Desarrollo técnico Kibernum.

Introducción.

El siguiente documento contempla el desarrollo de un API/REST para el manejo de una biblioteca. Se pensó en una autenticación mediante la implementación de Jason Web Token (jwt), donde los usuarios con rol ADMIN están autorizados para tareas como:

* Crear libros
* Eliminar libros
* Actualizar libros
* Listar todos los libros
* Listar reservas, ya sea por c´odigo de libro o por rut de alumno
* Reservar/pedir libros
* Eliminar/devolver libros.

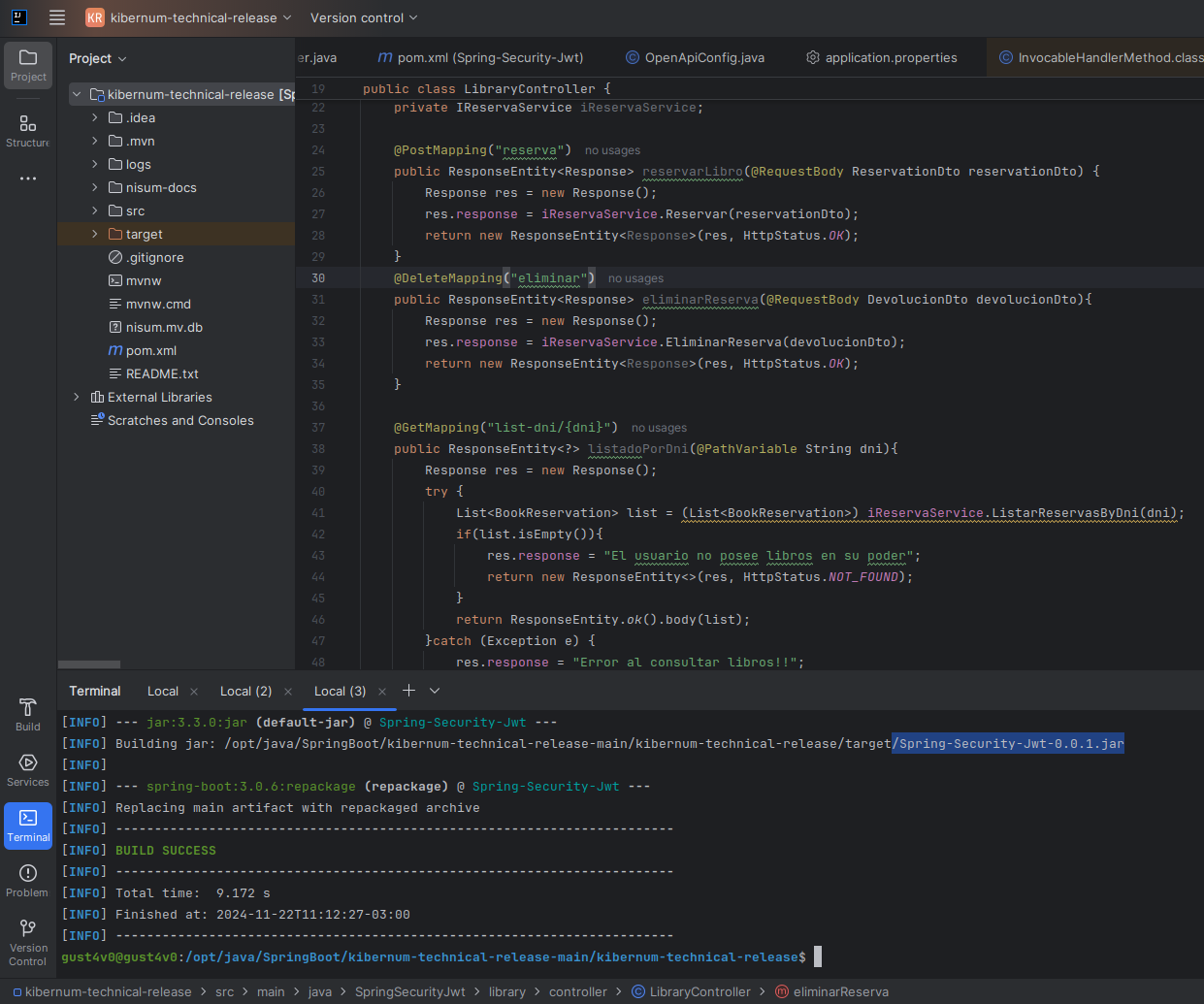
También se implementó la documentación de la api mediante Swagger utilizando el estandar de OpenApi.

Se utilizó una base de datos en memoria (H2) para facilitar las pruebas. Pero también se proporciona (en forma comentada) la configuración necesaria para utilizar una base de datos Mysql.

Se proporciona una collection en formato json para pruebas, esta collection es posible importarla con aplicaciones como insomnia o postman, a fin de facilitar las pruebas funcionales.

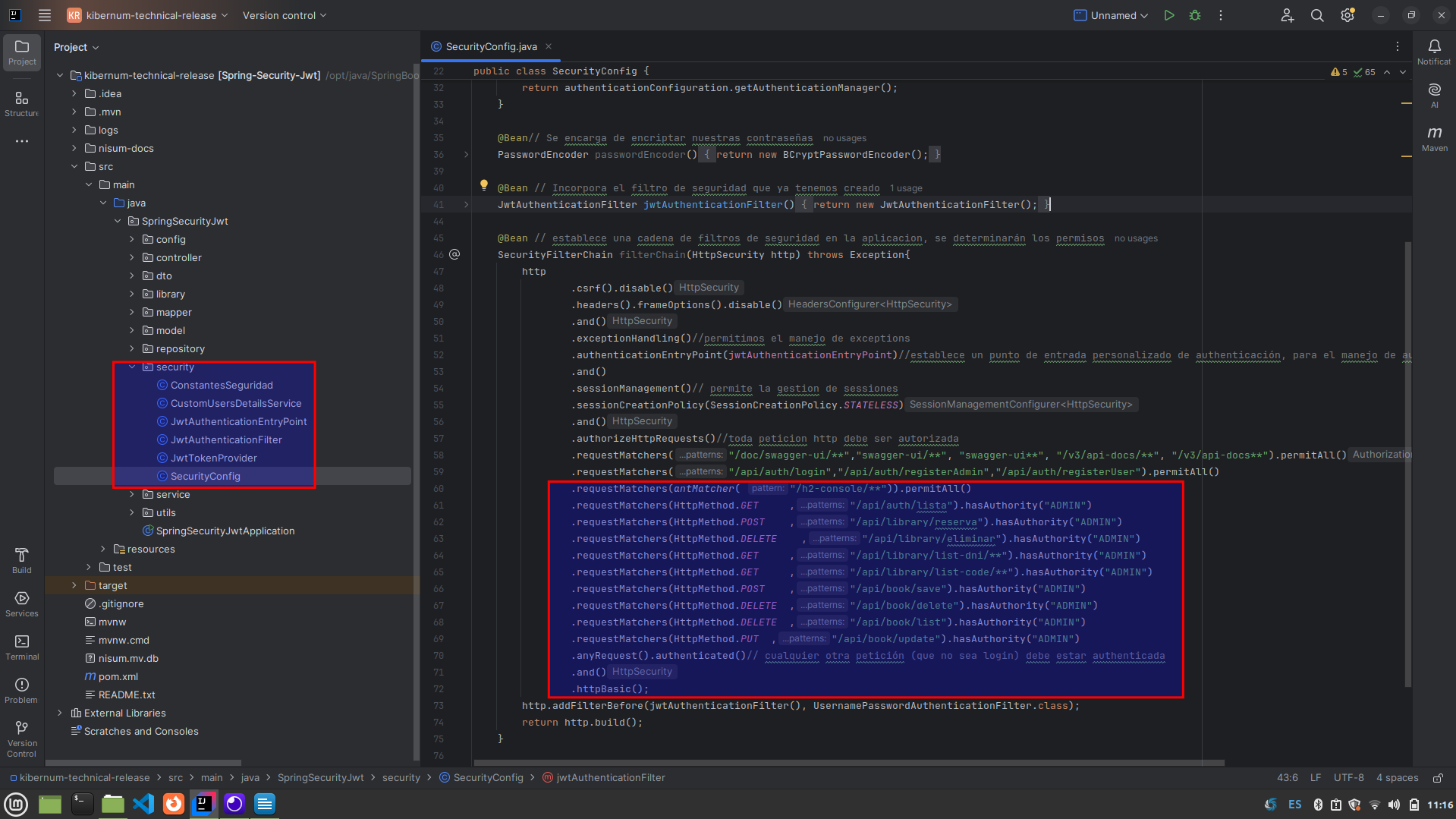
Detalles del proyecto.

