Institut Puig Castellar
Santa Coloma de Gramenet

Padel Tournament

CFGS Desenvolupament d'Aplicacions Multiplataforma

Raúl Olmedo Murillo

Curso académico

Resumen del proyecto

Al inicio del proyecto pretendía realizar una aplicación para Android la cuál me permitiese registrar diferentes tipos de jugadores y torneos de padel. Durante su desarrollo me encontré con varios problemas tales como, manejarme con Android, mi inexperiencia en el diseño de aplicaciones y de interfaces o la falta de tiempo.

A pesar de ello he conseguido que la aplicación guarde un registro de jugadores, torneos y partidos en una base de datos postgresql situada en un servidor dedicado que tengo en alquiler. Además realiza el emparejamiento tanto de equipos como de jugadores y te permite iniciar las diferentes rondas.

Aunque no he podido alcanzar los objetivos que pretendía al inicio del proyecto estoy contento con mi trabajo ya que me ha permitido tener un primer contacto con el sistema operativo Android, con sus Activitys y sus Fragment que casi consiguen hacerme desistir. Por otro lado me ha servido para realmente conocer el tiempo y el desgaste que puede conllevar diseñar, programar, probar y documentar una aplicación tu solo.

Abstract

It has been three very intense months.

Schedule changes, the pressure of creating your first application in short time and without having clearly the result of work.

Every day that passes I more understand this world, a labor sector that rewards effort and desire to improve, but also a labor sector that must learn to control, I must learn to disconnect from work as much as I liked me. Everything has a price.

On the other hand I have been abte to implements all kwonledge acquired from:

Linux: Connect by bash-terminal with Android and Servers, managament differents services.

Postgersql: The database was make with Postgresql. The managament of SGDB.

Android: The application is make for Android.

Java – SQL – XML: The lenguages of my app.

I could see how my brain adapts increasingly reading code or as I becomes easier compresion, although I not yet have the ease that I would like.

4 Palabras sobre el proyecto

Desafío – Android – Java - Database

Indice

1. Introducción

1.1 Contexto y justificación del trabajo

Un deporte que lleva ya unos años en auge es el Padel, es un deporte que por sus características y fácil aprendizaje permite que lo puedan practicar la mayoría de personas, sin importar su condición física o su edad. Muchos club de Padel han inaugurado recientemente sus instalaciones, en estos clubes se generan grupos de whatsapp para organizar partidos o torneos. Estos torneos siempre los tiene que organizar una persona, bien ya sea empleado del club o simplemente un usuario de este, pues bien, estos torneos suelen tener una modalidad llamada “Rondas Americanas”, la cuál hace que cada cierto tiempo cambies de contrincante y de pareja. Organizar este tipo de torneos cuando la participación es amplia es algo complejo y tedioso.

Objetivos

Nuestra App automatizará el proceso de la creación de diferentes torneos o ligas. Además almacenará un registro de todos los partidos y las estadísticas de sus Usuarios para mostrarlos en la aplicación.

Competencias a fomentar:

- Capacidad de plantear un problema, definir los objetivos de trabajo y analizar los resultados.
- Capacidad de resolver problemas.
- Capacidad de trabajar con Android y Java.
- Capacidad de crear una base de datos solvente
- Capacidad de presentar un trabajo públicamente y responder adecuadamente las preguntas del tribunal de evaluación.

Objetivos proyecto:

- Crear una base de datos solvente, que permita la creación de diferentes rankings según grupos o instalaciones y libere lo máximo posible de carga de trabajo a la aplicación.
- Aprender a trabajar con el sistema operativo Android y desarrollar una aplicación para este sistema operativo.
- Publicar mi primera aplicación en Android.

1.2 Objetivos de Trabajo

Conexión con base de datos

En un servidor se instalará Postgresql**,** en el cuál se creará una base de datos con:

- los datos de los usuarios. Nick, Email, total sets ganados, total sets perdidos, torneos ganados y torneos participado.
- Los partidos. Que tendrán unos participantes con diferentes puntuaciones, además de id del partido y su fecha.
- Los torneos. Que tendrán un número de partidos y las puntuaciones que se vayan generando.
- Las ligas.

De esta forma luego podremos acceder a estos datos ya sea para utilizarlos en forma gráfica por cada usuario, ver los torneos en los que se han participado, creación de un ranking, valoraciones de las pistas, etc. la opciones son infinitas.

Para no tener problemas con la LOPD, no vamos a solicitar muchos datos a los usuarios. Los principales serán nick y email, teniendo unos atributos secundarios como Torneos jugados o Set ganados.

De momento la conexión con la base de datos se hará por el puerto 5432 aunque estamos estudiando la posibilidad de utilizar el puerto 80 con un certificado SSL para tener una conexión más segura a través de Java.

Desarrollar una App para Android que nos permita:

- Añadir jugadores.
  - Añadir contactos o no contactos que serán los participantes de nuestros torneos o ligas.
  - Los jugadores aumentarán de nivel. El número de sets jugados y ganados en los partidos, si importar el resultado final será el nivelador para subir la experiencia del jugador y por lo tanto subir poco a poco su Nivel.

- La creación de torneos con sus diferentes modalidades:
  - Mixto, Masculino o Femenino

  - Individual o Dobles
  - americano,americano plus o por eliminación

- Instalaciones. Donde están y el número de pistas que queremos utilizar,enlaces a la web y a sus teléfonos de contacto. (Dato importante para la creación de torneos).

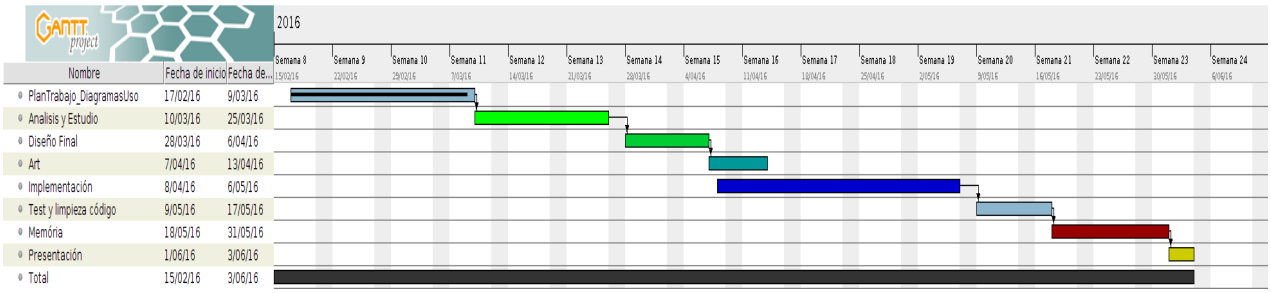
- Creación de ligas con nuestros contactos, las ligas tendrán automatizadas las siguientes tareas:
  - Las puntuaciones del último partido.
  - Los avisos de partido una semana y un día antes.
  - La clasificación .
- Un ranking.

1.3 Enfoque y método a seguir

1. Toma de Requisitos, entre mi hermano mayor (un fanático del Padel que organizar muchos torneos pero que no tiene ni idea de programación) y yo estamos decidiendo que requisitos son fundamentales para tener una aplicación de éxito.
2. Desarrollo del plan de trabajo.
3. Documentación de nuevas tecnologías.
4. Desarrollo de diagramas de uso.
5. Descripción de Requisitos Finales
6. Desarrollo de diagramas de clase y Entidades
7. Implementación
8. Depuración
9. Memoria técnica y presentación

1.4 Planificación del proyecto

Nombre Tarea	Fecha inicio	Fecha final
PLAN DE TRABAJO Y DIAGRAMAS DE USO **1.** Presentación Inicial **2.** Toma de Requisitos **3.** Plan de Trabajo **4.** Diagramas de uso	17/02/16	09/03/16
ANALISIS Y ESTUDIO **1.** Análisis y diseño de los diagramas UML **2.** Estudio de las APIS que voy a necesitar	10/03/16	25/03/16
DISEÑO FINAL	28/03/16	06/04/16
ART **1.** Pruebas con CSS y otros lenguajes para la visualización de la App **2.** Toma de decisión final con el Arte	07/04/16	13/04/16
IMPLEMENTACIÓN **1.** Creación de la base de datos **2.** Creación de la lógica del programa **3.** Creación de las interfaces del programa	08/04/16	06/05/16
TEST Y LIMPIEZA DEL CÓDIGO **1.** Depuración	09/05/16	17/05/16
MEMÓRIA **1.** Finalización de la memoria técnica	18/05/16	03/06/16
TOTAL	18/02/16	03/06/16



2. Resto de capítulos

Requisitos de la aplicación

La aplicación necesita:

1.- Poder crear jugadores, que posteriormente se añadirán a los torneos.

2.- Poder crear torneos de estos tipos:

a)Torneo normal. Por Eliminatorias.

b)Torneo Americano. Rondas (entre 20 y 30 min), número de rondas según participación y posibilidad de alquiler de pistas.

c)Torneo Americano Plus. Igual que el anterior pero también vamos cambiando de pareja.

d)Ligas.(**Objetivo alternativo**)

Tabla1 Tipos de Rondas Americanas Plus. Todos contra Todos

Nº Jugadores	Nº Pistas	Descanso	Duración partido	Nº Partidos	Duración Torneo
4	1	No	1:30	3	1:30
8	2	No	20	7	2:40
10	2	Si	20	9	3:00

Tabla2 Tipos de Rondas Americanas.

Todos los equipos contra todos los equipos. Lo importante de esta ronde es organizar una tarde divertida y amena, para ello habría que tener en cuenta el nivel de los jugadores.

