

DEPARTAMENTO:	CIENCIAS DE LA COMPUTACION	CARRERA:	SOFTWARE		
ASIGNATURA:	Pruebas de Software	NIVEL:	Sexto	FECHA:	7/2/2026
DOCENTE:	Ing. Luis Castillo, Mgtr.	PRÁCTICA N°:	2	CALIFICACIÓN:	

CI/CD usando GitHub Actions

Leonardo Vinicio Narváez Criollo

RESUMEN

El presente laboratorio describe la implementación de un flujo de trabajo de Integración Continua (CI) y una simulación de Entrega Continua (CD) utilizando GitHub Actions sobre un proyecto basado en Node.js. Durante la práctica, se configuró un entorno automatizado para la gestión de dependencias, la validación de la calidad del código mediante el análisis estático con ESLint y la verificación de la lógica de negocio a través de pruebas unitarias con el framework Jest. El proceso permitió observar cómo los disparadores (triggers) de Git facilitan la detección temprana de errores al ejecutar de forma automática los flujos de trabajo ante cada cambio en el repositorio. Finalmente, se concluyó que la automatización de estas etapas no solo reduce la probabilidad de introducir fallos en producción, sino que estandariza los criterios de calidad dentro de un equipo de desarrollo, optimizando los tiempos de entrega y la fiabilidad del software desarrollado.

Palabras Claves: Integración Continua, Automatización, Pruebas Unitarias.

1. INTRODUCCIÓN:

La integración continua (CI) es una práctica fundamental del desarrollo de software moderno. Este laboratorio tiene como propósito familiarizar con la automatización de tareas esenciales como la instalación de dependencias, la ejecución de pruebas unitarias y la verificación de calidad del código mediante ESLint, todo ello gestionado a través de GitHub Actions. A través de una aplicación sencilla en Node.js, se experimentará el poder de los flujos automatizados y se comprenderá la importancia de detectar errores temprano en el ciclo de vida del desarrollo.

2. OBJETIVO(S):

- 2.1 Configurar un flujo de integración continua (CI) en GitHub Actions que se active automáticamente con cada push o pull request a la rama principal del repositorio.
- 2.2 Implementar pruebas unitarias usando Jest, garantizando que la lógica del sistema funcione correctamente en cada actualización del código.
- 2.3 Aplicar análisis estático de código con ESLint, reforzando buenas prácticas de programación y detección temprana de errores o inconsistencias.
- 2.4 Simular un proceso de despliegue automatizado, demostrando cómo se automatizan las etapas previas al paso final de entrega continua (CD), aún sin depender de un proveedor de hosting.

3. MARCO TEÓRICO:

Para el desarrollo de esta práctica, es fundamental comprender los pilares de la automatización moderna en el ciclo de vida de desarrollo de software (SDLC):

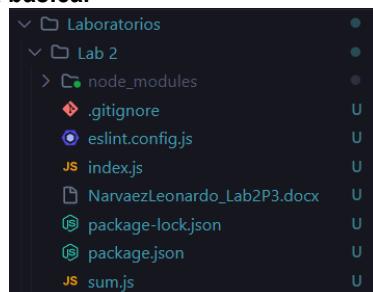
- **Integración Continua (CI) y Entrega Continua (CD):** la Integración Continua es una práctica de desarrollo en la que los desarrolladores integran su código en un repositorio compartido de forma frecuente. Cada integración es verificada por una compilación automatizada y pruebas para detectar errores lo antes posible. Por otro lado, la Entrega Continua asegura que el software pueda ser liberado a producción en cualquier momento de forma confiable.
- **GitHub Actions:** es una plataforma de automatización que permite crear flujos de trabajo (workflows) directamente en el repositorio de GitHub. Utiliza archivos de configuración en formato YAML para definir eventos (como un push o pull_request) que desencadenan una serie de trabajos (jobs) ejecutados en máquinas virtuales denominadas runners.

- **Jest y Pruebas Unitarias:** jest es un framework de pruebas de JavaScript diseñado con un enfoque en la simplicidad. En el contexto de CI, las pruebas unitarias permiten validar que pequeñas unidades de código (como funciones matemáticas) se comporten de la manera esperada ante diferentes entradas, sirviendo como la primera línea de defensa contra regresiones.
- **ESLint (Análisis Estático):** ESLint es una herramienta de análisis estático que identifica patrones problemáticos en el código JavaScript. Ayuda a mantener un estilo de codificación consistente y a prevenir errores comunes de sintaxis o lógica antes de que el código sea ejecutado.

4. DESCRIPCIÓN DEL PROCEDIMIENTO:

PARTE 1: Establecimiento de la estructura del proyecto base

Paso 1: Creación de la estructura básica.



Paso 2: Instalación de dependencias necesarias.

a. Creamos el archivo package.json para cargar las dependencias npm init -y

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> npm init -y
Wrote to D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2\package.json:

{
  "name": "lab-2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs"
}
```

b. Instalamos la dependencia de Express npm install express

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> npm install express
added 65 packages, and audited 66 packages in 6s

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

c. Instalamos las dependencias de Jest y ESLint npm install --save-dev jest eslint para que se puedan ejecutar en modo desarrollador

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> npm install --save-dev jest eslint
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lrucache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

added 357 packages, and audited 423 packages in 50s

83 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

PARTE 2: Creación de archivos base

Paso 1: Crear archivo index.js.

a. Usar el servidor express

```
Laboratorios > Lab 2 > JS indexjs > ...
1 const express = require('express');
2 const app = express();
3 const port = 3000;
```

b. Implementar un endpoint sencillo que responda con un mensaje

```
Laboratorios > Lab 2 > js index.js > ...
1 const express = require('express');
2 const app = express();
3 const port = 3000;
4 app.get('/', (req, res) => {
5   res.send('Integración continua trabajando');
6 })
```

c. Levantar el servidor en el puerto 3000

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> npm start
> lab-2@1.0.0 start
> node index.js

Servidor trabajando en el puerto 3000
```

Paso 2: Crear archivo sum.js.

- a. Crear una función que sume dos números pasados como parámetros

```
Laboratorios > Lab 2 > js sum.js > ...
1 function sum(a, b) {
2   return a + b;
3 }
4 module.exports = sum;
```

- b. Exportar la función.

```
module.exports = sum;
```

Paso 3: Crear archivo sum.test.js.

