Trabajo práctico de Gestión de Datos

CUPONETE

GRUPO_N (Número de grupo 35)

Alumnos

- *Juan Contardo* 1111437
- *Nicolás Nimis* 1226289
- *Carlos Rafellini* 1285774
- *Nahuel Branda* 1409487

INDICE

Definiciones

A continuación, detallamos las definiciones y asunciones que establecimos para salvar inconsistencias e indefiniciones del enunciado, de modo que se pueda entender el funcionamiento final de la aplicación y la forma de realizar las tareas solicitadas.

Usabilidad

La aplicación se ejecuta por defecto en modo maximizada. La primer vista que se tiene es la del Login, donde el usuario puede identificarse o registrarse según lo estipulado en el enunciado del TP. En caso de que se registre y el registro sea exitoso, la aplicación realizará automáticamente el logueo del usuario registrado.
Una vez identificado el usuario, podrá visualizar el menú con los elementos sobre los que tenga permiso y el contenedor principal de ventanas.
Cuando el usuario navegue a cualquier elemento del menú, se cargará la vista correspondiente en el contenedor principal de la aplicación. Dicha vista se mantiene en memoria, de modo que si luego se navega a alguna otra vista y se vuelve a la anterior se cargará el formulario anterior de modo que no se pierdan los datos y pueda volverse a lo que se estaba haciendo anteriormente.

Por lo general, las vistas se componen de tres partes:
- Grilla: muestra los datos correspondiente a esa vista de una forma amigable al usuario, para poder realizar sobre los elementos que la compongan las acciones posibles según el estado actual de la vista.

- Botonera: esta parte de la vista contiene a los controles que ejecutarán las acciones posibles sobre los elementos que se están viendo. A menos que no sea necesario, los controles de la botonera trabajarán con el elemento seleccionado de la grilla (esto es aplicable a las acciones de Modificar y Eliminar, sobre todo).
En caso de que la vista actual sea una grilla, se observará hacia la derecha la cantidad de resultados que muestran actualmente.

- Panel de Filtrado: en los casos especificados, se encuentra un panel con los filtros disponibles para filtrar la información mostrada en la grilla. Dichos filtros se diseñan en base al requerimiento y la guía de ABMs.

Sobre los Procesos

Rol por defecto

Para el caso de baja de un rol, a los usuarios que tengan dicho rol se les asignará como rol el 'Rol por Defecto', el cual no tendrá ningún permiso sobre la aplicación. Este rol se caracteriza por no ser editable ni eliminable.

Perfiles de usuario
Se crearán los perfiles de usuario indicados en el TP: Administrador, Proveedor, Cliente. Estos perfiles se asociarán a los usuarios al momento del alta de usuarios y cada perfil tendrá un rol asociado por defecto. De modo de no generar inconsistencias con el ABM de roles, los usuarios podrán registrarse con un **Perfil** determinado (para el caso será Proveedor o Cliente). La asignación de rol será la del rol por defecto que tenga el TP.

Sobre el administrador
El administrador se caracteriza por tener el rol de administrador, el cual tiene permisos sobre toda la aplicación. De todas maneras, cuando se comporte como proveedor tendrá acceso sobre su propia información (sus cupones creados, por ejemplo). De modo que no podrá registrar consumo de otros proveedores ni pedir devoluciones de otros clientes.

Sobre los clientes
La entidad modelada de Cliente deriva de la de Usuario (ver más adelante el detalle de la aplicación .NET), por lo que el ID de la tabla Ciente se relaciona con el ID de la tabla Usuario. De esta forma podemos hacer que el administrador pueda tener todos los perfiles del sistema, pudiendo relacionar a un usuario con cualquier perfil. De todos modos, no es posible desde el sistema que un usuario tenga varios perfiles ni cambiar el perfil de un usuario una vez establecido, según lo indicado en el enunciado.
Se definió 10 como valor Default en la tabla Clientes para el campo saldo, con lo que cada nuevo cliente tendrá $10 de crédito de regalo para cumplir con el requerimiento del enunciado.
Hay una constraints UNIQUE en Usuario para datos únicos como el Username, para que no se repita dicho dato al igual que otros campos de la entidad.

