

Universidad de Murcia  
Facultad de Informática

---

TÍTULO DE GRADO EN INGENIERÍA INFORMÁTICA

**Tecnologías de Desarrollo de Software**  
**PROFESOR: F. Javier Bermúdez Ruiz**

Apuesta Ya!

Autores:  
Cristina Pérez Orenes 48634419E  
Tony Wang Chen X1968208Y

Curso 12/13

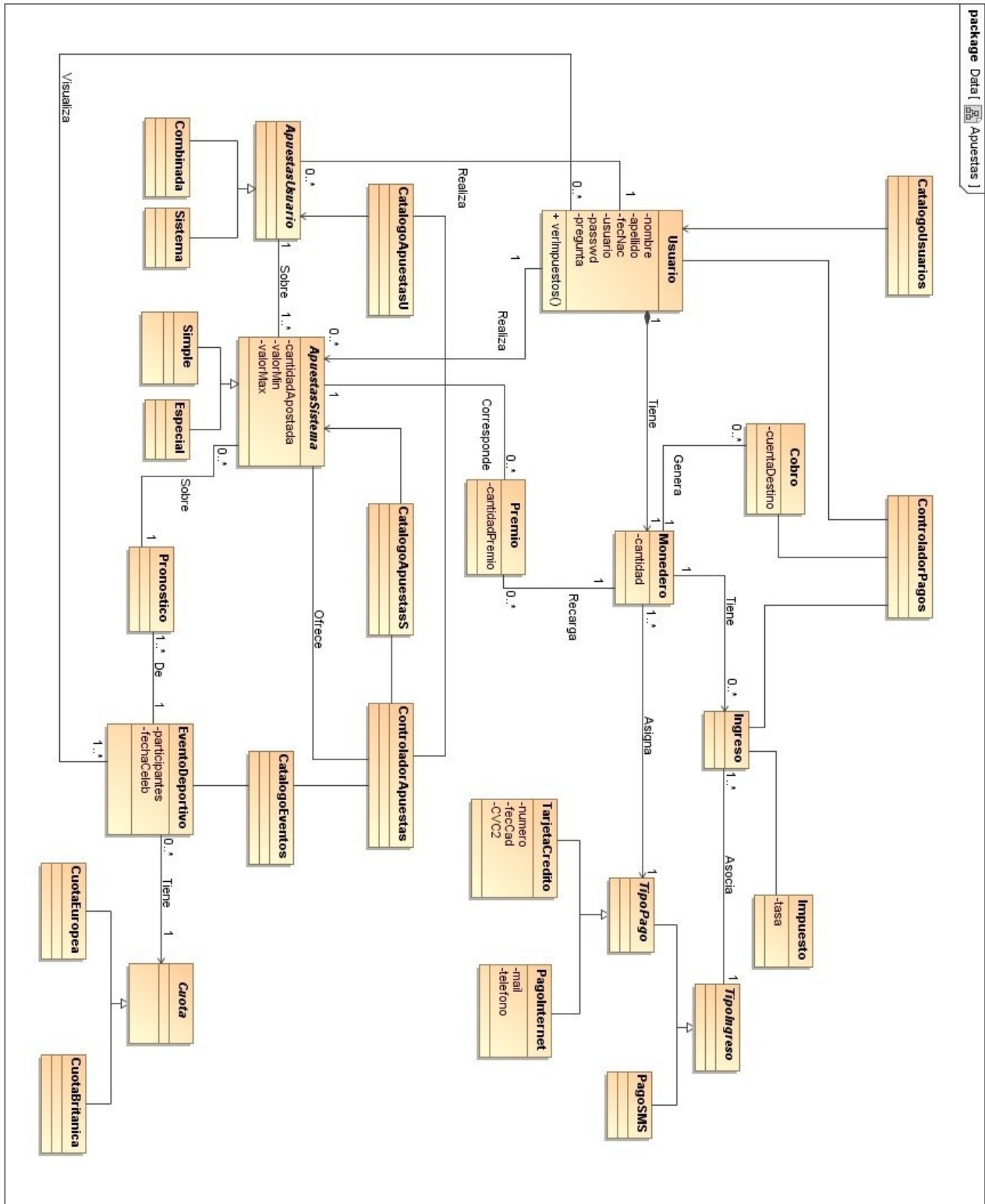
---

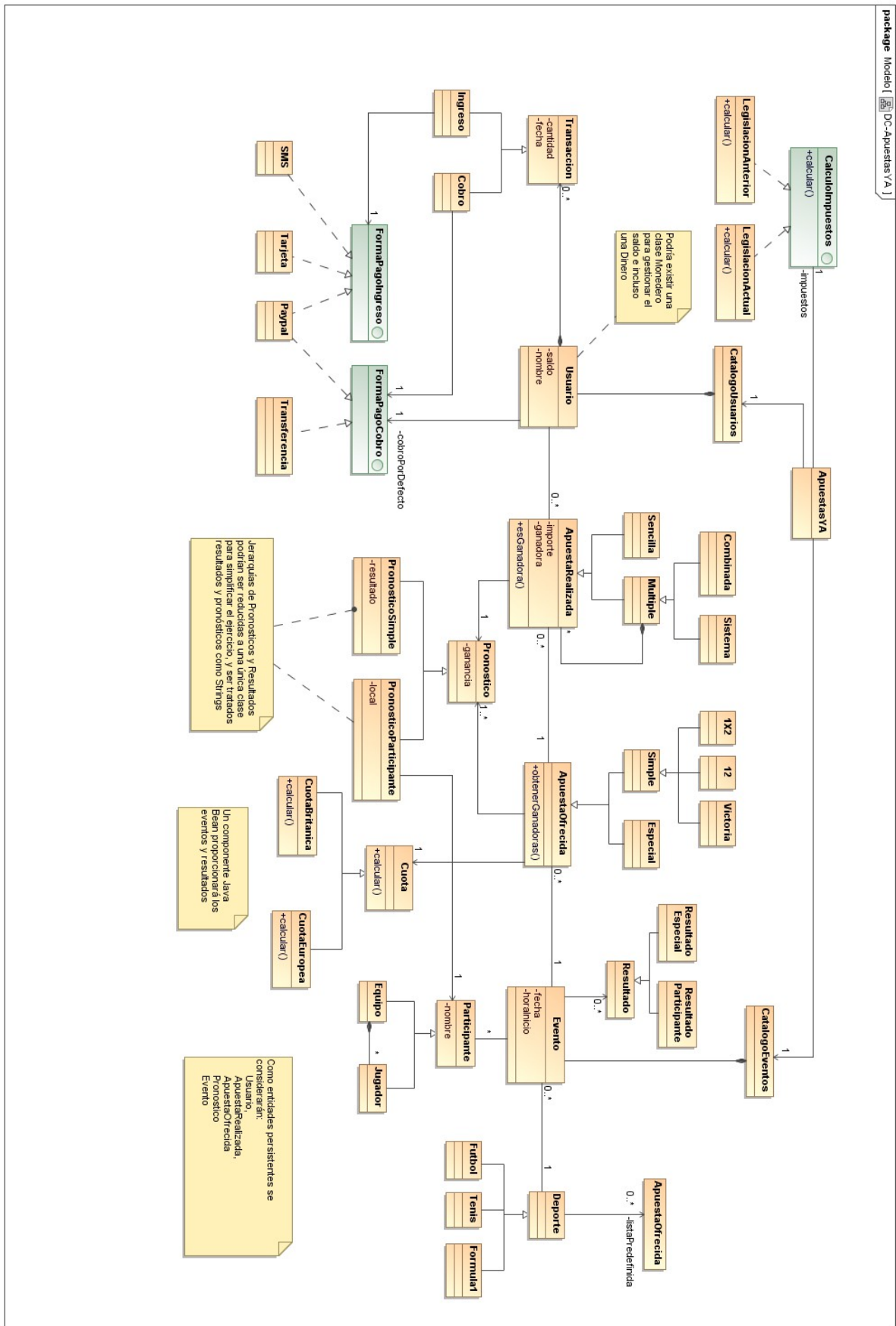
## Índice de contenido

1. Diagrama de clases.....	3
2. Realizar Apuesta: Diagrama de colaboración.....	5
3. Arquitectura de la aplicación y decisiones de diseño.....	6
4. Patrones de diseño.....	8
5. Componentes.....	12
6. Pruebas unitarias.....	13
7. Manual de usuario.....	15
7.1. Registro de un nuevo usuario.....	15
7.2. Sección de administradores.....	16
7.2.1. Crear un nuevo evento.....	17
7.2.2. Crear una nueva apuesta.....	17
7.2.3. Gestionar usuarios.....	18
7.2.4. Cargador de Resultados.....	18
7.3. Sección de usuario.....	19
7.3.1. Perfil, ingresos y cobros.....	19
7.3.2. Realizar apuestas.....	21
7.3.3. Información de apuestas y transacciones. Apuestas múltiples.....	22
8. Observaciones.....	24

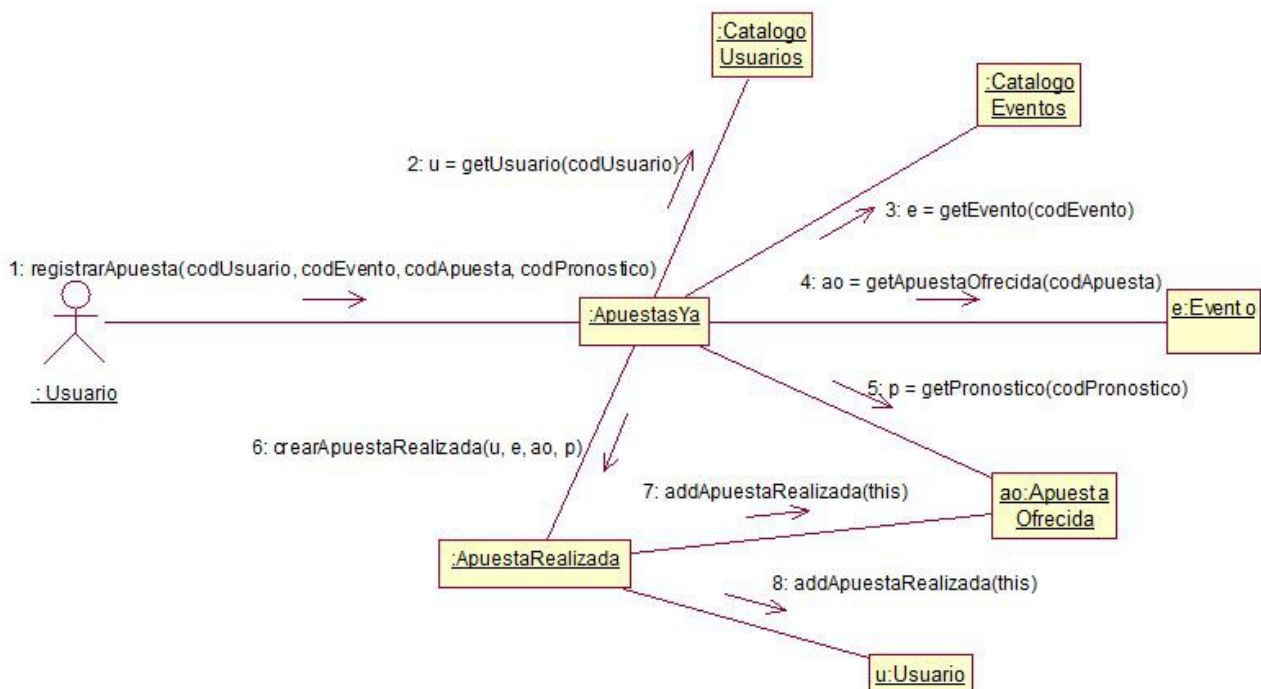
## 1. Diagrama de clases

Diagrama entregado:





## 2. Realizar Apuesta: Diagrama de colaboración



En el diagrama de colaboración se puede observar como se realiza la operación registrarApuesta donde, desde la interfaz, el usuario puede seleccionar una apuesta ofrecida en un evento y elegir un pronóstico. Estos parámetros son enviados al controlador ApuestasYa. El controlador se encarga de recuperar los objetos correspondientes a los códigos enviados desde la interfaz accediendo a los catálogos de usuarios y eventos. A continuación, con el objeto Evento obtiene el objeto de la apuesta ofrecida y, a través de ella, su pronóstico. Una vez obtenidos los objetos se crea la apuesta realizada y se añade a la lista de apuestas realizadas por el usuario y apuestas realizadas en la apuesta ofrecida. Pensamos que una posible forma de hacerlo más eficiente para evitar alta cohesión, podría ser que el usuario creara la apuesta realizada, pero por decisiones de diseño (para añadir la apuesta más rápidamente en la base de datos) hemos decidido que sea el controlador quien cree la apuesta realizada.

### 3. Arquitectura de la aplicación y decisiones de diseño

En este apartado enumeraremos aquellas decisiones de diseño importantes que se han tomado. Algunos diseños son diferentes al modelo de clases que se propone y otros por la falta de información del guión de prácticas por lo que se opta por intuir lo más lógico. Estas decisiones así como partes de la arquitectura de la aplicación las enumeramos a continuación:

- El deporte tiene también una lista de apuestas ofrecidas según el diagrama (se repite esta clase). Nosotros por facilidad y simplicidad, solamente lo ponemos en el evento. Cuando creamos el evento somos capaces de saber qué deporte es, por lo que creamos las apuestas correspondientes (1X2, 12 o Ganador) y lo asociamos a ese evento.
- Para asociar una apuesta realizada a su correspondiente ofrecida, lo que hacemos es quedarnos con el código de esa apuesta ofrecida como atributo. Lo guardamos como una cadena de caracteres, y no como el propio objeto, porque el mayoría de casos ni lo usamos y es tontería gastar memoria instanciándolo. Además, produce un bucle al intentar obtenerlas de la base de datos (ya que apuestas realizadas es de una apuesta ofrecida y esa apuesta ofrecida contiene una apuesta realizada).
- En el diagrama de clases el evento no conoce la cuota directamente. En el guión de prácticas se pide que el evento se cree con una cuota. Por tanto decidimos crear el evento con una cuota y que este sea el encargado de pasársela a la apuesta ofrecida.
- Las apuestas ofrecidas tendrán un atributo título de la apuesta para saber nosotros de qué tipo de apuesta se trata (más bien para saber en una especial por qué se está apostando y mostrarlo bien por pantalla, puesto que las otras apuestas se podrán mostrar de una manera u otra dependiendo de la clase que estemos creando [12, 1X2, ...]). Con esto, podemos luego comparar con los resultados del evento y asignar los ganadores.
- La aplicación está diseñada para que no se diferencien entre usuarios normales y administradores. Por tanto necesitamos el concepto de administrador porque será el que se encargue de gestionar los eventos (crear y borrarlos), gestionar los usuarios (borrarlos), crear las apuestas para los eventos, etc ... Esta decisión se ha tomado de esta manera porque en la especificación tampoco pone nada acerca de ello.
- En el modelo de clases, la clase evento tiene una relación de uno a muchos con la clase Resultado. Nosotros hemos hecho una relación de uno a uno, porque la clase Resultado correspondería a la la clase Resultado del componente. Un evento sólo puede tener un resultado posible. La clase ResultadosEvento tiene una lista de cada evento por separado (Resultado devuelve todos los resultados de todos los eventos que hay en el archivo XML) de forma que cuando enviamos un objeto ResultadosEvento a un evento, sólo contendrá los resultados de un evento concreto. Por tanto en la clase Evento no necesitaremos crear un nuevo Resultado ya que cuando comprobemos que el ResultadosEvento corresponde al evento actual, simplemente se lo pasamos al cierre de apuestas.
- Como las apuestas múltiples son de varios eventos, entonces el evento no puede saber qué apuestas son ganadoras. Lo que hacemos es que cada vez que cargamos nuevos resultados se comprueban las apuestas múltiples de los usuarios a los que atañe esos resultados para ver si tiene todas esas apuestas que compone la múltiple ya cerradas y si han resultado ganadoras. Por tanto no introducimos una apuesta múltiple en las apuestas realizadas sobre las ofertadas de un evento.
- Hay una parte que no entendimos muy bien y decidimos aplicar la lógica. Como hemos dicho anteriormente el diagrama de clases pone que la apuesta ofrecida tiene una cuota, pero la práctica pide que sea el evento que lo tenga. Luego nos pone que la ganancia neta se obtiene multiplicando por esa cuota, por lo que no le encontramos sentido ya que la cuota debe de ser algo fijo al evento. El método calcular de la cuota al final lo modificamos para que se le pase la ganancia del pronóstico (así no tiene un valor fijo, sino que depende de la apuesta que estemos realizando) y la cantidad apostada para que nos devuelva la ganancia. La clase cuota por lo tanto sólo nos sirve para indicarnos cómo se realizaría ese cálculo, y no para multiplicar por esta.
- La ganancia se calcula como un real aleatorio entre 1 y 4.
- El evento tiene un atributo cerrado o abierto que nos ayudará mucho a la hora de cargar resultados. Como los eventos forman parte de una lista de oyentes que reciben las actualizaciones de nuevos resultados, si cerramos un evento lo quitamos de esa lista. Por tanto ese evento no recibirá los nuevos resultados nunca y las apuestas ofrecidas tampoco y evitamos recorridos que no nos hacen falta y no tenemos que comprobar si las apuestas están o no cerradas.
- A la hora de calcular los impuestos, no podemos saber cuáles están pagados ya porque no hay

opción para ello. Lo que hacemos, es dependiendo de la legislación, calcular los impuestos sobre los ingresos que se han realizado el último año desde la fecha actual. Por tanto, según el día que le demos al botón de ver impuestos, podremos ver una cosa u otra. Otra aclaración es que los ingresos que hacemos manualmente para poder realizar las apuestas y no las que nos hacen por ganarlas, también cuentan para los impuestos. Como ya lo teníamos implementado para que se metieran a la lista de transacciones, hemos preferido dejarlo así.

- A la hora de cobrar el dinero, el controlador es el encargado de calcular las tasas y devolver la cantidad que realmente cobraríamos.
- La parte de cobrar mensualmente no la hemos hecho porque no hay manera posible de que la aplicación sea capaz de descontar por sí sola cada mes el dinero (cobrarlo). Podríamos haber puesto un botón no funcional, pero lo hemos visto innecesario.
- Como hemos dicho anteriormente el evento se puede cerrar por lo que es él el que se encarga de que no se vuelvan a iterar sobre las apuestas. Los atributos que tenemos de las apuestas para saber si son cerradas, finalizadas, etc.. son para información propia.
- La funcionalidad borrar evento está implementada pero esta no hace que se borren las apuestas asociadas por lo que si lo borramos, entonces la aplicación nos dará error. No lo hemos hecho porque no es nada que pidan en la práctica, pero la forma de hacerlo sería un borrado en cascada.
- Los eventos una vez que se cierran tampoco se borran. Se ocultan ante los usuarios normales para que no puedan realizar las apuestas pero siguen perteneciendo a la base de datos. Desde la parte de administración siguen siendo visibles y se le pueden añadir apuestas aunque no se vayan a visualizar.
- Los eventos y el saldo no se actualizan automáticamente. Por ejemplo si se recibe un ingreso de ganar una apuesta, hay que pulsar el botón actualizar para que se refleje. Si un evento recibe una nueva carga de resultados y se cierra, cuando volvemos a la ventana de ver los eventos también tenemos que actualizar o ese evento seguirá apareciendo aunque si realizamos apuestas sobre el no tendrán sentido puesto que no volverán a haber resultados para ese evento.

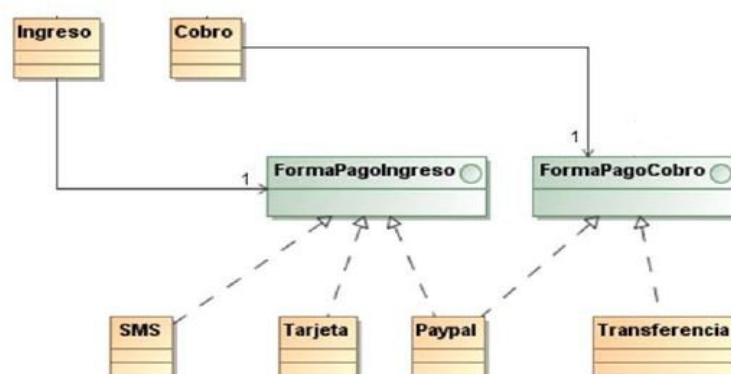
## 4. Patrones de diseño

En este apartado vamos a enumerar, al igual que en el anterior, los patrones que hemos usado o que pensamos que deberían de usarse pero por falta de tiempo o por tener ya mucha implementación hecha y no volver sobre lo hecho hemos aplicado:

- Patrón Singleton
  - Catálogos
  - Controlador
  - Componente
- Patrón Estrategia
  - Cuotas
  - Impuestos
  - Apuestas (Múltiples y Ofrecidas)
- Patrón Proxy
  - Instancias autoreferenciadas. Se produce un bucle al intentar obtener apuesta realizada y apuesta ofrecida de la base de datos (ya que apuestas realizadas es de una apuesta ofrecida y esa apuesta ofrecida contiene esa apuesta realizada)
  - Cuando se crean las apuestas por defecto, estamos creando el evento también. Las apuestas por defecto necesitan al evento y el evento las apuestas. Autoreferencia.
- Patrón Observer
  - Componente
  - Swing
- Patrón Adaptador
  - Persistencia con adaptador DAO
  - Sistema de pagos
- Patrón Composite
  - Apuestas múltiples
- Factoría Abstracta/Método Factoría
  - Base de datos (no conoce los objetos que va a tener dentro)
  - Swing

