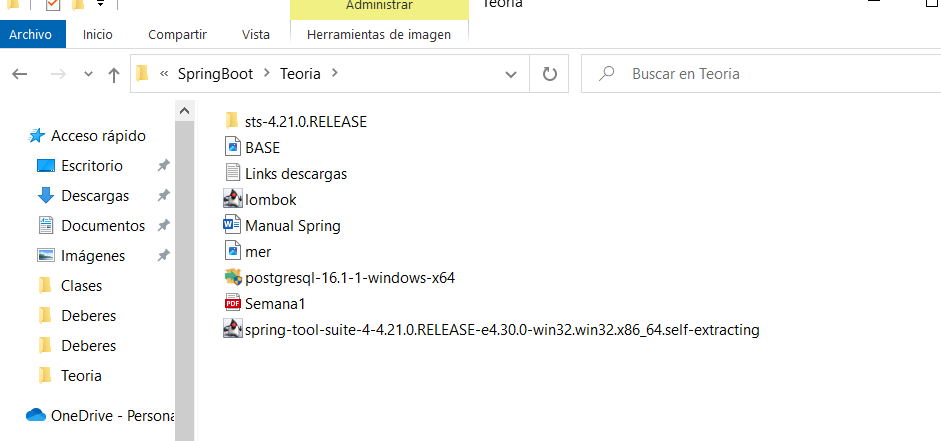
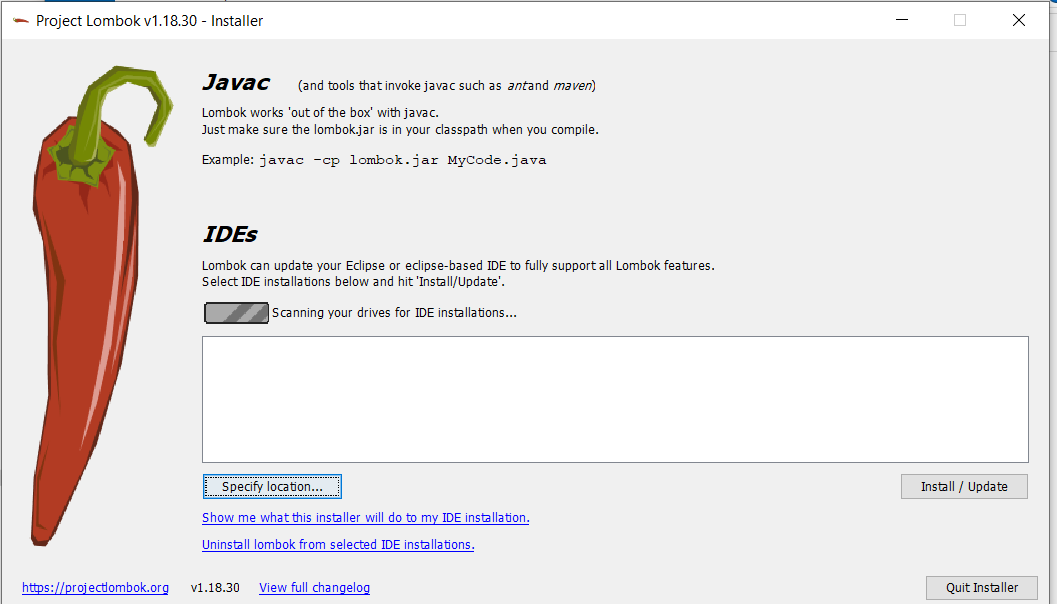
Previamente:

Instalar Spring  
Instalar Postgres  
Instalar Lombok (Opcional)

(Para instalar Lombok) colocar el lombok.jar al mismo nivel donde se encuntre el programa sts-REALESE



Ejecularmos Lomnok y escogemos “Especificar location!



