Proyecto Final

# Descripción de la temática

### Introducción

Tengo un micro-emprendimiento en casa de una peluquería, la cual es manejada por mi hermana que se comunica con sus clientes a través del uso de las apps de mensajería o redes sociales también, tanto para ofrecerles productos como para coordinar citas. Observo que podría ser más eficiente, ordenada y obtener mayor información del negocio si puede tener una página o app específicamente del local. Por tal motivo, me interesa poder armar una base de datos que contenga todas las relaciones que necesito para el armado de éste proyecto, en un principio buscare poder vender los productos de la peluquería y luego poder ir sumando, el tema de las citas, y como controlar a los peluqueros que trabajen en el local.

### Objetivos

Obtener mayor información del negocio (estadísticas de ventas, citas, ventas de los productos) y poder registrar las compras, servicios y citas de manera más clara tanto las fechas como su valor monetario. Logrando éste objetivo, busco poder abrir mi cabeza para la creación de otros proyectos de mayor envergadura y con ellos poder adentrarme en el ruedo de la programación y análisis de datos.

### Situación problemática

En éstos momentos la peluquería con la cual estoy trabajando no tiene ningún sistema, por lo tanto, no se tiene una clara información de los movimientos en la misma tanto por citas como por ventas. Tenemos citas que se contraponen muchas veces porque todo está escrito en papel o en el celular de una persona en particular sin tener con facilidad el acceso a un registro de todas las citas pactadas, permitiendo con ellos, ofrecer de manera correcta el día y horario para ser atendidos. Las ventas no están siendo controladas como tampoco tenemos un control de los productos que más se están vendiendo, lo que permitirá poder focalizar las compras y ser más eficientes en el local.

### Modelo de negocio

Con éste proyecto buscamos poder generar una mayor organización y eficiencia para el negocio, dejando atrás el uso de Excel y las cargas manuales diarias, por un sistema que permitirá:

* Cargar, modificar y eliminar Productos ofrecidos por el local.
* Cargar, modificar y eliminar servicios ofrecidos por el local.
* Tener un mayor control de los empleados y las tareas designadas para cada uno.
* Generar informes de ventas y servicios con los cuales se podrán tomar decisiones para el futuro del negocio.
* También podrán obtenerse estadísticas de los empleados que permitirán ver la eficiencia de los mismos.
* Estará diseñado para que cualquier usuario con una breve capacitación pueda usarlo, generando que no sea necesario tener grandes conocimientos técnicos para su utilización.
* Se podrá evitar perder información, porque todo tendrá un back up en la nube que permitirá incluso poder hacer consultas por fuera del local y con ellos trabajar desde donde sea más cómodo.
* Tendrán una administración que permitirá tener control de los usuarios que ingresen al sistema, no teniendo todos los mismos privilegios para borrar o editar e incluso poder ver o generar informes.

1. Diagramas entidad relación

<https://app.diagrams.net/#G15UG5BXzNau1tCvvkn-PzDL51XkGGUqmj>

1. Listado de tablas

Archivo Excel:” Descripcion\_TbPeluquerias\_CanoGabriel”

















1. Vistas

Archivo de creación: “Create\_View\_db.sql”

Vista 1: tb\_agrupodo\_por\_mes\_gastos\_clientes

Descripción: es una vista que permite visualizar, el total de gastos realizado de cada cliente por mes, mostrando, id\_cliente, nombre, apellido.

Objetivo: Visualizar el total de gastos mensuales por cliente.

Tablas/Datos: Las Tablas utilizadas son tb\_clientes,tb\_citas, tb\_citas\_serv y tb\_servicios.

Las cuales son unidas a través de Join, haciendo las siguientes combinaciones:

* Tb\_clientes unida con Tb\_citas a través del campo “id\_clientes”.
* Tb\_citas unida a Tb\_citas\_serv a través del campo “id\_citas”.}
* Tb\_citas\_serv unida a Servicios a través del campo “id\_servicios”

Luego usamos un Group by por el campo “Id\_clientes” de la Tb\_clientes y por mes usando la función “MONTHNAME” en el campo “Fecha” de la Tb\_citas

Los datos de cada tabla que se utilizan con sus respectivas funciones para esta vista son:

* `peluqueria`.`clientes`.`id\_Clientes` AS `id\_Clientes`,
* `peluqueria`.`clientes`.`Nombre` AS `Nombre`,
* `peluqueria`.`clientes`.`Apellido` AS `Apellido`,
* MONTHNAME(`peluqueria`.`citas`.`Fecha`) AS `MES`,
* SUM(`peluqueria`.`servicios`.`Precio`) AS `Total\_Gastado`

Vista 2: tb\_cantidad\_citas\_clientes

Descripción: es una vista creada para poder visualizar la cantidad de citas que tienen cada cliente

Objetivo: poder visualizar la cantidad de citas que tienen cada cliente

Tablas/Datos: Las tablas utilizadas son:

* Clientes
* Citas

Realizamos un Join con la tabla “Citas”, para poder contar las citas de cada cliente.

