DOCUMENTACIÓN

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)

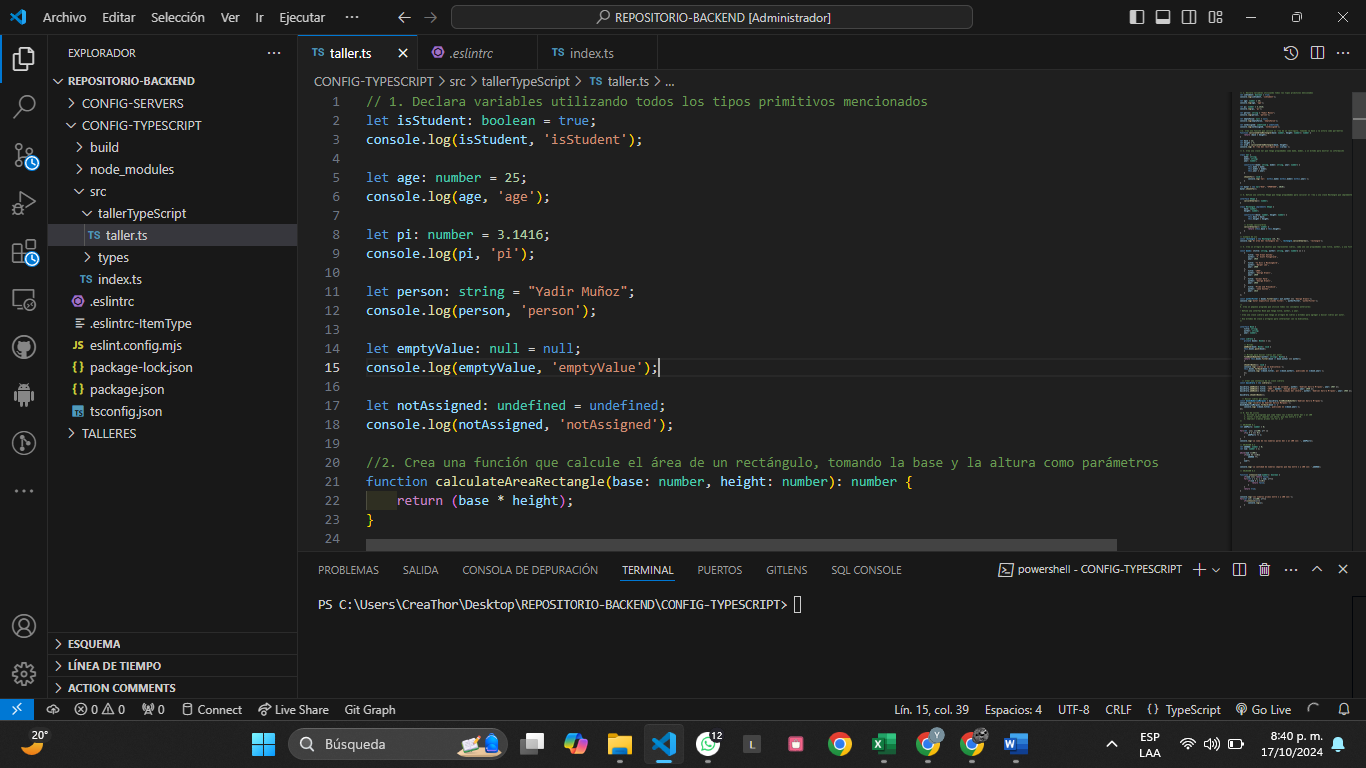
[Error 4: 11](#_Toc177911257)

[Error 5: 11](#_Toc177911258)

[Warning 1: 13](#_Toc177911259)

[Warning 2: 13](#_Toc177911260)

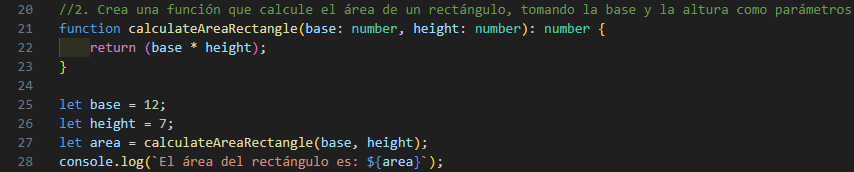
## Ejercicio 1



En este código se están declarando variables en TypeScript utilizando diferentes tipos de datos primitivos, seguido de la impresión de cada una en la consola para verificar sus valores:

1. **Booleano (boolean)**: La variable isStudent se inicializa con el valor true, que representa un valor de tipo booleano (verdadero o falso). Se utiliza console.log() para imprimir este valor con una etiqueta.
2. **Número (number)**: La variable age contiene el número 25, y pi almacena el valor 3.1416, ambos son números. Se utilizan números de tipo decimal en estos ejemplos.
3. **Cadena de texto (string)**: La variable person contiene el texto "Yadir Muñoz", que es una cadena de caracteres. Las cadenas en TypeScript se definen con comillas simples o dobles.
4. **Nulo (null)**: La variable emptyValue se asigna con el valor especial null, que indica la ausencia de valor intencionalmente.
5. **Indefinido (undefined)**: La variable notAssigned se asigna con undefined, que indica que la variable ha sido declarada, pero no se le ha dado un valor explícito.