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

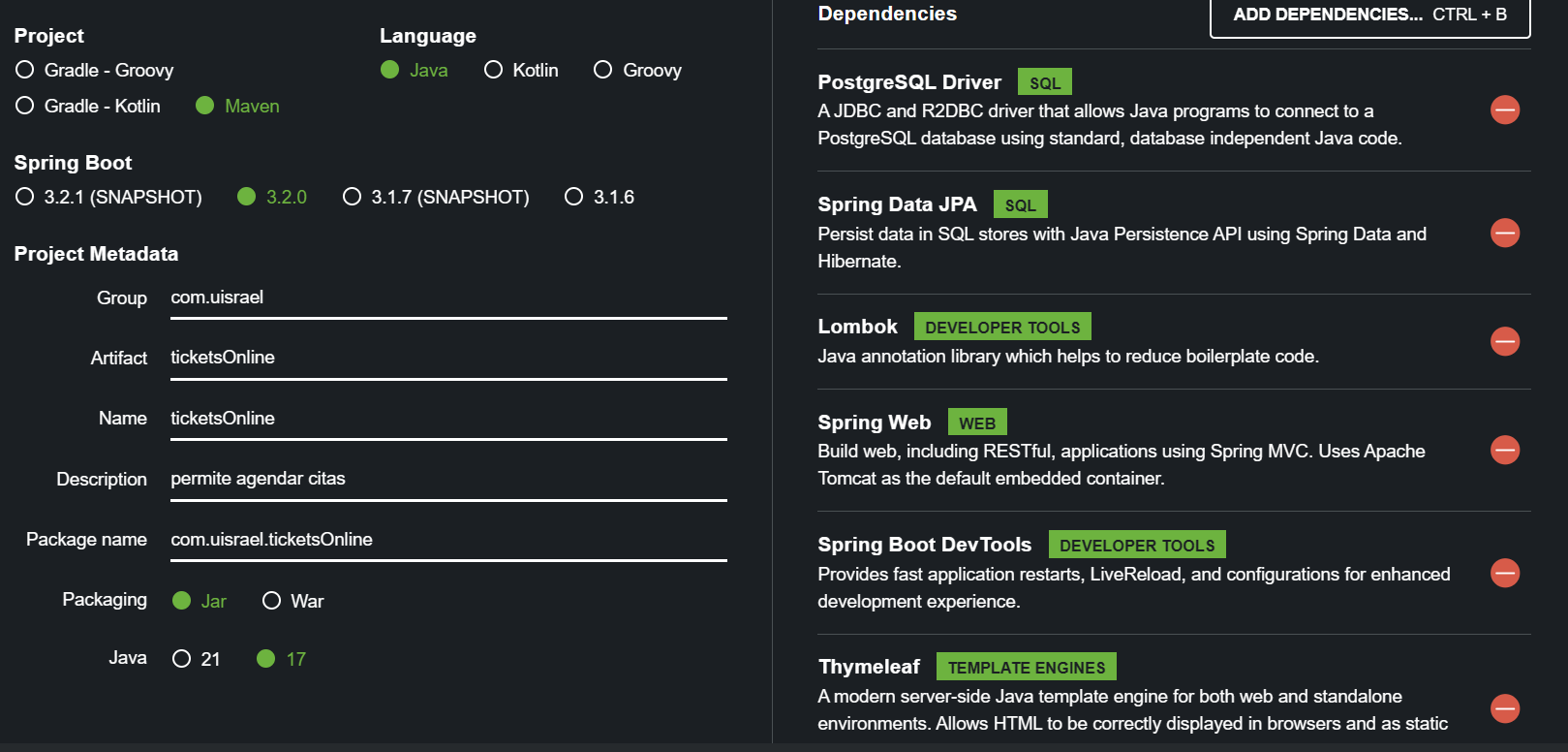
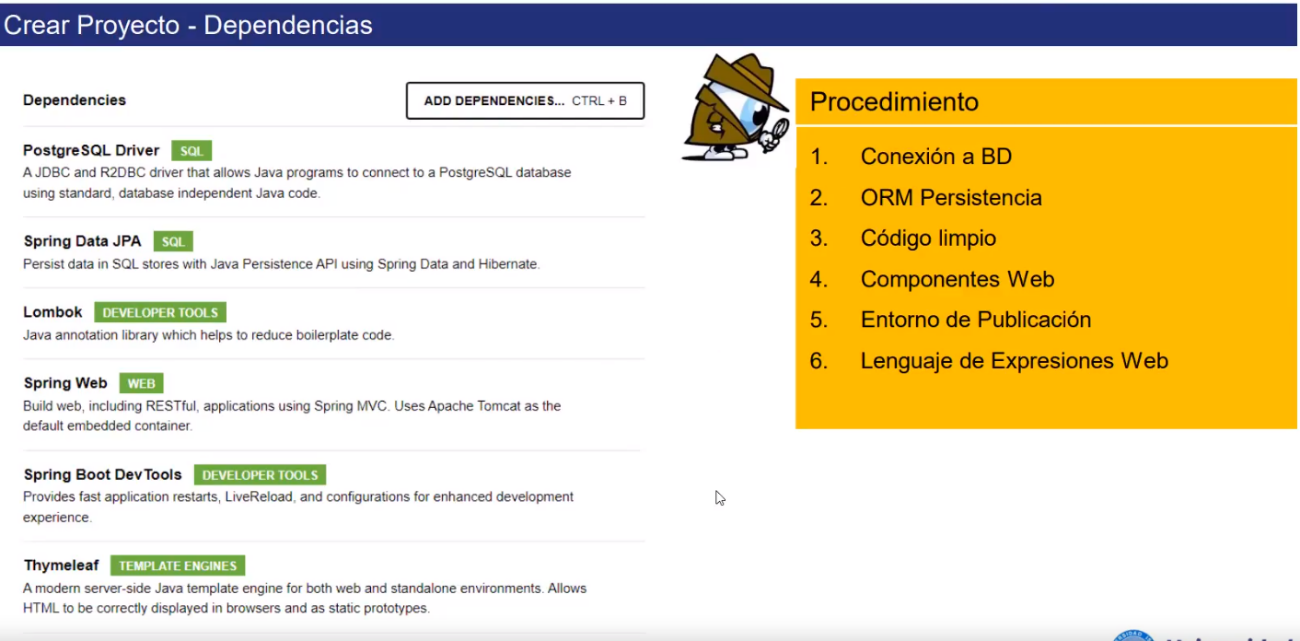
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Abrimos el workspace

