

Estrategia TP Gestión de Datos: PalcoNet

Curso: K3522

Nombre del grupo: MATE_LAVADO

Integrantes:

Chipian, Rocío, 1594746

Giorda Cristian, 1594369

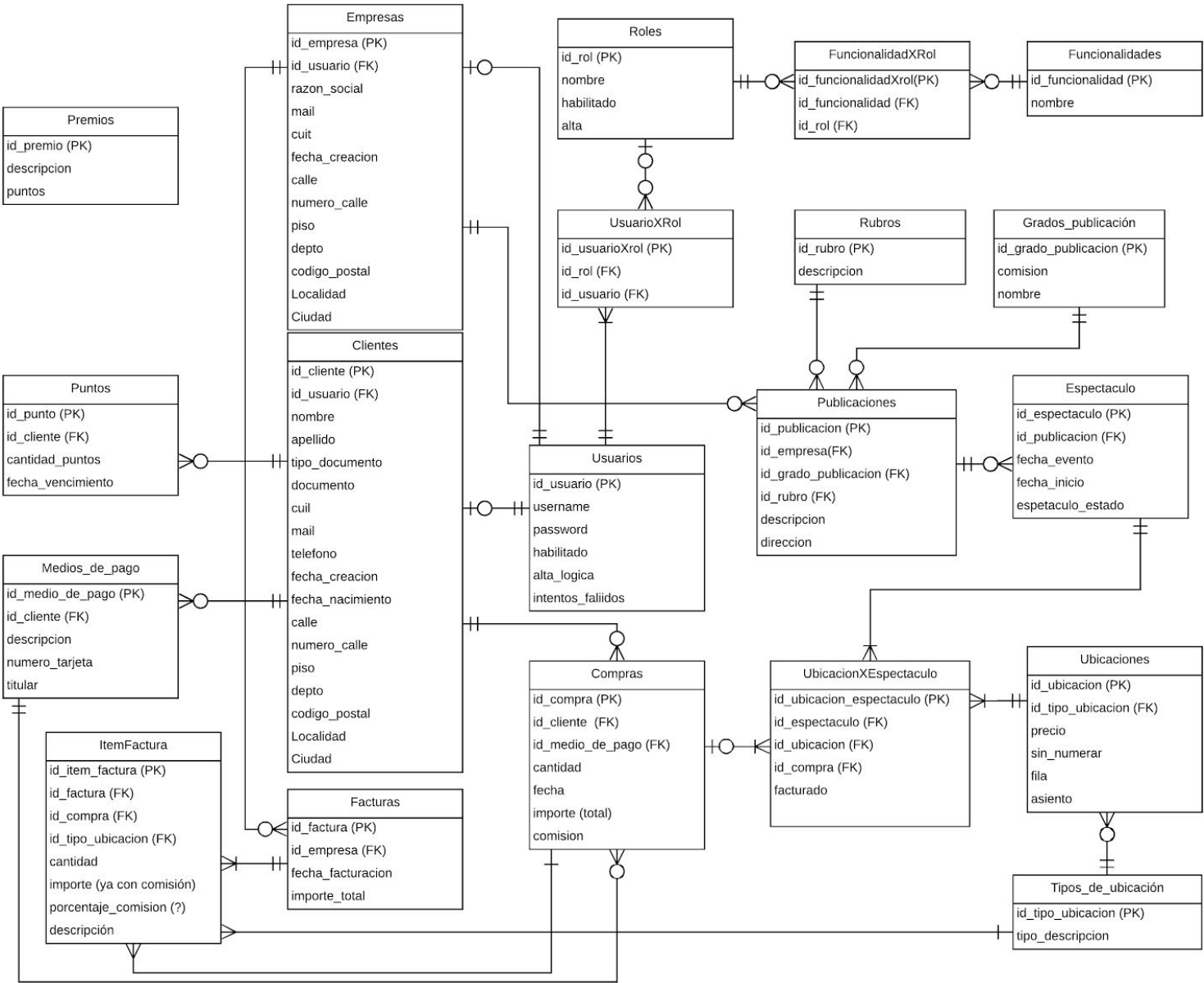
Mazzini Ivana, 1568322

Roldán Lucía, 1595313

Índice

DER	3
Entidades	4
Usuarios	4
UsuarioXRol	4
Roles	4
FuncionalidadXRol	4
Funcionalidades	5
Clientes	5
Medios_de_pago	5
Puntos	5
Empresas	6
Facturas	6
Compras	6
Publicaciones	6
Espectaculos	7
UbicacionXEspectaculo	7
Rubros	7
Grados_publicacion	7
TiposDeUbicacion	7
Ubicaciones	8
Premios	8
Funciones, Triggers y Stored Procedures	9
Lista completa de Stored Procedures	9
Lista Completa de Triggers	10
Lista completa de Funciones	10
Observaciones	11
Desarrollo del TP	13

