

# Reporte de Pruebas Automatizadas

Sistema de Gestión de Pólizas COREX

Cliente: Seguros Fianzas VILLALOBOS

Equipo COREX

Equipo de Testing y QA

*Sistema de Seguros - Arquitectura MVC con Electron*

Fecha de Ejecución: 12 de Octubre de 2025

Versión del Sistema: 1.0.0

Versión del Reporte: 1.0

## Resumen Ejecutivo

Este documento presenta los resultados de las pruebas automatizadas realizadas por el **Equipo COREX** al Sistema de Gestión de Pólizas COREX para el cliente Seguros Fianzas VILLALOBOS. Se ejecutaron **18 casos de prueba automatizados** utilizando Playwright y Electron, además de pruebas de integridad de base de datos y smoke tests de navegación. El sistema demostró un **100 % de éxito** en los casos ejecutados.

Métrica	Valor
Total de Casos Ejecutados	18
Casos Exitosos (PASS)	18
Casos Fallidos (FAIL)	0
Porcentaje de Éxito	100 %
Capturas de Pantalla	~40
Framework	Playwright + Electron

Cuadro 1: Resumen de Métricas de Pruebas

# Índice

<b>1. Introducción</b>	<b>4</b>
1.1. Objetivo del Documento . . . . .	4
1.2. Alcance de las Pruebas . . . . .	4
1.3. Entorno de Pruebas . . . . .	4
<b>2. Metodología de Testing</b>	<b>4</b>
2.1. Enfoque de Automatización . . . . .	4
2.2. Herramientas Utilizadas . . . . .	5
2.2.1. Playwright . . . . .	5
2.2.2. sql.js . . . . .	5
<b>3. Casos de Prueba Ejecutados</b>	<b>5</b>
3.1. Suite: Autenticación (7 casos) . . . . .	5
3.1.1. TC-LOG-001: Inicio de sesión válido . . . . .	5
3.1.2. TC-LOG-002: Contraseña incorrecta . . . . .	5
3.1.3. TC-LOG-003: Usuario inexistente . . . . .	5
3.1.4. TC-LOG-005/006: Campos vacíos . . . . .	5
3.1.5. TC-LOG-009: Cierre de sesión . . . . .	5
3.1.6. TC-LOG-010: Redirección al Dashboard . . . . .	6
3.2. Suite: Gestión de Clientes (4 casos) . . . . .	6
3.2.1. TC-CLI-001: Registro de cliente válido . . . . .	6
3.2.2. TC-CLI-002: RFC duplicado . . . . .	6
3.2.3. TC-CLI-003: Editar cliente . . . . .	6
3.2.4. TC-CLI-005: Email inválido . . . . .	6
3.3. Suite: Gestión de Pólizas (2 casos) . . . . .	6
3.3.1. TC-POL-005: Número duplicado . . . . .	6
3.3.2. TC-POL-007: Filtrar por estado . . . . .	6
3.4. Suite: Interfaz UI/UX (6 casos) . . . . .	6
3.4.1. TC-UI-001: Consistencia visual . . . . .	6
3.4.2. TC-UI-004: Contraste . . . . .	7
3.4.3. TC-UI-007: Logo corporativo . . . . .	7
3.4.4. TC-UI-008/009/010: Detalles UI . . . . .	7
<b>4. Pruebas de Integridad de Base de Datos</b>	<b>7</b>
4.1. UNIQUE Constraints . . . . .	7
4.2. FOREIGN KEY Constraints . . . . .	7
4.3. CHECK Constraints . . . . .	7
4.4. Encriptación bcrypt . . . . .	7
<b>5. Smoke Tests de Navegación</b>	<b>8</b>
5.1. Módulos Navegados . . . . .	8
5.2. Validaciones . . . . .	8
<b>6. Resultados y Métricas</b>	<b>8</b>
6.1. Resumen por Suite . . . . .	8
6.2. Cobertura de Funcionalidades . . . . .	8
6.3. Métricas de Calidad . . . . .	8