1. **CONSTRUIR ARQUITECTURA MAVEN**

METODO 1 **(RECOMENDADO)**: Usar sprimg initializer para crear el proyecto y agregar sus dependencias:

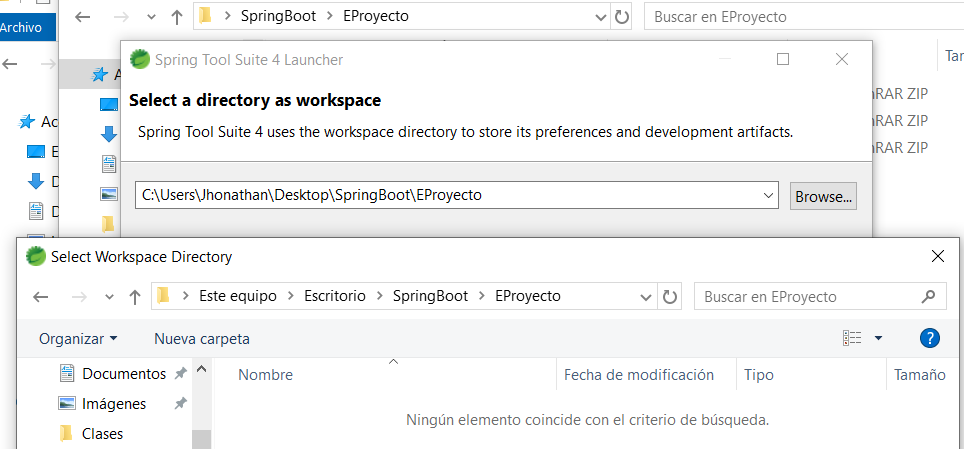


Vamos a crear una carpeta en el escritorio, y una carpeta dentro de ella, ahí colocaremos todos los proyectos que generemos por medio del navegador.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

abrir el proyecto con Spring, te pedirá dónde generar el .metadata; esgemos una carpeta Generica (La carpeta contenedora de los comprimdos)



Descomprimimos el proyecto que descargamos

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Y abrimos el proyecto en Spring:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

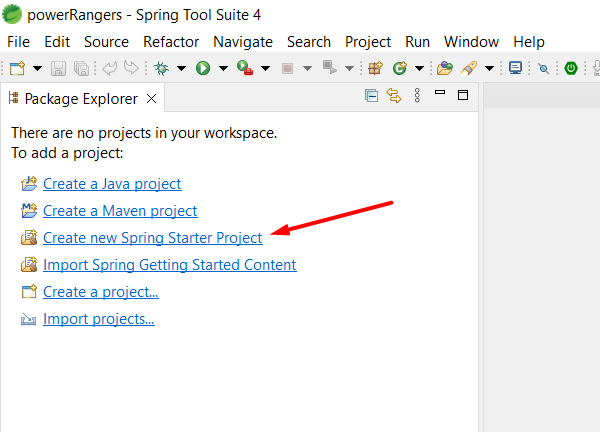
Verificar el archivo POM si tiene todas las dependencias  
Hacemos un update force de Maven

METODO 2: Abrimos Spring te pedirá dónde generar el .metadata; esgemos una carpeta Generica (La carpeta contenedora de los proyectos)

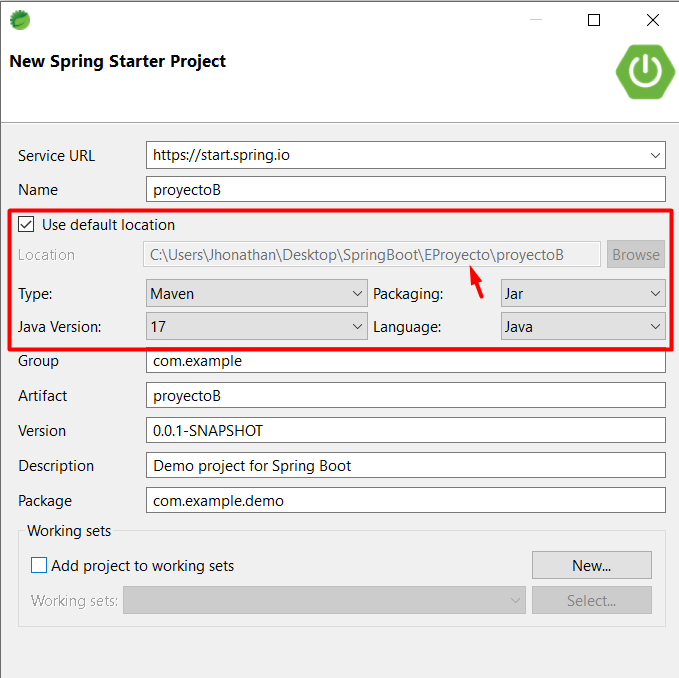
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

