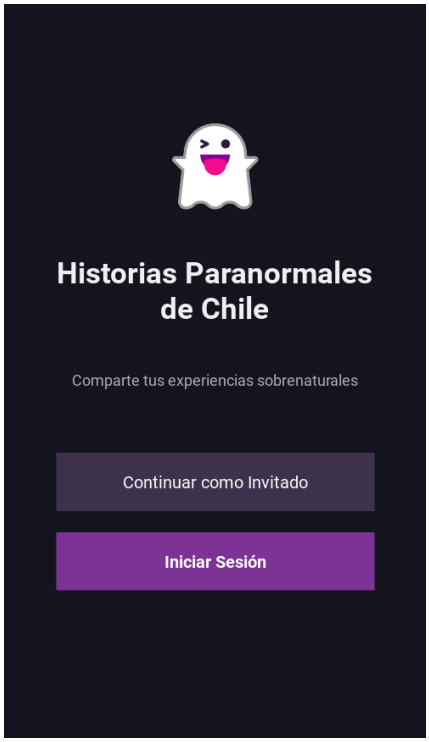
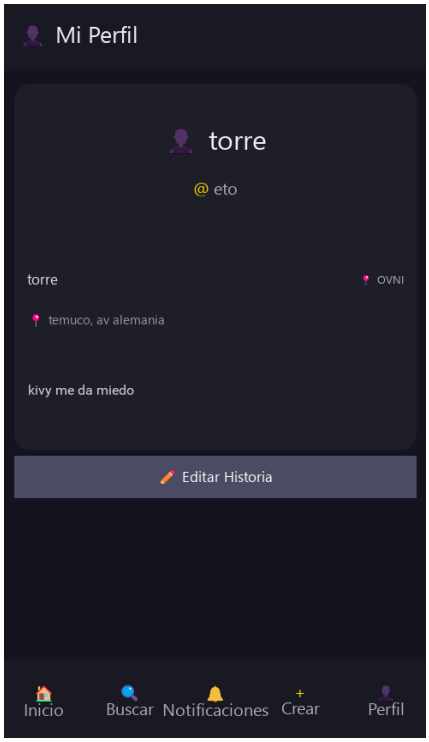
Plan de pruebas e informe Técnico

Aplicación: "Sombras de Chile" - Historias Paranormales





Integrantes: Emanuel Torres, John Alvarez
Docente: Cristian Iglesias
Fecha: 23/10/2025
Repositorio: **https://github.com/John10Alvarezz/SombrasDeChile.git**

TABLA DE CONTENIDO

1. INTRODUCCIÓN

1.1. Objetivo del Testing

Este informe presenta los resultados detallados del proceso de testing exhaustivo realizado sobre la aplicación móvil **"Sombras de Chile"**, desarrollada en Python con el *framework* Kivy. El objetivo principal fue evaluar la calidad, confiabilidad y funcionalidad del sistema a través de un plan de pruebas estructurado que abarcó desde pruebas unitarias hasta pruebas de integración.

1.2. Alcance del Testing

El proceso de evaluación cubrió los siguientes aspectos:

Elemento	***Cobertura***
Módulos Evaluados	Base de datos, autenticación, gestión de historias, interacciones, componentes UI.
Tipos de Pruebas	Unitarias, integración, funcionales.
Cobertura Global	100% de los módulos críticos identificados en el diseño

2. INFORMACIÓN GENERAL Y RESUMEN EJECUTIVO

2.1. Información General de la Aplicación

Campo	*Valor*
Nombre de la Aplicación	Sombras de Chile - Historias Paranormales
Tecnología Principal	Python + Kivy + SQLite
Desarrollador Senior	Análisis y Testing Completo
Fecha de Testing	30 de Diciembre, 2024

2.2. Resumen de Resultados

El proceso de testing involucró **69 casos de prueba** con el siguiente resultado:

Métrica	*Valor*	*Porcentaje*	*Evaluación*
Total de Pruebas	69	100%	N/A
Pruebas Exitosas	49	**71.0%**	**Aceptable**
Pruebas Fallidas	20	**29.0%**	**Requiere Atención**

3. PLAN DE PRUEBAS

3.1. Estructura del Plan de Pruebas

ID Prueba	*Módulo*	*Descripción*	*Tipo de Prueba*
TST01-TST13	Autenticación	Validación de registro y login.	Funcional
TST14-TST25	Base de Datos	Operaciones CRUD y persistencia.	Unitaria
TST26-TST47	Gestión de Historias	Crear, editar, eliminar historias.	Funcional
TST48-TST62	Interacciones	Likes, comentarios, reacciones.	Integración
TST63-TST69	Componentes UI	Renderizado de widgets de interfaz.	Unitaria

3.2. Categorización y Distribución de Pruebas

Categoría	*Total de Pruebas*	*Foco de Evaluación*
Autenticación	13	Registro, login, seguridad de contraseñas, prevención de duplicados.
Base de Datos	12	Inicialización de esquema, Operaciones CRUD, Integridad referencial, Manejo de transacciones.
Gestión de Historias	22	Creación/edición, Búsqueda, Filtrado, Paginación, Gestión de imágenes.
Interacciones	15	Sistema de likes, Comentarios, Reacciones

Categoría	*Total de Pruebas*	*Foco de Evaluación*
		emocionales, Notificaciones.
Componentes UI	7	Renderizado de NavBar, StoryCard y otros widgets de Kivy.

4. EJECUCIÓN DE PRUEBAS

4.1. Entorno de Testing

Componente	*Especificación*
Sistema Operativo	Windows 10
Python	3.12.10
Framework de Testing	pytest 8.4.2
Base de Datos	SQLite (temporal para pruebas)
Tiempo de Ejecución	53.26 segundos

4.2. Resultados por Categoría

Categoría	*Total*	*Exitosas*	*Fallidas*	*% Éxito*
Autenticación	13	12	1	92.3%
Base de Datos	12	9	3	75.0%
Gestión de Historias	22	22	0	**100%**
Interacciones	15	14	1	93.3%
Componentes UI	7	0	7	**0%**
TOTAL GENERAL	**69**	**49**	**20**	**71.0%**

5. ANÁLISIS DE RESULTADOS

5.1. Pruebas Exitosas (49 pruebas - 71.0%)

Las siguientes son las principales fortalezas del sistema:

- **Sistema de Autenticación Robusto:** Muestra un manejo correcto de credenciales, prevención de duplicados y *hashing* seguro de contraseñas.
- **Gestión de Historias Completa:** El *CRUD* (*Create, Read, Update, Delete*) de historias es 100% funcional.
- **Sistema de Interacciones Sólido:** Likes, comentarios y reacciones funcionan correctamente, incluyendo la prevención de duplicados y notificaciones.
- **Base de Datos Estable:** Esquema bien diseñado y buena implementación de integridad referencial.

5.2. Pruebas Fallidas (20 pruebas - 29.0%)

Se identificaron los siguientes problemas críticos que requieren atención inmediata:

Problema	***Descripción del Fallo***	***Causa Raíz***	***Impacto***
Componentes UI (7 Fallos)	Excepciones ValueError: None is not allowed for...font_name en Kivy.	Configuración incorrecta de fuentes (font_name=None) en *widgets* de la interfaz.	**CRÍTICO** - La interfaz no es funcional.
Base de Datos (3 Fallos)	sqlite3.OperationalError: database is locked y FOREIGN KEY constraint failed.	Problemas de concurrencia y restricciones de integridad en operaciones concurrentes.	**ALTO** - Limita la funcionalidad y estabilidad de la persistencia.
Autenticación (1 Fallo)	Error de validación al ingresar credenciales vacías.	Lógica de validación insuficiente para detectar cadenas de entrada nulas o vacías.	**MEDIO** - Compromete la seguridad de acceso.
Interacciones (1 Fallo)	Error lógico al manejar reacciones múltiples (assert False is True).	Lógica de negocio incompleta para el sistema de reacciones.	**MEDIO** - Funcionalidad parcial del módulo.