Compilación del proyecto

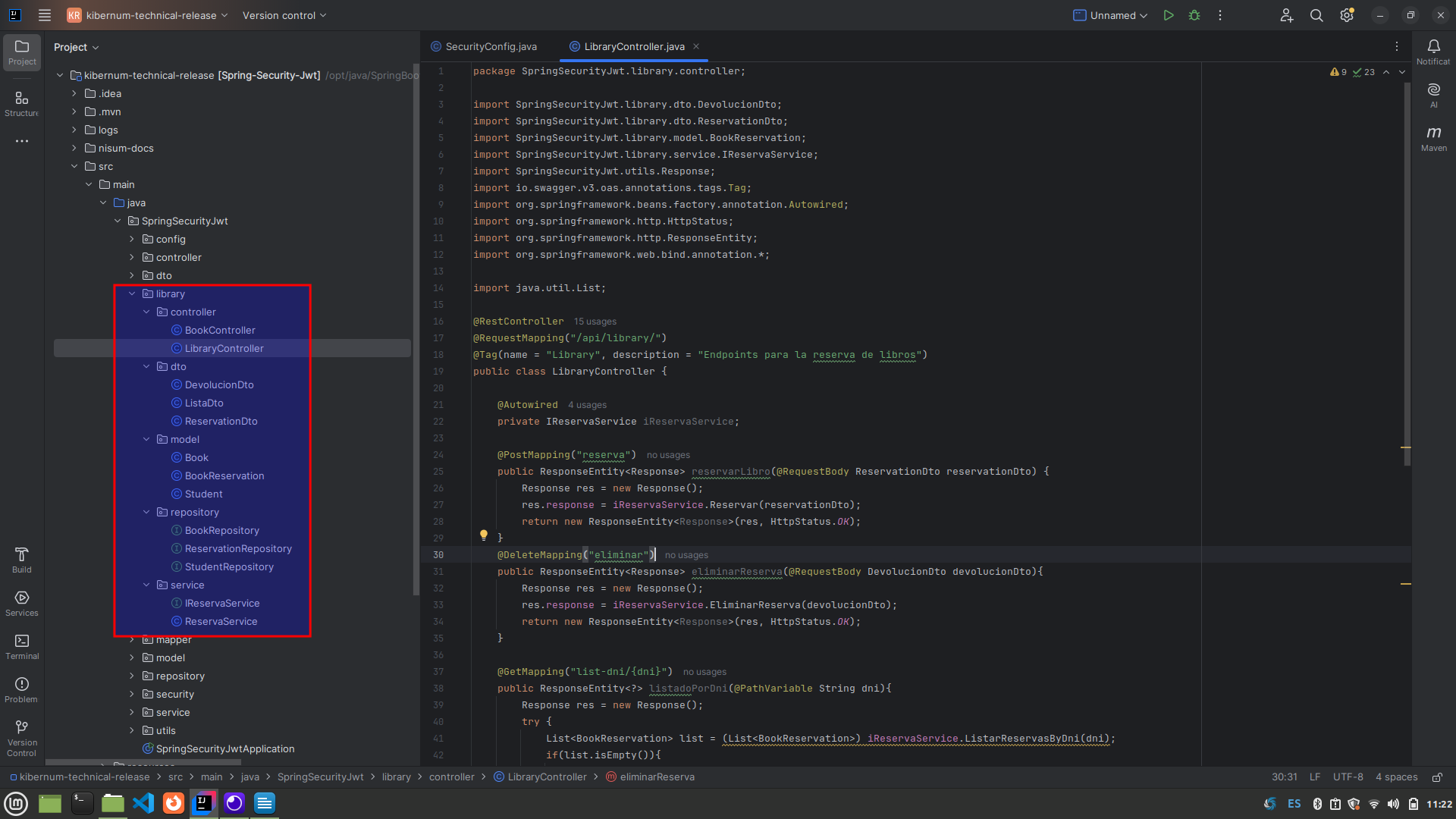


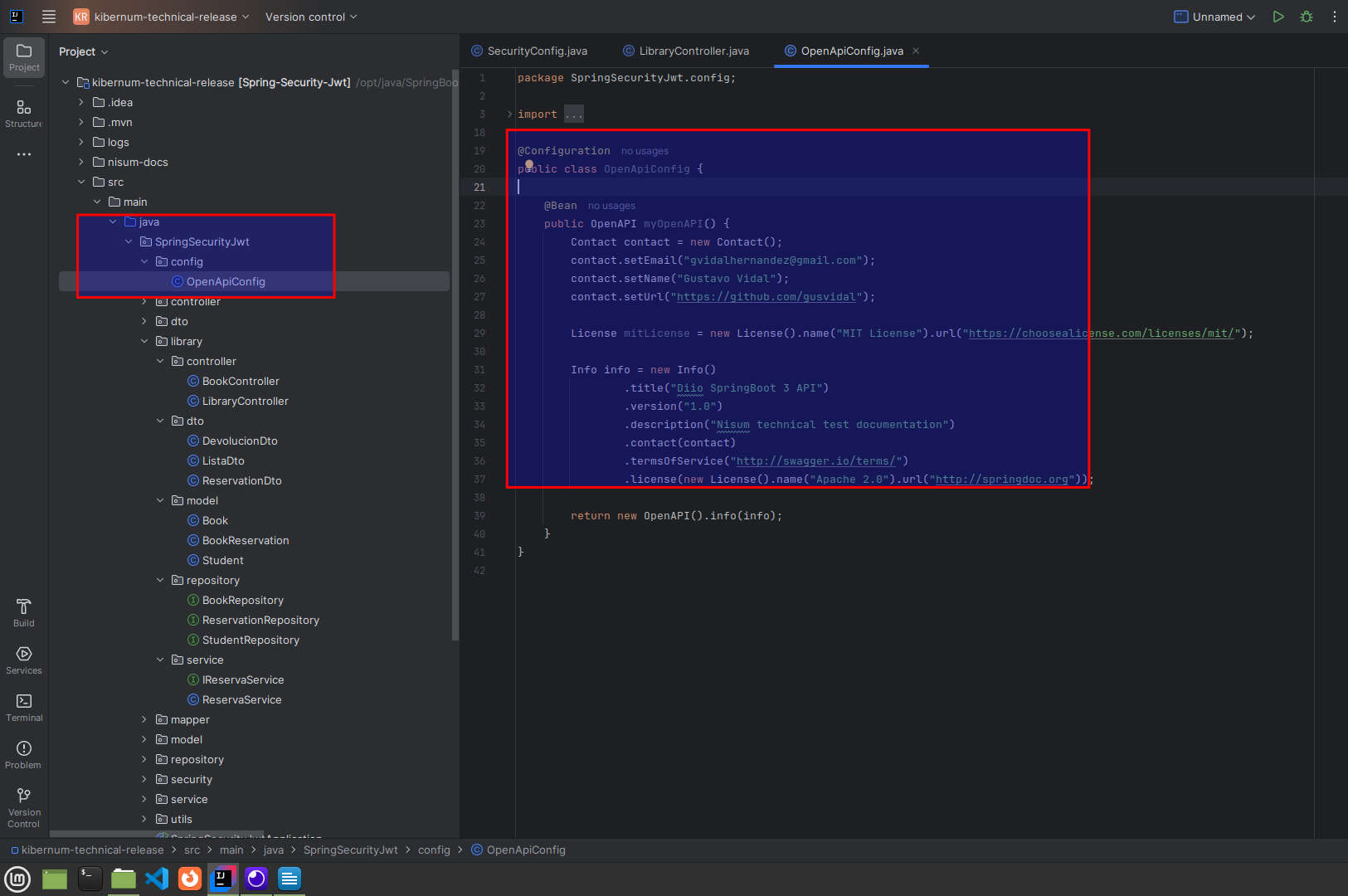
En las siguientes imágenes, se hace mención a las rutas relevantes del proyecto.

Sección de autenticación y autorización mediante jason web token, en la sección de la izquierda se muestra el directorio con la configuración de seguridad, en tanto, a la derecha se encuentran definidos los matchers necesarios para dar permisos sobre los endpoint del servicio.

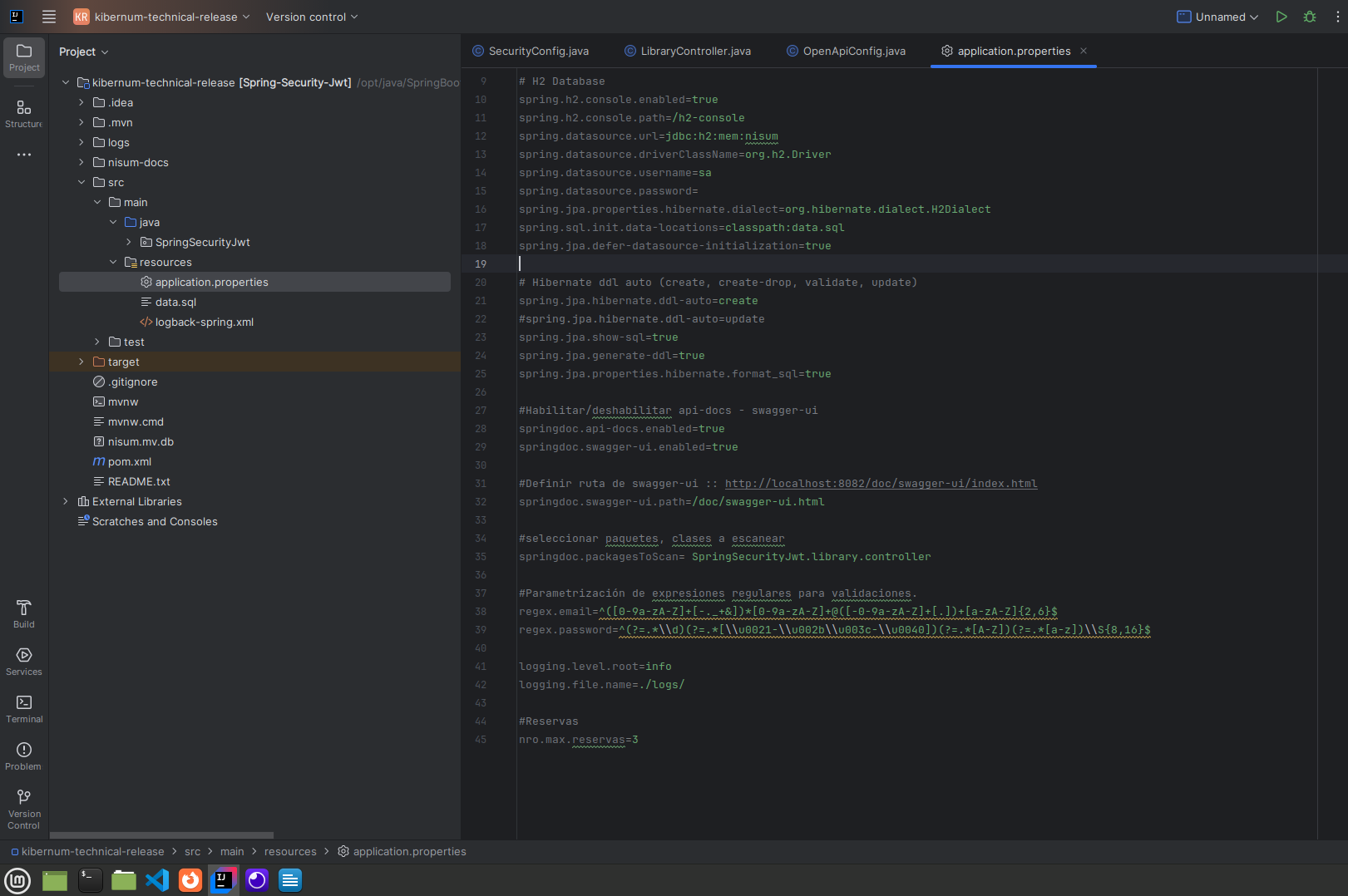


La siguiente imagen muestra el desarrollo referente a la librería, solicitado por Kibernum, se implementaron dos controller, uno para el manejo de los libros y otro para el manejo de las reservas de libros. Se desarrolló también, una capa de servicio con su respectiva interfaz.

