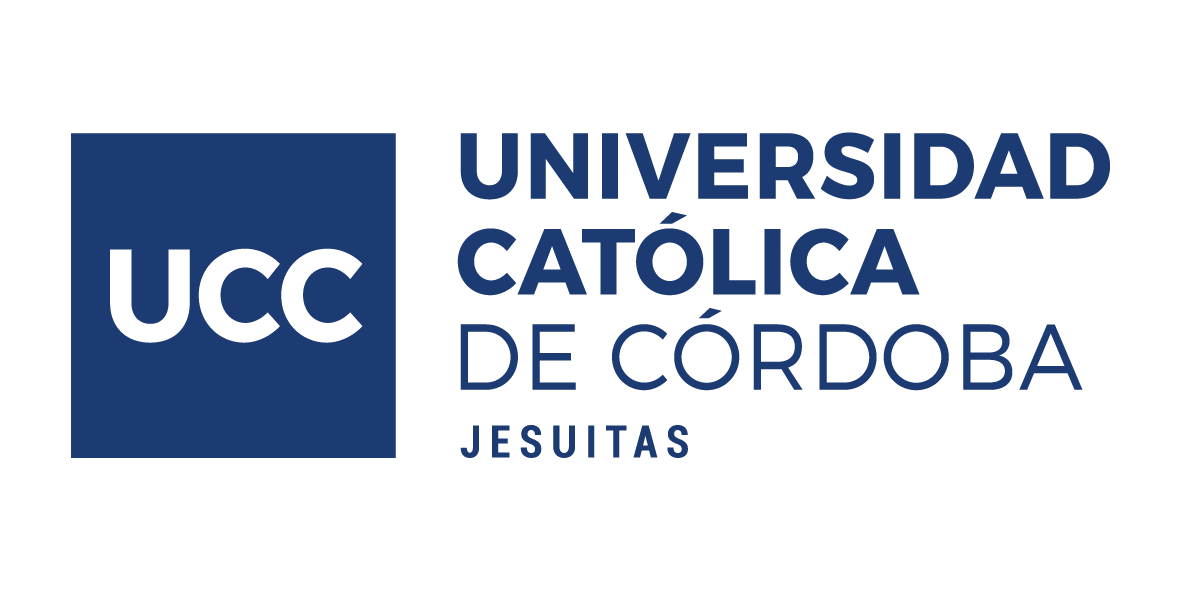
### **Trabajo Práctico N°6**

Alumno:

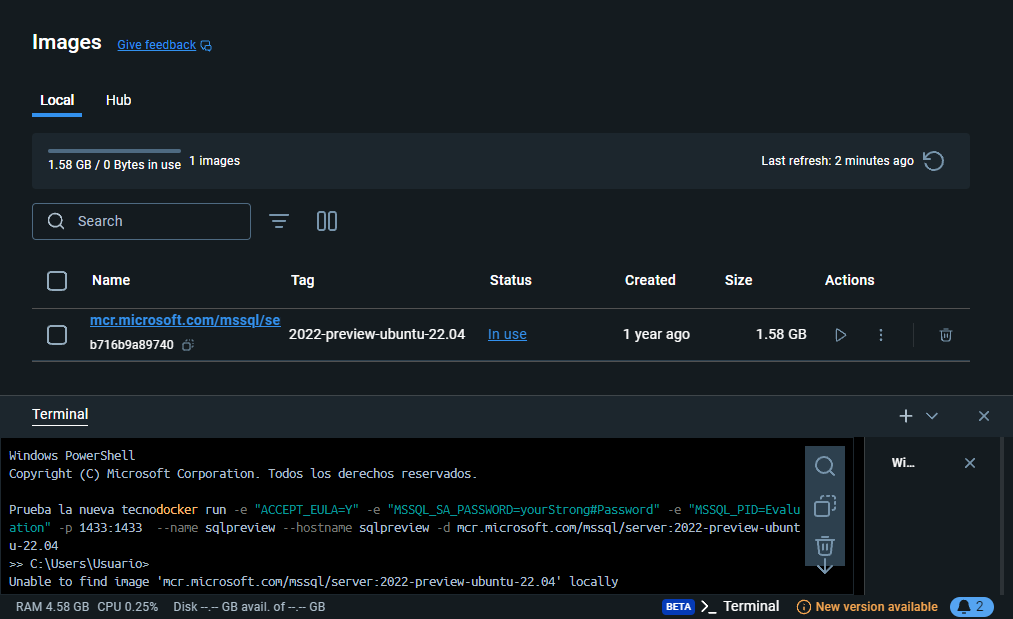
* Miron Juan Pablo (2100815).

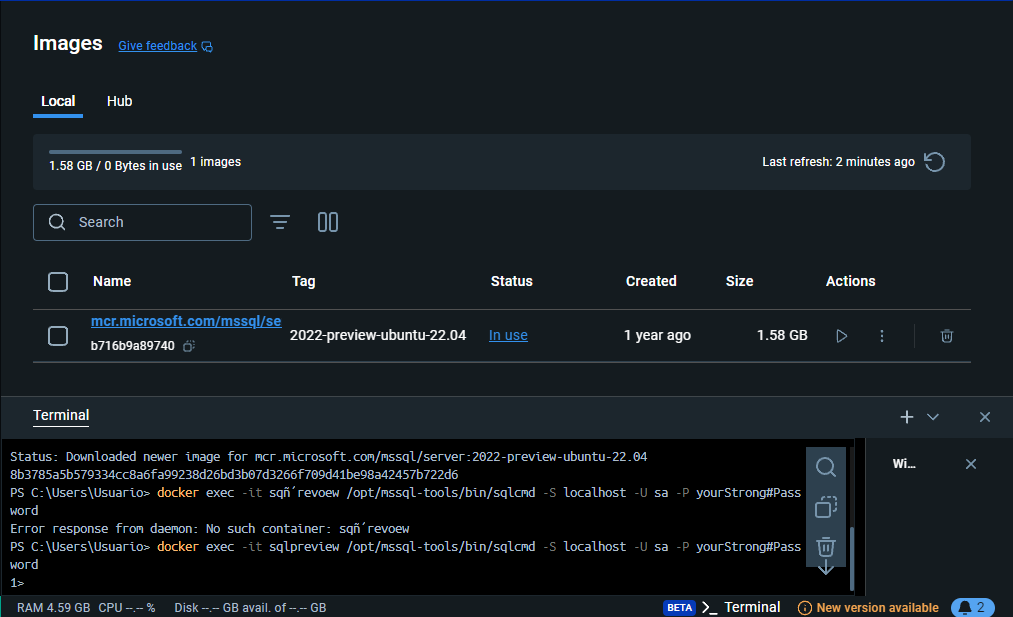
Materia: Ingeniería de Software III.

Docente: Schwindt Ariel.

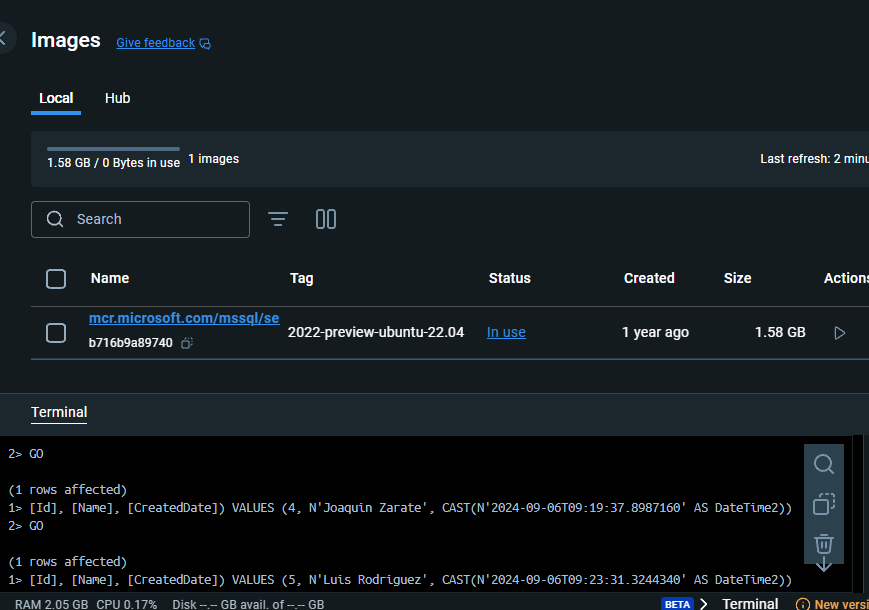


1. 1. Crear una BD Azure SQL Database (Ver Instructivo 5.1) o montar una imagen Docker de SQL Server como se solicitó en el punto 12 del [TP02]. (<https://github.com/ingsoft3ucc/TPS_2024/blob/main/trabajos/02-introduccion-docker.md>).

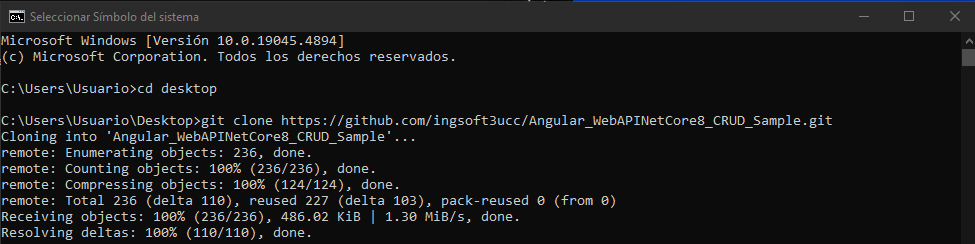




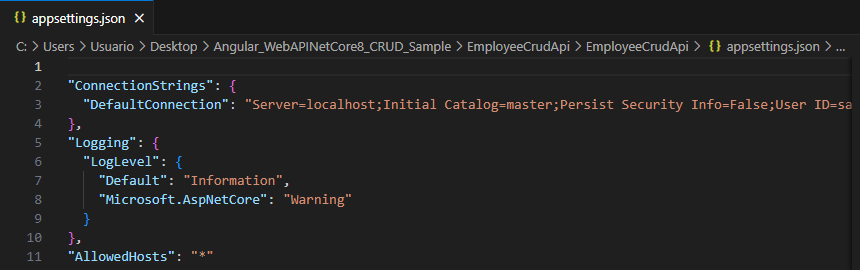
* 1. En caso de optar por la opción de montar la imagen de docker, una vez levantada el contenedor, conectarse y ejecutar el script.

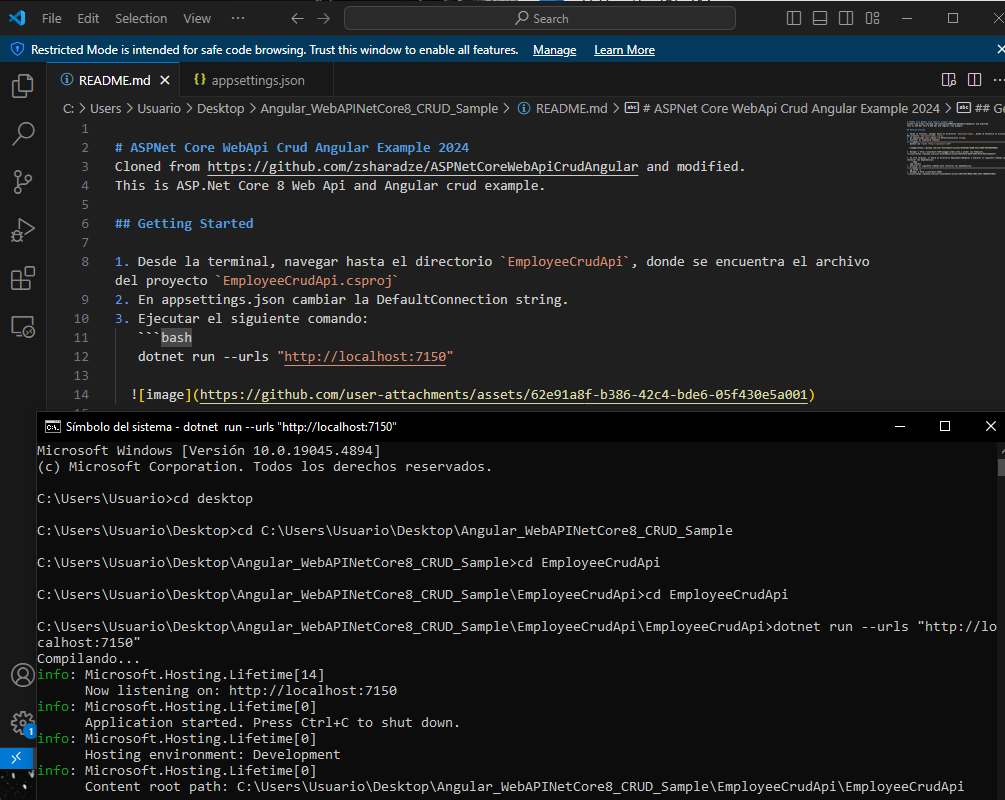


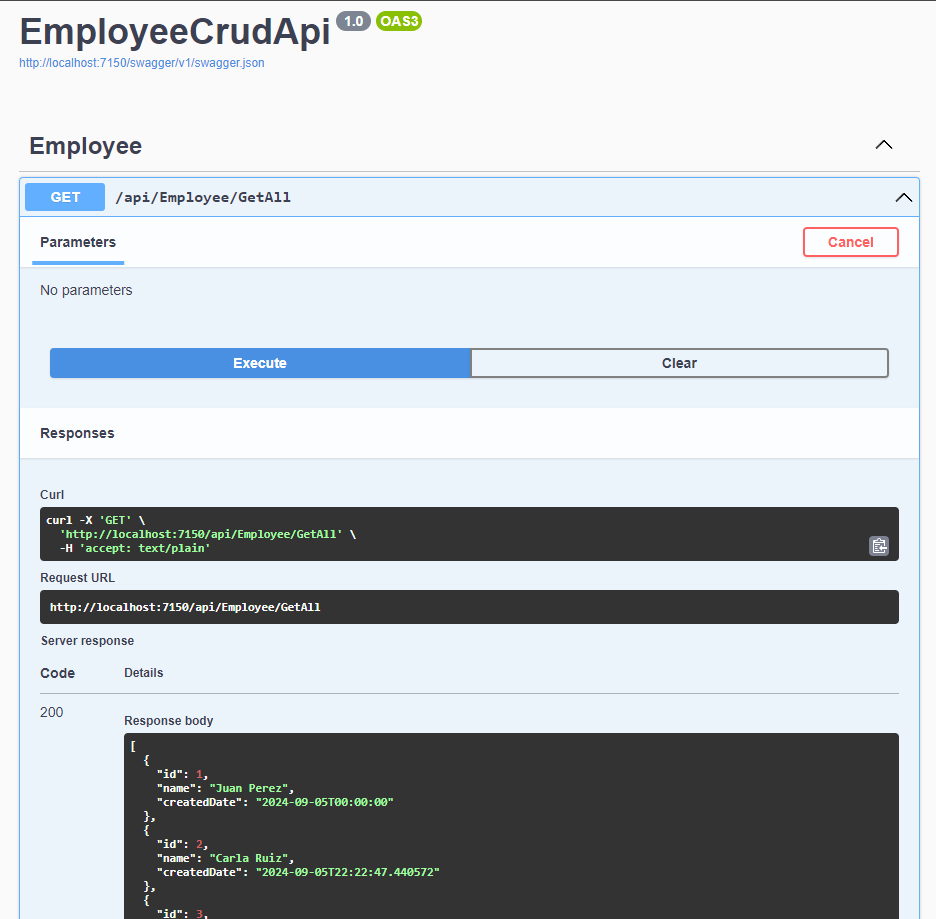
1. 1. Clonar el repo <https://github.com/ingsoft3ucc/Angular_WebAPINetCore8_CRUD_Sample.git>

