Microservicios – Arquitectura y Desarrollo

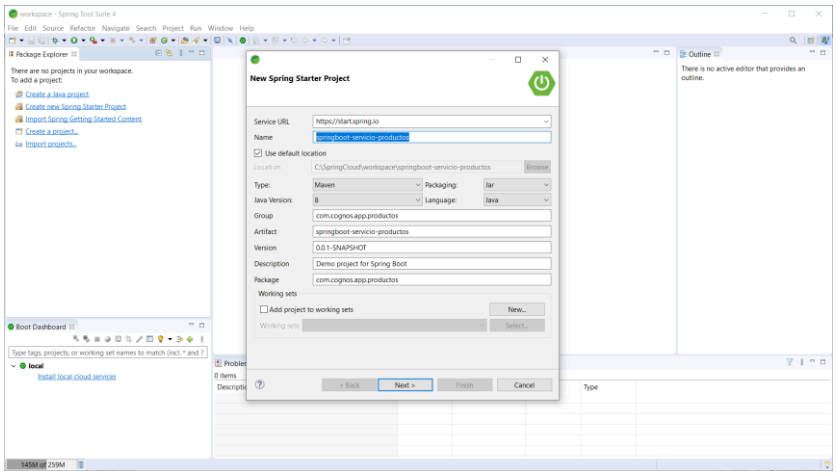
LAB Creando el Proyecto de Microservicios

Objetivos

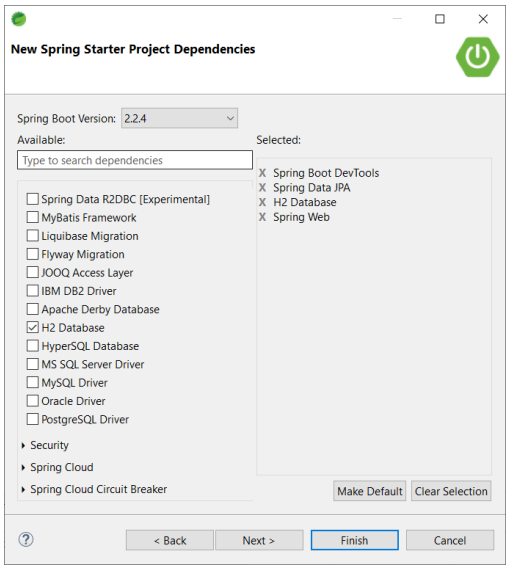
- Mostrar al participante el procedimiento para la creación de proyectos de Microservicios

Procedimiento

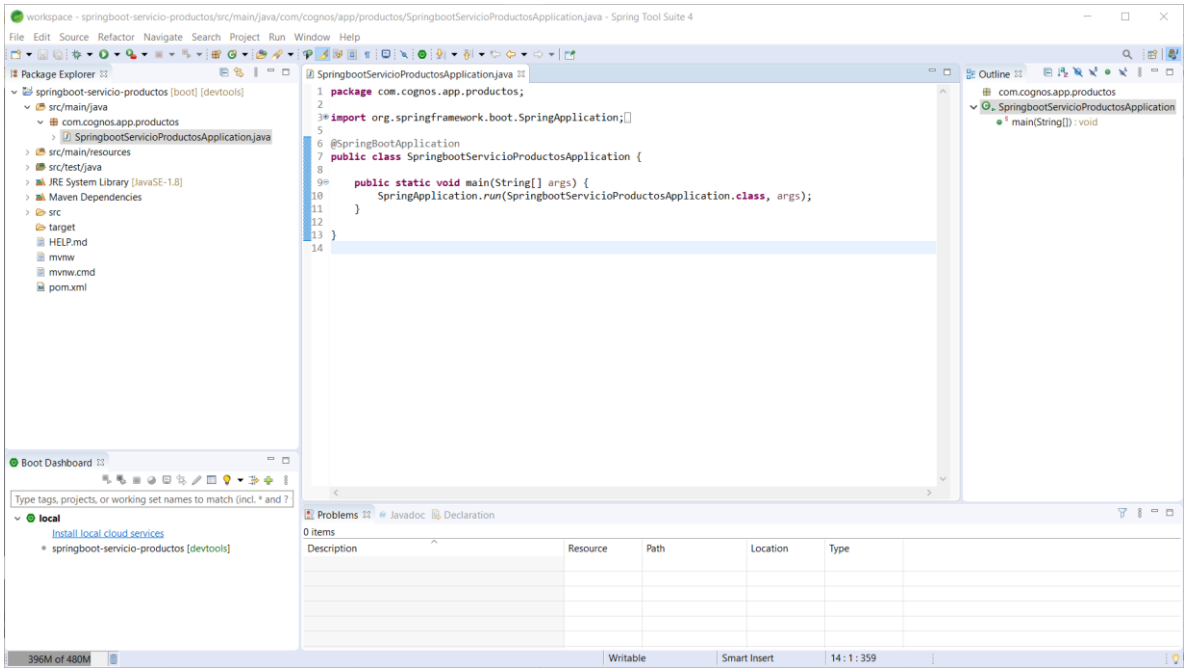
1. Crea el proyecto springboot-servicio-productos