### Patrón Adaptador

Para independizar nuestra aplicación de servicios externos como pueden ser un gestor de pagos (Paypal, Transferencia bancaria o a través de SMS) se podría usar el patrón Adaptador, el cual nos proporciona, a través de una interfaz, un conjunto de funciones que serán iguales para cualquier gestor de pagos que se desee añadir, solamente se tendrá que implementar una clase adaptadora que realice las funciones de la interfaz con el nuevo gestor:



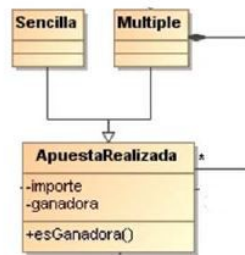
En nuestro caso tenemos, por ejemplo, la clase Ingreso, que realiza un ingreso a nuestra cuenta de ApuestasYa a través del gestor de pagos que tengamos activado. Esta clase tiene un objeto que implementa la interfaz FormaPagoIngreso. Esta interfaz reúne las funciones que necesita la clase Ingreso para conseguir su objetivo, así que, las clases SMS, Tarjeta y Paypal implementan la interfaz para adaptar sus drivers correspondientes a sus necesidades.

Este patrón también lo aplicamos a la hora de acceder a la base de datos, para independizarlo de la aplicación.



## Patrón Composite

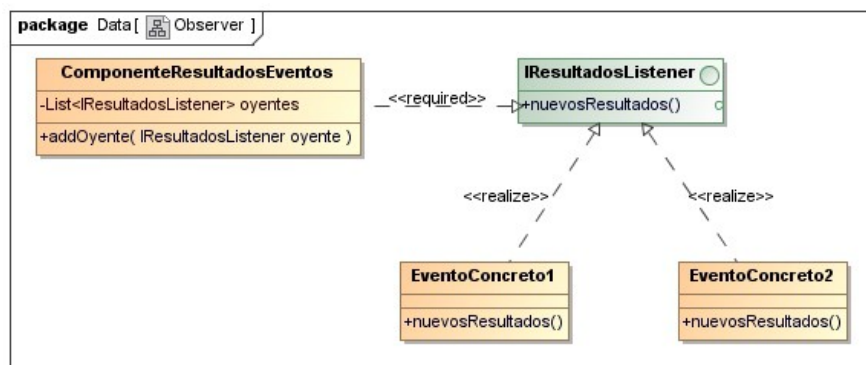
Por el simple hecho de usar Swing para la implementación de la interfaz de usuario, contamos con la utilización del patrón Composite. Este patrón nos da la posibilidad de añadir paneles dentro de otro panel, es decir, nos permite crear un objeto compuesto a partir de otros objetos (simples o compuestos). Este patrón también se da en las apuestas múltiples de nuestra aplicación:



Como podemos ver, una apuesta Multiple está formada por ApuestaRealizada, es decir, estará compuesta por apuestas sencillas u otras apuestas múltiples.

## Patrón Observer

Para la implementación del componente Resultado dado por los profesores, vamos a usar el patrón Observer. Este patrón se usa cuando queremos que muchas instancias de una clase puedan actuar ante un cambio. Esto se consigue haciendo que estas instancias implementen una interfaz con la función que se desea que realicen las instancias cuando sean avisadas del cambio. Por tanto, tendremos otra clase que será la encargada de avisar a las instancias oyentes:



La clase ComponenteResultadosEventos es la encargada de almacenar los oyentes que implementan la interfaz IResultadosListener y de ejecutar el método nuevosResultados() de la interfaz sobre los oyentes que en nuestro caso son los Eventos que esperan la llegada de su Resultado.

## Patrón Proxy

Este patrón se podría haber aplicado para los problemas anteriormente descritos pero al final se han solucionado de manera distinta. Tendríamos un objeto proxy que se encargaría de inicializar correctamente los objetos, como si de un intermediario se tratara.

## Patrón Singleton

Algunos objetos sólo deben de ser instanciados una vez y para ello podemos aplicar este patrón. En nuestra aplicación estos son el catálogo de usuarios, el catálogo de eventos, el controlador y el componente de los resultados

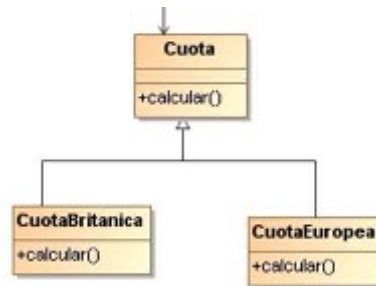
## Carga perezosa

Una posible mejora para optimizar la velocidad a la hora de cargar los datos de la base de datos cuando ejecutamos la aplicación, sería realizar la carga perezosa de los objetos. Esta carga se realizaría, a la hora de necesitar un objeto, creando el objeto en el justo momento de su utilización, en vez de crear todos los objetos al comienzo de la aplicación. En nuestro caso vemos que la aplicación tarda bastante en ejecutarse puesto que carga todos los usuarios, eventos, apuestas, ... y no sólo los necesarios.

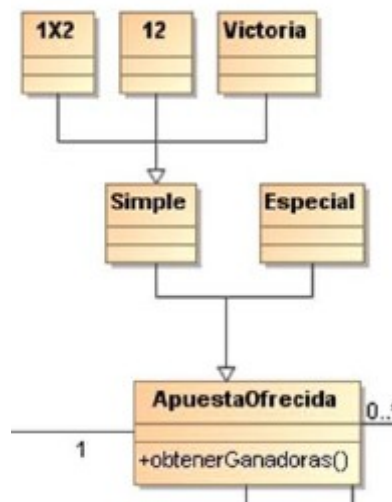
## Patrón estrategia

En muchos casos nos hemos encontrado una jerarquía de herencia en las cuales los objetos de la misma deben de realizar el mismo método de manera diferente. Para ello aplicamos el patrón estrategia, de tal forma que un objeto mismo sabe cómo debe de realizar su método.

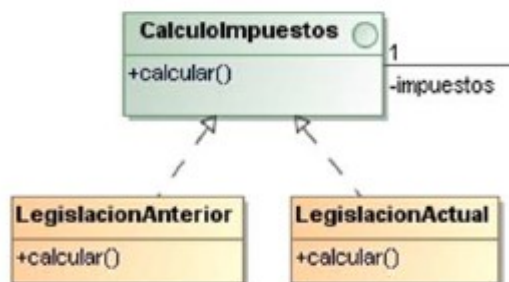
Por ejemplo dependiendo de la cuota debemos de calcularla de una manera u otra, pero no podemos hacer un análisis de caso sino simplemente llamar a su propio método calcular.



De forma análoga nos ocurre en las apuestas, a la hora de obtener las ganadoras dependiendo del tipo de apuesta será de una manera u otra.



El método `obtenerGanadoras()` se ha implementado para cada una de las distintas apuestas. Por último en las legislaciones se ha hecho también. Dependiendo del tipo que tengamos, podemos calcular los impuestos de una manera u otra.



Con este patrón podemos tener nuevos tipos de apuestas o cuotas, y no habría que modificar código sino simplemente generar nuevas formas de cálculo y se adaptaría perfectamente.

