

**Gestión de Datos**

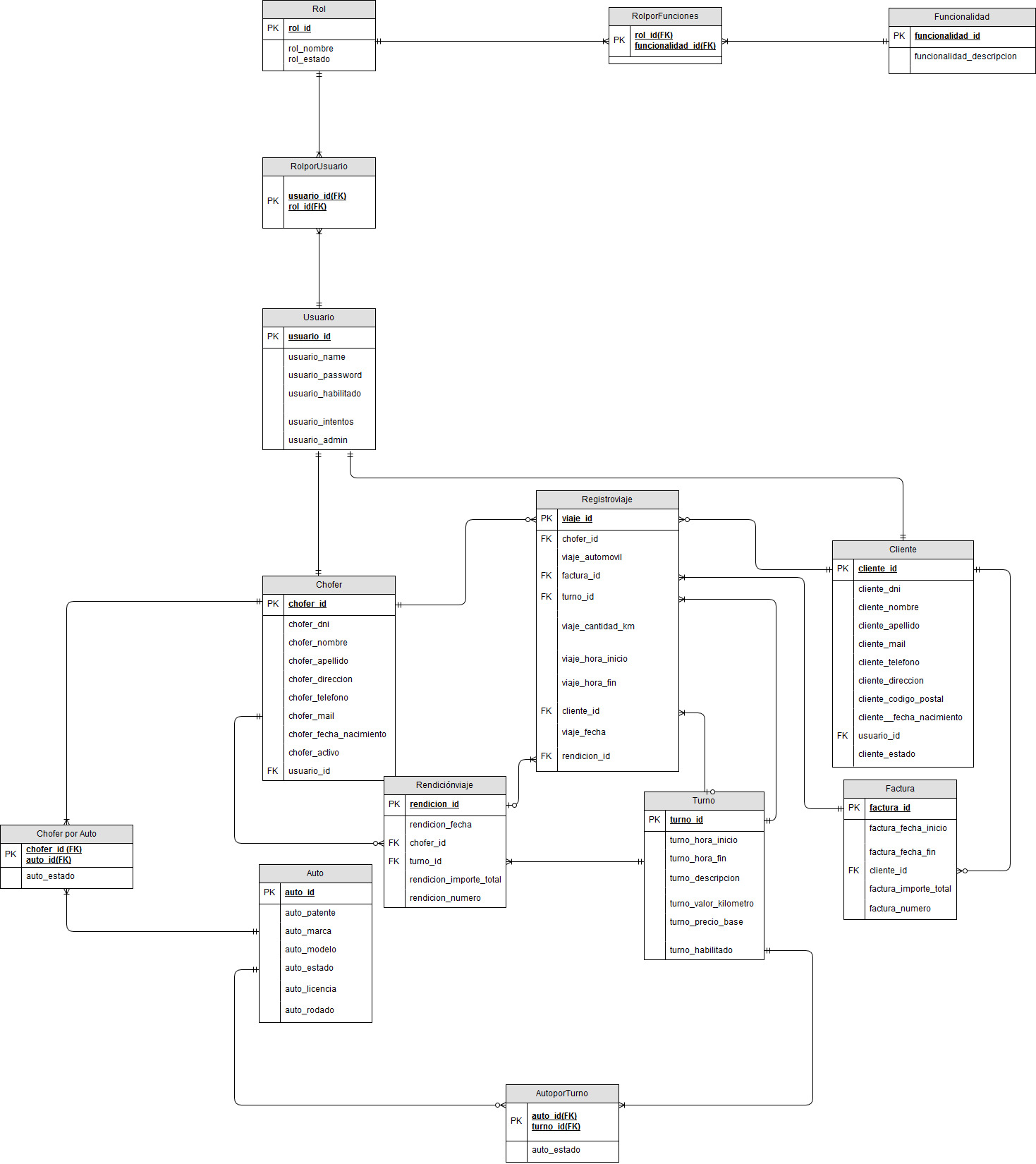
*Trabajo Práctico “UberFrba” (1C2017)*

Grupo N° 12 “Push\_it\_to\_the\_limit”

Integrantes:

* Amoroso, Lucas
* Ezeiza, Brian
* Marussi Lloret, Juan Martin
* Nobile, Nicolas

Índice:

**Diagrama de Entidad Relación:**

**Descripción de las entidades del DER:**

● **Funcionalidad**:  
 En esta entidad se almacenan todas las funcionalidades que se pueden realizar dependiendo del rol del usuario.

