Informe ET

Proyecto semestral:

“EduTech Innovators SPA”

Integrantes:

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

Índice

Problema

El problema principal de EduTech Innovators SPA es en su sistema de software monolítico actual, que en un inicio podría haber servido con un número limitado de usuarios, pero ahora que la demanda de usuario a crecido exponencialmente, este software que en un inicio funcionaba bien ahora está sufriendo problemas de rendimiento como tiempos lentos de respuesta, transacciones no procesadas a tiempo o incluso caídas del sistema. Esto debido a lo complejo que es escalar un sistema de software monolítico.

Análisis de requerimientos:

En cuanto a las necesidades del cliente, se organizó una lista de requerimientos, la cual se dividió en dos categorías: aquellos con interacción directa con el usuario (requisitos funcionales) y aquellos relacionados con el funcionamiento interno de la aplicación (requisitos no funcionales). A continuación, los ya mencionados:

Requisitos funcionales

- Gestión de usuarios
- Configuración de permisos
- Respaldo y restauración de datos
- Monitorización del sistema

Requisitos no funcionales

- Un sistema de microservicios
- Un motor de base de datos SQL
- Plan de migración
- Análisis de requerimientos
- Análisis del sistema actual
- Diseño de la nueva arquitectura.

Sistema actual

El sistema actual de EduTech Innovators SPA es un sistema monolítico, lo que significa que todas las funcionalidades están integradas en una única aplicación. Esto incluye la gestión de usuarios, la administración de cursos, la entrega de contenido educativo y la evaluación de estudiantes.

Como componentes posee una interfaz de usuario, manejo de inscripciones como a su vez el seguimiento del progreso una vez hecha la inscripción y un sistema de base de datos para almacenar la información de los usuarios, como así también, el contenido de los cursos.

Nueva arquitectura

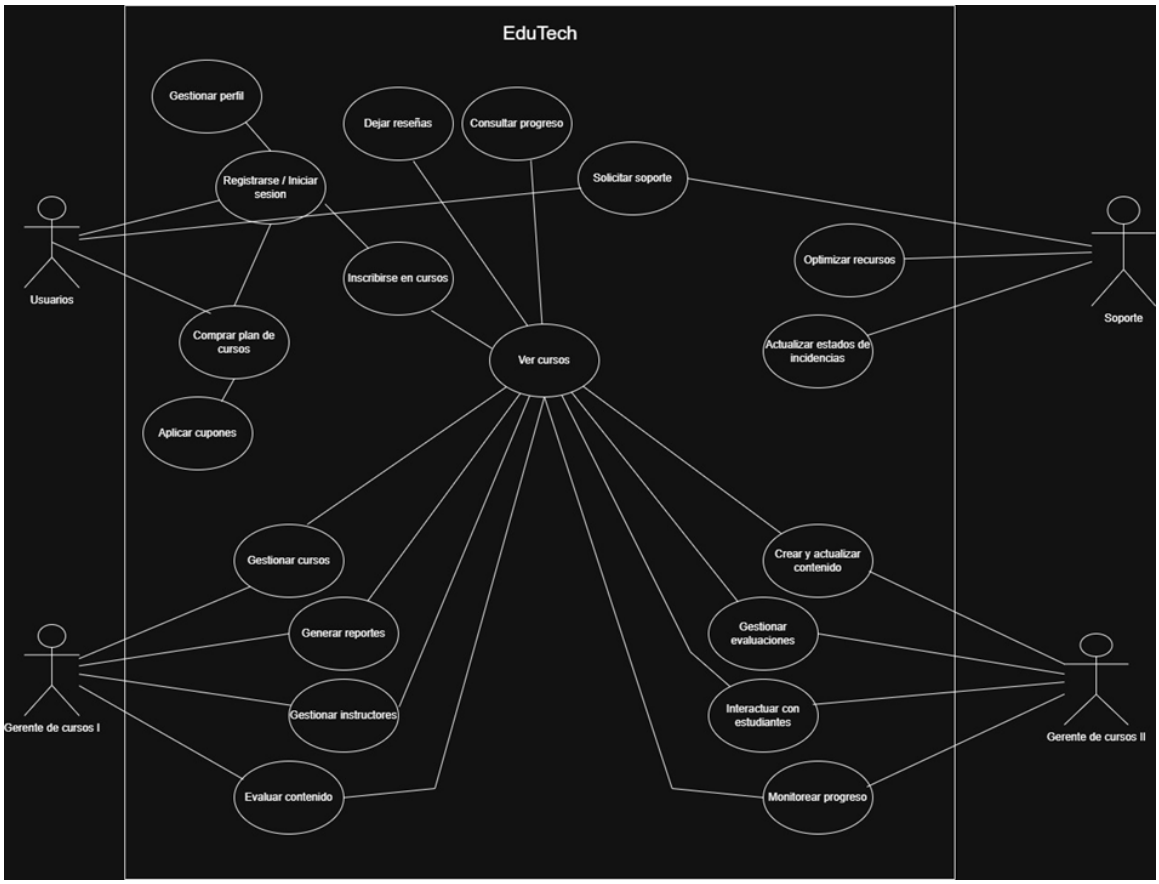
Teniendo esto en cuenta como equipo haciendo análisis de los requerimientos, decidimos que la forma más eficiente de llevar las necesidades actuales de la empresa es cambiando su estructura de sistema monolítico a una arquitectura de microservicios, debido a que este sistema se basa en dividir una aplicación en pequeños servicios independientes que se encargan de tareas específicas que se comunican entre sí mediante APIs o mensajes, cada microservicio es autónomo dando así una mayor flexibilidad y escalabilidad.

Esto es conveniente igual por el lado de tener una escalabilidad independiente, un desarrollo más ágil, la resiliencia que vendría siendo que cuando un microservicio falle, este no afectará a toda la aplicación, tendremos una flexibilidad tecnológica ya que los microservicios pueden ser desarrollados usando distintas tecnologías (como lenguajes de programación) y finalmente, tendrá un mantenimiento mucho más simple al ser su identificación y corrección de errores más sencilla.

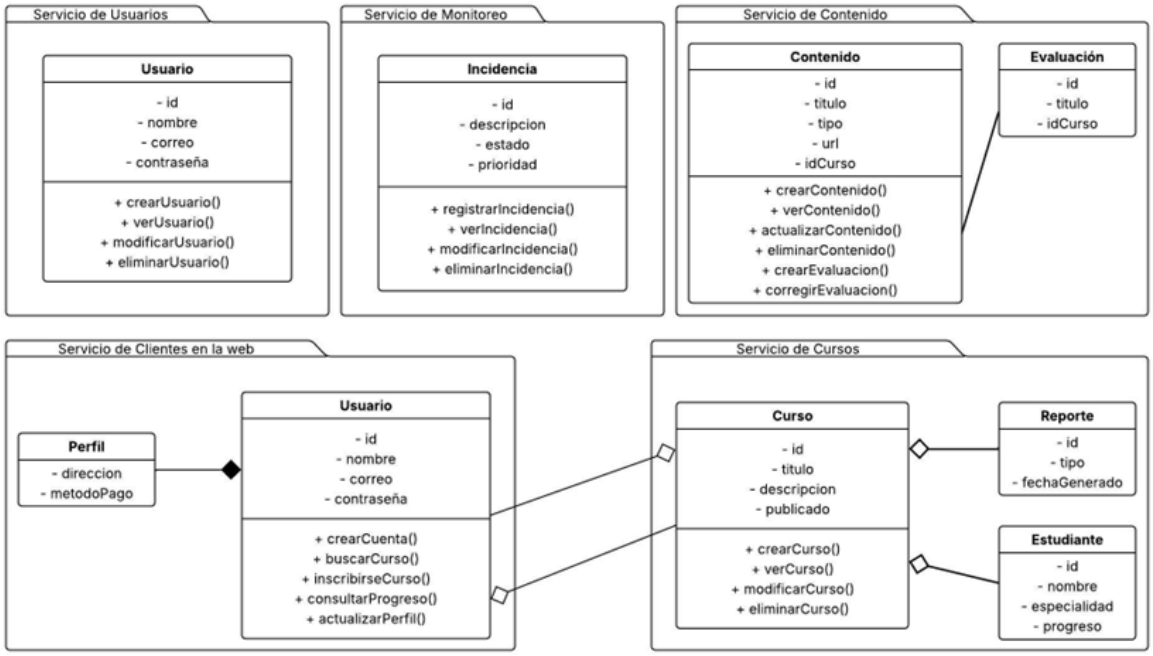
Esto resulta conveniente, ya que permite una escalabilidad independiente, facilita un desarrollo más ágil y aporta resiliencia: si un microservicio falla, no afectará al funcionamiento de toda la aplicación. Además, ofrece flexibilidad tecnológica, ya que cada microservicio puede desarrollarse con tecnologías distintas (como también diferentes lenguajes de programación). Por último, el mantenimiento se vuelve más sencillo, ya que la identificación y corrección de errores es más rápida y localizada.

