Memoria Descriptiva del Sistema de Generación de Proformas

Laureano Tupac Yupanqui, Natty Jimena
Manrique Silva, Ricardo Gabriel
Yauri Vargas, Jean Franco Matias

1. Propósito del aplicativo

El sistema de generación de proformas es una aplicación de escritorio, desarrollada para apoyar la gestión operativa en empresas.

El propósito principal de esta aplicación es centralizar y automatizar la información relacionada con:

- Empleados que intervienen en los procesos de atención y ventas.
- Inventarios.
- Proveedores.
- Empresas (clientes) que contratan servicios y compran.
- Órdenes de compra y proformas generadas en las operaciones cotidianas.
- Reportes operativos y comerciales para la toma de decisiones.

El aplicativo busca reemplazar procesos manuales o dispersos por un sistema único que garantice mayor control, trazabilidad y confiabilidad de la información.

2. Características del diseño del aplicativo: arquitectura básica

El diseño de la aplicación se basa en una arquitectura por capas, separada de la siguiente manera:

a. Capa de presentación

- Implementada con Tkinter y ttk (GUI nativa de Python).
- La clase principal MainApp define una ventana con pestañas que agrupa los módulos: Empleados, Inventario, Proveedores, Empresas, Orden de Compra, Proforma y Reportes.
- Cada pestaña combina formularios de entrada con tablas para visualizar registros y facilitar las operaciones.

b. Capa de negocios o lógica

- Implementada en Python en las clases MainApp, ConnectionWindow y LoginWindow.
- Responsabilidades principales:
  - Manejo del flujo de inicio: ventana de conexión → prueba de conexión → login por código de empleado.
  - Validaciones de formato (códigos EXXXX, RUC de 11 dígitos, enteros, decimales, fechas en formato YYYY-MM-DD, etc.).
  - Restricciones de negocio, como impedir eliminar al empleado que tiene la sesión activa.

- Orquestar las operaciones sobre la base de datos mediante procedimientos almacenados.

c. Capa de acceso a datos

- Encapsulada en la clase MySQLClient (archivo db.py), que ejecuta sentencias SQL y procedimientos almacenados invocando el cliente mysql desde Python.
- El acceso se parametriza a través de un archivo config.json, con:

```
{
 "host": "localhost",
 "port": 3306,

 "user": "root",
 "password": "root",
 "database": " "
}
```

- Se utilizan métodos como run_sql, select_rows, select_scalar y call_sp para estandarizar el acceso a la base de datos.

3. Herramientas utilizadas

Las principales herramientas y tecnologías utilizadas son:

a. Lenguaje de programación

- Python 3.x para la lógica del sistema y la interfaz gráfica.

b. Bibliotecas/módulos de Python

- tkinter y tkinter.ttk: construcción de ventanas, formularios, pestañas y tablas.
- subprocess: ejecución del cliente mysql para conectarse al servidor de base de datos.
- json y os: manejo de archivos de configuración (config.json) y rutas del sistema.

c. Gestor de base de datos

- MySQL, con la base de datos, tablas normalizadas y procedimientos almacenados para operaciones críticas.

d. Entorno de desarrollo

- Visual Studio Code para la edición del código fuente en Python.

e. Control de versiones

- Git para el control de cambios.
- GitHub como repositorio remoto del proyecto.

4. Justificación conceptual del aplicativo

La aplicación está definida como un Sistema de Informacion Transaccional orientado a apoyar las operaciones diarias de la empresa en lo referente a inventario, compras, clientes y proveedores.

Los fundamentos conceptuales son:

- **Sistemas de información**: La aplicacion se encarga de registrar, procesar, almacenar y recuperar información sobre empleados, repuestos, órdenes de compra, proformas, proveedores y empresas clientes, sirviendo de soporte a las actividades operativas.
- **Modelo de datos relacional**: Se opto por una base de datos relacional con entidades como Empleado, Repuesto, Proveedor, Empresa, Orden_Compra y Proforma, además de tablas auxiliares para teléfonos, correos y direcciones. Esto proporciona integridad referencial y reduce la redundancia.
- **Normalización y descomposición lógica**: Direcciones, teléfonos y correos se almacenan en tablas específicas, permitiendo que una empresa o proveedor tenga múltiples datos de contacto, alineado con la normalización de la base de datos.
- **Encapsulamiento de reglas de negocio en la base de datos**: La utilización de procedimientos almacenados permite que operaciones como alta y baja de repuestos, registro de órdenes de compra, ajustes de stock o gestión de proformas se realicen de forma controlada y consistente, independientemente del cliente de acceso.

Básicamente la aplicación ataca la necesidad de contar con un sistema que organice y haga visible la información principal para la empresa.

