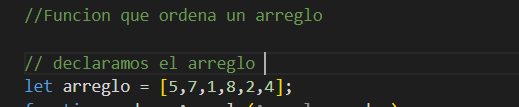


**Desarrollo de la función solicitada.**

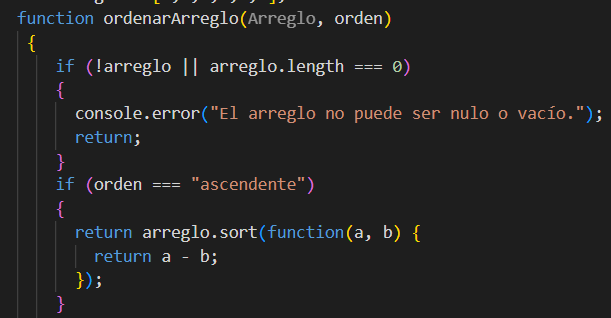
Para este ejercicio lo realicé en dos implementaciones usando el lenguaje de JavaScript.

**Implementación 1**

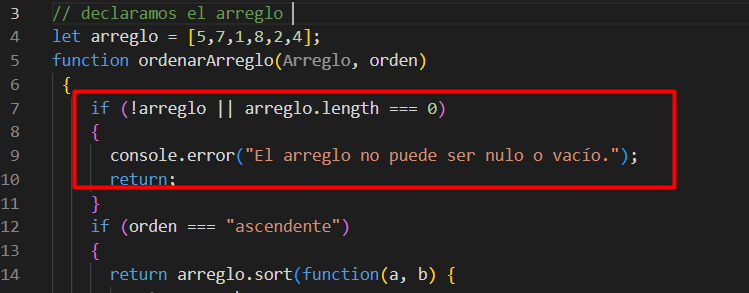
Declaramos un arreglo con 6 elementos



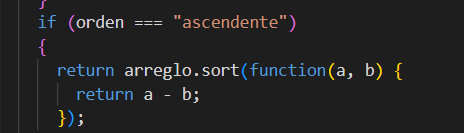
Realizamos una función de ordenamiento que reciba el arreglo anteriormente declarado de 6 elementos



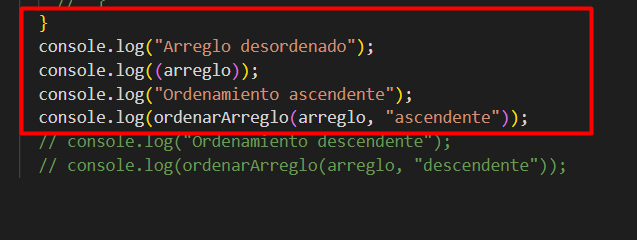
Se realiza una condición en donde valide si el arreglo este vacío muestre un mensaje de error



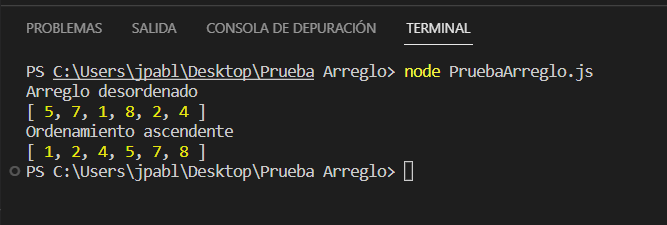
Una vez ejecutada la condición y si existen elementos en el arreglo procede a la condición de ordenamiento para este caso es ascendente.



Una vez ejecutado la condición de ordenamiento se imprime en pantalla el resultado



Validamos en pantalla el resultado del requerimiento.



**Código implementado**

**En parte del código esta comentado donde se realizaba ordenamiento descendente**

**// Declaramos el arreglo**

let arreglo = [5,7,1,8,2,4];

function ordenarArreglo(Arreglo, orden)

{

if (!arreglo || arreglo.length === 0)

{

console.error("El arreglo no puede ser nulo o vacío.");

return;

}

if (orden === "ascendente")

{

return arreglo.sort(function(a, b) {

return a - b;

});

}

**// else if (orden === "descendente")**

**// {**

**// return arreglo.sort(function(a, b) {**

**// return b - a;**

**// });**

**// }**

}

console.log("Arreglo desordenado");

console.log((arreglo));

console.log("Ordenamiento ascendente");

console.log(ordenarArreglo(arreglo, "ascendente"));

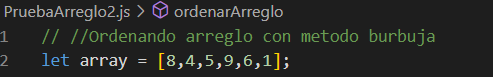
**// console.log("Ordenamiento descendente");**

