Niharika Singh Seguir
Entonces, ¿puedes paradigma?
27 de enero · 6 minutos de lectura

Cómo crear una API RESTful con autenticación en 5 minutos, todo desde tu línea de comando

Si el título de este artículo te entusiasma, amigo mío, estás a punto de alcanzar el **nivel 100** de satisfacción hasta el final. Iré rápidamente por el curso de este artículo:

1. **Lo que estamos a punto de crear:** API RESTful que maneja registros de alimentos en el menú de un restaurante. La base de datos utilizada en el back-end será MongoDB. (Literalmente puede usar cualquier base de datos aleatoria en este planeta. Hay una lista exhaustiva de conectores de bases de datos / conectores no basados en bases de datos compatibles con LoopBack a continuación).

2. **Qué es LoopBack:** en términos extremadamente simples, es un framework Node.js altamente extensible y de código abierto que se usa para crear API REST dinámicas y de extremo a extremo muy rápidamente. Las API generadas a través de LoopBack son Swagger API (el marco de API más popular del mundo, y verás por qué muy pronto). El front-end se puede hacer en cualquier marco de lo que estés enamorado; Angular o React.

3. **Creación de aplicaciones a través de CLI:** esta es la parte WOW que elimina toda la programación involucrada. LoopBack CLI es tan hermoso que todas las horas de trabajo de desarrollo se reducen a segundos. Aquí, estaríamos configurando nuestra base de datos usando CLI.

4. **Crear modelos de datos a través de CLI:** Nuevamente, sin programación. Todo a través de la hermosa CLI.

5. **Configuración de la autenticación a través de la CLI:** si tiene experiencia en la creación de API, ya sabe lo difícil que es restringir partes de la API mediante la autenticación. Configurar la autenticación basada en token usando Express + Node.js en el lado del servidor es un problema. ¡Todo ese dolor será quitado probando el elixir de LoopBack! Es la bebida del cielo.

Guía paso por paso:

Requisitos **previos:** asegúrese de que tiene instalado Node.js , Robomongo y el servidor MongoDB.

PASO 1: Instalar LoopBack CLI a través de NPM

Abra el terminal y escriba el siguiente comando para instalar LoopBack CLI para poder acceder al comando 'lb'. Solo a través del comando 'lb' podemos generar aplicaciones, modelos, fuentes de datos, etc. Para obtener más información: https://loopback.io/doc/en/lb2/Command-line-tools.html#using-yeoman

```
$ npm install -g loopback-cli
```

Asegúrate de instalarlo globalmente, o bien el comando 'lb' podría no funcionar para ti.

PASO 2: Creando la aplicación

Haga un directorio donde desea almacenar su proyecto. Lo llamaré 'restaurante-menú'. Asegúrate de haber abierto este directorio en tu terminal para que todos los archivos generados a través de LoopBack estén almacenados en esa carpeta.

Luego ingrese el siguiente comando:

```
$ lb
```

Se harán muchas preguntas, como las que se muestran en la imagen a continuación.

```
Niharikas-MacBook-Pro:restaurant-menu niharikasingh$ lb
? What's the name of your application? restaurant-menu
? Which version of LoopBack would you like to use? 3.x (current)
? What kind of application do you have in mind? api-server (A LoopBack API serve
r with local User auth)
Generating .yo-rc.json

I'm all done. Running npm install for you to install the required dependencies. If this fails, try running the command yourself.

   create .editorconfig
   create .eslintignore
   create .eslintrc
   create server/boot/root.js
   create server/middleware.development.json
   create server/middleware.json
   create server/server.js
   create README.md
   create server/boot/authentication.js
   create .gitignore
   create client/README.md
npm WARN deprecated nodemailer@2.7.2: All versions below 4.0.1 of Nodemailer are deprecated. See https://nodemailer.com/status/
npm notice created a lockfile as package-lock.json. You should commit this file.
added 502 packages in 23.551s

Next steps:

  Create a model in your app
    $ lb model

  Run the app
    $ node .

The API Connect team at IBM happily continues to develop,
support and maintain LoopBack, which is at the core of
API Connect. When your APIs need robust management and
security options, please check out http://ibm.biz/tryAPIC

Niharikas-MacBook-Pro:restaurant-menu niharikasingh$
```

Esto es lo que debería ser.