Diagrama de arquitectura de microservicios

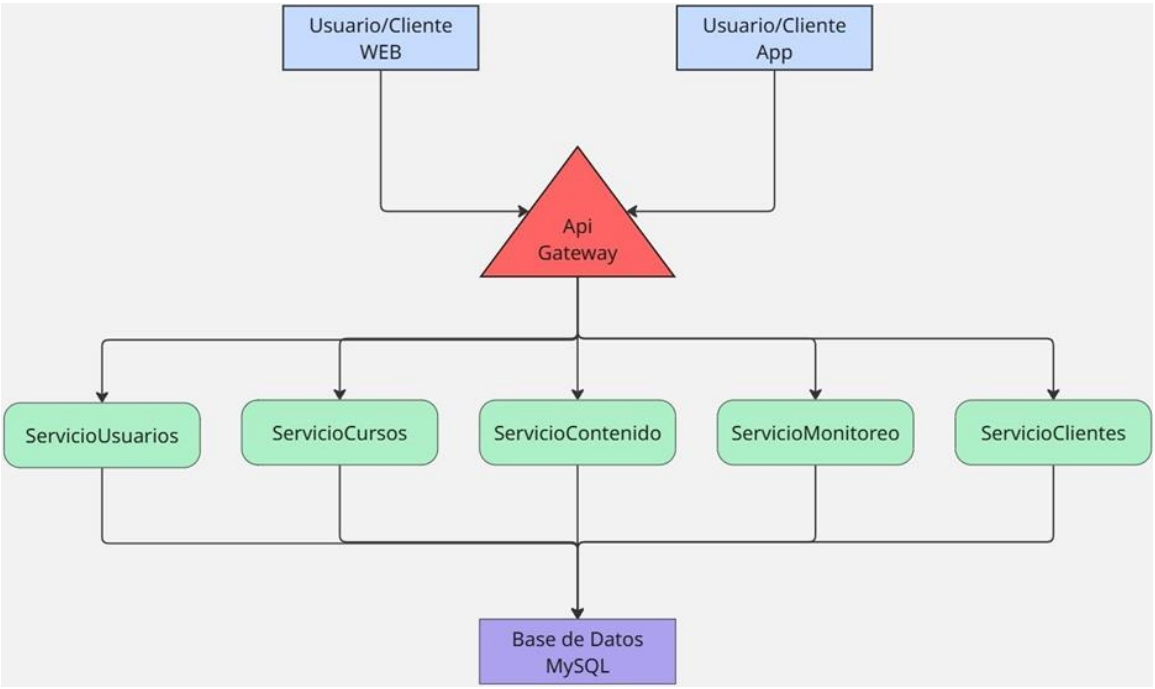
Para explicar esta nueva arquitectura, a continuación, se presentan tres diagramas que ilustran distintos aspectos clave del sistema.
El diagrama de casos de uso muestra las principales interacciones entre los actores y las funcionalidades del sistema:



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.



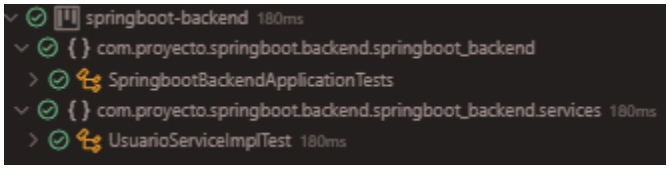
Plan de pruebas

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:

```java
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:"contraUsuariouno");
        Usuario usuario2 = new Usuario(Long.valueOf(1:204004008),nombre:"Usuario dos",correo:"usuariodos@duocuc.cl",contrasenia:"contraUsuariodos");
        Usuario usuario3 = new Usuario(Long.valueOf(1:205005007),nombre:"Usuario tres",correo:"usuariotres@duocuc.cl",contrasenia:"contraUsuariotres");

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

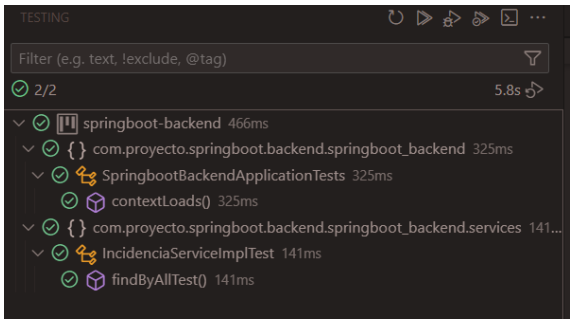


Pruebas unitarias del servicio de Mantenimiento:

pom.xml 1, M | IncidenciaServiceImplTest.java U | SpringbootBackendApplicationTests.java | Incidencia.java

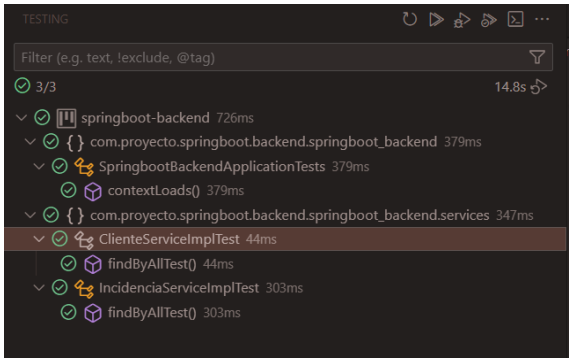
Proyecto-Semestral > springboot-backend > src > test > java > com > proyecto > springboot > backend > springboot_backend > services > IncidenciaServiceImplTest.java > IncidenciaServiceImplTest

```java
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.entities.Incidencia;
import com.proyecto.springboot.backend.springboot_backend.repository.IncidenciaRepository;


public class IncidenciaServiceImplTest {

    @InjectMocks
    private IncidenciaServiceImpl service;

    @Mock
    private IncidenciaRepository repository;

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

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

        this.chargeIncidencia();
    }

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

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

        assertEquals(expected:4, response.size());
        verify(repository, times(wantedNumberOfInvocations:1)).findAll();
    }

    public void chargeIncidencia(){
        Incidencia inci1 = new Incidencia(Long.valueOf(1:1), descripcion:"Primera prueba de Incidencias", estado:"Inactivo", prioridad:"Alta" );
        Incidencia inci2 = new Incidencia(Long.valueOf(1:2), descripcion:"Segunda prueba de Incidencias", estado:"Inactivo", prioridad:"Baja" );
        Incidencia inci3 = new Incidencia(Long.valueOf(1:3), descripcion:"Probando errores", estado:"Activo", prioridad:"Media" );
        Incidencia inci4 = new Incidencia(Long.valueOf(1:4), descripcion:"Probando soluciones", estado:"Inactivo", prioridad:"Alta" );

        list.add(inci1);
        list.add(inci2);
        list.add(inci3);
        list.add(inci4);
    }
}
```



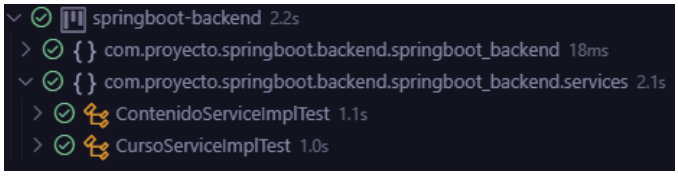
Pruebas unitarias del servicio de Clientes (en la web):

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

import static org.junit.jupiter.api.Assertions.assertEquals;
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.entities.Cliente;
import com.proyecto.springboot.backend.springboot_backend.repository.ClienteRepository;

public class ClienteServiceImplTest {

    @InjectMocks
    private ClienteServiceImpl service;

    @Mock
    private ClienteRepository repository;

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

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

        this.chargeCliente();
    }

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

        List<Cliente> response = service.findByAll();
        assertEquals(expected:4, response.size());
        verify(repository, times(wantedNumberOfInvocations:1)).findAll();
    }

    public void chargeCliente(){
        Cliente cli1 = new Cliente(Long.valueOf(1L), nombre:"Diego", correo:"di.zamorag@duocuc.cl", direccion:"Chiringuito Chatarra", metodoPago:"Visa" );
        Cliente cli2 = new Cliente(Long.valueOf(2L), nombre:"Maximiliano", correo:"mac@duocuc.cl", direccion:"Pisos picados", metodoPago:"Visa" );
        Cliente cli3 = new Cliente(Long.valueOf(3L), nombre:"Renato", correo:"ren@duocuc.cl", direccion:"Parque placentero", metodoPago:"Masterdcard" );
        Cliente cli4 = new Cliente(Long.valueOf(4L), nombre:"Anakin", correo:"darthvader@duocuc.cl", direccion:"Señorio de la sal", metodoPago:"Centurion" );

        list.add(cli1);
        list.add(cli2);
        list.add(cli3);
        list.add(cli4);
    }
}
```



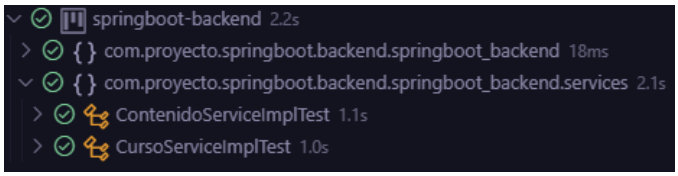
Pruebas unitarias del servicio de Cursos:

```java
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.entities.Curso;
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);
    }
```



Pruebas unitarias del servicio de Contenido:

```java
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.entities.Contenido;
import com.proyecto.springboot.backend.springboot_backend.repository.ContenidoRepository;


public class ContenidoServiceImplTest {

    @InjectMocks
    private ContenidoServiceImpl service;

    @Mock
    private ContenidoRepository repository;

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


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

        this.chargeContenido();
    }

    @Test
    public void findByAllTest(){

        when(repository.findAll()).thenReturn(list);

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

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

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

    public void chargeContenido(){
        Contenido cont1 = new Contenido(Long.valueOf(1:1), nombre:"Ecuaciones", tipo:"Materia");
        Contenido cont2 = new Contenido(Long.valueOf(1:2), nombre:"Sustantivos Propios", tipo:"Materia");
        Contenido cont3 = new Contenido(Long.valueOf(1:3), nombre:"Verbs", tipo:"Materia");

        list.add(cont1);
        list.add(cont2);
        list.add(cont3);
```



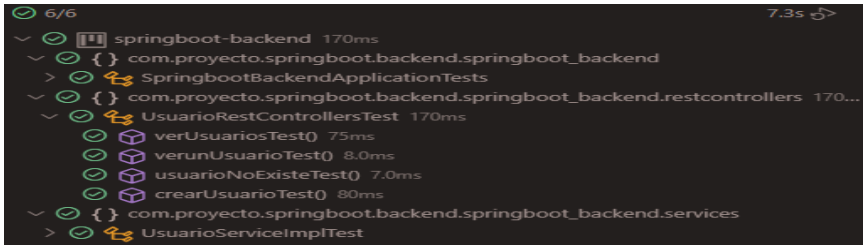
Ahora, las pruebas de integración:

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.

