TRABAJO PRACTICO

1º CUATRIMESTRE

2019

UTN – FACULTAD REGIONAL BUENOS AIRES

* **Título**: Gestión de Datos
* **Año**: 3°
* **Código**: 082030
* **Curso**: K3012
* **Grupo**: MACACO\_NOT\_NULL

INTEGRANTES LEGAJO

*COBOS, Bruno 159676-7*

*JUGO, German 158917-9*

*NIKCEVICH, Carlos Alexis 146698-7*

*VERDILE, Guillermo* *158962-3*

utn - frba

geStión de datos

Índice

[Introducción 2](#_Toc12894211)

[DER 3](#_Toc12894212)

[Desarrollo 3](#_Toc12894213)

[Crucero, Baja Crucero y Compañía 3](#_Toc12894214)

[Cabina y Tipo de Servicio 4](#_Toc12894215)

[Puerto, Tramo y Recorrido 4](#_Toc12894216)

[Viaje 5](#_Toc12894217)

[Usuario y Login 6](#_Toc12894218)

[Rol, Funcionalidad y Rol x Funcionalidad 7](#_Toc12894219)

[Reserva y Reserva Cabina 8](#_Toc12894220)

[Pasaje, Pago y Medio de Pago 9](#_Toc12894221)

[Script Inicial, Stored Procedures, Triggers y Funciones 9](#_Toc12894222)

[Procedures 10](#_Toc12894223)

[Funciones 12](#_Toc12894224)

[Triggers 12](#_Toc12894225)

[Decisiones de Diseño en la aplicación Desktop 12](#_Toc12894226)

# Introducción

A través de la Estrategia proponemos explicar las decisiones que fueron pertinentes tomar al momento de realizar la etapa del diseño de la aplicación Desktop con el framework de C# .Net versión 4.5, y su posterior implementación, como así también el modelado de los datos, implementado en MS SQL Server 2012. Adjunto se encuentra el DER correspondiente al trabajo práctico que refleja la integridad relacional del mismo.

Como puntos a destacar, mencionamos que el tiempo de migración promedio aproximado delos datos existentes esde 1:03 minutos en las pruebas realizadas por el grupo.

Al momento de empezar con la etapa de desarrollo del proyecto, analizamos la tabla maestra para poder obtener los requerimientos solicitados de una manera clara y concisa, respetando el modelo de datos y las reglas del negocio existente. Para documentar esta información utilizamos un diagrama de entidad relación que nos ayudó a poder identificar cada una de las distintas entidades que utilizamos a lo largo del desarrollo del trabajo práctico, como así también la relación que existía entre cada una de ellas y sus propiedades, como la cardinalidad y modalidad, haciendo principal foco en que el objetivo final es desarrollar un programa que permita administrar la venta y reserva de pasajes para los viajes de cruceros.

En la parte de pruebas de la aplicación, se asignan los siguientes nombres de usuarios con roles de administradores:

*jPerez*

*jGonzalez*

*aMontana*

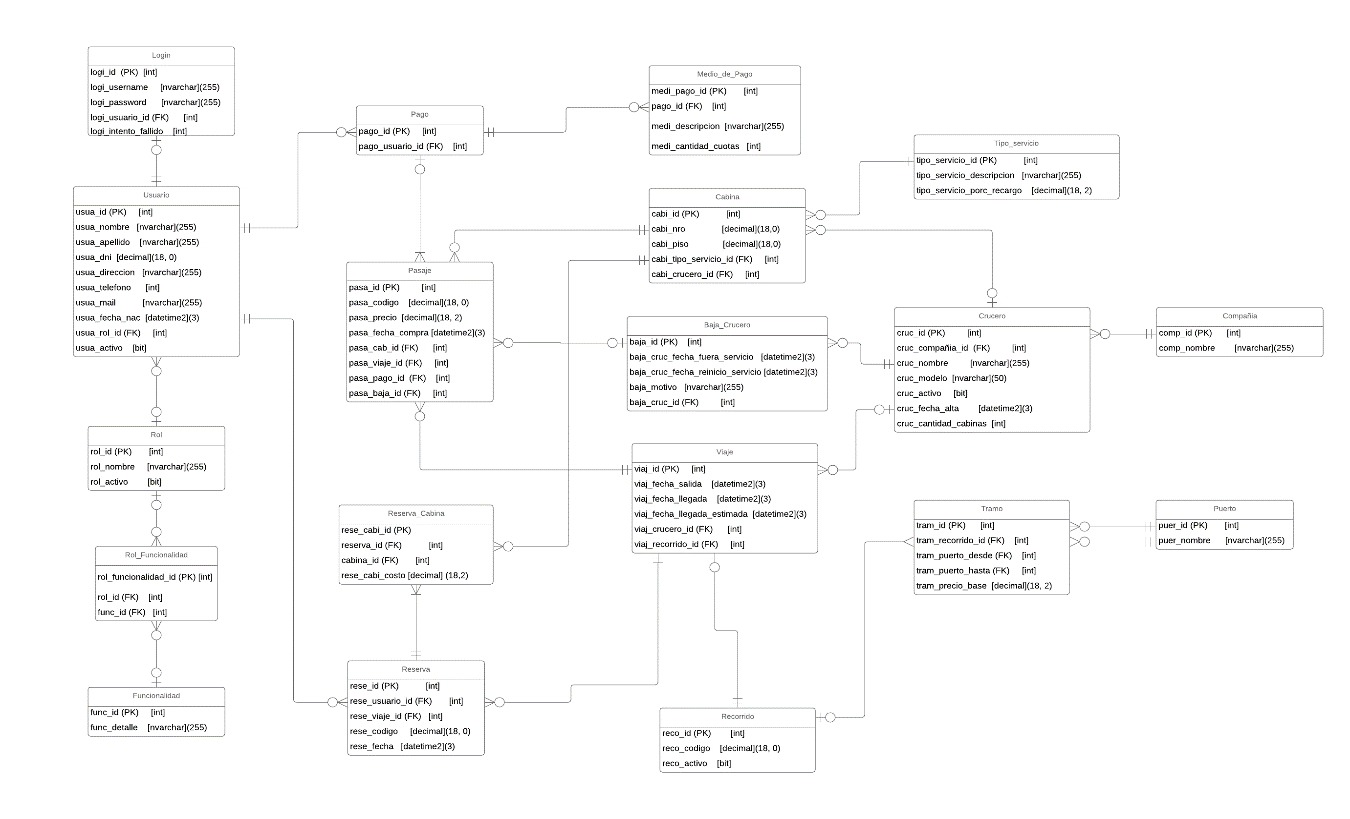
*mMozart*

*rtesoro*

*admin*

Aclaración importante*:* utilizar la contraseña “*w23e*” para el logueo como ADMINISTRADOR.

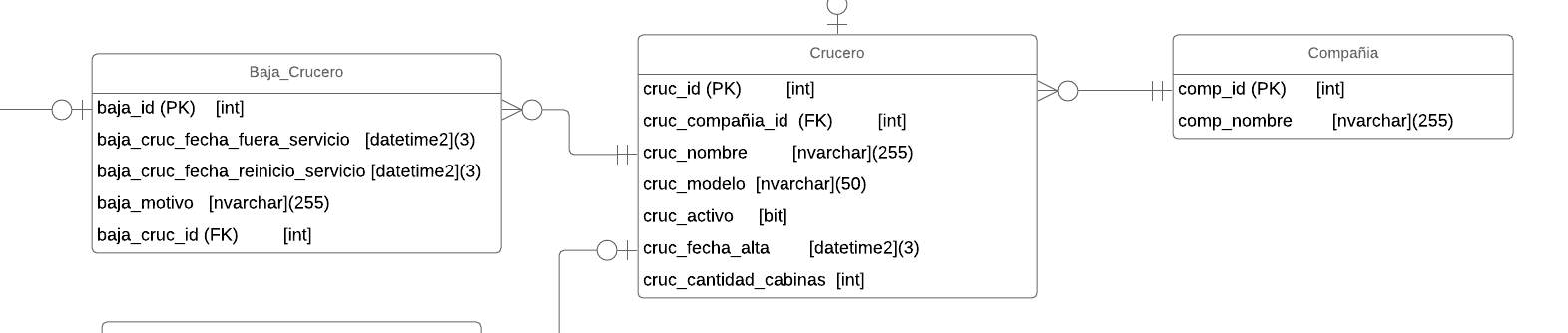
# DER



Aclaración importante: para más detalles, *ver archivo DER.jpg adjunto en el archivo zip.*