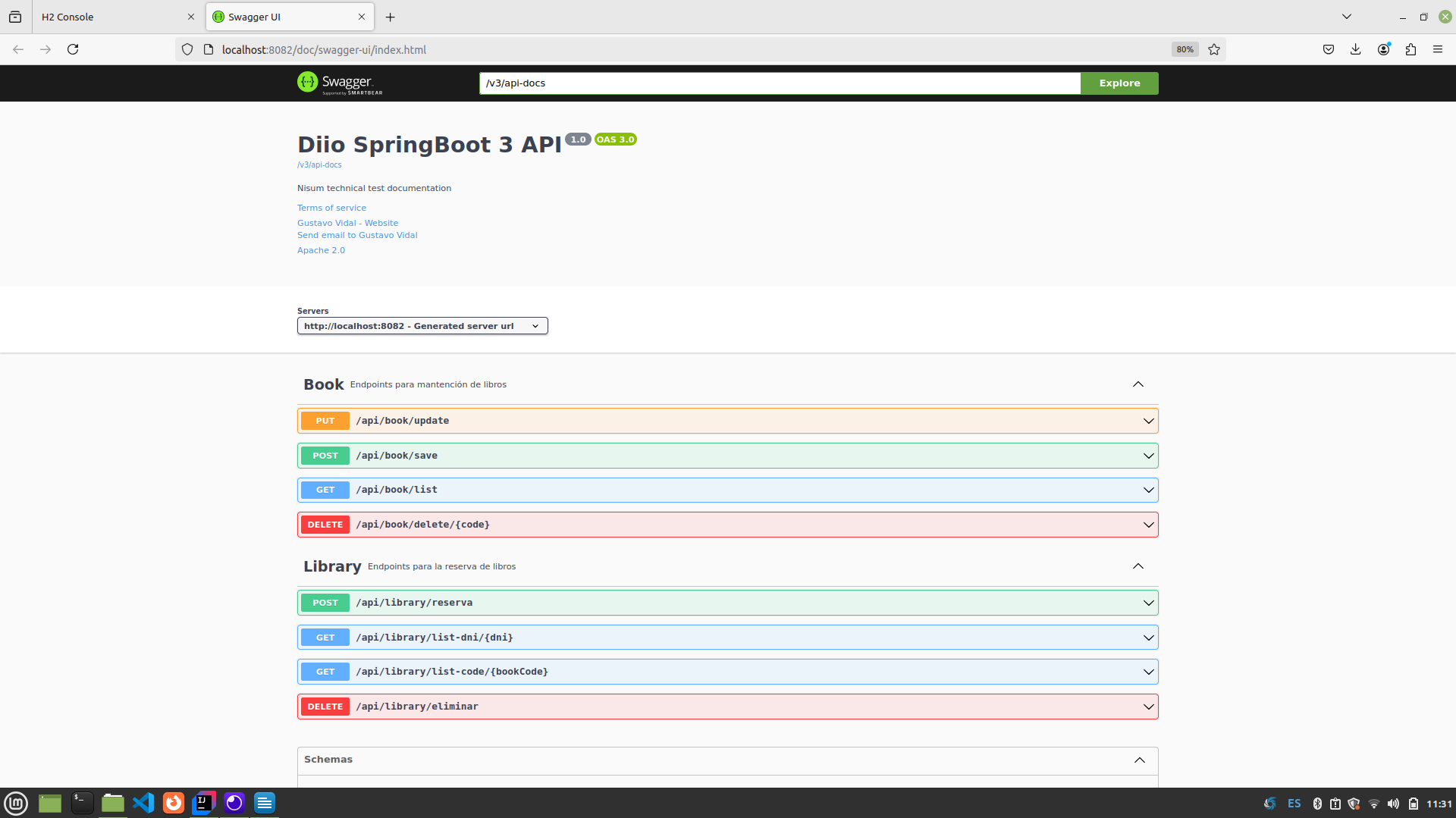
A continuación se muestra la configuración de OpenApi



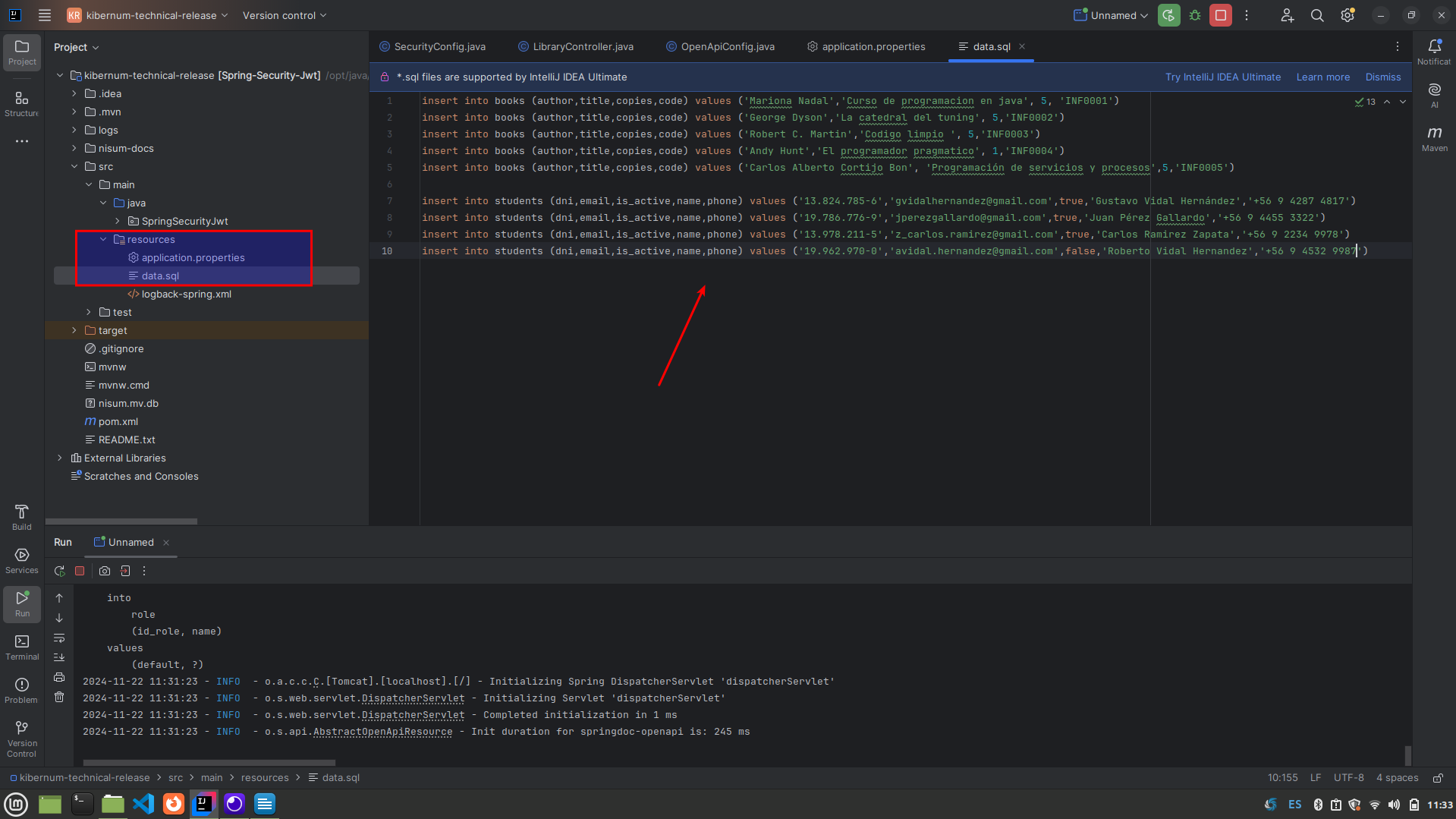
Así como también las configuraciones de url y los controllers a presentar.

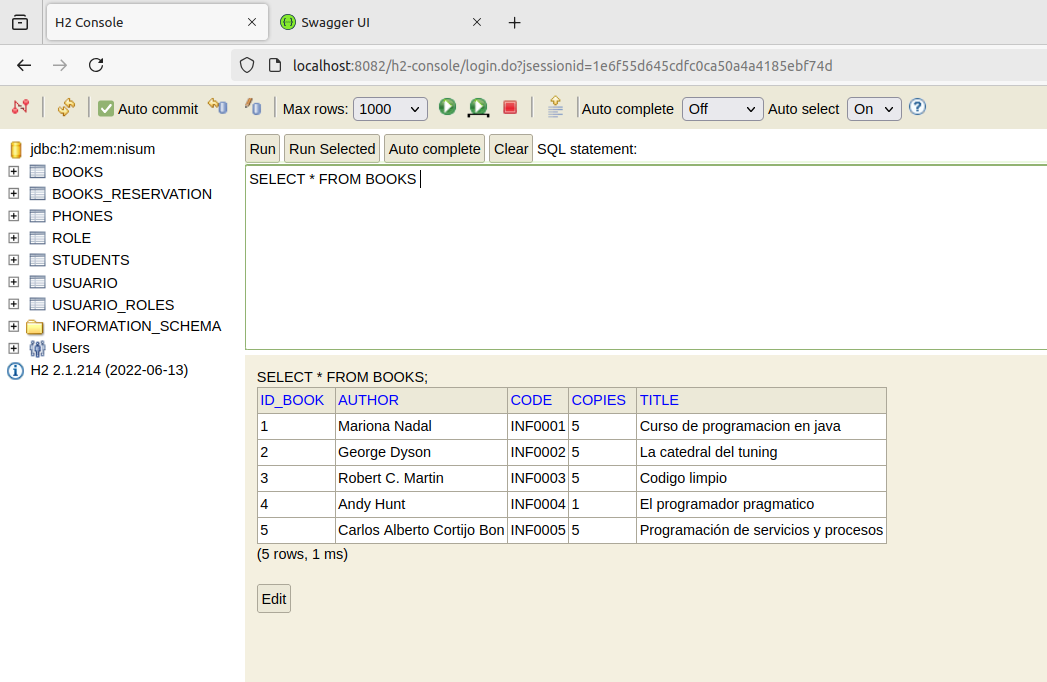


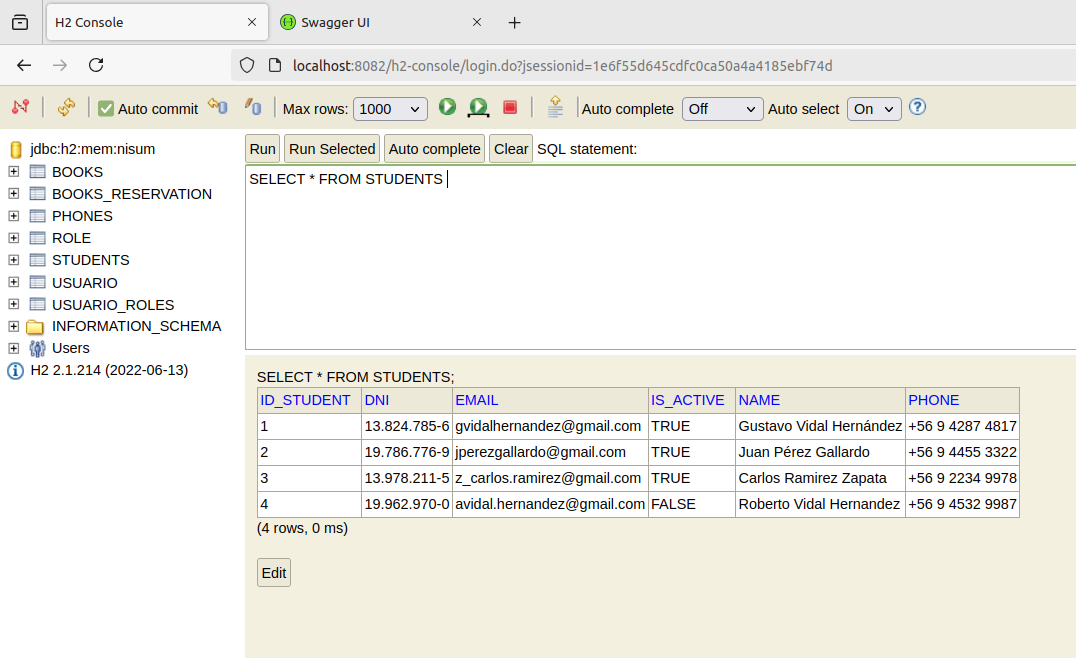
En la siguiente imagen, se puede apreciar Swagger en ejecución.



