funciones de la API.

Existen dos rutas protegidas: una requiere iniciar sesión y la otra, además de iniciar sesión, requiere tener permisos de administrador.

Datos de usuario:

- nombre de usuario
- contraseña
- Administrador(Bolean)

Almacenamiento de los usuarios:

- Utilizamos MongoDB para almacenar información de los usuarios, incluyendo contraseñas cifradas con la librería Bcrypt.

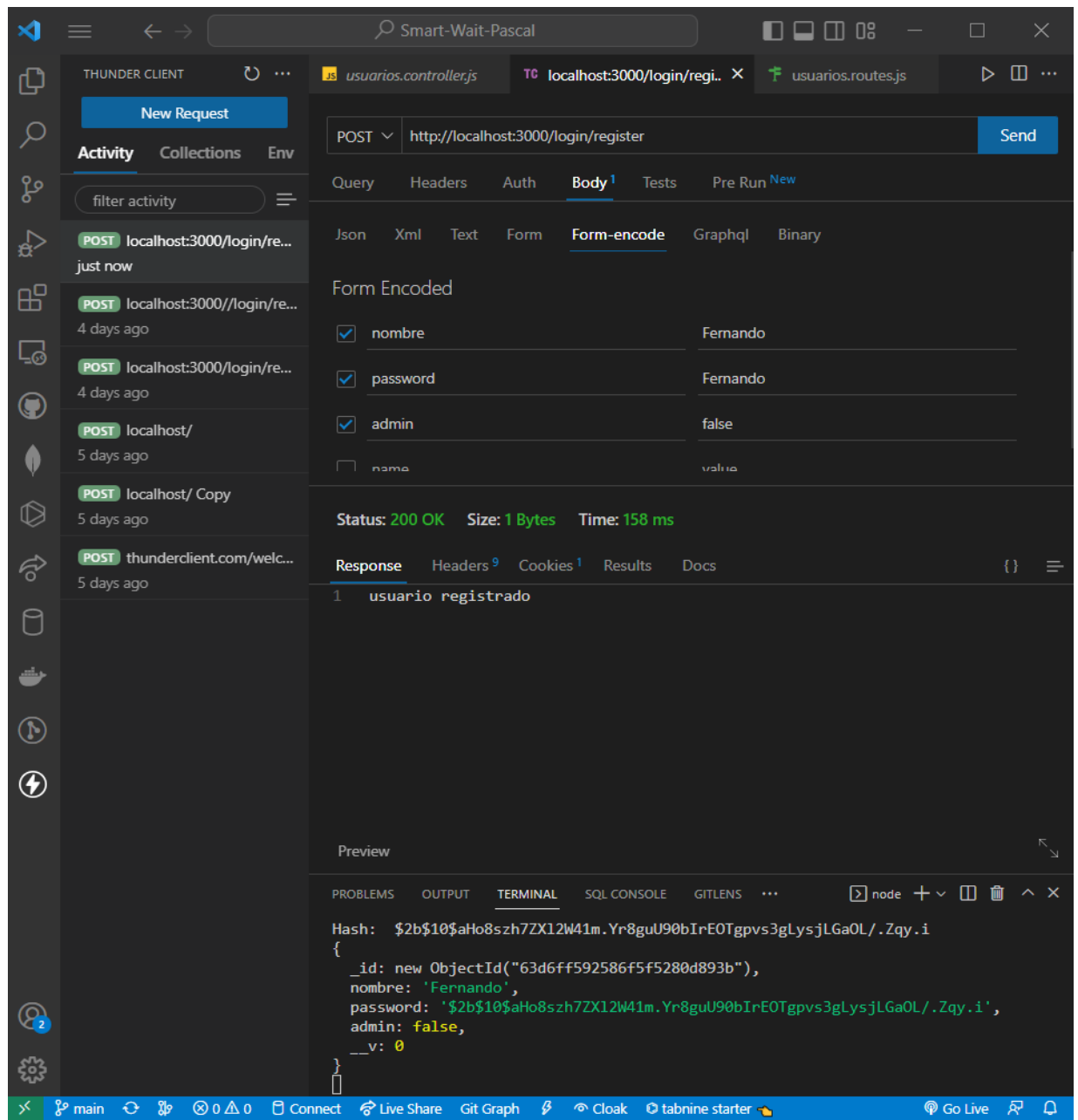
The screenshot displays the MongoDB Atlas web interface. On the left, a sidebar shows a search bar and a tree view of namespaces under the 'proyecto' folder, including 'servicios', 'tickets', 'usuarios' (highlighted), 'sample_mflix', and 'tutorial'. The main panel is titled 'Data Services' and shows the 'Find' tab for the 'usuarios' collection. At the top, it displays statistics: STORAGE SIZE: 36KB, LOGICAL DATA SIZE: 268B, TOTAL DOCUMENTS: 2, and INDEXES TOTAL SIZE: 36KB. Below this, there are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A filter bar is present with the text '{ field: 'value' }'. The query results section shows 'QUERY RESULTS: 1-2 OF 2'. The first document is for 'Pascal' with a hashed password and 'admin: true'. The second document is for 'Fernando' with a hashed password and 'admin: false'.

```
{
  "_id": ObjectId('63d19f60660c62a86a9359cf'),
  "nombre": "Pascal",
  "password": "$2b$10$Bv0QNFUuJB6a2nM9LRk3FuIpx5Y6J98hCkftf3oC5.2xrTPZqf80q",
  "admin": true,
  "__v": 0
}
```

```
{
  "_id": ObjectId('63d6ff592586f5f5280d893b'),
  "nombre": "Fernando",
  "password": "$2b$10$aHo8szh7ZXl2W41m.Yr8guU90bIrEOTgpvs3gLysjLGaOL/.Zqy.i",
  "admin": false,
  "__v": 0
}
```

