



UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

APLICACIONES DISTRIBUIDAS

GRUPO 2

MICROSERVICIOS Y ANGULAR 17

SISTEMA DE MATRÍCULAS Y
ESTUDIANTES

INTEGRANTES:

- Alisson Clavijo
- Adrian Iza
- Miguel Morales
- Cristopher Zambrano

TABLA DE CONTENIDOS

1. INTRODUCCION

2. OBJETIVOS

**3. DIAGRAMAS DE
ARQUITECTURA**

**4. DIAGRAMA FLUJO DE
DATOS**

5. DECISIONES DE DISEÑO


6. DESARROLLO

**7. PROBLEMAS ENCONTRADOS Y
CÓMO SE RESOLVIERON**

8. EJECUCIÓN

9. PRUEBAS UNITARIAS

**10. CONCLUSIONES Y
RECOMENDACIONES**



INTRODUCCION

INTRODUCCION

El desarrollo de sistemas informáticos eficientes y escalables es crucial en el ámbito educativo, especialmente cuando se trata de la gestión de matrículas y estudiantes.

Este proyecto se centra en la creación de un Sistema de Matrículas y Estudiantes utilizando una arquitectura de microservicios y el framework Angular.



OBJETIVOS

OBJETIVO GENERAL

Desarrollar un Sistema de Matrículas y Estudiantes basado en una arquitectura de microservicios utilizando Angular para el frontend y MySQL y PostgreSQL para la gestión de bases de datos, que permita la gestión eficiente de estudiantes y cursos, y facilite el proceso de matriculación.

OBJETIVOS ESPECIFICOS

1

Diseñar y desarrollar microservicios que permitan la creación, actualización, eliminación y visualización de registros de estudiantes.

2

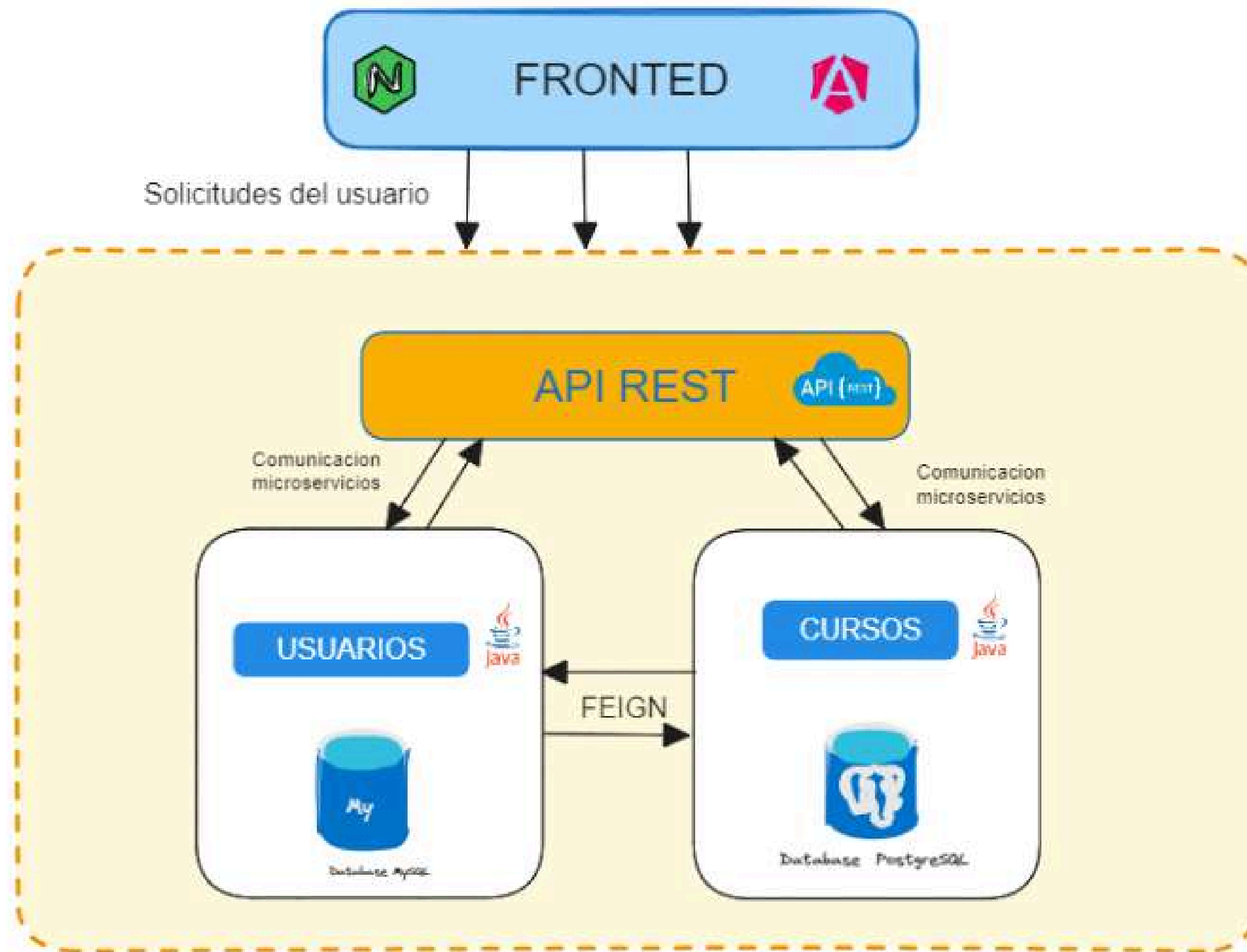
Crear microservicios dedicados a la gestión de cursos, incluyendo la creación, actualización, eliminación y visualización de cursos, utilizando PostgreSQL como sistema de gestión de base de datos.

3

Implementar funcionalidades que permitan a los estudiantes matricularse en los cursos disponibles a través de una interfaz desarrollada en Angular.



