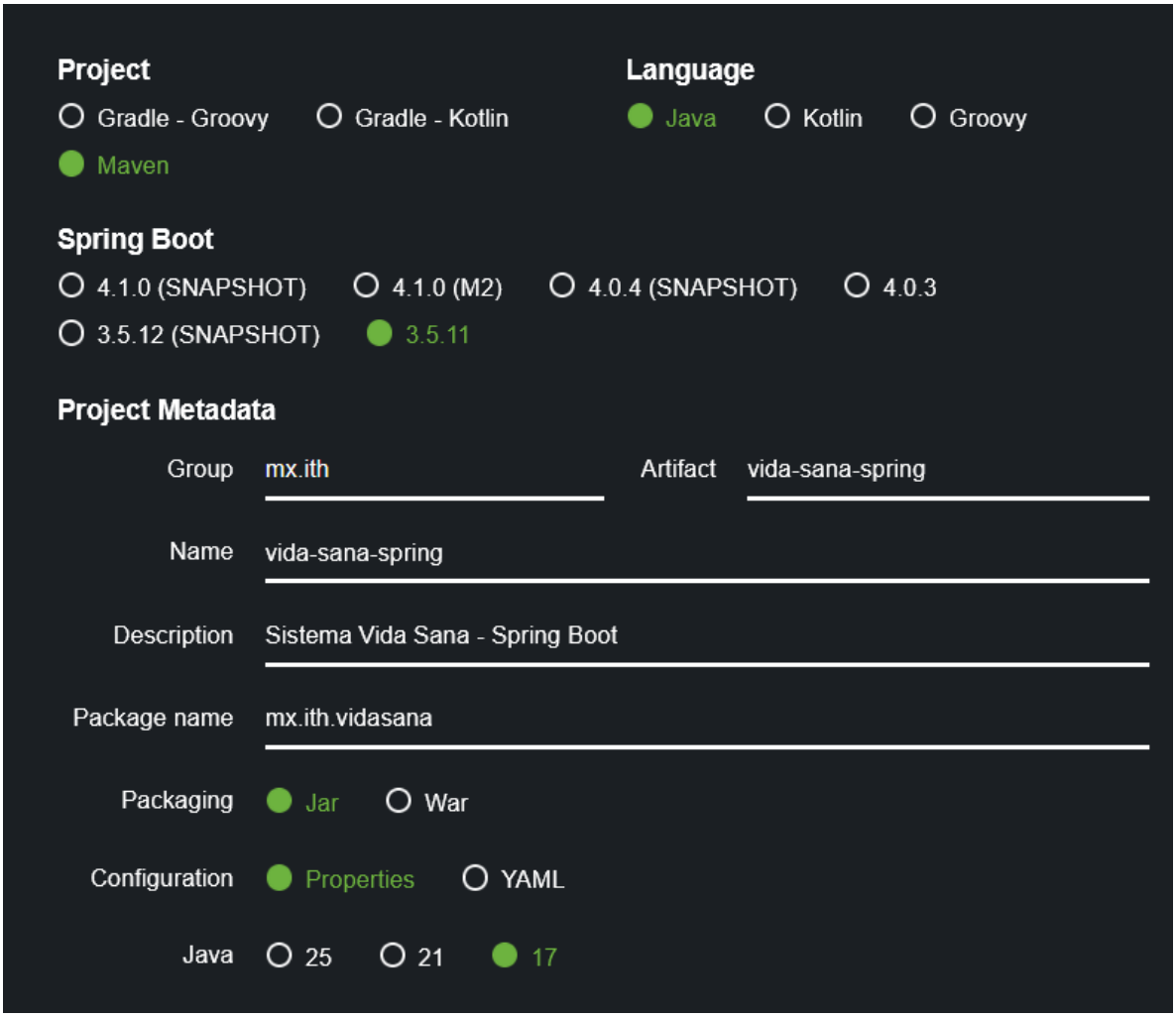
Práctica 2

Framework Spring-Boot

Vida Sana

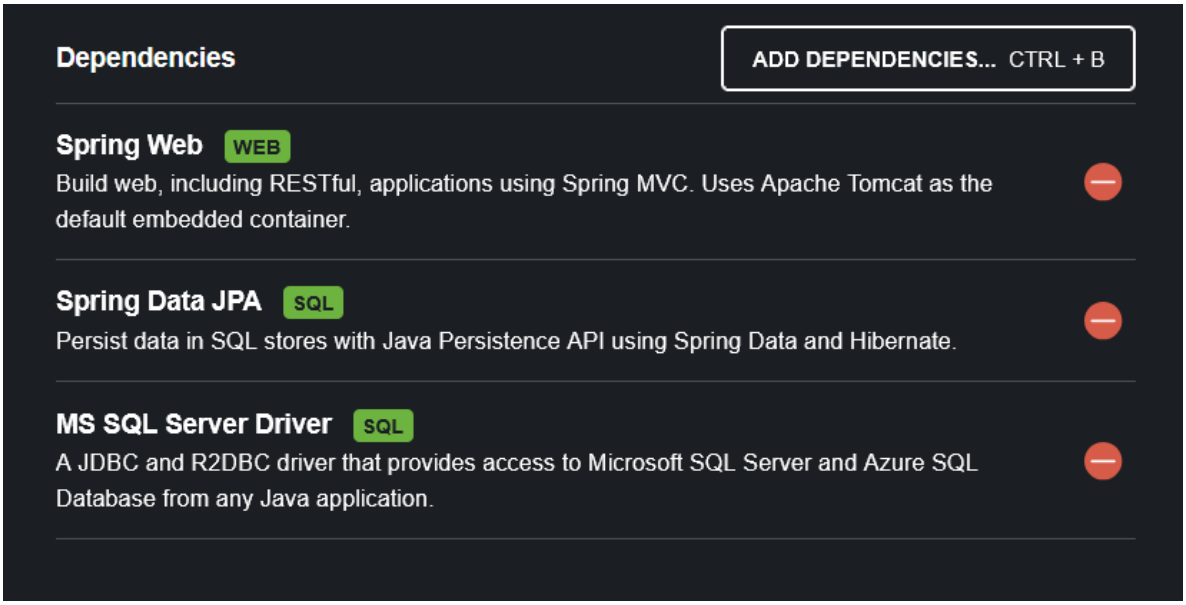
Spring Boot es una herramienta de código abierto basada en Java que simplifica drásticamente la creación, configuración y despliegue de aplicaciones web.

1) **Ingrese** a https://start.spring.io/ y cree un proyecto con los siguientes datos:



Antes de continuar, recuerde que debe tener su base de datos con su loginMedico y al menos una tabla llamada Pacientes.

Al lado derecho agregue las siguientes dependencias. Son las bibliotecas que su proyecto utilizará para crear los endpoints, conectarse a SQL Server, y acceder a SQL Server desde Java (JPA). Esta última API facilita mucho los métodos CRUD.



Presione el botón de "Generate", y se le generará un archivo vida-sana-spring.zip. Ubíquelo desde su carpeta de descargas, muévalo a un folder (cuyo nombre NO contenga espacios) y descomprima sus contenidos. Este archivo contiene todo lo necesario, inclusive un servidor de aplicaciones (Tomcat) para que la aplicación sea desplegada.