Para facilitar las pruebas, se crean objetos de datos de bases de datos, y se poblan las tablas con datos de ejemplo al momento de iniciar la aplicación.

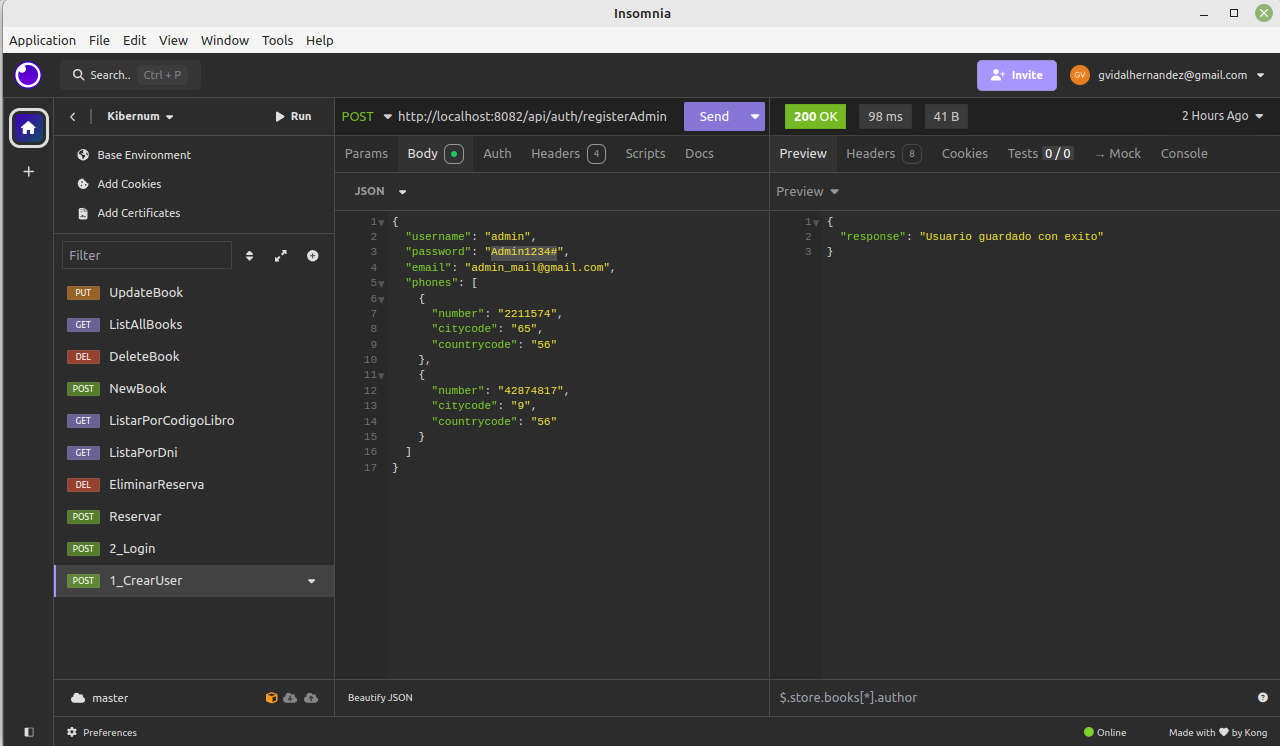




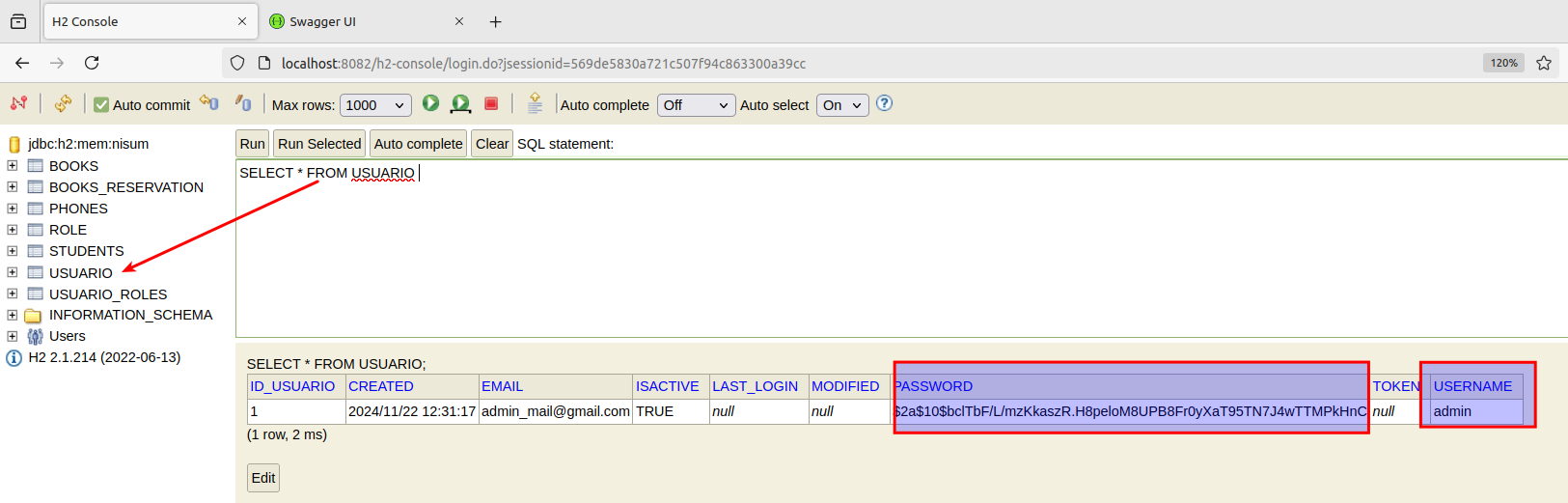


En cuanto a las pruebas funcionales, se adjunta un archivo readme.txt donde se disponen los endpoint y los json de entrada para poder realizar las pruebas funcionales del proyecto.

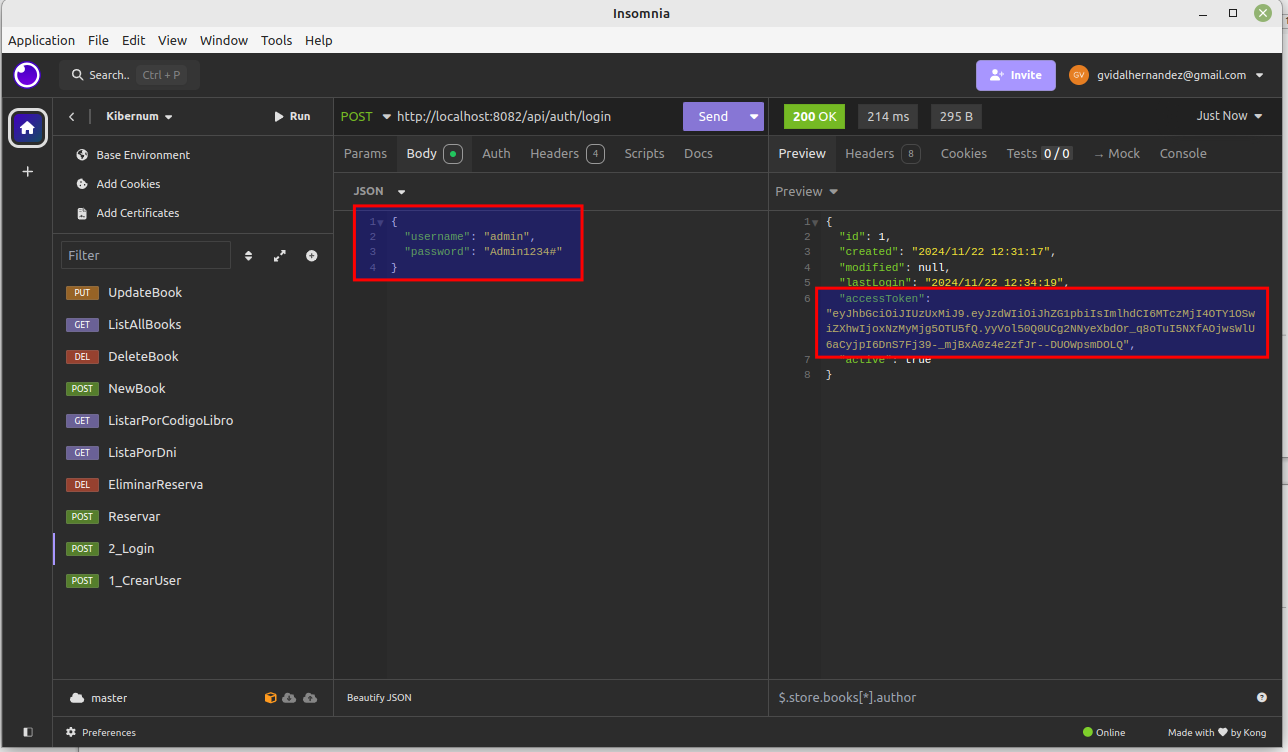
1.- CrearUser, esto creaá un usuario del tipo ADMIN en la base de datos



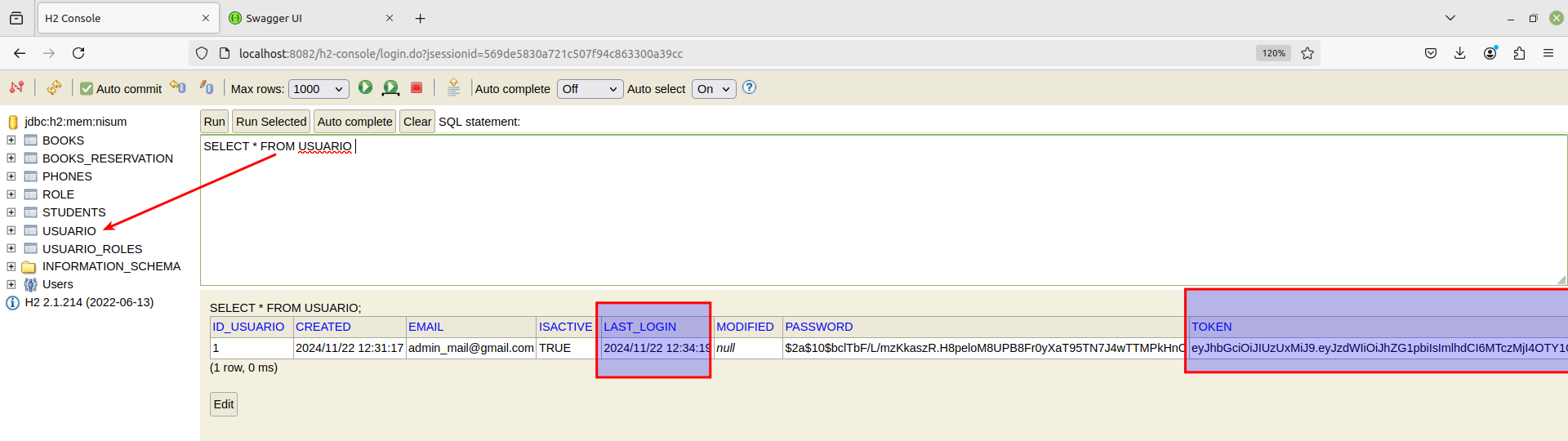
En base de datos, se guarda en la tabla USUARIO, de la siguiente forma



2.- Luego es necesario loguear al usuario recien creado, esto generará un token, el cual será solicitado en los demas endpoint.

