**Plan de Pruebas de Software**

***MinutIA***

***Fecha: 15/10/2024***

**Tabla de contenido**

Historial de Versiones 3

Información del Proyecto 3

Aprobaciones 3

Resumen Ejecutivo 4

Alcance de las Pruebas 4

Elementos de Pruebas 5

Nuevas Funcionalidades a Probar 6

Funcionalidades a No Probar 6

Enfoque de Pruebas (Estrategia) 7

Criterios de Aceptación o Rechazo 8

Criterios de Aceptación o Rechazo 9

Entregables 10

Recursos 10

Requerimientos de Entornos – Hardware 10

Requerimientos de Entornos – Software 10

Planificación y Organización 11

Procedimientos para las Pruebas 11

Matriz de Responsabilidades 11

Referencias 11

Glosario 13

# Historial de Versiones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fecha** | **Versión** | **Autor** | **Organización** | **Descripción** |
| **15/10/2024** | **1.0** | **Alejandro Valenzuela** | **Capstone** | **Se levanta plan se pruebas** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Información del Proyecto

|  |  |
| --- | --- |
| Proyecto | MinutIA |
| Fecha de preparación | 24/08/2024 |
| Cliente | Duocuc |
| Líder de Proyecto | Gerson Figelist |
| Líder de Pruebas de Software | Alejandro Valenzuela |

# Aprobaciones

|  |  |
| --- | --- |
| **Nombre y Apellido** | **Fecha** |
| Gerson Figelist |  |

# Resumen Ejecutivo

El objetivo de este plan de pruebas es verificar la correcta funcionalidad, integración, seguridad y rendimiento de la aplicación que consume los endpoints proporcionados por la API desarrollada en Django. La aplicación debe interactuar adecuadamente con los servicios backend, proporcionando una experiencia de usuario confiable y eficiente.

Este plan de pruebas también se propone establecer los **criterios de aceptación**, las **estrategias de prueba** y los **casos de prueba** necesarios para asegurar que la aplicación opere sin problemas en diferentes escenarios, como en condiciones de red inestables, manejo de errores y situaciones críticas relacionadas con la seguridad y la experiencia del usuario.

# Alcance de las Pruebas

## Elementos de Pruebas

Las pruebas incluirán la validación de los siguientes componentes:

* Comunicación entre la aplicación y la API.
* Manejo de errores en los endpoints.
* Seguridad y autenticación.
* Interfaz de usuario (UI) y experiencia del usuario (UX).
* Compatibilidad en diferentes dispositivos
* Comportamiento en situaciones offline.

## Nuevas Funcionalidades a Probar

* Registro de usuario
* Ingreso de productos por foto de boleta
* Ingreso de alimento manuales
* Edición de producto
* Eliminación de producto
* Eliminación de productos masivos
* Creación de minuta
* Consultar minuta
* Desactivar minuta
* Alerta de minuta
* Cumplimiento de objetivos
* Estadísticas de uso

## Funcionalidades a No Probar

* Pruebas de rendimiento a gran escala
  + Razón: La aplicación está en su fase inicial y las pruebas de rendimiento para grandes volúmenes de usuarios o datos no son una prioridad en esta etapa.
  + Riesgo: Si la aplicación crece rápidamente, podría haber problemas de rendimiento que no se detecten a tiempo.
* Soporte para múltiples dispositivos o sistemas operativos
  + Razón: La versión actual de la aplicación está optimizada solo para un sistema operativo móvil específico (Android o iOS). Las pruebas para otros dispositivos o sistemas operativos se realizarán más adelante.
  + Riesgo: Los usuarios en otros sistemas operativos podrían experimentar problemas, pero estos se gestionarán cuando la compatibilidad esté más desarrollada.
* Soporte para múltiples idiomas (si está en desarrollo futuro)
  + Razón: La aplicación solo está disponible en un idioma (por ejemplo, español), y el soporte multilingüe está planificado para versiones futuras.
  + Riesgo: La falta de pruebas para múltiples idiomas puede impactar en la experiencia de los usuarios internacionales cuando se implemente la funcionalidad.
* Funcionalidades de accesibilidad
  + Razón: Las pruebas de accesibilidad (para usuarios con discapacidades visuales, auditivas, etc.) no son parte de la fase inicial.
  + Riesgo: Los usuarios que necesiten estas funcionalidades podrían encontrar problemas, pero no afecta la mayoría de los usuarios en la versión actual

