



UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

Ingeniería de Software
Aplicaciones Distribuidas

MICROSERVICIOS Y

ANGULAR 17

GRUPO 6

NRC: 14930

DIAZ ADRIANA – MASACELA BRANDON – PILA
JOHANNA – PILATAXI DANIELA



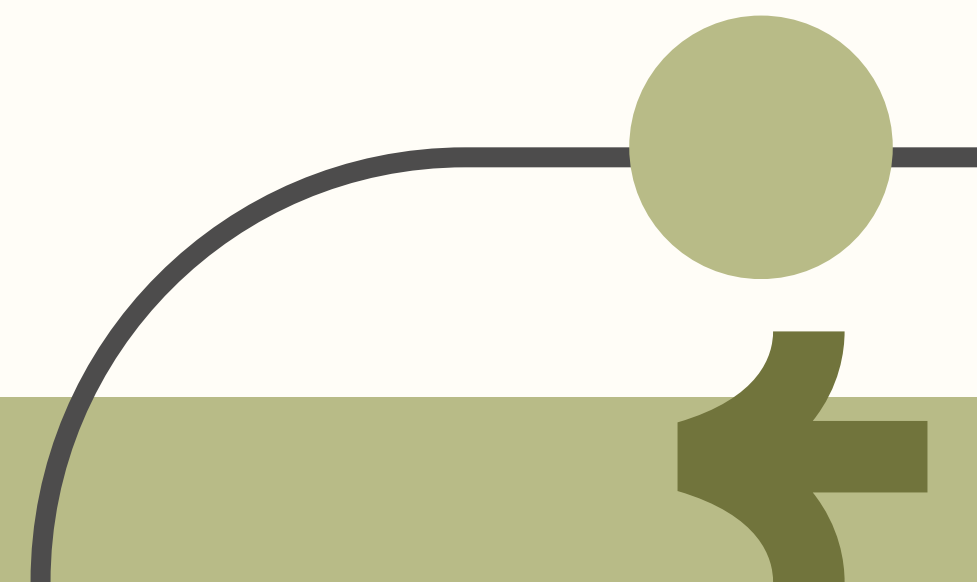
OBJETIVO GENERAL

Desarrollar un sistema de matrículas y gestión de estudiantes utilizando microservicios para el backend y Angular para el frontend, asegurando funcionalidad completa y eficiente a través de un proceso documentado y probado.



INTRODUCCIÓN

Al combinar microservicios y Angular nos permite ofrecer una arquitectura flexible y escalable. Los microservicios permiten dividir la aplicación en pequeños servicios independientes que pueden desarrollarse, desplegarse y gestionarse de forma autónoma. Esto facilita el mantenimiento y la evolución del sistema, permitiendo añadir nuevas funcionalidades sin afectar a los servicios existentes. Por otro lado, Angular proporciona una plataforma robusta para el desarrollo del frontend, ofreciendo una experiencia de usuario rica y dinámica.



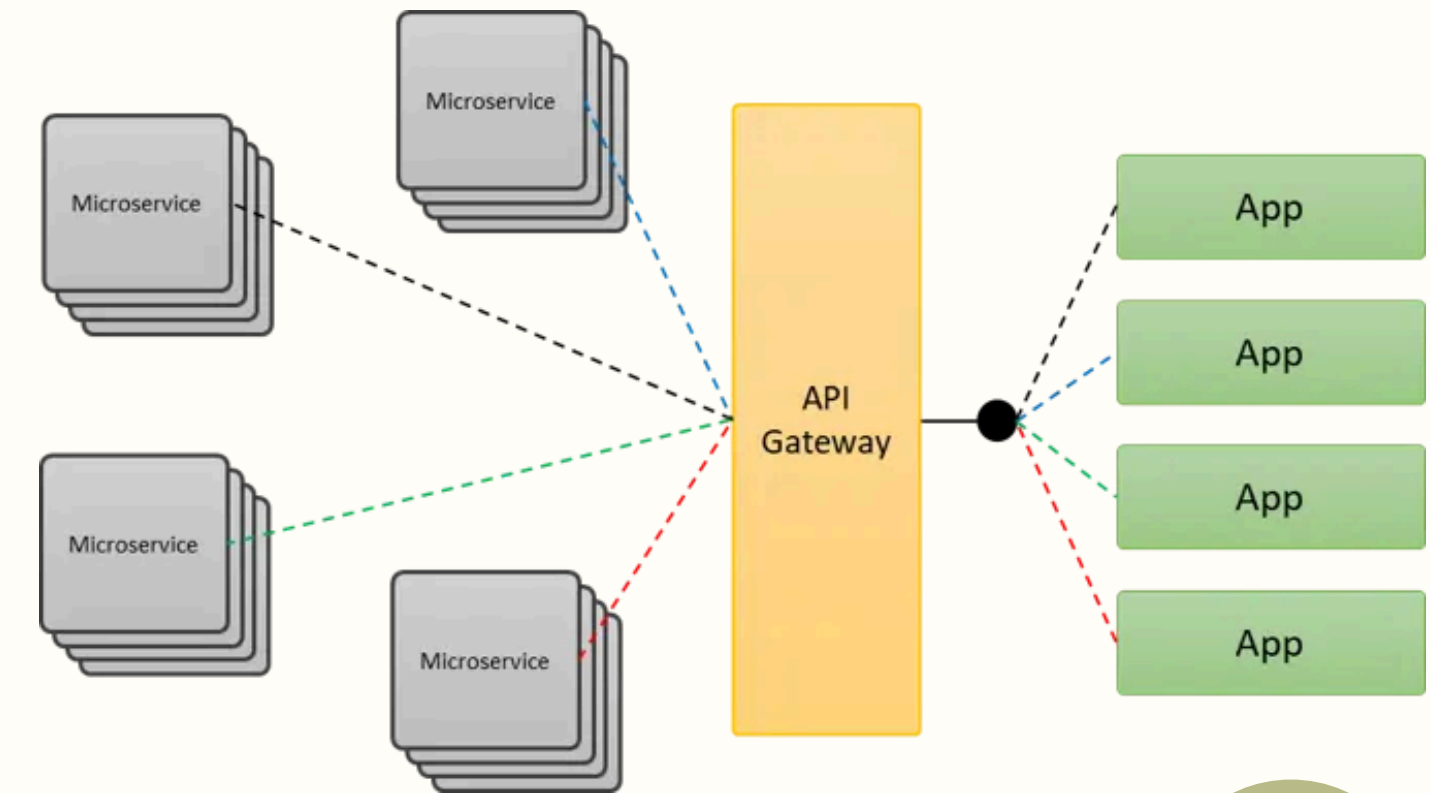
INTRODUCCIÓN

Microservicios

Son módulos ligeros que pueden servir como componentes básicos de las aplicaciones complejas basadas en la nube.

Angular

Es un framework de JavaScript de código abierto escrito en TypeScript. Su objetivo principal es desarrollar aplicaciones de una sola página.





01

Desarrollo
FrontEnd



Sistema de Gestión de Cursos

Bienvenido al Sistema de Gestión de Cursos

 Usuarios

 Cursos

Inscripciones

© 2024 Todos los derechos reservados.