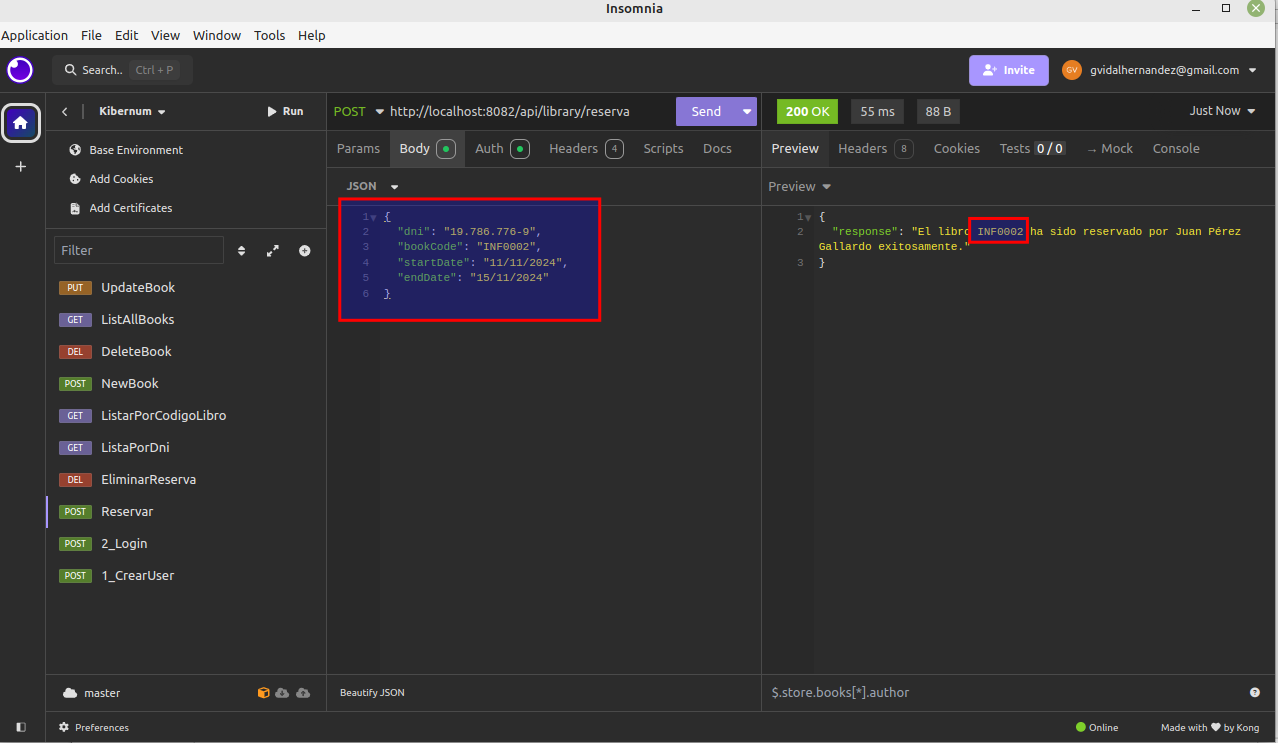


En base de datos, se almacena el token recien generado

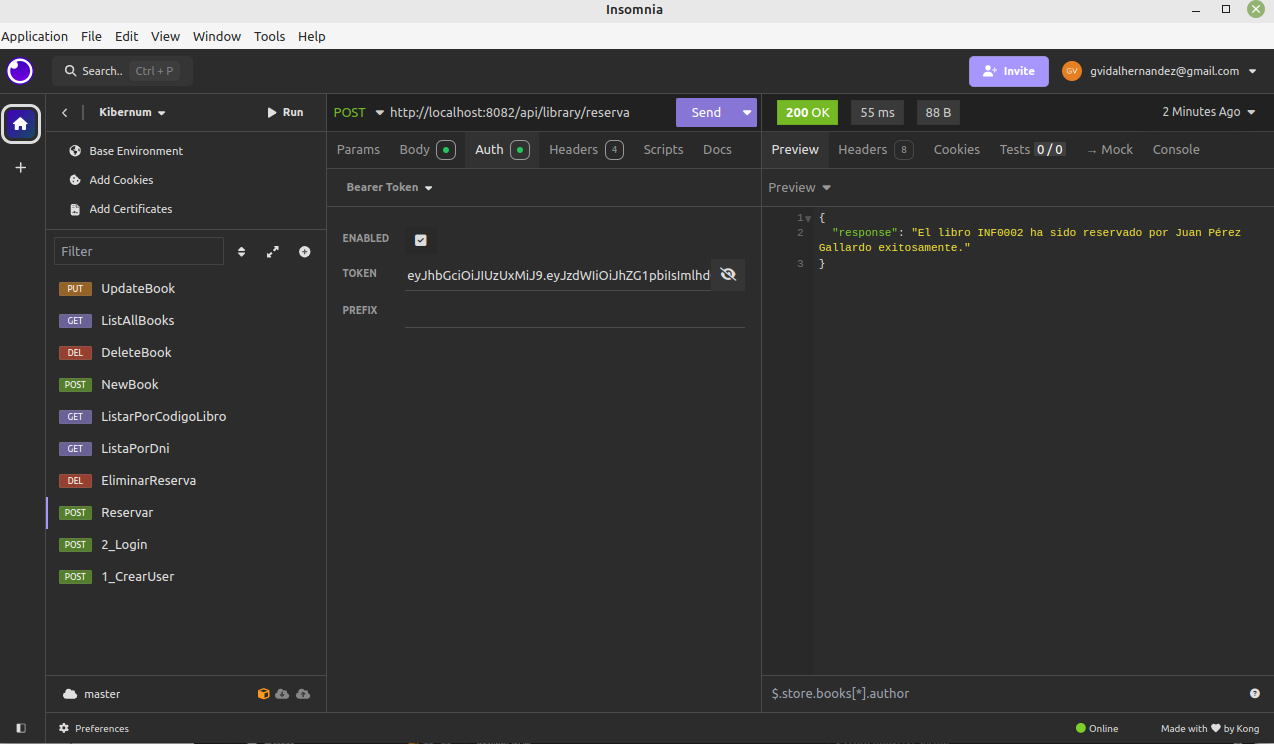


3.- Reservar

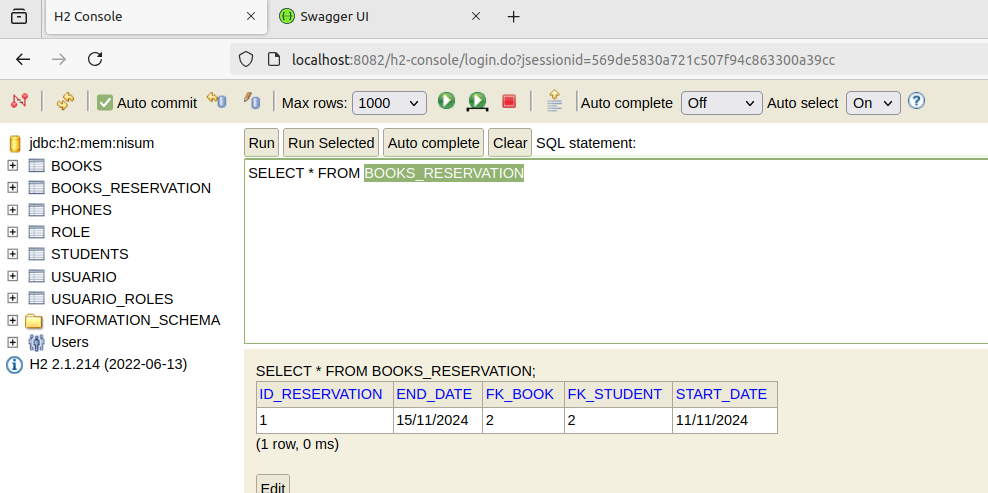
El endpoint permite reservar libros, requiere un json de entrada



Es necesario adjuntar el token, del tipo Bearer, como se ilustra en la siguiente imagen.

