

# **Informe N°3**

## **Proyecto semestral:**

### **“EduTech Innovators SPA”**

#### **Integrantes:**

- **Maximiliano Hernández**
- **Renato Valenzuela**
- **Diego Zamora**

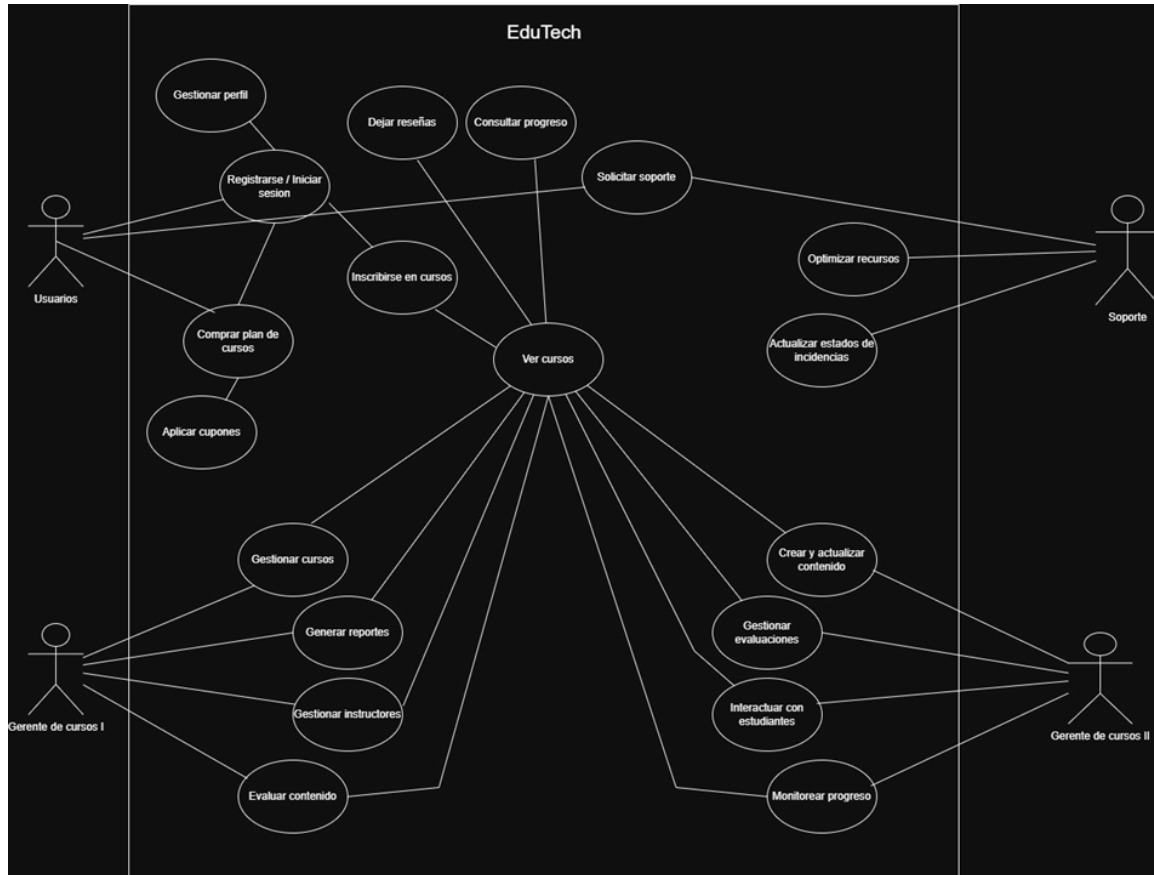
# Índice

- i. Diagrama de arquitectura de microservicios
- ii. Plan de pruebas: documentar con imágenes la codificación de las pruebas.  
Adicionalmente deben justificar qué herramientas y frameworks han utilizado para la creación de las pruebas.
  - ❖ Pruebas Unitarias
  - ❖ Pruebas de integración
- iii. Ejecución de pruebas:
  - ❖ Documentación de las pruebas ejecutadas.
  - ❖ Documentar con imágenes la ejecución de las pruebas implementadas.
- iv. Git-GitHub: explicar comandos utilizados (incluir print de pantalla) para la subida de archivos en el repositorio creado en GitHub, proyecto y base de datos. Adicionalmente en archivo Readme de Github añadir path de cada servicio REST con sus peticiones para su ejecución.
- v. Conclusión

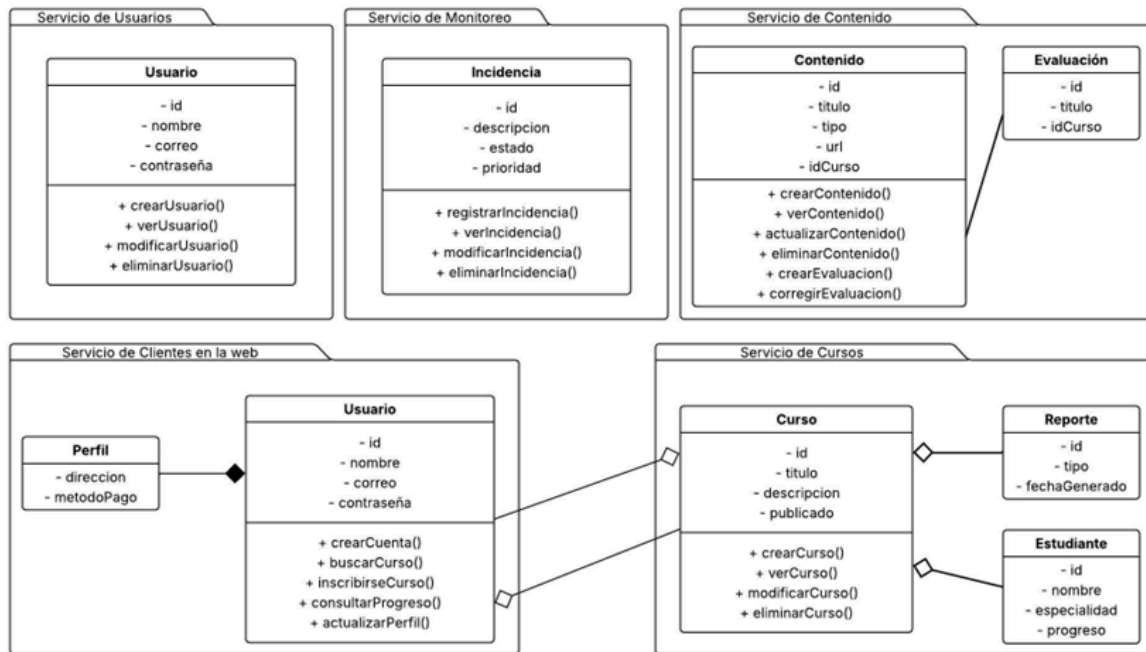
# Diagrama de arquitectura de microservicios

Para explicar esta nueva arquitectura, a continuación se presentan tres diagramas que ilustran distintos aspectos clave del sistema.

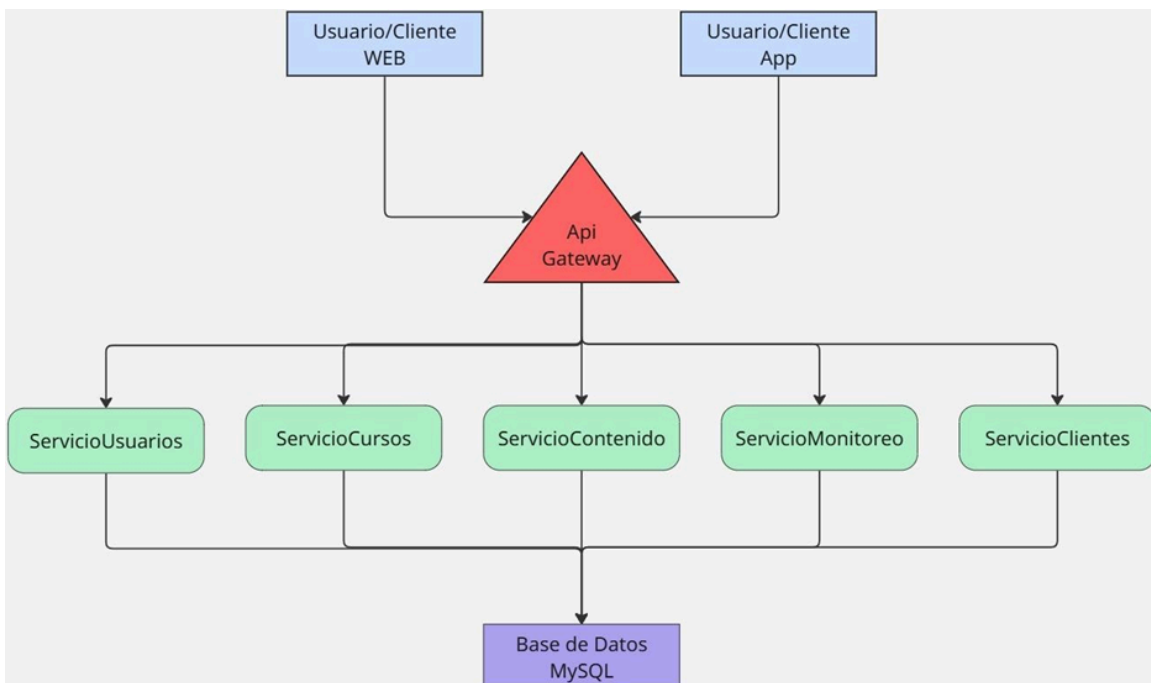
El diagrama de casos de uso muestra las principales interacciones entre los actores y las funcionalidades del sistema:



El *diagrama de clases* detalla la estructura lógica de los objetos y sus relaciones dentro de los microservicios.