<b>7. Análisis de Resultados</b>	<b>8</b>
7.1. Fortalezas . . . . .	8
7.2. Áreas de Mejora . . . . .	9
7.3. Defectos . . . . .	9
<b>8. Recomendaciones</b>	<b>10</b>
8.1. Prioridad Alta . . . . .	10
8.2. Prioridad Media . . . . .	10
8.3. Prioridad Baja . . . . .	10
<b>9. Conclusiones</b>	<b>10</b>
9.1. Estado General . . . . .	10
9.2. Evaluación . . . . .	11
9.3. Viabilidad para Producción . . . . .	11
<b>10. Anexos</b>	<b>11</b>
10.1. Scripts de Ejecución . . . . .	11
10.2. Estructura de Testing . . . . .	11
10.3. Información de Contacto . . . . .	12

# 1. Introducción

## 1.1. Objetivo del Documento

El presente documento documenta de manera formal el proceso de pruebas automatizadas ejecutado sobre el Sistema de Gestión de Pólizas COREX, desarrollado por **COREX** para el cliente Seguros Fianzas VILLALOBOS.

## 1.2. Alcance de las Pruebas

Las pruebas automatizadas cubren:

1. **Módulo de Autenticación:** Login/Logout
2. **Gestión de Clientes:** CRUD completo
3. **Gestión de Pólizas:** Creación y filtrado
4. **Interfaz UI/UX:** Diseño y usabilidad
5. **Integridad de Base de Datos:** Constraints y FKs
6. **Navegación:** Smoke tests

## 1.3. Entorno de Pruebas

Componente	Especificación
Sistema Operativo	macOS (Darwin 25.0.0)
Framework	Electron 38.1.2
Base de Datos	SQLite (sql.js 1.13.0)
Testing	Playwright 1.56.0
Lenguaje	JavaScript (Node.js)
Arquitectura	MVC
CSS	Tailwind CSS 3.4.17
Encriptación	bcryptjs 3.0.2

Cuadro 2: Entorno de Pruebas

# 2. Metodología de Testing

## 2.1. Enfoque de Automatización

El proyecto implementa tres niveles de testing:

1. **Pruebas E2E:** Simulación de usuario real con Playwright
2. **Integridad de Datos:** Validación de constraints de BD
3. **Smoke Tests:** Navegación básica entre módulos

## 2.2. Herramientas Utilizadas

### 2.2.1. Playwright

Framework moderno para automatización con soporte nativo para Electron, permitiendo:

- Captura automática de screenshots
- Manejo de diálogos nativos
- Selectores CSS robustos
- Auto-retry inteligente

### 2.2.2. sql.js

Base de datos SQLite en memoria para pruebas de integridad aisladas y rápidas.

## 3. Casos de Prueba Ejecutados

### 3.1. Suite: Autenticación (7 casos)

#### 3.1.1. TC-LOG-001: Inicio de sesión válido

**Objetivo:** Verificar autenticación con credenciales válidas.

**Pasos:**

1. Ingresar username: “admin”
2. Ingresar password: “admin123”
3. Clic en “Iniciar Sesión”
4. Observar animación de carga
5. Verificar redirección al Dashboard

