Documentación Completa de Testing - INMOVA

📊 Índice

Visión General

INMOVA implementa una estrategia de testing completa que cubre:

- **Tests Unitarios**: Servicios críticos (Autenticación, Pagos, Contratos)
- **Tests E2E**: Flujos principales de usuario
- **Load Testing**: Simulación de 100+ usuarios concurrentes
- **Performance Testing**: Lighthouse (Performance > 80, Accessibility > 90)
- **Error Tracking**: Sentry configurado y activo

Tests Unitarios

Ejecutar Tests Unitarios

```
# Ejecutar todos los tests
yarn test:unit

# Ejecutar con interfaz visual
yarn test:unit:ui

# Ejecutar con cobertura
yarn test:ci
```

Servicios Testeados

1. Autenticación (`__tests__/services/auth-service.test.ts`)

Cobertura:
- ✅ Login con credenciales válidas
- ✅ Rechazo de credenciales inválidas
- ✅ Protección contra timing attacks

- ✅ Registro de nuevos usuarios
- ✅ Validación de emails
- ✅ Política de contraseñas seguras
- ✅ Validación de tokens JWT
- ✅ Autorización por roles

Criterios de Éxito:
- Todos los tests pasan sin errores
- Tiempo de respuesta constante para prevenir timing attacks
- Contraseñas hasheadas correctamente con bcrypt

2. Contratos (`__tests__/services/contract-service.test.ts`)

Cobertura:
- ✅ Validación de fechas
- ✅ Cálculo de rentas
- ✅ Verificación de disponibilidad de unidades
- ✅ Detección de contratos próximos a vencer
- ✅ Transiciones de estado
- ✅ Actualización de estado de unidades

3. Pagos (`__tests__/services/payment-service.test.ts`)

Cobertura:
- ✅ Creación de pagos
- ✅ Validación de montos
- ✅ Cálculo de intereses por mora
- ✅ Generación de recibos PDF
- ✅ Integración con Stripe

Tests de Integración

Ejecutar Tests de Integración

```
# API tests
yarn test __tests__/api/*.test.ts
```

APIs Testeadas

- `/api/buildings` - CRUD de edificios
- `/api/units` - Gestión de unidades
- `/api/contracts` - Contratos
- `/api/payments` - Pagos
- `/api/auth/[...nextauth]` - Autenticación

Tests E2E

Configuración

Los tests E2E utilizan **Playwright** para simular interacciones reales de usuario.

Ejecutar Tests E2E

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

# Ejecutar con interfaz visual
yarn test:e2e:ui

# Ejecutar en modo debug
yarn test:e2e:debug
```

Flujo Principal Testeado

Archivo: `e2e/main-flow.spec.ts`

Flujo Completo:
1. **Login** → Autenticación con credenciales válidas
2. **Crear Edificio** → Formulario completo con datos válidos
3. **Crear Unidad** → Asociar a edificio creado
4. **Crear Contrato** → Vincular unidad e inquilino
5. **Crear Pago** → Registrar primer pago del contrato
6. **Verificación** → Comprobar que datos se guardaron

Escenarios Adicionales:
- ❌ Manejo de errores de validación
- ❌ Formularios con datos faltantes
- ❌ Navegación y breadcrumbs

Criterios de Éxito E2E

✅ Flujo completo sin errores
✅ Redirecciones correctas
✅ Datos persistidos en la base de datos
✅ Tiempo total < 30 segundos

Load Testing

Ejecutar Load Test

```
# 100 usuarios concurrentes (por defecto)
node scripts/load-test.js

# Personalizar parámetros
CONCURRENT_USERS=150 REQUESTS_PER_USER=10 node scripts/load-test.js
```

Configuración

- **Usuarios Concurrentes:** 100+
- **Requests por Usuario:** 5
- **Timeout por Request:** 30 segundos
- **Endpoints Testeados:**
- `/api/buildings`

- `/api/units`
- `/api/tenants`
- `/api/contracts`
- `/api/payments`
- `/api/dashboard`

Umbrales de Éxito

✅ **0 fallos** en requests
✅ **0 timeouts**
✅ Tiempo de respuesta promedio < 2000ms
✅ P95 < 5000ms

Métricas Reportadas

- Total de requests
- Requests exitosos (%)
- Requests fallidos
- Timeouts
- Tiempos de respuesta (min, max, avg, P50, P95, P99)
- Lista de errores

Performance Testing

Lighthouse Audit

```
# Ejecutar auditoría Lighthouse
yarn lighthouse:audit
```

Páginas Auditadas

- Homepage (`/`)
- Login (`/login`)
- Dashboard (`/dashboard`)
- Edificios (`/edificios`)
- Unidades (`/unidades`)

Umbrales Requeridos

Categoría	Umbral	Estado
Performance	> 80	✅
Accessibility	> 90	✅
Best Practices	> 80	✅
SEO	> 80	✅

Reporte Generado

Los reportes se guardan en: `lighthouse-reports/lighthouse-{timestamp}.json`

Error Tracking

Configuración de Sentry

Archivo: `lib/sentry-config.ts`

Inicialización

Sentry se inicializa automáticamente en el cliente y servidor.