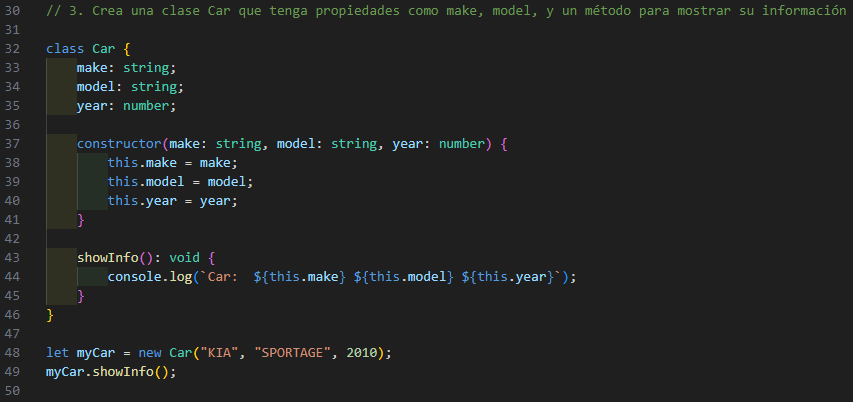
## Ejercicio 2



En este código se crea una función que calcula el área de un rectángulo, tomando como parámetros la base y la altura:

1. **Declaración de la función calculateAreaRectangle**: Esta función recibe dos argumentos: base y height, ambos de tipo number. La función devuelve un número que representa el área del rectángulo, calculada mediante la fórmula matemática base \* height.
2. **Asignación de valores**: Se declaran las variables base y height con los valores 12 y 7 respectivamente. Estos valores se utilizarán como entrada para la función.
3. **Llamada a la función**: La función calculateAreaRectangle es llamada pasando como argumentos base y height. El resultado, que es el área del rectángulo, se almacena en la variable area.
4. **Impresión del resultado**: Se utiliza console.log() para mostrar el resultado de la operación en la consola, con un mensaje que indica "El área del rectángulo es" seguido del valor calculado.

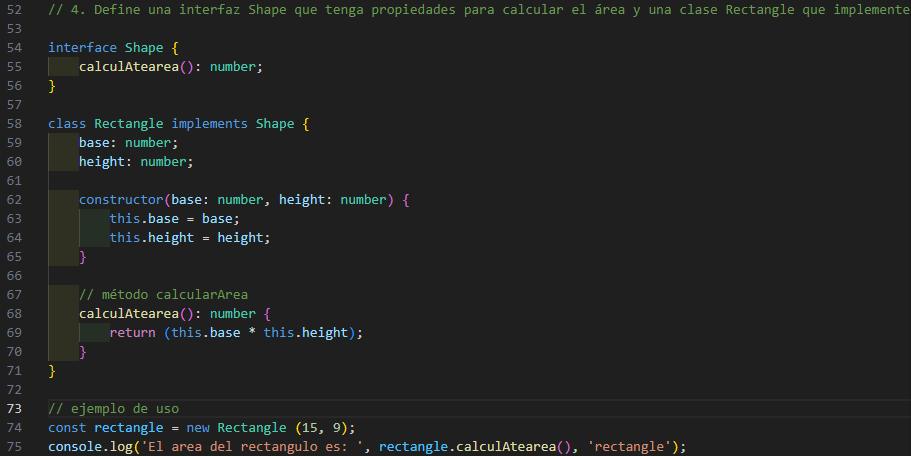
## Ejercicio 3



En este código se define una clase Car que representa un automóvil con propiedades y métodos:

1. **Definición de la clase Car**: La clase tiene tres propiedades: make (marca del auto), model (modelo del auto), y year (año de fabricación). Todas son de tipo string o number.
2. **Constructor**: El método constructor se usa para inicializar las propiedades de la clase. Cuando se crea un nuevo objeto de la clase Car, se le asignan valores a make, model y year mediante los parámetros que recibe el constructor.
3. **Método showInfo**: Este método imprime en consola la información del coche en el formato "Car: [make] [model] [year]". No devuelve ningún valor, solo ejecuta una acción, por eso su tipo de retorno es void.
4. **Creación de un objeto**: Se crea una instancia de la clase Car llamada myCar, asignándole los valores "KIA" para make, "SPORTAGE" para model y 2010 para year.
5. **Llamada al método showInfo**: Finalmente, se invoca el método showInfo para mostrar la información del coche almacenado en myCar en la consola.