5. Estructura

La estructura de la aplicación se organiza en módulos accesibles mediante pestañas en la ventana principal:

a. Módulo de conexión y login

i. ConnectionWindow

- Ventana inicial que permite configurar host, puerto, usuario, contraseña y base de datos.
- Incluye botón de “Probar conexión” y opción de “Guardar y continuar”, almacenando los datos en config.json.

ii. LoginWindow

- Solicita un código de empleado de la forma EXXXX.
- Consulta la tabla Empleado y, si encuentra el código, obtiene el nombre y permite el ingreso.
- En la barra superior del aplicativo se muestra la base de datos conectada y el empleado con sesión activa.

b. Módulo de Empleados

- Permite registrar, actualizar y eliminar empleados.
- Campos principales: código (EXXXX con prefijo fijo “E” y cuatro dígitos), nombre y teléfono.
- Se apoya en las tablas Empleado y Contacto_Empleado.
- Utiliza procedimientos almacenados como sp_Agregar_Empleado, sp_Actualizar_Empleado y sp_Eliminar_Empleado.
- Incluye validaciones básicas y la restricción de no permitir eliminar al empleado logueado.

c. Módulo de Inventario

- Gestiona el inventario de repuestos a través de la tabla Repuesto.
- Campos relevantes: Nro_Parte, Descripcion, Marca, Status, Precio_Unitario, Cantidad.
- Operaciones:
    - Crear, actualizar y eliminar repuestos.
    - Visualizar el inventario en un listado tipo tabla.
    - Cargar datos al formulario seleccionando un registro de la tabla.
- Submódulo de Ajuste de stock:
    - Permite ingresar un Nro_Parte, una cantidad y seleccionar la operación SUMA o RESTA.
    - Llama al procedimiento sp_Actualizar_Stock, que actualiza la cantidad y registra el ajuste de forma consistente.

d. Módulo de Proveedores

- Administra la información de proveedores que abastecen a empresas.
- Datos principales: RUC, Razón Social, Dirección, Teléfono y Email.
- Utiliza la tabla Proveedor y tablas asociadas Telefono_Proveedor y Email_Proveedor.
- Opera mediante procedimientos almacenados (sp_Agregar_Proveedor, sp_Actualizar_Proveedor, sp_Eliminar_Proveedor).
- Muestra una tabla con proveedores y contactos consolidados.

e. Módulo de Empresas

- Maneja los datos de empresas clientes que solicitan servicios o compran repuestos.

- Campos: RUC, Razón Social, FAX, Ciudad, Calle, Distrito, Teléfono y Correo.
- Utiliza Empresa, Direccion_Empresa, Telefono_Empresa y Correo_Empresa.
- Permite CRUD completo con selección desde tabla y carga de los datos al formulario.

f. Módulo de Orden de Compra

- Soporta el registro completo de órdenes de compra hacia proveedores.
- Datos principales:
    - Nro_Orden, Per_UM, Fecha_Entrega, Precio_Neto, Item, Cantidad, UM,
    - Forma_pago, Incoterms_2000, Desc_Orden,
    - Codigo_Empleado, RUC_Proveedor, RUC_Empresa, Nro_Parte.
- Mediante procedimientos almacenados (sp_Registrar_OrdenCompra, sp_Actualizar_OrdenCompra, sp_Eliminar_OrdenCompra) se asegura:
    - Registro de la orden.
    - Actualización correspondiente del stock de repuestos.
    - Reversión del stock cuando se elimina una orden.

g. Módulo de Proforma

- Gestiona proformas asociadas a repuestos y empleados.
- Campos: Nro_Proforma, Item, Cantidad, Fecha, Peso, Nro_Parte, Codigo_Empleado.
- Permite crear, actualizar y eliminar proformas utilizando procedimientos almacenados (sp_Agregar_Proforma, sp_Actualizar_Proforma, sp_Eliminar_Proforma).
- Facilita el análisis comercial por empleado y por repuesto.

h. Módulo de Reportes

Incluye un conjunto de reportes predefinidos, seleccionables desde un Combobox:

- Stock bajo con umbral configurable.
- Proveedores y contacto.
- Órdenes recientes (últimas 100).
- Órdenes por empresa.
- Top empresas por monto.
- Repuestos más comprados.

- Proformas por empleado.
- Proformas por repuesto.
- Empresas sin órdenes en N días.
- Proveedores sin órdenes.

El resultado se presenta en una tabla dinámica con columnas ajustadas al tipo de reporte, esto brinda una visión rápida para decisiones sobre stock, clientes prioritarios, proveedores activos/inactivos y comportamiento de la demanda de repuestos.

