**<JSJV>**

Plan de Pruebas de Aceptación

Versión: 0100

Fecha: 08/03/2024

[Versión del Producto]

| 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 y por escrito de la Junta de Andalucía. |
| --- |

**HOJA DE CONTROL**

| **Organismo** | Consejería de Economía, Innovación y Ciencia | | |
| --- | --- | --- | --- |
| **Proyecto** | JSJV | | |
| **Entregable** | Plan de Pruebas de Aceptación | | |
| **Autor** | Lavamatic la italiana | | |
| **Versión/Edición** | 0100 | **Fecha Versión** | 08/03/2023 |
| **Aprobado por** |  | **Fecha Aprobación** | DD/MM/AAAA |
|  |  | **Nº Total de Páginas** |  |

REGISTRO DE CAMBIOS

| **Versión doc** | **Causa del Cambio** | **Responsable del Cambio** | **Fecha del Cambio** |
| --- | --- | --- | --- |
| 0100 | Versión inicial | Valentina sanchez caicedo | 08/03/2023 |
|  |  |  |  |
|  |  |  |  |

CONTROL DE DISTRIBUCIÓN

| **Nombre y Apellidos** |
| --- |
| Julian Santiago Millan |
| Jaider Steven Aponte |
| Valentina Sanchez Caicedo |
|  |
|  |

**ÍNDICE**

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

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

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

[2 PLANES DE PRUEBA 5](#_heading=h.3znysh7)

[2.1 <Modulo> 5](#_heading=h.2et92p0)

[3 ANEXOS 6](#_heading=h.tyjcwt)

[4 GLOSARIO 7](#_heading=h.3dy6vkm)

[5 BIBLIOGRAFÍA Y REFERENCIAS 8](#_heading=h.1t3h5sf)

# INTRODUCCIÓN

## Objeto

El propósito principal de este documento es establecer un conjunto exhaustivo de pruebas que los usuarios del sistema llevarán a cabo con el fin de validar la funcionalidad y desempeño del Sistema diseñado para la lavandería Lavamatic la italiana. Este conjunto de pruebas se enfocará en evaluar la efectividad y conformidad del sistema con los requisitos previamente definidos, abarcando áreas críticas como servicios, asignación de actividades, gestión de inventarios y gestión humana. La ejecución exitosa de estas pruebas será determinante para la aceptación del sistema, asegurando así su capacidad para satisfacer las necesidades operativas y estratégicas de la lavandería.

## Alcance

Este documento establece los límites de las pruebas a ser realizadas por los usuarios del sistema para validar la funcionalidad del Sistema de la Lavandería Lavamatic italiana. Las pruebas se centrarán en validar las funcionalidades de servicios, asignación de actividades, gestión de inventarios y recursos humanos. El alcance excluye funcionalidades no relacionadas con estas áreas específicas.

# PLANES DE PRUEBA

* **Pruebas Funcionales:**
  + Verificación de la correcta entrada y procesamiento de órdenes de servicios de lavandería.
  + Validación de la asignación eficiente de tareas a través del sistema.
  + Comprobación de la actualización precisa de inventarios durante el proceso de lavandería.
  + Evaluación de la gestión adecuada de información de recursos humanos.
* **Pruebas de Rendimiento:**
  + Evaluación del rendimiento del sistema al procesar múltiples solicitudes de servicios concurrentes.
  + Medición del tiempo de respuesta del sistema durante la asignación y seguimiento de actividades.
* **Pruebas de Usabilidad:**
  + Evaluación de la interfaz de usuario para garantizar la facilidad de uso y comprensión.
  + Comprobación de la navegabilidad eficiente a través de las funciones clave del sistema.
* **Pruebas de Seguridad:**
  + Validación de la seguridad en el acceso a la información de recursos humanos.
  + Verificación de la integridad de los datos de inventario y servicios.

## 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. | |

* 1. **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. | |

* 1. **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. | |

* 1. **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. | |

# ANEXOS

<Introduzca contenido y borre cuadro>



# GLOSARIO

<Insertar comentario y eliminar cuadro>



| **Término** | **Descripción** |
| --- | --- |
| Orden de trabajo | Hace referencia a las facturas generadas en la lavandería. |
|  |  |
|  |  |

# 