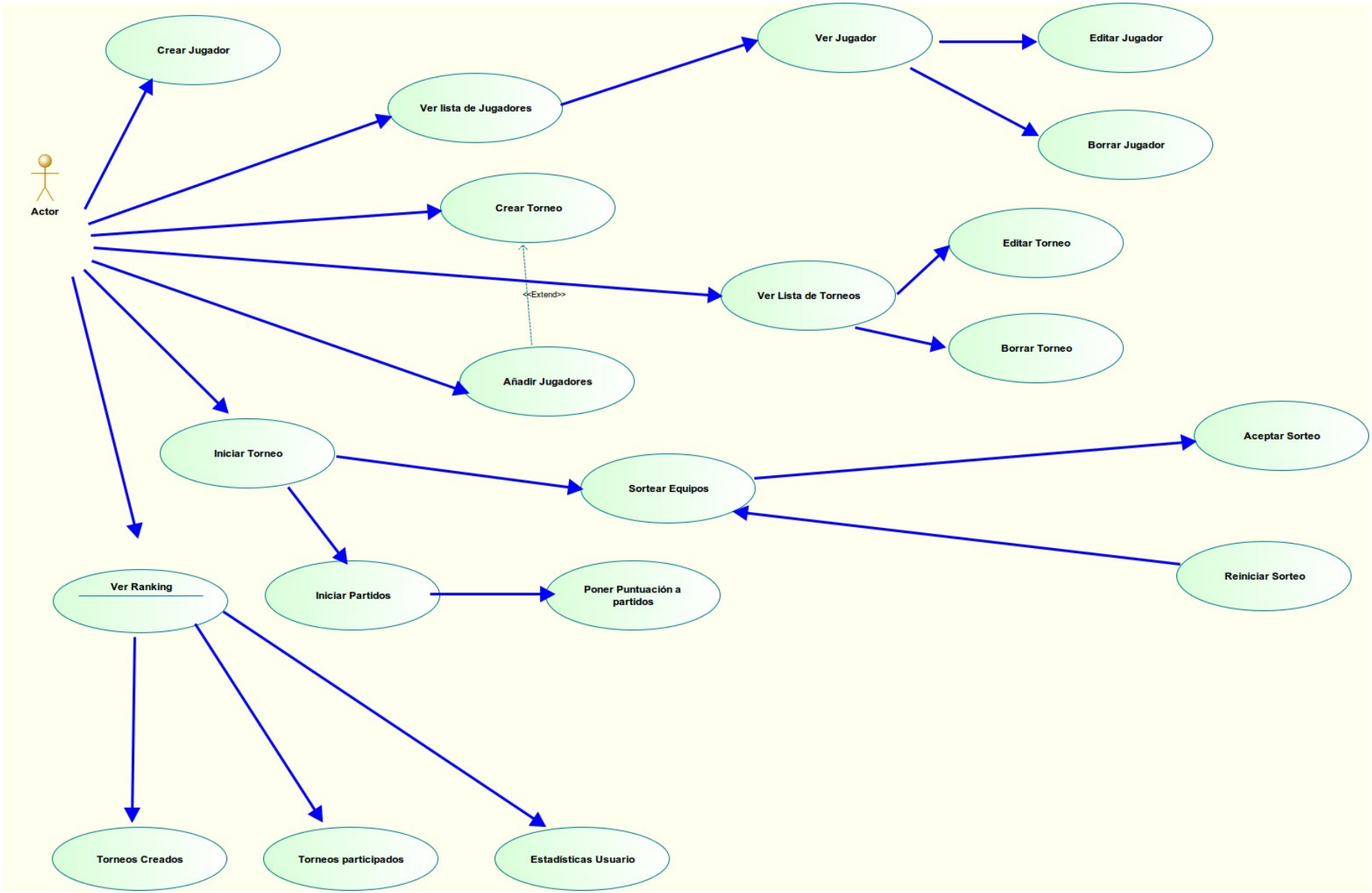
Nº Jugadores	Nº Pistas	Descanso	Duración partido	Nº Partidos	Duración Torneo
8	2	No	30	3	1:30
10	2	SI	20	4+1	2:00
12	3	No	20	5	2:00
14	3	SI	20	6+1	2:30
16	4	No	20	7	2:30
18	4	Si	20	7+1	2:40
20	5	No	20	9	3:00

Tabla4 Tipos de Ligas (Objetivo Alternativo)

3.- Un ranking. Con estadísticas de los jugadores y Torneos organizados.

4.- Geolocalización de las pistas(**Objetivo Alternativo**)

Diagramas / Diagramas de uso



Caso de uso nº 1

Usuario: *Crear Jugadores*

- *Resumen de la funcionalidad: El usuario creará jugadores que serán almacenados en una base de datos, para posteriormente ser añadidos en los Torneos que se crearán.*
- *Papel dentro del trabajo de usuario: Caso de uso principal.*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: El jugador no ha de existir.*
- *PostCondición: El jugador será almacenado en una base de datos.*
- *Descripción del caso de uso:*
  - *Común: El usuario introducirá los datos del Jugador(nombre,telf,correo,nivel).*
  - *Excepciones: El jugador a crear no puede existir, si alguno de los campos se repite el jugador no podrá ser creado.*

Caso de uso nº 2

Usuario: *Ver Jugadores*

- *Resumen de la funcionalidad: El usuario podrá ver una lista de los jugadores almacenados en la base de datos*
- *Papel dentro del trabajo de usuario: Caso de uso principal.*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Debe haber un usuario creado en la base de datos..*
- *PostCondición: Se mostrará una lista de los jugadores.*
- *Descripción del caso de uso:*
  - *Común: El usuario verá una lista de Jugadores.*
  - *Excepciones: Que no haya ningún jugador creado.*

Caso de uso nº 2.1

Usuario: *Ver Jugador*

- *Resumen de la funcionalidad: El usuario podrá ver los diferentes atributos de un jugador en concreto.*
- *Papel dentro del trabajo de usuario: Caso de uso secundario.*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Debe haber un usuario creado en la base de datos..*
- *PostCondición: Se mostrarán los datos de un jugador.*
- *Descripción del caso de uso:*
  - *Común: El usuario verá un jugador.*
  - *Excepciones: Que no haya ningún jugador creado.*

Caso de uso nº 2.1.A

Usuario: *Editar Jugador*

- *Resumen de la funcionalidad: El usuario podrá editar los diferentes atributos de un jugador en concreto.*
- *Papel dentro del trabajo de usuario: Caso de uso secundario.*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Debe haber un usuario creado en la base de datos..*
- *PostCondición: Se editarán y guardarán los datos de un jugador.*
- *Descripción del caso de uso:*
  - *Común: El usuario editará un jugador.*
  - *Excepciones: Que no haya ningún jugador creado.*

Caso de uso nº 2.1.B

Usuario: *Borrar Jugador*

- *Resumen de la funcionalidad: El usuario podrá borrar a un jugador en concreto.*
- *Papel dentro del trabajo de usuario: Caso de uso secundario.*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Debe haber un usuario creado en la base de datos..*
- *PostCondición: Se editarán y guardarán los datos de un jugador.*
- *Descripción del caso de uso:*
  - *Común: El usuario editará un jugador.*
  - *Excepciones: Que no haya un jugador seleccionado.*

Caso de uso nº 3

Usuario: *Crear Torneo*

- *Resumen de la funcionalidad: El usuario entrará en un menú para crear un Torneo nuevo.*
- *Papel dentro del trabajo de usuario: Caso de principal*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: El torneo no debe existir en la base de datos.*
- *PostCondición: Se almacenará el torneo en la base de datos.*
- *Descripción del caso de uso:*
  - *Común: El usuario creará un torneo.*
  - *Excepciones: El torneo no debe existir en previamente.*

.

Caso de uso nº 4

Usuario: *Ver listas de torneos*

- *Resumen de la funcionalidad: El usuario podrá ver una lista de torneos creados.*
- *Papel dentro del trabajo de usuario: Caso de secundario*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Los torneos deben haber sido creados previamente.*
- *PostCondición: Podremos elegir un torneo de la lista para su modificación o borrado de la base de datos*
- *Descripción del caso de uso:*
  - *Común: El usuario verá los torneos que ha creado.*
  - *Excepciones: Los jugadores no existan*

Caso de uso nº4.1

Usuario: *Editar Torneo*

- *Resumen de la funcionalidad: El usuario podrá editar un torneo de las lista mostrada en el caso anterior.*
- *Papel dentro del trabajo de usuario: Caso de secundario*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Los torneos deben haber sido creados previamente.*
- *PostCondición: Las modificaciones se almacenarán en la base de datos.*
- *Descripción del caso de uso:*
  - *Común: El usuario podrá cambiar los atributos principales del torneo como pueden ser Nombre, fecha de inicio, hora de inicio, hora de fin, etc.*
  - *Excepciones: No se podrá cambiar el tipo de torneo, si se ha creado un torneo para 16 personas no se podrá reemplazar por torneo para 20 personas.*

Caso de uso nº4.2

Usuario: *Borrar Torneo*

- *Resumen de la funcionalidad: El usuario borrará el torneo elegido*
- *Papel dentro del trabajo de usuario: Caso de secundario*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Los torneos deben haber sido creados previamente.*
- *PostCondición: El torneo desaparecerá de la base de datos.*
- *Descripción del caso de uso:*
  - *Común: Un torneo que nunca se utilizo, pruebas para crear torneos, etc. podrán ser eliminados mediante este uso.*
  - *Excepciones: Los torneos existirán previamente.*

Caso de uso nº5

Usuario: *Añadir jugadores a Torneo*

- *Resumen de la funcionalidad: El usuario añadirá los jugadores al torneo elegido.*
- *Papel dentro del trabajo de usuario: Caso de principal*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Los jugadores deben haber sido creados previamente.*
- *PostCondición: El sistema estará preparado para crear los emparejamientos.*

- *Descripción del caso de uso:*
    - *Común: El usuario elige los jugadores que participarán en el Torneo.*
    - *Excepciones: Los jugadores no existan*

Caso de uso nº 6

Usuario: *Iniciar Torneo.*

- *Resumen de la funcionalidad: El usuario elegirá uno de los torneos creados y le dará comienzo.*
- *Papel dentro del trabajo de usuario: Caso de principal*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: El torneo debe haber sido creado previamente y debe tener incluidos los jugadores.*
- *PostCondición: El sistema mostrará una pantalla con el número de rondas que se genera automáticamente dependiendo del número de participantes y equipos, que ya seleccionamos previamente al crear el torneo.*
- *Descripción del caso de uso:*
    - *Común: Iniciar Torneo, primero creando los emparejamientos y después guardando los resultados.*
    - *Excepciones: Debe haber en la base de datos un torneo con jugadores incluidos para poder empezar.*

Caso de uso nº 6.1

Usuario: *Crear emparejamientos*

- *Resumen de la funcionalidad: El programa creará los diferentes aparejamientos en las diferentes rondas y los mostrará*
- *Papel dentro del trabajo de usuario: Caso de principal*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: El tipo de torneo deberá haber sido elegido así como la introducción de los jugadores.*
- *PostCondición: El sistema mostrará los emparejamientos y las diferentes rondas.*
- *Descripción del caso de uso:*
    - *Común: El usuario permite al sistema iniciar los emparejamientos.*
    - *Excepciones: Que todos los jugadores que necesita el torneo estén introducidos en la base.*

Caso de uso nº 6.1

Usuario: *Aceptar emparejamientos.*

- *Resumen de la funcionalidad: El usuario acepta los emparejamientos y las rondas creadas y guarda el Torneo en la base de datos.*
- *Papel dentro del trabajo de usuario: Caso de principal*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: El usuario debe estar de acuerdo con los emparejamientos y las rondas.*
- *PostCondición: El sistema almacenará el Torneo en la base de datos*
- *Descripción del caso de uso:*
    - *Común: Almacenar un torneo.*

- *Excepciones:*

Caso de uso nº 6.3

Usuario: *Reiniciar emparejamientos.*

- *Resumen de la funcionalidad: El usuario no acepta los emparejamientos y las rondas creadas y reinicia la creación de estos.*
- *Papel dentro del trabajo de usuario: Caso de principal*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: El usuario no esta de acuerdo con los emparejamientos.*
- *PostCondición: El sistema creará de nuevo los emparejamientos y los enseñará por pantalla.*
- *Descripción del caso de uso:*
  - *Comuna: Reiniciar emparejamientos.*
  - *Excepciones:*

Caso de uso nº 6.4

Usuario: *Iniciar Rondas*

- *Resumen de la funcionalidad: El usuario podrá iniciar las diferentes rondas de un torneo.*
- *Papel dentro del trabajo de usuario: Caso de principal*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: El torneo debe haber sido iniciado previamente.*
- *PostCondición: El sistema mostrará una pantalla con La Ronda nº1 y un botón para iniciar el tiempo de ejecución de esta. Al finalizar el tiempo de la ronda, podremos poner la puntuación. El número de rondas vendrá predefinido en cada torneo, dependiendo de los partidos que se disputen.*
- *Descripción del caso de uso:*
  - *Común: Iniciar primera Ronda*
  - *Excepciones: No podrá iniciarse la ronda siguiente si la otra no ha finalizado*