Finalmente, el *diagrama de despliegue* representa la arquitectura física del sistema, incluyendo la distribución de los microservicios, sus bases de datos y las formas de comunicación entre ellos.



# Plan de pruebas

## A continuación las pruebas unitarias:

Para hacer las pruebas unitarias se usó el framework Mockito el cual nos sirve para simular objetos (como Usuarios, Incidencias, Cursos, etc) llamados “mocks” y ver si el CRUD funciona correctamente sin la necesidad de hacer conexiones reales (a la base de datos en este caso) y para ejecutar estos testeos se ocupa como motor de pruebas el framework de JUnit.

Pruebas unitarias del servicio Usuario:

```
package com.proyecto.springboot.backend.springboot_backend.services;

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

import java.util.ArrayList;
import java.util.List;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import com.proyecto.springboot.backend.springboot_backend.entities.Usuario;
import com.proyecto.springboot.backend.springboot_backend.repository.UsuarioRepository;

public class UsuarioServiceImplTest {

    @InjectMocks
    private UsuarioServiceImpl service;

    @Mock
    private UsuarioRepository repository;

    List<Usuario> list = new ArrayList<Usuario>();

    @BeforeEach
    public void init(){
        MockitoAnnotations.openMocks(this);

        this.chargeUsuario();
    }

    @Test
    public void findByAllTest(){
        when(repository.findAll()).thenReturn(list);

        List<Usuario> response = service.findByAll();

        assertEquals(expected:3, response.size());

        verify(repository, times(wantedNumberOfInvocations:1)).findAll();
    }

    public void chargeUsuario(){
        Usuario usuario1 = new Usuario(Long.valueOf(1:203003009), nombre:"Usuario uno", correo:"usuariouno@duocuc.cl", contrasenia:"contra\\suariouno");
        Usuario usuario2 = new Usuario(Long.valueOf(1:204004008), nombre:"Usuario dos", correo:"usuariodos@duocuc.cl", contrasenia:"contra\\suariodos");
        Usuario usuario3 = new Usuario(Long.valueOf(1:205005007), nombre:"Usuario tres", correo:"usuariotres@duocuc.cl", contrasenia:"contra\\suariotres");

        list.add(usuario1);
        list.add(usuario2);
        list.add(usuario3);
    }
}
```

```
✓ [U] springboot-backend 180ms
  ✓ {} com.proyecto.springboot.backend.springboot_backend
    > [U] SpringbootBackendApplicationTests
  ✓ {} com.proyecto.springboot.backend.springboot_backend.services 180ms
    > [U] UsuarioServiceImplTest 180ms
```

## Pruebas unitarias del servicio de Mantenimiento:

```

pom.xml | M | J IncidenciaServiceImplTest.java | X | SpringbootBackendApplicationTest.java | Incidencia.java
Projecto-Semestral > springboot-backend > src > test > java > com > proyecto > springboot > backend > springboot_backend > services > J IncidenciaServiceImplTest.java > IncidenciaServiceImplTest

1 package com.proyecto.springboot.backend.springboot_backend.services;
2
3 import static org.junit.jupiter.api.Assertions.*;
4 import static org.mockito.Mockito.times;
5 import static org.mockito.Mockito.verify;
6 import static org.mockito.Mockito.when;
7
8 import java.util.ArrayList;
9 import java.util.List;
10
11 import org.junit.jupiter.api.BeforeEach;
12 import org.junit.jupiter.api.Test;
13 import org.mockito.InjectMocks;
14 import org.mockito.Mock;
15 import org.mockito.MockitoAnnotations;
16
17 import com.proyecto.springboot.backend.springboot_backend.entities.Incidencia;
18 import com.proyecto.springboot.backend.springboot_backend.repository.IncidenciaRepository;
19
20
21 public class IncidenciaServiceImplTest {
22
23     @InjectMocks
24     private IncidenciaServiceImpl service;
25
26     @Mock
27     private IncidenciaRepository repository;
28
29     List<Incidencia> list = new ArrayList<Incidencia>();
30
31     @BeforeEach
32     public void init(){
33         MockitoAnnotations.openMocks(this);
34
35         this.chargeIncidencia();
36     }
37
38     @Test
39     public void findByAllTest(){
40         when(repository.findAll()).thenReturn(list);
41
42         List<Incidencia> response = service.findByAll();
43
44         assertEquals(expected:4, response.size());
45         verify(repository, times(wantedNumberOfInvocations:1)).findAll();
46     }
47
48     public void chargeIncidencia(){
49         Incidencia inci1 = new Incidencia(Long.valueOf(1:1), descripcion:"Primera prueba de Incidencias", estado:"Inactivo", prioridad:"Alta" );
50         Incidencia inci2 = new Incidencia(Long.valueOf(1:2), descripcion:"Segunda prueba de Incidencias", estado:"Inactivo", prioridad:"Baja" );
51         Incidencia inci3 = new Incidencia(Long.valueOf(1:3), descripcion:"Probando errores", estado:"Activo", prioridad:"Media" );
52         Incidencia inci4 = new Incidencia(Long.valueOf(1:4), descripcion:"Probando soluciones", estado:"Inactivo", prioridad:"Alta" );
53
54         list.add(inci1);
55         list.add(inci2);
56         list.add(inci3);
57         list.add(inci4);
58     }
59 }
60

```

```

TESTING
Filter (e.g. text, !exclude, @tag)
2/2 5.8s
springboot-backend 466ms
  com.proyecto.springboot.backend.springboot_backend 325ms
    SpringbootBackendApplicationTests 325ms
      contextLoads() 325ms
  com.proyecto.springboot.backend.springboot_backend.services 141ms
    IncidenciaServiceImplTest 141ms
      findByAllTest() 141ms

```

## Pruebas unitarias del servicio de Clientes (en la web):

```

Proyecto-Semestral > springboot-backend > src > test > java > com > proyecto > springboot > backend > springboot_backend > services > ClienteServiceImplTest.java > findByAllTest()

1 package com.proyecto.springboot.backend.springboot_backend.services;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.mockito.Mockito.times;
5 import static org.mockito.Mockito.verify;
6 import static org.mockito.Mockito.when;
7
8 import java.util.ArrayList;
9 import java.util.List;
10
11 import org.junit.jupiter.api.BeforeEach;
12 import org.junit.jupiter.api.Test;
13 import org.mockito.InjectMocks;
14 import org.mockito.Mock;
15 import org.mockito.MockitoAnnotations;
16
17 import com.proyecto.springboot.backend.springboot_backend.entities.Cliente;
18 import com.proyecto.springboot.backend.springboot_backend.repository.ClienteRepository;
19
20 public class ClienteServiceImplTest {
21
22     @InjectMocks
23     private ClienteServiceImpl service;
24
25     @Mock
26     private ClienteRepository repository;
27
28     List<Cliente> list = new ArrayList<Cliente>();
29
30     @BeforeEach
31     public void init(){
32         MockitoAnnotations.openMocks(this);
33
34         this.chargeCliente();
35     }
36
37     @Test
38     public void findByAllTest(){
39         when(repository.findAll()).thenReturn(list);
40
41         List<Cliente> response = service.findByAll();
42         assertEquals(expected:4, response.size());
43         verify(repository, times(wantedNumberOfInvocations:1)).findAll();
44     }
45
46     public void chargeCliente(){
47         Cliente cli1 = new Cliente(Long.valueOf(1i1), nombre:"Diego", correo:"di.zamora@duocuc.cl", direccion:"Chiringuito Chatarra", metodoPago:"Visa" );
48         Cliente cli2 = new Cliente(Long.valueOf(1i2), nombre:"Maximiliano", correo:"mac@duocuc.cl", direccion:"Pisos picados", metodoPago:"Visa" );
49         Cliente cli3 = new Cliente(Long.valueOf(1i3), nombre:"Renato", correo:"ren@duocuc.cl", direccion:"Parque placentero", metodoPago:"Masterdcard" );
50         Cliente cli4 = new Cliente(Long.valueOf(1i4), nombre:"Anakin", correo:"darthvader@duocuc.cl", direccion:"Señorio de la sal", metodoPago:"Centurion" );
51
52         list.add(cli1);
53         list.add(cli2);
54         list.add(cli3);
55         list.add(cli4);
56     }
57 }
58