Sobre los Proveedores
Hay constraints de UNIQUE en Proveedor para el CUIT y la Razón Social, para que no se repitan en el sistema cumpliendo con el requerimiento del enunciado.

Entidades Genéricas / tabla DetalleEntidad
Se separan los datos generales de una entidad (entendiendo como entidad a un Cliente, Proveedor, o cualquier otra entidad que figure en el sistema) en una tabla DetalleEntidad que contendrá los datos genéricos de todas las diferentes entidades.
Se determinó que la constraint de Email que en principio tomamos como NOT NULL, debe permitir valores nulos debido a que en la tabla maestra los proveedores no poseen dicho campo, con lo cual tomamos que es un nuevo requerimiento funcional del TP.
Hay una constraint de UNIQUE en DetalleEntidad para el teléfono, para que no se repitan.

Saldos de los clientes

El saldo actual de los clientes se manejará con Triggers. De modo que siempre que se detecte un insert en un pago, compra, gift card o devolución se actualizará el saldo del cliente con estos triggers.
Hay una restricción CHECK para garantizar que el saldo de un cliente sea siempre mayor o igual a cero, en caso de que cualquier operación intente establecer un saldo negativo se disparará una excepción. De este modo mantenemos la consistencia de los datos en nuestro esquema.

Sobre los Cupones
Se define como código de cupón a una secuencia de 10 letras mayúsculas y número de compra al número de compra realizada sobre un cupón, de modo que el código de compra es la concatenación del código de cupón con el número de compra.
Al crear un cupón se genera un código aleatorio. Se verifica siempre que dicho código no exista en la base de datos para no repetirlo. Para esta verificación se creó un índice UNIQUE en la tabla Cupon para el campo Codigo.
Se toman como fechas del cupón en la tabla:
- FechaPublicacion: la fecha a partir de la que se podrá comprar el cupón
- FechaVigencia: la fecha hasta la que se podrá comprar el cupón
- FechaVencimiento: la fecha hasta la que se podrá consumir el cupón

Se toman como precios del cupón en la tabla:
- Precio: el precio al que se compra el cupón. Lo que termina pagando el usuario
- PrecioOriginal: el precio sin el descuento. Es el precio ficticio, su función es sólo informativa

El cupón tiene un atributo 'Publicado', el cual indica si está aprobado o no por el administrador para poder ser visto por los usuarios.
Si bien se encontraron inconsistencias en las compras de cupones (varias compras con el mismo código de compra) se decidió importar dichos movimientos con el código original, no salvando la inconsistencia. Esta decisión se debe a que la compra de un usuario se identifica con el código, de modo que no vamos a modificar dicho valor ya que se perdería la referencia por parte del usuario.
Lo que podemos garantizar es que los códigos generados luego de la migración no tendrán dicha inconsistencia.

Compra de cupones
No se mostrarán a los clientes en los listados los cupones que no tengan stock para ser comprados.

Consumo de cupones
No se mostrarán a los proveedores en los listados para consumir compras las que estén vencidas o hayan sido devueltas o ya consumidas.

Facturación a proveedores

Se deberá seleccionar al proveedor y el rango de fechas para mostrar los cupones consumidos facturables, por defecto dicho rango será desde la fecha actual 6 meses para atrás. No se mostrarán cupones no consumidos o cupones ya facturados.

Parametrización

Fecha del sistema

Se define una entrada en el archivo App.Config para tomar la fecha del sistema. Dicha entrada se encuentra en el path configuration/appSettings/add(key="FechaSistema") y en el valor se encuentra la fecha en cuestión con el formato yyyy-mm-dd.

Conexión a la DB

Se define el connection string que utilizará la aplicación en el archivo App.config. Dicha entrada se encuentra en el path configuration/connectionStrings/add(key=”GrouponConnectionString”)

Criterios de migración de datos

Vamos a detallar a continuación los criterios utilizados para las nuevas funcionalidades especificadas en el enunciado del TP, las cuales no existían en la base de datos a importar:

Creación de los usuarios
Los usuarios se crearon teniendo en cuenta cuales son las restricciones aplicables de ciertos campos tanto para el cliente como para el proveedor. La contraseña por defecto será w23e para todos los usuarios, la misma se almacenará encriptada con SHA-256.