Esta entidad posee los siguientes campos:

* funcionalidad\_id : int – identity – Primary Key
* funcionalidad\_descripcion: varchar(100) - not null

Funcionalidades disponibles:

* ABM de Rol
* ABM de Clientes
* ABM de Automóviles
* ABM de Turnos
* ABM de Choferes
* Registro de Viajes
* Rendición de Viajes
* Facturación de Clientes
* Listado Estadístico

Consideraciones:

1. La funcionalidad de registro de usuario no se verá almacenada en esta entidad, ya que fue agregada en el formulario de login de los usuarios.

● **Rol**:   
En esta funcionalidad se almacenarán los distintos roles utilizados por los usuarios del sistema y si los mismos están habilitados o no para su uso.

Cuenta con los siguientes campos:

* rol\_id: INT IDENTITY(1,1) PRIMARY KEY
* rol\_nombre: VARCHAR(100) NOT NULL
* rol\_estado: BIT NOT NULL DEFAULT 1

Roles del Sistema:

* Administrativo
* Cliente
* Chofer

Consideraciones:

1. El campo “**rol\_estado**” se utiliza para comprobar si el rol está habilitado para ser asignado a un usuario (Al ser de tipo bit solo acepta los valores 1 o 0).

● **Rol por Funciones**:   
Funciona como intermediario entre las entidades Funcionalidad y Rol debido a la relación de muchos a muchos entre estas tablas.   
Como caso ejemplo, los roles cliente y chofer poseen el “Listado Estadístico” y el rol Administrativo tiene varios roles.

Esta entidad posee los siguientes campos:

* idFuncXRol: int identity(1,1) NOT NULL (No está en DER, pero si en migración ) REVISAR
* rol\_id: INTEGER - not null
* funcionalidad\_id: INTEGER - not null

Consideraciones:

1. Esta tabla cuenta con los campos “**rol\_id**” y “**funcionalidad\_id**” que serán foreign key de las tablas “**Rol**” y “**Funcionalidad**”.

● **Usuario:**Contiene a todos los usuarios que pueden utilizar el sistema.  
Desde el script inicial se incorpora un usuario Administrador, cuyo nombre es “user” y su contraseña es “w23e”.  
Los demás usuarios tendrán un usuario\_id con el cual se los identificará, además de nombre de usuario y contraseña (ambos coincidirán con el documento de quien se registre) la cual será encriptada con el algoritmo SHA256 y almacenada en la base de datos.  
Usaremos también el campo “**usuario\_intentos**” para registrar la cantidad de intentos fallidos con los que se intentó ingresar con ese nombre de usuario. Si algún usuario excediera los 3 intentos, se procederá a deshabilitarlo modificando el campo “**usuario\_habilitado**”.

Cuenta con los siguientes campos:

* usuario\_id: INT IDENTITY(1,1) PRIMARY KEY
* usuario\_name: VARCHAR(255) UNIQUE NOT NULL
* usuario\_password: NVARCHAR(255) NOT NULL
* usuario\_habilitado: [BIT] NOT NULL DEFAULT 1
* usuario\_intentos: [TINYINT] DEFAULT 0
* usuario\_admin: [BIT] NOT NULL DEFAULT 0 – REVISAR SI SE SIGUE UTILIZANDO

● **Rol por Usuario**:  
 Funcionará intermediario entre las entidades Usuario y Rol dado la relación de muchos a muchos entre esas tablas. Debido a que un Usuario puede tener más de 1 rol a la vez y un Rol puede estar presente en más de un usuario a la vez.

Cuenta con los siguientes campos:

* usuario\_id: INTEGER
* rol\_id: INTEGER

Esta tabla cuenta con los campos “**rol\_id**” y “**usuario\_id**” que son foreign key de las tablas “**Rol**” y “**Usuario**” y ambas forman una primary key compuesta.

● **Cliente:**

Contendrá a todos los clientes del sistema. Tendrá un ID para identificar a cada cliente además de una referencia al ID del usuario correspondiente.