Caso de uso nº 6.5

Usuario: *Poner Puntuación a Rondas*

- *Resumen de la funcionalidad: El usuario podrá puntuar el resultado de una ronda y darla por finalizada.*
- *Papel dentro del trabajo de usuario: Caso de principal*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: La ronda debe haber sido iniciada previamente.*
- *PostCondición: Una vez introducida la puntuación se dará por finalizada la ronda y se pasará a la siguiente o al final del torneo si fuese la última.*
- *Descripción del caso de uso:*
  - *Común: Poder puntuar los diferentes partidos en cada ronda.*
  - *Excepciones: Se deberá dar puntuación a todos los partidos de las ronda.*

Caso de uso nº 7

Usuario: *Ver ranking*

- *Resumen de la funcionalidad: Acceder a una actividad que mostrará 3 menús diferentes*
- *Papel dentro del trabajo de usuario: Caso secundario*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Debe haber almacenado en la base de datos un torneo finalizado.*
- *PostCondición: Se mostrarán 3 accesos a diferentes estadísticas*
    - *Torneos Creado*
    - *Torneos Participados*
    - *Estadísticas Jugador*
- *Descripción del caso de uso:*
    - *Común: Conocer los diferentes datos que crea la App.*
    - *Excepciones: Se deberá tener torneos creados y torneos participados.*

Caso de uso nº 7.1
Usuario: *Ver Torneos Creados*

- *Resumen de la funcionalidad: Acceder a las estadísticas de los Torneos que el usuario a ha creado.*
- *Papel dentro del trabajo de usuario: Caso secundario*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Debe haber almacenado en la base de datos un torneo finalizado.*
- *PostCondición: Se mostrarán datos como:*
    - *Número de jugadores participantes*
    - *Número de pistas alquiladas*
    - *Ganador*
    - *Segundo*
    - *Creador*
    - *Puntuación Ganador*
    - *Puntuación Creador*
- *Descripción del caso de uso:*
    - *Común: Conocer los diferentes estadísticas que genera la App.*
    - *Excepciones: Se deberá tener torneos creados y torneos participados.*

Caso de uso nº 7.2
Usuario: *Ver Torneos Participados*

- *Resumen de la funcionalidad: Acceder a las estadísticas de los Torneos que el usuario a ha participado*
- *Papel dentro del trabajo de usuario: Caso secundario*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Debe haber participado al menos en un torneo almacenado en la base de datos un torneo finalizado.*
- *PostCondición: Se mostrarán datos como:*
    - *Número de jugadores participantes*
    - *Número de juegos Ganados por el usuarios*

        - *Número de juegos Ganador por el Ganador*
        - *Ranking del torneo*
- *Descripción del caso de uso:*
    - *Común: Conocer los diferentes estadísticas que genera la App.*
    - *Excepciones: Se deberá tener torneos creados y torneos participados.*

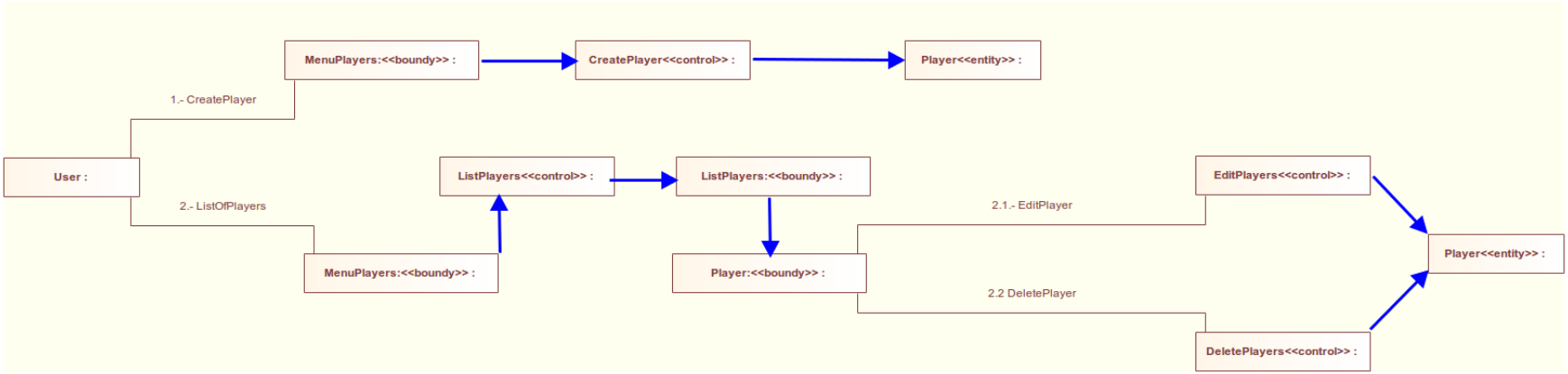
Caso de uso nº 7.3

Usuario: *Ver estadísticas usuario.*

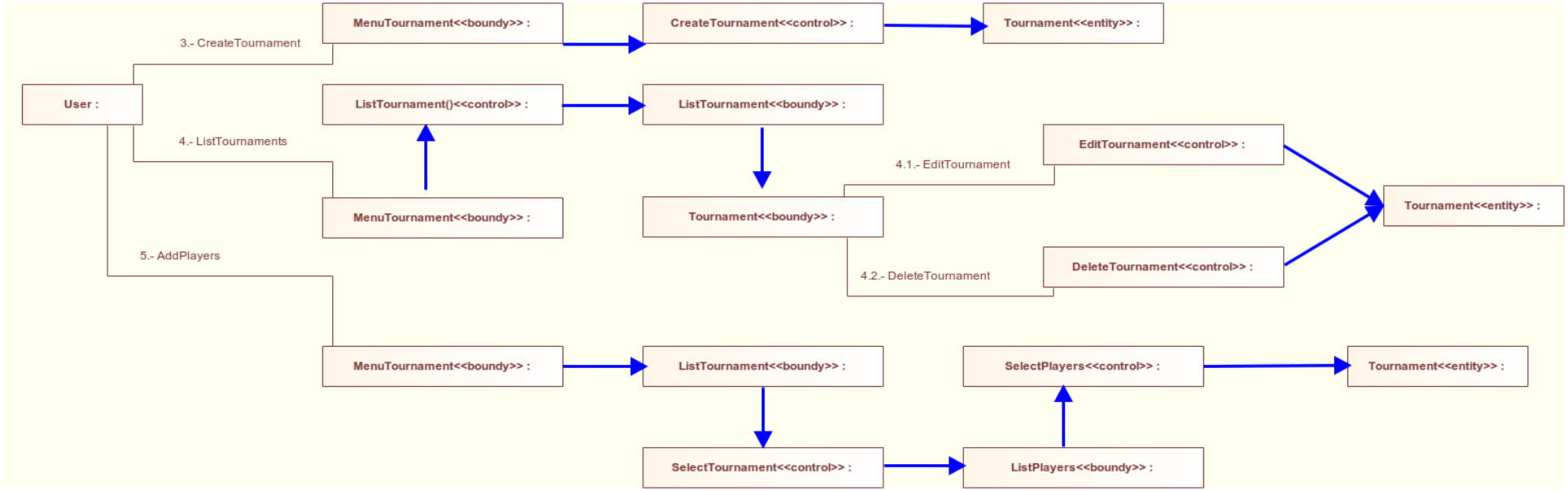
- *Resumen de la funcionalidad: Acceder a las estadísticas de un usuario*
- *Papel dentro del trabajo de usuario: Caso secundario*
- *Actores relaciones con el caso de uso: Usuario.*
- *Precondición: Debe haber participado al menos en un torneo almacenado en la base de datos un torneo finalizado.*
- *PostCondición: Se mostrarán datos como:*
        - *Número de sets ganados*
        - *Número de partidos participados*
        - *Número de torneos participados*
        - *Número de torneos ganados*
        - *Número de juegos Ganados por el usuarios*
        - *Número de juegos Ganador por el Ganador*
        - *Ranking del torneo*
- *Descripción del caso de uso:*
    - *Común: Conocer los diferentes estadísticas que genera la App.*
    - *Excepciones: Se deberá tener torneos creados y torneos participados.*

