UNIVERSIDAD TECNOLÓGICA NACIONAL
F.R.B.A.
INGENIERIA EN SISTEMAS DE INFORMACION

GESTION DE DATOS

ESTRATEGIA DE TRABAJO

FRBA HOTEL

FECHA: 11/11/2014

VERSIÓN: 1.0

COMPUMUNDO_HIPER_MEGA_RED

NOMBRE DEL INTEGRANTE	LEGAJO	E-MAIL
GOLOB, Víctor	137.977-0	victorgolob@gmail.com
GEREZ, Claudio	142.558-4	claudiomgerez@gmail.com
FERRARO, Tomás Ignacio	146.518-1	tetferraro@gmail.com
LEDESMA, Nicolás	147.062-0	n.e.ledesma@hotmail.com

BASE DE DATOS

CREACIÓN DEL NUEVO MODELO DE DATOS Y MIGRACIÓN DE LOS DATOS EXISTENTES

DIAGRAMAS

Partiendo de la **Tabla Maestra** y la documentación del enunciado, se realizó un **modelo desnormalizado** que cumpla las reglas de negocio actuales. Con dicho modelo, se normalizó el mismo hasta **Forma Normal 3** y luego se fueron tomando ciertos recaudos de duplicidad o simplificación de la información. Como ejemplo, uno de ellos fue el utilizar una tabla para los ítems de la factura; que aunque sean consumibles, es conveniente tenerlos separados en una tabla particular como resguardo de información y para evitar cálculos al momento de realizar consultas.

El Diagrama de Entidad-Relación (DER) que representa al modelo, se encuentra adjunto a este documento bajo el nombre de "**DER.png**".

MIGRACIÓN DE DATOS

Una vez realizado el nuevo modelo, se analizaron los datos del modelo anterior para poder realizar la migración de los mismos al nuevo sistema. Debido a ciertas **incompatibilidades** entre ambos modelos, se encontraron distintos problemas. A continuación se plantearán los mismos y las **soluciones** tomadas para realizar efectivamente la migración de los datos.

1. **Reservas y Facturas Inválidas**: Se encontraron varias entradas con valores erróneos en las reservas, así como también se encontraron varias entradas de facturas cuyos totales se encontraban mal calculados. Para evitar problemas de estabilidad en el sistema, se los delegó al **sistema de tablas para gestionar las entradas inválidas** y así evitar la pérdida de los datos migrados.

2. **Formato de Emails erróneo**: Todos los emails se encontraban con formatos inválidos para el estándar de la RFC 6531, **conteniendo mayúsculas, espacios y caracteres especiales inválidos** como: é, á, í, ú, ó, ü, etc. Para evitar problemas con el aplicativo, ya que utiliza la clase MailAddress, se alteran los emails con un **trigger** luego de cada insert o update de la siguiente manera:
    - Las mayúsculas se pasan a minúsculas.
    - Los espacios designan el inicio de la cadena.
    - Las vocales con tildes y/o diéresis se colocan en su forma normal.

3. **Datos Faltantes**: Debido a nuevas capacidades del modelo, fue necesario agregar **datos por defecto durante la migración**, como es el usuario que realizo las reservas, cancelaciones, estadías y facturas, como el "admin"; locaciones de los clientes (como son todos argentinos se supuso que provenían de la Ciudad Autónoma de Buenos Aires), DNIs para tipos de documentos, las fechas de creaciones de los hoteles, etc.

4. **Nacionalidades**: Para los Clientes, así como para la nacionalidad, se utiliza el **mismo campo** para definir el **país** de domicilio, a fin de homogeneizar. Además, todos los datos de los campos de nacionalidad, **se cambiaron a forma de país**, ya que el aplicativo posee una lista de todos los países existentes y admisibles.

DECISIONES DE DISEÑO

Junto con la creación del nuevo modelo se tomaron en cuenta ciertas decisiones a la hora de realizar el diseño del sistema. Las mismas fueron:

1. **Inmutabilidad de la Fecha de Creación de un Hotel**: La fecha de creación de los hoteles **no son modificables una vez creadas** debido a que no tiene ningún sentido, más que para subsanar un error. En tal caso, es recomendable realizar la creación del hotel nuevamente y eliminar el erróneo de la Base de Datos.

2. **Validación del Primer Log-in**: La validación del primer Log-in de un Usuario se realiza dentro de la base de datos en vez de la aplicación. Esto permite que sea la Base de Datos la cual decida si modifica o no el password del usuario, ya que por requerimiento "**el password es modificable una vez que el usuario se encuentre logueado**".

