

Adrián Hernando Gavilán

Proyecto fin de ciclo D.A.M

Curso: 2020/21

ORGANIZADOR DE TORNEOS

MEMORIA

***ÍNDICE***

[***OBJETIVOS*** 2](#_Toc72914622)

[***PLANIFICACIÓN Y DESARROLLO*** 2](#_Toc72914623)

[***RECOGIDA DE INFORMACIÓN*** 3](#_Toc72914624)

[*WPF* 4](#_Toc72914625)

[*C#* 5](#_Toc72914626)

[*Patrón MVVM* 6](#_Toc72914627)

[*LINQ* 9](#_Toc72914628)

[*Material Design XAML* 10](#_Toc72914629)

[***BASE DE DATOS*** 11](#_Toc72914630)

[*Modelo relacional* 11](#_Toc72914631)

[*Diagrama de clases* 12](#_Toc72914632)

[***MANUAL DE USUARIO*** 14](#_Toc72914633)

[***REFERENCIAS BIBLIOGRÁFICAS*** 21](#_Toc72914634)

# ***OBJETIVOS***

En este punto, se indicarán los objetivos principales que se presentan para la realización de este TFC.

En este trabajo se encara la realización de una aplicación que permite hacer una amplia gestión de un torneo, campeonato, liga sobre un deporte. Para ello, los objetivos en que se basa mi aplicación son los siguientes:

* Aprovechar la aplicación para facilitar la gestión de dicho torneo por parte del administrador.
* Facilitar por parte del usuario la consulta de fechas, lugares, resultados y clasificaciones mediante una posterior aplicación.
* Unir la relación que puede tener el deporte con la tecnología.
* Ahorrar tiempo tanto para los usuarios como para el administrador.

Por otra parte, están los objetivos académicos de la realización de este TFC, en mi caso, estos objetivos serían los siguientes:

* Conocer y adaptarme a un nuevo lenguaje de programación como es C#.
* Estudiar y conocer el patrón MVVM, que es muy usado con dichas herramientas.
* Estudiar y conocer la API Linq.
* Profundizar más en el IDE Visual Studio.

# ***PLANIFICACIÓN Y DESARROLLO***

En este punto, se indicarán las etapas para el desarrollo y la planificación que he llevado a cabo para la realización del proyecto.

* En primer lugar, fue elegir si hacerlo para escritorio o móvil, ya que en mis practicas del ciclo trabajo en una aplicación para escritorio. Es por ello que decidí enfocarlo para escritorio utilizando C# como lenguaje y Visual Studio como IDE con WPF.
* Después tuve que ponerme al día con el lenguaje mencionado anteriormente.
* De igual manera, ver las ventajas y conocer más a fondo cómo utilizar en WPF el lenguaje de marcado que se utiliza para la vista que es XAML.
* Elegir un patrón a usar, ya que me parece de lo más útil. (MVVM)
* Elegir dónde estaría alojada la base de datos. (Sql Server)
* Buscar una herramienta para gestionar dicha base de datos o hacerlo directamente sin herramientas externas. Definitivamente elegí hacerlo con Linq, una herramienta que proporciona el propio framework .NET.
* Por último, pensé en cambiar el diseño de alguna manera y encontré la librería para XAML Material Design.

Por otra parte, está la planificación y desarrollo de la propia aplicación que se muestra en los siguientes apartados:

* La aplicación la empecé realizando la gestión de todas las posibles partes que puede haber en un torneo, desde los jugadores de cada equipo a los campos donde se disputaran los encuentros y a sus posibles árbitros con su inserción, modificación, borrado y actualización.
* Después pasé a la realización del calendario. Esta consistió en crear un algoritmo que crea unas jornadas donde se iban a disputar los encuentros con una fecha y hora, lugar y un árbitro donde puedes introducirle el resultado que se usará posteriormente.
* Por último, realicé la tabla de clasificación donde se usan los resultados anteriormente nombrados para mostrar una clasificación del torneo escogido.

# ***RECOGIDA DE INFORMACIÓN***

En este apartado se indicarán los detalles de la recogida de información que se han llevado a cabo para la realización del proyecto con su correspondiente investigación.

## *WPF*

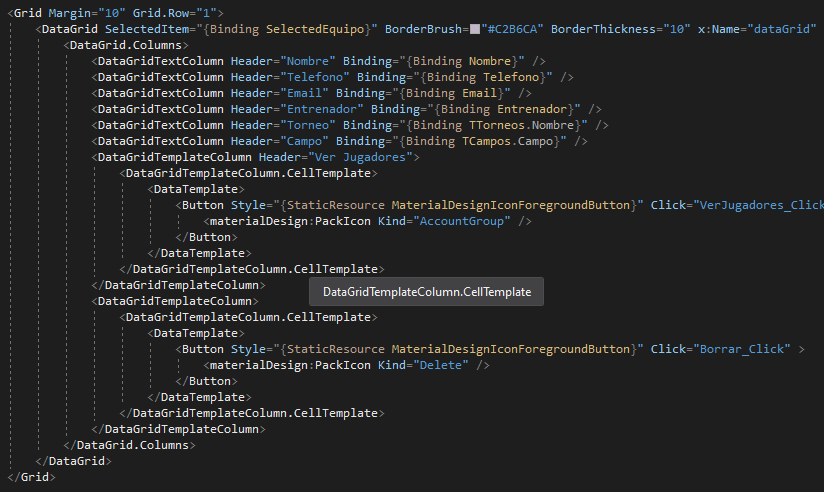
Windows Presentation Foundation te permite crear interfaces de usuario utilizando un lenguaje de marcado llamado XAML.

A diferencia de Windows Form en WPF existe un editor donde puedes escribir el código XAML directamente y esto ayuda a poder hacer un diseño mucho más optimo y trabajado.

Sus características principales son:

* Los gráficos en WPF están basados en vectores. Estos pueden ser escalados sin deformaciones.
* En WPF podemos crear elementos reutilizables para la interfaz gráfica.
* Permite el uso de Binding.
* Los estilos en WPF son muy parecidos a las hojas de estilos en cascada, CSS, para HTML.