Diagramas de colaboración

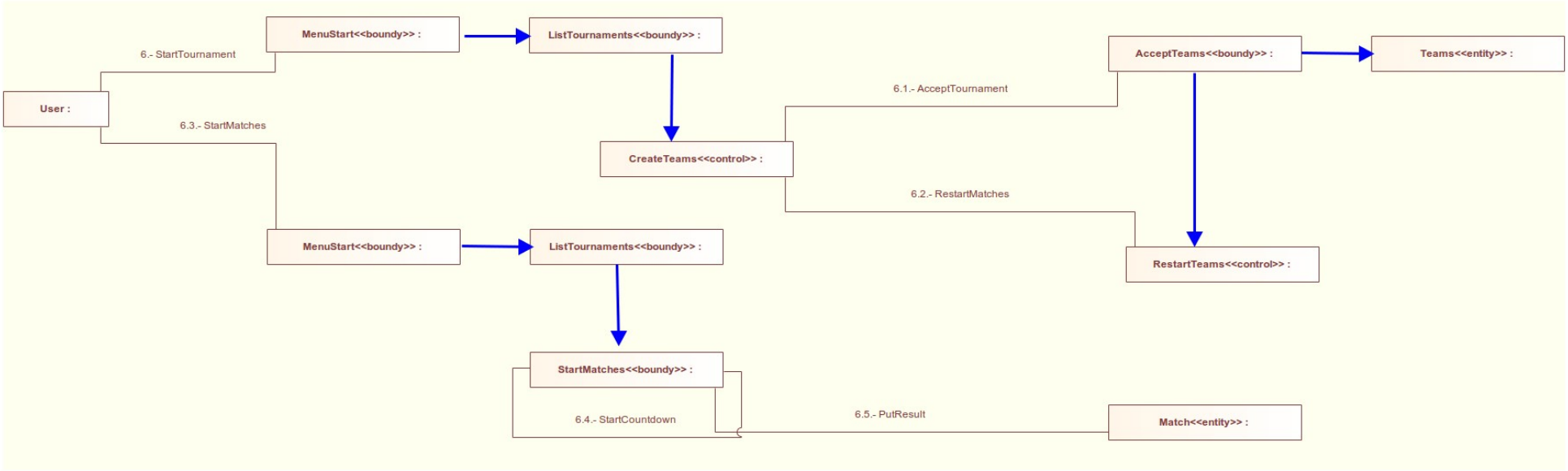
Colaboración con el MenuPlayer



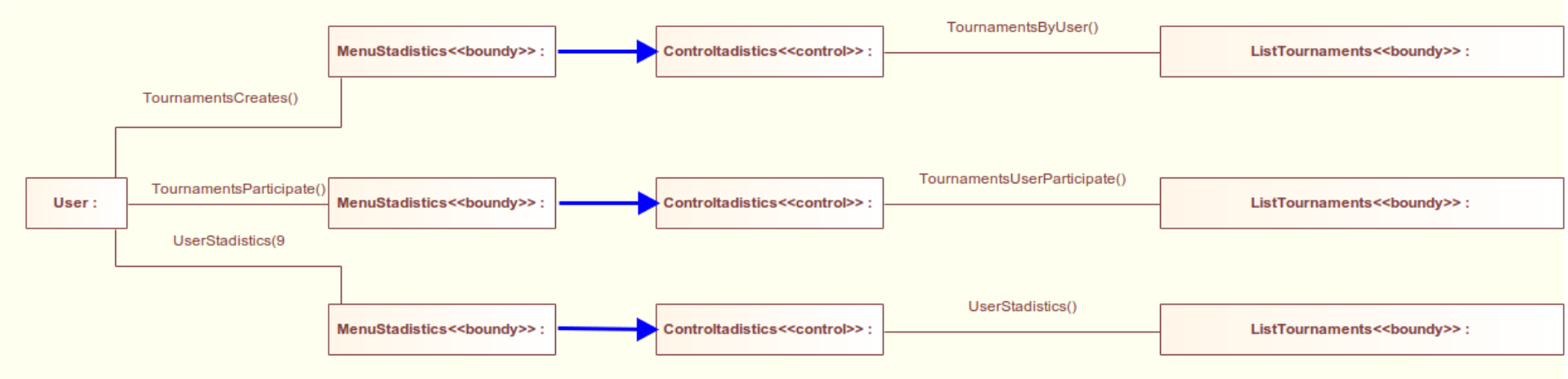
Colaboración con MenuTournament



Colaboración StartTournament

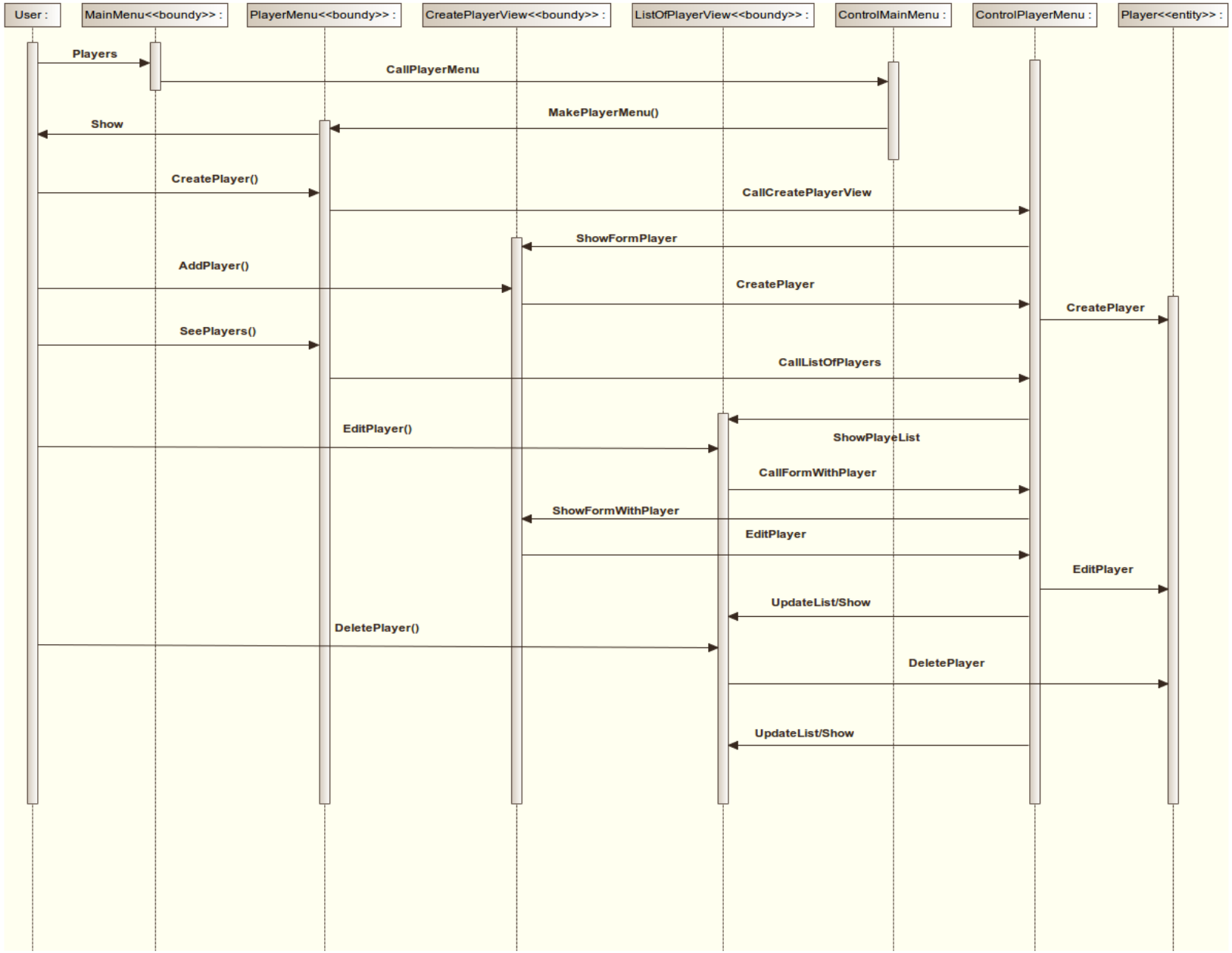


Colaboración MenuRanking

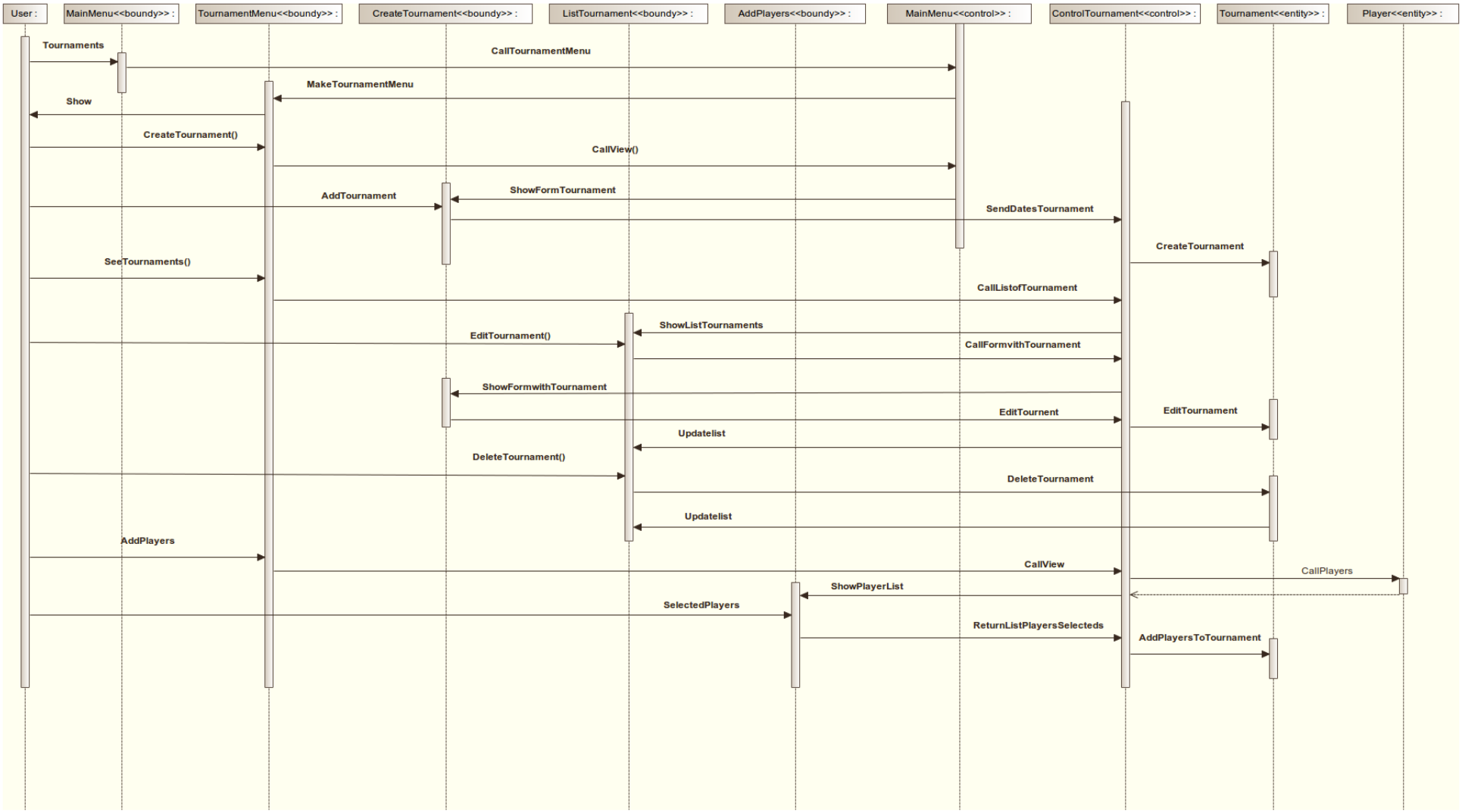


Diagramas de secuencia

1. *MenuPlayer*



2. *MenuTorneo*



3. *MenuStartTournament*

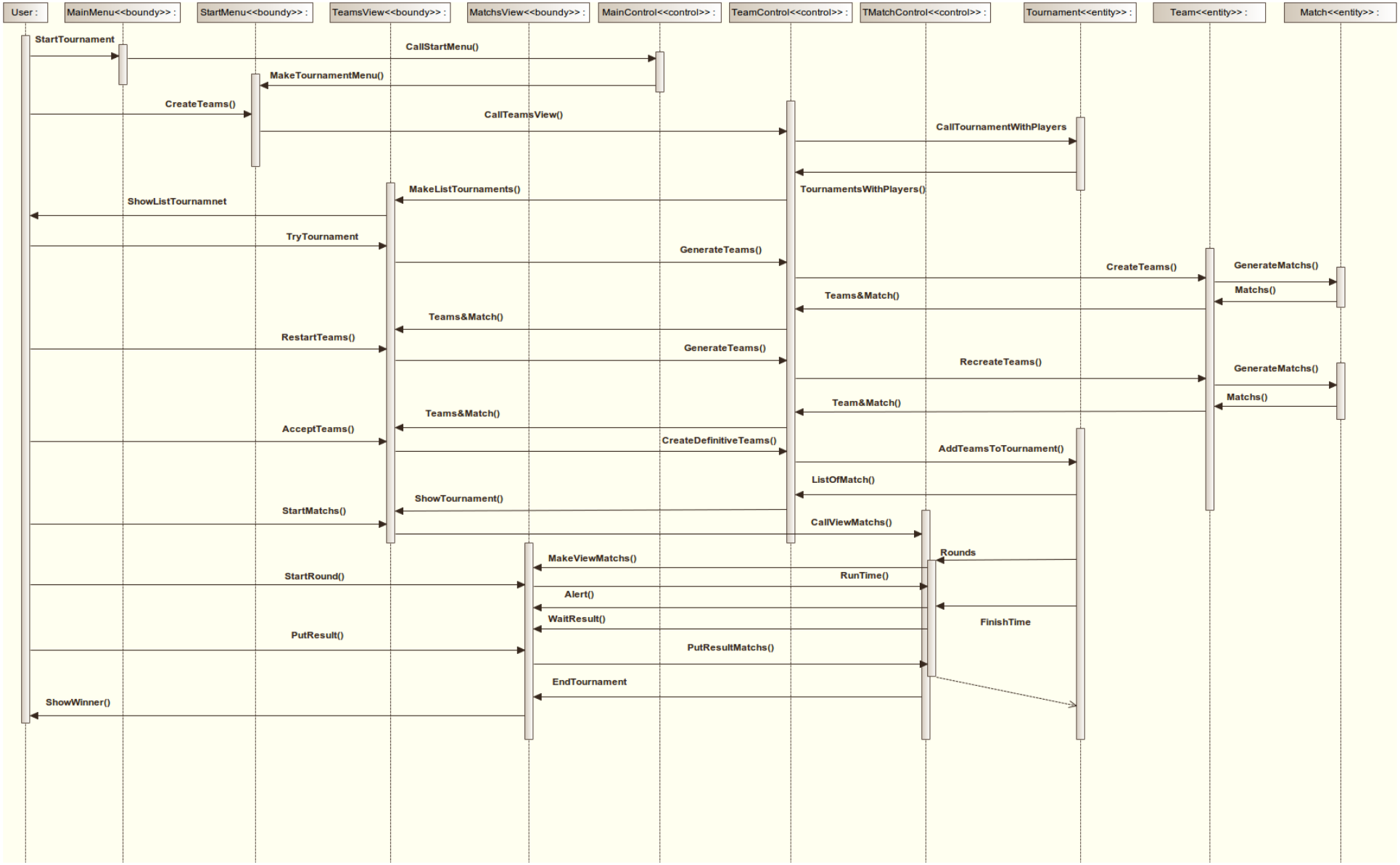


Diagrama de Clases

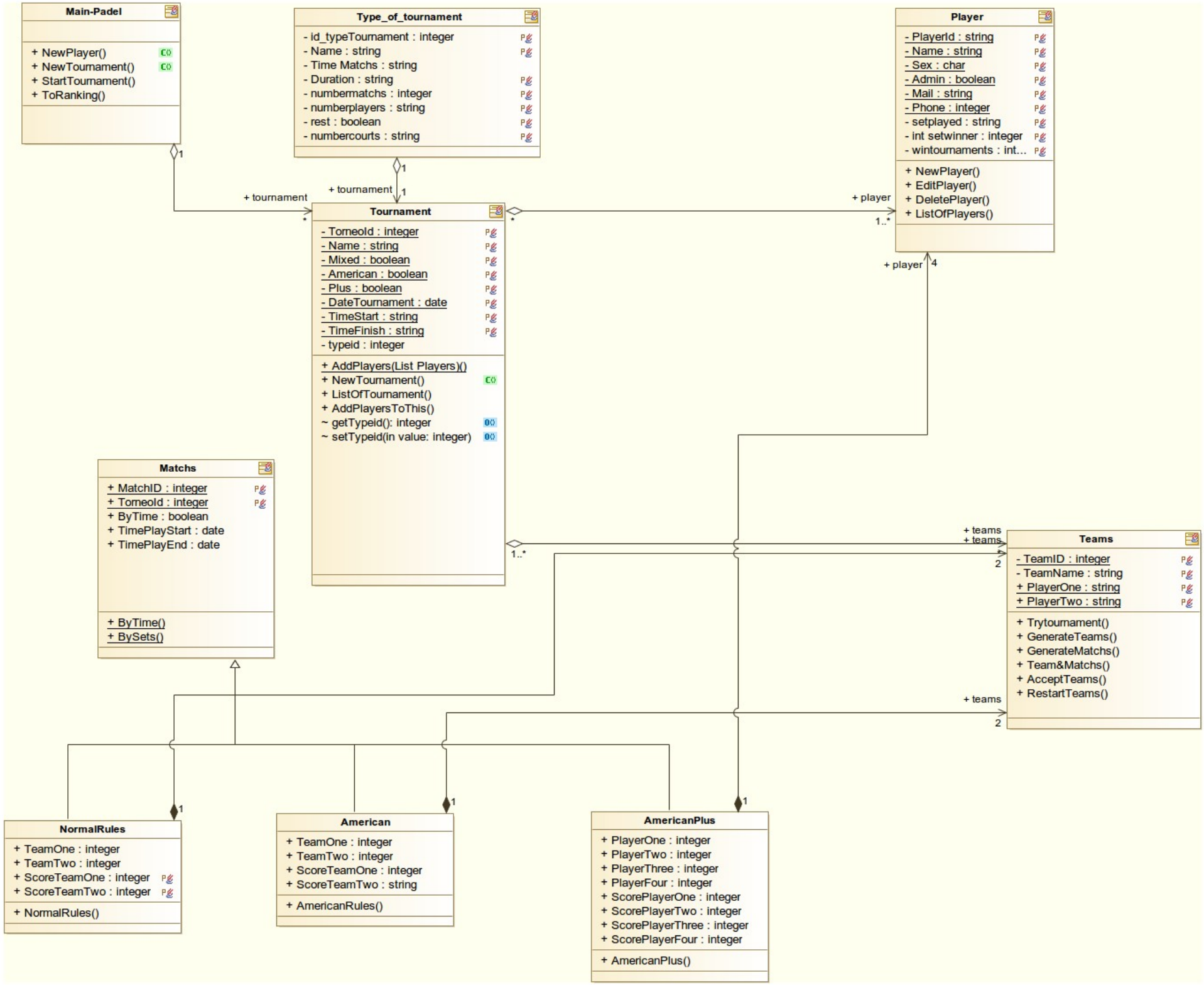

Main-Padel
+ NewPlayer()
+ NewTournament()
+ StartTournament()
+ ToRanking()
Type_of_tournament
- id_typeTournament : integer
- Name : string
- Time Matchs : string
- Duration : string
- numbermatchs : integer
- numberplayers : string
- rest : boolean
- numbercourts : string
Player
- PlayerId : string
- Name : string
- Sex : char
- Admin : boolean
- Mail : string
- Phone : integer
- setplayed : string
- int setwinner : integer
- wintournaments : int...
+ NewPlayer()
+ EditPlayer()
+ DeletePlayer()
+ ListOfPlayers()
Tournament
- TorneoId : integer
- Name : string
- Mixed : boolean
- American : boolean