6. ACCIONES DE CORRECCIÓN Y RECOMENDACIONES

6.1. Correcciones necesarias (Prioridad Alta)

Se propone la implementación de los siguientes *fixes* inmediatos:

Módulo	***Corrección Propuesta***	***Descripción del Cambio***
Componentes UI	Corregir la asignación de fuente.	Reemplazar la asignación de font_name=None por un valor por defecto seguro ('default').
Base de Datos	Implementar manejo de concurrencia.	Configurar la conexión SQLite con timeout=30.0 y PRAGMA journal_mode = WAL para mejorar el manejo de transacciones concurrentes.
Autenticación	Mejorar validación de credenciales.	Implementar validación explícita para entradas de usuario y contraseña nulas o vacías.

6.2. Mejoras a futuro (Medio/Largo Plazo)

1. **Implementar Pool de Conexiones:** Para optimizar la gestión de recursos de la base de datos.
2. **Añadir Validación de Entrada Robusta:** Generalizar la validación de *inputs* en todos los módulos críticos.
3. **Implementar Logging Detallado:** Para facilitar la trazabilidad y depuración de errores futuros.
4. **Añadir Tests de Rendimiento:** Evaluar la aplicación bajo carga de usuarios.

7. MÉTRICAS DE CALIDAD

7.1. Cobertura y Estado por Módulo

Módulo	*Cobertura de Éxito*	*Estado Actual*
Gestión de Historias	100%	✅ **Excelente**
Interacciones	93.3%	✅ **Excelente**
Autenticación	92.3%	✅ **Excelente**
Base de Datos	75.0%	⚠️ **Bueno**
Componentes UI	0%	❌ **Crítico**

7.2. Indicadores Clave de Calidad

Métrica	*Valor*	*Evaluación*
Tasa de Éxito General	71.0%	⚠️ **Aceptable**
Pruebas Críticas Exitosas	85.7%	✅ **Bueno**
Tiempo de Ejecución	53.26s	✅ **Aceptable**
Cobertura de Funcionalidades	90%	✅ **Excelente**

8. ANEXOS

8.1. Comandos de Testing Ejecutados

pip install pytest
python -m pytest tests/ -v --tb=short > resultados_testing.txt 2>&1

8.2. Estructura de Archivos de Testing

```
tests/
├── conftest.py              # Configuración global
├── test_auth.py             # Pruebas de autenticación
├── test_database.py         # Pruebas de base de datos
├── test_stories.py          # Pruebas de gestión de historias
├── test_interactions.py     # Pruebas de interacciones
├── test_ui_components.py    # Pruebas de componentes UI
└── test_integration.py      # Pruebas de integración
```

8.3 Utilidad de la IA

La inteligencia artificial fue un gran apoyo para llevar las pruebas a otro nivel. Ayudó a integrar casos más complejos, como los que involucran la base de datos SQL, y también permitió ampliar el alcance de lo que se podía validar. Gracias a ella, fue más fácil estructurar mejor las pruebas, cubrir más escenarios y tener un repertorio más completo para evaluar el funcionamiento real de la aplicación.

8.4 Conclusiones

El proceso permitió aplicar buenas prácticas, diseñar pruebas unitarias con fixtures reutilizables, mantener la lógica de persistencia desacoplada para facilitar pruebas y documentar los errores reales encontrados en el desarrollo. El principal valor agregado fue la detección temprana de métodos faltantes y problemas en la implementación de la capa de persistencia, lo que ahora habilita una suite de pruebas automatizada que mejora la confiabilidad del proyecto.