Por ello mi aplicación tiene la vista totalmente escrita a mano sin usar el cuadro de herramientas, porque es mucho más manejable, potente y lo más importante, no se deforma.

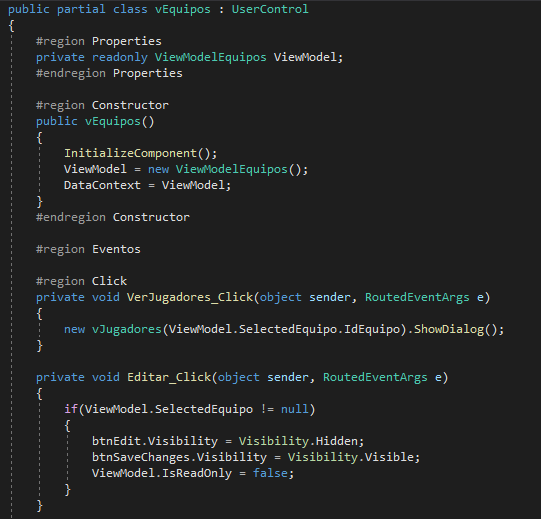
Ejemplo XAML de una parte de una vista de la aplicación:

## *C#*

Lenguaje nativo de .NET, C Sharp es una evolución de los lenguajes C y C++ que al cabo del tiempo ha ido añadiendo funcionalidades tomados de otros lenguajes como Java y Visual Basic.

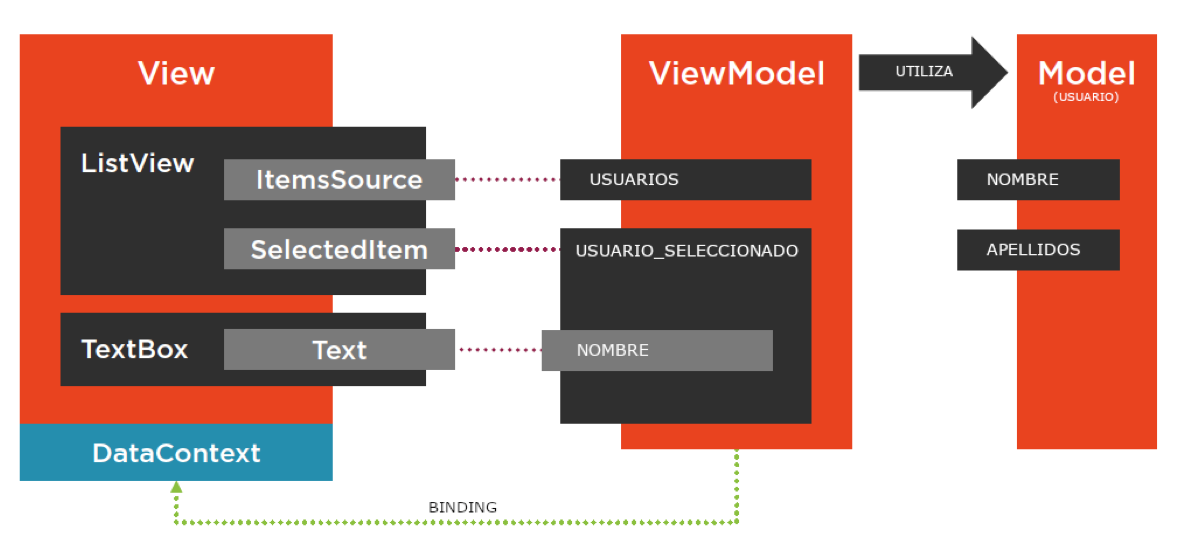
Unas de sus principales características son:

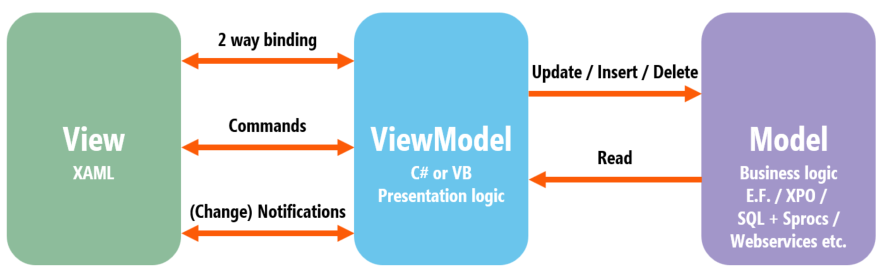
* Sencillez: Elimina gran cantidad de elementos que son innecesarios en .NET.
* Modernidad: Incorpora elementos que se han demostrado con el paso de los años que son útiles y esta a la orden del día. De ahí también la razón por la que lo escogí.
* Compatible: Mantiene una sintaxis muy parecida a otros lenguajes de programación para asi facilitar el trabajo al programador.
* Actualizado: Dispone de actualización y mejora continua.

Ejemplo de código en C# de una parte de la aplicación:

## *Patrón MVVM*

La recogida de información de este punto ha sido conocer cómo funciona la conexión entre el modelo y la vista mediante la vista modelo asi como aprender a llevarlo a cabo en mi aplicación.





* Vista: Se define en XAML, y no debe tener ninguna lógica en el código subyacente. Se une a la Vista Modelo solo mediante el enlace de datos (Binding).
* Modelo: Es el responsable de exponer los datos y su lógica.
* Vista Modelo: Es el encargado de conectar la vista con el modelo, conecta únicamente los datos que requieren las vistas y realiza la lógica necesaria para la preparación de dichos datos.

