UNIVERSIDADE DE VIGO

ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Memoria do Traballo de Fin de Grao que presenta

D. Santiago Andrés del Mazo

para a obtención do Título de graduado en Enxeñaría Informática

Ferramenta de trading empregando a API de Betfair

Xuño, 2018

Traballo de Fin de Grao Nº: EI-16/17-015

Titor/a: Francisco José Ribadas Pena

Área de coñecemento: Ciencias da computación e intelixencia artificial

Departamento: Informática.

Contenido

Introducción

Contexto y motivación

Desde hace unos pocos años la proliferación de negocios relacionados con las apuestas deportivas ha sido meteórica. En cualquier evento de índole deportivo encontramos referencias a ellas. Ya sea en forma de patrocinios, anuncios, etc.

Existen básicamente 3 tipos de casas de apuestas en la actualidad, que se definirán a continuación:

1. **Las casas de apuestas tradicionales.**
   Las tradicionales son el tipo más básico y son las que cualquier persona, ya sea profana o no, definiría si se le pregunta cómo funciona una casa de apuestas.
   Cuando un usuario decide apostar, es la casa quien acepta la apuesta y es la que promete pagar si la selección hecha por el usuario resulta acertada. En este tipo de negocio, es la casa quien marca la cuota a la que está dispuesta a aceptar la apuesta, sin posibilidad de negociación.
2. **Las casas de apuestas de intercambio.**
   Este segundo tipo es un poco más complejo de entender y salvo que la persona esté interesada en este mundo, no se suele conocer esta posibilidad.
   En este caso la casa de apuestas actúa de mero intermediario entre sus usuarios. Cuando se realiza una apuesta, esta no se produce contra la casa, si no que entra en un mercado en el cual espera a ser igualada por una apuesta contraria a la que el usuario realizó. ¿Qué gana la casa de apuestas entonces? Un porcentaje sobre las ganancias de la apuesta ganadora, que suele rondar el 5%. La gran ventaja es que es el usuario quien fija la cuota y será otro usuario el que decida si le parece justo o no aceptar una apuesta a esa cuota. El mercado se autorregula de forma completamente autónoma.
3. **Las casas de apuestas americanas.**
   Este tipo es el más desconocido por la gente de a pie. Funcionan de forma similar a las tradicionales, pero ofreciendo generalmente las mejores cuotas. Como las ganancias por parte de la casa son menores, existen restricciones para atenuar el efecto de cantidades ingentes de usuarios apostando a una determinada selección.

Este Trabajo Fin de Grado (TFG) se ha centrado en el caso de las apuestas de intercambio. Las casas de apuestas de intercambio tienen la particularidad de que a la casa le da exactamente igual quien gane la apuesta ya que ella sacará beneficio tanto si es uno el ganador como si es el otro. Esto ha provocado que lo único que les interese a las casas sea tener un volumen alto de apuestas y se han dado cuenta de que la mejor manera de lograrlo es facilitando la mayor cantidad de información a los usuarios.

Muchas las casas de apuestas de intercambio más populares ofrecen una API (Application Programming Interface) de programación contra la que realizar las mismas operaciones que se pueden realizar en sus aplicaciones web: listar mercados, realizar apuestas, cancelar apuestas, ver el historial, etc. A su vez, crear una aplicación propia basada en una API permite realizar tareas como: análisis de mercados por parte del

propio software, recolección de información, automatización de procesos, cotejar datos con los de otras fuentes de manera fácil, etc.

En el caso de este TFG lo que se ha desarrollado es una aplicación de escritorio que emula de manera sencilla las funcionalidades básicas que ofrece la web de la casa de apuestas Betfair Exchange y que permite la realización de apuestas de manera más rápida y cómoda.

API de Betfair

Betfair presenta 5 tipos de API dependiendo del tipo de aplicación que se quiera programar [1]:

1. **Exchange API.**
   Este tipo de API es la destinada a dar acceso a los distintos mercados de intercambio. Existen 2 alternativas para trabajar con ella:
   1. **JSON-RPC.**
      Se trata de un protocolo de llamada a procedimientos remotos.
   2. **JSON REST.**
      Se trata de una arquitectura, que apoyándose en el protocolo HTTP, proporciona una API de programación en la que se pueden usar los distintos métodos (GET, POST, PUT, DELETE, etc.) para así realizar distintas operaciones e intercambio de información entre el servicio web y el cliente.
2. **Exchange Historical Data [2].**
   Ofrece la posibilidad de recuperar información pasada de los mercados, como, por ejemplo, la evolución de las cuotas en un determinado evento desde su creación hasta su finalización.
3. **Games API.**
   Este tipo de API está destinada a la creación de aplicaciones que usen los distintos juegos de azar que alberga Betfair: blackjack, casino, póker, etc.
4. **Timeform API.**
   Se usa única y exclusivamente para las carreras de caballos. Permite recuperar información pasada, para así crear algoritmos y hacer predicciones.
5. **Vendor Web Apps API.**
   Sirve para crear aplicaciones web comerciales similares a la de Betfair. Sería una forma de montar tu propia casa de apuestas apoyándote en la de Betfair.

---

[1] https://developer.betfair.com/

[2] http://docs.developer.betfair.com/docs/display/1smk3cen4v3lu3yomq5qye0ni

En cuanto a las formas de autenticación disponibles para usar en una aplicación están las siguientes [3]:

1. **Non-interactive login.**
   Es la manera adecuada si se desea realizar la autenticación de manera completamente autónoma.
2. **Interactive login.**
   Como su propio nombre indica, es un método de autenticación que requiere la presencia del usuario.
    1. **Interactive login – API method.**
       Este método hace uso de un JSON API Endpoint para realizar la autenticación. Es la más simple y la más limpia.
    2. **Interactive login – Desktop application.**
       En cuanto a este método requeriría embeber parte de la página de Betfair en la aplicación. La gran ventaja es que permite crear flujos muy similares a los que tiene la propia aplicación web oficial.

Dado que la aplicación hará uso de mercados deportivos se ha elegido usar la Exchange API y el método de autenticación será el interactive login – API method.

En cuanto a la Exchange API requiere de lo siguiente [4]:

1. Una cuenta Betfair (que se abrirá a través de su aplicación web: https://www.betfair.es/).
2. Una application key (que será facilitada por la propia Betfair a través de las demos tools [5]).
3. Un sessionToken, que será facilitado tras realizar la autenticación usando una cuenta Betfair y su correspondiente application key. Este sessionToken será el que nos permita hacer uso de las distintas operaciones de las que provee la API.

---

[3] http://docs.developer.betfair.com/docs/pages/viewpage.action?pageId=3834909

[4] http://foroapuestas.forobet.com/betfair/55166-primeros-pasos-con-el-api-de-betfair-version-es.html

[5] https://developer.betfair.com/exchange-api/accounts-api-demo/

Objetivos

En este contexto, con el desarrollo del presente TFG se ha pretendido diseñar y construir una herramienta gráfica que, haciendo uso del Exchange API de Betair, presente la información de manera limpia, agradable y rápida al usuario, facilitando así las tareas típicas: búsqueda de eventos, consulta de cuotas, realización de apuestas, cancelación, etc.

Se ha buscado definir una arquitectura genérica para la interacción con el API que permita separar la extracción de la información, de la de presentación y la interacción con el usuario. La idea es que la interfaz gráfica pueda utilizar la información extraída de la API creando distintos menús en los que se pueda consultar y realizar operaciones. Entre las operaciones a las que se ha dado soporte están las siguientes:

1. Operaciones de autenticación (mantener la sesión abierta, abrir sesión y cerrar sesión).
2. Operaciones referentes a la cuenta (consultar fondos).
3. Operaciones referentes a los mercados (realizar apuesta, cancelar apuesta, listar apuestas realizadas, listar tipos de eventos, listar eventos, listar competiciones y consultar las distintas opciones dentro de un mercado).

Se ha planteado un diseño completamente modular, que permita introducir nuevas operaciones de manera sencilla. Además, se quiere que la aplicación cuente con soporte multi idioma y que facilite muchísimo la tarea de traducción a cuantos idiomas se necesiten.

Por último y dadas las características de la aplicación, se ha pensado en una lógica que refresque los mercados automáticamente cuando se esté dentro de ellos, cargando la nueva información cada poco tiempo. Por supuesto, los fondos deben actualizarse cuando se realice una apuesta, permitiendo también una actualización manual de estos en caso de querer ver si los fondos de la última apuesta ganada se han contabilizado correctamente.

Solución propuesta

Para conseguir los objetivos enumerados en la sección anterior, se ha planteado el desarrollo de una aplicación de escritorio monousuario que dé cabida a las funcionalidades anteriormente mencionadas.

Haciendo uso del lenguaje Java, en su versión 10, se ha creado un proyecto Maven, empleando SWING para la interfaz gráfica de usuario (GUI), aunque también cuenta con interfaz de consola más limitada.

Para el intercambio de datos se han creado diversas entidades y enumerados, para así poder recolectar la información que facilitan los servicios REST (REpresentational State Transfer) del API de Betfair Exchange, de la manera más limpia posible.

