UOC Universitat Oberta de Catalunya

Trabajo Final de Carrera
Microsoft .NET

SAAP
Sistema Automático de Apuestas de Póker

MEMORIA

Alumno: José María García Martínez
Consultor: Juan Carlos González Martín

Índice

1- Definición del proyecto

Betfair (http://www.betfair.com) es el mayor mercado de apuestas que existe hoy en día en Internet. Cada día se mueven cientos de millones de euros en transacciones de apuestas.

A diferencia de las tradicionales casas de apuestas en las que el usuario juega en contra de la casa, Betfair no juega contra el usuario, sino que él meramente se limita a ser un mero intermediario entre apostantes (por eso se habla de mercado de apuestas y no casa de apuestas). El beneficio de Betfair lo obtiene mediante el cobro de una pequeña comisión a los ganadores, por lo que le da igual que jugador gane, tan sólo le interesa que juego el mayor número de usuarios posible, puesto que así ganará más dinero. Esto quiere decir que Betfair no se dedica, como podría pasar en otras casas de apuestas, a manipular sus juegos para su ventaja, porque le da igual quien gane, ella ganará siempre que haya jugadores que jueguen, por lo que se trata de juegos más justos.

Dentro de los diversos mercados y juegos que ofrece Betfair nos vamos a centrar en unos juegos conocidos como X-Games que presentan una particularidad que los hace interesantes de cara a la informática: estos juegos tienen un API que permite que un sistema automático de apuestas diseñado por ordenador esté conectado a ellos continuamente realizando apuestas.

En cuanto a los X-Games nos vamos a centrar en uno en concreto, el poker de tipo Texas Hold'em, una variedad de poker muy jugada en Internet en la actualidad. El juego es muy sencillo, cada juego se compone de 4 jugadores y 4 manos. En la primera mano no se reparten todavía ni carta a los jugadores ni se levantan las de la mesa. En la segunda mano se reparte 2 cartas a cada jugador. En la tercera mano se levantan tres cartas de las 5 disponibles en la mesa. En la cuarta mano se levanta una

carta más de la mesa, quedando sólo una por levantar. El juego termina cuando se levanta la última carta y de ese modo se conoce al ganador.

El juego consiste en que durante las 4 manos se van viendo las cartas de los jugadores y las de la mesa, y se tiene que decir en cada mano cuál será el jugador que ganará. Evidentemente cuanto más tarde hagamos la apuesta la cuota será menor al tener mayor información para saber que jugador puede ganar.

Figura 1 – Juego Texas Hold'em en la página web de BetFair

Mediante este proyecto se creará una aplicación que se conecte a Betfair y realice simulaciones de apuestas de manera continua en uno de los X-Games, el de Texas Hold'em Poker, siguiendo una serie de estrategias predefinidas y mediante las cuales intentará asegurar una ganancia a corto plazo.

2- Objetivos

A nivel de la aplicación:

- Crear una aplicación que permita realizar apuestas automáticas.
- Permitir seleccionar diversas estrategias de juego, las cuales llevarán una pequeña descripción de lo que hacen y cómo funcionan.
- Grabar las partidas jugadas en una base de datos para su posterior análisis.

A nivel de proyecto:

- Demostrar que se puede ganar dinero a corto plazo, aunque sea en pequeñas cantidades, con sistemas automáticos de apuestas diseñados por ordenador.
- Comprender el funcionamiento de los API de los X-Games ofrecidos por Betfair.
- Introducirme en la programación mediante WPS en .NET.
- Aprender el funcionamiento de acceso a base de datos mediante LINQ.

2.1- Objetivos PEC 1 – Definición del proyecto

- Definir correctamente el ámbito del proyecto.
- Cuáles son sus objetivos principales.
- Establecer los hitos a alcanzar en cada PEC.
- Establecer las tecnologías a usar.
- Realizar una planificación inicial.

2.2- Objetivos PEC 2 – Análisis de la aplicación

- Realizar un análisis detallado de la aplicación.
- Crear los casos de uso.
- Definir el modelado de la base de datos.
- Diagramas de caso de uso y secuencia.
- Diagramas de clases.

2.3- Objetivos PEC 3 – Implementación

- Desarrollar la aplicación usando las tecnologías propuestas.
- Crear la base de datos.
- Testear la aplicación.
- Crear la documentación para el usuario.
- Crear la documentación para instalación/configuración.

3- Metodología y herramientas empleadas

Para el análisis y diseño de la aplicación se usarán los siguientes sistemas:

- Planificación de las fases del proyecto
- Recogida de datos y requisitos.
- Casos de uso.
- Diseño de la aplicación a través de UML (Unified Modeling Language) enfocado al desarrollo de una aplicación por objetos.
- Diseño visual de la interfaz gráfica.
- Implementación de la aplicación mediante .NET.

Para llevar a cabo todos estos objetivos se usarán las siguientes herramientas:

- Microsoft Project 2003 para la planificación de las fases y tareas del proyecto.
- Microsoft Visio 2003 para la creación de los diagramas UML.
- Microsoft Word 2003 para la documentación.
- Microsoft PowerPoint 2003 para la generación de las presentaciones.
- Microsoft Visual Studio C# 2010 Express.
- Microsoft SQL Server 2008 R2 Express.

Para poder usar la aplicación será necesario:

- Ordenador PC con sistema operativo Windows XP o superior.
- Net Framework 4.0.
- SQL Server 2008.

4- Licencia

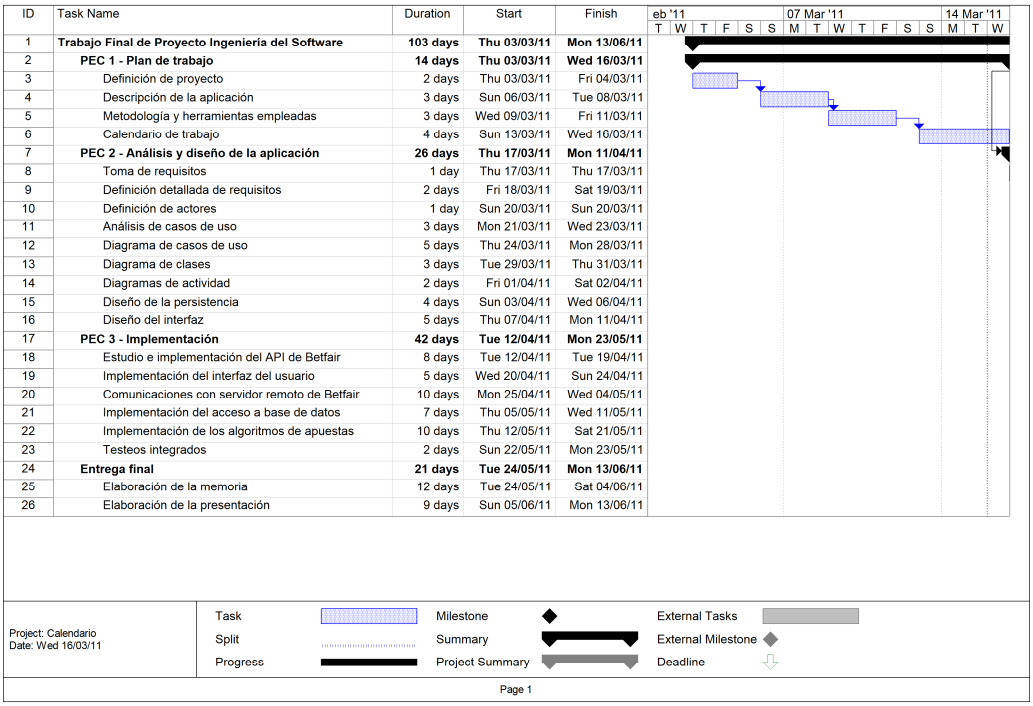
La aplicación se distribuirá gratuitamente a través de una licencia GPL 3 junto con su código fuente que permitirá el uso y modificación de la misma siempre y cuando se redistribuyan también esos cambios.

Se puede encontrar más información acerca de las condiciones de la licencia GPL 3 en la siguiente dirección: http://gplv3.fsf.org/

5- Planificación del proyecto

- *Planificación inicial*

En la siguiente captura de pantalla podemos ver cuál era la planificación inicial prevista para el desarrollo del proyecto al comienzo del mismo en la PEC 1:



ID	Task Name	Duration	Start	Finish
1	**Trabajo Final de Proyecto Ingeniería del Software**	**103 days**	**Thu 03/03/11**	**Mon 13/06/11**
2	**PEC 1 - Plan de trabajo**	**14 days**	**Thu 03/03/11**	**Wed 16/03/11**
3	Definición de proyecto	2 days	Thu 03/03/11	Fri 04/03/11
4	Descripción de la aplicación	3 days	Sun 06/03/11	Tue 08/03/11
5	Metodología y herramientas empleadas	3 days	Wed 09/03/11	Fri 11/03/11
6	Calendario de trabajo	4 days	Sun 13/03/11	Wed 16/03/11
7	**PEC 2 - Análisis y diseño de la aplicación**	**26 days**	**Thu 17/03/11**	**Mon 11/04/11**
8	Toma de requisitos	1 day	Thu 17/03/11	Thu 17/03/11
9	Definición detallada de requisitos	2 days	Fri 18/03/11	Sat 19/03/11
10	Definición de actores	1 day	Sun 20/03/11	Sun 20/03/11
11	Análisis de casos de uso	3 days	Mon 21/03/11	Wed 23/03/11
12	Diagrama de casos de uso	5 days	Thu 24/03/11	Mon 28/03/11
13	Diagrama de clases	3 days	Tue 29/03/11	Thu 31/03/11
14	Diagramas de actividad	2 days	Fri 01/04/11	Sat 02/04/11
15	Diseño de la persistencia	4 days	Sun 03/04/11	Wed 06/04/11
16	Diseño del interfaz	5 days	Thu 07/04/11	Mon 11/04/11
17	**PEC 3 - Implementación**	**42 days**	**Tue 12/04/11**	**Mon 23/05/11**
18	Estudio e implementación del API de Betfair	8 days	Tue 12/04/11	Tue 19/04/11
19	Implementación del interfaz del usuario	5 days	Wed 20/04/11	Sun 24/04/11
20	Comunicaciones con servidor remoto de Betfair	10 days	Mon 25/04/11	Wed 04/05/11
21	Implementación del acceso a base de datos	7 days	Thu 05/05/11	Wed 11/05/11
22	Implementación de los algoritmos de apuestas	10 days	Thu 12/05/11	Sat 21/05/11
23	Testeos integrados	2 days	Sun 22/05/11	Mon 23/05/11
24	**Entrega final**	**21 days**	**Tue 24/05/11**	**Mon 13/06/11**
25	Elaboración de la memoria	12 days	Tue 24/05/11	Sat 04/06/11
26	Elaboración de la presentación	9 days	Sun 05/06/11	Mon 13/06/11

Figura 2 – Planificación inicial del proyecto

- ***Planificación real***

Tras la realización de las tres PEC hubo que ir reajustando la planificación inicial, cambiando el tiempo previsto de realización de las tareas o moviendo tareas de un hito a otro. El resultado final puede verse en la siguiente captura de pantalla:

	Task Name	Duration	Start	Finish	Pred
1	**Trabajo Final de Proyecto Microsoft .NET**	**103 days**	**Thu 03/03/11**	**Mon 13/06/11**	
2	**PEC 1 - Plan de trabajo**	**14 days**	**Thu 03/03/11**	**Wed 16/03/11**	
3	Definición de proyecto	2 days	Thu 03/03/11	Fri 04/03/11	
4	Descripción de la aplicación	3 days	Sun 06/03/11	Tue 08/03/11	3
5	Metodología y herramientas empleadas	3 days	Wed 09/03/11	Fri 11/03/11	4
6	Calendario de trabajo	4 days	Sun 13/03/11	Wed 16/03/11	5
7	**PEC 2 - Análisis y diseño de la aplicación**	**26 days**	**Thu 17/03/11**	**Mon 11/04/11**	**2**
8	Análisis de requerimientos formales	2 days	Thu 17/03/11	Fri 18/03/11	6
9	Definición de actores	1 day	Sat 19/03/11	Sat 19/03/11	8
10	Análisis de casos de uso	5 days	Mon 21/03/11	Fri 25/03/11	9
11	Diagrama de casos de uso	3 days	Sat 26/03/11	Mon 28/03/11	10
12	Diseño de la arquitectura del sistema	2 days	Tue 29/03/11	Wed 30/03/11	11
13	Diagramas de secuencia	2 days	Fri 01/04/11	Sat 02/04/11	12
14	Diagrama de clases	5 days	Sun 03/04/11	Thu 07/04/11	13
15	Modelo de datos	3 days	Fri 08/04/11	Sun 10/04/11	14
16	Revisión calendario de trabajo	1 day	Mon 11/04/11	Mon 11/04/11	15
17	**PEC 3 - Implementación**	**42 days**	**Tue 12/04/11**	**Mon 23/05/11**	**7**
18	Diseño de la interfaz	2 days	Tue 12/04/11	Wed 13/04/11	16
19	Estudio de la aplicación de ejemplo de Betfair	18 days	Thu 14/04/11	Sun 01/05/11	18
20	Implementación del interfaz del usuario	7 days	Mon 02/05/11	Sun 08/05/11	19
21	Implementación del acceso a base de datos	7 days	Mon 09/05/11	Sun 15/05/11	20
22	Implementación de los algoritmos de apuestas	6 days	Mon 16/05/11	Sat 21/05/11	21
23	Testeos integrados	2 days	Sun 22/05/11	Mon 23/05/11	22
24	**Entrega final**	**21 days**	**Tue 24/05/11**	**Mon 13/06/11**	**17**
25	Elaboración de la memoria	12 days	Tue 24/05/11	Sat 04/06/11	23
26	Elaboración de la presentación	9 days	Sun 05/06/11	Mon 13/06/11	25

Figura 3 – Planificación real del proyecto

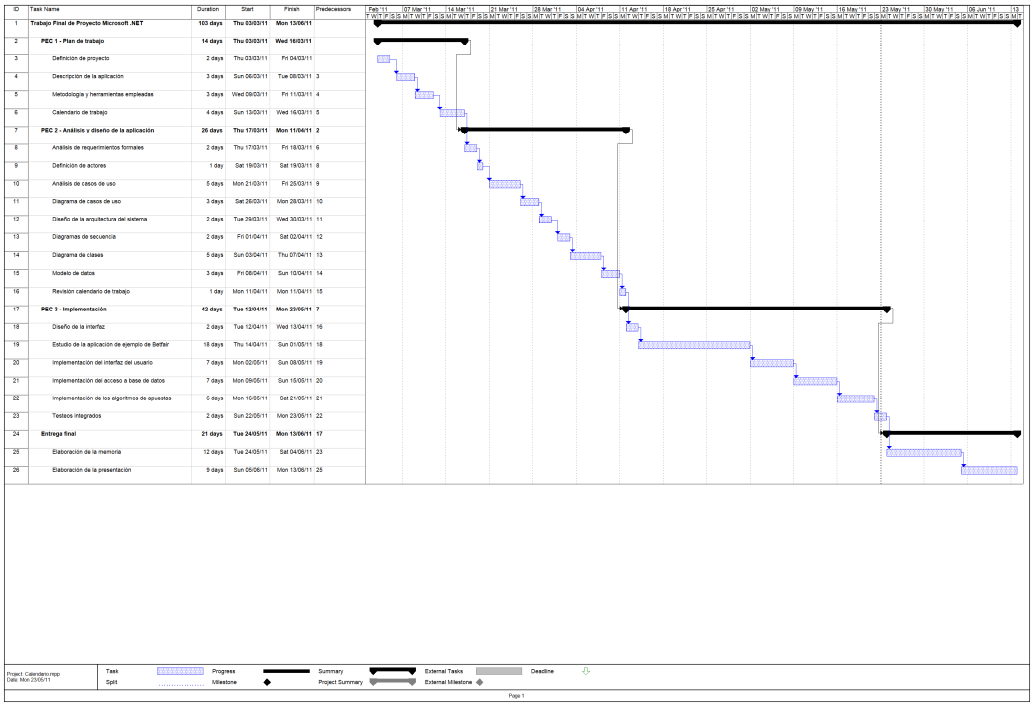


Figura 4 – Diagrama de Gantt de la planificación real

6- Análisis de requerimientos formales

La aplicación de simulación de apuestas automáticas en los X-Games de Betfair deberá contar con un sencillo interfaz gráfico que permita al usuario de manera sencilla elegir la estrategia que desea usar y la cantidad de dinero que desea apostar.