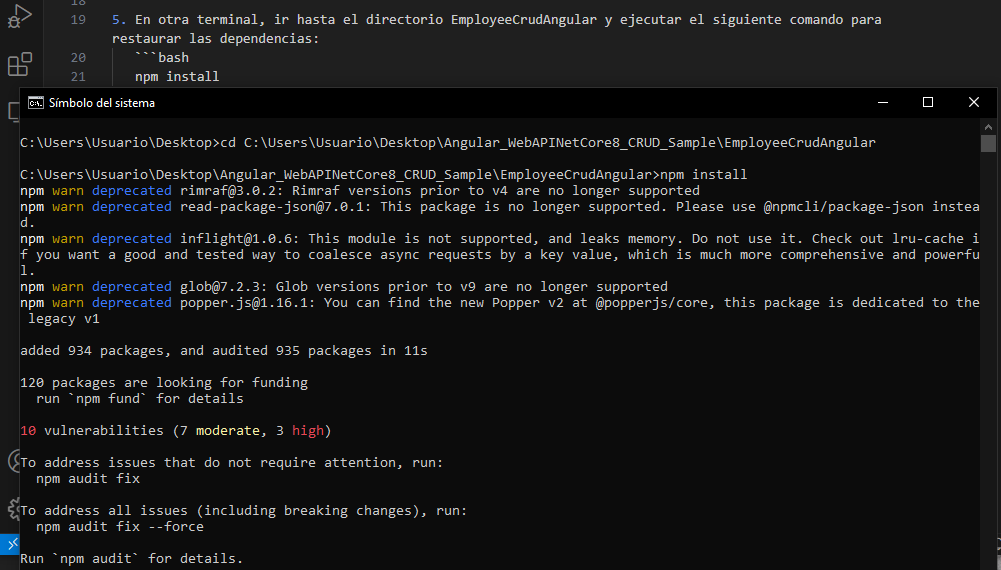


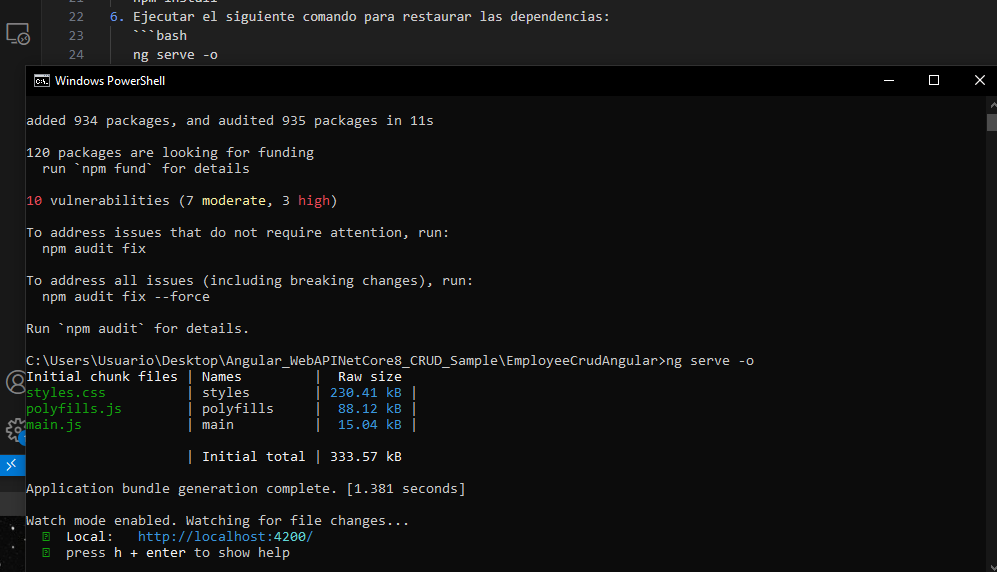
* 1. Seguir las instrucciones del README.md del repo clonado prestando atención a la modificación de la cadena de conexión en el appSettings.json para que apunte a la BD creada en 4.1

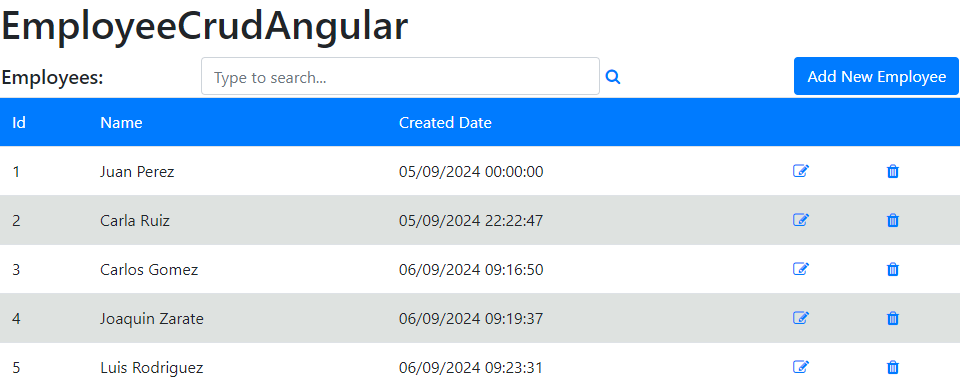




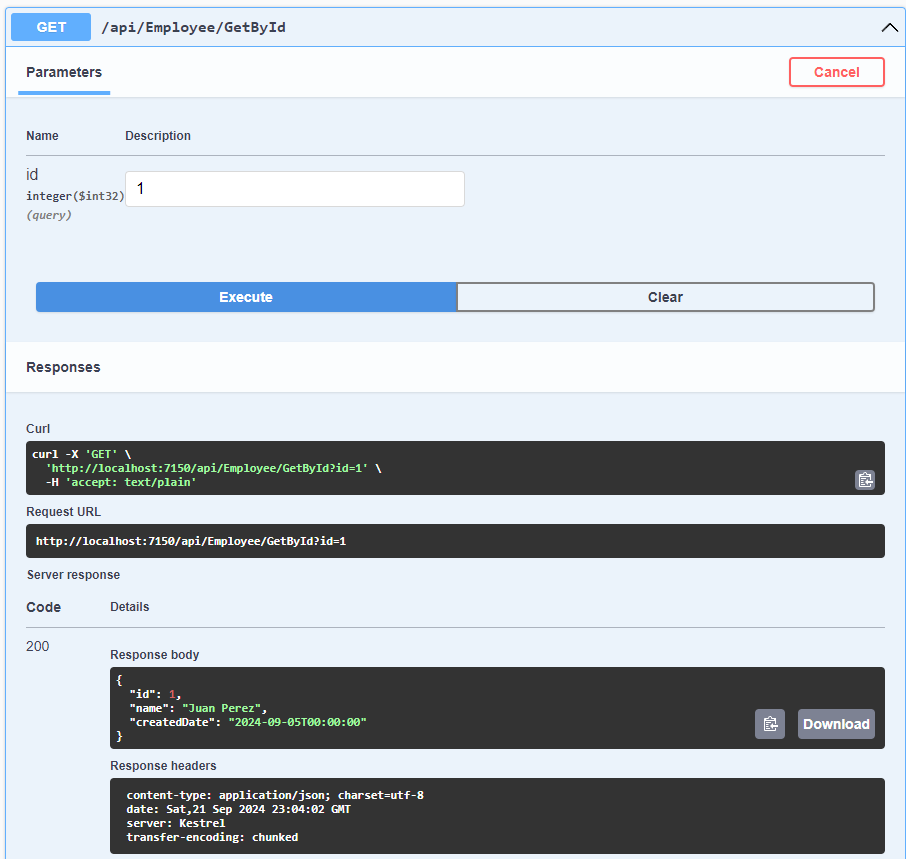




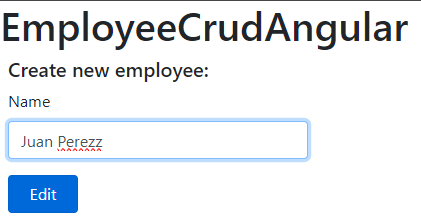


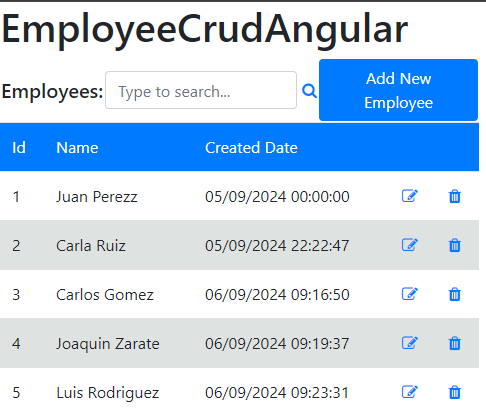


* 1. Navegar a http://localhost:7150/swagger/index.html y probar uno de los controladores para verificar el correcto funcionamiento de la API.



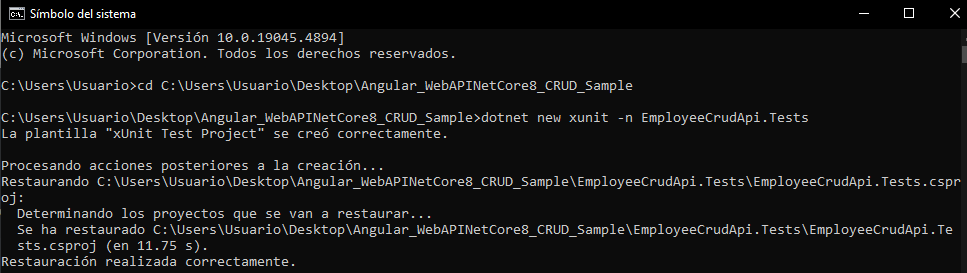
* 1. Navegar a http://localhost:4200 y verificar el correcto funcionamiento de nuestro front-end Angular.



