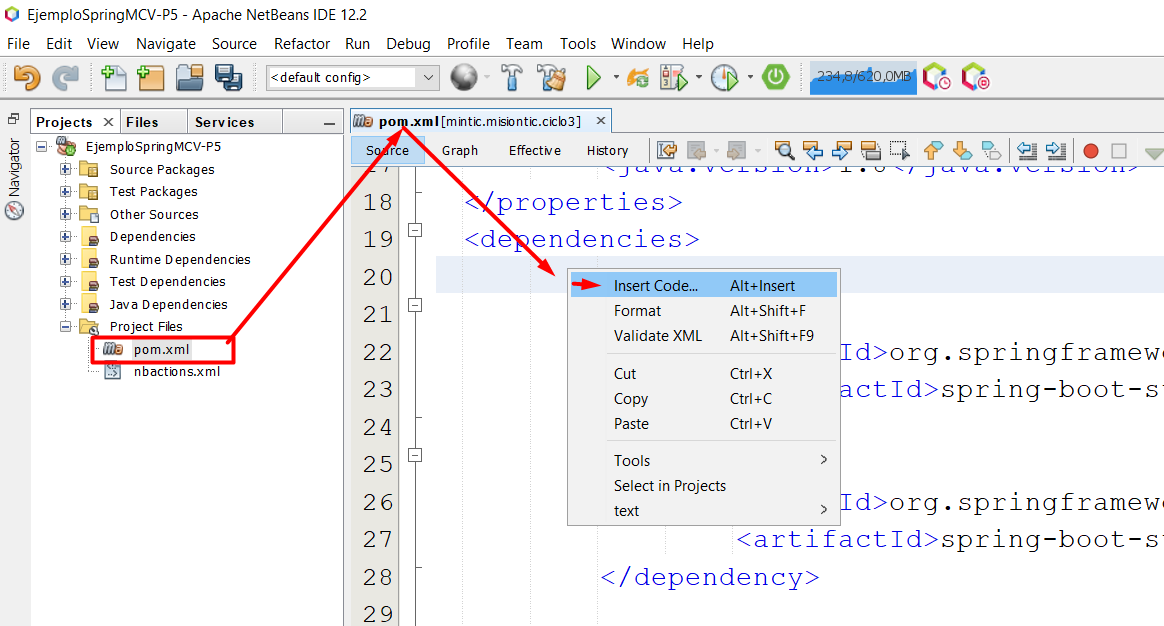
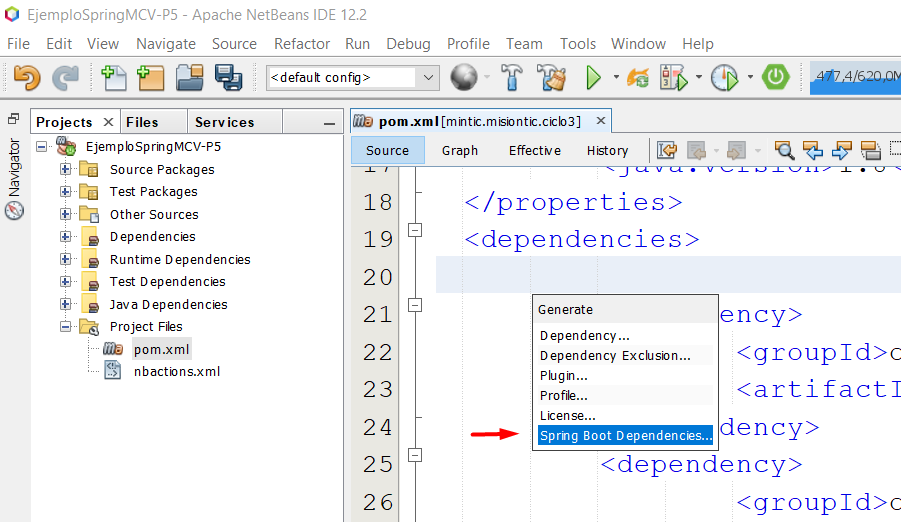
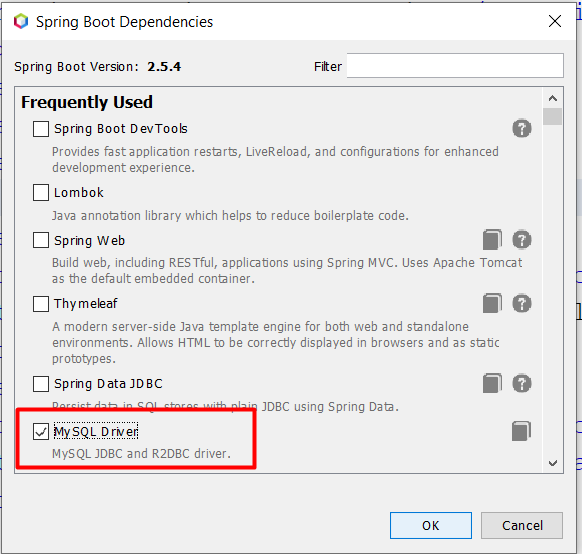
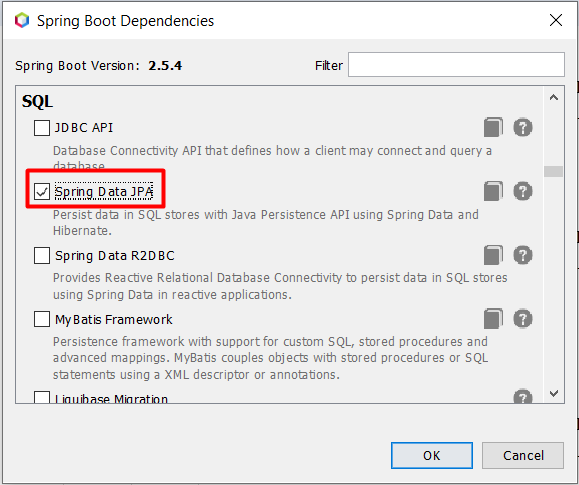
AGREGAR LAS DEPENDENCIAS NECESARIAS