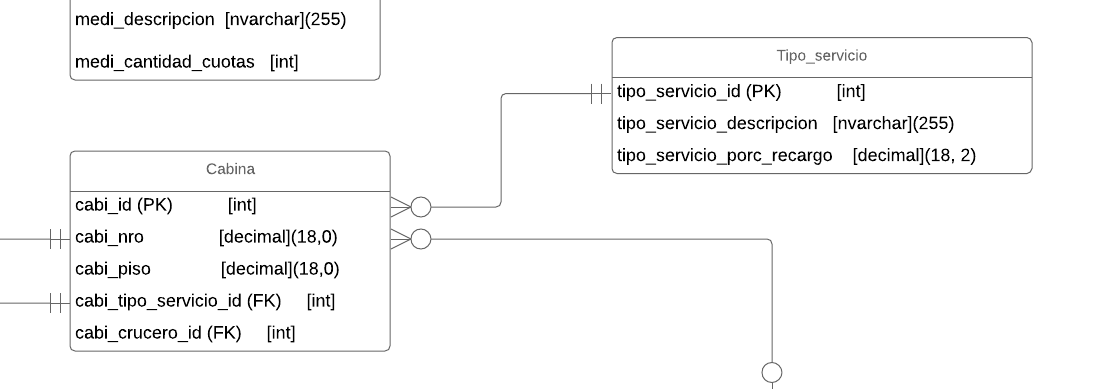
# Desarrollo

# Crucero, Baja Crucero y Compañía



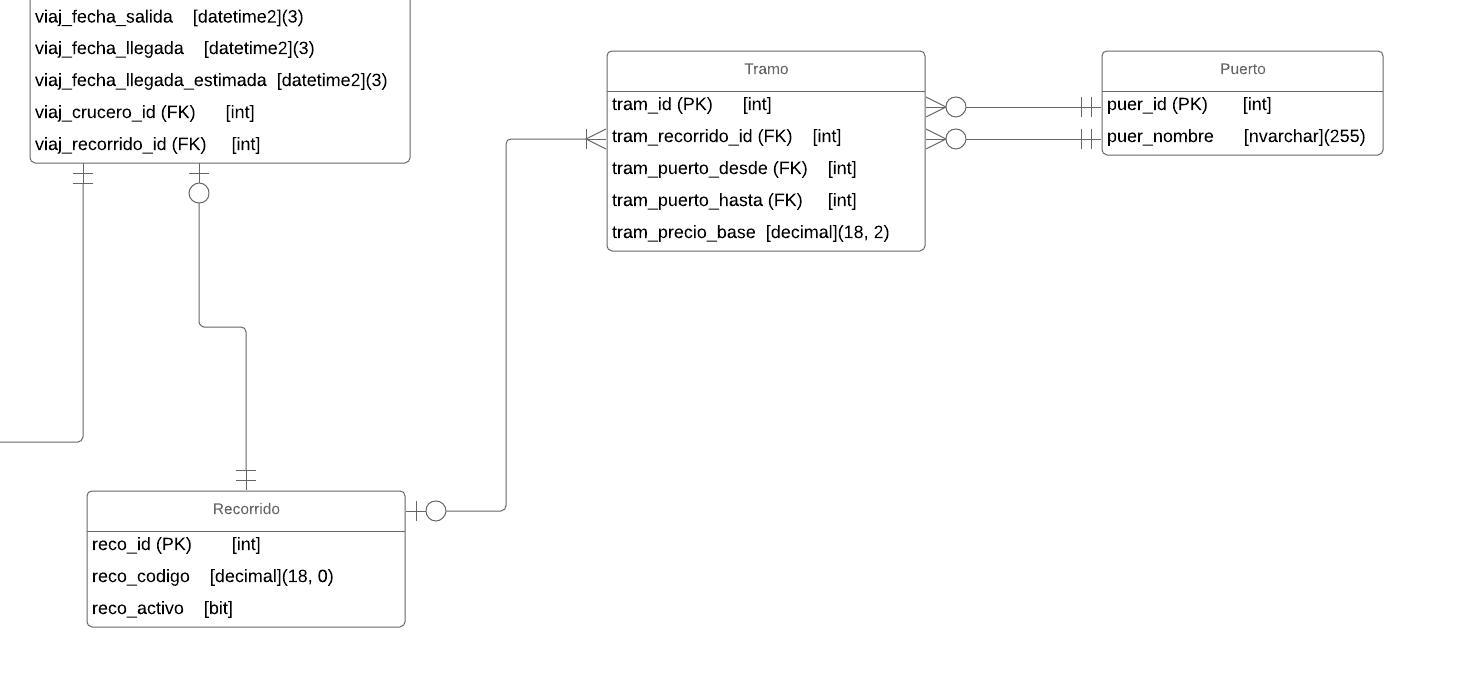
Para diseñar la tabla cruceros tomamos en cuenta que su PK es *cruc\_id,* como FK está *cruc\_compañía,* lo que nos permite en un futuro cambiar fácilmente al fabricante de cruceros en caso de ser necesario.También decidimos guardar el nombre, el modelo, la cantidad de cabinas y el estado del crucero lo representamos con los atributos de crucero activo y crucero fecha de alta. En este punto se decidió modelar una entidad aparte para la *Baja de Crucero*, que tiene como PK *baja\_id* y como FK a *baja\_cruc\_id,* además de guardar la fecha de fuera de servicio y fecha del reinicio de servicio como así también el motivo de la baja.

