ESTRATEGIA

ESTRATEGIA TRABAJO PRACTICO GESTION DE DATOS 2C 2019

Abstract

EN EL PRESENTE INFORME SE DETALLARÁ LAS ESTRATEGIAS, TECNOLOGIAS Y DESICIONES DE DISEÑO TOMADAS PARA LA REALIZACION DEL TRABAJO PRACTICO DE LA MATERIA GESTION DE DATOS

Ivan Casanova - Tomas Behringer – Deborah Pergamenik – Lautaro Hudson

Email Responsible: ivanrcasanova98@gmail.com

Indice

INTRODUCCIÓN

FRBA OFERTAS es una aplicación desktop cuyo propósito es la administración, compra y venta de ofertas provistas por proveedores. Para lograr dicho propósito, se implementan ciertas funcionalidades que les permiten a los usuarios poder utilizar la aplicación. Se implementa un sistema de Roles y permisos que además de proporcionar funcionalidad, brinda seguridad a la hora de usar la aplicación. Los usuarios se dividen principalmente en tres tipos de roles: los clientes, cuyo propósito es la compra de Ofertas y la carga de créditos. Los proveedores, los cuales crearan ofertas y canjear las ofertas de los clientes. Finalmente, el administrativo será el responsable de manejar las ABMs (o CRUD) de los clientes, proveedores y roles, la creación de ofertas, la facturación de los proveedores y contar con un listado estadístico de distintos parámetros.

Si bien por el momento solo existen tres tipos de usuarios el diseño del sistema permite la posibilidad de que en un futuro puedan agregarse nuevos roles con facilidad.

Las funcionalidades son los parámetros que permiten la creación de roles. Cada uno le permite al usuario poder ver y usar las herramientas que provee el programa. Se tomo precaución por cuestiones de seguridad de que un usuario solo pueda ver las funcionalidades que tiene asignadas y no todas las que hay en el sistema.

Por último, un usuario nuevo que ingrese al sistema podrá registrarse con los roles que tengan dicha funcionalidad. Por el momento solo la información personal de clientes y proveedores son persistidos.

Se implementaron distintos patrones de diseño y herramientas de SQL para poder lograr los requerimientos. Los mismos serán detallados más adelante en el documentó.

TECNOLOGIAS UTILIZADAS

Para la realización del presente trabajo practico, se utilizaron las herramientas:

- Microsoft Visual C# 2012 Express Edition (para la programación de la Aplicación Desktop)
- Microsoft SQL Server 2012 Express (para el manejo de la persistencia vía SQL)

COMPONENTE BASE DE DATOS/SQL SERVER

En esta sección explicaremos la parte de la base de datos relacional del trabajo practico. Esto se compondrá de:

- Proceso de migración
- Stored Procedures
- Triggers
- Functions

Cada uno de estos componentes será explicado en detalle y se justificaran ciertas decisiones de negocio a la hora de migrar a nuestro nuevo esquema.

PROCESO DE MIGRACION

Consiste en pasar toda la información de una base de datos anterior a nuestro nuevo diseño, para poder usar dichos datos en la aplicación. Para eso se utilizó una tabla maestra presentada por la catedra que contenía toda la información que debía ser migrada para el trabajo practico. Usaremos el término “Legacy” para aquella información perteneciente a la tabla maestra original.

El primer desafío fue la normalización de la tabla maestra, ya que muchos de los datos estaban inconexos o repetidos. Fue nuestra responsabilidad poder pasarla al diseño implementado por el grupo y que este tenga sentido.

Primero se crearon las tablas del diseño planteado en el DER con las Primary keys y Foreing keys indicadas. Luego se crearon las funciones y un usuario administrativo general para poder acceder al sistema libremente. Posterior a esto, se comenzó un proceso de limpieza de la tabla maestra. Esto consistía en buscar aquellos atributos que debían ser UNIQUE (dni e email para clientes, razón social y cuit para proveedores, etc.) y comenzar a limpiarlos. Esto se realizó mediante tablas temporales. Finalmente se comenzó el proceso de inserción de los datos legacy en el nuevo diseño.