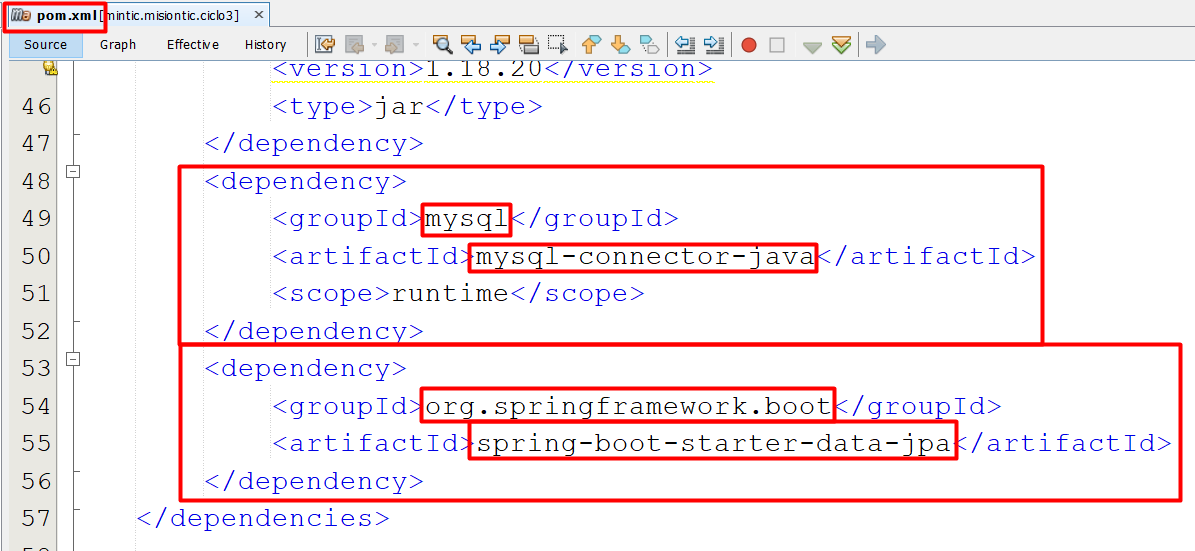




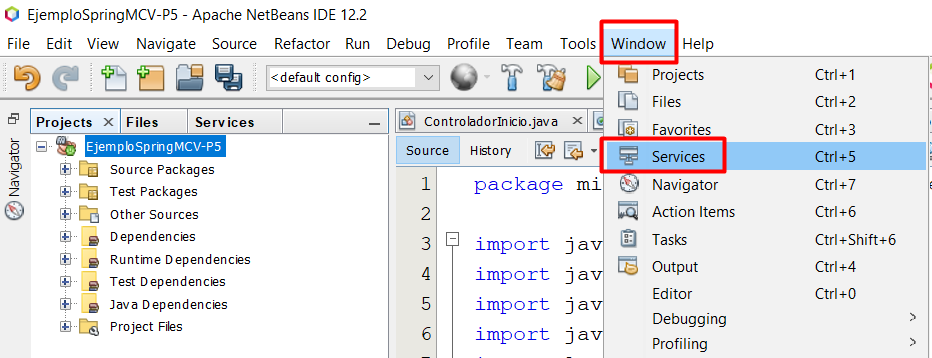


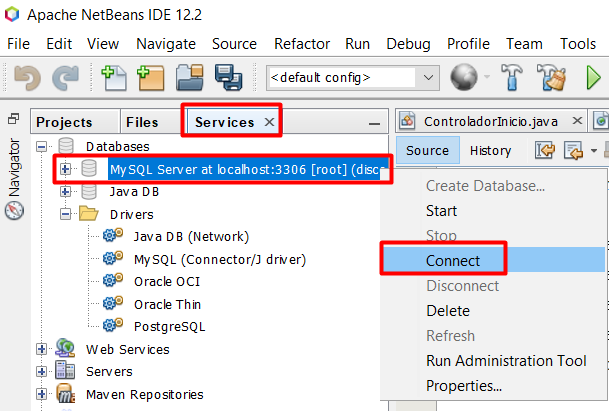
Para poder conectar nuestra API con una base de datos, debemos utilizar un driver correspondiente a esa base de datos. Y para manejar de forma mas sencilla las querys a esa base de datos, lo mas común es utilizar un ORM, para Java Spring Boot es mas comun es Spring Data JPA.

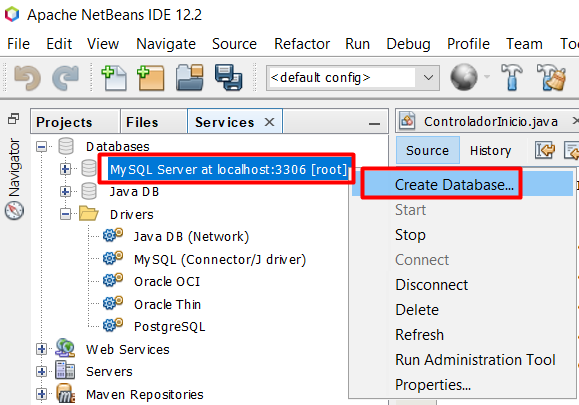


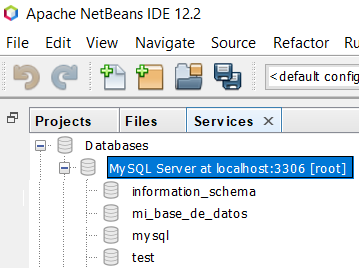


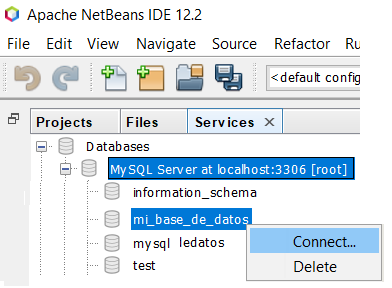
Desde netBeans podemos realizar una conexión a mySql muy sencillo