La aplicación se conectará al sistema de apuestas de Betfair por medio de su API, e irá mostrando en pantalla como transcurrirá la partida de Poker, mostrando las cartas de cada jugador y las de la mesa. Por lo tanto habrá 4 fases que mostrar en pantalla: el Deal (cuando todavía no se ha repartido carta a los jugadores), el Preflop (cuando se reparte 2 cartas a cada jugador), el Flop (cuando se han puesto 3 cartas sobre la mesa),

el Turn (cuando se ha puesto la cuarta carta sobre la mesa) y el River (cuando se pone la 5ª carta sobre la mesa y los jugadores enseñan su mano para ver quién es el ganador).

Durante las 4 primeras fases el sistema podrá ir realizando apuestas automáticas a las distintas manos de los jugadores dependiendo de la estrategia escogida así como de la cantidad indicada por el usuario. En la quinta fase, el River, se decidirá el ganador, que puede ser más de 1, un empate, en cuyo caso se llama un Dead Match, y en cuyo caso los beneficios se reparten entre los jugadores.

Se puede apostar a favor, es decir, que una mano será la ganadora, en cuyo caso se obtendría como ganancias el importe jugado por la cuota a favor de la apuesta, o a perdedor, es decir, que una mano no ganará, en cuyo caso lo que estamos apostando es la cantidad de dinero que queremos ganar y sería la que obtendríamos. Automáticamente cada vez que se apueste el sistema retirará ese importe de nuestros fondos, si es a favor retirará lo que hayamos jugado, si es a perdedor retirará la posible pérdida.

Una vez acabada la quinta mano, el River, y por lo tanto la partida, el sistema determinará si nuestras apuestas han sido ganadoras. Para calcular el importe de las apuestas ganadoras se usará el sistema comentado en el párrafo anterior para obtener las ganancias.

Las apuestas que hayamos perdido a favor simplemente se darán por perdidas, mientras que en las apuestas a perdedor que hemos fallado el importe de la pérdida se calculará multiplicando la cantidad apostada por su cuota de pérdida y se descontará de nuestros fondos.

Por ello es necesario mencionar en este punto que para poder realizar una apuesta a perdedor antes la aplicación debe de comprobar que nuestros fondos actuales podrían

cubrir la posible pérdida, dinero que retirará previamente antes de la apuesta a perdedor y que nos será devuelto junto con la ganancia si acertamos la apuesta.

7- Definición de actores

En el sistema encontramos los siguientes actores:

- **Usuario**: será el encargado de comenzar o realizar las apuestas, seleccionar el tipo de estrategia a seguir, establecer el límite de pérdidas y la cantidad a invertir en cada apuesta a favor a y a perdedor.

- **Sistema**: será el encargado de realizar las apuestas virtuales y la comunicación con el servidor remoto de Betfair para solicitar el estado de la jugada. Una vez terminada la jugada el sistema determinará las ganancias o pérdidas.

- **Servidor remoto**: será el encargado de devolver a la aplicación el estado del juego, las cartas en juego, tanto en el tablero como las de cada jugador, e indicará el ganador/es y perdedor/es de la partida.

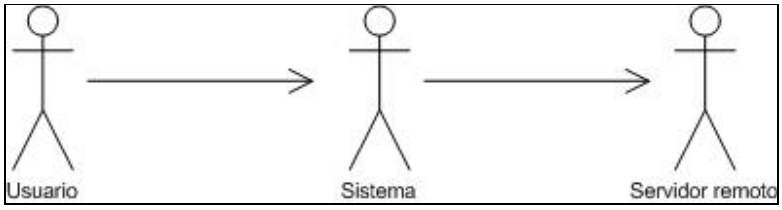


Figura 5 – Relación entre actores

8- Análisis y diagramas de casos de uso

8.1- Casos de uso de usuario

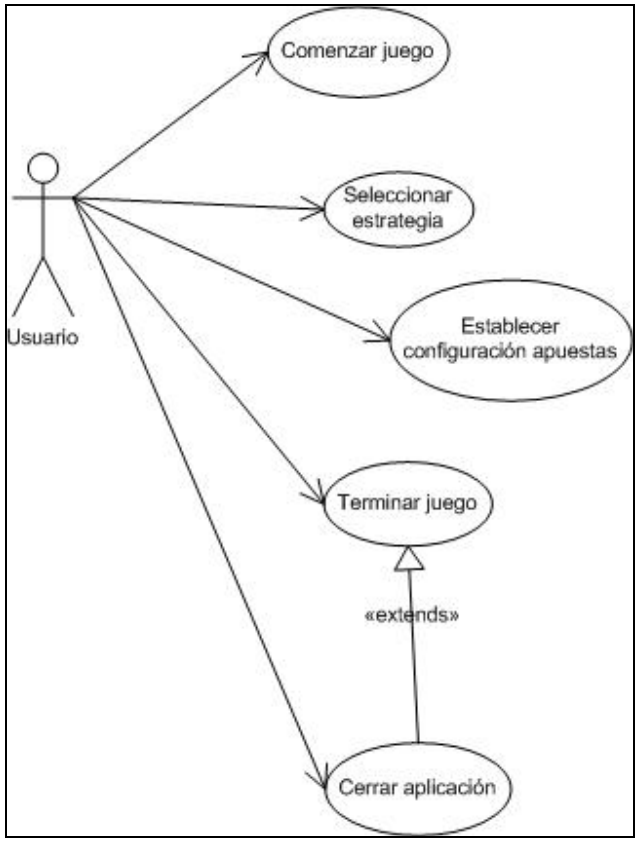


Figura 6 – Casos de uso para usuario

Caso de Uso 1 – Comenzar el juego
Resumen: el usuario pulsa el botón de comenzar juego.
Actores: usuario.
Casos de uso relacionados
Precondición: el juego no está arrancado.
Postcondición: se conecta al servidor remoto y arranca el juego.
Proceso principal: 1- El usuario pulsa el botón de comenzar juego.
Excepciones

1- El sistema no consigue conectarse al sistema remoto.
Observaciones

Caso de Uso 2 – Establecer configuración de apuestas
Resumen: el usuario establece la configuración para las apuestas.
Actores: usuario
Casos de uso relacionados
Precondición: el juego no está arrancado.
Postcondición: establece la configuración de apuestas a aplicar en el juego.
Proceso principal: 1- El usuario introduce los parámetros de configuración de apuestas.
Excepciones
Observaciones

Caso de Uso 3 – Seleccionar estrategia
Resumen: el usuario selecciona la estrategia a usar en el juego.
Actores: usuario.
Casos de uso relacionados
Precondición: el juego no está arrancado.
Postcondición: establece la estrategia a usar durante el juego.
Proceso principal: 1- El usuario selecciona la estrategia a seguir dentro de las disponibles.
Excepciones
Observaciones

Caso de Uso 4 – Terminar juego

Resumen: el usuario pulsa el botón de terminar juego.
Actores: usuario.
Casos de uso relacionados
Precondición: el juego está arrancado.
Postcondición: se termina el juego.
Proceso principal: 1- El usuario pulsa el botón de terminar juego. 2- El juego se para.
Excepciones
Observaciones

Caso de Uso 5 – Cerrar aplicación
Resumen: el usuario cierra la aplicación.
Actores: usuario.
Casos de uso relacionados: puede usar el caso de uso número 4 (terminar juego).
Precondición:
Postcondición: la aplicación se cierra.
Proceso principal: 1- El usuario pulsa el botón de cerrar o el aspa de la ventana. 2- Si el juego está arrancado, se mostrará un mensaje preguntando si desea realmente cerrar la aplicación. 3- Si se responde afirmativamente a ese mensaje o no estaba el juego arrancado, se cierra la ventana
Excepciones
Observaciones:

8.2- Casos de uso de sistema

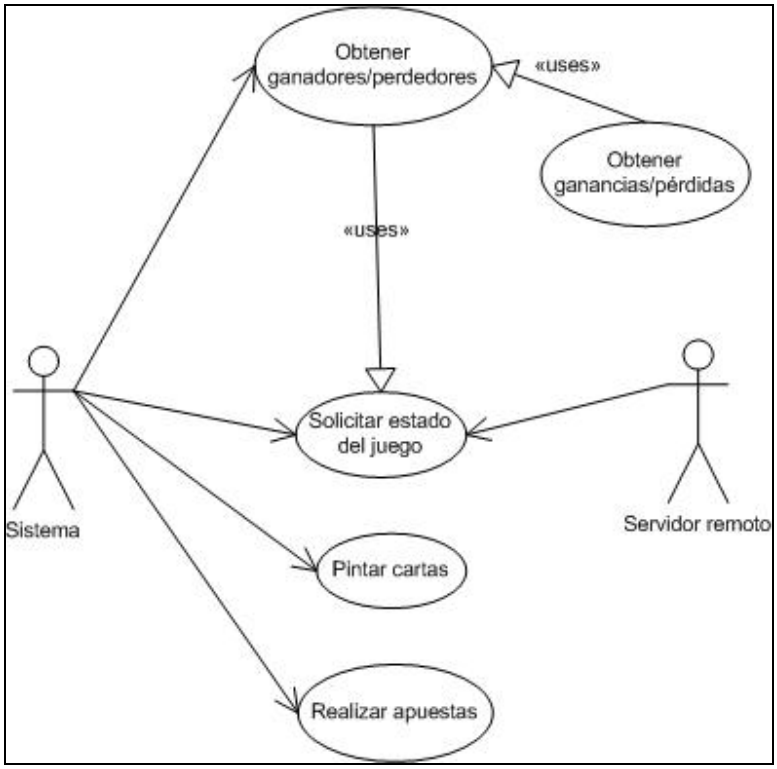


Figura 7- Casos de uso para sistema

Caso de Uso 6 – Solicitar estado del juego
Resumen: el sistema pregunta al servidor remoto por el estado de la partida.
Actores: sistema, servidor remoto.
Casos de uso relacionados
Precondición: el juego está arrancado.
Postcondición: se conecta al servidor remoto y obtiene el estado del juego.
Proceso principal: 1- El sistema se conecta al servidor remoto para ver el estado actual del juego. 2- El servidor remoto devuelve al sistema el estado del juego.
Excepciones

1- El sistema no consigue conectarse al sistema remoto.
Observaciones

Caso de Uso 7 – Pintar cartas
Resumen: el sistema pinta las cartas en pantalla.
Actores: sistema.
Casos de uso relacionados
Precondición: se han obtenido previamente los datos de las cartas del servidor.
Postcondición: pinta las cartas en pantalla.
Proceso principal: 1- El sistema pinta las cartas en la pantalla según los datos que le ha devuelto el servidor.
Excepciones
Observaciones

Caso de Uso 8 – Realizar apuestas
Resumen: el sistema realiza una apuesta virtual.
Actores: sistema.
Casos de uso relacionados
Precondición: el juego está arrancado.
Postcondición: se realiza una apuesta virtual.
Proceso principal: 1- El sistema realiza una apuesta virtual a una serie de jugadores y en un momento determinado del juego.
Excepciones
Observaciones

Caso de Uso 9 – Obtener ganadores/perdedores
Resumen: el sistema determina el ganador del juego.
Actores: sistema, servidor remoto.
Casos de uso relacionados: se usa el caso de uso número 6 solicitar estado del juego.
Precondición: el juego está arrancado.
Postcondición: se obtiene la lista de jugadores ganadores/perdedores.
Proceso principal: 1- El sistema se conecta al servidor remoto preguntando por el estado del juego. 2- El servidor remoto devuelve la lista de los jugadores ganadores.
Excepciones 1- El sistema no consigue conectarse al sistema remoto.
Observaciones

Caso de Uso 10 – Obtener ganancias/pérdidas
Resumen: el sistema obtiene las ganancias/pérdidas que se han producido cuando finaliza el juego.
Actores: sistema.
Casos de uso relacionados
Precondición: el juego no está arrancado.
Postcondición: se ha incrementado o disminuido la cantidad de saldo disponible en la cuenta dependiendo de las ganancias/pérdidas.
Proceso principal: 1- El sistema obtiene la lista de ganadores/perdedores. 2- En función de ellos calculas las ganancias o pérdidas producidas dependiendo de las apuestas realizadas durante el juego.
Excepciones
Observaciones

9- Diseño de la arquitectura del sistema

El sistema constará de los siguientes componentes:

- ***Servidor remoto de Betfair:*** el cual mediante un WebService estará constantemente escuchando llamadas de aplicaciones clientes.

- ***Aplicación cliente:*** basada en tecnología .NET Framework 4.0 se encargará de la comunicación con el servidor de Betfair mediante el uso del API de la misma empresa. También se encargará de la grabación de las partidas en la base de datos para su posterior análisis.

- ***Base de datos:*** basado en un servidor Microsoft SQL Server Express R2 se encargará de almacenar los datos relacionados con las partidas realizadas para su posterior análisis.

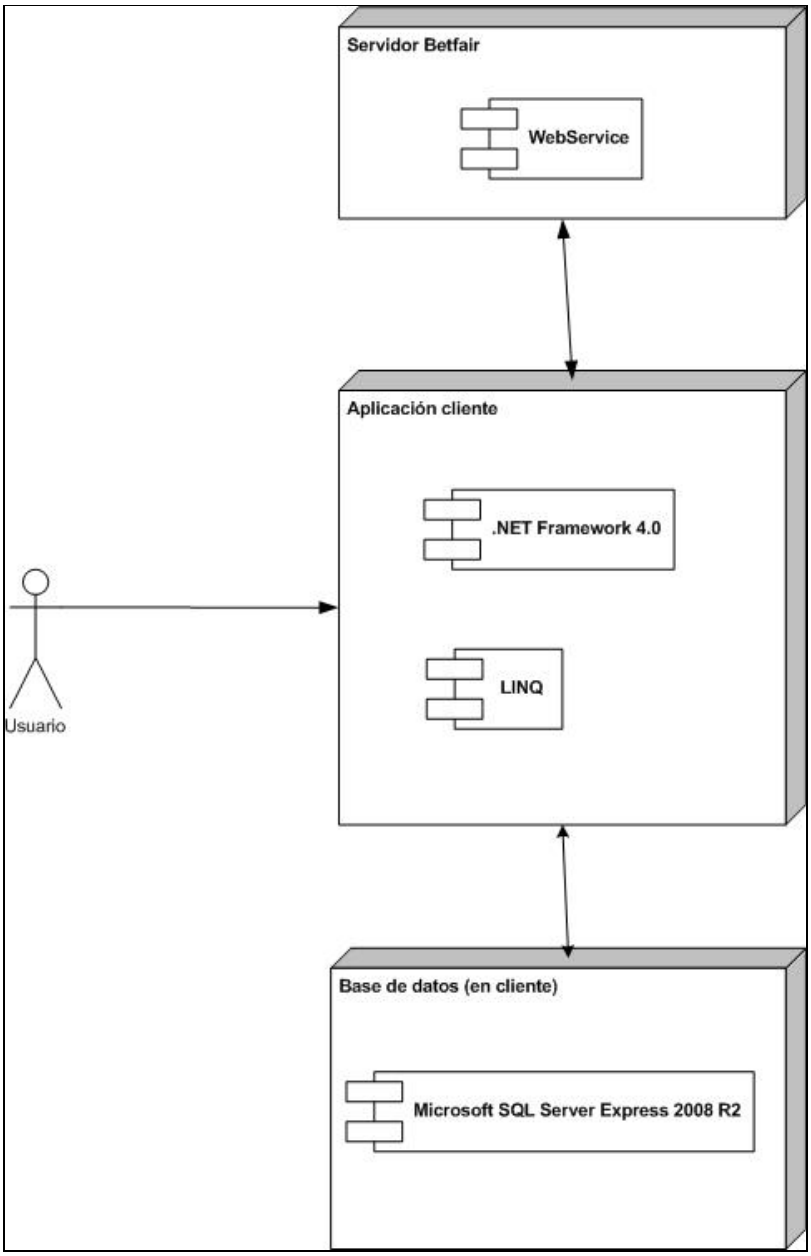


Figura 8- Arquitectura del sistema

10- Diagramas de secuencia

10.1 – Comenzar juego

Este diagrama de secuencia muestra el proceso del comienzo del juego, desde que el usuario pulsa el botón de comenzar el juego hasta que las cartas se pintan en la aplicación, así como la grabación de los datos de la partida en la base de datos.