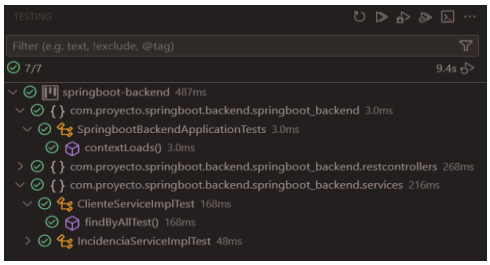
Estas pruebas permiten verificar que los controladores, servicios y otros componentes funcionen correctamente de forma integrada. Dependiendo de la configuración, las peticiones pueden interactuar con una base de datos real o en memoria. Para ejecutar estos testeos también se ocupa como motor de pruebas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

Pruebas de integración del servicio de Usuarios:

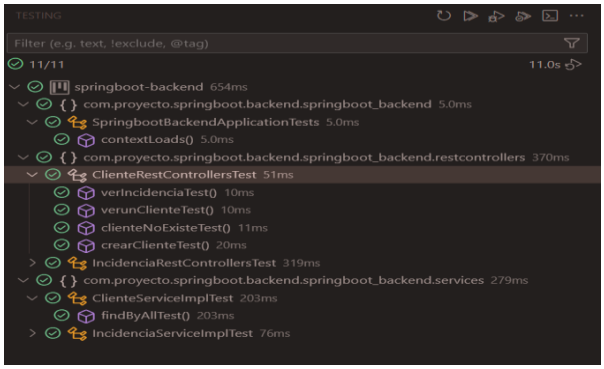
```java
package com.proyecto.springboot.backend.springboot_backend.restcontrollers;

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 UsuarioRestControllersTest {

  @Autowired
  private MockMvc mockmvc;

  @Autowired
  private ObjectMapper objectMapper;

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

  @Test
  public void verUsuariosTest() throws Exception{
    when(usuarioserviceimpl.findByAll()).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(1:203003009), nombre:"Usuario1", correo:"usuariouno@duocuc.cl", contrasenia:"contraUsuariouno");
    try{
      when(usuarioserviceimpl.findByRut(Long.valueOf(1:203003009))).thenReturn(Optional.of(unUsuario));
      mockmvc.perform(get(uriTemplate:"/api/usuarios/203003009")
      .contentType(MediaType.APPLICATION_JSON))
      .andExpect(status().isOk());
    }
    catch(Exception ex){
      fail("El Testing lanzo un Error"+ ex.getMessage());
    }
  }

  @Test
  public void usuarioNoExisteTest() throws Exception{
    when(usuarioserviceimpl.findByRut(Long.valueOf(1:203003009))).thenReturn(Optional.empty());
    mockmvc.perform(get(uriTemplate:"/api/usuarios/203003009")
    .contentType(MediaType.APPLICATION_JSON))
    .andExpect(status().isNotFound());
  }

  @Test
  public void crearUsuarioTest() throws Exception{
    Usuario unUsuario = new Usuario(Long.valueOf(1:203003009),nombre:"Usuario uno",correo:"usuariouno@duocuc.cl",contrasenia:"contraUsuariouno");
    Usuario otroUsuario = new Usuario(Long.valueOf(1:204004008),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());
  }
}
```



Pruebas de integración del servicio de Mantenimiento:

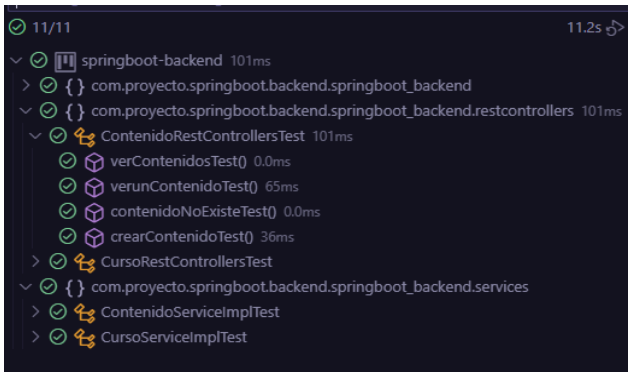


Pruebas de integración para el servicio de Cliente (en la web):



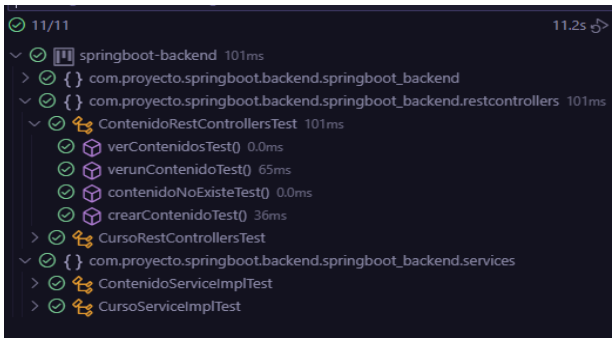
Pruebas de integración del servicio de Curso:

```java
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.Curso;
import com.proyecto.springboot.backend.springboot_backend.services.CursoServiceImpl;
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 CursoRestControllersTest {

  @Autowired
  private MockMvc mockmvc;

  @Autowired
  private ObjectMapper objectMapper;

  @MockitoBean
  private CursoServiceImpl cursoserviceimpl;
  private List<Curso> cursosLista;
  @Test
  public void verProductosTest() throws Exception{
    when(cursoserviceimpl.findByAll()).thenReturn(cursosLista);
    mockmvc.perform(get(uriTemplate:"/api/cursos")
    .contentType(MediaType.APPLICATION_JSON))
    .andExpect(status().isOk());
  }
  @Test
  public void verunCursoTest(){
    Curso unCurso = new Curso(Long.valueOf(1:1), titulo:"Ingles", descripcion:"Nivelacion", publicado:true);

    try{
      when(cursoserviceimpl.findById(Long.valueOf(1:1))).thenReturn(Optional.of(unCurso));
      mockmvc.perform(get(uriTemplate:"/api/cursos/1")
      .contentType(MediaType.APPLICATION_JSON))
      .andExpect(status().isOk());
    }
    catch(Exception ex){
      fail("El Testing lanzo un Error"+ ex.getMessage());
    }

  }
  @Test
  public void cursoNoExisteTest() throws Exception{
    when(cursoserviceimpl.findById(Long.valueOf(1:1))).thenReturn(Optional.empty());
    mockmvc.perform(get(uriTemplate:"/api/productos/10")
    .contentType(MediaType.APPLICATION_JSON))
    .andExpect(status().isNotFound());

  }
  @Test
  public void crearProductoTest() throws Exception{
    Curso unCurso = new Curso(Long.valueOf(1:1), titulo:"Ingles", descripcion:"Nivelacion", publicado:true);
    Curso otroCurso = new Curso(Long.valueOf(1:2), titulo:"Lenguaje", descripcion:"Nivelacion", publicado:true);
    when(cursoserviceimpl.save(any(type:Curso.class))).thenReturn(otroCurso);
    mockmvc.perform(post(uriTemplate:"/api/cursos")
    .contentType(MediaType.APPLICATION_JSON)
    .content(objectMapper.writeValueAsString(unCurso)))
    .andExpect(status().isCreated());
  }
}
```



Pruebas de integración del servicio de Contenido:

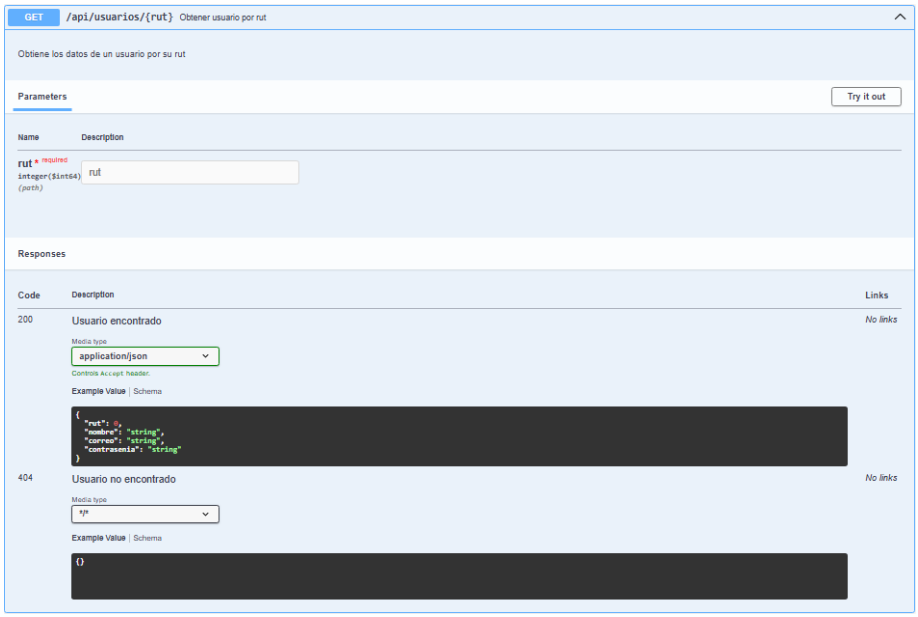
```java
import static org.mockito.Mockito.when;
import java.util.List;
import java.util.Optional;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
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 verContenidosTest() throws Exception{
    when(contenidoserviceimpl.findByAll()).thenReturn(contenidosLista);
    mockmvc.perform(get(uriTemplate:"/api/contenido")
    .contentType(MediaType.APPLICATION_JSON))
    .andExpect(status().isOk());
  }
  @Test
  public void verunContenidoTest(){
    Contenido unContenido = new Contenido(Long.valueOf(1:1), nombre:"Ingles", tipo:"Materia");
    try{
      when(contenidoserviceimpl.findById(Long.valueOf(1:1))).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(1:1))).thenReturn(Optional.empty());
    mockmvc.perform(get(uriTemplate:"/api/contenido/1")
    .contentType(MediaType.APPLICATION_JSON))
    .andExpect(status().isNotFound());
  }

  @Test
  public void crearContenidoTest() throws Exception{
    Contenido unContenido = new Contenido(Long.valueOf(1:1), nombre:"Matematicas", tipo:"Materia");
    Contenido otroContenido = new Contenido(Long.valueOf(1:2), 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());
  }
}
```