```

TESTING

Filter (e.g. text, !exclude, @tag)

3/3 14.8s

- springboot-backend 726ms
- com.proyecto.springboot.backend.springboot\_backend 379ms
  - SpringbootBackendApplicationTests 379ms
    - contextLoads() 379ms
- com.proyecto.springboot.backend.springboot\_backend.services 347ms
  - ClienteServiceImplTest 44ms
    - findByAllTest() 44ms
  - IncidenciaServiceImplTest 303ms
    - findByAllTest() 303ms

## Pruebas unitarias del servicio de Cursos:

```
package com.proyecto.springboot.backend.springboot_backend.services;

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.times;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.when;

import java.util.ArrayList;
import java.util.List;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import com.proyecto.springboot.backend.springboot_backend.entitiesCurso;
import com.proyecto.springboot.backend.springboot_backend.repository.CursoRepository;

public class CursoServiceImplTest {

    @InjectMocks
    private CursoServiceImpl service;

    @Mock
    private CursoRepository repository;

    List<Curso> list = new ArrayList<Curso>();

    @BeforeEach
    public void init(){
        MockitoAnnotations.openMocks(this);
        this.chargeCurso();
    }

    @Test
    public void findByAllTest(){
        when(repository.findAll()).thenReturn(list);

        List<Curso> response = service.findByAll();

        assertEquals( expected:3, response.size());

        verify(repository, times(wantedNumberOfInvocations:1)).findAll();
    }

    public void chargeCurso(){
        Curso prod1 = new Curso(Long.valueOf(1:1),titulo:"Matematicas",descripcion:"Nivelacion",publicado:true);
        Curso prod2 = new Curso(Long.valueOf(1:2),titulo:"Lenguaje",descripcion:"Nivelacion",publicado:true);
        Curso prod3 = new Curso(Long.valueOf(1:3),titulo:"Ingles",descripcion:"Nivelacion",publicado:true);

        list.add(prod1);
        list.add(prod2);
        list.add(prod3);
    }
}
```

```
✓ [✓] [I] springboot-backend 2.2s
> [✓] [✓] {} com.proyecto.springboot.backend.springboot_backend 18ms
✓ [✓] [✓] {} com.proyecto.springboot.backend.springboot_backend.services 2.1s
> [✓] [✓] [C] ContenidoServiceImplTest 1.1s
> [✓] [✓] [C] CursoServiceImplTest 1.0s
```



## Pruebas unitarias del servicio de Contenido:

```

1 package com.proyecto.springboot.backend.springboot_backend.services;
2
3 import static org.junit.jupiter.api.Assertions.*;
4 import static org.mockito.Mockito.times;
5 import static org.mockito.Mockito.verify;
6 import static org.mockito.Mockito.when;
7
8 import java.util.ArrayList;
9 import java.util.List;
10
11 import org.junit.jupiter.api.BeforeEach;
12 import org.junit.jupiter.api.Test;
13 import org.mockito.InjectMocks;
14 import org.mockito.Mock;
15 import org.mockito.MockitoAnnotations;
16
17 import com.proyecto.springboot.backend.springboot_backend.entities.Contenido;
18 import com.proyecto.springboot.backend.springboot_backend.repository.ContenidoRepository;
19
20
21 public class ContenidoServiceImplTest {
22
23     @InjectMocks
24     private ContenidoServiceImpl service;
25
26     @Mock
27     private ContenidoRepository repository;
28
29     List<Contenido> list = new ArrayList<Contenido>();
30
31     @BeforeEach
32     public void init(){
33         MockitoAnnotations.openMocks(this);
34         this.chargeContenido();
35     }
36
37
38
39     @Test
40     public void findByAllTest(){
41
42         when(repository.findAll()).thenReturn(list);
43
44         List<Contenido> response = service.findByAll();
45
46         assertEquals(expected:3, response.size());
47
48         verify(repository, times(wantedNumberOfInvocations:1)).findAll();
49     }
50
51     public void chargeContenido(){
52         Contenido cont1 = new Contenido(Long.valueOf(1:1), nombre:"Ecuaciones", tipo:"Materia");
53         Contenido cont2 = new Contenido(Long.valueOf(1:2), nombre:"Sustantivos Propios", tipo:"Materia");
54         Contenido cont3 = new Contenido(Long.valueOf(1:3), nombre:"Verbs", tipo:"Materia");
55
56         list.add(cont1);
57         list.add(cont2);
58         list.add(cont3);
59     }
60
61 }

```

```

✓ [icon] springboot-backend 2.2s
> ✓ {} com.proyecto.springboot.backend.springboot_backend 18ms
✓ [icon] {} com.proyecto.springboot.backend.springboot_backend.services 2.1s
> ✓ [icon] ContenidoServiceImplTest 1.1s
> ✓ [icon] CursoServiceImplTest 1.0s

```

## Ahora, las pruebas de integración:

Para hacer las pruebas de integración se usó una clase utilitaria que viene con Spring-Test llamada MockMvc el cual nos sirve para probar nuestra API Rest, osea, hace simulaciones de peticiones HTTP (Get, Post, Put, etc) sin la necesidad de levantar un servidor real.

Estas pruebas permiten verificar que los controladores, servicios y otros componentes funcionen correctamente de forma integrada. Dependiendo de la configuración, las peticiones pueden interactuar con una base de datos real o en memoria. Para ejecutar estos testeos también se ocupa como motor de pruebas el framework de JUnit.

### Pruebas de integración del servicio de Usuarios:

```
package com.proyecto.springboot.backend.springboot_backend_restcontrollers;

import static org.junit.jupiter.api.Assertions.fail;
import static org.mockito.Mockito.when;
import java.util.List;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.context.bean.override.mockito.MockitoBean;
import org.springframework.test.web.servlet.MockMvc;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.proyecto.springboot.backend.springboot_backend.entities Usuario;
import com.proyecto.springboot.backend.springboot_backend.services UsuarioServiceImpl;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static org.mockito.ArgumentMatchers.any;

@SpringBootTest
@AutoConfigureMockMvc
public class UsuarioRestControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @MockitoBean
    private UsuarioServiceImpl usuarioServiceImpl;
    private List<Usuario> usuariosLista;

    @Test
    public void verUsuariosTest() throws Exception {
        when(usuarioServiceImpl.findAll()).thenReturn(usuariosLista);
        mockMvc.perform(get(uriTemplate:"/api/usuarios")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk());
    }

    @Test
    public void verUnUsuarioTest(){
        Usuario unUsuario = new Usuario(Long.valueOf(1:203003000), nombre:"Usuario1", correo:"usuariouno@duocuc.cl", contrasenia:"contrausuariouno");
        try{
            when(usuarioServiceImpl.findById(Long.valueOf(1:203003000))).thenReturn(Optional.of(unUsuario));
            mockMvc.perform(get(uriTemplate:"/api/usuarios/203003000")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk());
        }
        catch(Exception ex){
            fail("El Testing lanza un Error"+ ex.getMessage());
        }
    }

    @Test
    public void usuarioNoExisteTest() throws Exception {
        when(usuarioServiceImpl.findById(Long.valueOf(1:203003000))).thenReturn(Optional.empty());
        mockMvc.perform(get(uriTemplate:"/api/usuarios/203003000")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isNotFound());
    }

    @Test
    public void crearUsuarioTest() throws Exception {
        Usuario unUsuario = new Usuario(Long.valueOf(1:203003000), nombre:"Usuario uno", correo:"usuariouno@duocuc.cl", contrasenia:"contrausuariouno");
        Usuario otroUsuario = new Usuario(Long.valueOf(1:204004000), nombre:"Usuario dos", correo:"usuariodos@duocuc.cl", contrasenia:"contrausuariodos");
        when(usuarioServiceImpl.save(any(type:Usuario class))).thenReturn(otroUsuario);
        mockMvc.perform(post(uriTemplate:"/api/usuarios")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(unUsuario)))
            .andExpect(status().isCreated());
    }
}
```

```

6/6 7.3s
springboot-backend 170ms
  com.proyecto.springboot.backend.springboot_backend
    SpringbootBackendApplicationTests
      com.proyecto.springboot.backend.springboot_backend.restcontrollers 170...
        UsuarioRestControllerTest 170ms
          verUsuariosTest() 75ms
          verUnUsuarioTest() 8.0ms
          usuarioNoExisteTest() 7.0ms
          crearUsuarioTest() 80ms
        com.proyecto.springboot.backend.springboot_backend.services
          UsuarioServiceImplTest