Metodología de desarrollo empleada

La metodología de desarrollo empleada ha sido UP (Unified Software Development Process) [6,7], por tratarse de un proceso de software estandarizado. El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental [8].

Las principales características del Proceso Unificado son:

- **Dirigido por casos de uso.**
  Un sistema software debe tener como base dar servicio a sus usuarios, por lo tanto, en el desarrollo del software se utilizan los casos de uso para representar los requisitos funcionales, dando paso finalmente a un modelo de casos de uso. Sin embargo, no son sólo una herramienta para especificar los requisitos de un sistema. También guían su diseño, implementación y prueba; esto es, guían el proceso de desarrollo, proporcionando un hilo conductor que avanza a través de una serie de flujos de trabajo que parten del modelo de casos de uso.
- **Centrado en la arquitectura.**
  Corresponde a las diferentes vistas del sistema en construcción, incluyendo los aspectos estáticos y dinámicos de éste. Se refleja en los casos de uso, plataforma de funcionamiento, módulos reutilizables, consideraciones de implantación, sistemas heredados, requisitos no funcionales, etc. La arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, ahora y en el futuro. Evolucionan en paralelo.
- **Iterativo e incremental.**

---

[6] http://cybertesis.uach.cl/tesis/uach/2003/bmfcis718d/xhtml/TH.4.xml

[7] https://sites.google.com/site/jcod3x/home/procesos/proceso-unificado-de-desarrollo

[8] https://es.wikipedia.org/wiki/Proceso_unificado

Se divide el desarrollo completo en mini proyectos que forman una iteración con todas las etapas del proceso, consiguiendo pequeñas versiones del producto a desarrollar. Con esta modalidad de trabajo se reducen los costos de riesgo, permite visualizar resultados a corto plazo, y adaptar el producto a necesidades cambiantes del usuario cuando, por lo general, no se tienen claros todos los requisitos.

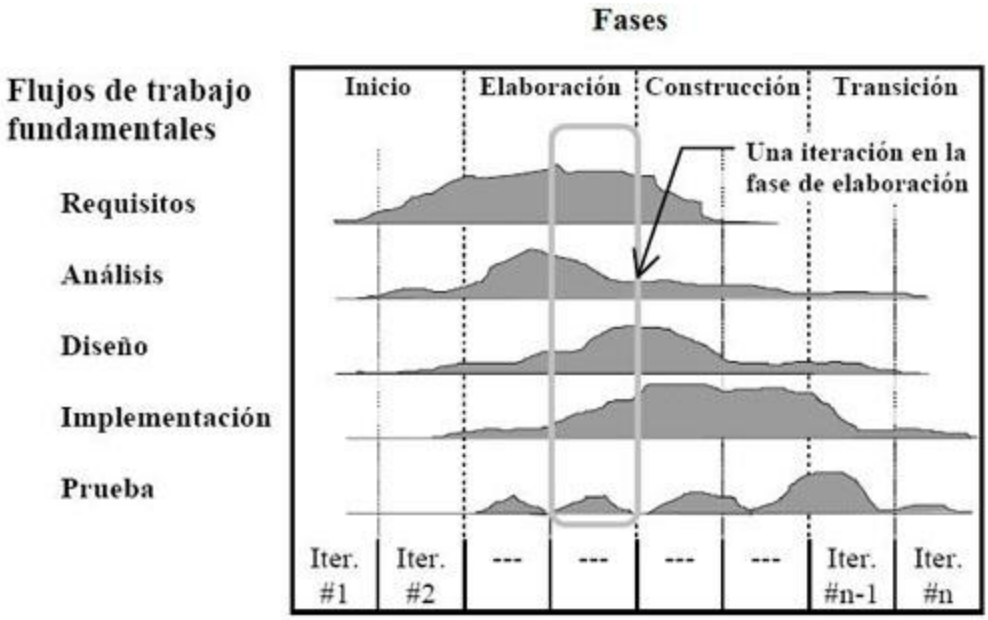
El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) para preparar todos los esquemas de un sistema software. UML es una parte esencial del Proceso Unificado – sus desarrollos fueron paralelos.

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes y consta de cuatro fases:

- **Inicio.**
  Se identifican los casos de uso más críticos junto con un plan de riesgo y estimación, se planifica en detalle la fase de elaboración.
- **Elaboración.**
  Se presenta un modelo detallado de los casos de uso y se diseña la arquitectura del sistema, esta se expresa en forma de vistas de todos los modelos del sistema. Además, se planifican las actividades posteriores y se estiman los recursos para terminar el proyecto.
- **Construcción.**
  Se crea el producto, que contiene todos los casos de uso que la dirección y el cliente han acordado para el desarrollo de la versión. Sin embargo, es posible que no esté libre de modificaciones. Muchos de estos defectos se descubrirán u solucionarán durante la fase de transición.
- **Transición.**
  Cubre el período durante el cual el producto se convierte en versión beta. En ese momento, un número reducido de usuarios con experiencia prueba el producto, e informa de defectos y deficiencias.

Además, en cada una de las fases, se repiten los distintos procesos de los que se compone el desarrollo de software:

- **Requisitos.**
- **Análisis.**
- **Diseño.**
- **Implementación.**
- **Pruebas.**


Fases
Flujos de trabajo fundamentales
Requisitos
Análisis
Diseño
Implementación
Prueba
Inicio
Elaboración
Construcción
Transición
Una iteración en la fase de elaboración
Iter. #1
Iter. #2