## Patrones en Swing

Swing usa como patrón de diseño el patrón modelo vista controlador. Además de ello usa el patrón Observer explicado anteriormente para capturar los eventos que ocurren en botones por ejemplo, y el patrón Composite en sus componentes (dentro de un panel podemos colocar otro panel). El patrón builder quizá también lo use a la hora de instanciar los diferentes componentes. Si quieres tener una ventana entera primero creas cada uno de los componentes y al final la ventana con ellos. Por último el

patrón factoría abstracta es el que se utiliza para crear los diferentes tipos de componentes para los distintos sistemas operativos.

### **Otros patrones**

En la práctica intentamos respetar los patrones GRASP (no violar el experto, que tenga bajo acoplamiento y alta cohesión, ...) y también la separación modelo vista mediante la implementación de un controlador.

## 5. Componentes

En este apartado vamos a explicar la funcionalidad que tiene nuestro componente luz y nuestro componente resultado y los pasos que sigue hasta calcular los ganadores.

### Componente Luz

El componente Luz ha sido implementado con los pasos que se muestran en la guía práctica de la asignatura. Se ha empaquetado e importado a nuestro proyecto para ser usado en la interfaz gráfica. Este componente aparece en el apartado del administrador donde podemos pulsarlo y aparece una nueva ventana donde elegir el fichero de resultados que se desea cargar. Mientras que estamos eligiendo el fichero, el botón Luz aparece como que se está pulsando y una vez elegido el fichero entonces aparece encendido indicando que el fichero ha sido cargado. Si volvemos a pulsar en él se apaga sin ninguna acción.

### Componente Resultado

Este componente no es como el anterior y no ha sido empaquetado ya que necesitamos recorrer toda su estructura para poder obtener los datos de los resultados.

Cuando pulsamos el botón Luz y elegimos un fichero, accedemos a la clase `ComponenteResultadosEventos` que es el encargado de almacenar la lista de oyentes. Esta lista almacena los Eventos que están pendientes de obtener un resultado, de esta forma cuando cargamos un archivo xml se llama a `asignarArchivo` con el nombre del fichero y se crea un objeto `Resultado` devuelto por la clase `CargadorResultados` dada por los profesores. Con este objeto `Resultado` se crea el objeto `ResultadosEventos`. La diferencia entre estas clases es que un `Resultado` contiene los resultados de muchos eventos distintos y de distintos deportes, mientras que `ResultadosEventos` contiene una lista de `Resultado` individuales, es decir, cada `Resultado` de la lista de `ResultadosEvento` tiene solamente un resultado de un único evento. De esta forma conseguimos que cuando le pasamos a los oyentes (eventos) el objeto `ResultadosEvento`, cada `Evento` puede recorrer la lista y encontrar su `Resultado` correspondiente.

Una vez que el `Evento` ha encontrado su resultado correspondiente en el método `nuevosResultados` (dado por la interfaz `IResultadosListener`, interfaz que gestiona los Eventos oyentes), envía el `Resultado` concreto a sus `ApuestasOfrecidas`. Cada tipo de `ApuestaOfrecida` buscará su pronóstico ganador en el resultado del evento, por ejemplo, si es una `ApuestaEspecial` que pregunta ¿Quién será el primer goleador? Esta apuesta calculará, dado el resultado del evento, quien ha sido el primer jugador en marcar un gol, entonces llamará al método `seleccionarGanadores` con el pronostico ganador. Éste método recorrerá toda la lista de `ApuestasRealizadas` sobre esa `ApuestaOfrecida` y marcará como ganadoras todas aquellas apuestas realizadas que contengan el pronóstico ganador. Recordamos que gracias a aplicar el patrón estrategia no tendremos que conocer que tipo de apuesta es, puesto que cada uno ya se encarga de calcular sus ganadores de la manera más adecuada.

## 6. Pruebas unitarias

Las clases sobre las que se han realizado pruebas unitarias son Evento y Usuario.

### Evento

Primero se ha creado una función `setUp()` con la etiqueta `@Before` para la inicialización de un objeto Evento. Las operaciones a las que se les va a realizar las pruebas son operaciones que no son de modificación u obtención, ya que no tiene sentido hacer las pruebas a operación que modifican directamente los atributos de la clase. A continuación se encuentran las operaciones de prueba:

- `testObtenerDeporte()`: Esta función realiza el test sobre la función `obtenerDeporte(String deporte)`. Su objetivo es modificar el atributo `Deporte` a través del String pasado como parámetro. Para ello ejecutamos la función sobre el evento creado en la inicialización con los strings "DeporteFutbol", "DeporteTenis" y "DeporteFormula1" y comprobamos que el atributo `deporte` de `Evento` ha sido modificado comprobando que es la clase que deseábamos con `assertEquals`.
- `testObtenerCuota()`: Al igual que con la operación anterior, se desea que se modifique el atributo `Cuota` ejecutando la función `obtenerCuota(String cuota)`. Probamos con las cadenas "CuotaEuropea" y "CuotaBritánica", e igual que antes, comprobamos con `assertEquals` que las clases sean las que esperábamos.
- `testRecuperarApuesta()`: Este método comprueba que el método `recuperarApuesta(int codApuesta)` devuelva la apuesta ofrecida correspondiente al código `codApuesta`. Para ello creamos una `ApuestaOfrecida` y le modificamos el código (para saber cuál es). A continuación añadimos la apuesta a la lista de apuestas ofrecidas del evento y, por último, comprobamos con el aserto que la apuesta ofrecida que devuelve el método `registrarApuesta()` es la misma que la que habíamos introducido anteriormente.
- `testModificarApuesta()`: El método `modificarApuesta(int codApuesta, ApuestaOfrecida apuesta)` modifica la apuesta que tiene `codApuesta` por la nueva apuesta. Para comprobar su correcto funcionamiento, creamos una primera `ApuestaOfrecida` igual a la anterior, la añadimos a la lista de apuestas ofrecidas del evento y guardamos la posición de la lista en la que se ha añadido esta apuesta. Creamos una nueva apuesta ofrecida distinta a la anterior y ejecutamos el método. Ahora solo nos queda comprobar que la apuesta ofrecida que hay en la posición de la lista donde se encontraba la primera, es igual que la segunda apuesta que hemos creado.
- `testAddApuesta()`: Esta función comprueba que se ha añadido correctamente una apuesta a la lista de apuestas. Como sabemos que cuando añadimos una apuesta se coloca la última en la lista, comprobamos que efectivamente la apuesta que hay en la última posición es igual a la que hemos creado anteriormente.
- `testNuevosResultados()`: Con este método de prueba se quiere comprobar que los resultados han sido cargados correctamente. Para ello creamos un objeto `ResultadosEvento` con un `Resultado` y le añadimos un `ResultadoFutbol` (por ejemplo). También tenemos que añadir al evento los participantes ya que no se los hemos añadido en la inicialización. Ejecutamos el método `nuevosResultados(resultadosEvento)` y compramos que el `Resultado` que guarda el evento es el mismo que el resultado futbol que hemos creado.