DER



Entidades

Usuarios

- id_usuario INT IDENTITY(1,1) PRIMARY KEY,
- username VARCHAR(255) UNIQUE,
 - Nombre de usuario que se utiliza en el login.
- password VARCHAR(255),
 - Encriptada utilizando SHA256.
- habilitado BIT,
 - Indica si el usuario está habilitado o no, si no esta habilitado no podrá ingresar a la aplicación.
- alta_logica DATETIME,
 - Indica cuando fue creado el usuario.
- intentos_fallidos INT,
 - Cuando estos son iguales a 3 el usuario queda inhabilitado.
- debe_cambiar_pass BIT;
 - Indica si el usuario fue generado automáticamente y por lo tanto debe cambiar su nombre de usuario y contraseña al ingresar por primera vez al sistema. Los administradores pueden generar usuarios ingresando un nombre de usuario y contraseña, esos usuarios no deben cambiar su nombre de usuario y contraseña al ingresar por primera vez a su cuenta.

UsuarioXRol

- id_usuarioXrol INT IDENTITY(1,1) PRIMARY KEY,
- id_usuario INT REFERENCES Usuarios,
- id_rol INT REFERENCES Roles;

Es una tabla intermedia entre Usuarios y Roles dado que entre ellos hay una relación de muchos a muchos, un usuario puede tener muchos roles y muchos usuarios pueden tener el mismo rol.

Roles

- id_rol INT IDENTITY(1,1) PRIMARY KEY,
- nombre CHAR(40),
 - Por ejemplo: “Cliente”, “Empresa” y “Administrador”.
- habilitado BIT,
 - Así como pueden deshabilitarse usuarios también pueden deshabilitarse roles, lo que significa que ningún usuario podrá loguearse con el rol determinado.
- alta BIT;
 - La baja es lógica y este bit sirve para indicar si el rol fue eliminado o no.

FuncionalidadXRol

- id_funcionalidadXRol INT IDENTITY(1,1) PRIMARY KEY,
- id_funcionalidad INT REFERENCES Funcionalidades,
- id_rol INT REFERENCES Roles;

Es una tabla intermedia dado que un rol puede incluir muchas funcionalidades y una misma funcionalidad puede ser incluida por muchos roles.

Funcionalidades

- id_funcionalidad INT IDENTITY(1,1) PRIMARY KEY
- nombre VARCHAR(100);
 - Por ejemplo: “Comprar”, “Generar Publicación” y “Generar rendición de comisiones”.

Clientes

- id_cliente INT IDENTITY(1,1) PRIMARY KEY
- id_usuario INT NOT NULL REFERENCES Usuarios,
- nombre NVARCHAR(255),
- apellido NVARCHAR(255),
- tipo documento CHAR(3),
 - Como “DNI” o “CI”
- documento NUMERIC(18,0),
- cuil NUMERIC(18,0),
- mail NVARCHAR(50),
- telefono NUMERIC(15),
- fecha_creacion DATETIME,
- fecha_nacimiento DATETIME,
- calle NVARCHAR(255),
- numero_calle NUMERIC(18,0),
- piso NUMERIC(18,0),
- depto NVARCHAR(255),
- codigo_postal NVARCHAR(50),
- localidad NVARCHAR(255),
- ciudad NVARCHAR(255);

Medios_de_pago

- id_medio_de_pago INT IDENTITY(1,1) PRIMARY KEY,
- id_cliente INT REFERENCES Clientes,
- descripcion VARCHAR(10) CHECK(descripcion IN ('Efectivo', 'Tarjeta')),
 - La aplicación solo permite el pago por medio de tarjeta pero agregamos la opción efectivo únicamente para la información previa a la aplicación dado que no se tiene registrado ningún dato sobre el medio de pago.
- nro_tarjeta NUMERIC(30),
 - Para los casos de efectivo el número de tarjeta es null

- titular NVARCHAR(50);