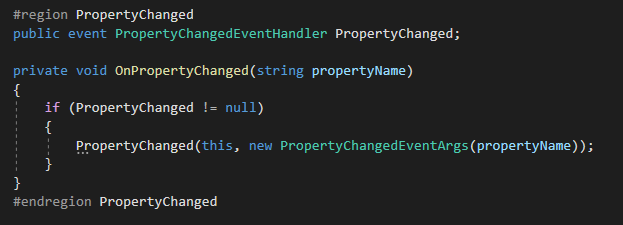
Por otra parte, se puede hacer la conexión de la vista modelo con la vista de diferentes maneras:

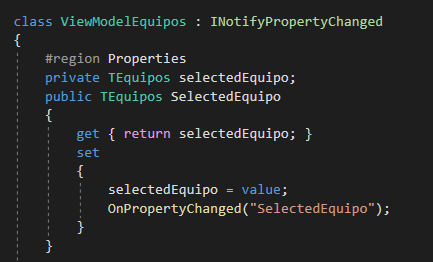
* Binding: Relaciona dos propiedades entre sí de forma que se mantengan sincronizadas.

De las propiedades más importantes que puede tener está el Modo que incluye diferentes valores:

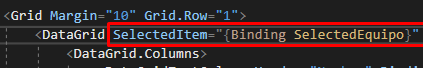
* + OneWay: Permite a las propiedades informar a la vista de los cambios, pero la propiedad no recibe información de los cambios que sufra en la vista.
  + TwoWay: En este caso el enlace es bidireccional, permite informar a la vista de los cambios y al revés.
  + OneTime: Lo mismo que OneWay, pero el enlace esta vez se realizará solo una vez.
  + OneWayToSource: Lo contrario a OneWay, solo permite a las propiedades informar de los cambios que se hayan hecho en la vista.
* Comandos: Permite controlar eventos desde la vista modelo.
* Notificaciones: Implementa la interfaz ***INotifyPropertyChanged***, que proporciona un método llamado ***RaisePropertyChanged*** y un evento llamado ***PropertyChanged***.

Dicho evento será lanzado cuando se actualice el valor de la propiedad deseada de la vista modelo y se lo notificará a la vista para que lo cambie al momento.

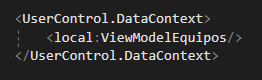
Ejemplo INotifyPropertyChanged:



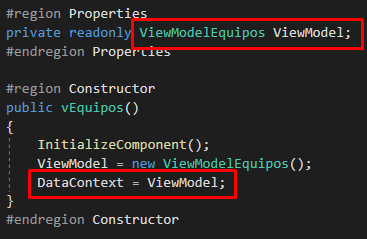
Ejemplo Binding XAML:



Ejemplo DataContext XAML: para enlazar la vista con la vista modelo



Ejemplo DataContext C#: para enlazar la vista con la vista modelo



## *LINQ*

LINQ (Language Integrated Query) es una sintaxis de consulta uniforme en C# y VB.NET utilizada para guardar y recuperar datos de diferentes fuentes.

La API LINQ en .Net implementan la interfaz IEnumerable y IQueryable:

* La clase Enumerable incluye métodos de extensión para las clases que implementan la interfaz IEnumerable.
* La clase Queryable incluye métodos de extensión para las clases que implementan la interfaz IQueryable.

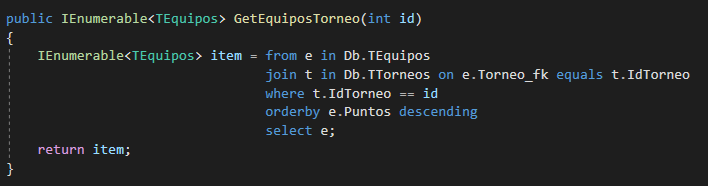
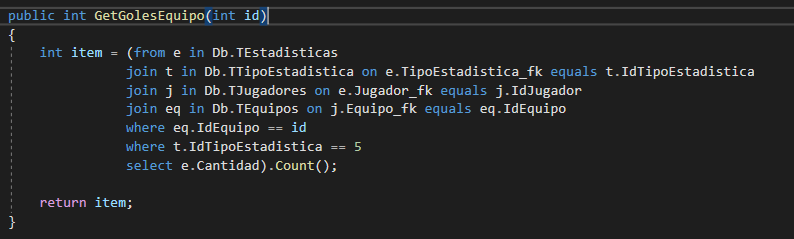
Con estas dos clases se pueden escribir consultas LINQ para recuperar datos de las colecciones integrada, la diferencia es que IQueryable ejecuta la consulta con filtros, mientras que IEnumerable primero ejecuta la consulta y luego filtra los datos según las condiciones.

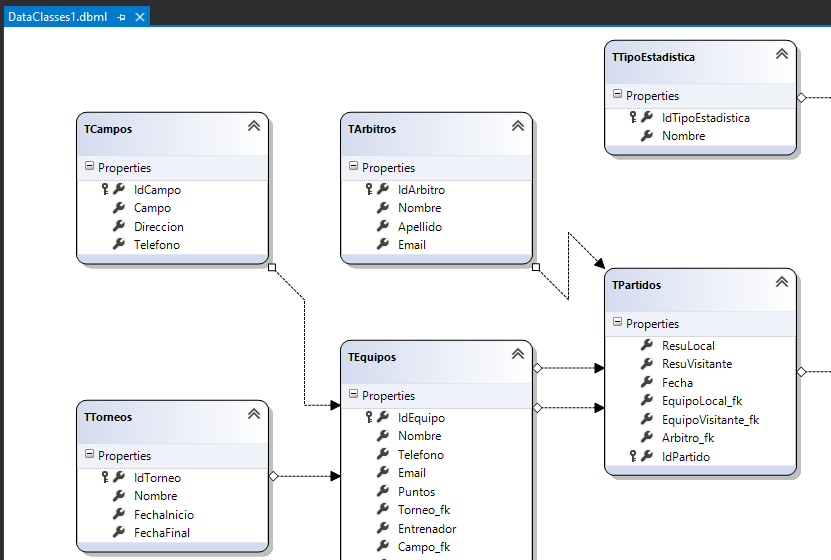
LINQ to SQL:

Es un componente de .NET que está diseñado para proporcionar acceso a datos almacenados en Sql Server mediante clases fuertemente tipadas mediante la implementación del ORM.