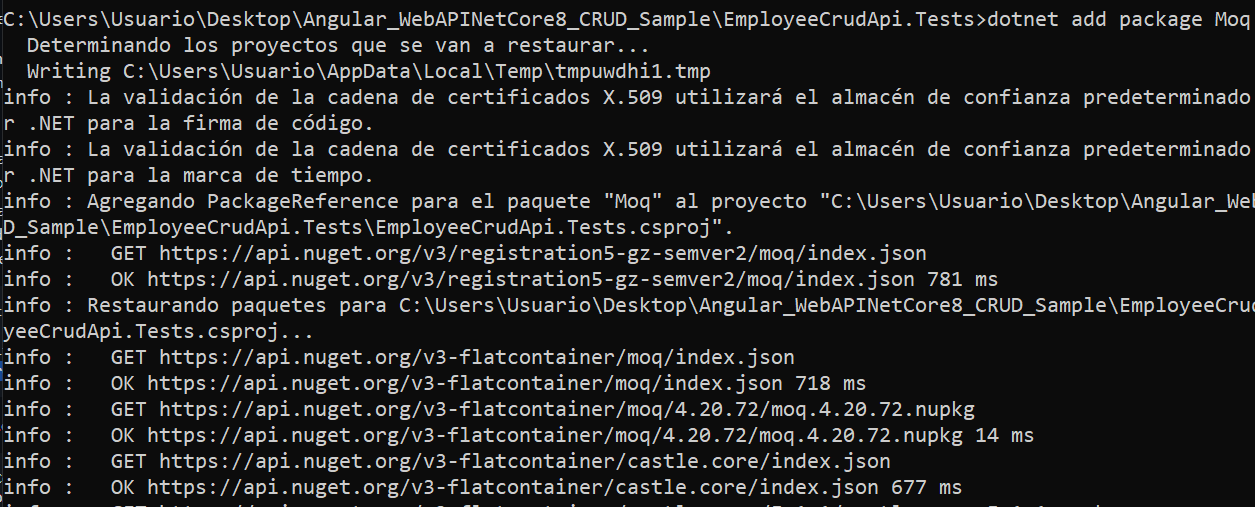


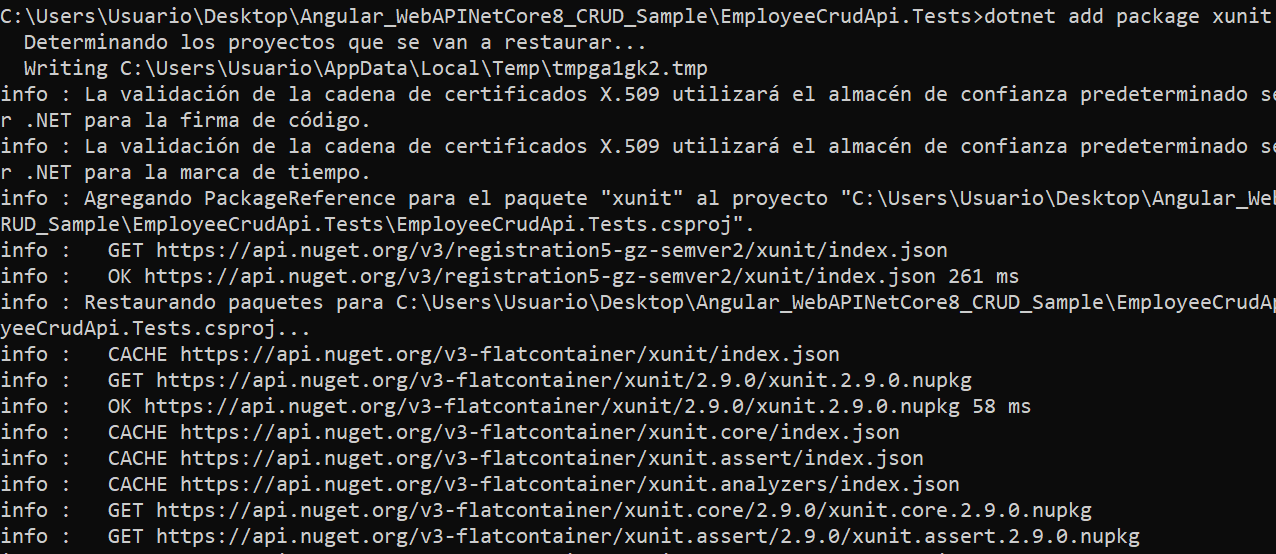
* 1. Una vez verificado el correcto funcionamiento de la Aplicación procederemos a crear un proyecto de pruebas unitarias para nuestra API.

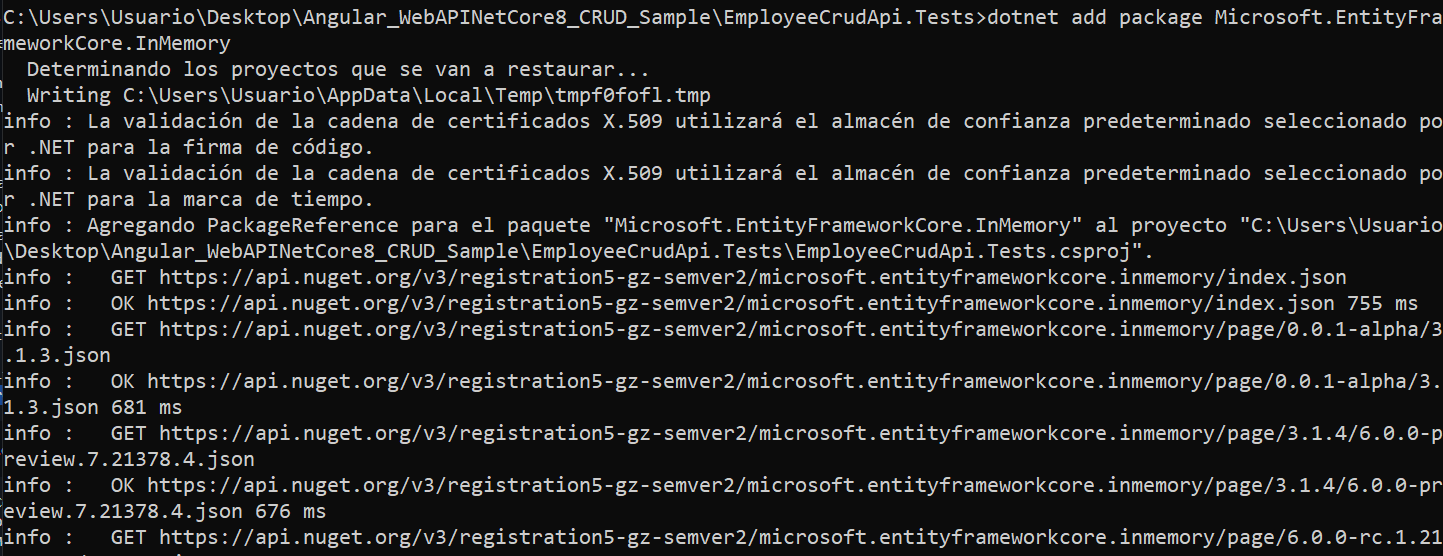
1. 1. En el directorio raíz de nuestro repo crear un nuevo proyecto de pruebas unitarias para nuestra API.



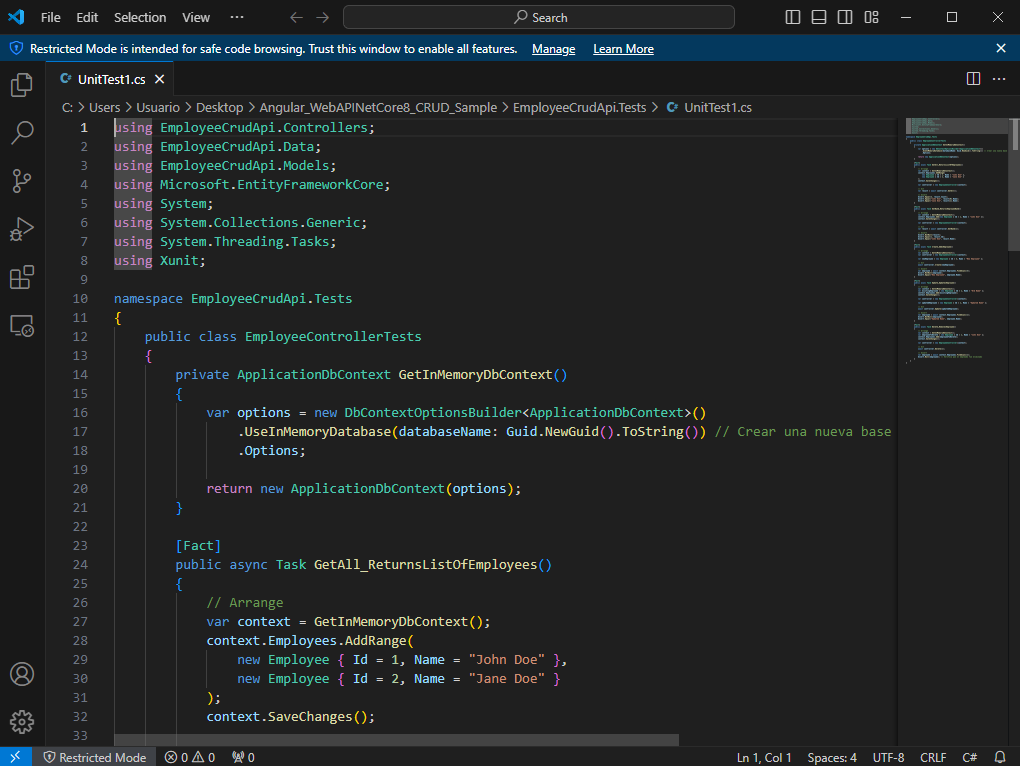
* 1. Instalar dependencias necesarias.



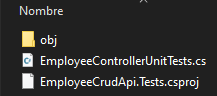




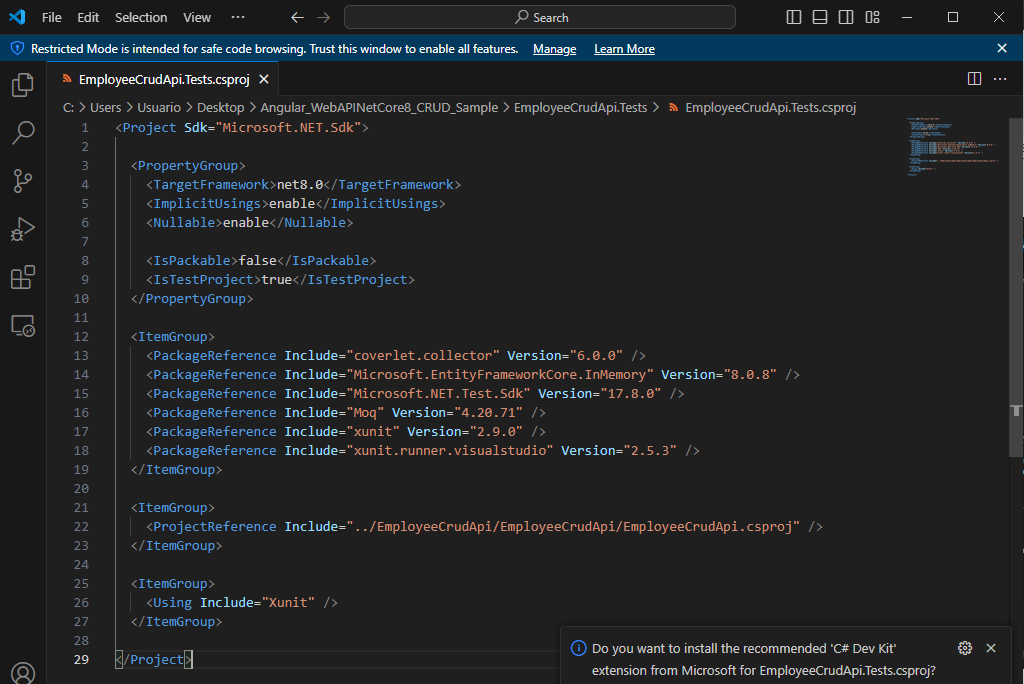
* 1. Editar archivo UnitTest1.cs reemplazando su contenido.



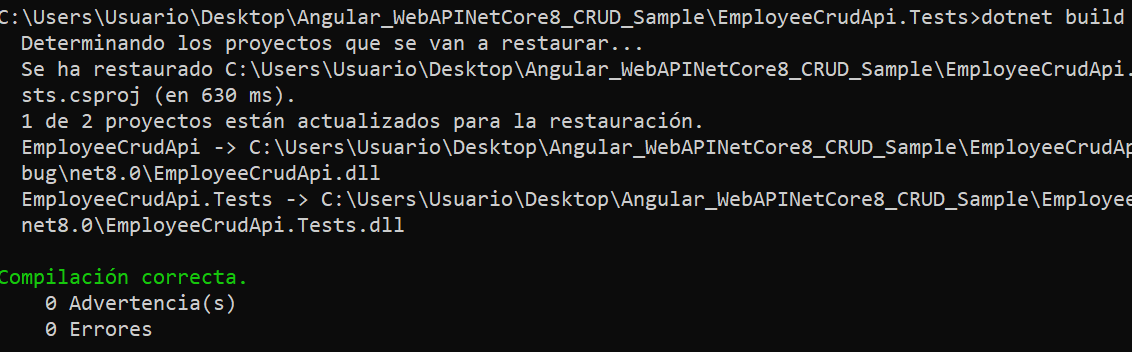
* 1. Renombrar archivo UnitTest1.cs por EmployeeControllerUnitTests.cs.

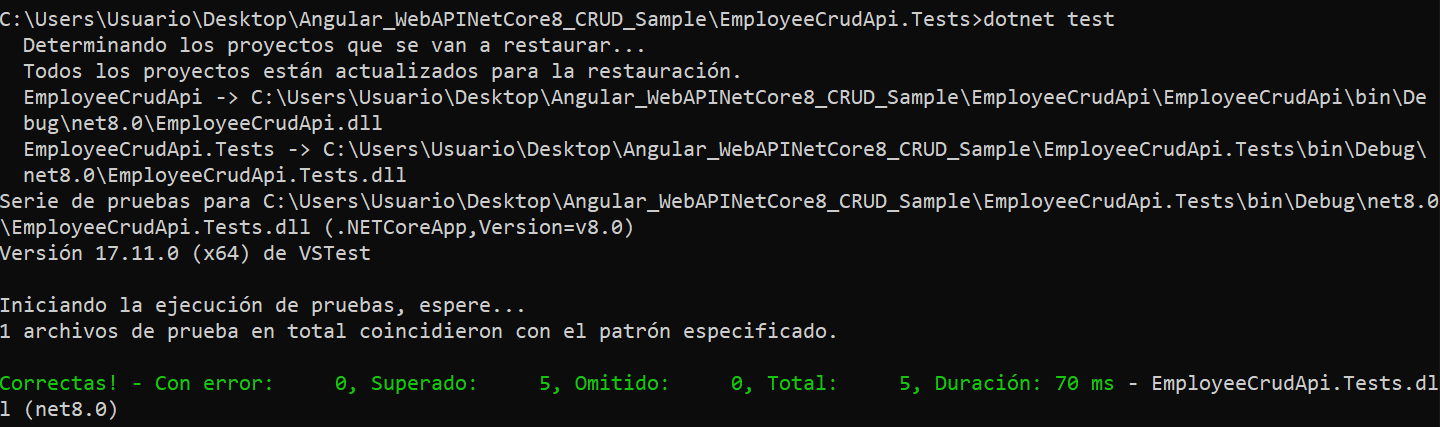


* 1. Editar el archivo EmployeeCrudApi.Tests/EmployeeCrudApi.Tests.csproj para agregar una referencia a nuestro proyecto de EmployeeCrudApi reemplazando su contenido.



