SuperBet

Taller de Sistemas de Información 2 - curso 2011

Grupo 4

Brian Hill

brian.ohillm@gmail.com Marcos Bellucci

delbetu@gmail.com Andrés Andina

caandina@gmail.com Martín Vázquez  
lobimar@gmail.com

**RESUMEN**

Los juegos de apuestas *on-line* están ampliamente difundidos en nuestros días. En este trabajo se expone el desarrollo de un sitio de características similares a los anteriormente mencionados, se verán las etapas vinculadas al análisis, diseño e implementación de los distintos módulos que conforman la solución. La plataforma de desarrollo utilizada es JEE (*Java Enterprise Edition*) [1], en conjunción con otras tecnologías que permiten la adición de ciertas características interesantes como ser consumo de noticias y multimedia.

El desarrollo aquí presentado aporta un enfoque particular, investigación en diversas tecnologías útiles a las funcionalidades a desarrollar y experiencia en la implementación de un Sistema de Información de mediano porte.

**Palabras Clave**

JEE, Apuestas *on-line*, *web Services*, MVC (*Model-View-Controller*), capas, integración, DAO.

# INTRODUCCIÓN

Debido a la proliferación de los juegos de apuestas *on-line*, lo que hace que su negocio sea bien entendido, y la multiplicidad de funcionalidades y variantes tecnológicas que un proyecto de esta clase permite, resulta interesante el desarrollo de un sistema de estas características como motivación para la investigación y ensayo de las diversas tecnologías que fueron integradas en la solución aquí presentada.

Algunas de las características más importantes de este proyecto son la multiplicidad de tecnologías integradas, el diseño amigable de la interfaz de usuario (especialmente en lo que toca al módulo de acceso público) y la organización en lo que al desarrollo se refiere, planteando iteraciones junto a sus respectivos hitos. Este trabajo resulta de relevancia para sus desarrolladores, dado que presentó una buena oportunidad para ensayar una gran variedad de tecnologías con las que nunca antes se había trabajado, además de contribuir con una posible solución a un Sistema de Información de mediano porte y futura referencia en lo que abarca el uso de las tecnologías JEE. Representa además el esfuerzo disciplinado y constante por parte de un equipo de desarrolladores en la construcción de una solución funcional y de relativa calidad.

Resulta importante destacar la separación en capas que propone esta solución, de esta forma se mantuvo un enfoque ordenado de desarrollo y se preservaron buenas cualidades en el software, como por ejemplo el bajo acoplamiento de los componentes pertenecientes a cada una de las capas. También es importante destacar la abundancia de requerimientos opcionales implementados, lo cual demuestra el ensayo de diversas tecnologías durante el desarrollo de esta solución.

Por último, es importante mencionar la separación de la solución en tres módulos de funcionalidades, a saber: módulo de acceso público, de administración y de moderación. En la sección 4 se describirán las responsabilidades que competen a cada uno de ellos, por el momento basta saber que el módulo de administración brinda infraestructura al de moderación y el módulo público es donde los usuarios apuestan.

Este trabajo se presenta en el marco de la edición 2011 de la asignatura Taller de Sistemas de Información 2 [2].

Conviene, en lo que sigue, presentar la organización de este artículo: en la sección siguiente se verán algunos de los conceptos utilizados a lo largo del desarrollo de la solución. En la sección 3 se describe el problema motivador del trabajo, a su vez, en la sección 3 se detallaran aspectos generales de la solución planteada, esto es, una explicación a alto nivel. En la sección 5 se describe la arquitectura general de la solución, luego, en la sección 6 se brinda una visión más profunda de la solución, abarcando detalles de implementación. La sección 7 brinda una evaluación del producto. Los últimos dos capítulos se concentran en detalles relacionados al desarrollo en sí mismo (hitos planteados, estimaciones de horas y demás), además de presentar sendas conclusiones y algunas líneas de desarrollo y/o investigación que pueden trazarse a futuro.

# MARCO CONCEPTUAL

En esta sección se describen algunos de los conceptos que resultó necesario abordar durante el desarrollo de la solución planteada en este trabajo.

## Composite view

En relación a las vistas de la aplicación se utilizó este patrón, de esta forma, se pudo dejar ciertas secciones de las páginas fijas (cómo ser *sidebars*, *headers* y el área de cada funcionalidad) minimizando la cantidad de código HTML y proponiendo un enfoque elegante. Se utilizaron *facelets* de JSF para lograr este cometido.

