# DOCUMENTACIÓN API CHAT

# Endpoints

## Recursos

A continuación, listare los Endpoints que contiene el proyecto. Primero iré detallando los recursos.

* Profiles:
* Users
* Chatrooms
* Messages

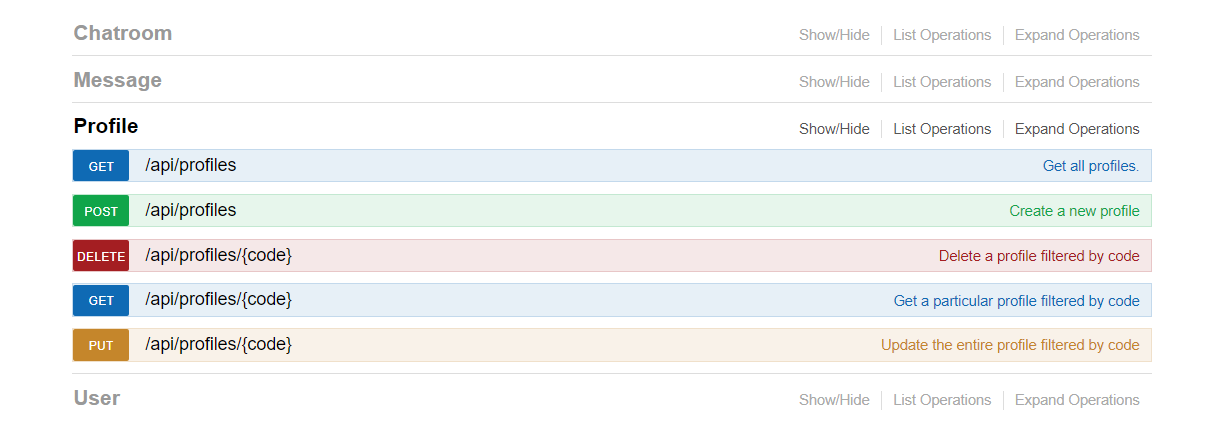
Todas a excepto del recurso Messages tiene la habilidad de crear, actualizar, eliminar y obtener recursos. En el caso del recurso Messages solo tiene el crear, eliminar y obtener por motivos de simplicidad.

## Operaciones por recursos

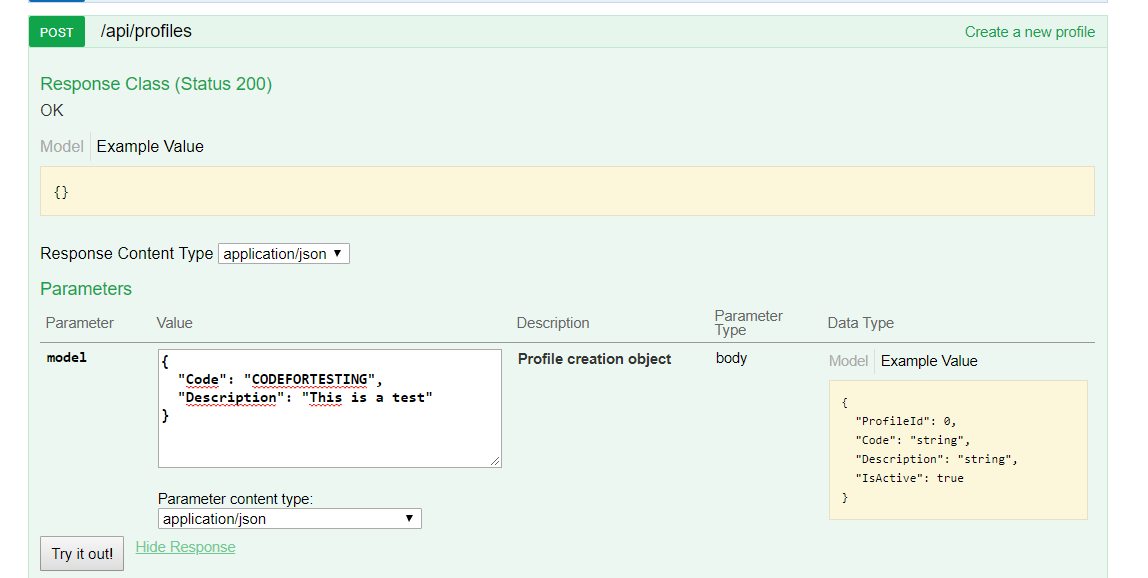
Estos datos pueden ser visualizado en el swagger que está integrado en la aplicación a través de la siguiente url: [***http://localhost:<port>/swagger/ui/index***](http://localhost:%3cport%3e/swagger/ui/index)***.*** En ella, se podrá observar todas las operaciones y su respectiva descripción. También se observa los inputs y output, los status code te ellos retornan e , incluso, puede realizar pruebas desde swagger ui.