Iter. #n-1
Iter. #n

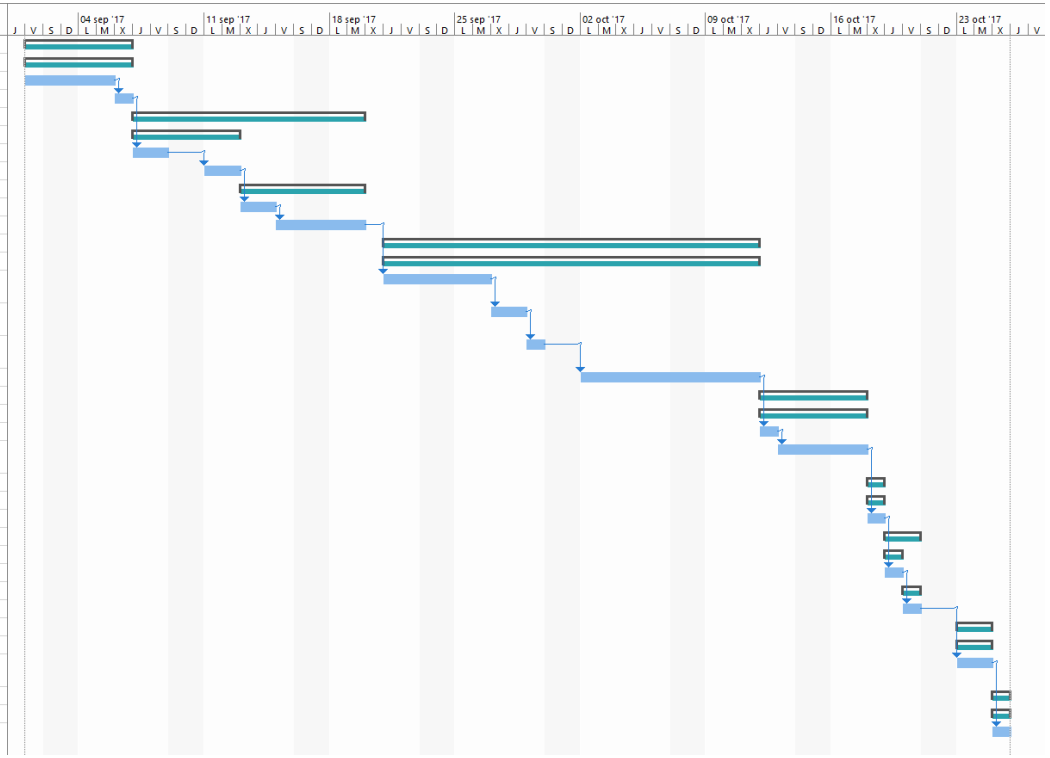
Figura 1

Planificación y seguimiento

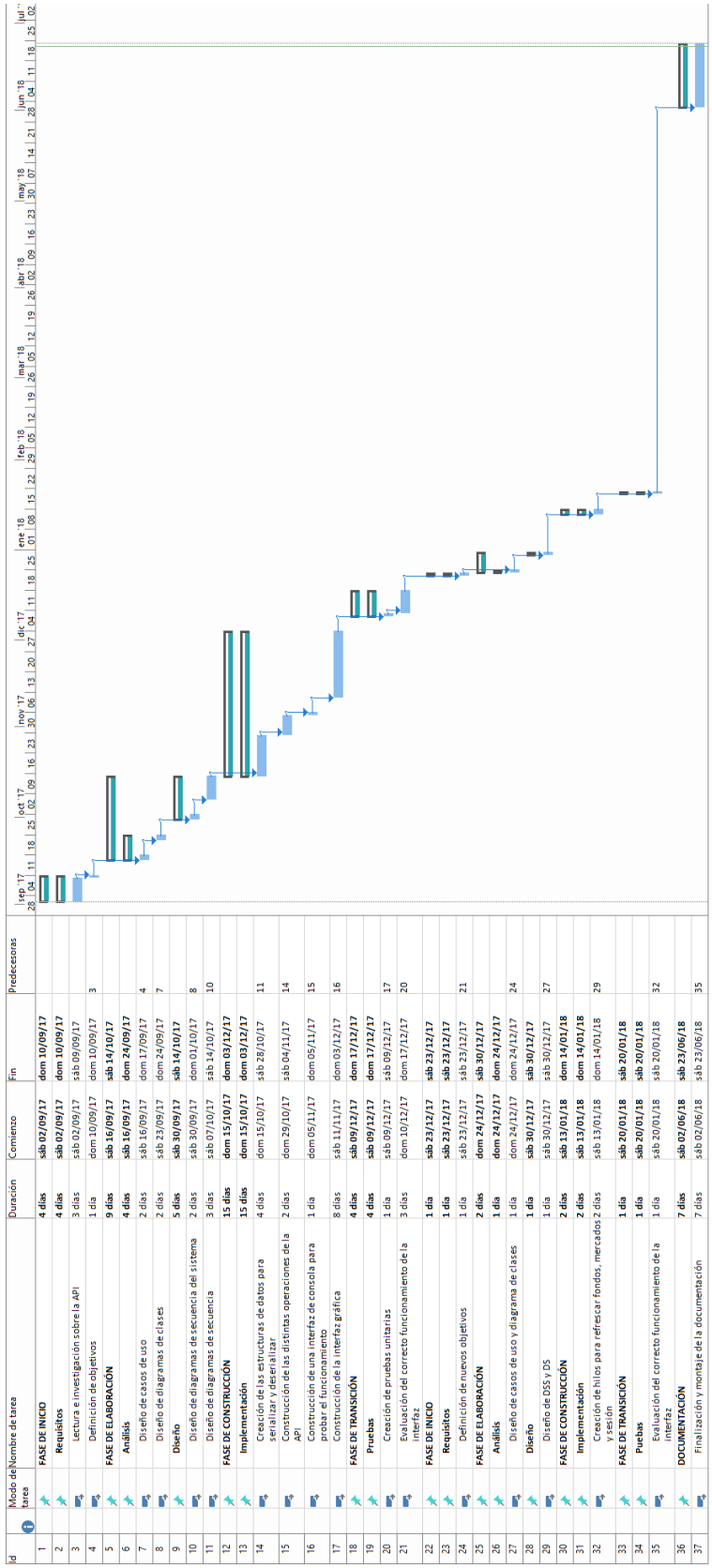
Planificación inicial

La planificación inicial se basó en una jornada de 8 horas de trabajo de lunes a viernes. Tras añadir las tareas se comprobó que serían unas 304 horas, que se corresponderían con 8 semanas de trabajo aproximadamente.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**FASE DE INICIO**	**4 días**	**vie 01/09/17**	**mié 06/09/17**	
2		**Requisitos**	**4 días**	**vie 01/09/17**	**mié 06/09/17**	
3		Lectura e investigación sobre la API	3 días	vie 01/09/17	mar 05/09/17	
4		Definición de objetivos	1 día	mié 06/09/17	mié 06/09/17	3
5		**FASE DE ELABORACIÓN**	**9 días**	**jue 07/09/17**	**mar 19/09/17**	
6		**Análisis**	**4 días**	**jue 07/09/17**	**mar 12/09/17**	
7		Diseño de casos de uso	2 días	jue 07/09/17	vie 08/09/17	4
8		Diseño de diagramas de clases	2 días	lun 11/09/17	mar 12/09/17	7
9		**Diseño**	**5 días**	**mié 13/09/17**	**mar 19/09/17**	
10		Diseño de diagramas de secuencia del sistem	2 días	mié 13/09/17	jue 14/09/17	8
11		Diseño de diagramas de secuencia	3 días	vie 15/09/17	mar 19/09/17	10
12		**FASE DE CONSTRUCCIÓN**	**15 días**	**jue 21/09/17**	**mié 11/10/17**	
13		**Implementación**	**15 días**	**jue 21/09/17**	**mié 11/10/17**	
14		Creación de las estructuras de datos para serializar y deserializar	4 días	jue 21/09/17	mar 26/09/17	11
15		Construcción de las distintas operaciones de la API	2 días	mié 27/09/17	jue 28/09/17	14
16		Construcción de una interfaz de consola para probar el funcionamiento	1 día	vie 29/09/17	vie 29/09/17	15
17		Construcción de la interfaz gráfica	8 días	lun 02/10/17	mié 11/10/17	16
18		**FASE DE TRANSICIÓN**	**4 días**	**jue 12/10/17**	**mar 17/10/17**	
19		**Pruebas**	**4 días**	**jue 12/10/17**	**mar 17/10/17**	
20		Creación de pruebas unitarias	1 día	jue 12/10/17	jue 12/10/17	17
21		Evaluación del correcto funcionamiento de la interfaz	3 días	vie 13/10/17	mar 17/10/17	20
22		**FASE DE INICIO**	**1 día**	**mié 18/10/17**	**mié 18/10/17**	
23		**Requisitos**	**1 día**	**mié 18/10/17**	**mié 18/10/17**	
24		Definición de nuevos objetivos	1 día	mié 18/10/17	mié 18/10/17	21
25		**FASE DE ELABORACIÓN**	**2 días**	**jue 19/10/17**	**vie 20/10/17**	
26		**Análisis**	**1 día**	**jue 19/10/17**	**jue 19/10/17**	
27		Diseño de casos de uso y diagrama de clases	1 día	jue 19/10/17	jue 19/10/17	24
28		**Diseño**	**1 día**	**vie 20/10/17**	**vie 20/10/17**	
29		Diseño de DSS y DS	1 día	vie 20/10/17	vie 20/10/17	27
30		**FASE DE CONSTRUCCIÓN**	**2 días**	**lun 23/10/17**	**mar 24/10/17**	
31		**Implementación**	**2 días**	**lun 23/10/17**	**mar 24/10/17**	
32		Creación de hilos para refrescar fondos, mercados y sesión	2 días	lun 23/10/17	mar 24/10/17	29
33		**FASE DE TRANSICIÓN**	**1 día**	**mié 25/10/17**	**mié 25/10/17**	
34		**Puebas**	**1 día**	**mié 25/10/17**	**mié 25/10/17**	
35		Evaluación del correcto funcionamiento de la interfaz	1 día	mié 25/10/17	mié 25/10/17	32



Seguimiento en el desarrollo



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**FASE DE INICIO**	**4 días**	**sáb 02/09/17**	**dom 10/09/17**	
2		**Requisitos**	**4 días**	**sáb 02/09/17**	**dom 10/09/17**	
3		Lectura e investigación sobre la API	3 días	sáb 02/09/17	sáb 09/09/17	
4		Definición de objetivos	1 día	dom 10/09/17	dom 10/09/17	3
5		**FASE DE ELABORACIÓN**	**9 días**	**sáb 16/09/17**	**sáb 14/10/17**	
6		**Análisis**	**4 días**	**sáb 16/09/17**	**dom 24/09/17**	
7		Diseño de casos de uso	2 días	sáb 16/09/17	dom 17/09/17	4
8		Diseño de diagramas de clases	2 días	sáb 23/09/17	dom 24/09/17	7
9		**Diseño**	**5 días**	**sáb 30/09/17**	**sáb 14/10/17**	
10		Diseño de diagramas de secuencia del sistema	2 días	sáb 30/09/17	dom 01/10/17	8
11		Diseño de diagramas de secuencia	3 días	sáb 07/10/17	sáb 14/10/17	10
12		**FASE DE CONSTRUCCIÓN**	**15 días**	**dom 15/10/17**	**dom 03/12/17**	
13		**Implementación**	**15 días**	**dom 15/10/17**	**dom 03/12/17**	
14		Creación de las estructuras de datos para serializar y deserializar	4 días	dom 15/10/17	sáb 28/10/17	11
15		Construcción de las distintas operaciones de la API	2 días	dom 29/10/17	sáb 04/11/17	14
16		Construcción de una interfaz de consola para probar el funcionamiento	1 día	dom 05/11/17	dom 05/11/17	15
17		Construcción de la interfaz gráfica	8 días	sáb 11/11/17	dom 03/12/17	16
18		**FASE DE TRANSICIÓN**	**4 días**	**sáb 09/12/17**	**dom 17/12/17**	
19		**Pruebas**	**4 días**	**sáb 09/12/17**	**dom 17/12/17**	
20		Creación de pruebas unitarias	1 día	sáb 09/12/17	sáb 09/12/17	17
21		Evaluación del correcto funcionamiento de la interfaz	3 días	dom 10/12/17	dom 17/12/17	20
22		**FASE DE INICIO**	**1 día**	**sáb 23/12/17**	**sáb 23/12/17**	
23		**Requisitos**	**1 día**	**sáb 23/12/17**	**sáb 23/12/17**	
24		Definición de nuevos objetivos	1 día	sáb 23/12/17	sáb 23/12/17	21
25		**FASE DE ELABORACIÓN**	**2 días**	**dom 24/12/17**	**sáb 30/12/17**	
26		**Análisis**	**1 día**	**dom 24/12/17**	**dom 24/12/17**	
27		Diseño de casos de uso y diagrama de clases	1 día	dom 24/12/17	dom 24/12/17	24
28		**Diseño**	**1 día**	**sáb 30/12/17**	**sáb 30/12/17**	
29		Diseño de DSS y DS	1 día	sáb 30/12/17	sáb 30/12/17	27
30		**FASE DE CONSTRUCCIÓN**	**2 días**	**sáb 13/01/18**	**dom 14/01/18**	
31		**Implementación**	**2 días**	**sáb 13/01/18**	**dom 14/01/18**	
32		Creación de hilos para refrescar fondos, mercados y sesión	2 días	sáb 13/01/18	dom 14/01/18	29
33		**FASE DE TRANSICIÓN**	**1 día**	**sáb 20/01/18**	**sáb 20/01/18**	
34		**Pruebas**	**1 día**	**sáb 20/01/18**	**sáb 20/01/18**	
35		Evaluación del correcto funcionamiento de la interfaz	1 día	sáb 20/01/18	sáb 20/01/18	32
36		**DOCUMENTACIÓN**	**7 días**	**sáb 02/06/18**	**sáb 23/06/18**	
37		Finalización y montaje de la documentación	7 días	sáb 02/06/18	sáb 23/06/18	35