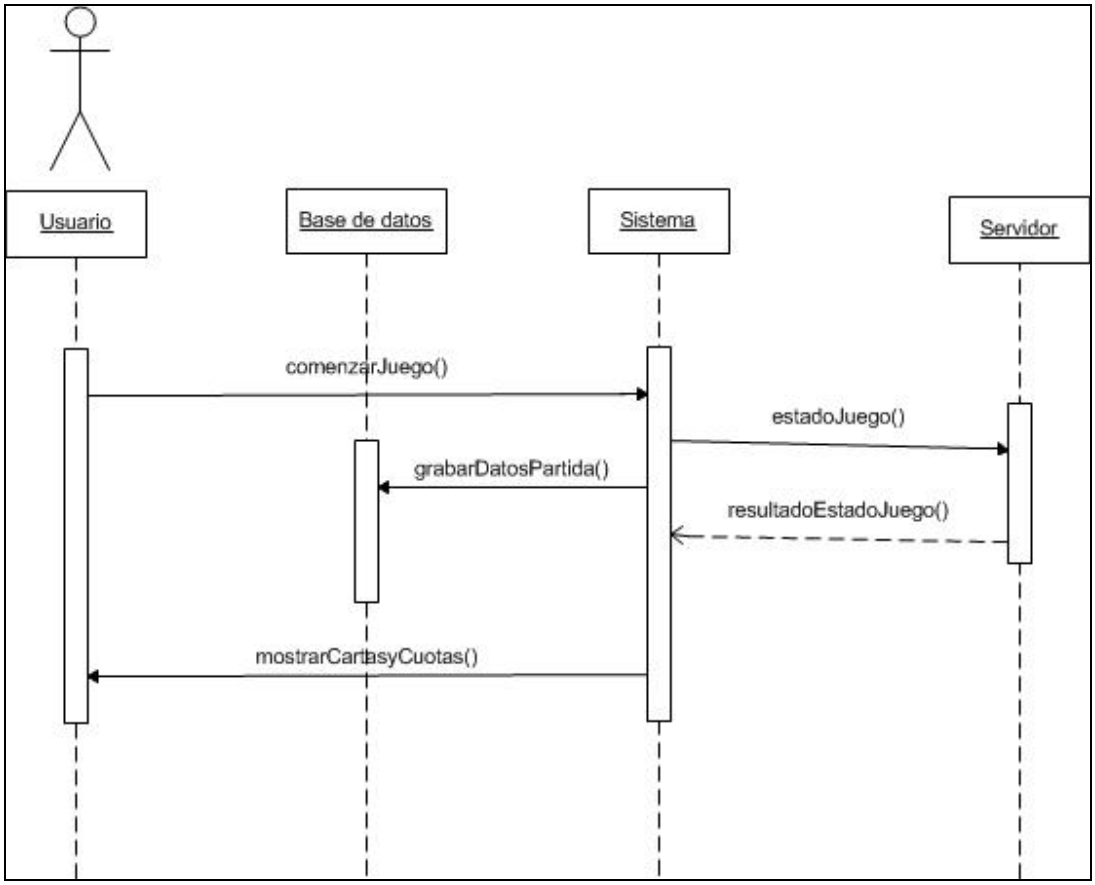


Figura 9- Comienzo de una partida

10.2 – Realizar una apuesta

Este diagrama de secuencia muestra el proceso de realización de una apuesta, desde que el usuario selecciona una estrategia a seguir y los parámetros de

configuración de la apuesta (por ejemplo, saldo a apostar) hasta que la apuesta es finalmente realizada.

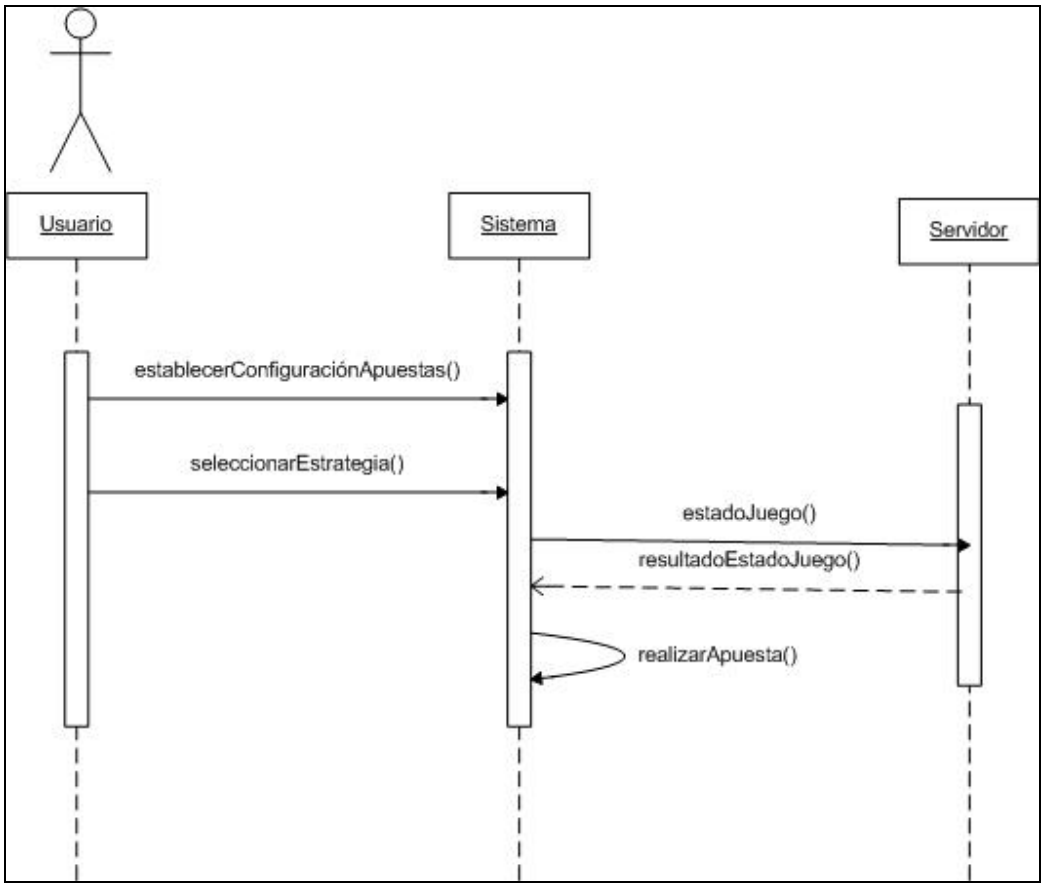


Figura 10- Realizar una apuesta

10.3 – Obtener ganador y ganancias/pérdidas

Este diagrama de secuencia muestra el proceso por el cual la aplicación obtiene cuál es el ganador/es y perdedor/es de la apuesta y en función de ello y en base a las apuestas realizadas previamente obtiene el correspondiente beneficio o pérdida.

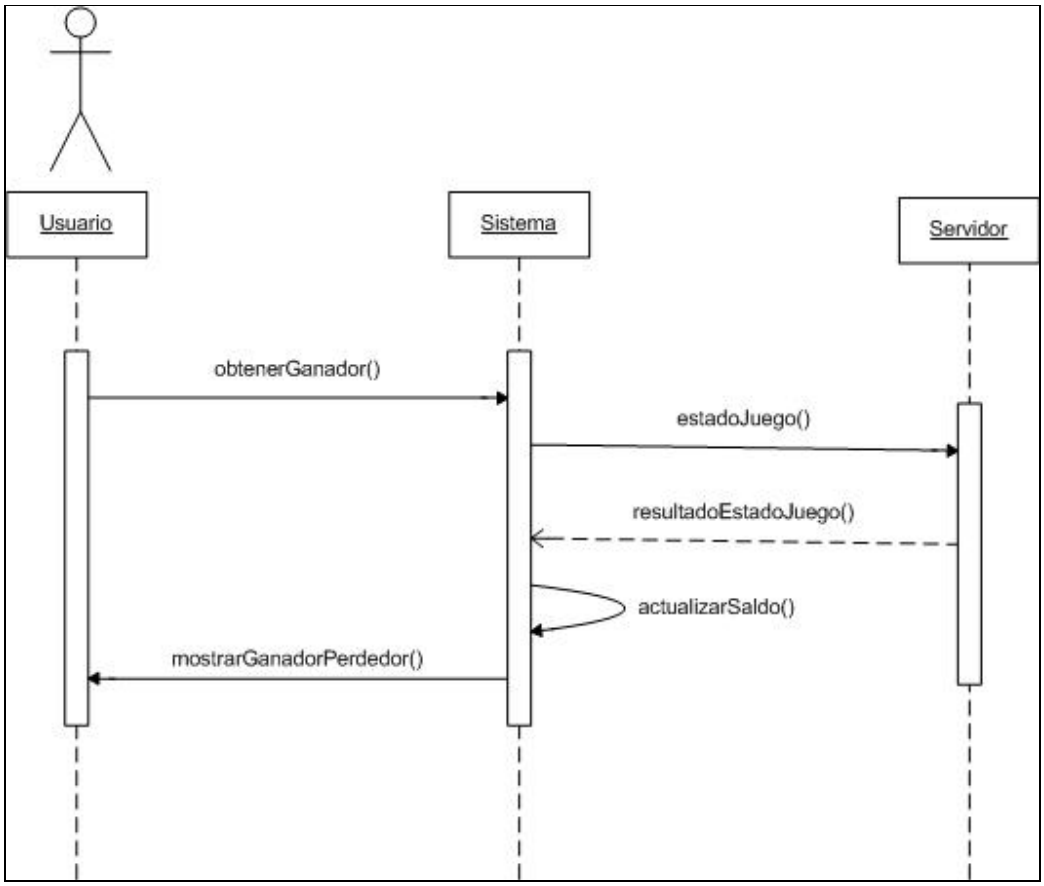


Figura 11- Obtener ganador y ganancias/pérdidas

11- Diagrama de clases

11.1 – Clases principales

Las siguientes son las clases principales del sistema. Toda la lógica de la realización de apuestas, aplicación de estrategias, pintado de cartas y control de ganadores/perdedores y ganancias/pérdidas son realizadas por ellas.

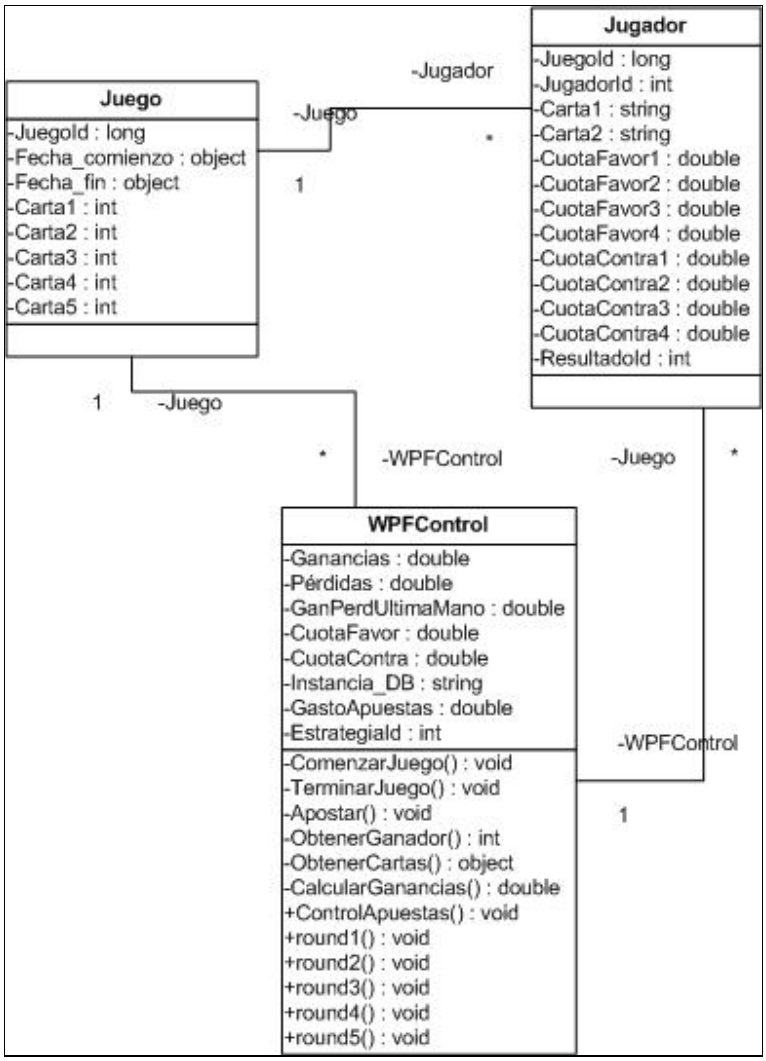


Figura 12 – Diagrama de clases del sistema

Nombre de clase	Descripción
Juego	Contendrá los datos del juego actual.
Jugador	Contendrá los datos de los jugadores.
WPFControl	Se encarga de realizar las apuestas, obtener los ganadores, y las ganancias y pérdidas.

11.2 – Clases gestoras

Las clases gestoras están encargadas del proceso de almacenamiento en la base de datos de todos los datos relativos a una partida para su posterior análisis: las cuotas, las cartas en la mesa y de los jugadores durante una partida, los ganadores/perdedores, etc.

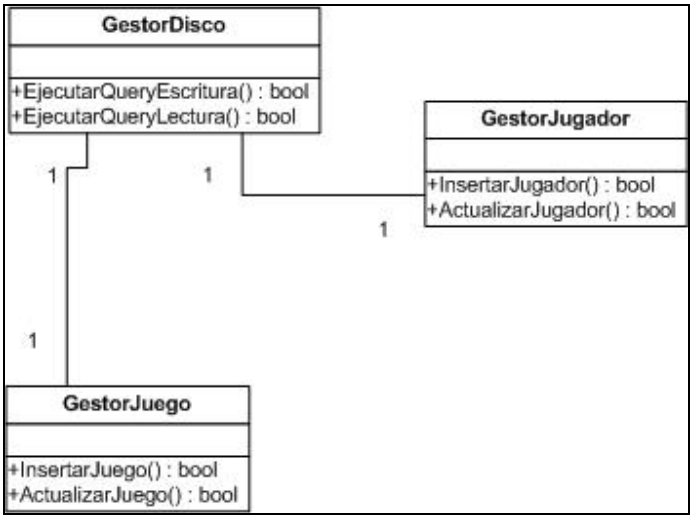


Figura 13 – Diagrama de clases gestoras

Nombre de clase	Descripción
GestorDisco	Realizará las operaciones generales de inserción/grabación en la base de datos.
GestorJugador	Grabará los datos de los jugadores durante la partida en la base de datos.
GestorJuego	Grabará los datos del juego actual en la base de datos.

12- Modelo de datos

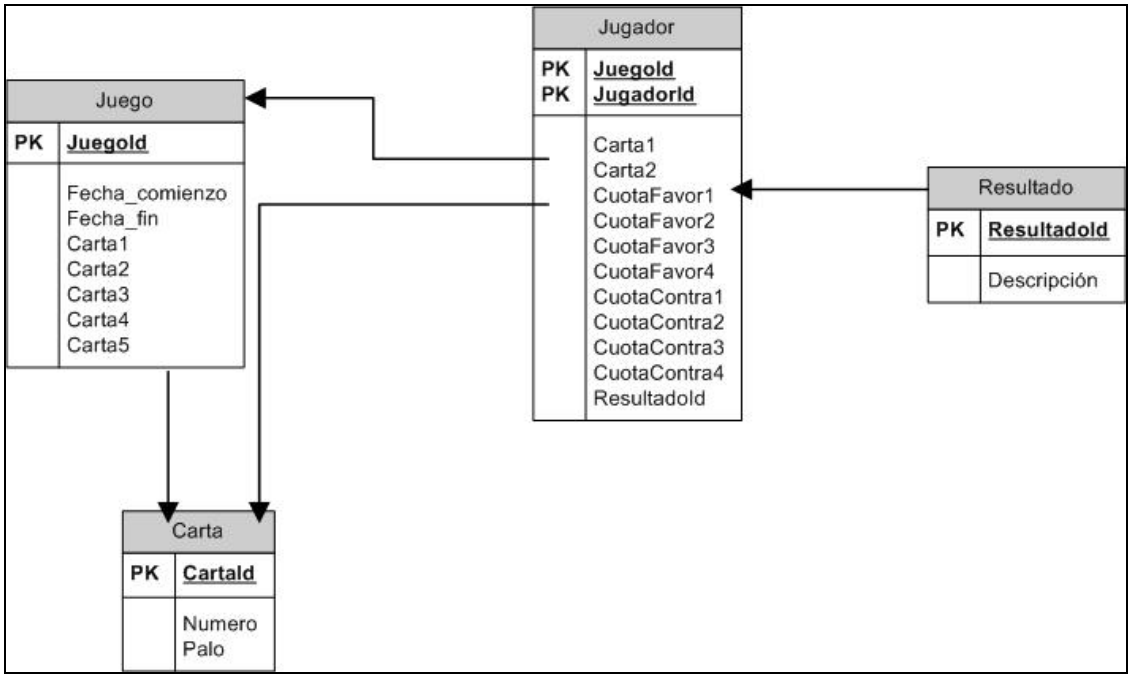


Figura 14 – Modelo de datos

Nombre de tabla	Descripción
Juego	Contendrá los datos de cada partida jugada por el sistema.
Cartas	Tabla maestra que contendrá el nombre de todas las cartas.
Jugador	Contendrá los datos de las cartas de los jugadores y del estado de éstos en el juego.
Resultado	Tabla maestra que contiene los 3 posibles estados de un jugador al final de la partida: ganador, empate, perdedor.