(Para navegar entre las opciones, use las teclas de flecha en su teclado)

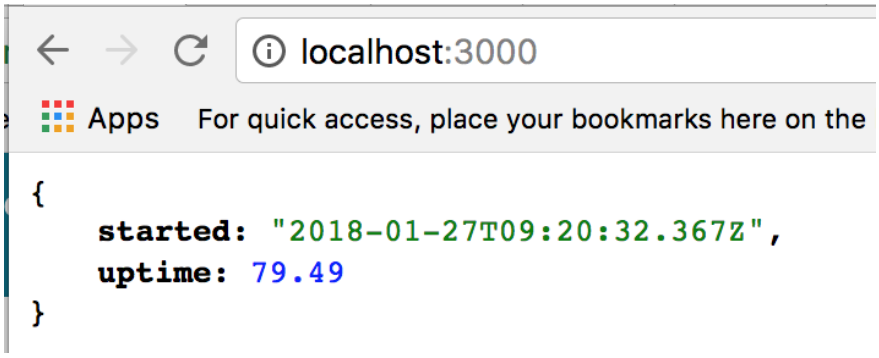
¡LA API SE CREA!

No estoy bromeando. No me creas? Ejecute la aplicación con el siguiente comando:

```
$ nodo.
```

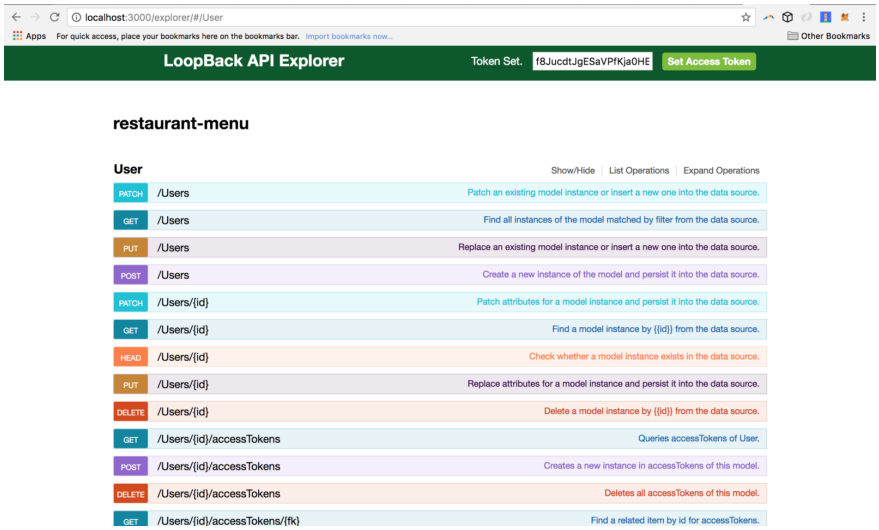
```
Niharikas-MacBook-Pro:restaurant-menu niharikasingh$ node .
Web server listening at: http://localhost:3000
Browse your REST API at http://localhost:3000/explorer
```

Si apunta a localhost: 3000, verá algo como esto:



Esto solo indicará cuándo se inició la API y cuántos segundos ha pasado.

Sin embargo, si vas a localhost: 3000 / explorer, verás el magnífico SwaggerAPI.



LoopBack ha configurado todas las rutas para usted:

GET usuarios, POST usuarios, PUT usuarios, ELIMINAR usuarios, Iniciar sesión, Cerrar sesión, Cambiar contraseña. Literalmente todo! De lo contrario, tomaría horas de trabajo codificar esto.

Abra esta carpeta en cualquier editor de texto. Estaría usando Atom.

PASO 3: conectando MongoDB

Si abre `datasources.json` en la carpeta Servidor, debería ver algo como:

```
{
  "db": {
    "nombre": "db",
    "conector": "memoria"
  }
}
```

Esto significa que actualmente, la fuente de datos que se utiliza es la memoria de nuestra computadora. Tenemos que cambiar esto a Mongo. Así que instalemos el conector mongo:

```
$ npm install --save loopback-connector-mongodb
```

Al lado, espero que mongod se esté ejecutando. Así es como sabrías que se está ejecutando:

```
2018-01-27T15: 01: 13.278 + 0530 I RED [thread1] esperando
conexiones en el puerto 27017
```

Ahora, conectemos el conector!

```
$ lb datasource mongoDS --connector mongoDB
```