Importación de clientes
Se crean usuarios para cada uno de los clientes donde se definió que el nombre del usuario va a ser el número de teléfono del mismo debido a que este valor es único para cada cliente.

Importación de proveedores
Se crean usuarios para cada uno de los proveedores donde se definió que el nombre del usuario va a ser la razón social del mismo debido a que este valor es única para cada cliente. Además como no tienen mail, se los ingresa con NULL en dicho campo.

Importación de cupones
Un nuevo requerimiento pide que los cupones posean un campo donde indica el plazo para retirarlo, el cual vamos a calcular como dos meses a partir de la fecha de publicación.
Otro nuevo requerimiento es que los cupones pueden tener una *CantidadPorUsuario* que nosotros estipulamos que para los cupones importados sean la cantidad total de los cupones posibles a la venta. Luego de analizar los datos de la tabla maestra, determinamos que cada cupón tiene un código único (por ejemplo WCXAPRGOPM) y luego en cada una de sus ventas se identifican agregando al final el número de compra para esa promoción en particular (por ejemplo WCXAPRGOPM1, WCXAPRGOPM2, etc). Aquí se encontraron inconsistencias debido que existe para un mismo cupón compras de diferentes proveedores con el mismo código, lo cual se detalló en el punto anterior de definiciones.

Importación de devoluciones de cupones
Otro de los nuevos requerimientos es incluir un motivo por la devolución. Se ingresa para todos los cupones importados la leyenda *'Devolucion en sistema previo'* en dicho campo, ya que no se encuentran datos en el esquema anterior que correspondan a esta columna.

Importación de las ciudades para los cupones
Como no se detallan las ciudades en las cuales se publicaron los cupones, elegimos tomar la ciudad del proveedor como valor por defecto.

