PAGO ELECTRÓNICO

ESTRATEGIA

Nombre de grupo: OOZMA_KAPPA

Integrantes:

Gino Ianuzzi (Responsable) 147.375-0

Camila García Santillán 147.570-8

María Florencia Maldonado 144.730-0

Guillermo Santillán 143.784-7

Número de grupo:

Curso: K3013

Índice

ESTRATEGIA

Para comenzar, el proyecto fue subdivido en 6 proyectos:

1. **Clases:**

En este proyecto se encuentran todas las clases del sistema. Cada clase se asemeja con cada tabla de la base. Cabe destacar que el manejo de estas clases fue realizado a conveniencia, donde pueden haber diferencias entre la clase y su entidad en la Base de Datos, como por ejemplo en la mayoría de los casos, en campos que son claves foráneas (relaciones con otras tablas), en la clase, directamente, el atributo pasa a ser la entidad relacionada en sí. Además, creamos una clase Base que posee todos los métodos comunes (los llamados a la base) a todas las clases como por ejemplo el guardar, eliminar, deshabilitar e insertar.

2. **Conexión:**

En este proyecto se encuentra una clase llamada SQLHelper, el cual se encarga de realizar todas las acciones que tengan que ver con la BD, parseando los parámetros, y dando a quien programe la aplicación una interfaz más amigable, evitando sentencias poco declarativas de la librería encargada de esta acción. Es, como bien lo dice su nombre, un helper.

3. **Excepciones:**

Este proyecto contiene todos los tipos de excepciones que entiende y maneja la aplicación, ellas son:

- La entidad buscada no se encuentra.
- No hay datos encontrados.
- ErrorConsulta: Excepción para cualquier tipo de storedprocedure o instrucción SQL que falle.
- Entidad existente.
- Badinsert: excepción para fallo de inserts.

4. **Pago Electrónico:**

En este proyecto se encuentra la UI del sistema. Contiene todos los ABM's, estadísticas, Inicio de sesión, y demás funcionalidades del sistema que el usuario final podrá manejar.

5. **Utilidades:**

En este proyecto creamos algunas clases que nos son de gran utilidad en el proyecto:

- Encryptor: se encarga de encriptar texto en el algoritmo SHA256

- Validator: Es la clase encargada de cualquier tipo de validación de campos ingresados por el usuario: nulidad del campo, si es numérico, si es decimal, etc.
- Manejador de combos.
- Manejador de diálogos: Crea un formulario nuevo con un solo campo de texto a completar. Sirve, por ejemplo, para el cambio de clave.

6. Log

En este proyecto se encuentra la clase FSLogger encargada de registrar los login exitosos o no, de los usuarios al ingresar a la aplicación. Consideramos que el Login Auditoria se registra en un archivo de texto plano, el cual se crea en el directorio:

PagoElectronico\PagoElectronico\bin\Debug

MODELO DE APLICACIÓN

La aplicación se divide en tres capas: Conexión, clases, interfaz del usuario. El usuario accederá visiblemente a la interfaz. Esta misma, se comunica con las clases, las cuales se encargarán, a su vez, de comunicarse con la capa de conexión, para acceder a las tablas de datos de la base y obtener/modificar/insertar datos.