13- Descripción técnica de la aplicación

Para el desarrollo de la aplicación se han usado las siguientes tecnologías:

- Microsoft Visual C# 2010 Express Service Pack 1.
- Microsoft SQL Server 2008 R2 Express.

La aplicación consiste en un sistema de emulación apuestas automáticas para el Poker modalidad Texas Hold'em que ofrece la famosa casa de apuestas por Internet BetFair. Como se comentó anteriormente esta empresa ofrece un sistema de apuestas mediante un API que permite la creación de aplicaciones cliente que estén conectadas 24x7 realizando apuestas en el sistema.

Para la realización del sistema de apuestas fue necesario usar el API en .NET que ofrece Betfair para desarrolladores. A su vez Betfair ofrece el API junto con una aplicación de código abierto de muestra, llamada UberClient, para que sea usada por los programadores.

Así que tomando como base el código fuente de esa aplicación, el UberClient, se modificó para ajustarla al proyecto que nos interesa, es decir, la creación de un sistema automático de apuestas. Por otra parte, debido al tiempo disponible para realizar la codificación del proyecto, apenas 1 mes y medio, era inviable realizar una aplicación de este tipo desde cero, por lo que esta fue también otra de las razones por las que se decidió basarse en esta aplicación.

Para ello hubo que modificar la aplicación para que sólo pudiera jugar al poker (pues está preparada para jugar con casi todos los juegos con API que ofrece Betfair). Posteriormente hubo que traducirla a español y lo más importante, hubo que añadir un control WPF que se encargaría de la gestión de todo el proceso de apuestas

automáticas, pues la aplicación que ofrece Betfair de ejemplo sólo permite realizar apuestas manuales.

Debido a la dificultad que implicaba eliminar los controles y formularios de la aplicación de ejemplo de Betfair, ya que al eliminar cualquiera de ellos salían luego cientos de errores de compilación, se optó por ocultarlos con lo que se consiguió salvar así este problema.

La aplicación de Betfair está realizada con WinForms ya que fue realizada con Visual Studio 2005 cuando la tecnología WPF todavía no había hecho su aparición en el mercado. Sin embargo queríamos que el módulo de apuestas automáticas estuviera realizado con WPF, por lo que de primeras existía un problema: los sistemas son incompatibles entre sí, o mejor dicho, no son capaces de verse ni de entenderse unos a otros. Un formulario WinForm no puede instanciar a uno WPF ni viceversa.

Tras buscar documentación al respecto se vio que existían dos posibilidades: o bien usar un formulario WPF y dentro un control WinForm, o bien usar un formulario WinForm y dentro un control WPF. Puesto que la aplicación de Betfair estaba realizada en WinForm, la decisión lógica que se usó fue usar el código existente de WinForm y añadir un control en WPF que contendría toda la lógica y el interfaz gráfico del módulo de apuestas automáticas.

Por lo tanto la arquitectura quedó así:

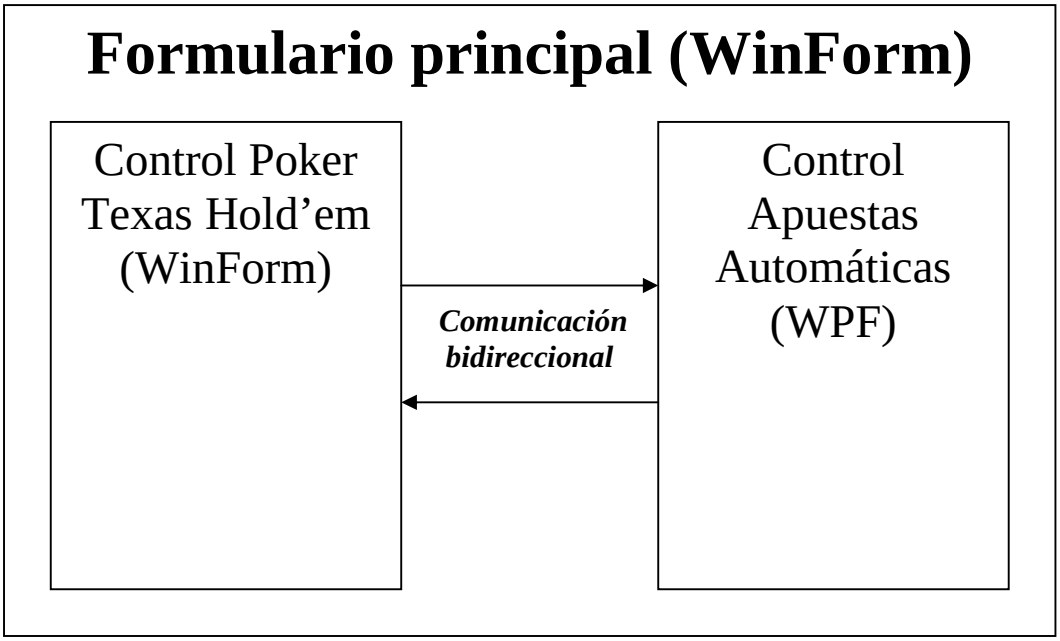


Figura 15 – Arquitectura de la aplicación

Tal como se muestra en el dibujo, la comunicación entre el control WinForm encargado de mostrar el juego de poker y el control WPF encargado de realizar y gestionar las apuestas automáticas es bidireccional, aunque prima la comunicación desde el componente WPF hacia el componente WinForm.

Quizás quede mucho más claro esto viendo una captura de pantalla de la aplicación:

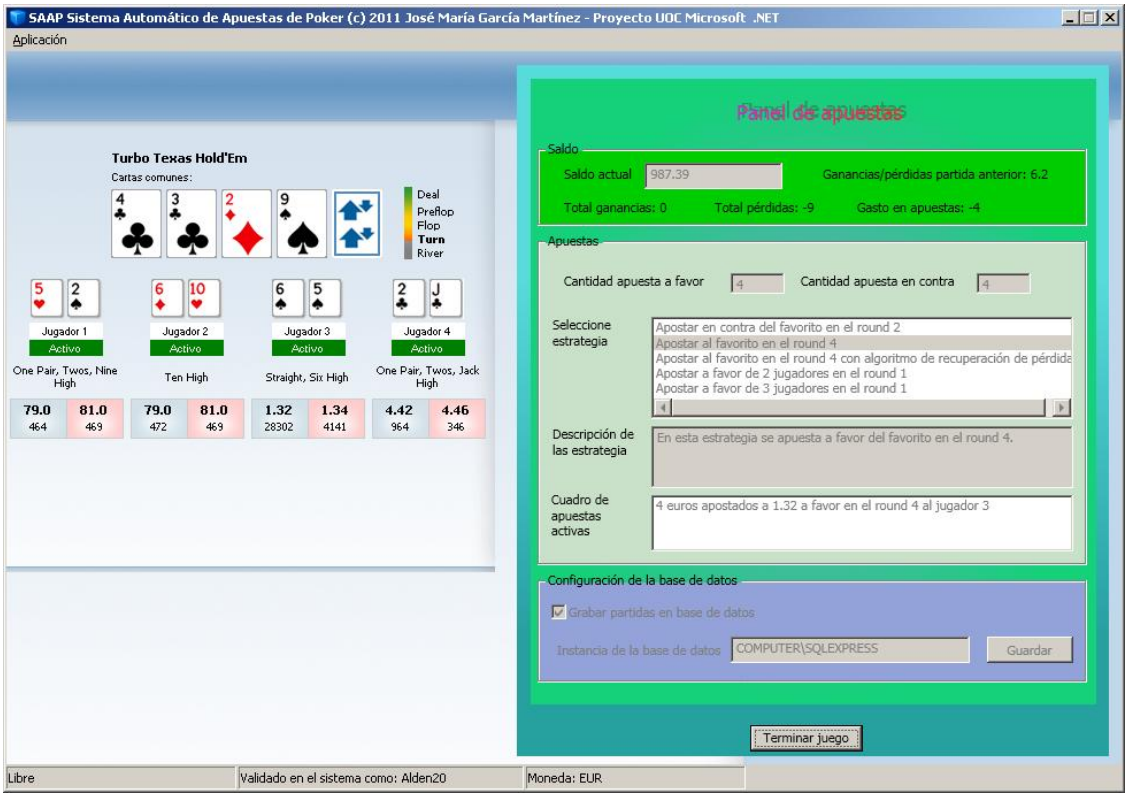


Figura 16 – La aplicación en activo efectuando una apuesta automática.

Como se ve puede apreciarse a la izquierda el control WinForm que muestra los datos relativos a la partida de poker, mientras que a la izquierda puede verse el control WPF que maneja todo el proceso de las apuestas.

En cuanto a la base de datos, su uso es opcional. Si el usuario así lo desea, se irán grabando en la base de datos los resultados de todas las partidas, se llegué o no a realizar apuestas sobre ellas. Para la grabación de los datos se ha usado la tecnología Linq de tal manera que las propias clases y objetos que se usan en memoria por parte de la aplicación para el proceso de apuestas son las mismas que se usan para la grabación en la base de datos.

Entrando más en detalle, el flujo de funcionamiento de la aplicación de apuestas automáticas es el siguiente:

1- El usuario se valida en el sistema.
2- Le aparece el control WPF de apuestas donde debe de seleccionar la estrategia a seguir, el importe por cada apuesta y el saldo inicial. Asimismo deberá elegir si desea grabar cada partida en la base de datos y si es así introducir el nombre de la instancia del SQL Server donde estará la base de datos Poker.
3- Cuando pulse el botón "comenzar juego" se iniciará el juego de poker. Se mostrará el control WinForm que controla el desarrollo del juego, en el que se irán mostrando las cartas comunes y de cada jugador, y las cuotas asociadas a cada uno de ellos.
4- Cada vez que se inicie un nuevo round en la partida el componente WinForm llamará al componente WPF para verificar si se ha de realizar alguna apuesta en el sistema.
5- Cuando termina la partida el componente WinForm llama al componente WPF que se encarga en este caso de obtener todas las ganancias y pérdidas que se producirán según las apuestas que se hayan hecho en el sistema, modificando los indicadores monetarios de pantalla en este sentido.
6- Si así se ha establecido, se grabará la partida jugada en la base de datos para su posterior análisis.

En cuanto a la estructura del código fuente es la siguiente:

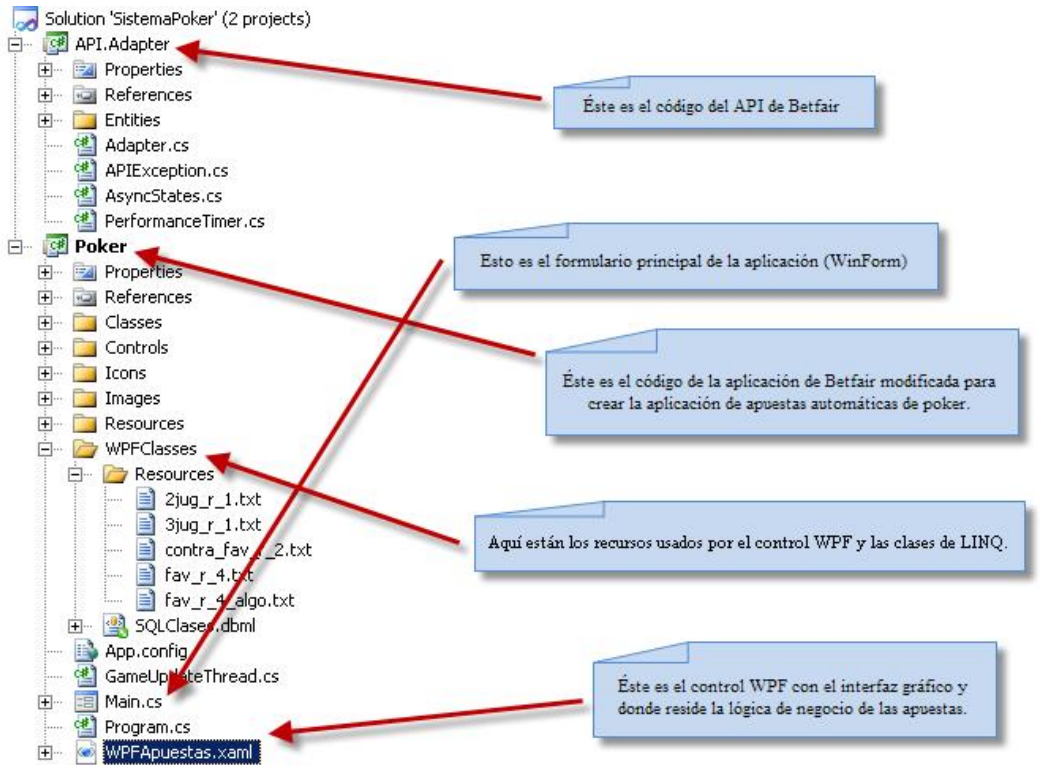


Figura 17 – Estructura del proyecto

Una descripción más detallada del árbol de la estructura sería el siguiente:

- **Proyecto API.Adapter**: contiene todo el código para el manejo del API de Betfair.
- **Proyecto Poker**: es la aplicación que se creado de apuestas automáticas de poker tomando como base la aplicación de ejemplo que ofrece Betfair. Dentro de este proyecto destacamos los siguientes carpetas y ficheros que son los principales usados por la aplicación:
    - **Controls/GameView**: contiene los controles WinForm para los juegos. En nuestro caso nos interesa el fichero llamado ***TexasHoldem.cs*** que contiene el control para jugar al poker.

- **WPFClasses**: contiene los recursos usados por el control WPF para las apuestas (en este caso ficheros de texto describiendo en qué consiste la estrategia de la apuesta) así como las clases necesarias para trabajar con LINQ.
- **GameUpdateThread.cs**: contiene el código que controla la partida y es lanzado en un nuevo hilo de ejecución.
- **Main.cs**: contiene el formulario principal (WinForm) en el que se alojan el control WinForm de poker y el control WPF de apuestas automáticas

14- Complejidad y problemas encontrados

Una vez que se decidió las tecnologías que se iban a usar para desarrollar el proyecto, lo primero que se hizo fue bajarse el código de la aplicación UberClient de Betfair para comprender el funcionamiento de la misma y del API.

El primer problema que se encontró fue que el código estaba desarrollado para Visual Studio 2005, por lo que al realizar la importación al Visual Studio 2010 salieron varios problemas en el código que tuvieron que ser subsanados por incompatibilidad con el .NET Framework 4.0.

Una vez solucionados éstos, el siguiente y el mayor problema durante todo el proyecto fue comprender el código y entender como funcionaba la aplicación y el proceso del juego. Esto llevó bastante tiempo hasta que se empezó a entender como trabajaba, sobre todo porque la aplicación es multithread y trabaja con varios hilos de ejecución, lo que dificultaba su comprensión, además teniendo en cuenta que el Visual Studio 2010 C# Express no incorpora el visor de threads que sí incorporan las versiones superiores, por lo que era incapaz de saber en que thread me encontraba.

Como he dicho, la comprensión del código llevó mucho tiempo y se ha comido bastante tiempo del proyecto.

El siguiente problema que se encontró era que el código estaba desarrollado con WinForms y la funcionalidad que queríamos desarrollar tenía que serlo con WPF, por lo que de primeras ambos sistemas son incompatibles, no pueden verse ni entenderse entre sí.

Así que hubo que realizar una búsqueda de información para ver si era factible de alguna manera que ambos sistemas pudieran comunicarse. Tras realizar dicho estudio se llegó a la siguiente conclusión: era posible, mediante un componente de .NET, que WinForm y WPF pudieran coexistir en un mismo proyecto, habiendo dos posibilidades de hacerlo: bien usando ese componente en un formulario WPF, lo que permitiría incluir un control WinForm en el mismo, o bien a la inversa, usando un formulario WinForm y dentro el componente que permitiría usar un control WPF.

Tras evaluar los pros y contras de ambas opciones, se llegó a la conclusión de que la opción más lógica era, puesto que la aplicación de Betfair está desarrollada con WinForms, usar un componente con un control WPF dentro de un formulario WinForm.