# Cabina y Tipo de Servicio



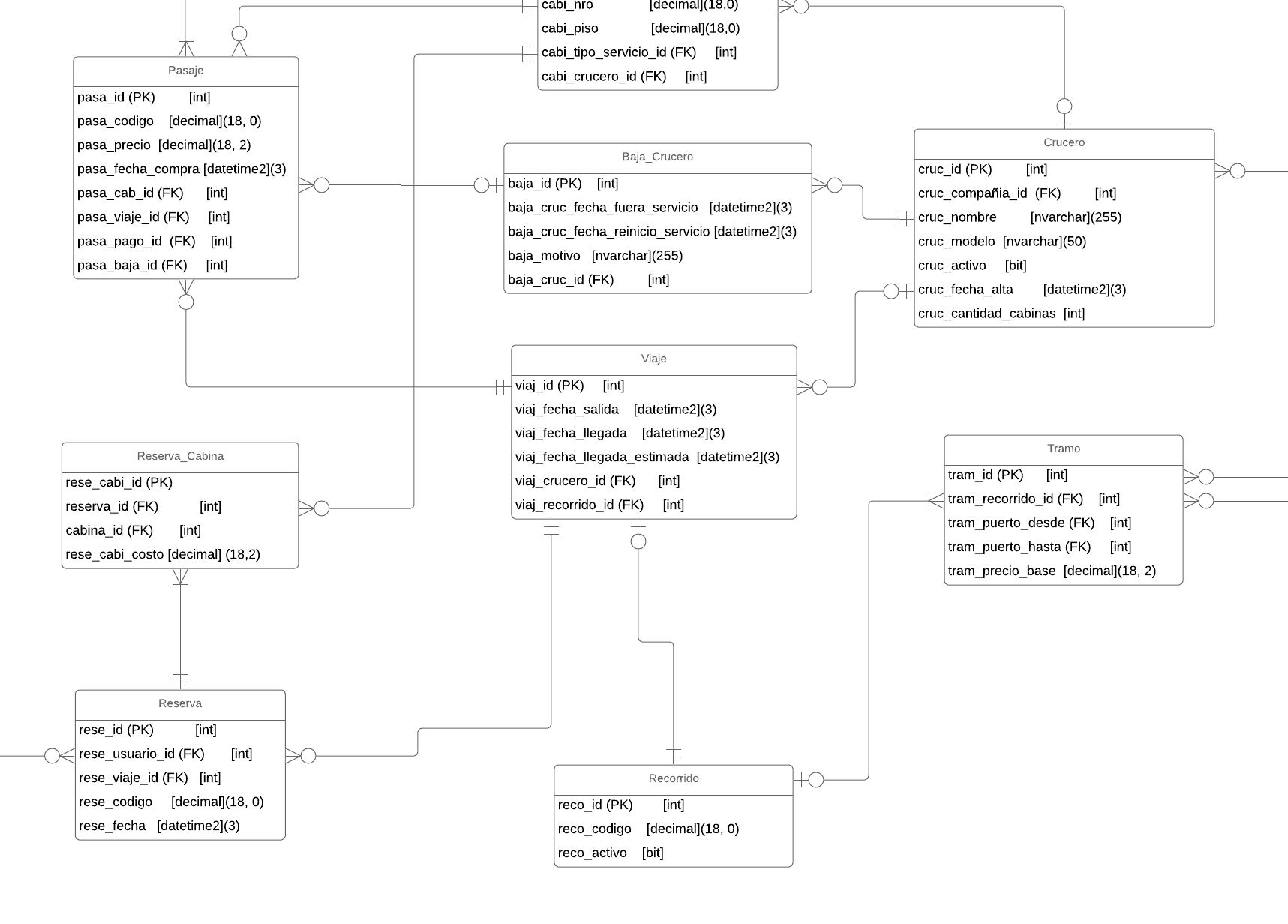
La tabla cabina tiene como PK a *cabi\_id*, como FK a *cabi\_crucero\_id*y*cabi\_tipo\_servicio,* permite guardar además el número y piso la cabina. Acá se decidió que la tabla Tipo\_servicio almacene los datos referidos a la descripción y porcentaje de recargo asociados al tipo de cabina.