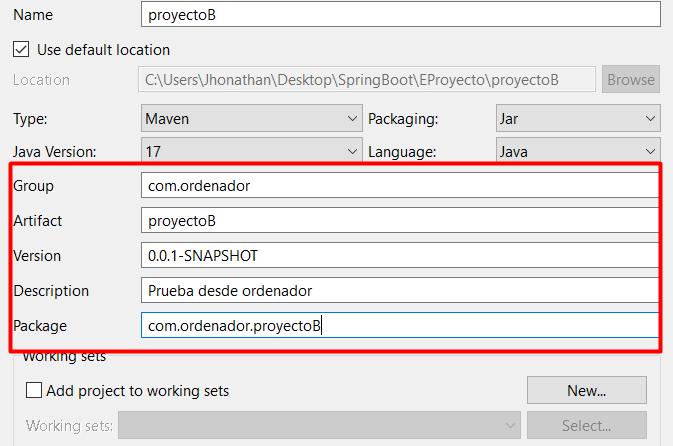
Descripción generada automáticamente

y creamos un proyecto Spring



Recuerda que la carpeta generica es la de EProyecto, esa contiene el .metadata





Equivale A:

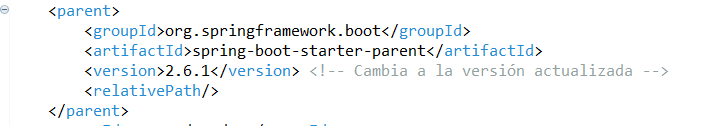
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

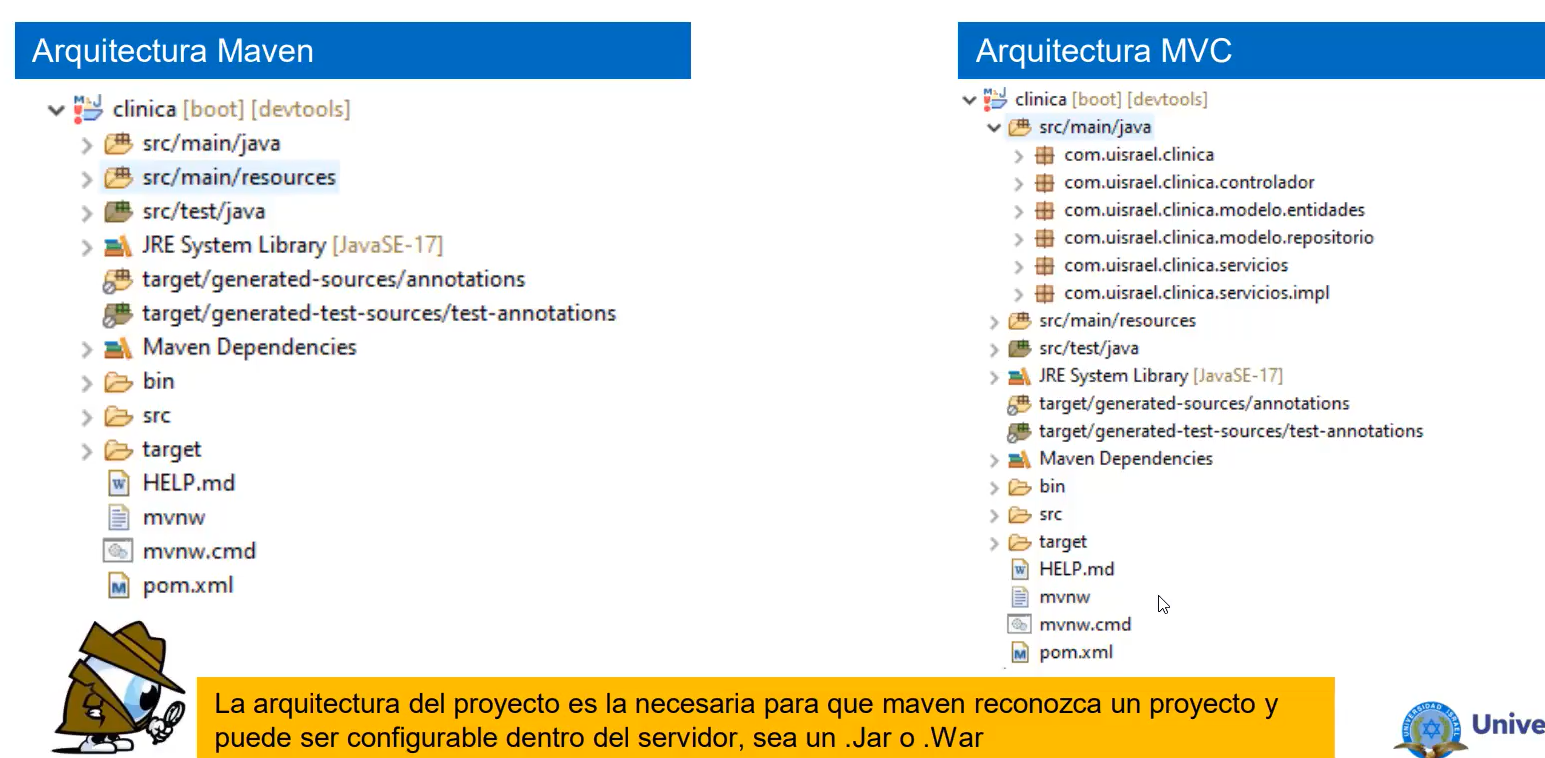
Descripción generada automáticamente

NOTA: La versión creada por ordenador puede no ser compatible con algunas cosas, suele marcar error en el parten del pom; y es necesario cambiarlo así:

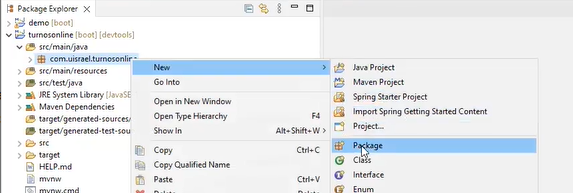


RECOMENDACION: Hacer por el método 1

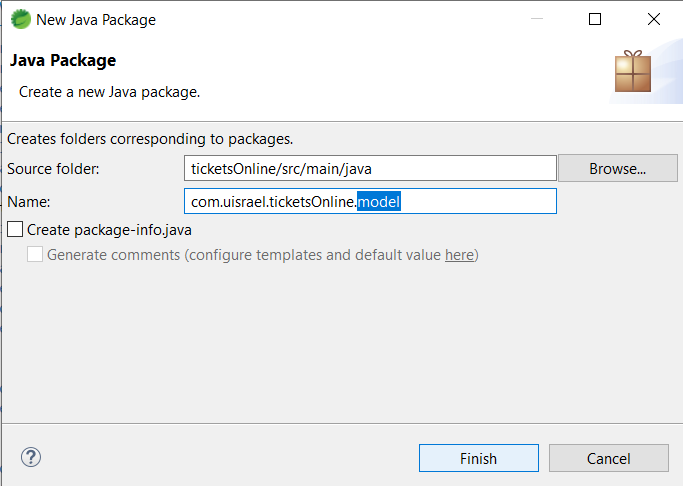
1. **CONSTRUIR ARQUITECTURA MVC**



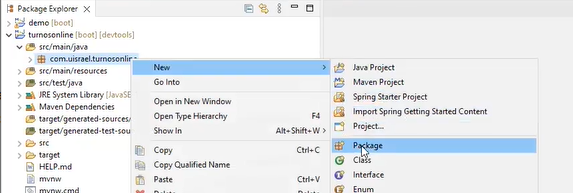
Primero crear modelo:



Agréguese **.model**  al final:



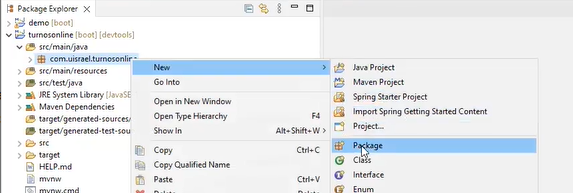
De la misma manera, crear el .**repository**



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Del mismo modo, crear el .**service**



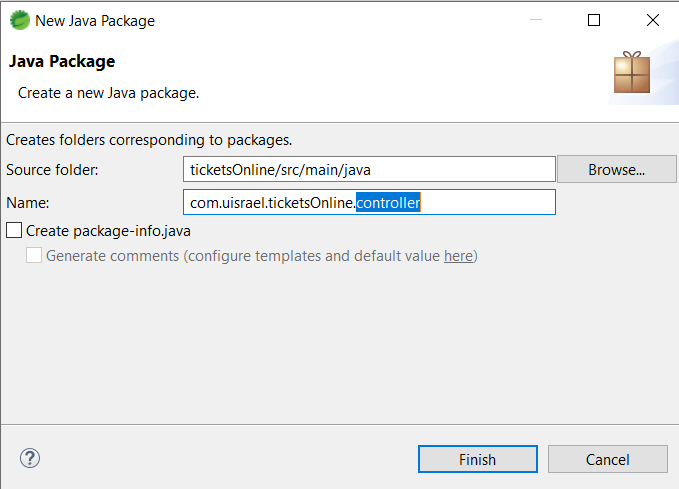
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

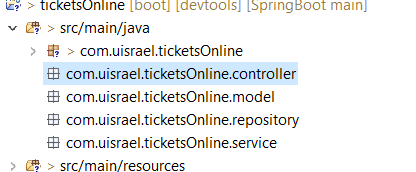
Y por último el controller:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