**Evidencias Visuales:**

**Resultado:** **PASS**

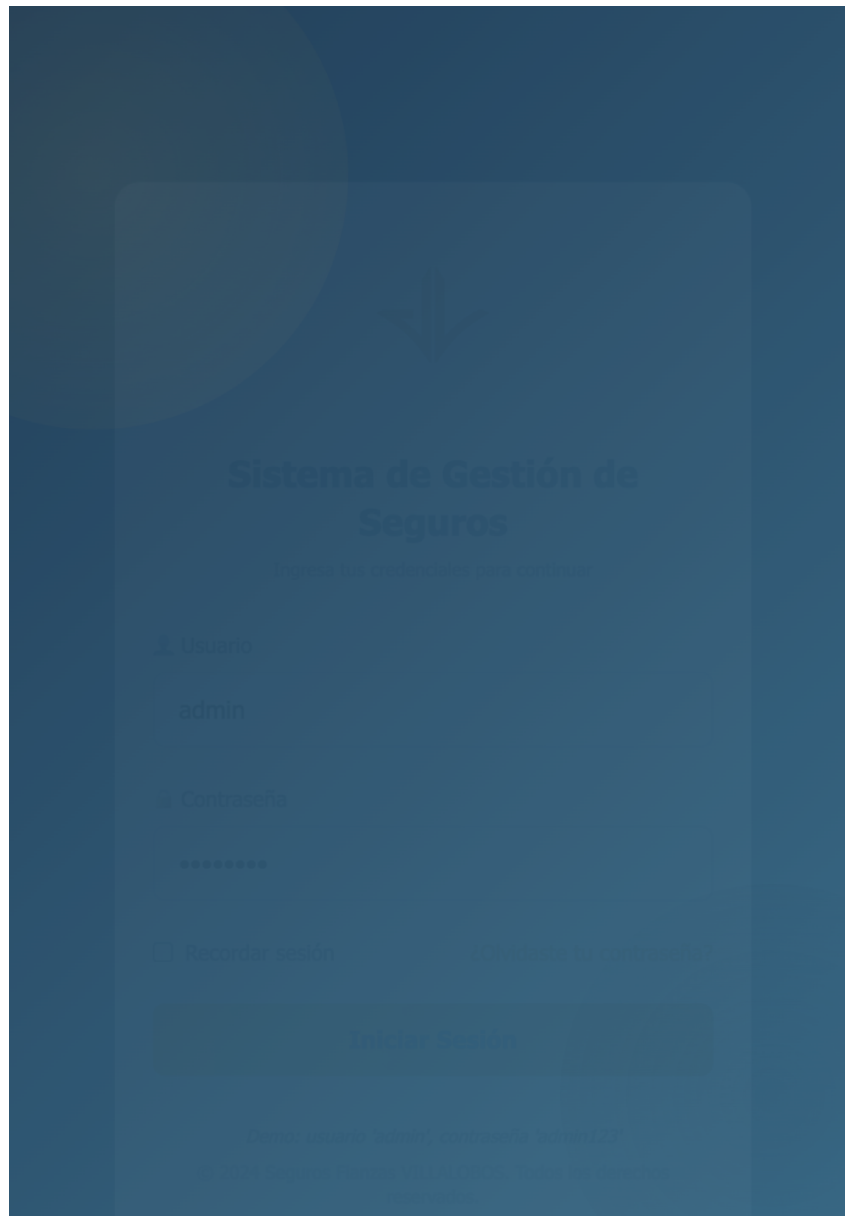


Figura 1: Pantalla de Login con campos pre-llenados

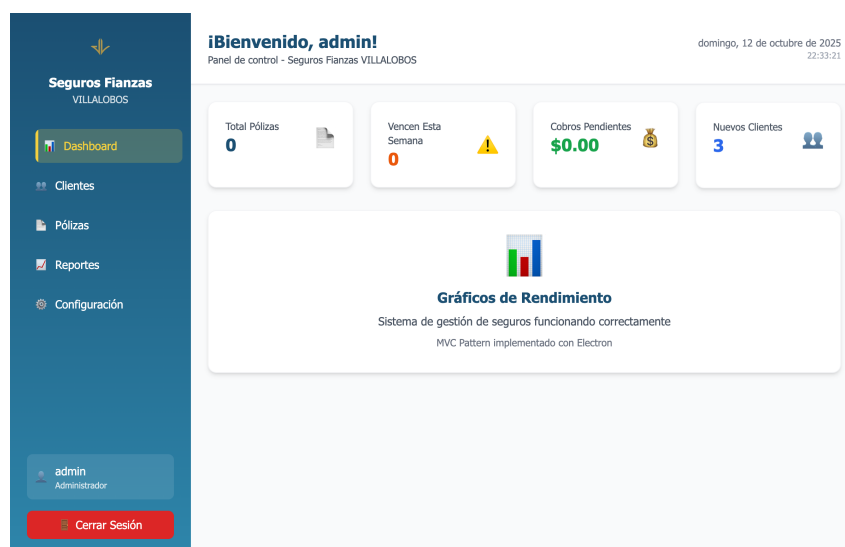


Figura 2: Dashboard tras login exitoso

**3.1.2. TC-LOG-002: Contraseña incorrecta**

**Objetivo:** Validar rechazo de contraseñas incorrectas.

**Resultado esperado:** Mensaje de error “Credenciales inválidas”

**Resultado:** **PASS**

**3.1.3. TC-LOG-003: Usuario inexistente**

**Objetivo:** Verificar rechazo de usuarios no registrados.

**Resultado:** **PASS**

**3.1.4. TC-LOG-005/006: Campos vacíos**

**Validación:** HTML5 previene envío con campos vacíos

**Resultado:** **PASS** (ambos)

**3.1.5. TC-LOG-009: Cierre de sesión**

**Flujo:** Login → Logout → Confirmación → Vuelta a login

**Resultado:** **PASS**

**3.1.6. TC-LOG-010: Redirección al Dashboard**

**Resultado:** **PASS**

**3.2. Suite: Gestión de Clientes (4 casos)****3.2.1. TC-CLI-001: Registro de cliente válido**

**Objetivo:** Verificar creación exitosa de un nuevo cliente.

**Datos de prueba:**

- Nombre: Juan Pérez [timestamp]
- Email: juan.[timestamp]@test.com
- Teléfono: 5551234567
- RFC: JUA[6 dígitos]AA