Cuenta con los siguientes campos:

* cliente\_id: INT IDENTITY(1,1) PRIMARY KEY
* cliente\_nombre: VARCHAR(255) NOT NULL
* cliente\_apellido: VARCHAR(255) NOT NULL
* cliente\_mail: VARCHAR(255)
* cliente\_telefono: NUMERIC(18,0) UNIQUE NOT NULL
* cliente\_direccion: VARCHAR(255) NOT NULL
* cliente\_codigo\_postal: INT
* cliente\_fecha\_nacimiento: DATETIME NOT NULL
* cliente\_dni: NUMERIC(18,0) UNIQUE NOT NULL
* usuario\_id: INT NOT NULL
* cliente\_estado: BIT NOT NULL DEFAULT 1

Consideraciones:

1. Describir por que usamos ID cuando tenemos DNI?
2. El campo “**usuario\_id**” es foreign key de la tabla “**Usuario**” ya que un cliente es un usuario.
3. Debido a que los números de teléfono son únicos para cada cliente (campo con atributo UNIQUE), no pueden existir clientes con el mismo número de teléfono.  
   Tenemos la misma situación con el DNI.
4. El campo “**cliente\_estado**” se utiliza para comprobar si el cliente está habilitado, pudiendo realizar viajes, o si el mismo esta deshabilitado (Al ser de tipo bit solo acepta los valores 1 o 0).

● **Chofer:**

Contendrá todos los choferes del sistema.

Posee los distintos campos:

* chofer\_id: INT IDENTITY(1,1) PRIMARY KEY
* chofer\_dni: NUMERIC(18,0) UNIQUE NOT NULL
* chofer\_nombre: [VARCHAR](255) NOT NULL
* chofer\_apellido: [VARCHAR](255) NOT NULL
* chofer\_direccion: [VARCHAR](255)
* chofer\_telefono: NUMERIC(18,0) UNIQUE
* chofer\_mail: VARCHAR(50)
* chofer\_fecha\_Nacimiento: DATETIME NOT NULL
* chofer\_estado: [bit] NOT NULL DEFAULT 1
* usuario\_id: INT

Consideraciones:

1. Describir porque usamos ID si tenemos DNI?
2. El campo “**usuario\_id**” es foreign key de la tabla “**Usuario**” ya que un chofer es un usuario.
3. El campo “**chofer\_estado**” se utiliza para comprobar si el chofer está habilitado, pudiendo realizar viajes, o si el mismo esta deshabilitado (Al ser de tipo bit solo acepta los valores 1 o 0).
4. Al igual que “**Cliente**” los números de teléfono y los DNI son únicos para cada chofer.

● **Auto:**

Contendrá todos los autos del sistema que serán utilizados en los viajes.

Cuenta con los siguientes campos:

* auto\_id: INT IDENTITY(1,1) PRIMARY KEY
* auto\_patente: VARCHAR(10) UNIQUE NOT NULL
* auto\_marca: VARCHAR(255) NOT NULL
* auto\_modelo: VARCHAR(255) NOT NULL
* chofer\_id: int NOT NULL
* auto\_estado: BIT not null DEFAULT 1
* auto\_licencia: VARCHAR(255)
* auto\_rodado: VARCHAR(10)

Consideraciones:

1. El campo “**chofer\_id**” es foreign key que relaciona con la entidad “**Chofer**” ya que el auto es manejado por un chofer.
2. Debido a que en el sistema no pueden existir autos con la misma patente (autos mellizos) se le asigna el atributo UNIQUE a “**auto\_patente**”.

● **Chofer por Auto:**

Funcionará intermediario entre las entidades Chofer y Auto por la relación de muchos a muchos entre esas tablas. Debido a que un Chofer puede haber usado más de 1 Auto para un viaje y un Auto puede haber sido utilizado por más de un Chofer.

Cuenta con los siguientes campos:

* chofer\_id; INT
* auto\_id: INT
* auto\_estado: BIT not null DEFAULT 1

Consideraciones:

1. Esta entidad cuenta con los campos “**chofer\_id**” y “**auto\_id**” que son foreign key de las tablas “**Chofer**” y “**Auto**” y ambas forman una primary key compuesta.
2. “**auto\_estado**” es un campo utilizado para verificar que el auto tenga asignado un chofer (Al ser de tipo bit solo acepta los valores 1 o 0).

● **Turno:**

Contendrá todos los turnos del sistema en los que se realizarán los viajes.