## Web Services

En lo que refiere a la integración con sistemas externos se utilizaron *web services*. Se exponen sendas interfaces para consumo tanto de operaciones como de estructuras de datos, las mismas se declaran en sus archivos *wsdl* correspondientes.

## Separación en capas

El estilo arquitectónico utilizado fue la separación en capas (estricto), lo cual propone tres niveles de abstracción que permiten alterar el comportamiento de los componentes de una capa en particular sin alterar la comunicación de esta con las restantes. Además propone un enfoque ordenado de cara a la implementación de la solución.

## DAO

El patrón de diseño *Data Access Object* permiten aislar a la aplicación del conocimiento acerca de la tecnología de persistencia subyacente (en el caso concreto de este desarrollo: *Hibernate* [3]).

# DESCRIPCIÓN DEL PROBLEMA

En esta sección se describirán, a groso modo, el dominio del problema y las funcionalidades que el mismo plantea. En referencia al dominio, las entidades predominantes son las siguientes: Eventos, Apuestas, Usuarios y Resultados (tanto de eventos como de apuestas de un usuario).

Cabe destacar que las Apuestas (en lo que sigue mencionaremos a las entidades con su primer letra en mayúscula), poseen un Tipo determinado, si bien esta entidad no reviste gran importancia a nivel de descriptivo, sí la tiene a nivel funcional, dado que los tipos de apuesta son los que determinan que podrá apostar cada Usuario, sobre un Evento determinado, y que no. Los resultados se interpretan de dos maneras distintas: como un Resultado seleccionado en la Apuesta de un Usuario y cómo el Resultado final de un Evento sobre el cual varios Usuarios apostaron.

La solución fue dividida en tres módulos diferentes, a saber: módulo de acceso público, módulo de administración y módulo de moderación. Las funcionalidades y propósito de cada uno de ellos se describen a continuación:

1.     Módulo público: es el módulo principal de la solución, en el mismo los Usuarios realizarán sus Apuestas, las funcionalidades principales son las siguientes:

* Apostar: es la funcionalidad principal de la solución, se pueden realizar tanto Apuestas combinadas (esto es, combinar Apuestas sobre diversos eventos que se permitan desde el módulo de moderación), como de otros tipos más simples. Además se permite apostar sobre Eventos publicados en sistemas externos.
* Gestionar perfil: se podrá actualizar la información referente a un Usuario público (datos personales, foto, etc.), así como depositar dinero en la cuenta virtual del mismo (acceso al sistema *PayPal* [4] mediante).
* Consultar noticias y acceso multimedia: se ofrece la visualización de noticias relacionadas a los deportes más importantes (fútbol, básquetbol, automovilismo, etc.), para que el Usuario pueda apostar con cierta información de primera mano. Además se proveen mapas con ubicaciones de los eventos más importantes.

2.     Módulo de administración: provee la infraestructura con la que trabajaran los Moderadores del sitio. Los únicos Usuarios que tienen permiso de ingresar en este módulo son los Administradores. Las funcionalidades más relevantes son:

* + Gestión de Moderadores: se podrán agregar nuevos Moderadores, editar sus datos y eliminarlos.
  + Gestión de Fuentes de Información: alta, baja y modificación de las Fuentes de noticias para el módulo de acceso público.
  + Gestión de Eventos: alta y modificación de los Eventos sobre los que se puede apostar. A su vez, se podrá combinar determinados Eventos para que los Usuarios públicos puedan realizar Apuestas combinadas.

3.     Módulo de moderación: modera la actividad de los Usuarios del módulo público. Los únicos usuarios que tienen acceso a este módulo son los Moderadores. Las principales funcionalidades se detallan a continuación:

* Gestión de Resultados: se podrá crear y modificar nuevos Resultados y relacionarlos con determinados Eventos.
* Pago de Apuestas: cuando finaliza un Evento y se han realizado Apuestas sobre él, un Moderador podrá pagar las mismas. Los montos en las cuentas de todos los Usuarios apostadores será modificado. También se pueden hacer pagos de apuestas combinadas.