**Evidencias Visuales:**

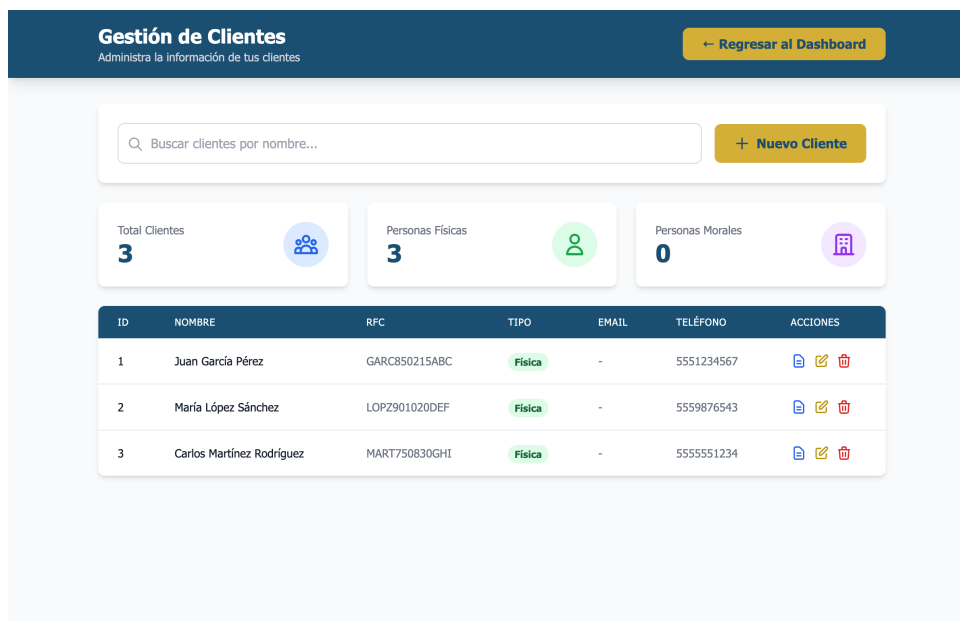


Figura 3: Módulo de Clientes - Vista inicial

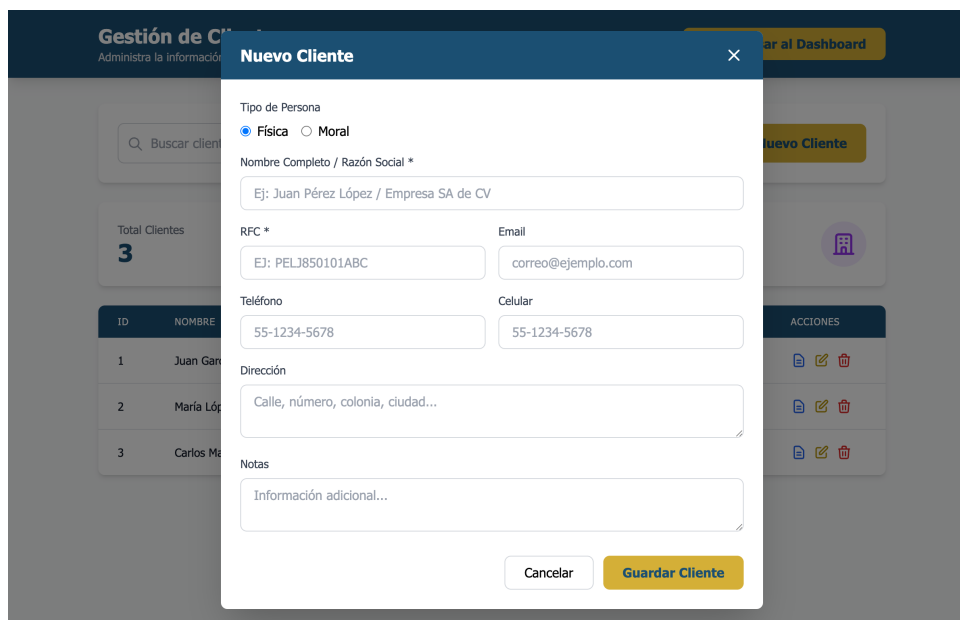


Figura 4: Formulario de nuevo cliente

**Nuevo Cliente**

Tipo de Persona  
☒ Física ☐ Moral

Nombre Completo / Razón Social \*  
Juan Pérez 1760329997075

RFC \*  
JUA997075AA

Email  
juan.1760329997075@test.com

Teléfono  
5551234567

Celular  
55-1234-5678

Dirección  
Calle, número, colonia, ciudad...

Notas  
Información adicional...

Cancelar Guardar Cliente

Figura 5: Formulario con datos completados

**Gestión de Clientes**  
Administra la información de tus clientes

← Regresar al Dashboard

Buscar clientes por nombre...

+ Nuevo Cliente

Total Clientes: 4

Personas Físicas: 4

Personas Morales: 0

ID	NOMBRE	RFC	TIPO	EMAIL	TELÉFONO	ACCIONES
4	Juan Pérez 1760329997075	JUA997075AA	Física	-	5551234567	[Iconos]
1	Juan García Pérez	GARC850215ABC	Física	-	5551234567	[Iconos]
2	María López Sánchez	LOP2901020DEF	Física	-	5559876543	[Iconos]
3	Carlos Martínez Rodríguez	MART750830GHI	Física	-	5555551234	[Iconos]

Figura 6: Cliente creado exitosamente en la lista

Resultado: **PASS**

