**UNIVERSIDAD TECNOLÓGICA NACIONAL**

***FACULTAD REGIONAL BUENOS AIRES***

*Ingeniería en Sistemas de Información*



Gestión de Datos

FRBACOMMERCE

1C – 2014

**Grupo: JJRD**

JONAS CASTEGLIONE 138116-7

DIEGO ALFONSIN 135151-5

RODRIGO ANTONIOLI 137709-7

JUAN URBANO 138538-0

Indice

[DER 3](#_Toc372273679)

[Consideraciones 4](#_Toc372273680)

[Estructuras 5](#_Toc372273681)

[Stored Procedures 6](#_Toc372273682)

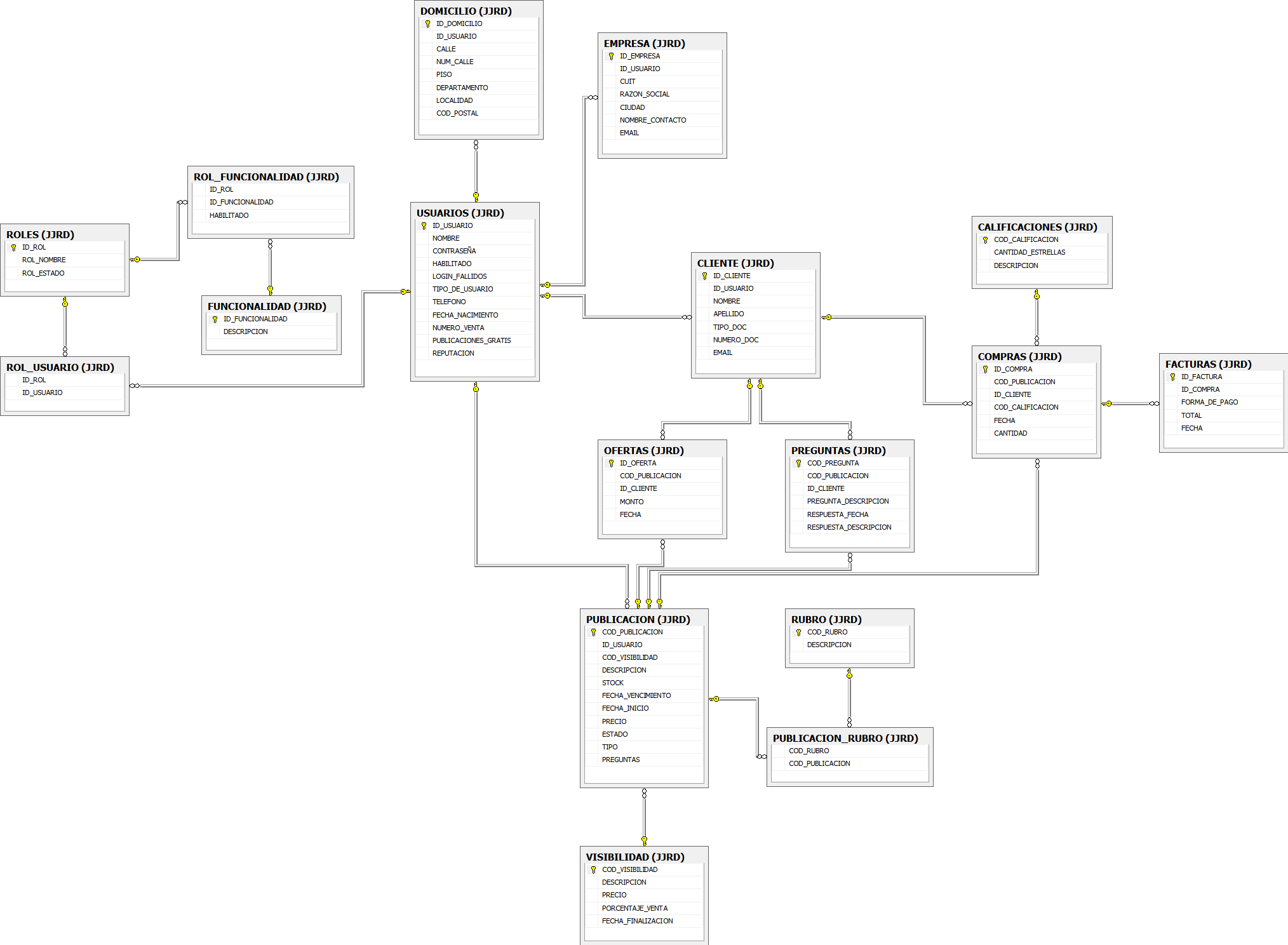
[Funciones 7](#_Toc372273683)

[Triggers 8](#_Toc372273684)

# DER

El siguiente es el DER de la solución planteada para la resolución del TP:

AGREGAR EL ULTIMO DER

**

Se recomienda para mayor legibilidad abrir el archivo *DER.png* que se encuentra en el mismo directorio que este documento.

**Lógica Principal**

La confección de la resolución del TP utiliza todo lo visto en clase en concepto de SQL y T SQL, a su vez añade en una parte dominante código ADO.NET para la ejecución de consultas desde la aplicación.

Del lado de la aplicación, la solución cuenta con una clase (.cs) principal llamada Query, la cual contiene la lógica para obtener datos múltiples en un DataTable (ObtenerDataTable), obtener un objeto como campo único (ObtenerUnicoCampo) y realizar la operaciones que no requieren la devolución de un resultado como INSERT, DELETE, UPDATE (Ejecutar).