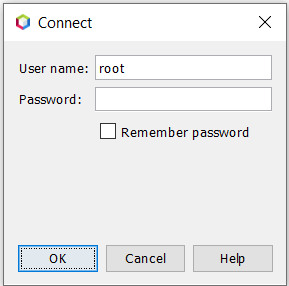


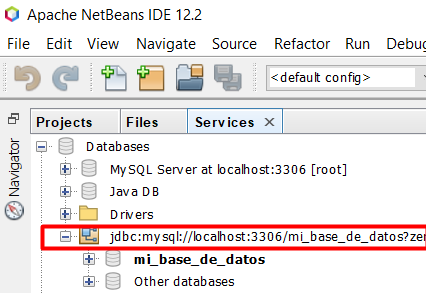


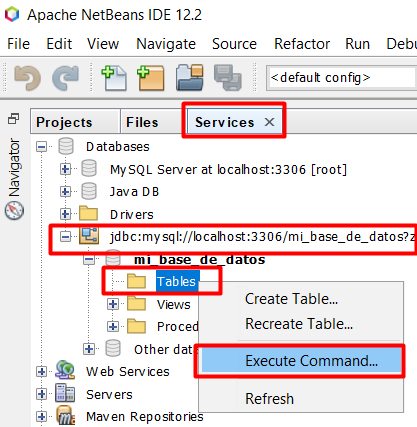


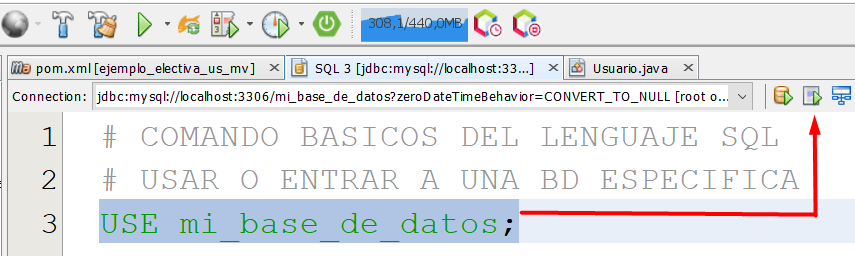


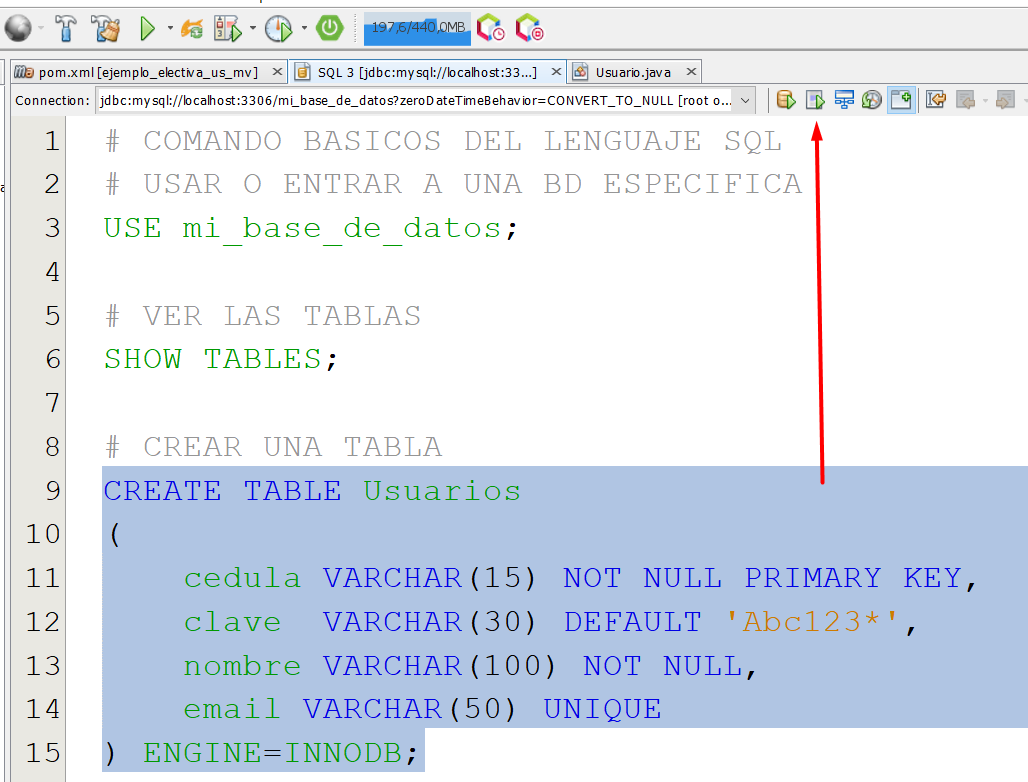


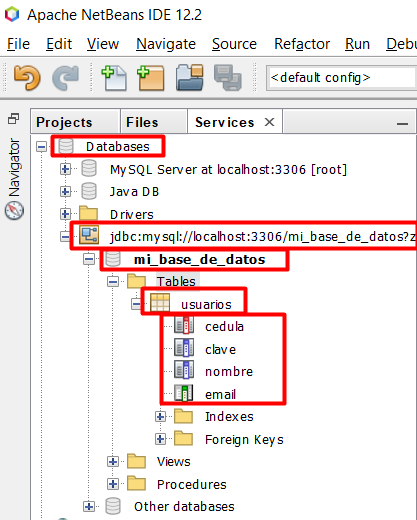




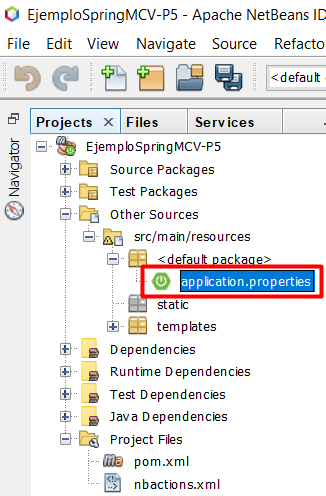




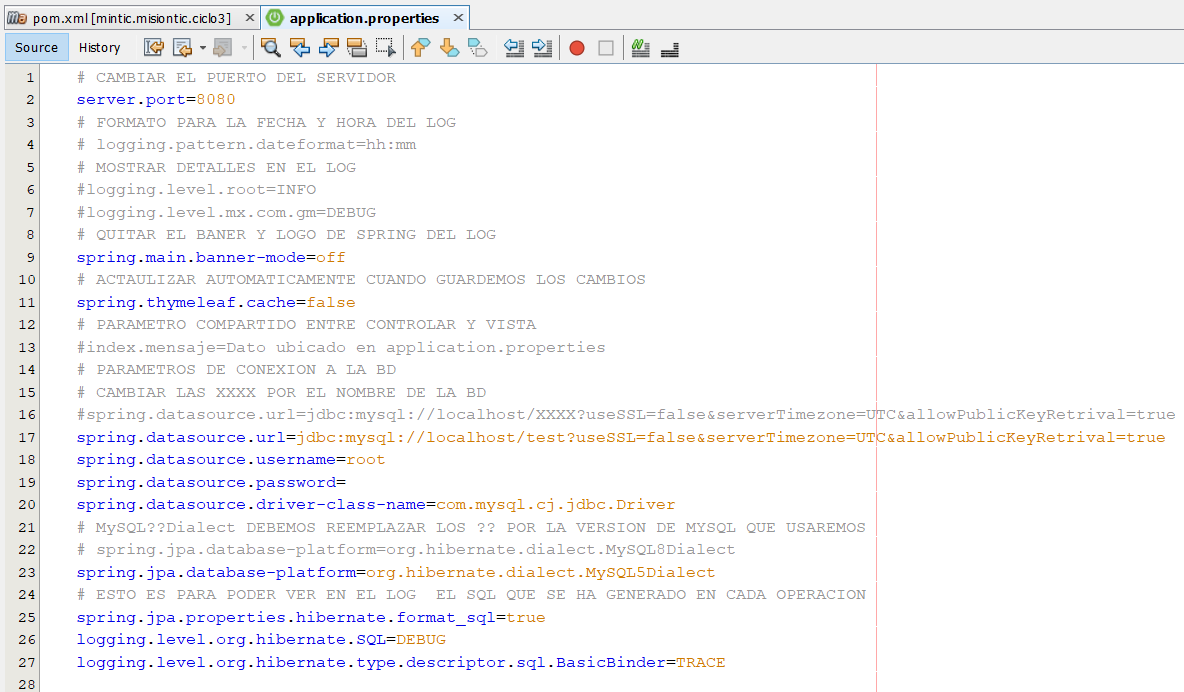




A partir de aquí, Spring Boot nos va a exigir la conexión a una base de datos. La forma mas común de hacer en cualquier framework es mediante las variables de entorno, en este caso, podemos declarar esa variables de entorno en el archivo aplication.properties



Declaramos los parámetros necesarios para realizar una conexión exitosa



*# CAMBIAR EL PUERTO DEL SERVIDOR*

server.port=8080

*# FORMATO PARA LA FECHA Y HORA DEL LOG*

*# logging.pattern.dateformat=hh:mm*

*# MOSTRAR DETALLES EN EL LOG*

*#logging.level.root=INFO*

*#logging.level.mx.com.gm=DEBUG*

*# QUITAR EL BANNER Y LOGO DE SPRING DEL LOG*

spring.main.banner-mode=off