Además de los requerimientos funcionales anteriores, se enriquecen las posibilidades de los Usuarios públicos mediante la implementación de una aplicación que permite hacer Apuestas desde dispositivos *Android* [5], además de la consulta del calendario deportivo, en el cual se puede apreciar el orden de los Eventos en el tiempo. La distribución de la aplicación en “la nube” (por medio de *OpenShift* [6]), permite tener libre acceso a la solución. Estos aspectos serán detallados en secciones posteriores.

# SOLUCIÓN PLANTEADA

A continuación se expondrán las decisiones de diseño que más repercusión tuvieron durante el desarrollo de la solución. Adicionalmente se explicarán todos los detalles de relevancia relacionados a los requerimientos opcionales desarrollados.

## Proyectos integrados a la solución

La solución cuenta con tres grandes módulos, los mismos constan de un proyecto dinámico Web y 2 proyectos EJB (*SuperBetWeb*, SuperBetNegocio y SuperBetPersistencia respectivamente), otros 2 módulos auxiliares implementados como proyectos EJB (SuperBetDominio y SuperBetMultimedia), y 2 proyectos más para la aplicación móvil, uno de naturaleza Web que es utilizado para la comunicación entre sistemas externos y para la establecida entre la aplicación móvil y la capa de negocio, y el proyecto para la implementación de la interfaz de la aplicación *Android*, a saber: *SuperBetService* y *SuperBetAndroid*, respectivamente.  
Como pudo deducirse de las secciones anteriores, existen tres actores que interactúan con el sistema, a saber:

* + Administrador: se encarga de manejar los moderadores, los eventos y las fuentes de noticias.
  + Moderador: se encarga de manejar los resultados de los eventos y de pagar a los usuarios cuando ganan la apuesta.
  + Público: son los usuarios que se registran en el sistema donde básicamente realizan apuestas a los eventos.

A continuación se da una explicación más sucinta de los proyectos antes mencionados.

### SuperBetWeb

Es la implementación de la capa de presentación donde se encuentran todas las interfaces por los que los distintos Usuarios del sistema interactúan con ella. Todos los Usuarios del sistema ingresan a él por la misma página de *login* pero una vez logueado se lo redirige a la página que le corresponde según su rol (usuario público, administrador o moderador). A continuación se da el detalle de las vistas de cada módulo.  
Módulo público:

* + *Home*: donde se pueden visualizar las noticias y videos sobre los Eventos de mayor relevancia.
  + Eventos: presenta el calendario de los mismos.
  + Apuestas: permite realizar Apuestas tanto simples como combinadas sobre los Eventos del sistema.
  + Ayuda: contiene toda la información necesaria para el correcto accionar de los Usuarios.

Módulo de administración:

* Eventos: gestión de Eventos sobre los que se podrá apostar.
* Moderadores: gestión de los Moderadores del sistema.
* Fuentes: gestión de fuentes de noticias.

Módulo de moderación:

* Resultados, gestión de Resultados y ejecución de pagos a los Usuarios que ganan las Apuestas.

Además de las vistas antes mencionadas, se implementaron otros requerimientos de interés, cómo ser la presentación de videos y noticias acerca de los Eventos más importantes que se hayan registrado, adicionalmente se hace un extenso uso del mapa provisto por *Google*, para ubicación de los Eventos que se hayan dado de alta en el sistema. En esta parte se integran las tecnologías de *Google maps* (*Prime Faces* provee un componente que facilita esta integración), el API de *Youtube* [7] y las librerías provistas por *Rome* [8] para el consumo de noticias.

### SuperBetNegocio

Es el nexo entre la capa de presentación y la de persistencia.

### SuperBetPersistencia

Representa la capa de persistencia que interactúa con una base de datos, esta última “corre” sobre un Servidor *MySQL*. El acceso a datos se hizo utilizando *Hibernate*.

### SuperBetDominio

Consiste de un simple Proyecto que contiene las clases de negocio y es incluido por todos los otros para poder realizar la comunicación de datos entre las distintas capas.

### SuperBetService

Este proyecto es una aplicación web que provee servicios tanto para ser consumidos por la aplicación *Android* como para ser consumidos por otros sistemas externos.  
Consiste de un proyecto web que ofrece *Web Services* de tipo *REST* para la comunicación entre la interfaz del dispositivo móvil y la capa de negocio, debido a que es el único método de comunicación posible entre ellos. A su vez, se utilizó *Web Services* SOAP para consumo de sistemas externos *SuperBet*.