3. **Intentos Fallidos de Log-in**: Los intentos fallidos de loggeo se registraran dentro del modelo a fin de realizar una **persistencia** de los mismos, como realizaría un sistema bancario. Esto genera que los mismos no sean volátiles y permite una mayor seguridad.

4. **Interacción con la Aplicación**: Para realizar una capa de abstracción entre nuestro modelo de datos y la aplicación, se crearon varios **Stored Procedures** que serán conocidos por la aplicación **como la interfaz de la Base de Datos.** Esta abstracción **reduce el acoplamiento** entre la aplicación y el modelo de datos, y permite que la aplicación solo conozca qué es lo que necesita y como obtenerlo, sin depender de los nombres de las tablas, sus atributos y relaciones.

5. **Hoteles y Roles**: A fin de simplificar el loggeo y la identificación en el sistema, los **roles** de un usuario son **indistintos** a sus **hoteles**, ya que no tendría sentido práctico aplicable que, por ejemplo, un usuario "Recepcionista" sea "Administrador" en otro Hotel.

TABLAS

Se trató de evitar la creación de tablas de datos Referenciales, debido a que el modelo de datos no posee relevancia en la utilización de los datos, por lo que para el mismo no generan ningún interés. Esto permite que el programador de la Aplicación se encargue de darle relevancia a los datos como el Tipo de Documento, País y Tipo de Cancelación y modifique de acuerdo a la modificación de la Aplicación.

A continuación se detallara el diseño de las tablas por Funcionalidad:

<u>Gestión de Usuarios:</u>

- **Usuarios**: Contiene los usuarios del sistema.
- **Roles**: Contiene los distintos tipos de roles.
- **Roles_X_Usuario**: Relaciona un usuario con uno o más roles.
- **Funcionalidades**: Contiene las funcionalidades del sistema.
- **Funcionalidades_X_Rol**: Relaciona las funcionalidades con los roles.
- **Hoteles_X_Usuarios**: Relaciona los usuarios con los hoteles.
- **Huespedes**: Contiene los huéspedes ingresados en el sistema.

<u>Gestión de Hoteles:</u>

- **Hoteles**: Contiene los hoteles del sistema.

- **Inhabilitaciones**: Contiene las inhabilitaciones de los hoteles.
- **Regímenes**: Contiene los diferentes regímenes de hoteles del sistema.
- **Regímenes_X_Hotel**: Relaciona los hoteles con los regímenes.
- **Habitaciones**: Contiene las habitaciones de los hoteles.
- **Tipo_Habitacion**: Contiene los datos del tipo de habitación.

Gestión de Reservas:

- **Reservas**: Contiene las Reservas del sistema.
- **Detalles_Reserva**: Contiene los detalles de las habitaciones reservadas.
- **CancelacionesReserva**: Contiene las Cancelaciones de las reservas.
- **Estadía**: Contiene la estadía registrada por reserva.
- **Consumibles**: Contiene los consumibles registrados en el sistema.
- **Consumibles_X_Estadia**: Relaciona la estadía con los consumibles registrados.

Gestión de Facturas:

- **Facturas**: Contiene las facturas realizadas.
- **Items_Factura**: Contiene los ítems de las facturas realizadas.

STORED PROCEDURES

Se crearon varios procedures a fin de servir como una interfaz del modelo de datos ante la Aplicación. Debido a esto, por cada tabla principal, se realizaron 4 **procedures básicos: get, insert, delete y update**. Todos poseen las siguientes condiciones:

- Si se ingresan campos como **-1** para los numéricos y **‘’** para los varchar() se los considera como “**elementos neutros**” o nulos.

- Los **procedures delete** son lógicos (pone en 0 su CampoBaja) en su mayoría, a excepción de los hoteles, que genera una inhabilitación, y de otros que son bajas físicas (usualmente correspondientes a las Tablas que realizan la Relación de M:M).

- Los **procedures de insert y update**, en campos no PK o no necesarios, no toman en cuenta los campos ingresados como “elementos neutros”.

- Los **procedures de get** devuelven todos los campos de la tabla si la PK ingresada es un “elemento neutro”.

