

CE3101 - Proyecto #2

# **Documentación evidencia de solución elaborada y planificación del trabajo**

---

Brayan Alfaro González



Julián Camacho Hernández  
Kevin Zeledón Salazar  
I Semestre 2021

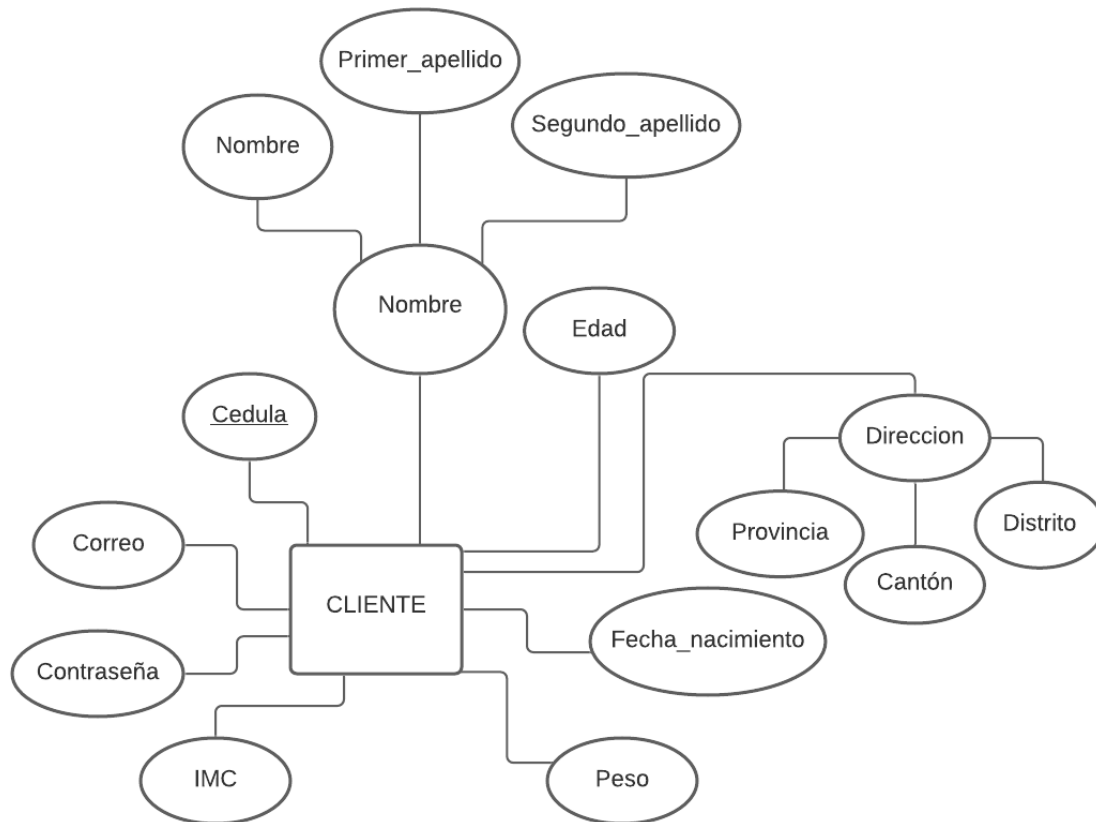


## Índice

<b>Índice</b>	<b>2</b>
<b>Modelo conceptual y diagrama relacional</b>	<b>3</b>
<b>Estructuras de datos desarrolladas</b>	<b>5</b>
<b>Descripción de los Stored Procedures, Triggers y Vistas implementados</b>	<b>8</b>
Stored Procedures	8
Triggers	9
Vistas	10
<b>Descripción detallada de la arquitectura desarrollada</b>	<b>11</b>
<b>Problemas sin solución</b>	<b>12</b>
<b>Problemas encontrados</b>	<b>12</b>
Problema 1:	12
Problema 2:	13
Problema 3:	13
<b>Conclusiones y recomendaciones</b>	<b>14</b>
Conclusiones:	14
Recomendaciones:	14
<b>Bibliografía</b>	<b>16</b>