*# ACTUALIZAR AUTOMÁTICAMENTE CUANDO GUARDEMOS LOS CAMBIOS*

spring.thymeleaf.cache=false

*# PARÁMETRO COMPARTIDO ENTRE CONTROLAR Y VISTA*

*#index.mensaje=Dato ubicado en application.properties*

*# PARAMETROS DE CONEXION A LA BD*

*# CAMBIAR LAS XXXX POR EL NOMBRE DE LA BD*

*#spring.datasource.url=jdbc:mysql://localhost/XXXX?useSSL=false&serverTimezone=UTC&allowPublicKeyRetrival=true*

spring.datasource.url=jdbc:mysql://localhost/**mi\_base\_de\_datos**?useSSL=false&serverTimezone=UTC&allowPublicKeyRetrival=true

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

*# MySQL??Dialect DEBEMOS REEMPLAZAR LOS ?? POR LA VERSIÓN DE MYSQL QUE USAREMOS*

*# spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect*

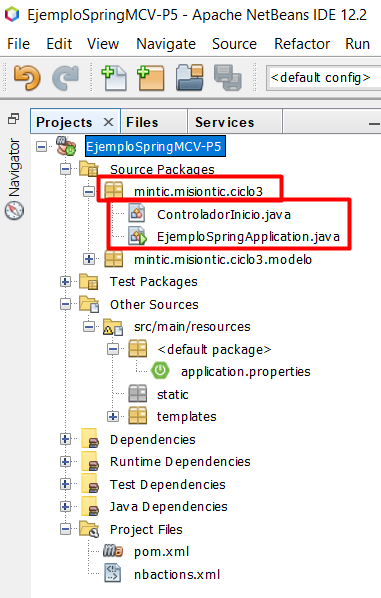
spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect

*# ESTO ES PARA PODER VER EN EL LOG DEL SQL QUE SE HA GENERADO EN CADA OPERACIÓN*

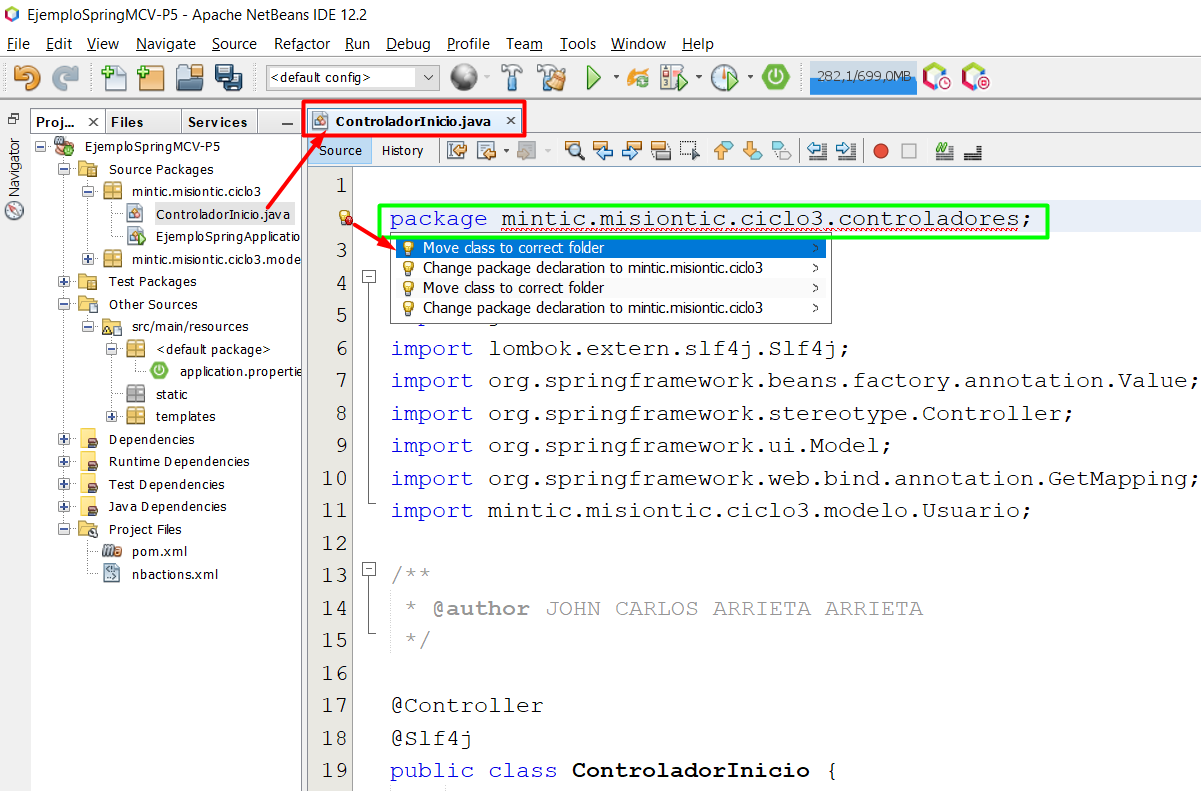
spring.jpa.properties.hibernate.format\_sql=true