**// console.log(ordenarArreglo(arreglo, "descendente"));**

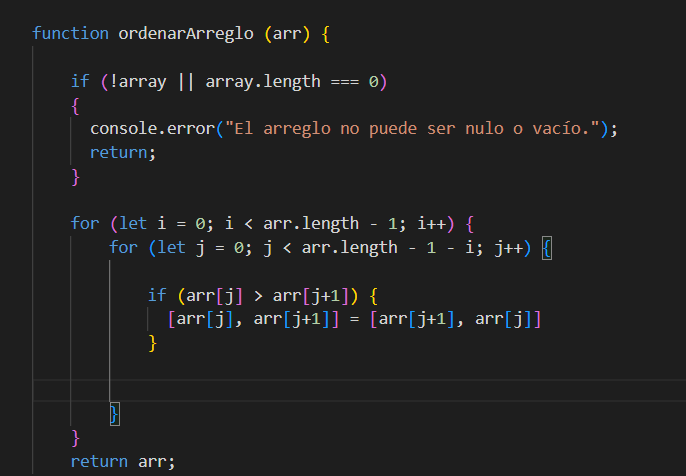
**Implementación 2**

La siguiente forma la realizo por la siguiente razón, en el enunciado del ejercicio mencionan que no se debe usar funciones automáticas propias del lenguaje, considerando ese punto procedo con la implementación del método de ordenamiento Burbuja.

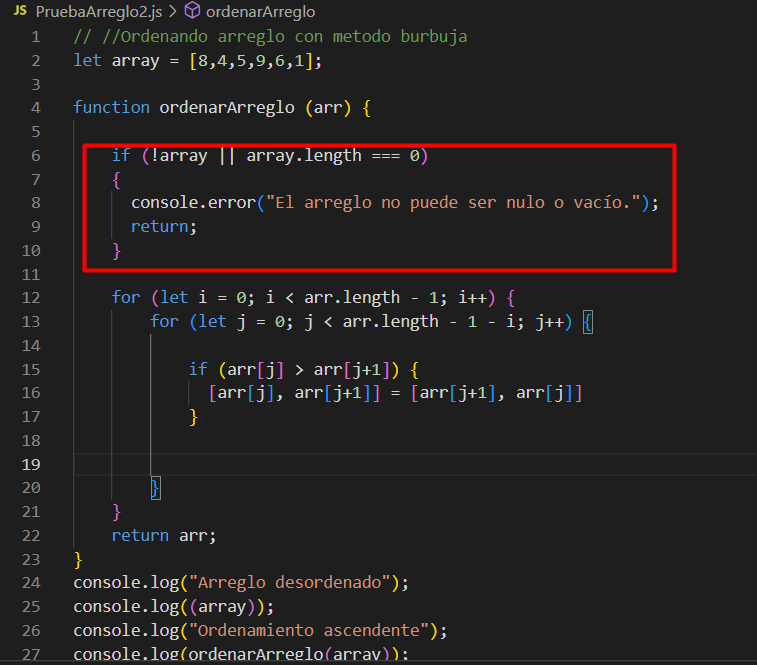
Declaramos un arreglo con 6 elementos



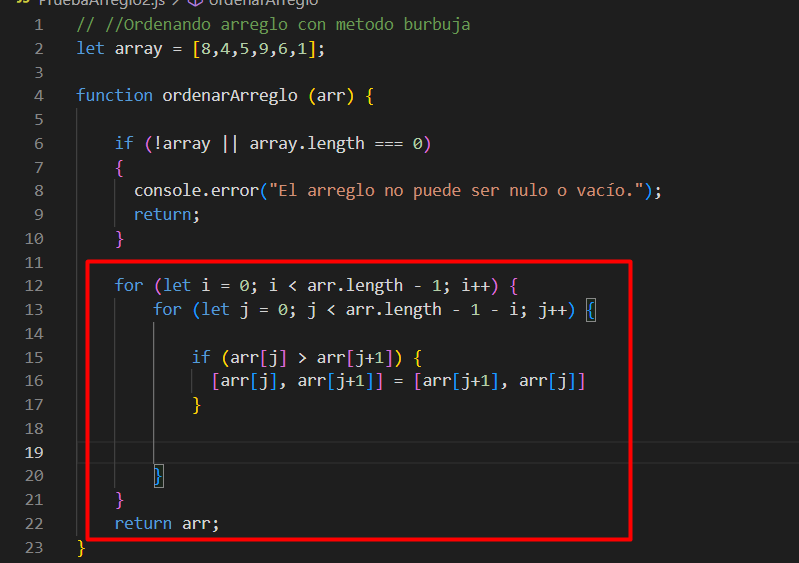
Realizamos una función de ordenamiento (método de burbuja) que reciba el arreglo anteriormente declarado de 6 elementos.