Ventajas de usar LINQ:

* Lenguaje familiar.
* Código reducido y legible.
* Forma estandarizada de consultar múltiples fuentes de datos en tu aplicación.
* Soporte Intellisense.

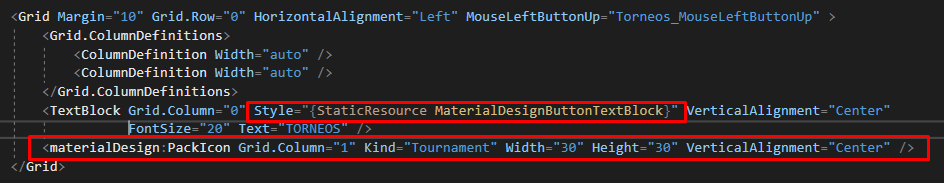
Ejemplo uso de LINQ en la aplicación:

Ejemplo de la clase Linq to Sql de la aplicación:

## *Material Design XAML*

Con este framework se le puede dar un formato mas moderno y diferente a los diseños y múltiples opciones que se ofrece en XAML de una sencilla forma.

Ejemplos uso en iconos en la aplicación:

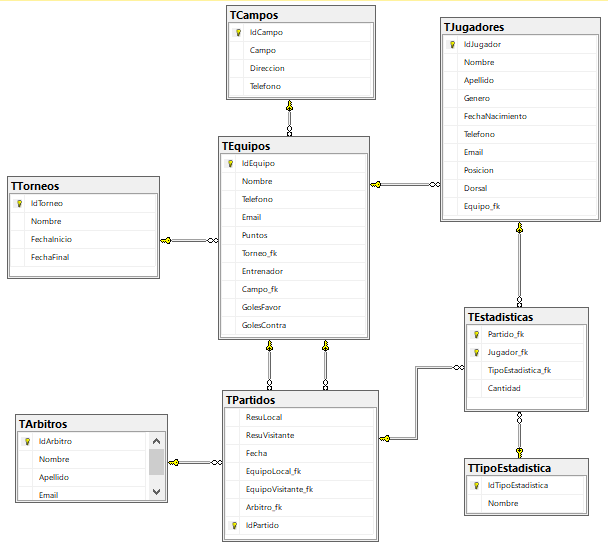


# ***BASE DE DATOS***

## *Modelo relacional*

El modelo relacional de una base de datos define la implementación lógica de los datos mediante tablas, campos y sus relaciones.

En este punto se mostrará cuál es el modelo relacional de la base de datos que se utiliza en la aplicación:

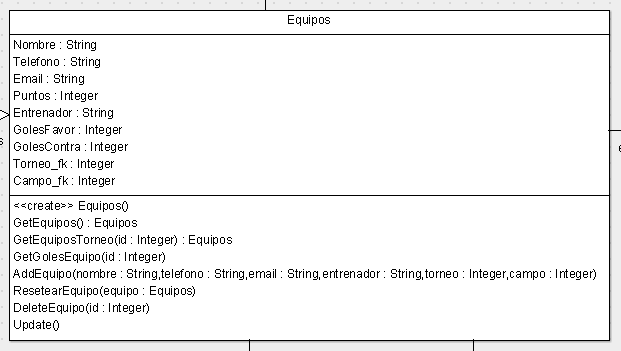


## *Diagrama de clases*

En este apartado se mostrará el diagrama de clases que se lleva a cabo a través de una división de capturas por clases.

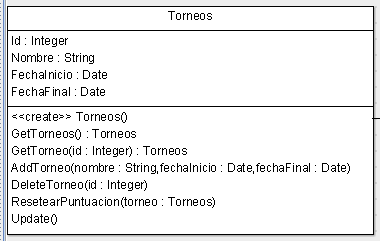
Se verá una representación gráfica de cada clase con su nombre, atributos y métodos.