Iremos tabla por tabla explicando como fue el proceso de migración y normalización para cada una. Para un mejor seguimiento, se recomienda revisar el DER al final del informe o en su versión imagen en el directorio donde se encuentre este documento.

Tabla Usuario

Uno de los principales inconvenientes de la tabla maestra es que no trabaja con usuarios. Es decir, había información de clientes y proveedores, pero la lógica de usuario depende de nosotros. La estrategia que utilizamos fue la siguiente: en caso de ser Cliente, se le asigna un usuario cuyo Username y Password sean su DNI. Esto permite que no haya usuarios repetidos. Para los proveedores se utilizó su CUIT el cual también es de carácter único. Los demás valores son los predeterminados de un usuario nuevo (habilitado, no bloqueado y con cantidad de intentos de login igual a cero). Dependiendo de la información que se insertase un usuario se lo vincula a un rol especifico.

Tabla Rol Por Usuario

A medida que se van insertando Clientes y Proveedores los usuarios generados reciben una entrada a esta tabla adquiriendo el Rol en cuestión para poder usar la Aplicación dentro del sistema.

Tabla Rol

Se cargaron los roles con los que debe empezar el sistema (cliente, proveedor, administrativo).

Tabla Función Por Rol

Se setearon las funciones predeterminados para cada uno de los roles predeterminados (cliente, proveedor, administrativo).

Tabla Función

Se cargaron las funciones predeterminadas dadas por el enunciado.

Tabla Cliente

Como mencionamos antes se utilizó un procedure que limpia las inconsistencias en la tabla maestra mediante el uso de tablas temporales y funciones provistas por SQL server (row_number()). La limpieza de la tabla permite que no puedan existir Clientes gemelos, es decir clientes con atributos únicos iguales. Posterior a esto se insertaron los datos referidos a clientes de la tabla maestra en nuestra Tabla Cliente respetando de no insertar varias veces el mismo DNI o email (uso de Distinct). Con respecto a los datos vacíos (dirección, piso, dpto), la aplicación proporcionara la posibilidad de que un Cliente pueda actualizar sus datos cuando lo vea necesario.

Tabla Tipo Pago

Como la tabla maestra la información referida tipos de pago en “crédito” o “débito”, se insertaron 2 filas con dichos parámetros.

Tabla Credito

Previo a los créditos se crea un trigger llamado “updatear_monto_por_carga” que por cada INSERT en la tabla Crédito se iría actualizando el monto de los Clientes correspondientes. Tomamos las filas de la tabla maestra en las cuales el monto no fuese nulo. Notamos que solo un cliente (llamado Marga Suarez) realizo todas las cargas de créditos en el sistema anterior, acumulando una cantidad descomunal de dinero. Debido a que no está especificado si el monto de cada cliente debe mantenerse o no después del proceso de migración decidimos que dicho usuario pueda mantener su monto en el nuevo diseño. Aquellos que no tengan ninguna carga empiezan en el sistema con el monto base de 200 pesos.

Tabla Rubro

Se insertan los distintos rubros que aparecen en la tabla maestra se usa un DISTINCT para no repetir.

Tabla Proveedor

Se utiliza un proceso similar al de tabla cliente. Mediante el uso de tablas temporales se filtran aquellos posibles proveedores con Razones sociales iguales o CUIT iguales. Tanto CUIT como RS serán de carácter unique, es decir se valida que provedorA y provedorB no tengan campos CUIT ni RS iguales. Finalmente se inserta en la nueva tabla y se crean los usuarios correspondientes a cada proveedor utilizando su CUIT como username y password. Con respecto a los datos vacíos (dirección, piso, dpto, etc.), la aplicación proporcionara la posibilidad de que un Proveedor pueda actualizar sus datos cuando lo vea necesario.

Tabla Ofertas

Se cargan las ofertas utilizando como distinción de cada una su código de oferta. Es decir, si aparecen dos ofertas con el mismo código, solo se insertará una en la nueva tabla. También se decidió poner como cantidad máxima por usuario, el stock disponible de la oferta. Si en la tabla maestra la Oferta_Cantidad era 5, un usuario podrá comprar hasta 5 unidades a la vez de dicho producto.