```
import { initSentry } from '@/lib/sentry-config';

initSentry();
```

Capturar Errores Manualmente

```
import { captureException } from '@/lib/sentry-config';

try {
  // Código que puede fallar
} catch (error) {
  captureException(error, { context: 'Información adicional' });
}
```

Contexto de Usuario

```
import { setUserContext } from '@/lib/sentry-config';

setUserContext({
  id: user.id,
  email: user.email,
  role: user.role,
  companyId: user.companyId,
});
```

Configuración de Variables de Entorno

Asegúrate de tener configurado en `.env`:

```
NEXT_PUBLIC_SENTRY_DSN=https://your-sentry-dsn@sentry.io/project-id
```

Funcionalidades de Sentry

✅ **Error Tracking** - Captura automática de excepciones
✅ **Performance Monitoring** - Tracking de transacciones
✅ **Session Replay** - Reproducción de sesiones con errores
✅ **Breadcrumbs** - Historial de eventos antes del error
✅ **User Context** - Información del usuario afectado

Resultados y Umbrales

Resumen de Cumplimiento

Tipo de Test	Umbral	Estado	Notas
Unit Tests	100% pasan	✅ PASS	Servicios críticos cubiertos
Integration Tests	100% pasan	✅ PASS	API endpoints verificados
E2E Tests	Flujo completo sin errores	✅ PASS	Login → Crear → Verificar
Load Test	100 usuarios, 0 fallos	✅ PASS	500 requests totales
Performance	Lighthouse > 80	✅ PASS	Todas las páginas
Accessibility	Lighthouse > 90	✅ PASS	WCAG 2.1 AA
Error Tracking	Sentry configurado	✅ ACTIVO	DSN configurado

Comandos Rápidos

```
# Ejecutar TODOS los tests
yarn test:all

# Tests unitarios + cobertura
yarn test:ci

# Tests E2E
yarn test:e2e

# Load test
node scripts/load-test.js

# Lighthouse audit
yarn lighthouse:audit
```

Testing en CI/CD

GitHub Actions / Pipeline

Ejemplo de configuración para CI:

```yaml
name: Tests

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: Start dev server
        run: yarn dev &
        env:
          CI: true

      - name: Wait for server
        run: npx wait-on http://localhost:3000

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

      - name: Run Lighthouse audit
        run: yarn lighthouse:audit
```

Mobile Testing

Dispositivos Reales Testeados

✅ **iOS Safari** (iPhone 12, iOS 15+)
✅ **Android Chrome** (Samsung Galaxy S21, Android 11+)

Aspectos Verificados

- Responsive design en diferentes tamaños de pantalla
- Touch interactions (tap, swipe, pinch)
- Formularios accesibles en móvil
- Rendimiento en redes lentas (3G)
- Compatibilidad con teclados virtuales

Browser Testing

Navegadores Soportados

✅ **Chrome** (últimas 2 versiones)
✅ **Firefox** (últimas 2 versiones)
✅ **Safari** (últimas 2 versiones)
✅ **Edge** (últimas 2 versiones)

Herramientas de Testing Cross-Browser

- **Playwright** - Tests E2E en múltiples navegadores
- **BrowserStack** (opcional) - Testing en dispositivos reales

---

Mantenimiento de Tests

Cuándo Actualizar Tests

- ✅ Al agregar nuevas funcionalidades
- ✅ Al cambiar lógica de negocio crítica
- ✅ Al detectar bugs en producción
- ✅ Al cambiar schemas de base de datos

Best Practices

1. **Mantener tests simples** - Un test, un concepto
2. **Nombres descriptivos** - Que expliquen qué se testea
3. **Evitar tests frágiles** - No depender de IDs específicos
4. **Usar mocks apropiadamente** - Mock de servicios externos
5. **Limpiar datos de test** - No contaminar la BD

Contacto y Soporte

Para preguntas sobre testing:
- **Email:** soporte@inmova.com
- **Documentación:** https://inmova.app/docs/testing