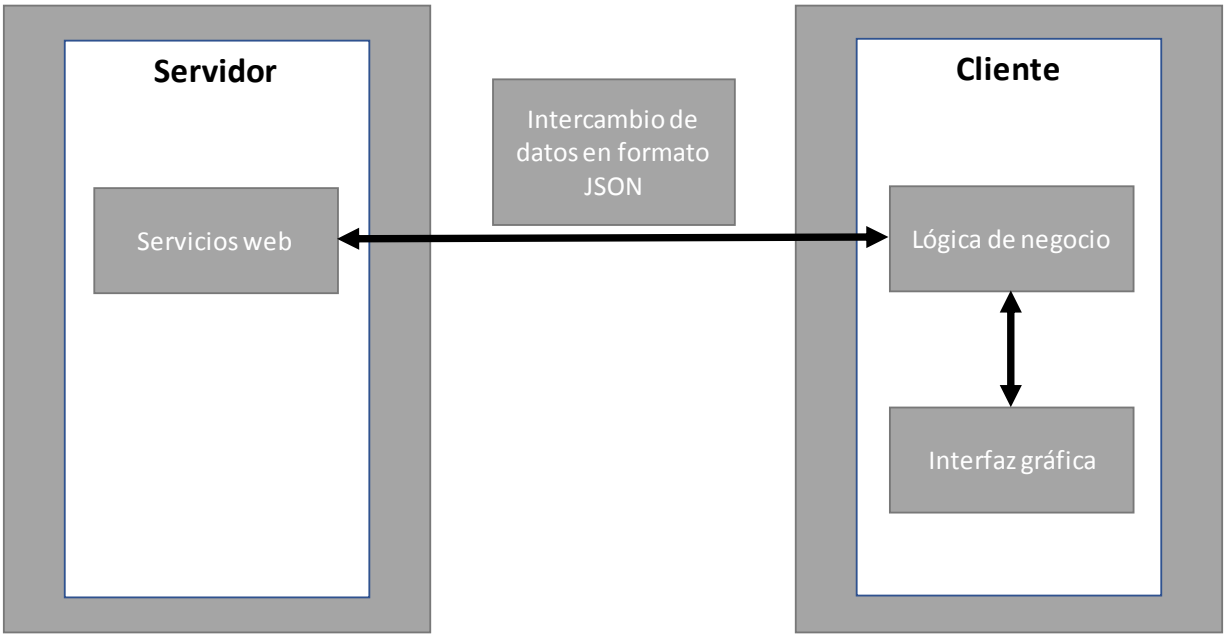
Desviaciones

El proyecto fue desarrollando en las horas disponibles fuera del horario laboral común de lunes a viernes. La falta de tiempo fue el principal motivo que causó que el desarrollo fuera lento, además de las múltiples complicaciones que ocurrieron durante el desarrollo de la interfaz gráfica y que requirieron una investigación más en profundidad para subsanarlas.

En cuanto al ensamblaje de la documentación final se decidió postergarla casi hasta el final nuevamente debido a la falta de tiempo.

Arquitectura general de la solución propuesta

Como se ha señalado, en este TFG se ha planteado el desarrollo de una aplicación de escritorio monousuario que ofrezca una interfaz gráfica para el acceso e interacción con el API de Betfair Exchange. La interfaz gráfica hace uso de la librería de componentes Swing y se comunica con el API de Betfair Exchange mediante peticiones HTTP (Hypertext Transfer Protocol), siguiendo una arquitectura REST (REpresentational State Transfer) y usando JSON (JavaScript Object Notation) como mecanismo de serialización de los datos intercambiados, tal y como se ilustra en la siguiente figura.



En cuanto a lógica de negocio tenemos distintas clases que son las encargadas de llamar a las distintas operaciones de los servicios web basados en REST, enviándoles los parámetros adecuados y recibiendo la información para presentarla adecuadamente en la interfaz gráfica.

Como se ha señalado, el intercambio de información se realiza apoyándose en el formato de intercambio de datos JSON y para la serialización y deserialización se apoya en múltiples clases creadas para tal menester.

De un modo resumido, el funcionamiento general de la herramienta desarrollada sigue un esquema donde la interfaz gráfica genera una petición, que es controlada por la lógica de negocio, que se encarga de serializar los datos, enviarlos al correspondiente endpoint del API REST de Betfair Exchange, recibir la respuesta, deserializarla y pasársela nuevamente a la interfaz gráfica.

Tecnologías empleadas

Plataforma y lenguaje de desarrollo

Para el desarrollo de la herramienta descrita en este documento se ha empleado el lenguaje de programación Java. Los motivos de su elección fueron puramente técnicos, ya que es un lenguaje con el que se contaba ya con abundante experiencia previa, debido a que se ha utilizado ampliamente en las distintas asignaturas de la carrera, lo que ha permitido un desarrollo más rápido. Java es un lenguaje consolidado y maduro, con una comunidad enorme, que siempre permite la aclaración de dudas con una simple búsqueda en Internet. Se trata también de un lenguaje que tiene una larga trayectoria y que cuenta con numerosas librerías y frameworks para hacer frente a cualquier problema sin tener que empezar de cero. Además, Betfair ofrece en su repositorio Github [9] código de ejemplo en este lenguaje [10], además de otros como: C#, Javascript, Perl, PHP, Python, VBA.

El IDE (Integrated Development Environment) elegido fue eclipse. Nuevamente por haber sido ampliamente utilizado durante las asignaturas de la carrera.

Librerías y herramientas utilizadas

- **Font-awesome [11]**
  Esta librería aporta un conjunto de iconos que se han usado en las distintas partes de la interfaz gráfica de la aplicación. Se ha optado por ella debido a que se había usado en otros desarrollos.
- **Gson [12]**
  Esta librería permite convertir estructuras de datos a JSON y viceversa de una manera bastante simple. Como se usan distintas estructuras para la serialización y deserialización a la hora de intercambiar datos con la API de Betfair resulta tremendamente útil. Se ha decidido usarla debido a que se encontró dentro de otro código de ejemplo que la empleaba y resultaba útil.
- **Junit [13]**

---

[9] https://github.com/betfair/API-NG-sample-code

[10] https://www.programcreek.com/java-api-examples/index.php?source_dir=betfair-master/src/com/mauriciotogneri/betfair/api/accounts/Login.java

[11] https://fontawesome.com/

[12] https://github.com/google/gson

[13] https://junit.org/junit5/

Esta librería, ampliamente conocida para realizar pruebas de código, permite realizar pruebas tanto unitarias como de integración. Se ha usado poco, debido a las características del proyecto. Se ha empleado por su conocimiento previo en desarrollos del tipo TDD (Test Driven Development).

- **Httpclient [**14**]**
  Esta librería permite el intercambio de datos con una API REST de manera fácil. Es la base de la aplicación. Entre todo el código de ejemplo que se consultó en un inicio se empleaba una librería parecida, pero que ya no tenía soporte, por lo que se buscó una similar que permitiera hacer lo mismo.
- **Log4j [**15**]**
  Esta librería resulta muy útil a la hora de desarrollar y también en producción para tratar de subsanar posibles errores que puedan ocurrir. Permite crear trazas de log en distintos niveles (warn, debug, etc.). Ampliamente usada en la comunidad Java. Se eligió por ser la más conocida y por haber trabajado anteriormente con ella.
- **Maven [**16**]**
  El proyecto se ha creado con Maven debido a que permite de manera simple la descarga e instalación de herramientas y librerías en el proyecto, eligiendo entre sus distintas versiones. Facilita muchísimo el desarrollo.
- **Swing [**17**]**
  Librería ampliamente conocida que ha servido para la creación de la interfaz gráfica, con sus múltiples pantallas. El haber contado con esta librería en otros proyectos fue clave para su utilización en este.