* 1. Ejecutar los siguientes comandos para ejecutar nuestras pruebas.
  2. Verificar que se hayan ejecutado correctamente las pruebas.

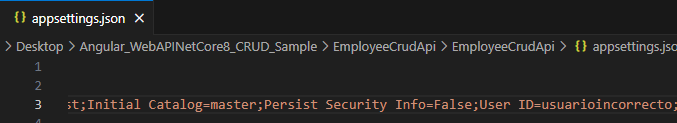


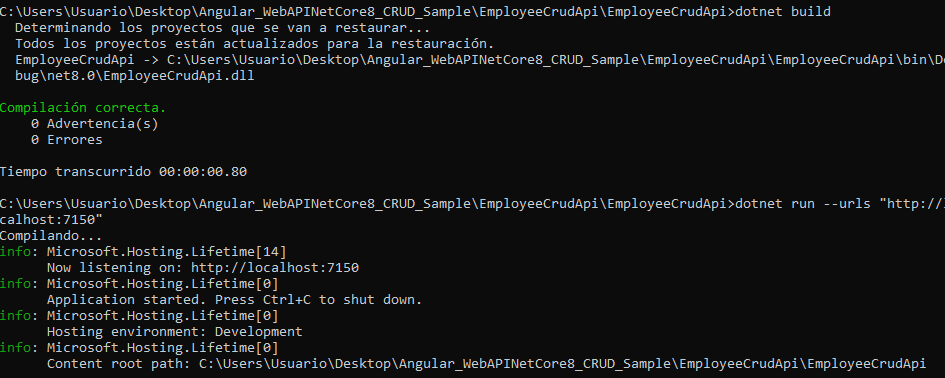


* 1. Verificar que no estamos usando una dependencia externa como base de datos.

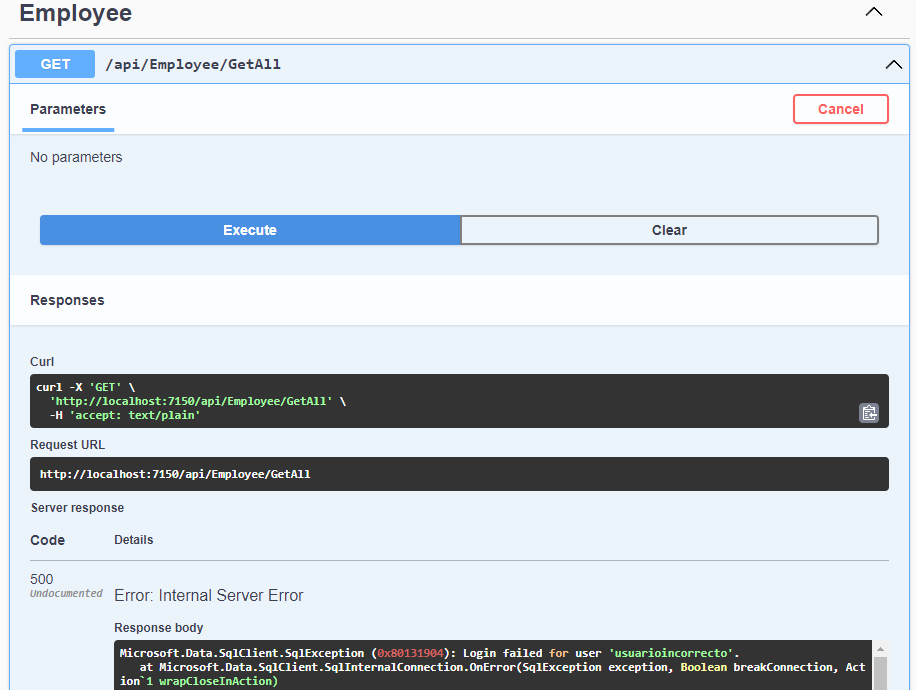
Esto se verifica por el hecho de que los datos cargados en la base de datos son distintos a los utilizados en el moqueo.

* 1. Modificar la cadena de conexión en el archivo appsettings.json para que use un usuario o password incorrecto y recompilar el proyecto EmployeeCrudApi.

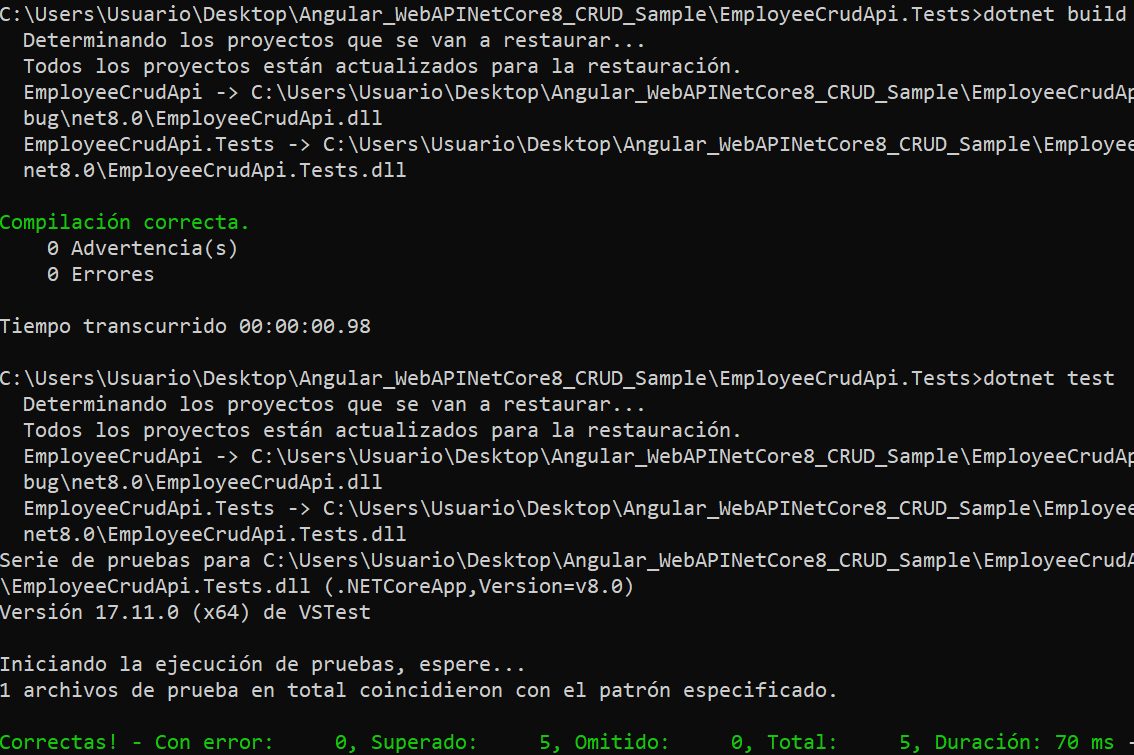




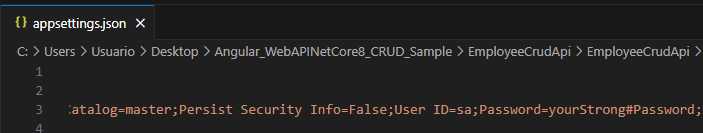
* 1. Verificar que nuestro proyecto ya no tiene acceso a la BD navegando a http://localhost:7150/swagger/index.html y probando uno de los controladores.

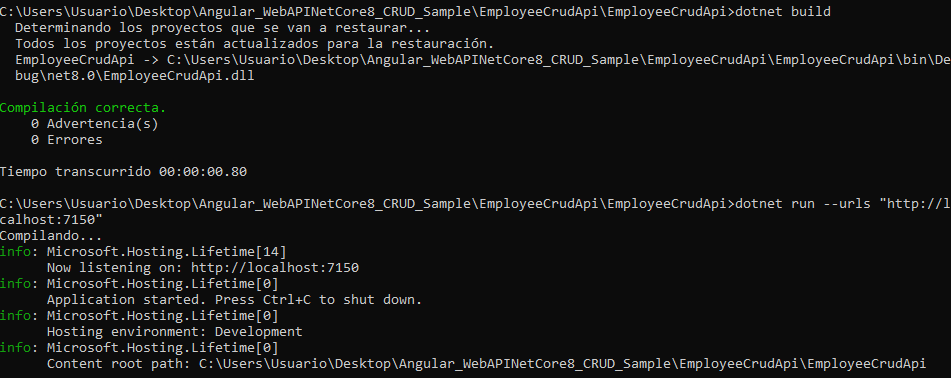


* 1. En la carpeta de nuestro proyecto EmployeeCrudApi.Tests volver a correr las pruebas.
  2. Verificar que se hayan ejecutado correctamente las pruebas inclusive sin tener acceso a la BD, lo que confirma que es efectivamente un conjunto de pruebas unitarias que no requieren de una dependencia externa para funcionar.

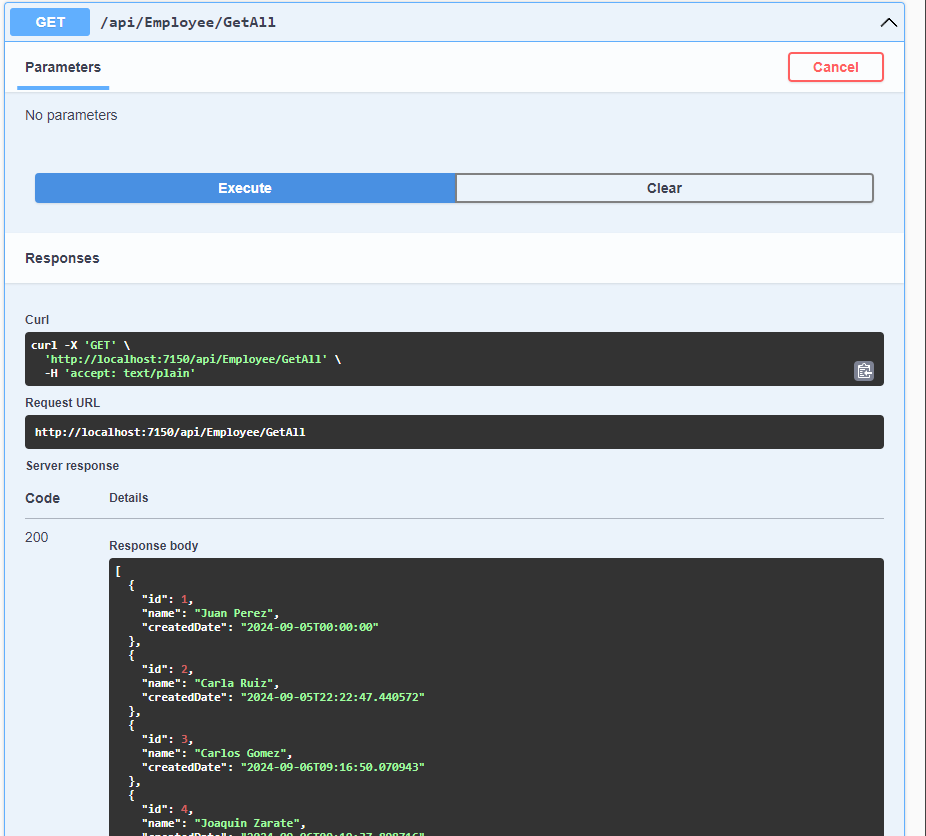


* 1. Modificar la cadena de conexión en el archivo appsettings.json para que use el usuario y password correcto y recompilar el proyecto EmployeeCrudApi.

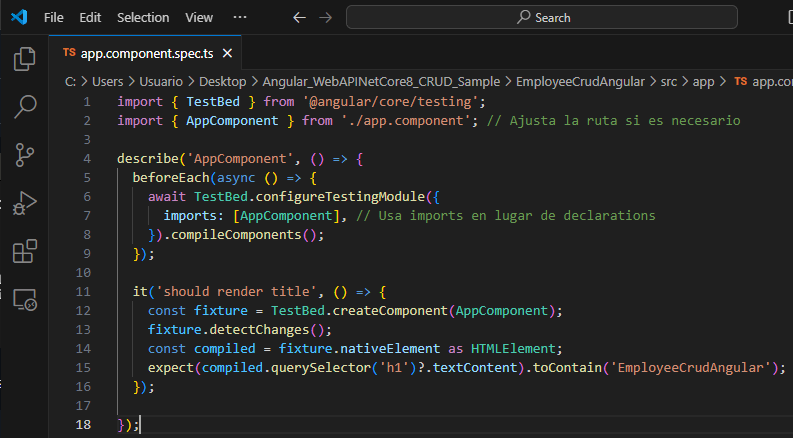




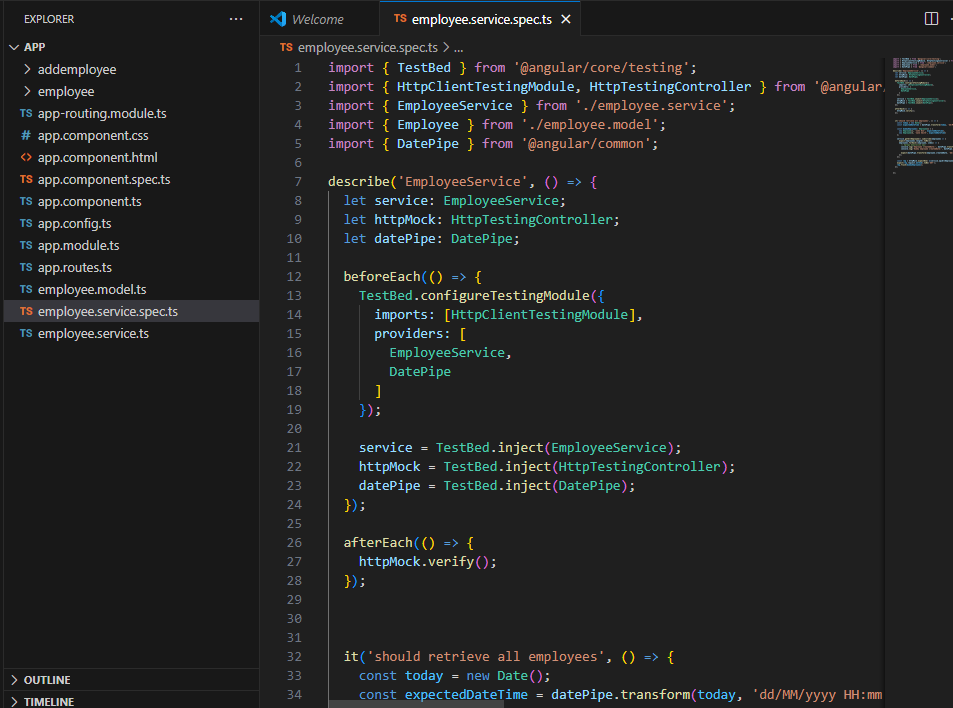
* 1. Verificar que nuestro proyecto vuelve a tener acceso a la BD navegando a http://localhost:7150/swagger/index.html y probando uno de los controladores.