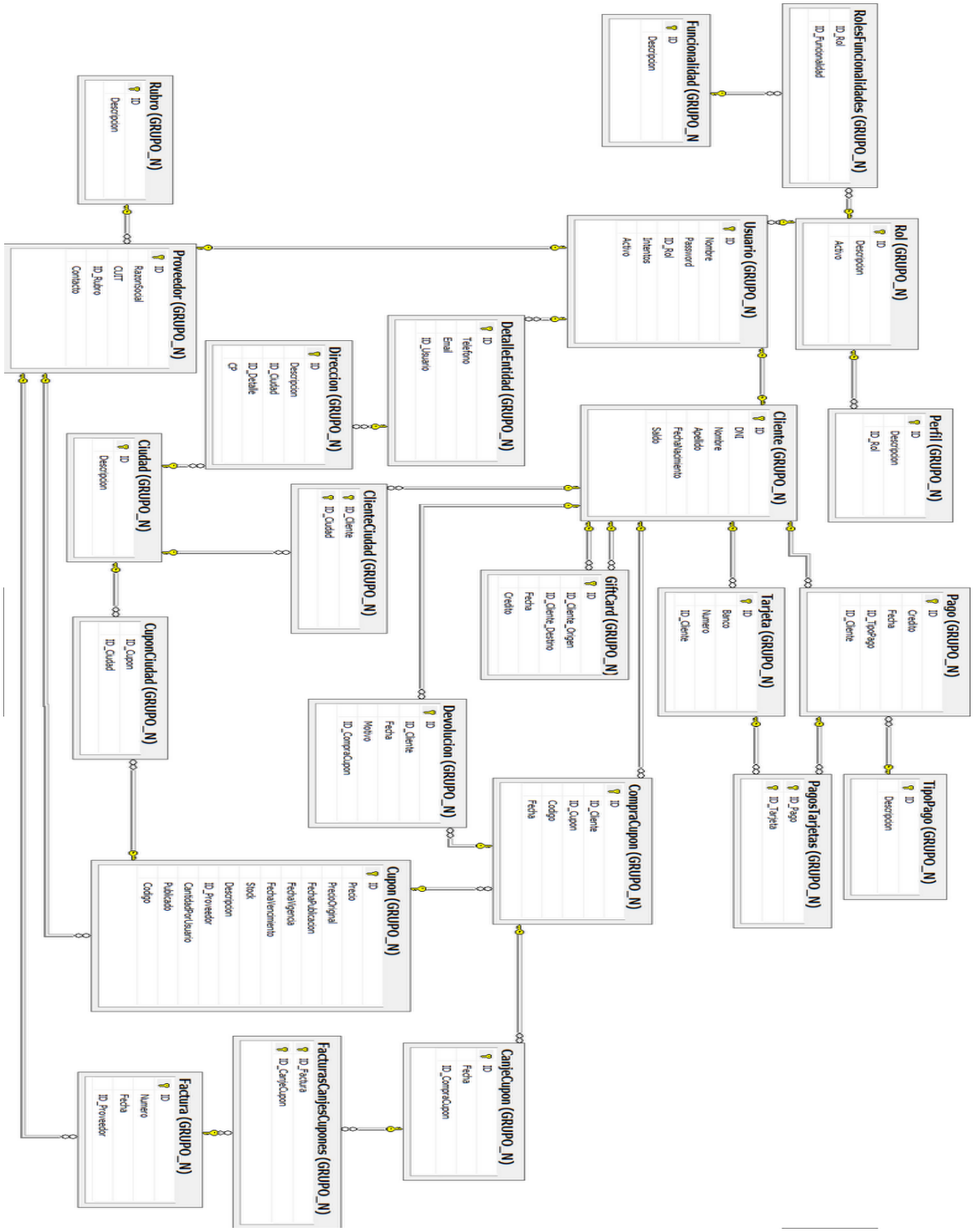
Importación de los pagos
En la importación de los pagos se toman en cuenta los registros de la tabla maestra que contienen valores no nulos para los campos `Carga_Credito y Carga_Fecha`, cada pago estará relacionado con un tipo de pago, un usuario y además los pagos que hayan sido hechos con tarjeta serán relacionados con la tarjeta mediante la tabla PagosTarjetas.

Importación de las tarjetas
Inmediatamente después de la importación de los pagos se importan las tarjetas. Como la tabla del sistema anterior no guardaba el banco ni el número correspondiente a la tarjeta se tomaron los criterios siguientes:
El banco al que corresponde la tajeta contendrá la cadena `'Banco de importacion previa'.`
Y el número de tarjeta conendrá la cadena `'Numero de importacion previa'.`

Importación de las pagos-tarjetas
Una vez ingresado los pagos y las tarjetas, se procede a cargar las relaciones pagos-tarjetas para ello se utilizan los pagos que fueron hechos con tarjeta, excluyendo cualquier otro tipo de pago.

Importación de las GiftCards
La importación de las GiftCards se realizan una vez que todos los clientes están cargados, ya que la carga de una requiere que tanto el ciente de origen como el de destino tengan su ID.
Una entrada de la tabla de GiftCard contendrá el cliente de origen, el de destino, la fecha en que se realiza la transferencia y el saldo en cuestión.

Diagrama de entidad - relación



Modelo de Datos

A continuación se especifican funcionalmente cada una de las tablas más relevantes del sistema. Como premisa, los campos IDs de las tablas serán enteros autoincrementales a menos que se especifique lo contrario.

Usuario
Es toda entidad que ingresa al sistema. Contiene información de su nombre, si se encuentra activo o no, la cantidad de ingresos fallidos al sistema y un ID que constituye como clave primaria. Además contiene una clave foránea llamada ID_Rol que relaciona a un usuario con un rol determinado.
Esta tabla contiene un index IX_Usuario (UNIQUE NONCLUSTERED), el cual nos asegura que no hayan nombres de usuarios repetidos para mantener la consistencia de datos.