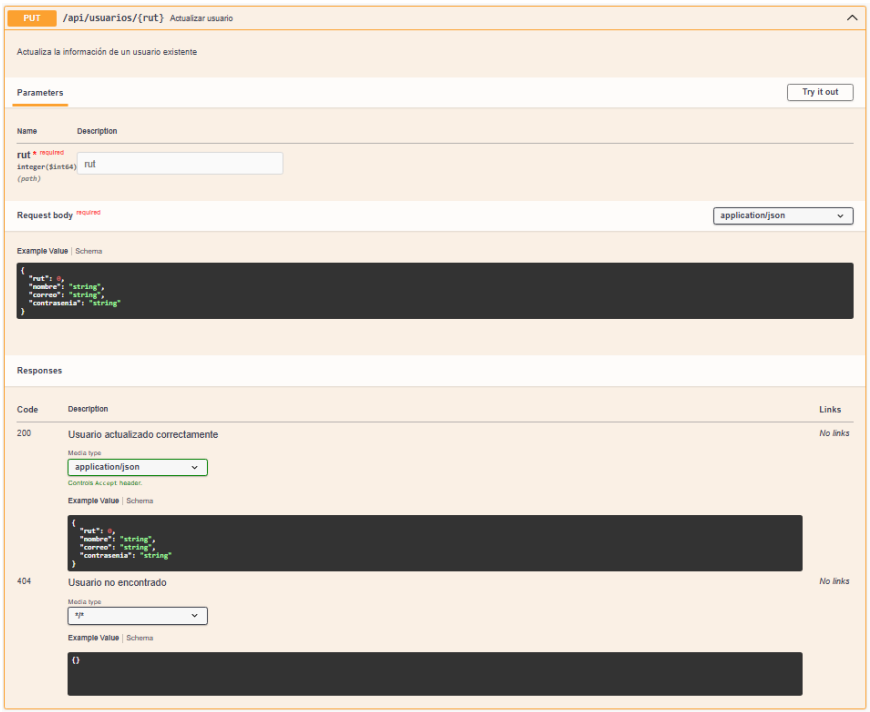

11/11
11.2s
springboot-backend 101ms
com.proyecto.springboot.backend.springboot_backend
com.proyecto.springboot.backend.springboot_backend.restcontrollers 101ms
ContenidoRestControllersTest 101ms
verContenidosTest() 0.0ms
verunContenidoTest() 65ms
contenidoNoExisteTest() 0.0ms
crearContenidoTest() 36ms
CursoRestControllersTest
com.proyecto.springboot.backend.springboot_backend.services
ContenidoServiceImplTest
CursoServiceImplTest

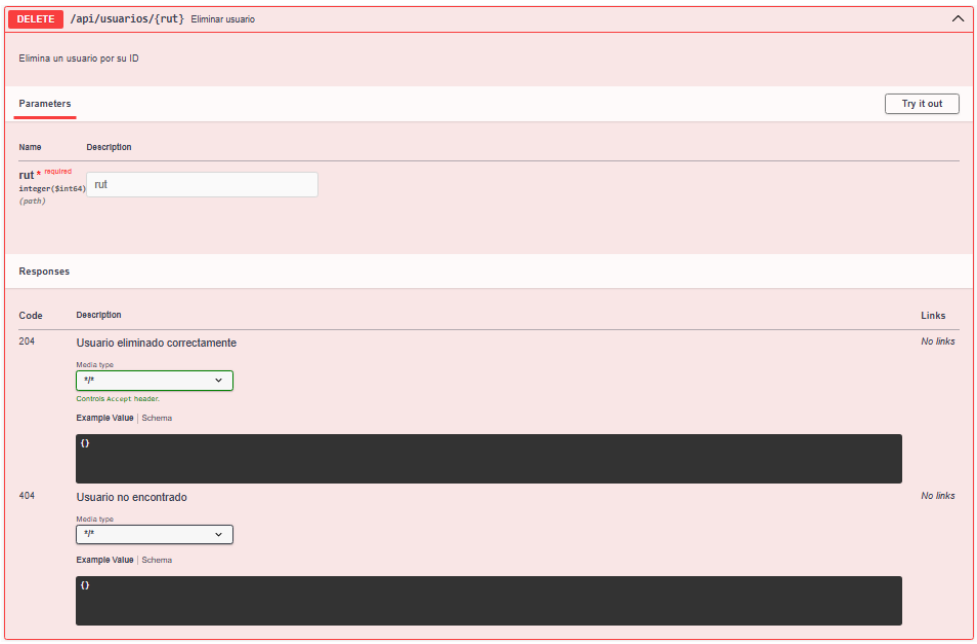
Ejecución de Pruebas

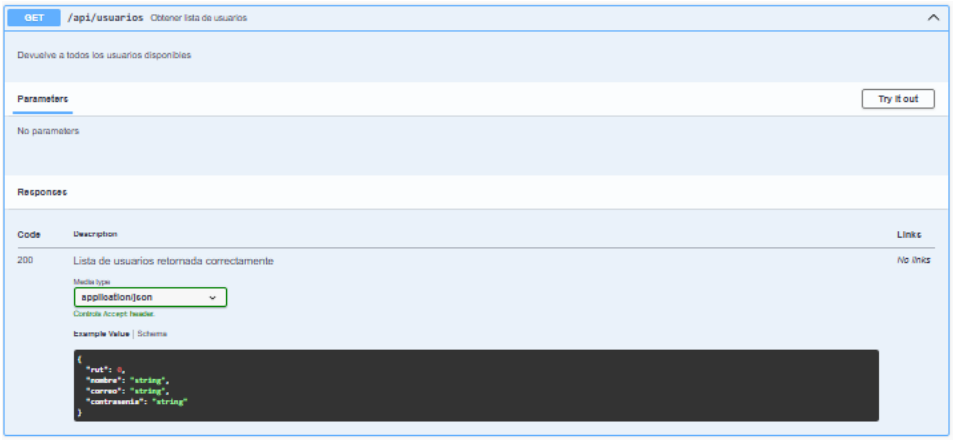
A continuación, la documentación de las pruebas vista en la herramienta llamada Swagger, la cual nos ayuda a probar los endpoints (Get, Post, Put, etc) desde el navegador sin necesidad de usar otras herramientas (como Postman). Permite ver: qué endpoints existen, qué parámetros necesita, qué devuelven,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

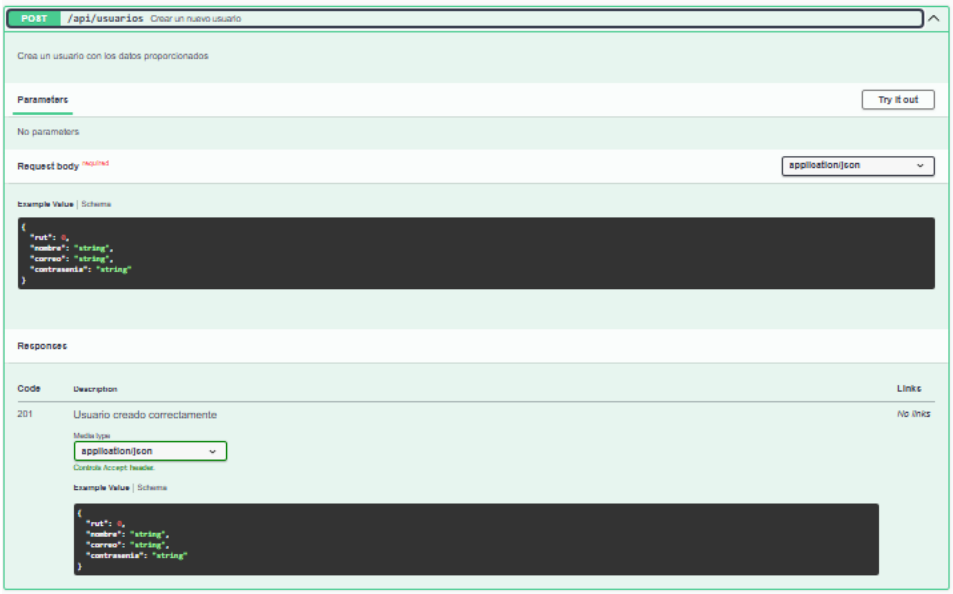
Ejecución de pruebas de integración del servicio de Usuario:



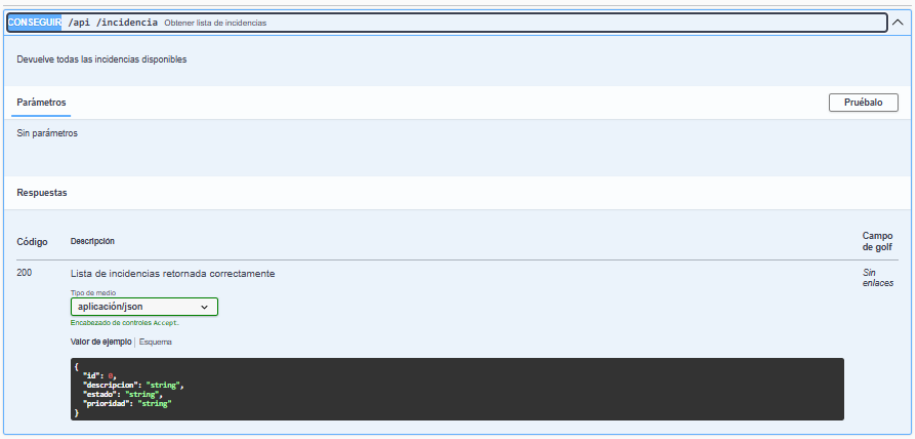


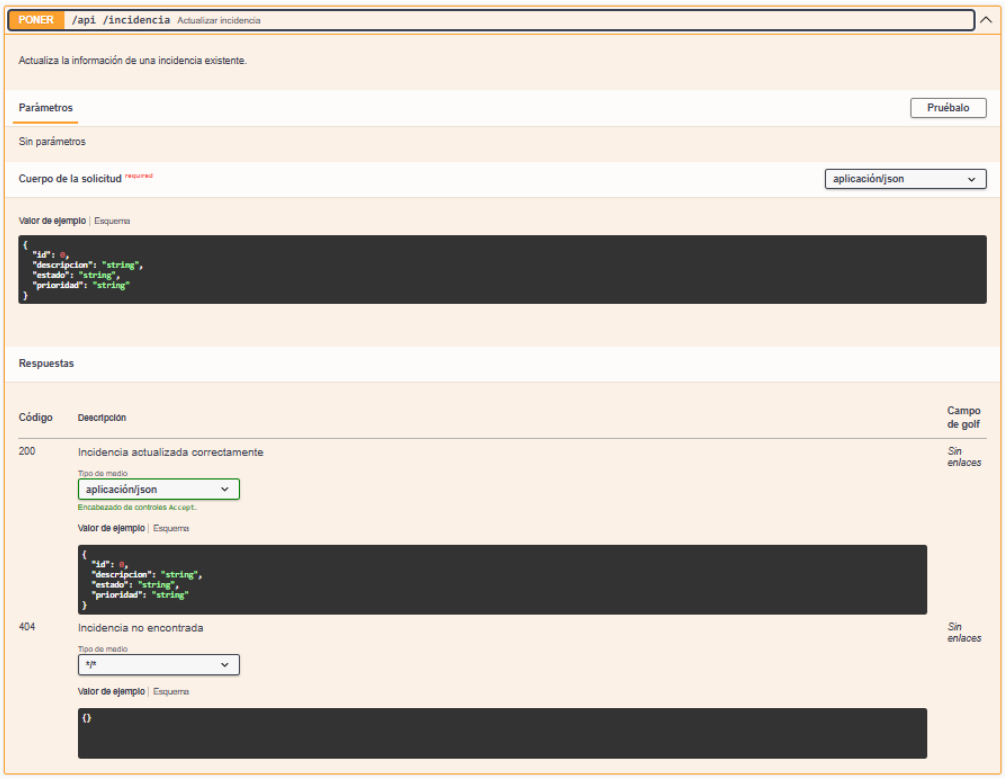

DELETE /api/usuarios/{rut} Eliminar usuario
Elimina un usuario por su ID
Parameters
Try it out
Name
Description
rut * required
integer($int64)
(path)
rut
Responses
Code
Description
Links
204
Usuario eliminado correctamente
No links
Media type
/
Controls Accept header.
Example Value | Schema
{}
404
Usuario no encontrado
No links
Media type
/
Example Value | Schema
{}

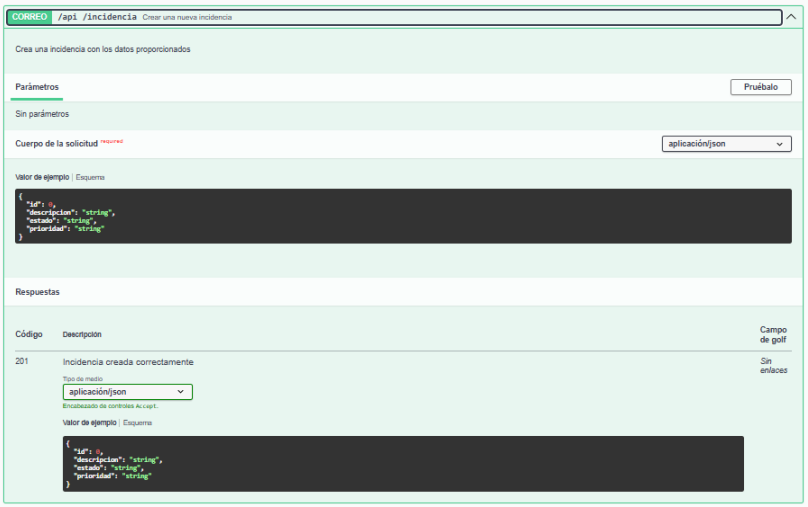

GET /api/usuarios Obtener lista de usuarios
Devuelve a todos los usuarios disponibles
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
Lista de usuarios retornada correctamente
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"rut": 0,
"nombre": "string",
"correo": "string",
"contrasenia": "string"
}

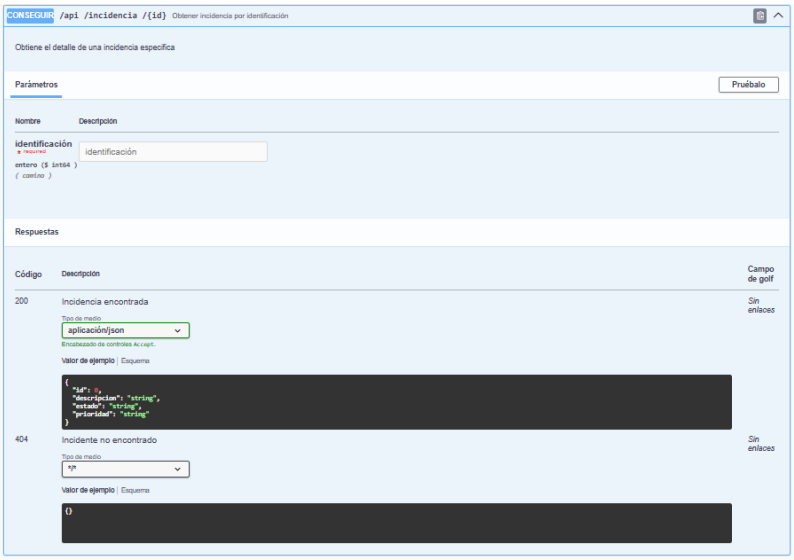

POST /api/usuarios Crear un nuevo usuario
Crea un usuario con los datos proporcionados
Parameters
Try it out
No parameters
Request body required
application/json
Example Value | Schema
{
"rut": 0,
"nombre": "string",
"correo": "string",
"contrasenia": "string"
}
Responses
Code
Description
Links
201
Usuario creado correctamente
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"rut": 0,
"nombre": "string",
"correo": "string",
"contrasenia": "string"
}

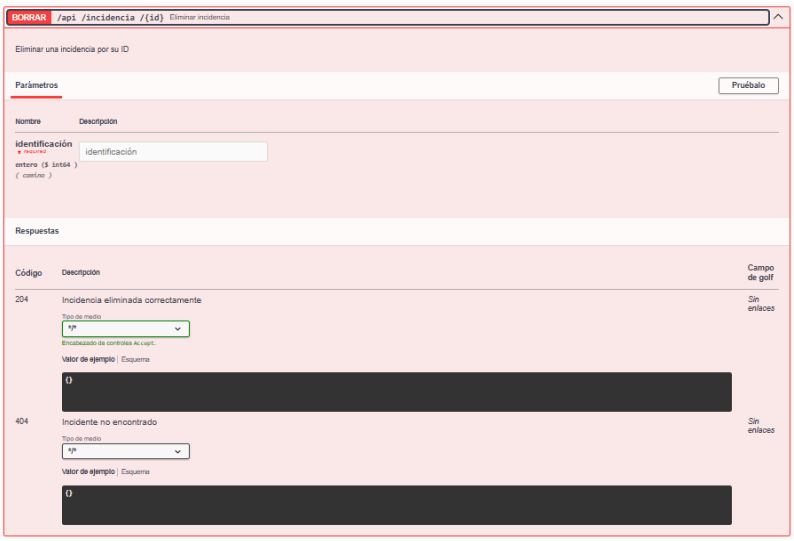
Ejecución de pruebas de integración del servicio de Mantenimiento:



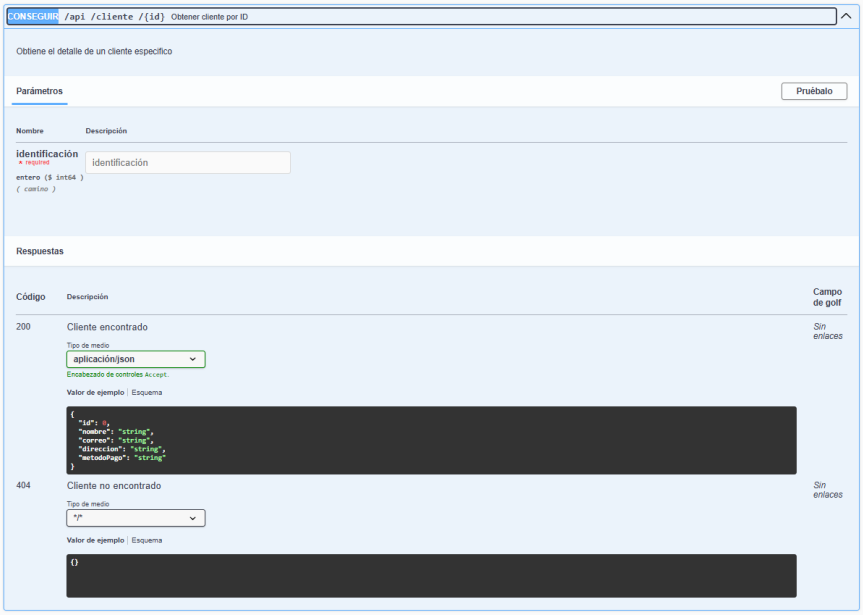


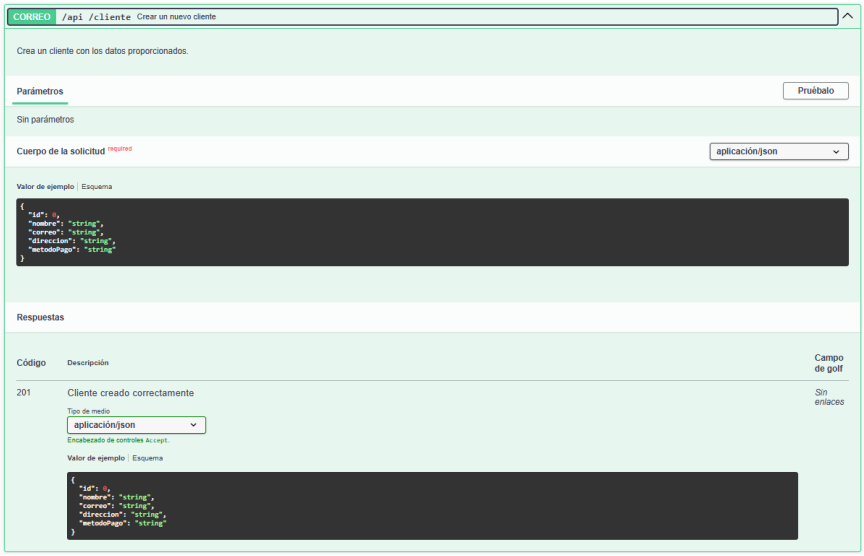

CORREO /api /incidencia Crear una nueva incidencia
Crea una incidencia con los datos proporcionados
Parámetros
Pruébalo
Sin parámetros
Cuerpo de la solicitud required
aplicación/json
Valor de ejemplo | Esquema
{
"id": 0,
"descripcion": "string",
"estado": "string",
"prioridad": "string"
}
Respuestas
Código
Descripción
Campo de golf
201
Incidencia creada correctamente
Sin enlaces
Tipo de medio
aplicación/json
Encabezado de controles Accept.
Valor de ejemplo | Esquema
{
"id": 0,
"descripcion": "string",
"estado": "string",
"prioridad": "string"
}