Esto hará muchas preguntas de la siguiente manera:

```
Niharikas-MacBook-Pro:restaurant-menu niharikasingh$ lb datasource mongoDS --connector mongoDB
? Enter the datasource name: mongoDS
? Select the connector for mongoDS: MongoDB (supported by StrongLoop)
? Connection String url to override other settings (eg: mongodb://username:passw
ord@hostname:port/database):
? host: localhost
? port: 27017
? user:
? password:
? database: food
Niharikas-MacBook-Pro:restaurant-menu niharikasingh$
```

Ahora modifíquelo `datasources.json` porque no deseamos usar memoria. Deseamos usar Mongo.

```
{
  "db": {
    "host": "localhost",
    "port": 27017,
    "url": "",
    "base de datos": "comida",
    "contraseña": "",
    "nombre": "mongoDS",
    "usuario": "",
    "conector": "mongodb"
  }
}
```

Entonces nuestra base de datos llamada: `food` se crea.

PASO 4: Creando modelos de datos

Ejecute el siguiente comando para crear modelos de datos:

```
$ lb modelo
```

```
Niharikas-MacBook-Pro:restaurant-menu niharikasingh$ lb model
? Enter the model name: dishes
? Select the datasource to attach dishes to: db (mongodb)
? Select model's base class PersistedModel
? Expose dishes via the REST API? Yes
? Custom plural form (used to build REST URL):
? Common model or server only? common
Let's add some dishes properties now.

Enter an empty property name when done.
? Property name: name
   invoke   loopback:property
? Property type: string
? Required? Yes
? Default value[leave blank for none]:

Let's add another dishes property.
Enter an empty property name when done.
? Property name: price
   invoke   loopback:property
? Property type: number
? Required? Yes
? Default value[leave blank for none]:

Let's add another dishes property.
Enter an empty property name when done.
? Property name:
Niharikas-MacBook-Pro:restaurant-menu niharikasingh$
```

Puede agregar tantas propiedades a un modelo en particular. Para dejar de ingresar más propiedades, simplemente presione Entrar para salir de la CLI.

Consulte `dishes.json` en la carpeta Common / Models.

```
{
  "nombre": "platos",
  "base": "PersistedModel",
  "idInjection": true,
  "opciones": {
    "validateUpsert": true
  },
  "properties": {
    "name": {
      "type": " string ",
      " required ": true
    },
    " price ": {
      " type ":" number ",
      " required ": true
    }
  },
  " validations ": [],
  " relations ": {},
  " acls ": [ ],
  "métodos": {}
}
```

También puede editar las propiedades de este archivo json. No es necesario usar CLI.

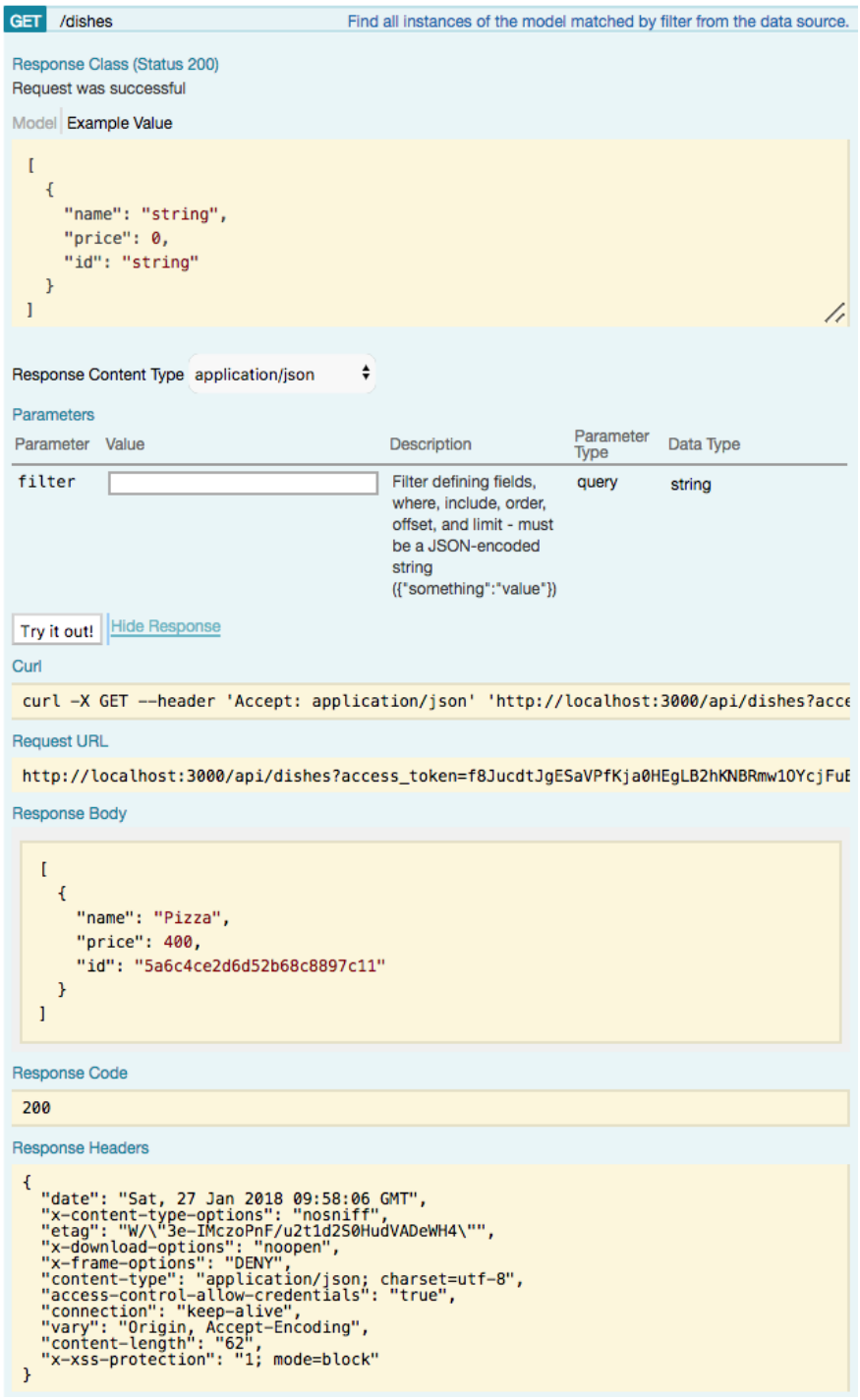
Ahora volvamos a ejecutar el servidor con el siguiente comando y diríjase a localhost: 3000 / explorer

```
$ nodo.
```