USUARIOS

Mis Usuarios

[Inicio](#) [+ Agregar Usuario](#)

ID	Nombre	Email	Operaciones
17	prueba final	final@gmail.com	✎ Editar 🗑 Eliminar
18	Momo Aquiles 1	momo.aquiles1@gmail.com	✎ Editar 🗑 Eliminar

Items per page: 5 11 - 12 of 12 [|<](#) [<](#) [>](#) [>|](#)

Registrar Usuario

Nombre

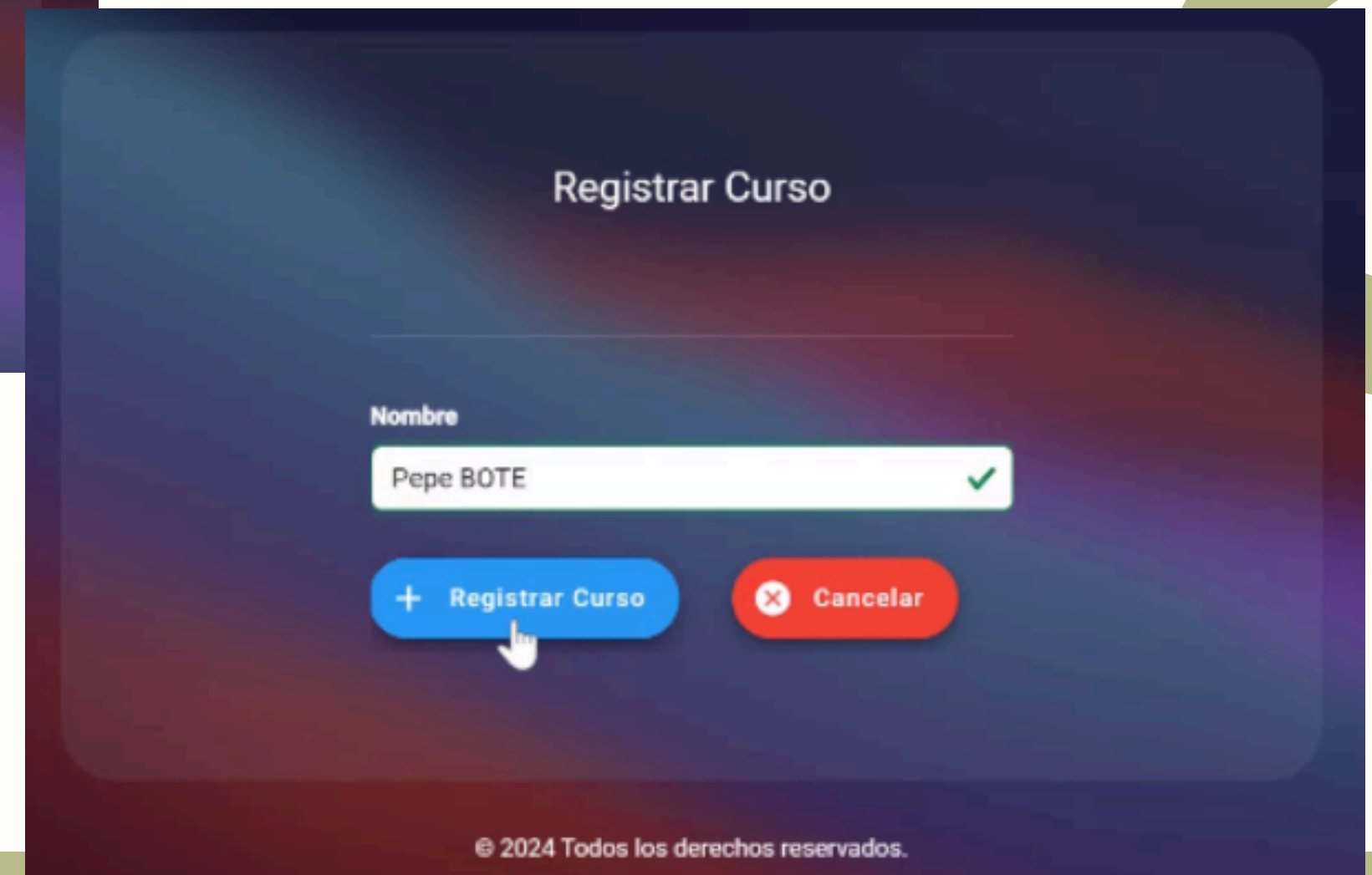
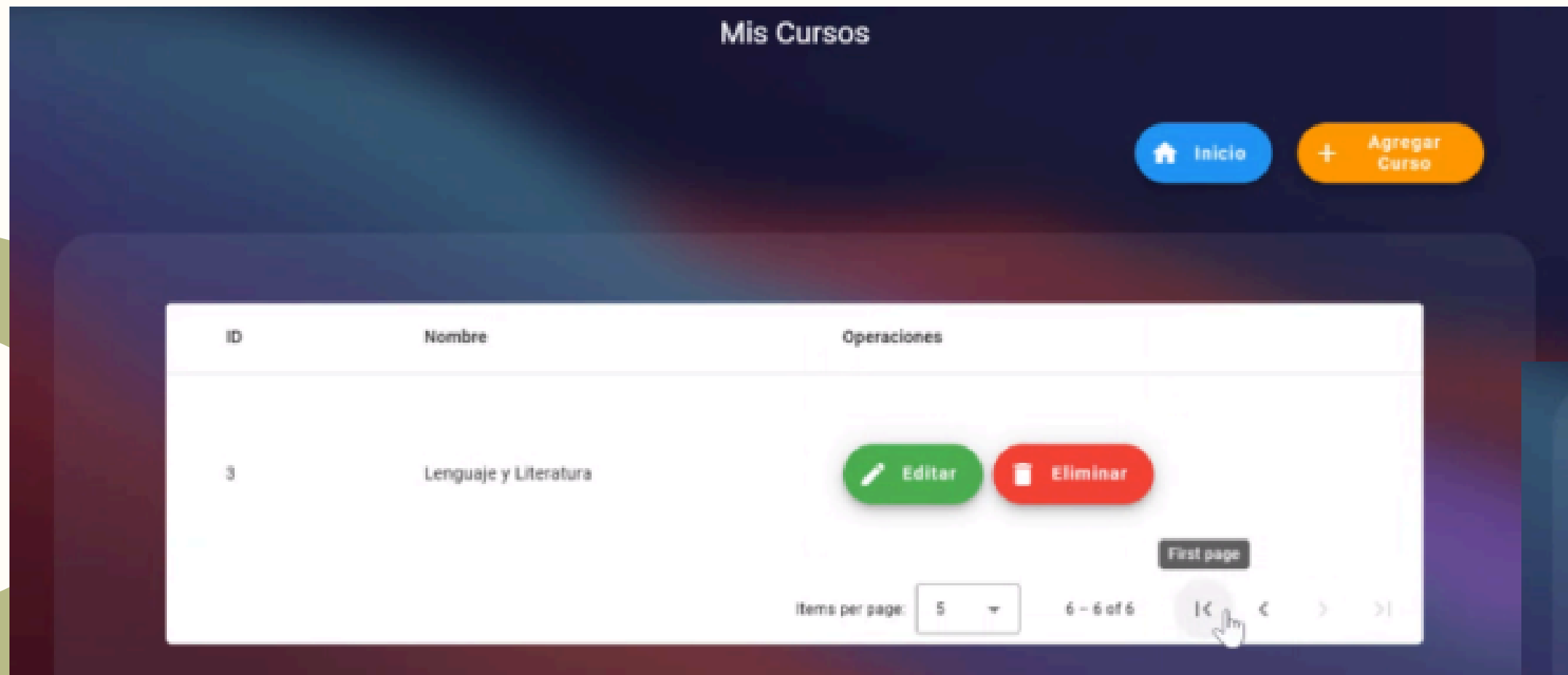
Correo electrónico