Operaciones

- Se registran en la aplicación:



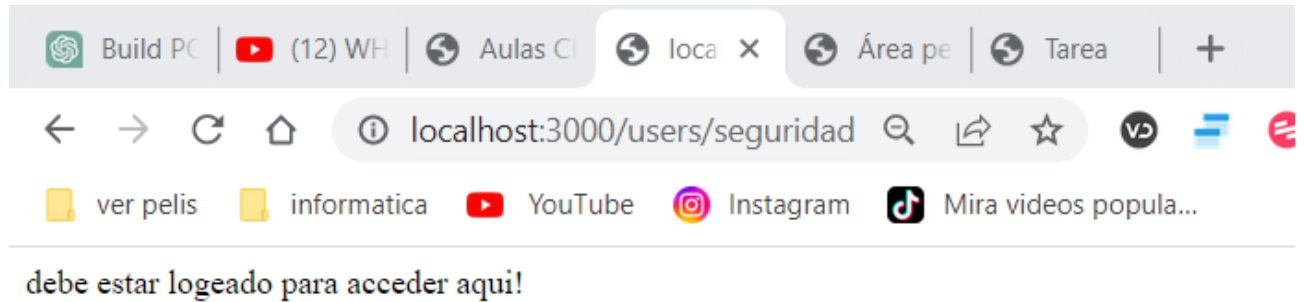
- **Login** (inicia una sesión si los datos de acceso son correctos)

```
Server is running on port 3000.
Server is connected to the database!
Ha entrado un cliente
[
  {
    _id: new ObjectId("63d19f60660c62a86a9359cf"),
    nombre: 'Pascal',
    password: '$2b$10$Bv0QNfUuJB6a2nM9LRk3FuIpx5Y6J98hCkftf3oC5.2xrTPZqf80q',
    admin: true,
    __v: 0
  }
]
Passwords match
```