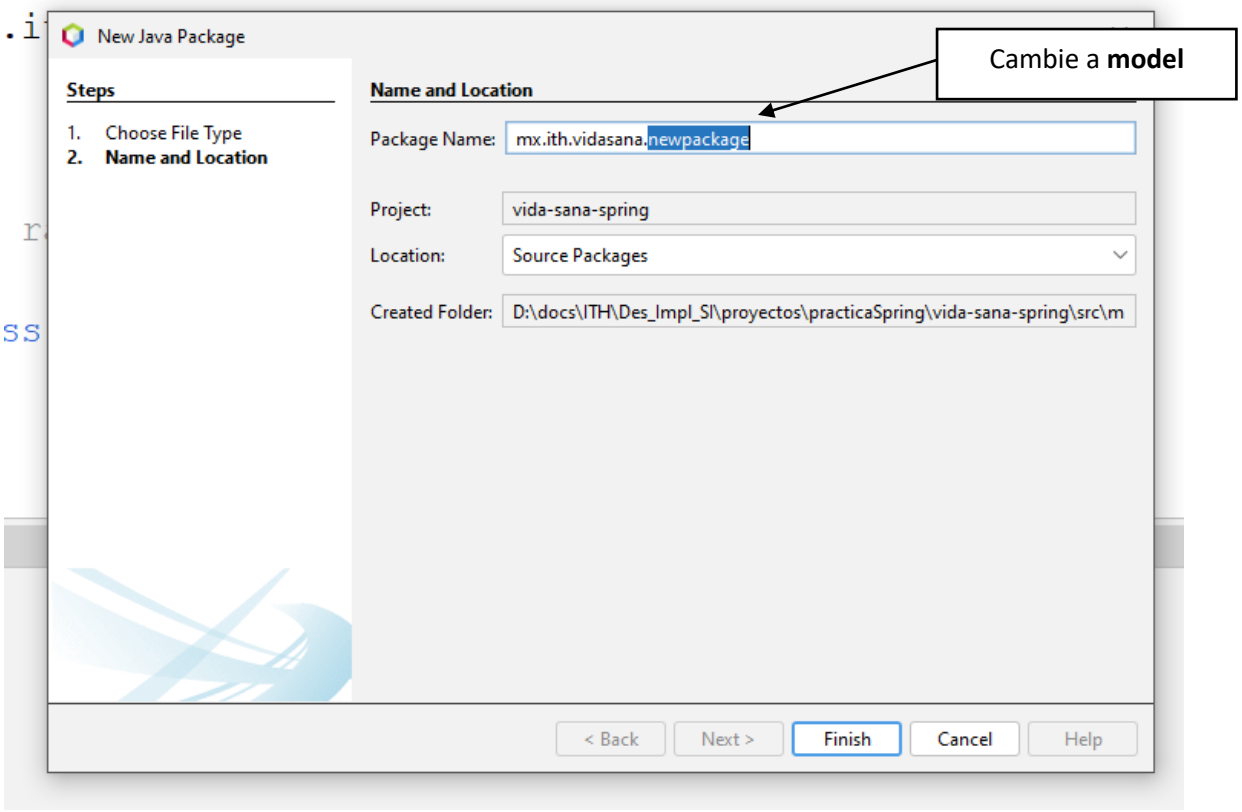
2) **Abra su proyecto en NetBeans (o VSCode)** y configure la conexión a SQL Server. Other Sources>src/main/resources>Default Package>application properties

