Desarrollo de Sistemas de información para la empresa Lavamatic la italiana

PLAN DE PRUEBAS Y

ACEPTACION DE USUARIOS

Versión: 0001

NORMA IEEE

Fecha: 16/03/2024

VERSION 01.

| *Queda prohibido cualquier tipo de explotación y, en particular, la reproducción, distribución, comunicación pública y/o transformación, total o parcial, por cualquier medio, de este documento sin el previo consentimiento expreso, protegidas es la Ley 23 de 1982, conocida como la "Ley de Derechos de Autor”. Colombiana.* |
| --- |

| **SENA** | **CENTRO DE SERVICIOS FINANCIEROS**  COORDINACIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN. | | |
| --- | --- | --- | --- |
| **Proyecto** | JSJV | | |
| **Entregable** | Manual de Usuario | | |
| **Autor** | TEAM SCRUM (Nº 1). Tecnólogo en Análisis y Desarrollo de Software | | |
| **Versión/Edición** | 0001 | **Fecha Versión** | 16/03/2024 |
| **Aprobado por** |  | **Fecha Aprobación** | DD/MM/AAAA |
| **ADSO FICHA Nº** | 2558723 | **Nº Total de Páginas** | 23 |

REGISTRO DE CAMBIOS

| **Versión** | **Causa del Cambio** | **Responsable del Cambio** | **Fecha del Cambio** |
| --- | --- | --- | --- |
| 0001 | Versión inicial | Valentina sanchez | 16/03/2024 |
| 0002 | N/A | N/A | N/A |
| 0003 | N/A | N/A | N/A |

CONTROL DE DISTRIBUCIÓN

| **Nombre y Apellidos (TEAM SCRUM)** |
| --- |
| Valentina sanchez |
| Jaidear aponte |
| Julian millan |
|  |
|  |

**ÍNDICE**

[1 INTRODUCCIÓN 4](#_heading=h.gjdgxs)

[1.1 Objeto 4](#_heading=h.30j0zll)

[1.2 Alcance 4](#_heading=h.1fob9te)

[2 CRITERIOS DE ACEPTACIÓN](#_heading=h.3znysh7) 7

[2.1 <Módulo Servicios>](#_heading=h.2et92p0) 7

2.2<Módulo Actividades> 12

2.3<Módulo Inventarios> 16

2.4<Módulo Gestión Humana>19

# INTRODUCCIÓN

El plan de pruebas pretende definir de manera clara, sin ambigüedades el alcance y objetivos específicos de las pruebas, de acuerdo con el nivel, tipo de la prueba y de las características de calidad para probar, las cuales son: facilidad de administración, seguridad, exactitud, completitud, recuperación ante fallas, consistencia, eficiencia, interfaz, integridad y resistencia.

# OBJETO

Realizar pruebas de funcionalidad de la página de <https://bae0-2800-e2-bf80-6a5-f4ad-e6eb-5c03-8f69.ngrok-free.app> con el fin de detectar fallos o inconsistencias en la aplicación.

# APLICACIÓN PARA EVALUAR

Se ha seleccionado la aplicación JSJV sobre la cual se realizarán pruebas funcionales y pruebas de usabilidad. Esta aplicación es de fácil acceso, dispone de diferentes módulos que permiten realizar pruebas de ingreso, consulta, manipulación de información, también permite realizar pruebas de usabilidad, rendimiento y seguridad

# ALCANCE

El tipo de pruebas a realizar es a nivel funcional y se validará la exactitud, completitud, consistencia, interfaz e integridad del sistema. El alcance para el proceso de pruebas funcionales comprende el Análisis que consta de la planeación, el Diseño consta del diseño de las pruebas y la ejecución consta de la ejecución, evaluación y cierre del proceso de pruebas funcionales. Se validará la funcionalidad de cada sistema y su integración con los otros sistemas

# **PLAN DE PRUEBAS**

# MATRIZ DE EJECUCIÓN

En esta matriz se muestran en detalle las fechas y horas planeadas, ejecutadas y el porcentaje de avance de las pruebas realizadas a la aplicación.

| JSJV | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **FECHA ESTIMADA** | | **TIEMPO ESTIMADO** | **FECHA REAL** | **TIEMPO** | **PORCENTAJE** |
| **MÓDULO** | **FASE** | **INICIAL** | **FINAL** | **HORAS** | **INICIAL FINAL** | **REAL EN HORAS** | **AVANCE** |
| Servicios | EJECUCIÓN | 07/03/24 | 16/03/24 | 1h 30 min | 07-03-2024 | 1h 00 min | 100% |
| Actividades | EJECUCIÓN | 12/03/2024 | 17/03/2024 | 1h 00 min | 12-03-2024 | 1h 00 min | 100% |
| Inventarios | EJECUCIÓN | 12/03/2024 | 17/03/2024 | 1h 00 min | 12-03-2024 | 1h 00 min | 100% |
| Gestión Humana | EJECUCIÓN | 12/03/2024 | 18/03/2024 | 1h 00 min | 12-03-2024 | 1h 00 min | 100% |

# **PRUEBA DE HUMO**

Al navegar por la aplicación, se está realizando una prueba inicial denominada prueba de humo o Smoke test, con el fin de identificar inconsistencias o fallas que impidan el funcionamiento básico del software.