- a. Usar el archivo con la función de suma

```
Laboratorios > Lab 2 > js sum.js > ...
1 const sum = require('../sum');
```

- b. Crear una prueba para la función de suma.

```
Laboratorios > Lab 2 > js sum.js > ...
1 const sum = require('../sum');
2
3 test('suma 1 + 2 debe ser 3', () => {
4   expect(sum(1, 2)).toBe(3);
5 });
```

Paso 4: Configurar package.json.

- a. Agregar o editar los scripts para start, test y lint

- b. Agregar la característica type para que ESLint funcione como módulo.

```
Laboratorios > Lab 2 > package.json > ...
1 [
2   {
3     "name": "lab-2",
4     "version": "1.0.0",
5     "description": "",
6     "main": "index.js",
7     "scripts": {
8       "start": "node index.js",
9       "test": "jest",
10      "lint": "eslint"
11    },
12    "type": "module",
13    "keywords": [],
14    "author": "",
15    "license": "ISC",
16    "dependencies": {
17      "express": "^5.2.1"
18    }
19  }
20]
```

Paso 5: Crear el archivo ESLint.

- a. Trabajar con reglas sencillas

```
Laboratorios > Lab 2 > eslint.config.js > [e] default
1  export default [
2    {
3      files: ["**/*.js"],
4      languageOptions: {
5        ecmaVersion: "latest",
6        sourceType: "module"
7      },
8      rules: {
9        semi: ['error', 'always'],
10       quote: ['error', 'single']
11     }
12   }
13 ]
14 ]
```

Paso 6: Ignorar node_modules.

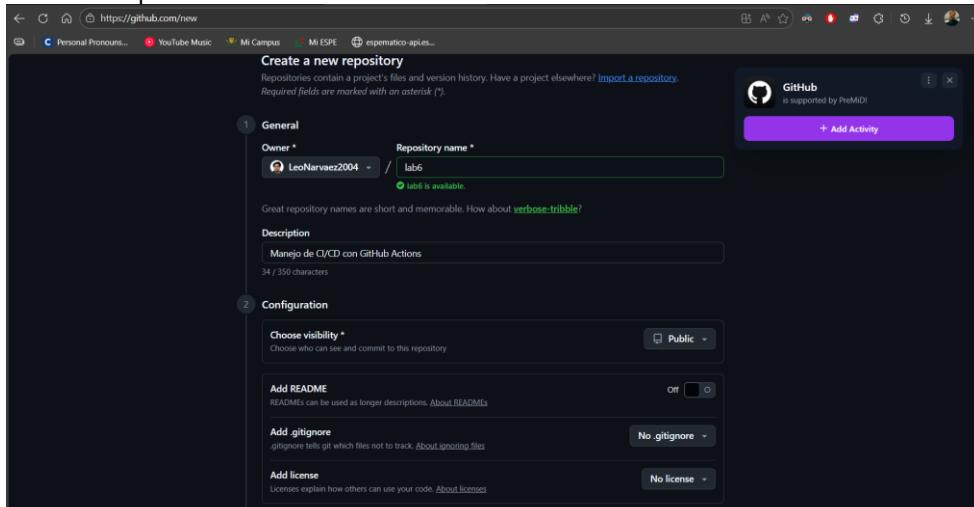
- a. En el archivo .gitignore ignorar todos los archivos que puedan causar conflictos para un proyecto NodeJS

```
Laboratorios > Lab 2 > .gitignore
1  node_modules/
2  .env
3  npmp-debug.log*
4  .Ds_Store
5  *.log
6  coverage/
```

PARTE 3: Configuración de Git

Paso 1: Crear repositorio en la cuenta de Git.

- a. Abrir la cuenta de Git en el navegador
b. Crear un nuevo repositorio vacío



Paso 2: Ejecución de comandos para clonar al repositorio.

- a. git init

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git init
Reinitialized existing Git repository in D:/Semestre VII/Pruebas de Software/Parcial III/Laboratorios/Lab 2/.git/
```

- b. git add .

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
```

- c. git commit -m "Proyecto base con CI"

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git commit -m "Proyecto base con CI"
[master (root-commit) 1b576cb] Proyecto base con CI
 8 files changed, 6061 insertions(+)
create mode 100644 .gitignore
create mode 100644 NarvaezLeonardo_Lab2P3.docx
create mode 100644 eslint.config.js
create mode 100644 index.js
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 sum.js
create mode 100644 sum.test.js
```

- d. git branch -M main

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git branch -M main
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2>
```

- e. git remote add origin https://github.com/TU_USUARIO/nombreRepositorio.git

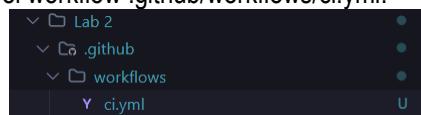
```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git remote add origin https://github.com/LeoNarvaez2004/lab6
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2>
```

f. `git push -u origin main`

```
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git push -u origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 16 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 97.42 KiB | 8.12 MiB/s, done.
Total 10 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/LeoNarvaez2004/lab6
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2>
```

Paso 3: Crear el workflow de GitHub Actions.

a. Crear un archivo nuevo para el workflow `.github/workflows/ci.yml`.



b. Configurar los triggers.

c. Configurar los trabajos a realizar

d. Configurar dentro de los trabajos los pasos a ejecutarse.

```
Laboratorios > Lab 2 > .github > workflows > ci.yml
1 name: CI Workflow
2
3 on:
4   push:
5     branches: [ main ]
6   pull_request:
7     branches: [ main ]
8
9 jobs:
10   build_and_test:
11     runs-on: ubuntu-latest
12     steps:
13       - name: Clonar repositorio
14         uses: actions/checkout@v2
15
16       - name: Configurar Node.js
17         uses: actions/setup-node@v2
18         with:
19           node-version: '20'
20
21       - name: Instalar dependencias
22         run: npm install
23
24       - name: Lint del código
25         run: npm run lint
26
27       - name: Ejecutar pruebas
28         run: npm test
29
30       - name: Simular despliegue
31         run: echo "Despliegue simulado: la aplicación pasa lint y pruebas."
```

Paso 4: Probar la CI.

a. Realizar un cambio al código.

```
Laboratorios > Lab 2 > index.js > app.get('/') callback
1 const express = require('express');
2 const app = express();
3 const port = 3000;
4 app.get('/', (req, res) => {
5   res.send(`Integración continua trabajando`)
6 })
7
8 app.listen(port, () => {
```