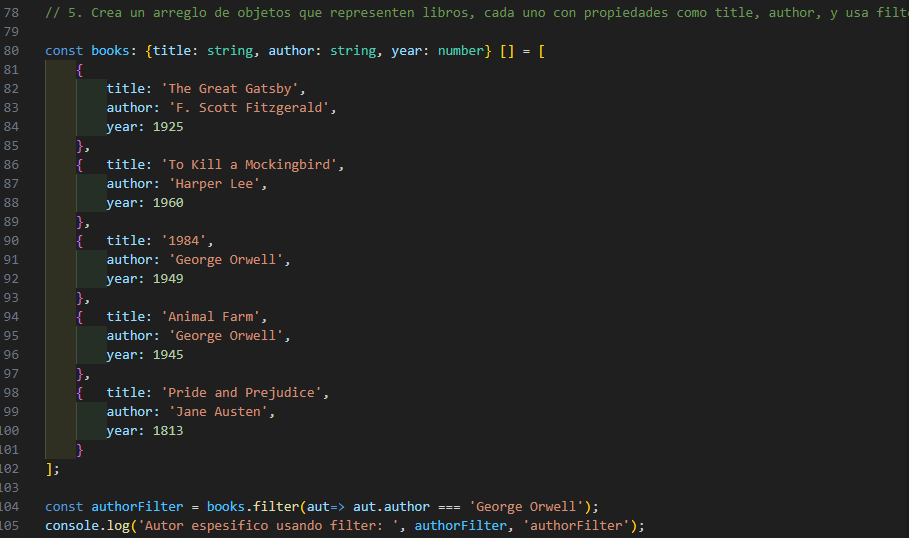
## Ejercicio 4



En este código se define una interfaz y una clase que implementa dicha interfaz para calcular el área de un rectángulo:

1. **Definición de la interfaz Shape**: La interfaz Shape actúa como un contrato que obliga a las clases que la implementen a tener un método llamado calculAtearea(), que devuelve un número. Este método debe calcular el área de la figura geométrica.
2. **Clase Rectangle que implementa la interfaz Shape**:
   * La clase Rectangle implementa la interfaz Shape, lo que significa que debe definir el método calculAtearea().
   * La clase tiene dos propiedades: base y height, que representan la base y la altura del rectángulo.
   * El constructor de la clase acepta base y height como parámetros y los asigna a las propiedades correspondientes.
3. **Método calculAtearea()**: Este método, definido en la clase Rectangle, cumple con la interfaz Shape. Calcula el área del rectángulo utilizando la fórmula base \* height y devuelve el resultado.
4. **Creación de un objeto Rectangle**: Se crea una instancia de Rectangle llamada rectangle con una base de 15 y una altura de 9.
5. **Cálculo del área**: Finalmente, se llama al método calculAtearea() para calcular e imprimir el área del rectángulo en la consola.

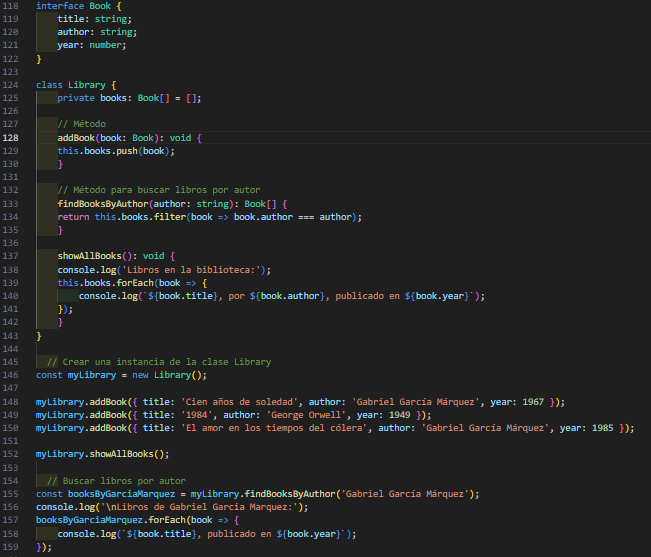
## Ejercicio 5



En este código se trabaja con un arreglo de objetos que representan libros, y se utiliza el método filter para obtener libros de un autor específico:

1. **Declaración del arreglo books**: Se crea un arreglo de objetos, donde cada objeto representa un libro con las propiedades title (título del libro), author (autor) y year (año de publicación). Este arreglo contiene varios libros de diferentes autores.
2. **Uso del método filter**: El método filter se aplica al arreglo books para crear un nuevo arreglo que contenga solo los libros cuyo autor sea "George Orwell". La función de flecha (aut => aut.author === 'George Orwell') verifica si el campo author del libro coincide con el autor especificado.
3. **Resultado filtrado**: El resultado del filtro se almacena en la variable authorFilter, que contiene solo los libros escritos por George Orwell.
4. **Impresión del resultado**: Finalmente, se usa console.log() para mostrar el arreglo filtrado en la consola, con un mensaje que indica que se ha filtrado por un autor específico.

