Gestión de Datos
Trabajo Práctico

2° Cuatrimestre 2018

Frba-PalcoNet

Grupo #42 - PMD		
Integrantes		
Nombre	**Legajo**	**E-mail**
Tonelli, Bruno	1590364	brunotonelli97@gmail.com
Lucero, Facundo	1555789	facundo.lucero@outlook.es
Peñoñori, Stefano	1521342	stefanonicolas@gmail.com

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Índice

1. Tablas

Para persistir las entidades principales del sistema, se crearon las tablas **Cliente, Espec_Empresa, Espectaculo.**

A su vez, ya que un Cliente y una Empresa pueden asociar un usuario, se creó la tabla **Usuario** para almacenar los datos sobre su cuenta.

Los espectáculos tienen publicaciones asociadas, para esto se creó la tabla **Publicación,** que representa una fecha y hora de ese espectáculo. Puede haber múltiples publicaciones sobre el mismo espectáculo (no misma fecha/hora). La tabla **Grado_Publicacion** abstrae la prioridad de esta publicación, y la comisión a cobrarle a la empresa por cada venta.

De cada publicación se requería saber las entradas compradas y disponibles. Para ello se agregó un campo Localidades en la tabla Publicación (entradas totales).

Una entrada se representa en la tabla **Ubicación**, cuya identidad está dada por el numero de asiento y letra de fila + la publicación a la que pertenece. Esta ubicación puede estar asociada (FK) a una compra, en tal caso no se encuentra disponible esa ubicación.

Una compra tiene un cliente asociado y en esa misma compra se pueden asignar varias ubicaciones, por lo que tiene un campo Cantidad.

Por cada compra se genera un registro de la tabla **Item_Factura**, que simboliza un ítem de la factura entregada a la empresa (tabla **Factura**) que realizó esa publicación. Ese ítem factura se realiza sobre las comisiones a cobrarle a la empresa.

Las tablas **Estado, Rol, Rubro** representan posibles valores que pueden adquirir campos de otra tabla (estado de publicación, rol de usuario, rubro del espectáculo respectivamente).

Un rol tiene varias funcionalidades (tabla **Funcionalidad**), por lo que se requiere una tabla **Func_Rol.** A su vez, un usuario puede tener varios roles, por lo que existe la tabla Rol_X_Usuario.

Por último, cada compra realizada por el cliente le genera puntos equivalentes a un 35% del total de la compra. Esos puntos vencen a 6 meses de realizada la compra. Se creó tabla **Puntos**.

Con los puntos, un cliente puede adquirir premios (tabla **Premio**). Tomamos la decisión de que puede adquirir el mismo premio múltiples veces y el premio puede ser canjeado por varios clientes, por lo que creamos la tabla **Premio_X_Cliente.**

2. Funciones

UbicacionesLibres: función que recibe una publicación y me devuelve la cantidad de asientos/ubicaciones libres. Se usa en conjunto con el trigger FinalizarPublicaciones.
localidadesVendidas: función que, dada una empresa, un año y un trimestre, me devuelve la cantidad de ubicaciones que vendió en ese periodo.
localidadesTotales: función similar a la anterior, devuelve el total de entradas que esa empresa tenía en ese periodo.
Ambas funciones se usan en conjunto para realizar el listado estadístico.

3. Triggers

ComprasNegativas: trigger que, al realizarse compras, verifica que no sean negativas (su total). Caso contrario, se elimina la compra.
FinalizarPublicaciones: trigger que, luego de realizada una compra, verifica la cantidad restante de ubicaciones disponibles en esa publicación. Si la cantidad es 0, se pasa a estado Finalizada esa publicación.
VerificarRol: trigger para que un usuario no pueda ser asignado un rol inhabilitado.
VerificarFechas: trigger para que la fecha de una publicación no sea mayor a la del evento.
VerificarEstado: trigger para que que una publicación ya finalizada no pueda cambiar su estado.

4. Migración

Decisiones:

- Todos los rubros presentes en la tabla maestra estaban vacios, se les asignó N/A al migrarlos.
- No se creó una tabla para formas de pago, se guarda su descripción en una columna de la tabla Compra.
- Se usa un procedure para cargar publicaciones en estado borrador, para poder testear funcionalidad Editar.
- Los tipos de ubicación vienen con un código. Asumimos que cada empresa podría llegar a ingresar descripciones/codigos de ubicaciones diferentes, por lo que solo se persiste la descripción del tipo, dentro de la tabla Ubicación.
- Asumimos que todos los clientes en Maestra, tienen tipo de documento “DNI”, ya que no existe esa columna en la tabla.
- Cada espectáculo tiene una única fecha en Maestra, se crea una publicación por cada uno.
- También se asignan los puntos a los clientes.
- Se requería un usuario **Administrador general**, para ello se creó un nuevo rol que tiene todas las funcionalidades, a su vez ese usuario tiene todos los roles asignados para poder testear visualmente la aplicación. Como las compras son

realizadas por clientes, y las publicaciones por empresa, se tuvieron que generar 1 cliente y 1 empresa para este usuario Administrador General.
- Se agregaron compras no facturadas, para poder **Generar rendición de comisiones**, ya que las compras de Maestra tienen su factura ya.
- Se agregaron premios (10)
- Se le agregaron 7500 puntos al administrador general. 1000 de esos puntos se encuentran vencidos.

5. Desarrollo de la aplicación

Se utilizaron los siguientes paquetes:

- Entity Framework para facilitar las consultas a la base de datos.
- Newtonsoft.Json para serializar/deserializar la configuración.
- PagedList para realizar las paginaciones sobre los grids de datos.

- Se generaron métodos de extensión.
- Se crearon User Controls para abstraer visual y funcionalmente los textbox de campos requeridos, premios y el listado de funcionalidades.
- La configuración se levanta al iniciar la aplicación, y se almacena de forma estática para su consulta durante la ejecución del programa.
- Las contraseñas se encriptan desde la aplicación.
- La sesión se guarda de forma estatica y se realizan consultas sobre ella para obtener el cliente/empresa que está logeado. También sobre el rol para ver los permisos en las vistas.
- Se delegó sobre la clase ValidadorCampos la totalidad de las validaciones sobre campos numéricos, not null, y validez de CUIT. Se utilizaron Error Providers de Windows Forms.
- En la clase ValidacionesInput se abstrae el checkeo de entidades que ya existen (verificación de la no repetición de empresas con tal cuil/razón, etc).
- Cuando una entidad es dada de baja (inhabilitada de forma lógica), se sigue mostrando en los grids, pero en rojo.
- Se optó por un archivo JSON simplificado para la configuración, ya que el App.config predeterminado contenía información innecesaria y de menos legible.