Tabla Compra

Se cargan todas las compras que aparecen en la tabla maestra. Debido a que no lo especifica asumimos que se compra una sola unidad de la oferta que se está comprando. Se las vincula mediante foreing keys a al cliente que aparezca en la tabla maestra y a la oferta misma. Una vez insertadas las compras se realizó un update para que el stock de las ofertas fuera consistente.

Tabla Cupones

Para la creación de cupones se decidió crear un cupón por cada compra realizada. Por ende, el número de compras y cupones es el mismo al migrar la tabla. Se decidió que el cupón tiene un código propio compuesto del código de la oferta más el valor de la primary key de la compra a la que pertenece el cupón pasado a base36. Ejemplo: si un cliente compra la Oferta de código 'AHRWT123' y es la compra número 36 en el sistema el código del cupón será: AHRWT12310. La decisión de usar base36 fue para acortar la longitud de los códigos a medida que se van insertando más compras en el sistema. Además, aseguran unicidad para poder distinguir compras de ofertas iguales realizadas por un mismo cliente. La fecha de vencimiento del cupón será 1 mes después de la fecha en la que se realizó la compra.

Tabla Factura

Se insertan las facturas ya realizadas en el sistema anterior usando como clave unique el número de oferta. Posterior a esto, las compras que fueron el campo Compra_Facturada de las compras que fueron facturadas se lo paso 1 para que no vuelva a facturarse cuando se utilice la aplicación.

Tabla Item Factura

Usando la tabla anterior y la tabla maestra se fueron insertando todas las foreing keys de las compras que pertenecían a una oferta.

Stored Procedures

Para cada una de las migraciones explicadas anteriormente se invento un stored procedure con el nombre 'pr_cargar_nombretabla' la funcionalidad de estos es la misma que la detallada en la sección de migración.

Otros procedures usados son:

- pr_bajaLogica: este stored procedure se repite 3 veces uno para cada abm (rol, cliente y proveedor) lo que hace es deshabilitar mediante un campo de tipo bit la posibilidad de utilizar un rol o usuario.
- pr_resetear_cant_login_fallido: se encarga de poner en 0 el contador de login fallidos cuando usuario entra con éxito a su cuenta.
- pr_borrar_relaciones_de_un_rol_x_usuario: se ejecuta cuando un rol es dado de baja un rol y se deben eliminar todas las entradas de dicho rol de la tabla Rol por usuario.

Triggers

- trigger_finalizar_oferta: se activa cuando el stock de una oferta llega a 0 realizando una baja lógica en la misma. La decisión de utilizar una baja lógica es que en algún momento se podría implementar la capacidad de reabastecer stock y la misma se pueda dar de alta.
- Trigger_compra_facturada: se activa cuando un administrador factura las ventas de un proveedor permitiendo que no se facturen 2 veces la misma compra.
- tr_bloquear_usuario: trigger que bloquea un usuario cuando entra 3 veces mal su contraseña en el login.
- Updatear_monto_por_carga: se encarga de actualizar el monto de los clientes a medida que los mismo van realizando cargas. Como se explicó anteriormente, este trigger se

encuentra previo a la migración por lo tanto el dinero cargado en el anterior sistema sigue siendo válido en este diseño.

- trigger_crear_cupon: crea cupones automáticamente a medida que se van realizando compras en el sistema. El método de creación es el detallado en Tabla Cupones en la parte de migración

Functions

- existe [Campo]: funciones para verificar desde la aplicación desktop si existe o no un determinado campo más que nada aquellos de carácter unique (dni, cuit, etc)
- fnBase36: función utilizada para convertir números decimales a base36, se usa en la creación de cupones
- validar_usuario: valida los datos que ingresa un usuario al realizar un login retornando un número que la aplicación interpretara de distintas maneras mediante un protocolo.
- Hash_Contraseña: función para encriptar con SHA256 las passwords de los usuarios

APLICACIÓN DESKTOP

En esta sección iremos explicando funcionalidad por funcionalidad como fuimos resolviendo cada uno de los requerimientos presentados en el trabajo practico. Posterior a esto, se detallará el contenido de cada una de las carpetas de la solución y una descripción de las clases/forms que contienen.