**Tipo de Prueba:** Funcional

**Nivel de Prueba:** Pruebas de Aceptación

**INGRESO AL MÓDULO DE SERVICIOS**

**DESCRIPCIÓN**, se puede navegar a través del módulo sin problema y no presenta bloqueo o inconsistencia. El módulo funciona de acuerdo con lo esperado.

**Lista de chequeo para realizar una prueba de humo**

| **DISEÑO DE ALTO NIVEL** | **SI CUMPLE** | **NO CUMPLE** |
| --- | --- | --- |
| 1. El usuario puede acceder a la aplicación |  |  |
| 2. La funcionalidad de la aplicación está disponible y accesible |  |  |
| 3. El usuario puede realizar la navegación elemental sin que la aplicación se bloquee |  |  |
| 4. El usuario puede navegar desde la aplicación |  |  |
| 5. La funcionalidad requerida está disponible y su contenido es suficiente |  |  |
| 6. El sistema es suficientemente estable y tiene suficiente funcionalidad para pasar a la fase de prueba |  |  |

**Tipo de Prueba:** Funcional

**Nivel de Prueba:** Pruebas de Aceptación

**INGRESO AL MÓDULO DE ACTIVIDADES**

**DESCRIPCIÓN**, se puede navegar a través del módulo sin problema y no presenta bloqueo o inconsistencia. El módulo funciona de acuerdo con lo esperado.

| **DISEÑO DE ALTO NIVEL** | **SI CUMPLE** | **NO CUMPLE** |
| --- | --- | --- |
| 1. El usuario puede acceder a la aplicación |  |  |
| 2. La funcionalidad de la aplicación está disponible y accesible |  |  |
| 3. El usuario puede realizar la navegación elemental sin que la aplicación se bloquee |  |  |
| 4. El usuario puede navegar desde la aplicación |  |  |
| 5. La funcionalidad requerida está disponible y su contenido es suficiente |  |  |
| 6. El sistema es suficientemente estable y tiene suficiente funcionalidad para pasar a la fase de prueba |  |  |

**Tipo de Prueba:** Funcional

**Nivel de Prueba:** Pruebas de Aceptación

**INGRESO AL MÓDULO DE INVENTARIOS**

**DESCRIPCIÓN**, se puede navegar a través del módulo sin problema y no presenta bloqueo o inconsistencia. El módulo funciona de acuerdo con lo esperado.

| **DISEÑO DE ALTO NIVEL** | **SI CUMPLE** | **NO CUMPLE** |
| --- | --- | --- |
| 1. El usuario puede acceder a la aplicación |  |  |
| 2. La funcionalidad de la aplicación está disponible y accesible |  |  |
| 3. El usuario puede realizar la navegación elemental sin que la aplicación se bloquee |  |  |
| 4. El usuario puede navegar desde la aplicación |  |  |
| 5. La funcionalidad requerida está disponible y su contenido es suficiente |  |  |
| 6. El sistema es suficientemente estable y tiene suficiente funcionalidad para pasar a la fase de prueba |  |  |

**Tipo de Prueba:** Funcional

**Nivel de Prueba:** Pruebas de Aceptación

**INGRESO AL MÓDULO DE RECURSOS HUMANOS**

**DESCRIPCIÓN**, se puede navegar a través del módulo sin problema y no presenta bloqueo o inconsistencia. El módulo funciona de acuerdo con lo esperado.

| **DISEÑO DE ALTO NIVEL** | **SI CUMPLE** | **NO CUMPLE** |
| --- | --- | --- |
| 1. El usuario puede acceder a la aplicación |  |  |
| 2. La funcionalidad de la aplicación está disponible y accesible |  |  |
| 3. El usuario puede realizar la navegación elemental sin que la aplicación se bloquee |  |  |
| 4. El usuario puede navegar desde la aplicación |  |  |
| 5. La funcionalidad requerida está disponible y su contenido es suficiente |  |  |
| 6. El sistema es suficientemente estable y tiene suficiente funcionalidad para pasar a la fase de prueba |  |  |

# **2. CRITERIOS DE ACEPTACIÓN**

El proceso de pruebas funcionales se da por terminado una vez que:

* Se han ejecutado el 100% de los casos de prueba diseñados para este proyecto y su resultado ha sido exitoso.
* El 100% de los defectos detectados en la ejecución de pruebas han sido solucionados y se ha validado dicha solución por parte de pruebas.
* Las pruebas de creación en los procesos finalizan después de hacer una instancia y crear un objeto.
* Las pruebas de eliminación en los procesos finalizan después de hacer una instancia y eliminar un objeto.