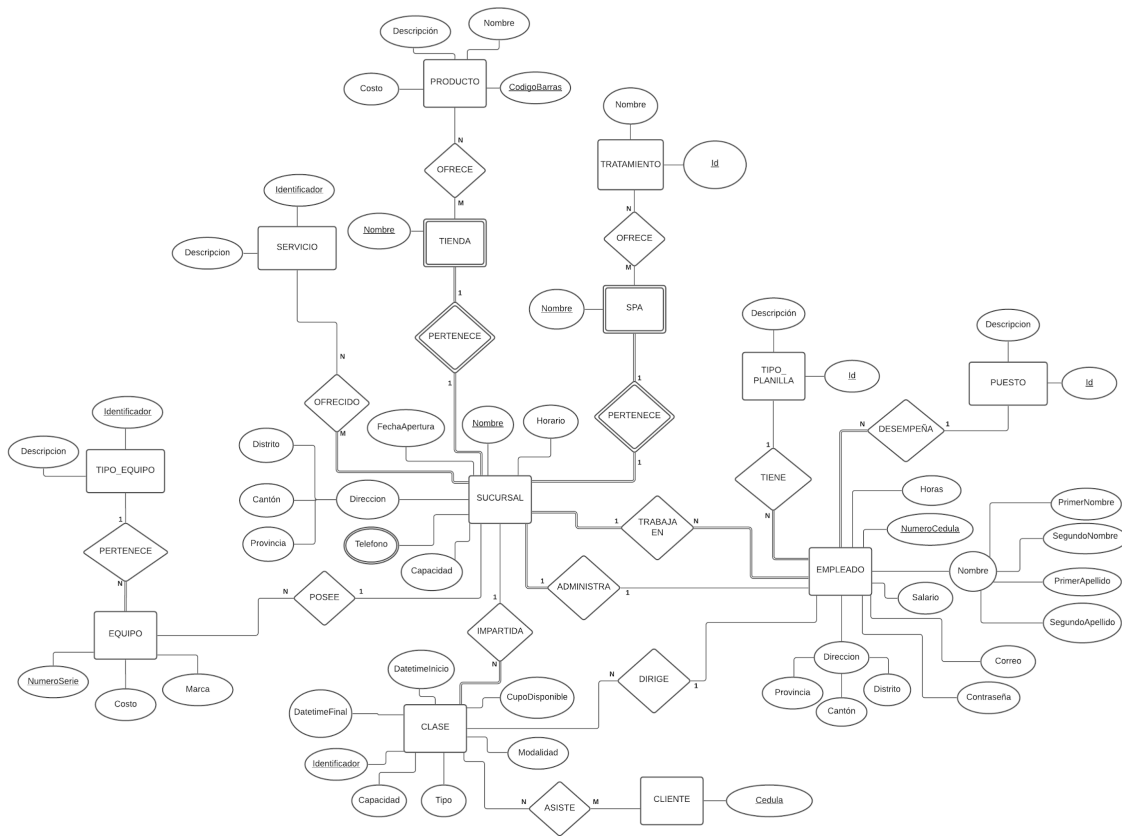
## Modelo conceptual y diagrama relacional

- Modelos del cliente
  - Modelo relacional

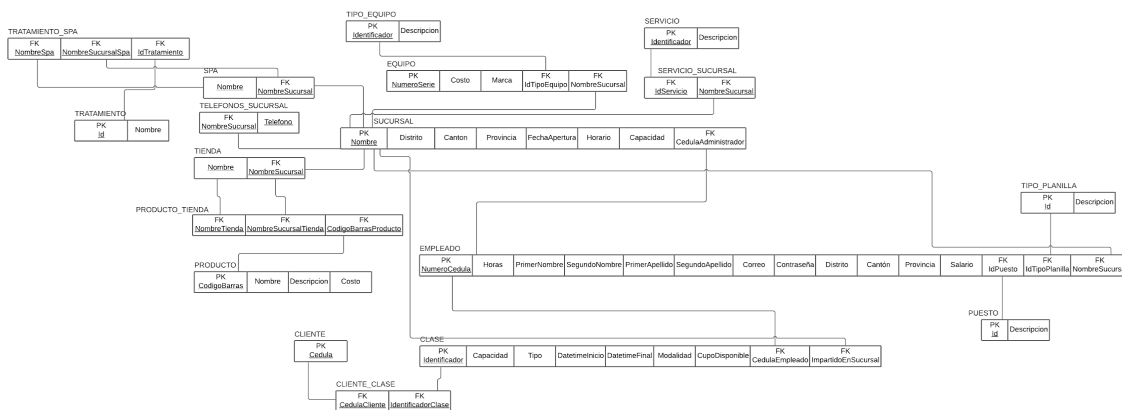


Cabe destacar que no existe diagrama relacional del cliente, debido a que esta estructura se va a desarrollar en una base de datos **no**-relacional, por lo que simplemente con el diagrama conceptual es suficiente para crear los modelos de la base de datos.