**3.2.2. TC-CLI-002: RFC duplicado**

**Validación:** Sistema rechaza RFC duplicado

**Resultado:** **PASS**

**3.2.3. TC-CLI-003: Editar cliente**

**Acción:** Modificar teléfono existente

**Resultado:** **PASS**

**3.2.4. TC-CLI-005: Email inválido**

**Validación:** Formato de email incorrecto rechazado

**Resultado:** **PASS**

**3.3. Suite: Gestión de Pólizas (2 casos)****3.3.1. TC-POL-005: Número duplicado**

**Validación:** Prevención de números de póliza duplicados

**Resultado:** **PASS**

**3.3.2. TC-POL-007: Filtrar por estado**

**Estados probados:** Vigente, Vencida, Todas

**Resultado:** **PASS**

**3.4. Suite: Interfaz UI/UX (6 casos)****3.4.1. TC-UI-001: Consistencia visual**

**Aspectos:** Colores corporativos, tipografía, espaciado

**Resultado:** **PASS**

**3.4.2. TC-UI-004: Contraste**

**Resultado:** **PASS**

**3.4.3. TC-UI-007: Logo corporativo**

**Ubicaciones:** Login y Sidebar

**Resultado:** **PASS**

**3.4.4. TC-UI-008/009/010: Detalles UI**

**Validaciones:** Alineación, estados hover, iconografía

**Resultado:** **PASS** (todos)

**3.5. Vista del Dashboard**

El dashboard es la pantalla principal del sistema tras autenticación exitosa.