Ejemplo:

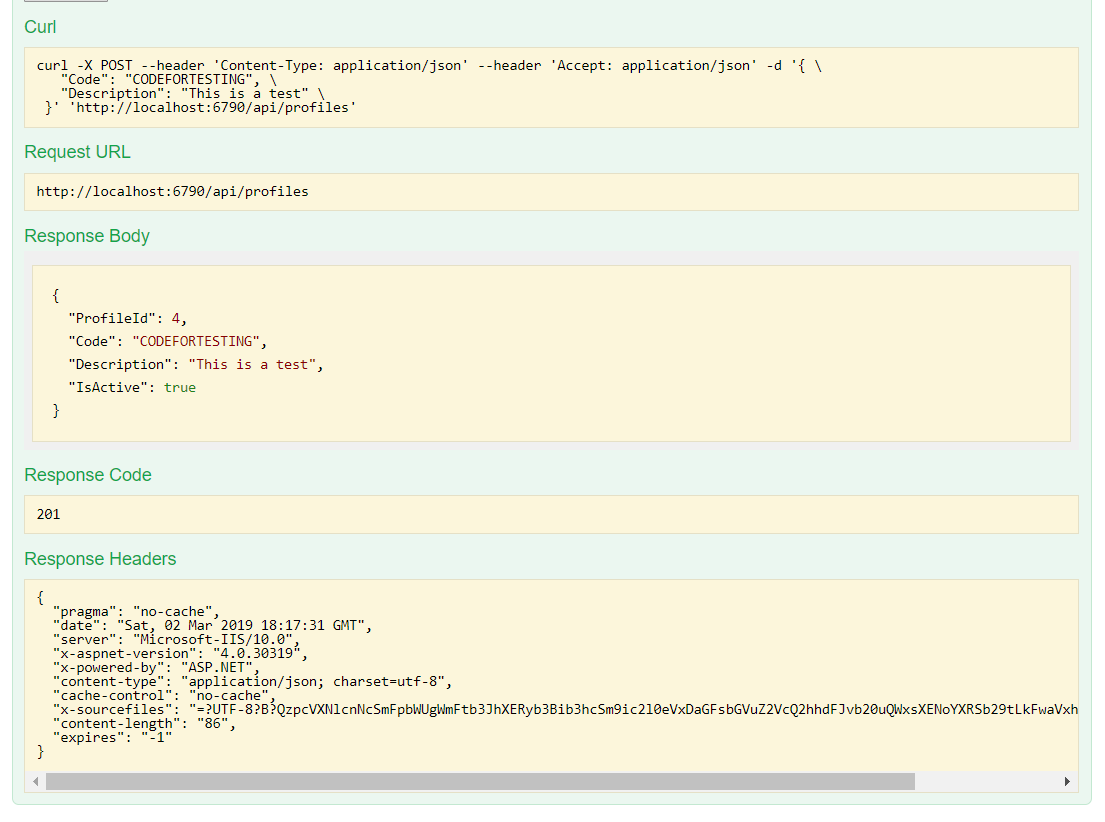
Lista de operaciones del recurso Profile



Ejecución del método post



Response del método post



## Detalles de los campos

Para las operaciones de cada recurso, se está utilizando un único view model. Es decir, tanto para el get, create o update se está utilizando el mismo objeto a pesar de que no todos sus atributos son requeridos en algunas operaciones. Es recomendable tener un view model por operación y usar data annotation para que se vea en swagger también. En este caso, también por cuestión de simplicidad, se apuntó a un solo objeto.