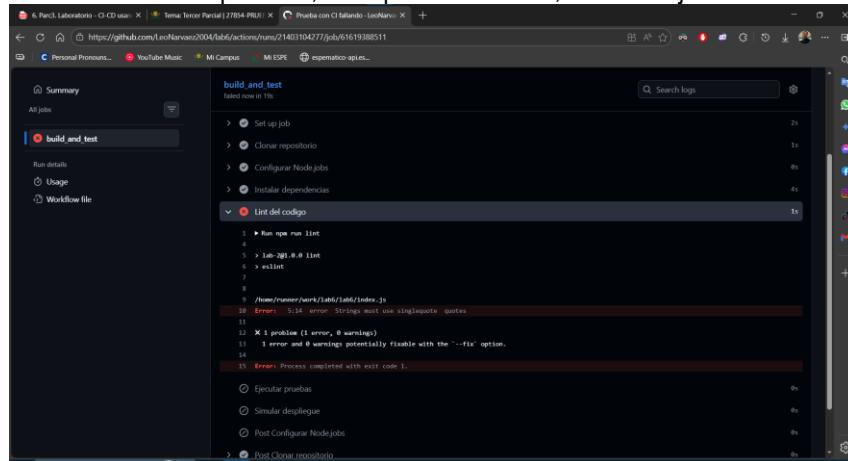
b. Ejecutar de nuevo los comandos para realizar un nuevo push.

```

● PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git add .
● PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git commit -m "Prueba con CI fallando"
[main ca8de9c] Prueba con CI fallando
 2 files changed, 1 insertion(+), 1 deletion(-)
● PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 144.03 KiB | 7.20 MiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/LeonNarvaez2004/lab6
 6ac8427..ca8de9c main -> main
○ PS D:\Semestre VII\Pruebas de Software\Parcial III\Laboratorios\Lab 2> 

```

c. Revisar en GitHub dentro del repositorio, en la pestaña Actions, como se ejecutan los Workflows



5. PREGUNTAS/ACTIVIDADES:

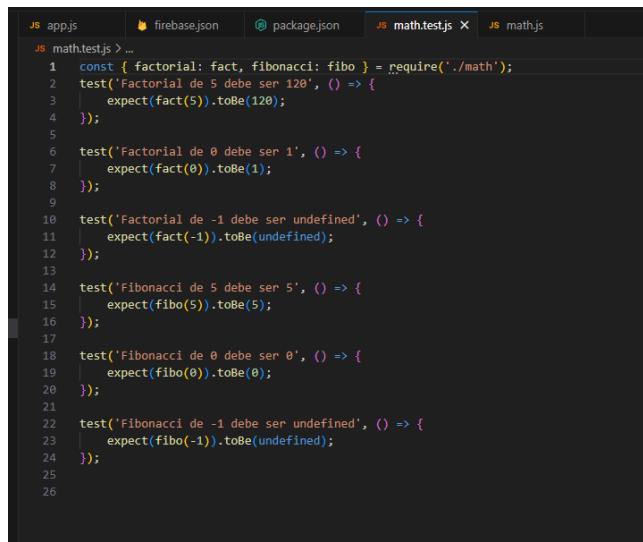
- Agregar más pruebas unitarias
 - Agregar al menos 2 funciones nuevas (por ejemplo, factorial, fibonacci) en un archivo math.js.

```

Laboratorios > Lab 2 > JS mathjs > ...
1  function factorial(n) {
2    let result = 1;
3    if (n < 0) {
4      return undefined;
5    } else if (n === 0) {
6      return 1;
7    }
8    for (let i = 1; i <= n; i++) {
9      result *= i;
10   }
11   return result;
12 }
13 module.exports = factorial;
14
15 function fibonacci(n) {
16   if (n < 0) return undefined;
17   if (n === 0) return 0;
18   if (n === 1) return 1;
19   let a = 0, b = 1;
20   for (let i = 2; i <= n; i++) {
21     let temp = a + b;
22     a = b;
23     b = temp;
24   }
25   return b;
26 }

```

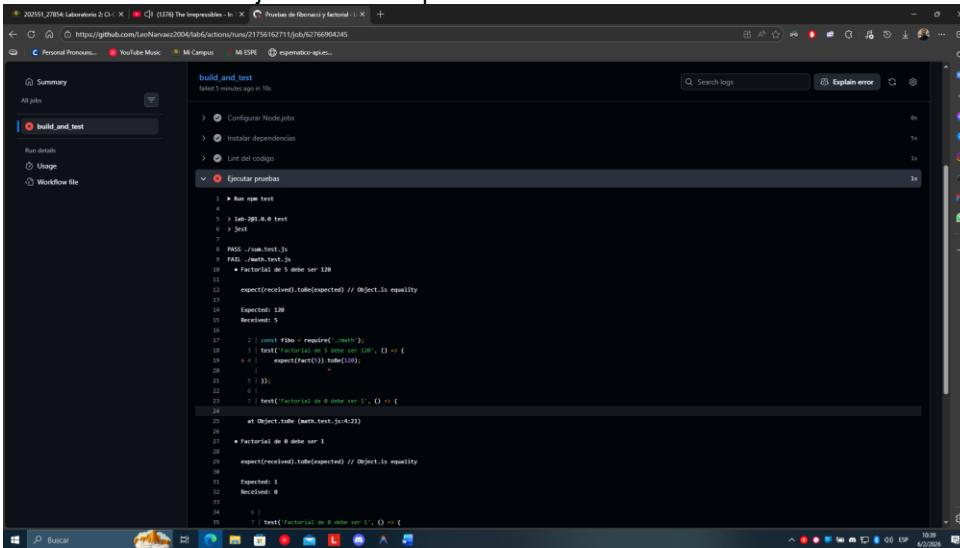
- Crear su correspondiente archivo math.test.js con pruebas Jest.



```

js app.js | firebase.json | package.json | JS math.test.js ✘ JS math.js
JS math.test.js > ...
1 const { factorial, fibonacci } = require('./math');
2 test('Factorial de 5 debe ser 120', () => {
3   expect(fact(5)).toBe(120);
4 });
5
6 test('Factorial de 0 debe ser 1', () => {
7   expect(fact(0)).toBe(1);
8 });
9
10 test('Factorial de -1 debe ser undefined', () => {
11   expect(fact(-1)).toBe(undefined);
12 });
13
14 test('Fibonacci de 5 debe ser 5', () => {
15   expect(fibo(5)).toBe(5);
16 });
17
18 test('Fibonacci de 0 debe ser 0', () => {
19   expect(fibo(0)).toBe(0);
20 });
21
22 test('Fibonacci de -1 debe ser undefined', () => {
23   expect(fibo(-1)).toBe(undefined);
24 });
25
26
  