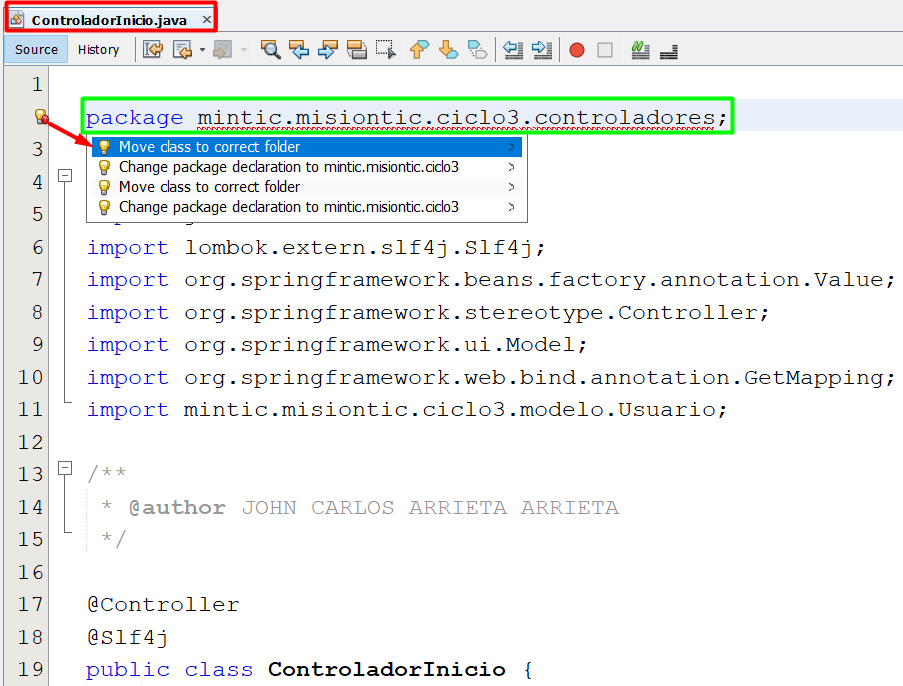
logging.level.org.hibernate.SQL=DEBUG

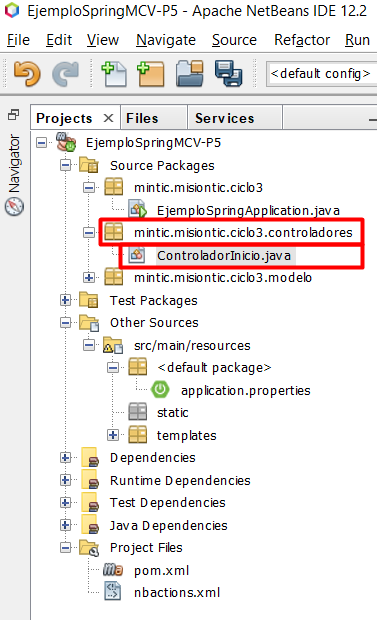
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE



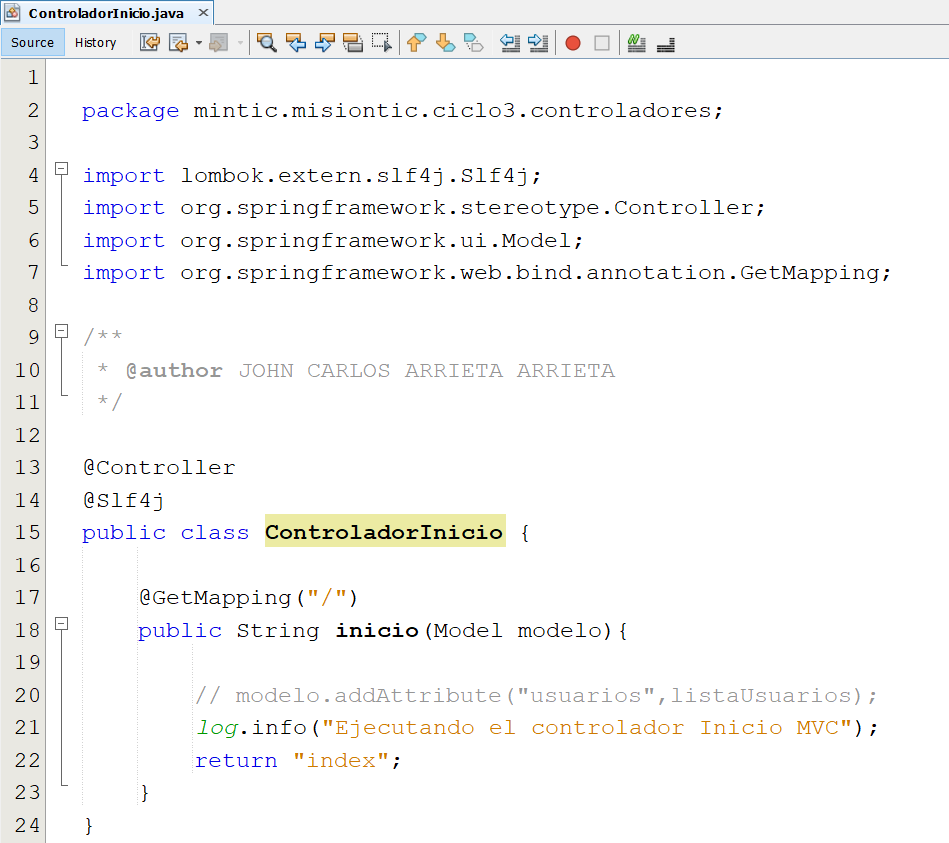
El framework detecta que un Controlador debe estar en una carpeta adecuada.