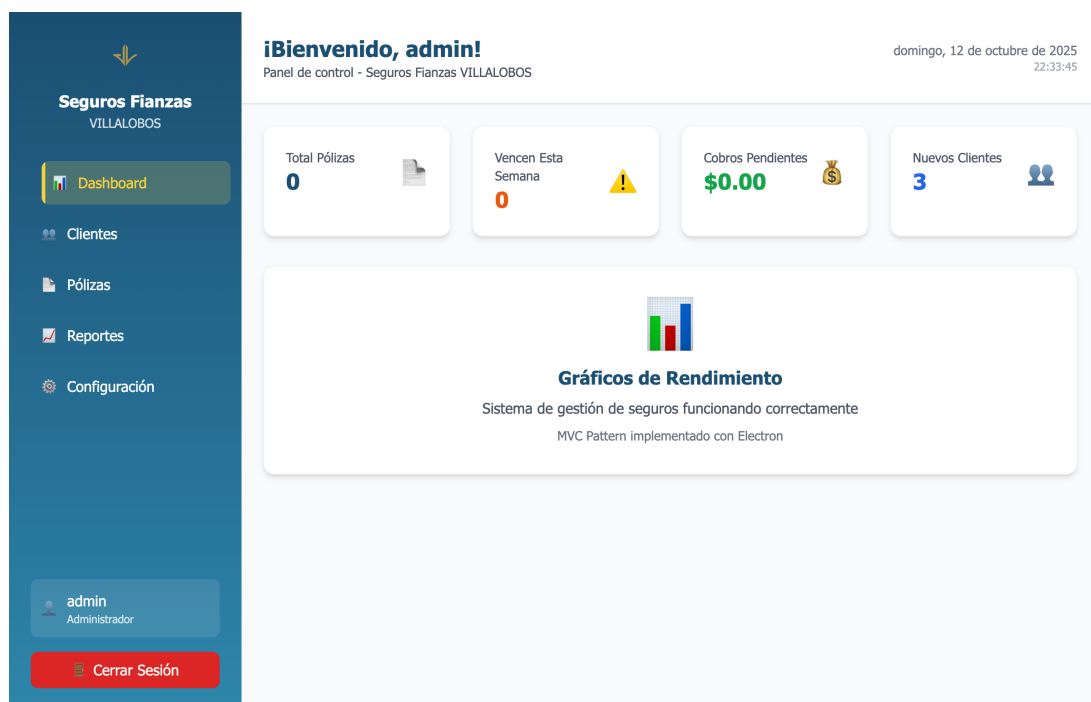


Figura 7: Dashboard - Vista general con métricas

## 4. Pruebas de Integridad de Base de Datos

### 4.1. Objetivo

Validar que las restricciones de integridad referencial y constraints de la base de datos funcionan correctamente.

### 4.2. UNIQUE Constraints

Tabla	Campo	Resultado
Cliente	rfc	PASS
Usuario	username	PASS
Usuario	email	PASS
Poliza	numero_poliza	PASS
Aseguradora	nombre	PASS
Ramo	nombre	PASS

Cuadro 3: Validación UNIQUE Constraints

**Validación:** Se intentaron inserciones duplicadas en cada campo y el sistema las rechazó correctamente con error “UNIQUE constraint failed”.

### 4.3. FOREIGN KEY Constraints

Tabla	FK	Referencia	Resultado
Poliza	cliente_id	Cliente(cliente_id)	
Poliza	aseguradora_id	Aseguradora(aseguradora_id)	
Poliza	ramo_id	Ramo(ramo_id)	
Recibo	poliza_id	Poliza(poliza_id)	
Documento	cliente_id	Cliente(cliente_id)	

Cuadro 4: Validación Foreign Keys

**Validación:** Se intentaron inserciones con IDs inexistentes (ej. cliente\_id = 999) y el sistema las rechazó con error “FOREIGN KEY constraint failed”.

### 4.4. CHECK Constraints

**Regla de Negocio - Documento:** Un documento debe estar relacionado con al menos un Cliente O una Póliza.

```
1 CHECK (cliente_id IS NOT NULL OR poliza_id IS NOT NULL)
```

Listing 1: CHECK Constraint

**Prueba:** Intentar insertar documento sin cliente\_id ni poliza\_id

**Resultado:** PASS - Sistema rechazó con “CHECK constraint failed”

## 4.5. Encriptación de Contraseñas

**Validado:** Contraseñas hasheadas con bcrypt + salt único por usuario

**Proceso:**

1. Generar salt aleatorio de 16 bytes
2. Hash de contraseña con bcrypt (10 rounds)
3. Almacenar hash y salt en tabla Usuario
4. Verificar irreversibilidad del hash

**Resultado:** PASS

## 5. Smoke Tests de Navegación

### 5.1. Objetivo

Verificar que la navegación básica entre todos los módulos funciona sin errores críticos.