```

- Asegurarse de que GitHub Actions ejecute todas las pruebas con éxito.
 - Primera ejecución de las pruebas con errores:



```

Workflow build_and_test
Last run 5 minutes ago in 1s
Run details
Usage
Workflow file
build_and_test
Summary
All jobs
| build_and_test
Run details
Usage
Workflow file
Ejecutar pruebas
1 ▶ run npm test
2 > lab-201-0-0 test
3 > jest
4
5 PASS ./sum.test.js
6 FAIL ./math.test.js
7 ● Factorial de 5 debe ser 120
8
9   expect(received).toBe(expected) // Object.is equality
10
11  Expected: 120
12  Received: 5
13
14  Expected: 120
15  Received: 5
16
17  const fibo = require('../src');
18  test('Factorial de 5 debe ser 120', () => {
19    expect(fact(5)).toBe(120);
20  });
21
22  const fact = require('../src');
23  test('Factorial de 0 debe ser 0', () => {
24    expect(fact(0)).toBe(0);
25  });
26
27  expect(received).toBe(expected) // Object.is equality
28
29  Expected: 0
30  Received: 0
31
32  ● Factorial de 0 debe ser 0
33
34  expect(received).toBe(expected) // Object.is equality
35
36  Expected: 0
37  Received: 0
38
39  ● test('Factorial de 0 debe ser 1', () => {
40
41  });
42
43  expect(received).toBe(expected) // Object.is equality
44
45  Expected: 1
46  Received: 0
47
48  ● Factorial de -1 debe ser undefined
49
50  expect(received).toBe(expected) // Object.is equality
51
52  Expected: undefined
53  Received: undefined
54
55  ● test('Factorial de -1 debe ser undefined', () => {
56
57  });
58
59  expect(received).toBe(expected) // Object.is equality
60
61  Expected: undefined
62  Received: undefined
63
64  ● Factorial de 5 debe ser 5
65
66  expect(received).toBe(expected) // Object.is equality
67
68  Expected: 5
69  Received: 5
70
71  ● Factorial de 0 debe ser 0
72
73  expect(received).toBe(expected) // Object.is equality
74
75  Expected: 0
76  Received: 0
77
78  ● Factorial de -1 debe ser undefined
79
80  expect(received).toBe(expected) // Object.is equality
81
82  Expected: undefined
83  Received: undefined
84
85  ● test('Factorial de -1 debe ser undefined', () => {
86
87  });
88
89  expect(received).toBe(expected) // Object.is equality
90
91  Expected: undefined
92  Received: undefined
93
94  ● Factorial de 0 debe ser 0
95
96  expect(received).toBe(expected) // Object.is equality
97
98  Expected: 0
99  Received: 0
100
101  ● Factorial de 5 debe ser 120
102
103  expect(received).toBe(expected) // Object.is equality
104
105  Expected: 120
106  Received: 5
107
108  ● Factorial de 0 debe ser 0
109
110  expect(received).toBe(expected) // Object.is equality
111
112  Expected: 0
113  Received: 0
114
115  ● Factorial de -1 debe ser undefined
116
117  expect(received).toBe(expected) // Object.is equality
118
119  Expected: undefined
120  Received: undefined
121
122  ● test('Factorial de -1 debe ser undefined', () => {
123
124  });
125
126  expect(received).toBe(expected) // Object.is equality
127
128  Expected: undefined
129  Received: undefined
130
131  ● Factorial de 0 debe ser 0
132
133  expect(received).toBe(expected) // Object.is equality
134
135  Expected: 0
136  Received: 0
137
138  ● Factorial de -1 debe ser undefined
139
140  expect(received).toBe(expected) // Object.is equality
141
142  Expected: undefined
143  Received: undefined
144
145  ● test('Factorial de -1 debe ser undefined', () => {
146
147  });
148
149  expect(received).toBe(expected) // Object.is equality
150
151  Expected: undefined
152  Received: undefined
153
154  ● Factorial de 0 debe ser 0
155
156  expect(received).toBe(expected) // Object.is equality
157
158  Expected: 0
159  Received: 0
160
161  ● Factorial de -1 debe ser undefined
162
163  expect(received).toBe(expected) // Object.is equality
164
165  Expected: undefined
166  Received: undefined
167
168  ● test('Factorial de -1 debe ser undefined', () => {
169
170  });
171
172  expect(received).toBe(expected) // Object.is equality
173
174  Expected: undefined
175  Received: undefined
176
177  ● Factorial de 0 debe ser 0
178
179  expect(received).toBe(expected) // Object.is equality
180
181  Expected: 0
182  Received: 0
183
184  ● Factorial de -1 debe ser undefined
185
186  expect(received).toBe(expected) // Object.is equality
187
188  Expected: undefined
189  Received: undefined
190
191  ● test('Factorial de -1 debe ser undefined', () => {
192
193  });
194
195  expect(received).toBe(expected) // Object.is equality
196
197  Expected: undefined
198  Received: undefined
199
200  ● Factorial de 0 debe ser 0
201
202  expect(received).toBe(expected) // Object.is equality
203
204  Expected: 0
205  Received: 0
206
207  ● Factorial de -1 debe ser undefined
208
209  expect(received).toBe(expected) // Object.is equality
210
211  Expected: undefined
212  Received: undefined
213
214  ● test('Factorial de -1 debe ser undefined', () => {
215
216  });
217
218  expect(received).toBe(expected) // Object.is equality
219
220  Expected: undefined
221  Received: undefined
222
223  ● Factorial de 0 debe ser 0
224
225  expect(received).toBe(expected) // Object.is equality
226
227  Expected: 0
228  Received: 0
229
230  ● Factorial de -1 debe ser undefined
231
232  expect(received).toBe(expected) // Object.is equality
233
234  Expected: undefined
235  Received: undefined
236
237  ● test('Factorial de -1 debe ser undefined', () => {
238
239  });
240
241  expect(received).toBe(expected) // Object.is equality
242
243  Expected: undefined
244  Received: undefined
245
246  ● Factorial de 0 debe ser 0
247
248  expect(received).toBe(expected) // Object.is equality
249
250  Expected: 0
251  Received: 0
252
253  ● Factorial de -1 debe ser undefined
254
255  expect(received).toBe(expected) // Object.is equality
256
257  Expected: undefined
258  Received: undefined
259
260  ● test('Factorial de -1 debe ser undefined', () => {
261
262  });
263
264  expect(received).toBe(expected) // Object.is equality
265
266  Expected: undefined
267  Received: undefined
268
269  ● Factorial de 0 debe ser 0
270
271  expect(received).toBe(expected) // Object.is equality
272
273  Expected: 0
274  Received: 0
275
276  ● Factorial de -1 debe ser undefined
277
278  expect(received).toBe(expected) // Object.is equality
279
280  Expected: undefined
281  Received: undefined
282
283  ● test('Factorial de -1 debe ser undefined', () => {
284
285  });
286
287  expect(received).toBe(expected) // Object.is equality
288
289  Expected: undefined
290  Received: undefined
291
292  ● Factorial de 0 debe ser 0
293
294  expect(received).toBe(expected) // Object.is equality
295
296  Expected: 0
297  Received: 0
298
299  ● Factorial de -1 debe ser undefined
300
301  expect(received).toBe(expected) // Object.is equality
302
303  Expected: undefined
304  Received: undefined