DIAGRAMAS DE ARQUITECTURA




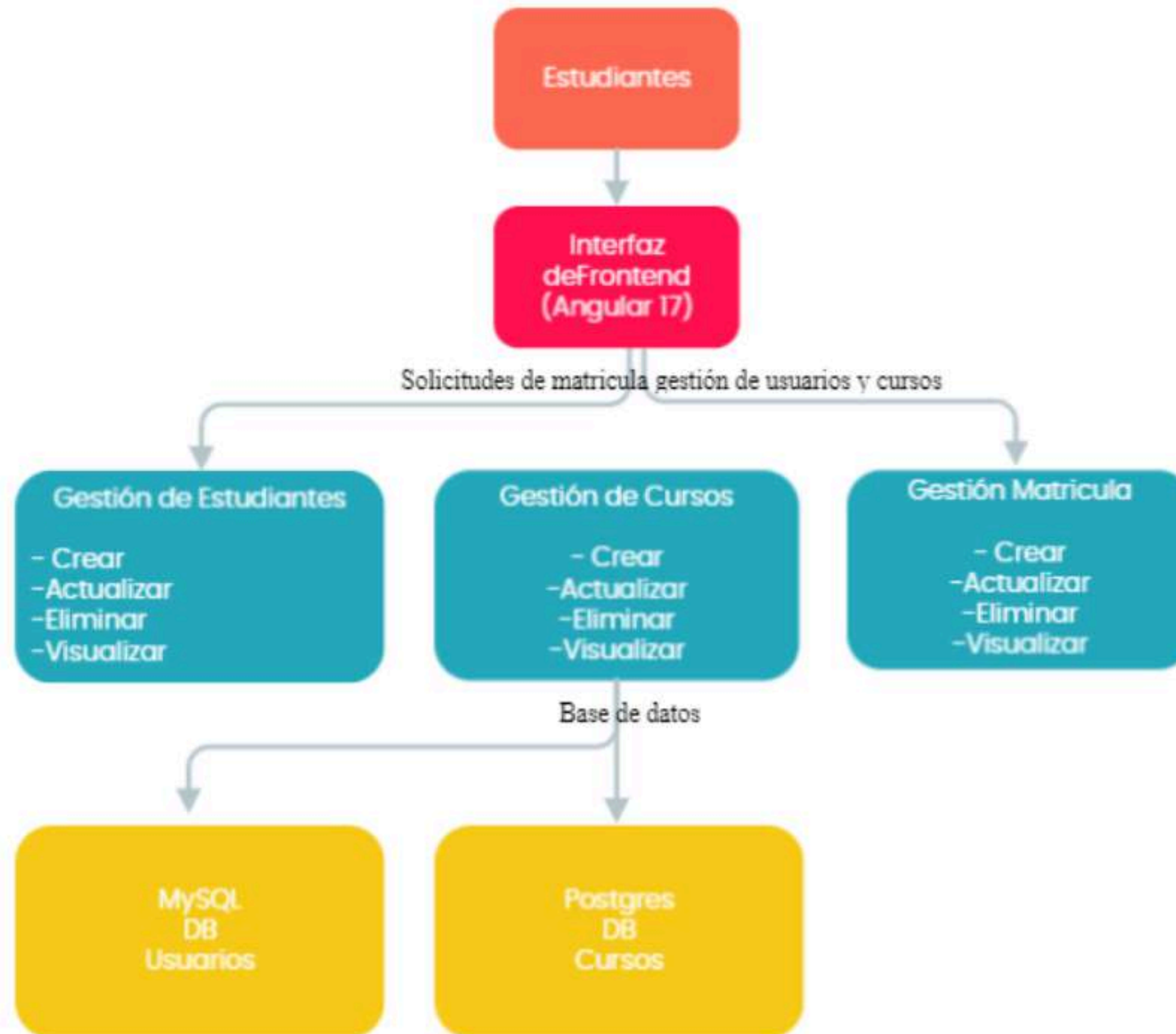


DIAGRAMA FLUJO DE DATOS

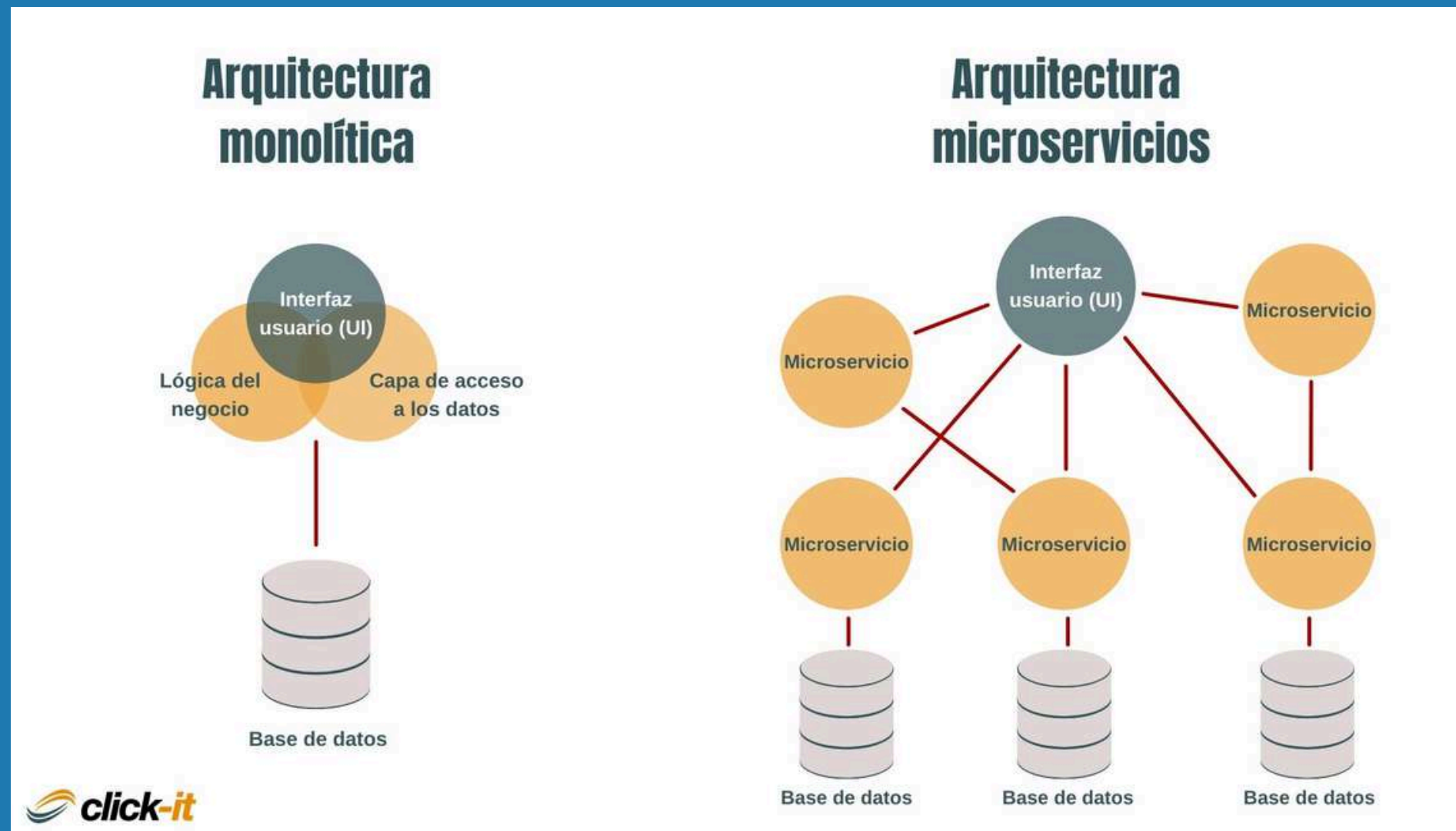




DECISIONES DE DISEÑO

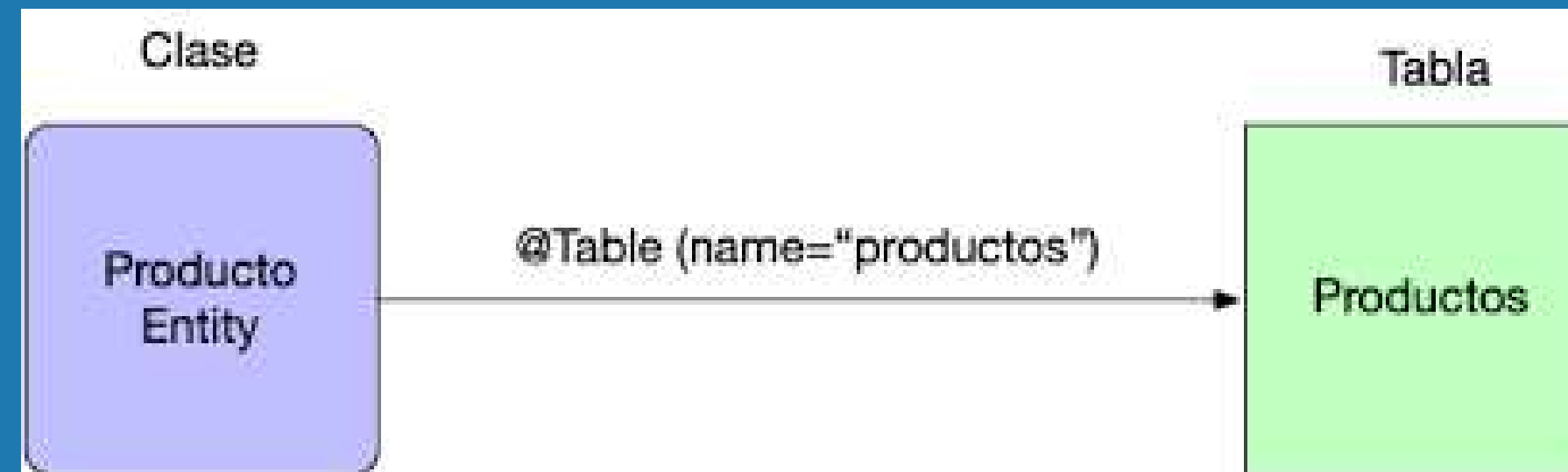
ARQUITECTURA DE MICROSERVICIOS

ESTE PROYECTO SE HA DIVIDIDO EN DOS MICRO SERVICIOS (MSVC-USUARIOS Y MSVC-CURSOS), CADA UNO ES RESPONSABLE DE LA GESTIÓN DE LOS USUARIOS Y DE LOS CURSOS RESPECTIVAMENTE.



DISEÑO DE ENTIDADES

LAS ENTIDADES COMO 'USER' Y 'COURSE' CON ANOTACIONES JPA PARA MAPEAR LAS TABLAS DE LA BASE DE DATOS. TAMBIÉN SE APLICARON ANOTACIONES COMO '@NOTEMPTY' Y '@EMAIL'.



SERVICIOS Y REPOSITORIOS

SERVICIOS: EN LOS SERVICIOS ES DONDE SE IMPLEMENTA LA LÓGICA DE LOS NEGOCIOS, ADEMÁS, SE IMPLEMENTÓ LA INYECCIÓN DE DEPENDENCIAS PARA MANEJAR LA INTERACCIÓN ENTRE LOS COMPONENTES.

REPOSITORIOS: UTILIZAMOS SPRING DATA JPA PARA MANEJAR LA PERSISTENCIA DE DATOS CON 'JPAREPOSITORY', LO QUE NOS AYUDA CON LA IMPLEMENTACIÓN DE OPERACIONES CRUD.



DESARROLLO

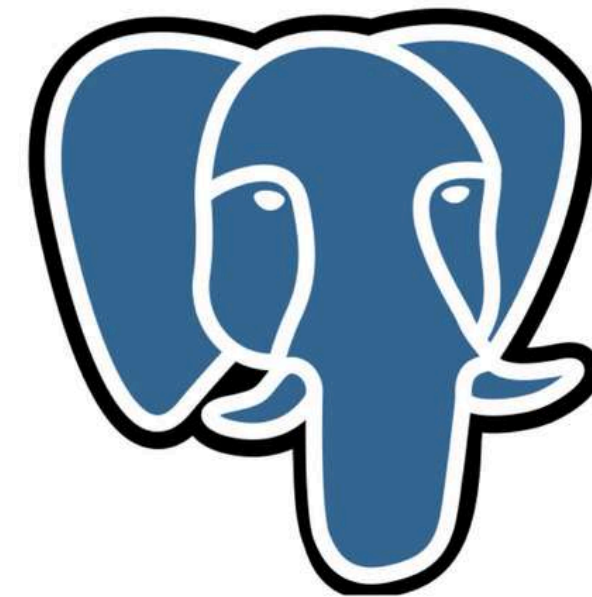



BACKEND

BASE DE DATOS

Los servicios están divididos entre el servicio de la gestión de usuarios (msvc-users), la cual utiliza una base de datos en MySQL para el almacenamiento de los usuarios y el servicio de gestión de cursos (msvc-courses) utiliza la base de datos PostgreSQL para el almacenamiento de los cursos.

Ambos servicios se comunican mediante Feign para las operaciones, además, ambos servicios tienen una configuración de seguridad para permitir consultas desde direcciones admitidas.



A graphic featuring a yellow circle partially visible behind a light yellow parallelogram. The parallelogram is framed by a thick red border. The word "FRONTEND" is written in bold black capital letters across the center of the yellow parallelogram.

FRONTEND

ANGULAR 17

Angular es un framework de diseño de aplicaciones y plataforma de desarrollo para crear aplicaciones de una sola página eficientes y sofisticadas.

Angular, que cuenta con el mantenimiento del equipo dedicado de Google, ofrece un amplio conjunto de herramientas, API y bibliotecas para simplificar y optimizar el flujo de trabajo de desarrollo. Angular ofrece una plataforma sólida sobre la que crear aplicaciones rápidas y confiables que se adaptan tanto al tamaño del equipo como al tamaño de la base del código.






PROBLEMAS ENCONTRADOS Y CÓMO SE RESOLVIERON

Comunicación entre Spring y Angular

- - Síntoma: Los servicios de Spring y la aplicación Angular no se comunicaban correctamente, lo que causaba errores en las consultas.
 - Causa: La configuración de seguridad en Spring no permitía que Angular realizara las solicitudes necesarias.

