Gestión de datos
1er Cuatrimestre

FRBA COMMERCE

ESTRATEGIA

Nombre y Apellido	Legajo
Joel Melamed (Responsable)	146.804-2
Ana Estévez	146.964-2
Victoria Cabrera	146.481-4

Nombre de grupo: ATJ
Número de grupo: 32
Curso: K3014
Profesor: Marcelo Moscuzza
Fecha de entrega: 18-06-14

Índice

ESTRATEGIA

Para comenzar, el proyecto fue subdivido en 5 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 BD, como por ejemplo, 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 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. **FRBA Commerce:**

En este proyecto se encuentra la UI del sistema. Contiene todos los ABM's, historiales, 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 validacion de campos ingresados por el usuario: nulidad del campo, si es numérico, si es decimal, el cuil, el cuit, etc.
- Manejador de combos.
- Manejador de dialogos: Crea un formulario nuevo con un solo campo de texto a completar. Sirve, por ejemplo, para el cambio de clave.

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 la 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 usuario para los clientes, y CUIT para las empresas
- Definimos la clave admin (encriptada bajo el algoritmo SHA256) como default para todos los usuarios (salvo el administrador)
- Los IDs de las tablas son autonuméricos
- Definimos que las publicaciones van a migrarse todas como FINALIZADAS en su estado ya que su fecha de vencimiento ya ocurrió.
- 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, dónde el eliminado representa la eliminación lógica. En la consigna del trabajo pedía que todas las bajas realizadas fueran lógicas. Por eso mismo es que decidimos agregar la columna “Activo” a las tablas de los ABMS que incluían eliminaciones.

2. CONSULTAS

Para realizar las consultas, utilizamos la libreriaSQLClient que nos provee System.Data de C#. Esta libreria 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.

Las storedprocedures las 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 sera "traerListado" + Nombre de la entidad + Condiciones
- Para las SP que usan INSERT, su nombre sera "insert" + Nombre de la entidad + Condiciones
- Para las SP que usan UPDATE, su nombre sera "update" + Nombre de la entidad + Condiciones
- Para las SP que usan DELETE, su nombre sera "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 las SP, según corresponda.

En cuanto a cómo ejecutamos estas SP desde el código, gracias a la librería, logramos ejecutar las 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"

Indices

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

3. SOBRE LA APLICACIÓN:

3. Registro de Usuario:

Para crear un usuario se debe ingresar Username, password, rol asignado y datos identificadores según el tipo de Usuario.

Decidimos reutilizar entonces los dos formularios correspondientes al ABM de Clientes y de Empresas.

El que quiera registrarse deberá entonces ingresar username, password y elegir (en un comboBox) un rol: Empresa o Cliente.Según lo seleccionado el el combo se traerá el formulario correspondiente y se podrán insertar los datos en la Base de Datos.

Como la consigna especifica que está la posibilidad de modificar la clave, tanto por el propio usuario como por el administrativo, decidimos agregarle una funcionalidad al administrativo que consista en "Administrar Usuarios". En el formulario de la misma, el administrativo puede tanto deshabilitar usuarios como cambiarles la clave. Nuevamente aprovechamos la reutilización de forms, y en el caso de que el administrativo decida cambiarle la contraseña a un usuario se invoca al mismo formulario que usaría un usuario para cambiar su propia clave.

4. ABM de Clientes

Decidimos que cuando se crea un nuevo cliente todos los campos sean obligatorios a excepción del campo número piso y número departamento en cuanto a la dirección.
En todos los abms decidimos poner los botones fuera de las grillas por una cuestión de interaz gráfica ya que consideramos que es más simple y queda mejor que esté así.

5. ABM de Empresas

Al igual que en el ABM de Clientes, decidimos que todos los campos sean obligatorios a excepción de esos dos.

8. Generar Publicación

Consideramos que cuando se genera una Publicación, todos los campos menos acepta preguntas son obligatorios.

9. Editar Publicación

Por cuestiones de dificultad de comprensión del enunciado, decidimos que al momento de editar una publicación, la misma puede pasar del estado publicada al estado finalizada.

10.Gestión de Preguntas

Este requerimiento decidimos ponerlo en el formulario de Mis Publicaciones, ya que cuando un usuario está consultando las publicaciones que realizó puede responder o ver las respuestas para alguna de sus publicaciones. Creemos que no es necesario un Gestionar Preguntas en el menú, ya que cuando un usuario tiene acceso a ver todas sus publicaciones entonces también lo tendrá para gestionar las preguntas para las mismas.