Funcionalidades

ABM de Rol, Cliente y Proveedor

Para cada uno de las ABM se utilizaron 3 forms: uno para dar alta, modificar y listado. Para no reutilizar código utilizamos un patrón de diseño decorator en el cual dependiendo la funcionalidad que deseamos utilizar, el form listado tendrá nuevas funcionalidades en tiempo de ejecución a su vez cada decorator tiene un atributo que especifica su tipo (rol, proveedor o cliente). Por ejemplo:

```
new FrbaOfertas.AbmCliente.ListadoCliente(new ListadoBaja(new ClienteHandler()));

new FrbaOfertas.AbmCliente.ListadoCliente(new ListadoModificar(new ClienteHandler()));
```

Tanto la lista para modificar como la lista para dar de baja utilizan el mismo listado, la diferencia será que los decorator (ListadoBaja y ListadoModificar) agregaran una columna nueva en el DataGridView para poder seleccionar.

Todos los forms de las ABM se diseñaron teniendo en cuenta las especificaciones de la guía de ABMs propuesta por la catedra.

Login y Seguridad

El login consiste en un solo form en el cual el usuario que ingresa al sistema deberá escribir su username y password. Cuando intente ingresar la Aplicación le preguntara a la base de datos si los mismos son correctos. Debido a que hay muchas alternativas posibles a la hora de ingresar (contraseña equivocada, usuario inexistente, usuario bloqueado, usuario inhabilitado) se implementó un protocolo en el cual dependiendo el valor de que devuelva la base de datos (a través de una función) la aplicación tomara una decisión.

Ejemplo: Si el usuario ingresa bien su usuario, pero mal su contraseña, la base de datos devolverá un 4 mostrando el mensaje de error correspondiente y aumentando la cantidad de logins fallidos en 1. En caso de estar inhabilitado devolverá un 2 y se mostrará un mensaje en pantalla explicando la situación.

Finalmente, si un usuario llegase a ingresar 3 veces mal su contraseña la base de datos (mediante un trigger) lo inhabilita lógicamente. En caso de ingresar de manera correcta la cantidad de intentos vuelve a 0.

Para el cambio de contraseña, cualquier usuario puede ingresar desde el menú a las opciones y cambiarla.

Registro de Usuario

Ingresando desde el form de login, el registro de un nuevo usuario que entra al sistema consiste en ingresar un username y password y seleccionar el Rol que desea crear (los predeterminados son cliente y proveedor). Una vez seleccionado el Rol, se abrirá un form para que el usuario pueda ingresar los datos de su rol. Esta demás decir que al igual que con las ABMs, todos los campos serán validados y obligatorios. Una vez ingresados los datos, la base de datos se encargará de hacer todos los inserts necesarios para que dicho usuario pueda entrar al sistema y tener las funcionalidades correspondientes.

Carga de crédito

Un cliente podrá cargar un monto determinado para poder hacer compras. El cliente especificará cuanta cantidad de dinero desea cargar y luego se le pedirá una tarjeta. Los datos de la tarjeta serán los mínimos y necesarios para realizar una transacción. Debido a que el sistema carece conexión con un banco para poder validar la tarjeta, se asumirá que todas las tarjetas son válidas y poseen fondos ilimitados (siempre y cuando cumplan con las validaciones mínimas).

Confección y publicación de Ofertas

Un proveedor podrá cargar los datos de su oferta y publicarlos en la sección de compras de ofertas de los usuarios. Todos los campos son validados. Para la generación de un código de oferta se utiliza la siguiente estrategia

Base36(Id del proveedor + Cuit del proveedor + número de oferta)

De esta forma se asegura que cada código de oferta que se cree sea único.

En caso de ser un administrador, habrá un form más en el que se tendrá que elegir el Proveedor que publicara la oferta.

