Proyecto de Bases de Datos I

Integrantes

1. Luis Fernando Molina Peraza
2. Paúl Andrés Rodríguez García

Elaboración

- Segundo semestre del 2022

Detalles

Pasos para inicializar el servidor

- Debe crearse la base de datos (ojalá con el nombre `Proyecto`, ya que algunos *scripts* están configurados para funcionar con ese nombre) y ejecutarse el *script* `CrearTablas.sql`.
- A partir de esto, hay que crear todos los SP y los *triggers* mediante los archivos que se proporcionan (algunos son para consultas, pero otros son necesarios para el procesamiento de los archivos XML). Muchos SP necesitan un nombre de usuario (normalmente es la variable `@inUsername`) para saber quién está tratando de utilizar ese SP (en la mayoría de los casos debe ser un administrador).
- Luego, se puede ejecutar la lectura de `Catalogo.xml` y `Operaciones.xml` con los scripts `LeerCatalogo.sql` y `LeerOperaciones.sql`, respectivamente. (Obviamente, hay que cambiar la ruta de los archivos para que coincidan con su ubicación).
- Aquí se pueden ejecutar los SP que crean los arreglos de pago, antes de leer `OperacionesSegundo.xml`.
- En este punto, se puede volver a correr `LeerOperaciones.sql` para leer el segundo archivo, pero hay que cambiar el nombre del archivo a `OperacionesSegundo.xml`, para que el archivo leído sea el segundo XML.

Cómo inicializar la página web

- Hay que instalar las siguientes bibliotecas de Python: `pyodbc` (para conectarse al servidor de la base de datos), `flask` (para crear el servidor web).
    - Al inicio del archivo logica.py, intenta conectarse primero a una computadora (remota, en un `try`) y luego a otra (local, en un `catch`), pero puede dejarse fijo para que se conecte a un solo servidor de base de datos.
    - Hay que reemplazar los valores de las variables que están ahí para que coincidan con la configuración del servidor y que pueda conectarse a él (por ejemplo, el nombre del servidor, el puerto, etc.).

Funciones no implementadas

- Se creó la tabla ErroresDefinidos para que cada código de error (por ejemplo, 50000 o 50001, etc.) tuviera su descripción en esa tabla. De esta forma, no se retornaría solo el có-

digo de salida, sino también una descripción. Así se evita en capa lógica asignarle la descripción a un código de error, ya que podría quedar inconsistente con la base de datos. Por ejemplo:

ID	resultCode	errorInfo
1	0	OK
2	50000	Error desconocido
3	50001	Credenciales inválidas
...	...	...

- Que se cancele el arreglo de pago al terminar de pagarlo o al dejar de pagar las facturas que contienen cuotas del arreglo.

Cuándo se cae el proyecto

- Algunos errores no son manejados adecuadamente desde la interfaz, entonces puede que después de recibir la respuesta del servidor se quede mostrando *Cargando…*, ya que dio un error al tratar de interpretar la respuesta del servidor. Lo correcto sería mostrar una alerta e indicar cuál fue el error. (La tabla `ErroresDefinidos` habría sido útil para esto, ya que la capa lógica recibiría el código de salida y el nombre del error, en lugar de solo recibir el código, y así se muestra directamente en la interfaz dicho nombre). Sin embargo, si la consulta no retornó un error, generalmente no hay problemas en la interfaz.
- Si se utiliza un archivo con multiples años, el sistema de generación de facturas generará resultados inconsistentes, pues nunca fue validado el año de ejecución (solamente los meses). Por lo tanto, el `15/1/2023` es considerado menor que `15/12/2022`.

Tablas con *trigger*

- `Factura`:
  - `AplicarAP`: Cuando a una factura se le asocia un pago, revisa si contiene un detalle de arreglo de pago. En ese caso, intenta aplicar el arreglo de pago (cambiando el estado de las facturas de **Pagado con arreglo de pago** a **Pagado normal**).
  - `OrdenCortaPagada`: Al pagar una factura, se encarga de revisar si ya se pagaron todas las facturas de esa propiedad. En ese caso, cambia el estado de la orden de corta a **Pago hecho** y genera la orden de reconexión.
- `Propiedad`:
  - `EventoDePropiedad`: Crea una entrada en la tabla de eventos al insertar, eliminar o actualizar una entrada en la tabla. Como el *trigger* no tiene acceso al usuario o a la IP, los deja en `NULL`, así que después de la inserción/actualización/eliminación, se hace un `UPDATE` en la tabla de eventos para reemplazar esos nulos por los valores verdaderos.
  - `CobrosPropiedad`: Asigna los detalles de una propiedad al ser creada.
  - `ActualizarCobros`: Similar a `CobrosPropiedad`, pero funciona al cambiar el tipo de uso o el tipo de zona de una propiedad. (Por ejemplo, le quita a la propiedad un concepto de cobro que no corresponde a su nuevo tipo de uso o tipo de zona).