# Puerto, Tramo y Recorrido



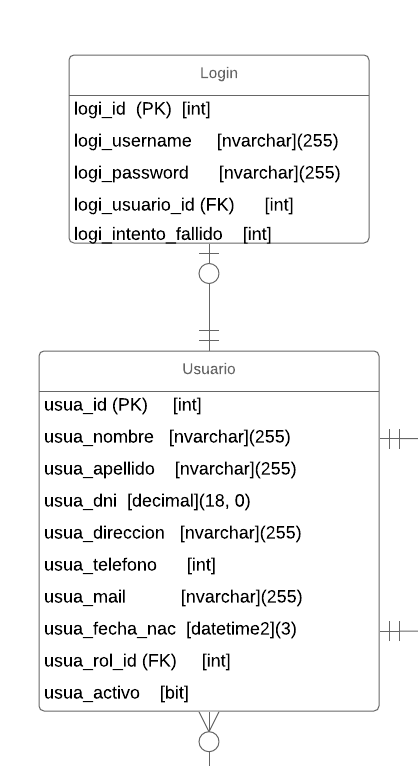
Sabiendo que cada ciudad tiene un solo puerto por las reglas del negocio existentes en el trabajo práctico, decidimos que cada tramo tenga un puerto de origen y un puerto de destino, Y por ende que un recorrido este formado por varios tramos. En la tabla Recorrido la PK es *reco\_id*, guarda también el código del recorrido y si está o no activo. Del Puerto, su PK es *puer\_id*, guardando el nombre de este. La entidad tramo rompe con la relación de muchos a muchos entre recorrido y puerto, por lo que además de heredar las PK De las dos entidades, tiene su propia PK *tram\_id*, y guarda el precio base por tramo.

# Viaje



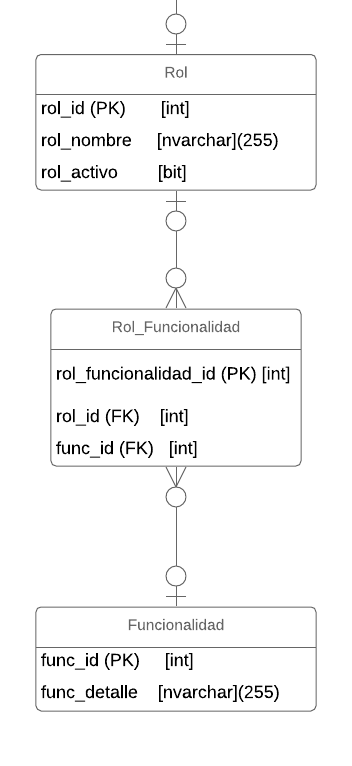
Para los viajes mencionamos que su PK es *viaj\_id*, sus FK son *viaj\_cruc\_id* y *viaj\_recorrido*, permitiendo persistir la fecha de salida, la fecha de llegada y la fecha de llegada estimada. Esta tabla se relaciona con las tablasCrucero,Pasaje,Reserva y Recorrido.

# Usuario y Login



La tabla Usuario tiene PK *usua\_id,* como FK a *usua\_rol\_id*. Guarda nombre, apellido, DNI, dirección, teléfono, mail, fecha de nacimiento y si está activo el usuario. Mientras que la tabla de Logintiene PK *logi\_id*y cómo FK a *logi\_usuario\_id.* Aquí se decidió utilizar el mecanismo de encriptación de hashing para los datos sensibles como lo son las contraseñas de los usuarios. Otro punto que consideramos para reforzar la seguridad es implementar un contador de intentos fallidos que bloquea un usuario cuando éste ingresa 3 veces mal su contraseña, cumpliendo así con uno de los requerimientos establecidos en el trabajo práctico.

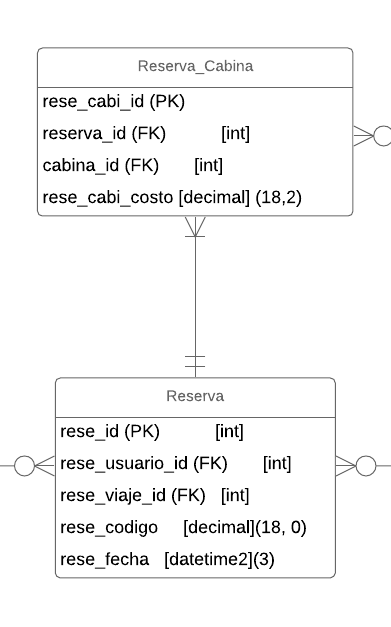
# Rol, Funcionalidad y Rol x Funcionalidad



Decidimos modelar a los usuarios con distintas funcionalidades de acuerdo con el rol que ocupen dentro del sistema, pudiendo ser *administradores* o *clientes* de este, con lo cual nos vimos obligados a contemplar la seguridad para los usuarios administradores, haciendo énfasis en evitar guardar contraseñas en texto plano. La tabla Rol tiene PK a*rol\_id*, guarda el nombre y si está activo o no el rol. La tabla Funcionalidad tiene PK *func\_id* y guarda el detalle de la misma. Para romper la relación de muchos a muchos se crea la entidad Rol x Funcionalidad, con su PK *rol\_funcionalidad\_id* y hereda como FK a las PKs de las 2 entidades.