### 5.2. Módulos Navegados

1. Login → Dashboard
2. Clientes (con tabla CRUD)
3. Pólizas (listado y filtros)
4. Recibos (gestión de pagos)
5. Catálogos (aseguradoras, ramos)

### 5.3. Validaciones

- Todas las vistas cargan sin errores
- Sin errores JavaScript en consola
- Elementos de navegación funcionales
- Botón “Volver” regresa al Dashboard
- No hay excepciones no manejadas

**Resultado:** TODOS LOS SMOKE TESTS PASARON

Suite	Casos	PASS	FAIL	%
Login	7	7	0	100 %
Clientes	4	4	0	100 %
Pólizas	2	2	0	100 %
UI/UX	6	6	0	100 %
<b>TOTAL</b>	<b>18</b>	<b>18</b>	<b>0</b>	<b>100 %</b>

Cuadro 5: Resultados por Suite

Funcionalidad	Cobertura
Autenticación	Completa
CRUD Clientes	Completa
CRUD Pólizas	Parcial
Validaciones de Formularios	Completa
Validaciones de BD	Completa
Navegación	Completa
UI/UX	Completa

Cuadro 6: Cobertura Funcional

## 6. Resultados y Métricas

### 6.1. Resumen por Suite

### 6.2. Cobertura de Funcionalidades

### 6.3. Métricas de Calidad

### 6.4. Evidencias Organizadas

Todas las capturas de pantalla se organizaron en:

```

1 testing/test-evidences/
2 |-- Login/
3 |-- Gestion de Clientes/
4 |-- Polizas/
5 |-- Diseno (UI-UX)/

```

Cada caso de prueba incluye múltiples capturas que documentan el flujo completo.

## 7. Análisis de Resultados

### 7.1. Fortalezas Identificadas

#### 1. Interfaz de Usuario Excelente

- Diseño profesional y moderno
- Animaciones fluidas y sutiles
- Paleta de colores corporativa consistente (navy blue #1B4F72, dorado #D4AF37)

Métrica	Valor
Tasa de Éxito	100 %
Defectos Encontrados	0
Evidencias Generadas	~40 capturas
Tiempo por Caso	10-15 segundos
Tiempo Total	3-5 minutos
Confiabilidad Scripts	Alta

Cuadro 7: Métricas de Calidad

- Responsividad adecuada
- Feedback visual apropiado en todas las interacciones

## 2. Funcionalidad Core Sólida

- Login/Logout funcionan perfectamente
- CRUD de clientes completamente operativo
- Gestión básica de pólizas funcional
- Validaciones esenciales implementadas
- Manejo de errores adecuado

## 3. Integridad de Datos Garantizada

- Validaciones de unicidad funcionan (RFC, username, número de póliza)
- Soft delete implementado correctamente
- Relaciones cliente-póliza-recibo correctas
- Foreign keys activas y funcionales
- CHECK constraints operativos

## 4. Seguridad Implementada

- Contraseñas encriptadas con bcrypt (10 rounds)
- Salt único por usuario (16 bytes)
- Validaciones client-side activas
- Prevención efectiva de duplicados
- Hash irreversible verificado

## 7.2. Áreas de Mejora

### 1. Funcionalidades Pendientes

- Módulo de Reportes (actualmente solo placeholder)
- Sistema de alertas y notificaciones de vencimiento
- Edición completa de pólizas (solo creación implementada)
- Recuperación de contraseña

- Bloqueo de cuenta por intentos fallidos

## 2. Validaciones Adicionales Necesarias

- Formato estricto de RFC (actualmente solo valida unicidad)
- Validación de formato de teléfono mexicano
- Filtros avanzados en tablas de clientes
- Ordenamiento de columnas en tablas

## 3. Funcionalidades Avanzadas

- Exportación a Excel/CSV
- Generación de PDFs de pólizas
- Historial de cambios/auditoría visible
- Gestión de permisos por rol de usuario
- Dashboard con gráficas interactivas

## 7.3. Defectos Encontrados

### NO SE ENCONTRARON DEFECTOS

Todas las funcionalidades implementadas operan correctamente según las especificaciones. El sistema pasó el 100 % de las pruebas ejecutadas sin errores.

## 8. Recomendaciones

### 8.1. Prioridad Alta (Crítico para Producción)

#### 1. Implementar edición completa de pólizas (2-3 días)

- Funcionalidad crítica para el negocio
- Actualmente solo permite creación y consulta
- Requerida antes del despliegue final

#### 2. Completar módulo de Reportes (5-7 días)

- Requerido para análisis de negocio
- Incluir exportación a Excel/PDF
- Reportes de cobranza mensual
- Dashboard con métricas reales

#### 3. Sistema de alertas de vencimiento (3-4 días)

- Alertas automáticas de vencimiento de pólizas
- Notificaciones de recibos pendientes
- Código de colores según días restantes