Contraseña

Repetir Contraseña

[+ Registrar Usuario](#) [✕ Cancelar](#)

CURSOS



INSCRIPCIONES

Detalle Cursos

Inicio

Ciencias Naturales

+ Asignar Usuario

ID Usuario	Nombre Usuario	Correo Electrónico
6	Brandon Masacela	brandonmasacela24@gmail.com
12	Myke Towers	myke.towers@gmail.com
17	prueba final	final@gmail.com

Items per page: 2 0 of 0

Inicio

Ciencias Naturales

+ Asignar Usuario

Inscribir a Ciencias Naturales

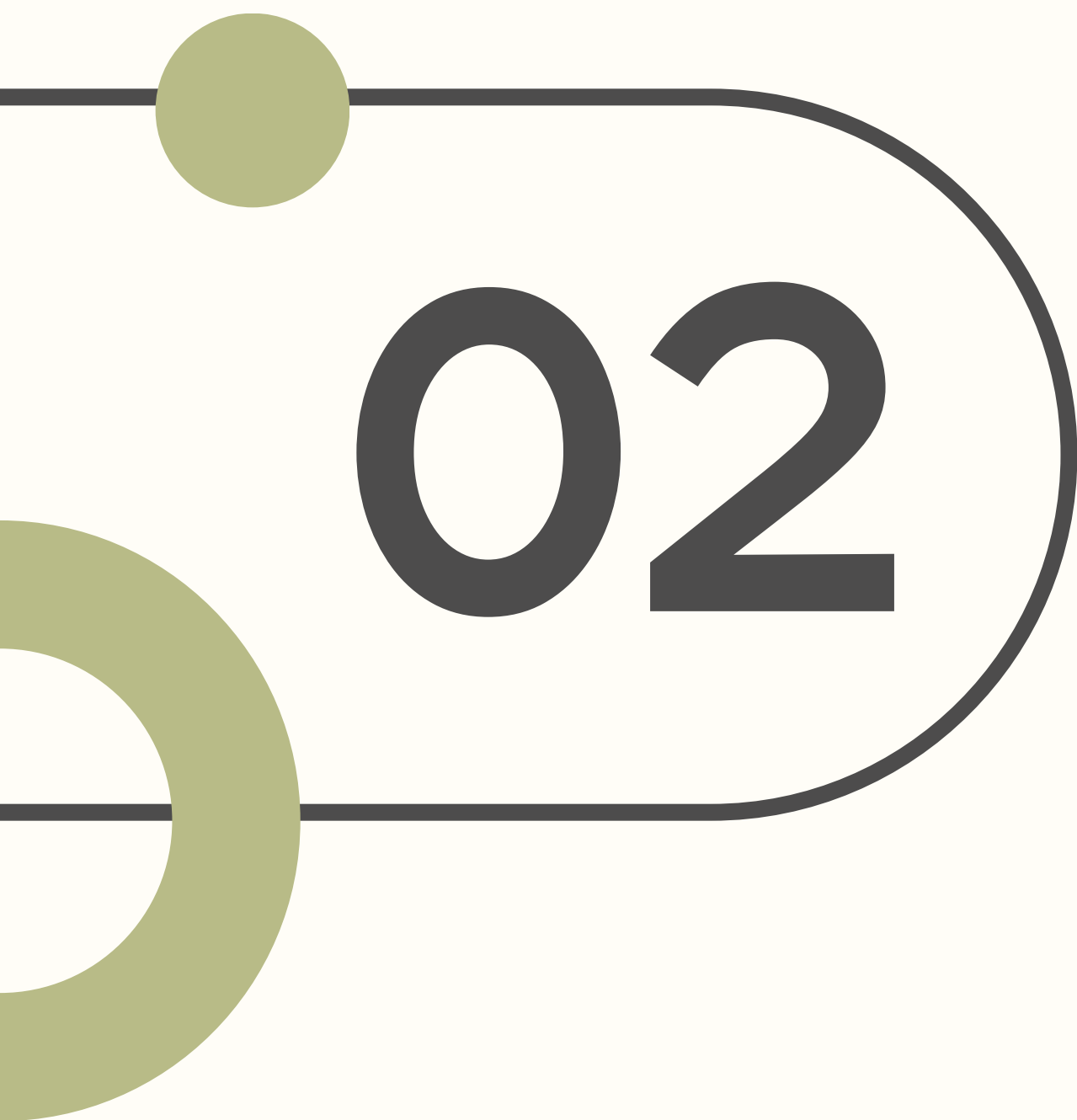
ID de Curso*

2

ID de Usuario*

Cancelar

+ Inscribir

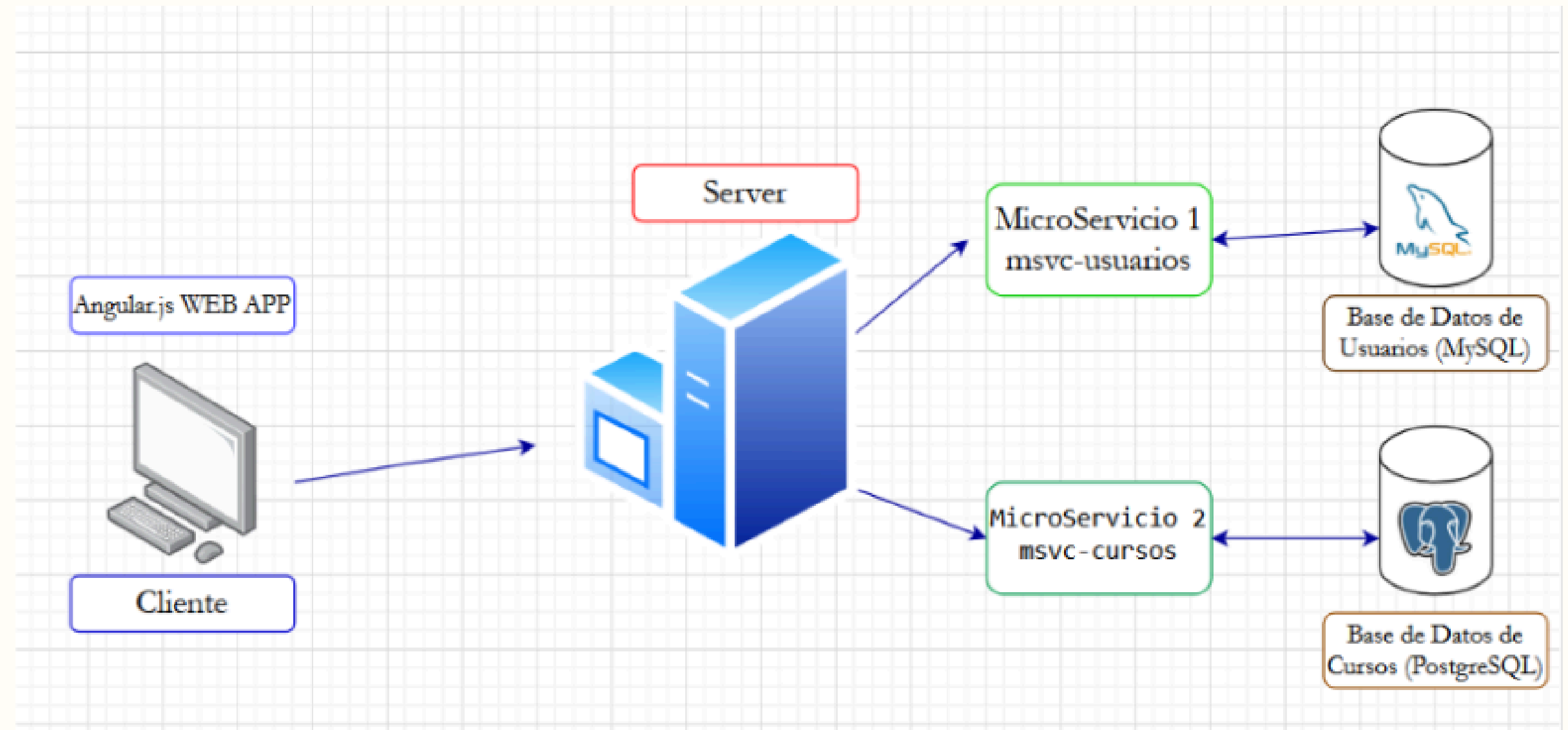


02

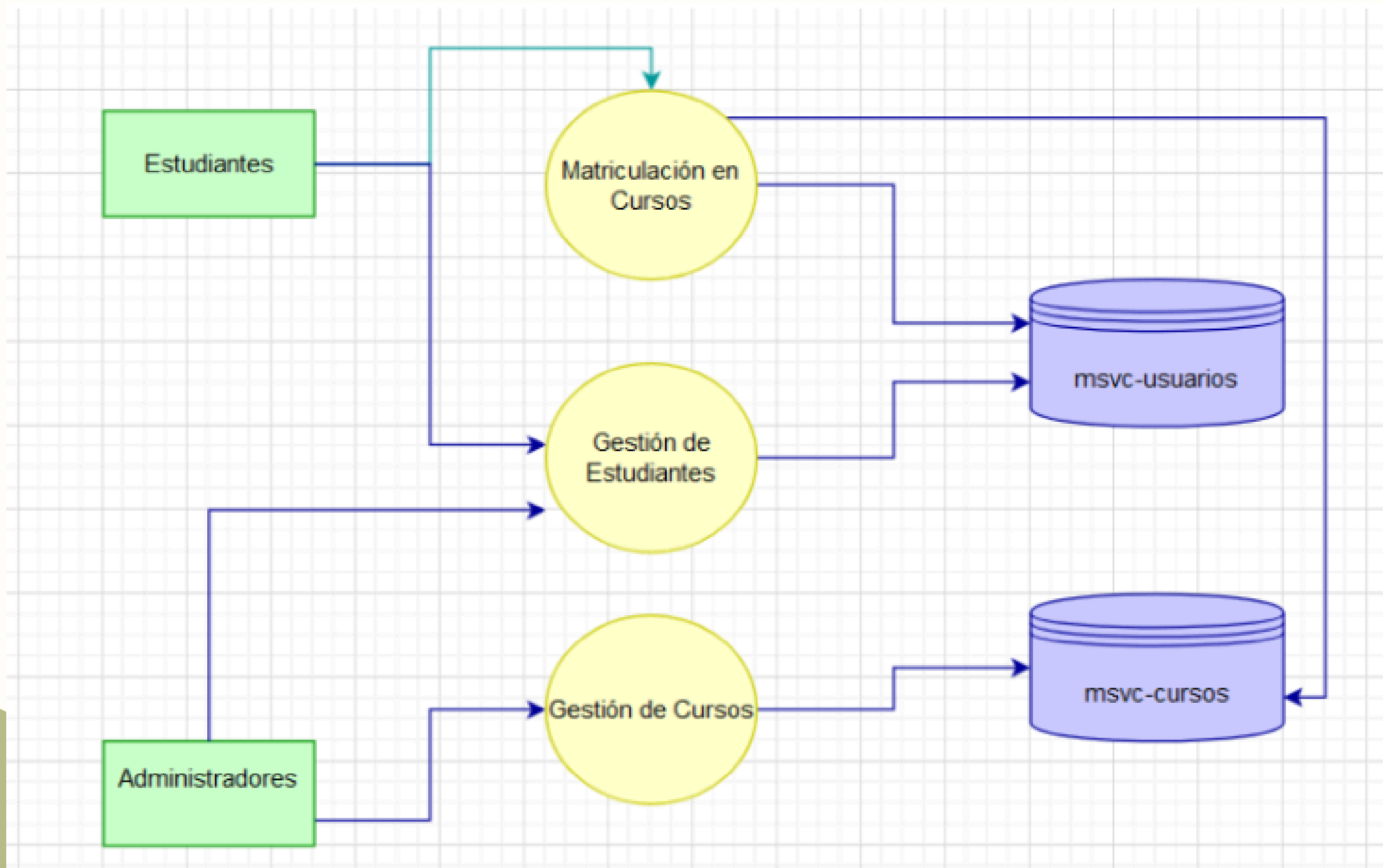
Diagrams

Arquitectura

Representación visual de los distintos componentes que forman un sistema y muestran cómo se comunican e interactúan entre sí.



