DOCUMENTACIÓN DE ERRORES, USANDO LINTER

ESTUDIANTE

YADIR ADRIAN MUÑOZ RIASCOS

PROFESOR: BRAYAN ARCOS

INSTITUTO TECNOLOGICO DEL PUTUMAYO

PROGRAMACIÓN BACKEND

MOCOA-PUTUMAYO

2024

Contenido

[Primeros pasos 4](#_Toc177911236)

[Dependencias y scripts de desarrollo y documentación de las mismas 5](#_Toc177911237)

[eslint 5](#_Toc177911238)

[typescript-eslint/eslint-plugin 5](#_Toc177911239)

[typescript-eslint/parser 5](#_Toc177911240)

[Eslint 5](#_Toc177911241)

[eslint-config-airbnb-base 5](#_Toc177911242)

[eslint-config-airbnb-typescript 6](#_Toc177911243)

[eslint-plugin-import 6](#_Toc177911244)

[Globals 6](#_Toc177911245)

[Typescript 6](#_Toc177911246)

[typescript-eslint 6](#_Toc177911247)

[test 6](#_Toc177911248)

[Lint 7](#_Toc177911249)

[watch 7](#_Toc177911250)

[start 7](#_Toc177911251)

[build 7](#_Toc177911252)

[Documentación de errores del archivo examplesTypes.ts 7](#_Toc177911253)

[Error 1: 9](#_Toc177911254)

[Error 2: 9](#_Toc177911255)

[Error 3: 10](#_Toc177911256)

[Error 4: 11](#_Toc177911257)

[Error 5: 11](#_Toc177911258)

[Warning 1: 13](#_Toc177911259)

[Warning 2: 13](#_Toc177911260)

[Ilustración 1 4](#_Toc177911261)

[Ilustración 2 7](#_Toc177911262)

[Ilustración 3 8](#_Toc177911263)

[Ilustración 4 8](#_Toc177911264)

[Ilustración 5 9](#_Toc177911265)

[Ilustración 6 9](#_Toc177911266)

[Ilustración 7 9](#_Toc177911267)

[Ilustración 8 10](#_Toc177911268)

[Ilustración 9 10](#_Toc177911269)

[Ilustración 10 10](#_Toc177911270)

[Ilustración 11 11](#_Toc177911271)

[Ilustración 12 11](#_Toc177911272)

[Ilustración 13 11](#_Toc177911273)

[Ilustración 14 12](#_Toc177911274)

[Ilustración 15 12](#_Toc177911275)

[Ilustración 16 12](#_Toc177911276)

[Ilustración 17 12](#_Toc177911277)

[Ilustración 18 13](#_Toc177911278)

[Ilustración 19 13](#_Toc177911279)

[Ilustración 20 14](#_Toc177911280)

# Primeros pasos

Lo primero que debemos tener es un proyecto bien configurado y estructurado, teniendo en cuenta las buenas prácticas de programación, siguiendo todos los pasos proporcionados por el profesor durante las sesiones y material compartido.

Una ves se cumplido el primer paso, el cual es tener un proyecto correctamente configurado usando typescript y eslint, podemos realizar una comprobación al código del proyecto para verificar que tengamos un código limpio.

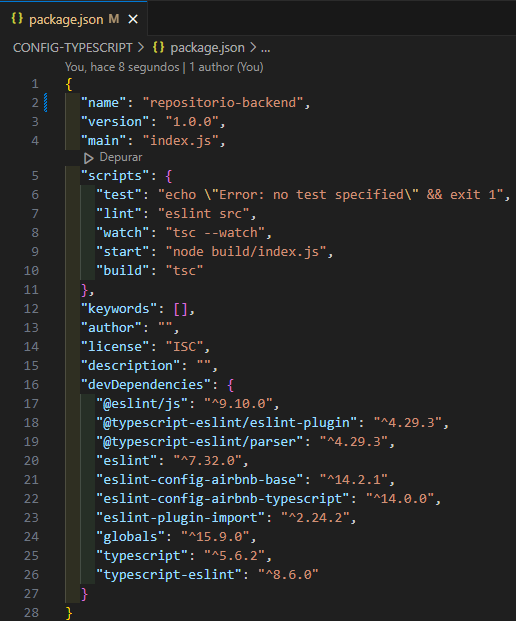
A continuación, mostrare las dependencias y los scripts que se estoy usando en mi proyecto:  