Clase *Equipos*

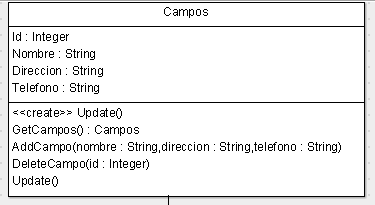


Clase *Torneos*

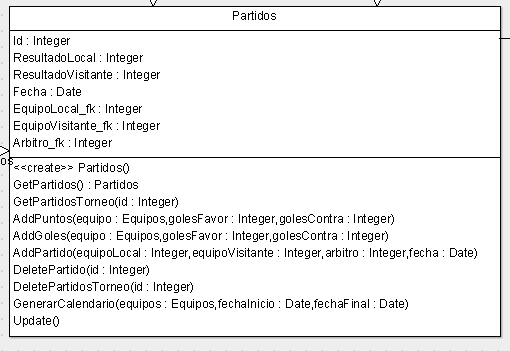
Equipos



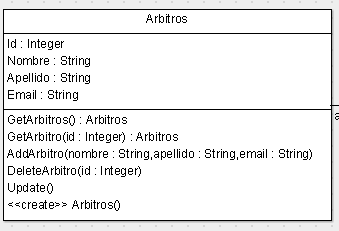
Clase *Campos*



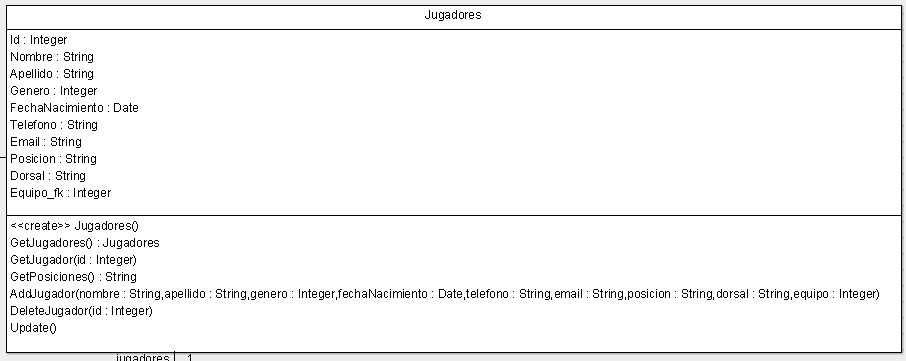
Clase *Partidos*



Clase *Árbitros*



*Clase Jugadores*



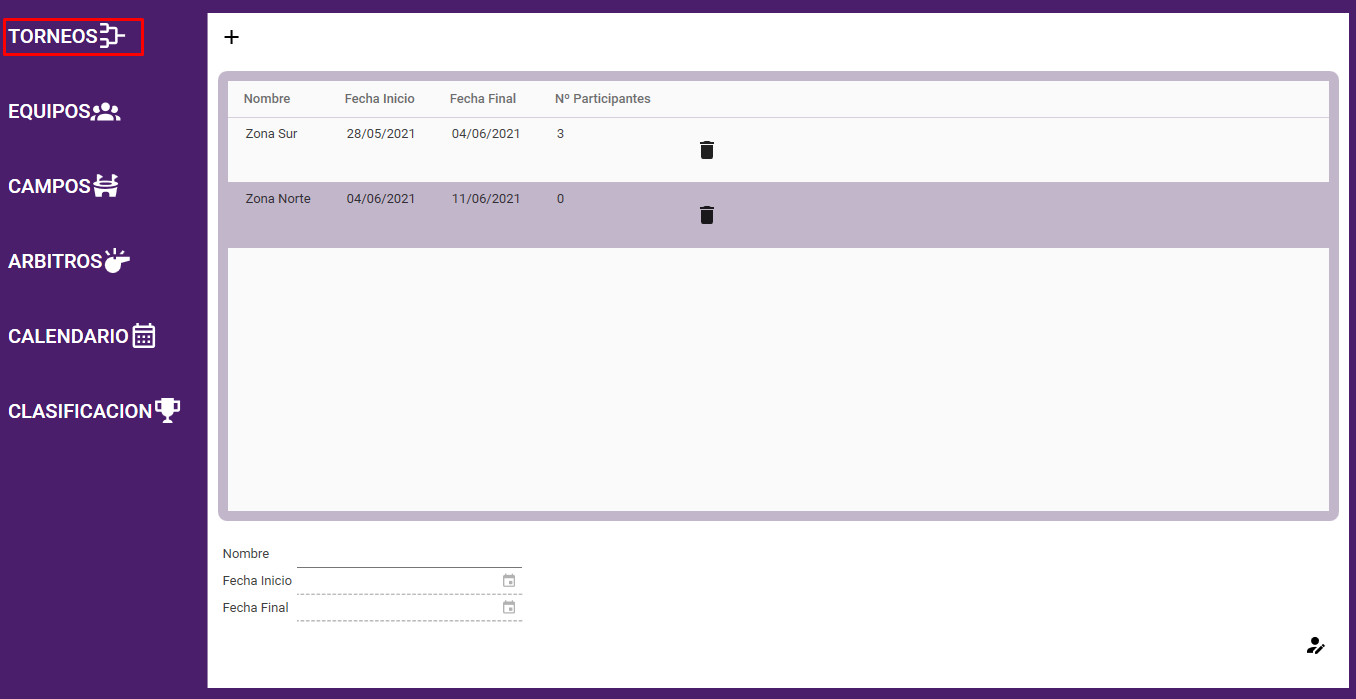
# ***MANUAL DE USUARIO***

Es este apartado se explicará el funcionamiento de cada ventana de la aplicación.

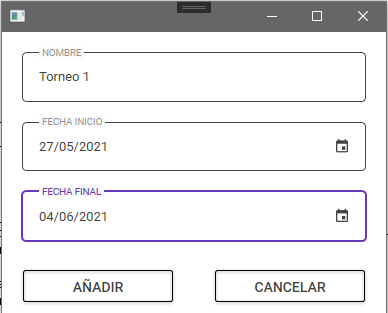
En las ventanas hay elementos en común:

* En la parte inferior de las ventanas será donde se pueden editar los campos de cada fila, pulsando un clic en la fila que deseemos nos aparecerán todos sus datos en los campos inferiores, pero solo para visualizarlos. Para poder editar dichos datos pulsar el botón con forma de pincel situado en la parte inferior derecha y se nos activara el modo editar, cuando hayamos hecho los cambios oportunos le volveremos a dar a este botón que ahora tendrá forma de disquete y se guardaran todos los cambios, si no le volvemos a pulsar y salimos de esa ventana desaparecerán esos cambios.
* En la parte superior izquierda habrá un botón con forma de el símbolo “+” que significará dar de alta, dependiendo la ventana en la que estemos nos aparecerá una ventana u otra donde se nos pedirán los campos oportunos para agregar campos a las tablas.
* Dentro de las filas habrá un botón con forma de papelera, dicho botón lo que hará es borrar el contenido de esa fila con lo que en cada caso conlleva que se explicara en cada ventana a continuación.
* En todas las ventanas de alta se controla que el usuario tenga que introducir todos los campos requeridos y de forma correcta.