```

Pruebas de integración del servicio de Mantenimiento:

```

package com.proyecto.springboot.backend.springboot_backend.controllers;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.proyecto.springboot.backend.springboot_backend.entities.Incidencia;
import com.proyecto.springboot.backend.springboot_backend.services.IncidenciaService;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.media.Content;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.responses.ApiResponses;
import io.swagger.v3.oas.annotations.tags.Tag;

@Tag(name = "Incidencias", description = "Operaciones relacionadas con Incidencias")
@RestController
@RequestMapping("/api/incidencia")
public class IncidenciaController {

    @Autowired
    private IncidenciaService service;

    @Operation(summary = "Obtener lista de incidencias", description = "Devuelve todas las incidencias disponibles")
    @ApiResponse(responseCode = "200", description = "Lista de incidencias retornada correctamente",
        content = @Content(mediaType = "application/json", schema = @Schema(implementation = Incidencia.class)))
    @GetMapping
    public List<Incidencia> findAll() {
        return service.findAll();
    }

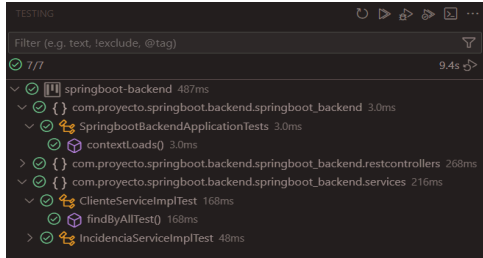
    @Operation(summary = "Obtener Incidencia por ID", description = "Obtiene el detalle de una incidencia especifica")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "Incidencia encontrada",
            content = @Content(mediaType = "application/json", schema = @Schema(implementation = Incidencia.class))),
        @ApiResponse(responseCode = "404", description = "Incidencia no encontrada")
    })
    @GetMapping("/{id}")
    public ResponseEntity<?> verIncidencia(@PathVariable Long id) {
        Optional<Incidencia> incidenciaOptional = service.findById(id);
        if (incidenciaOptional.isPresent()) {
            return ResponseEntity.ok(incidenciaOptional.orElseThrow());
        }
        return ResponseEntity.notFound().build();
    }

    @Operation(summary = "Crear una nueva incidencia", description = "Crea una incidencia con los datos proporcionados")
    @ApiResponse(responseCode = "201", description = "Incidencia creada correctamente",
        content = @Content(mediaType = "application/json", schema = @Schema(implementation = Incidencia.class)))
    @PostMapping
    public ResponseEntity<Incidencia> registrar(@RequestBody Incidencia unaIncidencia) {
        return ResponseEntity.status(HttpStatus.CREATED).body(service.save(unaIncidencia));
    }

    @Operation(summary = "Actualizar incidencia", description = "Actualiza la información de una incidencia existente")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "Incidencia actualizada correctamente",
            content = @Content(mediaType = "application/json", schema = @Schema(implementation = Incidencia.class))),
        @ApiResponse(responseCode = "404", description = "Incidencia no encontrada")
    })
    @PutMapping
    public ResponseEntity<?> modificar(@PathVariable Long id, @RequestBody Incidencia unaIncidencia) {
        Optional<Incidencia> incidenciaOptional = service.findById(id);
        if (incidenciaOptional.isPresent()) {
            Incidencia incidenciaExistente = incidenciaOptional.get();
            incidenciaExistente.setDescription(unaIncidencia.getDescription());
            incidenciaExistente.setEstado(unaIncidencia.getEstado());
            incidenciaExistente.setPrioridad(unaIncidencia.getPrioridad());
            Incidencia incidenciaModificada = service.save(incidenciaExistente);
            return ResponseEntity.ok(incidenciaModificada);
        }
        return ResponseEntity.notFound().build();
    }

    @Operation(summary = "Eliminar incidencia", description = "Elimina una incidencia por su ID")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "204", description = "Incidencia eliminada correctamente",
            content = @Content(mediaType = "application/json", schema = @Schema(implementation = Incidencia.class))),
        @ApiResponse(responseCode = "404", description = "Incidencia no encontrada")
    })
    @DeleteMapping("/{id}")
    public ResponseEntity<?> eliminar(@PathVariable Long id) {
        Incidencia unaIncidencia = new Incidencia();
        unaIncidencia.setId(id);
        Optional<Incidencia> incidenciaOptional = service.delete(unaIncidencia);
        if (incidenciaOptional.isPresent()) {
            return ResponseEntity.ok(incidenciaOptional.orElseThrow());
        }
        return ResponseEntity.notFound().build();
    }
}

```



Pruebas de integración para el servicio de Cliente (en la web):

```

package com.projecto.springboot.backend.springboot_backend.controllers;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.projecto.springboot.backend.springboot_backend.entities.Cliente;
import com.projecto.springboot.backend.springboot_backend.services.ClienteService;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.media.Content;
import io.swagger.v3.oas.annotations.media.Schema;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.responses.ApiResponses;
import io.swagger.v3.oas.annotations.tags.Tag;

@Tag(name = "Clientes", description = "Operaciones relacionadas con Clientes")
@RestController
@RequestMapping("/api/clientes")

public class ClienteController {

    @Autowired
    private ClienteService service;

    @Operation(summary = "Obtener lista de clientes", description = "Devuelve todos los clientes disponibles")
    @ApiResponse(responseCode = "200", description = "Lista de clientes retornada correctamente",
        content = @Content(mediaType = "application/json",
            schema = @Schema(implementation = Cliente.class)))
    @GetMapping
    public List<Cliente> list() {
        return service.findAll();
    }

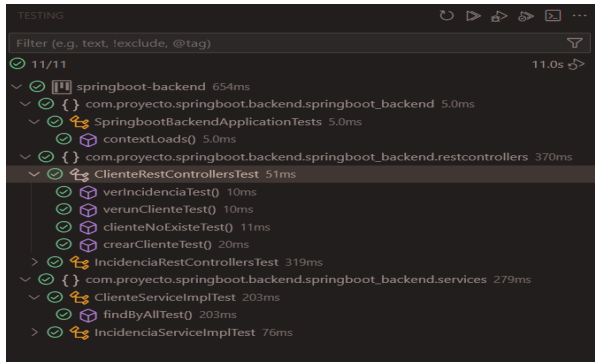
    @Operation(summary = "Obtener cliente por ID", description = "Obtiene el detalle de un cliente específico")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "Cliente encontrado",
            content = @Content(mediaType = "application/json", schema = @Schema(implementation = Cliente.class))),
        @ApiResponse(responseCode = "404", description = "Cliente no encontrado")
    })
    @GetMapping("/{id}")
    public ResponseEntity<?> verCliente(@PathVariable Long id) {
        Optional<Cliente> clienteOptional = service.findById(id);
        if (clienteOptional.isPresent()) {
            return ResponseEntity.ok(clienteOptional.orElseThrow());
        }
        return ResponseEntity.notFound().build();
    }

    @Operation(summary = "Crear un nuevo cliente", description = "Crea un cliente con los datos proporcionados")
    @ApiResponse(responseCode = "201", description = "Cliente creado correctamente",
        content = @Content(mediaType = "application/json", schema = @Schema(implementation = Cliente.class)))
    @PostMapping
    public ResponseEntity<Cliente> registrar(@RequestBody Cliente unCliente) {
        return ResponseEntity.status(HttpStatus.CREATED).body(service.save(unCliente));
    }

    @Operation(summary = "Actualizar cliente", description = "Actualiza la información de un cliente existente")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "Cliente actualizado correctamente",
            content = @Content(mediaType = "application/json", schema = @Schema(implementation = Cliente.class))),
        @ApiResponse(responseCode = "404", description = "Cliente no encontrado")
    })
    @PutMapping
    public ResponseEntity<?> modificar(@PathVariable Long id, @RequestBody Cliente unCliente) {
        Optional<Cliente> clienteOptional = service.findById(id);
        if (clienteOptional.isPresent()) {
            Cliente clienteExistente = clienteOptional.get();
            clienteExistente.setNombre(unCliente.getNombre());
            clienteExistente.setCorreo(unCliente.getCorreo());
            clienteExistente.setDireccion(unCliente.getDireccion());
            clienteExistente.setTelefono(unCliente.getTelefono());
            Cliente clienteModificado = service.save(clienteExistente);
            return ResponseEntity.ok(clienteModificado);
        }
        return ResponseEntity.notFound().build();
    }

    @Operation(summary = "Eliminar cliente", description = "Elimina un cliente por su ID")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "Cliente eliminado correctamente"),
        @ApiResponse(responseCode = "404", description = "Cliente no encontrado")
    })
    @DeleteMapping("/{id}")
    public ResponseEntity<?> eliminar(@PathVariable Long id) {
        Cliente unCliente = new Cliente();
        unCliente.setId(id);
        Optional<Cliente> clienteOptional = service.delete(unCliente);
        if (clienteOptional.isPresent()) {
            return ResponseEntity.ok(clienteOptional.orElseThrow());
        }
        return ResponseEntity.notFound().build();
    }
}