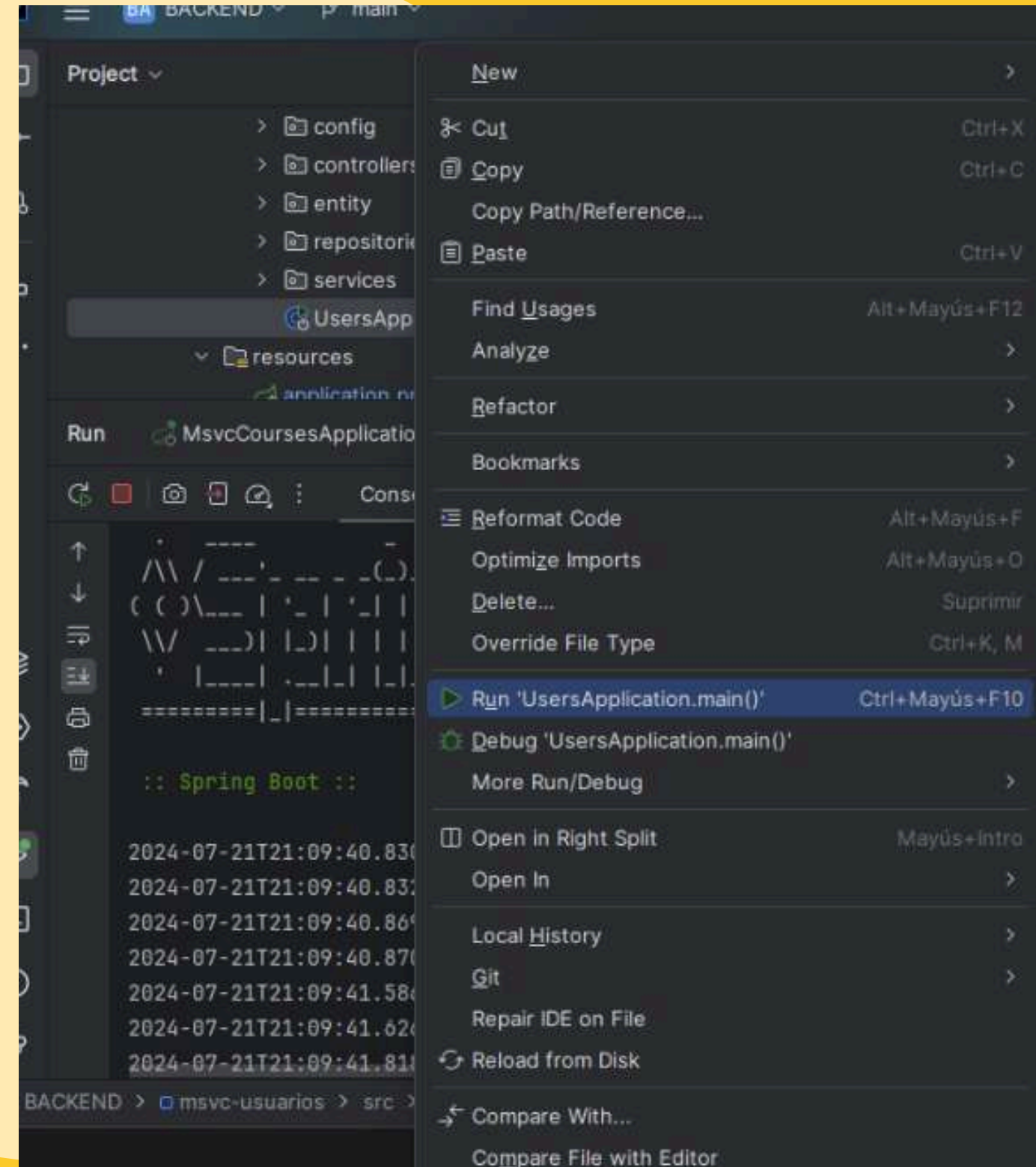
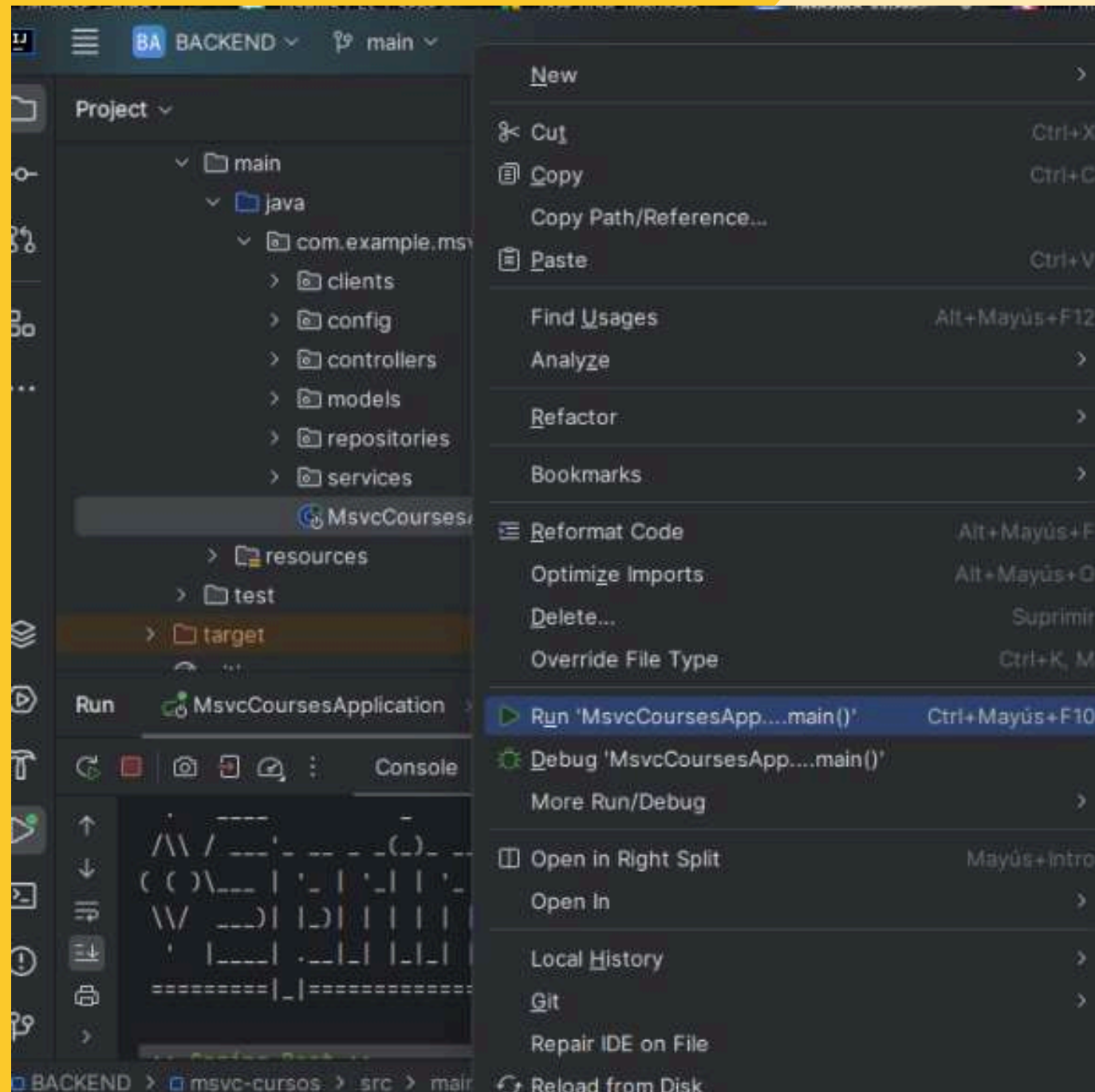
```
1 package com.example.msvc.cursos.config;
2
3 > import ...
4
5 @Configuration
6 public class SecurityConfig {
7     @Bean
8     public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
9         http.httpBasic(AbstractHttpConfigurer::disable);
10        http.csrf(AbstractHttpConfigurer::disable);
11        http.authorizeHttpRequests(authorizeRequests ->
12            authorizeRequests.anyRequest().permitAll()
13        );
14        http.cors(Customizer.withDefaults());
15
16        return http.build();
17    }
18
19     @Bean
20     CorsConfigurationSource corsConfigurationSource() {
21         final UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
22         final CorsConfiguration config = new CorsConfiguration();
23         config.setAllowCredentials(true);
24         config.addAllowedOrigin("http://localhost:4200");
25         config.addAllowedHeader("*");
26         config.addAllowedMethod("OPTIONS");
27         config.addAllowedMethod("HEAD");
28         config.addAllowedMethod("GET");
29         config.addAllowedMethod("PUT");
30         config.addAllowedMethod("POST");
31         config.addAllowedMethod("DELETE");
32         config.addAllowedMethod("PATCH");
33         source.registerCorsConfiguration(pattern: "**", config);
34         return source;
35     }
36 }
```

EJECUCIÓN

PRIMERO EJECUTAMOS EL BACKEND

- Abrimos IntelliJ IDEA
- Ejecutamos MSVC-CURSOS y MSVC-USUARIOS



EJECUTAMOS EL FRONTEND

Abrimos nuestro IDE en este caso visual code,
instalamos los módulos con el comando “npm i”

The image shows the Visual Studio Code interface with the Angular CLI start command executed in the terminal. The terminal output displays the command `npm start` and the resulting bundle sizes for the application.

Terminal Output:

```
PS C:\Users\Adrian\Desktop\proyectoMicroservicios Angular Grupo2-Vproyecto\FRONTEND> npm start

> npx -p @angular/cli start
> ng serve

Browser bundles
Initial chunk files
styles.css      277.63 kB
polyfills.js    93.88 kB
main.js         8.37 kB
chunk-V5HQALAb.js 1.58 kB

Lazy chunk files
chunk-LAPK4QNK.js 14.88 kB
chunk-STJRX2SR.js 11.28 kB
chunk-AXI867I6.js 7.54 kB
chunk-CCUMPGOV.js 1.13 kB
chunk-CG2CFQTV.js 368 bytes
chunk-L677T2UL.js 354 bytes
chunk-NE1H8CY5.js 348 bytes
chunk-BR2425KH.js 211 bytes

Initial total 380.58 kB

Application bundle generation complete. [3.716 seconds]

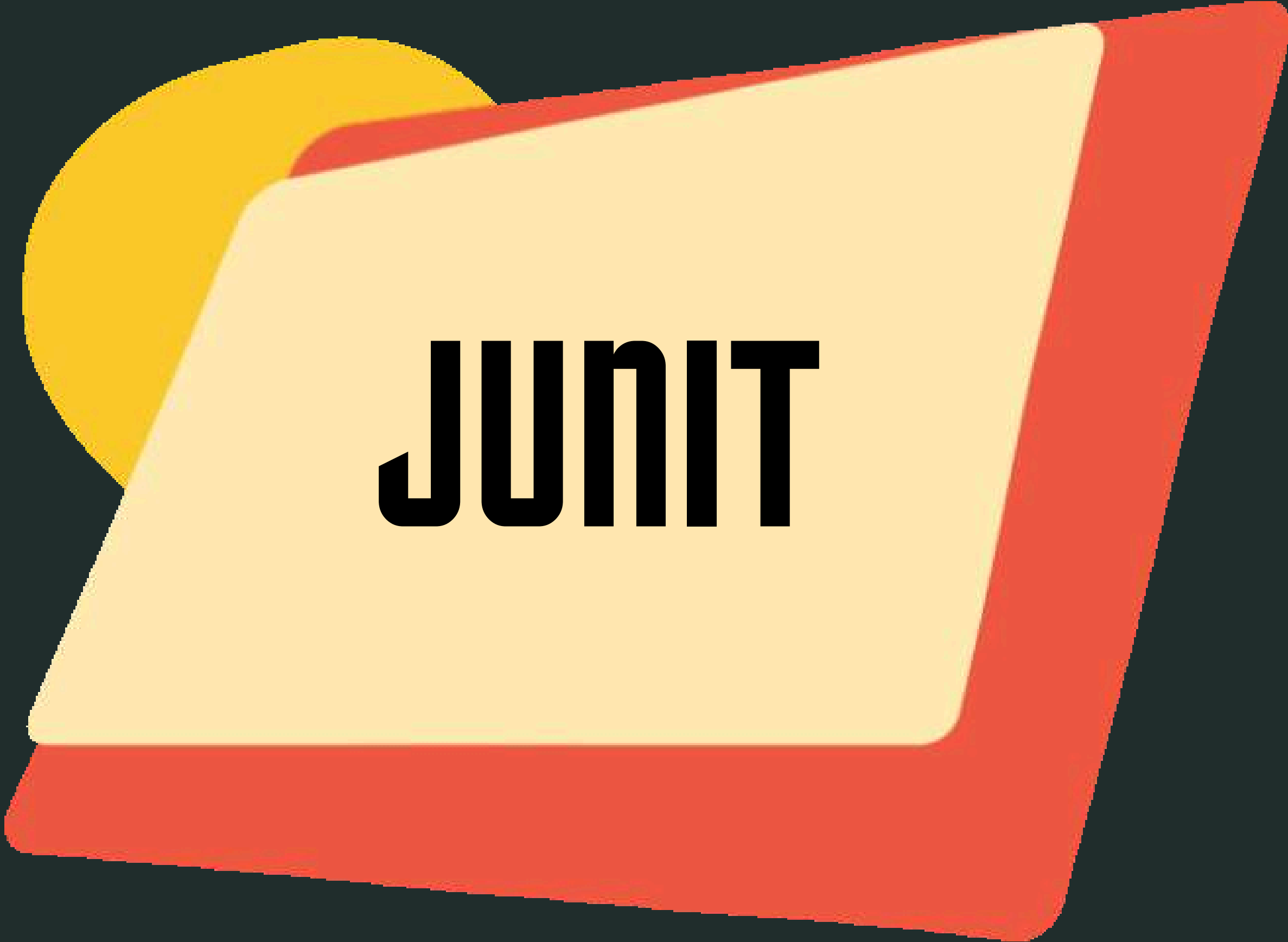
Match mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
→ Local: http://localhost:4200/
→ press h + enter to show help
```

The Explorer pane on the left shows the project structure, including the `angular.json` file. The Output pane on the right shows the bundle sizes for the application.

The bottom part of the image shows a web browser window displaying the application running on `localhost:4200/users`. The page has a sidebar with links for `Usuarios` and `Matriculaciones`. The main content area shows a `Formulario de Usuario` with fields for `Nombre`, `Email`, and `Password`, and a `Guardar` button.