Consideraciones y decisiones generales

1. CREACIÓN DE TABLAS Y MIGRACIÓN DE DATOS:

- Definimos DNI como username para los usuarios.
- Definimos la clave user (encriptado bajo el algoritmo SHA256) como default para todos los usuarios (salvo los administradores con clave w23e). La pregunta secreta es ¿Cuál es tu color preferido? Y la respuesta para todos es azul(encriptado bajo el algoritmo SHA256)
- Los IDs de las tablas son autonuméricos
- Definimos que, como no hay un ABM de funcionalidades en la aplicación, las decidimos nosotros y las aplicamos mediante un enum en la clase Funcionalidades
- Consideramos distinto desactivar que modificar, ya que modificar implica que una entidad puede volver a activarse.
- En algunas entidades pusimos campo Eliminado y campo Activo/Habilitado, donde el eliminado representa la eliminación lógica. En la consigna del trabajo pedía que todas las bajas realizadas fueran lógicas.
- Consideramos que en un principio todas las entidades se encuentran activas. Las cuentas no poseen transacciones a facturar.
- Para las cuentas definimos que todas las existentes en la tabla maestra eran gratuitas y les asignamos a todas el mismo saldo.
- Para el Tipo Cuenta agregamos atributo días de vigencia para establecer cuánto dura la suscripción a un determinado tipo. Por lo que al crear o modificar una cuenta, la fecha de cierre será la de creación/modificación más los días de vigencia.
  Además Supusimos que el tipo de cuenta tiene 3 costos asociados: por Apertura Cuenta, por Modificación y comisión por Transferencia.
- En Cuenta consideramos que la fecha apertura es desde el primer día en que se creó la cuenta. La fecha cierre corresponde a la fecha actual (la del sistema) más los días de vigencia del tipo cuenta.
- Todas las fechas utilizadas en la aplicación son sacadas del archivo de configuración.
- Para el tipo documento la tabla maestra solo contenía “Pasaporte”. Nosotros agregamos otros tipo documento como dni, LC y LE.
- Suponemos que todas las facturas que hay en la tabla maestra ya están facturadas, que se utilizará en el archivo de configuración una fecha mayor a las que ellas contienen.

2. CONSULTAS

Para realizar las consultas, utilizamos la libreriaSQLClient que nos provee System.Data de C#. Esta librería nos permite ejecutar tanto llamados a storedprocedures desde el código, como así también querys escritas en el código mismo.

Decidimos implementar storedprocedures y llamar a estas desde el código.

Los storedprocedures los fuimos desarrollando a medida que se iban necesitando con el desarrollo de la aplicación. Como las storedprocedures (SP) siempre son "SELECT", "INSERT", "UPDATE" o "DELETE", decidimos darle un nombre general default a cada una según son usadas:

- Para las SP que usan SELECT, su nombre será "traerListado" + Nombre de la entidad + Condiciones
- Para las SP que usan INSERT, su nombre será "insert" + Nombre de la entidad + Condiciones
- Para las SP que usan UPDATE, su nombre será "update" + Nombre de la entidad + Condiciones
- Para las SP que usan DELETE, su nombre será "delete" + Nombre de la entidad + Condiciones

A su vez, creamos otra abstracción para las desactivaciones y eliminaciones lógicas, en las que, si bien se ejecutan updates, les definimos a su nombre: *"deshabilitar" + Nombre de la Entidad / "delete" + Nombre de la Entidad respectivamente.*

Esto nos permitió, en nuestra clase Base definida previamente, abstraernos y usar para todas las clases las mismas llamadas a los SP, según corresponda.

En cuanto a cómo ejecutamos estos SP desde el código, gracias a la librería, logramos ejecutar los SP que realizan inserts/updates/deletes mediante el comando"ExecuteNonQuery", y las que realizan selects (es decir, solo traen resultados para leer), mediante el comando "ExecuteReader"

Índices

- Decidimos que no agregaríamos nuevos índices dado que bastaba con los generados por la constraint IDENTITY que usamos en la creación de las tablas.

3. SOBRE LA APLICACIÓN:

1. Registro de Usuario:

Para Loguearse en la aplicación, un usuario debe ingresar Username, password y rol asignado. Si el usuario posee un solo rol, ingresará directamente, caso contrario deberá elegir con qué rol desea ingresar.

El que quiera registrarse deberá entonces ingresar username, password y elegir (en un comboBox) un rol. Según el rol seleccionado y sus funcionalidades asociadas, las pestañas que aparecerán en la pantalla principal.