```

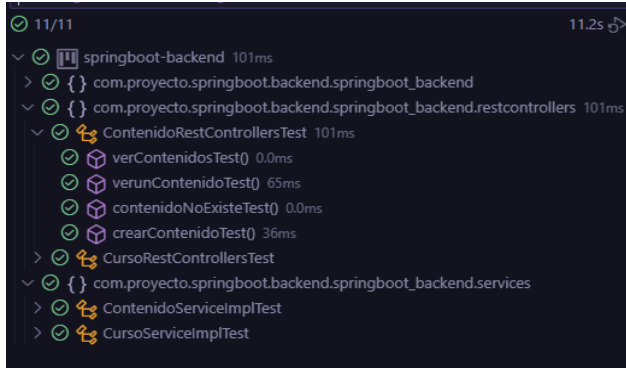


## Pruebas de integración del servicio de Curso:

```

1  import static org.mockito.Mockito.when;
2  import java.util.List;
3  import java.util.Optional;
4  import org.junit.jupiter.api.Test;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
7  import org.springframework.boot.test.context.SpringBootTest;
8  import org.springframework.http.MediaType;
9  import org.springframework.test.context.bean.override.mockito.MockitoBean;
10 import org.springframework.test.web.servlet.MockMvc;
11 import com.fasterxml.jackson.databind.ObjectMapper;
12 import com.proyecto.springboot.backend.springboot_backend.entities.Curso;
13 import com.proyecto.springboot.backend.springboot_backend.services.CursoServiceImpl;
14 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
15 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
16 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
17 import static org.mockito.ArgumentMatchers.any;
18
19 @SpringBootTest
20 @AutoConfigureMockMvc
21 public class CursoRestControllersTest {
22
23     @Autowired
24     private MockMvc mockMvc;
25
26     @Autowired
27     private ObjectMapper objectMapper;
28
29     @MockitoBean
30     private CursoServiceImpl cursoServiceImpl;
31     private List<Curso> cursosLista;
32
33     @Test
34     public void verProductosTest() throws Exception {
35         when(cursoServiceImpl.findByAll()).thenReturn(cursosLista);
36         mockMvc.perform(get(uriTemplate:"/api/cursos")
37             .contentType(MediaType.APPLICATION_JSON))
38             .andExpect(status().isOk());
39     }
40
41     @Test
42     public void verunCursoTest(){
43         Curso unCurso = new Curso(Long.valueOf(1L1), titulo:"Ingles", descripcion:"Nivelacion", publicado:true);
44
45         try{
46             when(cursoServiceImpl.findById(Long.valueOf(1L1))).thenReturn(Optional.of(unCurso));
47             mockMvc.perform(get(uriTemplate:"/api/cursos/1")
48                 .contentType(MediaType.APPLICATION_JSON))
49                 .andExpect(status().isOk());
50         }
51         catch(Exception ex){
52             fail("El Testing lanzo un Error"+ ex.getMessage());
53         }
54     }
55
56     @Test
57     public void cursoNoExisteTest() throws Exception{
58         when(cursoServiceImpl.findById(Long.valueOf(1L1))).thenReturn(Optional.empty());
59         mockMvc.perform(get(uriTemplate:"/api/productos/10")
60             .contentType(MediaType.APPLICATION_JSON))
61             .andExpect(status().isNotFound());
62     }
63
64     @Test
65     public void crearProductoTest() throws Exception{
66         Curso unCurso = new Curso(Long.valueOf(1L1), titulo:"Ingles", descripcion:"Nivelacion", publicado:true);
67         Curso otroCurso = new Curso(Long.valueOf(1L2), titulo:"Lenguaje", descripcion:"Nivelacion", publicado:true);
68         when(cursoServiceImpl.save(any(type:Curso.class))).thenReturn(otroCurso);
69         mockMvc.perform(post(uriTemplate:"/api/cursos")
70             .contentType(MediaType.APPLICATION_JSON)
71             .content(objectMapper.writeValueAsString(unCurso)))
72             .andExpect(status().isCreated());
73     }
74 }

```



Pruebas de integración del servicio de Contenido:

```
import static org.mockito.Mockito.when;
import java.util.List;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.context.bean.override.mockito.MockitoBean;
import org.springframework.test.web.servlet.MockMvc;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.proyecto.springboot.backend.springboot_backend.entities.Contenido;
import com.proyecto.springboot.backend.springboot_backend.services.ContenidoServiceImpl;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static org.mockito.ArgumentMatchers.any;
@SpringBootTest
@AutoConfigureMockMvc
public class ContenidoRestControllersTest {

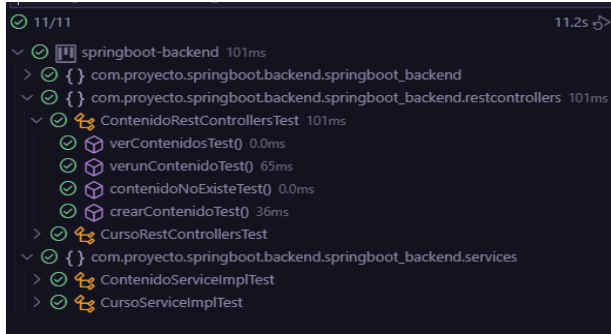
    @Autowired
    private MockMvc mockMvc;
    @Autowired
    private ObjectMapper objectMapper;
    @MockitoBean
    private ContenidoServiceImpl contenidoServiceImpl;
    private List<Contenido> contenidosLista;

    @Test
    public void verContentidosTest() throws Exception {
        when(contenidoServiceImpl.findAll()).thenReturn(contenidosLista);
        mockMvc.perform(get(uriTemplate:"/api/contenido")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk());
    }

    @Test
    public void verunContentidoTest(){
        Contenido unContenido = new Contenido(Long.valueOf(1L), nombre:"Ingles", tipo:"Materia");
        try{
            when(contenidoServiceImpl.findById(Long.valueOf(1L))).thenReturn(Optional.of(unContenido));
            mockMvc.perform(get(uriTemplate:"/api/contenido/1")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk());
        }
        catch(Exception ex){
            fail("El Testing lanzo un Error"+ ex.getMessage());
        }
    }

    @Test
    public void contenidoNoExisteTest() throws Exception {
        when(contenidoServiceImpl.findById(Long.valueOf(1L))).thenReturn(Optional.empty());
        mockMvc.perform(get(uriTemplate:"/api/contenido/1")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isNotFound());
    }

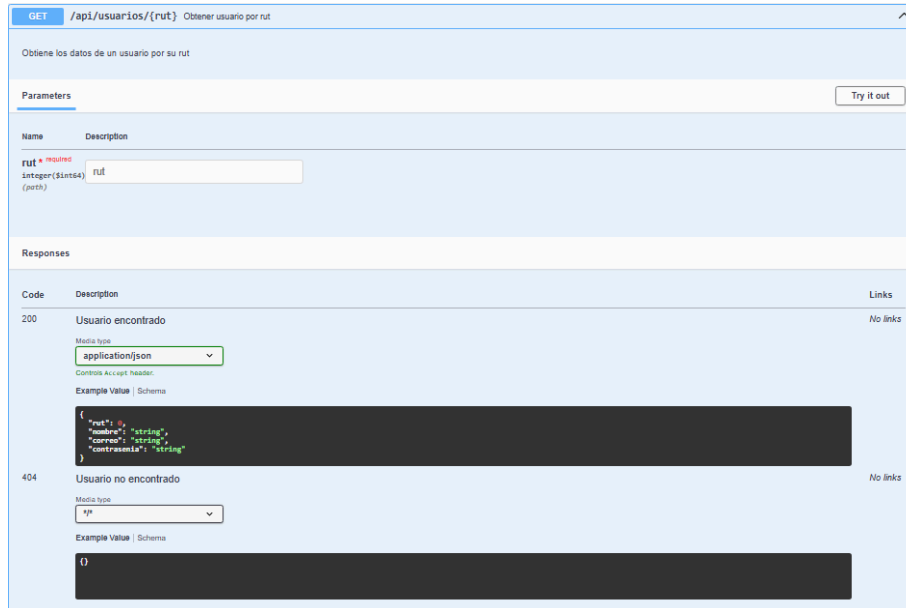
    @Test
    public void crearContentidoTest() throws Exception {
        Contenido unContenido = new Contenido(Long.valueOf(1L), nombre:"Matematicas", tipo:"Materia");
        Contenido otroContenido = new Contenido(Long.valueOf(1L), nombre:"Lenguaje", tipo:"Materia");
        when(contenidoServiceImpl.save(any(type:Contenido.class))).thenReturn(otroContenido);
        mockMvc.perform(post(uriTemplate:"/api/contenido")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(unContenido)))
            .andExpect(status().isCreated());
    }
}
```



# Ejecución de Pruebas

A continuación, la documentación de las pruebas vista en la herramienta llamada Swagger, la cual nos ayuda a probar los endpoints (Get, Post, Put, etc) desde el navegador sin necesidad de usar otras herramientas (como Postman). Permite ver: qué endpoints existen, qué parámetros necesita, qué devuelven, ejemplos de request y response.