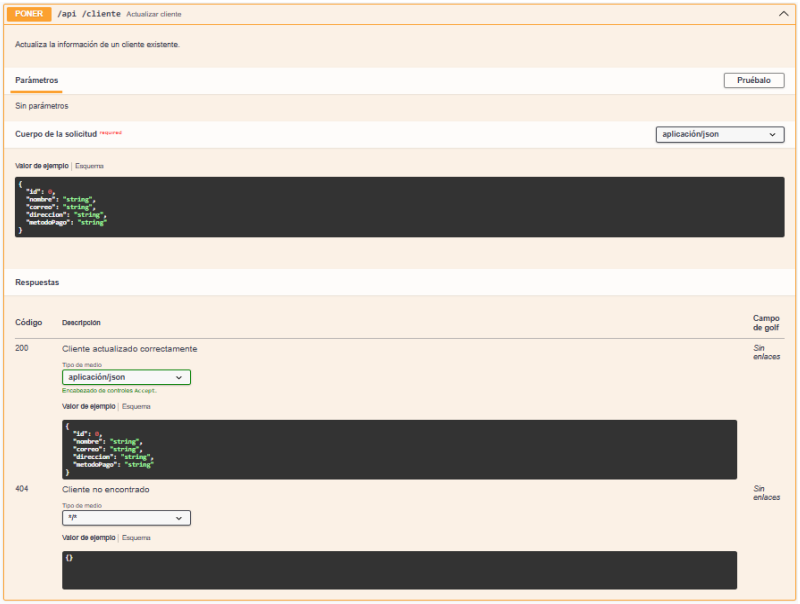

CONSEGUIR /api /incidencia /{id} Obtener incidencia por identificación
Obtiene el detalle de una incidencia especifica
Parámetros
Pruébalo
Nombre
Descripción
identificación
required
identificación
entero ($ int64)
(camino)
Respuestas
Código
Descripción
Campo de golf
200
Incidencia encontrada
Sin enlaces
Tipo de medio
aplicación/json
Encabezado de controles Accept.
Valor de ejemplo | Esquema
{
"id": 0,
"descripcion": "string",
"estado": "string",
"prioridad": "string"
}
404
Incidente no encontrado
Sin enlaces
Tipo de medio
/
Valor de ejemplo | Esquema
{}

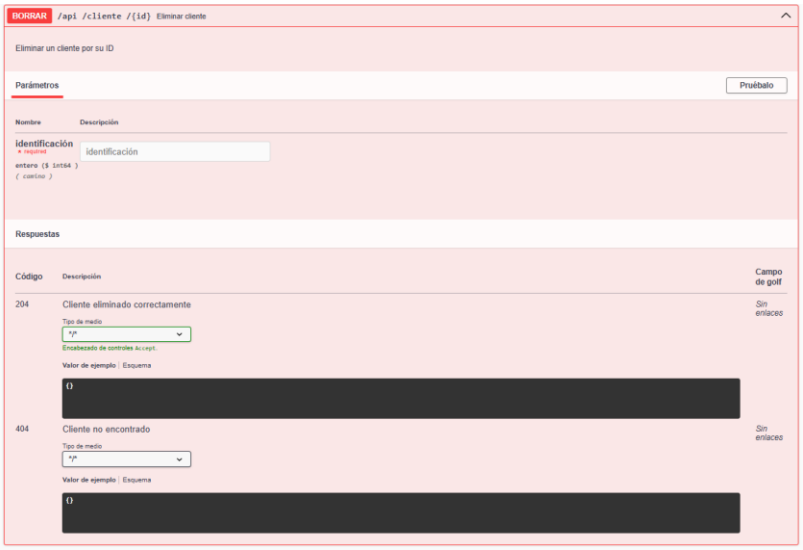

BORRAR /api /incidencia /{id} Eliminar incidencia
Eliminar una incidencia por su ID
Parámetros
Pruébalo
Nombre
Descripción
identificación
required
identificación
entero ($ int64)
(camino)
Respuestas
Código
Descripción
Campo de golf
204
Incidencia eliminada correctamente
Sin enlaces
Tipo de medio
/
Encabezado de controles Accept.
Valor de ejemplo | Esquema
{}
404
Incidente no encontrado
Sin enlaces
Tipo de medio
/
Valor de ejemplo | Esquema
{}

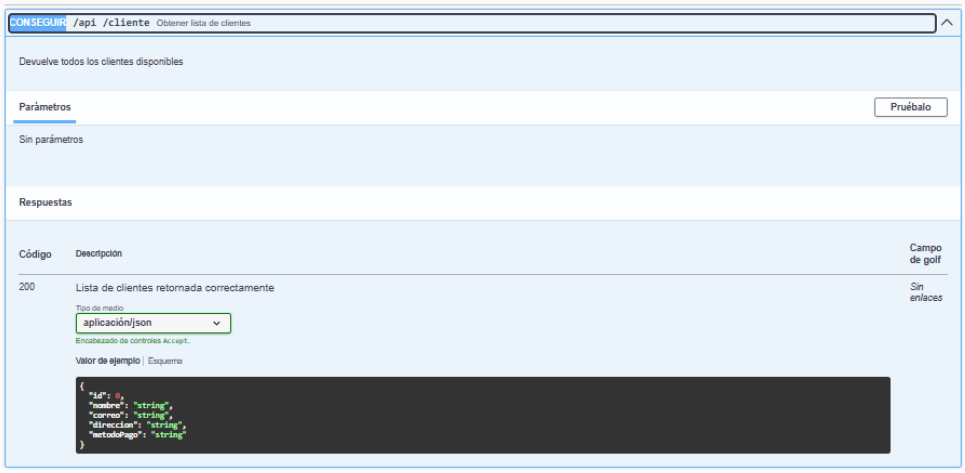
Ejecución de pruebas de integración del servicio de Clientes (en la web):



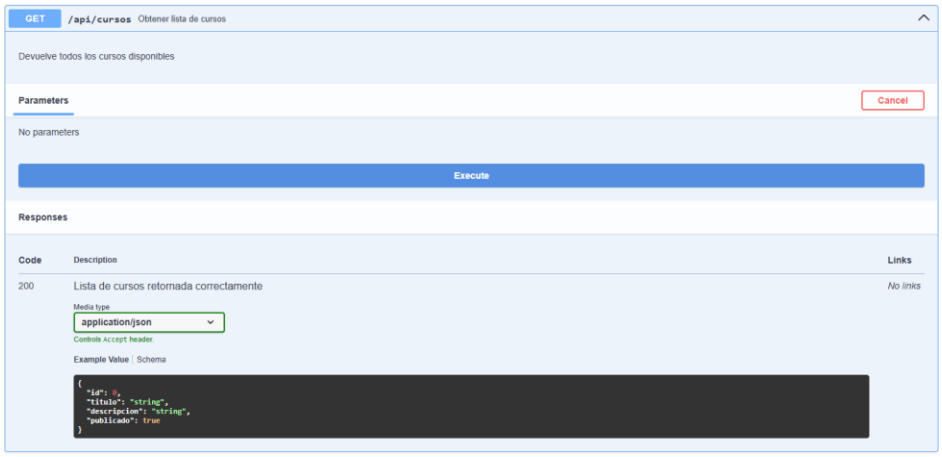


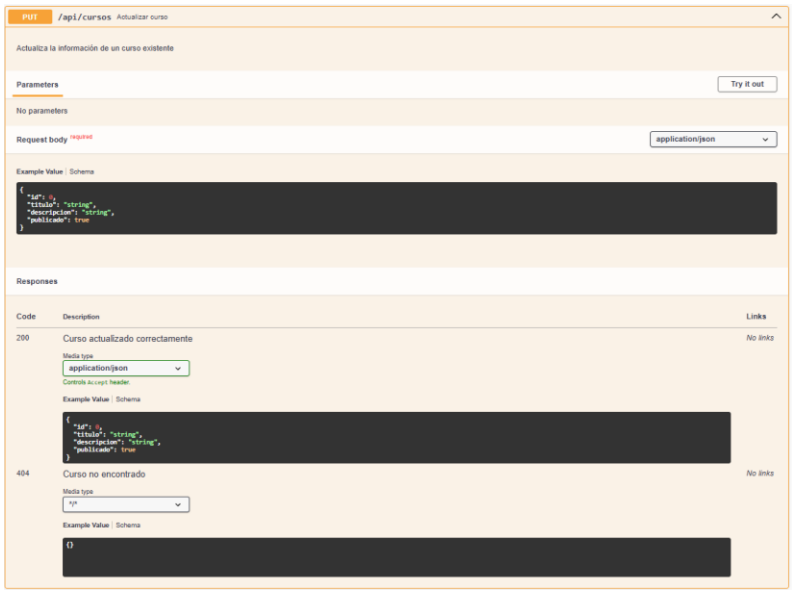

PONER /api /cliente Actualizar cliente
Actualiza la información de un cliente existente.
Parámetros
Pruébalo
Sin parámetros
Cuerpo de la solicitud
aplicación/json
Valor de ejemplo | Esquema
{
"id": 0,
"nombre": "string",
"correo": "string",
"direccion": "string",
"metodoPago": "string"
}
Respuestas
Código
Descripción
Campo de golf
200
Cliente actualizado correctamente
Sin enlaces
Tipo de medio
aplicación/json
Encabezado de controles Accept.
Valor de ejemplo | Esquema
404
Cliente no encontrado
Sin enlaces
Tipo de medio
/
Valor de ejemplo | Esquema
{}