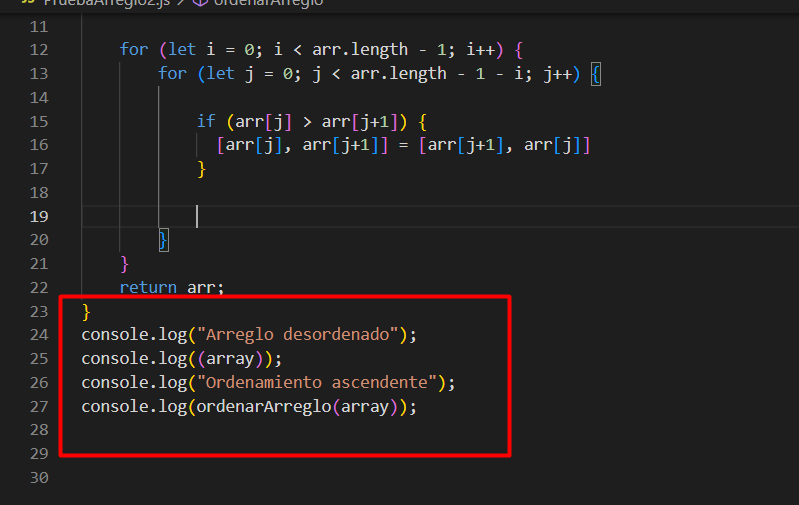
Se realiza una condición en donde valide si el arreglo este vacío muestre un mensaje de error



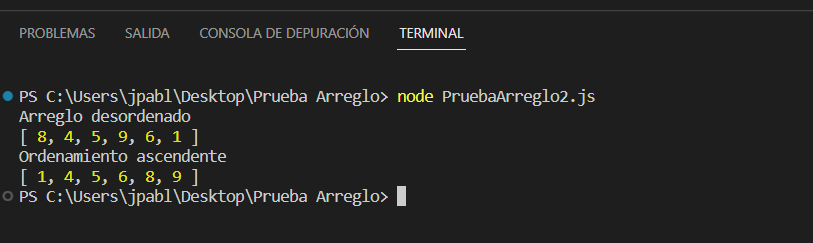
Se aplica el método de ordenamiento para que organice los elementos del array



Una vez ejecutado la condición de ordenamiento se imprime en pantalla el resultado.



Resultados obtenidos en pantalla.



**Código implementado**

//Ordenando arreglo con método burbuja

let array = [8,4,5,9,6,1];

function ordenarArreglo (arr) {

if (!array || array.length === 0)

{

console.error("El arreglo no puede ser nulo o vacío.");

return;

}

for (let i = 0; i < arr.length - 1; i++) {

for (let j = 0; j < arr.length - 1 - i; j++) {

if (arr[j] > arr[j+1]) {

[arr[j], arr[j+1]] = [arr[j+1], arr[j]]

}

}

}

return arr;

}

console.log("Arreglo desordenado");

console.log((array));

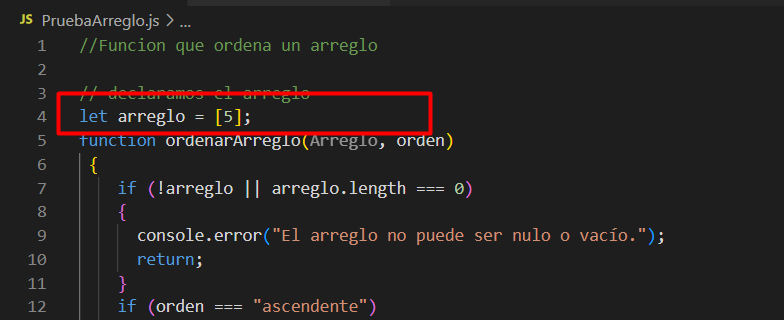
console.log("Ordenamiento ascendente");

console.log(ordenarArreglo(array));

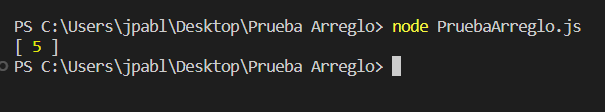
**Pruebas unitarias de la implementación 1**

1. **Verificar que no retorne lista vacía o null.**

Como se observa en el resultado no retorna el arreglo vacío, ya que existe una condición que muestra error al ingresar un arreglo vacío.

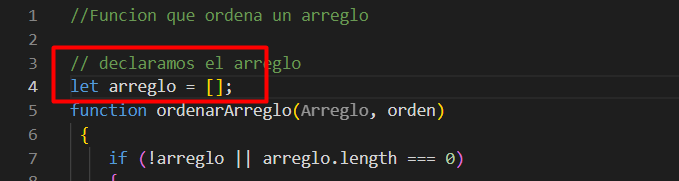


Resultado retornado

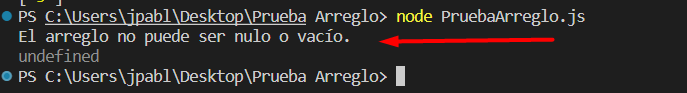


1. **Verificar en caso que venga la lista nula o vacía mensaje de error.**

Se declara el arreglo con 0 elementos.