Una vez elegida esta opción, hubo que implementarla y aquí evidentemente aparecieron problemas, pues aunque venía detallado en varias páginas web como realizarlo no conseguía dar con ello y siempre aparecía algún tipo de problema. Por fin después de unos días conseguí enterarme realmente de como funciona y logré insertar un control WPF dentro del componente de .NET y dentro éste a su vez del formulario WinForm.

El siguiente problema que apareció respecto a esto era conseguir la comunicación entre el control WinForm de poker y el control WPF, lo cual también me llevó unos días hasta que comprendí como se debía de llevar a cabo.

Una vez llevado a cabo todo esto el siguiente problema que me encontré, y el peor de todos, fue comprender como funcionaba el código de apuestas de Betfair, no tenía ni idea de dónde podía obtener los datos que necesitaba para las apuestas (cuotas de jugadores, favorito, ganadores y perdedores, etc). Además para añadir complejidad al asunto la aplicación es multithread y además ésta depende del estado del juego en el servidor, es decir, que el juego sigue corriendo en el servidor aunque el flujo del cliente esté detenido, lo que puede provocar que si se realiza un debugueo que dure más de unos minutos cuando continúe el flujo del juego éste lo haga en otro sitio del esperado.

El problema de la comprensión del código de Betfair para saber como se deberían implementar las apuestas, como funcionaban, donde podría obtener los datos que necesitaba, en qué momento estarían disponibles esos datos, etc. me llevó mucho tiempo y se comió bastante tiempo del proyecto.

Cuando ya por fin logré comprender como funcionaba todo y empecé a implementar el código, el problema que me encontré fue el de los testeos. Debido a que cada partida dura 5 minutos aproximadamente, y además teniendo en cuenta que algunas situaciones en la partida sólo se dan aleatoriamente (por ejemplo los empates o que se termine la partida en el round 4 y no en el 5), me llevó mucho tiempo realizar las pruebas y corrección de errores.

Por último también hubo problemas en la implementación de LINQ, primero para enterarme de cómo funcionaba, ya que yo siempre había tirado las SQL's por el método clásico de ADO (y ni siquiera hablo de ADO .NET, hablo del ADO de VB 6 ya que yo apenas he programado en .NET). Pero además resulta que el driver de

conexión a SQL que trae Visual Studio 2010 C# Express no permite conectarse a un servidor remoto, sino sólo a un fichero de base de datos local. Esto provocó que siempre saliera el siguiente mensaje de error cuando se intentaba usad: "*An attempt to attach an auto-named database for file XXXX failed. A database with the same name exists, or specified file cannot be opened, or it is located on UNC share*". Por más que intenté solucionarlo buscando información en Internet no hubo manera, el único modo que encontré que funcionase fue hacer la instancia por código y en lugar de a un fichero a una instancia de SQL Server en el equipo.

15- Testeos llevados a cabo

Se ha intentado probar la aplicación todo lo posible. No obstante los testeos han sido complicados hablando en tiempo ya que para que algunas situaciones pudieran darse han hecho falta que pasasen decenas de partidas.

En la siguiente tabla aparece una lista con las principales pruebas que se han realizado a la aplicación:

Validación correcta de usuario en Betfair.
Validación del formulario (campos vacíos, valores incorrectos, activación/desactivación de controles).
Apuestas correctas (importe adecuado, jugador elegido, round concreto).
Obtención de ganancias/pérdidas correctas (cálculo importe perdido en última apuesta, ganancias totales, pérdidas totales, saldo actual).
Aplicación correcta del algoritmo de recuperación de pérdidas (ecuación lineal).
Prueba de resistencia a bugs (8 horas seguidas corriendo la aplicación sin interrupciones realizando simulaciones de apuestas).
Obtención de ganancias/pérdidas correctas cuando la partida acaba en el round 4 y no en el 5 como suele ser lo normal.

Grabación correcta de los datos de las partidas en la base de datos.
Inicialización correcta las variables en comienzo/terminar partida.
No se realiza apuesta cuando no hay el importe mínimo en cartera.
Se realiza una apuesta por el importe establecido en el formulario cuando el algoritmo de recuperación devuelve un importe menor que 4 o mayor que 1000 para conseguir recuperar el dinero perdido en el juego anterior.

16- Mejoras

Debido al tiempo limitado para la fase de implementación en la PEC 3 hubo algunas cosas que fue necesario dejar en el tintero; otras fueron implementadas más escuetamente de lo que se hubiera deseado; y otras son mejoras a realizar a largo plazo, pues algunas son difíciles de implementar. Aquí se presentan algunas de esas ideas de mejora:

- **Modo off-line**: mediante este modo la aplicación juega contra las partidas almacenadas en la base de datos, emulando como si fueran partidas reales on-line, permitiendo de esta manera jugar cientos de partidas en pocos minutos, lo que es ideal para probar nuevas estrategias de juego y así no tener que esperar horas y horas en modo on-line para ver los resultados a medio plazo.
- **Editor de estrategias**: mediante esta posibilidad se permitiría al usuario que fuera capaz de crear sus propias estrategias de apuesta. La idea es compleja pero muy potente. Se trataría de implementar algún sencillo lenguaje de programación interpretado (modo script) que permitiera definir apuestas complejas.
- **Mejoras en el interfaz**: como por ejemplo implementar un nuevo control que mostrase algún tipo de aviso cuando no se pueda realizar una apuesta (como por ejemplo cuando el importe a apostar excede de los límites

fijados por Betfair o cuando el importe a apostar supera el saldo disponible en cuenta).

17- Manual de configuración

La configuración del sistema para poder ejecutar la aplicación es bastante sencilla, tan sólo se necesita los siguientes requisitos:

- Microsoft Visual Studio 2010.
- Microsoft SQL Server 2008 (es posible que funcione con versiones anteriores del SQL Server pero no se ha podido probar).
- Base de datos Poker restaurada.
- Conexión a Internet.
- Una cuenta en Betfair.

Respecto a la cuenta de Betfair, se ha creado una específicamente para este proyecto cuyos datos son:

Nombre de usuario: UOCProyecto
Password: password2011

En cuanto a la base de datos, se adjunta dentro del directorio del proyecto un backup de la base de datos Poker para ser restaurada en un SQL Server en local. El fichero se llama “Poker.bak” y estará en la raíz del directorio “SistemaPoker”.

En el manual de usuario se explica debidamente como restaurar la base de datos, por lo que me remito a ella para este punto.

Una vez restaurada la base de datos se puede ejecutar la aplicación sin problemas, tan sólo será necesario tener una conexión a Internet para que la aplicación se conecte al

servidor de Betfair, validarse con el usuario y password mostrados arriba y empezar el juego.

En cuanto al funcionamiento de la aplicación, también me remito al manual de usuario.

18- Manual de usuario

18.1 Restauración de base de datos

Lo primero que habrá que hacer es restaurar la base de datos Poker que se incluye junto con la aplicación de la siguiente manera:

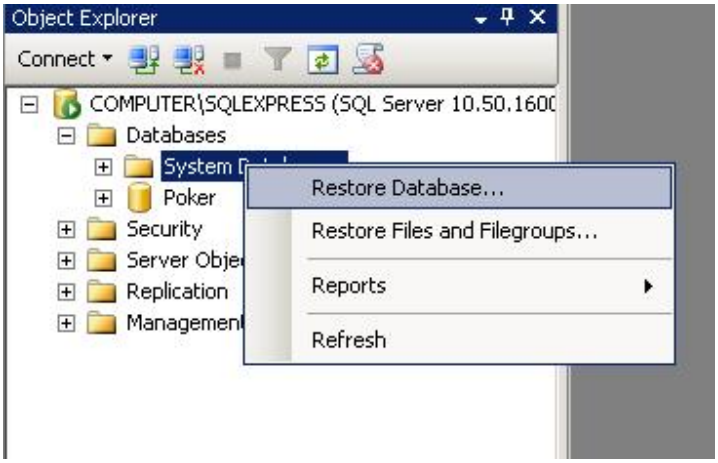


Figura 18 – Seleccionar opción de restaurar la base de datos

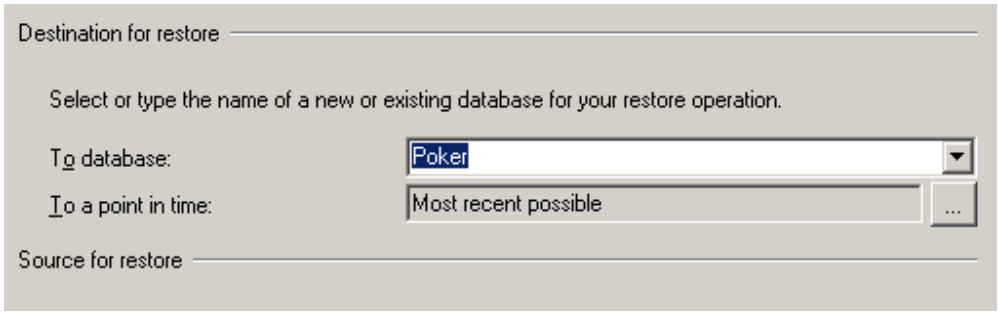


Figura 19 – Seleccionar nombre de la base de datos de destino

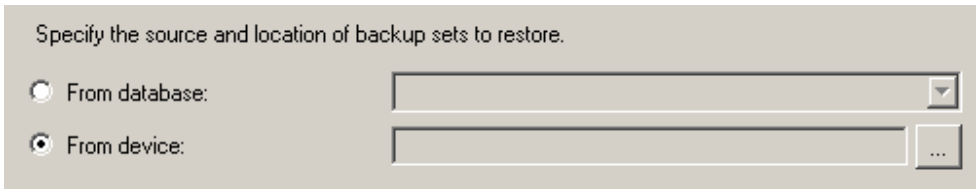


Figura 20 – Seleccionar origen del backup

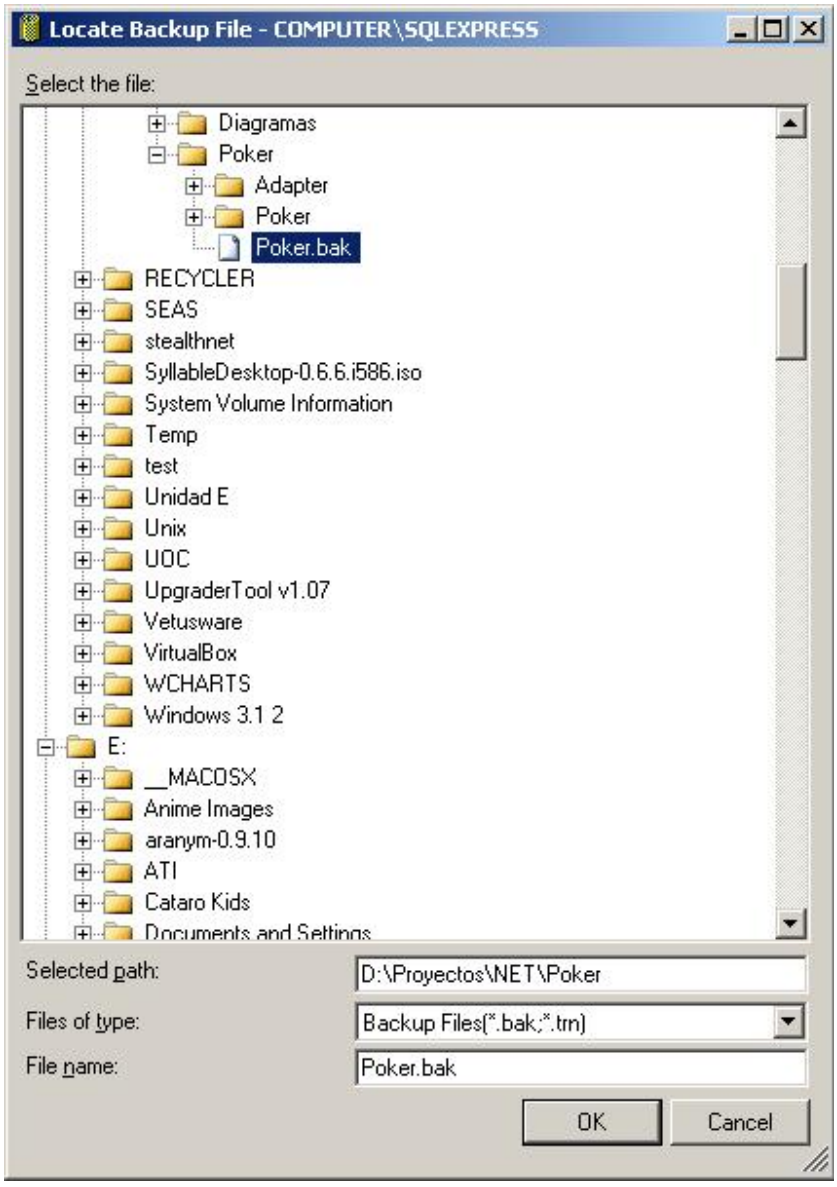


Figura 21 – Seleccionar el fichero de backup

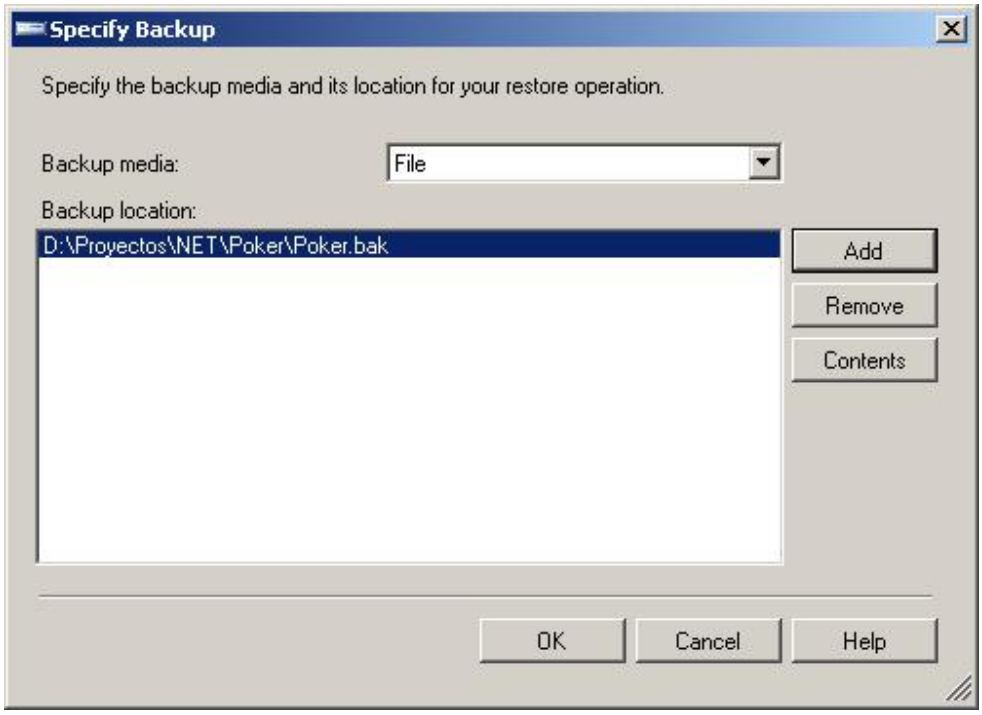


Figura 22 – Confirmar selección de fichero de backup

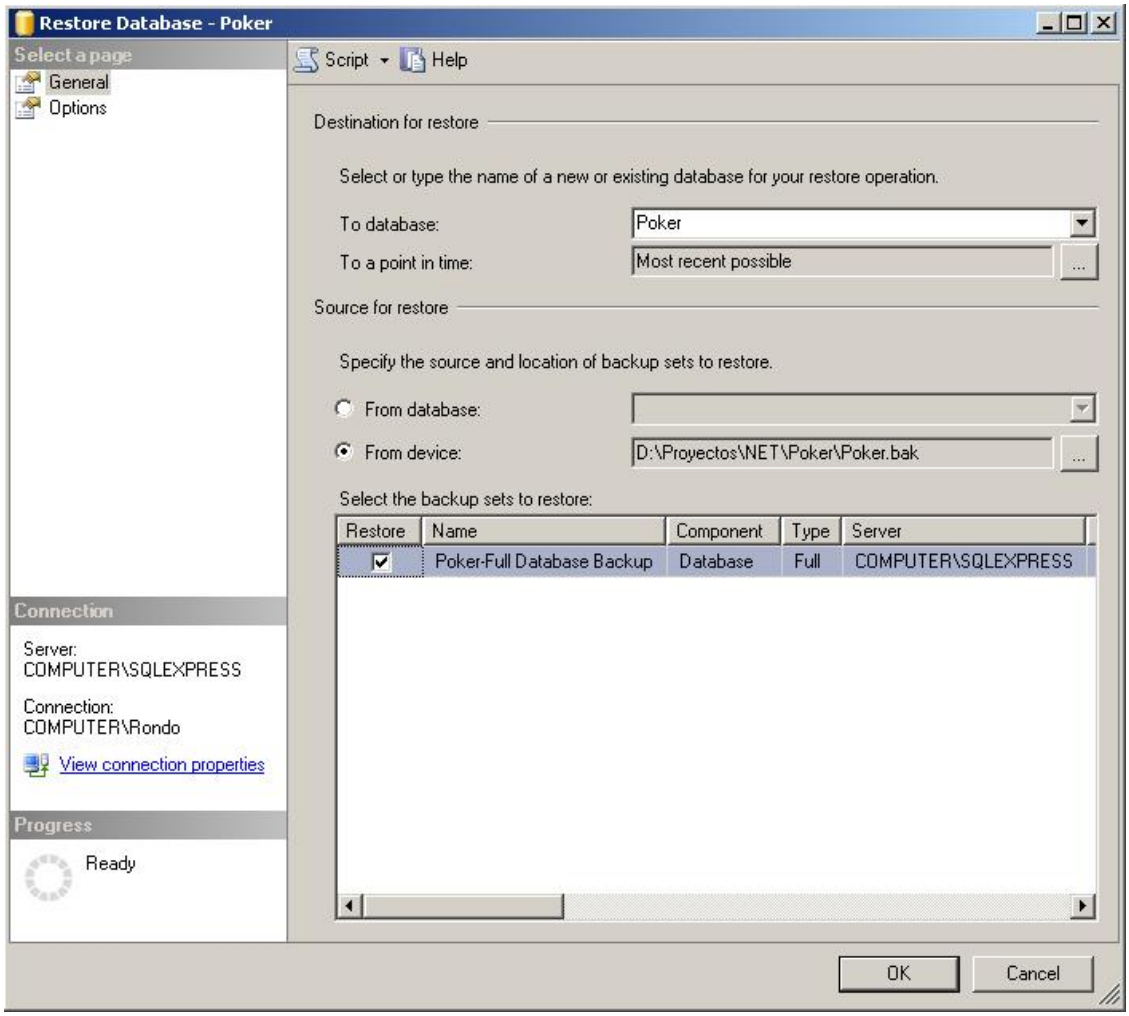


Figura 23 – Selección de backup a restaurar

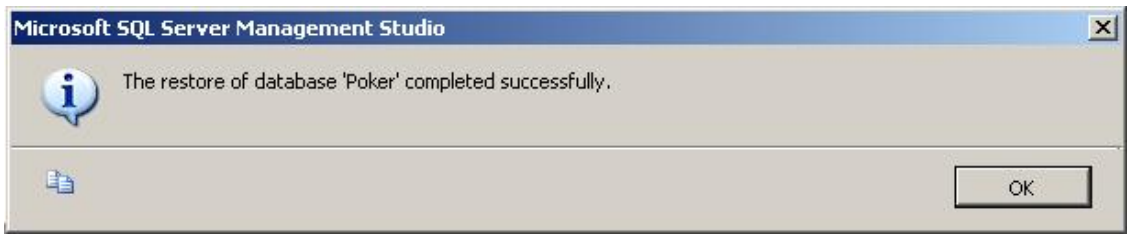


Figura 24 – Base de datos restaurada satisfactoriamente

Figura 25– Base de datos Poker restaurada en el servidor

18.2 Realizar apuestas con la aplicación

Lo primero que saldrá nada más ejecutar la aplicación es una pantalla solicitándonos que nos validemos en Betfair:

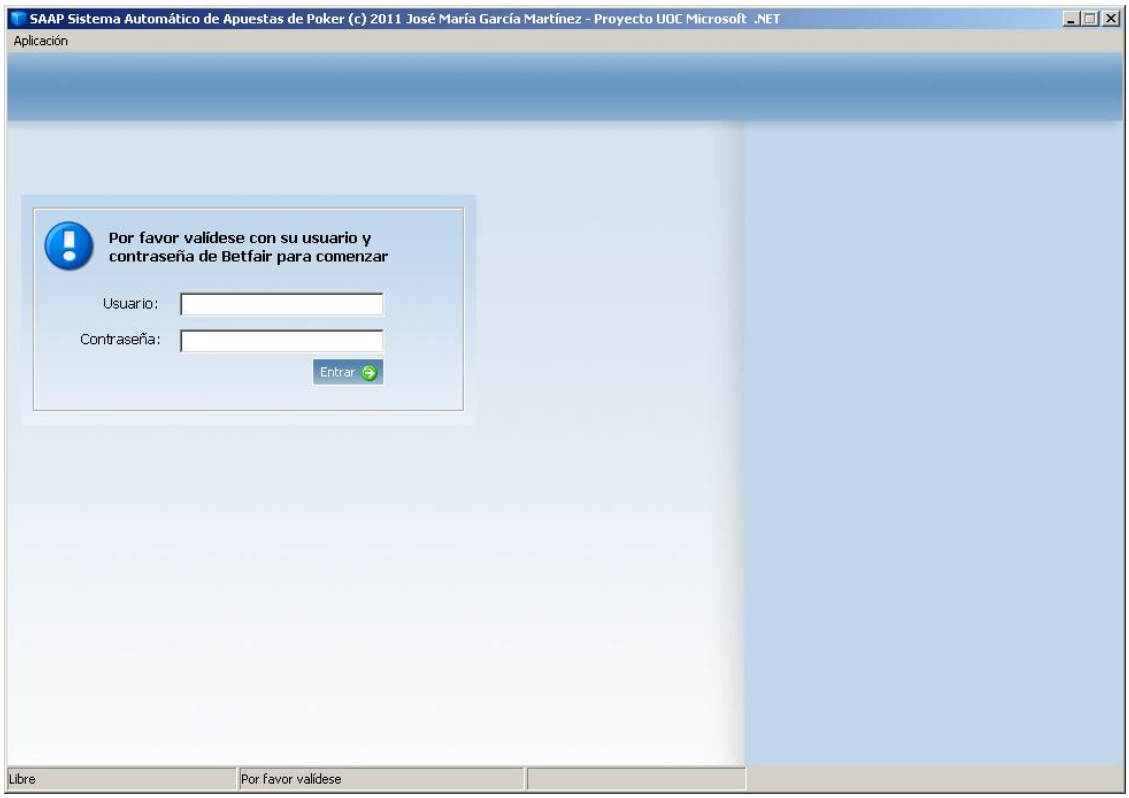


Figura 26 – Validación de usuario

Una vez que el usuario se valide con su nombre y contraseña de su cuenta de Betfair le saldrá la pantalla de apuestas automáticas:

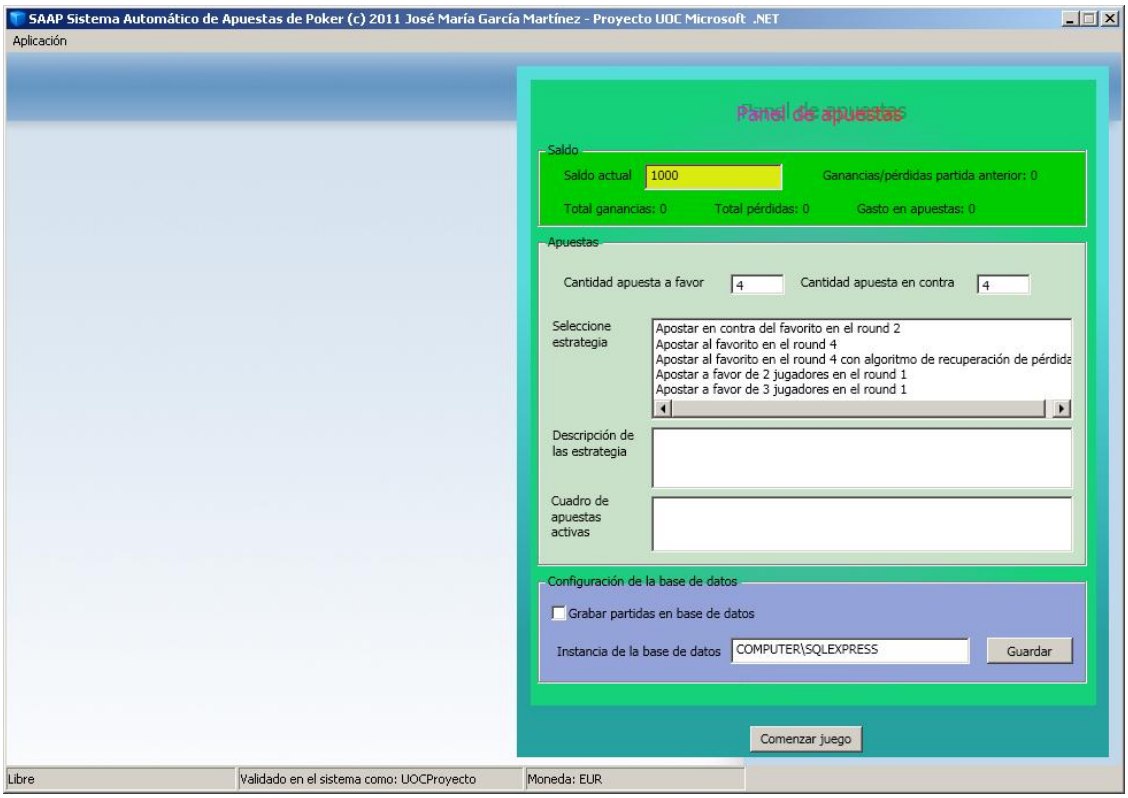


Figura 27 – Control de apuestas

Donde cada una de las opciones indica lo siguiente:

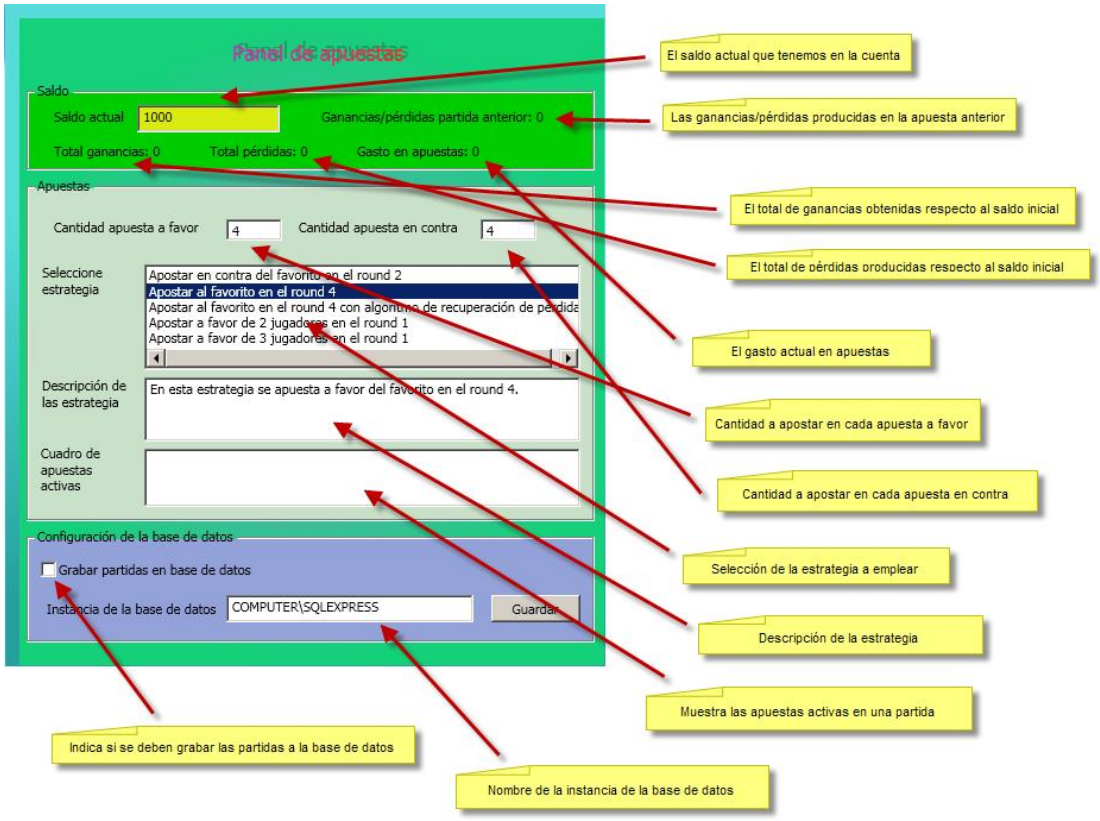


Figura 28 – Detalle del control de apuestas

Una vez que se pulse en el botón de comenzar la partida se mostrará lo siguiente:

Figura 29 – Partida de poker en ejecución

El control de la derecha es el componente WinForm que controla el desarrollo de la partida de poker. Los datos que muestra son los siguientes:

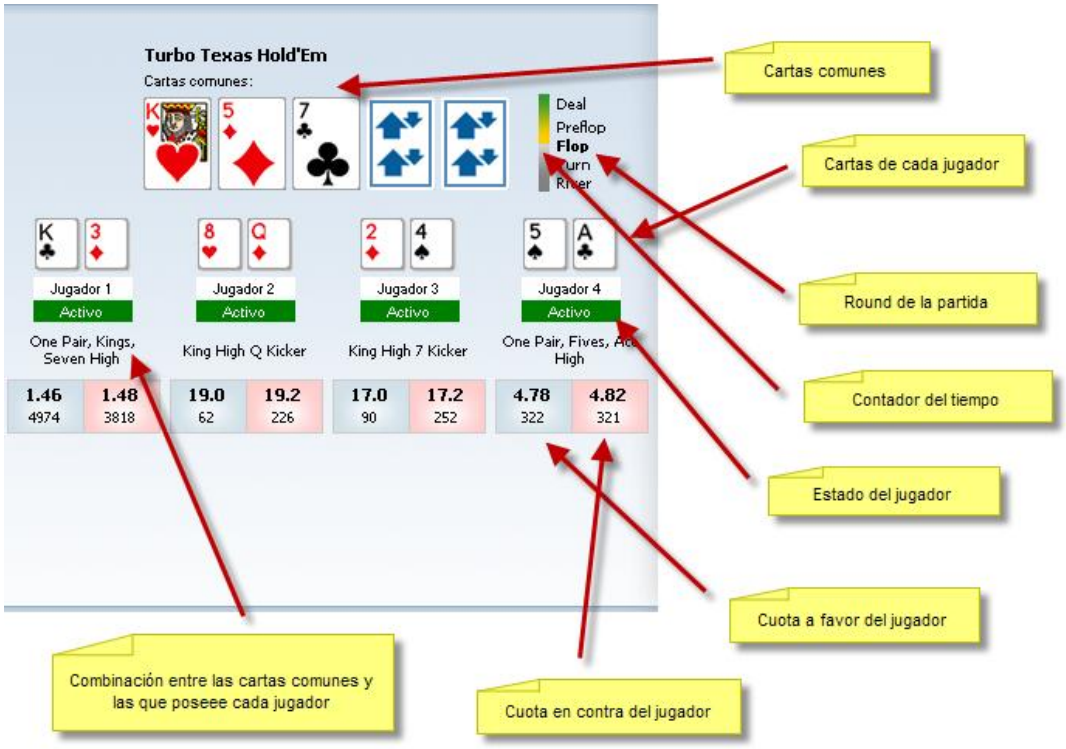


Figura 30 – Descripción del control de poker

El usuario sólo debe seleccionar una estrategia a seguir en el control de apuestas, establecer con que saldo en la cuenta comienza la partida y cuánto debe de ser el importe a usar en las apuestas a favor y en contra. Una vez realizado esto el usuario sólo tiene que pulsar en el botón “comenzar partida” y el sistema automático de apuestas comenzará a funcionar, sin tener que realizar nada más el usuario.

En cuanto a los controles del formulario de apuestas comentar lo siguiente:

- El campo del saldo actual no puede ser negativo.
- Los campos de cantidad apuesta a favor y cantidad apuesta en contra no pueden ser inferiores a 4 ni superiores a 1000 puesto que son los límites de apuestas que marca Betfair.

19- Conclusiones

Aunque no se han podido implementar todas las cosas que se tenían en mente para este proyecto, dado el corto espacio de tiempo para su desarrollo, se ha implementado lo suficiente como para mostrar como es posible crear un sistema automático de apuestas.

También, aunque las estrategias implementadas son bastante sencillas, permiten hacerse una idea de que es posible ganar dinero con este tipo de sistemas, por lo menos a corto plazo.

En cuanto a nivel técnico se ha hecho uso de tecnologías como C#, LINQ y WPF que eran condicionantes para el desarrollo del proyecto.