2. ABM de Clientes, Cuenta, Rol

Estas ABMs se comportan de manera similar. En un principio se muestra una grilla con los Clientes/Cuentas/Roles existentes, dando la posibilidad al usuario de elegir entre todos los campos uno de ellos para modificar, dar de baja o deshabilitar. También posee un botón para crear una nueva instancia de estas entidades.

Luego ya sea para modificar o crear nuevo se reutilizaron los formularios de creación/modificación de cada entidad.

En todos los ABMs decidimos poner los botones fuera de las grillas por una cuestión de interfaz gráfica ya que consideramos que es más simple y elegante que esté así.

3. Modificación Tipo Cuenta

Consideramos que para modificar el tipo cuenta se debe verificar que la cuenta no posee suscripciones pendientes de pago, ni comisiones por apertura/modificación del tipo cuenta anterior. Cuando se crea una cuenta o modifica el tipo cuenta, se agregan una transacción pendiente por cada día de vigencia del tipo cuenta. El costo de cada suscripción es el costo de apertura/modificación dividido en los días de vigencia.

4. Depósitos/Retiros/Transferencias

Al igual que antes poseen el mismo formato. Cada uno muestra los campos necesarios para realizar la transacción y el textbox importe el cual se verifica que sea del formato correcto (no nulo, decimal, mayor a cero, etc) para evitar inconsistencias al momento de enviar los datos a la base de datos. Antes de realizar un deposito/retiro/transferencias se verifica en la aplicación que la cuenta posee saldo disponible.

Una vez que se realiza alguna de estas transacciones se ejecuta en la base un trigger que actualiza los saldos de las cuentas. Además se lanza otro trigger (exceptuando a Retiros) que inserta estas transacciones en la tabla “Transacciones pendientes”, para llevar un registro de las cosas que aún no se han facturado.

5. Facturación

Para este requerimiento decidimos hacer un form que muestre una grilla para las transacciones pendientes de transferencias y depósitos, y otra grilla que muestre los cambios y aperturas de cuenta. Además si el cliente tiene cuentas que tengan suscripciones pendientes de pago, se

mostrara un groupbox donde podrá seleccionar una de esas cuentas y se mostrará la cantidad de suscripciones pendientes asociadas a esa cuenta. Luego el usuario ingresará la cantidad de suscripciones que desea facturar. (Consideramos que al crear o modificar una cuenta, se crea una suscripción por cada día de vigencia que tenga la Cuenta acorde a su tipo cuenta. El usuario podrá elegir X cantidad de suscripciones).

Una vez añadidas la cantidad de suscripciones a pagar se mostrará un form con los datos asociados a la factura, agrupando los ítems factura correspondientes, sus cantidades y subtotales.

Se considera que al facturar un cliente se le facturaran todas las comisiones por transferencias, depósitos (con costo cero, pero en la consigna especificaba que debía emitirse un comprobante), y modificaciones del tipo cuenta. Limpiando todas las transacciones pendientes asociadas al cliente, y solo x cantidad de suscripciones a tipo cuenta.

Una vez generada la Factura se insertarán los correspondientes ítems en la tabla ítems factura. Uno por cada tipo de transacción, haciendo una sumatoria de cantidades y precios por cada uno.