Ahora verá 2 modelos: `dishes` y `user`

Ahora vamos a publicar algunos `dish` .

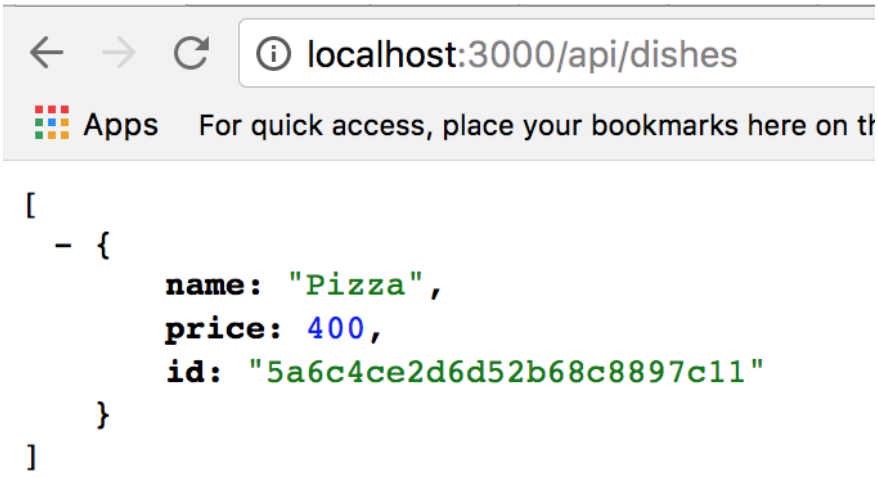
Ahora obtengamos lo mismo `dish` .



¡También puedes jugar con otras solicitudes HTTP!

También se puede acceder a estas API fuera del explorador:

http: // localhost: 3000 / api / platos



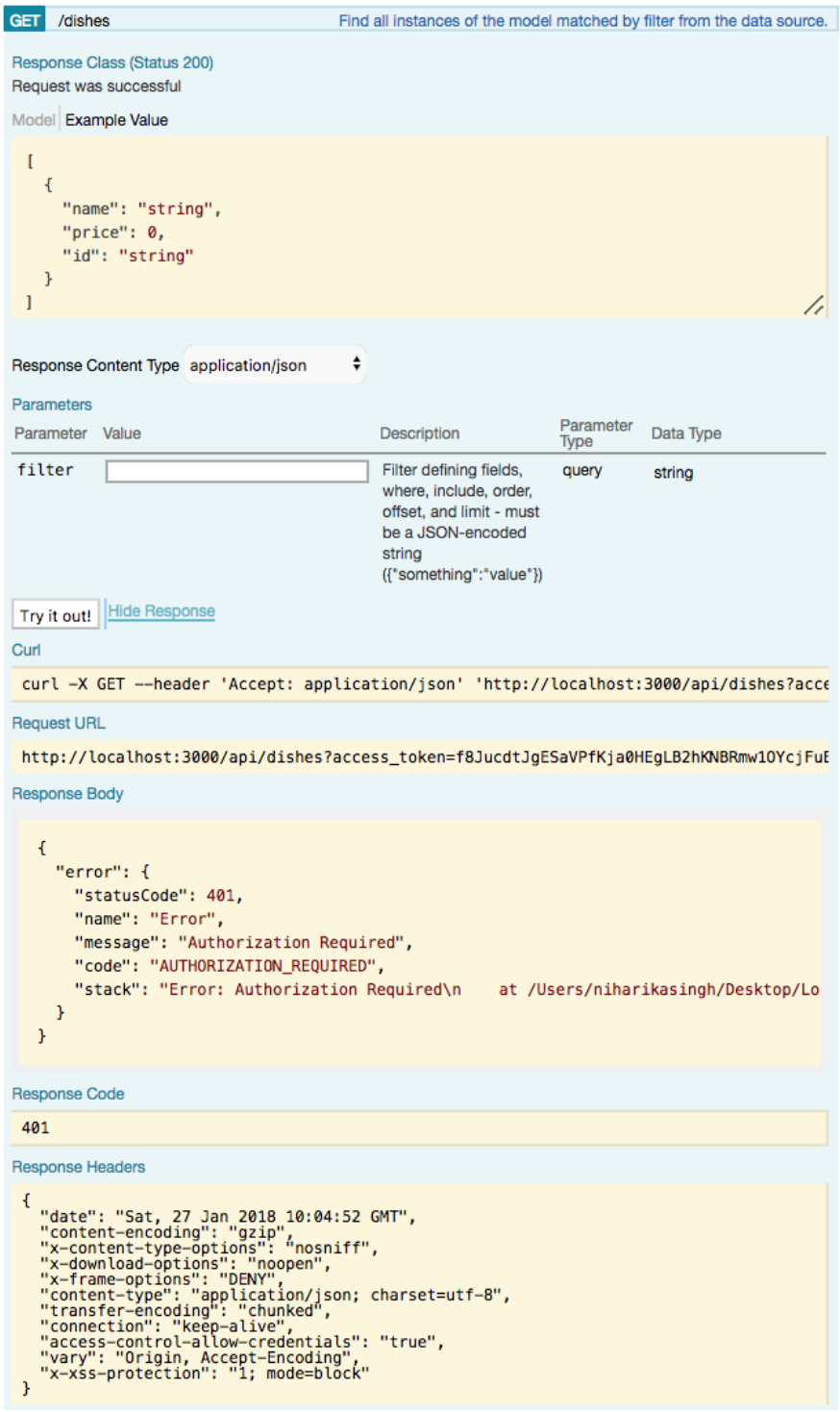
PASO 5: AUTENTICACIÓN: ¡Cereza en el pastel!

Para configurar la autenticación, ejecute el siguiente comando:

```
$ lb acl
```