A continuación, de detallaras los campos obligatorios y opciones de las operaciones por recurso:

**Profiles**

* Create
  + Obligatorio:
    - Code: código único del perfil
  + Opcional
    - Description: alguna descripción del perfil
* Put
  + Opcional
    - Description: alguna descripción del perfil
    - IsActive: determina si el perfil está activo o no
* Delete
  + Obligatorio:
    - Code: código único del perfil (en la url)
* GetByCode
  + Obligatorio:
    - Code: código único del perfil (en la url)
* GetAll: Sin input

**User**

* Create
  + Obligatorio:
    - Nickname: identificador del usuario
    - Code: código único del perfil
  + Opcional
    - Name: nombre del usuario
    - Lastname: apellido del usuario
* Put
  + Obligatorio:
    - Nickname: identificador del usuario (en la url)
    - Code: código único del perfil
  + Opcional
    - Name: nombre del usuario
    - Lastname: apellido del usuario
* Delete
  + Obligatorio:
    - Nickname: identificador del usuario (en la url)
* GetByNickname
  + Obligatorio:
    - Nickname: identificador del usuario (en la url)
* GetAll: Sin input

**Chatroom**

* Create
  + Obligatorio:
    - Owner: nickname del usuario
    - Name: nombre del chatroom
  + Opcional
    - Description: alguna descripción acerca del chatroom
* Put
  + Obligatorio:
    - Id: Chatroom id en la url
    - Owner: nickname de usuario (admin del Chatroom)
    - Name: nombre del chatroom
  + Opcional
    - Description: alguna descripción acerca del chatroom
    - IsActive: determina si el Chatroom está habilitado o no
* Delete
  + Obligatorio:
    - Id: id del Chatroom (en la url)
* GetById
  + Obligatorio:
    - Id: id del Chatroom (en la url)

**Messages**

* Create
  + Obligatorio:
    - Content: mensaje a enviar
    - Date: fecha del envió del mensaje
    - ChatroomId: Id del Chatroom a dónde va el mensaje
    - UserSender: nickname del usuario que envía el mensaje
* Get
  + Obligatorio:
    - Id: id del Chatroom (en la url)
* GetHistory
  + Obligatorio:
    - Id: id del Chatroom (en la url)

# Funcionamiento

## Scope

* Los usuarios pueden crear Chatrooms. Ya que no hay validaciones respecto a la cantidad pues pueden crear N.
* Los usuarios pueden mandar mensajes a cualquier Chatroom creado. No se pidió validaciones o, dicho de otro modo, no hay un requerimiento explicito que pida solicitar primero permiso para pertenecer al chat y, posteriormente, enviar mensajes. Actualmente, desde el api, basta con enviar el id del Chatroom al que se quiere enviar el mensaje.
* Cuando un usuario envía un texto como “/stock=APP”, se guardará en una cola y un job lo leerá eventualmente para que este envíe un mensaje personalidad que lo obtiene desde un archivo CSV. Bajo este escenario, será el bot el emisor más no ningún usuario.
* Los mensajes no pueden ser eliminados ni actualizados.
* Todos los mensajes tienen un emisor identificado con el nickname del usuario.

## Out of Scope

* Validaciones para unirser a un chat.
* Validaciones del contenido del mensaje a excepto de “/stock=APP” el cual tiene un comportamiento explicado en SCOPE.
* No hay forma de revertir el contenido de los mensajes ni eliminarlo.
* No hay notificación a los usuarios si hay nuevos mensajes en los Chatrooms.

# Conclusiones

El Chatroom api es simple sin mayor complejidad debido al tiempo dato en este challenge. Es importante mencionar que el tiempo también se invirtió en armar la arquitectura de la solución la cual tiene: Mappers, Dependency Injection, Unit test, Brokers, entre otros.

Gracias 😊