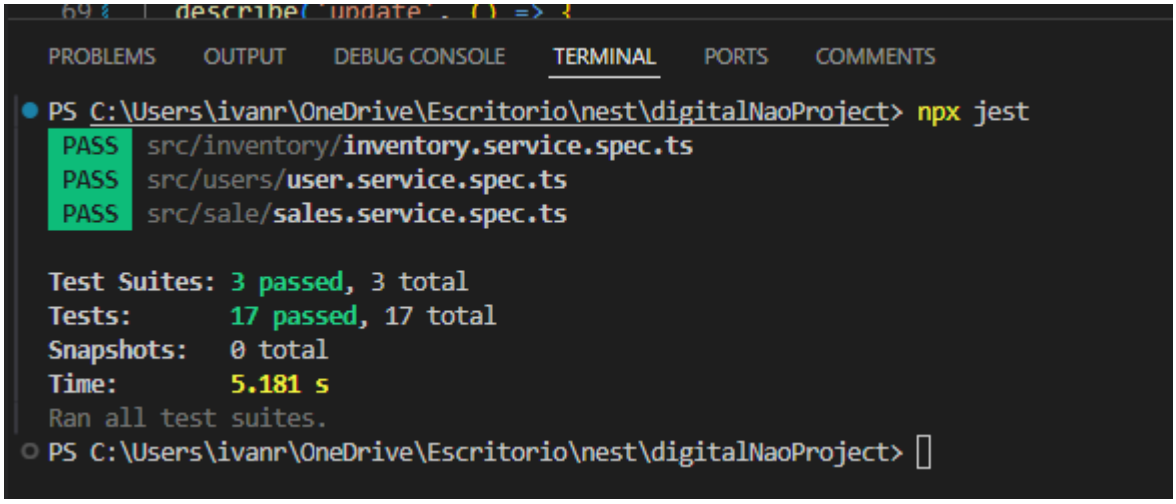

describe('update', () => {
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
PS C:\Users\ivanr\OneDrive\Escritorio\nest\digitalNaoProject> npx jest
PASS src/inventory/inventory.service.spec.ts
PASS src/users/user.service.spec.ts
PASS src/sale/sales.service.spec.ts
Test Suites: 3 passed, 3 total
Tests: 17 passed, 17 total
Snapshots: 0 total
Time: 5.181 s
Ran all test suites.
PS C:\Users\ivanr\OneDrive\Escritorio\nest\digitalNaoProject>

auth.module.ts | sales.service.spec.ts M | user.service.spec.ts M | inventory.service.spec.ts M | inventory.service.ts | create-product.dto.

src > sale > sales.service.spec.ts > describe('SaleService') callback

```ts
describe('SaleService', () => {
  const mockRepository = {
    delete: jest.fn().mockResolvedValue({ affected: 1 }),
  };

  beforeEach(async () => {
    const module: TestingModule = await Test.createTestingModule({
      providers: [
        SaleService,
        {
          provide: getRepositoryToken(Sale),
          useValue: mockRepository,
        },
      ],
    }).compile();

    service = module.get<SaleService>(SaleService);
    repository = module.get<Repository<Sale>>(getRepositoryToken(Sale));
  });

  it('should be defined', () => {
    expect(service).toBeDefined();
  });


  describe('findAll', () => {
    it('should return an array of sales', async () => {
      const result = await service.findAll();
      expect(result).toEqual(mockSaleArray);
      expect(repository.find).toHaveBeenCalled();
    });
  });
});
```

auth.module.ts | sales.service.spec.ts M | user.service.spec.ts M | inventory.service.spec.ts M | inventory.service.ts