Ilustración 1

# Dependencias y scripts de desarrollo y documentación de las mismas

Como se puede observar en la ilustración 1, tengo las siguientes dependencias y Scripts:

* eslint
* typescript-eslint/eslint-plugin
* typescript-eslint/parser
* eslint
* eslint-config-airbnb-base
* eslint-config-airbnb-typescript
* eslint-plugin-import
* globals
* typescript
* typescript-eslint

## eslint

Este paquete es parte de **ESLint**, que proporciona las reglas necesarias para analizar archivos JavaScript. Ayuda a encontrar y corregir errores en el código mediante la aplicación de reglas de estilo y convenciones específicas.

## typescript-eslint/eslint-plugin

Es un **plugin** para ESLint que agrega un conjunto de reglas específicas para **TypeScript**. Permite que ESLint analice archivos TypeScript y aplique reglas adicionales que son importantes para este lenguaje.

## typescript-eslint/parser

Es un **parser** que permite a ESLint comprender y analizar archivos TypeScript. Normalmente, ESLint solo puede trabajar con JavaScript, pero con este parser es capaz de procesar y analizar archivos .ts

## Eslint

ESLint es una herramienta que analiza tu código para detectar errores, malas prácticas y violaciones de estilo. Es configurable y permite crear reglas personalizadas o usar configuraciones prediseñadas.

## eslint-config-airbnb-base

Es un conjunto de reglas de ESLint que sigue la **guía de estilo de código de Airbnb** para JavaScript. Este paquete incluye reglas estrictas que se utilizan comúnmente en el entorno empresarial.

## eslint-config-airbnb-typescript

Esta configuración extiende las reglas de eslint-config-airbnb-base para adaptarse al uso de **TypeScript**. Ofrece reglas adicionales que garantizan que el código TypeScript siga las mejores prácticas.

## eslint-plugin-import

Este plugin para ESLint ayuda a **gestionar las declaraciones de importación** en tu proyecto. Asegura que los módulos se importen correctamente, que no haya módulos innecesarios y que las rutas de los archivos sean válidas.

## Globals

Ayuda a ESLint a identificar variables globales y a evitar errores de definición de variables, especialmente en entornos como Node.js o navegadores.

## Typescript

Permite escribir código JavaScript más seguro y predecible, gracias al tipado estático y a las herramientas de autocompletado y refactorización que proporciona.

## typescript-eslint

Permite aplicar reglas de ESLint a los archivos TypeScript, combinando las ventajas del linting con las capacidades avanzadas del compilador de TypeScript.

Scripts que tengo definidos:

* test
* lint
* watch
* start
* build

## test

Este script está configurado por defecto para mostrar un mensaje de error indicando que no se han definido pruebas (tests) para el proyecto. Luego, termina el proceso con un código de salida 1, que indica un error.

## Lint

Manteniene el código limpio, coherente y conforme a las reglas de calidad. Sirve para verificar que el código cumpla con las reglas establecidas en las configuraciones de ESLint, incluyendo las reglas específicas de TypeScript.

## watch

Facilita el desarrollo continuo al permitir que el código TypeScript se compile de manera automática sin necesidad de ejecutar manualmente el comando de compilación cada vez que se modifica un archivo.

## start

Iniciar la aplicación en producción. Después de compilar el código TypeScript, el archivo generado es ejecutado por **Node.js** para que la aplicación esté disponible.

## build

Realiza la compilación del proyecto, transformando todos los archivos .ts a .js para que puedan ser ejecutados en entornos donde solo se acepte JavaScript (como Node.js o navegadores).

