|  |
| --- |
| LOS\_JUS |
| Estrategia |
| FRBA-Commerce |

|  |
| --- |
|  |

Base de datos

A partir de la tabla Maestra provista creamos un modelo de datos para satisfacer las necesidades actuales del negocio. Se ampliarán algunos aspectos de este modelo en el siguiente trabajo.

# Tablas

Rol, Funcionalidad, RolxFuncionalidad, Usuario, RolxUsuario, Cliente, Empresa, Publicación, Pregunta, Visualización, PublicacionXVisualización, Rubro, RubroXPublicación, Operación, Calificación, Facturación, Ítem, Compra, Subasta, Sesión

# Procedimientos de migración

Migrar\_Usuarios\_cli, Migrar\_Usuarios\_emp, Migrar\_Rubros, Migrar\_Visualizacion, Migrar\_Publicaciones, Migrar\_Operaciones, Migrar\_Facturas

# Acotaciones

La estructura que usamos para ejecutar la migración, fue antes que nada, ir borrando (en caso que exista) de lo mínimo y más particular a lo general. Es decir, comenzamos buscando los procedimientos y borrándolos, para que en caso que existan al crear no nos de error. Luego pasamos a buscar las claves foráneas y primarias para borrarlas y así poder buscar y borrar las tablas en caso que existan. Por último lo que borramos es el esquema “Los\_Jus”.

Todo esto nos da una base limpia para comenzar a trabajar.

Una vez que tenemos la base limpia, procedemos de manera inversa a lo mencionado anteriormente.

Creamos el esquema, luego las tablas, luego las claves, después los procedimientos y funciones para que la aplicación funcione de manera correcta.

Adjuntamos un Diagrama de Entidad de Relaciones que clarifica un poco el modelo que usamos para el trabajo.

La migración de datos decidimos realizarla mediante procedimientos porque nos da la posibilidad de manejar objetos como cursores para recorrer fila a fila de una selección de registros e ir acomodando cada valor en las tablas que sea necesario.

Cuando migramos, aquellos datos que no estaban en la tabla Maestra pero que no podían ser null en el nuevo sistema (para que la base sea consistente), los inicializamos en un valor base a todos por igual para que posteriormente (y por fuera de este trabajo) si fuese necesario, se actualicen debidamente sus valores. Este es el caso por ejemplo del campo teléfono de empresa y cliente (actualmente en la migración lo insertamos como "4444444"). En el caso de la contraseña y usuario, se migraron los datos viejos con Username de empresa igual a CUIT, Username de Cliente igual a CUIL y la contraseña en ambos casos en "gd2014". En las tablas solo se guarda el Hash de las contraseñas.

La mayoría de las tablas tienen IDENTITY en sus claves primarias para facilitar el manejo de datos y prever inconsistencias. Las tablas cuyos códigos no son identity, se utilizo el siguiente manejo: son tablas cuya PK es exactamente igual al campo de otra tabla que si es identity (por ejemplo la relación entre los campos cli\_id y usu\_id) o son tablas que desde la Maestra venía un código determinado ya ingresado. Estos tipos de códigos eran consecutivos (aunque no empezaran desde 0) asi que para las nuevas inserciones, se tomó la ultima PK de esa tabla, y se le sumo uno.

El manejo de usuarios se realiza a través de tres tablas: Usuarios, Clientes y Empresa. Cliente-Usuario y Empresa-Usuario comparten un id, por lo que son fácilmente identificables, y mas simples de "joinear" a la hora de hacer consultas y procedimientos.

Todos las tablas tienen un campo "eliminado" y las principales (siguiendo con el enunciado) tiene también un campo "habilitado". El tratamiento de esos campos se hace a través de triggers sucesivos. En cada tabla hay un trigger, y cuando se realiza un update de campo "eliminado", el trigger se dispara y activa otra tabla, donde existe otro trigger, que a su vez se dispara. La idea es entonces, que si se elimina lógicamente cualquier objeto de la base de datos, todas las tablas asociadas a ese objeto también se darán de baja lógicamente, en forma de cascada.