Ejecución de pruebas de integración del servicio de Usuario:



**GET** /api/usuarios/{rut} Obtener usuario por rut

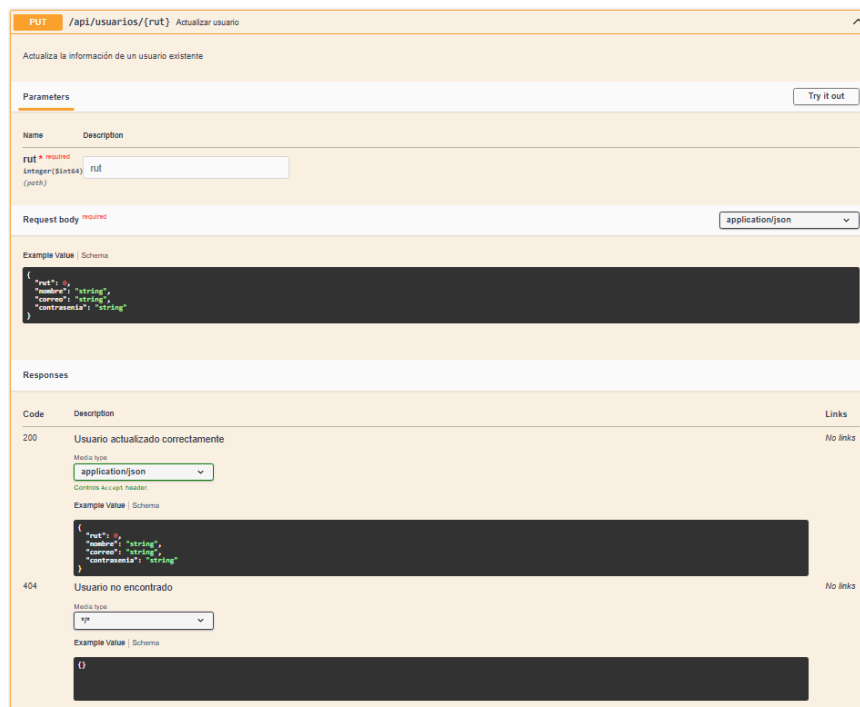
Obtiene los datos de un usuario por su rut

**Parameters**

Name	Description
rut <span>required</span>	rut

**Responses**

Code	Description	Links
200	Usuario encontrado	No links
404	Usuario no encontrado	No links



**PUT** /api/usuarios/{rut} Actualizar usuario

Actualiza la información de un usuario existente

**Parameters**

Name	Description
rut <span>required</span>	rut

**Request body** required

Media type: application/json

Example Value | Schema

```
{
  "rut": "12345678",
  "nombre": "string",
  "apellido": "string",
  "correo": "string",
  "contraseña": "string"
}
```

**Responses**

Code	Description	Links
200	Usuario actualizado correctamente	No links
404	Usuario no encontrado	No links



**DELETE** /api/usuarios/{rut} Eliminar usuario

Elimina un usuario por su ID

**Parameters** [Try it out](#)

Name	Description
<b>rut</b> <sup>required</sup> integer(\$int64) (path)	rut

**Responses**

Code	Description	Links
204	Usuario eliminado correctamente	No links
404	Usuario no encontrado	No links

**GET** /api/usuarios Obtener lista de usuarios

Devuelve a todos los usuarios disponibles

**Parameters** [Try it out](#)

No parameters

**Responses**

Code	Description	Links
200	Lista de usuarios retornada correctamente	No links

Media type: **application/json**

Example Value | Schema

```
{
  "rut": 0,
  "nombre": "string",
  "correo": "string",
  "contraseña": "string"
}
```

**POST** /api/usuarios Crear un nuevo usuario

Crear un usuario con los datos proporcionados

**Parameters** [Try it out](#)

No parameters

**Request body** <sup>required</sup> **application/json**

Example Value | Schema

```
{
  "rut": 0,
  "nombre": "string",
  "correo": "string",
  "contraseña": "string"
}
```

**Responses**

Code	Description	Links
201	Usuario creado correctamente	No links

Media type: **application/json**

Example Value | Schema

```
{
  "rut": 0,
  "nombre": "string",
  "correo": "string",
  "contraseña": "string"
}
```

## Ejecución de pruebas de integración del servicio de Mantenimiento:

**CONSEGUIR** /api/incidencia Obtener lista de incidencias

Devuelve todas las incidencias disponibles

**Parámetros** [Pruébalo](#)

Sin parámetros

**Respuestas**

Código	Descripción	Campo de golf
200	Lista de incidencias retornada correctamente	Sin enlaces

Tipo de medio:

Encabezado de controles: Accept

Valor de ejemplo | Esquema

```
{
  "id": 0,
  "descripcion": "string",
  "estado": "string",
  "prioridad": "string"
}
```

**PONER** /api/incidencia Actualizar incidencia

Actualiza la información de una incidencia existente.

**Parámetros** [Pruébalo](#)

Sin parámetros

Cuerpo de la solicitud **required**

Valor de ejemplo | Esquema

```
{
  "id": 0,
  "descripcion": "string",
  "estado": "string",
  "prioridad": "string"
}
```

**Respuestas**

Código	Descripción	Campo de golf
200	Incidencia actualizada correctamente	Sin enlaces
404	Incidencia no encontrada	Sin enlaces

Tipo de medio:

Encabezado de controles: Accept

Valor de ejemplo | Esquema

```
{
  "id": 0,
  "descripcion": "string",
  "estado": "string",
  "prioridad": "string"
}
```

Tipo de medio:

Valor de ejemplo | Esquema

```
()
```

**CORREO** /api /incidencia Crear una nueva incidencia

Crea una incidencia con los datos proporcionados

**Parámetros** [Pruébalo](#)

Sin parámetros

Cuerpo de la solicitud **required** aplicación/json

Valor de ejemplo | Estructura

```
{
  "id": 1,
  "descripcion": "string",
  "estado": "string",
  "prioridad": "string"
}
```

**Respuestas**

Código	Descripción	Campo de golf
201	Incidencia creada correctamente	Sin enlaces

Tip de medio: aplicación/json

Enviado de control de API:

Valor de ejemplo | Estructura

```
{
  "id": 1,
  "descripcion": "string",
  "estado": "string",
  "prioridad": "string"
}
```

**CONSEGUIR** /api /incidencia /{id} Obtener incidencia por identificación

Obtiene el detalle de una incidencia específica

**Parámetros** [Pruébalo](#)

Nombre	Descripción
identificación <b>required</b>	identificación

entero (3 dígitos)  
( cambio )

**Respuestas**

Código	Descripción	Campo de golf
200	Incidencia encontrada	Sin enlaces
404	Incidente no encontrado	Sin enlaces

Tip de medio: aplicación/json

Enviado de control de API:

Valor de ejemplo | Estructura

```
{
  "id": 1,
  "descripcion": "string",
  "estado": "string",
  "prioridad": "string"
}
```

Tip de medio: json

Enviado de control de API:

Valor de ejemplo | Estructura

```
{
}
```

**BORRAR** /api /incidencia /{id} Eliminar incidencia

Eliminar una incidencia por su ID

**Parámetros** [Pruébalo](#)

Nombre	Descripción
identificación <b>required</b>	identificación

entero (3 dígitos)  
( cambio )

**Respuestas**

Código	Descripción	Campo de golf
204	Incidencia eliminada correctamente	Sin enlaces
404	Incidente no encontrado	Sin enlaces

Tip de medio: json

Enviado de control de API:

Valor de ejemplo | Estructura

```
{
}
```

Tip de medio: json

Enviado de control de API:

Valor de ejemplo | Estructura

```
{
}
```

## Ejecución de pruebas de integración del servicio de Clientes (en la web):

**CONSEGUIR** /api /cliente /{id} Obtener cliente por ID

Obtiene el detalle de un cliente específico

**Parámetros** Pruébalo

Nombre	Descripción
<b>Identificación</b> <small>* required entero ( \$ int64 ) ( continue )</small>	Identificación

**Respuestas**

Código	Descripción	Campo de golf
200	<p>Cliente encontrado</p> <p>Tipo de medio: <span>aplicación/json</span></p> <p>Encabezado de control: Accept</p> <p>Valor de ejemplo   Esquema</p> <pre>{   "id": 1,   "nombre": "string",   "correo": "string",   "direccion": "string",   "telefono": "string" }</pre>	Sin enlaces
404	<p>Cliente no encontrado</p> <p>Tipo de medio: <span>*/*</span></p> <p>Valor de ejemplo   Esquema</p> <pre>()</pre>	Sin enlaces

**CREAR** /api /cliente Crear un nuevo cliente

Crea un cliente con los datos proporcionados.