Flujo de Datos



Traza el flujo de la información para cualquier proceso o sistema




03

Pruebas

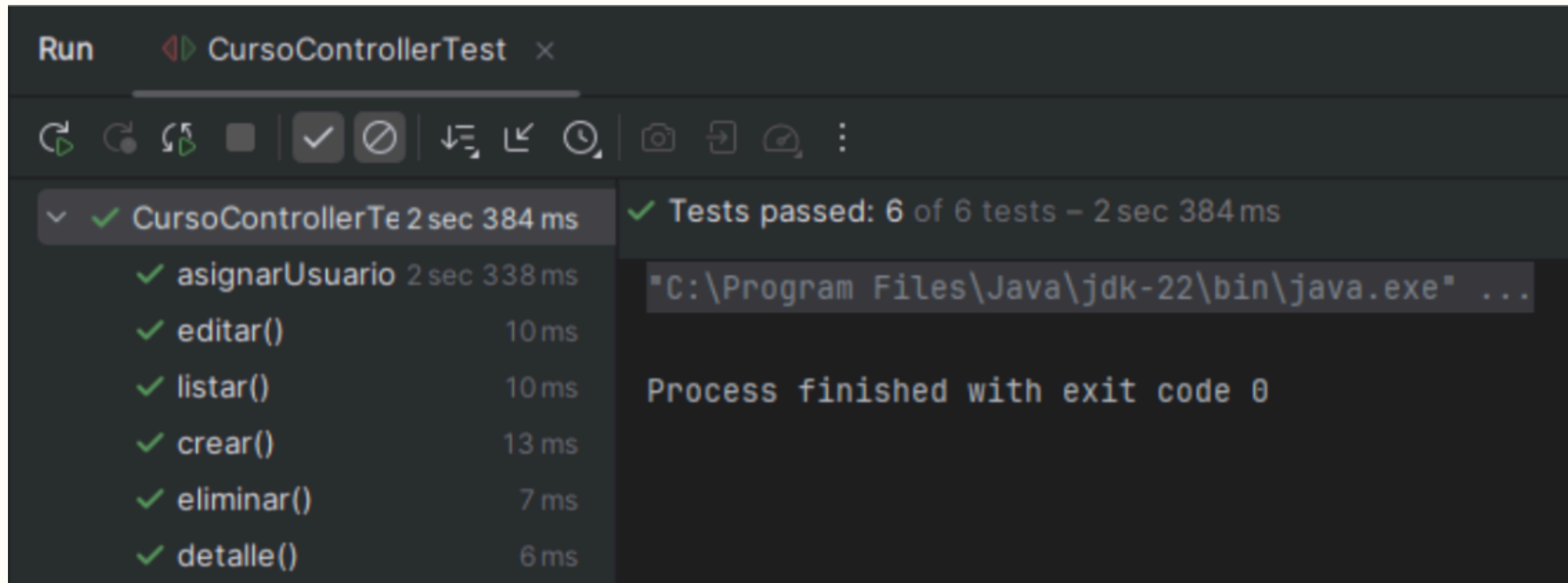


Pruebas Unitarias y de Integración para los microservicios backend JUnit

- Las pruebas unitarias, realizadas a nivel de componentes individuales, permiten verificar el comportamiento esperado de cada unidad de código.
 - Las pruebas de integración se centran en la interacción entre los distintos microservicios y otros sistemas, asegurando que los componentes se integren correctamente y cumplan con los requisitos funcionales y de rendimiento.
- 

→ Unitarias

- En esta prueba unitaria, se evalúan las funcionalidades del controlador `CursoController` mediante el uso de JUnit y Mockito para simular el comportamiento del servicio `CursoService`



The screenshot shows the 'Run' window of an IDE, specifically for the test class 'CursoControllerTest'. The window is divided into two main sections. The left section lists the test results for individual methods, all of which passed successfully. The right section provides a summary of the test run, indicating that all 6 tests passed within a total time of 2 seconds and 384 milliseconds. The command used to run the tests is also visible, showing the Java executable path.

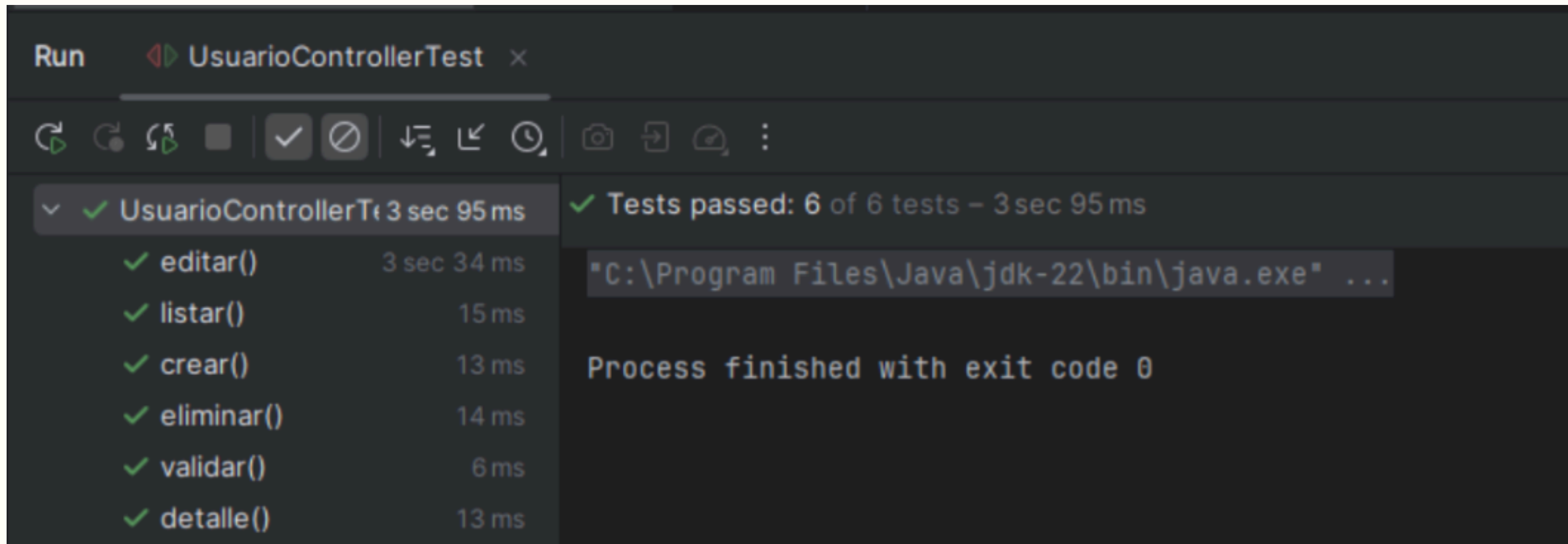
Test Method	Duration
✓ asignarUsuario	2 sec 338 ms
✓ editar()	10 ms
✓ listar()	10 ms
✓ crear()	13 ms
✓ eliminar()	7 ms
✓ detalle()	6 ms

✓ Tests passed: 6 of 6 tests – 2 sec 384 ms

"C:\Program Files\Java\jdk-22\bin\java.exe" ...

Process finished with exit code 0

- En esta prueba unitaria, se evalúan las operaciones del controlador `UsuarioController` utilizando JUnit y Mockito para simular el comportamiento del servicio `UsuarioService` y el objeto `BindingResult`



The screenshot shows the Run console of an IDE. At the top, the tab is labeled 'Run' and 'UsuarioControllerTest'. Below the tab is a toolbar with various icons. The main area is divided into two panels. The left panel shows a list of test methods, all of which passed successfully, indicated by green checkmarks. The right panel shows the overall test result, which is also a success, and the command used to run the tests.

Test Method	Duration
✓ editar()	3 sec 34 ms
✓ listar()	15 ms
✓ crear()	13 ms
✓ eliminar()	14 ms
✓ validar()	6 ms
✓ detalle()	13 ms

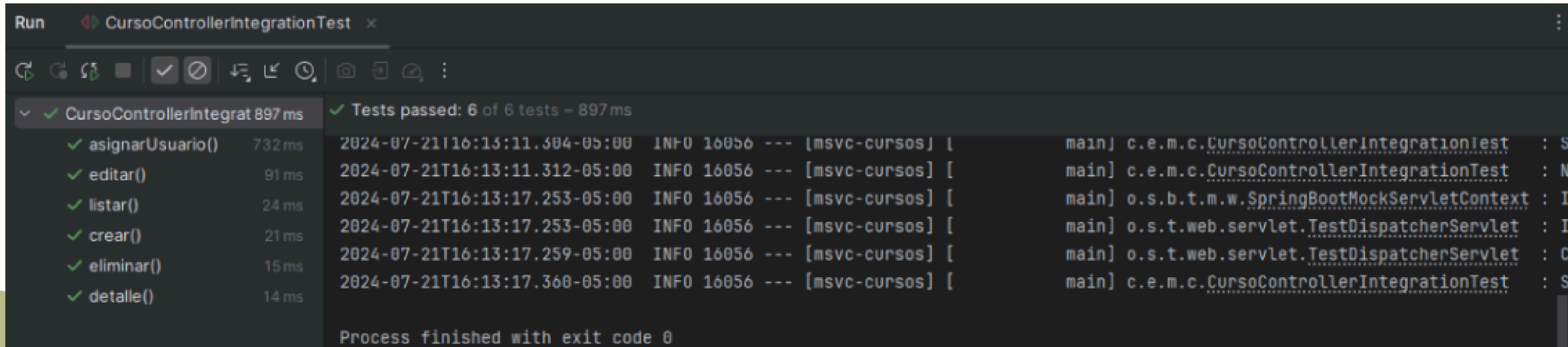
✓ Tests passed: 6 of 6 tests – 3 sec 95 ms

```
"C:\Program Files\Java\jdk-22\bin\java.exe" ...
```