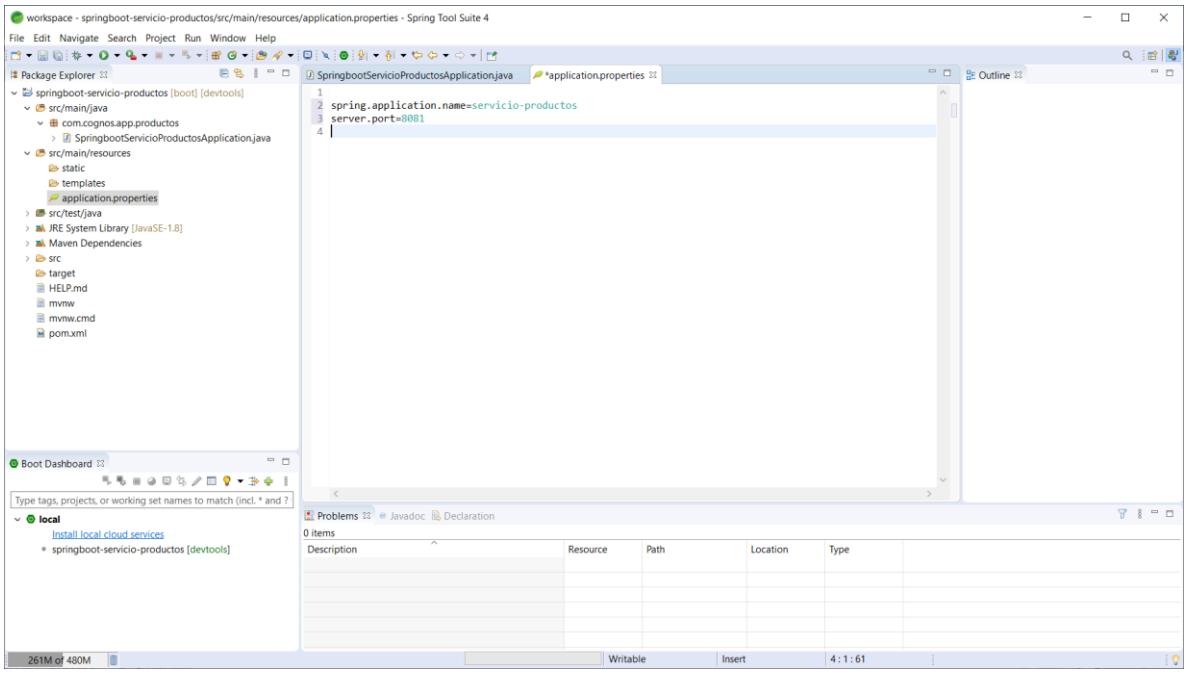
2. Selecciona las dependencias Spring Boot DevTools, Spring Web, H2 Database y Spring Data JPA



