# Pruebas

# Formato de archivos test

Usualmente, se utiliza la nomenclatura
[nombre de lo que se va a probar].test.js

# Formato de archivos test

Usualmente, se utiliza la nomenclatura
[nombre de lo que se va a probar].test.js

user.controller.test.js

# Tipos de pruebas

1. Unitarias
2. Integracion
3. End to end (E2E)

# Pruebas unitarias

Prueban las unidades mas pequeñas de un software

# Pruebas unitarias

Prueban las unidades mas pequeñas de un software

Por ej. funciones

# Pruebas de integracion

Prueban varias unidades pequeñas trabajando en conjunto

# Pruebas de integracion

Prueban varias unidades pequeñas trabajando en conjunto

Por ej., el proceso completo (ruta → controlador → base de datos)

# Pruebas end to end

Prueban la aplicacion completa en funcionamiento en un ambiente similar al de produccion

# Pruebas end to end

Prueban la aplicacion completa en funcionamiento en un ambiente similar al de produccion

Por ej., realizan requests desde el frontend al backend

# Recontextualizacion

Las pruebas e2e (end to end) requieren del frontend y el backend.

Por esto, es normal tener dos grupos de e2e distintos.

# Recontextualizacion de pruebas

1. E2E de aplicacion
2. E2E de backend
3. Integracion
4. Unitarias

# Pruebas unitarias

Prueban las unidades mas pequeñas de un software

Por ej. Funciones especificas

# Pruebas de integracion

Prueban varias unidades pequeñas trabajando en conjunto

Por ej., dos controladores que interactuan (users → products)

# Pruebas E2E de aplicacion

Prueban la aplicacion completa en funcionamiento en un ambiente similar al de produccion

Por ej., realizan requests desde el frontend al backend

# Pruebas E2E de backend

Prueban la aplicacion completa en funcionamiento

Por ej., el proceso completo (ruta → controlador → base de datos)

# Jest

```
npm install jest
```

# Realizar un test

```
test("descripcion del test (breve)", [test function]);
```

# "Expectativas"

```
expect(value).toBe(value);
```

# Caso 1

# Agrupar tests

```
describe("descripcion del grupo (breve)", [group function]);
```

# Caso 2

# Agrupar grupos

```
describe("descripcion del grupo (breve)", [group function]);
```

# Caso 3

# Manejo de errores y excepciones

```
expect(() => [function]).toThrow([error, o nada]);
```

# Caso 4

# Otros Matchers

.not
.toEqual
.toBeNull
.toBeUndefined
.toBeDefined
.toBeTruthy
.toBeFalsy
.toBeGreaterThan
.toBeGreaterThanOrEqual
.toBeLessThan
.toBeLessThanOrEqual

# Caso 5

# Montaje y Desmontaje

A veces, las pruebas requieren de recursos o cosas similares.

# Montaje y Desmontaje

| | |
|---|---|
| BeforeEach | Ejecuta antes de cada prueba |
| AfterEach | Ejecuta despues de cada prueba |
| BeforeAll | Ejecuta antes de todas las pruebas |
| AfterAll | Ejecuta despues de todas las pruebas |

# Montaje y Desmontaje

| | |
|---|---|
| BeforeEach | Ejecuta antes de cada prueba |
| AfterEach | Ejecuta despues de cada prueba |
| BeforeAll | Ejecuta antes de todas las pruebas |
| AfterAll | Ejecuta despues de todas las pruebas |

Ejemplos?

# Montaje, Desmontaje y Agrupacion

Los montajes y desmontajes toman en cuenta la agrupacion de pruebas.

# Caso 6

# Funciones Mock

```
jest.fn([function]);
jest.fn().mockReturnValue([value])
```

# Funciones Mock y Llamados

```
expect([mock]).toBeCalled()
expect([mock]).toBeCalledTimes([numero])
expect([mock].mock.calls[0][0]).toBe([param 1]);
expect([mock].mock.calls[0][1]).toBe([param 2]);
expect([mock].mock.results[0].value).toBe([resultado]);
```

# Caso 7

# Funciones Espia

```
jest
  .spyOn([objeto o clase], [nombre de function])
```

# Funciones Espia y Mock

```
jest
  .spyOn([objeto o clase], [nombre de function])
  .mockImplementation([function]);
```

# Caso 8