## Enfoque de Pruebas (Estrategia)

|  |  |  |
| --- | --- | --- |
| Fase de prueba | Descripción | Herramientas/Frameworks |
| Pruebas Funcionales | Verificar que la app realiza correctamente las operaciones básicas (GET, POST, PUT, DELETE). | Postman, pytest |
| Pruebas Unitarias | Comprobar el funcionamiento de funciones o componentes individuales de la app. | Django `unittest`, pytest |
| Pruebas de Integración | Validar cómo interactúan varias partes de la app (formulario, API, bases de datos). | Django `unittest`, pytest |
| Pruebas de UI/UX | Comprobar la interfaz de usuario, diseño y experiencia. | Pruebas manuales |
| Pruebas de Seguridad | Verificar que los datos sensibles se manejan de forma segura. | Pruebas manuales, Postman |
| Pruebas Offline | Simular el comportamiento de la app cuando no hay conexión. | Pruebas manuales |

# Criterios de Aceptación o Rechazo

## Criterios de Aceptación o Rechazo

* Registro de usuario
  + Aceptación:
    - El usuario debe poder registrarse con éxito proporcionando todos los campos obligatorios (nombre, edad, Sexo, etc.).
    - Al completar el registro, el sistema debe almacenar un json con datos del usuario de forma persistente en el localstrorage del teléfono.
  + Rechazo:
    - Si no se validan correctamente los campos obligatorios
    - Si el sistema permite el registro con datos incompletos o no válidos.
    - Si no se almacena la respuesta del endpoints registro en el teléfono
* Ingreso de productos por foto de boleta
  + Aceptación:
    - El OCR debe extraer con precisión los productos de la boleta, la cantidad y la unidad de medida.
    - Los productos deben guardarse en la base de datos de la dispensa del usuario correspondiente.
  + Rechazo:
    - Si los datos extraídos son incorrectos o están incompletos (productos mal identificados o cantidades erróneas).
    - Si el sistema no guarda los productos en la dispensa.
* Ingreso manual de alimentos
  + Aceptación:
    - El usuario debe poder ingresar manualmente los alimentos con nombre, cantidad y unidad de medida.
    - Los alimentos deben guardarse correctamente en la base de datos de la dispensa.
  + Rechazo:
    - Si los alimentos no se guardan correctamente en la dispensa.
    - Si el sistema permite ingresar alimentos con datos incompletos o inválidos.
* Edición de producto
  + Aceptación:
    - El usuario debe poder modificar los detalles de un producto (nombre, cantidad, unidad de medida) correctamente.
    - El producto editado debe actualizarse en la base de datos sin errores.
  + Rechazo:
    - Si los cambios no se reflejan correctamente en la dispensa.
    - Si el sistema permite guardar productos con datos incompletos o inválidos tras la edición.
* Eliminación de producto
  + Aceptación:
    - El usuario debe poder eliminar un producto de la dispensa, y el sistema debe reflejar esta acción correctamente en la base de datos.
  + Rechazo:
    - Si el producto no se elimina correctamente.
    - Si tras la eliminación persisten datos relacionados con el producto eliminado.
* Eliminación masiva de productos
  + Aceptación:
    - El usuario debe poder seleccionar y eliminar varios productos a la vez.
    - Todos los productos seleccionados deben eliminarse de la dispensa.
  + Rechazo:
    - Si no se eliminan todos los productos seleccionados.
    - Si la eliminación de productos afecta a otros elementos no relacionados.
* Creación de minuta
  + Aceptación:
    - El usuario debe poder crear una minuta, seleccionando alimentos y definiendo el cronograma de comidas.
    - La minuta debe guardarse correctamente en la base de datos y ser consultable.
  + Rechazo:
    - Si los datos de la minuta no se guardan correctamente.
    - Si el sistema permite crear minutas con información incompleta o errónea.
* Consulta de minuta
  + Aceptación:
    - El usuario debe poder consultar una minuta previamente creada con todos sus detalles.
  + Rechazo:
    - Si la minuta no muestra los datos correctos o no puede ser consultada.
* Desactivar minuta
  + Aceptación:
    - El usuario debe poder desactivar una minuta y no debe estar activa para el seguimiento de comidas.
  + Rechazo:
    - Si la minuta sigue activa después de haber sido desactivada.
* Alerta de minuta
  + Aceptación:
    - El sistema debe enviar alertas cuando sea el momento de seguir el cronograma de la minuta.
  + Rechazo:
    - Si las alertas no se envían o se envían en momentos incorrectos.
* Cumplimiento de objetivo
  + Aceptación:
    - El sistema debe mostrar un resumen de los objetivos cumplidos (por ejemplo, cumplimiento de la minuta).
  + Rechazo:
    - Si los objetivos no se calculan correctamente o la información presentada es incorrecta.
* Estadísticas de uso
  + Aceptación:
    - El sistema debe mostrar estadísticas claras y precisas sobre el uso de la aplicación y el cumplimiento de minutas.
  + Rechazo:
    - Si las estadísticas no se generan correctamente o contienen datos incorrectos.