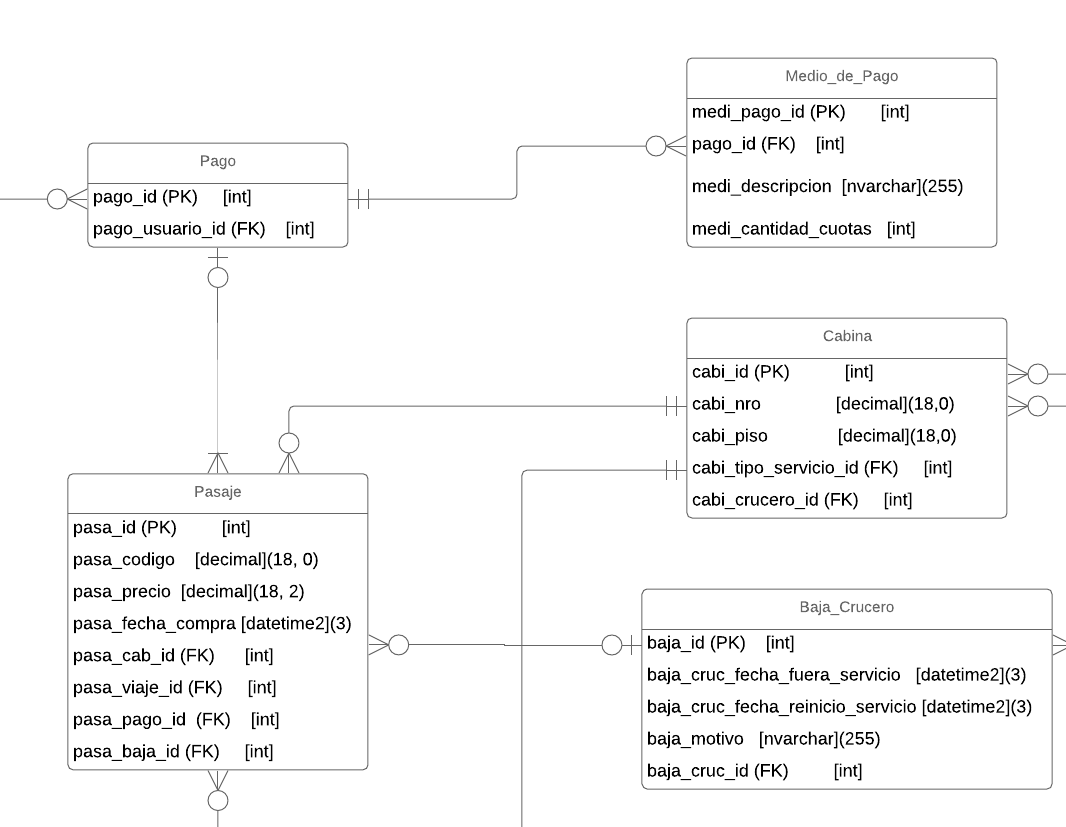
Con respecto a la aplicación desktop en la baja del rol nosotros permitimos dar de baja un rol destildando el checkbox logrando asi dar de baja un rol, en caso de que ya se encuentre destildado se verificara que se encuentre dado de baja, en caso de que tilde el checkbox de activo no ocurrirá ningún cambio debido a que esta pantalla no permite la dada de alta de los roles.

# Reserva y Reserva Cabina



Para verificar cuándo vence una reserva decidimos que en la tabla se guarde la fecha en la que fue efectuada la misma. Se asigna también un código para su posterior pago ya que es un compromiso de compra por parte del usuario respaldado por las reglas vigentes del negocio. La tabla reserva cabina guarda también el costo de acuerdo al tipo de servicio.

# Pasaje, Pago y Medio de Pago



Cuando una reserva necesita ser pagada se genera un pasaje con código de pasaje, el precio, la fecha de compra, la identificación de la cabina, la identificación del viaje, la identificación del pago y en el caso de que se de baja un crucero también se guarda la baja identificación. El pago lo asocia a un usuario qué puede elegir diferentes medios de pago qué debe especificar junto con la cantidad de cuotas FF siendo qué pagando de contado en efectivorepresenta una cuota.

# Script Inicial, Stored Procedures, Triggers y Funciones

Completada la etapa anterior, se confeccionó el script inicial que permitió hacer una migración de los datos existentes manteniendo la integridad con el nuevo sistema, Y permitiendo actualizar o eliminar filas dentro de los registros. De forma conjunta se desarrollaron los *procedures* para las entidades de *cruceros*, *recorridos*, *roles* y *viajes*en forma modular, para luego unificar todo en un único script.

# Procedures

PROCEDURE[MACACO\_NOT\_NULL].AltaRol:Este procedure da de alta un rol con una funcionalidad.

PROCEDURE[MACACO\_NOT\_NULL].BuscarRol: Este procedure trae una lista de roles los cuales cumplan con los filtros elegidos.