Luego usamos un Group by donde agrupamos por los siguientes campos y sus tablas:

* `clientes`.`id\_Clientes` ,
* `clientes`.`Nombre` ,
* `clientes`.`Apellido`

Los datos utilizados con sus respectivas funciones son los siguientes:

* `peluqueria`.`clientes`.`id\_Clientes` AS `id\_Clientes`,
* `peluqueria`.`clientes`.`Nombre` AS `Nombre`,
* `peluqueria`.`clientes`.`Apellido` AS `Apellido`,
* COUNT(`peluqueria`.`citas`.`id\_Citas`) AS `count(citas.id\_Citas)`

Vista3: tb\_clientes\_edad

Descripción: Es una lista que permite visualizar la edad de cada uno de los clientes.

Objetivo: Generar una vista que rápidamente permita ver los clientes más longevos.

Tablas/Datos: La tabla que se utiliza para generar la vista es Clientes, utilizando las siguientes columnas:

* `peluqueria`.`clientes`.`id\_Clientes` AS `id\_clientes`,
* `peluqueria`.`clientes`.`Nombre` AS `Nombre`,
* `peluqueria`.`clientes`.`Apellido` AS `Apellido`,
* (YEAR(NOW()) - YEAR(`peluqueria`.`clientes`.`Fecha\_Nacimiento`)) AS `Edad`

Generamos un campo llamado “Edad” realizando una operación que consiste en restar el año actual (obtenido usando la función Now() combinado con Year() para obtener solo el año) menos el año recibido en el campo “Fecha\_Nacimiento” donde usamos la función Year() para obtener solo el año.

Vista 4: tb\_detalles\_gastos\_clientes

Descripción: Es una vista que permite visualizar un detalle de gastos en los servicios por cliente, donde al estar ordenado de mayor a menor (Desc) por el campo “precio” de la tabla de “servicios”, podemos ver los cliente que gastaron más en servicios..

Objetivo: tener un detalle de los gastos de cada cliente en los servicios de la peluquería

Tablas/Datos: Las Tablas utilizadas son Clientes, Citas, Citas\_serv y Servicios.

Las cuales son unidas a través de Join, haciendo las siguientes combinaciones:

* Tb\_clientes unida con Tb\_citas a través del campo “id\_clientes”.
* Tb\_citas unida a Tb\_citas\_serv a través del campo “id\_citas”.}
* Tb\_citas\_serv unida a Servicios a través del campo “id\_servicios”

Los datos de cada tabla que se utilizan con sus respectivas funciones para esta vista son:

* `clientes`.`id\_Clientes` AS `id\_Clientes`,
* `clientes`.`Nombre` AS `nombre`,
* `clientes`.`Apellido` AS `Apellido`,
* `servicios`.`Nombre` AS `Servicio`,
* `citas`.`Fecha` AS `Fecha`,
* `servicios`.`Precio` AS `Precio``

Vista 5: tb\_total\_compras\_x\_proveedor

Descripción: Es una vista que nos permite ver el total de compras realizado por mes a cada proveedor.

Objetivo: Nos permite ver el total de compras realizado por proveedor, para identificar fácilmente el proveedor con el que más estamos invirtiendo.

Tablas/Datos: Las Tablas utilizadas son Proveedores y Detalles\_Compras. Las cuales son unidas a través de Join a trevés del campo “id\_proveedores”.

Los datos de cada tabla que se utilizan con sus respectivas funciones para esta vista son:

* `proveedores`.`Nombre` AS `Nombre`,
* MONTHNAME(`detalles\_compras`.`Fecha\_Compra`) AS `MES`,
* SUM(`detalles\_compras`.`Subtotal`) AS `Total\_Compras`

Genero un campo “Mes” para obtener el nombre del mes, aplicando la función “MONTHNAME” en el campo Fecha\_Compras de la tabla “Detalles\_Compras” y también, generamos otro campo llamado “Total\_Compras”, en el cual aplicamos la función SUM() al campo Subtotal de la tabla de “detalles\_compras”, de este modo obtenemos el total de compras en dinero.

Además, realizo un Group by por los siguientes campos:

* `Proveedores`.`Nombre`
* MONTHNAME(`detalles\_compras`.`Fecha\_Compra`)

Vista 6: tb\_Product\_mas\_vendido

Descripción: Es una vista que nos permite ver los productos que más se están vendiendo, junto con el valor monetario y cuando fue la última vez que se vendió dicho producto.