---

[14] https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient

[15] https://logging.apache.org/log4j/2.x/

[16] https://maven.apache.org/

[17] https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

Especificación y análisis de requisitos

En esta sección se presentará una visión estática de la aplicación: descripción de actores, especificación de requisitos y diagrama de casos de uso.

Descripción de actores

Al tratarse de una aplicación monousuario tan solo se cuenta con un único actor, el propio usuario, que será el que tenga disponibles todas las operaciones.

Especificación de requisitos

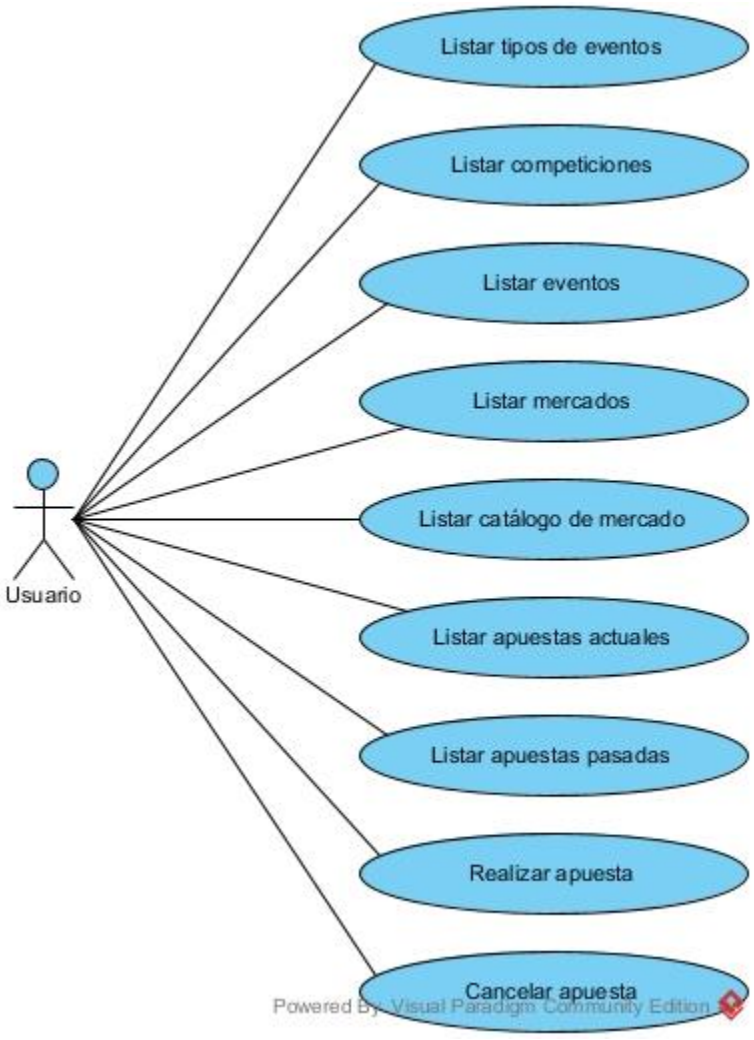
Requisitos funcionales

1. La aplicación debe permitir abrir sesión en Betfair.
2. La aplicación debe permitir cerrar sesión en Betfair.
3. La aplicación debe permitir navegar entre los diferentes mercados.
4. La aplicación debe permitir consultar los diferentes mercados.
5. La aplicación debe permitir realizar apuestas.
6. La aplicación debe permitir cancelar apuestas.
7. La aplicación debe permitir consultar las apuestas realizadas.

Requisitos no funcionales

1. La aplicación debe funcionar en Windows 8 y superiores.
2. La aplicación no debe consumir demasiados recursos.

Diagrama de casos de uso



Diseño de software

En esta sección se presentará una visión dinámica de la aplicación: diagrama de clases, diagramas de secuencia del sistema y diagramas de secuencia.

Diagrama de clases

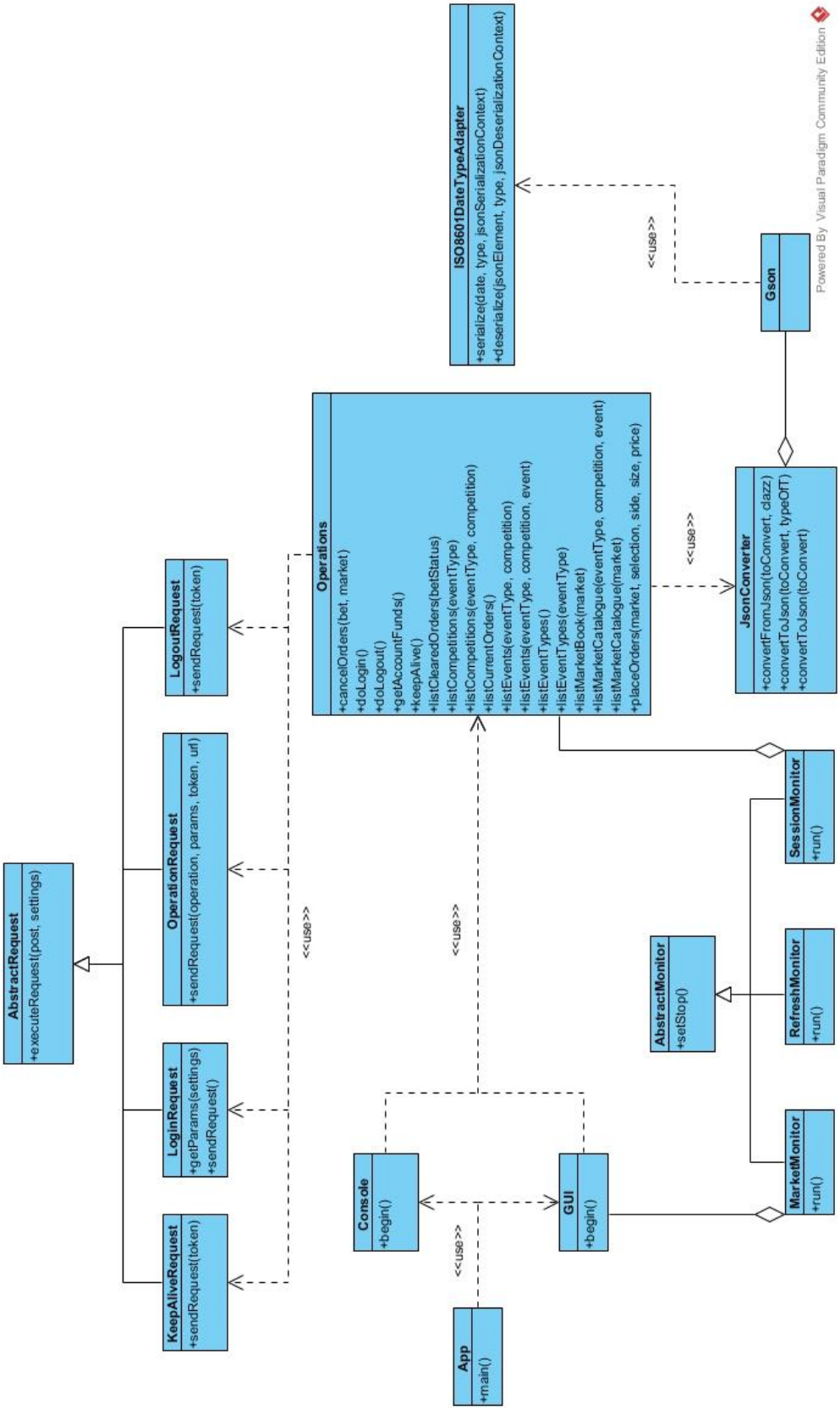
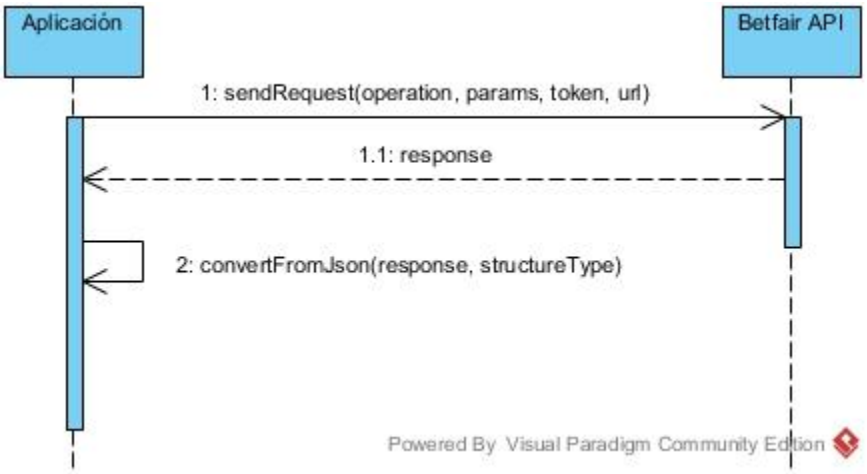


Diagrama de secuencia del sistema

Al tratarse de una aplicación que realiza peticiones a una API, los diagramas de secuencia del sistema son todos muy similares, cambiando el endpoint al que se realiza la petición y sus parámetros. Se detallan algunos a continuación a modo de ejemplo:

Listar tipos de eventos



Realizar apuesta

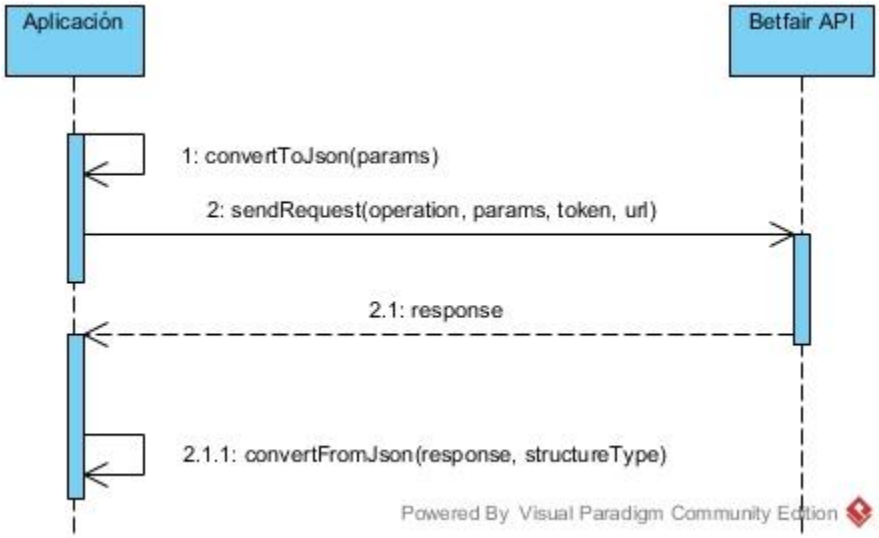
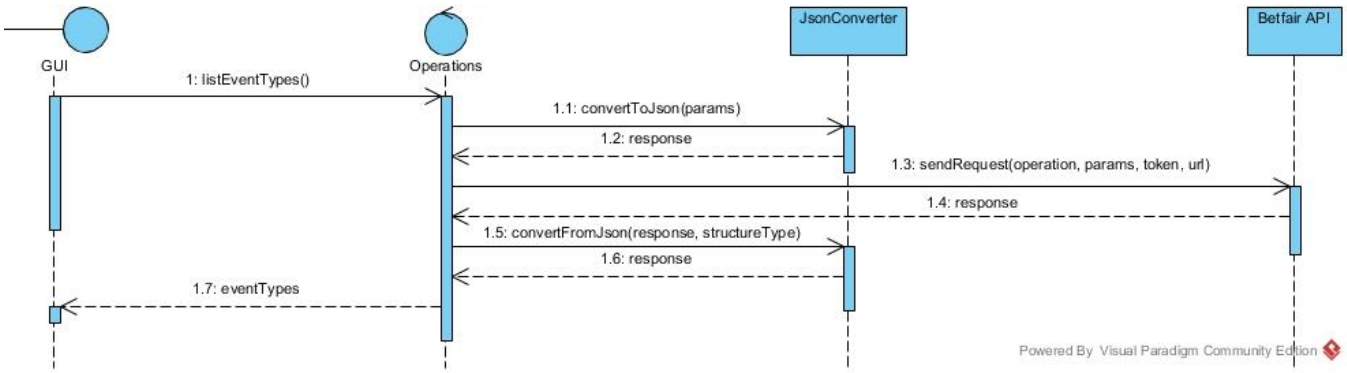
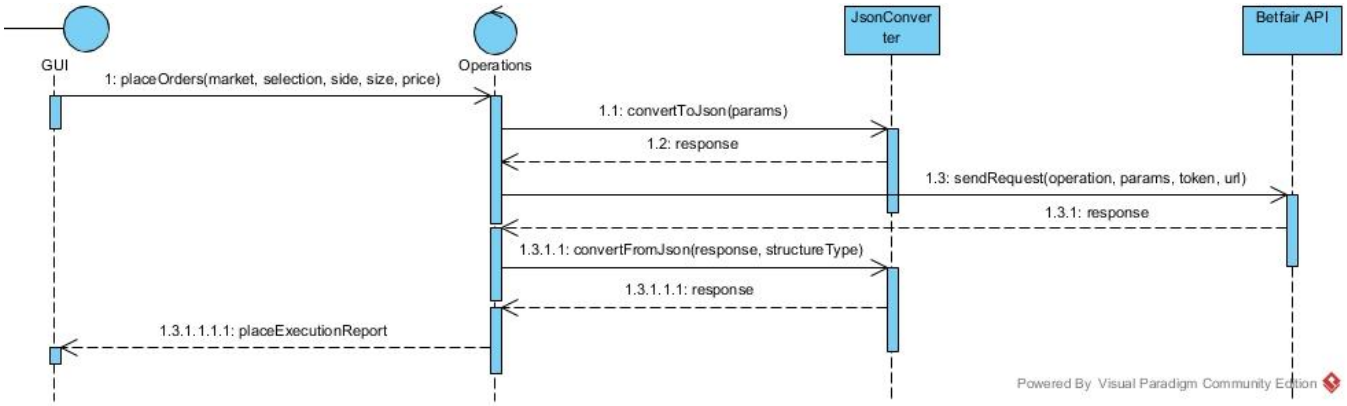


Diagrama de secuencia

Listar tipos de eventos



Realizar apuesta



Pruebas

Para comprobar la lógica de la aplicación se han creado pruebas unitarias y una interfaz de consola para así comprobar que la recepción y el envío de información era correcto.

- Las pruebas unitarias son meramente testimoniales debido a que lo único que cambia de una prueba a otra es la operación que realizar y los parámetros que se envían, pero en sí la prueba es la misma.
- En cuanto a la interfaz de consola se ha recogido todo el funcionamiento de la aplicación en él, no presentando la información de manera elegante, pero sí para que sea totalmente funcional y así poder probar todo.

En lo que respecta a la interfaz gráfica las pruebas han sido totalmente manuales. Se ha construido utilizando la lógica ya probada y a partir de ahí se ha empezado el desarrollo de las distintas pantallas de la interfaz.

Manual de usuario

Funcionalidades de la herramienta

- Listar tipos de eventos.
- Listar competiciones dentro de un tipo de evento.
- Listar eventos dentro de una competición.
- Listar mercados asociados a un evento.
- Consultar mercado.
- Realizar apuestas.
- Cancelar apuestas.
- Consultar fondos disponibles en la cuenta.
- Consultar apuestas abiertas.
- Consultar historial de apuestas.

Requisitos previos

- Apache Maven (en caso de querer realizar la compilación del código fuente).
- Conexión a internet.
- Java Runtime Environment (JRE) 10 o superior.
  Se trata de la máquina virtual de Java que permite ejecutar la aplicación en cualquier sistema operativo que permita la instalación del JRE.

Instalación y puesta en marcha

Antes de empezar es necesario configurar los parámetros del archivo Settings.properties. En él se deben especificar ciertos parámetros como la appkey, el nombre de usuario y la contraseña. El resto de parámetros pueden dejarse tal cual están.

Al tratarse de un proyecto Maven podemos realizar la compilación de la aplicación situándonos en la carpeta donde se encuentre la aplicación y ejecutando el siguiente comando [18]:

mvn package

A continuación, podríamos ejecutar la aplicación directamente con el siguiente comando:

java -jar target/JustBet.jar

En caso de querer ejecutar la aplicación con log, ejecutaríamos:

---

[18] https://www.mkyong.com/maven/how-to-create-a-jar-file-with-maven/

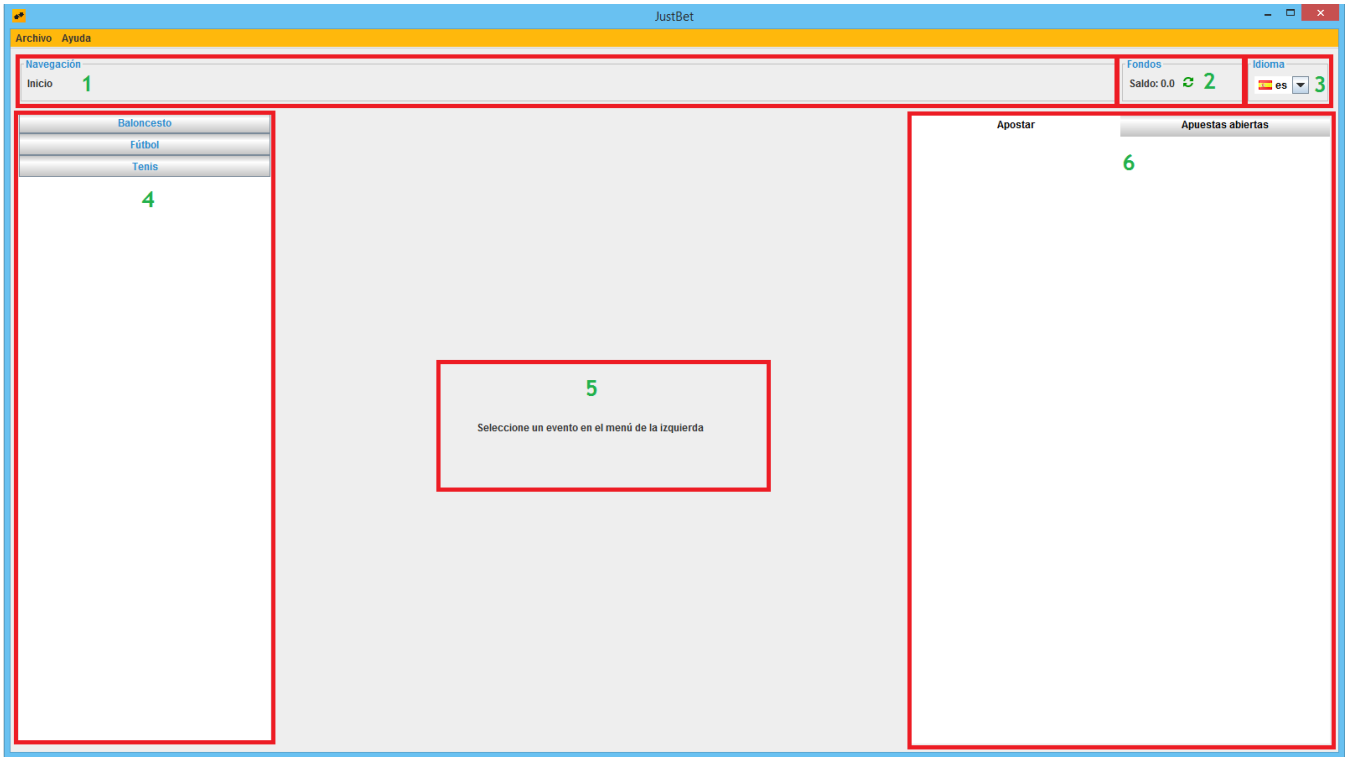
java -jar -Dlog4j.configuration=file:/full_path/log4j.properties target/JustBet.jar

Es posible ejecutar la aplicación en modo consola. Tan solo debemos pasarle un parámetro:

java -jar target/JustBet.jar -console

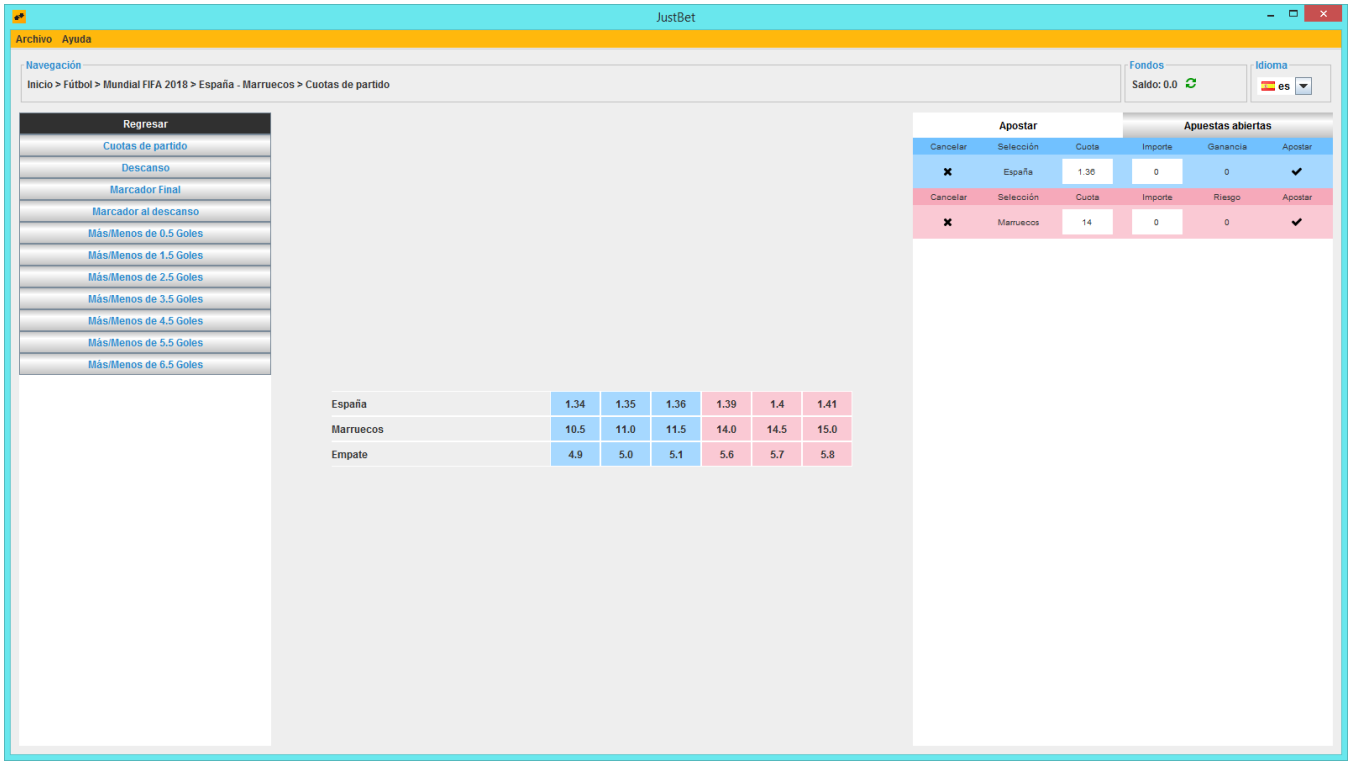
Instrucciones de uso

La pantalla principal del programa, una vez introducidas las credenciales para iniciar sesión correctamente es la siguiente:



1. **Navegación**. El usuario podrá conocer en cada momento donde se encuentra y así no perderse entre los distintos menús.
2. **Fondos**. Aquí se podrá observar y refrescar los fondos que se tengan según se vayan realizando apuestas.
3. **Idioma**. Permite el cambio de idioma en todo momento, refrescando todos los menús.
4. **Menús de selección de tipos de eventos, competiciones, eventos y mercados**. El usuario podrá desde aquí buscar el mercado en el que le interese apostar.
5. **Mercado**. Una vez seleccionado un mercado aparecerá aquí.
6. **Menús de apuestas**. Cuando se pinche en una cuota, el cupón de apuestas se cargará aquí. Se podrá cambiar la cuota al gusto (aunque no consiga igualarse), indicar el importe de apuesta a realizar y finalmente realizar la apuesta. Una vez realizada se podrá consultar en la otra pestaña que dice apuestas abiertas.

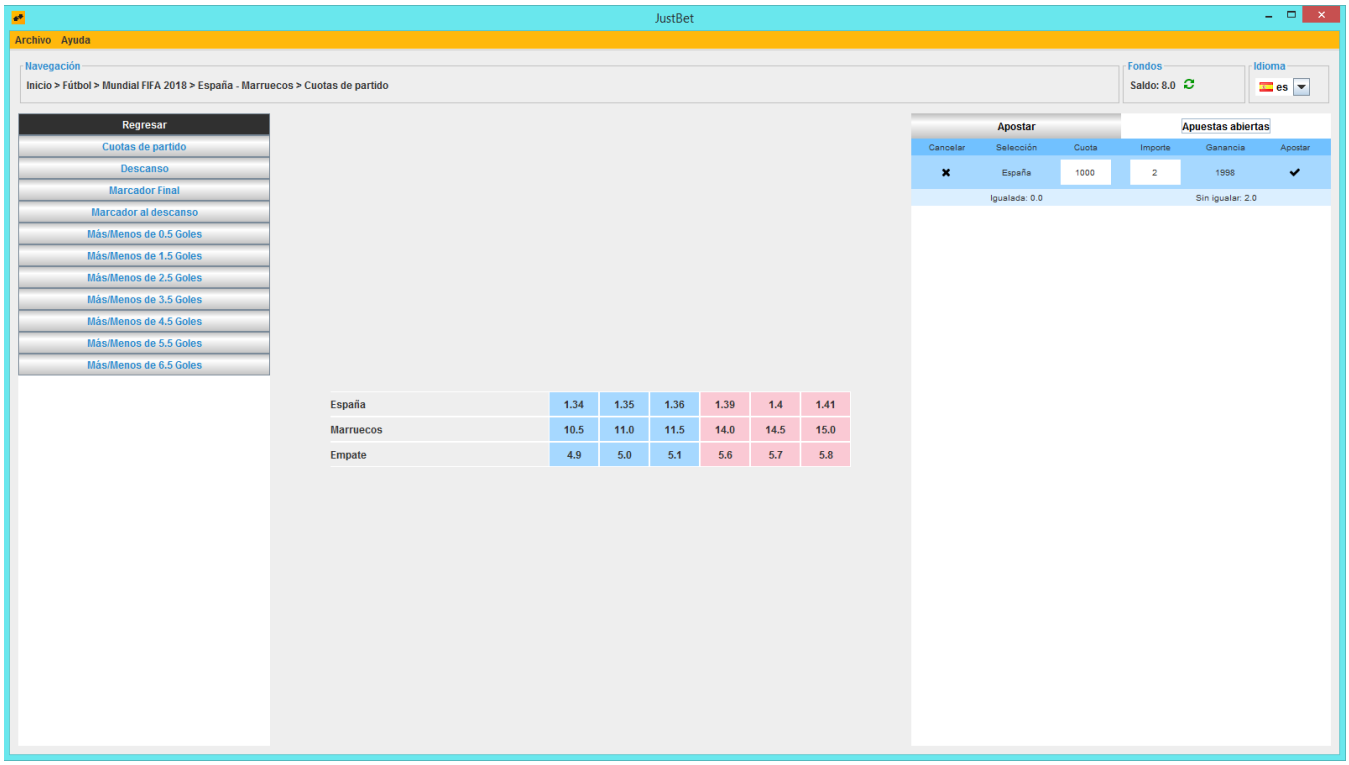
Así se vería la aplicación con un mercado cargado:



El azul representa las cuotas back, que son las normales, en las que el usuario vaticina que sucederá su pronóstico. El rosa representa las cuotas lay, las contrarias, el usuario vaticina que un determinado evento no sucederá.