PROCEDURE[MACACO\_NOT\_NULL].BajaRol: Este procedure establece en inactivo un rol.

PROCEDURE[MACACO\_NOT\_NULL].ModificarNombreRol: Se llama este procedure primero al querer modificar un rol

PROCEDURE[MACACO\_NOT\_NULL].AgregarFuncionalidadRol: Si en la tabla de funcionalidad pantalla de modificacion de Rol hay N filas, llamar a este procedure N veces

PROCEDURE[MACACO\_NOT\_NULL].HabilitarRol: Este procedure establece en activo un rol inactivo.

PROCEDURE[MACACO\_NOT\_NULL].LogearUsuario: Este procedure permite el ingreso de un usuario al sistema verificando su contraseña.

PROCEDURE[MACACO\_NOT\_NULL].GetCruceros: Obtiene una lista de cruceros de acuerdo a los que cumplan los filtros seteados.

PROCEDURE[MACACO\_NOT\_NULL].CreateCrucero: Crea un nuevo crucero.

PROCEDURE[MACACO\_NOT\_NULL].UpdateCrucero: Modifica los valores de un crucero.

PROCEDURE[MACACO\_NOT\_NULL].GetCabinasXPisoYServicio: Obtiene las cabinas con su tipo de servicio y piso de acuerdo al id del crucero ingresado.

PROCEDURE[MACACO\_NOT\_NULL].GetTramos: Obtiene una lista de tramos de acuerdo al codigo del recorrido ingresado.

PROCEDURE[MACACO\_NOT\_NULL].getRecorridos: Obtiene una lista de recorridos de acuerdo al codigo del recorrido ingresado y las ciudad de origen y destino.

PROCEDURE[MACACO\_NOT\_NULL].InsertRecorrido:Inserta todos los recorridos que tengan el codigo ingresado en una tabla de tramos.

PROCEDURE[MACACO\_NOT\_NULL].BajaRecorrido: Da de baja un recorrido salvo que el recorrido se encuentre en un viaje.

PROCEDURE[MACACO\_NOT\_NULL].GetPuertoByName: Obtiene todos los puertos que tengan el nombre ingresado.

PROCEDURE[MACACO\_NOT\_NULL].ModificarRecorrido: Modifica los valores de un recorrido.

PROCEDURE[MACACO\_NOT\_NULL].GetFuncionalidades: Obtiene una lista de funcionalidades de acuerdo a los filtros seleccionados.

PROCEDURE[MACACO\_NOT\_NULL].UpdateRol: Modifica los valores de un rol.

PROCEDURE[MACACO\_NOT\_NULL].GenerarViaje: Genera un nuevo viaje con la fecha, el crucero y el recorrido ingresado.

PROCEDURE[MACACO\_NOT\_NULL].OrdenarTramosRecorridos: Procedure que devuelve los tramos de los recorridos ordenados lógicamente.

PROCEDURE[MACACO\_NOT\_NULL].AgregarBajaCrucero: Se realiza la baja de un crucero de manera temporal.

PROCEDURE[MACACO\_NOT\_NULL].AgregarBajaCruceroDefinitivo: Realiza la baja de un crucero de manera definitva sin fecha de recuperacion y sin motivo.

PROCEDURE[MACACO\_NOT\_NULL].CancelarPasajes: Se ejecutar luego de agregar una baja a un crucero (siempre y cuando la accion posterior elegida por el admin era cancelar los pasajes vendidos del viaje) .

PROCEDURE[MACACO\_NOT\_NULL].PosponerViajes: Para posponer, una cierta cantidad de dias, los viajes del crucero que se acaba de dar de baja.

PROCEDURE[MACACO\_NOT\_NULL].ReemplazarCrucero: Reemplaza un crucero con otro el cual no se le superpongan los viajes que posee con los nuevos que tendria que realizar.

PROCEDUREMACACO\_NOT\_NULL.IdCruceroRemplazante: Obtiene el id de un crucero el cual puede reemplazar al crucero ingresado, si devuelve -1 no se encontro ninguno.

PROCEDURE[MACACO\_NOT\_NULL].CrearViaje: Creacion de un nuevo viaje

PROCEDURE[MACACO\_NOT\_NULL].ComprobarVigenciaReservasDelSistema: Comprobacion del vencimientos de todas las reservas.

PROCEDURE[MACACO\_NOT\_NULL].ComprobarVigenciaReserva:Comprobacion de vencimiento de reserva.

PROCEDUREMACACO\_NOT\_NULL.CabinasDisponiblesViaje: Obtiene la lista de las cabinas disponible para un viaje.