Objetivo: Poder tener un control de cuáles son los productos más se están vendidos en el local.

Tablas/Datos: Las Tablas utilizadas es “detalles\_ventas”.

Los datos de cada tabla que se utilizan con sus respectivas funciones para esta vista son:

* `id\_productos`,
* ‘count(id\_productos) AS Cant\_Vendidos’
* Sum(Subtotal) AS Ultim\_Vent

Realizo también un Group by por id\_productos para terminar ordenando el campo (alias en éste caso) “Cant\_vendidos” de manera descendente (desc).

1. Funciones

Archivo de creación: “Create\_Functions.sql”

Función 1: Calcular\_edad

Descripción: En este caso realicé una función que me permite calcular la edad enviando la fecha de nacimiento, con lo cual devuelve un valor entero indicando la edad o mejor dicho la diferencia entre la fecha enviada y la fecha actual.

Objetivo: Permitirme obtener la edad de los clientes o empleados sabiendo la fecha de nacimiento o incluso, poder usarla para medir la antigüedad de un cliente o empleado también.

Tablas/Datos: realice el siguiente calculo en la función:

*Resultado = (YEAR(NOW()) - YEAR(fecha\_recibida));*

Función 2: Calcular\_Descuento

Descripción: En este caso es una función que recibe 2 valores float, donde uno de los valores será el precio y el otro el porcentaje de descuento.

Objetivo: obtener fácilmente el precio de un producto aplicando el descuento que necesitemos a un producto y donde además ya tenga incluido el IVA también (por defecto la función tiene un 0.1 de IVA).

Tablas/Datos: realizamos el siguiente calculo con las variables float que recibimos en la función.

*Precio\_con\_Descuento= valor\_producto + iva - (valor\_producto \* porcentaje\_descuento);*

Return *Precio\_con\_Descuento;*

1. Stored Procedures

Archivo de creación: “Create\_Stored\_Procedures.sql”

Procedure1: Ingresar\_New\_Clientes:

Descripción: permite poder realizar un Insert a la tabla Clientes, donde exige para su carga que los campos indicados sean distintos de vacíos, para poder ejecutarse con éxito la carga, devolverá una variable que tendrá un mensaje indicando que la carga fue exitosa y además se ejecuta una consulta a la tabla "clientes” ordenada por id\_cliente de manera Desc con lo cual podremos ver el registro cargado. Ahora bien, si alguno de los campos ingresados está vacío, obtendremos en la variable de salida un mensaje indicando que la carga no fue exitosa y que tienen que completar todos los campos.

Objetivo: Poder realizar insert en la tabla cliente de manera más simple.

Tablas/Datos:

Utilizamos la tabla clientes y solicitamos los siguientes campos en el stored procedure para le **Insert** de nuevos clientes:

* **IN** id\_dni varchar(20),
* **IN** nomb varchar(50),
* **IN** apell varchar(50),
* **IN** correo varchar(100),
* **IN** f\_ingreso date,
* **IN** f\_nacimiento date,
* **OUT** Respuesta varchar(100)

Para poder asegurarnos que las fechas ingresadas por el usuario en los campos “f\_nacimiento” y “f\_ingreso” tenga el formato de mysql, utilizamos la función **str\_to\_date() para convertir la misma en el formato correcto.**

Procedure2: ORDENAR\_Campo\_Table

Descripción: Permite ordenar el campo de la tabla que necesitemos y lo va hacer de manera ascendente o Descendente, dependiendo del tercer valor que pasemos siendo “A” para Ascendente y “D” para Descendente, de compartir cualquier otro valor tomara el valor por defecto de la consulta.

Resuelto el stored procedures mostrara la tabla y el campos que pasamos ordenado según el valor que hayamos enviado.

Objetivo: poder obtener de manera rápida los datos de una tabla ordenados según la necesidad que tengamos.

Tablas/Datos: Solicitamos los siguientes campos en el stored procedures para poder hacer el ordenamiento:

* Campo\_tabla (campo de la tabla de productos por el cual queremos ordenar)
* Tabla (el nombre de la tabla que estamos por ordenar)
* tipo (tipo de ordenamiento A=Ascendente o D=Descendente)

Procedure3: Verif\_Stock\_Prod

Descripción: Permite controlar si tenemos stock del Id del producto que le pasemos como parámetro. El mismo necesita como parámetros el id del producto y una cantidad. Con estos datos el SP verifica en la tabla productos si contiene la cantidad del id producto solicitado. Si existe y tenemos la cantidad solicitada devolverá en la variable de Salida (@result) el valor 1, de lo contrario mostrará 0 (cero).