### Usuario

Al igual que para un Evento, para usuario creamos la clase `UsuarioTest`, donde probaremos el correcto funcionamiento de todos los métodos (menos getters y setters). Comenzamos creando el método de inicialización `setUp()` con la etiqueta `@Before` donde creamos un nuevo Usuario. La explicación de los métodos es la siguiente:

- `testComprobarIngreso()`: Esta función comprueba que `comprobarIngreso(String formaIngreso)` modifique el atributo `ingresoPorDefecto` en la clase que se desea dependiendo de la cadena de entrada ("Tarjeta", "SMS" o "Paypal"). Por lo tanto ejecutamos el método y comprobamos que el atributo es de la misma clase que la que queríamos.
- `testComprobarCobro()`: Igual que la función anterior pero para modificar el atributo `comprobarCobroPorDefecto` ("Transferencia" o "Paypal").

- `testAddTransaccion()`: Queremos comprobar que la transacción se ha añadido correctamente a la lista de transacciones de un usuario. Para ello creamos una nueva transacción y la añadimos. Por último obtenemos la transacción con la posición donde hemos añadido la transacción y la comparamos con la transacción creada anteriormente.
- `testAddApuesta()`: Igual que la función anterior pero con las apuestas. Creamos una nueva apuesta con una apuesta ofrecida y su pronóstico, la añadimos a la lista de apuestas realizadas y comprobamos que la última apuesta realizada de la lista es la misma que la que hemos creado anteriormente.

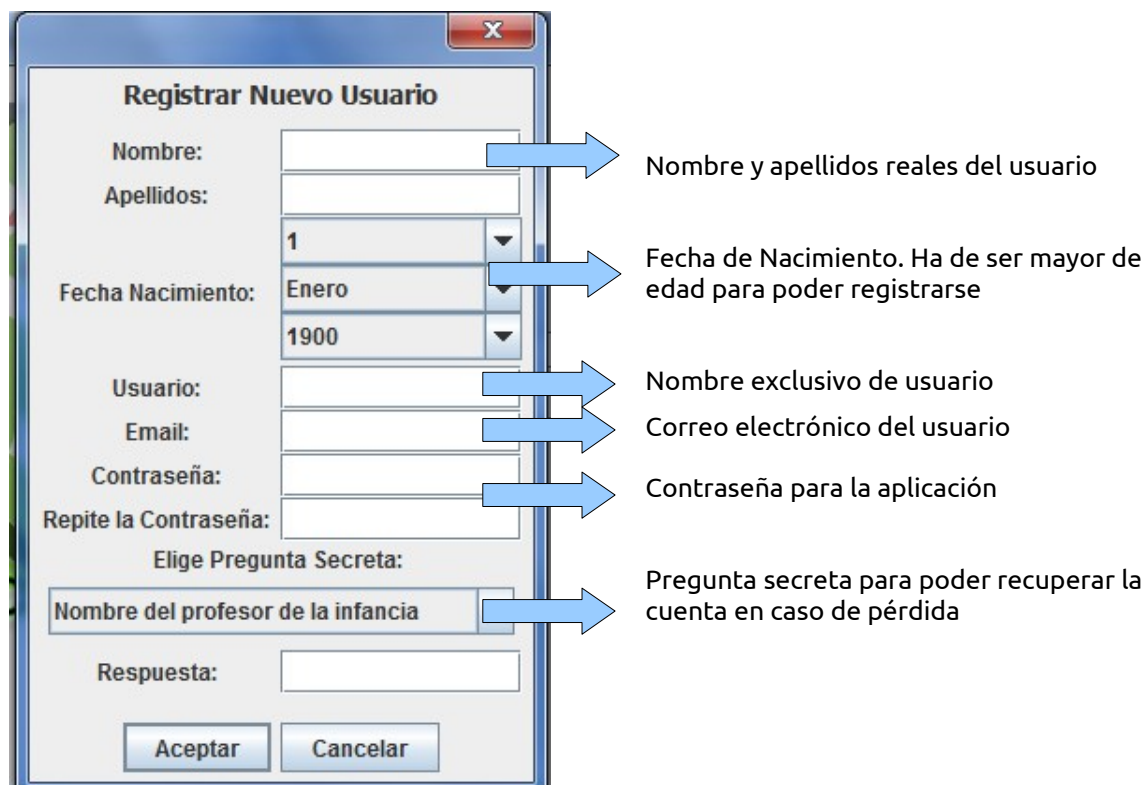
## 7. Manual de usuario

### 7.1. Registro de un nuevo usuario

Si es la primera vez que inicias la aplicación, deberás de registrarte antes de poder hacer nada. Para ello, desde la ventana de identificación hay que pinchar en el botón de "registrarse".



Acto seguido se abrirá otra ventana donde tendremos que completar correctamente cada uno de los campos que se nos solicitan:



Field	Description
Nombre:	Nombre y apellidos reales del usuario
Apellidos:	
Fecha Nacimiento:	Fecha de Nacimiento. Ha de ser mayor de edad para poder registrarse
Usuario:	Nombre exclusivo de usuario
Email:	Correo electrónico del usuario
Contraseña:	Contraseña para la aplicación
Repite la Contraseña:	
Elige Pregunta Secreta:	Pregunta secreta para poder recuperar la cuenta en caso de pérdida
Resposta:	

Si los datos que se introducen son correctos (mayoría de edad, nombre de usuario que no esté ya registrado, campos completos, contraseñas coincidentes..) entonces el usuario será registrado exitosamente (mostrará un mensaje) y volverá a la pantalla inicial.

Ahora podemos meter nuestro usuario y contraseña registrados y si todo es correcto se nos abrirá la pantalla principal siguiente:

Bienvenido a la aplicación de ApuestaYA! Tony Wang

Administrar Otros

Perfil Eventos Mis Apuestas Transacciones

**DATOS PERSONALES**

Usuario: lion

Nombre: Tony

Apellidos: Wang

Email: tony@um.es

Fecha de nacimiento: 28/05/1990

Contraseña:

Cambiar Datos Cambiar Contraseña

Actualizar datos

**DATOS DE INGRESO Y COBRO**

Ingreso por defecto: Paypal Modificar

Cobro por defecto: Paypal Modificar

Cantidad de ingreso/cobro:

Ingresar Cobrar

Ver Impuestos Actualizar

Saldo actual: 0 €

## 7.2. Sección de administradores

Desde la ventana principal, y siendo cualquier usuario (no se contempla por ahora que hayan usuarios especiales que son administradores), pulsamos en la esquina superior izquierda la opción de administrar y entramos como administrador. Está será la ventana que se abrirá:

Ventana de Administración de ApuestaYA!

Crear Eventos Administrar Eventos Administrar Usuarios Cargador Resultados

**Menu de Inserción de Nuevo Evento**

Nombre del Evento:

Deporte:   
☐ Futbol   
☐ Tenis   
☐ Formula 1

Fecha (dd/MM/aaaa):

Hora (HH:mm):

Cuota: Cuota Europea

**Jugadores de Tenis:**

Jugador 1:   
Jugador 2:

**Jugadores de Futbol:**

Equipo 1:   
E1-J1:   
E1-J2:   
E1-J3:   
E1-J4:   
E1-J5:   
E1-J6:   
E1-J7:   
E1-J8:   
E1-J9:   
E1-J10:   
E1-J11:   
Equipo 2:   
E2-J1:   
E2-J2:   
E2-J3:   
E2-J4:   
E2-J5:   
E2-J6:   
E2-J7:   
E2-J8:   
E2-J9:   
E2-J10:   
E2-J11:

**Corredores de F1:**

Corredor 1:   
Corredor 2:   
Corredor 3:   
Corredor 4:   
Corredor 5:   
Corredor 6:   
Corredor 7:   
Corredor 8:   
Corredor 9:   
Corredor 10:   
Corredor 11:   
Corredor 12:   
Corredor 13:   
Corredor 14:   
Corredor 15:   
Corredor 16:   
Corredor 17:   
Corredor 18:   
Corredor 19:   
Corredor 20:   
Corredor 21:   
Corredor 22:   
Corredor 23:   
Corredor 24:

Registrar Evento

Volver



### 7.2.1. Crear un nuevo evento

Vemos que en la ventana de administrador tenemos tres pestañas. Tendremos que pinchar encima de crear eventos (es la que se muestra nada más entrar como administrador). Una vez ahí tendremos que rellenar los correspondientes campos. No podremos dejar ninguno vacío o no se nos creará correctamente el evento. En la parte derecha, vemos que tenemos tres columnas donde no se pueden rellenar los datos; para que sean accesibles tenemos que seleccionar uno de los tres deportes que se nos ofertan.

