# PRÁCTICA 3 — Spring Boot + JPA + HTML + fetch + SQL Server (parte 2)

---

# 1. Abra su proyecto anterior *pacientes*

Inicie Visual Studio Code y abra la carpeta de su proyecto *pacientes* que hizo en la parte 1. En esta parte 1 hicimos un Controlador REST que escuchaba GET en el endpoint /api/pacientes y que, al acceder a él nos regresaba un JSON de la tabla pacientes. Desde el index.htnl accedimos a ese endpoint con la función fetch de Javascript. Recorrimos el resultado y afectamos la tabla para que se desplegara. También dimos estilo a la tabla utilizando Bootstrap.

# 2. Modificar controller REST

Vaya a su PacienteContoller y agregue un nuevo método HTTP al endpoint api//pacientes. En la primera parte de esta práctica, su endpoint **/api/pacientes** sólo contestaba a GET, e invocaba al método *findAll* de su JPA.  Ahora lo hará a POST. Cuando esto suceda, invocaremos el método *save* de esta misma API de persistencia de Java (JPA).

Agregue:

```
    @PostMapping
    public Paciente guardar(@RequestBody Paciente p) {
        return repo.save(p);
    }
```

Su archivo PacienteController.java ahora debe lucir de la siguiente manera:

(siguiente página)

## PacienteController.java

```
package com.demo.pacientes.controller;

import com.demo.pacientes.model.Paciente;
import com.demo.pacientes.repository.PacienteRepository;
import org.springframework.web.bind.annotation.*;

import java.util.List;

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

    private final PacienteRepository repo;

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

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

    @PostMapping
    public Paciente guardar(@RequestBody Paciente p) {
        return repo.save(p);
    }

}
```

# 3. form.html

Diríjase a la carpeta static de su proyecto, y haga un archivo nuevo llamado form.html (El nombre de form es genérico, y debería recibir algo más representativo como pacientes.html. Considere esto para su proyecto final) El código de este archivo form.html se encuentra en la siguiente página.

También incluya un hipervínculo hacia esta página form.html desde su index.html. Diríjase a su página index.html y en cuanto termina su tabla incluya la siguiente etiqueta

<a href="form.html">Nuevo paciente</a>

Este hipervínculo será sustituido por un menú de navegación para su proyecto final.

```
<!DOCTYPE html>
<html>
<head>
    <title>Nuevo paciente</title>
</head>
<body>

<h1>Nuevo paciente</h1>

<form id="form">

    Nombre:
    <input type="text" id="nombre">
    <br>

    Telefono:
    <input type="text" id="telefono">
    <br>

    <button type="submit">
        Guardar
    </button>

</form>

<script>
document.getElementById("form")
.addEventListener("submit", function(e){

    e.preventDefault();

    const datos = {

        nombre: document.getElementById("nombre").value,
        telefono: document.getElementById("telefono").value

    };

    fetch("/api/pacientes", {
        method: "POST",
        headers: {
            "Content-Type": "application/json"
        },
        body: JSON.stringify(datos)

    })
    .then(() => {
        window.location = "index.html";

    });

});

</script>

</body>
</html>
```

# 11. Ejecutar

Desde terminal:

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

Abrir:

http://localhost:8080/index.html

Pruebe el vínculo de “Nuevo Paciente”

Asignación: Diseñe estos mismos métodos para la empresa de su proyecto final.