Por lo tanto podemos decir que, aunque por supuesto todo es mejorable, se han conseguido cumplir los objetivos básicos que se habían propuesto al comienzo del proyecto.

20- Agradecimientos

- A mi consultor, Juan Carlos González Martín, por la ayuda prestada para la realización de este proyecto, y por estar siempre disponible para responder como el rayo cualquier duda o problema que se le plantease.

- A la UOC por permitirme llegar hasta donde he llegado permitiéndome compaginar trabajo, vida familiar y estudios.

*21- **Bibliografía***

Introducción a .NET
Jordi Conesa Caralt, Àngels Rius Gavidia, Jordi Ceballos Villach, David Gañán Jiménez
Editorial UOC, ISBN: 978-84-9788-875-2

Desarrollo de Software Orientado a Objetos
Fatos Xhafa
Editorial UOC, P02/75049/00158

Introducción a la Ingeniería del Software OO
Benet Campderrich Falgueras
Editorial UOC, P01/75007/00565

Diseño Orientado a Objetos
Benet Campderrich Falgueras; Recerca Informàtica, S.L.
Editorial UOC, P01/75007/00570

MSDN
Microsoft
http://msdn.microsoft.com/

WPF Tutorial - Using WPF In WinForms
http://www.switchonthecode.com/tutorials/wpf-tutorial-using-wpf-in-winforms

Tutorial de LINQ to SQL
Thinking in .NET
http://thinkingindotnet.wordpress.com/tutorial-de-linq-to-sql/

Developers Program

Betfair

http://bdp.betfair.com/index.php?option=com_weblinks&catid=60&Itemid=115

Anexo I

Código fuente principal de la aplicación (clase WPFApuestas.xaml.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO;
using System.Reflection;

using Betfair.Gamex.API.Entities;

using Poker.Classes;
using Poker.Controls;
using Poker.Controls.GameView;

//Clases para el control del juego y Linq
using Poker.WPFClasses;

namespace Poker
{
    /// <summary>
    /// Interaction logic for WPFApuestas.xaml
    /// </summary>
    ///

    //Control WPF para simulación de apuestas automáticas
    public partial class WPFApuestas : UserControl
    {
        public Juego objJuego;
        public Jugador J1, J2, J3, J4;
        public double TotalSaldo, TotalGanado, TotalPérdidas, UltimaGanancia, apuestas,
cant_comienzo;
        public double ApuestaFavor, ApuestaContra;
        public int roundId;
        public TexasHoldemView TexasControl;

        //La matriz de apuestas contendrá las apuestas que se realicen en cada partida
        private double[,] MatrizApuestas = new double[4, 4];

        public MainForm FPrincipal;

        //Constructor
        public WPFApuestas()
        {
            InitializeComponent();

            grid1.IsEnabled = true;

            //Creamos las apuestas
            ListEstrategia.Items.Add("Apostar en contra del favorito en el round 2");
            ListEstrategia.Items.Add("Apostar al favorito en el round 4");
            ListEstrategia.Items.Add("Apostar al favorito en el round 4 con algoritmo de
recuperación de pérdidas");
            ListEstrategia.Items.Add("Apostar a favor de 2 jugadores en el round 1");
```

```
            ListEstrategia.Items.Add("Apostar a favor de 3 jugadores en el round 1");


            try
            {
                //Cargamos el nombre de instancia del servidor SQL Server
                StreamReader re = File.OpenText("database.config");
                string input = re.ReadLine();

                TextInstancia.Text = input;
            }
            catch (Exception e)
            {
            }
        }

        //Devuelve si una cadena es un número o no
        //Parámetros: Expression -> la expresión que debe ser parseada
        public bool IsNumeric(object Expression)
        {
            bool isNum;
            double retNum;

            isNum = Double.TryParse(Convert.ToString(Expression),
System.Globalization.NumberStyles.Any,
System.Globalization.NumberFormatInfo.InvariantInfo, out retNum);
            return isNum;
        }


        //Valida el formulario del control verificando que todos los datos han sido
introducidos correctamente
        private Boolean validarForm()
        {
            Boolean ok = false;

            if (!IsNumeric(TextSaldo.Text))
            {
                ok = false;
                MessageBox.Show("El saldo total no es un número");
                TextGP.Focus();
            }
            else
            {
                if (!IsNumeric(TextFavor.Text))
                {
                    ok = false;
                    MessageBox.Show("El importe a apostar a favor no es un número");
                    TextFavor.Focus();
                }
                else
                {
                    if (!IsNumeric(TextContra.Text))
                    {
                        ok = false;
                        MessageBox.Show("El importe a apostar en contra no es un
número");
                        TextContra.Focus();
                    }
                    else
                    {
                        if (Convert.ToDouble(TextFavor.Text) < 4 ||
Convert.ToDouble(TextContra.Text) < 4 ||
                            Convert.ToDouble(TextFavor.Text) > 1000 ||
Convert.ToDouble(TextContra.Text) > 1000)
                        {
                            ok = false;
```

```
                            MessageBox.Show("El importe a apostar a favor y en contra
deben de estar comprendidos entre 4 y 1000");
                            TextFavor.Focus();
                        }
                        else
                        {
                            if (Convert.ToDouble(TextSaldo.Text) < 0)
                            {
                                ok = false;
                                MessageBox.Show("El importe del saldo no puede ser
negativo");
                                TextSaldo.Focus();
                            }
                            else
                            {
                                if (CheckGrabar.IsChecked.Value &&
TextInstancia.Text.Trim() == "")
                                {
                                    ok = false;
                                    MessageBox.Show("La instancia a la base de datos no
puede estar vacía");
                                    TextInstancia.Focus();
                                }
                                else
                                {
                                    ok = true;
                                }
                            }
                        }
                    }
                }

                return ok;
            }

            //Comienza o termina el juego
            private void BtnComenzar_Click(object sender, RoutedEventArgs e)
            {
                if (groupBox1.IsEnabled)
                {
                    if (validarForm())
                    {
                        FPrincipal.StartGame();
                        CrearObjetos();
                        BtnComenzar.Content = "Terminar juego";
                        groupBox1.IsEnabled = false;
                        groupBox2.IsEnabled = false;
                        groupBox3.IsEnabled = false;
                        objJuego.FechaComienzo=Convert.ToDateTime("1900-01-01");
                    }
                }
                else
                {
                    FPrincipal.EndGame();
                    TexasControl = null;
                    objJuego.FechaComienzo = Convert.ToDateTime("1900-01-01");
                    BtnComenzar.Content = "Comenzar juego";
                    groupBox1.IsEnabled = true;
                    groupBox2.IsEnabled = true;
                    groupBox3.IsEnabled = true;
                }
            }

            //Crea e inicializa los objetos necesarios para el proceso de apuestas
            private void CrearObjetos()
            {
                TotalSaldo = Convert.ToDouble(TextSaldo.Text);
```

```
            ApuestaFavor = Convert.ToDouble(TextFavor.Text);
            ApuestaContra = Convert.ToDouble(TextContra.Text);

            TexasControl = (TexasHoldemView)FPrincipal.m_ActiveGameControl;

            TotalGanado = 0;
            TotalPérdidas = 0;
            UltimaGanancia = 0;
            apuestas = 0;
            cant_comienzo = TotalSaldo;

            TextG.Text = "Total ganancias: " +  TotalGanado.ToString();
            TextP.Text = "Total pérdidas: " + TotalPérdidas.ToString();
            TextGP.Text = "Ganancias/pérdidas partida anterior: " +
UltimaGanancia.ToString();
            TextGPCurso.Text = "Gasto en apuestas: " + apuestas.ToString();

            if (TexasControl.roundId!=1) reinicializar();
        }


        //Reinicia los objetos cuando comienza una nueva partida
        private void reinicializar()
        {
            MatrizApuestas = new double[4, 4];

            objJuego = new Juego();

            J1 = new Jugador();
            J2 = new Jugador();
            J3 = new Jugador();
            J4 = new Jugador();

            J1.Jugador1 = 1;
            J2.Jugador1 = 2;
            J3.Jugador1 = 3;
            J4.Jugador1 = 4;

            ListApuestas.Items.Clear();
        }


        //Controla el desarrollo de las apuestas
        //Parámetros: status -> Indica el estado de la partida
        public void ControlApuestas(marketStatusEnum status)
        {
            if (TexasControl.roundId == 1) reinicializar();

            asignarValores();

            if (status != marketStatusEnum.SUSPENDED_GAME_SETTLING)
            {
                switch (TexasControl.roundId)
                {
                    case 1:
                        round1();
                        break;
                    case 2:
                        round2();
                        break;
                    case 3:
                        break;
                    case 4:
                        round4();
                        break;
                }
            }
            else
```

```
            {
                round5();
            }
        }

        //Chequea las cuotas de los jugadores ya que debido al multithread puede que
todavía no se hayan cargado
        public Boolean ChequearQuotas()
        {
            Boolean ok = true;
            double cuota;

            try
            {
                if (TexasControl.player1Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    cuota =
TexasControl.player1Selection.bestAvailableToBackPrices[0].Value;
                    cuota =
TexasControl.player1Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player2Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    cuota =
TexasControl.player2Selection.bestAvailableToBackPrices[0].Value;
                    cuota =
TexasControl.player2Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player3Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    cuota =
TexasControl.player3Selection.bestAvailableToBackPrices[0].Value;
                    cuota =
TexasControl.player3Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player4Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    cuota =
TexasControl.player4Selection.bestAvailableToBackPrices[0].Value;
                    cuota =
TexasControl.player4Selection.bestAvailableToLayPrices[0].Value;
                }
            }
            catch (Exception e)
            {
                ok = false;
            }

            return ok;
        }

        //Asigna los valoes de las cartas y las cuotas a los objetos en memoria
        private void asignarValores()
        {
            gameDataObject community = TexasControl.gameData[4];

            objJuego.Carta1 = community.property[0].value;
            objJuego.Carta2 = community.property[1].value;
            objJuego.Carta3 = community.property[2].value;

            if (community.property[3].value != "NOT AVAILABLE")
            {
                objJuego.Carta4 = community.property[3].value;
            }
            else
```

```
            {
                objJuego.Carta4 = "-1";
            }

            if (community.property[4].value != "NOT AVAILABLE")
            {
                objJuego.Carta5 = community.property[4].value;
            }
            else
            {
                objJuego.Carta5 = "-1";
            }

            J1.Carta1 = TexasControl.gameData[0].property[0].value;
            J1.Carta2 = TexasControl.gameData[0].property[1].value;

            J2.Carta1 = TexasControl.gameData[1].property[0].value;
            J2.Carta2 = TexasControl.gameData[1].property[1].value;

            J3.Carta1 = TexasControl.gameData[2].property[0].value;
            J3.Carta2 = TexasControl.gameData[2].property[1].value;

            J4.Carta1 = TexasControl.gameData[3].property[0].value;
            J4.Carta2 = TexasControl.gameData[3].property[1].value;

            roundId = TexasControl.roundId;

            if (roundId == 1)
            {
                J1.CuotaFavor1 =
TexasControl.player1Selection.bestAvailableToBackPrices[0].Value;
                J1.CuotaContra1 =
TexasControl.player1Selection.bestAvailableToLayPrices[0].Value;

                J2.CuotaFavor1 =
TexasControl.player2Selection.bestAvailableToBackPrices[0].Value;
                J2.CuotaContra1 =
TexasControl.player2Selection.bestAvailableToLayPrices[0].Value;

                J3.CuotaFavor1 =
TexasControl.player3Selection.bestAvailableToBackPrices[0].Value;
                J3.CuotaContra1 =
TexasControl.player3Selection.bestAvailableToLayPrices[0].Value;

                J4.CuotaFavor1 =
TexasControl.player4Selection.bestAvailableToBackPrices[0].Value;
                J4.CuotaContra1 =
TexasControl.player4Selection.bestAvailableToLayPrices[0].Value;
            }

            if (roundId == 2)
            {
                J1.CuotaFavor2 =
TexasControl.player1Selection.bestAvailableToBackPrices[0].Value;
                J1.CuotaContra2 =
TexasControl.player1Selection.bestAvailableToLayPrices[0].Value;

                J2.CuotaFavor2 =
TexasControl.player2Selection.bestAvailableToBackPrices[0].Value;
                J2.CuotaContra2 =
TexasControl.player2Selection.bestAvailableToLayPrices[0].Value;

                J3.CuotaFavor2 =
TexasControl.player3Selection.bestAvailableToBackPrices[0].Value;
                J3.CuotaContra2 =
TexasControl.player3Selection.bestAvailableToLayPrices[0].Value;
```

```
                J4.CuotaFavor2 =
TexasControl.player4Selection.bestAvailableToBackPrices[0].Value;
                J4.CuotaContra2 =
TexasControl.player4Selection.bestAvailableToLayPrices[0].Value;
            }

            if (roundId == 3)
            {
                if (TexasControl.player1Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    J1.CuotaFavor3 =
TexasControl.player1Selection.bestAvailableToBackPrices[0].Value;
                    J1.CuotaContra3 =
TexasControl.player1Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player2Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    J2.CuotaFavor3 =
TexasControl.player2Selection.bestAvailableToBackPrices[0].Value;
                    J2.CuotaContra3 =
TexasControl.player2Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player3Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    J3.CuotaFavor3 =
TexasControl.player3Selection.bestAvailableToBackPrices[0].Value;
                    J3.CuotaContra3 =
TexasControl.player3Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player4Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    J4.CuotaFavor3 =
TexasControl.player4Selection.bestAvailableToBackPrices[0].Value;
                    J4.CuotaContra3 =
TexasControl.player4Selection.bestAvailableToLayPrices[0].Value;
                }
            }

            if (roundId == 4)
            {
                if (TexasControl.player1Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    J1.CuotaFavor4 =
TexasControl.player1Selection.bestAvailableToBackPrices[0].Value;
                    J1.CuotaContra4 =
TexasControl.player1Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player2Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    J2.CuotaFavor4 =
TexasControl.player2Selection.bestAvailableToBackPrices[0].Value;
                    J2.CuotaContra4 =
TexasControl.player2Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player3Selection.status == selectionStatusEnum.IN_PLAY)
                {
                    J3.CuotaFavor4 =
TexasControl.player3Selection.bestAvailableToBackPrices[0].Value;
                    J3.CuotaContra4 =
TexasControl.player3Selection.bestAvailableToLayPrices[0].Value;
                }

                if (TexasControl.player4Selection.status == selectionStatusEnum.IN_PLAY)
```

```
                {
                    J4.CuotaFavor4 =
TexasControl.player4Selection.bestAvailableToBackPrices[0].Value;
                    J4.CuotaContra4 =
TexasControl.player4Selection.bestAvailableToLayPrices[0].Value;
                }
            }
        }

        //Reinicia la matriz de apuestas
        private void reiniciarMatriz()
        {
            MatrizApuestas = new double[4, 4];
        }


        //Calcula si la apuestas puede ser realizada en función del saldo disponible en
la cartera
        //Parámetros: TotalSaldo: el saldo total disponible en la cuenta
        //            Apuestas: el importe a apostar
        //            Cuota: la cuota del jugador
        private Boolean QuedaSaldo(double TotalSaldo, double Apuesta, double Cuota = 0)
        {
            if (Apuesta > 0)
            {
                if ((TotalSaldo - Apuesta) > 0)
                    return true;
                else
                    return false;
            }
            else
            {
                if (TotalSaldo + (Apuesta * Cuota) > 0)
                    return true;
                else
                    return false;
            }
        }

        //Realiza una apuesta
        //Parámetros: i -> fila de la matriz que indica el jugador
        //            a -> columna de la matriz que indica el round en el que se debe
apostar
        //            importe -> la cantidad a apostar
        private void Apostar(int i, int a, double importe)
        {
            string cadena="";
            double cuota=obtenerCuota(i,a,importe);

            if (QuedaSaldo(TotalSaldo, importe, cuota))
            {
                MatrizApuestas[i, a] = importe;

                TotalSaldo -= Math.Abs(importe);
                apuestas -= Math.Abs(importe);

                TextP.Text = "Total pérdidas: " + Math.Round(TotalPérdidas).ToString();
                TextSaldo.Text = Math.Round(TotalSaldo,2).ToString();
                TextGPCurso.Text = "Gasto en apuestas: " +
Math.Round(apuestas,2).ToString();

                cadena = Math.Abs(importe).ToString() +" euros apostados a ";

                cadena += cuota.ToString();

                if (importe>0)
                    cadena+=" a favor ";
                else
```

```
                cadena+=" en contra ";

            cadena+="en el round " + roundId + " al jugador " + (i+1);