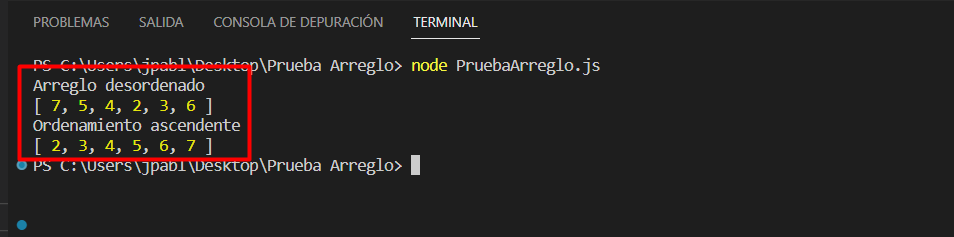


Muestra mensaje de error



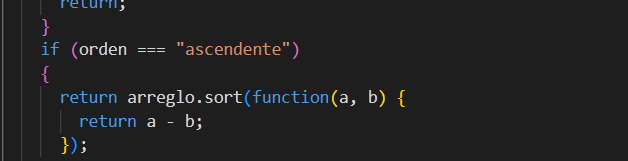
1. **Verificar que la lista devuelta sea de la misma longitud que la lista enviada.**

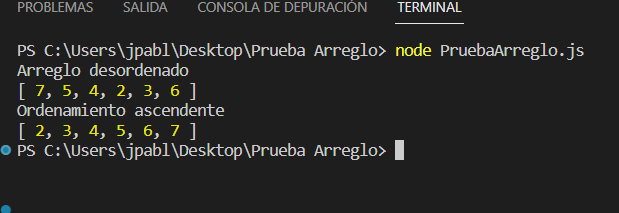
La función recibe un arreglo de 6 elementos y retorna como resultado un arreglo con 6 elementos.



1. **Verificar que la lista devuelta esté ordenada de acuerdo al parámetro enviado.**

El resultado obtenido de acuerdo al ordenamiento es código es el correcto en este caso es ascendente.

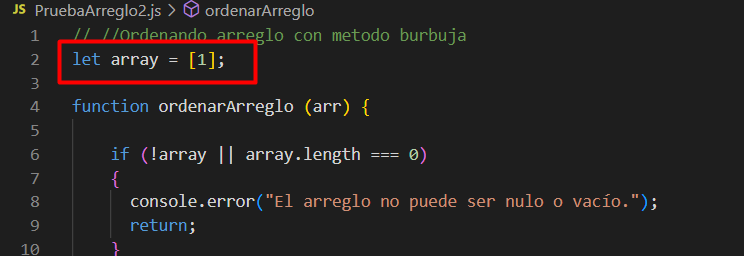




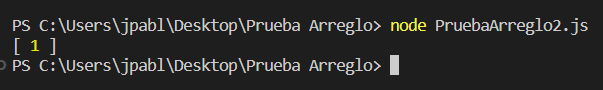
**Pruebas unitarias de la implementación 2**

1. **Verificar que no retorne lista vacía o null.**

Como se observa en el resultado no retorna el arreglo vacío, ya que existe una condición que muestra error al ingresar un arreglo vacío.

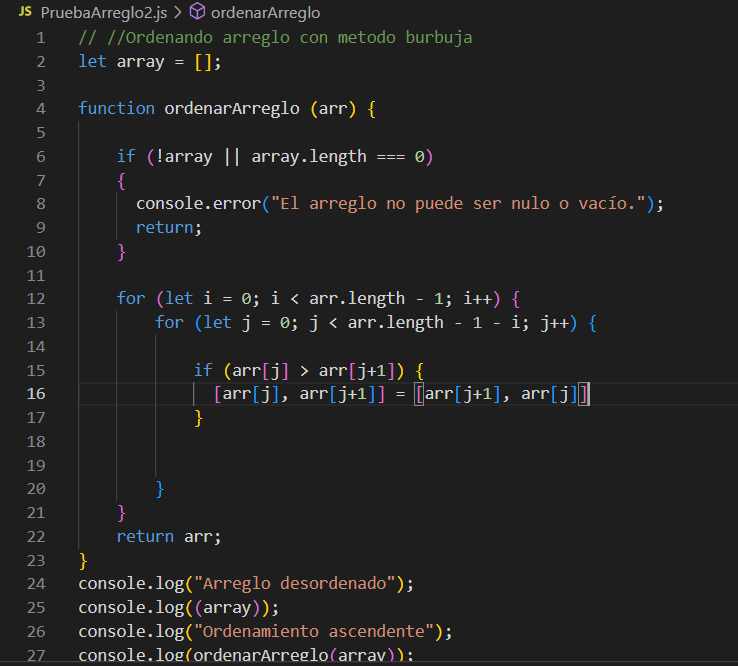


Resultado retornado

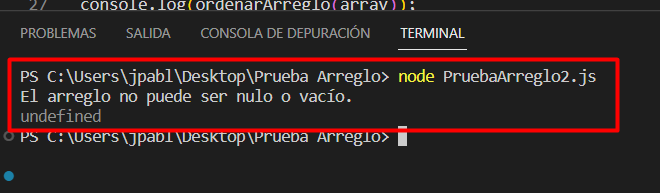


1. **Verificar en caso que venga la lista nula o vacía mensaje de error.**

Se declara el arreglo con 0 elementos.