Puntos

- id_punto INT IDENTITY(1,1) PRIMARY KEY,
- id_cliente INT REFERENCES Clientes,
- cantidad_puntos BIGINT,
 - La cantidad de puntos se equivale con el importe de la compra redondeado.
- fecha_vencimiento DATE;
 - Los puntos tienen un periodo de un 6 meses para ser utilizados

Empresas

- id_empresa INT IDENTITY(1,1) PRIMARY KEY
- id_usuario INT REFERENCES Usuarios,
- razon_social NVARCHAR(255) UNIQUE,
- mail NVARCHAR(50),
- cuit NVARCHAR(255),
 - Transformada para que no tenga guiones.
- fecha_creacion DATETIME,
- calle NVARCHAR(50),
- numero_calle NUMERIC(18,0),
- piso NUMERIC(18,0),
- depto NVARCHAR(50),
- codigo_postal NVARCHAR(50),
- ciudad NVARCHAR(255),
- localidad NVARCHAR(255);

Facturas

- id_factura INT PRIMARY KEY
- id_empresa INT REFERENCES Empresas,
- fecha_facturacion DATETIME,
- importe_total NUMERIC(18,2);

Compras

- id_compra INT IDENTITY PRIMARY KEY
- id_cliente INT REFERENCES Clientes,
- id_medio_de_pago INT REFERENCES Medios_de_pago,
- comision NUMERIC(3,3),
 - La comisión que se deberá cobrar al momento de facturar, esta se encuentra dada por el grado de la publicación al momento de realizar la compra.
- fecha DATETIME,
- importe INT;
 - Es la sumatoria del valor de cada entrada multiplicado por la cantidad de ese tipo de entradas compradas.

Publicaciones

- id_publicacion INT IDENTITY PRIMARY KEY
- id_empresa INT REFERENCES Empresas,
- id_grado_publicacion INT REFERENCES Grados_publicacion,
- id_rubro INT REFERENCES Rubros,
- descripcion NVARCHAR(255),
 - Descripción de la publicación, ejemplo: “Buenos Aires, Argentina - Roger Waters visitará la Argentina en Noviembre de 2018 con su nueva gira: Us + Them, con clásicos de Pink Floyd, algunas canciones nuevas y su trabajo como solista.”
- direccion VARCHAR(80);
 - Indica dónde se realizará el espectáculo.

Espectaculos

- id_espectaculo INT PRIMARY KEY
- id_publicacion INT REFERENCES Publicaciones,
- fecha_inicio DATETIME,
- fecha_evento DATETIME,
- estado_espectaculo CHAR(15) CHECK(estado_espectaculo IN ('Borrador', 'Publicada', 'Finalizada'))
 - Las publicaciones deben tener uno de esos tres estados.

UbicacionXEspectaculo

- id_ubicacion_espectaculo INT IDENTITY PRIMARY KEY
- id_espectaculo INT REFERENCES Espectaculos,
- id_ubicacion INT REFERENCES Ubicaciones,
- id_compra INT REFERENCES Compras,
- facturado BIT;
 - Indica si la ubicación ya fue facturada o no.

Es una tabla intermedia entre Ubicaciones y Espectaculos dado que puede haber muchos espectaculos (fechas) de un mismo show y por lo tanto tendrán los mismos asientos, muchas fechas y muchas ubicaciones.

Rubros

- id_rubro INT IDENTITY(1,1) PRIMARY KEY
- descripcion NVARCHAR(100);
 - Por ejemplo: “Concierto”, “Musical” y “Stand-Up”

Grados_publicacion

- id_grado_publicacion INT IDENTITY(1,1) PRIMARY KEY
- comision NUMERIC(3,3),

- Es el porcentaje del importe que se le cobrará a la empresa que haya publicado la publicación.
- nombre NVARCHAR(20);

TiposDeUbicacion

- id_tipo_ubicacion INT IDENTITY PRIMARY KEY
- descripcion NVARCHAR(255);

Ubicaciones

- id_ubicacion INT IDENTITY PRIMARY KEY
- codigo_tipo_ubicacion INT,
- tipo_ubicacion NVARCHAR(20),
 - Una descripción del tipo de asiento como: "Platea", "Pullman" o "Vip".
- fila VARCHAR(3),
 - Indica el número de fila, para el caso de que la entrada no sea numerada es null.
- asiento NUMERIC(18),
 - Para las entradas numeradas indica el número de asiento dentro de una fila, para las no numeradas simplemente el número de entrada.
- sin_numerar BIT,
 - Indica si la ubicación es numerada o no.
- precio NUMERIC(18);