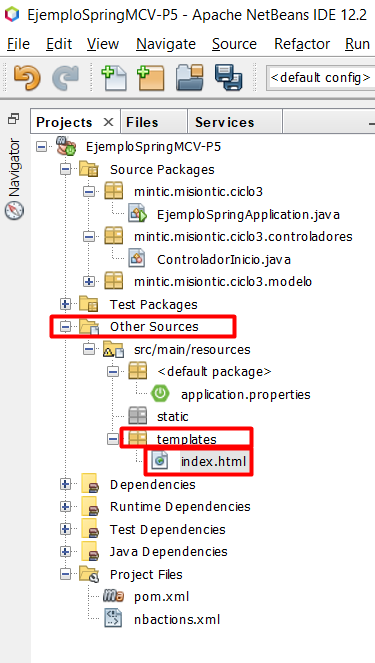




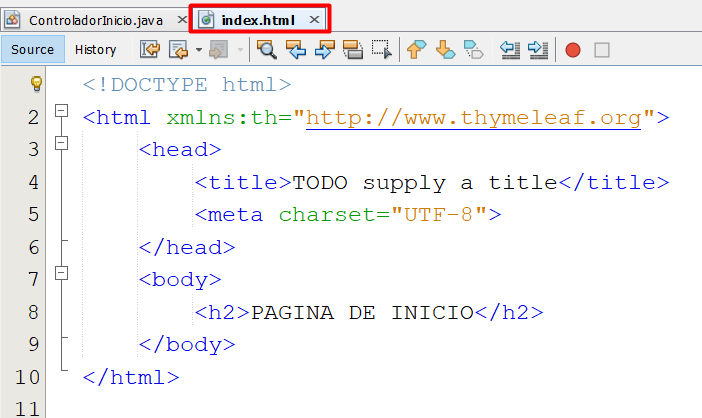


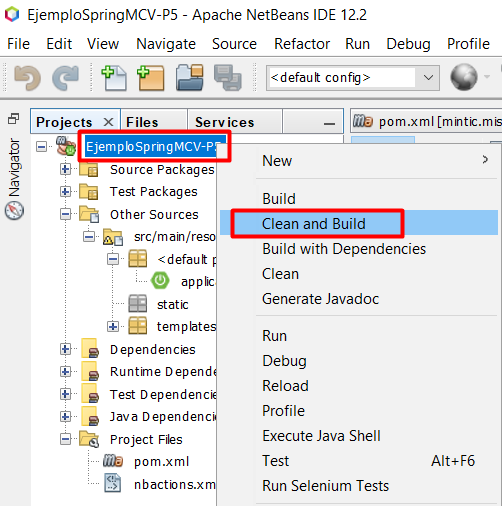
Para que nuestro controlar se comporte como tal, debemos usar la notacion @Controller o @RestController



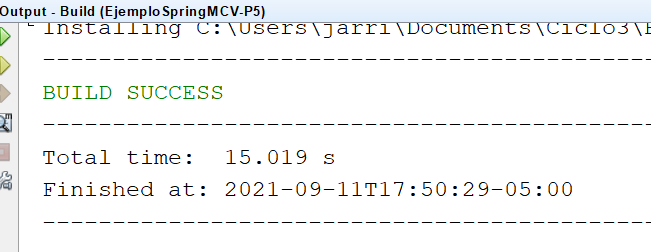


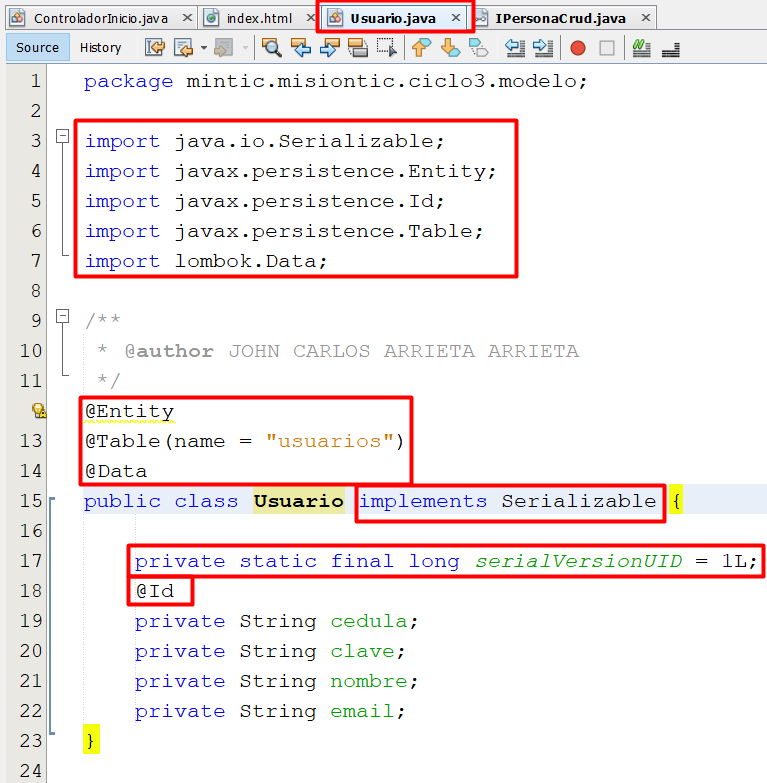
Generamos una vista html. Si necesitamos generar vistas dinámicas, debemos utilizar un motor de plantillas, en este caso, thymeleaf.