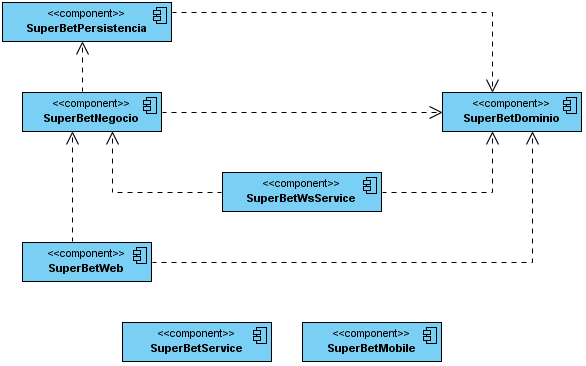
### SuperBetWsService

Proyecto en el cual se implementan los *Web Services* utilizados para integrar con sistemas externos.

### SuperBetMobile

Consiste de un proyecto tipo Android donde se pueden realizar apuestas desde un smartphone con dicho sistema operativo. En este proyecto se implemento la interfaz mobile.

En la siguiente figura se ilustra la relación existente entre los sistemas anteriormente descriptos.



**Dependencias de los proyectos de la solución**

## Utilización de Web Services e integración con sistemas externos

Para realizar la integración con el sistema externo se generó, en el sistema “de origen”, un único usuario G04 que representa a todos los usuarios que apuestan en el sistema externo. Esto fue una simplificación para poder realizar pagos a sus usuarios sin tenerlos registrados. Cuando un usuario externo apuesta en el sistema de origen, el mismo devuelve el id de apuesta generado y el sistema externo registra este id junto con el id del usuario apostador para luego poder pagarle. Una vez que el sistema de origen paga al usuario G04, el sistema externo es notificado, con lo cual procede a pagar a cada usuario previamente registrado. La modalidad de consumo es análoga para ambos sentidos de comunicación.

El sistema externo es emulado mediante una aplicación *swing* sencilla, que permite apostar y cobrar las apuestas correspondientes.

## Apuestas móviles

Se implementó una aplicación *Android* (versión 2.1) la cual accede al sistema a través de *Web Services* REST. Presentando una interfaz simple que permite Apuestas a Eventos tipo “resultado exacto”.

## Distribución en “la nube” (*OpenShift*)

    Se realizó el deploy de la aplicación en Openshift. La misma implicó crear una aplicación con JBoss, JDK y MySQL instalados. Además se debió crear la base de datos en la nube y algún usuario para poder utilizarla.

## Acceso multimedia

Se consume la API de *Google* para la obtención de videos de los eventos más importantes en el sistema. El filtro se realiza por agregación de palabras clave, aquellas que contenga la Competición a la que está asociada el Evento y el nombre del último.

# ARQUITECTURA DEL SISTEMA

El estilo arquitectónico utilizado es en capas estricto, esto permitió mantener un enfoque relativamente sencillo a la hora de implementar la solución, además de presentar una estructura ordenada de la misma.

A continuación daremos una breve explicación de las vistas que explican la arquitectura. Puede consultarse el documento de arquitectura [9] para visualizar los diagramas correspondientes.

## Vista lógica

### Presentación

En cada uno de los módulos se tienen vistas (en formato XHTML) y *Managed Beans* de sesión que presentan propiedades que son asignadas a los campos de las vistas. Los *Managed Beans* también presentan cierta lógica de presentación, si bien la finalidad de los mismos es actuar más como la parte del modelo en el patrón MVC. Como consecuencia de utilizar *Java Server Faces* se tienen *servlets,* provistos de forma implícita, que facilitan el tratamiento de los *requests* y *responses* vinculados a la comunicación de datos. La navegación se hace, mayormente por medio de reglas declaradas en el archivo *faces-config.xml*, aunque también se tienen algunas navegaciones implícitas en ciertas acciones asociadas a botones y demás controles que respondan a ciertos eventos de interacción. Las validaciones se hacen principalmente del lado del servidor. Los *Managed Beans* pueden comunicarse entre sí por medio de *Managed Properties*, que es una forma de inyección de dependencias. La comunicación con la capa lógica se hace por medio de búsquedas (*lookups*) JNI [10] de los componentes correspondientes en el negocio.