Si todo se rellena correctamente nos mostrará un mensaje de evento registrado con éxito y se vaciarán las casillas automáticamente por si queremos registrar otro evento.

### 7.2.2. Crear una nueva apuesta

Cuando hayamos terminado de crear los correspondientes eventos podremos crear apuestas para ellos. Los eventos al crearse siempre llevan apuestas por defecto dependiendo del deporte que se celebre. Es decir, si es un partido de fútbol ya tendrá como apuestas que el resultado sea 1, X o 2. Si es de tenis o fórmula 1 tendría como apuesta que el resultado sea un único ganador correspondiente al participante. Dicho esto, las apuestas que se crean desde la zona de administrador son las "especiales". Dichas apuestas son del tipo "¿Quién marcará antes un gol?", "¿Cuántos sets durará el partido?", etc..

Para hacerlo, pinchamos en la segunda pestaña "Administrar eventos"

Ventana de Administración de ApuestaYA!

Crear Eventos | Administrar Eventos | Administrar Usuarios

60-Roland Garros - Federer VS Nadal

**Crear Apuesta Especial**

**Apuestas de Tenis**

Pregunta Especial:

¿Cuántos sets durará el partido?

Posibles Pronósticos:

3

**Apuestas de Fútbol**

Pregunta Especial:

¿Quién será el primer goleador?

Posibles Pronósticos:

**Apuestas de F1**

Pregunta Especial:

¿Quién será el primero en entrar a boxes?

Posibles Pronósticos:

Actualizar Eventos | Eliminar Seleccionado | Crear Apuesta | Volver

A la izquierda tendremos toda la lista de eventos del que disponemos. Si hemos creado alguno y no nos aparece podremos actualizar la lista pinchando en "Actualizar Eventos". Desde esta ventana también tendremos la opción de eliminar eventos.

### 7.2.3. Gestionar usuarios

Desde esta ventana podremos ver los usuarios que tenemos registrados en nuestra aplicación y borrarlos seleccionándolos en la lista y pulsando el botón borrar.

Ventana de Administración de ApuestaYA!

Crear Eventos | Administrar Eventos | **Administrar Usuarios** | Cargar Resultados

**Relacion de Usuarios Registrados**

Codigo de usuario	Nombre Usuario	Nombre	Apellidos	Email	Fecha Nacimiento
1	lion	Tony	Wang	tony@um.es	28/05/1990

Eliminar Seleccionado

Volver

No se han implementado más métodos para la gestión de usuarios.

### 7.2.4. Cargador de Resultados

Esta sección sólo contará con un botón el cual podremos pulsar. Una vez que lo pulsemos se nos abrirá una ventana típica para seleccionar archivo. Podremos coger el archivo XML donde van los resultados para cargarlos en nuestra aplicación. Una vez hecho esto el sistema se encargará de cerrar los eventos y las apuestas relacionadas a ese archivo. No se pone ninguna imagen adjunta puesto que es muy sencillo de manejar.

## 7.3. Sección de usuario

### 7.3.1. Perfil, ingresos y cobros

Volvemos a poner la imagen de la ventana principal para que nos sea más sencillo entender el manual:

Bienvenido a la aplicación de ApuestaYA! Tony Wang

Administrar Otros

Perfil Eventos Mis Apuestas Transacciones

**DATOS PERSONALES**

Usuario: lion

Nombre: Tony

Apellidos: Wang

Email: tony@um.es

Fecha de nacimiento: 28/05/1990

Contraseña:

Cambiar Datos Cambiar Contraseña

Actualizar datos

**DATOS DE INGRESO Y COBRO**

Ingreso por defecto: Paypal Modificar

Cobro por defecto: Paypal Modificar

Cantidad de ingreso/cobro:

Ingresar Cobrar

Ver Impuestos Actualizar

Saldo actual: 0 €

Para cambiar los datos personales o contraseña tan solo deberemos de darle al botón correspondiente. En caso de los datos, al pulsar se nos pondrán las casillas editables. Cuando terminemos la edición debemos de introducir por seguridad la contraseña actual para poder llevar a cabo los cambios. Si la contraseña es correcta, al pulsar actualizar datos se nos habrá cambiado.

En caso de cambiar la contraseña, se nos abrirá una ventana en la cual deberemos introducir la nueva contraseña y la anterior para confirmar el cambio.

Cambiar de contraseña

Contraseña actual:

Nueva contraseña:

Repite la nueva contraseña:

OK Cancelar

Aunque se trate de un sistema externo, la aplicación permite cambiar el tipo de ingreso por defecto y cobro por defecto. Tan solo debemos de pulsar el botón correspondiente y se nos abrirá otra ventana donde podremos cambiarlo.

Dependiendo del tipo de cambio que queramos hacer, nos pedirán los datos necesarios. Una vez introducidos de manera correcta en la ventana principal de usuario veremos que la opción ha cambiado. En las imágenes siguientes vemos el cambio:

Opciones de Ingreso

Paypal Credito SMS

Email

Contraseña

Guardar

Cancelar

Opciones de Ingreso

Paypal Credito SMS

Ingrese su número de teléfono

620381613

Guardar

Cancelar

Bienvenido a la aplicación de ApuestaYA! Tony Wang

Administrar Otros

Perfil Eventos Mis Apuestas Transacciones

**DATOS PERSONALES**

Usuario

Nombre

Apellidos

Email

Fecha de nacimiento

Contraseña

Cambiar Datos Cambiar Contraseña

Actualizar datos

**DATOS DE INGRESO Y COBRO**

Ingreso por defecto

SMS Modificar

Cobro por defecto

Transferencia Modificar

Cantidad de ingreso/cobro

Ingresar Cobrar

Ver Impuestos Actualizar

Saldo actual: 1000 €

También hemos cambiado el tipo de cobro a transferencia introduciendo los 20 dígitos de la cuenta bancaria. Si queremos ingresar dinero para poder realizar apuestas tan solo tenemos que introducir la cantidad en la cajetilla y darle a ingresar. Si tenemos dinero en la cuenta, podremos cobrarlo haciéndolo de la misma forma. Todo estos métodos son robustos y comprueba que se puedan realizar. El botón actualizar sirve para refrescar el saldo, por si hemos ganado alguna apuesta y no se ve reflejado. Por último podemos ver los impuestos que tenemos que pagar desde la fecha actual hasta las apuestas de hace un año como máximo.