- Modelos del administrador
  - Modelo conceptual



- Diagrama relacional



Estos diagrama se puede apreciar mejor en su versión online en la siguiente dirección, bajo la pestaña de Administrador 2 y Cliente 2 para los diagramas del administrador y el cliente, respectivamente: <https://lucid.app/lucidchart/16744618-d731-497f-848a-685727e813ad/view>

## Estructuras de datos desarrolladas

En esta sección se presentan y se explica el uso de las estructuras de datos implementadas en el desarrollo de la tarea, específicamente las tablas de la base de datos.

Nombre	Descripción
SUCURSAL	Contiene todos los datos de los gimnasios/sucursales de la cadena. De ellos se almacena su nombre que debe ser único, datos de su ubicación como provincia, cantón y distrito, fecha de apertura, horario, capacidad y la cédula del empleado administrador.
EQUIPO	Almacena los datos de los equipos/máquinas que posee cada sucursal. Se guarda su número de serie único, costo, marca, tipo de equipo al que pertenece y el nombre de la sucursal en la que se encuentra.
TIPO_EQUIPO	Describe los tipos de las máquinas/equipos que pueden tener los gimnasios. De estos se conoce un identificador y una descripción.
SERVICIO	Contiene los datos de los servicios que se pueden ofrecer en cualquier sucursal de la cadena. De estos se conoce un identificador y una descripción.
SERVICIO_SUCURSAL	En esta tabla se determina la relación que existe entre los servicios y la sucursal, es decir, almacena cuáles servicios se ofrecen en cada sucursal. Para ello se guarda el identificador del servicio, así como el nombre de la sucursal.
SPA	Almacena los spa que son parte de las diferentes sucursales. Se conoce el nombre del spa y el nombre de la sucursal a la que pertenece.
TRATAMIENTO_SPA	Define los tratamientos que se brindan en cada uno de los spa. Para relacionarlos, se almacena el nombre del spa donde se ofrece, el nombre de la sucursal donde se encuentra dicho spa y el identificador del tratamiento.
TRATAMIENTO	Contiene los datos de los tratamientos que se pueden ofrecer en cualquier spa de cualquier sucursal de la cadena. De estos se conoce un identificador y un nombre.

TELEFONOS_SUCURSAL	Contiene los datos de los teléfonos que tiene cada una de las sucursales. Para relacionarlos, se almacenan cada uno de los teléfonos y el nombre de la sucursal a la que pertenecen.
TIENDA	Almacena las tiendas de conveniencia que son parte de las diferentes sucursales. Se conoce el nombre de la tienda y el nombre de la sucursal a la que pertenece.
PRODUCTO_TIENDA	Realiza la asociación entre los productos y las tiendas. Determina cuáles productos se venden en cada tienda. Para relacionarlos guarda el nombre de la tienda, el de la sucursal donde está la tienda y el código de barras de cada producto que se ofrece.
PRODUCTO	Describe los productos que pueden ser vendidos en las tiendas de conveniencia. De ellos se conoce su código de barras único, el nombre, una descripción y el costo por unidad.
EMPLEADO	Contiene todos los datos de los empleados de las sucursales. Se almacena su número de cédula, horas trabajadas, primer y segundo nombre, primer y segundo apellido, correo electrónico, contraseña, datos de su ubicación como provincia, cantón y distrito, salario, el identificador de su puesto, el de su tipo de planilla y el nombre de la sucursal donde labora.
TIPO_PLANILLA	Contiene los datos de los tipos de planilla a la que pueden pertenecer los empleados para su pago. De estos se conoce un identificador y una descripción.
PUESTO	Contiene los datos de los puestos que pueden ocupar los empleados. De estos se conoce un identificador y una descripción.
CLASE	En esta tabla se almacena la información de las clases que se imparten en las sucursales. Para determinarlo, se guarda un identificador, capacidad, tipo, fecha y hora de inicio, fecha y hora de finalización, modalidad, cupo disponible, cédula del empleado que la imparte, la sucursal donde se lleva a cabo y una imagen para su descripción.
CLIENTE_CLASE	Determina las clases a las que asiste un cliente. Para relacionarlas se almacena la cédula del cliente y el identificador de la clase a la que asiste.
CLIENTE	Contiene los datos de los clientes de las sucursales. Se



	guarda su número de cédula, nombre y apellidos, edad, fecha de nacimiento, peso, IMC, dirección, correo electrónico y contraseña.
--	---