PROCEDURE[MACACO\_NOT\_NULL].[GenerarReserva]: Generacion reserva

PROCEDURE[MACACO\_NOT\_NULL].AgregarCabina\_Reserva: Agregar 1 cabina a 1 reserva

PROCEDURE[MACACO\_NOT\_NULL].AgregarPasajeA\_Cliente: Procedure que agrega 1 pasaje, se ejecuta en el caso que se compre directamente un pasaje, sin pasar por la reserva

PROCEDURE[MACACO\_NOT\_NULL].AgregarPagoReserva\_Y\_PasajesAlCliente: Agrega los pasajes y el pago de la reserva al cliente de acuerdo al codigo de reserva.

PROCEDURE[MACACO\_NOT\_NULL].AgregarMedioDePago\_Al\_NuevoPago: Agrega los medios de pago a un nuevo pago.

PROCEDURE[MACACO\_NOT\_NULL].EliminarReserva: Elimina un reserva.

PROCEDURE[MACACO\_NOT\_NULL].[CrucerosConMasReparaciones]: Obtiene una lista de cruceros con mas reparaciones.

PROCEDURE[MACACO\_NOT\_NULL].[RecorridosConMasCabinasLibresEnSusViajes]:Obtiene una lista con los recorridos que posean mas cabinas libres en sus viajes.

PROCEDURE[MACACO\_NOT\_NULL].GetViajes: Devuelve todos los viajes que 'pasen' por el puerto origen o el destino.

PROCEDURE[MACACO\_NOT\_NULL].GetTipoServicioByDescription: Obtiene una lista con los tipos de servicio que tengan la descripcion ingresada.

procedure[MACACO\_NOT\_NULL].ObtenerCabinasDelCrucero: Obtiene una lista de cabinas del crucero ingresado.

PROCEDURE[MACACO\_NOT\_NULL].CreateOrUpdateCliente: Crea o modifica los valores de un cliente.

PROCEDURE[MACACO\_NOT\_NULL].VerificarViajeYaRerservadOComprado: Verifica si un viaje ya fue reservado o comprado.

# Funciones

FUNCTION[MACACO\_NOT\_NULL].GetPuertoId: Obtiene el id del puerto.

FUNCTION[MACACO\_NOT\_NULL].GetRecorridoIdByRecoCodigo: Obtiene los recorridos de acuerdo al código ingresado.

FUNCTION[MACACO\_NOT\_NULL].PrecioRecorrido: Obtiene el precio del recorrido.

FUNCTION[MACACO\_NOT\_NULL].ComprobarExistenciaReserva: Al ingresar el codigo de una reserva se debe verificar que exista alguna con ese numero.

FUNCTIONMACACO\_NOT\_NULL.ciudad\_origen: Obtiene la ciudad de origen de acuerdo al código de recorrido ingresado.

FUNCTIONMACACO\_NOT\_NULL.ciudad\_destino: Obtiene la ciudad de destiono de acuerdo al código de recorrido ingresado.

FUNCTION[MACACO\_NOT\_NULL].PrecioRecorrido: Obtiene el precio del recorrido.

FUNCTION[MACACO\_NOT\_NULL].DetallesReserva: Retorna toda la informacion asociada a una reserva.

FUNCTION[MACACO\_NOT\_NULL].ObtenerCabinas: Obtiene la lista de cabinas de acuerdo a los parámetros ingresados.

# Triggers

TRIGGER[MACACO\_NOT\_NULL].TRIGGER\_BLOQUEAR\_USUARIO\_POR\_LOGIN\_FALLIDO: Se ejecuta post login fallido.

TRIGGER[MACACO\_NOT\_NULL].DeleteReservasCabinas: Es ejecutado cuando se realiza el delete de una reserva.

# Decisiones de Diseño en la aplicación Desktop

Paralelamente se diseñaron las vistas del programa teniendo en cuenta los lineamientos pautados en el trabajo práctico, tanto para el desarrollo de los ABM, como para las clases auxiliares que utilizamos.

*En el caso de que un Crucero fuera dado de BAJA ya sea por fuera de servicio o por vida útil, se decidió qué ante esta situación, todos los pasajes que estén involucrados en dicho crucero se cancelen de forma automática.*

*Al momento de dar de ALTA un Crucero, cuándo se eligen la cantidad de cabinas con sus respectivos servicios, sólo se puede agregar un tipo de servicio por piso, con la intención de simplificar la manipulación de los datos y su representación en las vistas.*

*Cuando se desea MODIFICAR un Crucero que ya está creado, primero se lo busca desde el Listado Cruceros. Después se selecciona con la Celda de Modificar, y luego se abre la ventana de Modificación Crucero con los datos actuales del mismo. En este punto no se habilita la opción para Agregar o Eliminar cabinas del crucero que estás modificando, sólo se considera el hecho de poder cambiar el Nombre, Modelo o Compañía del Crucero.*