Plan de pruebas

### **ID: Test\_Plan\_001\_QA\_Justiona.IO**

## Introducción

Este plan de pruebas tiene como propósito detallar el enfoque y las actividades de aseguramiento de calidad para el proyecto *Justiona.io* .

A través de este plan, se busca garantizar que todas las funcionalidades operen de acuerdo con las especificaciones y se cumplan con los estándares de calidad establecidos por ISTQB. El objetivo principal es proporcionar un sistema seguro, accesible y eficiente que facilite la adherencia al tratamiento y optimice la gestión médica.

## Objetivos

* Verificar que la aplicación cumpla con los requisitos funcionales y no funcionales.
* Identificar y corregir defectos en la aplicación antes de su despliegue.
* Asegurar la seguridad, accesibilidad, y rendimiento de la aplicación.
* Validar que las diferentes funcionalidades (registro, gestión de usuarios, turnos médicos, tratamientos y recetas, y recordatorio de medicación) operen de manera efectiva e integrada.

## Criterios de Aceptación

* **Funcionales:**
  + Cada funcionalidad debe operar conforme a los requisitos especificados.
  + Los roles de usuario deben tener los permisos correctos.
* **No Funcionales:**
  + **Seguridad:** La aplicación debe cumplir con prácticas de seguridad, como el cifrado de contraseñas y la validación de correos electrónicos.
  + **Accesibilidad:** La aplicación debe ser accesible según los estándares WCAG (Web Content Accessibility Guidelines).
  + **Rendimiento:** La aplicación debe responder en un tiempo aceptable bajo condiciones de carga normal.

## Alcance

* **Funcionalidades a probar:**
  + Ingreso y registro.
  + Gestión de usuarios.
  + Turnos médicos.
  + Tratamientos y recetas.
  + Mis notas.
  + Recordatorio de medicación.
* **Tipos de pruebas:**
  + Pruebas de API con Postman.
  + Pruebas funcionales.
  + Pruebas de integración.
  + Pruebas de accesibilidad.
  + Pruebas de aceptación.
  + Pruebas de regresión (en caso de que aplique).
  + Automatización de pruebas de front-end con Selenium (si el tiempo lo permite).
* **Pruebas no incluidas:**
  + Pruebas Mobile.
  + Pruebas de Seguridad.
  + Pruebas de Carga y Rendimiento.
  + Pruebas de Usabilidad.
  + Pruebas de base de datos.
  + Pruebas en hardware e interfaces externas.
  + Pruebas de humo.

## Estrategia de Pruebas

* **Pruebas de API**:
  + Utilizar Postman para verificar los endpoints del backend.
  + Validar respuestas, códigos de estado, y tiempos de respuesta.
* **Pruebas Funcionales**:
  + Probar cada funcionalidad en detalle según las historias de usuario.
  + Asegurar que cada rol (administrador, médico, paciente) tenga los permisos adecuados.
* **Pruebas de Integración**:
  + Verificar la correcta interacción entre módulos (e.g., gestión de usuarios y turnos médicos).
* **Pruebas de Accesibilidad**:
  + Asegurar que la aplicación cumpla con los estándares de accesibilidad.
* **Pruebas de aceptación:**
  + Asegurar que las validaciones creadas, específicamente para los formularios, sean implementadas tanto en backend como en frontend.
* **Pruebas de regresión:**
  + Ejecución de las pruebas con mayor prioridad, en caso de que se haya modificado alguna funcionalidad.
* **Automatización con Selenium**:
  + Desarrollar scripts para pruebas repetitivas y críticas.

## Recursos

**Herramientas:**

* Navegadores: Firefox, Chrome, Edge, Opera.
* Postman para pruebas de API.
* Selenium, TestNG, Maven para pruebas de automatización.
* Trello, drive y discord para el seguimiento de defectos.

**Equipo:**

* Melisa Yunis: QA Analyst.

## Calendario

**Semana 0:**

* Armado del plan de pruebas.
* Revisión y planificación de funcionalidades entre todo el equipo.

**Semana 1:**

* Diseño de historias de usuario y validaciones.

**Semana 2:**

* Diseño de set de pruebas y casos de prueba.
* Ejecución de pruebas de API, pruebas funcionales y de aceptación.
* Documentación de los resultados obtenidos.

**Semana 3:**

* Continuar con la ejecución de pruebas de API, pruebas funcionales y de aceptación.
* Inicio de pruebas de integración.
* Documentación de los resultados obtenidos.
* Reporte de defectos y seguimiento.

**Semana 4:**

* Continuar con las pruebas de integración.
* Ejecución de pruebas de accesibilidad.
* Ejecución de pruebas de regresión (en caso de ser necesario).
* Documentación de los resultados obtenidos.
* Reporte de defectos y seguimiento.
* Automatización de pruebas de front-end y ejecución de las mismas (si el tiempo lo permite).

## Casos de prueba

Los casos de prueba serán diseñados y cargados en la planilla correspondiente. En la misma se deberá completar la información del caso de prueba, su detalles y al momento de ser ejecutado, colocar allí los resultados obtenidos. La planilla tendrá el siguiente formato:

| **ID #** | **Prioridad** | **Resumen** | **Precondiciones** | **Datos de Prueba** | **Pasos** | **Rdo Esperado** | **Rdo Obtenido** | **Pass/Fail** | **Severidad** | **#Etiquetas** | **Evidencia** | **Diseñado por** | **Ejecutado por** | **Fecha de Ejecución** | **Comentarios** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## Gestión de Defectos

Los defectos encontrados durante la ejecución de los casos de prueba, serán documentados y reportados en la planilla correspondiente, llenando allí todos los campos que se soliciten.

* Todos los defectos encontrados serán registrados y gestionados en la documentación de drive y Trello.
* Los defectos serán categorizados por severidad y prioridad (según corresponda en la clasificación indicada).
* El equipo de desarrollo será notificado para la corrección de defectos críticos, dándole así prioridad a su pronta solución.

### Prioridad

La prioridad de una falla puede clasificarse como:

* **Alta**: máxima prioridad de ejecución, la funcionalidad es una prioridad.
* **Media**: prioridad de ejecución media, la funcionalidad es importante pero no esencial.
* **Baja**: prioridad de ejecución baja, la prueba se ejecutará después de completar las anteriores. Su información es requerida pero no es bloqueante.

### Severidad

La severidad de una falla puede clasificarse como:

* **Crítica**: la falla encontrada está en una funcionalidad esencial o es bloqueante.
* **Alta**: la falla encontrada está en una funcionalidad principal.
* **Media**: la falla encontrada no limita el funcionamiento de otras actividades, pero es importante.
* **Baja**: la falla encontrada no influye directamente en una funcionalidad.
* **Muy Baja**: la falla encontrada no influye en una funcionalidad, o son principalmente errores visuales.

## Entregables

* Plan de pruebas documentado.
* Set de pruebas y casos de prueba detallados.
* Validaciones e historias de usuario.
* Informe de ejecución de pruebas de accesibilidad.
* Informe de ejecución de pruebas funcionales, de integración y API.
* Reporte de defectos.
* Datos de prueba para pruebas de API.

## Riesgos

* Tiempo insuficiente para automatización de pruebas.
* Tiempo insuficiente para la ejecución de las pruebas de API, funcionales y de integración.
* Problemas de integración entre módulos.
* Recursos limitados para pruebas de accesibilidad.

## Términos

**API**: Application Program Interface.