

# Resolución del ejercicio 1

Te proporcionamos algunas recomendaciones para la resolución

Prueba de métodos para “Persona” y “Producto” con Swagger

Para probar los métodos en Swagger, **solo deberá hacer clic en el botón TRY IT OUT y editar el JSON de ejemplo.**



Veamos la imagen del siguiente *slide*.



POST

/productos

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{
  "id": 1,
  "nombre": "Coca Cola",
  "precio": 2110,
  "descripcion": "Coca Cola",
  "urlFoto": "coca.gif"
}
```

Execute

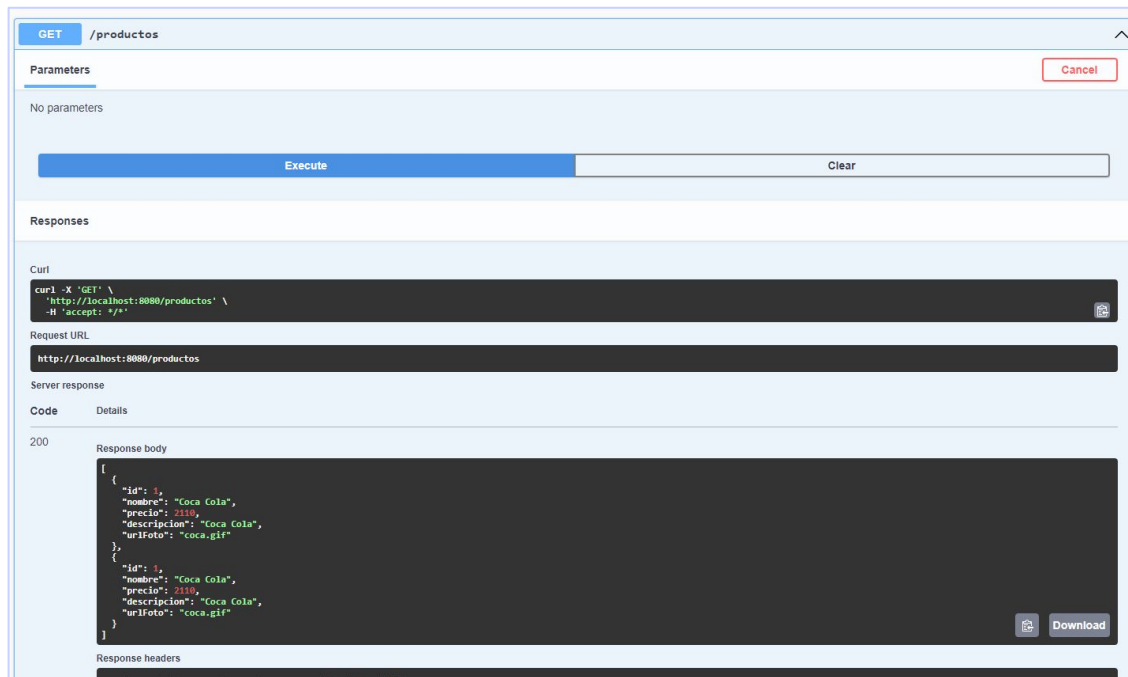
Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/productos' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 1,
    "nombre": "Coca Cola",
    "precio": 2110,
    "descripcion": "Coca Cola",
    "urlFoto": "coca.gif"
  }'
```

Para verificar que está funcionando correctamente se utilizará GET:



The screenshot shows a REST client interface with the following sections:

- GET /productos**: The method and endpoint.
- Parameters**: A section with a "Cancel" button and the text "No parameters".
- Execute**: A blue button to execute the request.
- Clear**: A button to clear the request.
- Responses**: A section containing:
  - Curl**: A code block with the command: 

```
curl -X 'GET' \
'http://localhost:8080/productos' \
-H 'accept: */*'
```
  - Request URL**: A code block with the URL: 

```
http://localhost:8080/productos
```
  - Server response**: A section with tabs for "Code" and "Details".
- 200**: The status code of the response.
- Response body**: A code block showing the JSON response:

```
{
  "id": 1,
  "nombre": "Coca Cola",
  "precio": 2110,
  "descripcion": "Coca Cola",
  "urlFoto": "coca.gif"
},
{
  "id": 1,
  "nombre": "Coca Cola",
  "precio": 2110,
  "descripcion": "Coca Cola",
  "urlFoto": "coca.gif"
}
```
- Response headers**: A section at the bottom.

## Resolución del ejercicio 2

Te proporcionamos algunas recomendaciones para la resolución

### Clase venta

```
import java.util.List;

public class Venta {
    private Long id;
    private Usuario usuario;
    private List<DetalleVenta>
detalleVentas;

    // Getters y Setters
}
```

```
public class DetalleVenta {
    private Producto producto;
    private int cantidad;

    // Getters y Setters
}
```

## Venta Controller



```
@RestController
@RequestMapping("/api/ventas")
public class VentaController {

    @Autowired
    private VentaService ventaService;

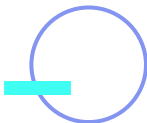
    @PostMapping
    public ResponseEntity<Venta> registrarVenta(@RequestBody Venta venta) {
        Venta nuevaVenta = ventaService.registrarVenta(venta);
        return ResponseEntity.ok(nuevaVenta);
    }

    @GetMapping
    public ResponseEntity<List<Venta>> obtenerTodasLasVentas() {
        return ResponseEntity.ok(ventaService.obtenerTodasLasVentas());
    }
}
```



## Venta Service

```
@Service
public class VentaService {
    private final List<Venta> ventas = new ArrayList<>();
    private final AtomicLong counter = new AtomicLong();
    public Venta registrarVenta (Venta venta) {
        venta.setId(counter.incrementAndGet());
        ventas.add(venta);
        return venta;
    }
    public List<Venta> obtener Todas Las Ventas() {
        return ventas;
    }
}
```



**¡Terminaste el módulo!**  
**Todo listo para rendir el examen**