

```
69 | describe('update', () => {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

● PS C:\Users\ivanr\OneDrive\Escritorio\nest\digitalNaoProject> npx jest

PASS src/inventory/inventory.service.spec.ts

PASS src/users/user.service.spec.ts

PASS src/sale/sales.service.spec.ts

Test Suites: 3 passed, 3 total

Tests: 17 passed, 17 total

Snapshots: 0 total

Time: 5.181 s

Ran all test suites.

○ PS C:\Users\ivanr\OneDrive\Escritorio\nest\digitalNaoProject>

```
TS auth.module.ts TS sales.service.spec.ts M X TS user.service.spec.ts M TS inventory.service.spec.ts M TS inventory.service.ts TS create-product.dto.ts

src > sale > TS sales.service.spec.ts > describe('SaleService') callback
9   describe('SaleService', () => {
23     const mockRepository = {
29       delete: jest.fn().mockResolvedValue({ affected: 1 }),
30     };
31   };
32   beforeEach(async () => {
33     const module: TestingModule = await Test.createTestingModule({
34       providers: [
35         SaleService,
36         {
37           provide: getRepositoryToken(Sale),
38           useClass: mockRepository,
39         },
40       ],
41     }).compile();
42
43     service = module.get<SaleService>(SaleService);
44     repository = module.get<Repository<Sale>>(getRepositoryToken(Sale));
45   });
46
47   it('should be defined', () => {
48     expect(service).toBeDefined();
49   });
50
51
52
53   describe('findAll', () => {
54     it('should return an array of sales', async () => {
55       const result = await service.findAll();
56       expect(result).toEqual(mockSaleArray);
57       expect(repository.find).toHaveBeenCalled();
58     });
59   });
60 }
```

```
TS auth.module.ts TS sales.service.spec.ts M TS user.service.spec.ts M X TS inventory.service.spec.ts M TS inventory.service.ts

src > users > TS user.service.spec.ts > describe('UserService') callback
9   describe('UserService', () => {
31     beforeEach(async () => {
42       service = module.get<UserService>(UserService);
43       repository = module.get<Repository<User>>(getRepositoryToken(User));
44     });
45
46     it('should be defined', () => {
47       expect(service).toBeDefined();
48     });
49
50     describe('createUser', () => {
51       it('should create a user', async () => {
52         const createUserDto: CreateUserDto = {
53           username: 'johndoe',
54           password: 'password123',
55         };
56         const result = await service.createUser(createUserDto);
57         expect(result).toEqual({ ...createUserDto, id: 1 });
58         expect(repository.create).toHaveBeenCalledWith(createUserDto);
59         expect(repository.save).toHaveBeenCalledWith({ ...createUserDto, id: 1 });
60       });
61     });
62
63     describe('findAll', () => {
64       it('should return an array of users', async () => {
65         const result = await service.findAll();
66         expect(result).toEqual(mockUserArray);
67         expect(repository.find).toHaveBeenCalled();
68       });
69     });
70
71     describe('findOne', () => {
72       it('should return a user', async () => {
73         const result = await service.findOne(1);
74         expect(result).toEqual(mockUser);
75         expect(repository.findOne).toHaveBeenCalledWith(1);
76       });
77     });
78
79     describe('delete', () => {
80       it('should delete a user', async () => {
81         const result = await service.delete(1);
82         expect(result).toEqual({ affected: 1 });
83         expect(repository.delete).toHaveBeenCalledWith(1);
84       });
85     });
86
87     describe('update', () => {
88       it('should update a user', async () => {
89         const result = await service.update(1, createUserDto);
90         expect(result).toEqual(createUserDto);
91         expect(repository.update).toHaveBeenCalledWith(1, createUserDto);
92       });
93     });
94
95     describe('register', () => {
96       it('should register a user', async () => {
97         const result = await service.register(createUserDto);
98         expect(result).toEqual({ token: 'token' });
99         expect(repository.create).toHaveBeenCalledWith(createUserDto);
100        expect(repository.save).toHaveBeenCalledWith({ ...createUserDto, id: 1 });
101      });
102    });
103  });
104 }
```

```
TS auth.module.ts TS sales.service.specs M TS user.service.specs M TS inventory.service.specs M X TS inventory.service.ts TS create-product

src > inventory > TS inventory.service.specs > ...
9 describe('InventoryService', () => {
10     //
11 }
12
13 describe('create', () => {
14     it('should create a product', async () => {
15         const createProductDto: CreateProductDto = {
16             name: 'Product A',
17             description: 'Description A',
18             quantity: 10,
19             price: 100,
20         };
21         const result = await service.create(createProductDto);
22         expect(result).toEqual({ ...createProductDto, id: 1 });
23         expect(repository.create).toHaveBeenCalledWith(createProductDto);
24         expect(repository.save).toHaveBeenCalledWith({ ...createProductDto, id: 1 });
25     });
26 });
27
28 describe('findAll', () => {
29     it('should return an array of products', async () => {
30         const result = await service.findAll();
31         expect(result).toEqual(mockProductArray);
32         expect(repository.find).toHaveBeenCalled();
33     });
34 });
35
36 describe('findOne', () => {
37     it('should return a single product by id', async () => {
38         const result = await service.findOne(1);
39         expect(result).toEqual(mockProduct);
40         expect(repository.findOneBy).toHaveBeenCalledWith({ id: 1 });
41     });
42 });
43 }
```