## ***2.1 Módulo Servicios***

| **Creación de Producto** | **CP -001** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de un producto en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código en la raíz del proyecto:   pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <PaSe crea un objeto por medio de la clase ProductList donde se implementan los siguientes atributos amut, description, status y category. | |
| **Resultado esperado:**  Que se cree adecuadamente un Producto en la base de datos | |
| **Resultado obtenido:**  No se mostro ningun error al momento de crear un Producto pero nos sugiere una actualización en un documento Django que no es necesario de momento | |

| **Creación de Tipo Servicios** | **CP -002** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de tipos de servicios en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <PaSe crea un objeto por medio de la clase TipeService donde se implementan los siguientes atributos descriptionTS, value. | |
| **Resultado esperado:**  Que se cree adecuadamente un Tipo Servicio en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear un Tipo Servicio pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Creación de Orden de Trabajo** | **CP-003** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de un Órdenes de Trabajo en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <PaSe crea un objeto por medio de la clase Income donde se implementan los siguientes atributos paymentMethod, para la creación de los métodos de pago que acepta la empresa. | |
| **Resultado esperado:**  Que se cree adecuadamente una Orden de Trabajo en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear una Orden de Trabajo pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Creación de Salidas** | **CP-004** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de Salidas en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <Pa  Se crean dos objetos necesarios para la creación de una salida, principalmente se genera un producto con los siguientes datos; amount, description, status, category y posteriormente se crea una salida con el producto creado, amountOutpus y dateOutput. | |
| **Resultado esperado:**  Que se cree adecuadamente una Salida en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear una Salida pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Creación de Ingresos** | **CP-005** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de un Ingreso en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <Pa  Creamos tres objetos necesarios para la creación de un ingreso un producto, una orden de trabajo y un ingreso con los siguientes datos paymentMethod, descriptionIncome, totalIncomey la orden de trabajo donde se relaciona el producto. | |
| **Resultado esperado:**  Que se cree adecuadamente un ingreso en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear un ingreso pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Producto** | **CP-006** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Productos. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <Pa  Se crea un producto con la misma lógica que en los anteriores puntos y posteriormente se elimina usando la función delete (). | |
| **Resultado esperado:**  Que se elimine adecuadamente un Producto en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de Eliminar un Producto pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Tipo Servicio** | **CP-007** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Ingresos. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <Pa  Se crea un Tipo Servicio con la misma lógica que en los anteriores puntos y posteriormente se elimina usando la función delete(). | |
| **Resultado esperado:**  Que se elimine adecuadamente un tipo de servicio en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de Eliminar un Tipo de servicio pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación Orden de Trabajo** | **CP-008** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Orden de Trabajo. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <Pa  Se crea un Tipo Servicio con la misma lógica que en los anteriores puntos y posteriormente se elimina usando la función delete(). | |
| **Resultado esperado:**  Que se elimine adecuadamente una orden de trabajo en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar una orden de trabajo pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Salidas** | **CP-009** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Salidas. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <Pa  Se crea una salida con la misma lógica que en los anteriores puntos y posteriormente se elimina usando la función delete(). | |
| **Resultado esperado:**  Que se elimine adecuadamente una salida en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar una salida pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Ingresos** | **CP-010** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Ingresos. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings servicio/tests.py | |
| **Pasos:**  <Pa  Se crea un Ingreso con la misma lógica que en los anteriores puntos y posteriormente se elimina usando la función delete(). | |
| **Resultado esperado:**  Que se elimine adecuadamente un ingreso en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar un ingreso pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