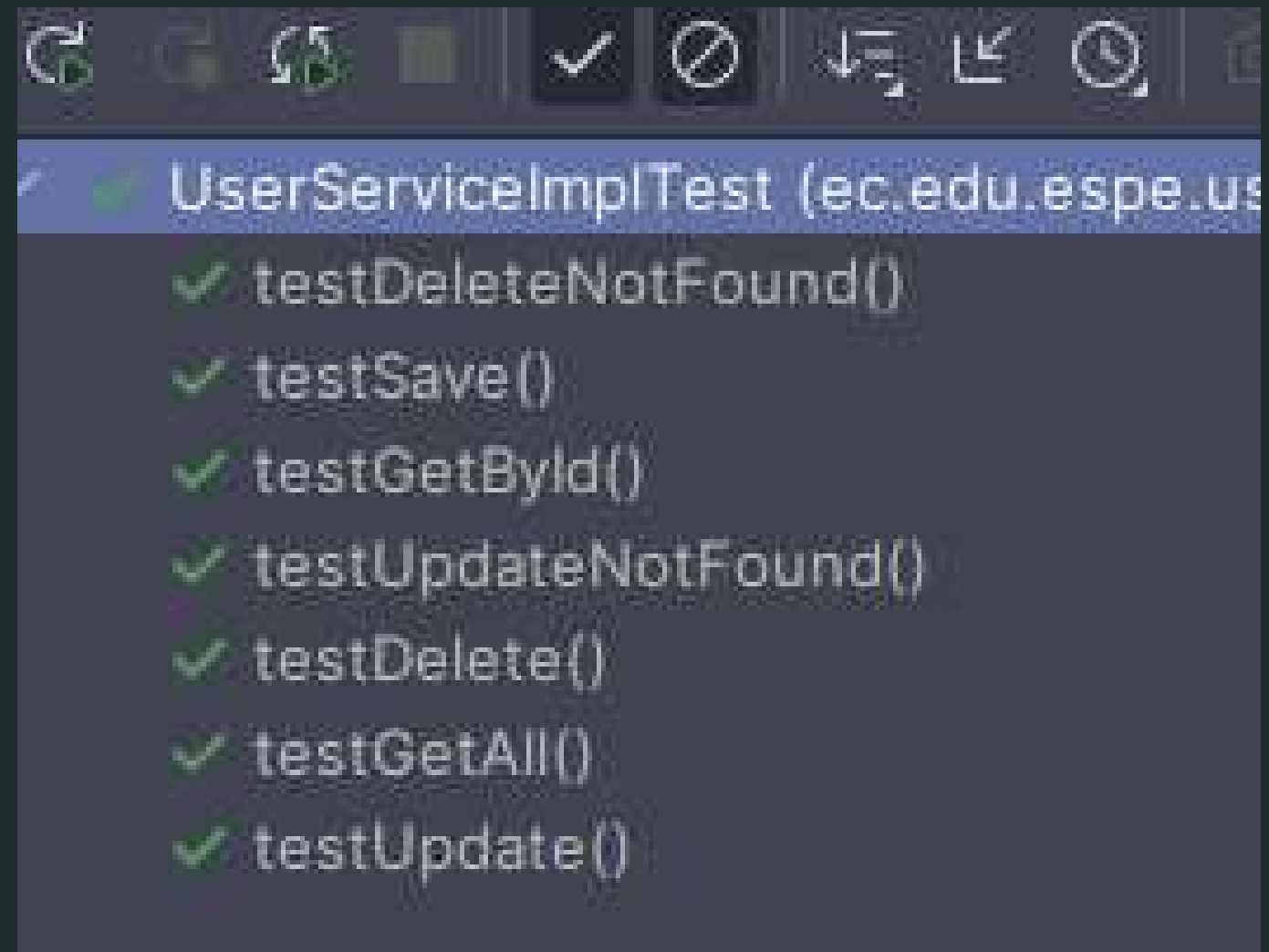
PRUEBAS UNITARIAS



JUNIT

Pruebas Unitarias

msvc-usuarios



msvc-cursos



Pruebas de Integración

H2

- Base de datos Embebida
- application.properties

```
# Configuración para los test de integración
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

msvc-usuarios

```
@Test
@Transactional
public void testSaveAndFindUser() {
    User user = new User();
    user.setName("John Doe");
    user.setEmail("john.doe@example.com");
    user.setPassword("password");

    User savedUser = userService.save(user);
    assertThat(savedUser).isNotNull();
    assertThat(savedUser.getId()).isNotNull();

    User foundUser = userService.getById(savedUser.getId()).orElse( other: null);
    assertThat(foundUser).isNotNull();
    assertThat(foundUser.getId()).isEqualTo(savedUser.getId());
}
```

```
@Test
@Transactional
public void testSaveAndDeleteUser() {
    User user = new User();
    user.setName("John Doe");
    user.setEmail("john.doe@example.com");
    user.setPassword("password");

    User savedUser = userService.save(user);
    assertThat(savedUser).isNotNull();
    userService.delete(savedUser.getId());

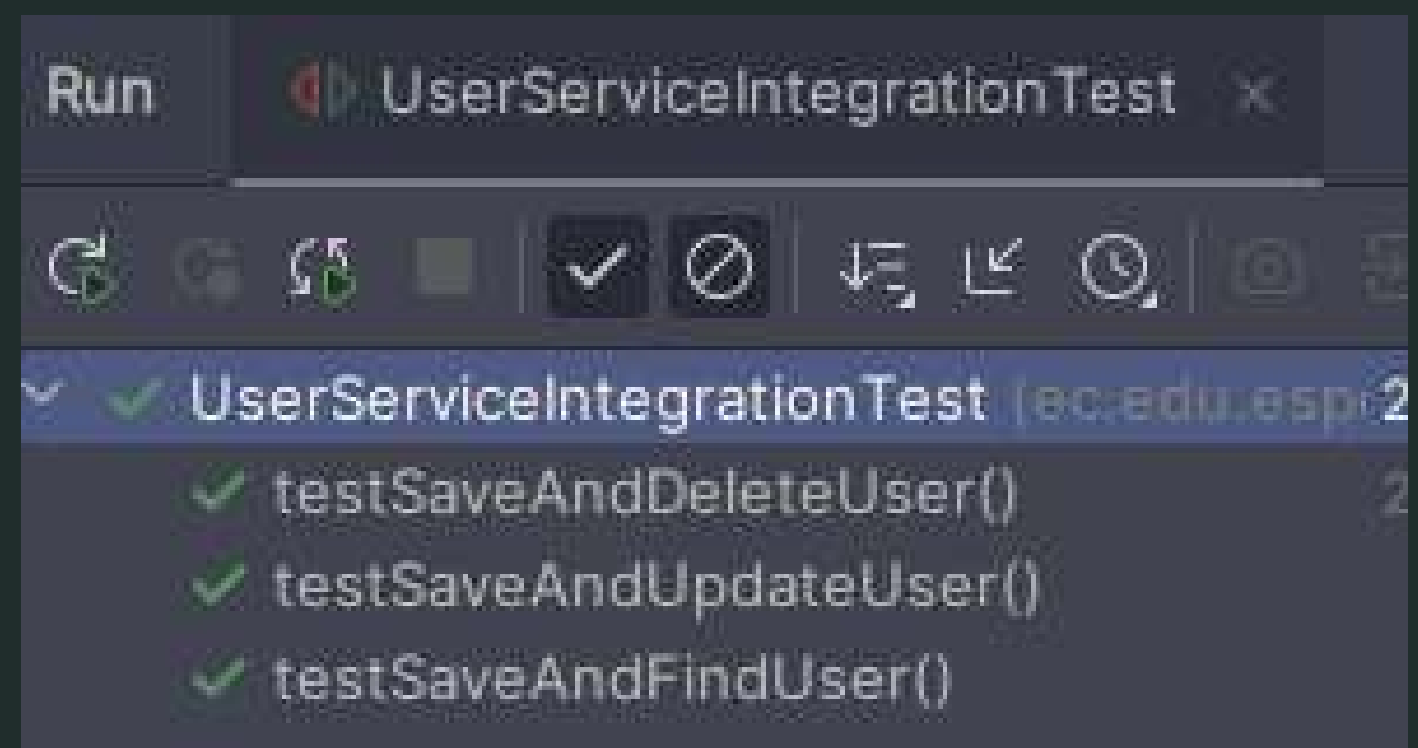
    User foundUser = userService.getById(savedUser.getId()).orElse( other: null);
    assertThat(foundUser).isNull();
}
```

```
@Test
@Transactional
public void testSaveAndUpdateUser() {
    User user = new User();
    user.setName("John Doe");
    user.setEmail("john.doe@example.com");
    user.setPassword("password");

    User savedUser = userService.save(user);
    assertThat(savedUser).isNotNull();

    savedUser.setName("Jane Doe");
    User updatedUser = userService.save(savedUser);
    assertThat(updatedUser).isNotNull();
    assertThat(updatedUser.getId()).isEqualTo(savedUser.getId());
    assertThat(updatedUser.getName()).isEqualTo( expected: "Jane Doe");

    User foundUser = userService.getById(updatedUser
        .getId()).orElse( other: null);
    assertThat(foundUser).isNotNull();
    assertThat(foundUser.getId()).isEqualTo(updatedUser.getId());
    assertThat(foundUser.getName()).isEqualTo( expected: "Jane Doe");
}
```





JASMINE Y KARMA

PRUEBAS UNITARIAS

LAS PRUEBAS SE HAN LLEVADO A CABO UTILIZANDO EL FRAMEWORK DE PRUEBAS JASMINE EN CONJUNTO CON ANGULAR TESTBED Y HTTPCLIENTTESTINGMODULE PARA LA SIMULACIÓN DE SOLICITUDES HTTP.



Karma v 6.4.3 - connected; test: complete;

DEBUG

Chrome 126.0.0.0 (Windows 10) is idle

Jasmine 4.6.1

Options

Ran 1 of 17 specs - run all

finished in 0.048s

Incomplete: fit() or fdescribe() was found, 1 spec, 0 failures, randomized with seed 52809

EnrollmentService
• should be created

EnrollmentComponent
• should create

UsersComponent
• should create

SidebarComponent
• should create

UsersComponent
• should create

AppComponent
• should create the app
• should have the 'msvcApp' title
• should render title

CourseService
• should be created

UserService
• should retrieve users from the API via GET
• should be created

DialogComponent
• should create

CoursesComponent
• should create

DialogService
• should cancel dialog
• should be created
• should show dialog
• should confirm dialog

SERVICIOS PROBADOS: ENROLLMENTSERVICE

Resultado de la Prueba: Exitosa

- Descripción: Se verificó la correcta creación del servicio EnrollmentService.
- Método probado: should be created
- Propósito: Garantizar que el servicio EnrollmentService se inicializa correctamente y está listo para ser utilizado.
- Verificación de Servicio: Confirma que el servicio está correctamente configurado en el módulo y se puede inyectar sin errores.

```
15 | providers: [EnrollmentService]
16 | });
17 | service = TestBed.inject(EnrollmentService);
18 | httpMock = TestBed.inject(HttpTestingController);
19 | });
20 |
21 | afterEach(() => {
22 |   httpMock.verify();
23 | });
24 |
25 | fit('should be created', () => {
26 |   expect(service).toBeTruthy();
27 | });
28 |
```

Karma v 6.4.3 - connected; test: complete;

DEBUG

Chrome 126.0.0.0 (Windows 10) is idle

Jasmine 4.6.1

Options

Ran 1 of 17 specs - run all

finished in 0.048s

Incomplete: fit() or fdescribe() was found, 1 spec, 0 failures, randomized with seed 52809

EnrollmentService
• should be created

USERSERVICE

Resultado de la Prueba: Exitosa

- Descripción: Se realizaron dos pruebas principales para verificar el funcionamiento del servicio UserService
- Método probado: should be created
- should be created: Se confirmó la correcta creación del servicio.
- should retrieve users from the API via GET: Se validó que el método getUsers realiza una solicitud GET a la API y devuelve una lista de usuarios.
- Verificación de API: Asegura que la interacción con la API para recuperar usuarios funcione como se espera.
- Verificación de Servicio: Confirma que el servicio esté disponible para la lógica de negocio relacionada con los usuarios.

```
describe('UserService', () => {
  beforeEach(() => {
    imports: [HttpClientTestingModule],
    providers: [UserService]
  });
  service = TestBed.inject(UserService);
  httpMock = TestBed.inject(HttpTestingController);
});

afterEach(() => {
  httpMock.verify();
});

it('should be created', () => {
  expect(service).toBeTruthy();
});

it('should retrieve users from the API via GET', () => {
  const dummyUsers: User[] = [
    { id: 1, name: 'User 1', email: 'user1@example.com' },
    { id: 2, name: 'User 2', email: 'user2@example.com' }
  ];

  service.getUsers().subscribe(users => {
    expect(users.length).toBe(2);
    expect(users).toEqual(dummyUsers);
  });

  const req = httpMock.expectOne(`${environment.apiUrl}/users`);
  expect(req.request.method).toBe('GET');
  req.flush(dummyUsers);
});
```

UserService

- should retrieve users from the API via GET
- should be created

COURSESERVICE

Resultado de la Prueba: Exitosa

- Descripción: Se verificó la correcta creación del servicio CourseService.
- Método probado: should be created
- Propósito: Garantizar que el servicio CourseService se inicializa correctamente y está listo para ser utilizado.
- Verificación de Servicio: Confirma que el servicio está disponible para manejar operaciones relacionadas con los cursos, como obtener, crear, actualizar y eliminar cursos.

Complexity is 4 Everything is cool!