Process finished with exit code 0

→ Integración

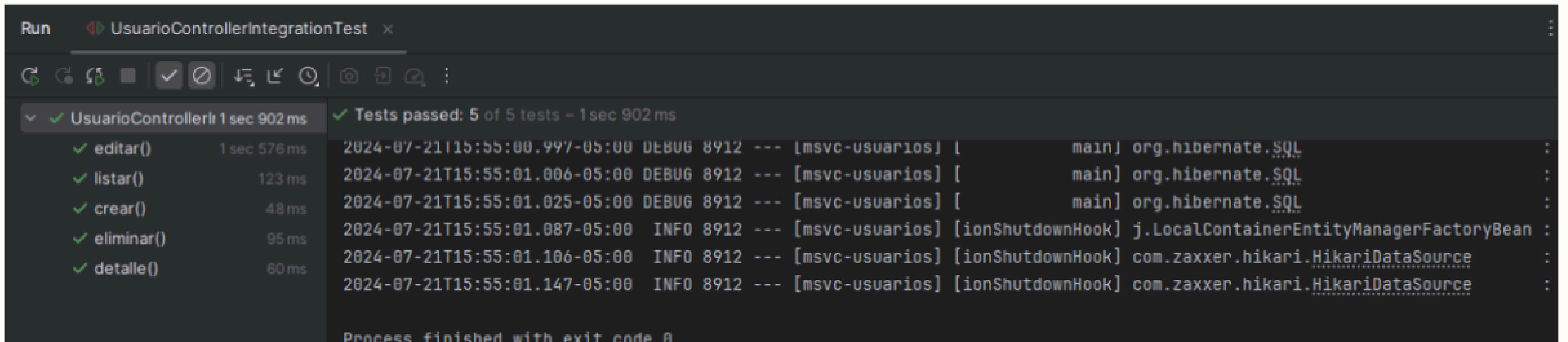
Se valida cómo el CursoController interactúa con CursoService y realiza operaciones HTTP correctamente, utilizando el enfoque mocks (@MockBean) para evitar depender de la implementación real del servicio, controlando el comportamiento del mismo y verificar cómo el controlador interactúa con él,



The screenshot shows the Run console of an IDE with the following content:

```
Run  CursoControllerIntegrationTest x
[Icons: Run, Debug, Test, etc.]
✓ CursoControllerIntegrat 897 ms  ✓ Tests passed: 6 of 6 tests - 897 ms
  ✓ asignarUsuario() 732 ms  2024-07-21T16:13:11.304-05:00 INFO 16056 --- [msvc-cursos] [main] c.e.m.c.CursoControllerIntegrationTest : S
  ✓ editar() 91 ms  2024-07-21T16:13:11.312-05:00 INFO 16056 --- [msvc-cursos] [main] c.e.m.c.CursoControllerIntegrationTest : M
  ✓ listar() 24 ms  2024-07-21T16:13:17.253-05:00 INFO 16056 --- [msvc-cursos] [main] o.s.b.t.m.w.SpringBootMockServletContext : I
  ✓ crear() 21 ms  2024-07-21T16:13:17.253-05:00 INFO 16056 --- [msvc-cursos] [main] o.s.t.web.servlet.TestDispatcherServlet : I
  ✓ eliminar() 15 ms  2024-07-21T16:13:17.259-05:00 INFO 16056 --- [msvc-cursos] [main] o.s.t.web.servlet.TestDispatcherServlet : C
  ✓ detalle() 14 ms  2024-07-21T16:13:17.360-05:00 INFO 16056 --- [msvc-cursos] [main] c.e.m.c.CursoControllerIntegrationTest : S
Process finished with exit code 0
```

- En este caso Validamos la integración completa con la base de datos y verificamos la funcionalidad completa del sistema, utilizando el enfoque con una base de datos en memoria (@SpringBootTest)




The screenshot shows the Run console of an IDE with the following content:

```
Run  UsuarioControllerIntegrationTest x
[Icons: Run, Debug, Test, Stop, Checkmark, Cancel, Expand, Copy, Paste, Refresh, Search, Zoom In, Zoom Out, Fullscreen, Help]
✓ UsuarioControllerIntegrationTest 1 sec 902 ms  ✓ Tests passed: 5 of 5 tests - 1 sec 902 ms
  ✓ editar() 1 sec 576 ms  2024-07-21T15:55:00.997-05:00 DEBUG 8912 --- [msvc-usuarios] [main] org.hibernate.SQL :
  ✓ listar() 123 ms  2024-07-21T15:55:01.006-05:00 DEBUG 8912 --- [msvc-usuarios] [main] org.hibernate.SQL :
  ✓ crear() 48 ms  2024-07-21T15:55:01.025-05:00 DEBUG 8912 --- [msvc-usuarios] [main] org.hibernate.SQL :
  ✓ eliminar() 95 ms  2024-07-21T15:55:01.087-05:00 INFO 8912 --- [msvc-usuarios] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean :
  ✓ detalle() 60 ms  2024-07-21T15:55:01.106-05:00 INFO 8912 --- [msvc-usuarios] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource :
  2024-07-21T15:55:01.147-05:00 INFO 8912 --- [msvc-usuarios] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource :
  Process finished with exit code 0
```



Pruebas unitarias para los componentes y servicios de Angular con Jasmine y Karma

- **Jasmine es un marco de pruebas para JavaScript que facilita la escritura de tests legibles y expresivos**
 - **Karma es un ejecutor de pruebas que permite correr estos tests en varios navegadores y plataformas de manera automatizada.**
- 