Configure el archivo *properties* (recuerde cambiar al nombre de su servidor donde dice *suServidor*, o en su defecto utilizar localhost:1433

```
spring.datasource.url=jdbc:sqlserver://suServidor;databaseName=vida_sana;encrypt=true;trustServerCertificate=true

spring.datasource.username=loginMedico

spring.datasource.password=medico123

spring.jpa.hibernate.ddl-auto=none

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServerDialect

spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

Note cómo ahora configuramos un archivo properties en lugar de hacer nuestro string de conexión.

3) **Crear entidad Paciente.**
   Dentro de *Source Packages*, haga click derecho a su paquete mx.ith.vidasana, y seleccione new package, de tal manera que el nombre del paquete nuevo que está a punto de crear le anteceda el nombre su paquete padre, ingrese *model.* Al final el nombre de su package deberá quedar mx.ith.vidasana.model. Si está utilizando VSCode, vaya hasta donde está su archivo vidaSanaApplication.java y a ese nivel cree folders llamados *model*, *controller* y *repository*.



   Dentro del package *model* cree una clase llamada Paciente. Click derecho a su package *model*, New> Java Class.

Recuerde que por cada atributo que declare en esta clase, deberá existir un campo en su tabla *Pacientes* en su base de datos.

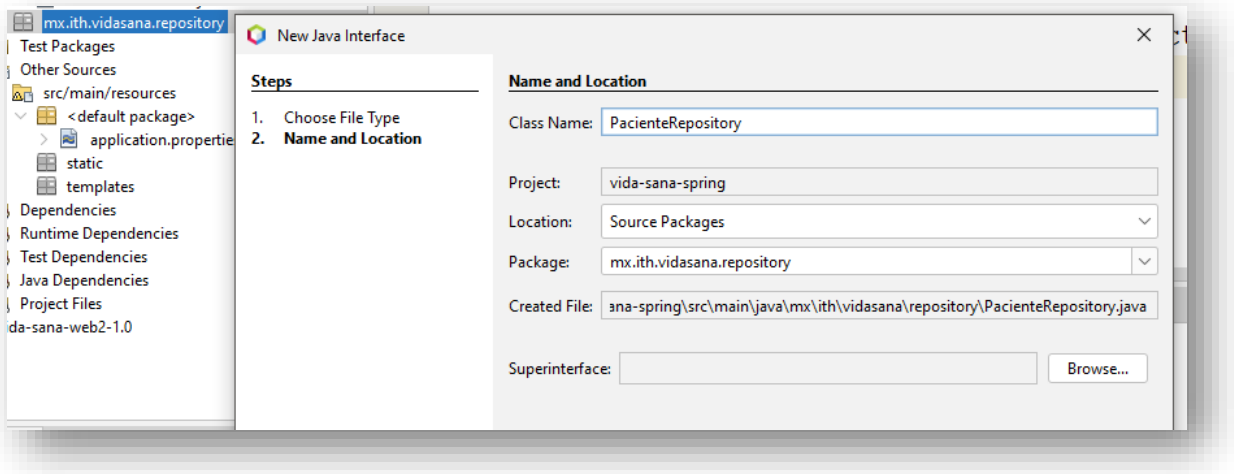
Vea las anotaciones Entity, y *Table*. En esta parte se hace un verdadero mapeo desde un objeto de la clase Paciente hacia una tabla *Pacientes*. También note cómo cada atributo se mapea hacia una columna, así como se define el ID.

```
@Entity
@Table(name = "Pacientes")
public class Paciente {

   @Id
   @Column(name = "IdPaciente")
   private int idPaciente;

   @Column(name = "Nombre")
   private String nombre;
   @Column(name = "Telefono")
   private String telefono;


   public int getIdPaciente() {
      return idPaciente;
   }

   public void setIdPaciente(int idPaciente) {
      this.idPaciente = idPaciente;
  }

   public String getNombre() {
      return nombre;
   }

   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   public String getTelefono() {
      return telefono;

   }

   public void setTelefono(String telefono) {
      this.telefono = telefono;
   }

}
```

4) **Cree otro package** llamado repository (Esto es el equivalente de nuestro paquete DAO de prácticas anteriores)

Lo creará de la misma manera que los otros paquetes: Click derecho a su paquete mx.ith.vidasana, New>Java Package y llámelo repository. (Si utiliza VSCode ya creó esta carpeta en un paso anterior)

5) **Cree una interface**. En su package repository, haga click derecho New>Java Interface y llámela *PacienteRepository.*



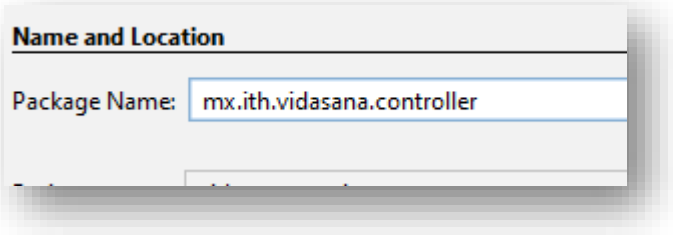
Agregue el siguiente *extends*¸y corrija los import necesarios.

```
public interface PacienteRepository  extends JpaRepository<Paciente, Integer> {


}
```

6) **Cree el Controlador.** El Controller centraliza las llamadas de los clientes (si es un GET, POST, etc)

Haga click derecho a su paquete mx.ith.vidasana, y seleccione New> Java package, de tal manera que el nombre del paquete nuevo que está a punto de crear le anteceda el nombre su paquete padre, ingrese *controller*

Al final el nombre de su *package* deberá quedar mx.ith.vidasana.controller



Si utiliza VSCode ya creó esta carpeta en un paso anterior.

Agregue una clase en este paquete (click derecho a su package controller, New> Java Class) y llámela PacienteController

Agregue la anotación @RestController y arregle los import.