- Plus : boolean
- DateTournament : date
- TimeStart : string
- TimeFinish : string
- typeid : integer
+ AddPlayers(List Players)()
+ NewTournament()
+ ListOfTournament()
+ AddPlayersToThis()
~ getTypeid(): integer
~ setTypeid(in value: integer)
Matchs
+ MatchID : integer
+ TorneoId : integer
+ ByTime : boolean
+ TimePlayStart : date
+ TimePlayEnd : date
+ ByTime()
+ BySets()
Teams
- TeamID : integer
- TeamName : string
+ PlayerOne : string
+ PlayerTwo : string
+ Trytournament()
+ GenerateTeams()
+ GenerateMatchs()
+ Team&Matchs()
+ AcceptTeams()
+ RestartTeams()
NormalRules
+ TeamOne : integer
+ TeamTwo : integer
+ ScoreTeamOne : integer
+ ScoreTeamTwo : integer
+ NormalRules()
American
+ TeamOne : integer
+ TeamTwo : integer
+ ScoreTeamOne : integer
+ ScoreTeamTwo : string
+ AmericanRules()
AmericanPlus
+ PlayerOne : integer
+ PlayerTwo : integer
+ PlayerThree : integer
+ PlayerFour : integer
+ ScorePlayerOne : integer
+ ScorePlayerTwo : integer
+ ScorePlayerThree : integer
+ ScorePlayerFour : integer
+ AmericanPlus()
+ tournament
+ tournament
+ player
+ player
+ teams
+ teams
+ teams
1
*
2
4