305
306  ● test('Factorial de -1 debe ser undefined', () => {
307
308  });
309
310  expect(received).toBe(expected) // Object.is equality
311
312  Expected: undefined
313  Received: undefined
314
315  ● Factorial de 0 debe ser 0
316
317  expect(received).toBe(expected) // Object.is equality
318
319  Expected: 0
320  Received: 0
321
322  ● Factorial de -1 debe ser undefined
323
324  expect(received).toBe(expected) // Object.is equality
325
326  Expected: undefined
327  Received: undefined
328
329  ● test('Factorial de -1 debe ser undefined', () => {
330
331  });
332
333  expect(received).toBe(expected) // Object.is equality
334
335  Expected: undefined
336  Received: undefined
337
338  ● Factorial de 0 debe ser 0
339
340  expect(received).toBe(expected) // Object.is equality
341
342  Expected: 0
343  Received: 0
344
345  ● Factorial de -1 debe ser undefined
346
347  expect(received).toBe(expected) // Object.is equality
348
349  Expected: undefined
350  Received: undefined
351
352  ● test('Factorial de -1 debe ser undefined', () => {
353
354  });
355
356  expect(received).toBe(expected) // Object.is equality
357
358  Expected: undefined
359  Received: undefined
360
361  ● Factorial de 0 debe ser 0
362
363  expect(received).toBe(expected) // Object.is equality
364
365  Expected: 0
366  Received: 0
367
368  ● Factorial de -1 debe ser undefined
369
370  expect(received).toBe(expected) // Object.is equality
371
372  Expected: undefined
373  Received: undefined
374
375  ● test('Factorial de -1 debe ser undefined', () => {
376
377  });
378
379  expect(received).toBe(expected) // Object.is equality
380
381  Expected: undefined
382  Received: undefined
383
384  ● Factorial de 0 debe ser 0
385
386  expect(received).toBe(expected) // Object.is equality
387
388  Expected: 0
389  Received: 0
390
391  ● Factorial de -1 debe ser undefined
392
393  expect(received).toBe(expected) // Object.is equality
394
395  Expected: undefined
396  Received: undefined
397
398  ● test('Factorial de -1 debe ser undefined', () => {
399
400  });
401
402  expect(received).toBe(expected) // Object.is equality
403
404  Expected: undefined
405  Received: undefined
406
407  ● Factorial de 0 debe ser 0
408
409  expect(received).toBe(expected) // Object.is equality
410
411  Expected: 0
412  Received: 0
413
414  ● Factorial de -1 debe ser undefined
415
416  expect(received).toBe(expected) // Object.is equality
417
418  Expected: undefined
419  Received: undefined
420
421  ● test('Factorial de -1 debe ser undefined', () => {
422
423  });
424
425  expect(received).toBe(expected) // Object.is equality
426
427  Expected: undefined
428  Received: undefined
429
430  ● Factorial de 0 debe ser 0
431
432  expect(received).toBe(expected) // Object.is equality
433
434  Expected: 0
435  Received: 0
436
437  ● Factorial de -1 debe ser undefined
438
439  expect(received).toBe(expected) // Object.is equality
440
441  Expected: undefined
442  Received: undefined
443
444  ● test('Factorial de -1 debe ser undefined', () => {
445
446  });
447
448  expect(received).toBe(expected) // Object.is equality
449
450  Expected: undefined
451  Received: undefined
452
453  ● Factorial de 0 debe ser 0
454
455  expect(received).toBe(expected) // Object.is equality
456
457  Expected: 0
458  Received: 0
459
460  ● Factorial de -1 debe ser undefined
461
462  expect(received).toBe(expected) // Object.is equality
463
464  Expected: undefined
465  Received: undefined
466
467  ● test('Factorial de -1 debe ser undefined', () => {
468
469  });
470
471  expect(received).toBe(expected) // Object.is equality
472
473  Expected: undefined
474  Received: undefined
475
476  ● Factorial de 0 debe ser 0
477
478  expect(received).toBe(expected) // Object.is equality
479
480  Expected: 0
481  Received: 0
482
483  ● Factorial de -1 debe ser undefined
484
485  expect(received).toBe(expected) // Object.is equality
486
487  Expected: undefined
488  Received: undefined
489
490  ● test('Factorial de -1 debe ser undefined', () => {
491
492  });
493
494  expect(received).toBe(expected) // Object.is equality
495
496  Expected: undefined
497  Received: undefined
498
499  ● Factorial de 0 debe ser 0
500
501  expect(received).toBe(expected) // Object.is equality
502
503  Expected: 0
504  Received: 0
505
506  ● Factorial de -1 debe ser undefined
507
508  expect(received).toBe(expected) // Object.is equality
509
510  Expected: undefined
511  Received: undefined
512
513  ● test('Factorial de -1 debe ser undefined', () => {
514
515  });
516
517  expect(received).toBe(expected) // Object.is equality
518
519  Expected: undefined
520  Received: undefined
521
522  ● Factorial de 0 debe ser 0
523
524  expect(received).toBe(expected) // Object.is equality
525
526  Expected: 0
527  Received: 0
528
529  ● Factorial de -1 debe ser undefined
530
531  expect(received).toBe(expected) // Object.is equality
532
533  Expected: undefined
534  Received: undefined
535
536  ● test('Factorial de -1 debe ser undefined', () => {
537
538  });
539
540  expect(received).toBe(expected) // Object.is equality
541
542  Expected: undefined
543  Received: undefined
544
545  ● Factorial de 0 debe ser 0
546
547  expect(received).toBe(expected) // Object.is equality
548
549  Expected: 0
550  Received: 0
551
552  ● Factorial de -1 debe ser undefined
553
554  expect(received).toBe(expected) // Object.is equality
555
556  Expected: undefined
557  Received: undefined
558
559  ● test('Factorial de -1 debe ser undefined', () => {
560
561  });
562
563  expect(received).toBe(expected) // Object.is equality
564
565  Expected: undefined
566  Received: undefined
567
568  ● Factorial de 0 debe ser 0
569
570  expect(received).toBe(expected) // Object.is equality
571
572  Expected: 0
573  Received: 0
574
575  ● Factorial de -1 debe ser undefined
576
577  expect(received).toBe(expected) // Object.is equality
578
579  Expected: undefined
580  Received: undefined
581
582  ● test('Factorial de -1 debe ser undefined', () => {
583
584  });
585
586  expect(received).toBe(expected) // Object.is equality
587
588  Expected: undefined
589  Received: undefined
590
591  ● Factorial de 0 debe ser 0
592
593  expect(received).toBe(expected) // Object.is equality
594
595  Expected: 0
596  Received: 0
597
598  ● Factorial de -1 debe ser undefined
599
600  expect(received).toBe(expected) // Object.is equality
601
602  Expected: undefined
603  Received: undefined
604
605  ● test('Factorial de -1 debe ser undefined', () => {
606
607  });
608
609  expect(received).toBe(expected) // Object.is equality
610
611  Expected: undefined
612  Received: undefined
613
614  ● Factorial de 0 debe ser 0
615
