**UTN – GESTION DE DATOS**

2014 2c

**GAME OF QUERYS**

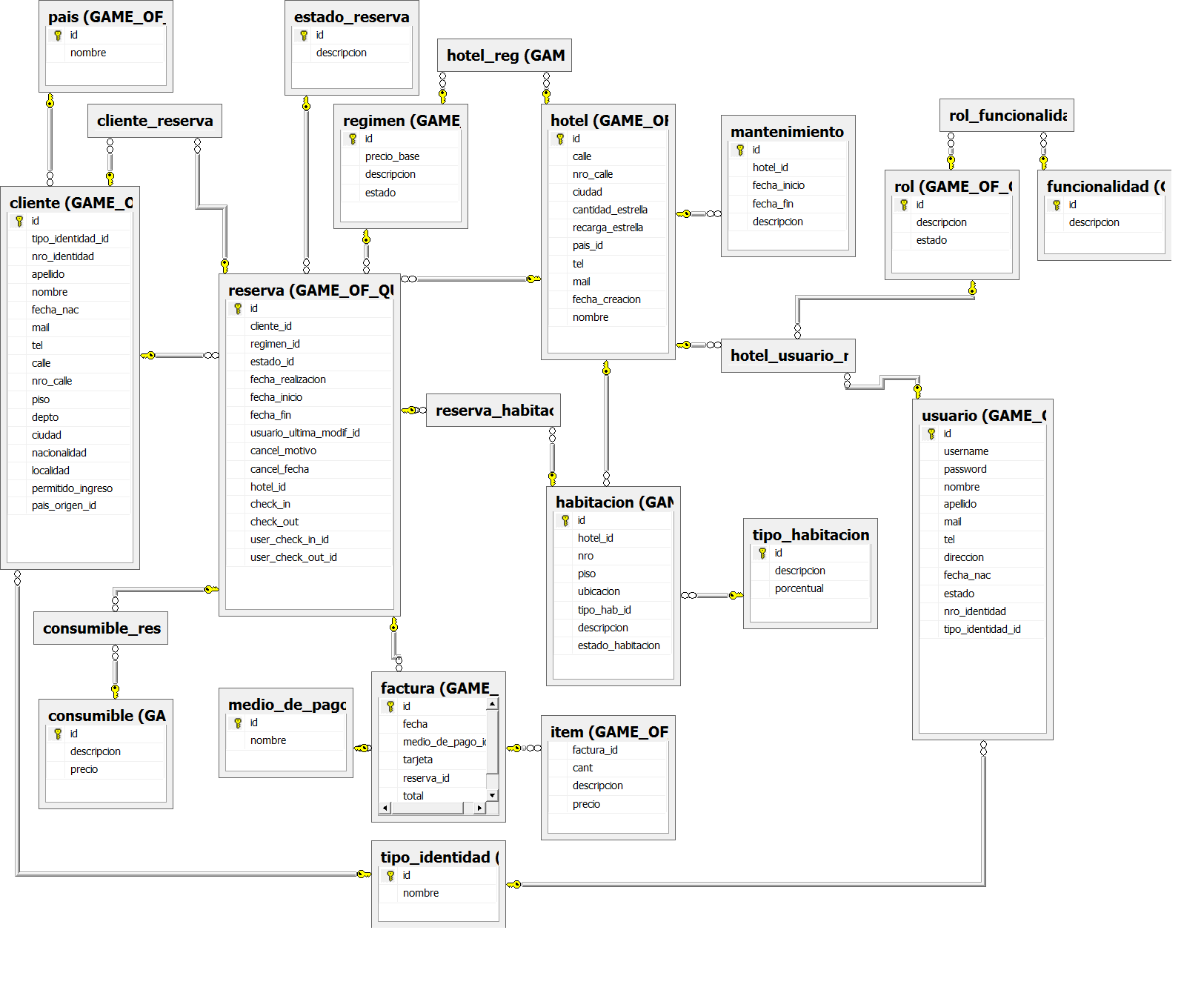
**FRBA – Hoteles, Estrategias**

**Pablo Pace**

**Florencia Marcaccio**

**Hernán Loiacono**

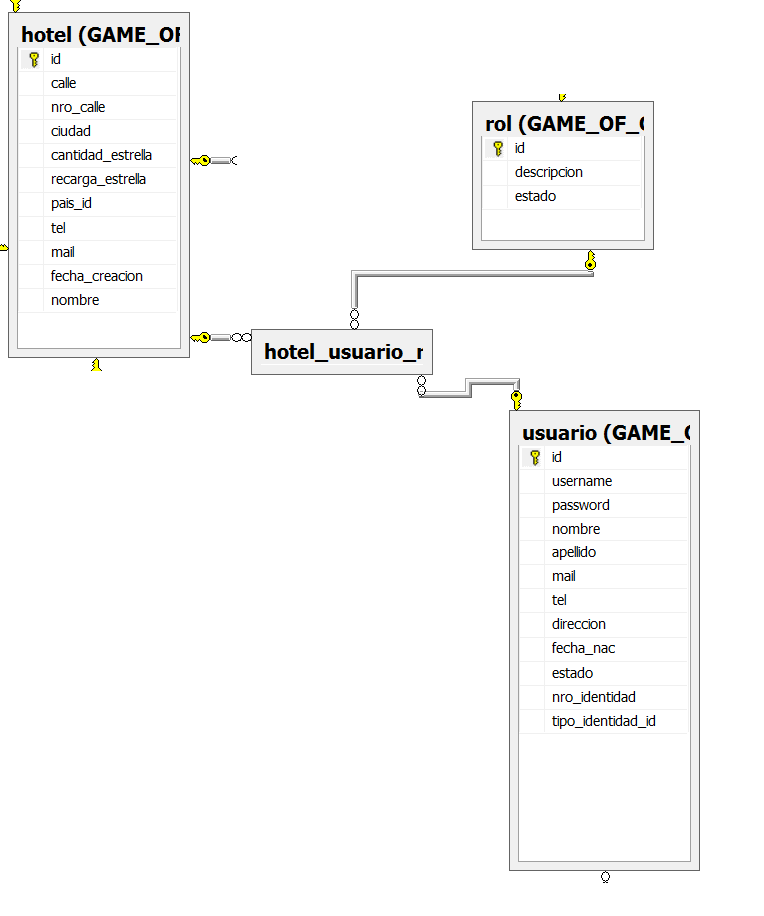
**Julián Zelayeta**



# Roles

1. Se consideró un rol adicional: admin. Este rol es el único que puede acceder al abm de Roles y hacer modificaciones en los mismos.
2. Se estableció que las ABM de Hotel y de Usuario sean funcionalidades exclusivas de los administradores comunes o de 'admin', el administrador general.
3. Se creó un trigger para respetar las restricciones mencionadas en los puntos 1 y 2.

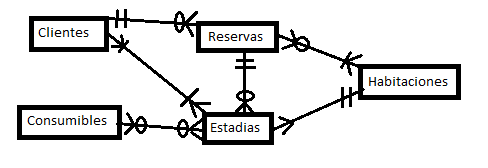
# Login

1. Se diseñó el modelo de datos con el objetivo de que un mismo usuario tuviera distintos roles en distintos hoteles. Para ello utilizamos una tabla **hotel\_usuario\_rol** con 3 FKs. Por cuestiones de tiempo, se decidió realizar que la implementación del rol sea independiente del hotel. Es decir, un mismo usuario tendrá los mismos roles en todos sus hoteles asignados.
2. La password **w23e** para el usuario **admin** se hardcodeó hasheada en sha256 en la migración.
3. Para el usuario 'admin' se consideró que trabaja como administrador general para todos los hoteles. Esto quiere decir que, al igual que los otros usuarios que trabajan en más de un hotel, va a tener que elegir el hotel que desea y en esa sesión solo podrá realizar funciones para ese hotel.

# Hoteles

1. Cada habitación de cada hotel tiene un id único. El numero de habitación para un mismo hotel es único. Pero se puede repetir el numero de habitación para distintos hoteles.
2. Se pueden borrar los mantenimientos con fecha en el futuro. Mantenimientos viejos y los que se encuentran en curso no se pueden pueden borrar.
3. No se pueden realizar mantenimientos en fechas donde tengo reservas o estadías.
4. En “All Inclusive moderado” se cobran todos los consumibles.
5. No se pueden modificar los regímenes de un hotel si se tienen reservas o estadías en curso bajo ese régimen.
6. Se decidió que los administradores pueden realizar modificaciones y bajas en cualquier hotel de la cadena.
7. Al realizar el alta de un hotel se asigna como administrador a la persona que realizó el alta.