Premios

- id_premio INT IDENTITY(1,1) PRIMARY KEY
- descripcion VARCHAR(110),
 - Ejemplo: "Set de platos", "Fin de semana en Tandil" y "Pava electrica"
- puntos BIGINT;
 - Los puntos que se necesitan para adquirir el premio

ItemFactura

- id_item_factura INT IDENTITY(1,1) PRIMARY KEY,
- id_factura INT REFERENCES Facturas,
- id_compra INT REFERENCES Compras,
- id_tipo_ubicacion INT REFERENCES TiposDeUbicacion,
 - Para saber el tipo de las entradas.
- cantidad INT,
 - La cantidad de entradas que están incluidas.
- importe NUMERIC(18,2),
 - Es el precio multiplicado por la cantidad de entradas.
- comision NUMERIC(3,3),
 - El porcentaje de comisión que tenía la publicación según su grado al momento de la compra.
- descripcion NVARCHAR(60);

- Indica que es una facturación de comisiones.

ItemFactura representa cada línea que se encuentra en una factura, o sea, indica qué se factura, el valor y detalla la cantidad de entradas y el precio de ellas.

Funciones, Triggers y Stored Procedures

Adoptamos, como estrategia de trabajo, desarrollar Stored Procedures para prácticamente todas las consultas y/o actualizaciones a la base de datos, de forma que puedan ser ejecutadas desde Visual Studio, con los parámetros correspondientes, de ser necesario. Al trabajar de esta manera ganamos una capa de seguridad de los datos de la base. Además creamos algunos triggers y funciones.

Algunos **triggers** relevantes que realizamos son:

- Trigger “finalizarEspectaculoAgotado” AFTER UPDATE de UbicacionXEspectaculo, que verifica si se vendieron todas las localidades para un Espectáculo, y de ser así, lo finaliza. Consideramos que una publicación está finalizada si todos sus espectáculos están finalizados (se comprueba al traer las publicaciones).

Algunos **stored procedures** relevantes que realizamos son:

- Procedure “BuscarPublicacionesPorCriterio” como las publicaciones se pueden buscar por uno o más criterios creamos una query dinámica. El stored procedure recibe la descripción, rubros y fecha de inicio y fin. Estos campos pueden ser nulos o no por lo que el procedure verifica si son nulos y en caso de no serlo agrega una cláusula al where de la query que se comenzó a generar anteriormente. Al tener la query completa guardada en una variable de tipo varchar la ejecuta obteniendo así los resultados.
- Procedure “agregarUbicaciones” se ocupa de analizar si la ubicación es numerada o no y llama a otro stored procedure según corresponda.

Lista completa de Stored Procedures

- actualizarGradoPublicacion_sp
- actualizarPublicacion_sp
- actualizarUsuarioYContrasenia_sp
- agregarEspectaculo_sp
- agregarFactura_sp
- agregarFuncionalidadARol_sp
- agregarRol_sp
- agregarUbicaciones_sp
- agregarUbicacionNumerada_sp
- agregarUbicacionSinNumerar_sp
- agregarUbicacionXEspectaculo_sp
- asientosDisponiblesSegunEspectaculoYFila_sp
- borrarPuntos_sp
- buscarClientePorUsername_sp
- buscarComprasNoFacturadas_sp
- buscarEmpresaPorCriterio_sp
- buscarEmpresaPorUsername_sp
- buscarEspectaculosBorradorPorPublicacion_sp

