Test Plan — Restful-Booker-Platform

Versión: 1.0

Fecha: Marzo 2026

Autor: Adriana Troche — Senior QA Engineer

Aplicación bajo prueba: https://automationintesting.online

---

1. Objetivo y Alcance

Objetivo

Validar la funcionalidad de la plataforma Restful-Booker a través de pruebas manuales y automatizadas, cubriendo los flujos críticos de negocio tanto en la interfaz web como en la API REST.

Este proyecto forma parte de un portafolio QA profesional que demuestra la aplicación de un enfoque **shift-left**: el análisis y diseño de casos precede a la automatización, y la automatización complementa — no reemplaza — el criterio de calidad.

Alcance — Dentro

Módulo	Canal	Tipo de prueba
Autenticación de administrador	API	Manual + Automatizada
Gestión de habitaciones (CRUD)	API	Manual + Automatizada
Gestión de reservas (CRUD)	API	Manual + Automatizada
Formulario de contacto	Web UI	Manual + Automatizada
Flujo de reserva (guest)	Web UI	Manual + Automatizada
Validación cruzada UI vs API	Web + API	Automatizada

Alcance — Fuera

- Panel de administración (branding, configuración avanzada)
- Pruebas de performance y carga
- Pruebas de accesibilidad
- Compatibilidad cross-browser (solo Chromium en esta versión)
- Pruebas en dispositivos móviles

---

2. Estrategia de Pruebas

Enfoque general

Se aplica un modelo de **pirámide de pruebas**: mayor cobertura en API (más rápidas, más estables) y pruebas UI focalizadas en flujos end-to-end críticos.

```
[UI E2E] ← 6 tests — flujos críticos del usuario final
[API Tests] ← 13 tests — contratos, validaciones, seguridad
[Exploración manual] ← flujos no automatizados, edge cases, UX
```

Tipos de prueba aplicados

Pruebas funcionales positivas — verifican que el sistema hace lo que debe hacer con datos válidos.

Pruebas funcionales negativas — verifican el comportamiento del sistema ante datos inválidos, campos vacíos y combinaciones incorrectas.

Pruebas de seguridad básica — validan que los endpoints protegidos rechazan requests sin autenticación (HTTP 401).

Validación de schema — verifican que la estructura de respuesta de la API cumple el contrato esperado (tipos de datos, campos requeridos).

Pruebas exploratorias manuales — sesiones de exploración libre para identificar comportamientos no cubiertos por los casos formales.

Herramientas

Propósito	Herramienta
Automatización UI + API	Playwright 1.58.2 + TypeScript
Patrón de diseño	Page Object Model (POM)
CI/CD	GitHub Actions
Reporte de resultados	Playwright HTML Report
Pruebas manuales de API	Postman
Gestión de casos y bugs	Este repositorio (docs/)

Datos de prueba

Todos los datos de prueba están centralizados en `fixtures/test-data.ts`. No se usan datos hardcodeados en los tests. Las fechas de reserva se generan dinámicamente para garantizar disponibilidad en el entorno compartido.

---

3. Análisis de Riesgos

#	Riesgo	Probabilidad	Impacto	Mitigación
R1	Entorno compartido con otros testers — datos inconsistentes	Alta	Alto	Fechas dinámicas, cleanup en `afterEach`, uso de IDs propios
R2	Diferencias entre documentación de la API y comportamiento real	Alta	Medio	Exploración previa y ajuste de tests según respuestas reales

#	Riesgo	Probabilidad	Impacto	Mitigación
R3	Inestabilidad del entorno demo (Cloudflare, timeouts)	Media	Medio	Reintentos configurados, manejo de ECONNRESET en auth tests
R4	Cambios en el DOM que rompan selectores UI	Media	Alto	Selectores semánticos y por rol sobre selectores frágiles (xpath, clases CSS)
R5	Datos de reserva que colisionen con otros usuarios	Media	Bajo	Fechas 3000+ días en el futuro para garantizar disponibilidad
R6	Falta de cobertura en flujos de administrador (UI)	Alta	Medio	Documentado en scope out — candidato para versión 2.0

---

4. Criterios de Entrada y Salida

Criterios de entrada — condiciones para iniciar pruebas

- [ ] Entorno disponible y accesible: `https://automationintesting.online`
- [ ] Credenciales de administrador válidas confirmadas
- [ ] Proyecto configurado y compilando sin errores (`tsc --noEmit`)
- [ ] Al menos una habitación disponible en el sistema (Room ID 1)

Criterios de salida — condiciones para considerar las pruebas completas

- [ ] 100% de los casos de prueba ejecutados
- [ ] 0 defectos críticos o bloqueantes abiertos
- [ ] Todos los tests automatizados pasando en CI/CD
- [ ] Reporte HTML generado y publicado
- [ ] Bugs encontrados documentados con severidad y pasos de reproducción

Criterios de suspensión

- Entorno caído por más de 2 horas consecutivas
- Más del 30% de los tests fallando por causas del entorno (no del código)

---

5. Métricas y Reportes

Métricas de cobertura

Métrica	Valor actual
Total de casos de prueba	19
Casos automatizados	19

Métrica	Valor actual
Casos manuales documentados	En progreso
Tests API	13 (68%)
Tests UI	6 (32%)
Tasa de éxito última ejecución	19/19 — 100% ■

Distribución por tipo

Tipo	Cantidad
Positivos	11
Negativos	6
Schema / Contrato	1
Seguridad básica	2
E2E (UI)	2

Reporte de resultados

- **Reporte automático:** Playwright HTML Report — generado en cada ejecución de CI/CD
- **Reporte manual:** documentado en `docs/bug-reports/`
- **Frecuencia:** en cada push y Pull Request (GitHub Actions)

Hallazgos del entorno real

Durante la implementación se identificaron diferencias entre la documentación oficial y el comportamiento real de la API:

Diferencia	Esperado	Real
Endpoint de autenticación	`POST /api/auth`	`POST /api/auth/login`
Código de error — password incorrecto	403	401
Código de error — POST sin auth	403	401
Respuesta al crear habitación	201 + objeto room	200 + `{"success":true}`
Código al eliminar recursos	202	200
GET /api/booking	Sin parámetros	Requiere `?roomid=<id>`
POST /api/booking	Sin roomid	Requiere campo `roomid`

Estos hallazgos demuestran la importancia de la exploración previa y el valor del QA como primera línea de validación del contrato API.

---

6. Entregables

- [x] Test Plan (`docs/test-plan.md`)
- [ ] Casos de prueba manuales — Web (`docs/test-cases/web-test-cases.md`)
- [ ] Casos de prueba manuales — API (`docs/test-cases/api-test-cases.md`)

- [ ] Reporte de bugs (`docs/bug-reports/`)
- [x] Suite de automatización Playwright + TypeScript (`tests/`)
- [ ] Pipeline CI/CD (`.github/workflows/playwright.yml`)
- [x] Reporte HTML de resultados

Proyecto de portafolio QA — Adriana Troche · Senior QA Engineer