## Descripción de los Stored Procedures, Triggers y Vistas implementados

### Stored Procedures

Nombre	Parametros	Descripción
GeneracionPlanilla		Este stored procedure se encarga de devolver la información de la planilla mensual, tomando en cuenta los diferentes tipos de planillas y los diferentes casos de cálculo.
EMPLEADOSelectExtended	@NumeroCedula: Cédula del empleado, o null para obtenerlos todos	Este stored procedure devuelve la información de los empleados y le añade los detalles del tipo de planilla y tipo de puesto.
CLASEFilter	@Cedula: Cédula del cliente @Tipos: Lista de tipos de clases @Sucursales: Lista de sucursales @Inicio: Fecha mínima @Fin: Fecha máxima	Este stored procedure se utiliza para obtener las clases existentes que cumplan con los criterios de búsqueda indicados, y en las que el usuario indicado no se encuentre registrado.
CLASESCliente	@Cedula: Cédula del cliente	Este stored procedure se usa para obtener las clases en las que un cliente se encuentra registrado.
TIPOS_CLASESelect		Este stored procedure devuelve los diferentes tipos de clase que existen.
SUCURSALCopy	Parámetros de los datos de la nueva sucursal: @NewNombreSucursal @NewDistrito @NewCanton @NewProvincia @NewFechaApertura @NewHorario @NewCapacidad @NewCedulaAdministra	Este stored procedure se encarga de crear una nueva sucursal con los datos que se brindan como parámetros. Posteriormente, toma el nombre de la sucursal de la cual se quiere copiar los datos y se copia lo siguiente: los tratamientos del Spa, los productos de la tienda, y las clases que se imparten.

	dor @OldNombreSucursal: Nombre de la sucursal de la cual se quiere copiar los datos	
CLASESCopy	@fecha_inicio_copiar: fecha desde la cual se quieren copiar las actividades @fecha_final_copiar: fecha hasta la cual se quieren copiar las actividades @diff: diferencia entre la fecha de inicio de copiar y fecha de inicio donde se van a pegar @sucursal: nombre de la sucursal de la cual se quiere copiar sus clases	Este stored procedure se encarga de copiar las actividades de un periodo a otro. Toma las fechas por copiar y crea nuevas clases, cuya fecha es la fecha de inicio más la diferencia. Si se indica el nombre de la sucursal, solo copia las clases de ese gimnasio.
TratamientoUnassigned		Devuelve los tratamientos que no están asignados a ningún spa
EquipoUnassigned		Devuelve el equipo que no ha sido asignado a ninguna sucursal
ProductoUnassigned		Devuelve los productos que no están asignados a ninguna sucursal

\* Adicionalmente, se implementaron stored procedures para todas las operaciones de CRUD en las diferentes tablas.

## Triggers

Nombre	Descripción
ReducirCuposDisponibles	Este trigger se encarga de disminuir el cupo disponible en una clase luego de que se realice una inserción en la tabla CLIENTE_CLASE.
PrimerosTratamientos	Crea los tratamientos iniciales de cada spa después de la creación de un spa.
ProteccionPrimerosTratamientos	Evita que se actualice un tratamiento con el

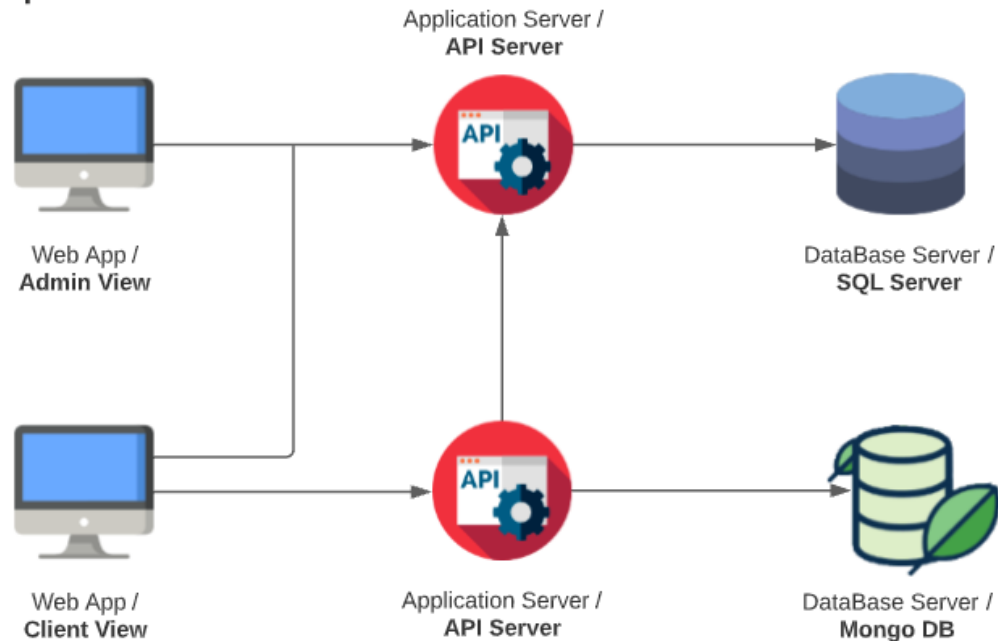
	nombre de los primeros tratamientos (“Masaje Relajante”, “Masaje Descarga Muscular”, “Sauna” o “Baños a vapor”)
--	---

