INMOVA - Configuración de Testing

📚 Documentación Completa del Sistema de Testing

Este documento describe la configuración completa del sistema de testing implementado en INMOVA, incluyendo tests unitarios, de integración y end-to-end (E2E).

🛠️ Tecnologías Utilizadas

1. Jest - Tests Unitarios

- **Propósito**: Testing de funciones, utilidades y componentes aislados
- **Configuración**: `jest.config.js`
- **Setup**: `jest.setup.js`
- **Entorno**: jsdom para simular el navegador

2. Vitest - Tests de Integración

- **Propósito**: Testing rápido y moderno con HMR
- **Configuración**: `vitest.config.ts`
- **Setup**: `vitest.setup.tsx`
- **UI**: Interfaz web interactiva para debugging

3. Playwright - Tests E2E

- **Propósito**: Testing de flujos completos de usuario
- **Configuración**: `playwright.config.ts`
- **Tests**: Directorio `e2e/`
- **Navegadores**: Chromium (configurable para Firefox y Safari)

4. Testing Library

- `@testing-library/react`: Utilidades para testing de componentes React
- `@testing-library/jest-dom`: Matchers personalizados para aserciones DOM
- `@testing-library/user-event`: Simulación de interacciones de usuario

📁 Estructura de Archivos

```
nextjs_space/
├── __tests__/                # Tests unitarios (Jest)
│   ├── components/
│   │   ├── button.test.tsx
│   │   └── kpi-card.test.tsx
│   └── lib/
│       ├── utils.test.ts
│       └── sanitize.test.ts
├── e2e/                      # Tests E2E (Playwright)
│   ├── auth.spec.ts
│   ├── buildings.spec.ts
│   ├── contracts.spec.ts
│   ├── dashboard.spec.ts
│   ├── documents.spec.ts
│   ├── maintenance.spec.ts
│   ├── navigation.spec.ts
│   ├── payments.spec.ts
│   └── tenants.spec.ts
├── jest.config.js            # Configuración Jest
├── jest.setup.js             # Setup Jest (mocks globales)
├── vitest.config.ts          # Configuración Vitest
├── vitest.setup.tsx          # Setup Vitest
└── playwright.config.ts      # Configuración Playwright
```

🚀 Scripts de Testing

Ejecutar desde el directorio `nextjs_space/`:

Jest (Tests Unitarios)

```
# Modo watch (desarrollo)
yarn test

# Ejecución única con coverage
yarn test:ci
```

Vitest (Tests de Integración)

```
# Modo watch
yarn test:unit

# Con UI interactiva
yarn test:unit:ui
```

Playwright (Tests E2E)

```
# Ejecutar todos los tests E2E
yarn test:e2e

# Con UI interactiva
yarn test:e2e:ui

# Modo debug
yarn test:e2e:debug
```

Todos los Tests

```
yarn test:all
```

📝 Convenciones de Naming

Tests Unitarios

- **Ubicación**: `__tests__/[categoria]/[nombre].test.ts(x)`
- **Naming**: `describe('NombreComponente', () => { ... })`

Tests E2E

- **Ubicación**: `e2e/[feature].spec.ts`
- **Naming**: `test('should do something', async ({ page }) => { ... })`

🧰 Configuraciones Importantes

Jest

```
// jest.config.js
{
  testEnvironment: 'jest-environment-jsdom',
  moduleNameMapper: {
    '^@/(.*)$': '<rootDir>/$1'
  },
  setupFilesAfterEnv: ['<rootDir>/jest.setup.js'],
  collectCoverageFrom: [
    'lib/**/*.{js,jsx,ts,tsx}',
    'app/**/*.{js,jsx,ts,tsx}',
    '!**/*.d.ts',
  ]
}
```

Playwright

```
// playwright.config.ts
{
  testDir: './e2e',
  use: {
    baseURL: 'http://localhost:3000',
    trace: 'on-first-retry',
    screenshot: 'only-on-failure',
  },
  webServer: {
    command: 'yarn dev',
    url: 'http://localhost:3000',
  }
}
```

🎯 Coverage Goals

Objetivos de Cobertura

- **Funciones críticas**: 90%+
- **Componentes UI**: 80%+
- **Utilidades**: 85%+
- **APIs**: 75%+