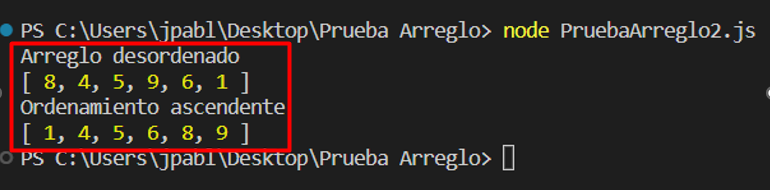


Muestra mensaje de error



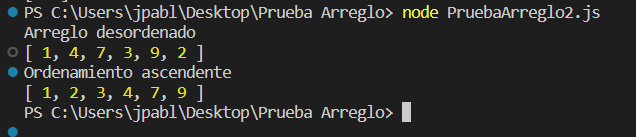
1. **Verificar que la lista devuelta sea de la misma longitud que la lista enviada.**

La función recibe un arreglo de 6 elementos y retorna como resultado un arreglo con 6 elementos.



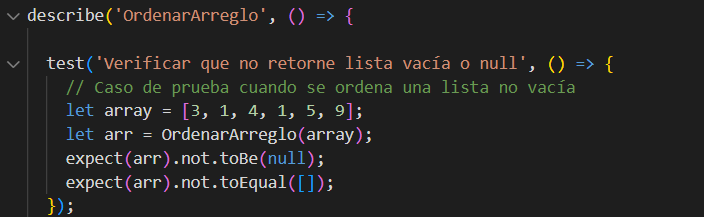
1. **Verificar que la lista devuelta esté ordenada de acuerdo al parámetro enviado.**

El resultado obtenido de acuerdo al ordenamiento es código es el correcto en este caso es ascendente.

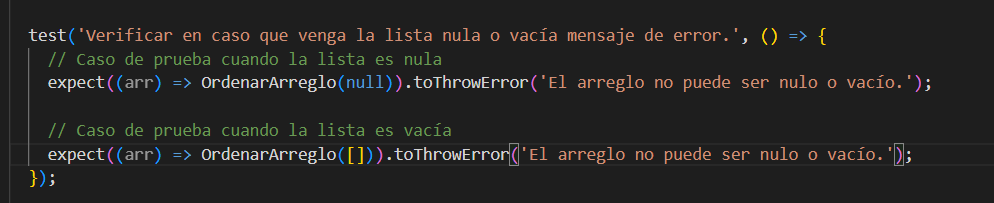


**Utilizado Jets para pruebas unitarias de la implementación 2.**

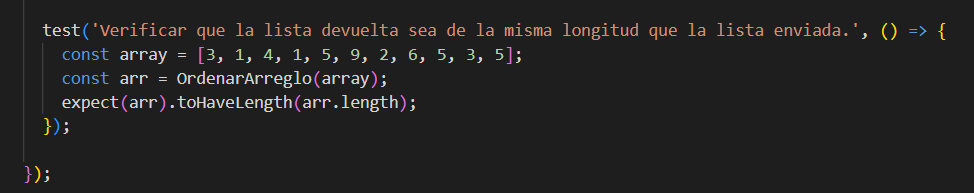
1. Verificar que no retorne lista vacía o null.



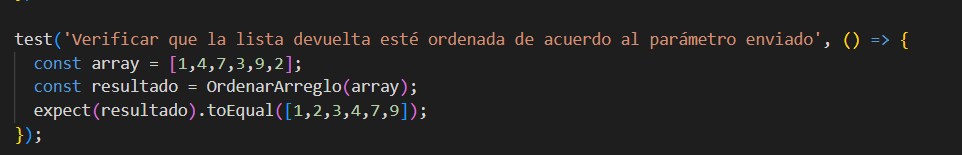
2. Verificar en caso que venga la lista nula o vacía mensaje de error.



