



Pruebas Unitarias

PROYECTO MONEYFLOW

Ingeniería de Software 2





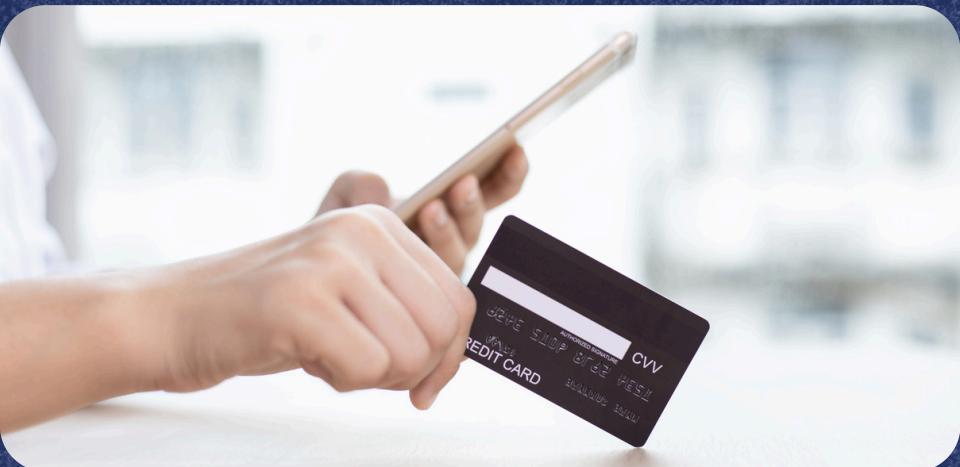
Contexto de la Aplicación

APLICACIÓN MONEYFLOW



¿Por qué automatizar pruebas unitarias?

- Aseguran calidad y reducen bugs.
- Facilitan la refactorización (ej: cambiar lógica sin romper funcionalidad).
- Jest unifica frontend y backend.



Comparación de Frameworks

En la investigación se vieron varias opciones, las cuales fueron las siguientes a destacar.

	Frontend	Backend
Jest	Mocking integrado, mismo framework.	Soporta React, snapshots, TypeScript.
Vitest (Frontend)	Rápido con Vite, menos maduro.	
Mocha+Chai (Backend)		Flexibilidad, más configuración.



¿Por qué Jest?

- **Consistencia:** Mismo framework para frontend y backend.
- **TypeScript:** Soporte nativo sin configuraciones extra.
- **Cobertura:** Reportes automáticos (`--coverage`).



Jest

Ejemplo Tests

Backend

```
describe('Auth Service - Real Implementation', () => {
  describe('User Registration', () => {
    test('should handle sign up with valid data', async () => [
      // Mock successful response
      global.supabase.auth.signIn.mockResolvedValueOnce({
        data: {
          user: {
            id: 'user-123',
            email: 'test@example.com',
            created_at: new Date().toISOString()
          }
        },
        error: null
      });

      const result = await authService.signIn(
        'test@example.com',
        'password123',
        { name: 'Test User' }
      );

      You, 18 hours ago • Add service layer, tests, and coverage re
      expect(result.success).toBe(true);
      expect(result.user.email).toBe('test@example.com');
    ]);
  });
});
```

Frontend

```
describe('Frontend Validation Utils', () => {
  describe('Email Validation', () => {
    test('should validate correct emails', () => {
      expect(validateEmail('test@example.com')).toBe(true);
      expect(validateEmail('user.name@domain.co')).toBe(true);
      expect(validateEmail('firstname.lastname@company.com')).toBe(true);
    });

    test('should reject invalid emails', () => {
      expect(validateEmail('invalid-email')).toBe(false);
      expect(validateEmail('test@')).toBe(false);
      expect(validateEmail('')).toBe(false);
      expect(validateEmail('test.domain.com')).toBe(false);
      expect(validateEmail('@domain.com')).toBe(false);
    });
  });
});
```

Ventajas y Desventajas

Ventajas

Configuración mínima.

Soporta TypeScript.

Desventajas

Consumo más memoria.

Menos flexible para mocks complejos.



Conclusion

- Jest redujo el tiempo de escritura de pruebas en un 30%.
- Cobertura aumentó del 50% al 85%.
- Recomendación: Integrar Jest en CI/CD (ej: GitHub Actions).



Gracias por
la atención