Cuenta con los siguientes campos:

* turno\_id: INT IDENTITY(1,1) PRIMARY KEY
* turno\_hora\_inicio: NUMERIC(18,0) NOT NULL
* turno\_hora\_fin: NUMERIC(18,0) NOT NULL
* turno\_descripcion: VARCHAR(255)
* turno\_valor\_kilometro: NUMERIC(18,2) NOT NULL
* turno\_precio\_base: NUMERIC(18,2) NOT NULL
* turno\_habilitado: BIT NOT NULL DEFAULT 1

Consideraciones:

1. Utilizamos un campo “**turno\_habilitado**” debido a que si el sistema posee turnos que se superponen solo 1 de ellos va a estar habilitado en un instante dado.

● **Auto por Turno:**

Funcionará intermediario entre las entidades Auto y Turno por la relación de muchos a muchos entre esas tablas. Debido a que un Auto puede haber usado más de 1 Turno para un viaje y en un Turno puede haber sido utilizado más de un Auto.

Cuenta con los siguientes campos:

* auto\_id: int
* turno\_id: INTEGER
* auto\_estado: BIT not null DEFAULT 1

Consideraciones:

1. Esta entidad cuenta con los campos “**auto\_id**” y “**turno\_id**” que son foreign key de las tablas “**Auto**” y “**Turno**” y ambas forman una primary key compuesta.
2. “**auto\_estado**” es un campo utilizado para verificar que el auto tenga asignado un turno(Al ser de tipo bit solo acepta los valores 1 o 0).

● **Factura:**

Esta tabla contiene todos los datos pertinentes a la factura emitida a un cliente.

Cuenta con los campos:

* factura\_id: INT IDENTITY(1,1) PRIMARY KEY
* factura\_fecha\_inicio: DATETIME
* factura\_fecha\_fin: DATETIME
* factura\_numero: numeric(18,0) unique not null
* cliente\_id: INT NOT NULL REFERENCES [PUSH\_IT\_TO\_THE\_LIMIT].Cliente
* factura\_importe\_total: NUMERIC(18,2)

Consideraciones:

1. El campo “**cliente\_id**” es foreign key que relaciona esta tabla con la tabla “**Cliente**”

● **Rendición Viaje:**

Esta tabla contiene todos los datos pertinentes a la rendición emitida a un chofer.

Cuenta con los campos:

* rendicion\_id: INT IDENTITY(1,1) PRIMARY KEY
* rendicion\_fecha: DATETIME NOT NULL
* rendicion\_numero: numeric(18,0) not null
* chofer\_id: int NOT NULL
* turno\_id: int NOT NULL
* rendicion\_importe\_total: NUMERIC(18,2) NOT NULL

Consideraciones:

1. El campo “**chofer\_id**” es foreign key ya que relaciona con la entidad “**Chofer**” ya que se le hace la rendición a un chofer determinado y “**turno\_id**” también es foreign key ya que relaciona con la entidad “**Turno**” dado que se le hace la rendición por el turno.

● **Registro Viaje:**

Contiene toda información sobre los viajes que se realizaron entre clientes y choferes en un turno determinado.

Cuenta con los siguientes campos:

* viaje\_id: INT IDENTITY(1,1) PRIMARY KEY
* chofer\_id: INT NOT NULL
* viaje\_automovil: VARCHAR(8) NOT NULL,
* factura\_id: INT NOT NULL
* turno\_id: INT NOT NULL
* viaje\_cantidad\_km: NUMERIC(18,0) NOT NULL
* rendicion\_id: INT
* viaje\_fecha: VARCHAR(15) NOT NULL
* viaje\_hora\_inicio: VARCHAR(10) not null
* viaje\_hora\_fin: VARCHAR(10)
* cliente\_id: INT NOT NULL

Consideraciones:

1. El campo “**chofer\_id**” es foreign key, relaciona con la entidad “**Chofer**” debido a que un chofer determinado realiza el viaje.  
   El campo “**factura\_id**” es foreign key, relaciona con la entidad “**Facutra**” ya que al cliente se le va a facturar este viaje.  
   El campo “**turno\_id**” es foreign key, relaciona con la entidad “**Turno**” debido que el viaje se realiza en un turno determinado.  
   El campo “**rendicion\_id**” es foreign key, relaciona con la entidad “**Rendicion**” porque al chofer se le hace la rendición por el viaje realizado.  
   El campo “**cliente\_id**” es foreign key, relaciona con la entidad “**Cliente**” debido a que un cliente determinado realiza el viaje.
2. En los campos “**viaje\_fecha**”, “**viaje\_hora\_inicio**”, “**viaje\_hora\_fin**” en lugar de ser de tipo DATETIME usamos VARCHAR porque la función usada en la migración para obtener la fecha devuelve un VARCHAR.
3. “**viaje\_hora\_fin**” acepta valores de tipo NULL ya que en la tabla maestra esta columna está vacía.