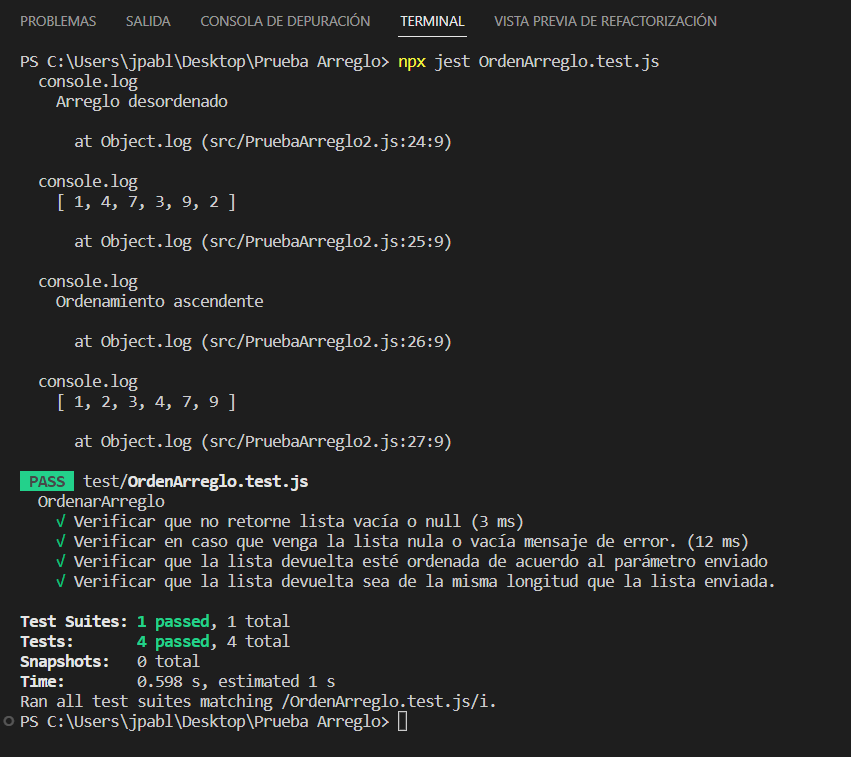
3. Verificar que la lista devuelta sea de la misma longitud que la lista enviada.



4. Verificar que la lista devuelta esté ordenada de acuerdo al parámetro enviado.



Resultados obtenidos aprobados.



**Código implementado para pruebas unitarias**

const {OrdenarArreglo} = require ('../src/PruebaArreglo2')

describe('OrdenarArreglo', () => {

test('Verificar que no retorne lista vacía o null', () => {

// Caso de prueba cuando se ordena una lista no vacía

let array = [3, 1, 4, 1, 5, 9];

let arr = OrdenarArreglo(array);

expect(arr).not.toBe(null);

expect(arr).not.toEqual([]);

});

test('Verificar en caso que venga la lista nula o vacía mensaje de error.', () => {

// Caso de prueba cuando la lista es nula

expect((arr) => OrdenarArreglo(null)).toThrowError('El arreglo no puede ser nulo o vacío.');

// Caso de prueba cuando la lista es vacía

expect((arr) => OrdenarArreglo([])).toThrowError('El arreglo no puede ser nulo o vacío.');

});

test('Verificar que la lista devuelta esté ordenada de acuerdo al parámetro enviado', () => {

const array = [1,4,7,3,9,2];

const resultado = OrdenarArreglo(array);

expect(resultado).toEqual([1,2,3,4,7,9]);

});

test('Verificar que la lista devuelta sea de la misma longitud que la lista enviada.', () => {

const array = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5];

const arr = OrdenarArreglo(array);

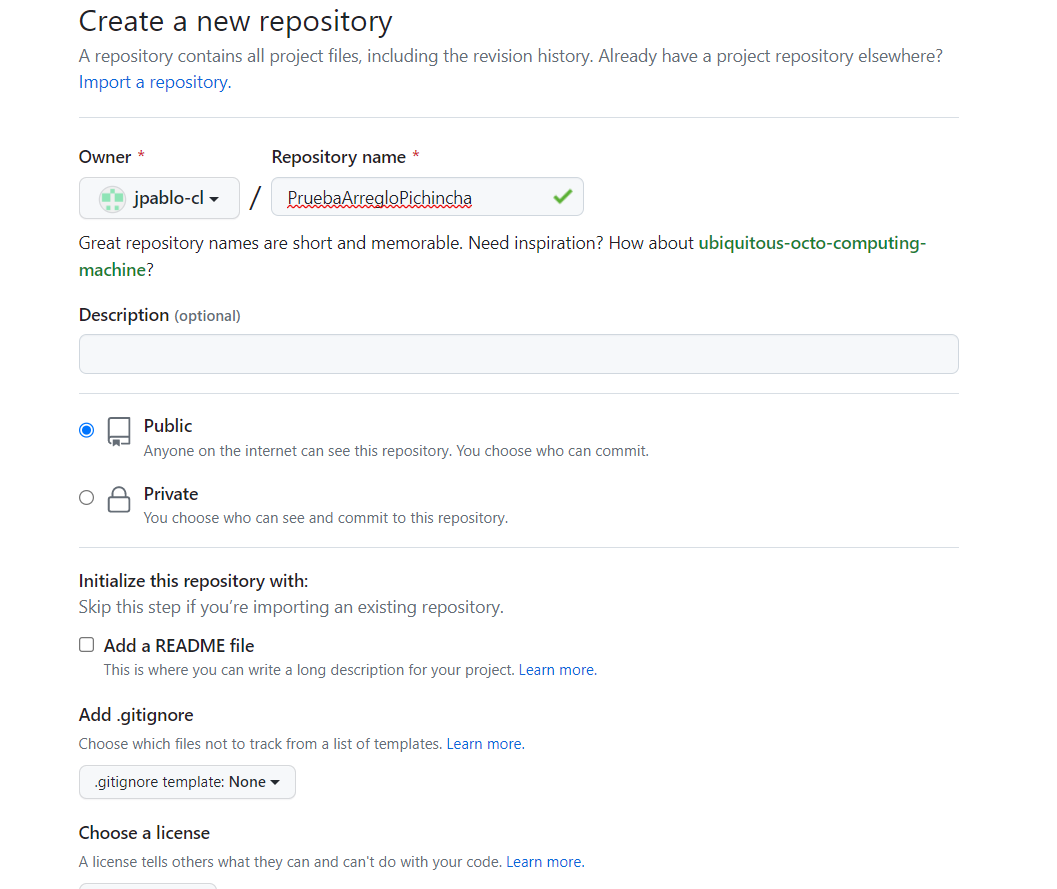
expect(arr).toHaveLength(arr.length);