Comprar Oferta

Un cliente podrá comprar ofertas que sean publicadas en la fecha estipulada por los proveedores, tendrán un listado de todas las ofertas publicadas y podrán filtrar por nombre. Cuando seleccionen la oferta, tendrán que elegir la cantidad que desean comprar. Esta cantidad esta provista por la base de datos, pero en caso de que el stock actual de una oferta sea menor que la cantidad máxima permitida, el usuario solo podrá comprar el stock restante. De más está decir que el monto del cliente es validado a la hora de realizar una compra y se le avisara por pantalla en caso de tener créditos insuficientes. Una vez realizada la compra se mostrará por pantalla el

código del cupón que el usuario deberá presentar a un proveedor para que pueda consumir la oferta. En caso de que un usuario se olvide un cupón a canjear, el form de compra ofrece un Historial para ver todas las compras realizadas con sus respectivos cupones detallando fecha de vencimiento y si el mismo ya fue consumido o no.

Entrega/Consumo de oferta

El proveedor podrá buscar entre los códigos de los cupones pertenecientes a sus ofertas compradas por clientes. Cuando se ingrese un código de cupón valido, el campo Fecha_Consumido del cupón se actualizará con la fecha en la que el mismo fue canjeado.

ACLARACION: en caso de ser un administrador general las funciones de clientes y proveedores descritas tendrán un form extra el cual consiste en un listado donde el admin deberá elegir el cliente/proveedor sobre el cual desea aplicar la funcionalidad. En caso de ser un Rol nuevo e intentar acceder a alguna de estas funcionalidades, la aplicación no se lo permitirá debido a que precisa atributos característicos de un cliente o proveedor (por ejemplo, monto).

Facturación a Proveedor

Un administrador primero seleccionará el intervalo de fechas sobre el cual facturar, luego elegirá el proveedor y finalmente podrá ver un listado con todas las ofertas vendidas por dicho proveedor en el periodo especificado. Cuando seleccione la opción de facturar aparecerá un mensaje con el monto de la cantidad facturada y el número de factura. El número de factura será el siguiente al ultimo numero de factura generado por el sistema a partir de los números de factura migrados de la tabla maestra. Todas las compras tienen un atributo que le permite al sistema saber si dicha compra ya fue facturada o no por lo tanto no puede haber una compra facturada 2 veces.

Listado Estadístico

Se utiliza un único form en el cual se le consulta a la base de datos mediante una query dinámica (dependiendo el año, semestre, etc.) Para mostrar los datos pedidos en pantalla.

Carpetas

Abms

- Alta: form que permite dar de alta clientes se utiliza tanto en la abm como en el registro de un usuario cliente.
- Listado: listado que dependiendo el decorator tendrá distintas funcionalidades, utiliza querys dinámicas para aplicar los filtros.
- Modificacion: similar al Form de alta cliente solo que presenta ciertos atributos que solo un administrador puede modificar (modificar monto, habilitar/deshabilitar).

En el caso de rol se implementa una lista para poder quitar o agregar roles nuevos.

BaseDeDatos

- Conexión.cs: contiene todas las funciones necesarias para poder conectarse con la base de datos esto permite que no se repita el mismo código a lo largo de toda la aplicación.

CargaCredito

- CargarCredito.cs: form que permite a un cliente ingresar la cantidad a cargar.
- CargaTarjeta.cs: form que permite cargar una tarjeta para una carga.

ComprarOferta

- ComprarOferta.cs: form que permite al cliente buscar las ofertas publicadas por los proveedores con la posibilidad de comprar o ver historial. Las ofertas que aun no han sido publicadas o ya vencieron no aparecen en la lista.
- CantidadOfe.cs: cuando se selecciona una oferta a comprar se abre este form para pedirle al usuario que ingrese la cantidad que desea comprar. La misma estará limitada por la cantidad máxima de cada oferta.
- HistorialCupones.cs: form que permite a un usuario visualizar las compras que realizo con sus respectivos cupones para retirarlos.

ConectorDB

Contiene todas las funciones necesarias para recuperar la información de la base de datos que puede llegar a requerir la aplicación. Están divididas por las tablas sobre las que actúan (ej: FuncionesCliente.cs son funciones sobre la tabla Cliente). La mayoría de las querys SQL se encuentran en estas clases.