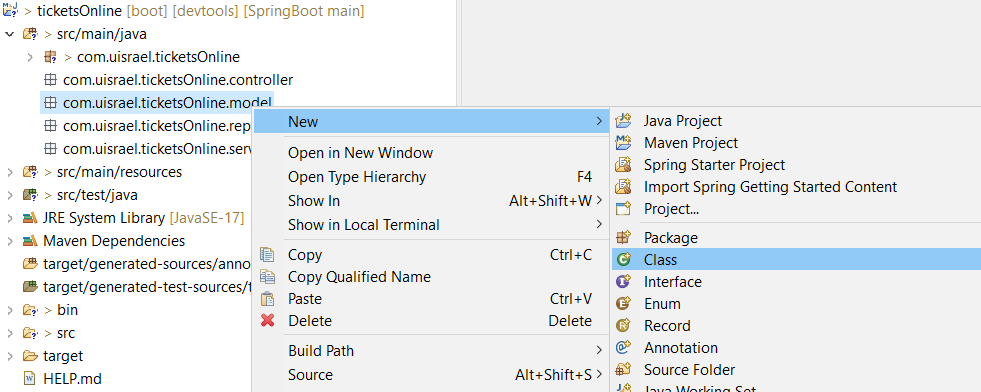


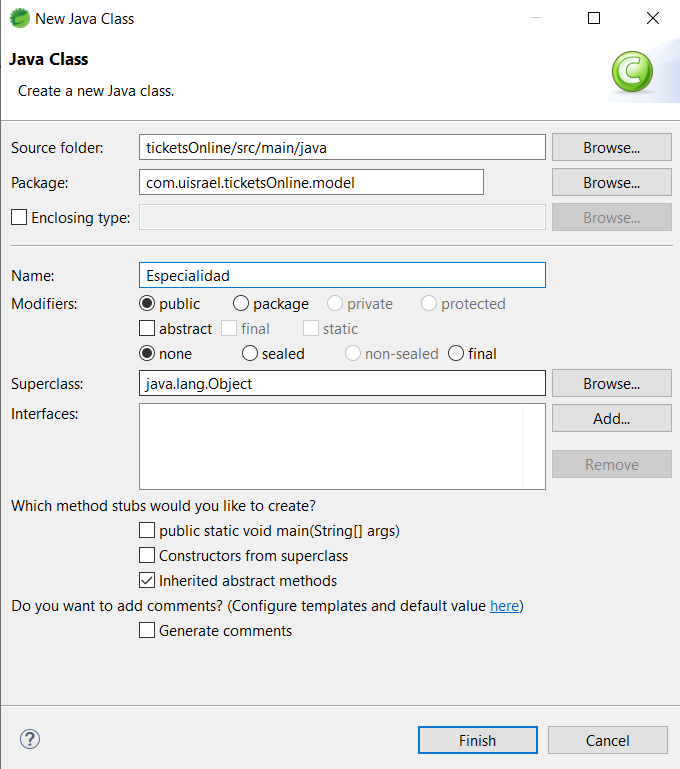
Resultado:



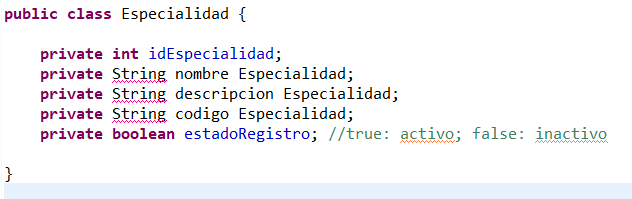
1. **Construir objetos**

**Fase 1:** En model creamos una **clase,** en formato camel case; para el nombre de la clase la primera es mayúscula; ejemplo: Especialidad, Paciente, Médico, HospitalMilitar, etc.





Ahora definimos los atributos de clase, para eso usamos camel case; por ejemplo: nombreDoctor, turnoPaciente, etc. (Tiene que ser los mismos del MER)



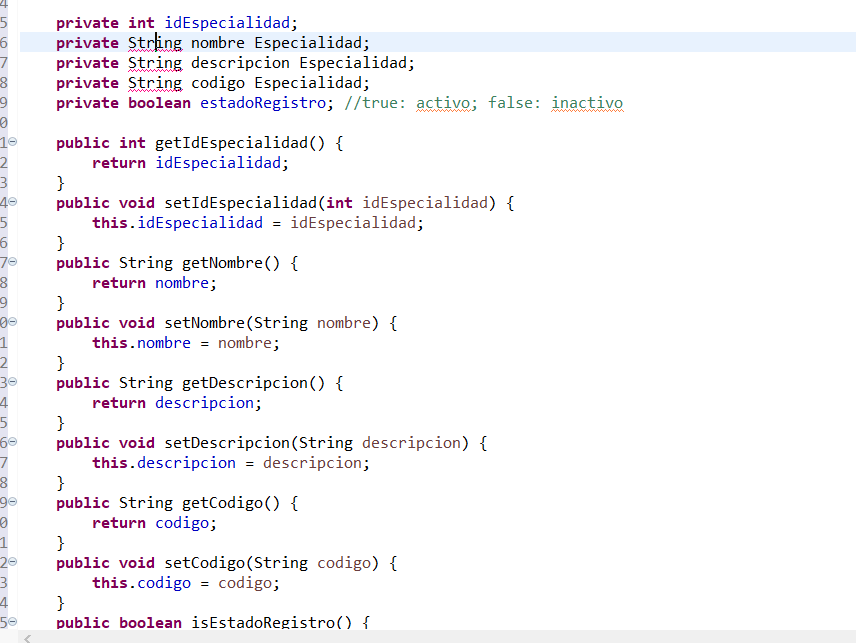
Ahora generamos los getters y setters.