Diagrama Entidad – Relación

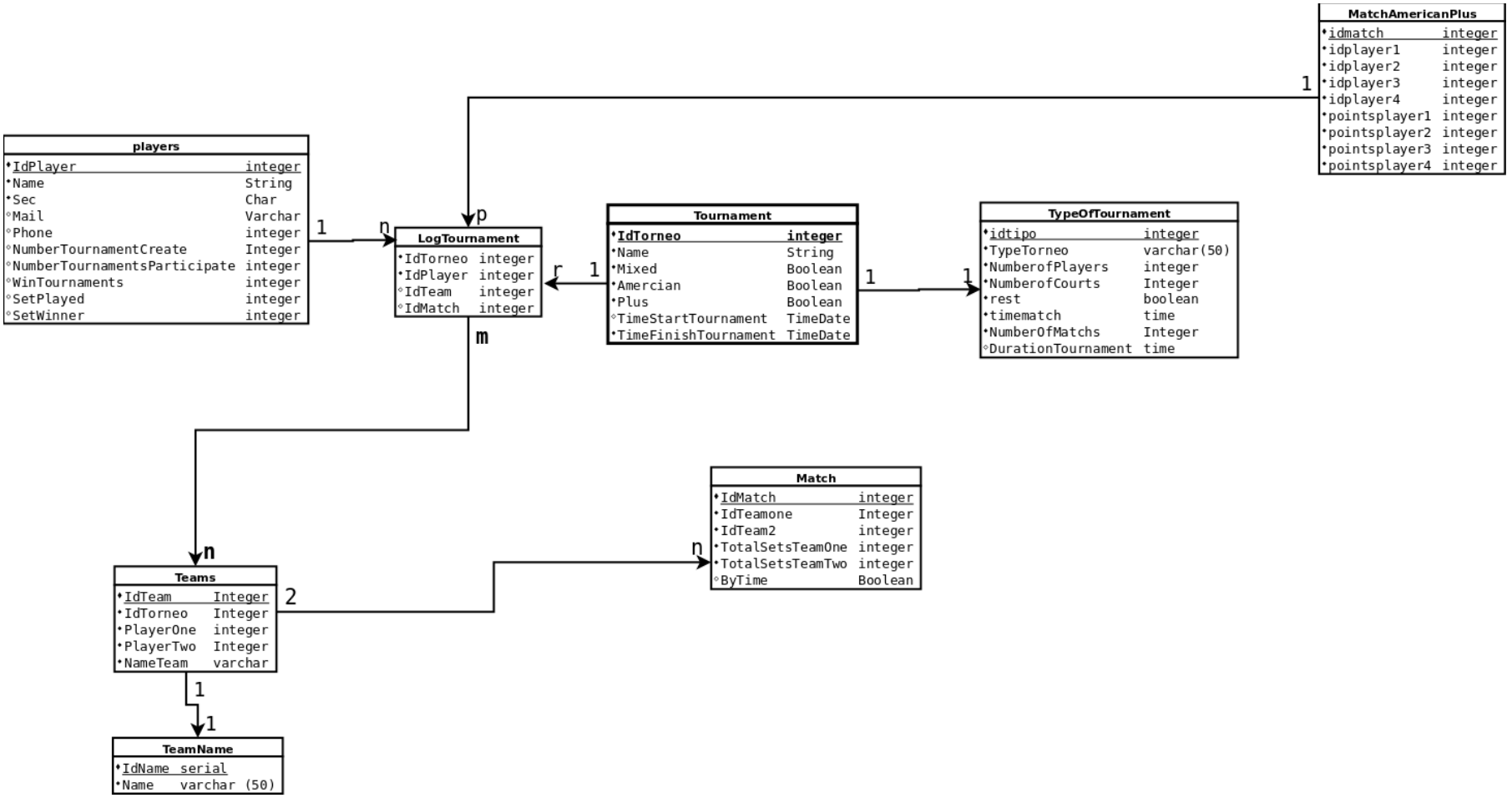
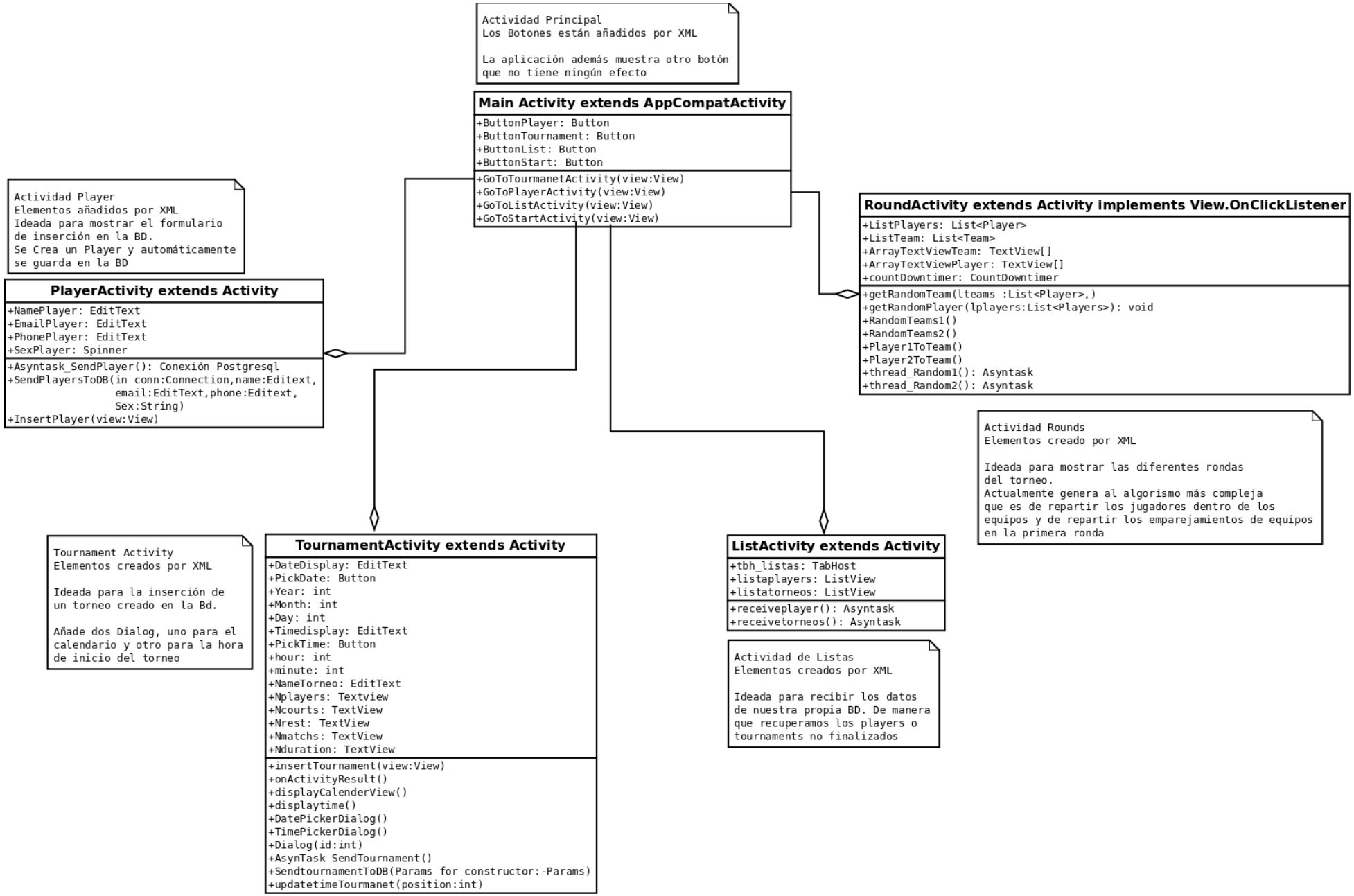


Diagrama de Activitys

Diagrama con el estado actual de la aplicación


Actividad Principal
Los Botones están añadidos por XML
La aplicación además muestra otro botón que no tiene ningún efecto
Main Activity extends AppCompatActivity
+ButtonPlayer: Button
+ButtonTournament: Button
+ButtonList: Button
+ButtonStart: Button
+GoToTourmanetActivity(view:View)
+GoToPlayerActivity(view:View)
+GoToListActivity(view:View)
+GoToStartActivity(view:View)
Actividad Player
Elementos añadidos por XML
Ideada para mostrar el formulario de inserción en la BD.
Se Crea un Player y automáticamente se guarda en la BD
PlayerActivity extends Activity
+NamePlayer: EditText
+EmailPlayer: EditText
+PhonePlayer: EditText
+SexPlayer: Spinner
+Asyntask_SendPlayer(): Conexión Postgresql
+SendPlayersToDB(in conn:Connection,name:Editext, email:EditText,phone:Editext, Sex:String)
+InsertPlayer(view:View)
RoundActivity extends Activity implements View.OnClickListener
+ListPlayers: List<Player>
+ListTeam: List<Team>
+ArrayTextViewTeam: TextView[]
+ArrayTextViewPlayer: TextView[]
+countDowntimer: CountDowntimer
+getRandomTeam(lteams :List<Player>,)
+getRandomPlayer(lplayers:List<Players>): void
+RandomTeams1()
+RandomTeams2()
+Player1ToTeam()
+Player2ToTeam()
+thread_Random1(): Asyntask
+thread_Random2(): Asyntask
Actividad Rounds
Elementos creado por XML
Ideada para mostrar las diferentes rondas del torneo.
Actualmente genera al algorismo más compleja que es de repartir los jugadores dentro de los equipos y de repartir los emparejamientos de equipos en la primera ronda
Tournament Activity
Elementos creados por XML
Ideada para la inserción de un torneo creado en la Bd.
Añade dos Dialog, uno para el calendario y otro para la hora de inicio del torneo
TournamentActivity extends Activity
+DateDisplay: EditText
+PickDate: Button
+Year: int
+Month: int
+Day: int
+Timedisplay: EditText
+PickTime: Button
+hour: int
+minute: int
+NameTorneo: EditText
+Nplayers: Textview
+Ncourts: TextView
+Nrest: TextView
+Nmatchs: TextView
+Nduration: TextView
+insertTournament(view:View)
+onActivityResult()
+displayCalenderView()
+displaytime()
+DatePickerDialog()
+TimePickerDialog()
+Dialog(id:int)
+AsynTask SendTournament()
+SendtournamentToDB(Params for constructor:-Params)
+updatetimeTourmanet(position:int)
ListActivity extends Activity
+tbh_listas: TabHost
+listaplayers: ListView
+listatorneos: ListView
+receiveplayer(): Asyntask
+receivetorneos(): Asyntask
Actividad de Listas
Elementos creados por XML
Ideada para recibir los datos de nuestra propia BD. De manera que recuperamos los players o tournaments no finalizados

Requisitos Formularios

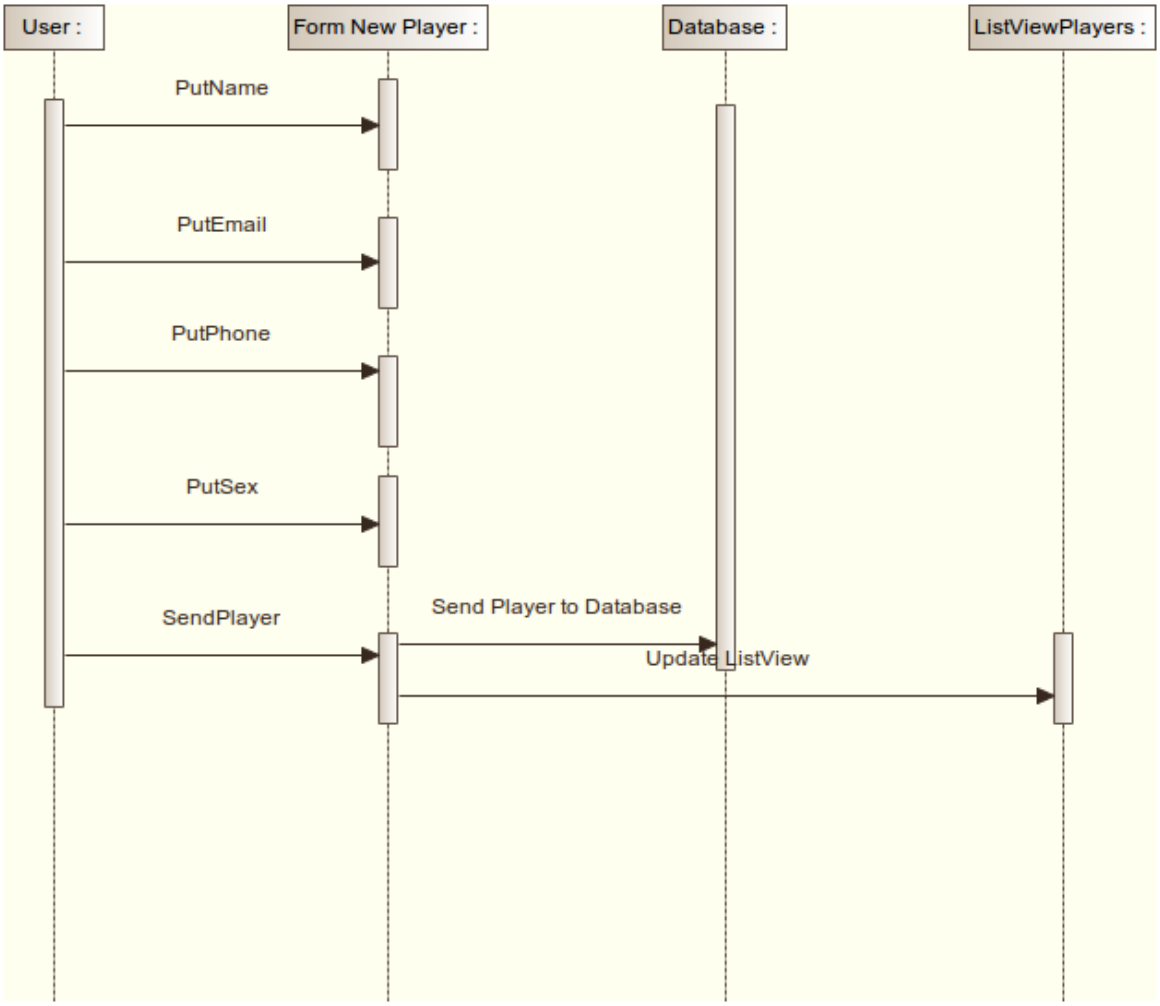
Formulario “NewPlayer”

Requisitos

- Informe de Requisitos

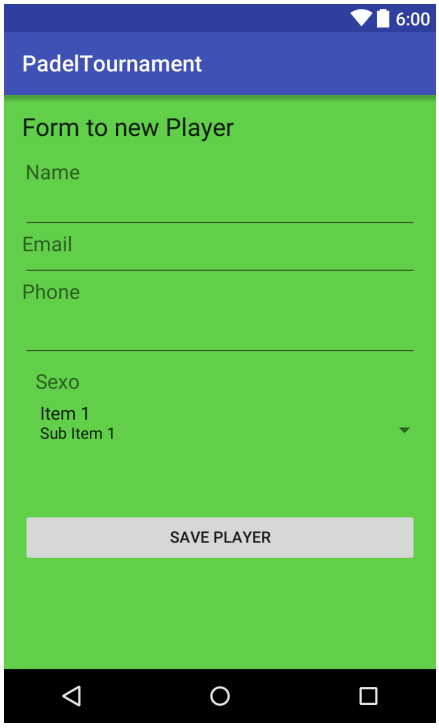
  Este formulario esta diseñado para la creación de un nuevo objeto de la clase jugador, para su posterior uso en los torneos y su almacenaje en la base de datos remota.