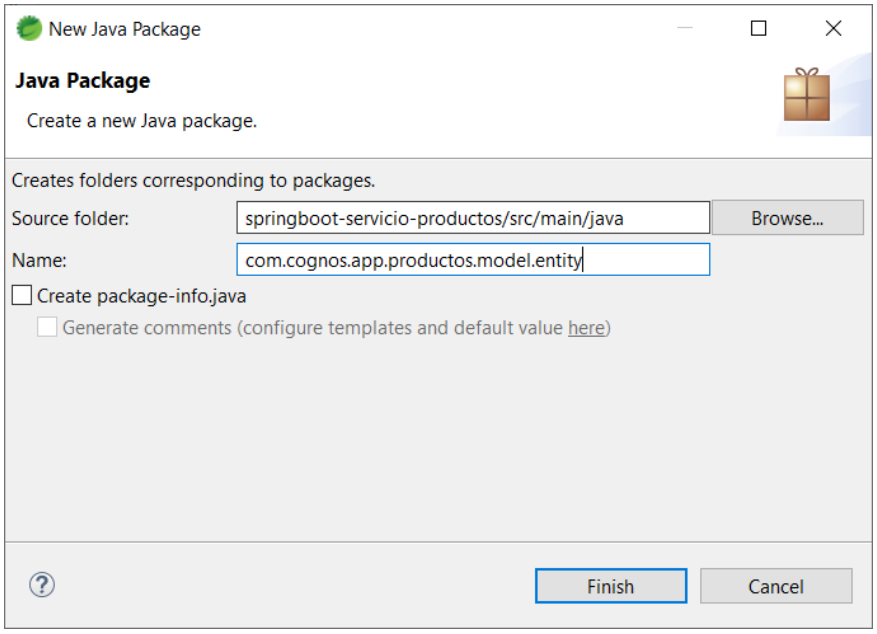
3. Verifica que se haya creado el Proyecto con éxito.



4. Edita el archivo application.properties



5. Crea el paquete com.cognos.app.productos.model.entity



6. Crea la entidad Producto en el paquete com.cognos.app.productos.model.entity

```
package com.cognos.app.productos.model.entity;

import java.io.Serializable;
import java.util.Date;
```

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name="productos")
public class Producto implements Serializable{

	private static final long serialVersionUID = 3526278020978785736L;

	@Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	private String nombre;
	private Double precio;
	@Column(name = "create_at")
	@Temporal(TemporalType.DATE)
	private Date createAt;
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public Double getPrecio() {
		return precio;
	}
	public void setPrecio(Double precio) {
		this.precio = precio;
	}
	public Date getCreateAt() {
		return createAt;
	}
	public void setCreateAt(Date createAt) {
		this.createAt = createAt;
	}


}
```

7. Crea el paquete `com.cognos.app.productos.model.dao y la interface ProductoDao`

```
package com.cognos.app.productos.model.dao;

import org.springframework.data.repository.CrudRepository;

import com.cognos.app.productos.model.entity.Producto;

public interface ProductoDao extends CrudRepository<Producto, Long> {

}
```

8. Crea el paquete `com.cognos.app.productos.model.service` y la interface ProductoService

```
package com.cognos.app.productos.model.service;

import java.util.List;

import com.cognos.app.productos.model.entity.Producto;

public interface ProductoService {

	public List<Producto> findAll();
	public Producto findById(Long id);
}
```

9. Implementa la interface ProductoService con la clase ProductoServiceImpl

```
package com.cognos.app.productos.model.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.cognos.app.productos.model.dao.ProductoDao;
import com.cognos.app.productos.model.entity.Producto;

@Service
public class ProductoServiceImpl implements ProductoService {

	@Autowired
```

```
	private ProductoDao productoDao;

	@Override
	@Transactional(readOnly = true)
	public List<Producto> findAll() {
		return (List<Producto>) productoDao.findAll();
	}

	@Override
	@Transactional(readOnly = true)
	public Producto findById(Long id) {
		return  productoDao.findById(id).orElse(null);
	}

