# ESCUELA COLOMBIANA DE INGENIERÍA
# ARQUITECTURAS DE SOFTWARE (ARSW)
## Integración Spring–MVC–REST

## 1. SPRING MVC REST

En este laboratorio se crea un api REST que permite realizar XXX mediante peticiones HTTP utilizando JSON para el intercambio de información. Para esto, descargue el código disponible en `https://github.com/ARSW-ECI/spring-rest.git`.

En este repositorio se encuentran dos proyectos diferentes:

- Un manejador de ordenes de comida basado en Spring
- Un proyecto Maven con un arqueotipo JavaEE-Web, configurado para trabajar con Spring Framework Boot.

### 1.1. Parte I

En este ejercicio, usted creará un API REST que permita manipular un manejador de órdenes a través de peticiones HTTP, y usando el formato JSON para el intercambio de información. Para esto:

1. Revise la funcionalidad del proyecto de ordenes de comida (principalmente la clase ManejadorOrdenes).
2. En el directorio `spring-sample-project-food-orders` ejecute:

   ```
   mvn compile
   mvn exec:java
   ```

   para compilar y ejecutar la aplicación.
3. Revise las clases del proyecto `spring-rest-echo` principalmente la clase `EchoController`.
4. Cree el jar del projecto `spring-rest-echo`

   ```
   mvn package
   java -jar target/rest-ws-0.1.0.jar
   ```

5. En un navegador haga pruebas con las direcciónes:

   *a*) `http://localhost:8080/echo/hello%20world`

   *b*) `http://localhost:8080/hecho/hello%20world`

   Modificando la última parte de la dirección url.

6. Copie las clases y el archivo de configuración spring del proyecto de ordenes al proyecto spring-rest en sus respectivas rutas.

7. habilite la carga automática del contexto del archivo de configuración de Spring copiado. Para esto revise la clase `Application`, descomente y revise los nombres de los archivos de configuración apropiados.

8. Cree la clase que servirá como manejador de las nuevas peticiones REST para los recursos correspondientes a las ordenes. Dicha clase debe tener las anotaciones:

```
@RestController
@RequestMapping("/ordenes")
```

9. A esta clase agregue como atributo el Manejador de Ordenes, y use la anotación `@Autowired` para que la misma sea inyectada.

10. En dicha clase agregue un método que permita manejar las peticiones GET a todas las ordenes, y a una orden en particular

```
@RequestMapping(method = RequestMethod.GET)
@ResponseBody
public List<Orden> consords() {
  ...
}
@RequestMapping(value="/{numorden}",method = RequestMethod.GET)
@ResponseBody
public Orden consord(@PathVariable int numorden) {
  ...
}
```

11. Para implementar los métodos anteriores:

    *a*) Use el manejador de órdenes para obtener la lista actual de ordenes.

    *b*) Para la operación que retorna una orden en particular, si el número de orden no existe, se arrojará una excepción de tipo OperationFailedException (la cual equivaldrá a un código 404 http, de lado del cliente).

12. Desde un browser puede probar el envío de una petición GET a uno de los pedidos, por ejemplo: `http://<host>:puerto/<aplicación>/rest/orden/1`

## 1.2. Parte II

1. Agregue el manejo de peticiones POST (creación de nuevas ordenes), de manera que un cliente http pueda registrar una nueva orden haciendo una petición POST al recurso 'ordenes', y enviando como contenido de la petición todo el detalle de dicho recurso a través de un documento JSON.

```
@RequestMapping(method = RequestMethod.POST)
public ResponseEntity<?> persist(@RequestBody Orden o) {
  //agregar la orden
  return new ResponseEntity<>(HttpStatus.CREATED);
}
```

2. Para probar que el recurso 'ordenes' acepta e interpreta correctamente el verbo POST, use el comando curl de Unix. Este comando tiene como parámetro el tipo de contenido manejado (en este caso JSON), y el 'cuerpo del mensaje' que irá con la petición, lo cual en este caso debe ser un documento JSON equivalente a la clase Cliente:

```
curl -i -X POST -HContent-Type:application/json
     -HAccept:application/json <URL del recurso cliente>
     -d '<DocumentoJSON>'
```

3.