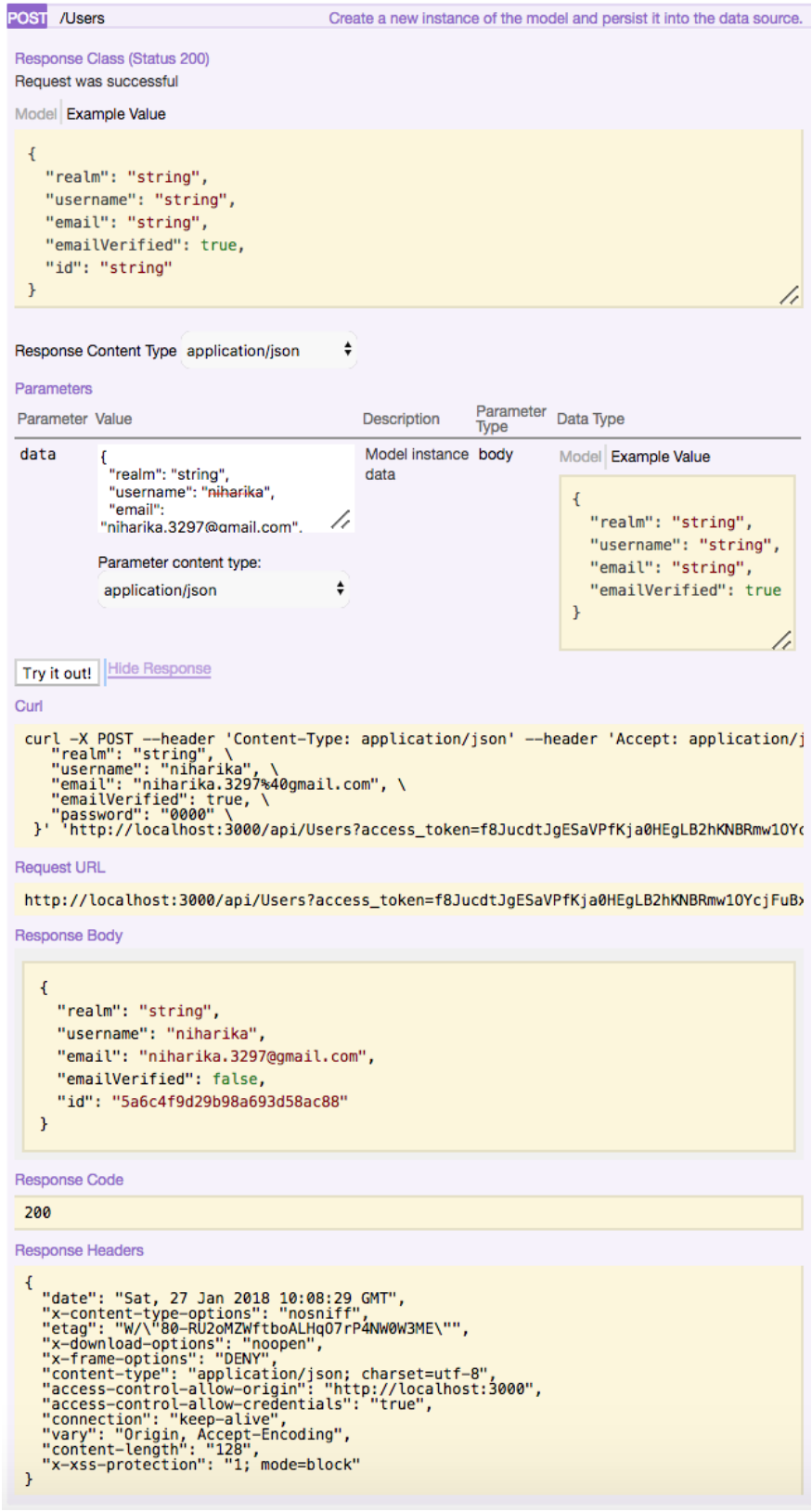
```
Niharikas-MacBook-Pro:restaurant-menu niharikasingh$ lb acl
[? Select the model to apply the ACL entry to: (all existing models)
[? Select the ACL scope: All methods and properties
? Select the access type: All (match all types)
? Select the role Any unauthenticated user
? Select the permission to apply Explicitly deny access
[Niharikas-MacBook-Pro:restaurant-menu niharikasingh$
```

Ahora, intentemos OBTENER el `dishes` . Antes de eso, vuelva a ejecutar el servidor.