            ListApuestas.Items.Add (cadena);
        }
    }

    //Realiza las apuestas en el round1
    private void round1()
    {
        string estrategia = (string)ListEstrategia.SelectedItem;

        reiniciarMatriz();

        apuestas = 0;
        TextGPCurso.Text = "Gasto en apuestas: " + apuestas.ToString();

        objJuego.FechaComienzo = DateTime.Now;

        switch (estrategia)
        {
            case "Apostar a favor de 2 jugadores en el round 1":
                Apostar(0, 0, ApuestaFavor);
                Apostar(1, 0, ApuestaFavor);
                break;
            case "Apostar a favor de 3 jugadores en el round 1":
                Apostar(0, 0, ApuestaFavor);
                Apostar(1, 0, ApuestaFavor);
                Apostar(2, 0, ApuestaFavor);
                break;
        }
    }

    //Realiza las apuestas en el round2
    private void round2()
    {
        string estrategia = (string)ListEstrategia.SelectedItem;

        switch (estrategia)
        {
            case "Apostar en contra del favorito en el round 2":
                Apostar(Convert.ToInt32(ObtenerFavorito(true)), 1, -ApuestaContra);
                break;
        }
    }

    //Realiza las apuestas en el round4
    private void round4()
    {
        string estrategia = (string)ListEstrategia.SelectedItem;

        switch (estrategia)
        {
            case "Apostar al favorito en el round 4":
                Apostar(Convert.ToInt32(ObtenerFavorito(true)), 3, ApuestaFavor);
                break;
            case "Apostar al favorito en el round 4 con algoritmo de recuperación de
pérdidas":
                Apostar(Convert.ToInt32(ObtenerFavorito(true)), 3,
AlgoritmoRecuperacionPerdida(ObtenerFavorito(false)));
                break;
        }
    }

    //Calcula mediante una ecuación lineal el importe que se debe apostar dada una
cuota para recuperar el dinero
    //perdido en el anterior juego
```

```
        //Parámetros: cuota -> la cuota del jugador del que se quiere realizar la apuesta
        private double AlgoritmoRecuperacionPerdida(double cuota)
        {
            //Apuesta mínima en BetFair
            double resultado = 4;

            if (UltimaGanancia < 0)
            {

                /*Ecuación líneal que viene de x * cuota = x + ultima_pérdida para la
obtención del dinero perdido
                  a la que además se le debe de sumar el 2.5% que se quedaría Betfair de
beneficio*/
                resultado = (Math.Abs(UltimaGanancia) / (cuota - 1)) +
(((Math.Abs(UltimaGanancia) / (cuota - 1)) * 2.5) / 100);

                /*Si la apuesta es <4 ó >1000 Betfair no dejará hacerla, por lo que la
igualamos a 4 para que haga la apuesta
                  mínima*/
                if (resultado < 4 || resultado > 1000) resultado = 4;
            }

            return Math.Round(resultado,2);
        }

        //Obtiene el jugador favorito en el round en curso
        //Parámetros: jug_favorito -> indica si lo que se ha de retornar es el número del
jugador favorito o su cuota
        private double ObtenerFavorito(Boolean jug_favorito)
        {
            double cuota = 1000000;
            int favorito = 0;

            if (TexasControl.player1Selection.status == selectionStatusEnum.IN_PLAY &&
                TexasControl.player1Selection.bestAvailableToBackPrices[0].Value < cuota)
            {
                cuota = TexasControl.player1Selection.bestAvailableToBackPrices[0].Value;
                favorito = 0;
            }

            if (TexasControl.player2Selection.status == selectionStatusEnum.IN_PLAY &&
                TexasControl.player2Selection.bestAvailableToBackPrices[0].Value < cuota)
            {
                cuota = TexasControl.player2Selection.bestAvailableToBackPrices[0].Value;
                favorito = 1;
            }

            if (TexasControl.player3Selection.status == selectionStatusEnum.IN_PLAY &&
                TexasControl.player3Selection.bestAvailableToBackPrices[0].Value < cuota)
            {
                cuota = TexasControl.player3Selection.bestAvailableToBackPrices[0].Value;
                favorito = 2;
            }

            if (TexasControl.player4Selection.status == selectionStatusEnum.IN_PLAY &&
                TexasControl.player4Selection.bestAvailableToBackPrices[0].Value < cuota)
            {
                cuota = TexasControl.player4Selection.bestAvailableToBackPrices[0].Value;
                favorito = 3;
            }

            if (jug_favorito)
                return favorito;
            else
                return cuota;
        }

        //Obtiene la cuota de un jugador en un round concreto
```

```
//Parámetros: i -> indica el jugador
//            a -> indica el round
private double obtenerCuota(int i, int a)
{
    return obtenerCuota(i, a, MatrizApuestas[i, a]);
}

//Obtiene la cuota de un jugador en un round concreto
//Parámetros: i -> indica el jugador
//            a -> indica el round
//            importe -> indica si ha de obtener la cuota a favor o en contra
private double obtenerCuota(int i, int a, double importe)
{
    var dValue = J1.CuotaFavor1;

    if (importe > 0)
    {
        switch (i)
        {
            case 0:
                switch (a)
                {
                    case 0:
                        dValue = J1.CuotaFavor1;
                        break;
                    case 1:
                        dValue = J1.CuotaFavor2;
                        break;
                    case 2:
                        dValue = J1.CuotaFavor3;
                        break;
                    case 3:
                        dValue = J1.CuotaFavor4;
                        break;
                }
                break;
            case 1:
                switch (a)
                {
                    case 0:
                        dValue = J2.CuotaFavor1;
                        break;
                    case 1:
                        dValue = J2.CuotaFavor2;
                        break;
                    case 2:
                        dValue = J2.CuotaFavor3;
                        break;
                    case 3:
                        dValue = J2.CuotaFavor4;
                        break;
                }
                break;
            case 2:
                switch (a)
                {
                    case 0:
                        dValue = J3.CuotaFavor1;
                        break;
                    case 1:
                        dValue = J3.CuotaFavor2;
                        break;
                    case 2:
                        dValue = J3.CuotaFavor3;
                        break;
                    case 3:
                        dValue = J3.CuotaFavor4;
                        break;
```

```
                }
                break;
            case 3:
                switch (a)
                {
                    case 0:
                        dValue = J4.CuotaFavor1;
                        break;
                    case 1:
                        dValue = J4.CuotaFavor2;
                        break;
                    case 2:
                        dValue = J4.CuotaFavor3;
                        break;
                    case 3:
                        dValue = J4.CuotaFavor4;
                        break;
                }
                break;
        }
    }
    else
    {
        switch (i)
        {
            case 0:
                switch (a)
                {
                    case 0:
                        dValue = J1.CuotaContra1;
                        break;
                    case 1:
                        dValue = J1.CuotaContra2;
                        break;
                    case 2:
                        dValue = J1.CuotaContra3;
                        break;
                    case 3:
                        dValue = J1.CuotaContra4;
                        break;
                }
                break;
            case 1:
                switch (a)
                {
                    case 0:
                        dValue = J2.CuotaContra1;
                        break;
                    case 1:
                        dValue = J2.CuotaContra2;
                        break;
                    case 2:
                        dValue = J2.CuotaContra3;
                        break;
                    case 3:
                        dValue = J2.CuotaContra4;
                        break;
                }
                break;
            case 2:
                switch (a)
                {
                    case 0:
                        dValue = J3.CuotaContra1;
                        break;
                    case 1:
                        dValue = J3.CuotaContra2;
                        break;
```

```
                        case 2:
                            dValue = J3.CuotaContra3;
                            break;
                        case 3:
                            dValue = J3.CuotaContra4;
                            break;
                    }
                    break;
                case 3:
                    switch (a)
                    {
                        case 0:
                            dValue = J4.CuotaContra1;
                            break;
                        case 1:
                            dValue = J4.CuotaContra2;
                            break;
                        case 2:
                            dValue = J4.CuotaContra3;
                            break;
                        case 3:
                            dValue = J4.CuotaContra4;
                            break;
                    }
                    break;
            }
        }

        return Convert.ToDouble(dValue);
    }

    //Esta función calcula el resultado de las apuestas cuando concluye el juego, sea
éste en el round 5 como es normal o
    //en el round 4 como algunas veces puede ocurrir
    private void round5()
    {
        int number_winners = 0;
        selectionStatusEnum status = 0;
        double dValue = 0;
        double current_perdidas = 0, current_ganancias = 0;

        J1.ResultadoId = Convert.ToInt32(TexasControl.player1Selection.status);
        J2.ResultadoId = Convert.ToInt32(TexasControl.player2Selection.status);
        J3.ResultadoId = Convert.ToInt32(TexasControl.player3Selection.status);
        J4.ResultadoId = Convert.ToInt32(TexasControl.player4Selection.status);

        //Si no hay ninguna apuesta en el sistema no tenemos que calcular nada
        if (apuestas != 0)
        {
            if (TexasControl.player1Selection.status == selectionStatusEnum.WINNER ||
                TexasControl.player1Selection.status ==
selectionStatusEnum.TIED_DEAD_HEAT)

                number_winners++;

            if (TexasControl.player2Selection.status == selectionStatusEnum.WINNER ||
                TexasControl.player2Selection.status ==
selectionStatusEnum.TIED_DEAD_HEAT)

                number_winners++;

            if (TexasControl.player3Selection.status == selectionStatusEnum.WINNER ||
                TexasControl.player3Selection.status ==
selectionStatusEnum.TIED_DEAD_HEAT)

                number_winners++;

            if (TexasControl.player4Selection.status == selectionStatusEnum.WINNER ||
```

```
                    TexasControl.player4Selection.status ==
selectionStatusEnum.TIED_DEAD_HEAT)

                    number_winners++;

                for (var i = 0; i < MatrizApuestas.GetLength(1); i++)
                {
                    for (var a = 0; a < MatrizApuestas.GetLength(0); a++)
                    {
                        if (MatrizApuestas[i, a] != 0)
                        {
                            switch (i)
                            {
                                case 0:
                                    status = TexasControl.player1Selection.status;
                                    break;
                                case 1:
                                    status = TexasControl.player2Selection.status;
                                    break;
                                case 2:
                                    status = TexasControl.player3Selection.status;
                                    break;
                                case 3:
                                    status = TexasControl.player4Selection.status;
                                    break;
                            }

                            dValue = obtenerCuota(i, a);

                            if (MatrizApuestas[i, a] > 0)
                            {
                                if (status == selectionStatusEnum.WINNER ||
                                    status == selectionStatusEnum.TIED_DEAD_HEAT)
                                {
                                    //Betfair se lleva un 2.50% de beneficio de la
ganancia obtenida
                                    //Fórmula: (importe apostado * (cuota/número de
ganadores)) - 2.50%
                                    current_ganancias += ((MatrizApuestas[i, a] * (dValue
/ number_winners)) - ((MatrizApuestas[i, a] * (dValue / number_winners) * 2.50) / 100));
// +MatrizApuestas[i, a];
                                }
                                else
                                {
                                    //No hace falta porque descontamos la apuesta cuando
se realiza
                                    //current_perdidas -= MatrizApuestas[i, a];

                                    //Si hemos perdido la apuesta se suma al saldo de
dinero pérdido.
                                    //TotalPérdidas += MatrizApuestas[i, a];
                                }
                            }
                            else
                            {
                                if (status == selectionStatusEnum.LOSER)
                                {
                                    //Betfair se lleva un 2.50% de beneficio de la
ganancia obtenida
                                    //Fórmula: importe apostado + (importe apostado -
2.50%)
                                    current_ganancias += Math.Abs(MatrizApuestas[i, a]) +
(Math.Abs(MatrizApuestas[i, a]) - ((Math.Abs(MatrizApuestas[i, a]) * 2.50) / 100));
                                }
                                else
                                {
                                    //Fórmula: (importe apostado * cuota) - importa
apostado
```

```
                                    current_perdidas += (MatrizApuestas[i, a] * dValue) -
MatrizApuestas[i, a];
                                }
                            }
                        }
                    }
                }


                if (current_ganancias + current_perdidas != 0)
                {
                    UltimaGanancia = TotalSaldo;
                    TotalSaldo += current_ganancias + current_perdidas;
                    UltimaGanancia = TotalSaldo - UltimaGanancia;
                }
                else
                {
                    UltimaGanancia = apuestas;
                }

                if (TotalSaldo - cant_comienzo > 0)
                {
                    TotalGanado = TotalSaldo - cant_comienzo; //+= current_ganancias;
                    TotalPérdidas = 0;
                }
                else
                {
                    TotalPérdidas = TotalSaldo - cant_comienzo; //-= current_perdidas;
                    TotalGanado = 0;
                }

                TextSaldo.Text = Math.Round(TotalSaldo, 2).ToString();
                TextGP.Text = "Ganancias/pérdidas partida anterior: " +
Math.Round(UltimaGanancia, 2).ToString();
                TextGPCurso.Text = "Gasto en apuestas: 0";
                TextP.Text = "Total pérdidas: " + Math.Round(TotalPérdidas,
2).ToString();
                TextG.Text = "Total ganancias: " + Math.Round(TotalGanado, 2).ToString();
                ListApuestas.Items.Clear();
            }

            GrabarJuego();
        }

        //Graba la partida en la base de datos
        private void GrabarJuego()
        {
            // Northwnd inherits from System.Data.Linq.DataContext.
            //Po nw = new Juego1(@"Poker.mdf");
            //SQLClasesDataContext Poker = new SQLClasesDataContext(@"Poker.mdf");

            if ((objJuego.FechaComienzo != Convert.ToDateTime("1900-01-01")) &&
CheckGrabar.IsChecked.Value)
            {
                SQLClasesDataContext Poker;

                try
                {
                   Poker = new SQLClasesDataContext("Server=" + TextInstancia.Text +
";Database=Poker;Trusted_Connection=True");

                    objJuego.FechaFin = DateTime.Now;

                    Poker.Juegos.InsertOnSubmit(objJuego);

                    Poker.SubmitChanges();

                    J1.JuegoId = objJuego.JuegoId;
```

```
                    J2.JuegoId = objJuego.JuegoId;
                    J3.JuegoId = objJuego.JuegoId;
                    J4.JuegoId = objJuego.JuegoId;

                    Poker.Jugadors.InsertOnSubmit(J1);
                    Poker.Jugadors.InsertOnSubmit(J2);
                    Poker.Jugadors.InsertOnSubmit(J3);
                    Poker.Jugadors.InsertOnSubmit(J4);

                    Poker.SubmitChanges();
                }
                catch (Exception e)
                {
                    MessageBox.Show("Se produjo un fallo al intentar grabar la partida en
la base de datos." +
                                    "Compruebe que el nombre de la instancia es correcta
y que en ella existe la base de datos" +
                                    "Poker. Si este error sigue ocurriendo desactive por
favor la opción de grabar la partida en la base de datos.");
                }

            }
        }

        //Muestra la descripción de cada estrategia
        private void ListEstrategia_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            string estrategia = (string)ListEstrategia.SelectedItem;
            Stream s = null;

            Assembly a = Assembly.Load("Poker");

            switch (estrategia)
            {
                case "Apostar a favor de 2 jugadores en el round 1":
                    s =
a.GetManifestResourceStream("Poker.WPFClasses.Resources.2jug_r_1.txt");
                    break;
                case "Apostar a favor de 3 jugadores en el round 1":
                    s =
a.GetManifestResourceStream("Poker.WPFClasses.Resources.3jug_r_1.txt");
                    break;
                case "Apostar en contra del favorito en el round 2":
                    s =
a.GetManifestResourceStream("Poker.WPFClasses.Resources.contra_fav_r_2.txt");
                    break;
                case "Apostar al favorito en el round 4":
                    s =
a.GetManifestResourceStream("Poker.WPFClasses.Resources.fav_r_4.txt");
                    break;
                case "Apostar al favorito en el round 4 con algoritmo de recuperación de
pérdidas":
                    s =
a.GetManifestResourceStream("Poker.WPFClasses.Resources.fav_r_4_algo.txt");
                    break;
            }

            if (s != null)
            {
                using (Stream stream = s)
                {
                    byte[] buffer = new byte[stream.Length];
                    stream.Read(buffer, 0, buffer.Length);
                    TextDescripcion.Text = Encoding.Default.GetString(buffer);
                }
            }
        }
```

```
        //Guarda el nombre de la instancia de SQL Server en un fichero
        private void BtnGuardar_Click(object sender, RoutedEventArgs e)
        {
            System.IO.StreamWriter file = new System.IO.StreamWriter("database.config");

            try
            {
                file.WriteLine(TextInstancia.Text);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Se produjo un error al intentar grabar el fichero de
configuración.");
            }
            finally
            {
                file.Close();
            }
        }
    }
}
```