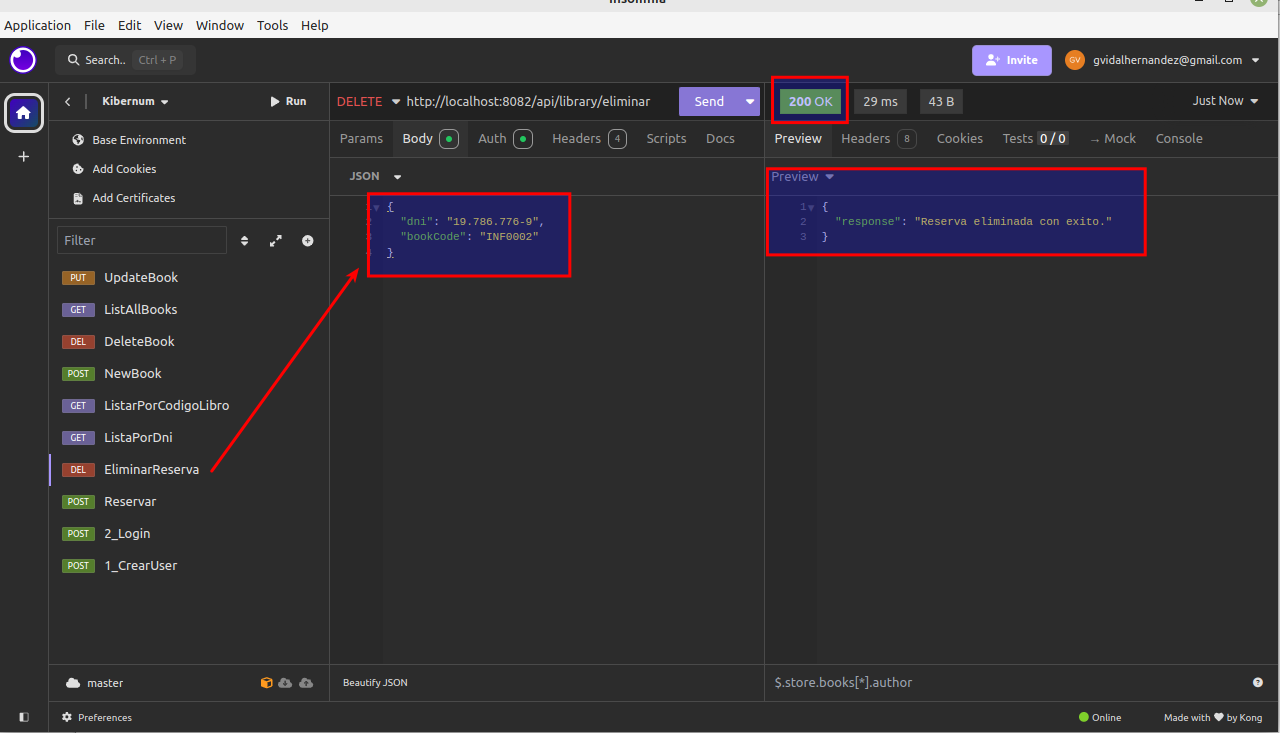


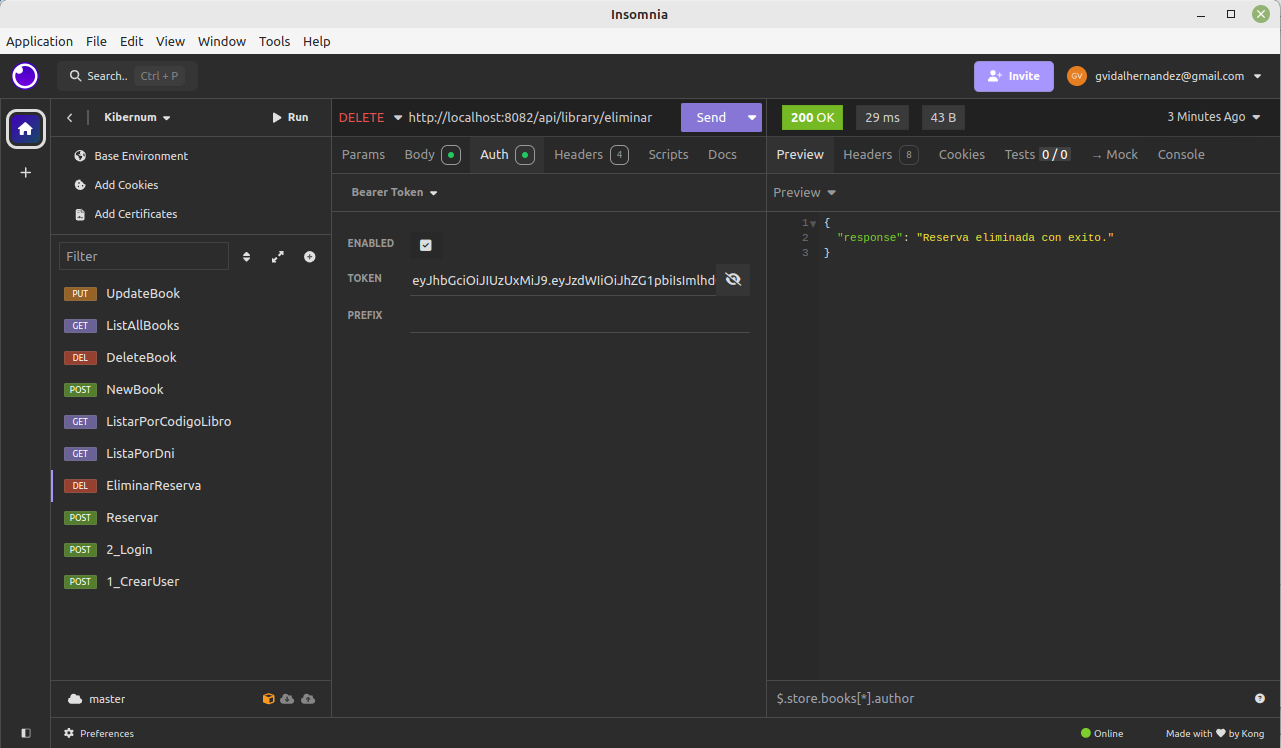
En base de datos, queda guardado en la tabla BOOKS\_RESERVATION



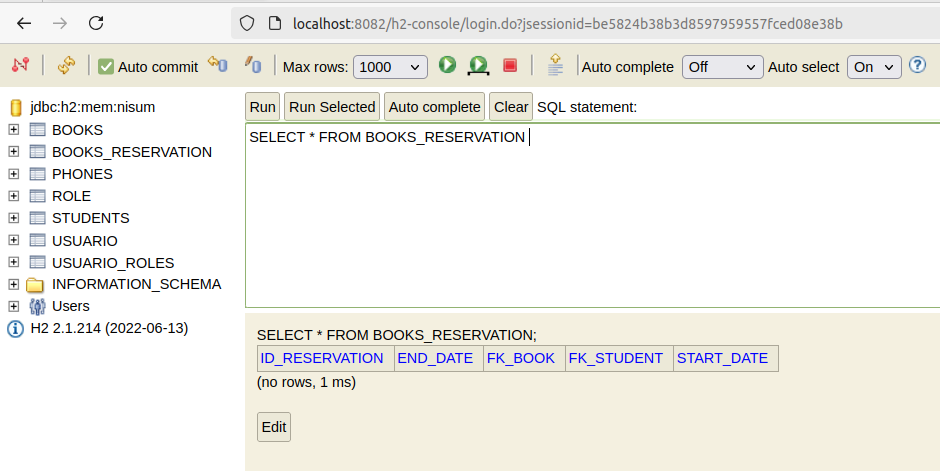
4.- EliminarReserva

Para eliminar la reserva, se debe enviar un json con el rut y el codigo del libro. Y adjuntar el token

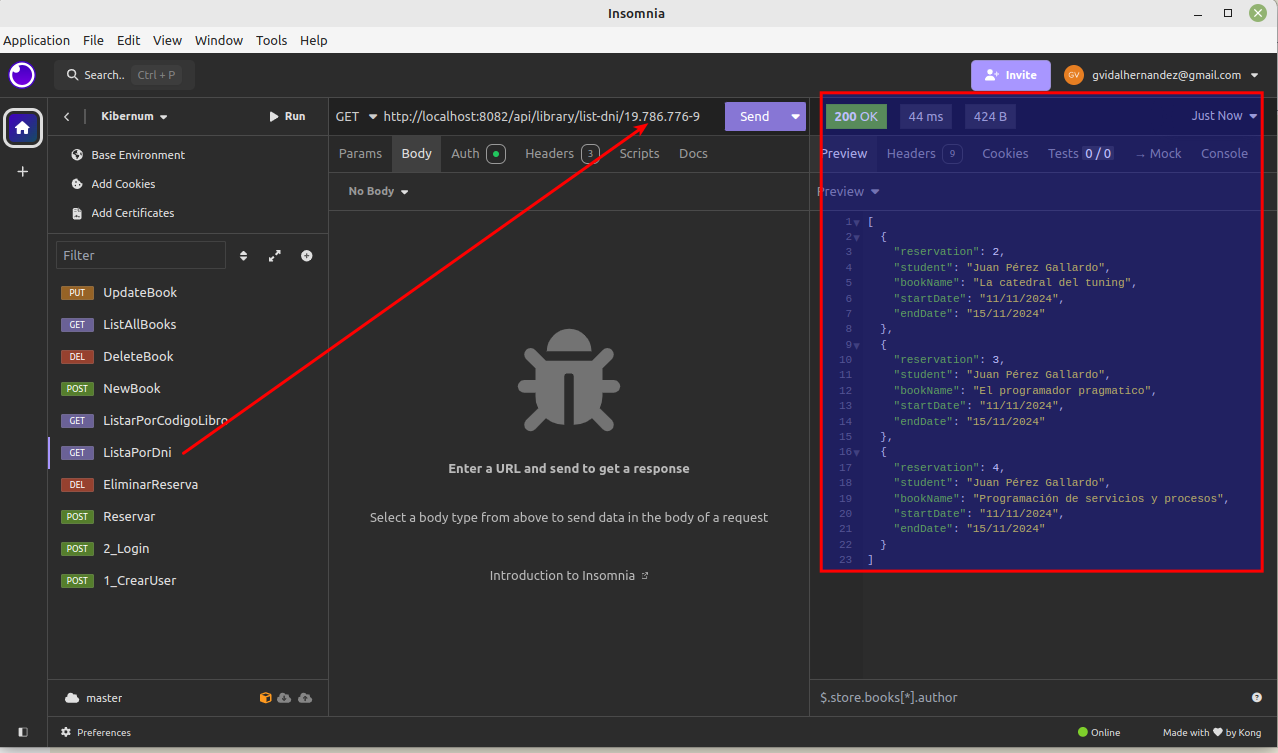




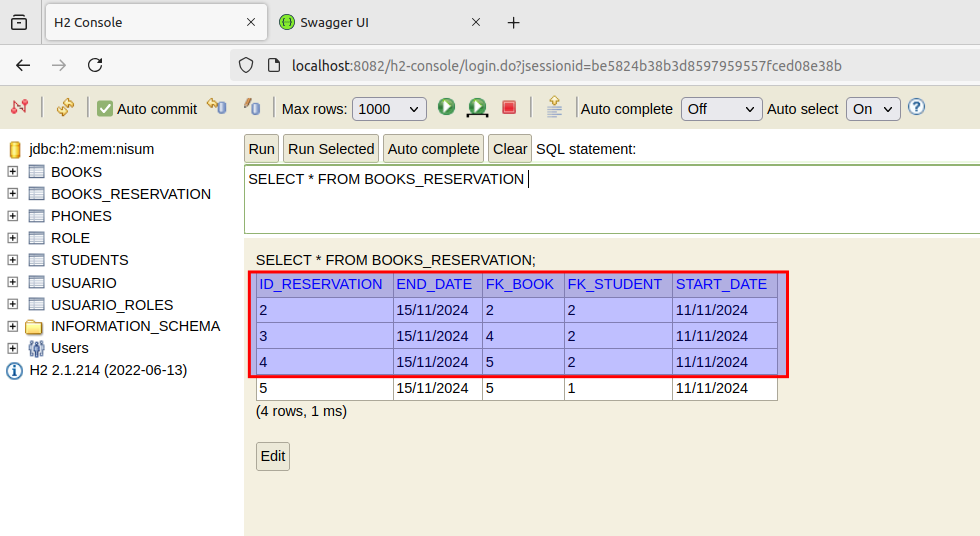
En la base de datos, la reserva que ejecutamos en el paso 3, ya no existe.