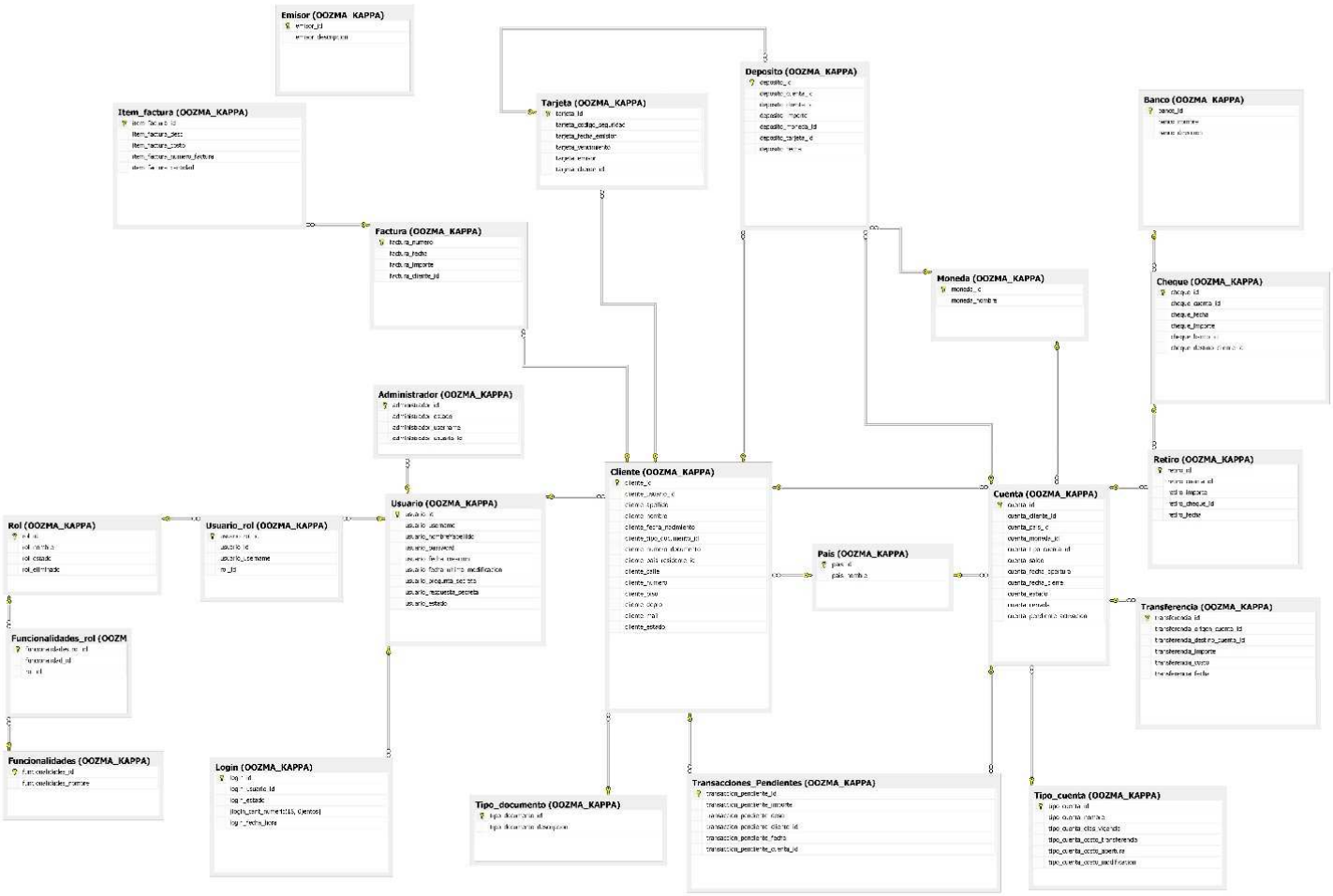
6. Listado Estadístico

Para este requerimiento decidimos hacer un mismo form donde mediante un combo box se pueda seleccionar el listado estadístico que se quiera consultar.

Nosotros decidimos usar un combo box para que el usuario seleccione el trimestre, ya que de esta manera predeterminamos los 4 trimestres del año y sus fechas de inicio y fin. Así se nos facilitó el trabajo a la hora de realizar las consultas a la base de datos.

Como cada listado es diferente decidimos que de acuerdo al que se le elija, se configurará una grilla distinta con las columnas correspondientes para el listado que se muestre.

DER (Diagrama Entidad Relación)



Nota: Para verlo ampliado, por favor, abrir imagen "DER OOZMA_KAPPA.jpg" en el directorio raíz