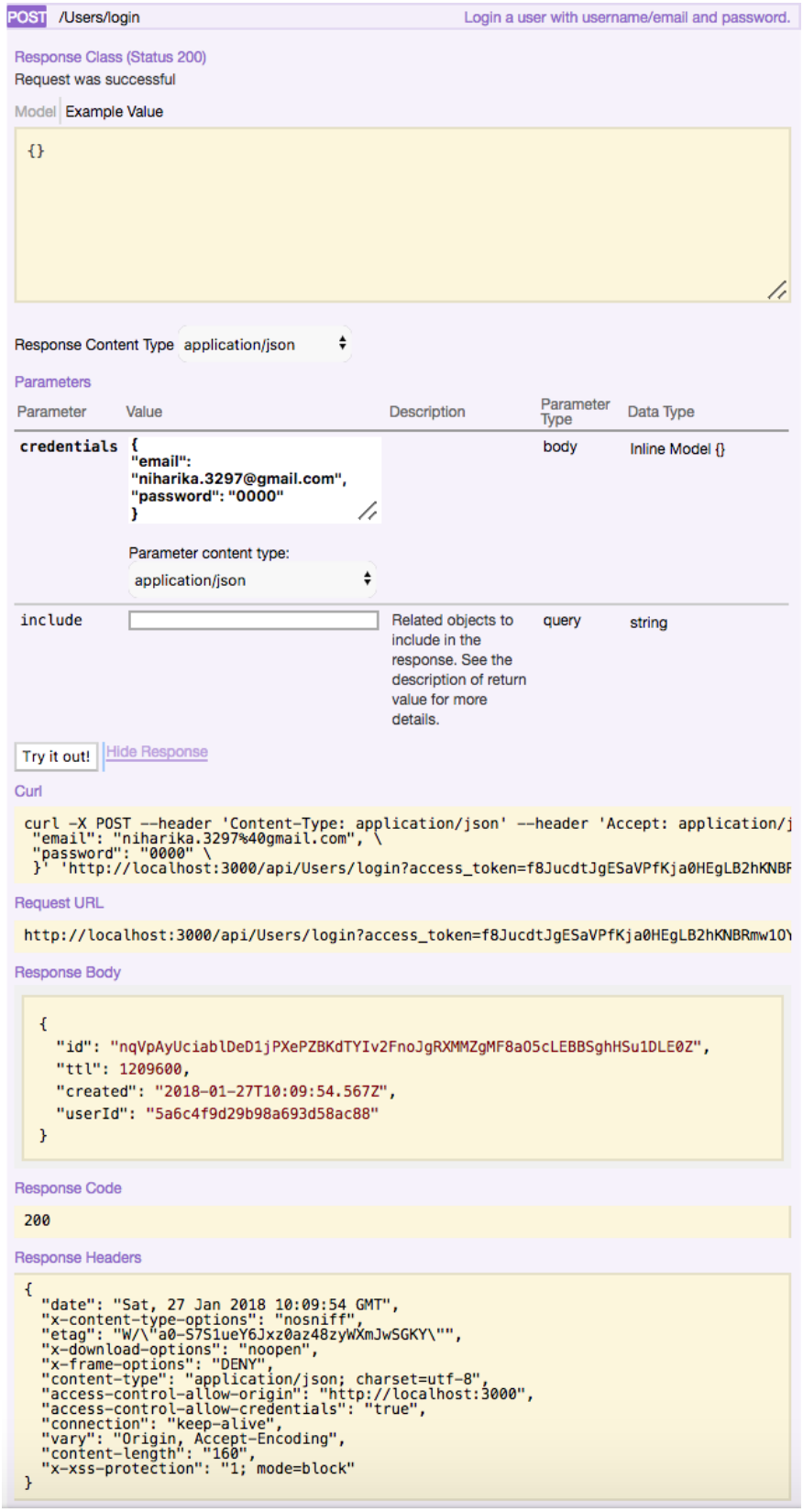


Por lo tanto, NO CONSIGUE ningún plato. Como se esperaba. Porque no hemos iniciado sesión. Nos alerta al decir que se requiere autenticación.

¡Seamos autenticados! Para eso, tenemos que registrarnos primero. Así que su mensaje en `users` .