### Negocio

Los componentes de negocio fueron separados en Interfaces y sus implementaciones. Actualmente, todas las interfaces son locales y se orientan hacia ciertas funcionalidades, de forma de que los *Managed Beans* en los que se inyecten tengan acceso sólo a determinadas funciones, de esta forma se separan las responsabilidades de los diversos componentes de negocio y se separa la especificación de la implementación. Cada clase de implementación se corresponde con un *Enterprise Java Bean* sin estado. La inyección de dependencias respecto a los componentes de la capa de persistencia se hace de la misma manera que en el caso anteriormente descripto. Se proveen además interfaces para comunicación con dispositivos móviles, las mismas son implementadas por *web services* de tipo REST (*Representational State Transfer*) [11], una forma de lograr que ni el cliente, ni el servidor necesiten mantener estado alguno para lograr establecer cierta comunicación.

### Servicios

Provee los servicios necesarios para realizar apuestas en el sistema desde un dispositivo *Android* o desde un sistema externo. La comunicación fue implementada mediante servicios REST (caso Android) y *Web Services* SOAP para la comunicación con sistema externos, en el primer caso se utilizó la librería *jax-rs*, mientras que para el segundo se recurrió a la librería *jax-ws*.

### Persistencia

Los componentes de esta capa están compuestos de interfaces locales y *EJB* sin estado, que implementan dichas interfaces. Para poder acceder a la base de datos de la aplicación, se utilizaron los mecanismos provistos por *JPA*. Para poder utilizarlo en un contexto de *Java EE* se debió inyectar un *Entity Manager* en cada uno de las clases de implementación. Al igual que en la capa lógica, se utilizaron interfaces que separan la especificación de la implementación*.* Como se comentó anteriormente, se utilizó el patrón de diseño DAO (*Data Access Object*), de esta forma se logró abstraer aún más la implementación concreta del acceso a datos del resto de la solución.

## Vista física

La separación en proyectos y las inyecciones de componentes mediante JNI permite que la arquitectura de la solución sea distribuible. Esto permite que el sistema escale si en un escenario eventual se multiplica el número de usuarios de la misma. Esta distribución de las capas y la comunicación entre ellas no afectan el hecho de que la aplicación pueda ser “deployada” en un medio de *Cloud Computing* como *OpenShift*.

# IMPLEMENTACIÓN

## Detalles de algunas funcionalidades

### Feeds y Multimedia

Se provee un método para cargar sitios de fuentes externas, para ello, en el módulo de administrador se incluyó una funcionalidad que, dado el nombre y la URL en la que se encuentran los *feeds* de determinada fuente, da de alta una nueva fuente de noticias. Luego, en la portada del módulo público se extraen los *feeds* y se los expone a los usuarios.

La tecnología utilizada para consumir los *feeds* es *Rome*.

### Seguridad

Se hizo hincapié en el control de acceso a recursos Web por parte de los Usuarios “logueados”. Para ello se pensó, en una primera instancia, en un enfoque *rol-based-access-control*, que es el proporcionado por *JBoss,* a través de la definición de un dominio de seguridad. Sin embargo, el *login* por formulario que propone rompe la estética de la Web, además de consumir mucho tiempo en pruebas y por lo tanto se decidió implementar el control de acceso a los recursos Web mediante un *servlet filter*.

### Google maps

La integarción con google maps se hizo a través de un componente provisto por *Prime Faces*. Con la ayuda de unas funciones simples de JavaScript, el componente pudo utilizarse para seleccionar ubicaciones en el mapa y para marcar los Eventos dados de alta.

### Integración con Pay Pal

Para el depósito de dinero de los usuarios en el sistema se utilizó la API propuesta por *Pay Pal*. El pago en el sentido *usuario-sistema*, se hace mediante una solicitud a *Pay Pal*, así se genera un ticket temporal, que el Usuario debe confirmar, con el cual se valida un transferencia de dinero entre la cuentas del usuario y la cuenta del sistema *SuperBet* (la banca). Para el pago en el sentido *sistema-usuario* se realiza una transferencia directa entre cuentas. El sistema *Pay Pal* permite hacer las pruebas en un área de pruebas: el *SandBox* del desarrollador.