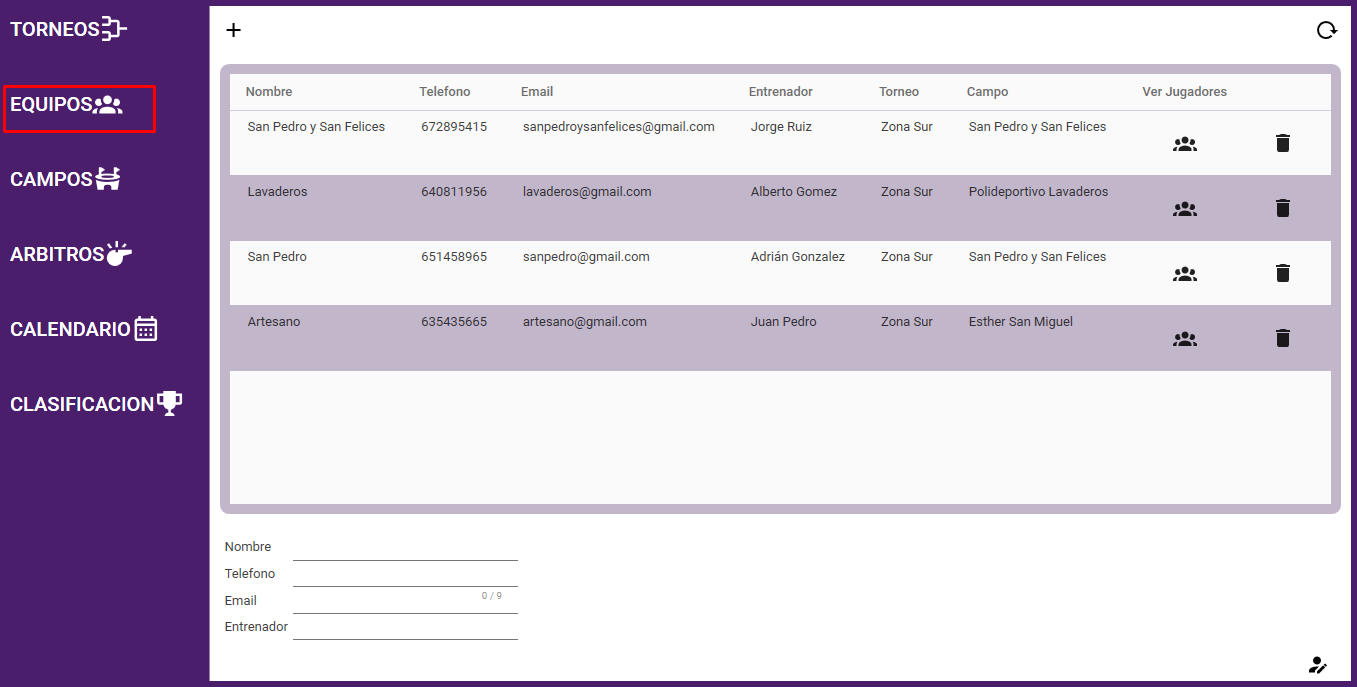
En la ventana Torneos se mostrarán los torneos con su nombre, fecha de inicio del torneo, la fecha final del torneo y su número de participantes, en este caso el botón de borrar eliminara por completo los equipos del torneo en caso de tenerlos y todos sus jugadores.



Alta torneos donde se pedirá un nombre para el torneo con una fecha de fin y otra de inicio.



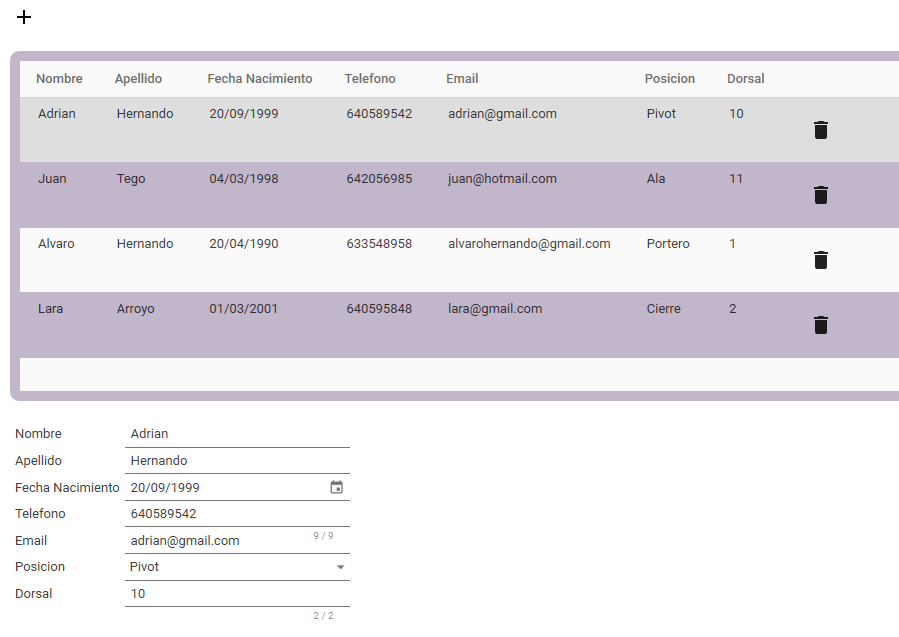
Ventana Equipos se mostrarán los equipos con su nombre, datos de contacto, entrenador, torneo al que pertenece, campo al que pertenece y un botón en la columna ver jugadores que lleva a la ventana Jugadores donde se mostrarán los jugadores del equipo de la fila seleccionado, en este caso el botón de borrar eliminara al equipo seleccionado con todos sus jugadores en el caso de tenerlos.



Alta de equipos se pedirán los datos anteriormente mencionados de la ventana equipos, pero en torneo y campos será un seleccionable ya que tienen que pertenecer a un torneo y campo ya existentes por lo que habrá que crearlos con anterioridad al equipo.