- buscarEspectaculosPorPublicacion_sp
- buscarPublicacionesPorCriterio_sp
- buscarPublicacionesPorEmpresa_sp
- buscarUbicacionesPorEspectaculo_sp
- buscarUbicacionesPorPublicacionEdicion_sp
- buscarUsuarioPorCriterio_sp
- crearItemFactura_sp
- cuilEsValido_sp
- cuitEsValido_sp
- deshabilitarUsuario_sp
- elClienteExiste_sp
- elClienteTieneInfoCompleta_sp
- eliminarEspectaculo_sp
- eliminarFuncionalidadesRol_sp
- eliminarPublicacion_sp
- eliminarRol_sp
- eliminarUbicacion_sp
- filasDisponiblesSegunEspectaculo_sp
- getDatosTarjeta_sp
- getFuncionalidadesDeRol_sp
- getFuncionalidadesDeUsuario_sp
- getMediosDePago_sp
- getPremios_sp
- getPublicacionesDeUsuario_sp
- getPuntos_sp
- getRolesDeUsuario_sp
- getRolesHabilitados_sp
- getRubros_sp
- getRubrosDePublicacion_sp
- habilitarUsuario_sp
- historialClienteConOffset_sp
- laEmpresaExiste_sp
- laEmpresaTieneInfoCompleta_sp
- modificarCliente_sp
- modificarEmpresa_sp
- modificarNombreRol_sp
- modificarRol_sp
- modificarTarjeta_sp
- modificarTitularTarjeta_sp
- obtenerDatosAdicionalesCliente
- obtenerDatosAdicionalesEmpresa
- registrarCompra_sp
- registrarCompraExU_sp
- registrarCompraUbicacion_sp
- registrarMedioDePago_sp
- registrarPublicacion_sp
- registroCliente_sp
- registroClienteConUsuario_sp
- registroEmpresa_sp

- registroEmpresaConUsuario_sp
- top5ClienteComprasParaUnaEmpresa_sp
- top5ClientesPuntosVencidos_sp
- top5EmpresasLocalidadesNoVendidas_sp
- traerTodasRazonesSociales_sp
- ubicNumeradaDisponiblesSegunEspectaculoYTipoUbicacion_sp
- ubicSinNumerarDisponiblesSegunEspectaculoYTipoUbicacion_sp
- usuarioEsCliente_sp
- usuarioEsEmpresa_sp
- vaciarEspectaculosPublicacion_sp
- verificarLogin_sp

Lista Completa de Triggers

- insertarNuevoEspectaculo
- insertarNuevaFactura
- rollInhabilitado
- insertarNuevaCompra
- finalizarEspectaculo

Lista completa de Funciones

- getCantidadEntradasVendidas
- getCantidadEntradasPublicacion
- fn_ValidarDigitoVerificador
- fn_cuitDeEmpresa
- fn_cuilDeCliente

Observaciones

- Tanto las empresas como los clientes tienen un usuario, con un nombre de usuario (único) y password (encriptado).
- Agregamos campos para la baja lógica y para verificar si se pasó de los 3 intentos máximos para conectarse.
- Se proponen 3 roles que son: empresa, cliente y administrativo (los manejamos como ids y cadenas de texto).
- El rol también puede inhabilitarse
- Se proponen diferentes funcionalidades a las que cada rol tiene acceso, las cuales también manejamos con ids y una descripción.
- Al registrar una empresa se verifica que el CUIT sea válido
- Tanto para esta tabla como para los clientes decidimos no normalizar el domicilio (que incluiría a los campos calle, numero_de_calle, piso, depto, código postal, algo que por ejemplo en esta migración no ocurre nunca) ya que lo que se gana es muy poco (que justo dos domicilios de un cliente o empresa coincidan) en comparación con tener que hacer un join adicional para acceder a estos datos al agregar una tabla adicional.
- Al registrar un cliente se verifica que el numero de documento y el CUIL sean validos.
- Una empresa tiene muchas publicaciones, cada publicación es propia de una única empresa. Consideramos que la publicación tiene los datos propios del evento, aquellos que son comunes a las diferentes fechas en las que podría realizarse (como la dirección y la descripción).
- En el caso de que la empresa quiera hacer una misma publicación en diferentes lugares deberán ingresarse como publicaciones diferentes.
- El grado de la publicación (alto, medio, bajo) determina la comisión y también el orden a la hora de mostrar las diferentes publicaciones del sistema.
- Los espectáculos son las diferentes “instancias” de cada publicación. Contiene la publicación asociada, y tanto la fecha de publicación como la fecha en la que se desarrollaría el evento. Esto es en parte para facilitar la generación de un mismo evento en diferentes fechas. Además, cada espectáculo tiene su propio estado de publicación, que puede ser “Borrador”, “Publicada” o “Finalizada”. Esto es así porque los Espectáculos deben poder finalizarse manualmente, cuando se agotan las entradas, y luego de la fecha del evento.
- Las publicaciones siempre tienen las mismas ubicaciones, a su vez cada espectáculo tiene esas mismas ubicaciones, pero se repiten N veces siendo N la cantidad de espectáculos dentro de esa publicación. Aparece otra relación muchos a muchos y creamos la tabla intermedia UbicacionXEspectaculo.
- Para ingresar las ubicaciones en principio solicitamos el ingreso del precio y la cantidad total de las ubicaciones y una descripción (tipo de asiento).