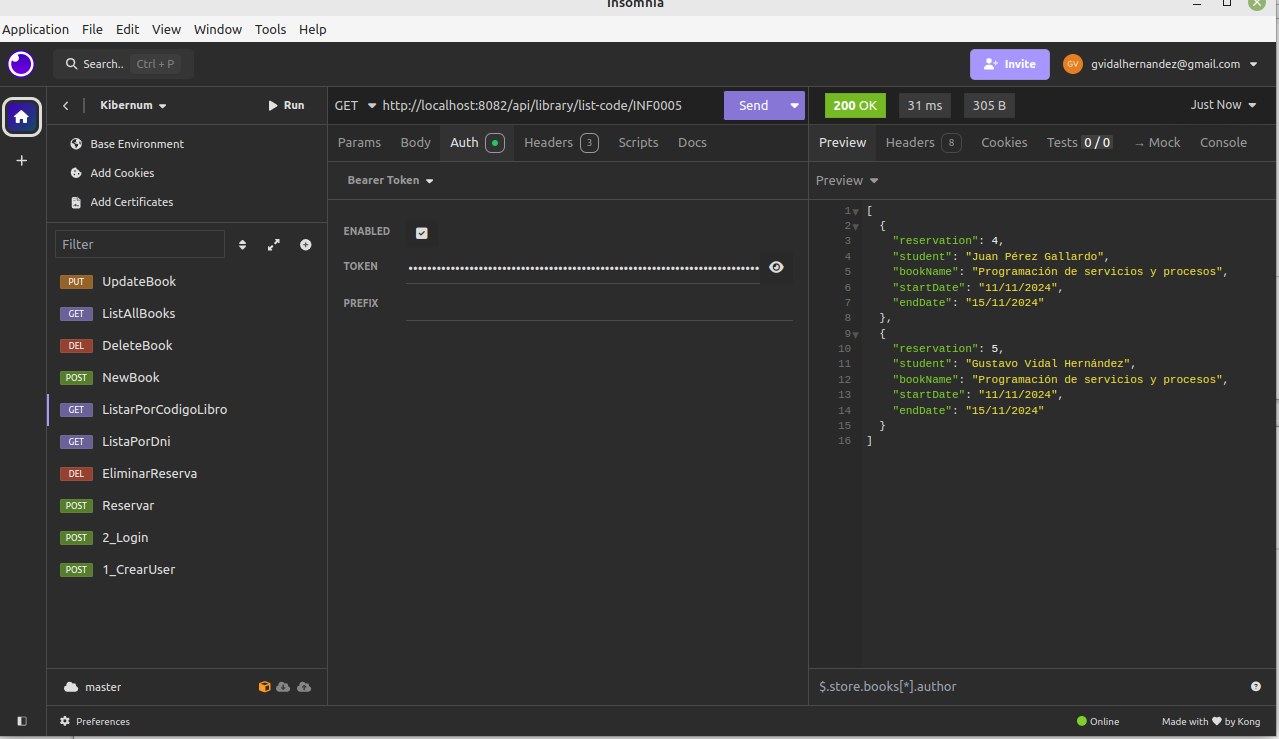
5.- ListarPorDni entrega un listado de todos los libros pedidos por un usuario en particular (dni)



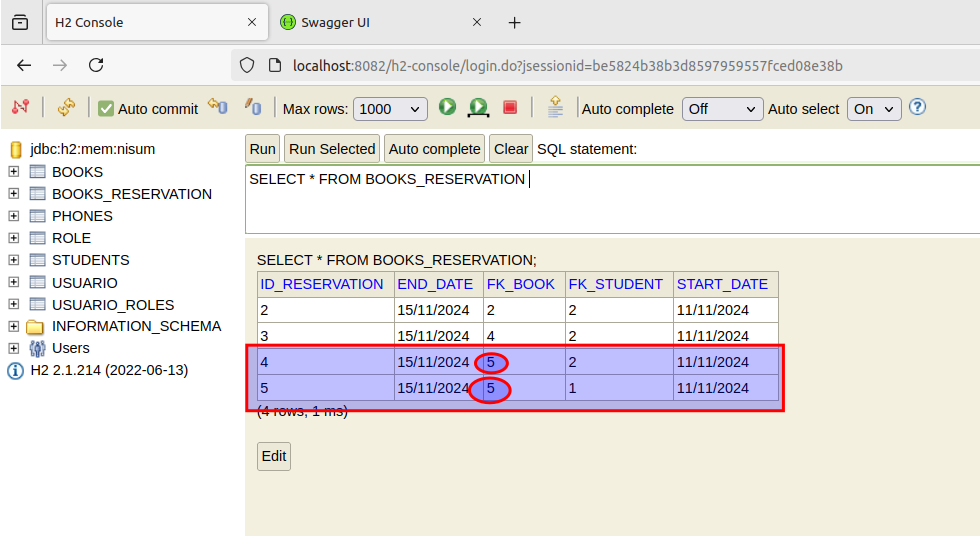
En la base de datos, observamos:



6.- ListarPorCodigoLibro muestra todas las peticiones realizadas por alumnos de un libro en particular.

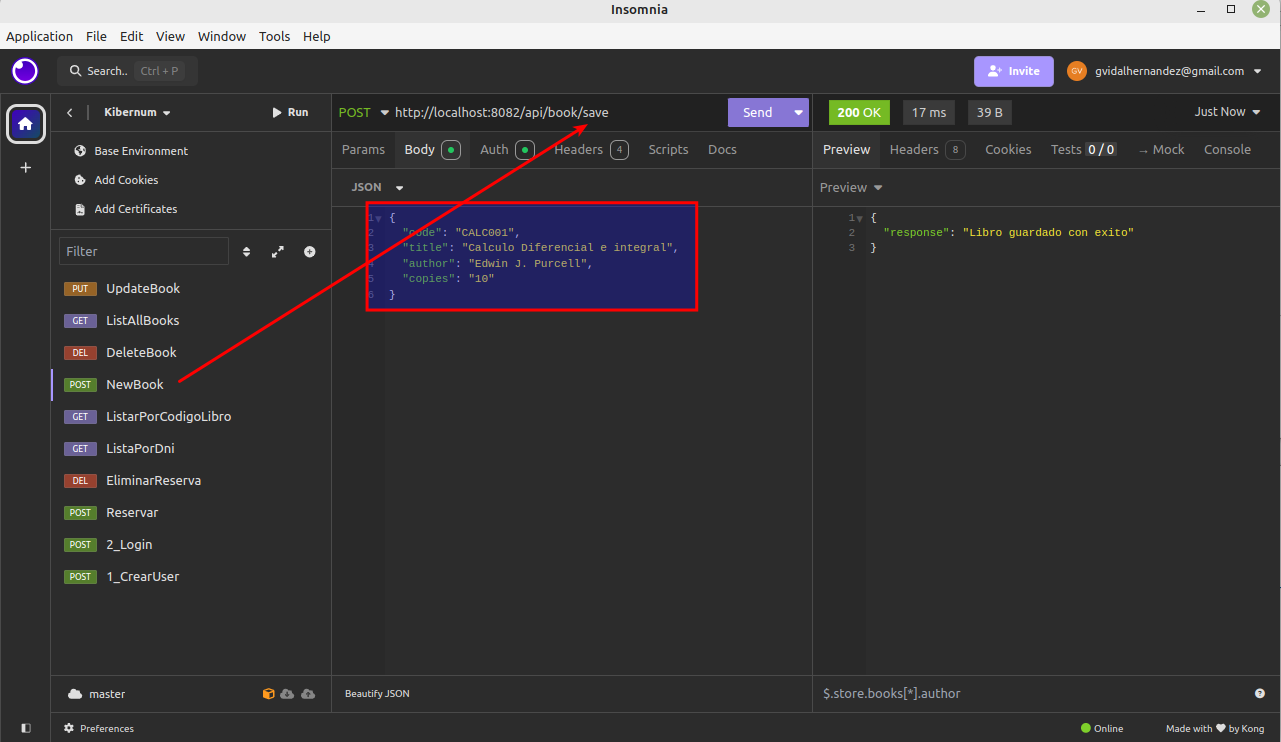


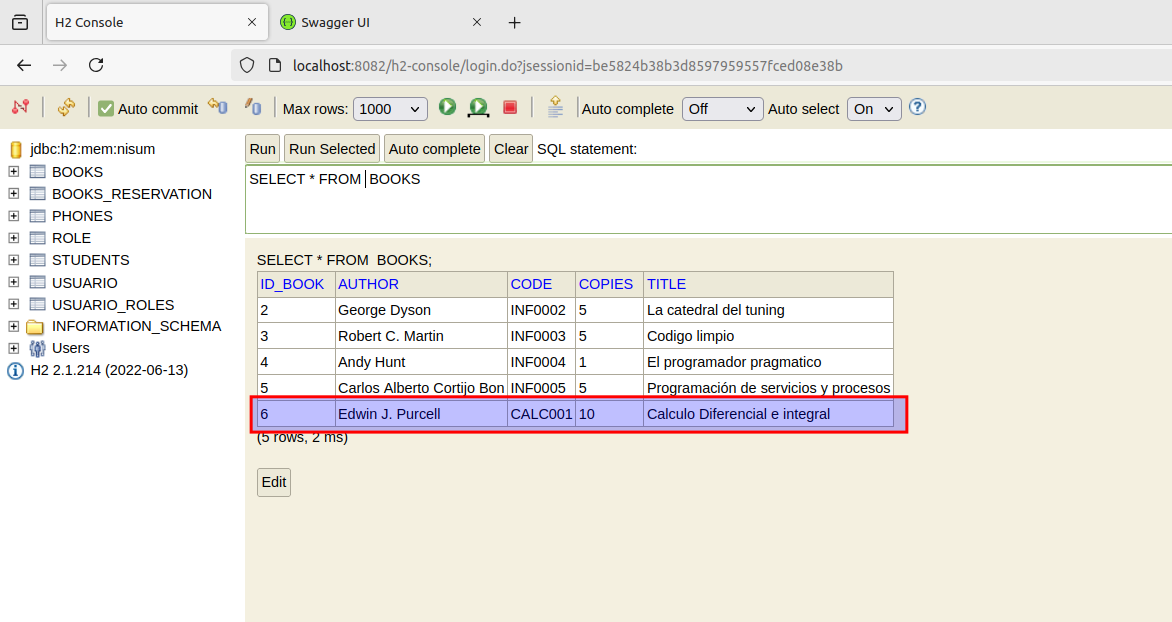
En base de datos podemos ver:



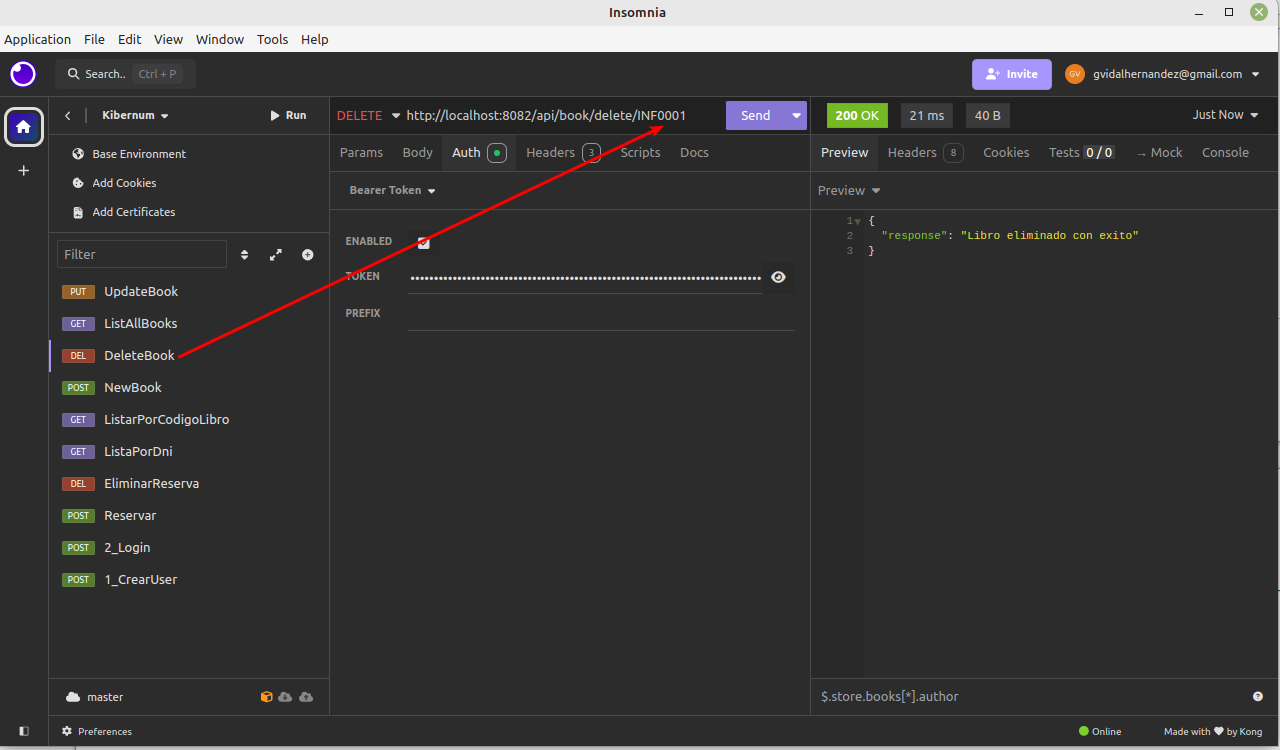
En cuanto al mantenedor de libros se refiere, podemos crear, eliminar, actualizar y listar (CRUD) libros del sistema.

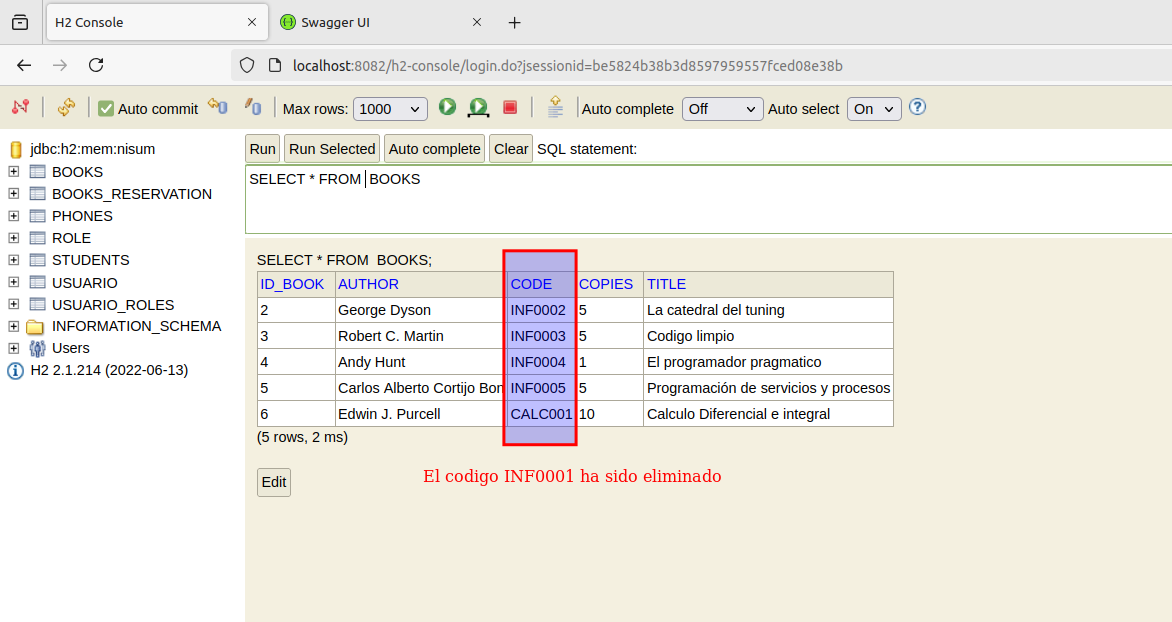
1.- Crear un nuevo libro.



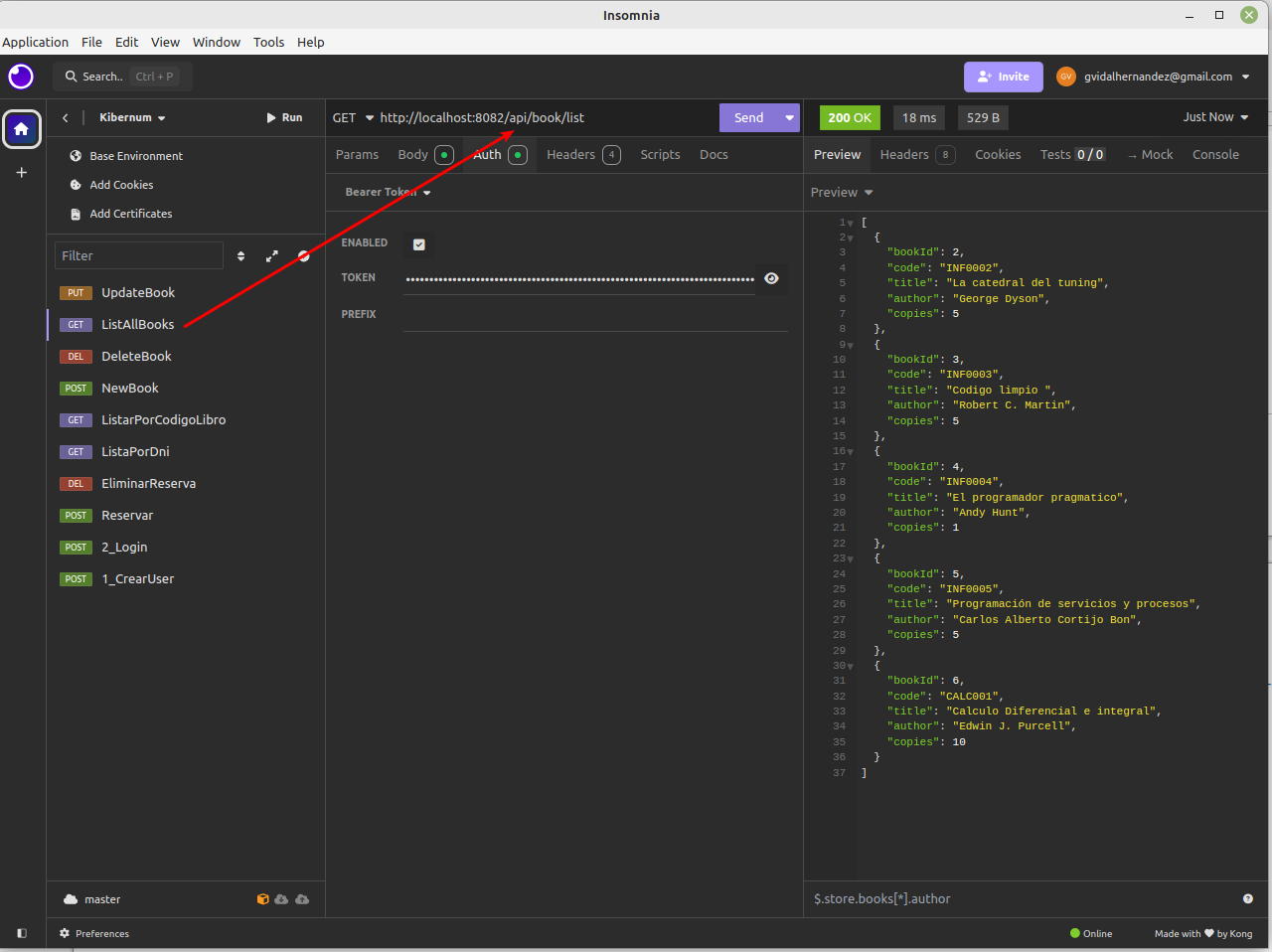


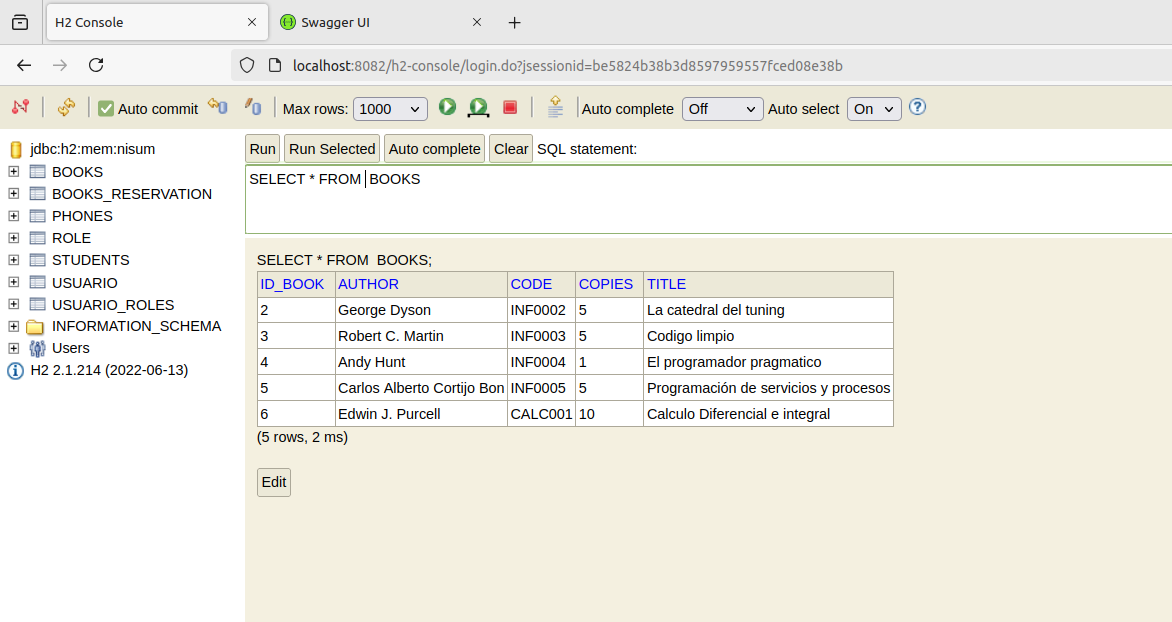
2.- Eliminar libro





3.- Listar todos los libros





4.- Actualizar libro