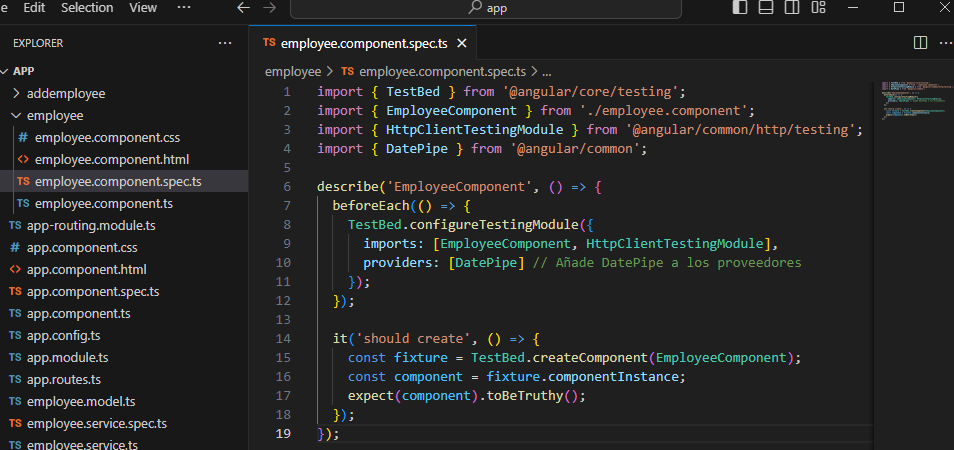
1. 1. Nos posicionamos en nuestro proyecto de front, en el directorio EmployeeCrudAngular/src/app.
   2. Editamos el archivo app.component.spec.ts reemplazando su contenido.



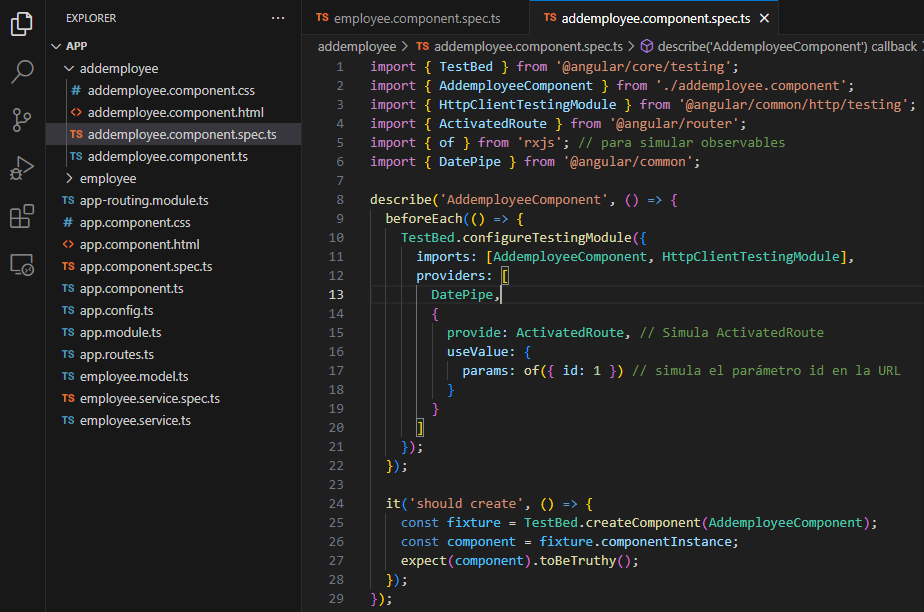
* 1. Creamos el archivo employee.service.spec.ts reemplazando su contenido.



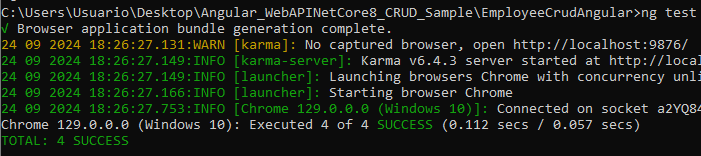
* 1. Editamos el archivo employee.component.spec.ts ubicado en la carpeta employee reemplazando su contenido.



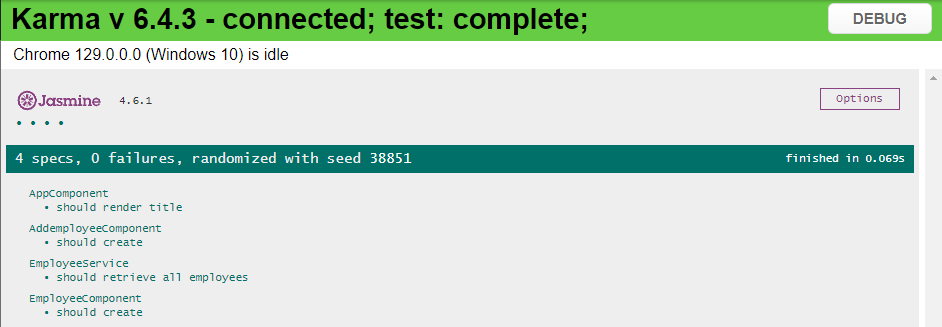
* 1. Editamos el archivo addemployee.component.spec.ts ubicado en la carpeta addemployee reemplazando su contenido.



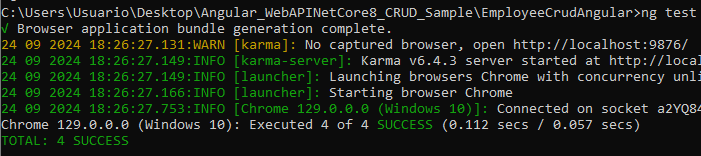
* 1. En el directorio raíz de nuestro proyecto EmployeeCrudAngular ejecutamos el comando.



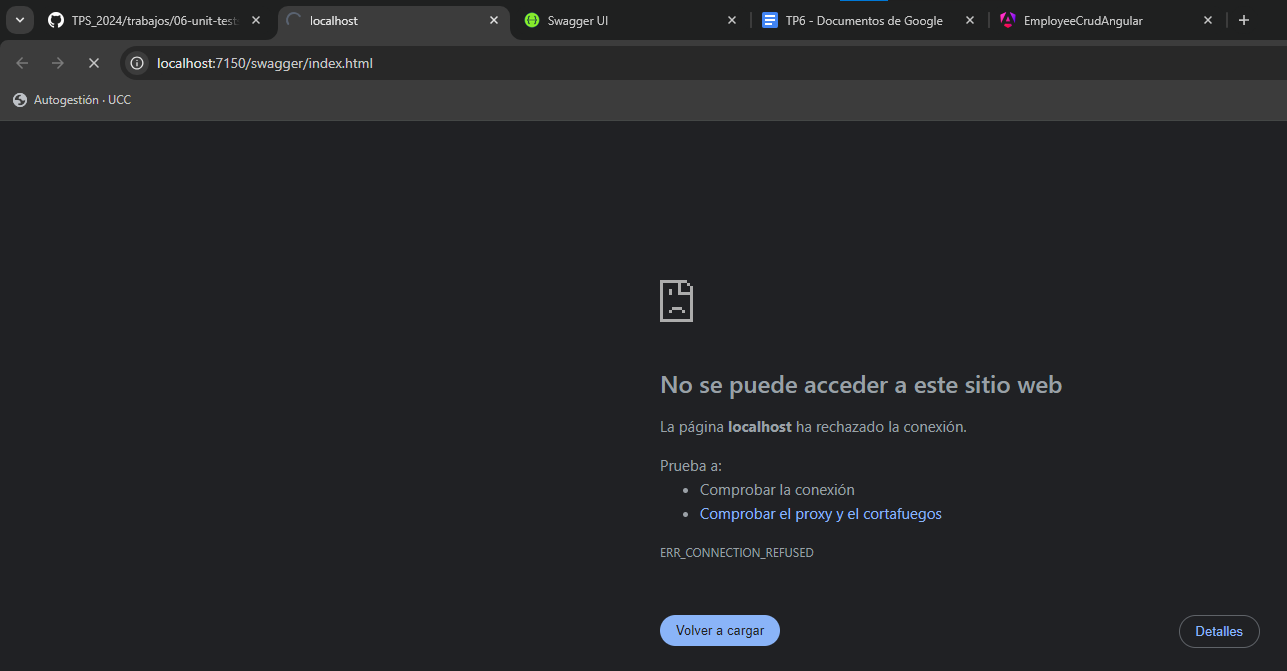
* 1. Vemos que se abre una ventana de Karma con Jasmine en la que nos indica que los tests se ejecutaron correctamente.



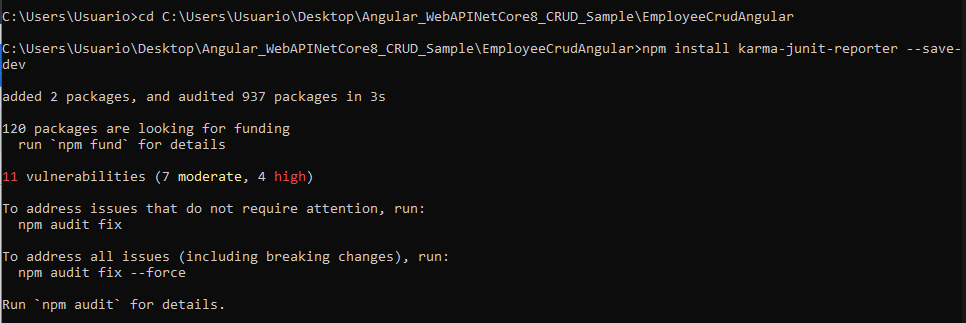
* 1. Vemos que los tests se ejecutaron correctamente.



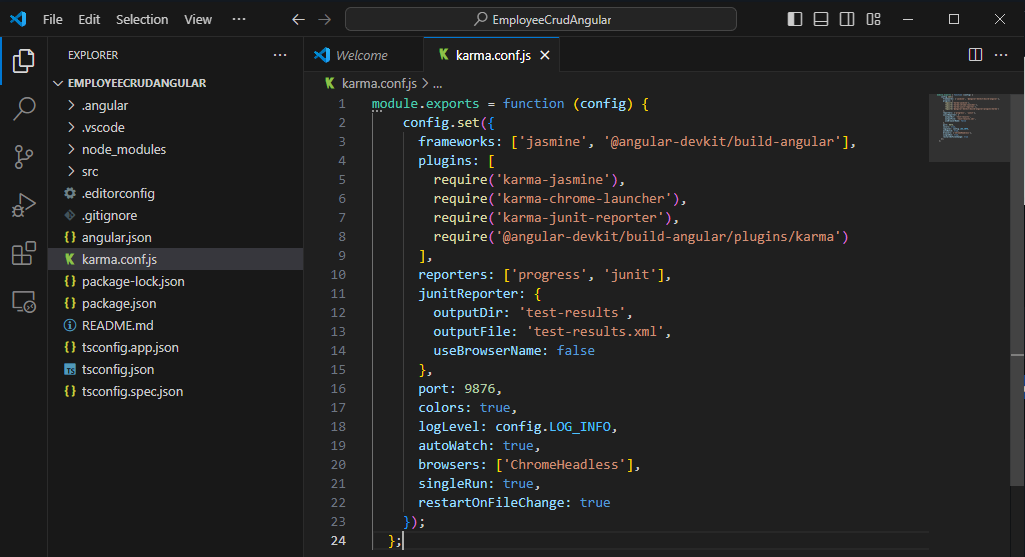
* 1. Verificamos que no esté corriendo nuestra API navegando a http://localhost:7150/swagger/index.html y recibiendo esta salida.



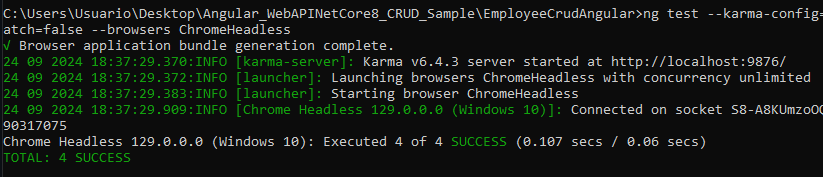
1. 1. Instalamos dependencia karma-junit-reporter.