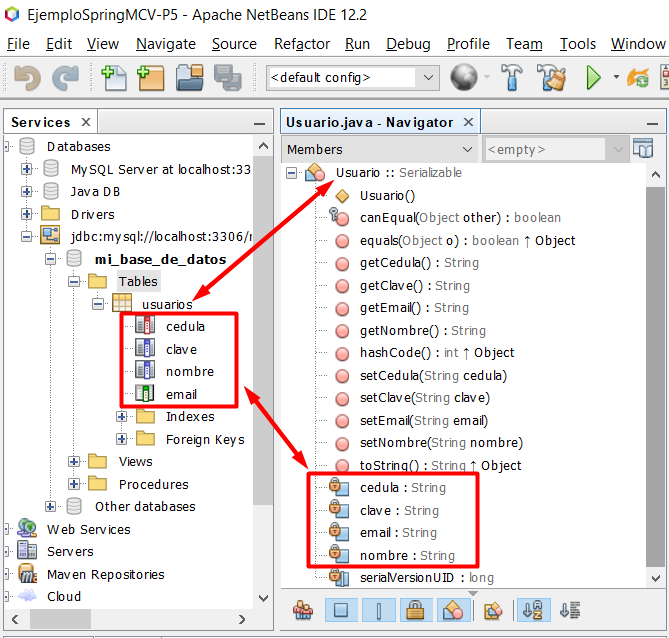




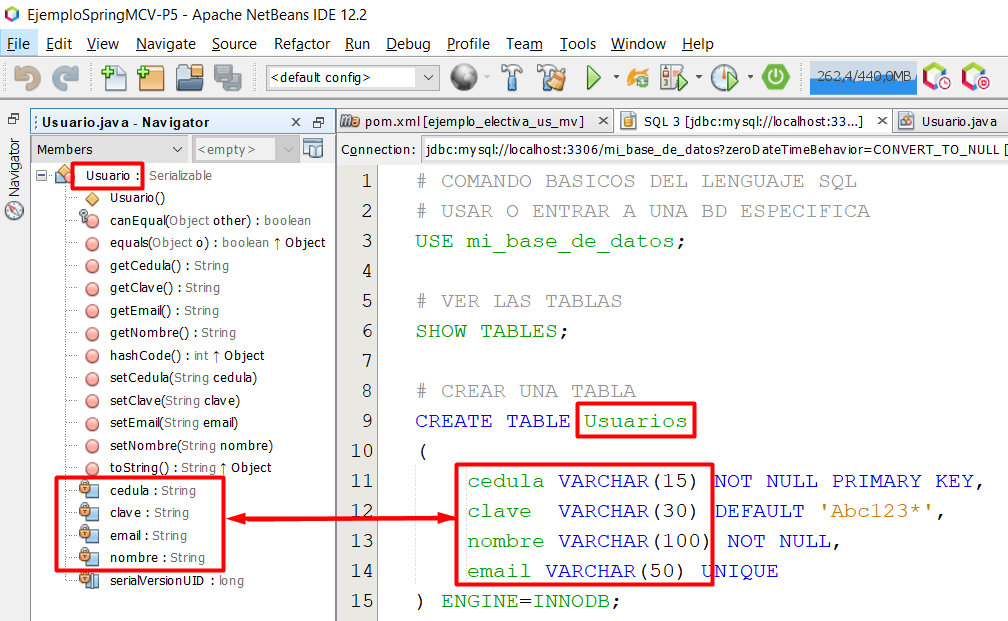
Para que Spring boot pueda manejar nuestra clase Usuario como una entidad o tabla en la base de datos, debemos utilizar la notación @Entity

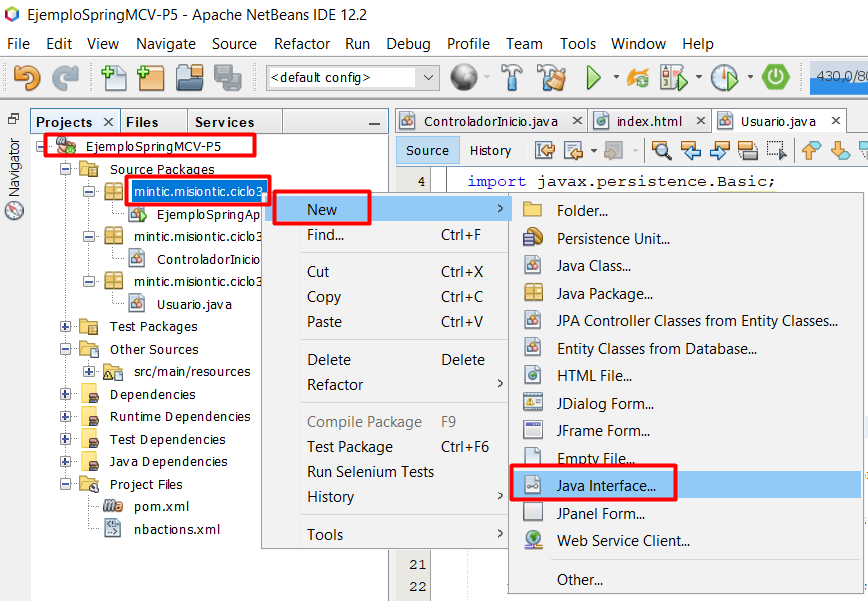




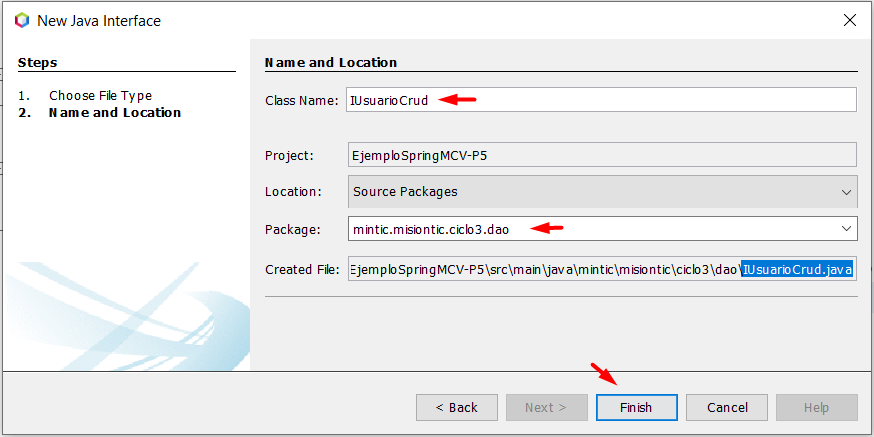


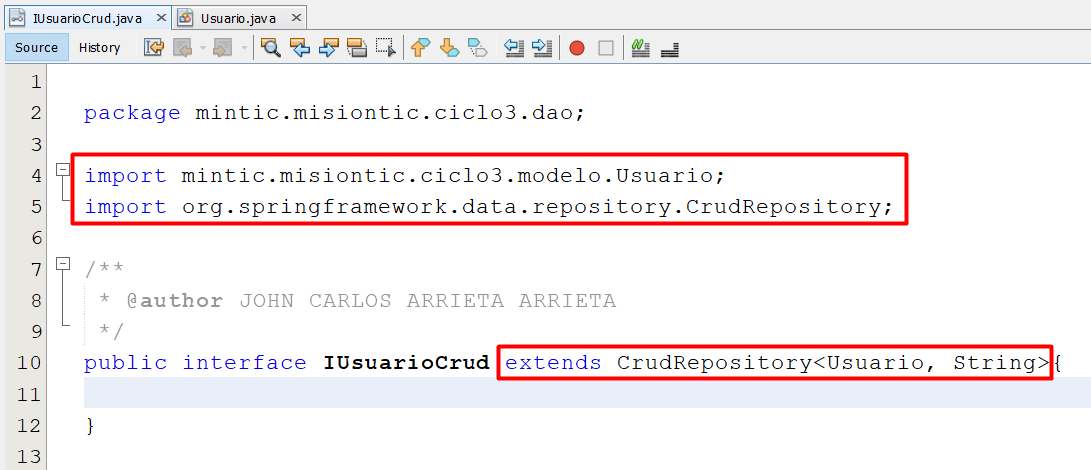
Podemos crear la tabla manualmente en la base de datos