6. Normativas utilizadas

Para el desarrollo de la aplicación se consideró las siguientes normas y buenas prácticas:

- Buenas prácticas de programación en Python
  - Uso de clases (MySQLClient, MainApp, ventanas de conexión y login) para organizar el código.
  - Nombres descriptivos para variables, métodos y atributos, alineados con las recomendaciones de estilo de PEP 8.
  - Validaciones de datos y manejo básico de excepciones para evitar errores en tiempo de ejecución.
- Buenas prácticas de bases de datos
  - Diseño relacional con claves primarias y foráneas para garantizar la integridad referencial.
  - Separación de tablas de contacto y dirección para proveedores y empresas, reduciendo redundancia y permitiendo múltiples datos de contacto.
  - Uso de procedimientos almacenados para centralizar reglas de negocio sensibles.
- Control de versiones y documentación
  - Uso de Git para registrar cambios incrementales.
  - Repositorio en GitHub que almacena el código, scripts de base de datos y configuraciones necesarias para desplegar el sistema.

7. Objetivos del aplicativo

a. Objetivo general

- Un sistema de gestión de repuestos, órdenes de compra, proformas, proveedores, empresas y empleados para las empresas, que centralice la información y apoye las operaciones diarias dentro de las empresas.

b. Objetivos específicos

- Mantener un registro actualizado y confiable de empleados, proveedores y empresas clientes.
- Controlar el inventario de repuestos, incluyendo altas, bajas y ajustes de stock asociados a órdenes de compra.
- Gestionar el ciclo de órdenes de compra y proformas de manera estructurada, relacionando correctamente empleados, proveedores, empresas y repuestos.
- Proporcionar reportes operativos que permitan identificar stock bajo, clientes con mayor volumen de compras, repuestos más demandados y actores inactivos (empresas o proveedores).
- Reducir el uso de hojas de cálculo y registros manuales, disminuyendo errores, tiempos de búsqueda y pérdida de información.

8. Alternativas: opciones en caso de no contar con el aplicativo

De no contar con la aplicación desarrollada, algunas de las alternativas serian:

a. Uso de hojas de cálculo

- Ventajas: herramientas conocidas y de fácil acceso.
- Desventajas: falta de integridad entre archivos, riesgo alto de errores de fórmula o de versión, difícil auditoría y trazabilidad de la información.

b. Registros manuales

- Elevado riesgo de pérdida o deterioro de documentos.
- Dificultad para generar reportes o consultar históricos.
- Procesos lentos y poco eficientes para una empresa que maneja múltiples repuestos y clientes.

c. Software genérico de gestión

- Podría no ajustarse a la realidad específica de las empresas.
- Requeriría adaptar procesos a las limitaciones del software y no al revés.
- Usualmente con licencias y módulos que encarecen la solución.

d. ERP o sistema comercial especializado

- Ofrece alta funcionalidad, pero a costa de mayores costos de licenciamiento, consultoría e implementación, poco justificados para el tamaño y alcance operativo de la empresa.
- Podría incluir módulos que las empresas no necesitan, generando complejidad adicional sin valor agregado.

9. Justificación económica: comparación con alternativas

Desde un punto de vista económico:

- La aplicación fue desarrollada con software libre: Python, Tkinter y MySQL.
- No se pagan licencias por usuario ni por servidor de aplicación, lo que reduce el coste total de propiedad (TCO) para la empresa o para la institución educativa que lo utiliza como prototipo.

a. Comparacion con ERP

- Se evitarían costos iniciales de adquisición, que pueden ser altos para una pequeña o mediana empresa del rubro mantenimiento y reparación de vehículos.
- Se evitan además las cuotas recurrentes por soporte, actualización o ampliación de módulos.

b. Comparacion con hojas de cálculo o registros manuales

- Costos ocultos en tiempo de trabajo adicional, errores de inventario, pedidos mal realizados, falta de información para negociar con proveedores o priorizar clientes.
- Mayor probabilidad de decisiones subóptimas por falta de datos consolidados.

Viendo la comparación la aplicacion representa una solución económica que maximiza la relación costo/beneficio: inversión en desarrollo y capacitación inicial, a cambio de una herramienta adaptada a las necesidades específicas del negocio.

10. Justificación práctica: ¿por qué es mejor su aplicativo?

La aplicación es una mejor opción practica frente a las alternativas por los siguientes motivos:

- Está diseñado específicamente para el contexto de una empresa de mantenimiento y reparación de vehículos y maquinaria, con foco en la gestión de repuestos, proveedores, empresas y órdenes de compra.
- Centraliza en una sola interfaz funciones que de otro modo estarían dispersas (inventario, contactos, órdenes, proformas y reportes).
- Su interfaz gráfica es simple y directa, pensada para usuarios operativos sin necesidad de conocimientos avanzados en informática.
- Permite generar reportes clave sin recurrir a herramientas externas, lo que agiliza la toma de decisiones operativas y comerciales.
- Al utilizar procedimientos almacenados e integridad referencial, garantiza que el stock, los clientes, los proveedores y las órdenes estén siempre coherentes.
- Es escalable y modificable: al ser código propio, se pueden agregar nuevos campos, reportes o módulos según evolucionen los procesos de la empresa.

11. Anexos:

Enlace Github: https://github.com/FrostGZ/Sistema_de_Generacion_de_Proformas