Rol
Contiene el nombre e identificador de los únicos cuatro roles posibles hasta el momento en el sistema: Rol por defecto, Administrador, Proveedor y Cliente. Cada uno se va a relacionar con las funcionalidades a las que puede acceder en el sistema mediante otra tabla. Por otro lado, se indica si está activo o no para la implementación de bajas lógicas. El campo ID constituye la clave primaria del mismo.

Funcionalidad
Contiene la descripción de cada funcionalidad que se puede realizar con la aplicación Desktop de este trabajo práctico, y además un campo ID siendo ésta su clave primaria. Las funcionalidades que se encuentren aquí registradas funcionarán como permisos para los roles asignados en la aplicación.

RolesFuncionalidades
Tabla que permite relacionar cada Rol con las funcionalidades de la aplicación a las que puede acceder. ID_Funcionalidad y ID_Rol son los campos que terminan constituyendo la combinación de ambas tablas en clave primaria de esta tabla.

DetalleEntidad
Esta tabla contiene los campos comunes tanto de proveedores como clientes. Para mantener la coherencia de los datos los clientes y proveedores en su primary key tienen un campo ID que es el mismo que la tabla usuario. Por lo tanto esta tabla tiene una foreign key ID_Usuario para relacionar el detalle con un usuario para que luego se relacione según corresponda a un cliente/proveedor.
Para garantizar que el número telefónico sea único como lo indica el enunciado, se creó el índice IX_DETALLEENTIDAD_TELEFONO como UNIQUE.

Direccion
Esta tabla extiende el detalle de una entidad, persistiendo aquí los datos de la dirección de dicha entidad. Contiene la descripción de la dirección como texto (en un nvarchar(255)), el código postal y una relación con la ciudad a la que pertenece la dirección (FK ID_Ciudad), además de la relación con el detalle al que corresponde esta dirección (FK ID_Detalle). El código postal podrá ser nulo ya que no es un dato existente en el esquema actual, no así los otros datos de esta entidad.

Cliente
Tiene la información de cada cliente, con sus datos personales y su saldo. ID es la clave primaria que como ya mencionamos antes es el mismo ID de la tabla Usuarios, para generalizar esta entidad y poder reutilizar módulos del sistema.
Esta tabla contiene la restricción CK_Cliente_Saldo para garantizar que el saldo sea siempre positivo y mantener así la consistencia de datos en nuestro esquema.

ClienteCiudad
Esta tabla permite relacionar a los clientes con sus ciudades de preferencia. Se registran qué ciudades (FK ID_Ciudad) prefieren los clientes para visualizar ofertas (FK ID_Cliente)

Proveedor
Tiene la información de cada cliente, con sus datos personales e ID_Rubro que es una clave foránea contra la PK de la tabla rubro que indica a qué se dedica el mismo (relación Proveedor - Rubro)
ID es la clave primaria que como ya mencionamos antes es el mismo ID de la tabla Usuarios. Para garantizar que el CUIT y la Razón Social sean únicos se crearon dos índices UNIQUE-NONCLUSTERED: IX_Proveedor_CUIT y IX_Proveedor_RSocial.

Factura
Contiene un campo ID autonumérico como PK, el número y monto de la misma. Por otra parte posee una FK ID_Proveedor que relaciona a cada factura con un proveedor. El detalle de lo que contiene la factura se encuentra en la tabla FacturasCanjesCupones.

FacturasCanjesCupones
Relaciona una factura con múltiples cupones canjeados que figuran en la tabla CanjeCupon mediante el campo ID_CanjeCupon (FK). La combinación de ambos conforma una PK. Aquí se encuentran los cupones canjeados que conforman una factura, relacionando ambas entidades en esta tabla.

Cupon
Contiene toda la información de una publicación que realiza un usuario con el perfil de proveedor en nuestro sistema para conformar una promoción. A su vez tiene una FK ID_Proveedor para poder relacionar un cupón con un ID de proveedor en particular. Contiene todos los datos de un cupón disponible para la visualización de los usuarios. La publicación

de los cupones depende del campo Publicado, el cual se establece en ‘1’ cuando algún administrador publica el cupón para que pueda ser comprado por los clientes.
Esta tabla tiene tres índices:
- IX_Cupon_Codigo: es un índice UNIQUE NONCLUSTERED, para garantizar que los códigos de cupón sean únicos no repetibles.
- IX_Cupon_FechaPublicacion y IX_Cupon_FechaVigencia: la necesidad de crear estos índices deriva de las búsquedas de cupones por estos campos. La velocidad de búsqueda se vería muy afectada sin estos índices.