**2.2 Módulo Actividades**

| **Creación de Estado Actividad** | **CP-010** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de un estado de actividad en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings actividades/tests.py | |
| **Pasos:**  <Pa  Se crea un objeto con el dato de statecAc siguiendo la lógica de la empresa siendo los datos Manchado, Realizada, En proceso, Pendiente. | |
| **Resultado esperado:**  Que se cree adecuadamente un estado de actividad en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear un estado de actividad pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Creación de Novedad** | **CP-011** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de una novedad en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings actividades/tests.py | |
| **Pasos:**  <Pa  Se crea una un objeto con los siguientes datos dateN y stateN siguiendo la lógica de la empresa siendo los datos Manchado, Roto, Quemado, Decolorado, Ninguno. | |
| **Resultado esperado:**  Que se cree adecuadamente una novedad en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear una novedad pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Creación de Actividad** | **CP-012** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de una Actividad en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings actividades/tests.py | |
| **Pasos:**  <Pa  Se crea siete objetos novelity, stateAc, poductlist orden\_job, machine, input, inventory y actividad donde esta última se creará con los siguientes datos; dateAc la fecha actual, timeAc, la hora actual, una descripción de la actividad, novelityAc los diferentes novedades de una actividad, stateAc con los estados que puede tener una actividad, ordenJob que referencia a una orden de trabajo, un inventario donde se crean un insumo y máquina y posteriormente el valor de la actividad. | |
| **Resultado esperado:**  Que se cree adecuadamente una actividad en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear una actividad pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Estados Actividad** | **CP-014** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Estado Actividad. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings actividades/tests.py | |
| **Pasos:**  <Pa  Siguiendo la lógica con la creación de StatecAc se crea un estado de actividad y posteriormente se elimina con delete(). | |
| **Resultado esperado:**  Que se Elimine adecuadamente un estado de actividad en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar un estado de actividad pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Novedades** | **CP-015** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Novedad Actividad. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings actividades/tests.py | |
| **Pasos:**  <Pa  Siguiendo la lógica con la creación de Novelty se crea un novedad de actividad y posteriormente se elimina con delete(). | |
| **Resultado esperado:**  Que se Elimine adecuadamente una novedad en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar una novedad pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Actividades** | **CP-016** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Actividad. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings actividades/tests.py | |
| **Pasos:**  <Pa  Siguiendo la lógica con la creación de Activity se crea una actividad y posteriormente se elimina con delete(). | |
| **Resultado esperado:**  Que se Elimine adecuadamente una Actividad en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar una Actividad pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

**2.3 Modulo Inventario**

| **Creación de Insumo** | **CP-017** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de un estado de un Insumo en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings inventario/tests.py | |
| **Pasos:**  <Pa  Se crea un objeto referenciando a un Insumo, con los siguientes datos supplier\_nameI nombre del Proveedor, amoutI la cantidad de insumo y product\_type el tipo que corresponda con un insumos de lavanderia. | |
| **Resultado esperado:**  Que se cree adecuadamente un insumo en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear un insumo pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Creación de Máquina** | **CP-018** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de una Máquina en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings inventario/tests.py | |
| **Pasos:**  <Pa  Se crea un objeto referenciando a una máquina, con los siguientes datos supplier nameM nombre de proveedor, amountM cantidad de las maquinas, machine type el tipo de máquina, description las características de la máquina. | |
| **Resultado esperado:**  Que se cree adecuadamente una máquina en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear un máquina pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Creación de Inventario** | **CP-019** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de un Inventario en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings inventario/tests.py | |
| **Pasos:**  <Pa  Se crea un objeto referenciando a un Inventario, donde se generan instancias de insumo y máquinas para crear un inventario. | |
| **Resultado esperado:**  Que se cree adecuadamente un inventario en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear un inventario pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Insumo** | **CP-020** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Insumos.. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings inventario/tests.py | |
| **Pasos:**  <Pa  Siguiendo la lógica para crear un insumo, se genera uno para eliminarlo posteriormente con el método delete(). | |
| **Resultado esperado:**  Que se elimine adecuadamente un insumo en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar un insumo pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Máquinas** | **CP-021** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Máquinas. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings inventario/tests.py | |
| **Pasos:**  <Pa  Siguiendo la lógica para crear una máquina, se genera una para eliminarlo posteriormente con el método delete(). | |
| **Resultado esperado:**  Que se Elimine adecuadamente una máquina en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar una máquina pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Inventario** | **CP-022** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Inventarios. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings inventario/tests.py | |
| **Pasos:**  <Pa  Siguiendo la lógica para crear un Invenatrio, se genera uno para eliminarlo posteriormente con el método delete(). | |
| **Resultado esperado:**  Que se Elimine adecuadamente un inventario en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar un inventario pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