616  expect(received).toBe(expected) // Object.is equality
617
618  Expected: 0
619  Received: 0
620
621  ● Factorial de -1 debe ser undefined
622
623  expect(received).toBe(expected) // Object.is equality
624
625  Expected: undefined
626  Received: undefined
627
628  ● test('Factorial de -1 debe ser undefined', () => {
629
630  });
631
632  expect(received).toBe(expected) // Object.is equality
633
634  Expected: undefined
635  Received: undefined
636
637  ● Factorial de 0 debe ser 0
638
639  expect(received).toBe(expected) // Object.is equality
640
641  Expected: 0
642  Received: 0
643
644  ● Factorial de -1 debe ser undefined
645
646  expect(received).toBe(expected) // Object.is equality
647
648  Expected: undefined
649  Received: undefined
650
651  ● test('Factorial de -1 debe ser undefined', () => {
652
653  });
654
655  expect(received).toBe(expected) // Object.is equality
656
657  Expected: undefined
658  Received: undefined
659
660  ● Factorial de 0 debe ser 0
661
662  expect(received).toBe(expected) // Object.is equality
663
664  Expected: 0
665  Received: 0
666
667  ● Factorial de -1 debe ser undefined
668
669  expect(received).toBe(expected) // Object.is equality
670
671  Expected: undefined
672  Received: undefined
673
674  ● test('Factorial de -1 debe ser undefined', () => {
675
676  });
677
678  expect(received).toBe(expected) // Object.is equality
679
680  Expected: undefined
681  Received: undefined
682
683  ● Factorial de 0 debe ser 0
684
685  expect(received).toBe(expected) // Object.is equality
686
687  Expected: 0
688  Received: 0
689
690  ● Factorial de -1 debe ser undefined
691
692  expect(received).toBe(expected) // Object.is equality
693
694  Expected: undefined
695  Received: undefined
696
697  ● test('Factorial de -1 debe ser undefined', () => {
698
699  });
700
701  expect(received).toBe(expected) // Object.is equality
702
703  Expected: undefined
704  Received: undefined
705
706  ● Factorial de 0 debe ser 0
707
708  expect(received).toBe(expected) // Object.is equality
709
710  Expected: 0
711  Received: 0
712
713  ● Factorial de -1 debe ser undefined
714
715  expect(received).toBe(expected) // Object.is equality
716
717  Expected: undefined
718  Received: undefined
719
720  ● test('Factorial de -1 debe ser undefined', () => {
721
722  });
723
724  expect(received).toBe(expected) // Object.is equality
725
726  Expected: undefined
727  Received: undefined
728
729  ● Factorial de 0 debe ser 0
730
731  expect(received).toBe(expected) // Object.is equality
732
733  Expected: 0
734  Received: 0
735
736  ● Factorial de -1 debe ser undefined
737
738  expect(received).toBe(expected) // Object.is equality
739
740  Expected: undefined
741  Received: undefined
742
743  ● test('Factorial de -1 debe ser undefined', () => {
744
745  });
746
747  expect(received).toBe(expected) // Object.is equality
748
749  Expected: undefined
750  Received: undefined
751
752  ● Factorial de 0 debe ser 0
753
754  expect(received).toBe(expected) // Object.is equality
755
756  Expected: 0
757  Received: 0
758
759  ● Factorial de -1 debe ser undefined
760
761  expect(received).toBe(expected) // Object.is equality
762
763  Expected: undefined
764  Received: undefined
765
766  ● test('Factorial de -1 debe ser undefined', () => {
767
768  });
769
770  expect(received).toBe(expected) // Object.is equality
771
772  Expected: undefined
773  Received: undefined
774
775  ● Factorial de 0 debe ser 0
776
777  expect(received).toBe(expected) // Object.is equality
778
779  Expected: 0
780  Received: 0
781
782  ● Factorial de -1 debe ser undefined
783
784  expect(received).toBe(expected) // Object.is equality
785
786  Expected: undefined
787  Received: undefined
788
789  ● test('Factorial de -1 debe ser undefined', () => {
790
791  });
792
793  expect(received).toBe(expected) // Object.is equality
794
795  Expected: undefined
796  Received: undefined
797
798  ● Factorial de 0 debe ser 0
799
800  expect(received).toBe(expected) // Object.is equality
801
802  Expected: 0
803  Received: 0
804
805  ● Factorial de -1 debe ser undefined
806
807  expect(received).toBe(expected) // Object.is equality
808
809  Expected: undefined
810  Received: undefined
811
812  ● test('Factorial de -1 debe ser undefined', () => {
813
814  });
815
816  expect(received).toBe(expected) // Object.is equality
817
818  Expected: undefined
819  Received: undefined
820
821  ● Factorial de 0 debe ser 0
822
823  expect(received).toBe(expected) // Object.is equality
824
825  Expected: 0
826  Received: 0
827
828  ● Factorial de -1 debe ser undefined
829
830  expect(received).toBe(expected) // Object.is equality
831
832  Expected: undefined
833  Received: undefined
834
835  ● test('Factorial de -1 debe ser undefined', () => {
836
837  });
838
839  expect(received).toBe(expected) // Object.is equality
840
841  Expected: undefined
842  Received: undefined
843
844  ● Factorial de 0 debe ser 0
845
846  expect(received).toBe(expected) // Object.is equality
847
848  Expected: 0
849  Received: 0
850
851  ● Factorial de -1 debe ser undefined
852
853  expect(received).toBe(expected) // Object.is equality
854
855  Expected: undefined
856  Received: undefined
857
858  ● test('Factorial de -1 debe ser undefined', () => {
859
860  });
861
862  expect(received).toBe(expected) // Object.is equality
863
864  Expected: undefined
865  Received: undefined
866
867  ● Factorial de 0 debe ser 0
868
869  expect(received).toBe(expected) // Object.is equality
870
871  Expected: 0
872  Received: 0
873
874  ● Factorial de -1 debe ser undefined
875
876  expect(received).toBe(expected) // Object.is equality
877
878  Expected: undefined
879  Received: undefined
880
881  ● test('Factorial de -1 debe ser undefined', () => {
882
883  });
884
885  expect(received).toBe(expected) // Object.is equality
886
887  Expected: undefined
888  Received: undefined
889
890  ● Factorial de 0 debe ser 0
891
892  expect(received).toBe(expected) // Object.is equality
893
894  Expected: 0
895  Received: 0
896
897  ● Factorial de -1 debe ser undefined
898
899  expect(received).toBe(expected) // Object.is equality
900
901  Expected: undefined
902  Received: undefined
903
904  ● test('Factorial de -1 debe ser undefined', () => {
905
906  });
907
908  expect(received).toBe(expected) // Object.is equality
909
910  Expected: undefined
911  Received: undefined
912
913  ● Factorial de 0 debe ser 0
914
915  expect(received).toBe(expected) // Object.is equality
916
917  Expected: 0
918  Received: 0
919
920  ● Factorial de -1 debe ser undefined
921
922  expect(received).toBe(expected) // Object.is equality
923
924  Expected: undefined
925  Received: undefined
926