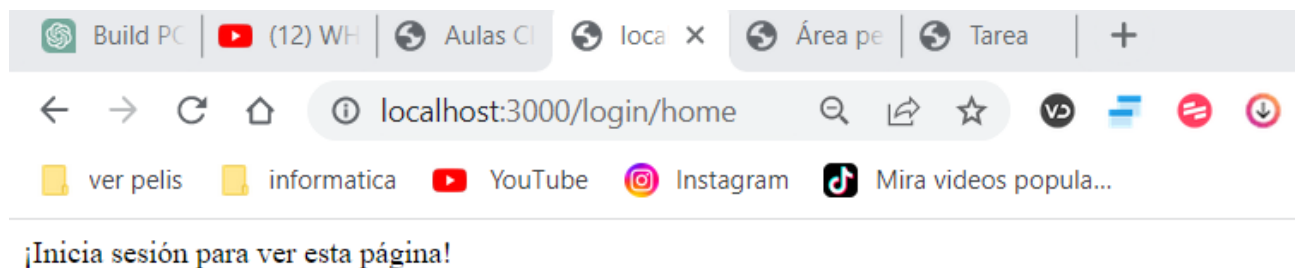
- **Logout** (elimina la información de sesión)

1. **Acceso denegado** a una operación 'privada' que necesite autenticación.

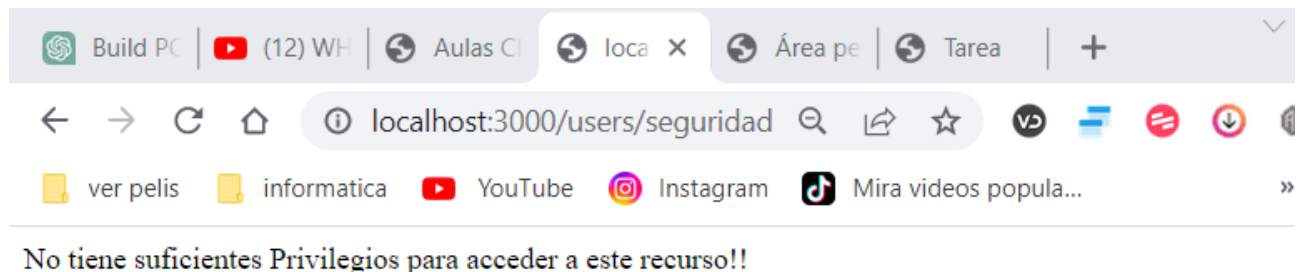
Para acceder a esta operación "privada" que sería users/seguridad que devuelve todos los usuarios administradores, es necesario estar autenticado y tener permisos de administrador:



Para acceder a esta operación "privada" que sería login/home es necesario estar autenticado:

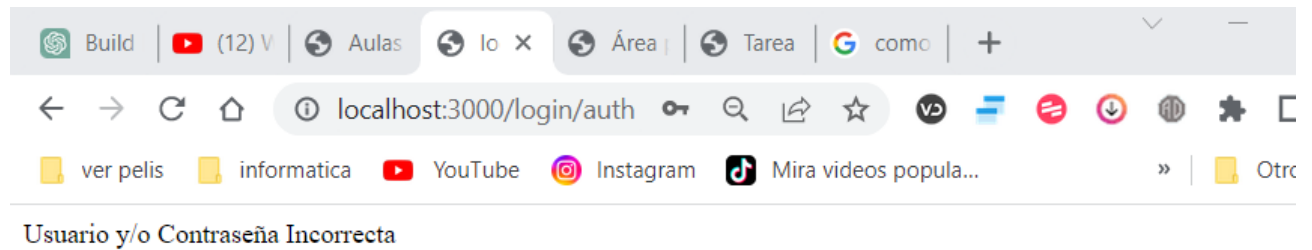


Aquí se muestra que un usuario que no es administrador no puede acceder al recurso users/seguridad:



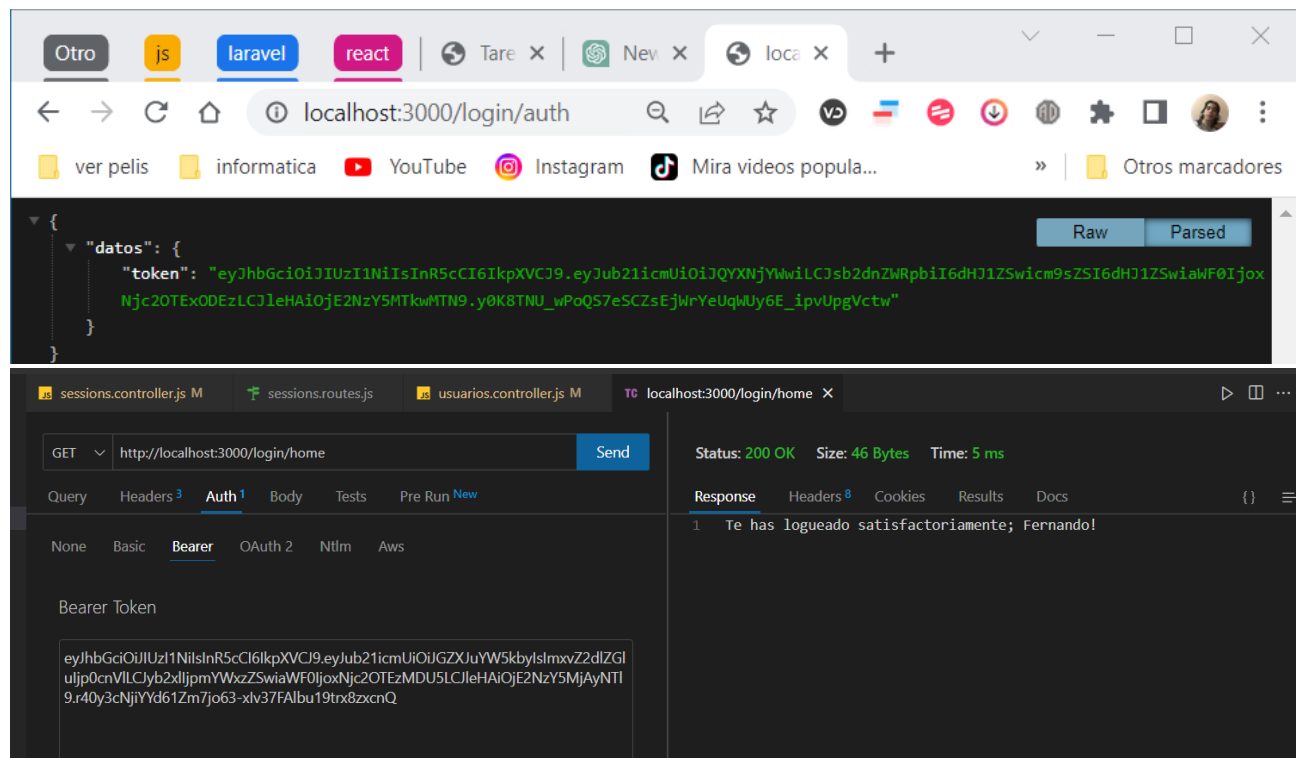
2. Intento de **login fallido**:

Aquí intentamos conectarnos con un usuario o contraseña que no es correcta:



3. **Login correcto**:

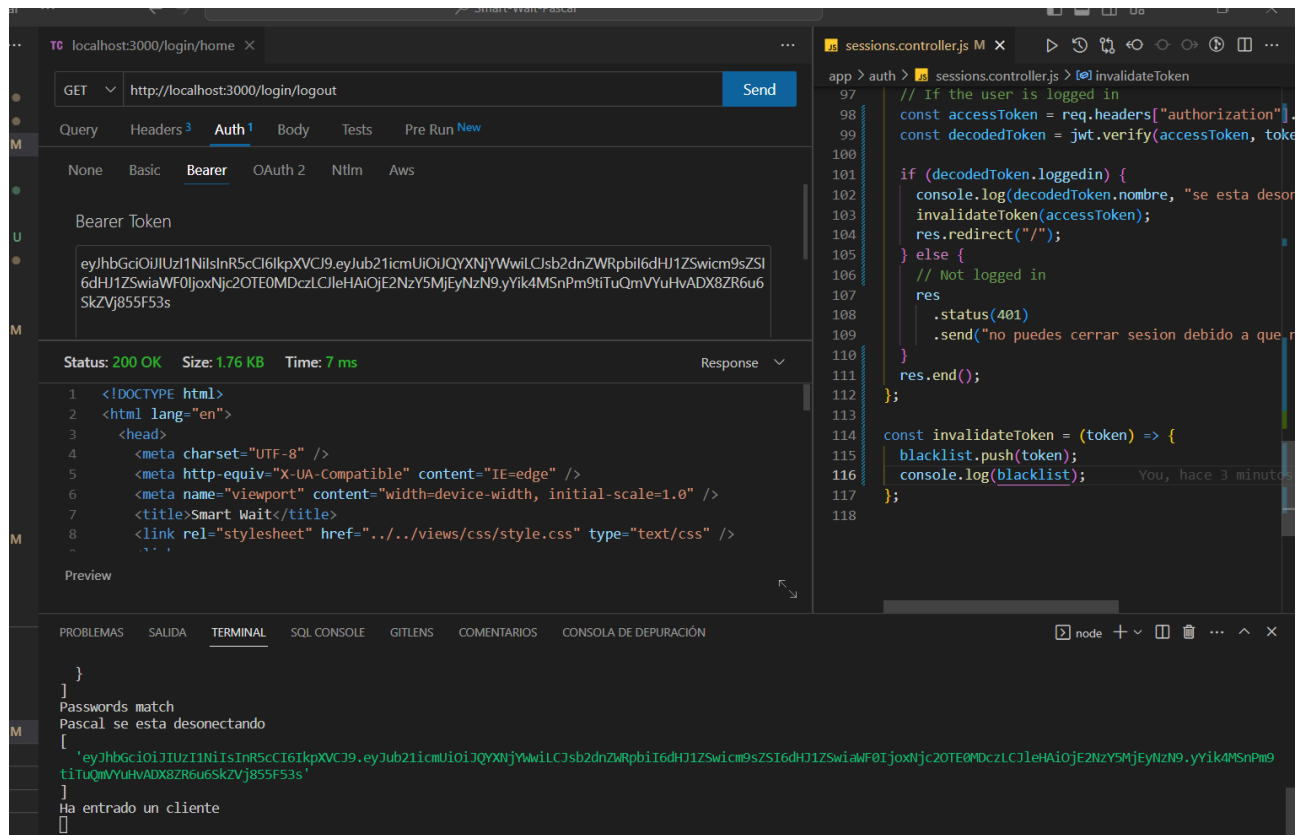
Aquí nos hemos conectado con el usuario Fernando y ya tenemos acceso al recurso login/home pero no a users/seguridad ya que no es admin:



The screenshot displays the Swagger UI interface for a REST API. The top navigation bar shows the API name 'sessions.controller.js M' and the user 'usuarios.controller.js M'. The main area is divided into two panels. The left panel shows the request details for the endpoint 'http://localhost:3000/users/seguiridad'. The request method is 'GET'. The request body is empty. The response status is '200 OK', the size is '149 Bytes', and the time is '101 ms'. The response body is a JSON object: { '_id': '63d19f60660c62a86a9359cf', 'nombre': 'Pascal', 'password': '\$2b\$10\$8v0QWfuuJB6a2nM9LRk3FuIpx5Y6J98hCkftf3oC5.2xrTPZqf80q', 'admin': true, '__v': 0 }. The right panel shows the response details, including the status '200 OK', size '149 Bytes', and time '101 ms'. The response body is a JSON object: { '_id': '63d19f60660c62a86a9359cf', 'nombre': 'Pascal', 'password': '\$2b\$10\$8v0QWfuuJB6a2nM9LRk3FuIpx5Y6J98hCkftf3oC5.2xrTPZqf80q', 'admin': true, '__v': 0 }.

5. Logout.

aquí nos desconectamos del servidor e intentamos acceder la misma operación de antes acceder al recurso login/home



6. **Prueba de acceso a la misma operación ‘privada’** después de hacer logout. Debería dar un mensaje de acceso denegado o algo así.

The image shows a REST client interface with a dark theme. At the top, the URL bar shows 'localhost:3000/login/home'. The request method is 'GET'. The 'Auth' tab is selected, showing 'Bearer' authentication. The token is a long alphanumeric string. The 'Response' tab shows a status of '401 Unauthorized', size of '26 Bytes', and time of '5 ms'. The response body is a JSON object: {"error": "Expired Token"}. At the bottom, there is a terminal window showing a Node.js script running. The script defines a JWT token and a function to log in a client. The terminal output shows the token being logged and the client being logged in.