Objetivo: poder realizar el control de stock y con ello obtener una bandera que me puede servir para otros SP o procesos que necesite realizar en la base de datos.

Tablas/Datos: Utilizamos las tablas **productos** y solicitamos los siguientes campos en el SP:

* **IN** id\_prod **INT**
* **IN** cantCtrl **INT**
* **OUT** Result **INT**

Procedure4: Carga\_Ventas\_Product

Descripción: Permite cargar la venta de un producto. Utilizamos TCL dentro del SP, para iniciar una transacción y si no se obtiene ningún error confirmamos los cambios utilizando “--Commit”.

1 - Solicitaremos para la carga los siguientes parámetros:

1. id\_product (Es el id del producto que se está vendiendo),
2. Cant\_prod (es la cantidad que se está vendiendo del producto),
3. id\_Empleado (agregamos el Id del empleado que está realizando la venta),
4. bandera\_ventas (donde tendremos que poner el valor 0 (cero) si estamos cargando el primer producto de una nueva venta, de lo contrario, tendremos que poner el valor 1 (uno), si estamos cargando el producto de una venta que aún se está llevando a cabo. Si enviamos un valor diferente los mencionados obtendremos un error al ejecutar el SP.

2 - Pasado los parámetros de la carga verificaremos utilizando otro SP (Verif\_Stock\_Prod) para controlar si tenemos stock del producto que estamos vendiendo para realizar la transacción.

3 - En caso de tener el stock tenemos que restar de la tabla productos la cantidad que se está comprando

4 - Agregamos en la tabla Ventas un registro con una ID de ventas.

5 - Agregamos un registro por cada producto que se compró en la tabla "detalles\_ventas" donde repetiremos o no según sea el caso el ID de la venta que fue agregado en la tabla "Ventas".

Una vez ejecutado el SP podemos obtener algunos de los siguientes mensajes:

* Si la carga es correcta mostrara “Carga OK de la venta”.
* Si no tenemos stock del producto que estamos cargando mostrara “No tiene Stock para realizar la venta”
* Si enviamos mal el valor de la bandera de venta, nos mostrara “ERROR en el valor enviado de bandera\_ventas (es el cuarto valor puesto en el SP”).

Además, el SP también cuanta con una variable de salida (Resp) donde mostrara el valor 1(uno) si la carga fue exitosa de lo contrario mostrara 0 (cero).

Objetivo: poder realizar la carga de las ventas en la base de datos.

Tablas/Datos: Utilizamos las tablas **productos**, **ventas** y **detalles\_ventas** y solicitamos los siguientes campos en el SP:

* **IN** id\_product **INT**
* **IN** Cant\_prod **INT**
* **IN** id\_Empleado **INT**
* **IN** bandera\_ventas **INT**
* **OUT** Resp **INT**

1. Trigger

Archivo creación: “Create\_Trigger.sql”

Trigger a la Tabla Productos

1. *Generamos 3 Trigger (after) en la tabla "****Productos****" para el* ***insert, update y delete****,*

Descripción: de éste modo guardo en una tabla de auditoría llamada “**\_audit\_new\_Productos**”, los campos que considero importante.

Objetivo: Poder tener un control de los cambios que se realizan en la tabla Productos.

1. *Creo un Trigger (BEFORE) en la tabla "****Productos****" para asegurarme que siempre el Precio\_unitario sea mayor a 0*

*.*

Descripción: Cuando agregamos un producto si no especificamos el stock el mismo se agrega por defecto en 1.

Objetivo: asegurarme que siempre el stock tenga mínimo 1 al momento de ser insertado un nuevo producto en la tabla Productos.

Trigger a la Tabla Clientes

1. Generamos un Trigger (after) en la tabla "**Clientes**" *para el* ***insert, update y delete****.*

Descripción: guardo en una tabla de auditoría llamada “**\_audit\_new\_clientes**” los campos que considero importante.

Objetivo: Poder tener un control de los cambios que se realizan en la tabla "clientes".

1. Archivos SQL

Script creación de tablas e inserción de datos:

* “Create\_Tables\_db.sql”
* “Script\_INSERT\_datos\_en\_db.sql”

Repositorio Gut Hub, donde podrán encontrar todos los Script:

1. El reporte generado en base a la información almacenada en las tablas.

Todas las vistas que generé me van a servir como informes, sin embargo, puedo destacar que las vistas “tb\_Product\_mas\_vendido” y “tb\_total\_compras\_x\_proveedor”, son las que ejecutare normalmente.

1. Herramientas y tecnologías utilizadas

* Git hub
* <https://app.diagrams.net/> (de google, es la que utilice para hacer el diagrama de E-R)
* Excel
* Word
* Whatsapp
* Pizarrón blanco