```
describe('CourseService', () => {
```

```
  let service: CourseService;
```

```
  let httpMock: HttpTestingController;
```

```
  beforeEach(() => {
```

```
    TestBed.configureTestingModule({
```

```
      imports: [HttpClientTestingModule],
```

```
      providers: [CourseService]
```

```
    });
```

```
    service = TestBed.inject(CourseService);
```

```
    httpMock = TestBed.inject(HttpTestingController);
```

```
  });
```

```
  afterEach(() => {
```

```
    httpMock.verify();
```

```
  });
```

```
  it('should be created', () => {
```

```
    expect(service).toBeTruthy();
```

```
  });
```

```
});
```

CourseService

- should be created

COMPONENTES PROBADOS: DIALOGCOMPONENT

Resultado de la Prueba: Exitosa

- Descripción: Se verificó la correcta creación del componente DialogComponent.
- Método probado: should create
- Propósito: Asegurar que el componente se inicializa sin errores y está listo para su uso en la interfaz de usuario.
- Verificación de Componente: Confirma que el componente se instancia correctamente, lo cual es fundamental para su uso en la interfaz de usuario.

```
1 import { ComponentFixture, TestBed } from '@angular/core/testing';
2
3 import { DialogComponent } from '../dialog.component';
4
5 describe('DialogComponent', () => {
6   let component: DialogComponent;
7   let fixture: ComponentFixture<DialogComponent>;
8
9   beforeEach(async () => {
10     await TestBed.configureTestingModule({
11       imports: [DialogComponent]
12     })
13     .compileComponents();
14
15     fixture = TestBed.createComponent(DialogComponent);
16     component = fixture.componentInstance;
17     fixture.detectChanges();
18   });
19
20   it('should create', () => {
21     expect(component).toBeTruthy();
22   });
23 });
24
```

DialogComponent

- should create

SERVICIOS DE DIÁLOGO: DIALOGSERVICE

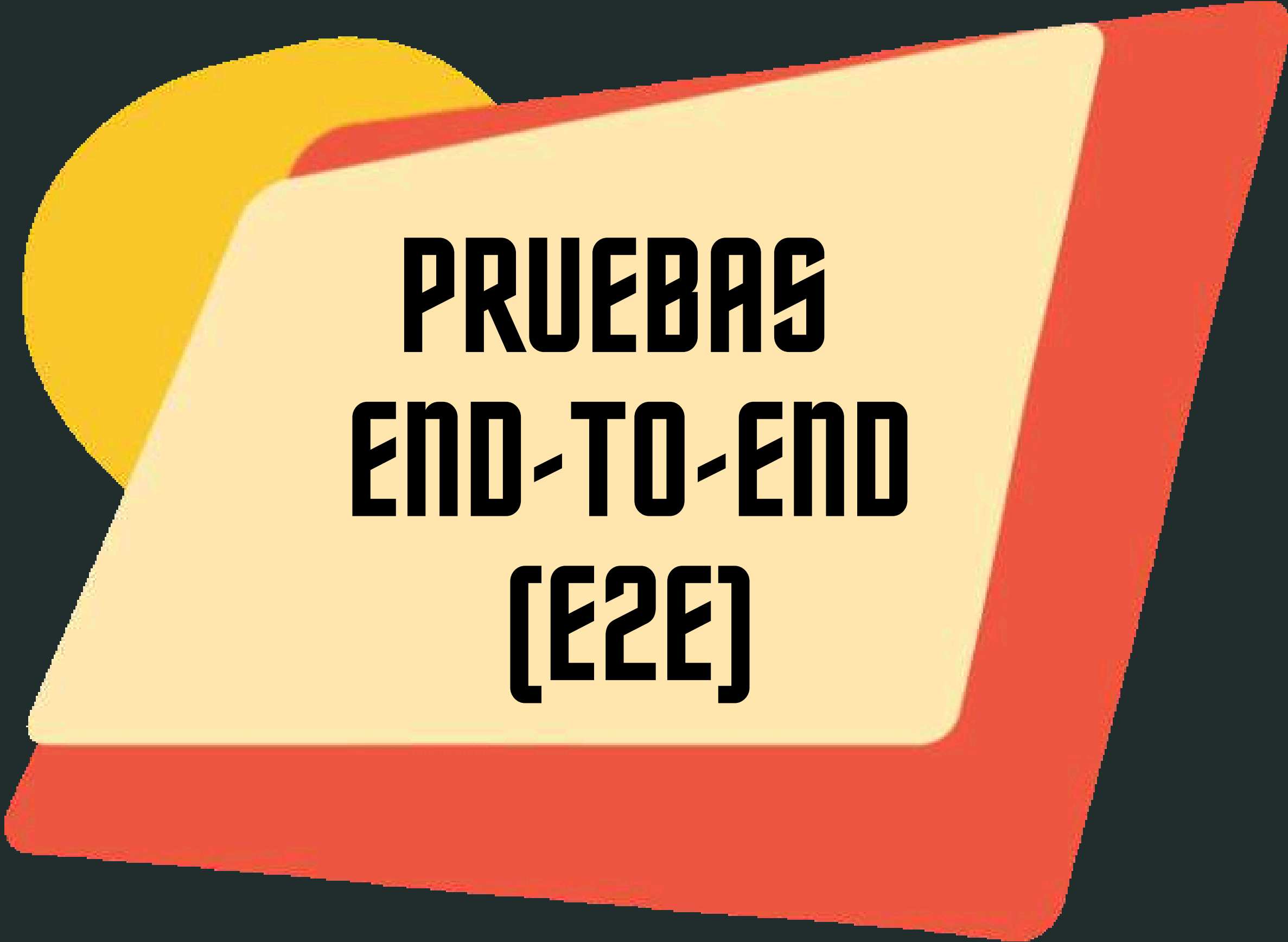
```
3 Complexity is 9 It's time to do something...
4 describe('DialogService', () => {
5   let service: DialogService;
6
7   beforeEach(() => {
8     TestBed.configureTestingModule({});
9     service = TestBed.inject(DialogService);
10  });
11
12  it('should be created', () => {
13    expect(service).toBeTruthy();
14  });
15
16  it('should show dialog', () => {
17    const options: DialogOptions = { message: 'Test Message' };
18    service.show(options);
19
20    service.onShow().subscribe(res => {
21      expect(res).toEqual(options);
22    });
23  });
24
25  it('should cancel dialog', () => {
26    service.cancel();
27
28    service.onCancel().subscribe(() => {
29      expect(true).toBe(true);
30    });
31  });
32
33  it('should confirm dialog', () => {
34    service.confirm();
35
36    service.onConfirm().subscribe(() => {
37      expect(true).toBe(true);
38    });
39  });
40 });
```

Resultado de la Prueba: Exitosa

- Descripción: Se realizaron varias pruebas para verificar el correcto funcionamiento del servicio DialogService.
- Métodos probados:
- should be created: Se confirmó la correcta creación del servicio.
- should show dialog: Se verificó que el método show emite los valores correctos para mostrar un diálogo.
- should cancel dialog: Se validó que el método cancel emite correctamente la cancelación del diálogo.
- should confirm dialog: Se aseguró que el método confirm emite correctamente la confirmación del diálogo.
- Verificación de Funcionalidad de Diálogo: Asegura que la funcionalidad de los diálogos, como mostrar, cancelar y confirmar, funcione correctamente en la aplicación, permitiendo una interacción adecuada del usuario.

DialogService

- should be created
- SPEC HAS NO EXPECTATIONS should confirm dialog
- SPEC HAS NO EXPECTATIONS should cancel dialog
- SPEC HAS NO EXPECTATIONS should show dialog



**PRUEBAS
END-TO-END
(E2E)**

The text is centered on a light yellow rectangular background with rounded corners. This yellow area is set against a larger, tilted red background. To the left of the yellow area, a portion of a yellow circle is visible. The entire composition is on a dark blue background.



CYPRESS

Realizamos el primer comando para poder guardar como una dependencia de desarrollo

```
PS C:\Users\Ali\Desktop\Distribuidas\Git\Microservicios_Angular_Grupo2-\Proyecto\FRONTEND>
  npm install cypress --save-dev