Muchos de los campos de las tablas tienen DEFAULT para facilitar el proceso de inserción desde la aplicación y para evitar dispara errores en la aplicación como cuando el DataReader de una consulta intenta leer un campo null.

Todos los tipos de datos ya existentes en la Maestra fueron respetados, tanto en la migración como en nuevas inserciones desde la aplicación.

Se realizaron muchos procedimientos y triggers, para "aliviar" la lógica de la aplicación y evitar consultas complejas desde Visual C#.

Aplicación

Encaramos la aplicación con el principal objetivo de darle visibilidad a los datos, consultas, procedimientos y funciones de la Base de Datos.

Por lo tanto, la lógica del tratamiento de los datos se encuentra en la Base, y en la aplicación nos dedicamos a comprobar los datos para luego, mostrarlos, almacenarlos o darles algún tratamiento con las funciones hechas.

El Log In tiene una lógica simple, utilizando el protocolo de encriptación según lo estipulado. Los datos del usuario se almacenan en el momento de ingreso al sistema en la tabla Sesión, a donde se irá a buscar en caso de necesitar el id del que esta navegando el sistema, su rol o su username.

El menú principal da la opción de elegir con que rol ingresar al sistema (en el caso que tenga mas de uno). El administrador puede entrar a cualquiera de las funcionalidades del sistema aunque no sea exclusivamente propia.

Las funcionalidades son consultadas a la base y se despliegan en un ComboBox para asegurarse de que si en algún momento se modifican las funcionalidades del rol en la base, la aplicación continúe ejecutando correctamente los datos.

El administrados es el único del sistema que puede ingresar a los ABMs e ingresar nuevos usuarios, roles, y demás objetos del sistema.

En el caso del cliente o empresa, si se quisiera crear un usuario que conste con dos roles, se puede dar de alta un nuevo cliente (desde alta ABM de cliente) o una nueva empresa (desde alta AMB de empresa) y pedir que comparta el usuario con otro de diferente rol.

Desde la pantalla modificación, el administrador podrá, además de modificar los datos que aparecen en pantalla, seleccionar la opción "habilitar" para volver a habilitar un Rol, Empresa o Cliente inhabilitado.

En cuanto a la parte de facturación, tratamiento de comisiones por visibilidad y premios, se maneja todo desde la aplicacion e implementando funciones desde la base. Por ejemplo se opto por no poner un contador para ver cuando se "bonificaba" una determinada compra, porque complejizaba la base, y era más eficiente trabajarlo desde el programa.

El tratamiento de tipos de datos y comprobación de datos se realiza desde la aplicación. Los tipos de datos numéricos con decimales se trabajan en dos textBox para evitar que el usuario del sistema ingrese datos erróneos. Se controlan también aquellos datos que deben ser únicos en la base, consultando previo a la inserción si no existe ese valor en la Base. Sin embargo, solo por precaución, utilizamos algunos try y catch en lugares críticos para poder manejar correctamente el tratamiento de errores y excepciones. Cualquier error ocurrido se avisara pertinentemente con un MessegeBox, informando el error y su tipo.

En numerosas pantallas agregamos también un leyenda con información que se activa, brindando información importante al pasar por encima de cualquier control.

El manejo de consultas y display de datos se hace todo a través de DataGridView, y se utilizan DataReaders para poder acceder a cada registro y cada columna. En algunos casos complejos (como por ejemplo, en facturación) nos vimos en la necesidad de usar DataReaders encadenados, por lo que nuestra cadena de conexión a la base establece también la condición MARS ( "MultipleActiveResultSets=True") .

La paginación de las publicaciones fue una de las partes más complejas y laboriosas de todas. Es por esto, que decidimos poner la lógica utilizada en una clase aparte (clase sin form) y luego llamar los métodos para luego poder manejar los registros correctamente.

Todas las ventanas de la Aplicación tienen el mismo tratamiento: no pueden ser modificadas de tamaño (FixedSingled), no pueden ser maximizadas (pero si minimizadas) y aparecen en el centro de la pantalla. Se priorizo el uso de pantallas contenedoras (como las de los ABM) porque resultaban más intuitivas para el usuario.