# Entregables

* + - Plan de pruebas
    - Casos de prueba
      * Archivo Excel que contiene el listado detallado de los casos de prueba, incluyendo:
        + ID del caso de prueba
        + Descripción del caso de prueba
        + Pasos para seguir para ejecutar la prueba
        + Datos de prueba utilizados
        + Resultado esperado
        + Resultado obtenido
        + Estado (Aprobado/Rechazado)
        + Observaciones
    - Evidencias de Pruebas

# Recursos

## Requerimientos de Entornos – Hardware

Especificaciones mínimas:

* + - Procesador: Intel Core i5 o superior
    - Memoria RAM: 8 GB o más
    - Espacio en disco: 256 GB o más
    - Sistema Operativo: Windows 10 / Linux Ubuntu
    - Resolución de pantalla: 1920x1080 o superior

## Requerimientos de Entornos – Software

* + - Entorno de desarrollo: Visual Studio Code, PyCharm, u otro editor de código a tu elección.
    - Entorno Backend:
    - Servidor Local (localhost): Tu servidor está alojado en localhost, por lo que no necesitarás configurar acceso a servidores remotos.
    - Django: La aplicación backend desplegada localmente.
  + Base de datos local: MySQL o PostgreSQL corriendo en localhost (en tu caso en el puerto 3307).
* Herramientas de pruebas: Postman o Insomnia para probar la API REST.

# Planificación y Organización

## Procedimientos para las Pruebas

Las pruebas seguirán una metodología ágil, integrándose dentro de los ciclos de desarrollo. Se realizarán de forma iterativa y continua, conforme se vayan completando las historias de usuario y cumpliendo los objetivos de cada sprint. El enfoque de las pruebas será principalmente funcional, pero también incluirá pruebas unitarias, de integración, y de usabilidad.