## Productos y Herramientas

A continuación se presenta un cuadro con la evaluación de algunas de las herramientas más relevantes utilizadas durante el desarrollo de la solución.

## Problemas Encontrados

* **Implementación de *web services*:** surgieron conflictos con las *ip* dinámicas y los *wsld* generados.
* **Presentación:** debido a la poca experiencia del equipo de desarrollo con la librería de componentes utilizada (*primeFaces*) se experimentaron complicaciones para dar a la presentación ciertos comportamientos. Esto también se debió al hecho de que varios componentes de la librería no funcionaban de la forma que se esperaba.
* **Multimedia:** la comprensión de cómo hacer los filtros de búsqueda fue la principal dificultad en este aspecto. La abundancia de documentación hizo difícil hallar los puntos específicos que se necesitaban.
* **Persistencia:** *Hibernate* presentó ciertas dificultades de comprensión, tanto para construir el modelo de datos (a través de entidades), como para persistir aquellas entidades que se relacionaban con otras.

# EVALUACIÓN DE LA SOLUCIÓN

En líneas generales se cuenta con una solución de calidad, con una gran variedad de funcionalidades y buenos preceptos arquitectónicos. A continuación se presentan algunos puntos fuertes y limitaciones que la solución presenta a juicio de los desarrolladores.     
Algunas de las buenas propiedades más relevantes de la solución planteada son las siguientes:

* Escalabilidad: debido a la encapsulación de capas en proyectos independientes (conectadas mediante inyecciones JNI), lo cual permite que la solución sea escalable. Pueden distribuirse un gran número de instancias de la solución y de esa forma atender a una cantidad de usuarios eventualmente grande.
* Flexibilidad: pueden agregarse proyectos nuevos a la solución o bien extender las capas ya presentes y de esta forma extender las funcionalidades del sistema sin necesidad de hacer *refactoring* ni replantear la distribución de la arquitectura.
* Amigabilidad: la interfaz de usuario es fácil de utilizar y presenta un diseño agradable.

A continuación se exponen algunas de las limitaciones detectadas:

* Niveles de seguridad básicos: resulta necesario analizar mecanismos de seguridad más avanzados (encriptación de claves y del código de la aplicación, etc.)
* No se analizaron problemas de concurrencia, ni se ha indagado en cuestiones tales como picos de acceso a la solución.

Tabla 1. Evaluación de Productos y Herramientas

|  |  |  |  |
| --- | --- | --- | --- |
| **Producto** | **Puntos Fuertes** | **Puntos Débiles** | **Evaluación General** |
| PrimeFaces | - Gmaps - Gran variedad de componentes | - Facilidad de uso | - Algunos componentes no funcionan correctamente |
| Hibernate | - Permite abstraerse de la generación “a mano” del modelo de datos. | - Algunas consultas pueden tardar demasiado | - Persistir entidades relacionadas lleva a *workarounds* no deseables |
| Android | - Buena documentación en la web | - Testear la aplicación lleva mucho tiempo | - Testear la aplicación lleva mucho tiempo |
| API de Youtube | - Buena documentación | - API razonablemente bien diseñada | - Difícil entender el mecanismo de búsqueda de videos/imágenes |
| Rome | - Facilidad de uso | - No puede restringirse el número de *feeds* en la consulta | - Debido al poco ensayo con esta tecnología no se puede emitir un juicio |
| JSF | - Facilita el desarrollo de la interfaz de usuario | - Abundancia de librerías de componentes que permiten extender el comportamiento de la interfaz | - Difícil de configurar si no se tiene un servidor de aplicaciones que lo tenga previamente configurado |

# DESARROLLO DEL PROYECTO

A continuación se expone la planificación que se siguió a lo largo del proyecto, cabe destacar que hubo múltiples variaciones y restablecimientos del alcance, los mismos no serán expuestos. La planificación será expuesta enumerando los hitos que se plantearon.   
  
*Prototipo inicial, 6 Octubre:*

1. Gestión de Usuarios y sus Cuentas

* Registro de Usuarios.
* Edición de los Datos del Perfil (foto, datos personales, ubicación).
* Depósitos de dinero en su cuenta.