CrearOferta

- CrearOfertaAdministrador.cs: form para la carga de ofertas siendo un administrador
- CrearOfertaProveedor.cs: form para la carga de ofertas siendo proveedor

DatosPersonales->Usuarios

- DatosPersonalesUsuario.cs: permite a un usuario cambiar su contraseña. En caso de ser admin se puede acceder a un listado de usuario para cambiar la contraseña de otros.
- ListadoUsuario.cs: en caso de ser admin se podrá elegir el usuario a cambiar la contraseña
- ModificarUsuario.cs: form que permite modificar los datos de un usuario

EntregaOferta

- EntregarOferta.cs: consiste en un buscador de cupones utilizando una query que filtra por el código del cupón. Cuando se seleccione un cupón este será canjeado. Los cupones que ya están vencidos no aparecen en la lista.

Facturar

- FacturarProveedor.cs: primer form en el cual se elegirá el periodo a facturar
- FacturarListaProveedores.cs: el administrador deberá elegir al proveedor que facturará
- OfertasAdquiridasFacturasDeProveedor.cs: form que muestra todas las ofertas vendidas por el proveedor seleccionado previamente. Al facturar se imprime por pantalla el monto y el número de factura.

Home

- MenuPrincipal.cs: desde acá se puede acceder a todas las funcionalidades que un usuario tenga a su disposición. Se utilizo un menuStrip para ir verificando que funcionalidades tiene o no un usuario. En caso de no tener alguna funcionalidad el menuStrip la esconde y es inaccesible. También se consideraron los casos en los cuales rol cliente o proveedor pueden estar dados de baja, por lo tanto, no se podrá dar de alta dichos roles.

ListadoEstadistico

- ListadoEstadistico.cs: único form que contiene los datagridview y filtros para realizar las querys pedidas.

Login

- LOGIN.cs: form de login donde un usuario intenta ingresar al sistema. Tiene un botón para que el usuario pueda registrarse en caso de no tener cuenta.
- RegistroUsr.cs: Consiste en un form donde el usuario a registrarse elige su usuario y contraseña junto con el Rol que desea crear. Luego de esto se abrirá una pantalla de alta correspondiente al rol elegido.

Modelo

Contiene todas las clases que necesita el sistema

- ABMHandler: contiene las clases para que un listado sepa el tipo ABM que es (cliente, proveedor o rol).
- GuardarDB: es un state que le permite al form de darAlta saber si se está realizando un alta a través de un registro o a través de la ABM.
- Listado: Contiene los decorator necesarios para poder modificar el listado en tiempo de ejecución. Agregan una columna extra en el dataGridView del listado para determinadas funcionalidades (baja, modificación, selección,etc.).
- Roles: Los roles que usa el sistema de forma predeterminada

Luego están todas las clases que utiliza el sistema para persistir (Compra.cs, Cuenta.cs, etc.). Esto permite que solo se deba pasar una clase a la hora de persistir y no todos los atributos. Además, facilita la recuperación de una clase cuando hay que traer información de la base de datos.

PropiedadesConfig

- Config.settings: archivo config que contiene la fecha de sistema que usa la aplicación y el connection string para conectarse a la DB

Utils

Consiste de herramientas varias que se utilizan a lo largo de todo el código

- HashConsoleApp.cs: función que permite el hash de strings SHA256
- ProtocoloSQL.cs: Protocolo que utiliza el login para responder ante distintas situaciones
- Transicion.cs: contiene las funciones para wrappear la transición entre forms
- Valilador.cs: contiene todas las funciones de validación que usa el sistema, desde verificar si un campo está vacío o comprobar que un campo unique ingresado no se encuentre en la base de datos.

Conclusión

Con la realización del trabajo presentado, podemos asegurar una asimilación con los conceptos vistos en la materia permitiéndonos en un futuro realizar nuevos proyectos con aplicaciones que usen bases de datos relacionales. La parte de SQL nos permitió explayar todos los conocimientos de Base de datos vistos en clase mientras que la Aplicación fue un primer acercamiento a sistemas

basados en la gestión de datos en los cuales una buena validación y diseño son claves para su correcto funcionamiento a largo plazo.

Der