- Caso de uso y diseño de interacción.
  - Caso de uso: Cada vez que queramos añadir un nuevo jugador en nuestra aplicación deberemos rellenar el formulario correspondiente y guardar esos datos en la base de datos.
  - Diseño de interacción.



- Diagrama de secuencia (FORMS,DIALOG)
- Detectar elementos funcionales.
    - EditText para el rellenado de campos
    - Button para el envió del nuevo jugador.
- Evaluación de elementos.

  Tenemos varios elementos en este formulario:
    - TextView. Para las cabeceras de los campos de inserción:
        - Name, Email, Phone y Sexo.
    - EditText. Para el campo Name.
    - EditText(tipo email). Para el campo Email.
    - EditText(tipo teléfono). Para el campo Phone.
    - Spinner con las letras V y M. Para el campo Sexo
    - Button. Para enviar actualizar la base de datos y el ListView que muestra todos los jugadores.
- Prototipos



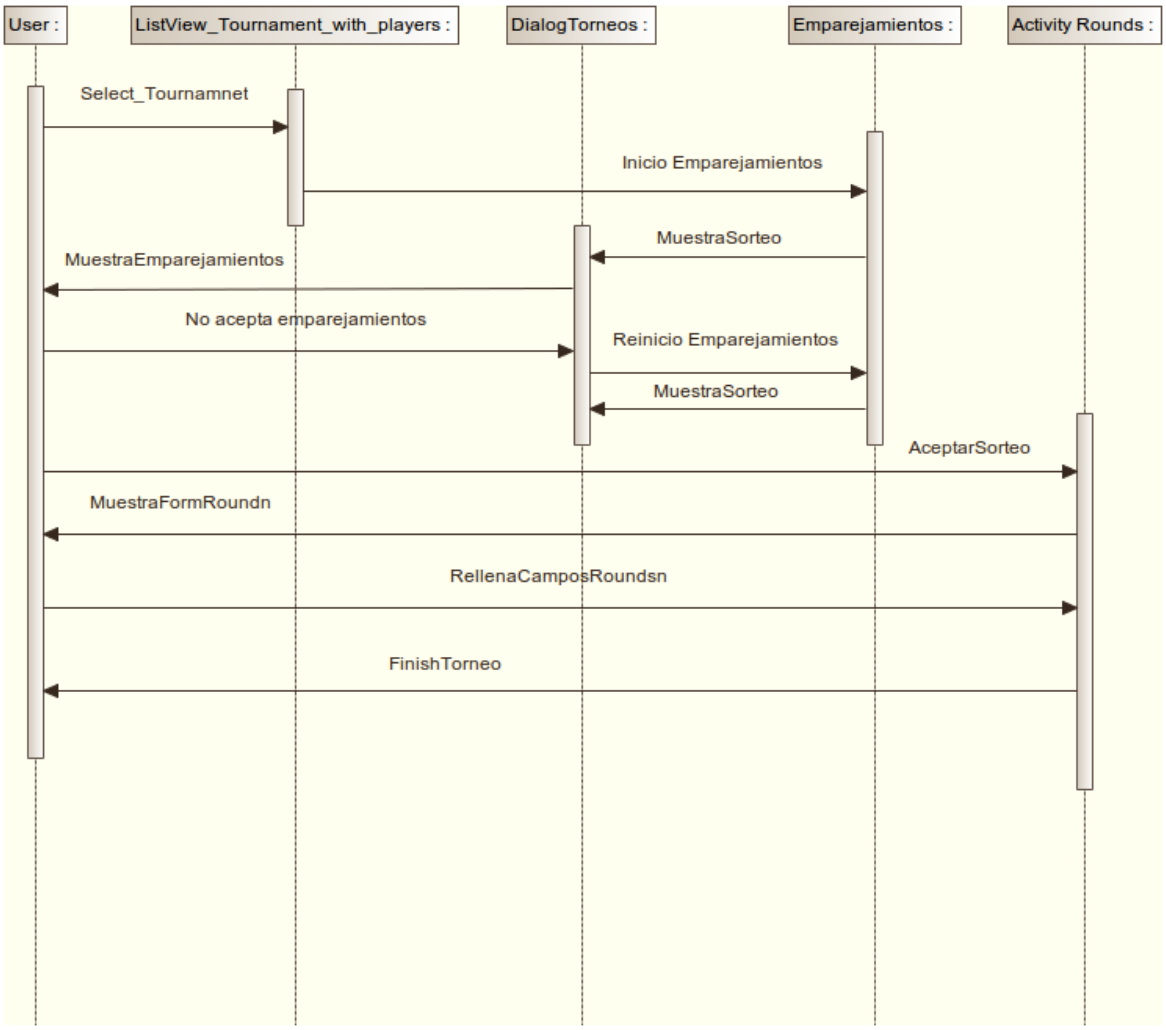
Informe de formulario “NewTournament”

Requisitos

- Informe de Requisitos

  Este formulario esta diseñado para la inserción en la bd y la creación en memoria de un nuevo objeto Tournament.

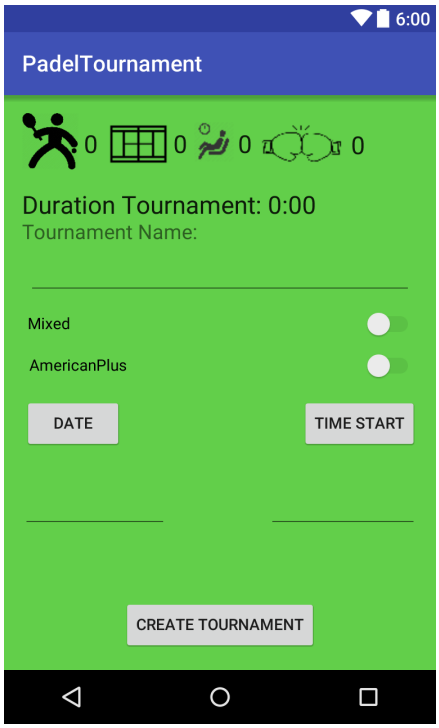
- Caso de uso y diseño de interacción.
  - Caso de uso:
    - Cada vez que queramos crear un nuevo torneo esta será el formulario a rellenar.
  - Diseño de interacción.



- Diagrama de secuencia (DIALOG,ACTIVITY)
- Detectar elementos funcionales.
    - Activity activity_tournament.
        - CustomDialog. Nos mostrará una lista de tipos de torneos que se iniciará cada vez que entremos en este formulario
        - 4 Image View, iconos que representan el número de jugadores por torneo, el número de pistas asociadas al torneo, si tiene descanso y el número de partidos.

        - Dos TextView para los títulos de los campos y para mostrar la duración total del torneo.
        - Tres EditText donde recogeremos el nombre del torneo la fecha y la hora de inicio.
        - Dos Switch uno para definir si son mixtos y el otro el tipo de torneo. Según el tipo de torneo utilizaremos un algoritmo diferente para los emparejamientos.
        - Dos CustomDialog más uno para introducir la fecha y otro para introducir la hora de inicio.
        - CalendarView y TimerView

- Prototipo


6:00
PadelTournament
0
0
0
0
Duration Tournament: 0:00
Tournament Name:
Mixed
AmericanPlus
DATE
TIME START
CREATE TOURNAMENT

Estado actual de las tecnologías

Servidor Dedicado.

Ya que tengo a mi disposición un servidor dedicado con varias direcciones IP estáticas y diferentes servicios se esta utilizando para realizar las pruebas y almacenar los diferentes datos que genera la aplicación, pasamos a una breve descripción del servidor:

Debian
Conexión de red 100 Mbps
CPU Intel Atom N2800 1.86GHz 4 Núcleos
4GB RAM
Dos discos de 500GB en Raid1

Este servidor alberga servicios como apache, bind9, iptables, mysql y postgresql. La aplicación únicamente utilizará el sistema gestor de bases de datos.

Android

Android es un sistema operativo basado en linux y diseñado para todo tipo de smartphones, tables o dispositivos móviles Actualmente tiene una cuota de entre el 81% y 88% del mercado mundial. (Digo entre porque tras comprobar varias fuentes de información ninguna coincide y se hace difícil aportar una cifra exacta).

Android nos aporta algunos detalles que su competidor directo IOS nunca podrá asumir:

1. Múltiples dispositivos
2. Múltiples precios.
3. Personalización
4. Widgets
5. Launchers
6. Roms cocinadas (Nos da la posibilidad de retocar y mejorar nuestro propio sistema operativo)
7. Perfecta integración con los servicios de Google.
8. Mayor cantidad de aplicaciones y juegos gratuitos.
9. Mayor velocidad de mejora.

IDE Android Studio

Ya que voy a realiza una aplicación para Android que mejor IDE utilizar que el suyo propio. Android Studio es una herramienta para el desarrollo de aplicaciones en todo tipo de sistemas móviles. Su primera versión estable fue publicada en diciembre del 2014.
Esta basado en el software IntelliJ de JetBrains es publicado de forma gratuita bajo la licencia de Apache 2.0.
Require de un equipo de potencia media en la actualidad, estos son sus requisitos:

- *2GB de RAM (4GB recomendados)*
- *1 GB para Android SDK*
- *Monitor de 1280 X 800*
- *Java Development Kit 7*

Nos puede facilitar mucho el trabajo gracias a su intuitiva interfaz, a la cantidad de documentación que hay en la red y a que es para algunas cosas bastante automático, por ejemplo cada que creamos una activity el genera la correspondiente línea en el android_manifest.xml.
Por remarcar un parte negativa diré el poco tiempo que lleva en uso en comparación con otros IDES como Eclipse o NetBeans.

Android SDK

El SDK (Software Develepoment Kit) de Android incluye un conjunto de herramientas de desarrollo. En las que intervienen estas herramientas:

1. *Un depurador de código.*
2. *Una biblioteca.*
3. *Un simulador de teléfono basado en QEMU.*
4. *Documentación.*
5. *Ejemplos de código.*
6. *Tutoriales.*

Sus actualizaciones están coordinadas con el desarrollo general de Android.

JDBC Postgresql

La Api jdbc_postgresql hecha en JAVA 1.1.
Esta Api provee a la aplicación de un driver de tipo 4. Esto nos indica que el driver esta escrito en JAVA y las comunicaciones con el sistema de bd se realiza mediante protocolos de internet. De esta manera consiguen tener que el driver pueda usarse en cualquier plataforma.