2. Gestión de Eventos Deportivos

* Dar de Alta un Evento Deportivo Simple

3. Gestión de Apuestas

* Permitir ingreso de apuestas simples para los Eventos que lo permitan.
* Pagar las apuestas ganadoras y cobrar Simples

*Funcionalidades básicas,* *20 Octubre*

1. Gestión de Usuarios y sus Cuentas

* Retiros de dinero en su cuenta.

2. Gestión de Eventos Deportivos

* Dar de Alta un Evento Combinado.
* Configurar las listas de eventos a los que se podrá apostar en forma combinada.
* Configurar los Calendarios de Eventos: el mismo incluye fecha, deporte, campeonato y país.
* Lugar o región, etc.

3. Gestión de Apuestas

* Permitir ingreso de apuestas combinadas para los grupos de eventos que se determinen.
* Pagar las apuestas ganadoras y cobrar Combinadas.

4. Información Adicional para los usuarios

* Generación de un área de novedades dónde se muestren los resultados de los eventos
* recientes a los que el usuario apostó.
* Generación de un área de información dónde se muestren novedades de sitios de noticias.
* relativas a los eventos sobre los que el usuario tiene apuestas vigentes.
* Historial de Apuestas Realizadas por el Usuario.

*Funcionalidades opcionales y mejoramiento de funcionalidades básicas, 1 al  31 Octubre*

1. Apuestas entre distintos Sistemas *Superbet*

2. Apuestas entre Usuarios.

3. Apuestas en Tiempo Real.

4. Integración con *Google Maps* para la visualización de Eventos por zonas geográficas.

5. Permitir a los Usuarios acceso desde dispositivos móviles a su cuenta y realizar Apuestas desde los mismos.

6. Brindarles a los Usuarios información multimedia sobre los eventos actuales.

*14 de Noviembre – Entrega Final.*  
  
Se planificaron entonces 3 iteraciones. En la primera iteración se registraron atrasos que fueron acarreados e incrementados en las siguientes iteraciones. Por esto se debió dejar de lado algunas funcionalidades opcionales tal como la apuesta de un Usuario contra otro Usuario y la implementación de interfaces de tipo GWT. El suma, el tiempo dedicado por semana fue superior al estimado inicialmente: 15 horas semanales por integrante.

# CONCLUSIONES Y TRABAJO A FUTURO

El desarrollo de la interfaz gráfica fue de los puntos más engorrosos debido, en parte, a la inexperiencia de los desarrolladores en este campo (cabe destacar que las nociones de diseño de interfaz con las que se cuentan no son óptimas) y por otro lado a que la biblioteca de componentes utilizada (*primeFaces*) posee muchos componentes que no funcionan como se espera en un principio, como consecuencia de esto se tuvo que incurrir en un extenso método de ensayo y error hasta dar con las soluciones adecuadas.   
Al principio de la implementación se incurrieron en dificultades para establecer un modelo de datos adecuado y en la persistencia de entidades, finalmente, luego de chocar con unas cuantas excepciones y errores varios, se entendió el mecanismo mediante el cual persiste *Hibernate* y se logró establecer los *workarounds* necesarios para acceder a los datos.     
  
Entre las posibilidades de mejora de la solución, se exponen los siguientes puntos:

* Estudios de concurrencia y mejoras en la seguridad:
* Registro de errores en tiempo de ejecución: para lo cual pueden utilizarse herramientas como *log4j*.
* Generación de reportes.
* Aumentar la cantidad y calidad del testing realizado y de esta forma detectar más posibilidades de mejora.

# REFERENCIAS

1. JEE – *Java Enterprise Edition* <http://www.oracle.com/technetwork/java/javaee/overview/index.html>
2. Laboratorio de Integración de Sistemas. Taller de Sistemas de Información 2 – Trabajo Obligatorio. Año 2011.
3. Hibernate – Jboss Community. <http://www.hibernate.org/>
4. PayPal - *developer site* <https://developer.paypal.com/>
5. Android - <http://www.android.com/>
6. Openshift - <https://openshift.redhat.com/app/>
7. API de datos de YouTube - <http://code.google.com/intl/es-ES/apis/youtube/getting_started.html>
8. Rome - <http://java.net/projects/rome/>
9. Documento de arquitectura – TSI 2 – Grupo 4
10. JNI – *Java Native Interface* <http://java.sun.com/docs/books/jni/>
11. REST – *Representational State Transfer Web Services* <http://www.oracle.com/technetwork/articles/javase/index-137171.html>