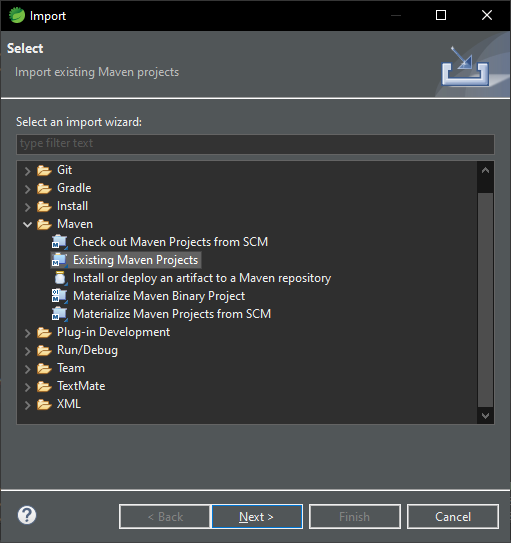
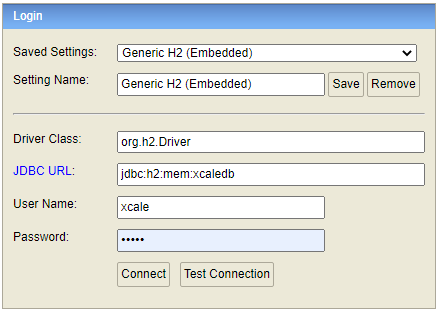
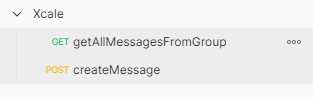
* Descargarse la carpeta **Fernando-Aramayo** del repositorio de GIT y luego importar como proyecto de Maven. En eclipse **File->Import**



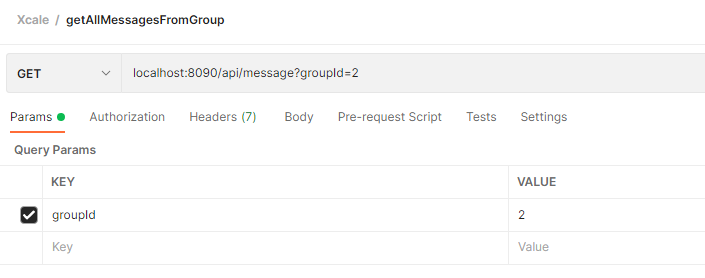
* La aplicación está configurada para ejecutarse en el puerto 8090, localhost:8090/
* La aplicación usa la base de datos H2 y se configuro para ejecutarse como base de datos en memoria, para simplificar las pruebas.
* Para ingresar en el administrador de la BD: [**http://localhost:8090/h2-console**](http://localhost:8090/h2-console)  
  user: xcale pass: xcale



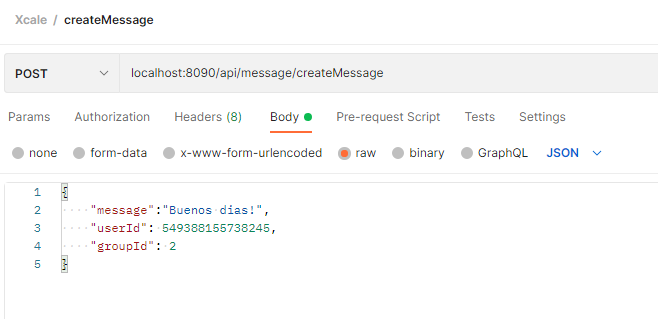
* src\main\resources\data.sql: Contiene sentencias SQL para una configuración previa (creación de tablas e inserción de contenido), como ser los mensajes, usuarios, grupos y la asociación entre los integrantes de un grupo. Cuando el sistema inicia, ejecuta los scripts correspondientes, y cuando se detiene el server se pierde el contenido de la BD, porque está en memoria.
* Ejecutar la aplicación como “Spring Boot App”
* Validaciones realizadas en la lógica de negocio:   
  1) En la creación de un nuevo mensaje, se valida que el usuario que está creando el nuevo mensaje pertenezca al grupo en el cual desea publicar. Hace uso de la Excepción UserException.  
  2) En la creación de un nuevo mensaje, se valida que:   
  - El atributo mensaje sea recibido por el backend(!=null), que no sea un mensaje vacío y la longitud máxima del mensaje por BD y validación es de 250 caracteres como máximo.  
  - El atributo correspondiente al Id del usuario sea recibido por el backend(!=null). El atributo debe estar presente en el Body del POST.   
  - El atributo correspondiente al Id del grupo sea recibido por el backend(!=null) . El atributo debe estar presente en el Body del POST.   
  Las validaciones hacen uso de la Excepción ResponseValidException.
* Se realizó un custom @ControllerAdvice para permite manejar excepciones en toda la aplicación en un componente de manejo global. Puede verse como un interceptor de excepciones lanzadas por métodos anotados con @GetMapping y similares. Generado un ResponseEntity con mensajes custom de error.
* Se proporciona una “**collection**” de Postman para poder realizar las pruebas de la Aplicación desarrollada. Importar el file “Xcale.postman\_collection.json” y usar el GET y POST disponible



* **GET** getAllMessagesFromGroup: Para obtener todos los mensajes de un determinado grupo. Al modificar el valor del param, traerá los mensajes para el grupo seleccionado.



* **POST** createMessage: Para crear un mensaje con una determinada persona y en un determinado grupo y generar nuevas notificaciones de mensajes recibidos. Se debe modificar la info que se encuentra en el Body de esta solicitud POST.



* src\test\java: Se crearon los Test unitarios haciendo uso de JUnit y Mockito.
* Se realizaron comentarios en el código para documentar la funcionalidad del mismo.
* Se realizaron mappers mediante MapStruct, utilizando anotaciones, que nos brindan la capacidad de generar mapeos entre diferentes objetos en tiempo de compilación. En nuestro ejemplo entity -> entityDTO y entityDTO -> entity.