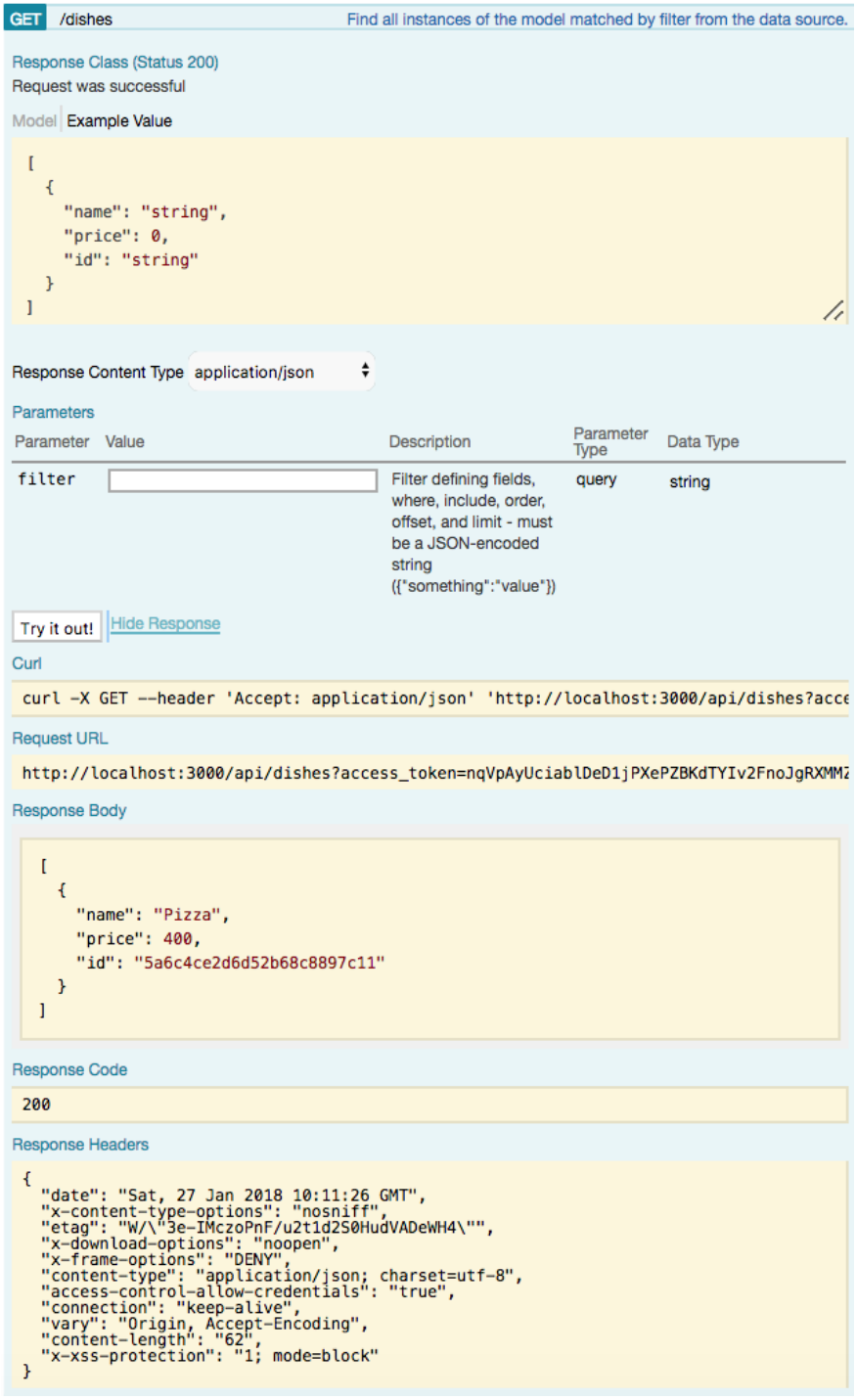
Ahora, inicie sesión.



Ahora, copie la ID en el cuerpo de la respuesta y péguelo en el campo Token de acceso en la parte superior de la página.

Ahora estamos autenticados. HURRA.

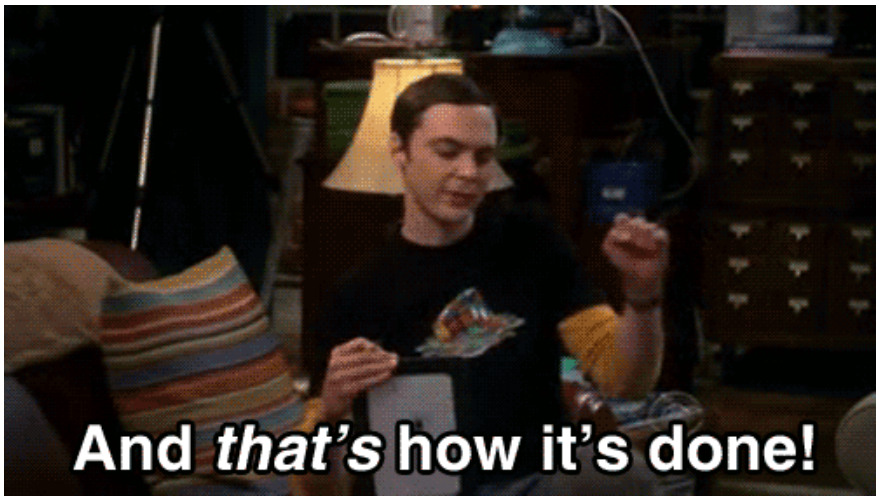
Ahora, obtengamos el `dishes` nuevo.



¡HOORAY!

Felicidades si has alcanzado este paso con éxito. Muy orgulloso de ti

Los próximos pasos serían crear un front-end alrededor de esta API que se haría más tarde.



Adiós amigos!
Feliz codificación.