});

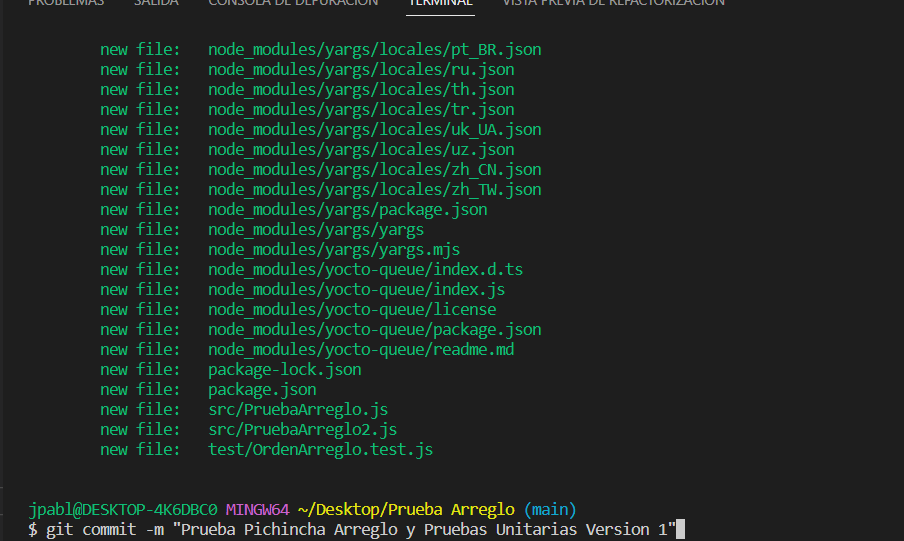
});