Se verifica si la descripción ya existe en la tabla TiposDeUbicaciones, de existir se le asigna a la nueva ubicación ese id. De no existir, se crea un nuevo registro. Esto es común a todas las ubicaciones

Dentro de las variantes propuestas por los datos de la tabla maestra, limitamos las posibilidades a 2 casos puntuales:

- Ubicaciones numeradas: Que tienen un número de asiento y un número de fila.
- Ubicaciones sin numerar: Simplemente se indica la cantidad de entradas, y son consideradas como equivalentes entre sí.

Lo importante es que se pueden combinar ambas alternativas, una empresa puede generar una publicación que, por ejemplo, tenga tanto ubicaciones numeradas como sin numerar.

- UbicacionXEspectaculo, además de vincular ambas tablas, representa la entrada individual de una ubicación en un espectáculo particular. Tiene una referencia a una compra, de ser NULL significa que aún no ha sido comprada, caso contrario fue vendida (y se la puede facturar).
- Una compra representa la compra de un cliente a una o muchas UbicacionesXEspectaculo, pagada a través de un medio de pago y en una determinada fecha. También tiene pre calculado el importe.
- En algún momento se factura a las diferentes empresas, se buscan los espectáculos finalizados de las empresas, las UbicacionesXEspectaculos asociadas que tengan una compra y de esas compras, aquellas que no tengan facturas, pasan a estar facturadas.
- Un cliente tiene muchos medios de pago, la descripción tiene 2 posibilidades: efectivo o tarjeta. Efectivo representaría el medio empleado en el modelo anterior, mientras que a partir de ahora, al existir únicamente la posibilidad de pago electrónico, deberá asociarse una tarjeta. Se almacena el número de la tarjeta y el titular de la misma
- Cada cliente tiene asociados varios puntos, los cuales tienen una cantidad y una fecha de vencimiento. Consideramos que se gasten siempre los puntos más cercanos al vencimiento.
- En el momento en que un cliente quiere ingresar, en el login se verifica que posea todos sus datos obligatorios, en caso de no tenerlos se abrirá una pantallita para que pueda completarlos ya que podrían ser necesarios para las operaciones que este quiera realizar.
- Con respecto a los roles definimos que tanto el cliente como la empresa solo podrán tener sus respectivas funcionalidades, una empresa no podrá ser cliente ni viceversa. En el único caso que tendremos todas las funcionalidades es en el administrador a quien se le asignarán desde un principio por definición por lo que una vez que ingrese éste en el login directamente podrá elegir entre todas las funcionalidades, no deberá ingresar en la selección del rol que le corresponde, ya que hay un rol específicamente definido para éste que incluye todo lo que tiene permitido hacer.
- Para las compras efectuadas y facturadas antes de la migración no tenemos forma de saber el porcentaje de comisión. Por lo cual el campo comisión queda en null.
- A las publicaciones creadas antes de la migración se les asigna el grado de la publicación bajo. Esto es porque estas publicaciones se encuentran a la venta y todas las publicaciones que aparezcan como “Publicadas” deben tener un grado asignado.
- Con respecto a los medios de pago: consideramos que no es una buena práctica guardar el código de seguridad de las tarjetas en nuestra base de datos. Si el sistema se implementase realmente, se debería hacer una verificación con un componente externo. En el trabajo está el lugar donde se verificaría pero siempre que se haya completado con un código de seguridad la verificación da verdadero.