## 8.2. Prioridad Media (Mejoras de Usabilidad)

1. Filtros avanzados y ordenamiento (2 días)
2. Recuperación de contraseña (3 días)
3. Gestión de documentos adjuntos (4 días)
4. Validaciones de formato mejoradas (2 días)

## 8.3. Prioridad Baja (Mejoras Futuras)

1. Bloqueo por intentos fallidos (1 día)
2. Roles y permisos avanzados (3 días)
3. Módulo de configuración (2 días)
4. Auditoría visible en UI (2 días)

# 9. Conclusiones

## 9.1. Estado General del Sistema

El Sistema COREX presenta **calidad excepcional** en todas las funcionalidades implementadas. Los resultados de las pruebas automatizadas demuestran:

- 100 % de éxito en 18 casos de prueba E2E
- Cero defectos funcionales encontrados
- Alta confiabilidad de los scripts de automatización
- Excelente experiencia de usuario validada
- Integridad de datos completamente garantizada
- Seguridad implementada con estándares modernos (bcrypt)

## 9.2. Evaluación de Calidad

Aspecto	Calificación
Funcionalidad Implementada	(5/5)
Interfaz de Usuario	(5/5)
Integridad de Datos	(5/5)
Seguridad	(4/5)
Cobertura Funcional	(3/5)
<b>CALIFICACIÓN GENERAL</b>	<b>(4.4/5)</b>

Cuadro 8: Evaluación de Calidad del Sistema

**Nota:** La cobertura funcional es 3/5 debido a módulos pendientes (Reportes, Alertas), pero todo lo implementado funciona perfectamente.

### 9.3. Viabilidad para Producción

#### RECOMENDACIÓN: APTO PARA PRODUCCIÓN CON CONDICIONES

El sistema desarrollado por COREX puede desplegarse en producción para:

- Gestión completa de clientes (CRUD operativo)
- Creación y consulta de pólizas
- Filtrado básico de información
- Autenticación segura de usuarios con bcrypt
- Navegación fluida entre módulos

**Condición:** Implementar edición de pólizas antes del despliegue final para funcionalidad completa.

### 9.4. Valor del Testing Automatizado

La implementación de pruebas automatizadas por el Equipo COREX proporciona:

1. **Confianza en la calidad** mediante validación objetiva
2. **Detección temprana** de defectos antes de producción
3. **Documentación ejecutable** de comportamientos esperados
4. **Regresión continua** para prevenir bugs en futuras versiones
5. **Evidencias visuales** para stakeholders no técnicos ( 40 capturas)
6. **Reducción de costos** al automatizar validaciones repetitivas

## 10. Anexos

### 10.1. Scripts de Ejecución

```
1 # Pruebas E2E (18 casos)
2 node testing/test-automation.js
3
4 # Pruebas de integridad BD
5 node testing/db_integrity.test.js
6
7 # Smoke tests navegacion
8 node testing/ui_smoke.test.js
9
10 # Ejecutar todas
11 npm test
```

Listing 2: Ejecutar Suite Completa

## 10.2. Estructura del Proyecto de Testing

```

1 testing/
2 |-- test-automation.js           (Suite E2E principal)
3 |-- db_integrity.test.js        (Pruebas BD)
4 |-- ui_smoke.test.js            (Smoke tests)
5 |-- testlink_mapping.json       (Mapeo TestLink)
6 |-- test-evidences/             (Capturas)
7     |-- Login/
8     |-- Gestion de Clientes/
9     |-- Polizas/
10    |-- Diseno (UI-UX)/

```

## 10.3. Información del Proyecto

Campo	Información
Empresa Desarrolladora	COREX
Equipo	Equipo COREX - Testing y QA
Cliente	Seguros Fianzas VILLALOBOS
Sistema	COREX v1.0.0
Fecha de Ejecución	12 Octubre 2025
Framework Testing	Playwright + Electron
Base de Datos	SQLite (sql.js)

Cuadro 9: Información del Proyecto

## 10.4. Contacto

Para consultas sobre este reporte o el sistema, contactar al:

**Equipo COREX**

Departamento de Testing y QA

Sistema de Gestión de Pólizas COREX

## Fin del Reporte

*Desarrollado por COREX para Seguros Fianzas VILLALOBOS*

**Calidad Asegurada - 100 % de Éxito en Pruebas**