Desarrollamos ésta clase con la funcionalidad extra de poder ejecutar Stored Procedures indicando el tipo de operación que se va a realizar mediante un atributo de la misma. Además, se posee una lista de parámetros de tipo SqlParameter de forma tal que los mismos se correspondan con los pedidos del procedimiento. No obstante, en forma mayoritaria, las consultas estan

realizadas mediante consultas almacenadas en variables string que luego son pasadas por el constructor de la clase para que éste las ejecute. Sin embargo, las funciones principales del TP estan almacenadas la base de datos y son llamadas a la aplicación mediante la funcionalidad descrita en el párrafo anterior.

Un dato importante es que la fecha del sistema debe modificarse desde el Program.cs, algo que nos pareció lógico siendo este el punto de entrada de la aplicación. El trabajo en su totalidad respeta como fecha actual la fecha del sistema almacenada en Properties.Settings.Default.FechaSistema y se encuentra actualmente seteada en la fecha "2013-10-01" dado que la mayoría de los datos de la tabla maestra eran cercanos a esa fecha.

Desde el lado de SQL, la migración fue realizada por la confección de diversos stored procedures que son ejecutados desde el script\_creacion\_inicial.sql entregado. No obstante se han generado otros procedimientos, funciones, triggers y vistas para mantener los datos lo más consistentes posibles. La lógica para la utilización completa de la aplicación como un administrador consiste en varios pasos que vamos a dejar asentados aquí para que se facilite la evaluación de nuestro trabajo:

1. Iniciar sesión en el sistema indicando el usuario y contraseña correspondiente. Existe un lote de usuarios que se pueden utilizar y todos funcionan con la misma contraseña indicada en el enunciado. Los usuarios son ADMIN (admin), ESOMOZA(admin), PALUNNI (admin).

2. Dependiendo el perfil del usuario con el que ingresó al sistema se habilitarán los ABM correspondientes.

3. Existe una opción para que, una vez ingresado al sistema con un usuario, se pueda cambiar el mismo y volver a loguearse.

# Consideraciones

* Todos los campos creados a partir de la normalización respetan los mismos tipos de datos que la tabla MAESTRA.
* El DER fue diseñado con la herramienta provista por Microsoft SQL Server Management Studio, de manera que refleja transparente y confiablemente el modelo de datos real.
* Para la carga inicial se crearon diversos procedimientos que agrupan la lógica de carga de cada tabla. La ejecución de estos procedimientos se realiza en un orden establecido para respetar la integridad referencial de las diversas tablas.
* No se podrán crear funcionalidades nuevas, sólo se le podrán habilitar o no a los roles nuevos las funcionalidades que ya existen actualmente en el sistema.
* En nuestro sistema la fecha del sistema es variable desde la aplicación, a fin de facilitar las pruebas que sean necesarias.

# Estructuras:

A continuación se realiza una descripción de las diferentes estructuras empleadas de la base.

## Stored Procedures

**Procedimientos para la carga inicial**

La carga inicial de las diferentes tablas que componen el modelo se agrupó en diferentes SP para un manejo más cómodo. La lógica es similar para todos:

1. Creación de la tabla
2. Inserción de datos en forma manual o tomándolos de la tabla maestra.

La lista de procedimientos que tienen esta funcionalidad es la siguiente: PIER.CREACIONDEUSUARIOS, PIER.CREARROLES, PIER.CREACIONDEFUNCIONALIDADES, PIER.ROL\_FUNCIONALIDAD, PIER.CREAPACIENTES, PIER.CREAMEDICOS, PIER.CREATURNOS, PIER.CREAESPECIALIDADES, PIER.CREAMEDICOSESPEC, PIER.CREAPLANES, PIER.CREAHORASTURNOS, PIER.CREABONOS

**JJRD.LOGUEAR**

Este procedimiento se dispara cuando un usuario realiza un login a la aplicación. Tiene como funcionalidad principal realizar la verificación de un logueo. En el caso de ser incorrecto se incrementará la cantidad de logins incorrectos para que al alcanzar los 3 consecutivos proceder ala inhabilitación del mismo.

**JJRD.CONSULTARAFILIADOS**

El procedimiento se encarga de devolver los datos de un afiliado consultando por su código de afiliado.

**JJRD.InsertPaciente**I

El procedimiento da de alta un nuevo paciente con todos sus datos personales.

**PIER.UpdatePaciente**

Permite actualizar los datos del paciente: dirección, teléfono, mail y plan.

**PIER.AgregarProfesional**

El procedimiento da de alta un nuevo profesional con todos sus datos personales.

**PIER.UpdateProfesional**

Permite actualizar los datos del profesional.

**PIER.BajaProfesional**

El sp realiza la baja lógica de un profesional (identificado con su matrícula) y procede a eliminar los turnos que tenía asignado.

**PIER.InsertPacienteConyuge**

El procedimiento contiene la lógica de asociación de un conyuge a un afiliado, respestando la numeración correspondiente para asignar el número de afiliado.

**PIER.CONSULTARMEDICO**

El procedimiento se encarga de devolver los datos de un médico consultando por su matrícula.

## Funciones

## 

## Triggers

**PIER.bajaAfiliados**

Este trigger se dispara cuando el campo Habilitado de la tabla Pacientes es actualizado. Se procede a insertar el paciente dado de baja a la tabla de pacientes histórica.