```java
@RestController
public class PacienteController {
    @GetMapping("/hola")
    public String hola(){
        return "Hola desde Spring";
    }
}
```

Agregue un endpoint llamado hola como se muestra arriba.

7) Ejecute su proyecto

Abra una consola de comandos, sitúese en la carpeta de su proyecto y ejecute el siguiente comando

```
mvnw spring-boot:run
```

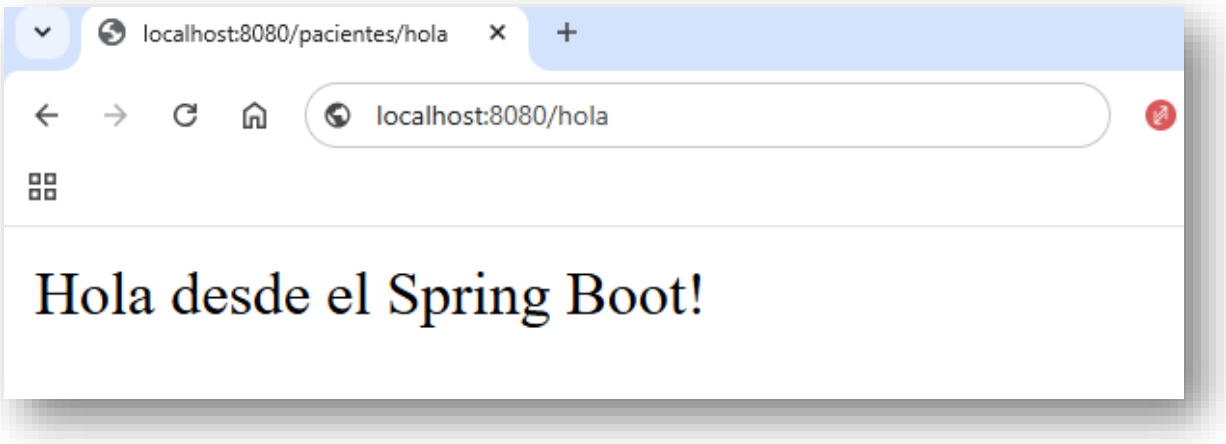
Si abre una terminal desde VSCode, a este comando le antepone un **./**