**Stored Procedures:**

Alta de entidades: Utilizamos stored procedures del formato “crear\_entidad” para crear cada una de las entidades utilizadas por el sistema. Ej: “crear\_chofer, crear\_cliente”

“pr\_crear\_usuario\_con\_valores”, “InsertarFuncXRol”, “InsertarRol” no cumpliría con el patrón pero no creo que se fijen tanto

Acá se haría tedioso ir nombrando y describiendo uno por uno si no tenemos un formato general

Agregados a entidades: formato “pr\_agregar\_entidad\_a\_entidad”  
Ej: “pr\_agregar\_rol\_a\_usuario, pr\_agregar\_turno\_a\_automovil”  
Se consideraría modificación?

**Triggers utilizados:**

**actualizacion\_turno:** Al momento de actualizar un turno se lo deshabilita y se inserta un nuevo turno con los datos del original modificados

**insertar\_turno:** Este trigger tiene varias funcionalidades con respecto a verificar que a la hora de ingresar un turno…

1. Sus horarios no se superpongan con la de otro turno activo, de superponerse los horarios se realiza un rollback dejando todo como estaba.
2. No tenga exactamente los mismos datos que un turno ya presente en el sistema, ya sea un turno habilitado o deshabilitado.
3. Sus horarios sean valores coherentes.

Si no se pudiera agregar el turno el trigger lanzara una excepción.

Observación: Tanto los Stored Procedures como los triggers están contenidos en el script inicial. No en la aplicación desktop, aunque esta sea la que los utilice.

**Consideraciones ABMs:**

Para las altas de chofer, cliente, turno, automóvil y usuario (Facturacion, Rendicion y viaje?) utilizamos los stored procedures y triggers antes mencionados además para las modificaciones y bajas usamos querys realizadas por la aplicación desktop hacia la BD.  
IMPORTANTE: Acá tendríamos problemas de consistencia de porque no lo hacemos todo en el script o todo desde querys y seguro no les va a cerrar.

●**ABM Auto**

1. Editar auto (botón modificar de la grilla de autos):
   1. Al el campo chofer si el DNI ingresado es el mismo que estaba, se habilita el chofer para ese auto en caso que estuviese deshabilitado, sino quiere decir que se modificaron cualquiera del resto de los datos del auto.   
      SI el DNI ingresado es distinto al que estaba, se deshabilita el chofer anterior para este auto y además se ingresa en la tabla “choferporAuto” el nuevo chofer para ese auto.
   2. Cada vez que se intenta agregar un chofer a un auto, tanto sea habilitando uno que ya existía o asignando uno nuevo, se valida que el chofer ya no tenga un coche activo.
   3. Cuando se le intenta agregar un auto a chofer que ya tiene uno activo muestra un mensaje de error y no deja guardar cualquier cambio al auto
   4. Es importante entender que por ejemplo cuando yo solo le modifico la marca a un auto al hacer click en guardar se va a intentar habilitar ese chofer a ese auto (no importa si no lo modifique ) y Por ahí no me deja porque ya le asigne otro coche activo al chofer . (Valga la redundancia No entiendo)
   5. Si no se quiere agregar ningún turno nuevo, en el combobox turno de esta pantalla se debe seleccionar la opción de ninguno.
   6. Si al modificar un auto se desea deshabilitar al chofer se tilda el checkbox de "Deshabilitar Chofer".
2. Mostrar Autos(FiltroAuto)
   1. En el menú del ABM, hay un checkbox que se pone en true (tilde) si el auto está habilitado y además si el chofer tiene habilitado ese auto. lo mismo ocurre para los turnos que tiene asociado este auto.

**Base de datos:**

**Auto:**

-Un auto puede tener varios turnos, habilitados o deshabilitados  
-Un auto puede tener varios choferes   
-Un chofer no puede tener más de un auto activo (puede tener más pero el resto tienen que estar deshabilitados para ese chofer)