En el caso de ver las respuestas, simplemente se muestra una grilla con un listado de las respuestas que otorgó por cada pregunta que le hicieron y la información de la publicación, y en el caso de responder preguntas, se le muestra un listado con todas aquellas preguntas que le hicieron y que debe responder.

11. Comprar/Ofertar

Las publicaciones que se muestran en el formulario de Ver publicaciones, se hacen por medio de una grilla paginada de 10 páginas

12. Calificar al Vendedor

Al realizar la migración, nos dimos cuenta que los valores correspondientes a la cantidad de estrellas de las calificaciones realizadas en la tabla maestra varían entre 1 y 10. Sin embargo, en la consigna del trabajo, mencionaba que las calificaciones únicamente podían valen del 1 al 5, por lo que tanto cuando se inserta una nueva calificación en la tabla, la cantidad de estrellas se inserta multiplicado por 2.

Además, creímos conveniente la utilización de un trigger para que cada vez que se inserte una calificación para un usuario, se actualice la reputación del vendedor que está siendo calificado.

La reputación la definimos como el promedio de estrellas obtenidas. Esto es, la suma de estrellas obtenidas por todas las calificaciones divido la cantidad de veces que un usuario fue calificando. De esta manera, en el trigger realizamos un update de la tabla clientes o empresa según corresponda y seteamos la reputación con este cálculo, contemplando ahora, la inserción de una nueva calificación

13.Historial Cliente

En este requerimiento decidimos hacer un mismo formulario con una misma grilla para las 4 operaciones. El ítem de Calificaciones otorgadas y recibidas, lo distinguimos como dos operaciones diferentes ya que un mismo usuario consulta las calificaciones que otorgó a otros y las que recibió de otros.

14. Facturar Publicaciones

Para este requerimiento decidimos hacer un form que consista en una grilla de publicaciones a rendir. En la misma se mostraran todas aquellas publicaciones pendientes que tenga el usuario para facturar ordenadas por antigüedad. Luego se requiere que el usuario ingrese la cantidad de publicaciones que el mismo quiere facturar y que seleccione la forma de pago del combo box. El combo box es cargado al principio con la descripción de todas las formas de pago disponibles que se encuentran en la Tabla Forma_Pago de la Base de Datos.

Cuando el usuario apreta el botón Facturar, primero se validan los campos ingresados, es decir que se verifica que la cantidad no se vacía y que sea un número. Luego, mediante un for que va desde 0 hasta la cantidad de publicaciones a rendir ingresadas, se guardan las publicaciones futuras a facturar en una lista auxiliar que será utilizada más adelante.

Después, por cada publicación en esa lista, se obtendrá primero el listado de compras que fueron realizadas así el usuario puede pagar las comisiones correspondientes a la cantidad de

compras (tanto de compra directa como por ofertas) por publicación. Entonces, cada publicación (por cada compra directa) significará un nuevo Item_Factura en la factura final, donde el código de publicación del ítem será el correspondiente al de la publicación, la cantidad será la correspondiente a la compra que se realizó, el precio será el porcentaje de la visibilidad de la publicación por el precio de la compra por la cantidad que se vendieron en esa compra.

Al mismo tiempo, una compra puede haber sido realizada también por ofertas. Entonces se obtienen todas las ofertas que hayan sido hechas según el código de publicacion. Nuevamente cada oferta significará un nuevo ítem en la factura, donde el código publicacion sigue siendo el código de la publicación, la cantidad será 1, y el monto del ítem será el precio de la oferta por el porcentaje de visbilidad de la publicación.

Finalmente se agrega un último ítem_factura a la factura que es el ítem propio de la publicación en sí haciendo referencia al costo de publicar según la visibilidad.

Al momento de crear una factura, la forma de pago se obtiene según lo seleccionado por el usuario en el combo box, y el precio total de la factura equivale a la suma de todos los montos de los ítems que van a pertenecer a esa factura.

Cuando la factura se inserta en la tabla, decidimos que devuelva su id de factura y que se guarde en una variable. Este va a ser utilizado luego para que cuando se inserten los ítems de la factura en la tabla ítem_factura, lo hagan con ese número de factura.