## Matriz de Responsabilidades

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Responsable | Informado | Aprobador |
| Gerson Figelist |  | x | x |
| Diego Meynard | x |  |  |
| Alejandro Valenzuela bovet | x |  |  |

# Referencias

Lista de todos los documentos que pueden citarse como apoyo o para ampliar el contenido del plan de pruebas. Algunos ejemplos de lo que se puede hacer referencia aquí son:

* + - Plan de Proyecto.
    - Diagrama de arquitectura
    - Diagrama de modelo
    - Documentación Endpoints
    - Casos de prueba

# 

# Glosario

* **API (Application Programming Interface):** Conjunto de protocolos y herramientas que permiten a diferentes aplicaciones comunicarse entre sí, facilitando el intercambio de datos y funcionalidades.
* **Back-End:** Parte del sistema que gestiona la lógica de negocio, bases de datos y la comunicación con el front-end, generalmente oculta para el usuario final.
* **Base de Datos:** Sistema que permite almacenar, organizar y gestionar grandes volúmenes de información estructurada, accesible por una o más aplicaciones.
* **Criterios de Aceptación:** Condiciones específicas que una funcionalidad debe cumplir para que se considere que está completa y funcionando correctamente.
* **Deploy (Despliegue):** Proceso mediante el cual se pone en producción o disponible para los usuarios finales una aplicación o sistema.
* **EndPoint:** URL o punto de acceso de una API que permite realizar peticiones específicas, como obtener datos o enviar información.
* **Front-End:** Parte visible de una aplicación, que interactúa directamente con el usuario a través de interfaces gráficas o web.
* **LangChain:** Herramienta de desarrollo para aplicaciones de IA que ayuda a conectar modelos de lenguaje con fuentes de datos, APIs y flujos de trabajo personalizados.
* **Pruebas Funcionales:** Pruebas destinadas a verificar que cada funcionalidad de la aplicación cumple con los requerimientos especificados, según los criterios de aceptación.
* **Pruebas de Integración:** Proceso de prueba que valida cómo interactúan entre sí los diferentes módulos o servicios de un sistema.
* **Pruebas Unitarias:** Pruebas diseñadas para verificar que componentes individuales del código, como funciones o métodos, funcionan correctamente de forma aislada.
* **Postman:** Herramienta utilizada para probar APIs, permitiendo realizar solicitudes HTTP y analizar las respuestas para asegurarse de que los servicios funcionan correctamente.
* **Pull Request:** Solicitud para fusionar el código nuevo o actualizado en el repositorio principal de un proyecto, tras su revisión por otros desarrolladores.
* **Regresión:** Pruebas realizadas para asegurar que un cambio o adición en el sistema no ha roto o afectado negativamente las funcionalidades existentes.
* **Sprint:** Ciclo corto de desarrollo, generalmente de 1 a 4 semanas, dentro de la metodología ágil, donde se completan tareas o funcionalidades específicas.
* **Sprint Backlog:** Lista de las tareas o funcionalidades que se deben completar en un sprint, según las prioridades definidas por el equipo.
* **Test Case (Caso de Prueba):** Conjunto de condiciones y pasos diseñados para verificar que una funcionalidad o característica de un sistema funcione como se espera.
* **Test Plan (Plan de Pruebas):** Documento que describe el enfoque, los recursos y el cronograma de las actividades de prueba necesarias para garantizar la calidad del software.
* **Test Report (Informe de Pruebas):** Documento que recoge los resultados de las pruebas realizadas, incluyendo los casos de éxito y fallo, así como los errores encontrados.
* **User Story (Historia de Usuario):** Descripción breve y simple de una funcionalidad desde la perspectiva de un usuario final, enfocada en lo que este desea lograr con el sistema.
* **Mock:** Simulación de un objeto o componente del sistema utilizado durante las pruebas para replicar comportamientos sin interactuar con los sistemas reales.
* **JSON (JavaScript Object Notation):** Formato de intercambio de datos legible por humanos y utilizado comúnmente para enviar datos entre un servidor y una aplicación.
* **Repositorio:** Espacio de almacenamiento de código fuente, generalmente gestionado a través de plataformas como GitHub o GitLab, donde los desarrolladores pueden colaborar y mantener el código del proyecto.