**Parámetros** Pruébalo

Sin parámetros

Cuerpo de la solicitud required aplicación/json

Valor de ejemplo | Esquema

```
{
  "id": 1,
  "nombre": "string",
  "correo": "string",
  "direccion": "string",
  "telefono": "string"
}
```

**Respuestas**

Código	Descripción	Campo de golf
201	<p>Cliente creado correctamente</p> <p>Tipo de medio: <span>aplicación/json</span></p> <p>Encabezado de control: Accept</p> <p>Valor de ejemplo   Esquema</p> <pre>{   "id": 1,   "nombre": "string",   "correo": "string",   "direccion": "string",   "telefono": "string" }</pre>	Sin enlaces

**PUT** /api /cliente Actualizar cliente

Actualiza la información de un cliente existente.

**Parámetros** Pruébalo

Sin parámetros

Cuerpo de la solicitud new aplicación/json

Valor de ejemplo | Esquema

```
{
  "id": 1,
  "nombre": "string",
  "apellido": "string",
  "direccion": "string",
  "telefono": "string"
}
```

**Respuestas**

Código	Descripción	Campo de golf
200	Cliente actualizado correctamente	Sin enlaces
404	Cliente no encontrado	Sin enlaces

Valor de ejemplo | Esquema

```
{
  "id": 1,
  "nombre": "string",
  "apellido": "string",
  "direccion": "string",
  "telefono": "string"
}
```

**DELETE** /api /cliente /{id} Eliminar cliente

Eliminar un cliente por su ID

**Parámetros** Pruébalo

Nombre Descripción

Identificación identificación

entero (32 bits) (código)

**Respuestas**

Código	Descripción	Campo de golf
204	Cliente eliminado correctamente	Sin enlaces
404	Cliente no encontrado	Sin enlaces

Valor de ejemplo | Esquema

```
{
  "id": 1,
  "nombre": "string",
  "apellido": "string",
  "direccion": "string",
  "telefono": "string"
}
```

**GET** /api /cliente Obtener lista de clientes

Devuelve todos los clientes disponibles

**Parámetros** Pruébalo

Sin parámetros

**Respuestas**

Código	Descripción	Campo de golf
200	Lista de clientes retornada correctamente	Sin enlaces

Valor de ejemplo | Esquema

```
{
  "id": 1,
  "nombre": "string",
  "apellido": "string",
  "direccion": "string",
  "telefono": "string"
}
```

## Ejecución de pruebas de integración del servicio de Cursos:

**GET** `/api/cursos` Obtener lista de cursos

Devuelve todos los cursos disponibles

**Parameters** Cancel

No parameters

**Execute**

**Responses**

Code	Description	Links
200	Lista de cursos retornada correctamente	No links

Media type:

Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "title": "string",
  "description": "string",
  "publicado": true
}
```

**PUT** `/api/cursos` Actualizar curso

Actualiza la información de un curso existente

**Parameters** Try it out

No parameters

**Request body** required

Example Value | Schema

```
{
  "id": 0,
  "title": "string",
  "description": "string",
  "publicado": true
}
```

**Responses**

Code	Description	Links
200	Curso actualizado correctamente	No links
404	Curso no encontrado	No links

Media type:

Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "title": "string",
  "description": "string",
  "publicado": true
}
```

Media type:

Example Value | Schema

```
{
}
```

**POST** /api/cursos Crear un nuevo curso

Crea un curso con los datos proporcionados

**Parameters** Try it out

No parameters

**Request body** required application/json

Example Value | Schema

```
{
  "id": 0,
  "titulo": "string",
  "descripcion": "string",
  "publicado": true
}
```

**Responses**

Code	Description	Links
201	Curso creado correctamente	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "titulo": "string",
  "descripcion": "string",
  "publicado": true
}
```

**GET** /api/cursos/{id} Obtener curso por ID

Obtiene el detalle de un curso específico

**Parameters** Try it out

Name	Description
<b>id</b> <small>required</small> integer(int64) (path)	<input type="text" value="id"/>

**Responses**

Code	Description	Links
200	Curso encontrado	No links
404	Curso no encontrado	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "titulo": "string",
  "descripcion": "string",
  "publicado": true
}
```

Media type: text

Example Value | Schema

```
{}
```

DELETE /api/cursos/{id} Eliminar curso

Elimina un curso por su ID

Parameters
Try it out

Name	Description
<b>id</b> <small>required</small> <small>integer(\$int64)</small> <small>(path)</small>	<input type="text" value="id"/>

Responses

Code	Description	Links
204	Curso eliminado correctamente <div> Media type  <input type="text" value="*/"/> </div> <div> Controls Accept header </div> <div> Example Value   Schema </div> <div> 0 </div>	No links
404	Curso no encontrado <div> Media type  <input type="text" value="*/"/> </div> <div> Example Value   Schema </div> <div> 0 </div>	No links



## Ejecución de pruebas de integración del servicio de Contenido:

**GET** /api/contenido Obtener lista de contenido

Devuelve todos los contenidos disponibles

**Parameters** [Try it out](#)

No parameters

**Responses**

Code	Description	Links
200	Lista de contenidos retornada correctamente	No links

Media type:

Controls Accept header:

Example Value | Schema

```
{
  "id": 1,
  "nombre": "string",
  "tipo": "string"
}
```

**PUT** /api/contenido Actualizar contenido

Actualiza la información de un contenido existente

**Parameters** [Try it out](#)

No parameters

**Request body** <sup>required</sup>

Example Value | Schema

```
{
  "id": 1,
  "nombre": "string",
  "tipo": "string"
}
```

**Responses**

Code	Description	Links
200	Contenido actualizado correctamente	No links
404	Contenido no encontrado	No links

Media type:

Controls Accept header:

Example Value | Schema

```
{
  "id": 1,
  "nombre": "string",
  "tipo": "string"
}
```

**POST** /api/contenido Crear un nuevo contenido

Crea un contenido con los datos proporcionados

**Parameters** [Try it out](#)

No parameters

**Request body** <sup>required</sup>

Example Value | Schema

```
{
  "id": 1,
  "nombre": "string",
  "tipo": "string"
}
```

**Responses**

Code	Description	Links
201	Contenido creado correctamente	No links

Media type:

Controls Accept header:

Example Value | Schema

```
{
  "id": 1,
  "nombre": "string",
  "tipo": "string"
}
```

**GET** /api/contenido/{id} Obtener contenido por ID

Obtiene el detalle de un contenido específico

**Parameters** [Try it out](#)

Name	Description
<b>id</b> * required Integer(\$int64) (path)	id

**Responses**

Code	Description	Links
200	Contenido encontrado Media type: application/json Controls Accept header Example Value   Schema <pre>{   "id": 1,   "nombre": "string",   "tipo": "string" }</pre>	No links
404	Contenido no encontrado Media type: */* Example Value   Schema <pre>{}</pre>	No links

**DELETE** /api/contenido/{id} Eliminar contenido

Elimina un contenido por su ID

**Parameters** [Try it out](#)

Name	Description
<b>id</b> * required Integer(\$int64) (path)	id

**Responses**

Code	Description	Links
204	Contenido eliminado correctamente Media type: */* Controls Accept header Example Value   Schema <pre>{}</pre>	No links
404	Contenido no encontrado Media type: */* Example Value   Schema <pre>{}</pre>	No links

# Git - GitHub

Para el flujo de trabajo se utilizaron tres branches distintas, uno por cada integrante donde primero, en la carpeta donde cada uno trabajó se descargó en archivo **.zip** el main en vez de hacer un git clone, luego desde ese punto se continuó trabajando.