- Para realizar las pruebas de componentes y servicios es necesario considerar archivos de prueba (*.spec.ts) para cada componente y servicio.

```
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component';

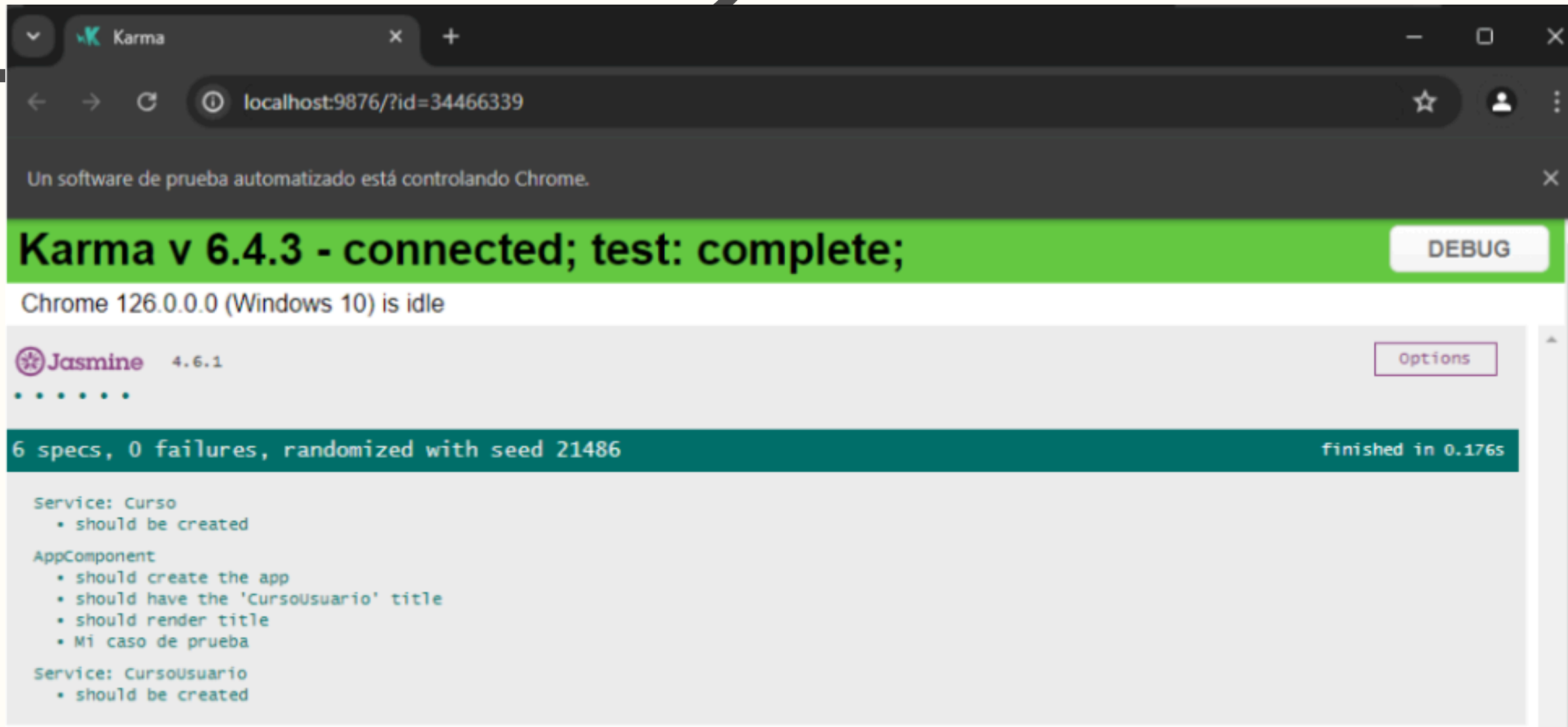
describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [AppComponent],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have the 'CursoUsuario' title`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('CursoUsuario');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('h1')?.textContent).toContain('Hello,
CursoUsuario');
  });

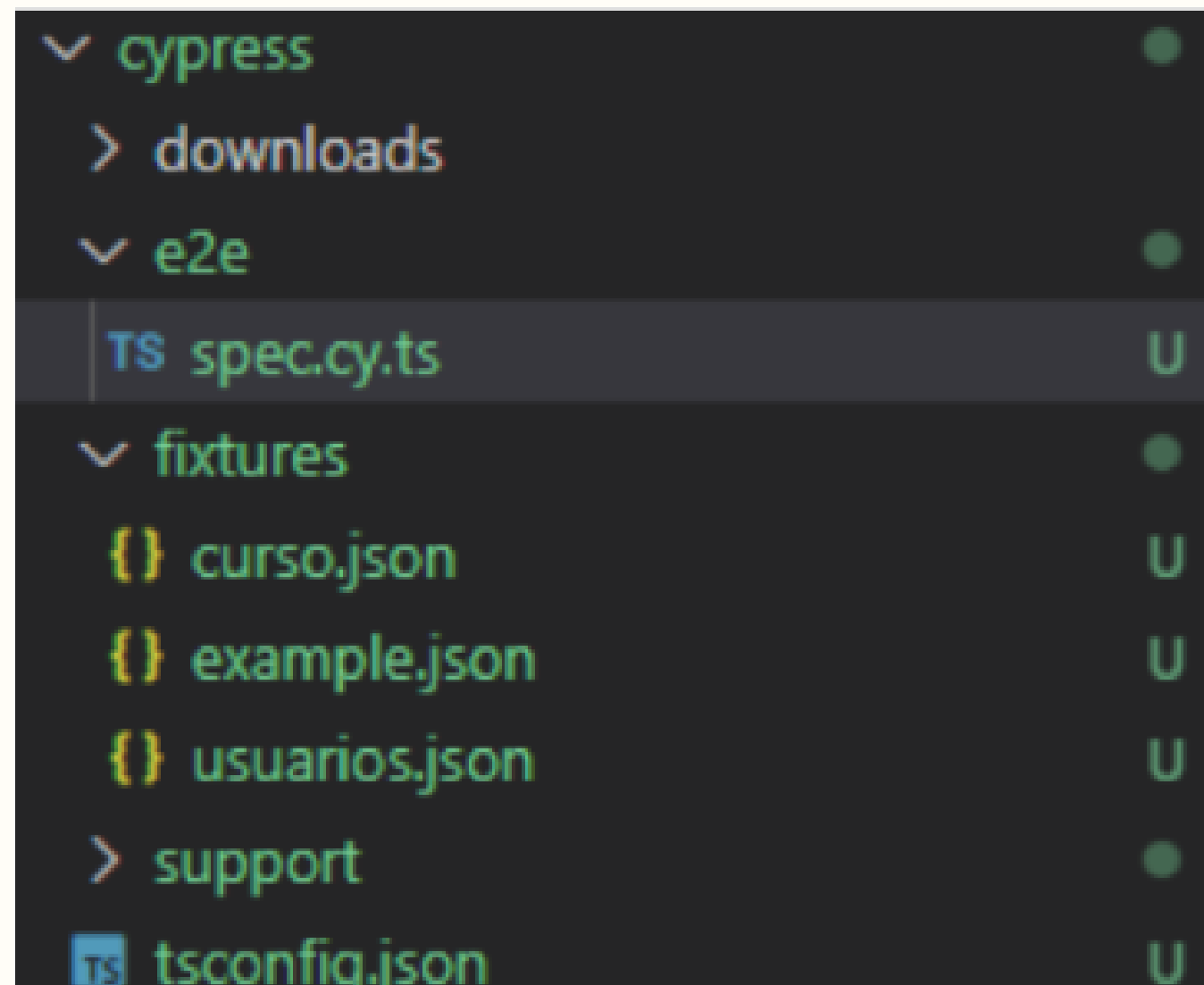
  it('Mi caso de prueba', () => {
    expect(1).toBe(1);
  });
});
```



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  COMENTARIOS  node - serv

PS D:\MIO\7MO SEMESTRE\DISTRIUBIDAS\CursoUsuario\src\app\services> ng test
❖ ✓ Browser application bundle generation complete.
21 07 2024 19:38:22.027:WARN [karma]: No captured browser, open http://localhost:9876/
21 07 2024 19:38:22.236:INFO [karma-server]: Karma v6.4.3 server started at http://localhost:9876/
21 07 2024 19:38:22.238:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
21 07 2024 19:38:22.286:INFO [launcher]: Starting browser Chrome
21 07 2024 19:38:23.685:INFO [Chrome 126.0.0.0 (Windows 10)]: Connected on socket g5zEz51yFQ68Y6feAAAB with id 34466339
Chrome 126.0.0.0 (Windows 10): Executed 6 of 6 SUCCESS (0.004 secs / 0.134 secs)
TOTAL: 6 SUCCESS
```

Pruebas end-to-end (E2E) para la aplicación Angular

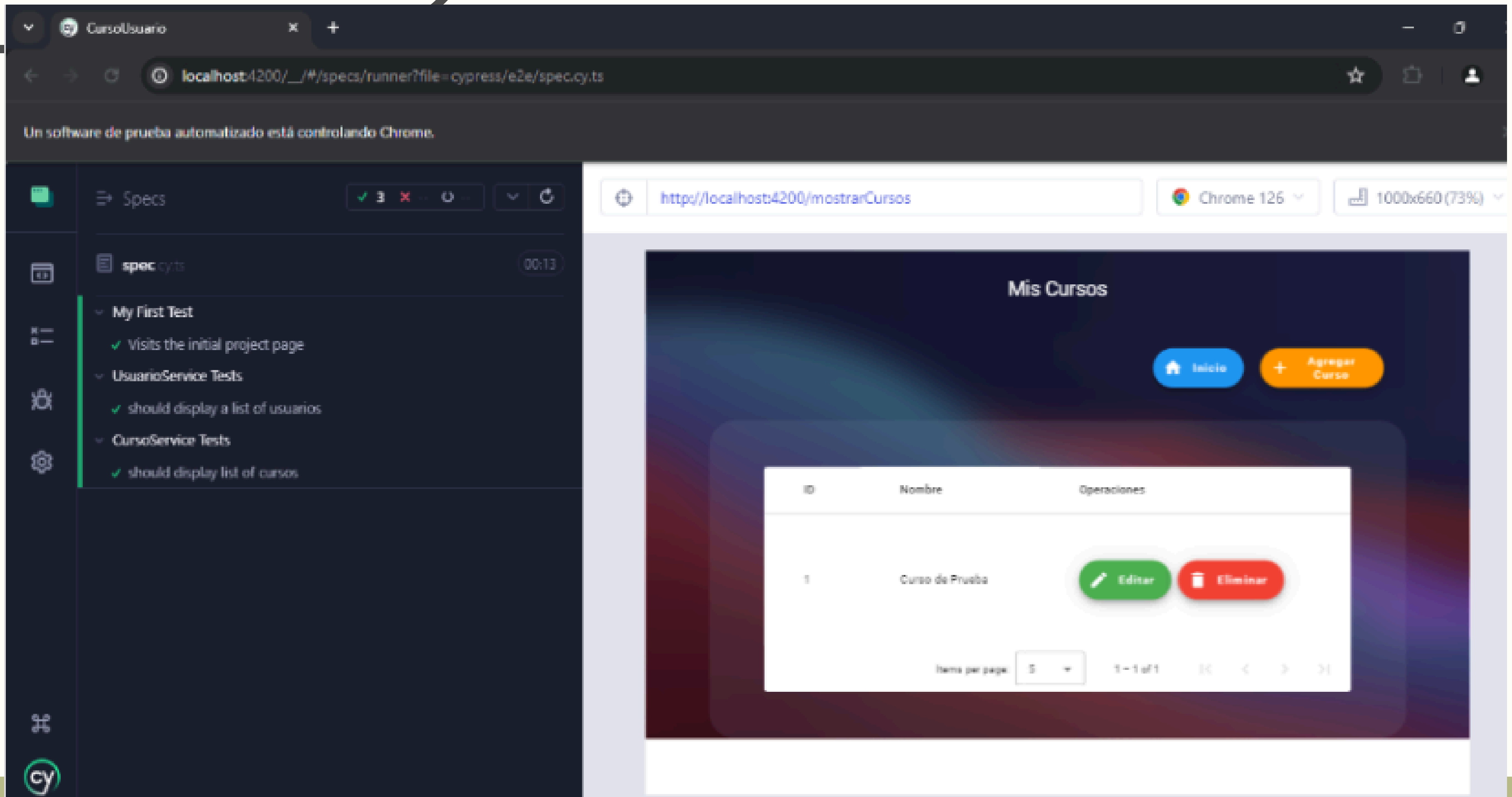


- Las pruebas end-to-end (E2E) simulan escenarios de uso reales, verificando que todas las partes de la aplicación funcione correctamente en conjunto, desde la interfaz de usuario hasta los servicios backend.

```
describe('CursoService Tests', () => {
  beforeEach(() => {
    cy.intercept('GET', 'http://localhost:8002/curso/MostrarCursos', {
      fixture: 'curso.json'
    }).as('getCurso');
  });

  it('should display list of cursos', () => {
    cy.visit('/');
    cy.wait(5000); // Espera a que se cargue la lista de cursos
    cy.wait('@getCurso'); // Espera a que se cargue la lista de cursos
    cy.get('.mat-elevation-z8').should('exist');
    cy.get('.mat-elevation-z8 tbody tr').should('have.length', 1); // asume
    // que tienes 1 curso en el fixture
    cy.get('.mat-elevation-z8 tbody tr td').eq(1).should('contain', 'Curso de
    Prueba');
  });
});
```

- En los archivo **spec.cy.ts** configuraremos la codificación para que realice estas pruebas considerando los archivos **.json**, los cuales simulamos datos que deberían estar en el proyecto



→ Pruebas

- En el ámbito del backend, se llevaron a cabo pruebas unitarias e integración utilizando JUnit y Mockito, así como pruebas con mocks para validar el comportamiento de los microservicios
- Por otro lado, en el entorno del frontend, se realizaron pruebas unitarias y de integración para los componentes y servicios de Angular utilizando Jasmine y Karma. Las pruebas end-to-end (E2E) también se ejecutaron para validar la aplicación Angular en escenarios reales.





Conclusión

El uso de una arquitectura basada en microservicios y Angular ha demostrado ser una solución efectiva y escalable al momento de el desarrollo de un Sistema de Matrículas y Gestión de Estudiantes. Al adaptar los microservicios nos permite que cada componente del sistema funcione de manera independiente, facilitando el mantenimiento y la actualización de los servicios sin interrumpir la operación general del sistema. Angular, por su parte, nos ha proporcionado una plataforma robusta y dinámica para el desarrollo del frontend adaptándonos a un lenguaje con interfaces intuitivas y responsivas.

Recomendaciones

- Implementar medidas de seguridad robustas, como autenticación, autorización y encriptación, para proteger las comunicaciones remotas y prevenir ataques.
- Diseñar un sistema de manejo de errores eficaz que contemple la recuperación de fallos y tiempos de espera, para mejorar la resiliencia del sistema.
- Verificar que las pruebas realizadas cumplan con las expectativas al momento del desarrollo del sistema.



Muchas
GRACIAS