927  ● test('Factorial de -1 debe ser undefined', () => {
928
929  });
930
931  expect(received).toBe(expected) // Object.is equality
932
933  Expected: undefined
934  Received: undefined
935
936  ● Factorial de 0 debe ser 0
937
938  expect(received).toBe(expected) // Object.is equality
939
940  Expected: 0
941  Received: 0
942
943  ● Factorial de -1 debe ser undefined
944
945  expect(received).toBe(expected) // Object.is equality
946
947  Expected: undefined
948  Received: undefined
949
950  ● test('Factorial de -1 debe ser undefined', () => {
951
952  });
953
954  expect(received).toBe(expected) // Object.is equality
955
956  Expected: undefined
957  Received: undefined
958
959  ● Factorial de 0 debe ser 0
960
961  expect(received).toBe(expected) // Object.is equality
962
963  Expected: 0
964  Received: 0
965
966  ● Factorial de -1 debe ser undefined
967
968  expect(received).toBe(expected) // Object.is equality
969
970  Expected: undefined
971  Received: undefined
972
973  ● test('Factorial de -1 debe ser undefined', () => {
974
975  });
976
977  expect(received).toBe(expected) // Object.is equality
978
979  Expected: undefined
980  Received: undefined
981
982  ● Factorial de 0 debe ser 0
983
984  expect(received).toBe(expected) // Object.is equality
985
986  Expected: 0
987  Received: 0
988
989  ● Factorial de -1 debe ser undefined
990
991  expect(received).toBe(expected) // Object.is equality
992
993  Expected: undefined
994  Received: undefined
995
996  ● test('Factorial de -1 debe ser undefined', () => {
997
998  });
999
1000  expect(received).toBe(expected) // Object.is equality
1001
1002  Expected: undefined
1003  Received: undefined
1004
1005  ● Factorial de 0 debe ser 0
1006
1007  expect(received).toBe(expected) // Object.is equality
1008
1009  Expected: 0
1010  Received: 0
1011
1012  ● Factorial de -1 debe ser undefined
1013
1014  expect(received).toBe(expected) // Object.is equality
1015
1016  Expected: undefined
1017  Received: undefined
1018
1019  ● test('Factorial de -1 debe ser undefined', () => {
1020
1021  });
1022
1023  expect(received).toBe(expected) // Object.is equality
1024
1025  Expected: undefined
1026  Received: undefined
1027
1028  ● Factorial de 0 debe ser 0
1029
1030  expect(received).toBe(expected) // Object.is equality
1031
1032  Expected: 0
1033  Received: 0
1034
1035  ● Factorial de -1 debe ser undefined
1036
1037  expect(received).toBe(expected) // Object.is equality
1038
1039  Expected: undefined
1040  Received: undefined
1041
1042  ● test('Factorial de -1 debe ser undefined', () => {
1043
1044  });
1045
1046  expect(received).toBe(expected) // Object.is equality
1047
1048  Expected: undefined
1049  Received: undefined
1050
1051  ● Factorial de 0 debe ser 0
1052
1053  expect(received).toBe(expected) // Object.is equality
1054
1055  Expected: 0
1056  Received: 0
1057
1058  ● Factorial de -1 debe ser undefined
1059
1060  expect(received).toBe(expected) // Object.is equality
1061
1062  Expected: undefined
1063  Received: undefined
1064
1065  ● test('Factorial de -1 debe ser undefined', () => {
1066
1067  });
1068
1069  expect(received).toBe(expected) // Object.is equality
1070
1071  Expected: undefined
1072  Received: undefined
1073
1074  ● Factorial de 0 debe ser 0
1075
1076  expect(received).toBe(expected) // Object.is equality
1077
1078  Expected: 0
1079  Received: 0
1080
1081  ● Factorial de -1 debe ser undefined
1082
1083  expect(received).toBe(expected) // Object.is equality
1084
1085  Expected: undefined
1086  Received: undefined
1087
1088  ● test('Factorial de -1 debe ser undefined', () => {
1089
1090  });
1091
1092  expect(received).toBe(expected) // Object.is equality
1093
1094  Expected: undefined
1095  Received: undefined
1096
1097  ● Factorial de 0 debe ser 0
1098
1099  expect(received).toBe(expected) // Object.is equality
1100
1101  Expected: 0
1102  Received: 0
1103
1104  ● Factorial de -1 debe ser undefined
1105
1106  expect(received).toBe(expected) // Object.is equality
1107
1108  Expected: undefined
1109  Received: undefined
1110
1111  ● test('Factorial de -1 debe ser undefined', () => {
1112
1113  });
1114
1115  expect(received).toBe(expected) // Object.is equality
1116
1117  Expected: undefined
1118  Received: undefined
1119
1120  ● Factorial de 0 debe ser 0
1121
1122  expect(received).toBe(expected) // Object.is equality
1123
1124  Expected: 0
1125  Received: 0
1126
1127  ● Factorial de -1 debe ser undefined
1128
1129  expect(received).toBe(expected) // Object.is equality
1130
1131  Expected: undefined
1132  Received: undefined
1133
1134  ● test('Factorial de -1 debe ser undefined', () => {
1135
1136  });
1137
1138  expect(received).toBe(expected) // Object.is equality
1139
1140  Expected: undefined
1141  Received: undefined
1142
1143  ● Factorial de 0 debe ser 0
1144
1145  expect(received).toBe(expected) // Object.is equality
1146
1147  Expected: 0
1148  Received: 0
1149
1150  ● Factorial de -1 debe ser undefined
1151
1152  expect(received).toBe(expected) // Object.is equality
1153
1154  Expected: undefined
1155  Received: undefined
1156
1157  ● test('Factorial de -1 debe ser undefined', () => {
1158
1159  });
1160
1161  expect(received).toBe(expected) // Object.is equality
1162
1163  Expected: undefined
1164  Received: undefined
1165
1166  ● Factorial de 0 debe ser 0
1167
1168  expect(received).toBe(expected) // Object.is equality
1169
1170  Expected: 0
1171  Received: 0
1172
1173  ● Factorial de -1 debe ser undefined
1174
1175  expect(received).toBe(expected) // Object.is equality
1176
1177  Expected: undefined
1178  Received: undefined
1179
1180  ● test('Factorial de -1 debe ser undefined', () => {
1181
1182  });
1183
1184  expect(received).toBe(expected) // Object.is equality
1185
1186  Expected: undefined
1187  Received: undefined
1188
1189  ● Factorial de 0 debe ser 0
1190
1191  expect(received).toBe(expected) // Object.is equality
1192
1193  Expected: 0
1194  Received: 0
1195
1196  ● Factorial de -1 debe ser undefined
1197
1198  expect(received).toBe(expected) // Object.is equality
1199
1200  Expected: undefined
1201  Received: undefined
1202
1203  ● test('Factorial de -1 debe ser undefined', () => {
1204
1205  });
1206
1207  expect(received).toBe(expected) // Object.is equality
1208
1209  Expected: undefined
1210  Received: undefined
1211
1212  ● Factorial de 0 debe ser 0
1213
1214  expect(received).toBe(expected) // Object.is equality
1215
1216  Expected: 0
1217  Received: 0
1218
1219  ● Factorial de -1 debe ser undefined
1220
1221  expect(received).toBe(expected) // Object.is equality
1222
1223  Expected: undefined
1224  Received: undefined
1225
1226  ● test('Factorial de -1 debe
```