Luego de haber hecho todos los avances, primero se hizo un “**git add .**” para subir los archivos modificados y nuevos:

```
renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (renato)
$ git add .
```

```
Diego@Papudiego MINGW64 ~/OneDrive/Desktop/Proyecto semestral/Proyecto-Semestral
(diego)
$ git add .
warning: in the working copy of 'springboot-backend/pom.xml', LF will be replace
d by CRLF the next time Git touches it
warning: in the working copy of 'springboot-backend/src/main/java/com/proyecto/s
pringboot/backend/springboot_backend/controllers/IncidenciaController.java', LF
will be replaced by CRLF the next time Git touches it
```

```
maxim@DESKTOP-8IGITE2 MINGW64 ~/OneDrive/Documentos/proyectoSemestral/Proyecto-Semestral (mac)
$ git add .
warning: in the working copy of 'springboot-backend/pom.xml', LF will be replaced by CRLF the next time Git touches i
warning: in the working copy of 'springboot-backend/src/main/java/com/proyecto/springboot/backend/springboot_backend/
controllers/CursoController.java', LF will be replaced by CRLF the next time Git touches it
```

Luego, se hizo un commit y un push para subir los cambios hechos en cada branch:

```
renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (renato)
$ git commit -m "Cambios hechos a Usuario, pruebas unitarias y de integracion"
[renato ad098a0] Cambios hechos a Usuario, pruebas unitarias y de integracion
2 files changed, 0 insertions(+), 0 deletions(-)
rename Renato.txt => Carpeta para la foto/Renato.txt (100%)
create mode 100644 Carpeta para la foto/Renato.txt

renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (renato)
$ git push origin renato
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 371 bytes | 371.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/RenatoValenzuela262/Proyecto-Semestral.git
84e67ed..ad098a0 renato -> renato
```

```
Diego@Papudiego MINGW64 ~/OneDrive/Desktop/Proyecto semestral/Proyecto-Semestral
(diego)
$ git commit -m "Tests e integracion de los servicios de incidencias y clientes"
[diego aa0974e] Tests e integracion de los servicios de incidencias y clientes
7 files changed, 365 insertions(+), 3 deletions(-)
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/bac
kend/springboot_backend/restcontrollers/ClienteRestControllersTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/bac
kend/springboot_backend/restcontrollers/IncidenciaRestControllersTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/bac
kend/springboot_backend/services/ClienteServiceImplTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/bac
kend/springboot_backend/services/IncidenciaServiceImplTest.java

Diego@Papudiego MINGW64 ~/OneDrive/Desktop/Proyecto semestral/Proyecto-Semestral
(diego)
$ git push origin diego
Enumerating objects: 45, done.
Counting objects: 100% (45/45), done.
Delta compression using up to 22 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (28/28), 5.23 KiB | 1.31 MiB/s, done.
Total 28 (delta 9), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (9/9), completed with 6 local objects.
To https://github.com/RenatoValenzuela262/Proyecto-Semestral.git
 35736e2..aa0974e  diego -> diego
```

```
maxim@DESKTOP-8IGITE2 MINGW64 ~/OneDrive/Documentos/proyectoSemestral/Proyecto-Semestral (mac)
$ git commit -m "Contenido y Cursos, pruebas unitarias y de integracion"
[mac 3edcb47] Contenido y Cursos, pruebas unitarias y de integracion
8 files changed, 364 insertions(+), 1 deletion(-)
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/back
end/springboot_backend/restcontrolle
s/ContenidoRestControllersTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/back
end/springboot_backend/restcontrolle
s/CursoRestControllersTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/back
end/springboot_backend/services/Cont
nidoServiceImplTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/back
end/springboot_backend/services/Curs
ServiceImplTest.java

maxim@DESKTOP-8IGITE2 MINGW64 ~/OneDrive/Documentos/proyectoSemestral/Proyecto-Semestral (mac)
$ git push origin mac
Enumerating objects: 48, done.
Counting objects: 100% (48/48), done.
Delta compression using up to 6 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (30/30), 5.13 KiB | 875.00 KiB/s, done.
Total 30 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), completed with 5 local objects.
To https://github.com/RenatoValenzuela262/Proyecto-Semestral.git
 fba7e64..3edcb47  mac -> mac
```

Finalmente, en la rama main se hace un merge para combinar los avances hechos en las tres branches:

```
renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral/Proyecto-S
emestral (main)
$ git merge renato main
Updating 863b0a0..ad098a0
Fast-forward
 Carpeta para la foto/Renato 2.txt          | 0
 Carpeta para la foto/Renato.txt           | 0
 springboot-backend/pom.xml                 | 16 +++++
 .../controllers/UsuarioController.java     | 90 ++++++-----
 .../springboot_backend/entities/Usuario.java | 23 +++++
 .../services/UsuarioService.java          | 4 +-
 .../services/UsuarioServiceImpl.java       | 11 +--
 .../src/main/resources/application.properties | 6 +-
 .../UsuarioRestControllersTest.java        | 79 ++++++
 .../services/UsuarioServiceImplTest.java    | 55 ++++++
10 files changed, 233 insertions(+), 51 deletions(-)
 create mode 100644 Carpeta para la foto/Renato 2.txt
 create mode 100644 Carpeta para la foto/Renato.txt
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/bac
kend/springboot_backend/restcontrollers/UsuarioRestControllersTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/bac
kend/springboot_backend/services/UsuarioServiceImplTest.java
```

```
renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral/Proyecto-Semestral (main)
$ git commit -m "Mezcla de branch renato y main"
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral/Proyecto-Semestral (main)
$ git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/RenatoValenzuela262/Proyecto-Semestral.git
 863b0a0..ad098a0  main -> main
```

```
renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main)
$ git merge mac main
```

```
renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main|MERGING)
$ git commit -m "Merge mac con main"
[main 408f2a1] Merge mac con main

renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main)
$ git push
Enumerating objects: 57, done.
Counting objects: 100% (57/57), done.
Delta compression using up to 12 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (23/23), 1.73 KiB | 441.00 KiB/s, done.
Total 23 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/RenatoValenzuela262/Proyecto-Semestral.git
 ad098a0..408f2a1  main -> main
branch 'main' set up to track 'origin/main'.
```

```
renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main|MERGING)
$ git merge diego main
fatal: You have not concluded your merge (MERGE_HEAD exists).
Please, commit your changes before you merge.

renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main|MERGING)
$ git commit -m "Merge diego con main"
[main 2f9e3ab] Merge diego con main

renax@DESKTOP-37USU06 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main)
$ git push
Enumerating objects: 60, done.
Counting objects: 100% (60/60), done.
Delta compression using up to 12 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (24/24), 2.04 KiB | 522.00 KiB/s, done.
Total 24 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (7/7), completed with 7 local objects.
To https://github.com/RenatoValenzuela262/Proyecto-Semestral.git
 408f2a1..2f9e3ab  main -> main
```

Las ramas de “mac” y de “diego” dieron unos errores al hacer merge debido a configuraciones específicas que tuvieron que cambiar en sus estaciones de trabajo, pero se resolvió (por eso los errores en las imágenes).

Finalmente, el archivo final pasa todos los tests:

```

✓ 26/26 11.1s
✓ [ ] springboot-backend 522ms
  ✓ { } com.proyecto.springboot.backend.springboot_backend 2.0ms
    > ✓ [ ] SpringbootBackendApplicationTests 2.0ms
  ✓ { } com.proyecto.springboot.backend.springboot_backend.restcontrollers 291...
    > ✓ [ ] ClienteRestControllersTest 26ms
    > ✓ [ ] ContenidoRestControllersTest 26ms
    > ✓ [ ] CursoRestControllersTest 31ms
    > ✓ [ ] IncidenciaRestControllersTest 182ms
    > ✓ [ ] UsuarioRestControllersTest 26ms
  ✓ { } com.proyecto.springboot.backend.springboot_backend.services 229ms
    > ✓ [ ] ClienteServiceImplTest 37ms
    > ✓ [ ] ContenidoServiceImplTest 35ms
    > ✓ [ ] CursoServiceImplTest 34ms
    > ✓ [ ] IncidenciaServiceImplTest 89ms
    > ✓ [ ] UsuarioServiceImplTest 34ms

```

Readme:

<https://github.com/RenatoValenzuela262/Proyecto-Semestral/blob/main/README.md>

## Conclusión

En conclusión, el análisis y documentación del proceso de pruebas para el sistema de microservicios representa un componente esencial en la consolidación de una arquitectura robusta y escalable para *EduTech Innovators SPA*. Este informe abarcó desde el diseño de la arquitectura y codificación de pruebas, hasta la ejecución documentada y el control de versiones a través de GitHub.

La inclusión de pruebas unitarias e integrales, acompañadas de su respectiva justificación técnica y uso de frameworks especializados, permitió validar el correcto funcionamiento de los servicios. Así mismo, la evidencia gráfica del proceso de implementación y prueba proporciona transparencia y trazabilidad.

Por otro lado, el uso de comandos Git y la correcta gestión de ramas aseguran buenas prácticas en la colaboración del equipo, permitiendo mantener versiones limpias, organizadas y documentadas del código.

En resumen, este trabajo no solo fortalece la calidad técnica del proyecto, sino que establece un precedente para futuras etapas de desarrollo, garantizando mantenibilidad, escalabilidad y eficiencia operativa.