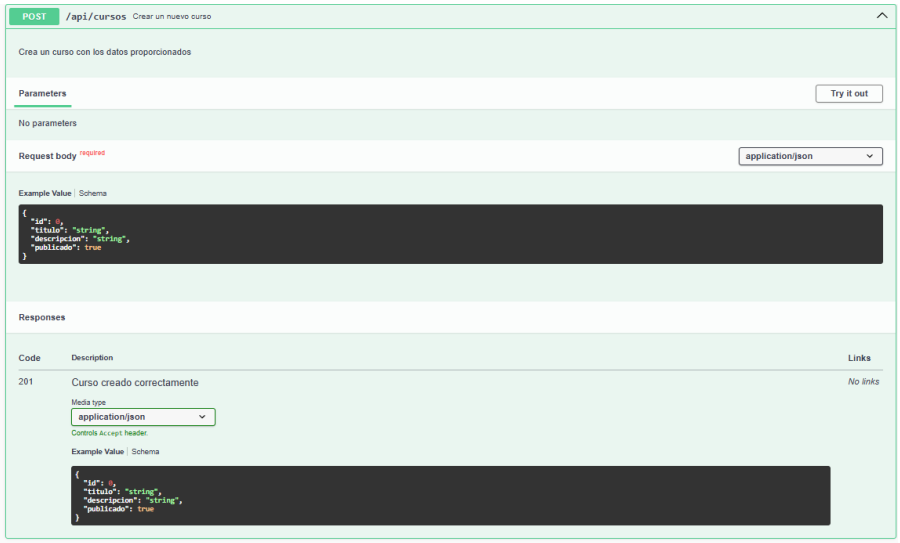

BORRAR /api /cliente /{id} Eliminar cliente
Eliminar un cliente por su ID
Parámetros
Pruébalo
Nombre
Descripción
identificación
* required
entero ($ int64)
(camino)
identificación
Respuestas
Código
Descripción
Campo de golf
204
Cliente eliminado correctamente
Sin enlaces
Tipo de medio
/
Encabezado de controles Accept.
Valor de ejemplo | Esquema
{}
404
Cliente no encontrado
Sin enlaces
Tipo de medio
/
Valor de ejemplo | Esquema
{}

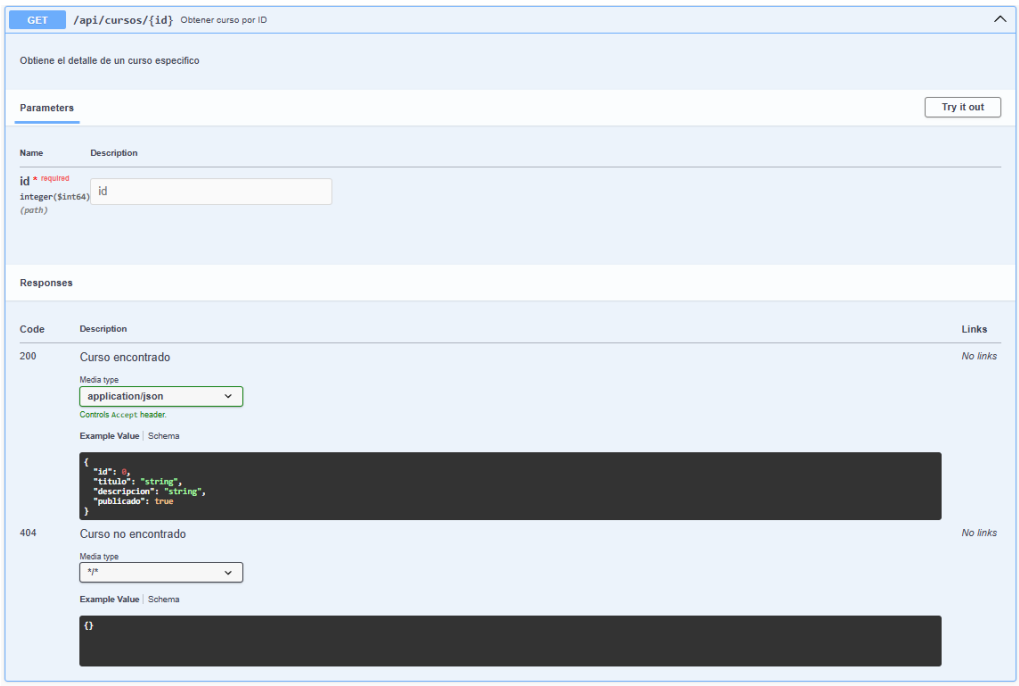

CONSEGUIR /api /cliente Obtener lista de clientes
Devuelve todos los clientes disponibles
Parámetros
Pruébalo
Sin parámetros
Respuestas
Código
Descripción
Campo de golf
200
Lista de clientes retornada correctamente
Sin enlaces
Tipo de medio
aplicación/json
Encabezado de controles Accept.
Valor de ejemplo | Esquema
{
"id": 0,
"nombre": "string",
"correo": "string",
"direccion": "string",
"metodoPago": "string"
}

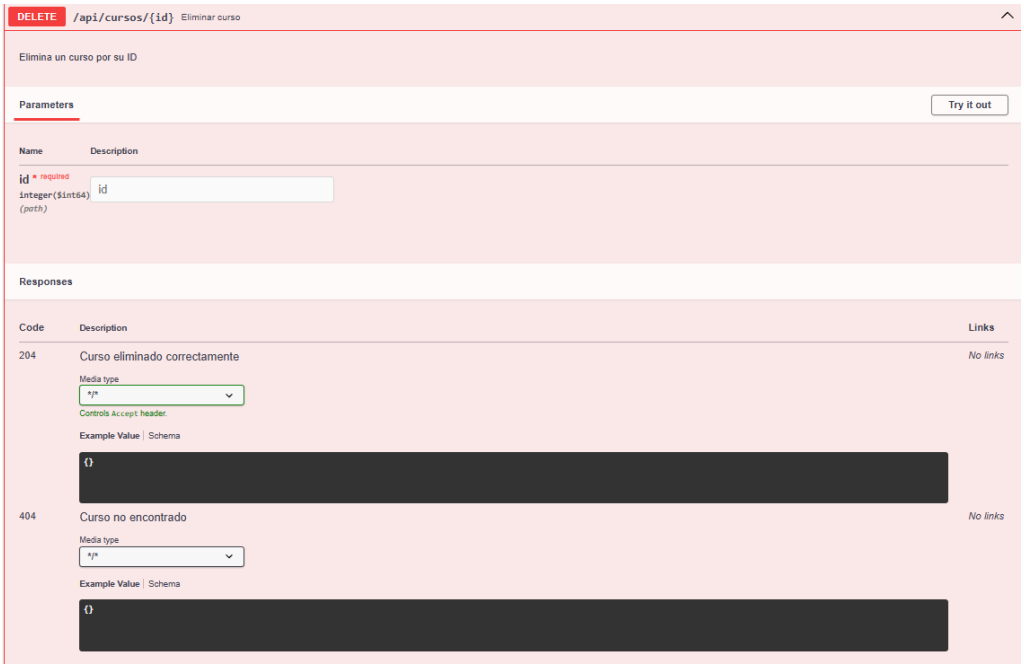
Ejecución de pruebas de integración del servicio de Cursos:



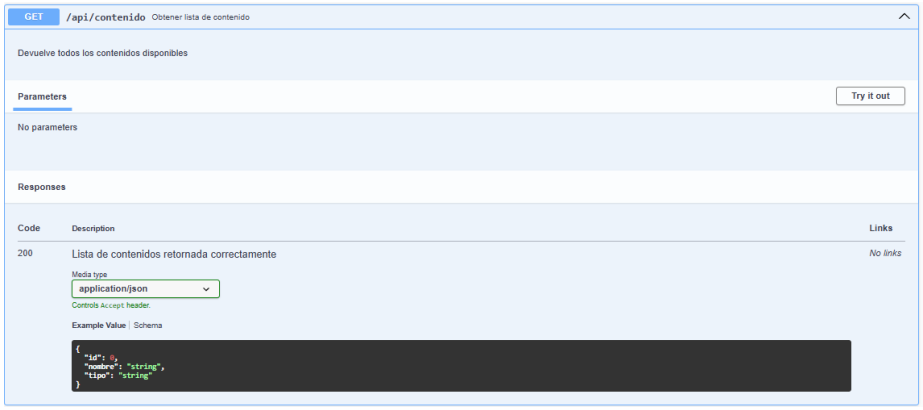


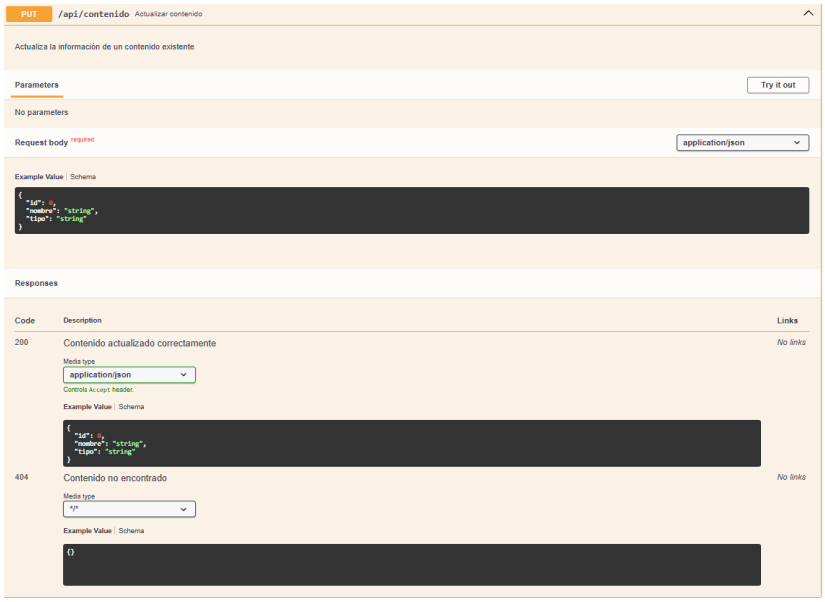


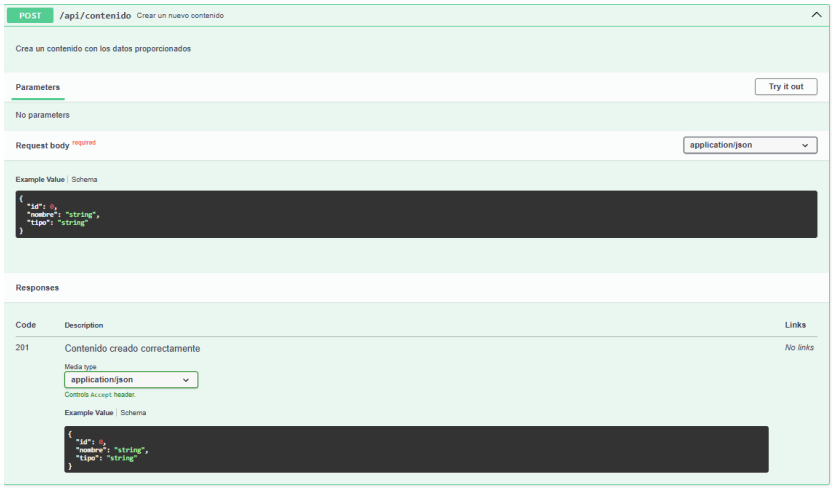

GET
/api/cursos/{id}
Obtener curso por ID
Obtiene el detalle de un curso especifico
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
Links
200
Curso encontrado
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"titulo": "string",
"descripcion": "string",
"publicado": true
}
404
Curso no encontrado
No links
Media type
/
Example Value | Schema
{}

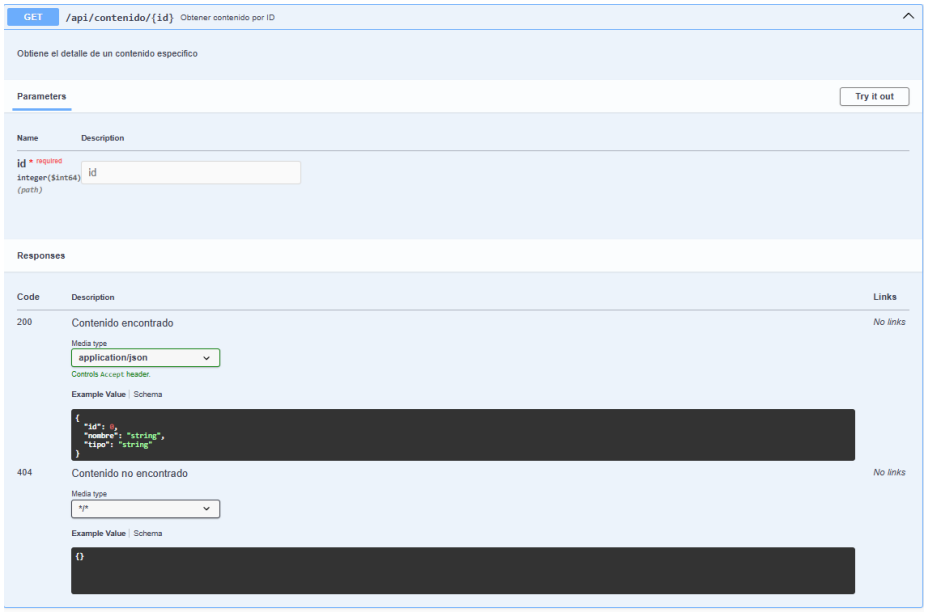

DELETE
/api/cursos/{id}
Eliminar curso
Elimina un curso por su ID
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
Links
204
Curso eliminado correctamente
No links
Media type
/
Controls Accept header.
Example Value | Schema
{}
404
Curso no encontrado
No links
Media type
/
Example Value | Schema
{}

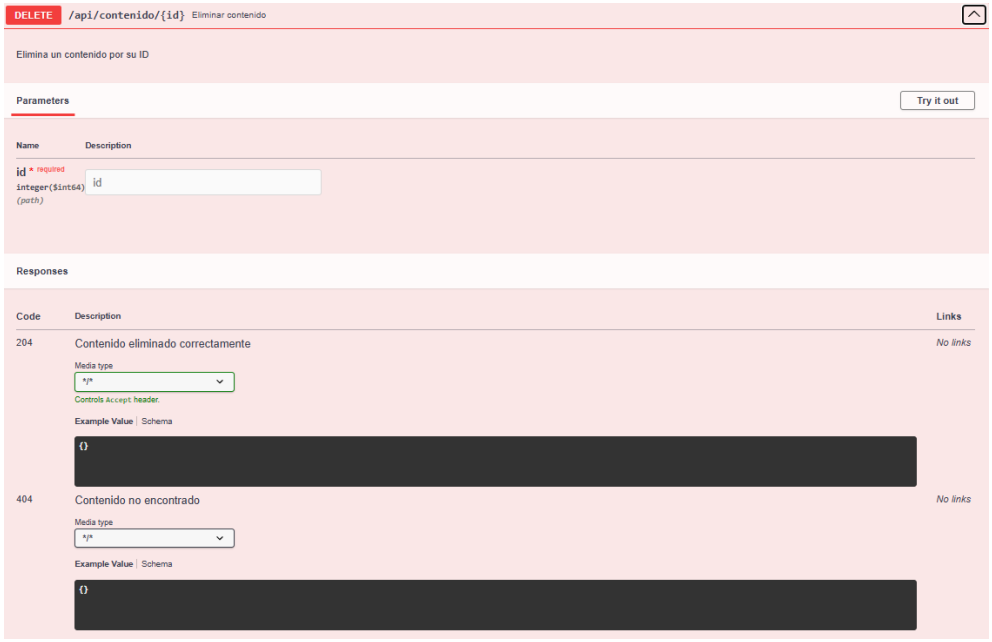
Ejecución de pruebas de integración del servicio de Contenido:








GET
/api/contenido/{id}
Obtener contenido por ID
Obtiene el detalle de un contenido especifico
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
Links
200
Contenido encontrado
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"nombre": "string",
"tipo": "string"
}
404
Contenido no encontrado
No links
Media type
/
Example Value | Schema
{}


DELETE
/api/contenido/{id}
Eliminar contenido
Elimina un contenido por su ID
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
Links
204
Contenido eliminado correctamente
No links
Media type
/
Controls Accept header.
Example Value | Schema
{}
404
Contenido no encontrado
No links
Media type
/
Example Value | Schema
{}

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-8IG1TE2 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/
ontrollers/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 2.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-8IG1TE2 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/backend/springboot_backend/restcontrolle
s/ContenidoRestControllersTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/backend/springboot_backend/restcontrolle
s/CursoRestControllersTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/backend/springboot_backend/services/Cont
nidoServiceImplTest.java
 create mode 100644 springboot-backend/src/test/java/com/proyecto/springboot/backend/springboot_backend/services/Curs
ServiceImplTest.java

maxim@DESKTOP-8IG1TE2 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-37USUO6 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-37USUO6 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-37USUO6 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
```

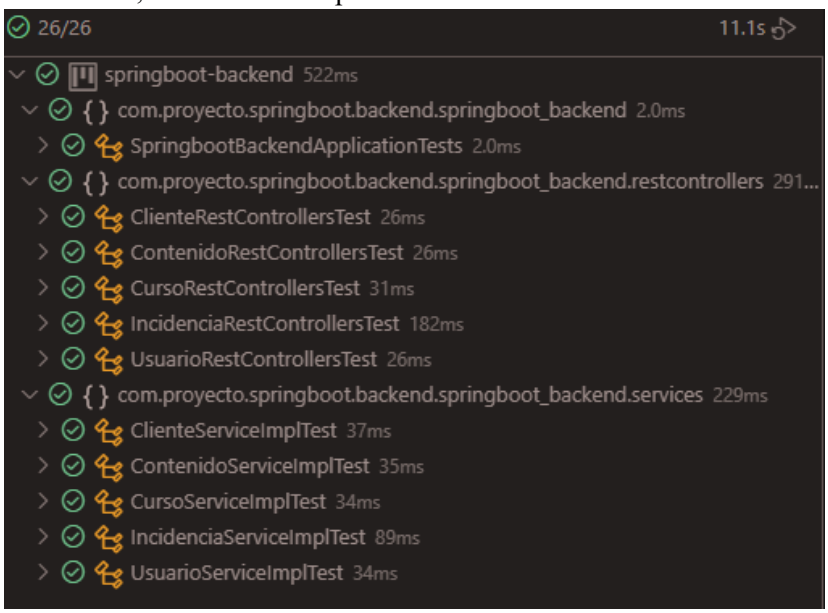
```
renax@DESKTOP-37USUO6 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main)
$ git merge mac main
```

```
renax@DESKTOP-37USUO6 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main|MERGING)
$ git commit -m "Merge mac con main"
[main 408f2a1] Merge mac con main

renax@DESKTOP-37USUO6 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-37USUO6 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-37USUO6 MINGW64 ~/OneDrive/Documentos/ProyectoSemestral (main|MERGING)
$ git commit -m "Merge diego con main"
[main 2f9e3ab] Merge diego con main

renax@DESKTOP-37USUO6 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:



Readme:
https://github.com/RenatoValenzuela262/Exp3_Hernandez_Valenzuela_Zamora/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.