```
./mvnw spring-boot:run
```

Observe Spring Boot iniciar

Una de las últimas líneas de la console le notifica que el servidor Tomcat ha iniciado en el puerto 8080. Abra su navegador y haga solicitud al endpoint hola

http://localhost:8080/hola

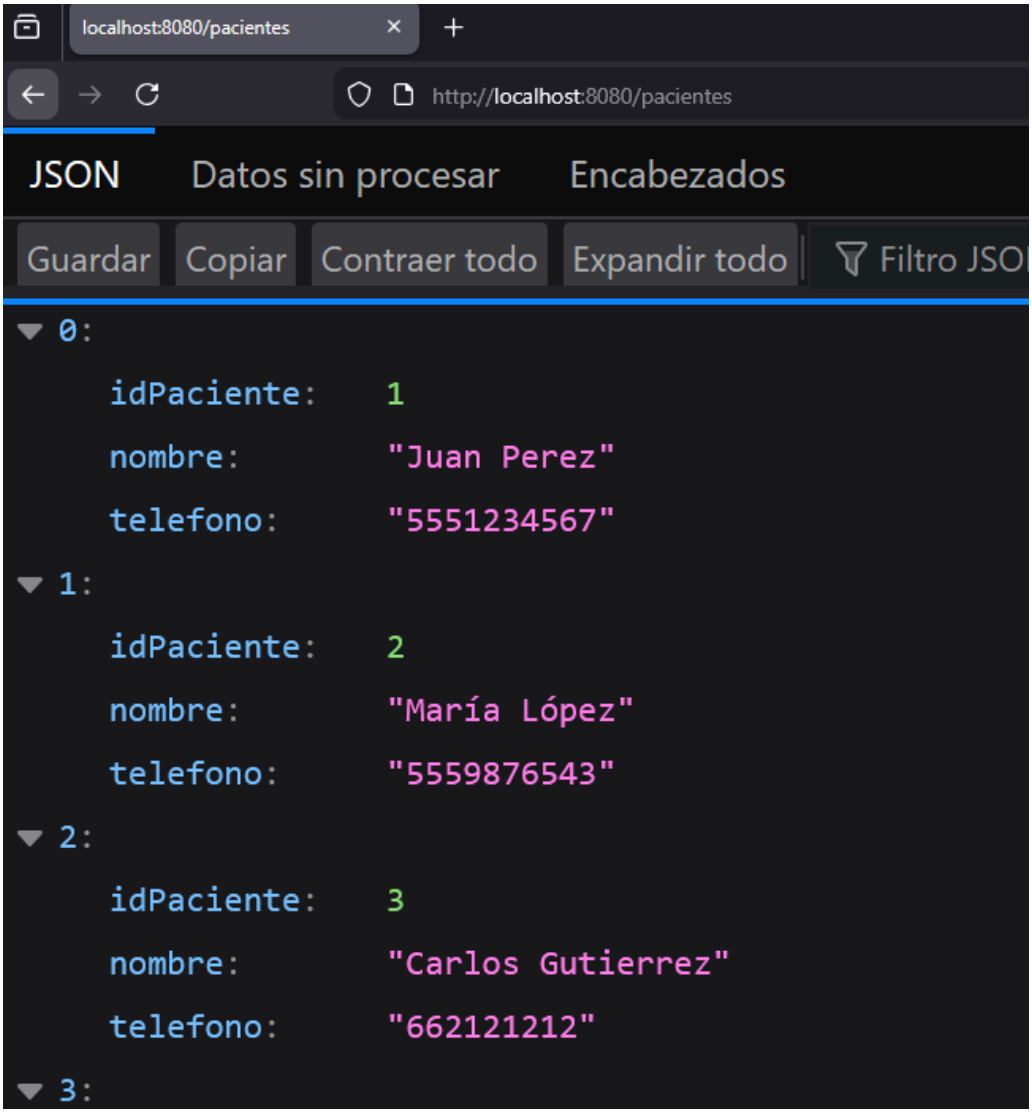


Note como en este punto no dependemos de NetBeans para ejecutar nuestro proyecto, por lo que prácticamente puede utilizar Visual Studio Code o cualquier otro IDE compatible con Java para administrar su proyecto.

8) **Mostrar pacientes.** Modifique su PacienteController de tal manera que va a hacer otro endpoint padre llamado “pacientes”. Note como estamos mandando llamar un método llamado findAll. Éstos son métodos de la biblioteca de Spring JPA.

```
@RestController
@RequestMapping("/pacientes")
public class PacienteController {

        private final PacienteRepository repo;
       public PacienteController(PacienteRepository repo) {
              this.repo = repo;
       }

    @GetMapping
   public List<Paciente> listar() {
      return repo.findAll();
   }


   @GetMapping("/hola")
   public String hola(){
      return "Hola desde Spring!";
   }

}
```

Haga solicitud al endpoint pacientes y verá el listado de sus pacientes en formato JSON a través de su navegador.



Hasta este punto ya está haciendo una operación real en su base de datos a partir de la llamada de un URL, con el verbo GET. Recuerde que este diseño de acomodar las llamadas mediante URLs para hacer las operaciones en una BD se conoce como arquitectura REST.

Adjunte evidencia como esta última captura de pantalla junto con las respuestas a las siguientes preguntas y súbalo a Google Classroom.

1. ¿Qué es RestController?
2. ¿Qué es JSON?
3. ¿Para qué tenemos que crear un model?
4. ¿Por qué cree que en ningún lado estamos viendo un “SELECT * FROM Pacientes”?