Ver Reportes de Coverage

```
yarn test:ci
open coverage/lcov-report/index.html
```

🔧 Mocks Globales

Next.js Router

```
jest.mock('next/navigation', () => ({
  useRouter: () => ({
    push: jest.fn(),
    replace: jest.fn(),
    // ...
  }),
}));
```

NextAuth

```
jest.mock('next-auth/react', () => ({
  useSession: () => ({
    data: null,
    status: 'unauthenticated',
  }),
}));
```

Next Image

```
jest.mock('next/image', () => ({
  default: (props) => <img {...props} />
}));
```

✅ Best Practices

1. Escribir Tests Legibles

```
test('should display user name when logged in', () => {
  // Arrange
  const user = { name: 'John Doe' };

  // Act
  render(<UserProfile user={user} />);

  // Assert
  expect(screen.getByText('John Doe')).toBeInTheDocument();
});
```

2. Usar Data-Testid para Selectores Estables

```
// Componente
<button data-testid="submit-button">Submit</button>

// Test
const button = screen.getByTestId('submit-button');
```

3. Simular Interacciones Reales

```
import { userEvent } from '@testing-library/user-event';

test('should submit form on button click', async () => {
  const user = userEvent.setup();
  render(<LoginForm />);

  await user.type(screen.getByLabelText('Email'), 'test@example.com');
  await user.click(screen.getByRole('button', { name: 'Login' }));

  expect(mockSubmit).toHaveBeenCalled();
});
```

4. Tests E2E Resilientes

```
// Usar localizadores semánticos
await page.getByRole('button', { name: 'Login' }).click();

// Esperar a estados específicos
await page.waitForURL('/dashboard');

// Capturar errores de red
page.on('pageerror', (error) => {
  console.error('Page error:', error);
});
```

🐛 Debugging

Jest

```
# Ejecutar un test específico
yarn test path/to/test.test.ts

# Modo debug con Node inspector
node --inspect-brk node_modules/.bin/jest --runInBand
```

Playwright

```
# UI interactiva
yarn test:e2e:ui

# Modo debug con Playwright Inspector
yarn test:e2e:debug

# Ver trace de un test fallido
npx playwright show-trace trace.zip
```

📦 Dependencias Instaladas

```
{
  "devDependencies": {
    "jest": "latest",
    "jest-environment-jsdom": "latest",
    "@testing-library/react": "latest",
    "@testing-library/jest-dom": "latest",
    "@testing-library/user-event": "latest",
    "@playwright/test": "latest",
    "vitest": "latest",
    "@vitejs/plugin-react": "latest",
    "@vitest/ui": "latest",
    "jsdom": "latest"
  }
}
```

📊 CI/CD Integration

GitHub Actions Example

```yaml
name: Test

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Install dependencies
        run: yarn install

      - name: Run unit tests
        run: yarn test:ci

      - name: Install Playwright browsers
        run: npx playwright install --with-deps

      - name: Run E2E tests
        run: yarn test:e2e

      - name: Upload coverage
        uses: codecov/codecov-action@v3
```

📚 Recursos Adicionales

- Jest Documentation (https://jestjs.io/)
- Vitest Documentation (https://vitest.dev/)
- Playwright Documentation (https://playwright.dev/)
- Testing Library (https://testing-library.com/)
- Next.js Testing Guide (https://nextjs.org/docs/testing)

❓ Troubleshooting

Error: “Cannot find module ‘@/...’“

➡️ Verificar `moduleNameMapper` en `jest.config.js`

Error: “ReferenceError: window is not defined”

➡️ Asegurar que `testEnvironment: 'jsdom'` esté configurado

Tests E2E lentos

➡️ Considerar ejecutar en paralelo: `workers: 4` en `playwright.config.ts`

Tests flaky (intermitentes)

➡️ Añadir `retries: 2` en la configuración de Playwright

🔄 Actualizaciones

Última actualización: 2 de diciembre de 2025
Versión: 1.0
Mantenido por: Equipo de Desarrollo INMOVA

📧 **Contacto**: Para dudas sobre testing, contactar al equipo de QA en `qa@inmova.com`