}
```

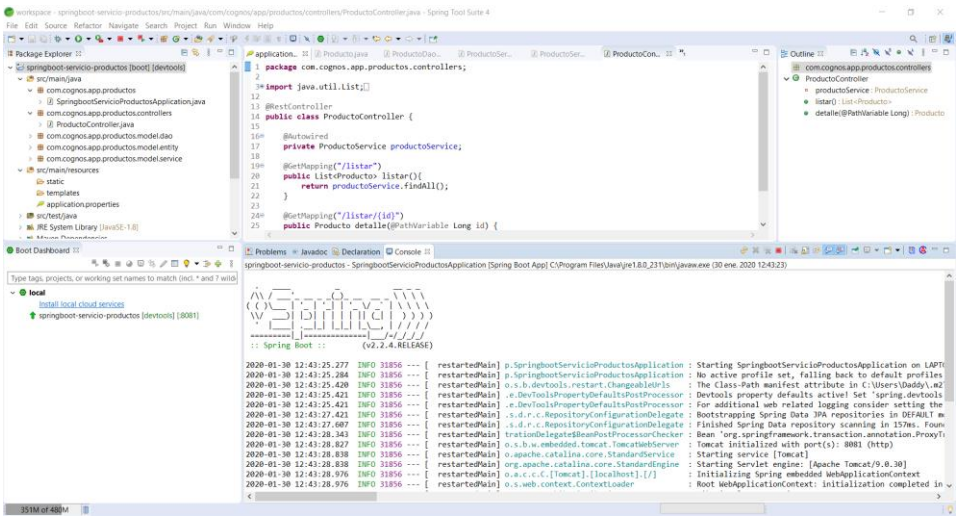
10. Crea el paquete `com.cognos.app.productos.controllers` y crea el controlador ProductoController

```
package com.cognos.app.productos.controllers;


import java.util.List;


import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;


import com.cognos.app.productos.model.entity.Producto;

import com.cognos.app.productos.model.service.ProductoService;


@RestController

public class ProductoController {


	@Autowired

	private ProductoService productoService;
```

```
	@GetMapping("/listar")
	public List<Producto> listar(){
		return productoService.findAll();
	}