- La carga de datos de una publicación (y también su edición) es lenta, esto es porque se deben insertar muchas filas en la base de datos. Para que el usuario no piense que el programa “se quedó colgado” pusimos una progress bar que muestra el progreso.
- Las compras pueden ser facturadas de a partes. Para otorgarle flexibilidad al administrador, este puede elegir facturar solo algunas entradas de una compra.
- Las tablas Factura e ItemFactura presentan un esquema Maestro-detalle. Factura es el maestro, contiene información general como a quien se le factura y la fecha de facturación. Mientras que ItemFactura es el detalle, cada ítem de una factura contiene información específica de un tipo de asiento que fue facturado incluyendo la cantidad, la comisión y el subtotal.
- Las publicaciones anteriores a la migración no tienen información sobre cuál es su rubro, por lo que decidimos que el campo id_rubro quede en null para estos casos.
- Al momento de crear un usuario puede suceder que haya campos que no se puedan repetir, para estos casos la verificación se realiza cuando se quiere guardar el usuario. Si se llegasen a encontrar campos repetidos se le informa al usuario cuáles podrían ser para que los cambie.
- El top 5 de Empresas con Localidades No Vendidas es filtrado por fecha y grado y ordenado por cantidad de localidades no vendidas. Ya que así tendremos el verdadero TOP 5 sabiendo cuales son las 5 empresas con mayor cantidad de localidades no vendidas, lo que no nos pasaba cuando lo probamos ordenando por fecha y grado.
- En el ABM Rubro comenzamos la implementación permitiendo que a partir del id de la publicación se conozca el rubro de ésta, pero por cuestiones de tiempo y prioridades no llegamos a implementar la parte de la modificación aunque aparece la opción en la pantalla.
- Tomamos como decisión pedirle a todos los usuarios que provienen de la migración renovar usuario y contraseña al querer iniciar sesión, ellos podrán mantener sus datos viejos solo deberán reescribirlos y confirmarlos en la pantalla cuando se les solicite.
- En el momento de la compra cuando se le pide al usuario que seleccione qué ubicación desea, se mostrará el tipo de ubicación junto al precio, ya que éste puede variar indistintamente del tipo de ubicación. Lo que nosotros preferimos es que el cliente conozca el precio antes de seleccionar cual es la ubicación que prefiere, ya que si no puede pasar que, por motivo del precio luego quiera modificar el asiento varias veces hasta encontrar el precio justo para el tipo de asiento que buscaba.

Desarrollo del TP

Para el desarrollo del TP, el primer paso fue diseñar el DER. Lo realizamos en base a reiteradas lecturas del enunciado y comparando las entidades requeridas por el mismo con los datos presentes en la Tabla Maestra, para encontrar el modelo que cumpla con los requerimientos y mejor se adapte a los datos preexistentes.

En segundo lugar, migramos todas las columnas existentes en la Tabla Maestra, dejando en NULL aquellos campos inexistentes, como por ejemplo el CUIL de los clientes e información de los Medios de Pago.

También insertamos información correspondiente a los datos migrados para que se mantengan consistentes con nuestro modelo, como el Total de las Compras (que calculamos) y el grado de la publicación (bajo para todas las publicaciones). De la misma manera, utilizamos información de las tablas para modelar el Sistema de Puntos, el cual era uno de los requerimientos del enunciado, y no se encontraba presente en el modelo original.

Para mostrar los datos obtenidos de las queries en las distintas pantallas de nuestra aplicación, creamos clases en Visual Studio que se correspondieran con las filas retornadas por estas consultas. Por ejemplo, creamos la clase "Factura" con tres atributos, "Empresa", "FechaDeFacturación" e "ImporteTotal", los cuales se corresponden con el retorno de la consulta a la base de datos para obtener las facturas. Gracias a estos objetos, pudimos utilizar métodos de C# que nos facilitaron la representación de la información en las distintas Tablas, ComboBoxes, etc.

Inicialmente, dividimos el trabajo en dos partes: el diseño de las pantallas, y la migración de la base y la creación de las consultas necesarias a la misma, las cuales se iban creando a partir de la lectura del enunciado, y de las sugerencias de las personas encargadas del diseño de las pantallas, que podían visualizar más claramente la información que debía ser mostrada.

Para las consultas, adoptamos la estrategia de escribir la menor cantidad de código SQL en Visual Studio, ya que al tratarse de strings, era más propenso a errores de typeo, y por el contrario, creamos Stored Procedures para las mismas. Luego, estos SPs podían ser llamados desde Visual Studio. Así, logramos desacoplar lo más posible las consultas a la base del desarrollo de la interfaz gráfica, y pudimos tener bien organizado nuestro código SQL.

Luego, nos encargamos conjuntamente de la integración de las pantallas con las consultas a la base. En este proceso, fueron surgiendo nuevas consultas y triggers a desarrollar, y modificaciones a realizar a los ya existentes.

Finalmente, nos dedicamos también conjuntamente al testeo del TP. En general, algún integrante se encargaba de poner a prueba la aplicación, mientras el resto realizaba las correcciones que iban siendo necesarias. Paralelamente a esta última etapa, fuimos redactando la Estrategia, dejando constancia de las características más relevantes de nuestro modelo.