6. CONCLUSIONES:

- Se logró configurar exitosamente un workflow de GitHub Actions, demostrando que la automatización mediante archivos YAML permite centralizar la lógica de integración, eliminando la necesidad de procesos manuales para la validación del código.
- La implementación de pruebas unitarias con Jest permitió verificar la integridad de la lógica aritmética del sistema, asegurando que nuevas funcionalidades (como factorial o fibonacci) no rompan el comportamiento existente, cumpliendo así con el principio de regresión.
- El uso de ESLint garantizó que el código cargado al repositorio cumpla con estándares de calidad definidos, lo que facilita la mantenibilidad del proyecto y reduce la deuda técnica desde las etapas iniciales del desarrollo.
- A través de la simulación del despliegue automatizado, se comprendió la importancia de las etapas de "pre-flight" (instalación, linting y testing), las cuales actúan como filtros críticos que garantizan que solo el código estable pueda avanzar hacia una fase de producción o entrega final

7. RECOMENDACIONES:

- Modularización de Pruebas: Se recomienda separar las pruebas unitarias por módulos funcionales (por ejemplo, math.test.js separado de api.test.js) para facilitar la depuración cuando un flujo de trabajo de CI falle.
- Gestión de Secretos: Aunque en esta práctica no se conectó a un proveedor de hosting real, se recomienda para proyectos futuros utilizar el apartado de Secrets de GitHub para manejar credenciales de despliegue de forma segura.
- Optimización del Cache: Para reducir el tiempo de ejecución en GitHub Actions, es aconsejable configurar el caché de las dependencias de node_modules dentro del archivo YAML, evitando la descarga repetitiva de paquetes en cada ejecución.

8. BIBLIOGRAFÍA:

GITHUB. (2024). GitHub Actions Documentation. Recuperado de: <https://docs.github.com/en/actions>.

JEST. (2024). Jest: Delightful JavaScript Testing. Recuperado de: <https://jestjs.io/>.

FOWLER, M. (2006). Continuous Integration. MartinFowler.com. Recuperado de: <https://martinfowler.com/articles/continuousIntegration.html>.

SOMMERVILLE, I. (2011). Ingeniería de Software. 9na Edición. Pearson Educación.