added 114 packages, and audited 1102 packages in 21s

159 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



CYPRESS

Realizamos el segundo comando que usa el cypress que se desacrargo en nuestros modules y con opens nos abra una interfaz.

```
PS C:\Users\Ali\Desktop\Distribuidas\Git\Microservicios_Angular_Grupo2-\Proyecto\FRONTEND> node_modules/.bin/cypress open
○ It looks like this is your first time using Cypress: 13.13.1

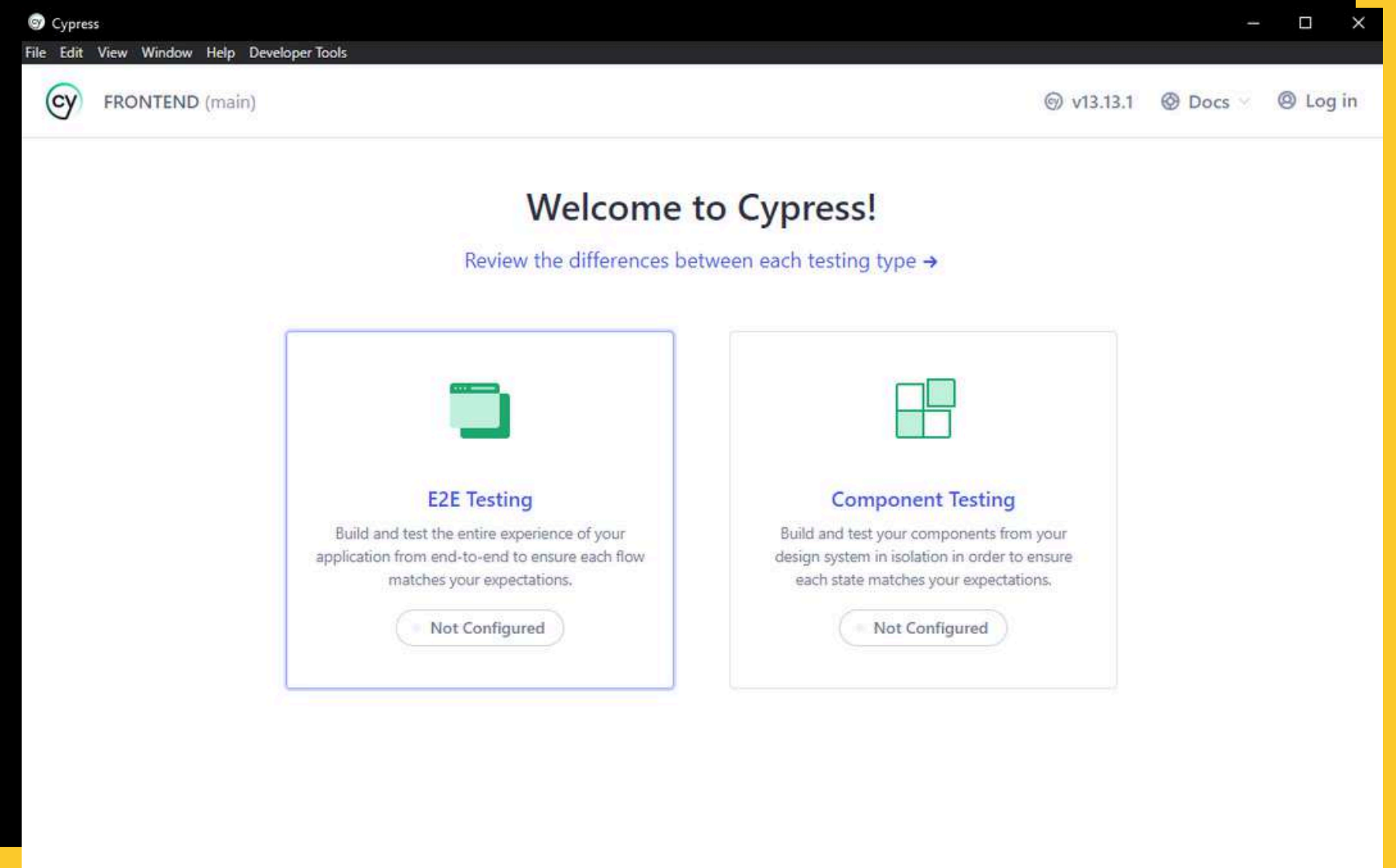
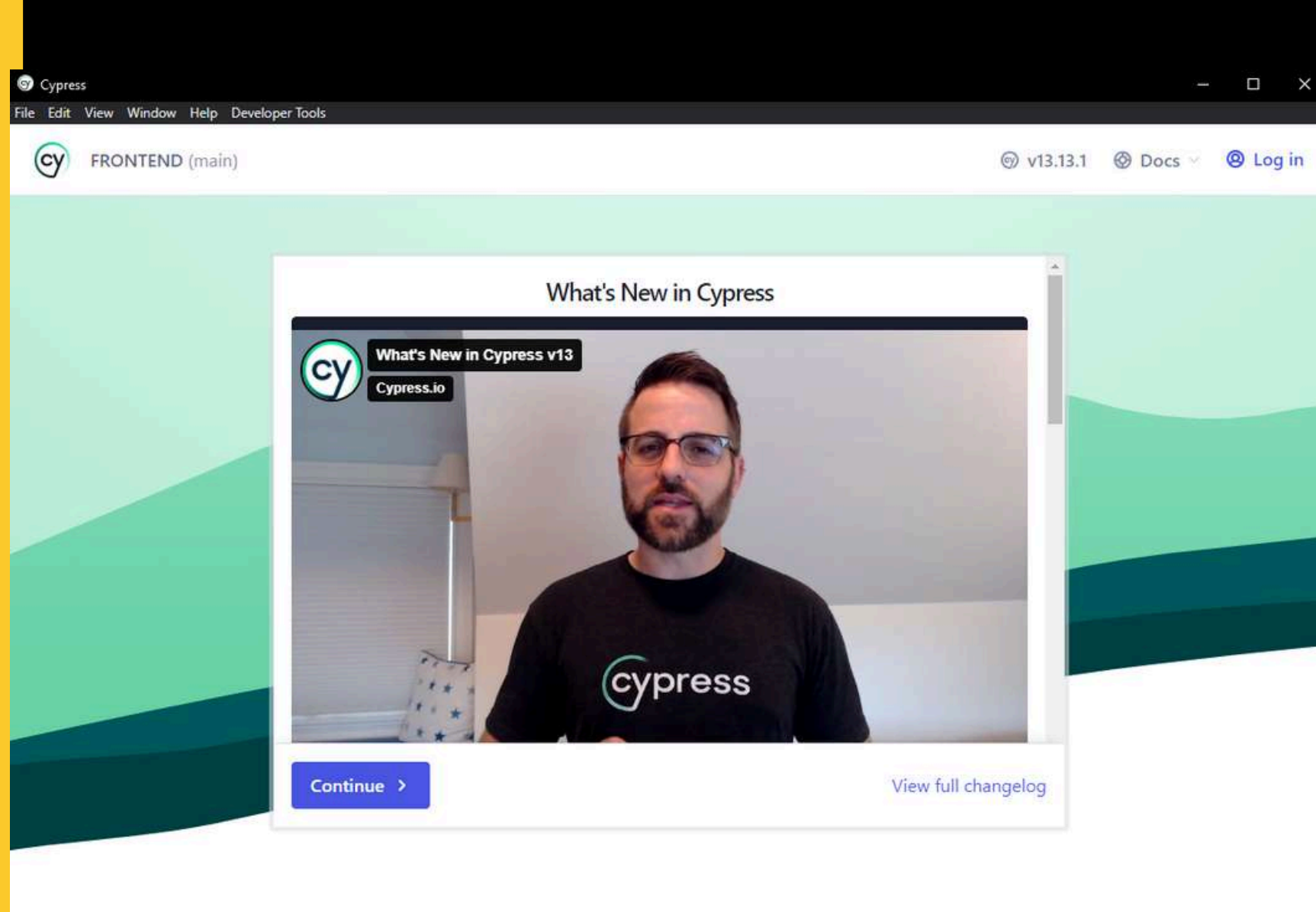
✓ Verified Cypress! C:\Users\Ali\AppData\Local\Cypress\Cache\13.13.1\Cypress

Opening Cypress...

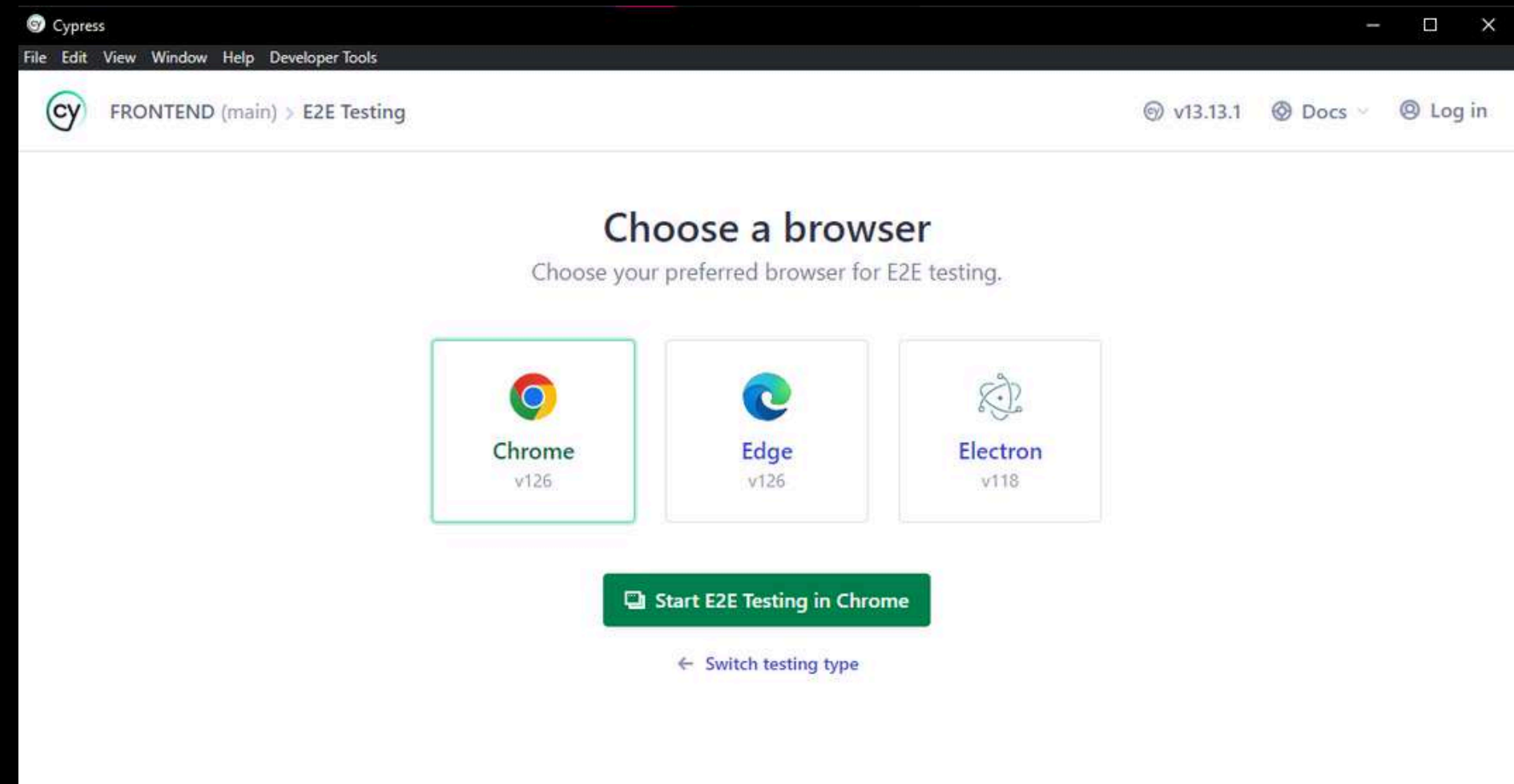
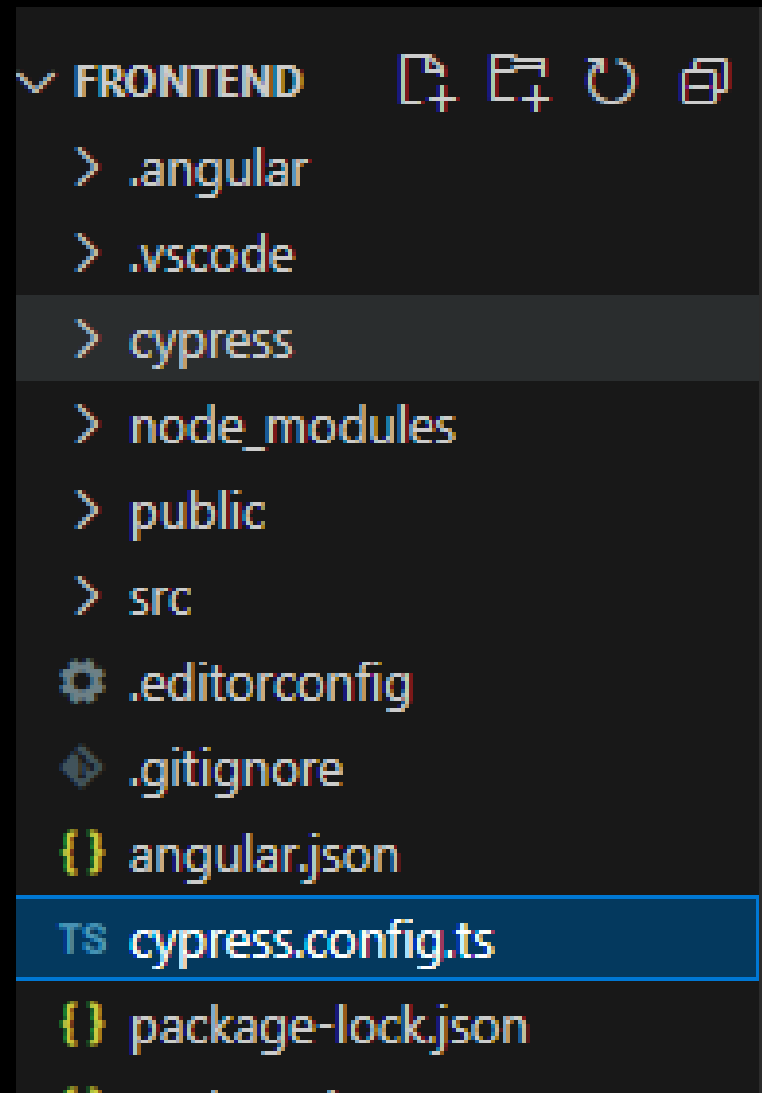
DevTools listening on ws://127.0.0.1:59197/devtools/browser/66d6dd43-bb88-4caf-a0e1-2a7490a4b580
Missing baseUrl in compilerOptions. tsconfig-paths will be skipped
```


CYPRESS

Realizamos el segundo comando que usa el cypress que se desacrargo en nuestros modules y con opens nos abraira una interfaz.