## Vistas

Nombre	Descripción
EMPLEADO_CLASES	Esta vista consiste en el nombre y número de cédula de los empleados que se les paga por clase, así como el número de clases dadas en el último mes.
EMPLEADO_HORAS	Esta vista consiste en el nombre y número de cédula de los empleados que se les paga por hora, así como el número de horas impartidas en el último mes.
PLANILLA_MES	Esta vista da los datos de planilla para aquellos empleados que laboran por mes.
PLANILLA_HORAS	Esta vista da los datos de planilla para aquellos empleados que laboran por horas, tomando en cuenta la vista EMPLEADO_HORAS.
PLANILLA_CLASES	Esta vista da los datos de planilla para aquellos empleados que laboran por clases, tomando en cuenta la vista EMPLEADO_CLASES.

## Descripción detallada de la arquitectura desarrollada

### Arquitectura del Software



El software para este proyecto consta de dos aplicaciones web, que son la vista de administrador y la vista de cliente. Mediante estas apps, los usuarios clientes y administradores son capaces de gestionar y acceder a toda la funcionalidad del software.

La aplicación de administrador se conecta a un API que se encarga de procesar la información e interactuar al ser conectado con una base de datos desarrollada en SQL Server. La vista del cliente se conecta a dos APIs, la anteriormente mencionada que se relaciona con SQL Server y otra que se encarga de gestionar la información con una base de datos no relacional de MongoDB. A su vez, ambos APIs comparten información e interactúan entre sí.

## Problemas sin solución

- El tamaño de la imagen de fondo en la appweb del cliente a veces no es lo suficientemente larga para las dimensiones del dispositivo. Se intentaron usar diferentes configuraciones de css pero nunca se encontró una solución al problema.

## Problemas encontrados

- **Problema 1:**
  - Descripción: Para el despliegue y copia de actividades en el calendario se presentaron problemas con los tipos de datos de fecha. Esto se produjo porque el tipo Date en typescript no es compatible con el tipo datetime que se trabaja en la base de datos y que recibía el stored procedure de copia de actividades. Esto debido a que se observó que se utilizan formatos diferentes como el ISO8601 o el datetime ('DD/MM/YYYY HH:mm:ss:nnnn')
  - Intentos de solución: Se intentó enviar desde el frontend al api las fechas por copiar en tipo Date, pero al convertirlo en este tipo de dato, se guardaba en un formato no compatible para realizar el stored procedure correctamente.
  - Solución encontrada: La solución encontrada fue enviar desde el frontend un string de la fecha en el formato datetime que es compatible con la base de datos. De esa manera la base lo interpretaba como una fecha y se lograba ejecutar el stored procedure. Y para recibir datos de fechas desde el api en el frontend, se volvía a hacer la conversión de tipo de dato para ser desplegado correctamente.
  - Recomendaciones y conclusiones:
    - Se concluye que los tipos de datos de fecha pueden necesitar conversiones dependiendo del formato que se necesite.
    - Se recomienda el uso de un único formato en todas las partes de las aplicaciones para evitar errores de este tipo.
  - Bibliografía consultada:
    - typescript/javascript, C. and Aygun, A. (2019) Converting ISO-8601 string to local date in typescript/javascript, Stack Overflow. Disponible en: <https://stackoverflow.com/questions/54250630/converting-iso-8601-string-to-local-date-in-typescript-javascript> [Accesado: 11 de junio de 2021].