Podemos además apreciar que se han cargado en el menú de apuestas un par de selecciones, cada una con su color dependiendo del tipo de apuesta que se trate.

Tras realizar una apuesta, desaparecerá de la pestaña apostar y pasará a la de apuestas abiertas.


JustBet
Archivo Ayuda
Navegación
Inicio > Fútbol > Mundial FIFA 2018 > España - Marruecos > Cuotas de partido
Fondos
Saldo: 8.0
Idioma
es
Regresar
Cuotas de partido
Descanso
Marcador Final
Marcador al descanso
Más/Menos de 0.5 Goles
Más/Menos de 1.5 Goles
Más/Menos de 2.5 Goles
Más/Menos de 3.5 Goles
Más/Menos de 4.5 Goles
Más/Menos de 5.5 Goles
Más/Menos de 6.5 Goles
España 1.34 1.35 1.36 1.39 1.4 1.41
Marruecos 10.5 11.0 11.5 14.0 14.5 15.0
Empate 4.9 5.0 5.1 5.6 5.7 5.8
Apostar
Apuestas abiertas
Cancelar Selección Cuota Importe Ganancia Apostar
España 1000 2 1998
Igualada: 0.0
Sin igualar: 2.0

Aportaciones del TFG

La principal aportación del trabajo desarrollado es la propia herramienta realizada, que ofrece una interfaz sencilla y amigable que replica parte de las funcionalidades de la web de Betfair Exchange, haciendo uso del API REST de este servicio. De un modo más concreto, se ha conseguido que todos los puntos de la sección de objetivos fueran cubiertos:

1. Se ha creado una interfaz gráfica amigable, atractiva y simple, que permite las tareas típicas que necesita una aplicación de apuestas.
2. Se ha creado una arquitectura genérica. Utilizando las virtudes de la programación orientada a objetos, ha sido posible crear una lógica genérica para el intercambio de datos con la API y luego métodos específicos que usasen esta lógica genérica evitando así la reutilización de código.
3. Se han definido todas las operaciones necesarias de autenticación, referentes a la cuenta y referentes a mercados.
4. Se le ha dado soporte multi idioma. Por ahora solo con español e inglés, pero resulta de lo más simple introducir nuevos idiomas.
5. Se ha dotado a la aplicación de varios hilos de ejecución para el mantenimiento de la sesión abierta, el refresco de mercados y el refresco de los fondos.

Conclusiones

Conocimientos adquiridos

La creación de la aplicación me ha permitido refrescar conocimientos y mejorarlos en lo relativo a cómo el diseñar un buen diagrama de clases es sumamente importante como base para llevar a cabo una implementación manejable y correcta. Antes de realizar este desarrollo, había creado un prototipo más sencillo y resultaba un poco caótico. Tras consultar código oficial en el repositorio de Betfair y darles una vuelta a mis diagramas de clase, terminé dándome cuenta de que este diseño era lo más adecuado para lo que quería hacer.

En cuanto a las herramientas que no había utilizado y me resultaron tremendamente útiles han sido:

- Gson, que simplifica enormemente la serialización y deserialización de datos en formato JSON
- Httpclient, que encapsula de forma conveniente las operaciones relacionadas con el transporte de datos sobre el protocolo HTTP.

Adecuación de las decisiones técnicas tomadas

Ya que, al final, lo más laborioso ha sido la creación de la interfaz gráfica, las decisiones más importantes han sido referentes a ella.

- He tenido que crear múltiples clases para darle más vistosidad a la GUI. Por ejemplo, extender la clase JComboBox para crear un componente combobox específico para esta aplicación, que me permitiese introducir una bandera y un texto.
- Uso del patrón Singleton para recuperar instancias de objetos desde cualquier parte de la GUI donde los necesitase sin tener que hacer pasos de parámetros infinitos.
- La posibilidad de cambiar de idioma en cualquier momento sin tener que cerrar y volver a abrir la aplicación me ha obligado a añadir bastante código adicional.

En cuanto al lenguaje de programación utilizado, no ha habido demasiados problemas ya que es un lenguaje en el que dispongo de cierta experiencia y la metodología de desarrollo la habíamos usado en varias ocasiones durante la carrera. No obstante, este proyecto ha servido para poner en práctica esa experiencia en un contexto mucho más amplio y exigente que el de las prácticas realizadas durante la titulación.

Trabajo futuro

En cuanto a posibles mejoras a realizar sobre la aplicación desarrollada, se podrían agrupar según la parte de la misma a la que afecten.

Interfaz gráfica

La principal mejora sería evitar que el refresco del mercado incluya a todo el panel. Sería mucho más elegante, que en vez de borrar todo el panel y montarlo de nuevo, que simplemente se modificasen los números.

En cuanto a los mensajes de error, algunos mensajes se devuelven directamente en inglés aun teniendo la aplicación en español. Sería deseable traducirlos todos.

En función de las apuestas realizadas, mostrar en el propio mercado cuánto dinero ganaremos o perderemos dependiendo de cada resultado. Al final es una aplicación de trading, así que si realizamos varias apuestas en un mismo mercado debería reflejarse en valores absolutos cuánto vamos a ganar y cuánto vamos a perder sin tener que calcularlo nosotros.

Lógica de negocio

De cara a la evolución futura y ante un previsible uso más extensivo de la aplicación desarrollada, resultaría útil programar algún tipo de soporte del concepto stop loss [19]. De esta forma, el usuario, podría configurar un mecanismo mediante el cual, si su apuesta empieza a ir realmente mal, directamente se realice un trading para evitar seguir perdiendo dinero. Técnicamente, este tipo de añadido supondría dotar a la aplicación de un hilo aparte responsable de seguir las apuestas en curso y actuar al detectar el tipo de pérdida configurado.

Persistencia de apuestas

Aunque la aplicación nos permite consultar un histórico, que al final es el que nos ofrece Betfair, no existe una persistencia de las apuestas en local. Podría resultar interesante que se exportase en un formato de hoja de cálculo (Excel u Open/LibreOffice Calc) según las vayamos realizando para así poder ir viendo nuestros progresos.

---

[19] https://esbolsa.com/blog/analisis-tecnico/stop-loss-importante-inversiones/

Referencias

[1] https://developer.betfair.com/ (en línea, 02/09/2017)

[2] http://docs.developer.betfair.com/docs/display/1smk3cen4v3lu3yomq5qye0ni (en línea, 23/06/2018)

[3] http://docs.developer.betfair.com/docs/pages/viewpage.action?pageId=3834909 (en línea, 23/06/2018)

[4] http://foroapuestas.forobet.com/betfair/55166-primeros-pasos-con-el-api-de-betfair-version-es.html (en línea, 15/10/2017)

[5] https://developer.betfair.com/exchange-api/accounts-api-demo/ (en línea, 15/10/2017)

[6] http://cybertesis.uach.cl/tesis/uach/2003/bmfcis718d/xhtml/TH.4.xml (en línea, 17/06/2018)

[7] https://sites.google.com/site/jcod3x/home/procesos/proceso-unificado-de-desarrollo (en línea, 17/06/2018)

[8] https://es.wikipedia.org/wiki/Proceso_unificado (en línea, 17/06/2018)

[9] https://github.com/betfair/API-NG-sample-code (en línea, 22/10/2017)

[10] https://www.programcreek.com/java-api-examples/index.php?source_dir=betfair-master/src/com/mauriciotogneri/betfair/api/accounts/Login.java (en línea, 22/10/2017)

[11] https://fontawesome.com/ (en línea 22/10/2017)

[12] https://github.com/google/gson (en línea 22/10/2017)

[13] https://junit.org/junit5/ (en línea 22/10/2017)

[14] https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient (en línea 22/10/2017)

[15] https://logging.apache.org/log4j/2.x/ (en línea 22/10/2017)

[16] https://maven.apache.org/ (en línea 22/10/2017)

[17] https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html (en línea 22/10/2017)

[18] https://www.mkyong.com/maven/how-to-create-a-jar-file-with-maven/ (en línea, 20/01/2018)

[19] https://esbolsa.com/blog/analisis-tecnico/stop-loss-importante-inversiones/ (en línea, 17/06/2018)