# Documentación de errores del archivo examplesTypes.ts

Para este caso tenemos el siguiente ejemplo que se realizó en clases:

Texto

Descripción generada automáticamente

Ilustración 2

Como se puede observar en la imagen, tengo un archivo **examplesTypes.ts** el contiene unas variables de te tipo **number**, **string**, **boolean** y **any** con su respectivo valor.

Al ejecutar el código nos muestra en consola los resultados de forma correcta tal como se debería mostrar según las variables definidas con su respectivo valor asignado. En la siguiente ilustración se muestra los resultados de la ejecución:

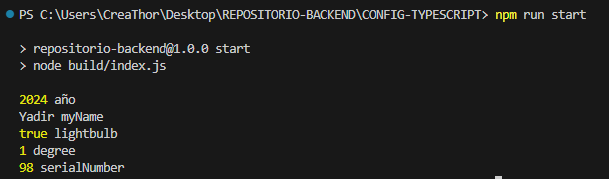


Ilustración 3

La finalidad de esta documentación es poder identificar errores usando el comando npm run lint ya que este nos proporciona reglas necesarias para analizar nuestro código ayudando a encontrar y corregir errores.

Al ejecutar el comando npm run lint este nos muestra las recomendaciones o sugerencias que debemos adoptar en nuestro código, nos señala que fragmento o parte del código debemos hacer una mejora para tener un código mas limpio y fluido. A continuación, en la siguiente ilustración se muestra las recomendaciones que debo adoptar en mi código:

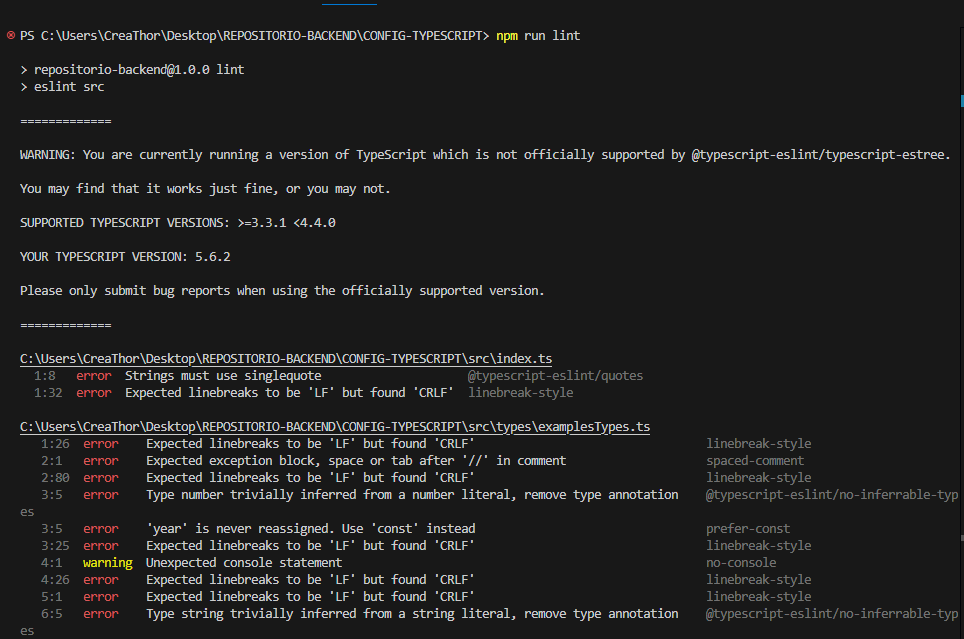


Ilustración 4

Error 1: Para solucionar el primer error:

error Strings must use singlequote @typescript-eslint/quotes que se muestra en la ilustración 4, sería la siguiente: lo que nos sugiere es que usemos comillas simples (‘’) en lugar de comillas dobles (“ ”) en la importación del archivo **examplesTypes** en el archivo **index.ts.** A continuación, mostrare la solución que se implemento para corregir el error:

inicialmente la importación estaba de la siguiente manera:

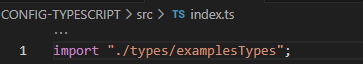
 x

Ilustración 5

Solución implementada:

**✓**

Ilustración 6

Error 2: El siguiente error o sugerencia que se presenta es el siguiente:

error Expected linebreaks to be 'LF' but found 'CRLF' linebreak-style

indica que tu archivo utiliza saltos de línea en el formato CRLF. La solución que implemente para este error fue modificar el archivo .eslintrc y agrega la regla linebreak-style como se muestra a continuación:

reglas antes de corregir el error:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza mediax

Ilustración 7

Regla implementada para poder evitar el error:

**✓**

Ilustración 8

También se puede solucionar de otra manera mas sencilla y es haciendo lo siguiente:   
En la esquina inferior derecha de la ventana de VS Code, hay un botón que muestra el tipo de saltos de línea del archivo, el cual dice CRLF.

Se da clic en este botón y selecciona LF en el menú desplegable. En la siguiente ilustración se puede observar la opción:

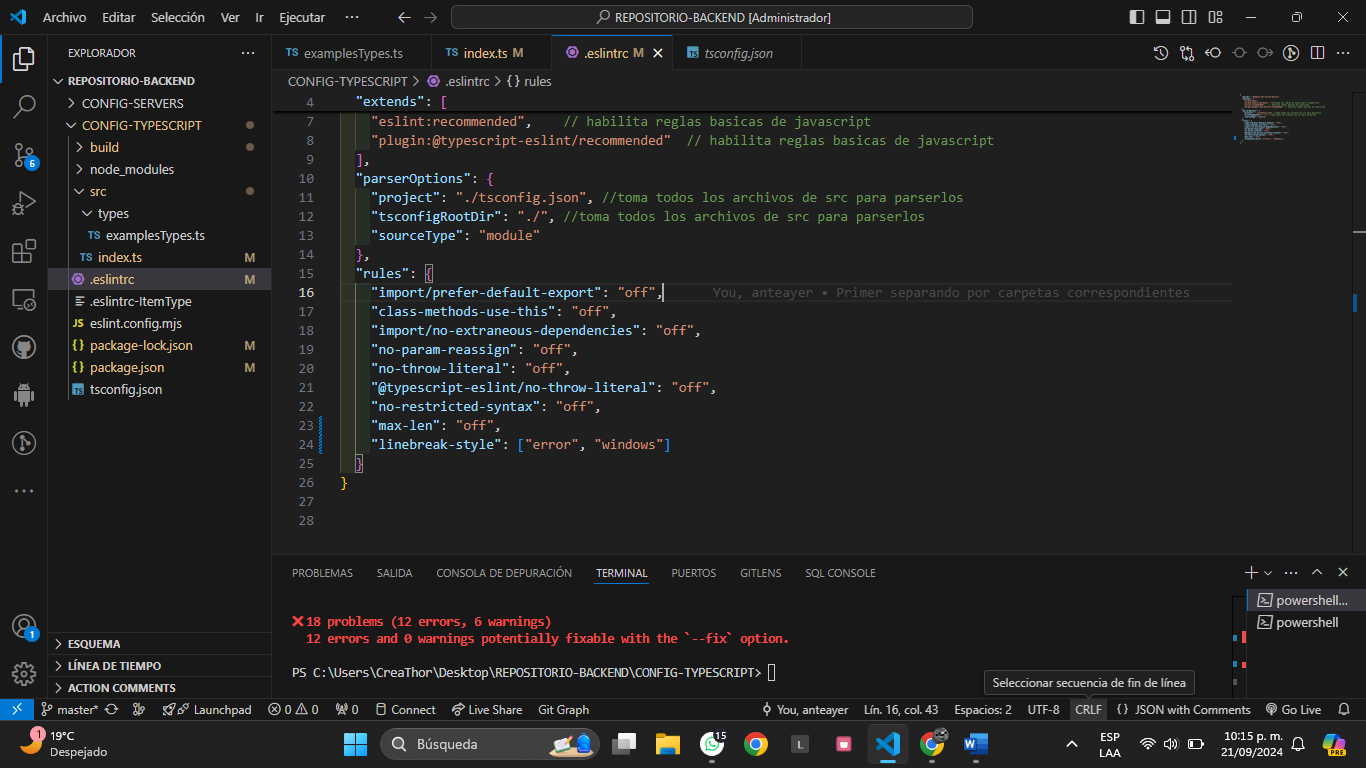


Ilustración 9

Luego de solucionar los dos primeros errores, compile, seguidamente ejecute el comando npm run lint para ver las recomendaciones o errores que me mostraba, y me di cuenta que el nuero de errores de sintaxis había reducido significativamente, ahora solo muestra las siguientes sugerencias para abordar y corregir:

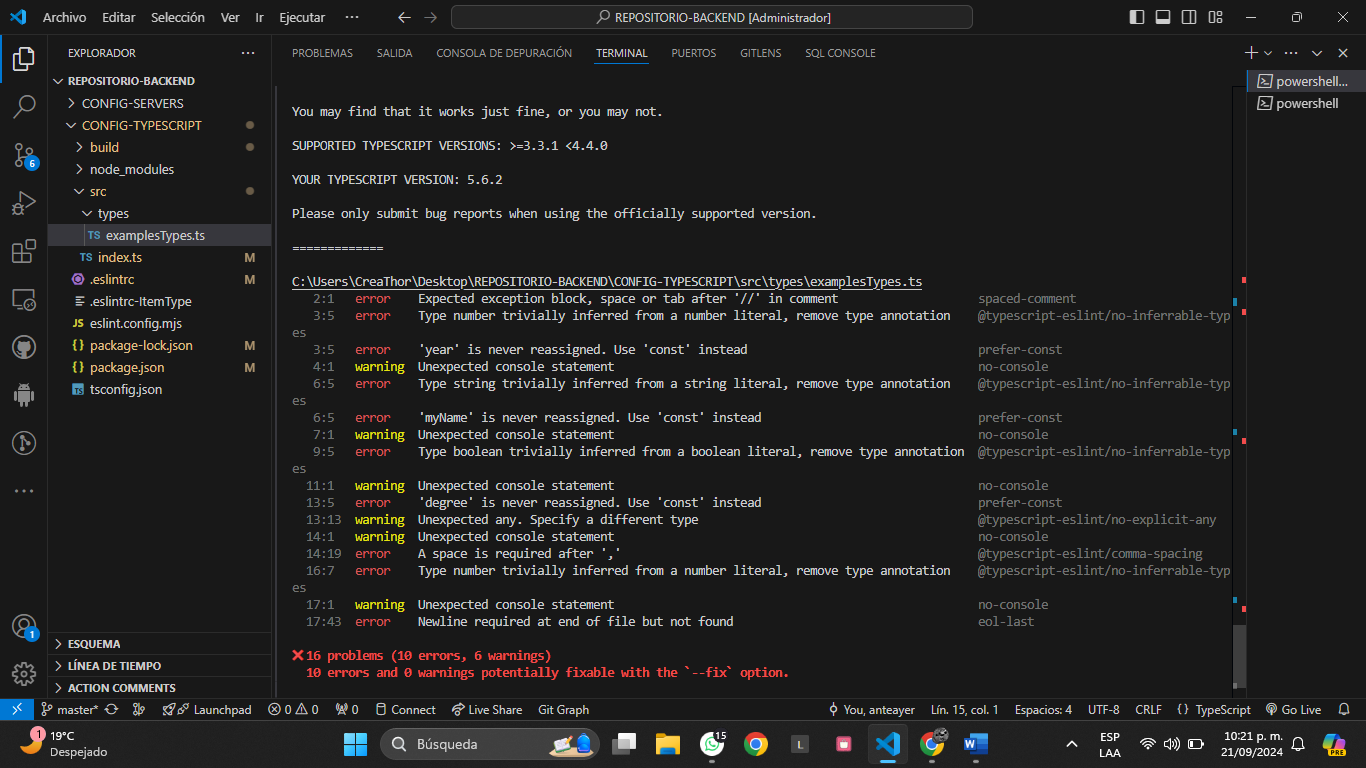


Ilustración 10

Error 3: para abordar el error:

error Expected exception block, space or tab after '//' in comment

significa que **ESLint** espera un espacio o tabulación después de los dos caracteres **//** en los comentarios de una sola línea. Menciona que el error esta relacionado con el archivo **examplesTypes.ts,** como se puede observar en la ilustración 2, estoy usando **//** para comentar, el inconveniente está relacionado con la línea 2, en ese comentario se puede evidenciar que después de **//** no hay un espacio. Para solucionarlo es muy fácil, simplemente añadimos un espacio entre **//** y el comentario, como se puede ver en la siguiente ilustración:

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración 11

Error 4: para abordar el error:

Type number trivially inferred from a number literal, remove type annotation

significa que TypeScript ha detectado una anotación de tipo redundante en el código. Esto ocurre cuando se asigna un tipo explícito a una variable o constante que ya puede inferirse automáticamente a partir del valor que se asignas.

Para solucionarlo, simplemente elimino la anotación de tipo redundante, así:

 **✓**

Ilustración 12

Error seria tener definido la variable **year** de la siguiente manera:

 **x**

Ilustración 13

Error 5: para abordar el error:

error 'year' is never reassigned. Use 'const' instead

indica que he declarado una variable con **let**, pero esa variable nunca cambia su valor después de ser asignada inicialmente. El error esta relacionado con la línea 3, relacionado con la línea anteriormente corregida. Para corregir el inconveniente simplemente cambio **let** por **const**, lo que indica que la variable **year** no será reasignada, así:

**✓**

Ilustración 14

He visto mas recomendaciones acerca de utilizar **const** en ves de **let** ya que el valor que le asignando no lo estoy cambiando. Entonces corregiré los errores relacionados con este mismo patrón y definiré con **const** las variables:



Ilustración 15



Ilustración 16

Error 6: el siguiente error:

error Type string trivially inferred from a string literal, remove type annotation

indica que **TypeScript** detecta que he declarado una variable con una anotación de tipo redundante, porque el tipo ya se puede inferir directamente a partir del valor asignado.

Para solucionar este inconveniente, simplemente quitamos el tipo de dato asignado, ya que la variable actúa como una anotación, infiriendo el tipo de dato que se espera:

Texto

Descripción generada automáticamente**✓**

Ilustración 17

Antes:

Texto

Descripción generada automáticamente***X***

Ilustración 18

Warning 1: para solucionar la sugerencia:

**warning Unexpected console statement**

Las declaraciones **console** se suelen utilizar para depurar, pero dejarlas en el código final puede generar problemas de rendimiento, exponer información innecesaria o comprometer la seguridad en aplicaciones en producción.

Dado que no estamos en un entorno de producción podemos desactivar esta regla usando la regla en el archivo **.eslintrc**:

"no-console": "off"

Regla añadida:



Ilustración 19

Warning 2: para solucionar la sugerencia:

warning Unexpected any. Specify a different type

se produce cuando **TypeScript** detecta que se está utilizando el tipo **any**, pero se espera que se especifique un tipo más preciso. Esto ocurre porque **any** desactiva la verificación estricta de tipos, permitiendo cualquier valor.

Para evitar este tipo de sugerencia podemos cambiar el tipo any a un tipo de dato más específico como **string**, **number, boolean.** según sea el caso**.**

Por buenas prácticas también es bueno usar **unknown,** ya que requiere que se verifique con una condición el valor antes de ser usado.

Aborde la solución cambiando **any** por **number:**



Ilustración 20

**✓**

Aquí finaliza la documentación de los errores y recomendaciones que nos ofrece las reglas de TypeScript para un código limpio y permitiendo la optimización de un proyecto.

Anexo link de mi repositorio de **hithub:**

https://github.com/1007748140/REPOSITORIO-BACKEND