# Reservas

1. Se había considerado realizar una tabla de estadías, a modo de tener una estadía por cada habitación de la reserva. De esta manera podríamos tener checkins y checkouts distintos para cada habitación. Finalmente se decidió pasar las fechas de chekins y checkouts a la tabla reservas ya que todos los huéspedes de la reserva deben realizar el checkout el mismo día y se factura todo a una sola persona.
2. Todos los huéspedes de una determinada habitación realizan el checkout en la misma fecha.
3. Se decidió incluir el atributo hotel\_id en la entidad Reserva para evitar consultas muy extensas. Ya que se podría obtener el hotel de una reserva a partir de las habitaciones que se reservaron.
4. Para realizar la baja o modificación de una reserva es necesario el código de reserva que se comunicó al generar la reserva.

# Estado de reservas

1. Notamos que la última fecha de check out que aparecía en la tabla maestra es '31-12-2016' por lo que no tiene lógica que la 'fecha de hoy' sea menor a esta. Por ello, la fecha actual es simulada y se trae desde archivo de configuración: Frbahotel/App.config. Elegimos que sea '01-01-2017'.
2. En todas las reservas que se encontraron en la tabla se les puso su estado según su fecha con respecto de la máxima fecha encontrada, '31-12-2016' (recordar que esta fecha la tomamos como si fuera el día de ayer):

* Las reservas cuya fecha de inicio es igual o menor a esta y no tiene registrado el check in se las registró como 'Cancelada por no-show'.
* Las reservas cuya fecha de inicio es igual o menor a esta y tiene registrado el check in se las registró como 'con ingreso'.
* Las reservas cuya fecha de inicio es mayor a esta, es decir que su fecha de inicio es la de 'hoy' o mayor, se registran como 'correctas'. Sin embargo, no existen reservas con fecha de inicio mayor al 31-12-2016.

# Países

1. Los países se cargan durante la migración en SQL directamente en la base de datos.

# Consumibles

1. Realizamos una funcionalidad llamada 'AltaConsumible' cuyo fin es agregar ítems a los que ya tiene en stock el hotel.
2. El 'listadoConsumibles' se utiliza para buscar productos que puedan existir en el stock del hotel.
3. Hay dos formas de registrar consumibles para cada habitación. La primera, durante la estadía se puede ingresar al formulario de 'Registrar Consumibles' y elegir lo que desea una determinada habitación. La otra, durante el check out, se pueden ingresar consumibles que no se hayan registrado a lo largo de la estadía.
4. Se decidió no registrar en la factura qué consumió cada habitación, pero lo puede ver cuando hace el check out y en la base.

# Regímenes

1. Cada reserva, independientemente de la cantidad de habitaciones, puede tener un único régimen.

# Estadías

1. No se puede realizar el check in de reservas canceladas, ya que sus habitaciones pueden haber sido asignadas a otras reservas.
2. Solo se puede realizar el check in en el día de la fecha de inicio de la reserva.

# Listado estadístico

1. Los listados se realizaron como stored procedure para mejorar su desempeño, ya que establecen su plan de ejecución en la primer compilación y lo reutilizan en las siguientes invocaciones.

# Consultas

1. Se agrego en la pantalla principal una pestaña de 'Consultas', en donde se ofrecen 3 tipos de consultas sobre estadías y habitaciones ocupadas del hotel:

* Listar habitaciones ocupadas
* Listar habitaciones de una estadía en particular, especificando el código de reserva
* Consultar código de reserva de una habitación en particular

1. Estas consultas pueden ser de ayuda, por ejemplo, para recordar qué números de habitación pertenecen a una estadía sin tener que recurrir a la base. Recordar que para hacer el check out de una estadía se debe ingresar el número de cualquiera de sus habitaciones, por lo que estas consultas pueden ser de mucha ayuda.