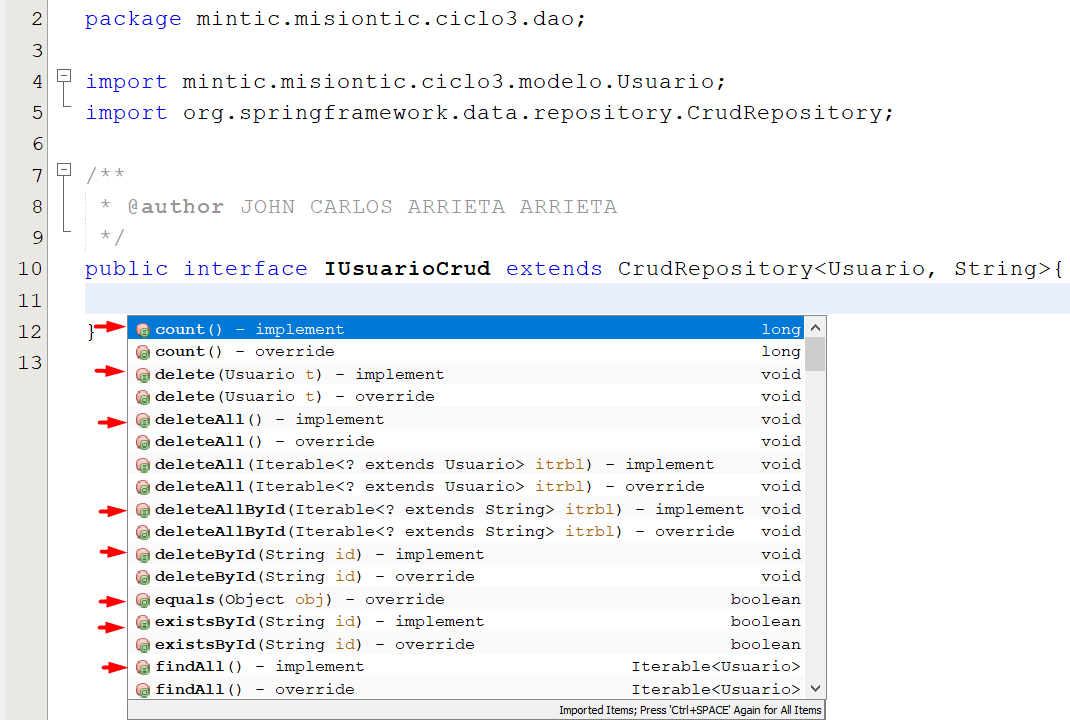


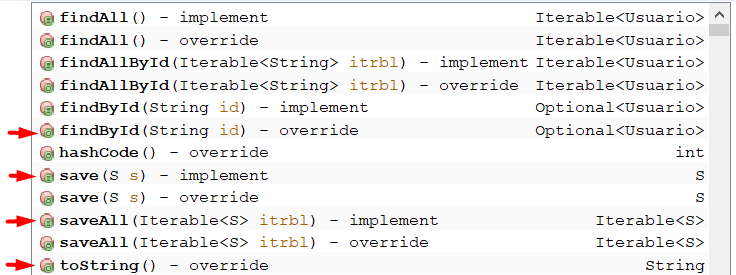


Para que nuestro modelo tenga la funciones de un CRUD, o todos los metodos necesarios para gestionar la base de datos, es necesario crear una interfaz que herede de un repositorio que contenga esas funciones. Éste recibe el modelo Usuario, y el tipo de dato de su llave primaria, al igual que JpaRepository.



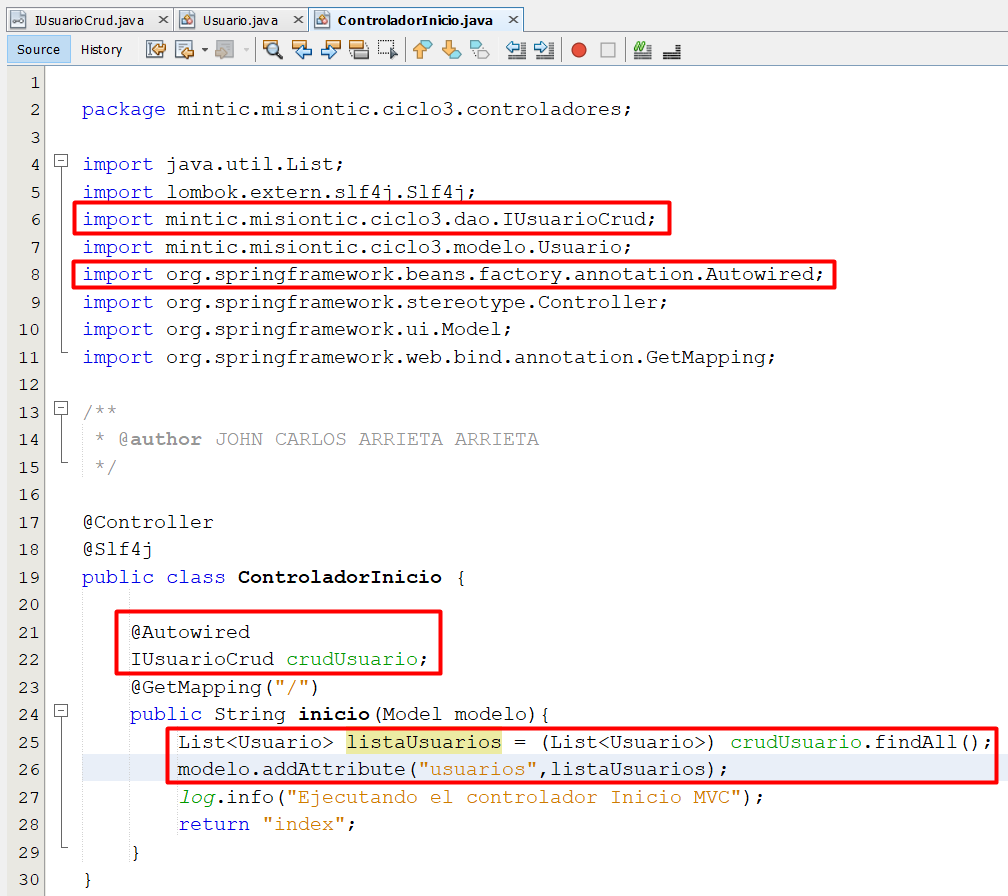






Para poder utilizar los métodos antes mostrador, debemos sobre escribir los métodos de la interface creada anteriormente. La notación @Autowired, nos permite eso facilmente

Añadir la lista de usuarios y añadirlo al modelo.addAtribute podemos renderizar esa informacion en la vista.



Ahora, para implementar los datos obtenidos previamente, debemos utilizar el motor de plantillas para poder mapear esa información.