Riesgos

Tiempo para la ejecución

Dependiendo del número de objetivos el tiempo podría ser un problema. Aunque también puede ser un plus para mi experiencia y entrega al trabajo de programación.

Android

Hay muchas formas de realizar una aplicación en Android elegir la correcta para mi caso quizás sea un futuro error ya que no tengo mucha experiencia con este sistema operativo.

Inexperiencia en el desarrollo de aplicaciones

Al ser neófito en el desarrollo de aplicaciones la carga de trabajo para una persona sola puede ser excesiva, ya que las prisas y el stress que se generará puede provocar más errores de los que ya mi inexperiencia aportará.

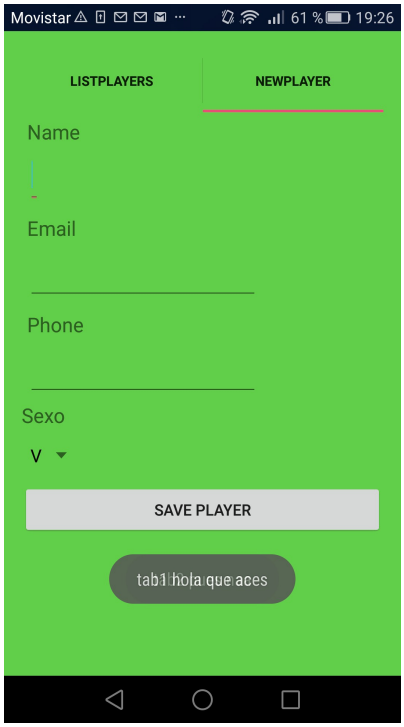
3. Conclusiones

¿ Que he aprendido con la realización del proyecto ?

Realizar este proyecto me ha servido para adquirir nuevos conocimientos y aptitudes en diferentes campos.

Java y Android

Toda el código esta escrito en Java y XML. Al principio del desarrollo, diseñe las interfaces para que algunas de ellas incluyeran un Tabhost para mostrar e insertar la diferente información. Una muestra:



Como Android recomienda los Tabhost deberían usarse con Fragments, objetos con métodos muy parecidos a los de Activity, así que no lo dude y me lance ha lo que iba a ser mi mayor creación.

Con la interfaz orientada a el objeto Player tuve problemas pero conseguí que fuese funcional y práctica. La interfaz de Tournament fue otra historia, ya que debía incluir un objeto de la clase DialogAlert. Después de varios intentos por mostrar una lista de objetos customizada, tuve que optar por mostrar un Array en el dialog, pero el imprevisto final vino al no poder estabilizar el formulario de creación. En ese momento decidí cambiar mis clases de Fragment a Activity como hemos visto en clase.

Cabe añadir que el inicio del proyecto se inicio junto con el inicio de las clase se Android.

Todo este proceso me ha ayudado a aprender:

- El intercambio de información entre Activitys, DialgosAlerts, Fragments.
- Ha asumir de mejor manera conceptos como Context y Views.
- Diseño de formularios para varios tipos de resoluciones, obtención y recepción de información desde Android a una bd postgresql.
- Asyntask de android.

Diseño de aplicaciones

Gracias al proyecto he podido mejorar mi aptitudes en el diseño de aplicaciones, las charlas con los profesores y compañeros sobre este proyecto y los otros ayudan a abrir la mente a nuevas maneras de pensar, procesar y ejecutar. Ha sido muy gratificante ver esto de cerca y participar.
El diseño de la aplicación es funcional al 100%. Lástima no haber tenido más tiempo o más conocimientos en Android para haber mostrado una demostración completa.

Gestión de tiempo

Actualmente puedo ser más concreto a la hora de generar un plan de trabajo que cuando inicie el proyecto. He visto que el tiempo necesario para acabar con éxito de algunas tareas depende mucho de la información de la que disponemos para trabajar y de nuestra habilidad para desarrollar e interactuar tanto con el código como los diferentes servicios o servidores.

Herramientas utilizadas

Modelio para el diseño de los diagramas de la aplicación.
Día para el diseño de los diagramas de entidad relación.
Pgadmin y Postgresql, tanto para la gestión como para la creación de la base de datos.
Android Studio para la creación de las interfaces y todo el código de la aplicación.
Gant para el desarrollo del plan del trabajo.
Prezi para el diseño de la presentación del proyecto.
Herramientas ofimáticas para la creación de la memoria.

Mejora de habilidad de búsqueda

Hoy en día en muy importante saber buscar en internet, durante el proceso de escritura del código me he encontrado muchas dificultades por lo comentado más arriba, esta ha hecho que mejore mis dotes de

búsqueda en internet gracias a la constancia y algunos tutoriales de como usar Google.

¿Se ha cumplido con los objetivos?

Aunque en la demostración se cumplen todos los objetivos en parte, no se ha cumplido con los objetivos. Por ejemplo solo se puede crear un tipo de torneo en la actualidad.
Quizás los objetivos eran demasiados presuntuosos para mí nivel de programación y mi conocimiento de Android. Pienso que si hubiese hecho la aplicación para un S.O. como linux no hubiese tenido tanto problemas.

También me he equivocado en otras puntos como pueden ser:
El tiempo que he ocupado en el estudio de diferentes formas de mostrar la aplicación que lo podía haber ocupado en desarrollo o mejora de la bd, ya que cuando me decidí por diferentes formas no pude mostrar crear ninguna de ellas.
El diseño de la aplicación se puede mejorar y ampliar. Implementando usuarios y rankings de Google, mejoras en los atributos de player, más tipos de torneos y sorteos pero la base actual es 100% solvente.

La creación de la BD de datos en Postgresql no fue un error pero tampoco fue muy acertado, ya que la conexión con la base de datos se debía hacer bajo un proceso(Thread) y costo asimilarlo. Por suerte las clases sobre Asyntask justo comenzaban cuando inicie los primeros intentos de conexión.

¿Se ha seguido la planificación?

Se ha seguido aunque no era del todo buena, ya que tenía dedicado demasiado tiempo el estudio de diferentes formas de diseño de layouts que luego no ha podido utilizarse por su dificultad.

Cabe añadir que hoy 18 de Mayo se esta empezando a retocar la memoria del proyecto tal y como se puso en la planificación.

Lineas de trabajo futuro

1. Geolocalización.
2. Utilización de la Api de Google para usuarios.
3. Utilización de la Api de Google para juegos y su posible adaptación a el proyecto.
4. Mejora visual del proyecto.
5. Mejora la bd para que liberé de la mayor carga posible a la aplicación, mejora de su seguridad y una mejora en las copias de seguridad. Implementación de Servidores Esclavos – Maestros
6. Una mayor utilización de CustomFragments y CustmoDialogs.

7. La integración en la app de las diferentes modalidades que se tomaron como ejemplo en su día.

Conclusión

El proceso de realización ha sido tedioso y estresante, *una buena práctica para la vida real*. Ha servido para recordarme el largo camino que aún me queda por recorrer para poder realizar una aplicación decente.

Aun así lo volvería a hacer solo ya que todos los fallos de diseño, errores de planificación, errores de código son míos e intentaré no volver a repetirlos.
Al final ha sido una experiencia gratificante.

Gracias a las habilidades adquiridas en durante estos dos años y podido crear algo de la nada.

4. Glosario

Términos y acrónimos generales.

1. **BD**. Base de datos.
2. **APP**. Aplicación.
3. **Api**. Aplication Programming Interface. Es el conjunto de subrutinas, funciones y métodos que ofrece cierta dicha biblioteca para ser utilizado por otro software como una capa de abstracción.
4. **Postgresql**. Sistema gestor de bases de datos más potente que Mysql.
5. **QUEMU**. Es un emulador de procesadores basado en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped).

Términos y acrónimos concretos de la aplicación

6. **Player**. Objeto java de una clase que he creado para gestionar la participación en los diferentes torneos.
7. **Tournament**. Objeto java de una clase que he creado para poder gestionar los diferentes Torneos.
8. **Set**. Forma en la que se puntuarán los partidos de padel. Equivalente a los goles en el futból.

Términos y acrónimos concretos de Android

9. **Activity**. Es como una ventana centrada en lo que usuario puede hacer.
10. **Fragment**. Representa un comportamiento o un porción de interfaz.
11. **Dialog**. Es una pequeña venta que pide al usuario que tome un decisión o introducir información adicional.
12. **CustomFragment**. Es un Fragment personalizado.
13. **CustomDialog**. Es un Dialog personalizado, al cuál podemos poner estilos, listas personalizadas, etc.
14. **Views**. Representan el bloque de construcción básico para los componentes de interfaz de usuario. Ocupa el área rectangular en la pantalla y es responsable de la elaboración y gestión de eventos. Se utiliza para crear componentes de interfaz interactivos.
15. **Context.** Contiene la información global sobre un entorno de aplicación, permite el acceso a los recursos específicos de la aplicación.
16. **Layout.** Definen la estructura visual de una interfaz de usuario.
17. **Tabhost.** Contenedor para una vista de ventana con pestañas. Este objeto tiene dos hijos: un conjunto de etiquetas de las solapas en las que el usuario puede hacer click para cambiar de vista y un objeto FrameLayout que muestra el contenido de la solapa que se ha pulsado.
18. **TextView.** Muestra el texto para el usuario y opcionalmente les permite editarlo.
19. **EditText.** Es una fina capa sobre un TextView que se configura para poder ser editado.

20. **Switch.** Es un interruptor de palanca de dos estados que te permite seleccionar entre dos opciones.
21. **ImageView.** Muestra una imagen arbitraria, tal como un icono. Puede cargar imágenes de diversas fuentes.
22. **ListView.** Una vista que muestra elementos de una lista de desplazamiento vertical. Los artículos viene de un **ListAdapter** asociado a con este tipo de vista.
23. **ListAdapter.** Es el puente entre un ListView y los datos que se quieren mostrar en el.
24. **CustomListView.** Es un ListView personalizado. Por ejemplo podemos querer tener una serie de imágenes y a continuación unos TextView que vayan cambiado según los elementos de la lista.
25. **CountDowntimer.** Debido a que el cronometro de Android solo cuenta hacia delante tiene en su API una herramienta que te permite una cuenta atrás desde un punto de tiempo concreto.
26. **Calendar.** Es una clase abstracta para la conversión entre un objeto Date y un conjunto de campos integer como el año, el mes, el dia o la hora.
27. **CalendarView.** Una vista o widget de calendario que nos permite visualizar y seleccionar fechas.
28. **TimerView.** Una vista o widget de una unidad de tiempo que nos permite visualizar y seleccionar un horario diferente.
29. **Asyntask.** Clase que nos permite trabajar de un manera más fácil y rápida con la multitarea de JAVA. Esta clase es exclusiva de Android.

5. Bibliografía

Web en la que he podido mirar infinidad de ejemplos.
http://stackoverflow.com/

Web de Android
https://developer.android.com

Ejemplo para la Alerta con el Calendario y el Reloj
http://www.mysamplecode.com/2011/10/android-datepickerdialog.html
http://es.androids.help/q6339

El gran libro de Android
Jesús Tomás Gironés
3ª Edición

Ejemplos de clase de Programación

Por desgracia no he podido recuperar todos las web que visite, entre las prisas y el stress muchas no quedaron guardadas.