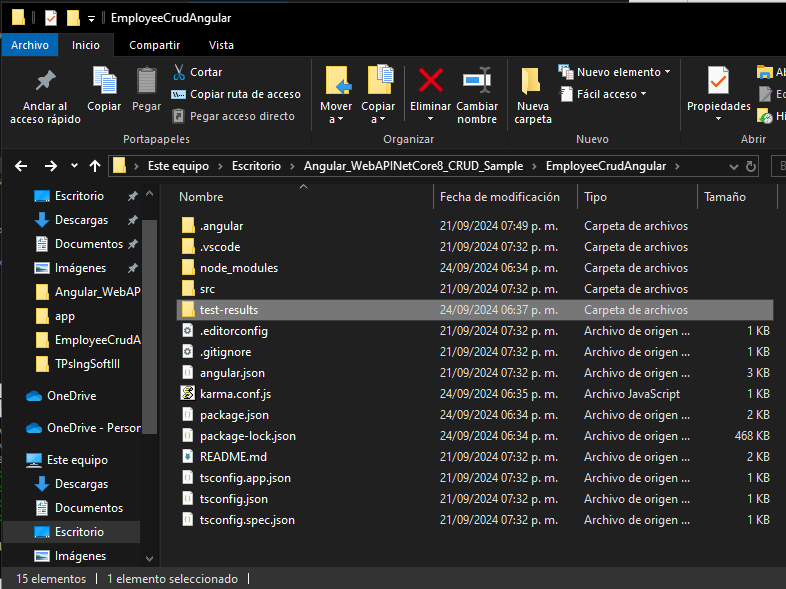
* 1. En el directorio raíz de nuestro proyecto (al mismo nivel que el archivo angular.json) creamos un archivo karma.conf.js con el siguiente contenido.

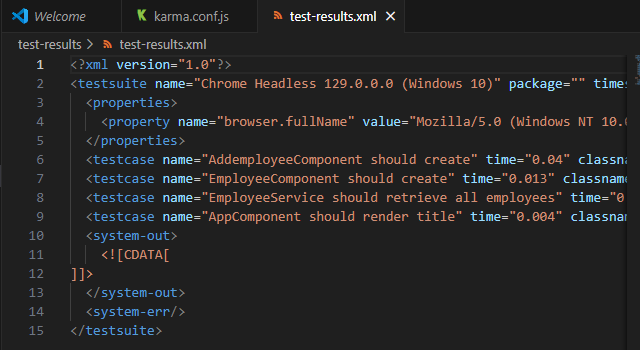


* 1. Ejecutamos nuestros test de la siguiente manera.

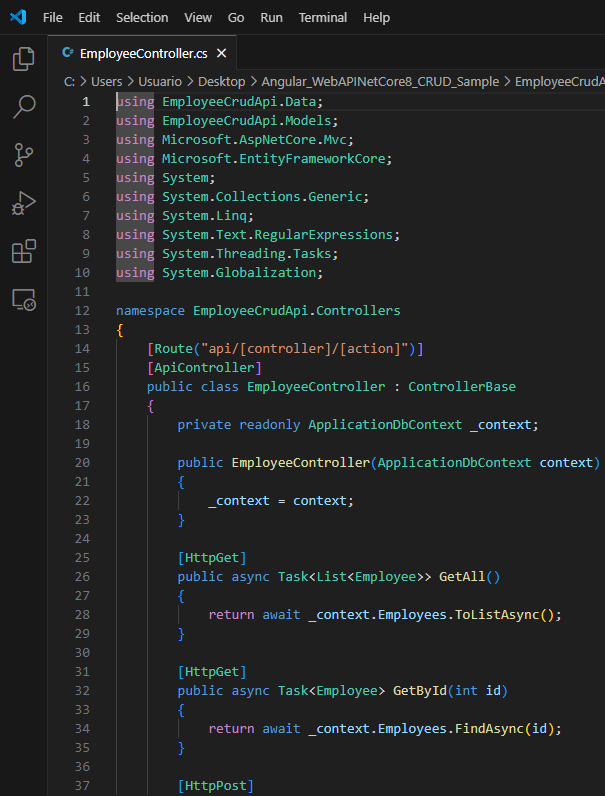


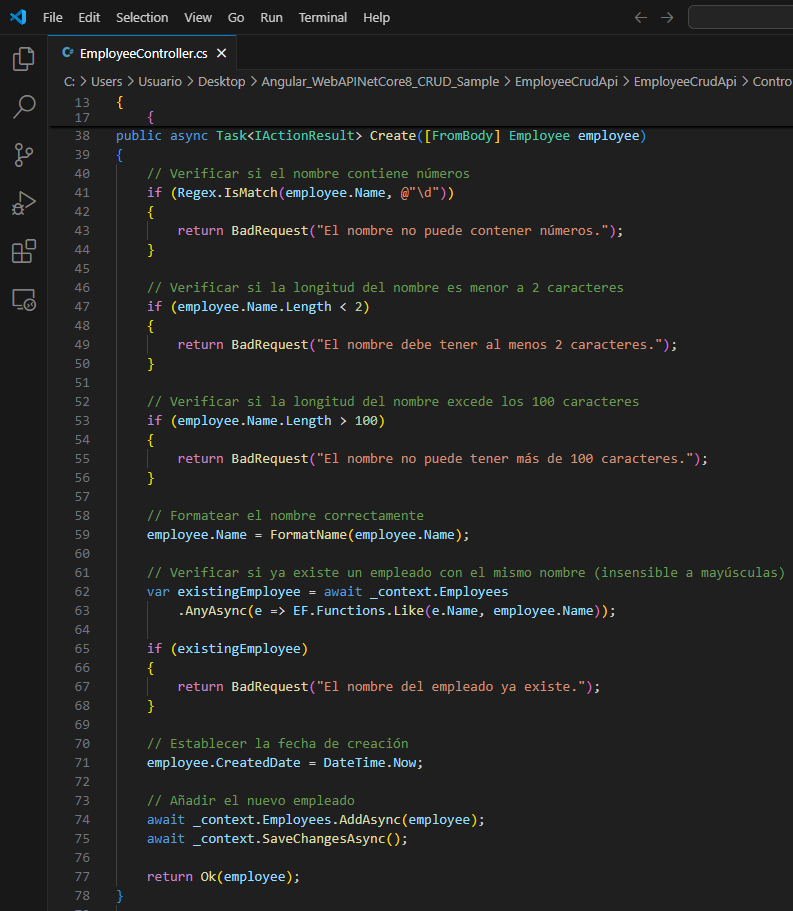
* 1. Verificamos que se creó un archivo test-result.xml en el directorio test-results que está al mismo nivel que el directorio src.

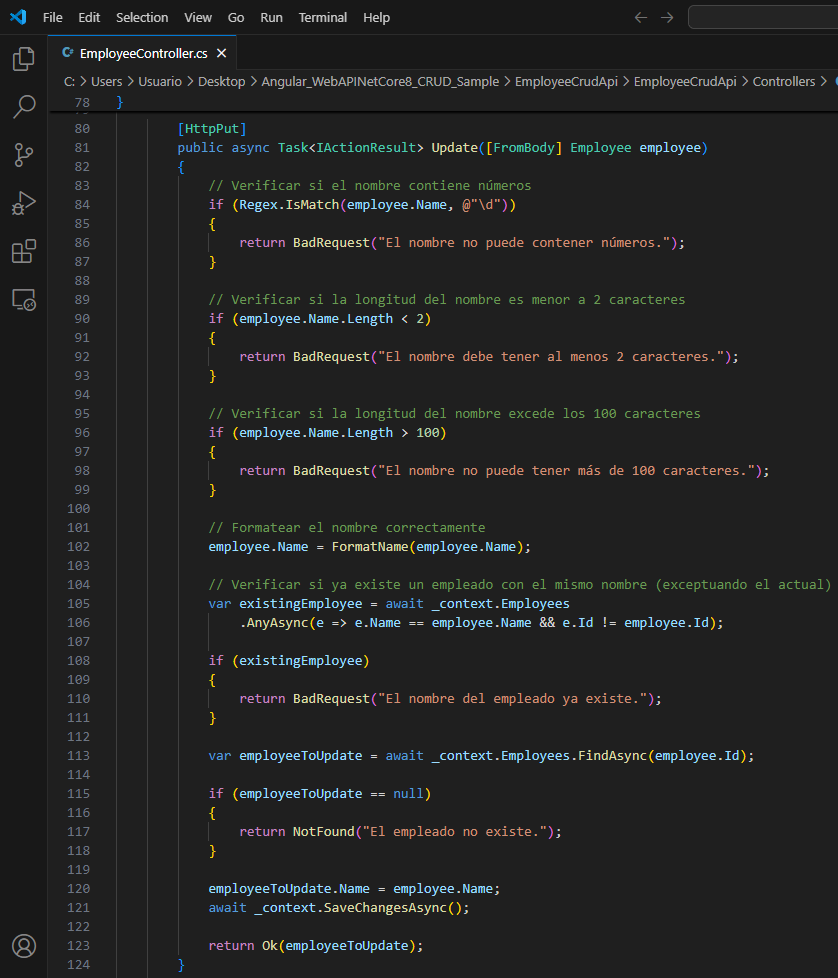


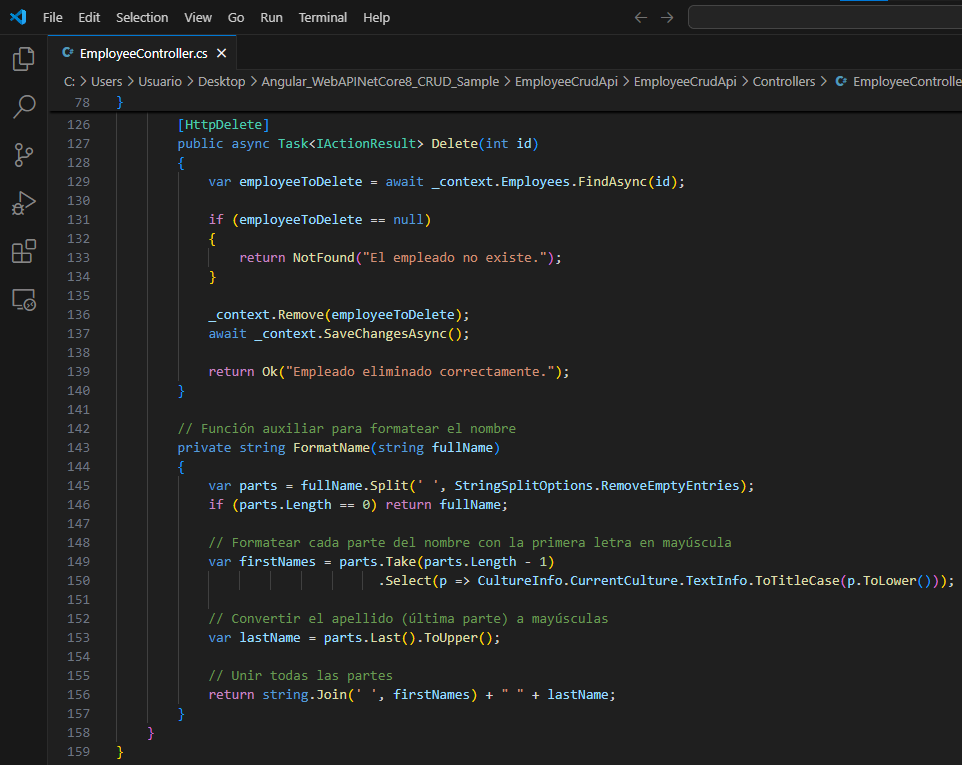


1. Realizar al menos 5 de las siguientes modificaciones sugeridas al código de la API:
   1. Al agregar y al editar un empleado, controlar que el nombre del empleado no esté repetido.
   2. La longitud máxima del nombre y apellido del empleado debe ser de 100 caracteres.
   3. Almacenar el nombre en la BD siempre con la primera letra de los nombres en Mayúscula y todo el apellido en Mayusculas. Ejemplo, si recibo juan carlos chamizo, se debe almacenar como Juan Carlos CHAMIZO.
   4. Validar que el nombre tenga un número mínimo de caracteres, por ejemplo, al menos dos caracteres para evitar entradas inválidas como "A".
   5. Verificar que el nombre no contenga números, ya que no es común en los nombres de empleados.

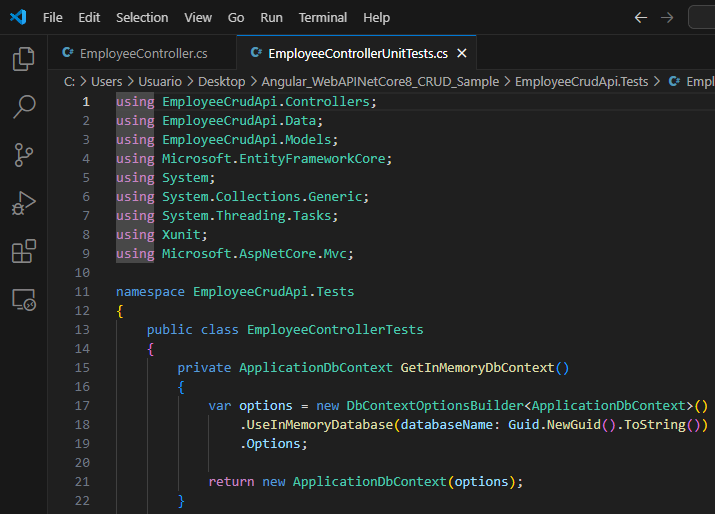


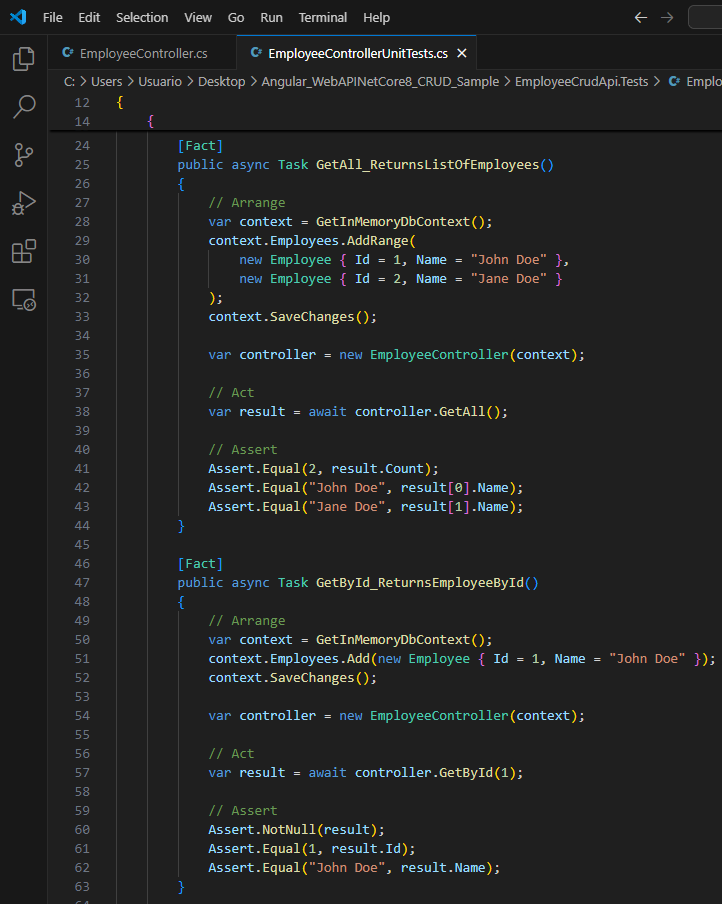


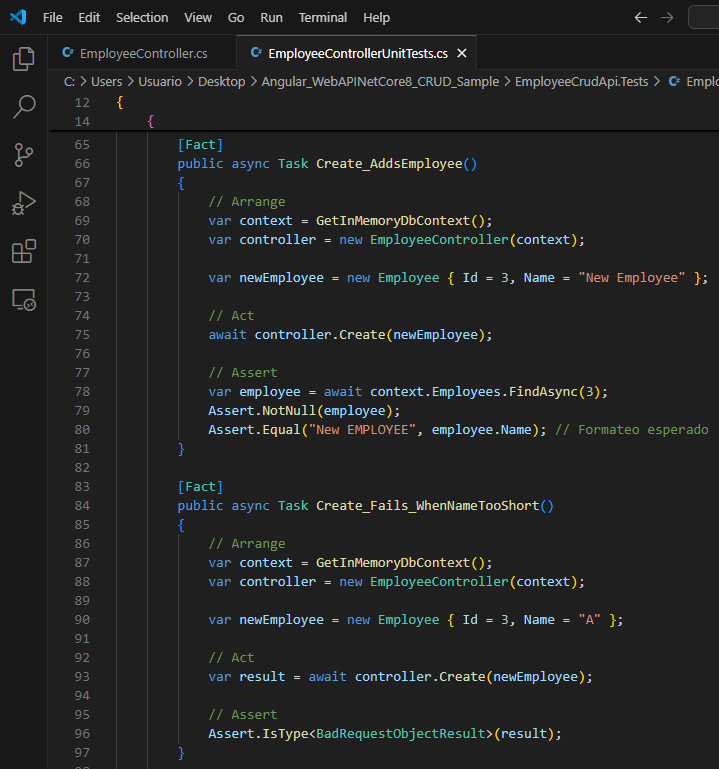


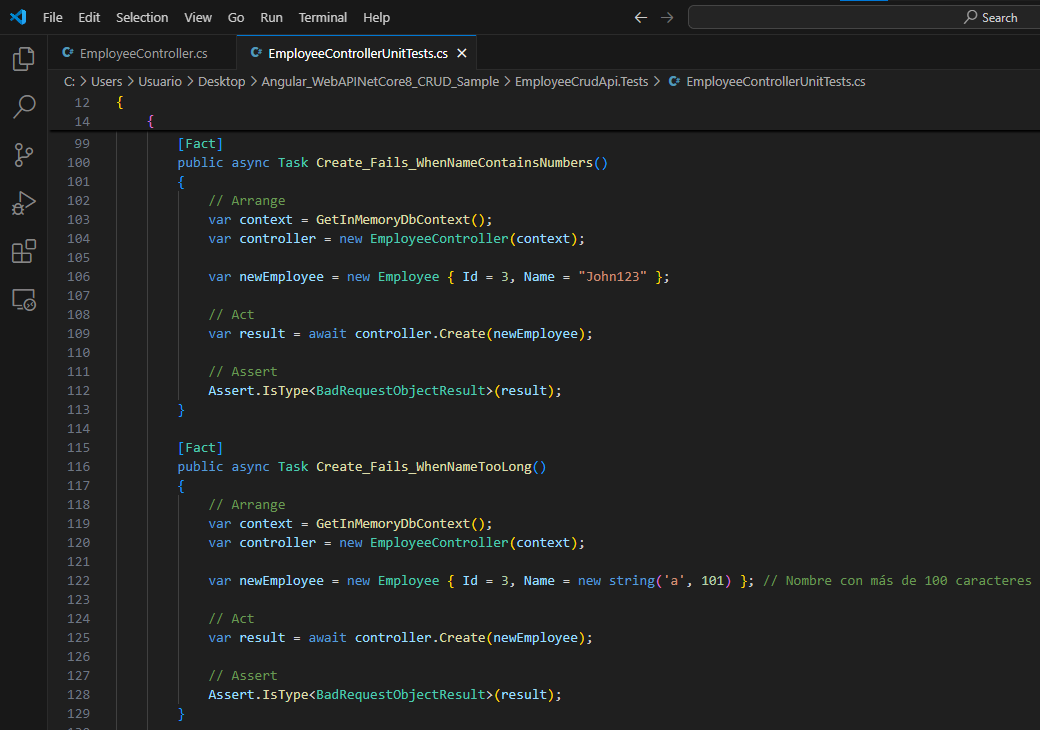


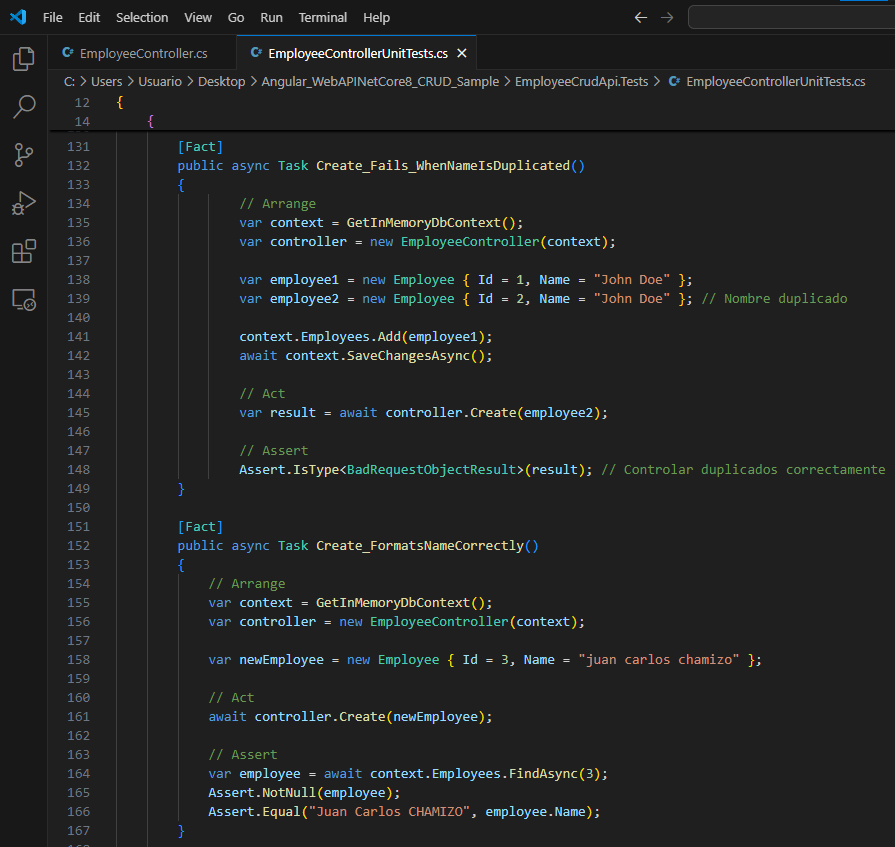
1. Crear las pruebas unitarias necesarias para validar las modificaciones realizadas en el código.

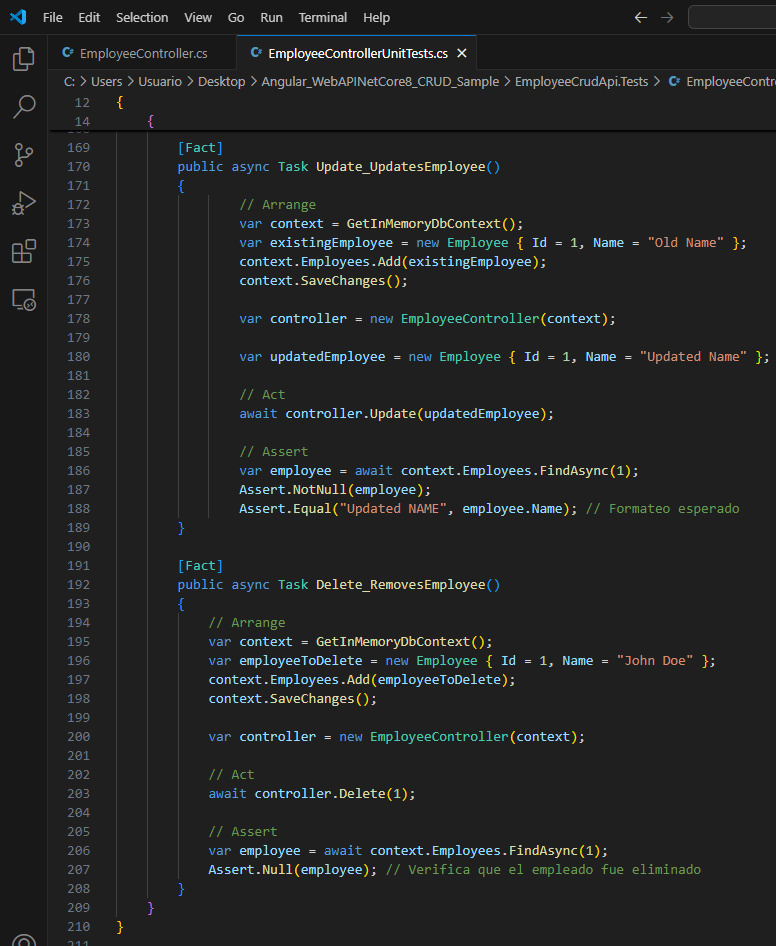


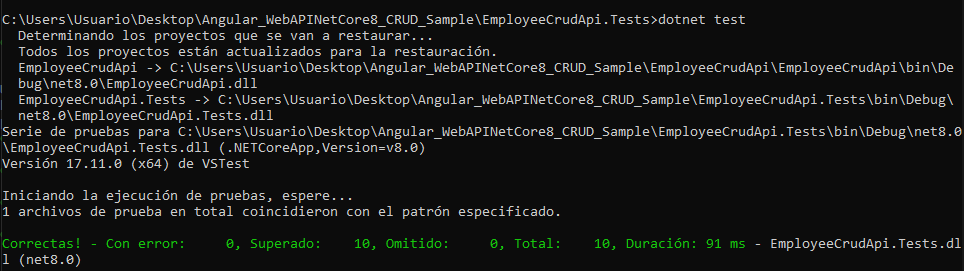




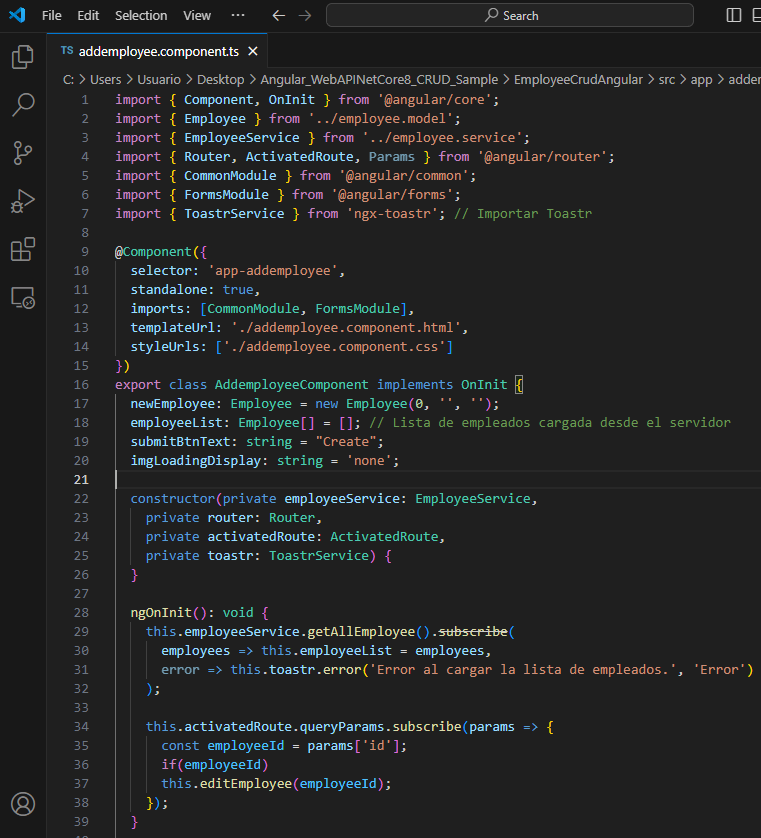


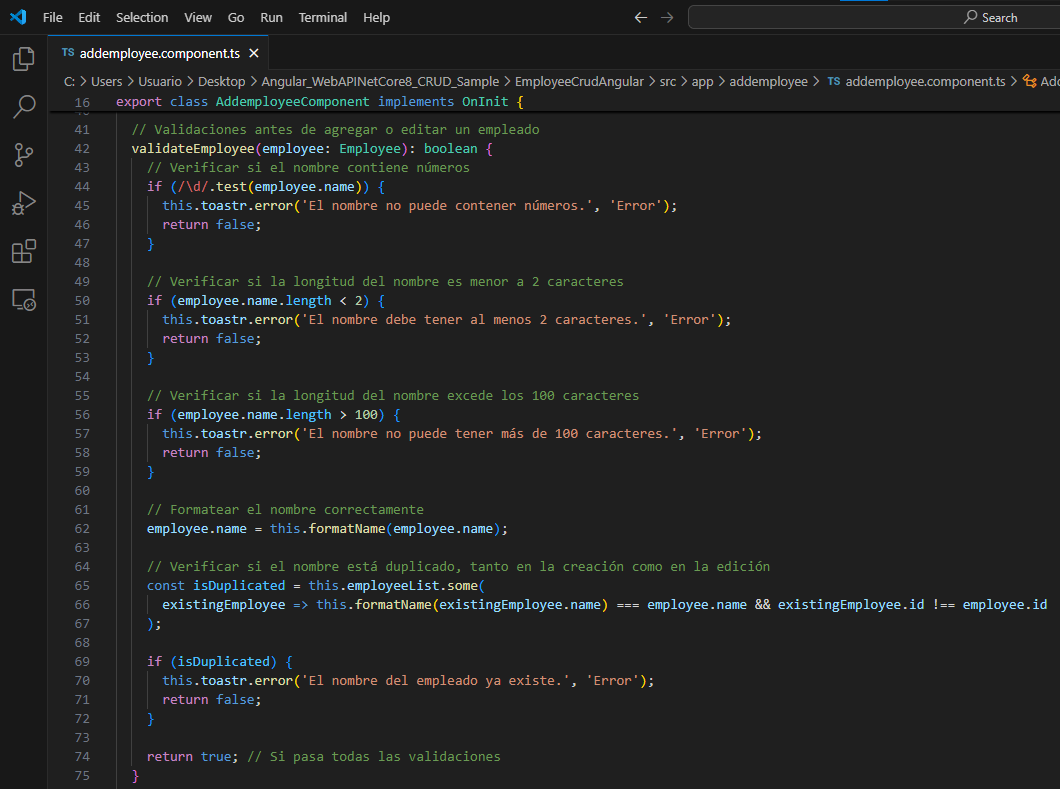


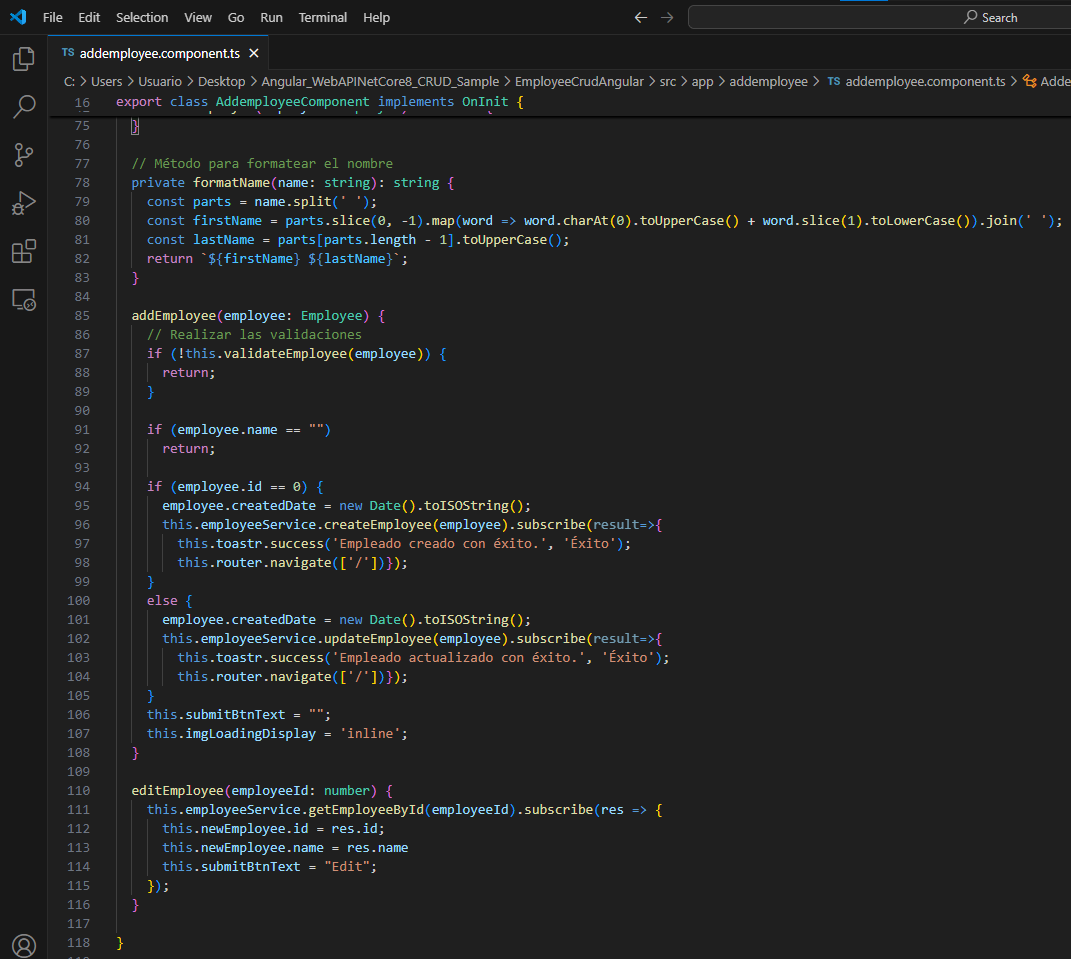




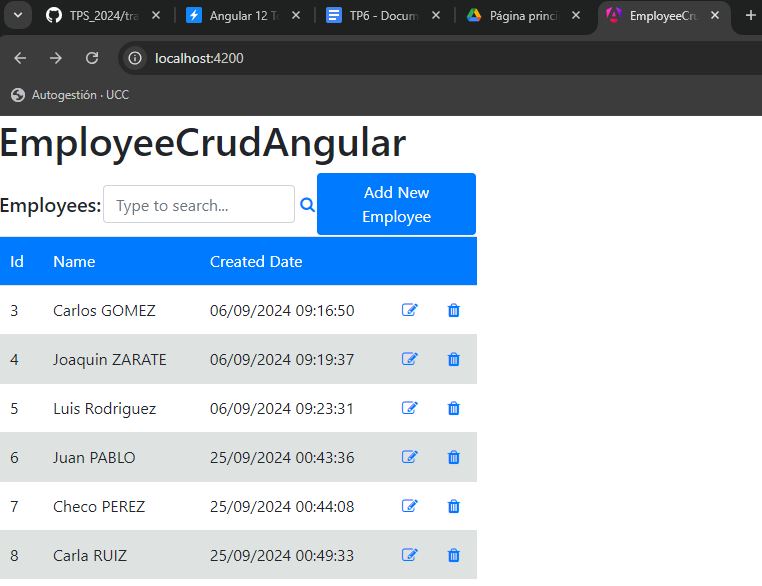
1. 1. Realizar en el código del front las mismas modificaciones hechas a la API.

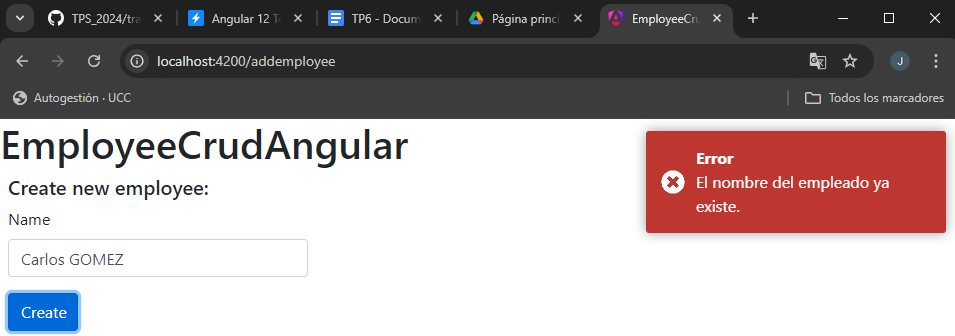


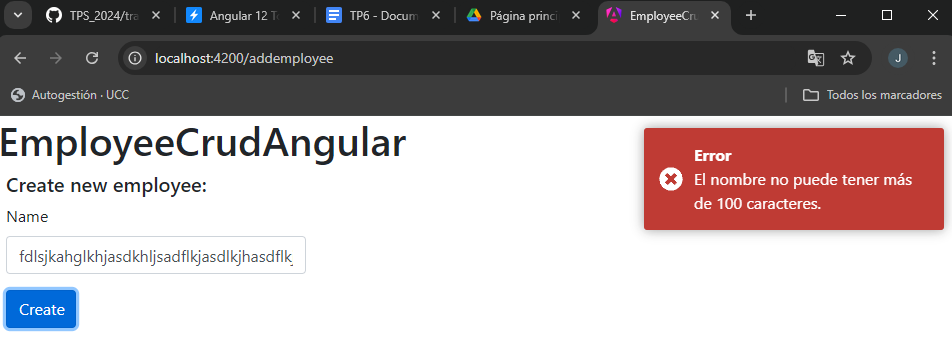


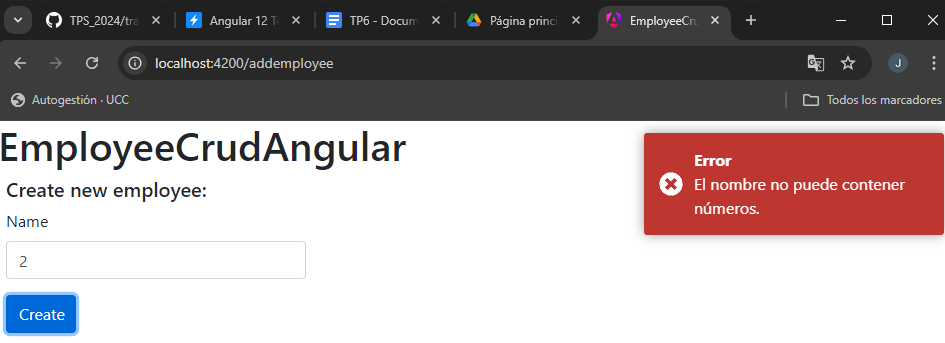


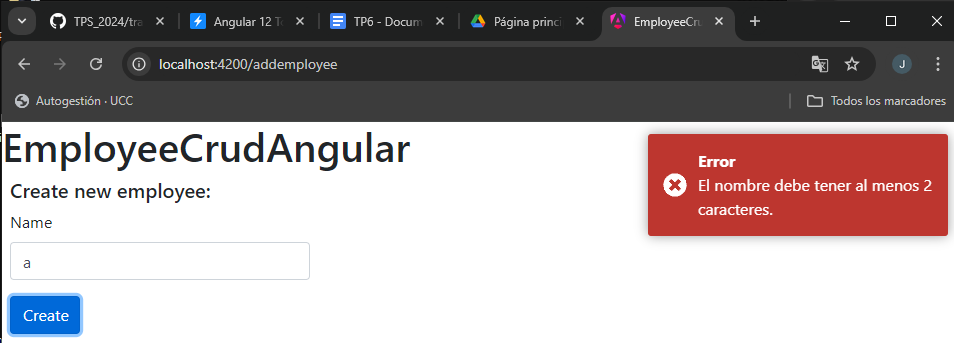
* 1. Las validaciones deben ser realizadas en el front sin llegar a la API, y deben ser mostradas en un toast.

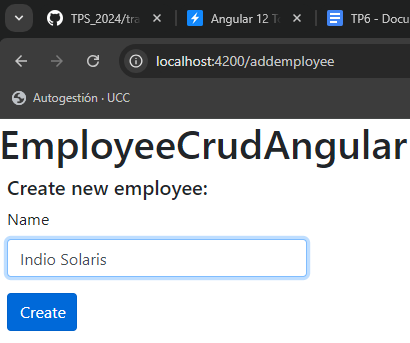


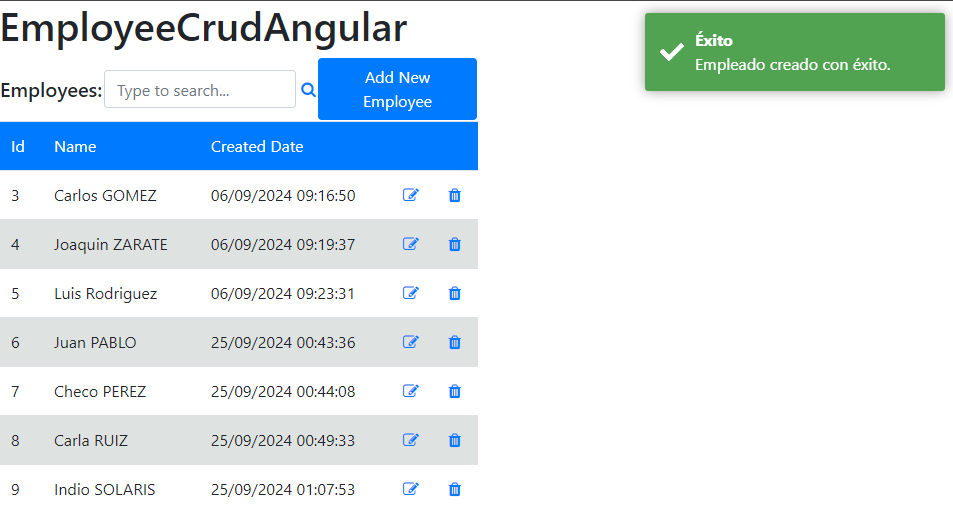












* 1. Crear las pruebas unitarias necesarias en el front para validar las modificaciones realizadas en el código del front.