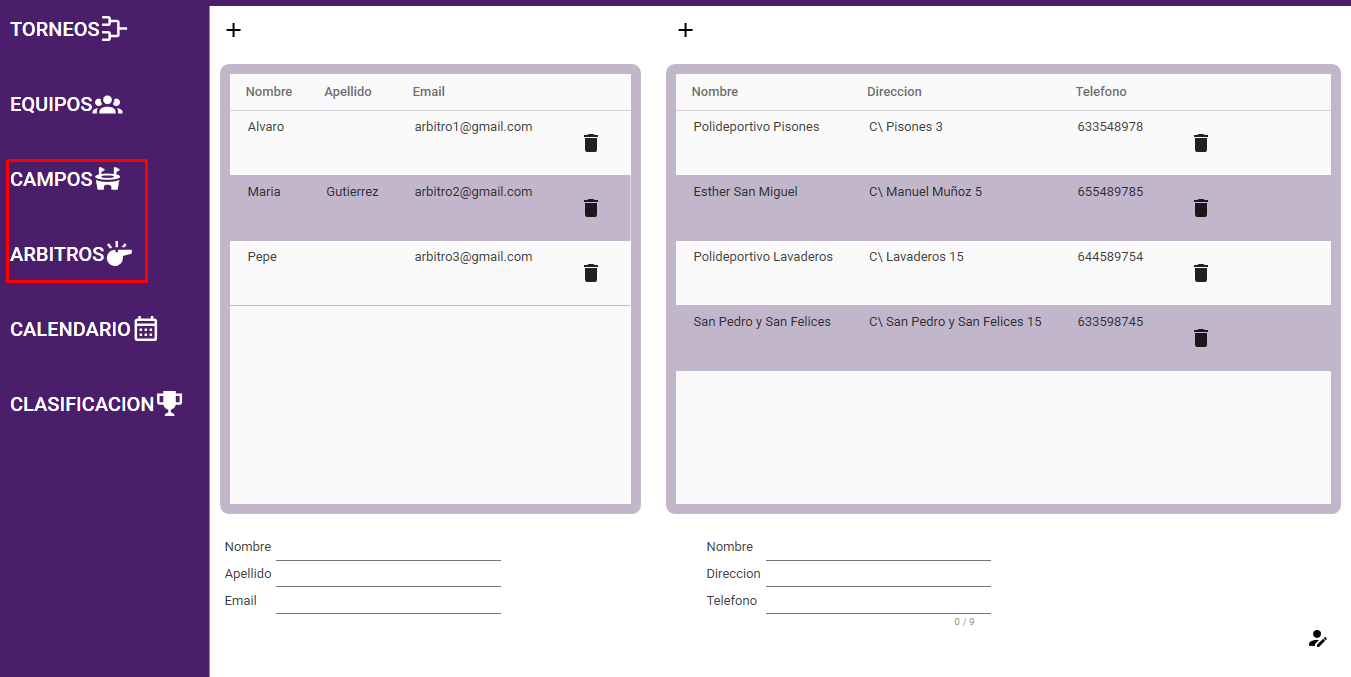
En la ventana Jugadores que aparece al pulsar el botón ver jugadores que hay en cada fila de la ventana Equipos aparecerán el nombre completo, fecha, datos de contacto, posición y dorsal del jugador, en este caso la posición al editarla será un seleccionable de las posiciones de futbol sala.



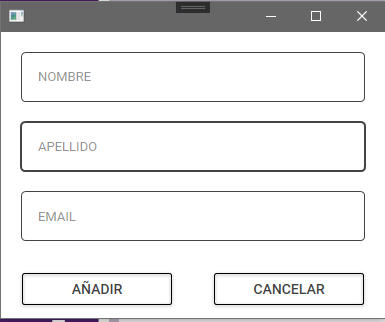
Alta de jugadores contendrá opciones para todos los campos anteriormente mencionados que hacen a un jugador asi también siendo un desplegable la posición.



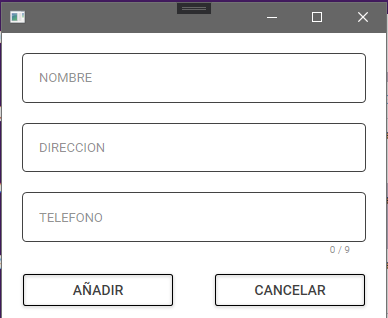
Ventana Árbitros y Campos forma los árbitros, con su nombre completo y email, y los campos con su nombre, dirección y teléfono.



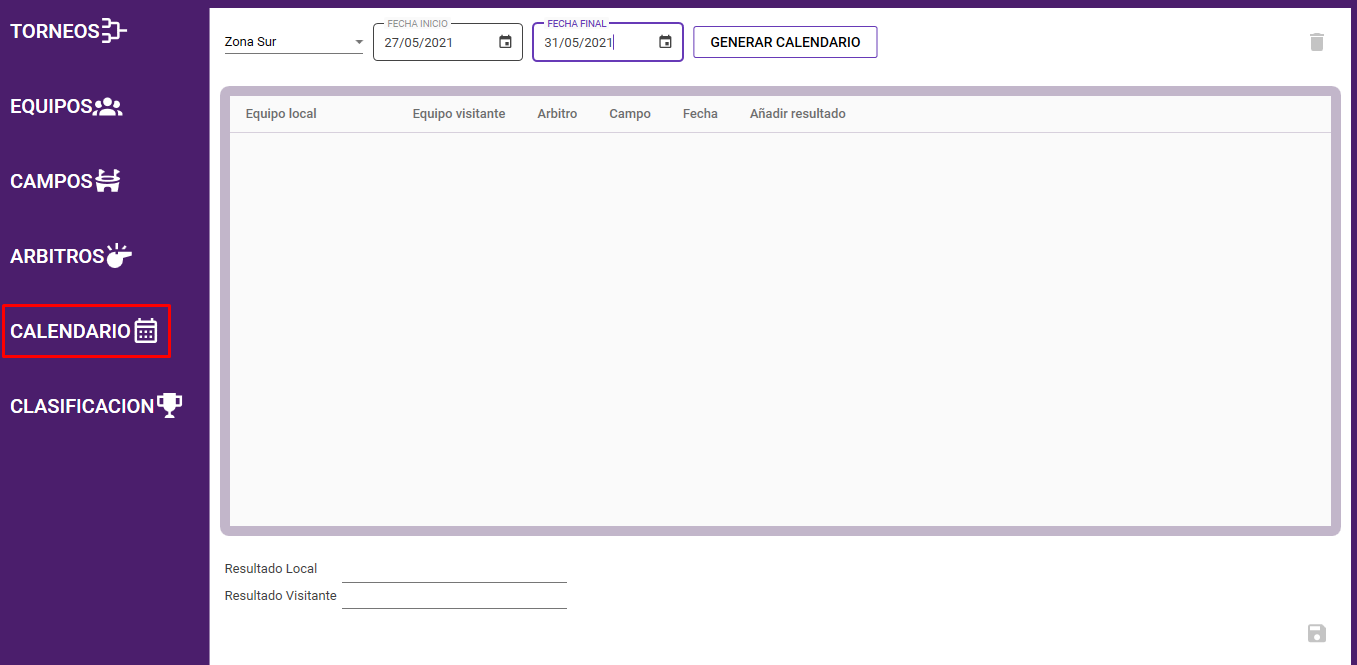
Alta de árbitros que tendrá la posibilidad de crearlo con el nombre completo y un email para contactar en caso de partido.



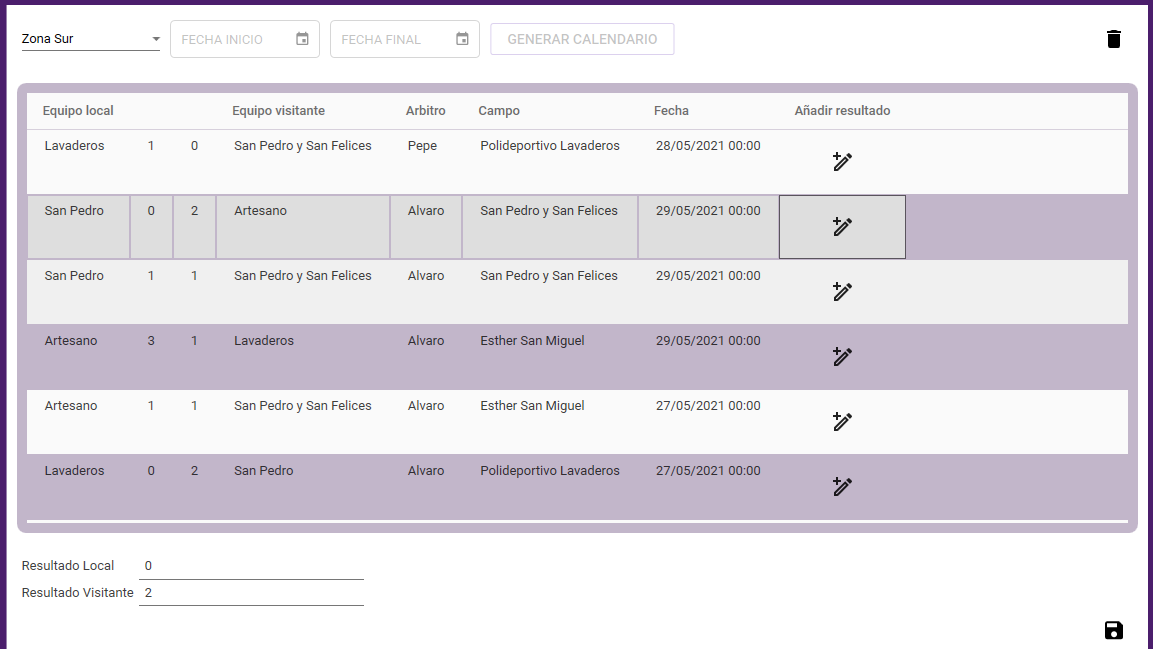
Alta de campos con un nombre, dirección para su localización y un teléfono para el contacto.



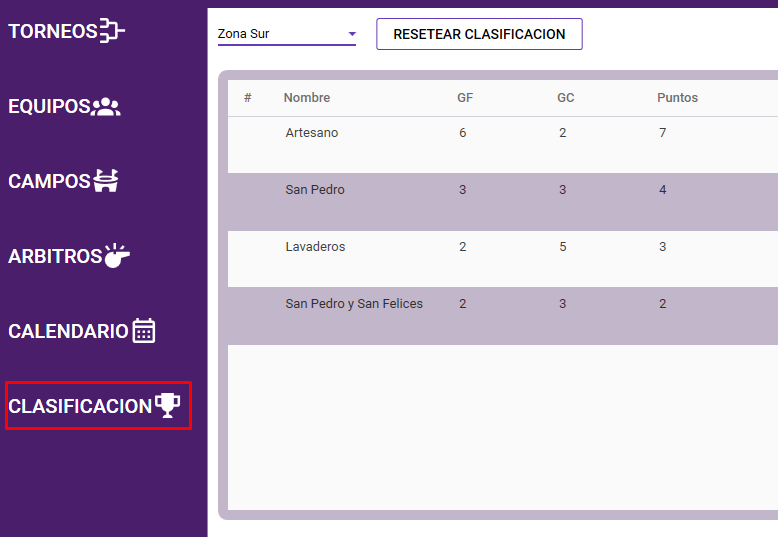
Ventana Calendario donde se crearán los calendarios de un torneo seleccionando un torneo de los existentes en el desplegable y fijándole una fecha concreta de inicio y final para que se jueguen sus partidos entre dichas fechas. Después pulsamos el botón “Generar Calendario” y nos creara varios partidos en las fechas seleccionadas con los equipos jugando todos contra todos del torneo seleccionado.



Por otra parte, en esta ventana al tener ya el calendario creado tenemos la opción de introducir los resultados que después serán utilizados para obtener la clasificación, si pulsamos el botón añadir resultado podremos introducir el resultado en cada fila de cada partido, en este caso el botón de borrar situado en la esquina superior derecha eliminara el calendario de los partidos del torneo seleccionado.



Por último, la ventana Clasificación seleccionando un torneo te muestra su clasificación con el nombre, goles a favor, goles en contra y puntos que lleva sumados que dependerán de los resultados que se hayan introducido en la ventana Calendarios, también existe la opción con el botón “Resetear clasificación” de resetear todos los puntos a cero para volver a empezar.



# ***REFERENCIAS BIBLIOGRÁFICAS***

*C#*:

<https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/>

*Linq*:

<https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries>

<https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/sql/linq/>

<https://www.tutorialsteacher.com/linq/linq-api>

*Patrón MVVM*:

<https://docs.microsoft.com/es-es/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern>

<https://www.c-sharpcorner.com/UploadFile/raj1979/simple-mvvm-pattern-in-wpf/>

*Material Design XAML*:

<http://materialdesigninxaml.net/>