**2.4 Módulo Gestion Humana**

| **Creación de Horarios** | **CP-023** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de un horario en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings gestionHumana/tests.py | |
| **Pasos:**  <Pa  Se crea un objeto que hace referencia a un horario con los siguientes campos; AdmissionDate la fecha de ese horario, starDate el inicio en horas de ese horario y finalDate en horas cuando se finalizó ese horario. | |
| **Resultado esperado:**  Que se cree adecuadamente un horario en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear un horario pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Creación de Salarios** | **CP-024** |
| --- | --- |
| **Descripción:**  Validar los campos necesarios para la creación de un Salario en el sistema operativo y que se almacenan adecuadamente. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings gestionHumana/tests.py | |
| **Pasos:**  <Pa  Se crean siete objetos los cuales son necesarios para calcular un salario los cuales son; input\_obj el insumo, machine\_obj la máquina, inventory con las instancias de input\_obj y machine\_obj, activity referente con una actividad, schedule referente a un horario y salario con los siguientes datos detailSalary el detalle del salario, descriptionAccount la descripción de pago que se realizará, numberAccount el número de cuenta, totalSalary que se le pagará al empleado, Activity referenciando a una actividad y schedule referenciando al horario. | |
| **Resultado esperado:**  Que se cree adecuadamente un salario en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear un salario pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Horarios** | **CP-025** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Horarios.. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings gestionHumana/tests.py | |
| **Pasos:**  <Pa  Siguiendo la lógica de creación de un horario, se instancia uno para posteriormente eliminarlo con la función delete(). | |
| **Resultado esperado:**  Que se Elimine adecuadamente un horario en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar un horario pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Eliminación de Salarios** | **CP-026** |
| --- | --- |
| **Descripción:**  Eliminar adecuadamente los registros en la base de datos en el apartado de Salarios. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Conectarse a la base de datos 3. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings gestionHumana/tests.py | |
| **Pasos:**  <Pa  Siguiendo la lógica de creación de un salario, se instancia uno para posteriormente eliminarlo con la función delete(). | |
| **Resultado esperado:**  Que se Elimine adecuadamente un salarios en la base de datos. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de eliminar un salarios pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Test creacion Usuarios** | **CP-027** |
| --- | --- |
| **Descripción:**  Se somete a estrés el sistema operativo para crear 100 usuarios al tiempo. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. Importar la clase usuarios y su respectivo modelo 3. Conectarse a la base de datos 4. Ejecutar el siguiente código:pytest -v --ds=JSJV.settings gestionHumana/tests.py | |
| **Pasos:**  <Pa  Por medio de un bucle for se crean una cantidad de usuarios para ver el estrés que se pueda soportar el sistema operativo. | |
| **Resultado esperado:**  Que el sistema soporta la creación de 100 usuarios al mismo tiempo | |
| **Resultado obtenido:**  No se mostró ningún error al momento de crear 100 usuarios pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Test login Credenciales válidas** | **CP-028** |
| --- | --- |
| **Descripción:**  Que se ejecuten adecuadamente las validación al momento de crear un usuario. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. De la librería pytest importar Client 3. Importar de las URLS el reverse 4. Conectarse a la base de datos 5. Ejecutar el siguiente código: pytest -v --ds=JSJV.settings gestionHumana/tests.py | |
| **Pasos:**  <Pa  Se crea un usuario para ejecutar las validaciones consulta en cuanto a correo y nombre para confirmar la existencia de dicho usuario en la base de datos. | |
| **Resultado esperado:**  El funcionamiento adecuado de las validaciones al ingresar al sistema como un usuario y se muestran adecuadamente los mensajes de Bienvenida al usuario. | |
| **Resultado obtenido:**  No se mostró ningún error al momento de ejecutar las validaciones, pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |

| **Test login Credenciales inválidas** | **CP-029** |
| --- | --- |
| **Descripción:**  Que se ejecuten adecuadamente las validaciones y mensajes al momento de crear un usuario de manera incorrecta. | |
| **Prerrequisitos**   1. Instalar la librería Pytest Django 2. De la librería pytest importar Client 3. Importar de las URLS el reverse 4. Conectarse a la base de datos 5. Ejecutar el siguiente código: pytest -v --ds=JSJV.settings gestionHumana/tests.py | |
| **Pasos:**  <PaSe crea un usuario para ejecutar las validaciones consulta en cuanto a correo y nombre para confirmar la existencia de dicho usuario en la base de datos. | |
| **Resultado esperado:**  El funcionamiento adecuado de las validaciones al ingresar al sistema como un usuario y se muestran adecuadamente los mensajes de error . | |
| **Resultado obtenido:**  No se mostró ningún error al momento de ejecutar los mensajes de error, pero nos sugiere una actualización en un documento Django que no es necesario de momento. | |