Ver Impuestos

Este año tiene que pagar 210€

Aceptar

### 7.3.2. Realizar apuestas

Si le damos a la ventana de Eventos podremos realizar apuestas. Para ello tienen que haber eventos abiertos. Hemos creado uno de prueba y como es reciente tendremos que darle al botón de actualizar para visualizarlo.

The screenshot shows a web application window titled 'Bienvenido a la aplicación de ApuestaYA! Tony Wang'. It has a navigation bar with 'Administrar' and 'Otros'. Below it are tabs for 'Perfil', 'Eventos', 'Mis Apuestas', and 'Transacciones'. The 'Eventos' tab is active, showing a list of events on the left with '30-Roland Garros' selected. On the right, the details for 'Roland Garros' are displayed: 'Tipo de Evento: Partido de Tennis', 'Fecha de Celebración: 09/09/2013', 'Hora de Celebración: 17:00', and 'Cuota: Europea'. There is a button 'Ver Apuestas Disponibles' and a section 'Filtrar Eventos' with radio buttons for 'Todos' (selected), 'Fútbol', 'Tennis', and 'Fórmula 1'. At the bottom is an 'Actualizar Eventos' button.

Si pinchamos sobre cualquier evento de la lista, a la derecha podremos ver los datos del mismo y darle a realizar apuestas. Si queremos encontrar solo eventos de un determinado deporte podremos filtrarlos con la opción que hay disponible. Al darle a ver apuestas disponibles se nos abrirá:

The screenshot shows a window titled 'Apuestas para Roland Garros'. It has two main sections: 'Selecciona Pronóstico' and 'Información'. In the 'Selecciona Pronóstico' section, there are two dropdown menus. The first shows '25-¿Qué tenista se alzará con la victoria?' and the second shows '19-Pronostico: Nico almagro'. In the 'Información' section, there is a 'Ganancia con Pronóstico' field showing '1,32' and a 'Cantidad a Apostar' field which is empty. At the bottom is a 'Realizar Apuesta' button.

En esta ventana podremos elegir en la parte superior la apuesta que queremos realizar con su correspondiente pronóstico. Cada pronóstico tendrá una ganancia (por euro apostado). Ponemos la cantidad que queremos apostar, en nuestro caso pondremos 10 euros y le damos a realizar apuesta.

### 7.3.3. Información de apuestas y transacciones. Apuestas múltiples

Si pinchamos en la pestaña de Mis Apuestas podremos ver la relación de apuestas sencillas que tenemos hechas con sus respectivos datos:

Cod...	Evento	Fecha	Pronostico elegido	Cant...	Gan...	Finalizada	Ganadora
40	Roland Garros	09/09/2013 ...	Ganara el Tenista: Nico almagro...	10.0 €	1,32	No	No

Para realizar una apuesta múltiple tendremos que darle al botón que lo indica y tendremos una ventana como la siguiente:

Evento	Pronostico elegido
Roland Garros	Ganara el Tenista: Nico almagro

Evento	Pronostico elegido
Roland Garros	Ganara el Tenista: Nico almagro
Roland Garros	Ganara el Tenista: Nico almagro
Roland Garros	Ganara el Tenista: Nico almagro
Roland Garros	Ganara el Tenista: Nico almagro

Tipo de Apuesta: ☐ Apuesta Combinada ☒ Apuesta Sistema

Cantidad:

Combinacion Sistema:  /

A la izquierda tendremos las apuestas aún abiertas que tenemos y que podemos usar para realizar una apuesta múltiple y a la derecha las apuestas que estamos seleccionando para realizar la múltiple. Podremos borrarlas usando los botones del centro. Tendremos que elegir si queremos que sea una apuesta combinada o una sistema con la combinación.

Si todo está correcto entonces la crearemos y podremos verlo en la ventana de ver apuestas múltiples que se muestra en la siguiente página.

Apuestas múltiples de Tony

### Relación de Apuestas Múltiples

Apuestas (Códigos)	Tipo	Cantidad	Pronostico	Finalizada	Ganadora
40, 40, 40, 40	Sistema	5.0	3/4	No	No

Cerrar

Por último tendremos en la pestaña de transacciones todo lo correspondiente con ingresos y cobros que se han realizado en la cuenta.



## 8. Observaciones

Tenemos aquí una tabla resumen del tiempo que se ha empleado para realizar la práctica aproximadamente:

Tipo de trabajo	Tiempo dedicado		
	Día	Hora	Minutos Totales
Diagrama de clases	27/10/12	13:00 a 15:00	120
	20/10/12	17:00 a 17:40	40
	25/10/12	09:00 a 09:30	30
	28/10/12	17:30 a 18:00	30
Elaborado de la memoria	20/10/12	16:30 a 17:00	30
	30/10/12	10:30 a 11:00	30
	24/11/12	19:30 a 20:00	150
	27/01/13	11:00 a 12:50	110
	12/08/13	16:00 a 18:30	150
	19/08/13	16:00 a 19:30	210
	20/08/13	10:00 a 11:20	80
	21/08/13	18:00 a 19:30	90
	22/08/13	16:00 a 19:00	180
Interfaz (y algo de implementación)	23/11/12	19:00 a 23:00	240
	25/11/12	17:00 a 18:00	60
	07/12/12	12:40 a 14:10	90
	07/12/12	15:30 a 21:00	330
	08/12/12	16:00 a 17:30	90
	08/12/12	17:30 a 2:00	510
	09/12/12	10:00 a 13:30	210
	15/01/13	11:30 a 13:30	120
Persistencia (y algo de implementación)	08/12/12	17:30-20:50	200
	09/12/12	15:00 a 19:30	270
	05/01/13	21:00 a 00:30	210
	06/01/13	17:00 a 18:00	60
	15/01/13	11:30 a 13:30	120
	15/01/13	15:30 a 18:00	150
	16/01/13	19:15 a 20:45	90
	16/01/13	22:00 a 00:40	160
	17/01/13	19:00 a 20:30	90
	17/01/13	22:30 a 0:30	120
	18/01/13	11:00 a 13:00	120
	18/01/13	15:45 a 20:00	225
	24/01/13	21:20 a 0:00	160
	25/01/13	16:15 a 20:45	270



	26/01/13	16:45 a 18:00	75
	12/08/13	16:00 a 20:00	240
	15/08/13	16:00 a 20:00	240
	21/08/13	16:30 a 18:00	90
	21/08/13	22:00 a 2:30	270
Componente e Implementación	05/08/13	16:00 a 19.45	225
	06/08/13	9:30 a 13:30	240
	06/08/13	16:00 a 19:00	180
	10/08/13	15:00 a 17:00	120
	19/08/13	16:00 a 19:30	210
	20/08/13	16:00 a 20:00	240
	20/08/13	22:00 a 1:00	180
Pruebas Unitarias	07/08/13	10:00 a 13:40	220
	07/08/13	16:00 a 18:30	150

La práctica se ha realizado en un total de 130 horas aproximadamente. Pensamos que hay muchas partes en la especificación que no terminan de quedar claras (no obstante, siempre tenemos ayuda del profesor). Además, la práctica trata de un ámbito en el que los alumnos por norma general no está familiarizado por lo que a veces cuesta más entenderlo que implementarlo. Por último, la carga de trabajo que tiene esta práctica también es muy grande.