Ciudad
En esta tabla se centralizan todas las ciudades que utiliza en sistema. La misma tiene un campo ID como PK y la descripción, donde figura el nombre de la ciudad.

CuponCiudad
Tabla que relaciona un cupón con las ciudades donde se van a encontrar disponible para su publicación. Relaciona los IDs de las tablas Cupon y Ciudad, estableciendo de esta manera las ciudades en las que estará disponible el cupón.

GiftCard
Tabla donde figuran las giftcard que un cliente le envía a otro con un monto que figura en el campo crédito. A su vez tienen dos FK que se relacionan con el ID del cliente llamadas ID_Cliente_Origen e ID_Cliente_Destino.
Esta tabla tiene asociada el trigger GiftCardInserted, índice For Insert que incrementa el crédito del destinatario de la gift card y reduce el crédito de quien la compra.

Pago
En esta tabla figura el monto de crédito cargado y la fecha de carga. Se relaciona con la tabla cliente (ID_Cliente) y la tabla TipoPago (ID_TipoPago) para establecer quién realizó el pago y con qué modalidad.
Esta tabla tiene asociada el trigger PagoInserted, índice For Insert que incrementa el crédito de quien realizó el pago según corresponda.

TipoPago
Contiene los medios de pago disponibles en el sistema. Esta tabla se relacionará con la de Pagos para indicar la modalidad del pago realizado.

Tarjeta
Contiene los datos de una tarjeta de crédito para pagos realizados por un cliente. Tiene un campo ID que es clave primaria y una clave foránea que relaciona a esta tarjeta con un cliente llamado ID_Cliente. En un futuro podría mejorarse el sistema, permitiendo que a la hora de realizar un pago con tarjeta se seleccione una de las tarjetas disponibles para realizar el pago.

PagosTarjetas

Relaciona un pago con una tarjeta en el caso que fuese del tipo determinado. Contiene los campos necesarios para relacionar el pago de un cliente con la tarjeta correspondiente.

CompraCupon
Contiene los datos de las compras realizadas con los clientes. Esta tabla relaciona a un cliente con un cupón, mientras que contiene también el código otorgado para el cupón en cuestión y la fecha en la que se realizó la compra.
Esta tabla tiene asociada el trigger CompraCuponInserted, índice For Insert que debita el valor del cupón al saldo del cliente que realizó la compra, además de actualizar el stock de cupones disponibles restando uno a la cantidad.

Devolucion
En esta tabla se registran las devoluciones de las compras de cupones realizadas por los usuarios. Contiene las relaciones con los clientes (FK ID_Cliente) y compras de cupones (FK ID_CompraCupon), de modo que un registro en esta tabla indicará que esa compra ha sido devuelta. Cada registro tendrá la fecha de devolución y el motivo de la devolución ingresado por el usuario (dato no obligatorio).
Esta tabla tiene asociada el trigger DevolucionInserted, índice For Insert que acredita el valor del cupón al saldo del cliente que realizó la devolución, además de actualizar el stock de cupones disponibles incrementando en 1.

CanjeCupon
Contiene los registros de los cupones que han sido canjeados (o consumidos), según lo indiquen los proveedores. Aquí se establecen qué compras (FK ID_CompraCupon) han sido canjeadas en qué fecha.
Esta tabla posee un índice IX_CanjeCupon_Fecha. La necesidad de creación de este índice se debe a la demora detectada en la generación de facturas. Al tener que consultar los cupones consumidos en un rango de fechas, dicha consulta se ejecuta con una diferencia de performance notable agregando este índice a la tabla.

