**CHF INVENTARIO TI**

Fecha: 18-07-2024

Por Jean Apablaza – Desarrollador J

**Descripción:** Una aplicación web que puede controlar el flujo de asignación y devolución de equipos/celulares/accesorios etc. Capaz de guardar historial y mantener un orden en de un inventario.

**Objetivo:** Desarrollar una aplicación web con base de datos para facilitar la gestión del flujo de entrega y devolución de equipos tecnológicos mientras se mantiene registro de inventario.

**Usuarios**: Este software esta destinado para uso y manejo del equipo de informática.

**Requerimientos:**

1. Mantener un registro de equipos con sus respectivas características.
2. Implementar un mantenedor (master) para los siguientes elementos:
   * Equipo
   * Persona
   * Área
   * Tipo de equipo
3. Tener visualización de una lista de personas con equipos asignados, presentada en forma de grilla (grid).
4. Añadir una interfaz para el flujo asignación y devolución de equipos del área hacia la persona para mejor manejo de inventario.
5. Incorporar una barra de búsqueda para filtrar tanto personas como equipos.
6. Incluir un campo de "observaciones" para cada equipo, así como un historial de equipo que permita registrar información relevante para futuras referencias.
7. Esta herramienta solo estará al alcance del área de TI y sistemas.

**Requisitos app:**

* Sistema operativo Windows (Se necesita ejecutar un .exe)
* SQL Server (Base de datos de personas con sus respectivos datos, nombres, apellidos, fechas de nacimiento y correo)
* Bloc de notas (o cualquier otra aplicación editora de texto para configurar el archivo config.ini)

**Requisitos no funcionales:**

Disponibilidad: La aplicación estará operativa en el momento de su ejecución. Si se cierra la consola de ejecución, se apaga el servidor, hay un corte de internet del servidor, o cualquier otra interrupción similar, la web app dejará de funcionar.

Seguridad: Acceso restringido solo al área de TI y sistemas.

Rendimiento: La aplicación debe ser capaz de manejar múltiples usuarios simultáneamente sin degradar el rendimiento.

**Arquitectura**

Se consideró como Modelo-Vista-Controlador (MVC) utilizando:

* **Lenguaje de Programación**: Python
* **Librería para Aplicaciones Web**: Flask
* **Gestor de Base de Datos**: SQL Server

**Instalación \***

**Configuración \***

**Estructura del Proyecto**

src/

├── templates/

│ ├── index.html

│ ├── ...

├── static/

│ ├── css/

│ │ ├── styles.css

│ ├── js/

│ │ ├── scripts.js

├── app.py

├── ...

config.ini

**Conexión a la Base de Datos**

* **Driver**: pyodbc
* **Configuración**: Utiliza los detalles del archivo config.ini para conectarse a SQL Server.

**Lógica de Negocio**

1. **Modelos**: Definidos en módulos Python para interactuar con SQL Server.
2. **Vistas**: Templates HTML en src/templates utilizando Jinja2 para generar HTML dinámico.
3. **Controladores**: Definidos en src/app.py y otros módulos Python para manejar solicitudes HTTP y coordinar la lógica de negocio.

**Pruebas**

* Solo pruebas operativas, de los controles de flujo y creación, eliminación y actualización de registros.

**Documentación del Código\***