src > users > user.service.spec.ts > describe('UsersService') callback

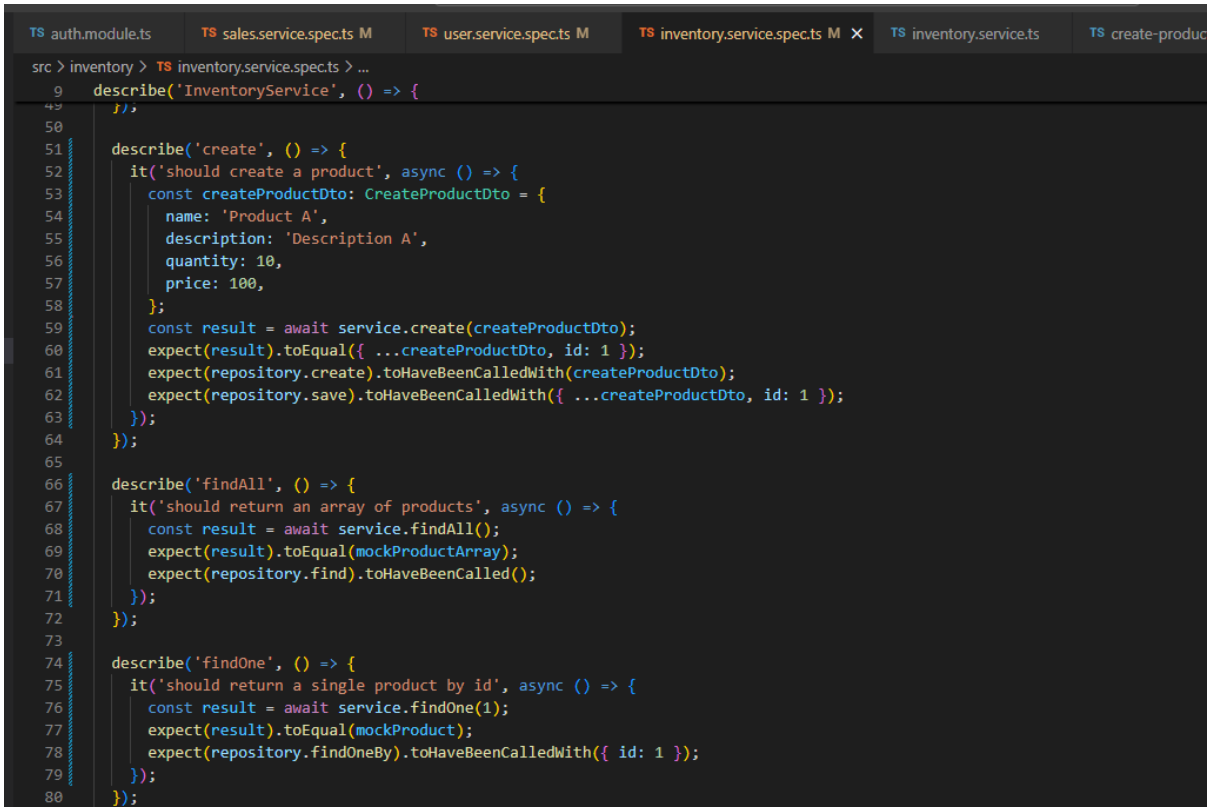
```ts
describe('UsersService', () => {
  beforeEach(async () => {
    service = module.get<UsersService>(UsersService);
    repository = module.get<Repository<User>>(getRepositoryToken(User));
  });

  it('should be defined', () => {
    expect(service).toBeDefined();
  });

  describe('createUser', () => {
    it('should create a user', async () => {
      const createUserDto: CreateUserDto = {
        username: 'johndoe',
        password: 'password123',
      };
      const result = await service.createUser(createUserDto);
      expect(result).toEqual({ ...createUserDto, id: 1 });
      expect(repository.create).toHaveBeenCalledWith(createUserDto);
      expect(repository.save).toHaveBeenCalledWith({ ...createUserDto, id: 1 });
    });
  });

  describe('findAll', () => {
    it('should return an array of users', async () => {
      const result = await service.findAll();
      expect(result).toEqual(mockUserArray);
      expect(repository.find).toHaveBeenCalled();
    });
  });

  describe('findOne', () => {
```


auth.module.ts
sales.service.spec.ts M
user.service.spec.ts M
inventory.service.spec.ts M
inventory.service.ts
create-produc
src > inventory > inventory.service.spec.ts > ...
describe('InventoryService', () => {
});
describe('create', () => {
it('should create a product', async () => {
const createProductDto: CreateProductDto = {
name: 'Product A',
description: 'Description A',
quantity: 10,
price: 100,
};
const result = await service.create(createProductDto);
expect(result).toEqual({ ...createProductDto, id: 1 });
expect(repository.create).toHaveBeenCalledWith(createProductDto);
expect(repository.save).toHaveBeenCalledWith({ ...createProductDto, id: 1 });
});
});
describe('findAll', () => {
it('should return an array of products', async () => {
const result = await service.findAll();
expect(result).toEqual(mockProductArray);
expect(repository.find).toHaveBeenCalled();
});
});
describe('findOne', () => {
it('should return a single product by id', async () => {
const result = await service.findOne(1);
expect(result).toEqual(mockProduct);
expect(repository.findOneBy).toHaveBeenCalledWith({ id: 1 });
});
});