Challenge Tech Support Analyst - Solvento
Cabrera Ábrego Raúl Andrés

Introducción

Este documento acompaña el desarrollo de un sistema de planificación de entregas para Solvento, una tienda de comercio electrónico. El objetivo principal del proyecto es demostrar la habilidad de programación para resolver problemas técnicos comunes en un entorno productivo. El script resultante gestiona la logística de asignación de viajes y camiones, integrando los procesos necesarios para la organización eficiente de las entregas diarias.

Supuestos y Justificaciones

- Lenguaje de Programación: Se eligió Python por su simplicidad y familiaridad, así como por su eficiente conexión con sistemas de bases de datos a través de bibliotecas como SQLite3, que permite ejecutar SQL directamente desde Python. Esta conexión directa demuestra cómo se realizarían las operaciones de base de datos en un entorno de producción real.
- Base de Datos: A diferencia de una simulación, esta solución implementa una conexión real a una base de datos SQLite, utilizando el módulo sqlite3 de Python. Esto permite ejecutar consultas SQL directamente, gestionando datos persistentes y demostrando operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sin usar un ORM. Esta implementación refleja un escenario más cercano a aplicaciones de producción donde se requiere manipulación directa de la base de datos.
- Versionamiento: El código se mantiene en un repositorio público basado en GIT, facilitando el control de versiones y permitiendo un seguimiento transparente del desarrollo y los cambios realizados en el proyecto. Esto es esencial para la colaboración y el mantenimiento continuo del software.
- Frontend: Aunque la solución no implementa un frontend tradicional, se utiliza la visualización directa en Jupyter Notebook para mostrar los resultados de las operaciones de base de datos. Se hacen uso de características de impresión en Python para mostrar los datos de manera clara y estructurada, lo que facilita la comprensión del flujo de datos y los resultados de las consultas.
- Modelado de Clases y Funciones: Se diseñaron funciones específicas para manejar la lógica de negocio relacionada con la asignación de camiones y la planificación de viajes. Cada función cumple roles específicos como la creación de viajes, asignación de camiones basada en la capacidad de peso y disponibilidad, demostrando una clara separación de responsabilidades y una estructura de código modular.
- Planificación de Entregas: La funcionalidad de planificación y asignación utiliza consultas SQL para interactuar con la base de datos y calcular

dinámicamente las asignaciones basadas en el peso total de los artículos de las compras vinculadas a cada viaje y la capacidad de los camiones disponibles. Esto asegura que las asignaciones de camiones cumplan con los requerimientos de carga y disponibilidad.

- Optimización: Aunque el enfoque principal es la funcionalidad, se ha prestado atención a la optimización del acceso a la base de datos y la eficiencia de las consultas SQL. El uso de índices, consultas bien estructuradas y transacciones asegura que el sistema sea no solo funcional sino también eficiente, equilibrando complejidad y rendimiento.

Metodología

El desarrollo se llevó a cabo siguiendo una metodología ágil e iterativa, comenzando con la creación de esquemas de base de datos y la definición de las tablas necesarias. Posteriormente, se desarrolló la lógica de negocio utilizando consultas SQL para manejar las operaciones en la base de datos. Cada funcionalidad implementada fue seguida por pruebas inmediatas para validar la integridad y el correcto funcionamiento antes de proceder con el siguiente bloque de funcionalidades.

Clases y Entidades

Dado que el proyecto se basa en el uso directo de una base de datos con el lenguaje SQL, las "clases" tradicionales son representadas como tablas en la base de datos, y las operaciones se manejan a través de funciones en Python que ejecutan comandos SQL:

Item

Representa un artículo en el inventario. Se almacena en la base de datos con campos para el ID del artículo, nombre, peso y precio.

Purchase

Refleja una compra realizada por un cliente. Contiene información como el ID de la compra, el nombre del cliente, la fecha de la compra, la dirección de entrega y el estado de la compra, además de estar vinculada a los artículos comprados y sus cantidades a través de una tabla de relación.

Truck

Representa un camión en la flota de entrega (en este caso, un barco). La tabla correspondiente almacena detalles como el número de matrícula del camión, el peso máximo soportado, la dirección donde está estacionado el camión, los días que trabaja y su disponibilidad.

Trip

Encapsula un viaje de entrega. Incluye el ID del viaje, los destinos (representados como códigos postales o lugares específicos), el camión asignado, la fecha planificada para el viaje y el camión asignado, gestionando todo a través de la

base de datos.

Address

Aunque no se maneja como una entidad separada en este sistema, la información de la dirección se incluye dentro de las compras y puede extenderse a una tabla dedicada si se requiere manejar múltiples direcciones por cliente.

Esta estructura garantiza que el sistema es extensible y mantiene la integridad de los datos a través de relaciones bien definidas entre las tablas, permitiendo operaciones complejas y consultas eficientes que son esenciales para la logística y la planificación de entregas.

Funciones y Simulación de Base de Datos

get_all_items()

Devuelve una lista de todos los artículos en la base de datos. Esta función realiza una consulta SQL para recuperar todos los registros de la tabla de artículos y los convierte en diccionarios para su fácil manipulación en Python.

get_all_trucks()

Recupera todos los camiones disponibles en la base de datos, incluyendo detalles como su capacidad de carga y disponibilidad. Similar a *get_all_items()*, devuelve los resultados en forma de diccionarios.

get_all_purchases()

Obtiene todas las compras registradas, mostrando detalles relevantes como el cliente, los artículos comprados y la fecha de la compra. Esta función combina datos de múltiples tablas para proporcionar una vista completa de cada compra.

get_all_trips()

Consulta todos los viajes planificados, incluyendo los códigos postales de destino, los camiones asignados y las fechas de los viajes. Ofrece una visión completa del plan de entregas programadas.

assign_trucks_to_trips(planning_date)

Esta función asigna camiones a los viajes planificados basándose en el peso total del viaje y la disponibilidad de los camiones. Se asegura de que los camiones asignados puedan manejar la carga prevista y actualiza la base de datos para reflejar las asignaciones y cambiar el estado de disponibilidad de los camiones.

print_truck_assignments()

Imprime los detalles de los viajes asignados, incluyendo el peso del viaje y la

capacidad del camión asignado. Esta función es útil para verificar las asignaciones y asegurar que todas sean adecuadas según las capacidades de los camiones.

Visualización de Datos

Utiliza pandas para crear DataFrames que resumen los componentes de los datos (artículos, compras, camiones, viajes) y los presenta en un formato tabular claro y estructurado. Esta visualización facilita la comprensión del estado actual del sistema y ayuda en la toma de decisiones operativas.

Widgets de Interacción

La función *print_truck_assignments()* actúa como una forma interactiva de visualización dentro del entorno de Jupyter Notebook, proporcionando detalles en tiempo real sobre los viajes planificados y los camiones asignados.