Aplicación .NET

Para el desarrolo de dicha aplicación, diseñamos una arquitectura básica para poder desarrollar sobre ella nuestros módulos. Dicha arquitectura se compone de 4 componentes:
- GrouponDesktop
- DataAccess
- GrouponDesktop.Common
- GrouponDesktop.Business

Lo que buscamos con este diseño es obtener un menor acoplamiento entre las capas, para poder colaborar en el desarrollo y mantenimiento de la aplicación de forma conjunta y lograr un código de calidad y más entendible. Detallamos a continuación cada uno de ellos.

Componentes

GrouponDesktop

Este componente es en donde se encuentra la UI desarrollada sobre Winforms para el usuario. Contiene un Core que le brinda servicios para gestionar las diferentes vistas y datos de sesión del usuario.

DataAccess

Esta capa es la encargada de gestionar las conexiones a la DB. Brinda los componentes necesarios para poder establecer una conexión y realizar consultas, abstrayendo a las capas superiores de estas tareas. Provee servicios de administración de transacciones y ejecución de consultas a las capas superiores, además de la gestión de conexiones.
Como estrategia, decidimos utilizar Stored Procedures en toda la aplicación para evitar tener código SQL embebido en el lado de .NET. De esta manera, podemos mantener la lógica del motor SQL desacoplada de la lógica de negocios y obtenemos una mejor mantenibilidad y seguridad evitando SQL Injection.

GrouponDesktop.Common

Este componente contiene las definiciones de entidades de negocio que se utilizarán en la aplicación. La idea es realizar una solución orientada a objetos, de modo que las entidades que representen a las intervinientes en los procesos se encuentran modeladas en esta capa.

GrouponDesktop.Business

Esta capa se encarga de aislar las operaciones CRUD de la lógica de negocios, de modo que provee servicios a la UI (contenida en GrouponDesktop). En este componente se encuentran los managers de las entidades de negocio, desacoplando dichas tareas de entre la capa superior y la de acceso a datos.

Manejo de Excepciones

Dentro del Core de la solución se creó un ExceptionHandler para el manejo de excepciones no controladas en la aplicación, de modo que toda excepción no controlada será manejada por este componente.
El caso especial es para las excepciones generadas por la capa de acceso a datos, de modo que hay un manejador especial para las SqlExceptions. Desde la aplicación, se define un mensaje de error para mostrar al usuario en un diccionario que tiene como entrada el nombre de la constraint violada. De esta forma, si se captura una SqlException se busca en el mensaje de la excepción el nombre de la constraint violada y se le muestra al usuario el mensaje correspondiente.

Seguridad

Se toma como 'Funcionalidad' del sistema a los módulos sobre los que puede tener acceso un usuario. De esta forma, un rol tendrá acceso a los módulos según las funcionalidades asignadas.
Las funcionalidades se encuentran catalogadas en el siguiente enum (dentro de GrouponDesktop.Common):

```
public enum Functionalities
{
    AdministrarClientes,
    AdministrarProveedores,
    AdministrarCupones,
    CargarCredito,
    CrearCupon,
    ComprarCupones,
    GiftCards,
    Facturar,
    VerHistorialCupones,
    ListarEstadisticas,
    PedirDevoluciones,
    PublicarCupones,
    RegistrarConsumoCupones,
    AdministrarRoles
}
```

Estas funcionalidades están populadas también en la base de datos en la tabla Funcionalidad. Los componentes Core de GrouponDesktop obtendrán la configuración de funcionalidades para el rol del usuario logueado, de modo de listar en el menú sólo los módulos a los que el usuario tiene acceso.

Para definir el permiso requerido para una vista en particular, se debe especificar el atributo PermissionRequired en la clase que define a la vista, indicando el tipo de permiso que se requiere para visualizar dicho formulario. Por ejemplo, para visualizar el formulario de proveedores se requieren permisos sobre la funcionalidad de AdministrarProveedores, con lo que el atributo declarado quedaría de esta manera:
[PermissionRequired(Functionalities.AdministrarProveedores)]

En caso de que un formulario no sea navegable desde el menú, se indicará dicha necesidad con el atributo [NonNavigable].