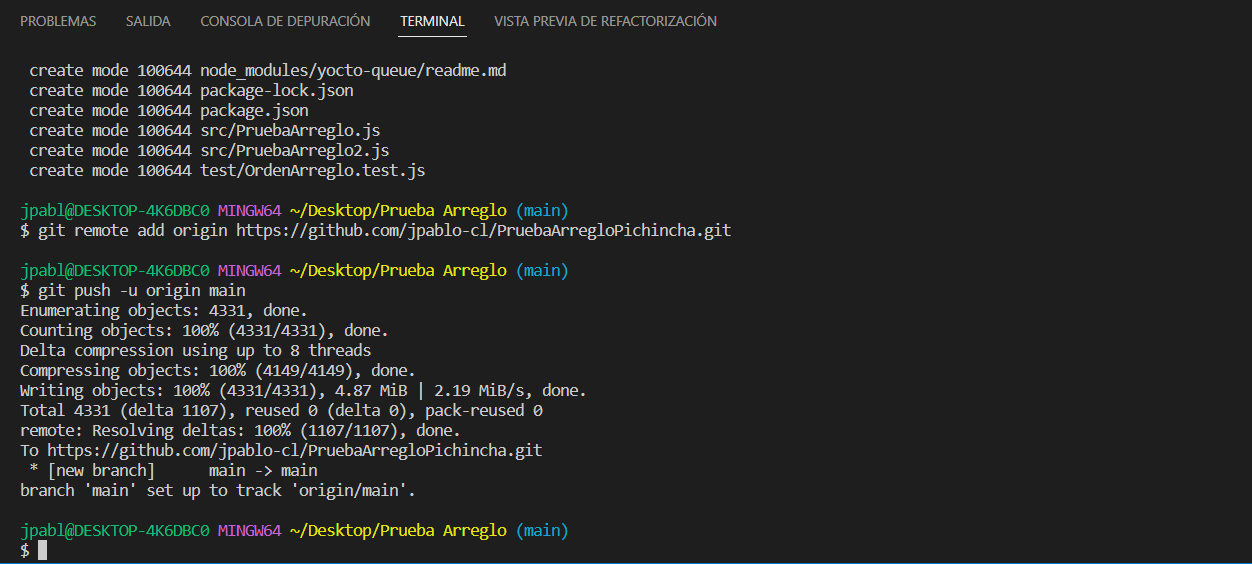
**Cargar la implementación en GitHub**

Creamos el repositorio en GitHub con el nombre **PruebaArregloPichincha**

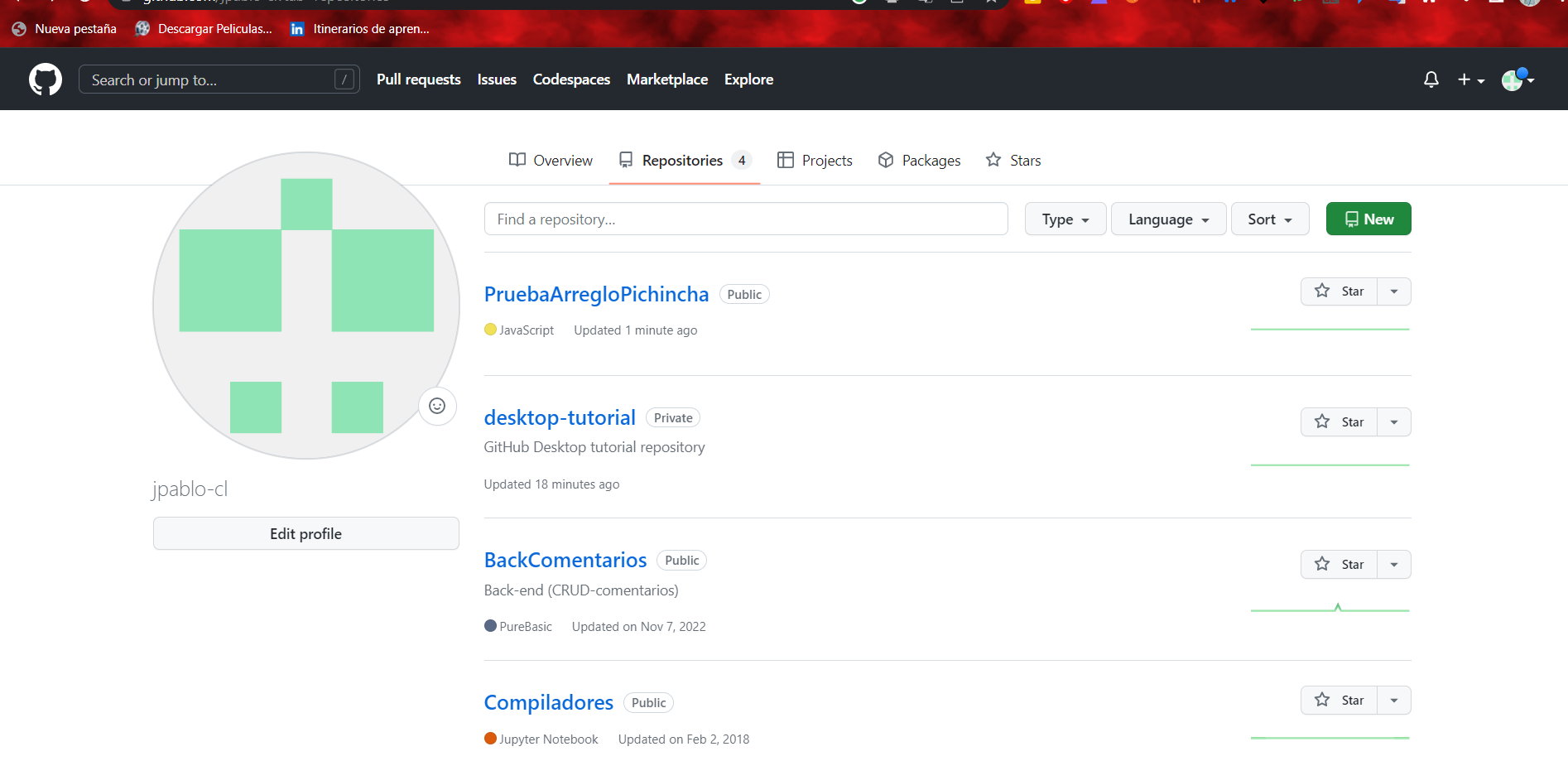


Se usa comando de GIT





Validamos que se suba en GITHUB



Validamos que se subieron todos los archivos.