CYPRESS



Una vez ejecutada nos crea la carpeta "cypress" y un archivo "cypress.json" donde pondremos las configuraciones necesarias como la URL de la aplicación que vamos a testear

CYPRESS

Configuramos las pruebas e2e para ejecutar con cypress

TS cypress.config.ts X

TS cypress.config.ts > ...

```
1  import { defineConfig } from "cypress";
2
3  export default defineConfig({
4    e2e: {
5      baseUrl: "http://localhost:4200/",
6      setupNodeEvents(on, config) {
7        // implement node event listeners here
8      },
9    },
10  });
11
```

TS cypress.config.ts

{} package.json ●

{} package.json > {} scripts > e2e

```
1  {
2    "name": "msvc-app",
3    "version": "0.0.0",
4    "scripts": {
5      "ng": "ng",
6      "start": "ng serve",
7      "build": "ng build",
8      "watch": "ng build --watch --configuration development",
9      "test": "ng test",
10     "serve:ssr:msvcApp": "node dist/msvc-app/server/server.mjs",
11     "e2e": "cypress open"
12   },

```

Configuramos la URL base, con nuestra aplicacion

CYPRESS

En una terminal
ejecutamos la
aplicacion

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

chunk-S5EDBUCL.mjs | user-routes | 384 bytes |
chunk-YSANAP7J.mjs | - | 246 bytes |

Application bundle generation complete. [7.503 seconds]

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request tr
ansformations.
Re-optimizing dependencies because lockfile has changed
  → Local: http://localhost:4200/
  → press h + enter to show help
Re-optimizing dependencies because lockfile has changed
█
```

```
● PS C:\Users\Ali\Desktop\Distribuidas\Git\Microservicios_Angular_Grup
o2-\Proyecto\FRONTEND> npm run e2e

> msvc-app@0.0.0 e2e
> cypress open

DevTools listening on ws://127.0.0.1:61202/devtools/browser/234d82da
-72bc-49a0-9957-71845f39f311
Missing baseUrl in compilerOptions. tsconfig-paths will be skipped
PS C:\Users\Ali\Desktop\Distribuidas\Git\Microservicios_Angular_Grup
o2-\Proyecto\FRONTEND> █
```

En otra terminal levantamos
cypress y como ya creamos el
script que nos permitía acceder
a cypress open , usamos el
comando "npm run e2e"

CYPRESS

Escribimos el
test, para
poder probar
que se ingrese
un usuario

TS cypress.config.ts

JS add-user.cy.js ●

JS add-user-correct.cy.js ●

JS todo.cy.js

{ } package.json

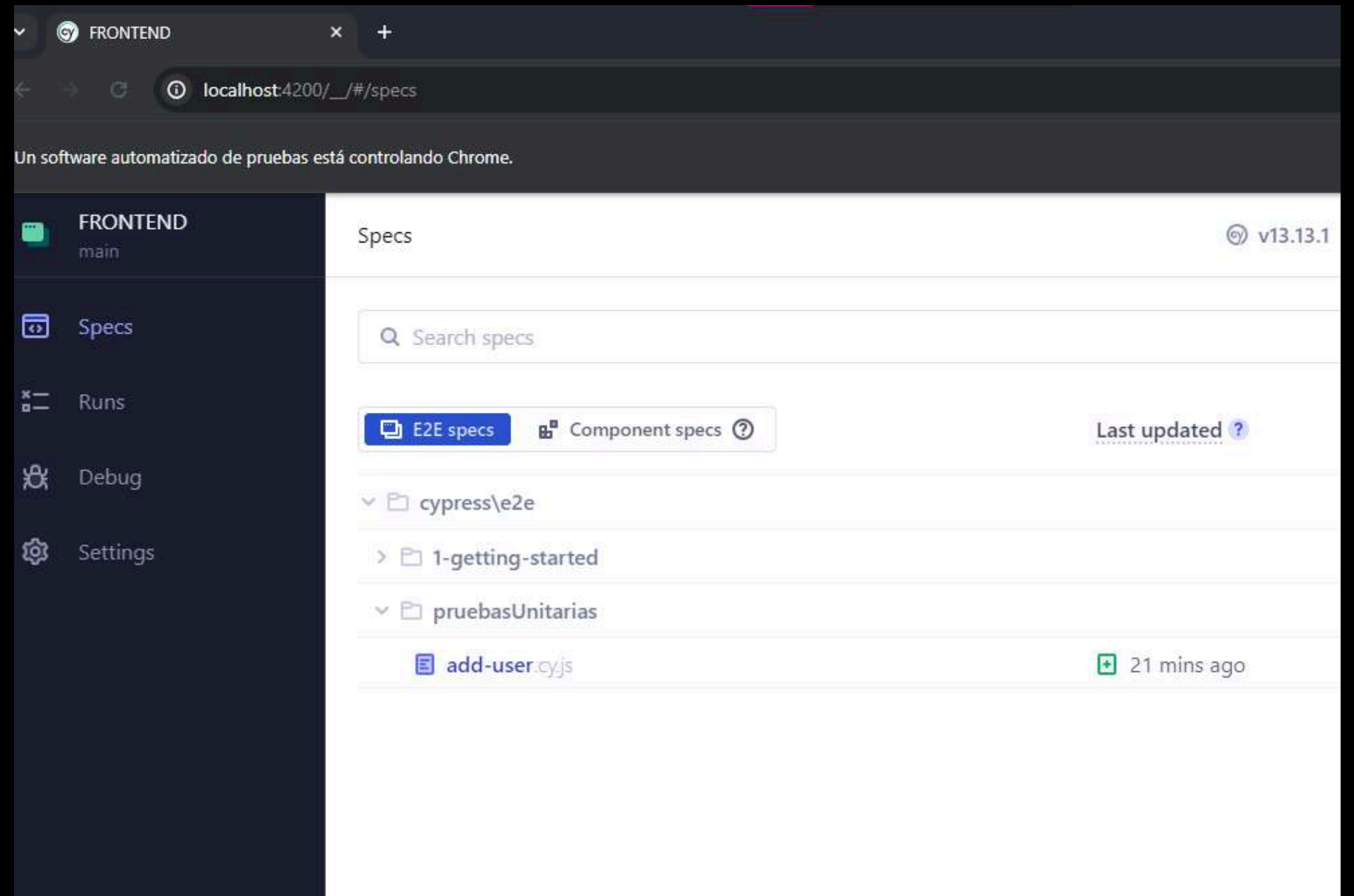
cypress > e2e > pruebasUnitarias > JS add-user.cy.js > ...

```
1 describe('add user test', () => {
2   it('Deberia no ingresar el usuario, si solo se llena el nombre', () => {
3     cy.visit('/users');
4
5     // Ajustar el selector para encontrar el input de nombre correctamente
6     cy.get('input[name="name"]').type('Alisson');
7   });
8 });
```

10

CYPRESS

Ya se nos refleja el test dentro de la interfaz de cypress, le damos Click, nos mostrara otra interfaz donde se probara el test.



CYPRESS

Probara el test
En este caso con la
prueba nos da error
debido a que
deberíamos ingresar el
email y la contraseña
para que se cree el
usuario.

The screenshot shows the Cypress test runner interface on the left and the application under test on the right.

Cypress Test Runner (Left):

- Specs: 1 test, 0 failures, 0 retries.
- add-user.cy.js: 00:02
- add user test: Deberia ingresar usuario
- TEST BODY:
 - 1 visit /users
 - 2 get input[name="name"]
 - 3 -type Alisson
- (xhr) POST 400 http://localhost:8001/users

Application Interface (Right):

http://localhost:4200/users

Cursos

Usuarios

Matriculaciones

Id	Nombre	Email	Password	Acciones
1	Ali	anclavijo@espe.edu.ec	ali	<button>Editar</button> <button>Eliminar</button>

Formulario de Usuario

Nombre

Alisson

Email

Email

Password

Password

Guardar

Error: Validation failed for object='user', Error count: 2

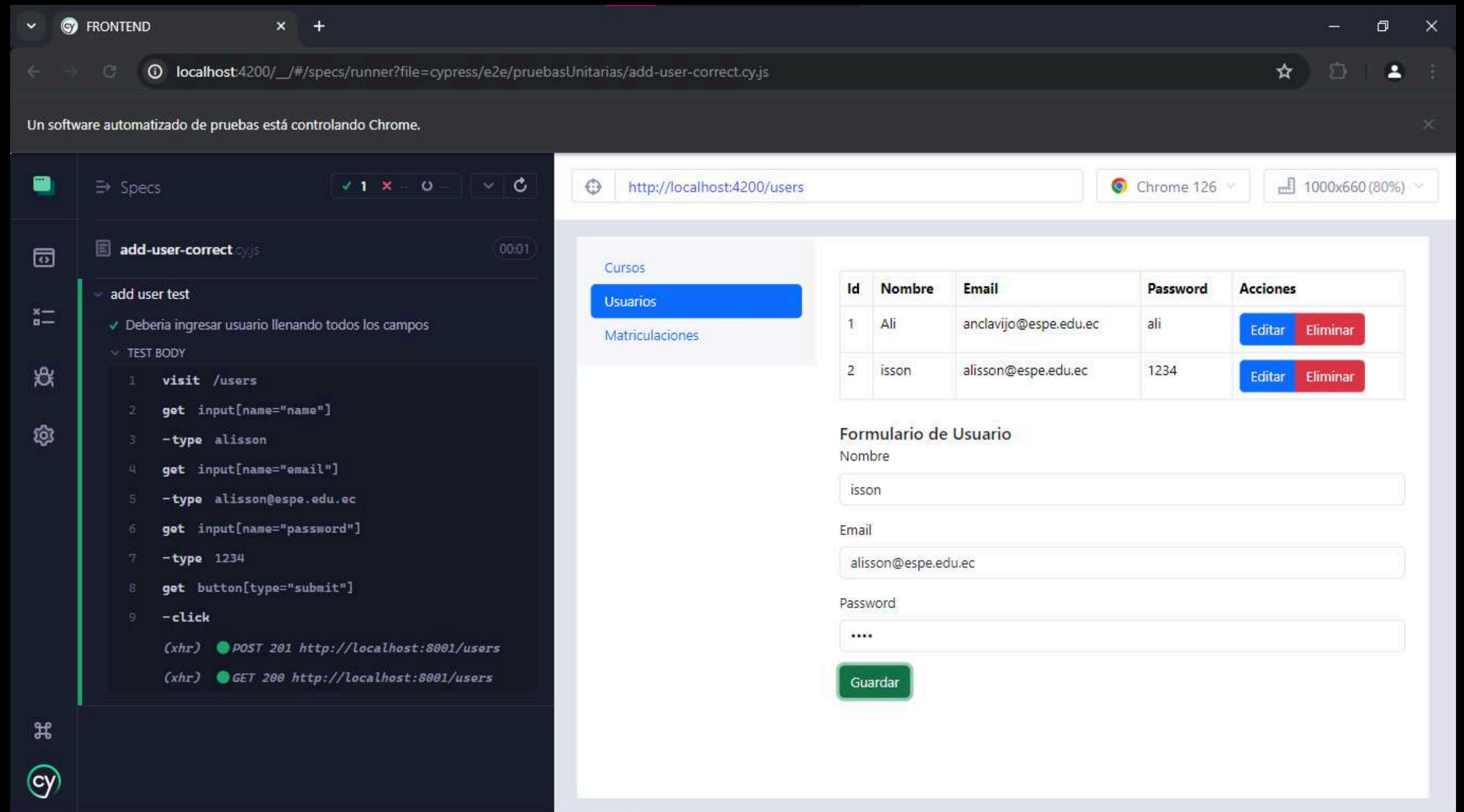
CYPRESS

Escribimos el test, para poder probar que se ingrese un usuario llenando todos los campos.