	@GetMapping("/mostrar/{id}")
	public Producto detalle(@PathVariable Long id) {
		return productoService.findById(id);
	}
}
```

11. Prueba la aplicación.

Run > Run As > Spring Boot App



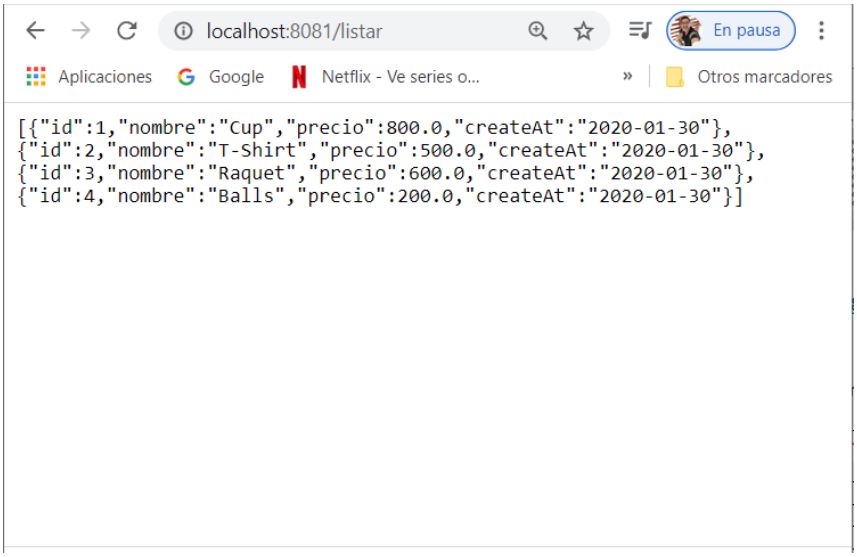
Abre el browser y navega: http://localhost:8081/listar

Nota: De momento como no hay productos en el BD se mostrará un arreglo vacío.

12. Agrega en **resources** el archivo **import.sql** con el siguiente contenido.

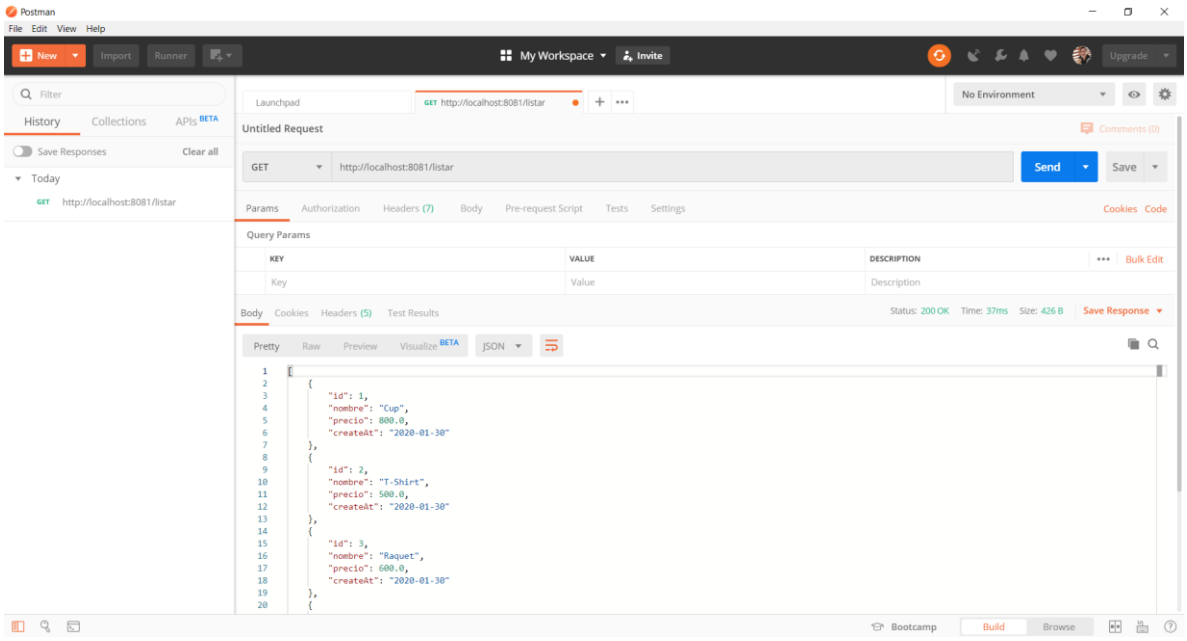
```
insert into productos (nombre, precio, create_at) values ('Cup', 800, now());
insert into productos (nombre, precio, create_at) values ('TShirt', 500, now());
insert into productos (nombre, precio, create_at) values ('Raquet', 600, now());
insert into productos (nombre, precio, create_at) values ('Balls', 200, now());
```

13. Ejecuta nuevamente la aplicacion si es que no lo estaba y abre la url http://localhost:8081/listar



14. Instala postman, descárgalo desde https://www.getpostman.com

15. Realiza un request a http://localhost:8081/listar usando postman


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs BETA
Save Responses
Clear all
Today
GET http://localhost:8081/listar
Launchpad
GET http://localhost:8081/listar
No Environment
Untitled Request
Comments (0)
GET
http://localhost:8081/listar
Send
Save
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 37ms
Size: 426 B
Save Response
Pretty
Raw
Preview
Visualize BETA
JSON
[
{
"id": 1,
"nombre": "Cup",
"precio": 800.0,
"createAt": "2020-01-30"
},
{
"id": 2,
"nombre": "T-Shirt",
"precio": 500.0,
"createAt": "2020-01-30"
},
{
"id": 3,
"nombre": "Raquet",
"precio": 600.0,
"createAt": "2020-01-30"
},
{
Bootcamp
Build
Browse