***FORMA 1 Si se desea trabajar de manera tradicional***

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Escogemos de todos, y se genera:



Ahora podemos ver como existen en nuestra clase los métodos:  
Interfaz de usuario gráfica, Texto, Aplicación

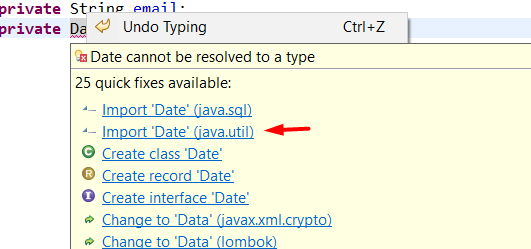
Descripción generada automáticamente

Creamos ahora del mismo modo para el médico:

Texto

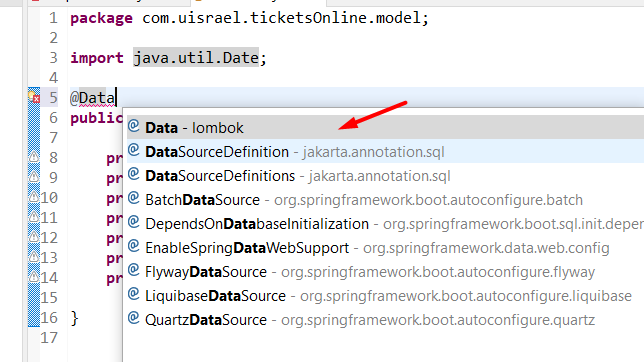
Descripción generada automáticamente

Si trabajamos con Date (fechas) hacemos el import de java.util



Hágase para todas las entidades.

***FORMA 2 (Válido sólo si tienes Lombok instalado)***



Esto generará todos los getters y setters de manera automática y abstracta:

Interfaz de usuario gráfica, Aplicación, Word

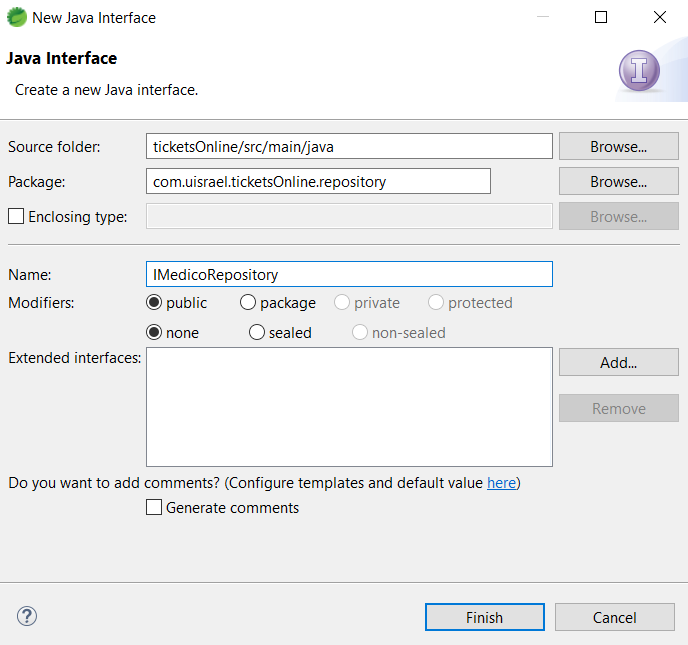
Descripción generada automáticamente

Hágase lo mismo para todas las demás clases

**Fase 2**: Trabajamos sobre el repository, vamos a crear una **interfaz**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



Aquí declaramos los métodos Insertar, Actualizar, Eliminar, etc. (Veremos luego)  
Creamos una interfaz para todas las clases creadas:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Fase 3**: Trabajamos sobre el service, vamos a crear una **interfaz**

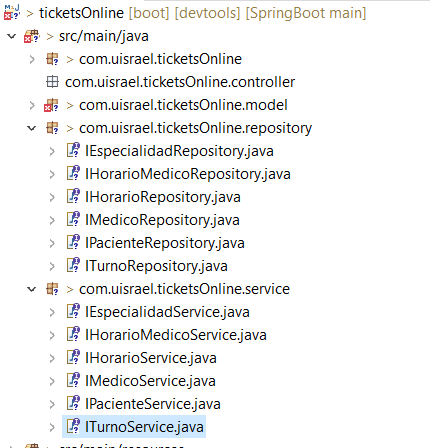
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