- angular-calendar, L. (2017) Load events from db on angular-calendar, Stack Overflow. Disponible en:  
<https://stackoverflow.com/questions/46455828/load-events-from-db-on-angular-calendar> [Accesado: 11 de junio de 2021].
- JavaScript, C., Pandey, M. and Rajagopal, P. (2014) Change ISO Date String to Date Object - JavaScript, Stack Overflow. Disponible en:  
<https://stackoverflow.com/questions/27012854/change-iso-date-string-to-date-object-javascript> [Accesado: 11 de junio de 2021].

## ● Problema 2:

- Descripción: Se requería pasar una lista de datos por medio de un parámetro a un stored procedure definido en la base de datos; sin embargo, no existen métodos internos de SQL para hacer esto posible.
- Intentos de solución: Se intentó buscar algún tipo de dato que sirviera como lista, pero no se encontró ninguno.
- Solución encontrada: La solución encontrada consistió en generar un nuevo tipo de dato, el cual se define como una tabla que cuenta con un solo atributo. De esta manera, cuando se desea enviar una lista de datos al stored procedure, estos se agregan a una variable definida por el tipo de dato creada, y se envía como parámetro. Para utilizar los datos dentro del stored procedure se realizan operaciones de SQL sobre el parámetro recibido.
- Recomendaciones y conclusiones:
  - Se concluye que en ciertas situaciones es necesario introducir nuevos tipos de datos en SQL para satisfacer necesidades para las cuales no se encuentra una solución implementada nativamente.
  - Se recomienda realizar una investigación exhaustiva antes de introducir nuevos tipos de datos, ya que si no se siguen las buenas prácticas se podría caer en algún tipo de redundancia con los tipos de datos, lo que afectaría el buen desarrollo de la base de datos.
- Bibliografía consultada:
  - A. Little, "How do I pass a list as a parameter in a stored procedure?", *Stack Overflow*, 2017. [En línea]. Disponible en:  
<https://stackoverflow.com/questions/42448596/how-do-i-pass-a-list-as-a-parameter-in-a-stored-procedure>. [Accessed: 15- Jun- 2021].

## ● Problema 3:

- Descripción:
- Intentos de solución:

- Solución encontrada:
- Recomendaciones y conclusiones:
  - Se recomienda i
- Bibliografía consultada:
  -


## Conclusiones y recomendaciones

### Conclusiones:

- Se concluye que el desarrollo de front end es una actividad especializada que no se debe subestimar, ya que existen muchos factores a investigar con respecto a las diferentes herramientas disponibles, como css, html, angular, bootstrap, etc. Esto dado que se comprobó que el diseño visual de un sitio web es una labor no trivial que requiere de conocimientos de diseño gráfico, y de las herramientas necesarias para materializar dicho diseño.
- Se concluye que el uso de las herramientas proveídas por .NET Framework para la conexión a la base de datos facilitan el desarrollo del backend, ya que reducen la carga de los programadores en tareas que son triviales, como la conexión a los stored procedures.
- Se concluye que el uso de stored procedures, triggers y vistas pueden facilitar mucho tareas que son muy repetitivas dentro de la base de datos y permiten un mejor orden y manejo de los datos dentro de la misma.

### Recomendaciones:

- Se recomienda que al desarrollar un proyecto web, hayan programadores especializados en la parte de front end para así agilizar el avance del proyecto, así como para lograr un mejor resultado visual.
- Se recomienda el uso de ADO.NET, proveído por .NET Framework, para la conexión desde la aplicación .NET a la base de datos. Ya que mediante esta herramienta se generan de manera sencilla los diferentes modelos en C# concernientes a las tablas de la base de datos y los objetos retornados por los stored procedures.

- 
- Se recomienda el uso de las diferentes herramientas de sql server management studio para el mantenimiento y la creación de las bases de datos hechas en sql server, debido a que proveen mucha funcionalidad para el programador.





## Bibliografía

- [1] "Angular", *Angular.io*. [En línea]. Disponible en: <https://angular.io/guide/component-interaction#parent-and-children-communicate-via-a-service>. [Accesado: 15- Jun- 2021].
- [2] "HTML 5.2", *W3.org*, 2021. [En línea]. Disponible en: <https://www.w3.org/TR/html52/>. [Accesado: 15- Jun- 2021].
- [3] ".NET documentation", *Docs.microsoft.com*, 2021. [En línea]. Disponible en: <https://docs.microsoft.com/en-us/dotnet/> [Accesado: 15- Jun- 2021].
- [4]"SQL Server technical documentation - SQL Server", *Docs.microsoft.com*. [En línea]. Disponible en: <https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15>. [Accesado: 15- Jun- 2021].