```
TS cypress.config.ts JS add-user.cy.js JS add-user-correct.cy.js X JS todo.cy.js {} package.json
cypress > e2e > pruebasUnitarias > JS add-user-correct.cy.js > ...
1 describe('add user test', () => {
2   it('Deberia ingresar usuario llenando todos los campos', () => {
3     cy.visit('/users');
4
5     // Ajustar el selector para encontrar el input de nombre correctamente
6     cy.get('input[name="name"]').type('alisson');
7
8     // Ajustar el selector para encontrar el input de email correctamente
9     cy.get('input[name="email"]').type('alisson@espe.edu.ec');
10
11    // Ajustar el selector para encontrar el input de password correctamente
12    cy.get('input[name="password"]').type('1234');
13
14    // Aquí puedes agregar un selector para el botón de envío y hacer clic en él
15    cy.get('button[type="submit"]').click();
16  });
17 });
18
```

CYPRESS

Probara el test
En este caso al llenar
todos los campos , el
usuario es creado con
exito.



The image shows a Cypress test runner interface on the left and a web application on the right. The Cypress runner is running a test named 'add user test' which is passing. The test body includes commands to visit the '/users' endpoint, get input values for name, email, and password, and click the submit button. The application on the right shows a 'Usuarios' (Users) section with a table of users and a 'Formulario de Usuario' (User Form) with fields for name, email, and password, and a 'Guardar' (Save) button.

Cypress Test Runner:

- Specs: 1 passed, 0 failed, 0 pending
- Test Name: add user test
- Test Body:
 - 1 visit /users
 - 2 get input[name="name"]
 - 3 -type alisson
 - 4 get input[name="email"]
 - 5 -type alisson@espe.edu.ec
 - 6 get input[name="password"]
 - 7 -type 1234
 - 8 get button[type="submit"]
 - 9 -click
- Logs:
 - (xhr) POST 201 http://localhost:8001/users
 - (xhr) GET 200 http://localhost:8001/users

Web Application:

http://localhost:4200/users

Cursos

Usuarios

Matriculaciones

Id	Nombre	Email	Password	Acciones
1	Ali	anclavijo@espe.edu.ec	ali	<button>Editar</button> <button>Eliminar</button>
2	isson	alisson@espe.edu.ec	1234	<button>Editar</button> <button>Eliminar</button>

Formulario de Usuario

Nombre

isson

Email

alisson@espe.edu.ec

Password

....

Guardar

CYPRESS

Escribimos el
test, ahora
para cursos.

El test debería
crear
correctamente
un curso.

TS cypress.config.ts

JS add-user.cy.js ●

JS add-course.cy.js X

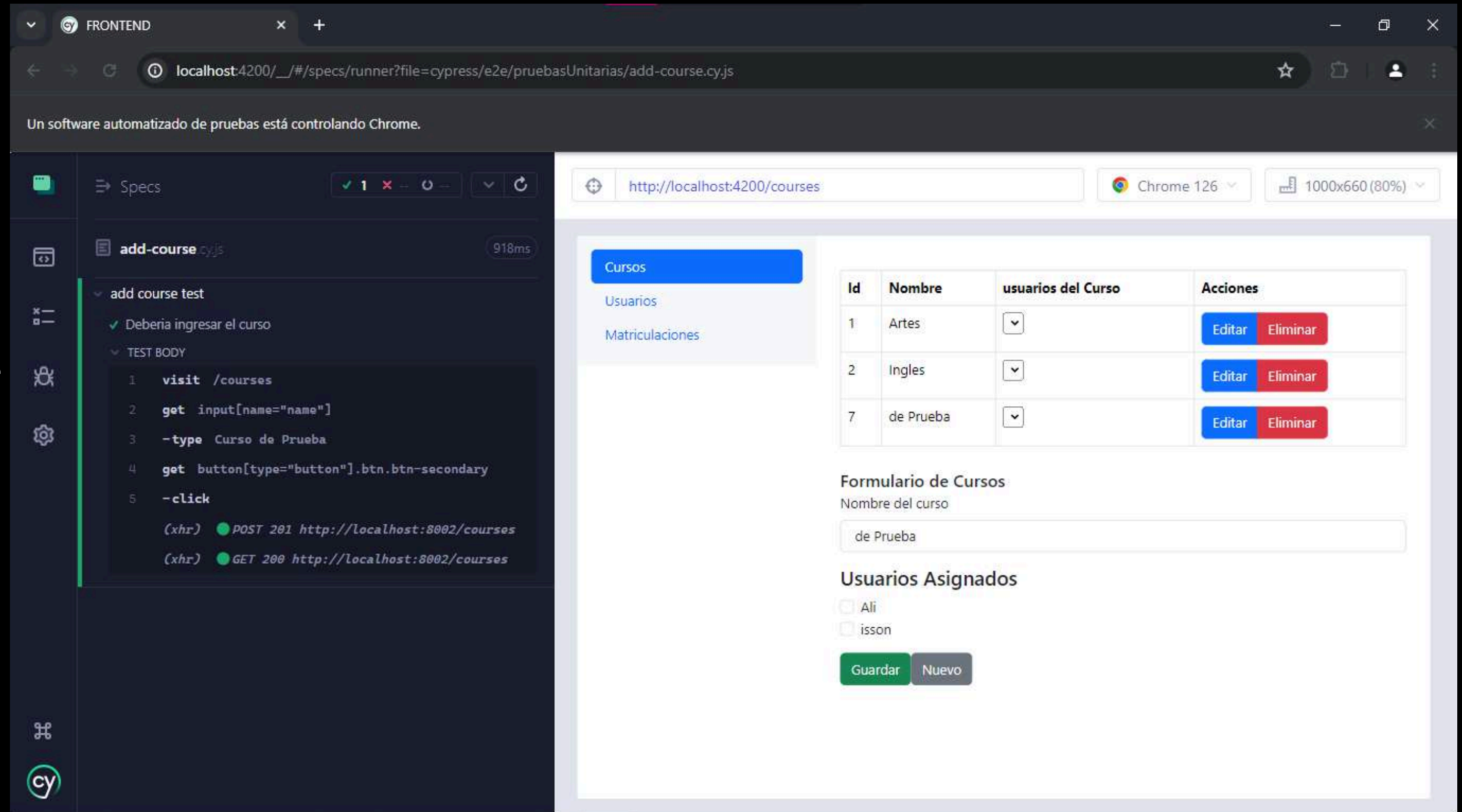
TS courseUser.model.ts

cypress > e2e > pruebasUnitarias > JS add-course.cy.js > describe('add course test') callback

```
1 describe('add course test', () => {
2   it('Deberia ingresar el curso', () => {
3     cy.visit('/courses');
4
5     // Ingresar el nombre del curso
6     cy.get('input[name="name"]').type('Curso de Prueba');
7
8     cy.get('button[type="button"].btn.btn-secondary').click();
9
10  });
11 });
12
```

CYPRESS

Se crea
correctamente
el curso.



The image shows a Cypress test runner interface on the left and a web application on the right. The test runner is running a test named 'add course test' which includes a 'visit' command and a 'click' command. The test results show a successful POST request to 'http://localhost:8002/courses' and a successful GET request to 'http://localhost:8002/courses'. The web application on the right displays a list of courses with columns for Id, Nombre, usuarios del Curso, and Acciones. The 'Cursos' tab is selected, showing a table with 3 rows. Below the table is a 'Formulario de Cursos' section with a text input field containing 'de Prueba'. At the bottom, there is a 'Usuarios Asignados' section with two checkboxes, 'Ali' and 'isson', and two buttons, 'Guardar' and 'Nuevo'.

Un software automatizado de pruebas está controlando Chrome.

Specs

add-course.cy.js

add course test

✓ Deberia ingresar el curso

TEST BODY

```
1 visit /courses
2 get input[name="name"]
3 -type Curso de Prueba
4 get button[type="button"].btn.btn-secondary
5 -click
```

(xhr) POST 201 http://localhost:8002/courses

(xhr) GET 200 http://localhost:8002/courses

http://localhost:4200/courses

Chrome 126

1000x660 (80%)

Cursos

Usuarios

Matriculaciones

Id	Nombre	usuarios del Curso	Acciones
1	Artes	▼	Editar Eliminar
2	Ingles	▼	Editar Eliminar
7	de Prueba	▼	Editar Eliminar

Formulario de Cursos

Nombre del curso

de Prueba

Usuarios Asignados

☐ Ali

☐ isson

Guardar Nuevo

CYPRESS

Escribimos el
test, ahora
para matricula.

El test debería
registrar
correctamente
al estudiante
en el curso.

```
TS cypress.config.ts JS add-user.cy.js JS add-course.cy.js JS add-matricula.cy.js X
cypress > e2e > pruebasUnitarias > JS add-matricula.cy.js > ...
1 describe('add course test', () => {
2   it('Deberia seleccionar y matricular al estudiante en el curso', () => {
3     cy.visit('/enrollment');
4
5     // Seleccionar un usuario
6     cy.get('select#userSelect').select('Ali');
7
8     // Seleccionar un curso
9     cy.get('select#courseSelect').select('Artes');
10
11    // Hacer clic en el botón de matricular
12    cy.get('button.btn.btn-primary.mt-3').click();
13
14  });
15 });
16
```


CYPRESS

Se realiza
correctamente
la matricula

The image shows a Cypress test runner interface on the left and a web application on the right. The Cypress runner is displaying a test suite named 'add-matricula.cy.js' with a single test 'add course test' that passed. The test body includes commands for visiting the enrollment page, selecting a user and course, and clicking the enroll button. The application on the right shows the 'Matricular Estudiante' form with 'Ali' selected as the user and 'Artes' as the course. A success message 'User enrolled successfully!' is displayed at the bottom of the form.

Un software automatizado de pruebas está controlando Chrome.

Specs

add-matricula.cy.js

add course test

✓ Deberia seleccionar y matricular al estudiante en el curso

TEST BODY

```
1 visit /enrollment
2 get select#userSelect
3 -select Ali
4 get select#courseSelect
5 -select Artes
6 get button.btn.btn-primary.mt-3
7 -click

(xhr) PUT 200 http://localhost:8002/courses/8/assign-user/1
```

http://localhost:4200/enrollment

Chrome 126

1000x660 (64%)

Cursos

Usuarios

Matriculaciones

Matricular Estudiante

Selecciona un usuario

Ali

Selecciona un curso

Artes

Enroll

User enrolled successfully!

CYPRESS

En caso de que el estudiante ya este matriculado en el curso y se vuelva a realizar la prueba nos reflejara que no se puede realizar la matricula


The screenshot displays the Cypress test runner interface. The top bar shows the browser window with the URL `localhost:4200/_/#/specs/runner?file=cypress/e2e/pruebasUnitarias/add-matricula.cy.js`. Below the browser window, a message states: "Un software automatizado de pruebas está controlando Chrome."

The left sidebar shows the "Specs" panel with the file `add-matricula.cy.js` selected. The test suite `add course test` is expanded, showing a single test: "Deberia seleccionar y matricular al estudiante en el curso". The test body is visible, showing the following commands:

```
1 visit /enrollment
2 get select#userSelect
3 -select Ali
4 get select#courseSelect
5 -select Artes
6 get button.btn.btn-primary.mt-3
7 -click
```

The test result shows a failure: `(xhr) PUT 500 http://localhost:8002/courses/8/assign-user/1`.

The right sidebar shows the web application state. The URL is `http://localhost:4200/enrollment`. The page title is "Matricular Estudiante". The form includes two dropdown menus: "Selecciona un usuario" (with "Ali" selected) and "Selecciona un curso" (with "Artes" selected). Below the dropdowns is an "Enroll" button. A light blue error message box at the bottom of the form states: "An unexpected error occurred."



CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- La arquitectura de microservicios, dividida en servicios de usuarios y cursos, permite una alta cohesión y baja dependencia, facilitando el mantenimiento y la escalabilidad del sistema.
- El uso de Feign para la comunicación entre microservicios asegura una integración robusta y eficiente.

RECOMENDACIONES

- Continuar optimizando la seguridad y el rendimiento del sistema para garantizar la protección de los datos y una experiencia de usuario fluida.
- Mantener un enfoque en la escalabilidad y flexibilidad del sistema para adaptarse a futuras necesidades y cambios en el entorno educativo.

**GRACIAS POR
SU ATENCIÓN**

GRUPO 2