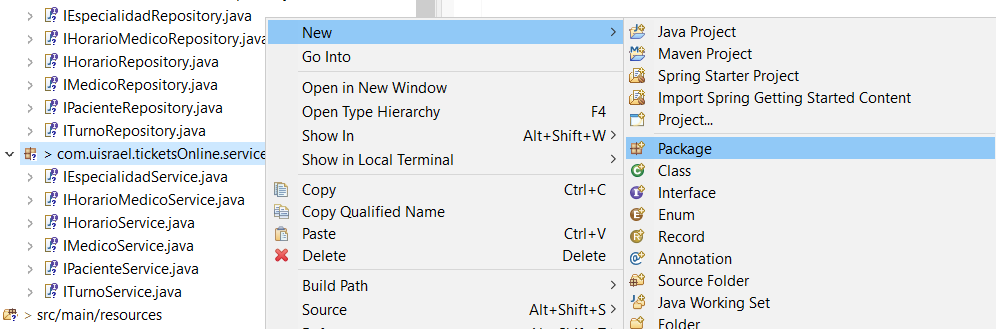
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

De la misma manera creamos para cada entidad su respectivo interfaz de servicio:



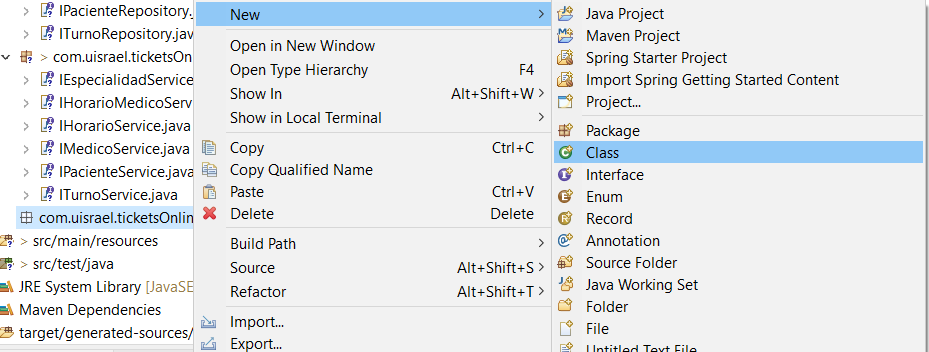
Ahora creamos un paquete dentro del servicio para implementar la interfaz.

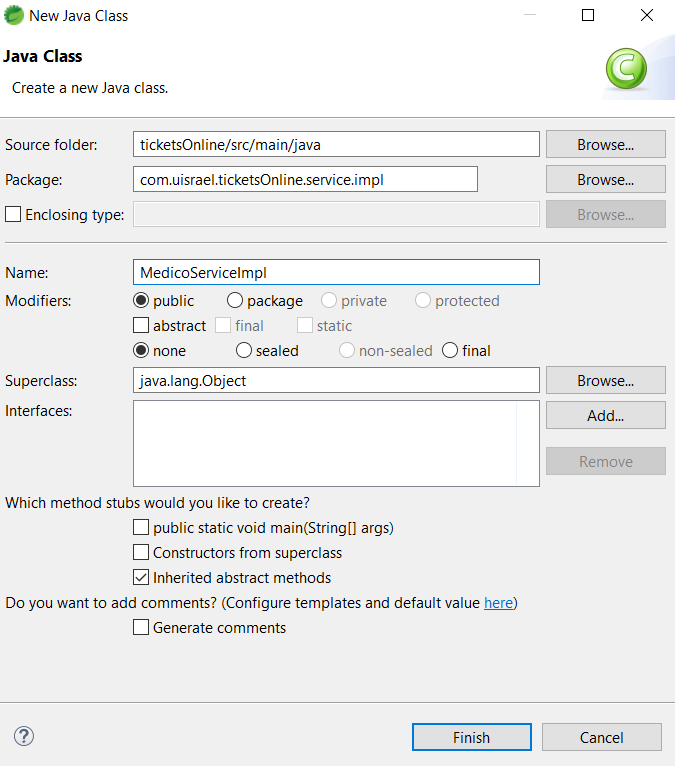


Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Y dentro de ese paquete creamos las clases respectivas:





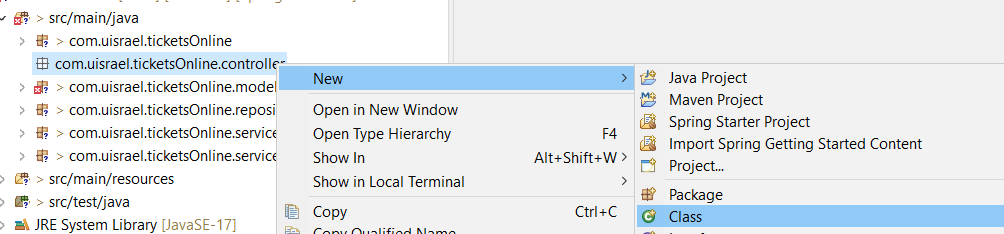
Esta clase implementa esta interfaz:

Texto, Aplicación

Descripción generada automáticamente

Hacemos igual, las demás clases para cada interfaz.

**Fase 4**: Trabajamos sobre el controlador, vamos a crear una **clase**



Creamos el controlador para cada entidad:

