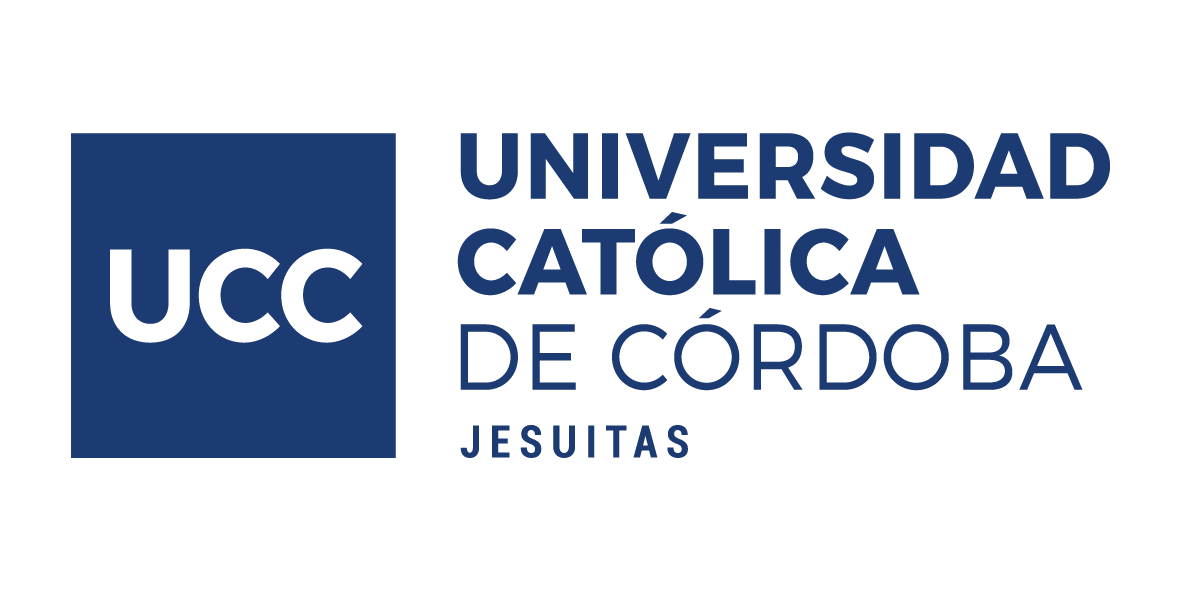
### **Trabajo Práctico N°7**

Alumno:

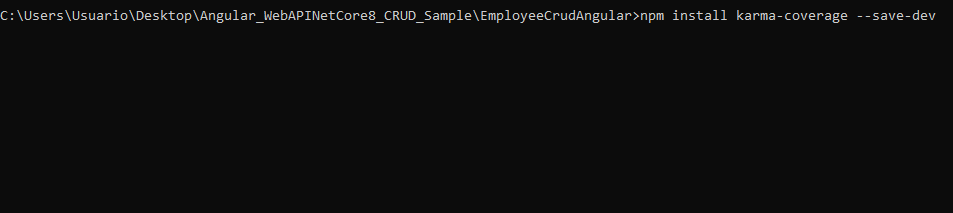
* Miron Juan Pablo (2100815).

Materia: Ingeniería de Software III.

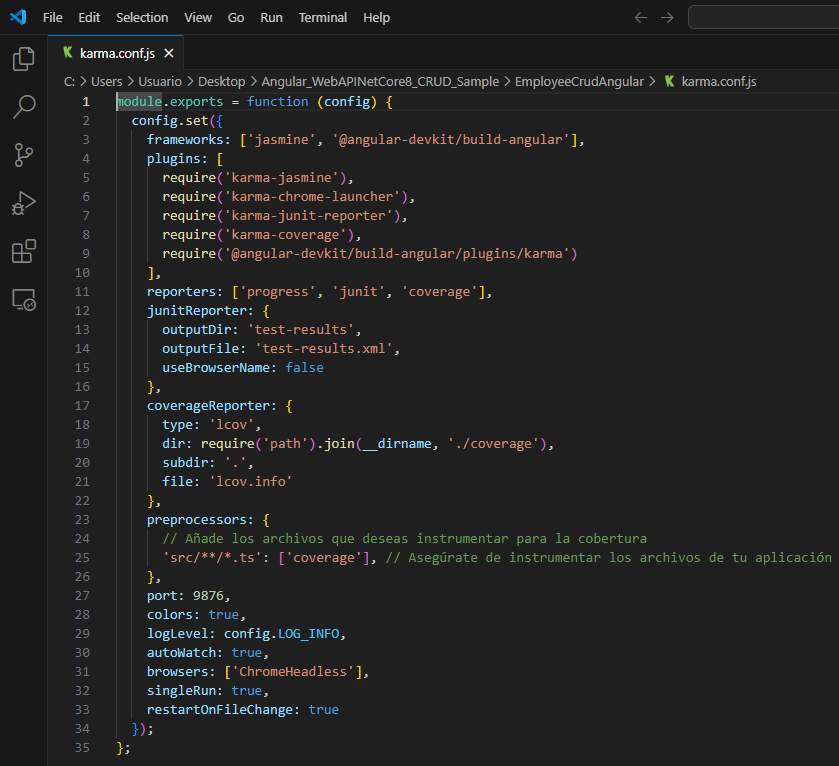
Docente: Schwindt Ariel.



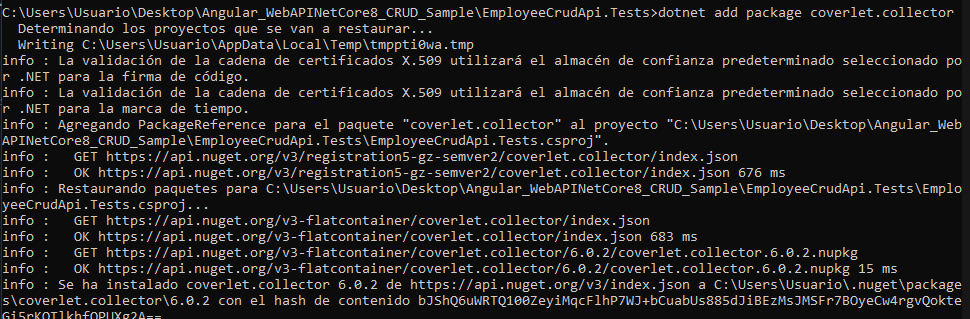
1. Agregar Code Coverage a nuestras pruebas unitarias de backend y frontend e integrarlas junto con sus resultados en nuestro pipeline de build.
   1. En el directorio raíz de nuestro proyecto Angular instalar el siguiente paquete.



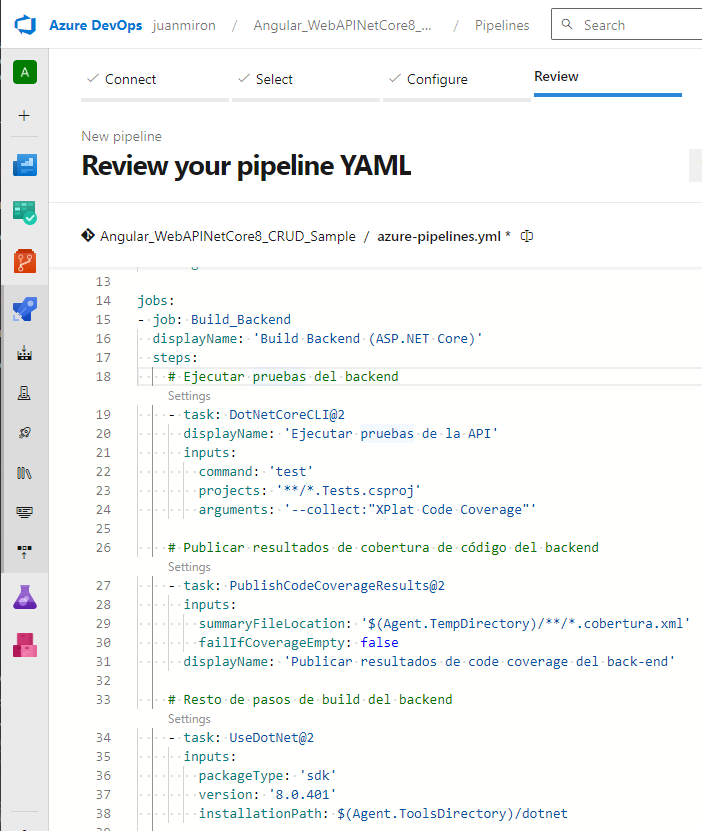
* 1. Editar nuestro archivo karma.conf.js para que incluya reporte de cobertura.



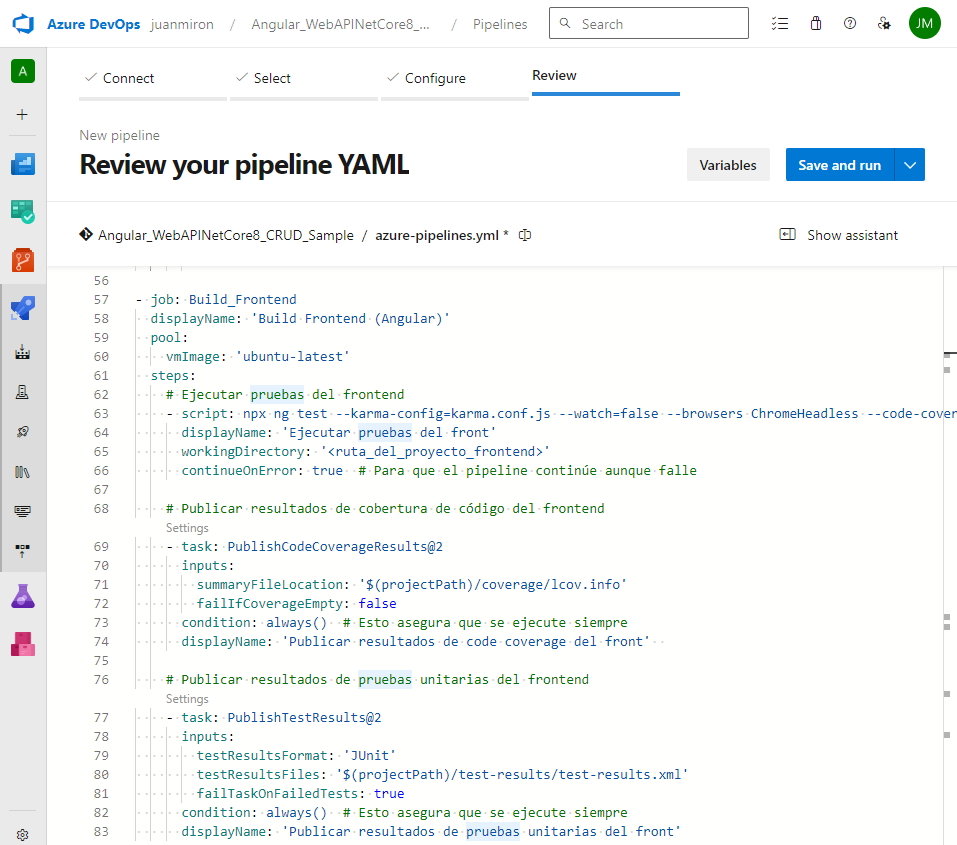
* 1. En el dir raíz del proyecto EmployeeCrudApi.Tests ejecutar.



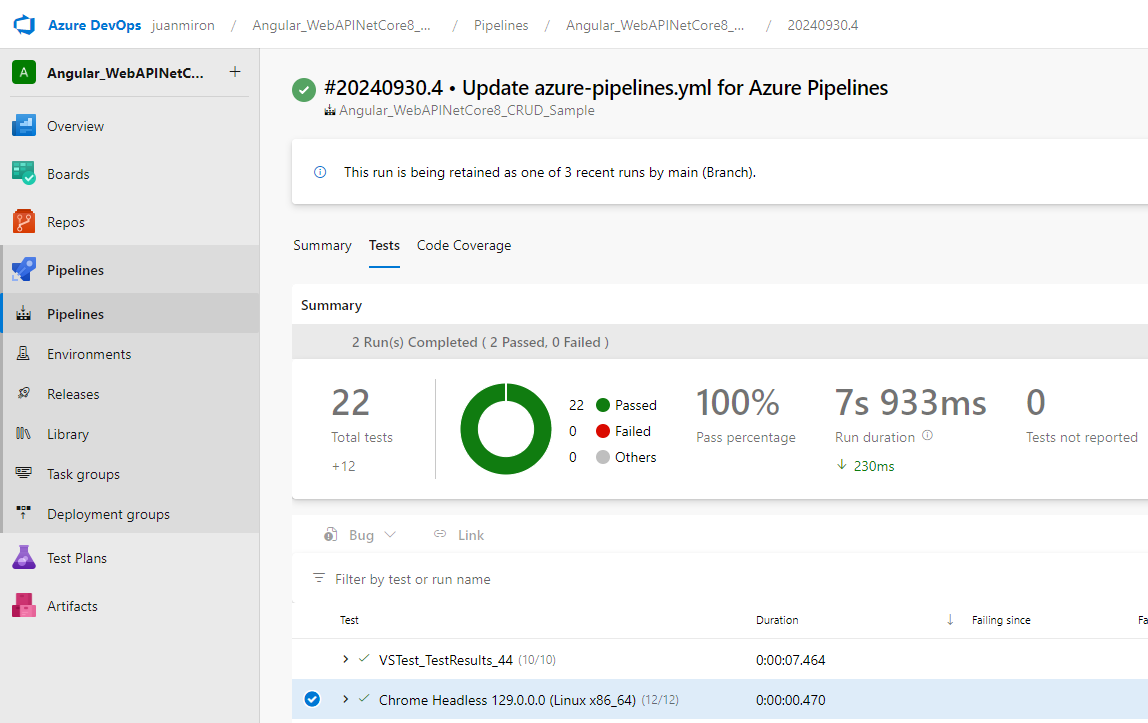
* 1. Agregar a nuestro pipeline ANTES del Build de Back la tarea de test con los argumentos especificados y la de publicación de resultados de cobertura.

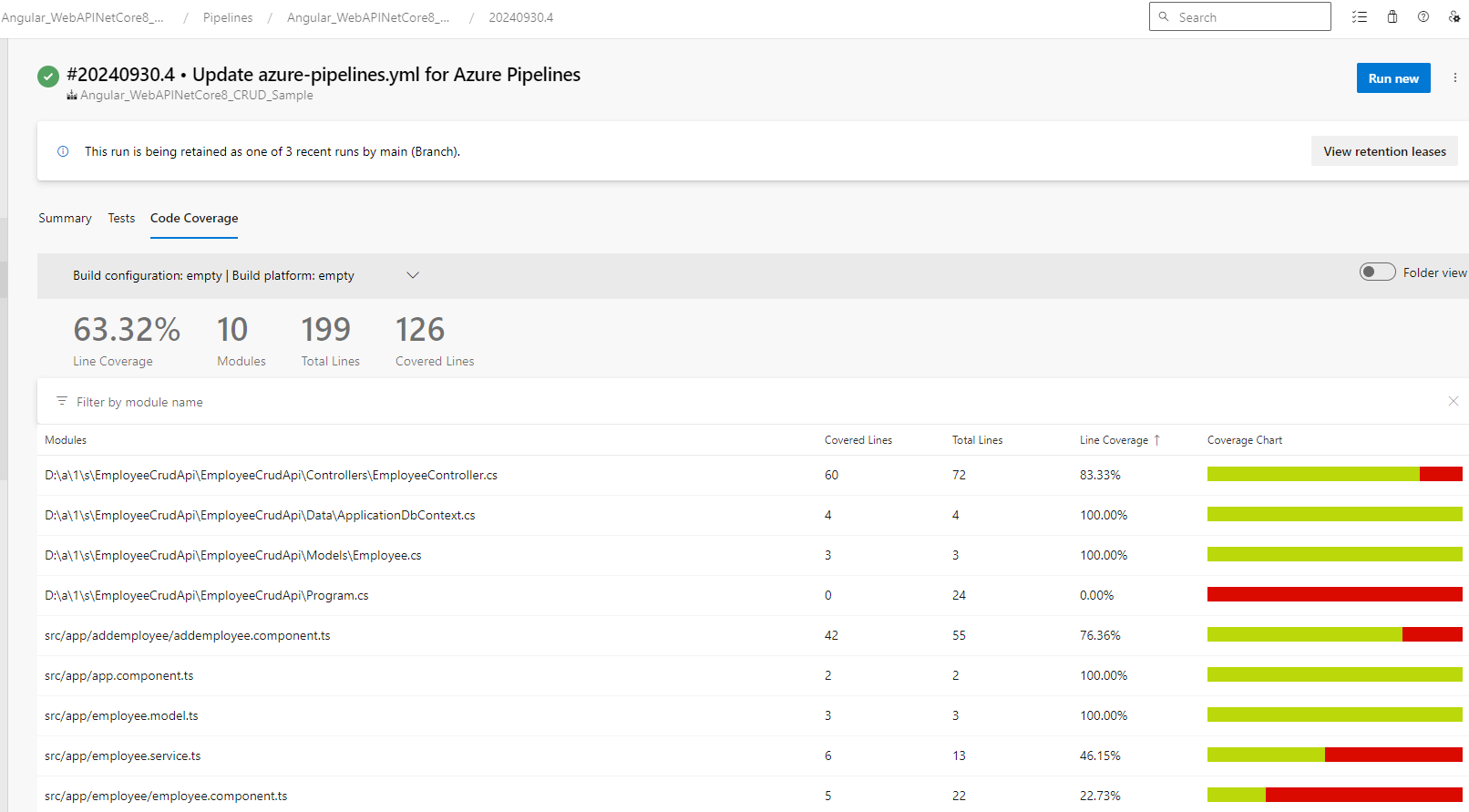


* 1. Agregar a nuestro pipeline ANTES del Build de front la tarea de test y la de publicación de los resultados.



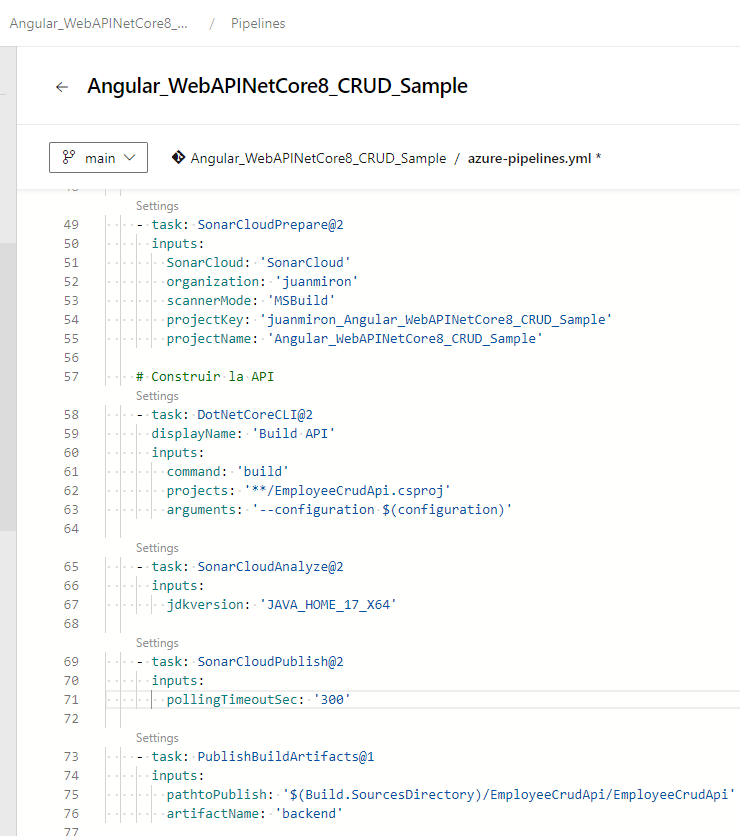
* 1. Ejecutar el pipeline y analizar el resultado de las pruebas unitarias y la cobertura de código.



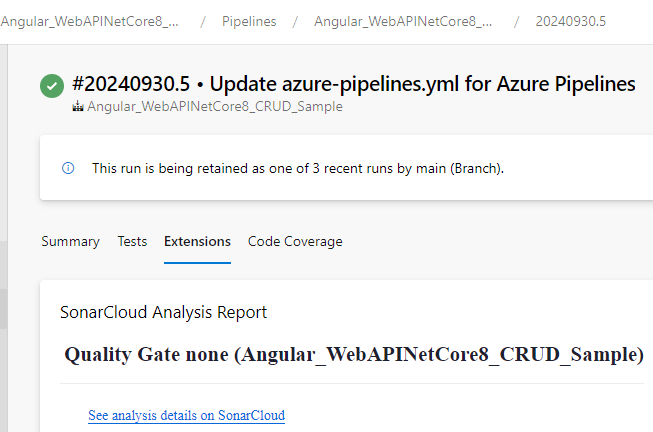


1. Agregar Análisis Estático de Código con SonarCloud.
   1. Integraremos SonarCloud para analizar el código fuente. Configurar SonarCloud en nuestro pipeline siguiendo instructivo 5.1.

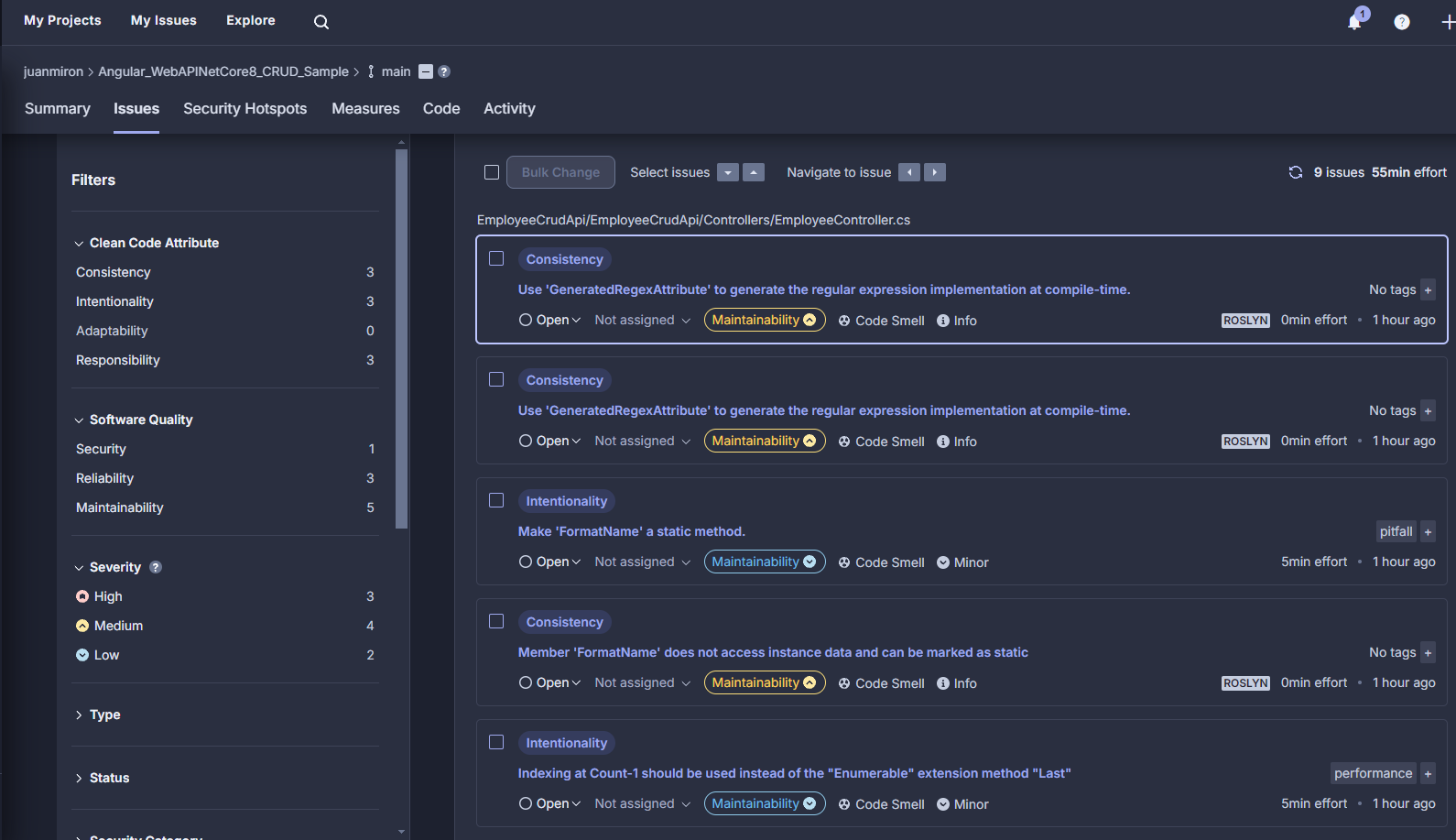
Antes de nuestra tarea de Build del Back y después de nuestra tarea de Build del Back:



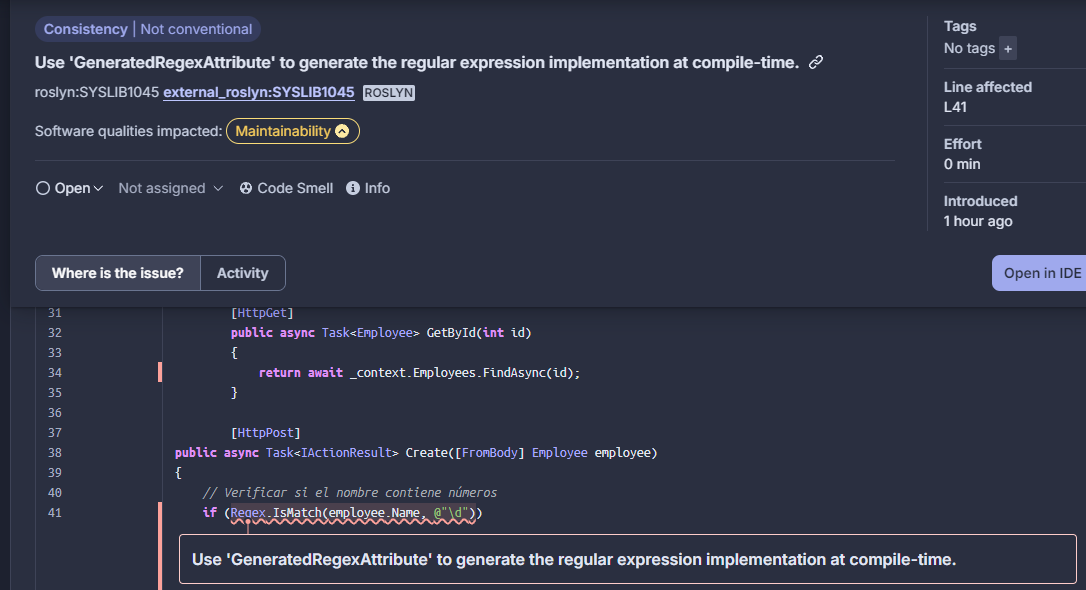
* 1. Vemos el resultado de nuestro pipeline, en extensions tenemos un link al análisis realizado por SonarCloud.



* 1. Ir al link y analizar toda la información obtenida. Detallar en la entrega del TP los puntos más relevantes del informe, qué significan y para qué sirven.

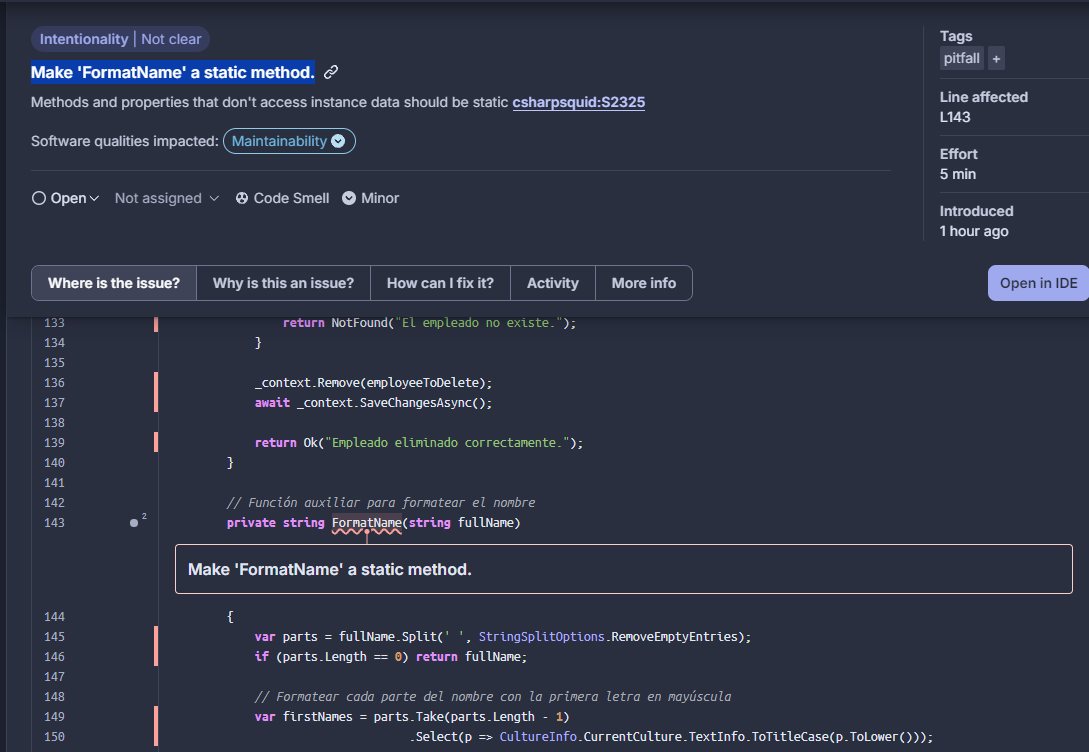


* + 1. Use 'GeneratedRegexAttribute' to generate the regular expression implementation at compile-time:



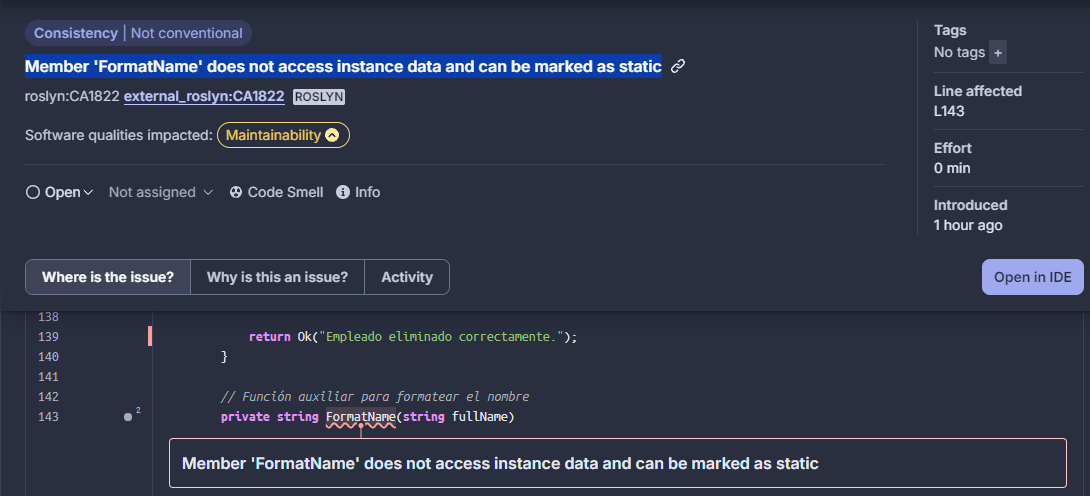
Nos indica que utilizar este atributo tiene un impacto medio a la hora de compilar, y podría cambiarse por otro para verificar si el nombre contiene un número.

* + 1. Make 'FormatName' a static method:



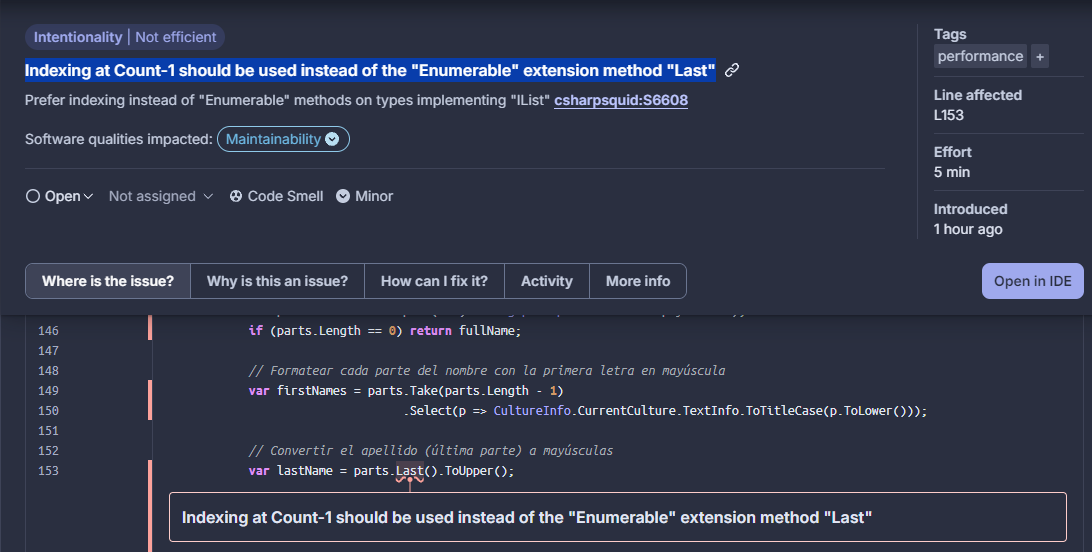
Nos indica que si la función para formatear el nombre no accede a instancias de datos, debería implementarse como una función estática. No es un error pero no queda claro por qué no es estática.

* + 1. Member 'FormatName' does not access instance data and can be marked as static.



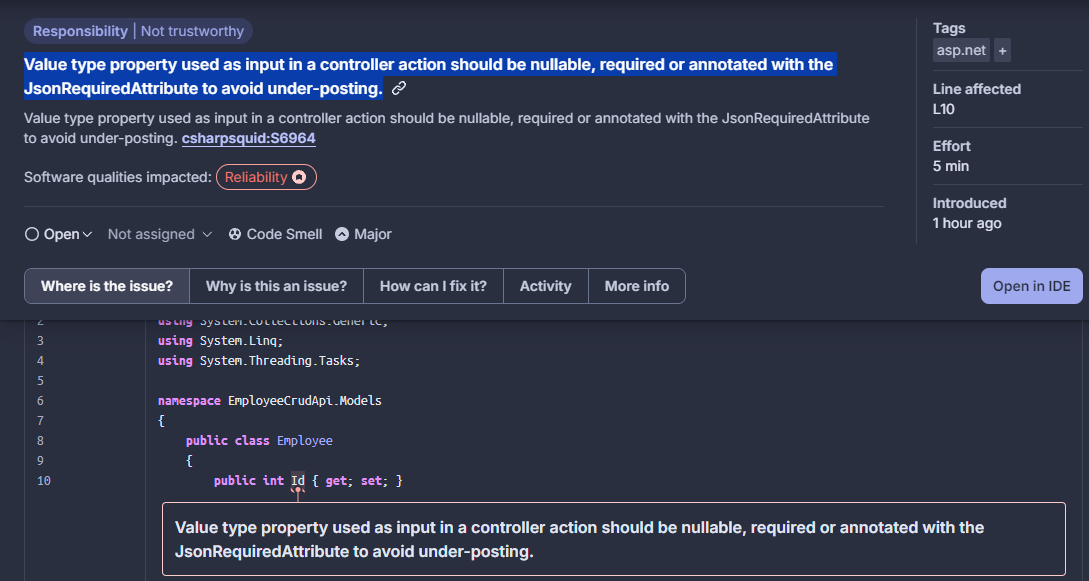
Es el mismo aviso que en el caso anterior, sólo que se titula como algo que no es consistente.

* + 1. Indexing at Count-1 should be used instead of the "Enumerable" extension method "Last":



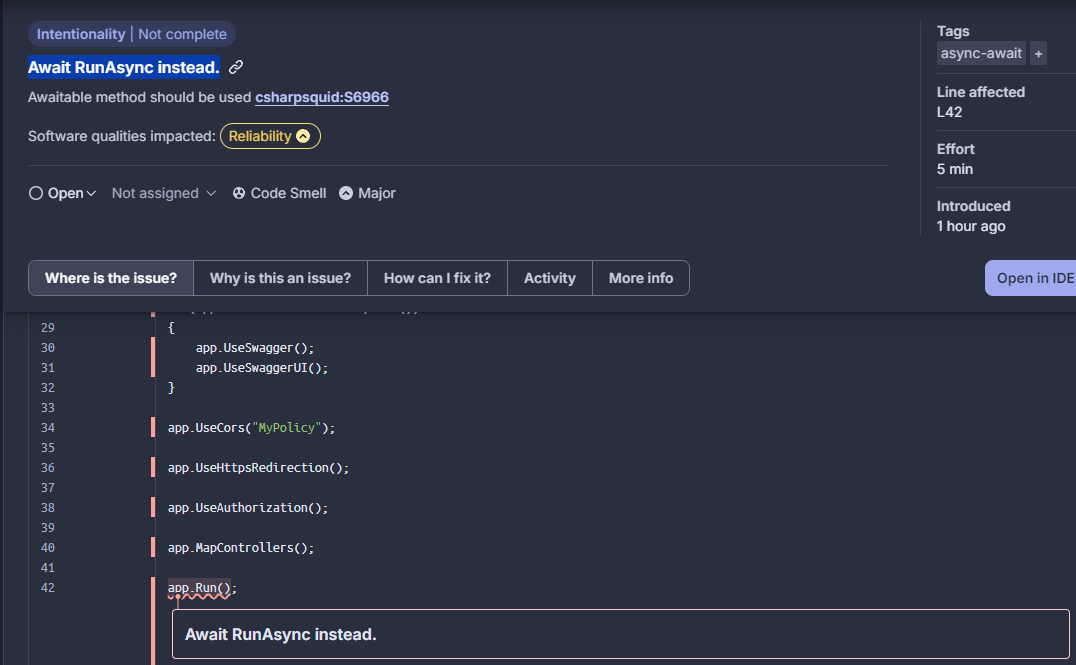
Este aviso entiendo que quiere decir que una mejor forma que usar el método “Last” para acceder al apellido es utilizar un contador para recorrer el nombre completo y extraerlo, ya que de otra forma no parece tener sentido.

* + 1. Value type property used as input in a controller action should be nullable, required or annotated with the JsonRequiredAttribute to avoid under-posting.



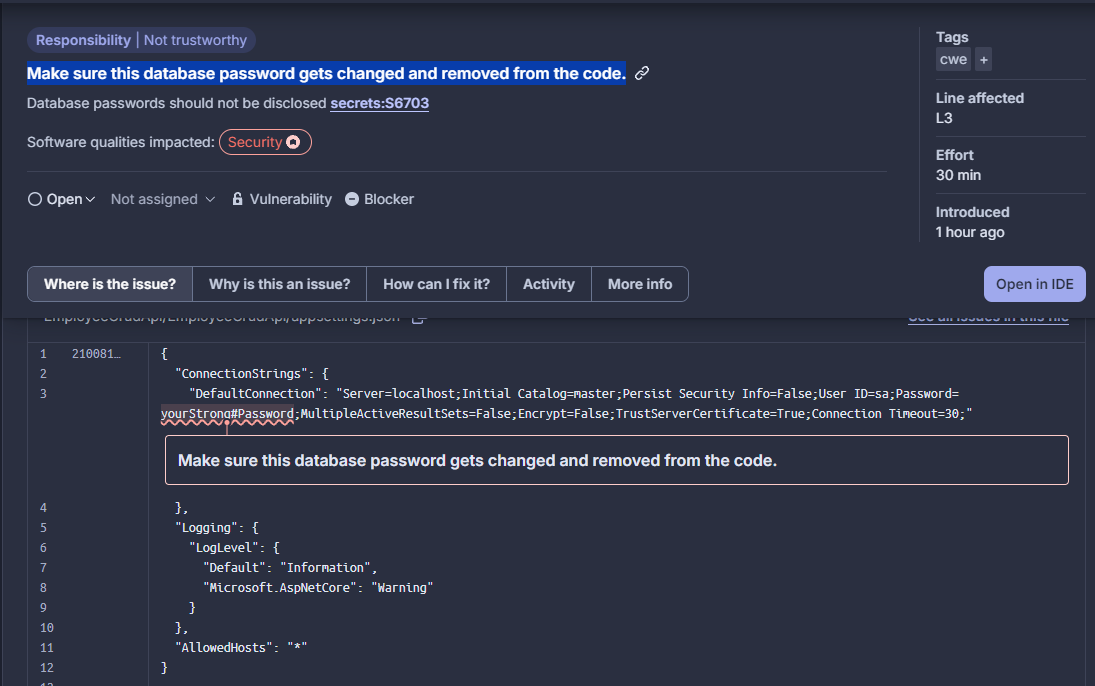
Este mensaje significa que cuando propiedades de tipo de valor se usan como parámetros de entrada en un controlador de ASP.NET, puede haber un riesgo de "under-posting", que ocurre cuando no se envían todos los datos necesarios al servidor.

* + 1. Await RunAsync instead.



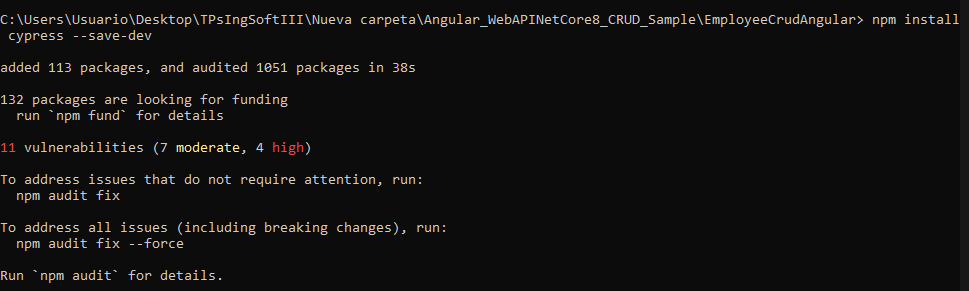
Se indica que hay que usar otro método que espere a la sincronización ante de correr el código para que estén todos los datos necesarios disponibles.

* + 1. Make sure this database password gets changed and removed from the code.

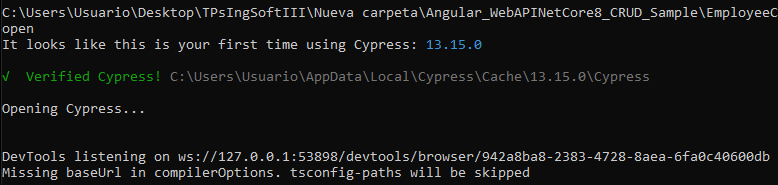


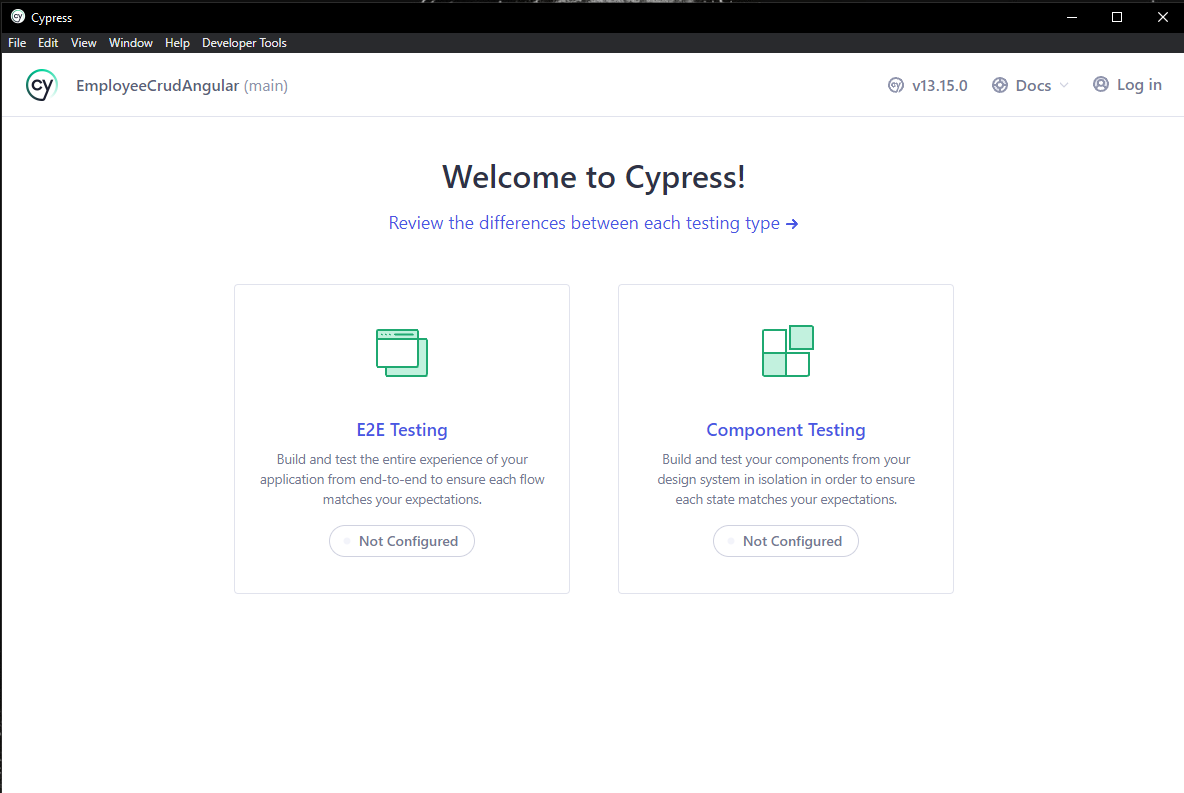
Es un claro error de seguridad ya que la contraseña de la base de datos está “hardcodeada” y esto implica una vulnerabilidad muy importante para cualquier sistema.

1. Pruebas de Integración con Cypress.
   1. En el directorio raíz de nuestro proyecto Angular instalar el siguiente paquete.

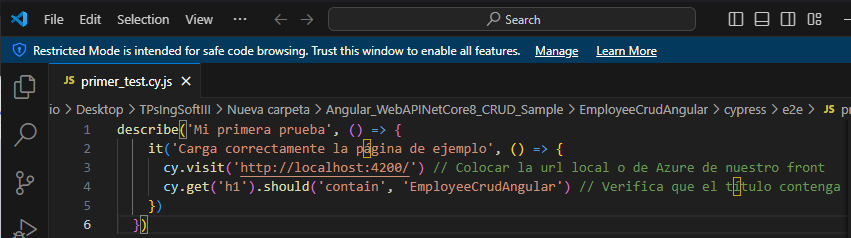
****

* 1. Abrir Cypress.

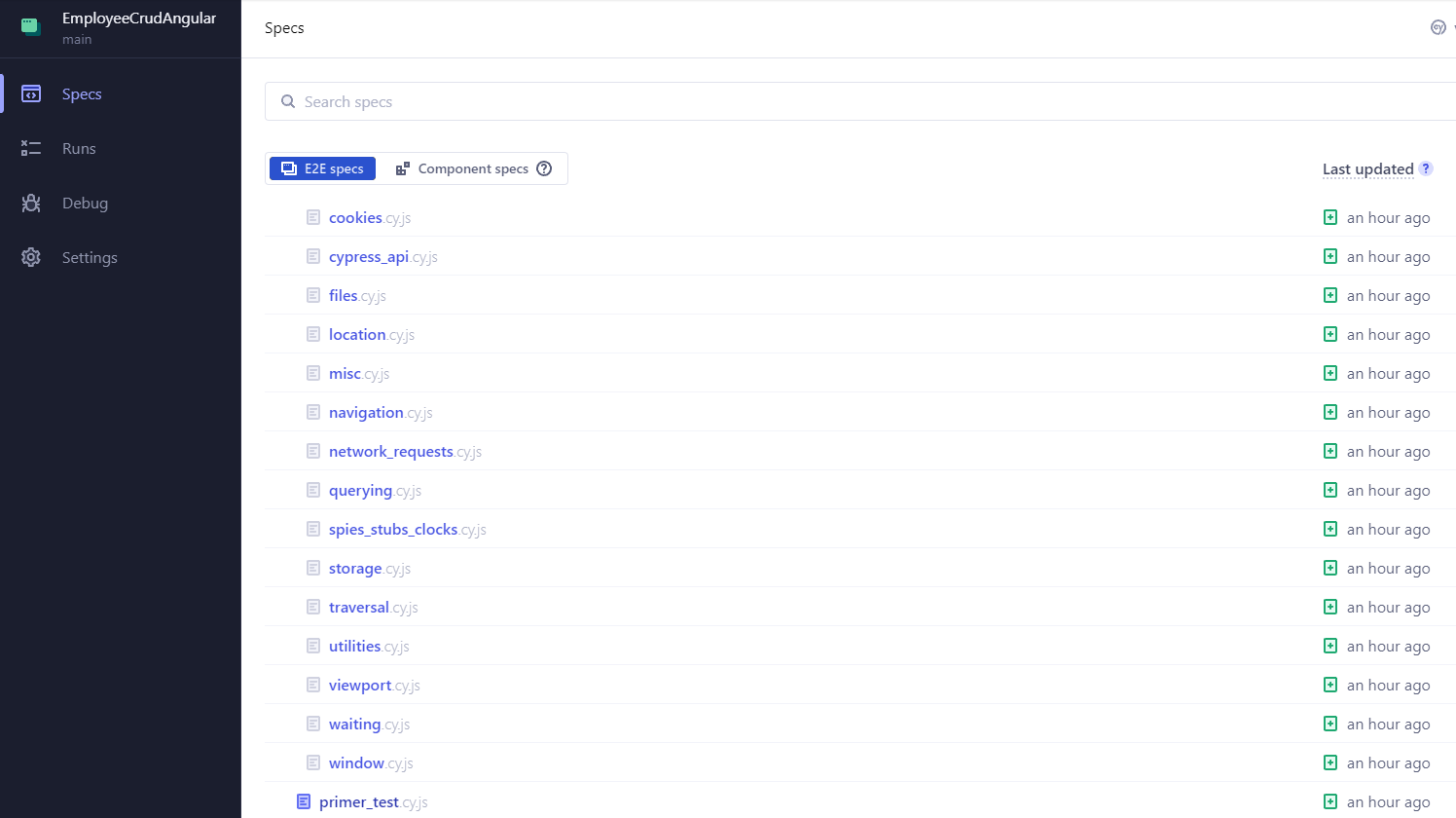


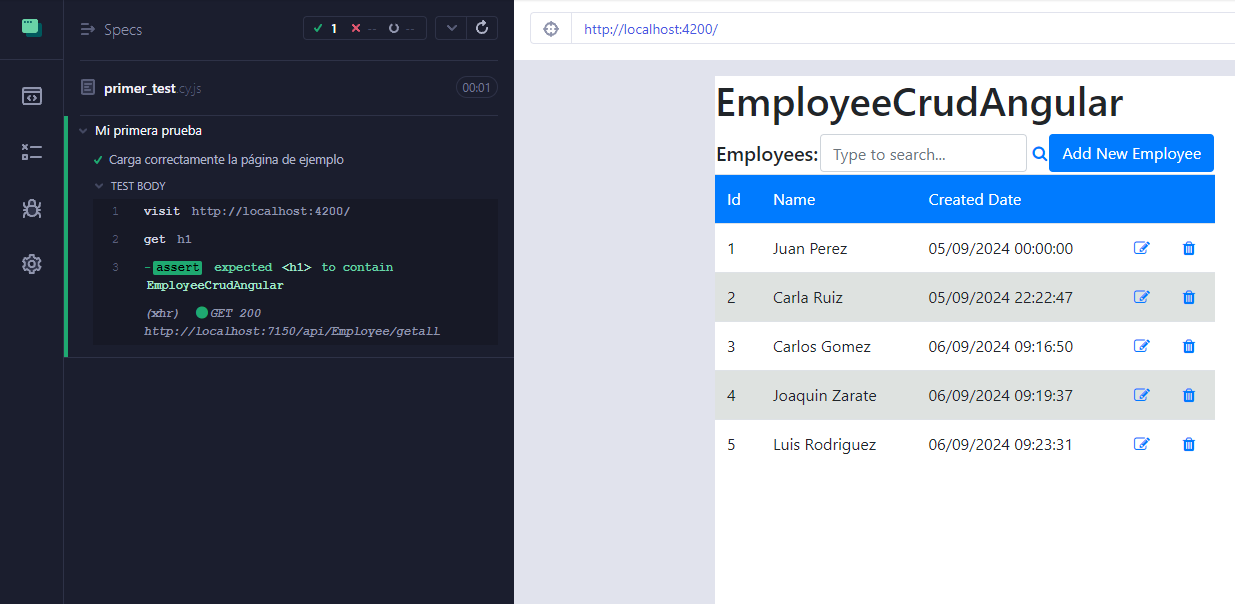


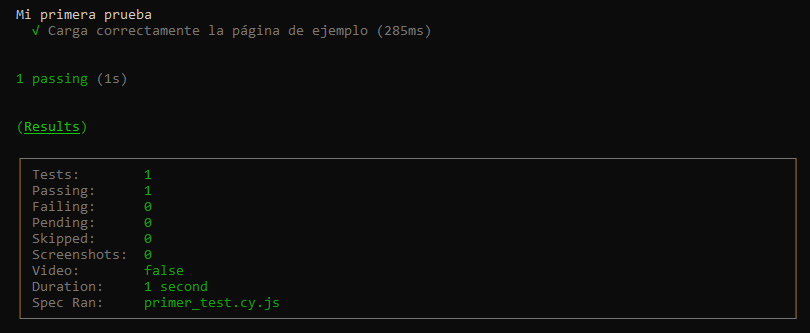
* 1. Inicializar Cypress en nuestro proyecto como se indica en el instructivo 5.2.
  2. Crear nuestra primera prueba navegando a nuestro front.



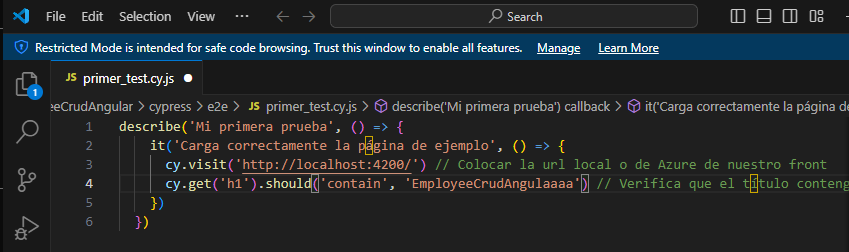
* 1. Correr nuestra primera prueba.

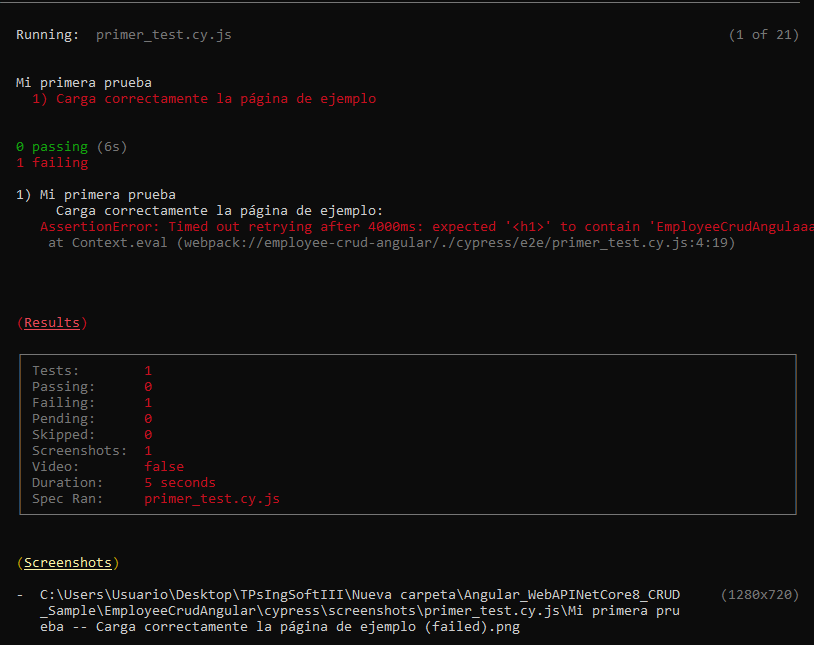




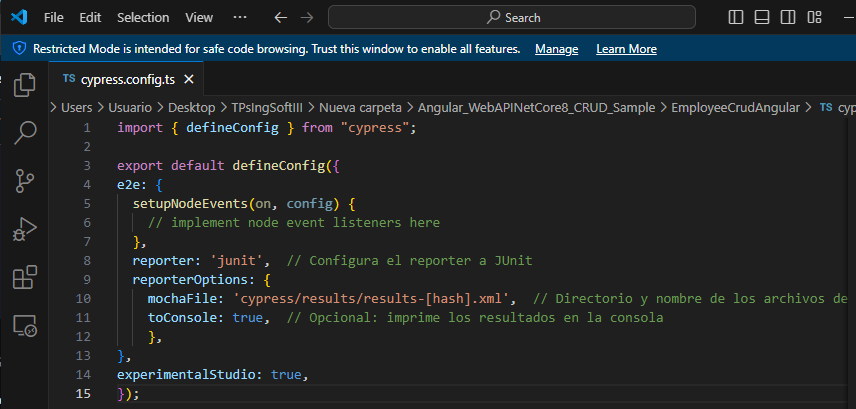


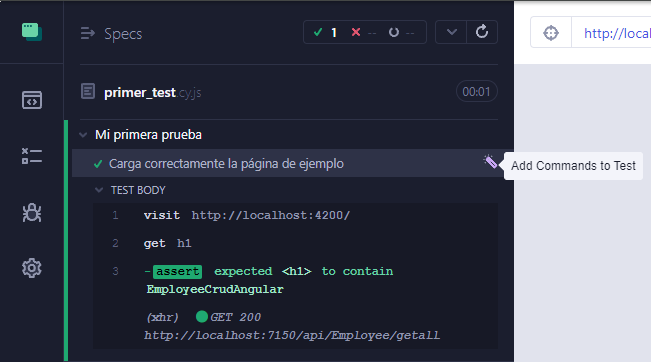
* 1. Modificar nuestra prueba para que falle.

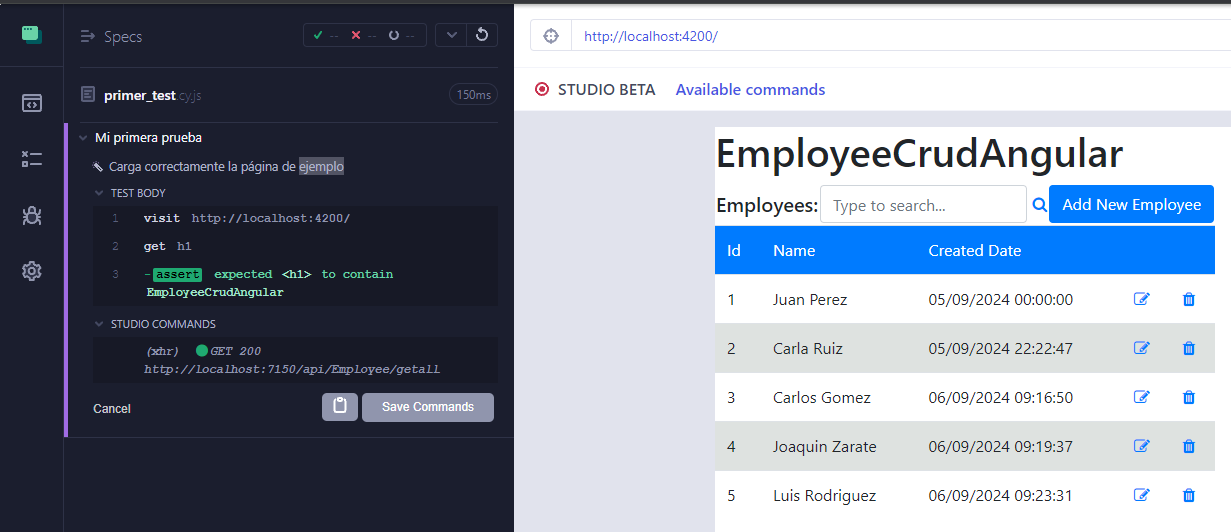




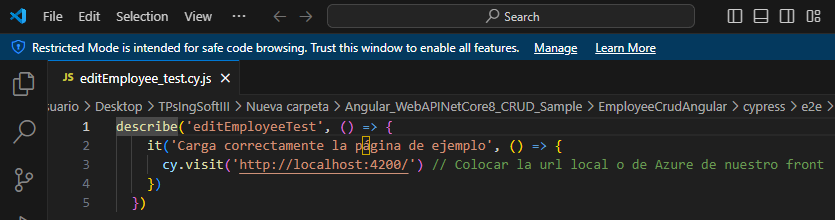
* 1. Grabar nuestras pruebas para que Cypress genere código automático y genere reportes.



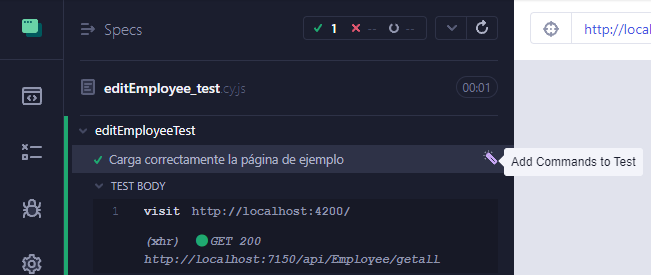


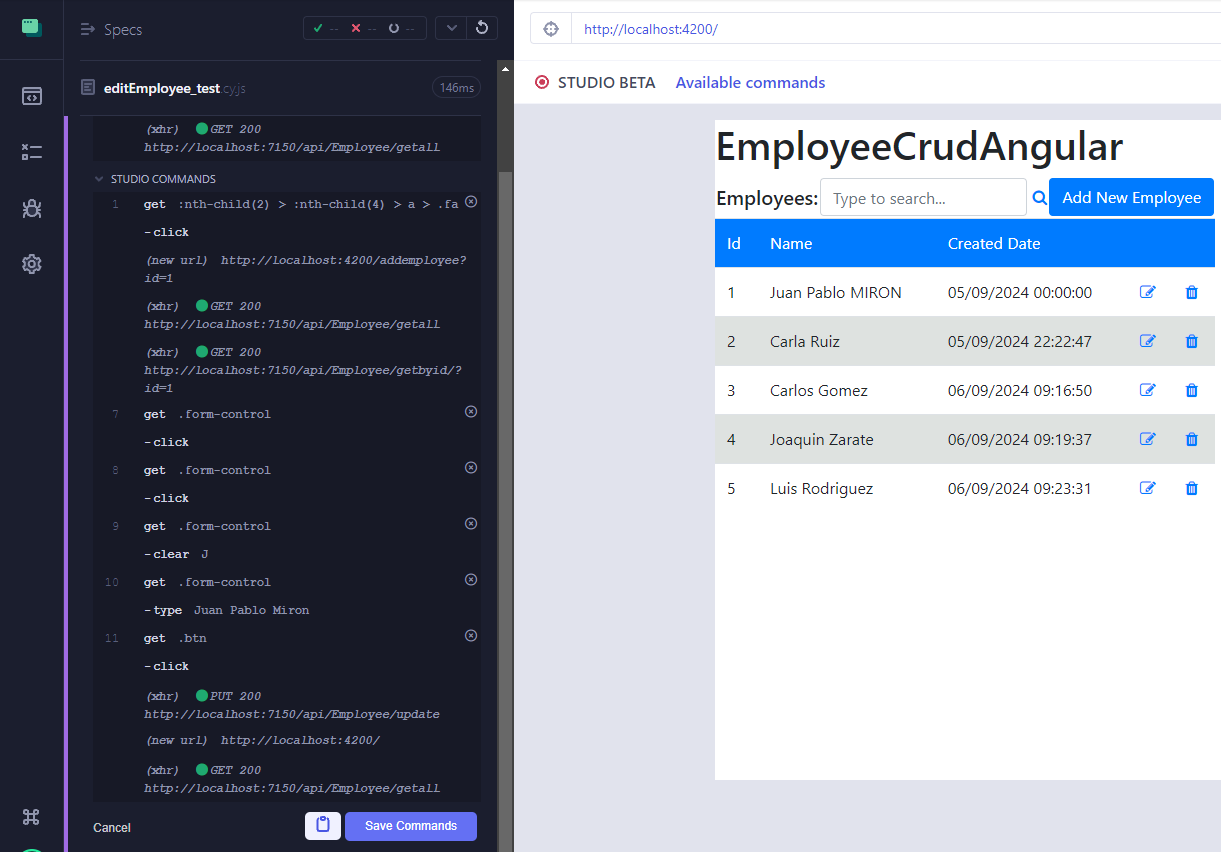


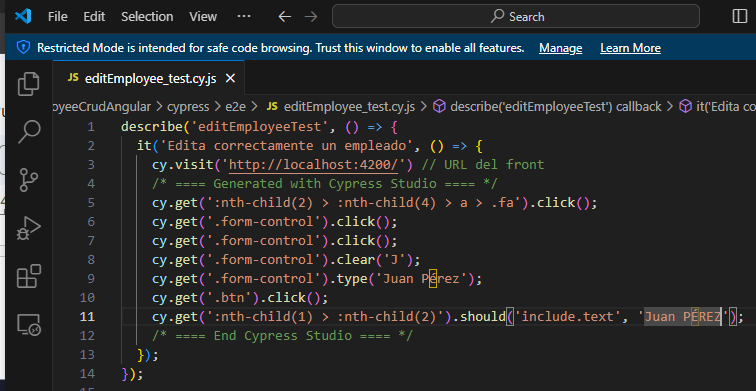
* 1. Hacemos prueba de editar un empleado.

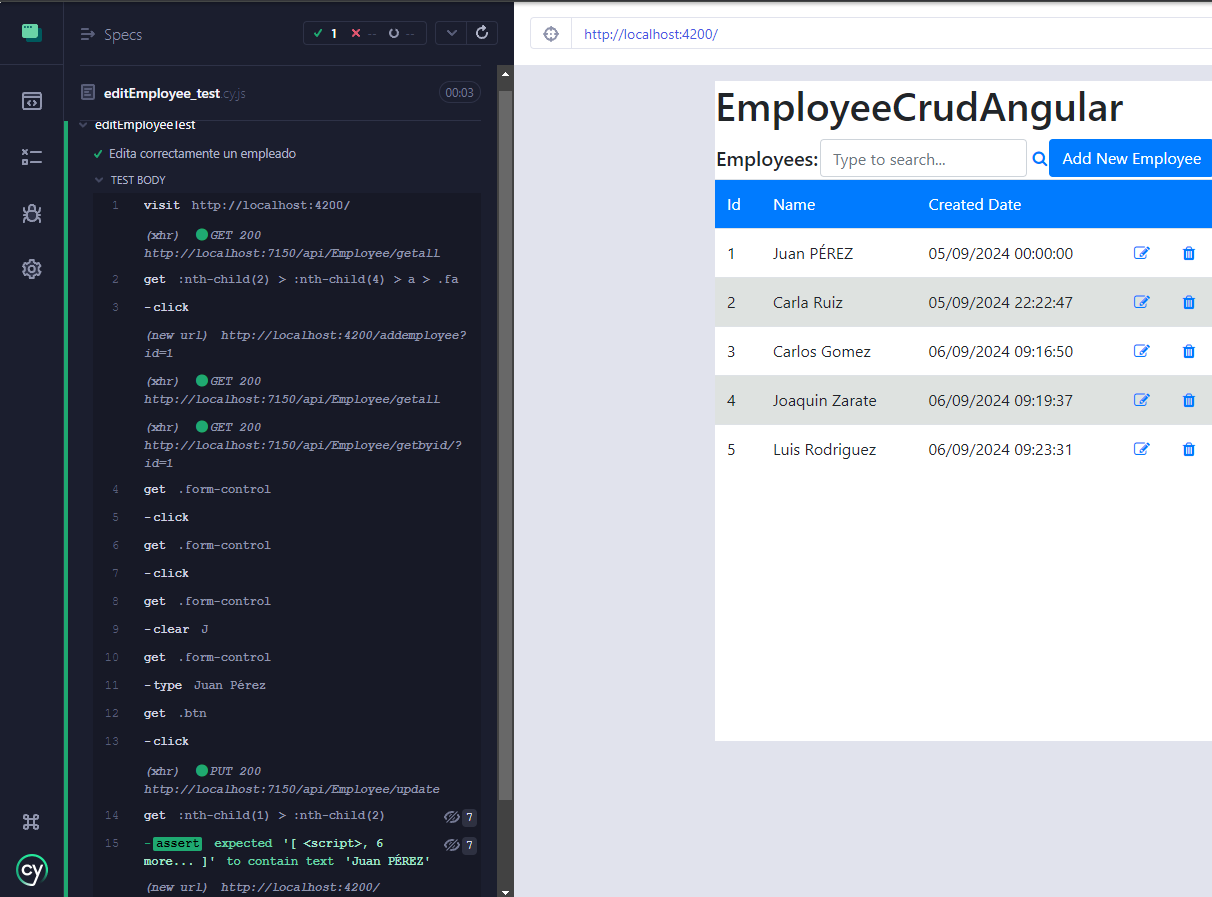






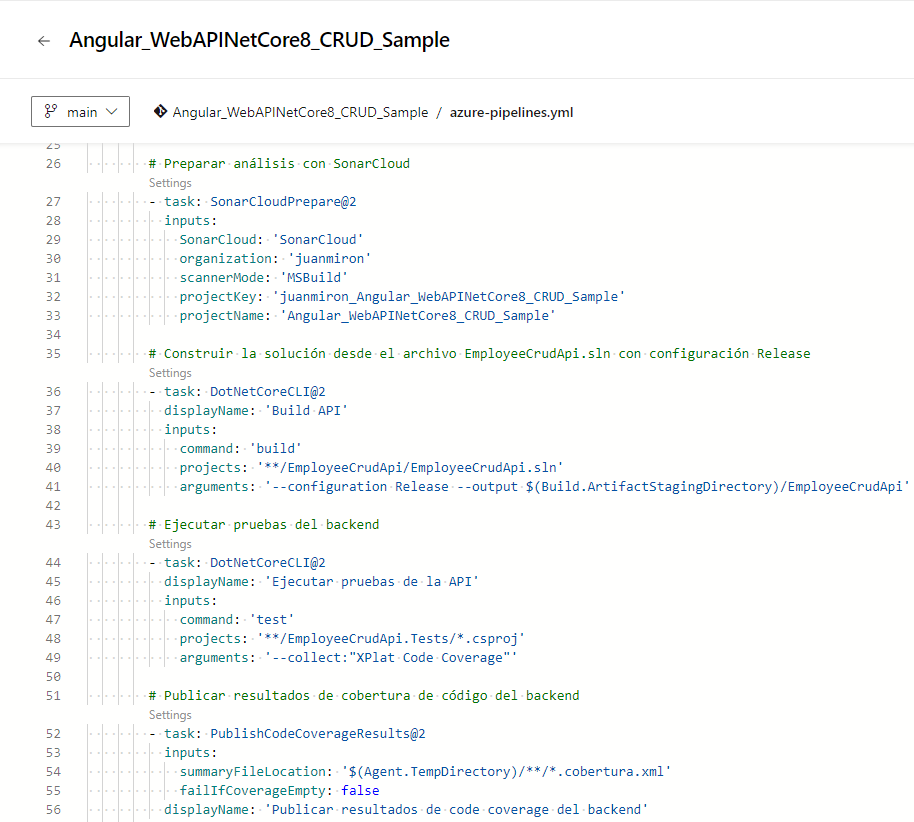


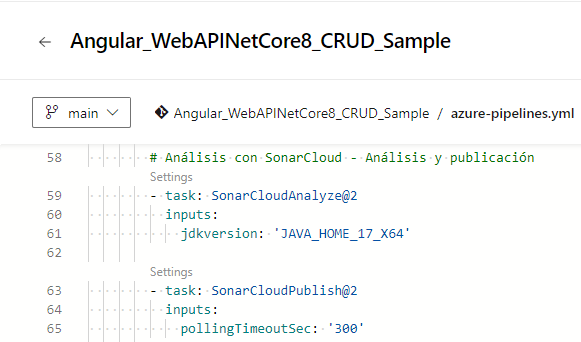




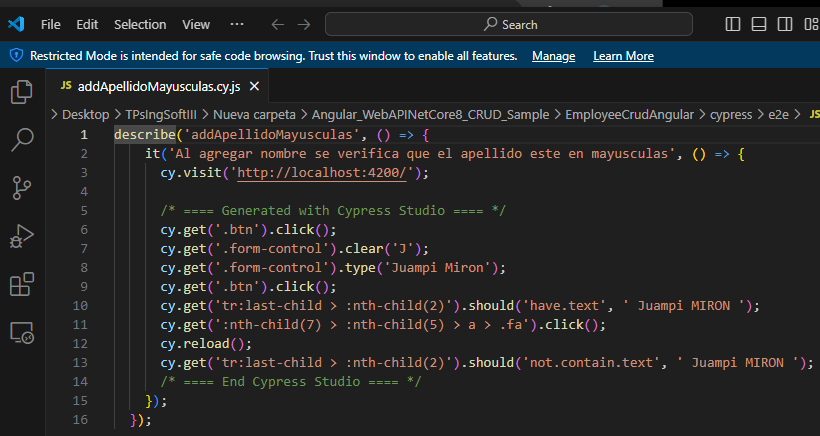
1. Desafíos:

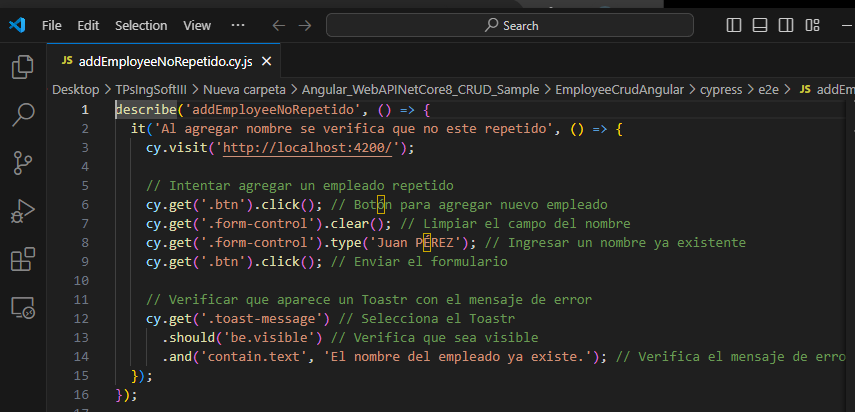
* Integrar en el pipeline SonarCloud para nuestro proyecto Angular, mostrar el resultado obtenido en SonarCloud

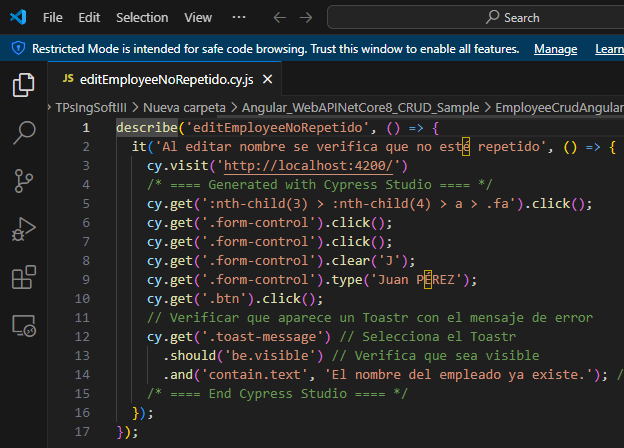


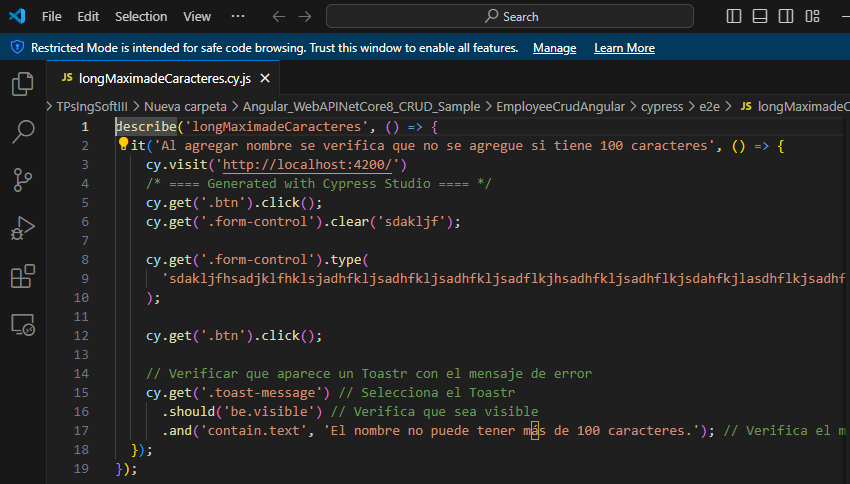


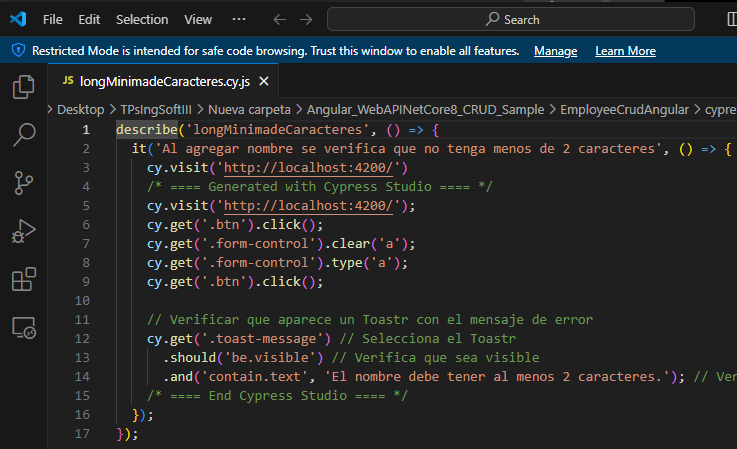
* Implementar en Cypress pruebas de integración que incluya los casos desarrollados como pruebas unitarias del front en el TP06.

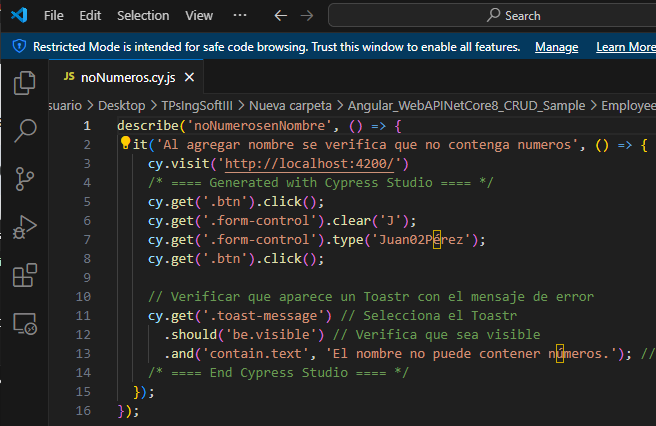












* Incorporar al pipeline de Deploy la ejecución de las pruebas de integración y la visualización de sus resultados.

Hasta el momento no fue posible lograr el deploy de la webapp.