Además, el sistema posee otros tipos de procedures como:

SP Facturar: Genera una factura para una reserva.

```
@codReserva numeric(18),
@tipoPago   varchar(50),
@codTarjetaCredito   varchar(19) (Se manda el “elemento neutro” en caso de ser pago en efectivo”)
```

SP getHotelesMayorCancelaciones: Obtiene los 5 Hoteles con Mayor Cancelaciones

```
@opcion  numeric(1),
@anio  numeric(4)
```

*Donde Opción es:

1. Enero a Marzo
2. Abril a Junio
3. Julio a Septiembre
4. Octubre a Diciembre
5. Todo el Año

SP getHotelesMayorConsumiblesFacturados: Obtiene los 5 Hoteles con Mayor cantidad de Consumibles Facturados

```
@opcion  numeric(1),
@anio  numeric(4)
```

SP getHotelesMayorDiasInhabilitado: Obtiene los 5 Hoteles con Mayor Tiempo de Inhabilitación acumulado.

```
@opcion  numeric(1),
@anio  numeric(4)
```

SP getHabitacionesMayorCantOcupadas: Obtiene los 5 Hoteles con Mayor Cantidad de Ocupaciones

```
@opcion  numeric(1),
@anio  numeric(4)
```

SP getMejorCliente: Obtiene los 5 Mejores Huespedes

```
@opcion  numeric(1),
@anio  numeric(4)
```

FUNCTIONS

Para evitar la duplicación de código debido a poseer tanta cantidad de procedures, se crearon las siguientes funciones:

```
[totalConsumibles] (@codReserva  numeric(18)) RETURNS numeric(18,2)
```

Realiza el cálculo del subtotal por consumibles para la factura.

```
[precioRegimen] (@codReserva  numeric(18)) RETURNS numeric(3,2)
```

En función de una reserva, devuelve el precio del régimen del hotel.

```
[recargoEstrella] (@codReserva  numeric(18)) RETURNS numeric(2)
```

Realiza el cálculo del recargo por estrellas del hotel.

```
[porcentualHabitacion] (@codReserva numeric(18)) RETURNS numeric(2)
```

En función de una reserva, devuelve el porcentual del tipo de habitación.

```
[get_ultimoNumeroFactura] () RETURNS numeric(18)
```

Retorna el número de la última factura realizada.

```
[get_ultimoCodigoReserva] () RETURNS numeric(18)
```

Retorna el número de la última reserva realizada.

TRIGGERS

Como se mencionó anteriormente, el sistema solo posee un trigger, siendo el mismo para chequear los emails luego de ser ingresados y ajustar su formato.

APLICACIÓN DESKTOP

INTERFAZ GRÁFICA

DISEÑO GENERAL DE LA APLICACION

La Aplicación Desktop está compuesta por 2 (dos) proyectos, la **interfaz gráfica** (FrbaHotel) y **las Clases de Dominio y Datos** (DOM).

El proyecto DOM está compuesto por una sección de **clases de Dominio**, donde se especifican las clases del modelo de negocios**, clases auxiliares** de Documentos, Dirección, PasswordEncoding (encriptación de contraseñas). Además, posee un módulo de persistencia de datos utilizando el patrón **DAO** (Data Access Objects).

PERSISTENCIA DE DATOS

Como se expuso anteriormente, la persistencia de datos se realiza a partir del patrón de diseño DAO. El mismo consiste en **nuclear la conexión con la Base de Datos** en clases estáticas con el prefijo "DAO" y continuando con el nombre de la Clase de Dominio asociada al objeto, como "DAOUsuario". Estas clases "DAO" heredan su capacidad de conexión y comunicación con la Base de Datos de una **clase abstracta** llamada "**SqlConector** ".

Estas clases estáticas poseen métodos para obtener tanto como una **DataTable** como una lista de la clase de dominio; insertar, borrar y actualizar, y algunas clases poseen la capacidad de interactuar con las relaciones M:M. Todos los métodos estáticos hacen referencia a los métodos de comunicación del "SqlConnector" enviando **el nombre del procedure** a ejecutar y pasando por array de objetos los **parámetros** del mismo.

CLASES ESTÁTICAS

Las clases estáticas del sistema se dividen en: las clases de manejo de persistencia de datos; y la clase **Globals**.

La segunda mencionada se encarga de tener todas las **funcionalidades comunes** a las ventanas de la **Interfaz Gráfica**, como para realizar el intercambio de ventanas, utilización del **archivo de configuración** para obtener de la cadena con los datos para conectar con la Base de Datos y la fecha del sistema. Además, posee **atributos estáticos globales**, como un array de cadenas de los nombres de los países existentes.

DECISIONES DE DISEÑO

Para esta parte del sistema se consideraron ciertos aspectos no previstos en el enunciado:

1. Cuando un usuario queda **deshabilitado por 3 intentos fallidos** de Login, esta des habilitación es **permanente** hasta que un administrador lo habilite nuevamente.

2. Al ingresar como **invitado** al sistema, se utilizara un pseudo-usuario llamado "invitado" que será el encargado **representar al agente externo** para realizar las reservas.