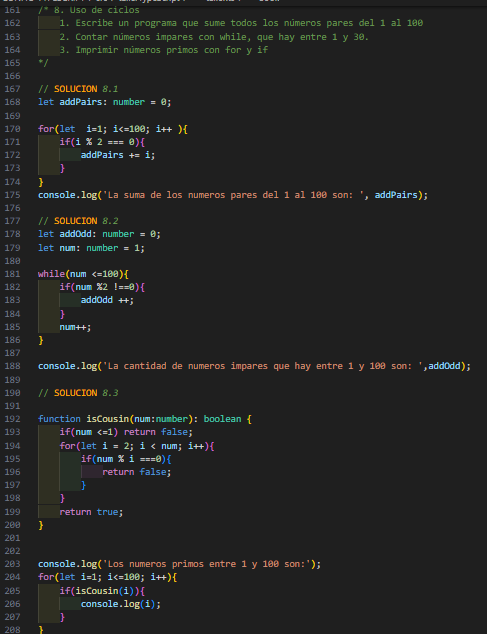
## Ejercicio 6



En este código se define una clase para gestionar una colección de libros en una biblioteca, con métodos para agregar y buscar libros:

1. **Interfaz Book**: Se define la interfaz Book, que especifica la estructura de un objeto libro con tres propiedades: title (título del libro), author (autor) y year (año de publicación). Esto asegura que cualquier libro añadido a la biblioteca siga esta estructura.
2. **Clase Library**:
   * La clase tiene una propiedad privada books, que es un arreglo de objetos del tipo Book. Al ser privada, solo se puede acceder a esta propiedad desde dentro de la clase.
   * **Método addBook**: Este método recibe un objeto de tipo Book como parámetro y lo añade al arreglo books usando el método push().
   * **Método findBooksByAuthor**: Este método busca y devuelve los libros que coinciden con el autor proporcionado como argumento. Utiliza el método filter() para recorrer el arreglo books y comparar el campo author de cada libro.
   * **Método showAllBooks**: Este método imprime todos los libros almacenados en la biblioteca, recorriendo el arreglo books con forEach() y mostrando el título, autor y año de cada libro.
3. **Uso de la clase Library**:
   * Se crea una instancia de la clase Library llamada myLibrary.
   * Se agregan tres libros a la biblioteca usando el método addBook().
   * El método showAllBooks() se llama para mostrar todos los libros en la biblioteca.
4. **Búsqueda de libros por autor**:
   * Se utiliza el método findBooksByAuthor() para obtener todos los libros escritos por "Gabriel García Márquez".
   * Los libros encontrados se imprimen en la consola mostrando el título y el año de publicación.

## Ejercicio 8



Este código realiza tres tareas utilizando ciclos para trabajar con números pares, impares y primos:

1. **Suma de números pares del 1 al 100 (Solución 8.1)**:
   * Se inicializa la variable addPairs en 0 para almacenar la suma de los números pares.
   * Un ciclo for recorre los números del 1 al 100. Dentro del ciclo, se verifica si un número es par con la condición i % 2 === 0. Si es par, se suma a addPairs.
   * Al final, se imprime la suma de todos los números pares.
2. **Contar números impares entre 1 y 100 usando while (Solución 8.2)**:
   * Se inicializa la variable addOdd en 0 para contar los números impares y la variable num en 1 para empezar el conteo.
   * Se usa un ciclo while que continúa mientras num sea menor o igual a 100. Dentro del ciclo, se verifica si el número es impar usando num % 2 !== 0. Si es impar, se incrementa el contador addOdd.
   * Al final, se imprime la cantidad de números impares.
3. **Imprimir números primos entre 1 y 100 (Solución 8.3)**:
   * Se define la función isCousin(num) que verifica si un número es primo. Si el número es menor o igual a 1, no es primo. Si es divisible por algún número entre 2 y num-1, tampoco es primo.
   * Un ciclo for recorre los números del 1 al 100. Para cada número, llama a la función isCousin(), y si la función devuelve true, el número es primo y se imprime.

Enlace del video sustentación:

<https://drive.google.com/file/d/1uTkmeTIZ2sAB2bmtkI8v-32jNb-wmJg1/view?usp=sharing>

Enlace de GitHub:

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