15. Listado Estadístico

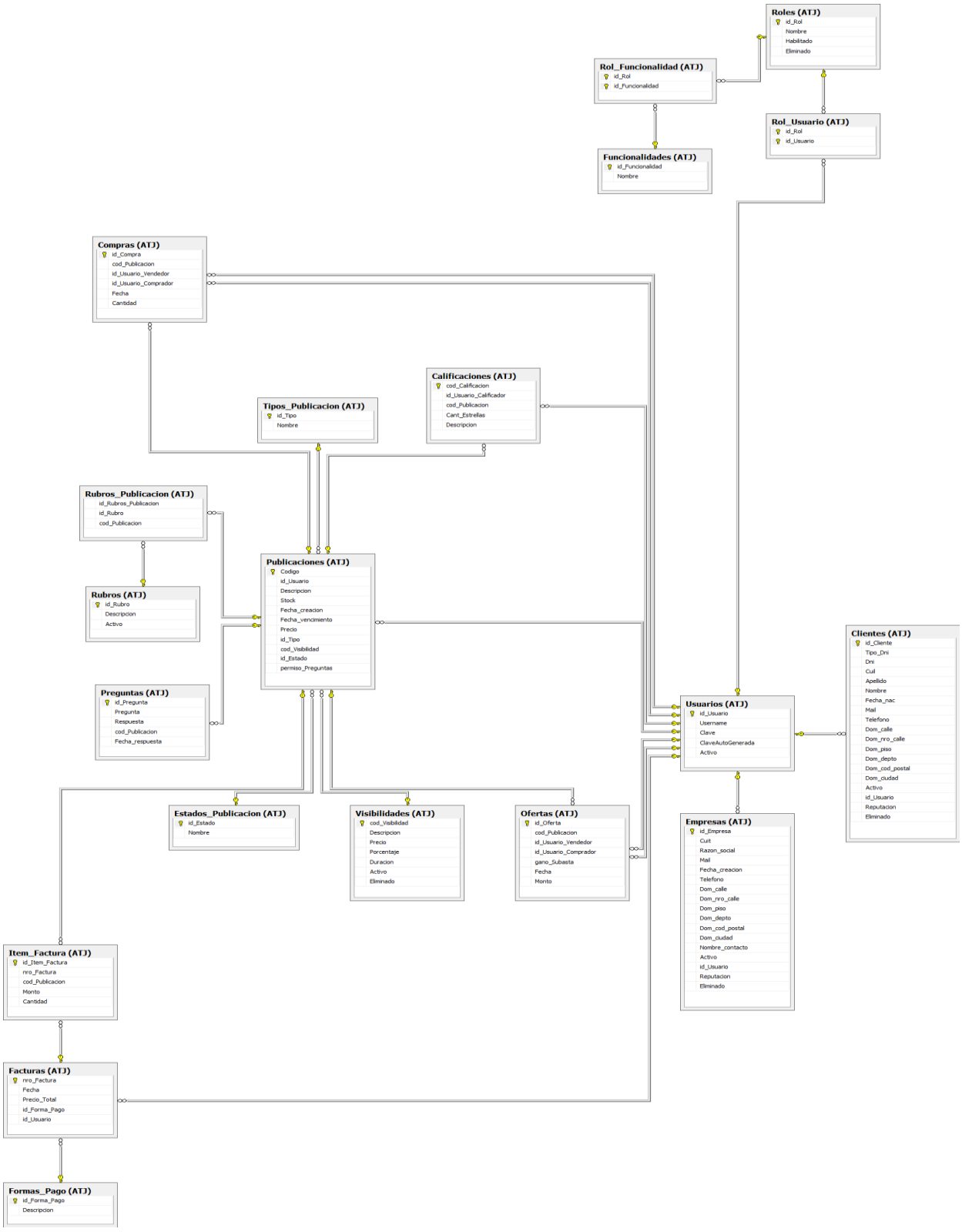
Para este requerimiento decidimos hacer un mismo form donde mediante un combo box se pueda seleccionar el listado estadístico que se quiera consultar.

En el caso del primer listado, de vendedores con mayor cantidad de productos no vendidos, decidimos no considerar el año como un filtro más ya que nos pareció que era irrelevante. Esto se justifica ya que apenas se ingresa al form, lo primero que el usuario debe ingresar es el año por el cual quiere consultar el listado, entonces no tenía sentido hacer un filtro cuando ya el listado iba a estar “filtrado” por un año.

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.png" en el directorio raíz